CertyIQ



CertyIQ

Premium exam material

Get certification quickly with the CertyIQ Premium exam material.
Everything you need to prepare, learn & pass your certification exam easily. Lifetime free updates
First attempt guaranteed success.

https://www.CertyIQ.com

About CertyIQ

We here at CertyIQ eventually got enough of the industry's greedy exam paid for. Our team of IT professionals comes with years of experience in the IT industry Prior to training CertIQ we worked in test areas where we observed the horrors of the paywall exam preparation system.

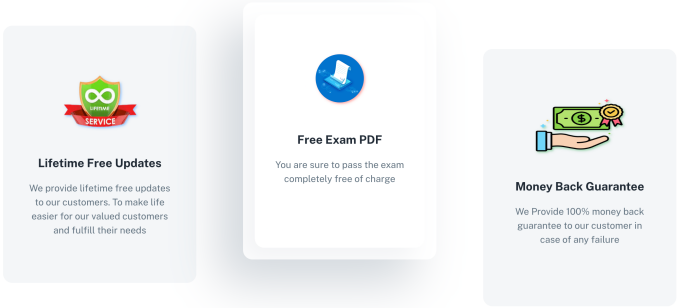
The misuse of the preparation system has left our team disillusioned. And for that reason, we decided it was time to make a difference. We had to make In this way, CertyIQ was created to provide quality materials without stealing from everyday people who are trying to make a living.

Doubt Support

We have developed a very scalable solution using which we are able to solve 400+ doubts every single day with an average rating of 4.8 out of 5.

https://www.certyiq.com

Mail us on - certyiqofficial@gmail.com



John

October 19, 2022

★★★★★

Thanks you so much for your help. I scored 972 in my exam today. More than 90% were from your PDFs!

Dana

September 04, 2022

★★★★★

Thanks a lot for this updated AZ-900 Q&A. I just passed my exam and got 974, I followed both of your Az-900 videos and the 6 PDF, the PDFs are very much valid, all answers are correct. Could you please create a similar video/PDF for DP900, your content/PDF's is really awesome. The team did a really good job. Thank You 😊.

Ahamed Shibly

2 months ago

★★★★★

Customer support is realy fast and helpful, I just finished my exam and this video along with the 6 PDF helped me pass! Definitely recommend getting the PDFs. Thank you!

October 22, 2022

★★★★★

Passed my exam today with 891 marks. Out of 52 questions, 51 were from certyiq PDFs including Contoso case study.
Thank You certyiq team!

Henry Rome

2 months ago

★★★★★

These questions are real and 100 % valid. Thank you so much for your efforts, also your 4 PDFs are awesome, I passed the DP900 exam on 1 Sept. With 968 marks. Thanks a lot, buddy!

Esmaria

2 months ago

★★★★★

Simple easy to understand explanations. To anyone out there wanting to write AZ900, I highly recommend 6 PDF's.Thank you so much, appreciate all your hard work in having such great content. Passed my exam Today - 3 September with 942 score.

Microsoft

(AZ-400)

Microsoft Azure DevOps Solutions

Total: **624 Questions**

Link: https://certyiq.com/papers/microsoft/az-400

Question: 1 CertyIQ

You are configuring project metrics for dashboards in Azure DevOps.
You need to configure a chart widget that measures the elapsed time to complete work items once they become active.
Which of the following is the widget you should use?

A. Cumulative Flow Diagram
B. Burnup
C. Cycle time
D. Burndown

Answer: C

Explanation:

The correct answer is Cycle Time. Cycle time specifically measures the elapsed time it takes for a work item to move from one stage to another, typically from "active" to "done." This aligns directly with the requirement of measuring the time to complete a work item after it becomes active. The widget visualizes this time, helping teams understand their efficiency in completing work. Cumulative Flow Diagrams (CFD) provide a broad overview of workflow, showing the number of work items in different states over time, but doesn't directly provide the elapsed time. Burnup charts track the total scope completed over time and the total scope required, but don't inherently indicate cycle time. Burndown charts track the remaining work to be done in a sprint or release, against the time remaining, also not related to the time it takes for a work item to progress from active to complete. Only Cycle Time provides the specific metric required for visualizing and understanding the speed at which work moves through the active stages of the workflow, providing insights on potential bottlenecks and areas for improvement.

Further resources:

Cycle Time, Lead Time, and Throughput: https://learn.microsoft.com/en-us/azure/devops/boards/reports/cycle-lead-time?view=azure-devops
Add and configure Cycle Time and Lead Time widgets: https://learn.microsoft.com/en-us/azure/devops/boards/dashboards/cycle-time-lead-time?view=azure-devops

Question: 2 CertyIQ

You need to consider the underlined segment to establish whether it is accurate.
The Burnup widget measures the elapsed time from creation of work items to their completion.
Select `No adjustment required` if the underlined segment is accurate. If the underlined segment is inaccurate, select the accurate option.

A. No adjustment required.
B. Lead time
C. Test results trend
D. Burndown

Answer: B

Explanation:

The underlined segment in the original statement is inaccurate. The Burnup widget does not measure the elapsed time from work item creation to completion. Instead, it visualizes the amount of work completed over

time against the total scope of work. It helps track progress towards a goal by showing the upward trend of completed work.

Lead time is the correct term to describe the elapsed time from when a work item is created to when it is completed. It measures the time it takes for a task to move through the entire workflow, from start to finish.

Burndown charts track the amount of work remaining over time, showing a downward trend as work is completed. Test results trend widgets display how testing results change over time, revealing trends and indicating the quality of the software being developed. They don't measure the elapsed time.

Therefore, the correct answer is **B. Lead time** because lead time is the metric used to measure the time between work item creation and completion.

Further reading:

Azure DevOps widgets: https://learn.microsoft.com/en-us/azure/devops/report/dashboards/widget-catalog?view=azure-devops
Understanding Lead Time and Cycle Time: https://www.atlassian.com/agile/metrics/cycle-time

Question: 3 CertyIQ

You are making use of Azure DevOps manage build pipelines, and also deploy pipelines.
The development team is quite large, and is regularly added to.
You have been informed that the management of users and licenses must be automated when it can be.
Which of the following is a task that can't be automated?

A. Group membership changes
B. License assignment
C. Assigning entitlements
D. License procurement

Answer: D

Explanation:

The correct answer is D. Here's why:

While Azure DevOps provides robust capabilities for automating user management, certain aspects remain outside the scope of automation, particularly the procurement of licenses.

Group Membership Changes (A): Azure DevOps supports integrating with Azure Active Directory (Azure AD) groups. Automating group membership changes in Azure AD directly translates to automated permission updates within Azure DevOps, streamlining user access management.

License Assignment (B): After licenses are available, Azure DevOps allows for automated assignment of licenses to users. This can be achieved through group rules in Azure Active Directory that automatically assign Azure DevOps licenses upon a user's addition to a specific group.

Assigning Entitlements (C): Entitlements, which grant specific access levels and permissions within Azure DevOps, can also be automated. Group rules in Azure AD can be configured to assign specific entitlements to users based on group membership.

License Procurement (D): The acquisition of Azure DevOps licenses typically involves a financial transaction with Microsoft. This process is not fully automated within Azure DevOps and typically requires manual intervention to select the required number of licenses and complete the purchase. You can set up billing and

link subscriptions, but the initial decision and action to procure licenses require human input. While tools like Azure Cost Management can help track license usage and forecast future needs, they do not automatically purchase additional licenses.

In summary, while Azure DevOps excels at automating user management tasks like group membership, license assignment, and entitlement assignment, the initial purchase of licenses (procurement) requires manual action and cannot be entirely automated.

Authoritative Links:

Manage users and access in Azure DevOps: https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/add-users-team-members?view=azure-devops
About Azure DevOps billing: https://learn.microsoft.com/en-us/azure/devops/billing/overview/what-is-billing?view=azure-devops
Automate Azure DevOps license management: https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/assign-access-levels-automatically?view=azure-devops

Question: 4 CertyIQ

You have been tasked with strengthening the security of your team's development process.
You need to suggest a security tool type for the Continuous Integration (CI) phase of the development process.
Which of the following is the option you would suggest?

A. Penetration testing
B. Static code analysis
C. Threat modeling
D. Dynamic code analysis

Answer: B

Explanation:

The correct answer is **B. Static code analysis**.

Static code analysis is the ideal security tool for the CI phase because it examines the source code before it's compiled and deployed. This proactive approach allows developers to identify potential vulnerabilities early in the development lifecycle, reducing the cost and effort associated with fixing them later. It doesn't require the code to be running, so it can be easily integrated into the CI pipeline. Static analysis tools can detect common security weaknesses like SQL injection, cross-site scripting (XSS), and buffer overflows based on predefined rules and patterns. Integrating this type of tool in the CI phase helps prevent insecure code from being merged into the main codebase, thereby minimizing the attack surface of the application. This approach embodies the shift-left security philosophy, promoting earlier detection and remediation of security flaws.

Penetration testing (A) and dynamic code analysis (D) are more suitable for later stages of the software development lifecycle, such as testing or production, as they require a running application to assess its security. Threat modeling (C) is a valuable practice, but it is more focused on identifying potential threats and attack vectors during the design phase, rather than directly analyzing code within the CI pipeline.

Further resources on static code analysis and its role in secure DevOps:

OWASP Static Analysis: https://owasp.org/www-project-static-analysis/
Microsoft's SDL Practices: https://www.microsoft.com/en-us/securityengineering/sdl/practices (This page highlights Static Analysis as a critical part of the Software Development Lifecycle)

Question: 5 CertyIQ

Your company is currently making use of Team Foundation Server 2013 (TFS 2013), but intend to migrate to Azure DevOps.
You have been tasked with supplying a migration approach that allows for the preservation of Team Foundation Version Control changesets dates, as well as the changes dates of work items revisions. The approach should also allow for the migration of all TFS artifacts, while keeping migration effort to a minimum.
You have suggested upgrading TFS to the most recent RTW release.
Which of the following should also be suggested?

A. Installing the TFS kava SDK
B. Using the TFS Database Import Service to perform the upgrade.
C. Upgrading PowerShell Core to the latest version.
D. Using the TFS Integration Platform to perform the upgrade.

Answer: B

Explanation:

The correct answer is **B. Using the TFS Database Import Service to perform the upgrade.**

Here's why:

The primary goal is to migrate from TFS 2013 to Azure DevOps while preserving historical data like changeset dates and work item revision dates with minimal effort. The TFS Database Import Service is specifically designed for this scenario. It's the recommended and officially supported method for migrating on-premises TFS databases directly to Azure DevOps Services. It handles the complex process of transferring data, including version control history, work items, builds, and tests.

Option A, Installing the TFS kava SDK, is incorrect. There's no such thing as the TFS kava SDK.

Option C, Upgrading PowerShell Core to the latest version, is partially relevant for certain scripting tasks around the migration, but it is not the core method for the database migration itself. While PowerShell might be used for pre-migration checks or post-migration validations, it doesn't directly handle the data transfer and historical data preservation.

Option D, Using the TFS Integration Platform to perform the upgrade, is also incorrect. The TFS Integration Platform is a deprecated tool and is not the recommended approach for migrating to Azure DevOps Services. It was primarily intended for synchronizing data between different TFS instances or other ALM tools, not for full-fidelity migrations with history preservation.

The Database Import Service performs a high-fidelity migration, ensuring that critical historical data is accurately transferred to Azure DevOps Services. Upgrading to the latest RTW release of TFS before using the Database Import Service is a best practice because it maximizes compatibility and minimizes potential issues during the migration process.

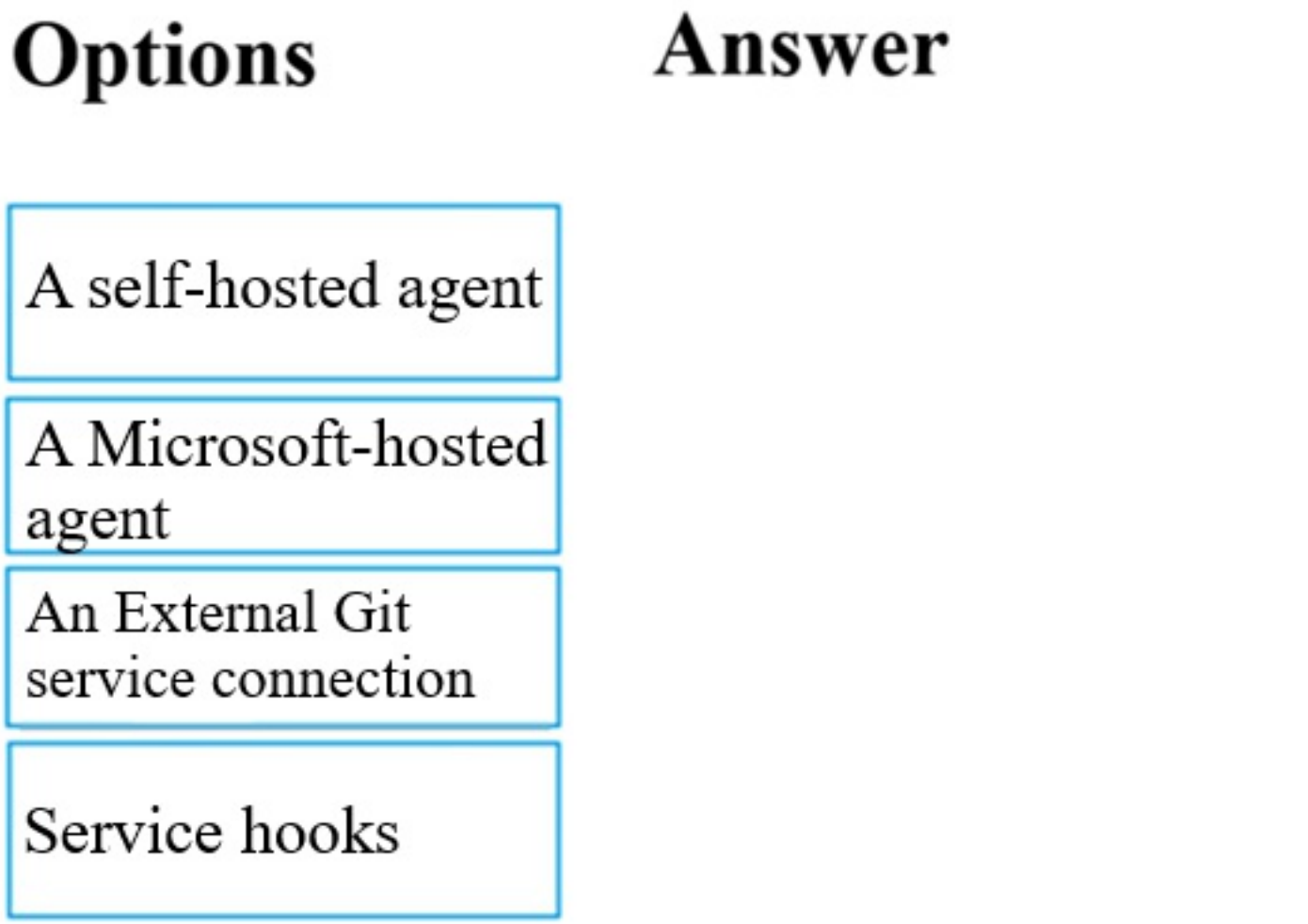
In summary, the TFS Database Import Service is the only option that directly addresses the requirements of preserving historical data and performing a complete migration from TFS 2013 to Azure DevOps Services with minimal effort. The key here is high-fidelity migration with history.

Authoritative Links:

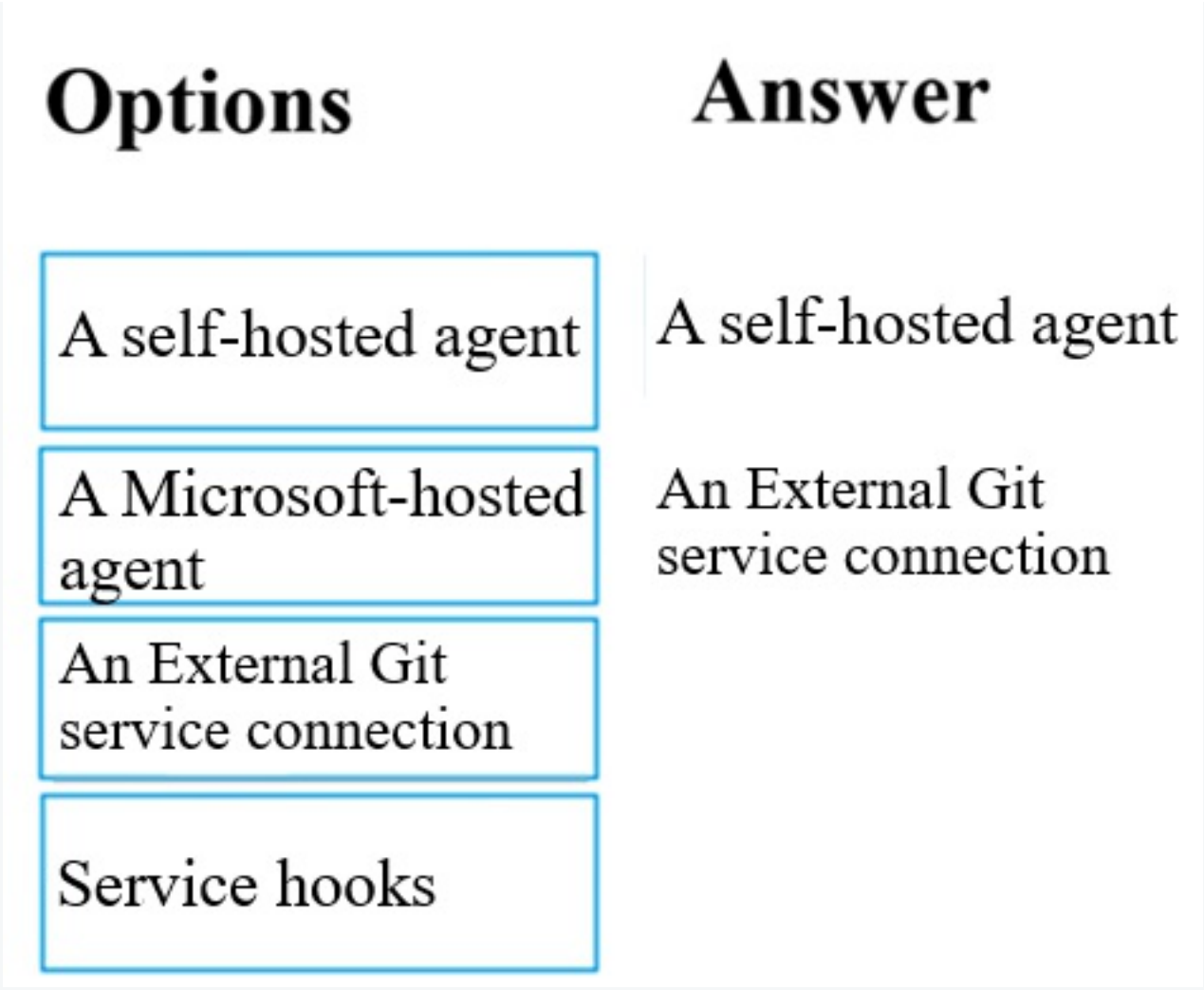
Migrate data from Azure DevOps Server to Azure DevOps Services: https://learn.microsoft.com/en-us/azure/devops/migrate/migration-overview?view=azure-devops
Overview of the data migration tool for Azure DevOps: https://learn.microsoft.com/en-us/azure/devops/migrate/data-migration-tool?view=azure-devops

Question: 6 CertyIQ

DRAG DROP -
You have an on-premises Bitbucket Server with a firewall configured to block inbound Internet traffic. The server is used for Git-based source control.
You intend to manage the build and release processes using Azure DevOps. This plan requires you to integrate Azure DevOps and Bitbucket.
Which of the following will allow for this integration? Answer by dragging the correct options from the list to the answer area.
Select and Place:



Answer:



Explanation:

Reference:
https://docs.microsoft.com/en-us/azure/devops/pipelines/repos/pipeline-options-for-git
" target="_blank">

Feature	Azure Pipelines	TFS 2017.2 and higher	TFS 2017 RTM	TFS 2015.4	TFS 2015 RTM
Branch	Yes	Yes	Yes	Yes	Yes
Clean	Yes	Yes	Yes	Yes	Yes
Tag or label sources	Project; Classic only	Team project	Team project	Team project	No
Report build status	Yes	Yes	Yes	No	No
Checkout submodules	Yes	Yes	Yes	Yes	Yes

Question: 7 CertyIQ

You are currently developing a project for a client that will be managing work items via Azure DevOps.
You want to make sure that the work item process you use for the client allows for requirements, change requests, risks, and reviews to be tracked.
Which of the following is the option you would choose?

A. Basic
B. Agile
C. Scrum
D. CMMI

Answer: D

Explanation:

The correct answer is D (CMMI) because it offers the most comprehensive and structured approach to work item tracking, explicitly supporting requirements, change requests, risks, and reviews – all vital for the client's needs. Agile and Scrum, while excellent for iterative development, lack the robust, formalized tracking features inherent in CMMI for these specific item types. Basic is too limited. CMMI, designed for process maturity and improvement, provides specific work item types and workflows tailored to managing these different aspects of a project. Requirements are managed and tracked as discrete work items. Change requests follow defined workflows for approval and implementation. Risks are identified, assessed, and mitigated through dedicated work items. Reviews are formally conducted and documented, ensuring quality control. Agile primarily focuses on user stories and sprint planning. Scrum revolves around product backlog items and sprint burndown charts. Basic offers minimal process customization. CMMI, on the other hand, aligns with a more structured and formal approach suitable for organizations needing rigorous tracking and documentation for audit or compliance purposes, which is implied by the client's requirement for tracking requirements, change requests, risks and reviews.

Further reading on Azure DevOps process templates can be found here:

Choose a process - Azure Boards | Microsoft Learn
About the CMMI process - Azure Boards | Microsoft Learn

Question: 8 CertyIQ

Note: The question is included in a number of questions that depicts the identical set-up. However, every question has a distinctive result. Establish if the solution satisfies the requirements.
You run the Register-AzureRmAutomationDscNode command in your company's environment.
You need to make sure that your company's test servers remain correctly configured, regardless of configuration drift.
Solution: You set the -ConfigurationMode parameter to ApplyOnly.
Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution does not meet the goal of ensuring test servers remain correctly configured against configuration drift. Setting -ConfigurationMode to ApplyOnly means the DSC configuration will be applied once,

but any subsequent drift from the desired state will not be automatically corrected. To ensure ongoing consistency and remediation of configuration drift, you need to use ApplyAndAutoCorrect.

ApplyOnly is suitable when you want to apply a configuration one time and don't want DSC to enforce compliance afterward. However, in scenarios like maintaining the desired state of test servers where configuration drift can occur, this mode is inadequate. To effectively manage configuration drift and ensure the test servers remain correctly configured, ApplyAndAutoCorrect should be used so that DSC will automatically correct any deviations from the defined configuration at regular intervals.

Therefore, the proposed solution of using ApplyOnly is not sufficient, and the answer is No.

Supporting Links:

Desired State Configuration Configuration Modes: https://learn.microsoft.com/en-us/powershell/dsc/concepts/configurations#configuration-modes
Azure Automation DSC Overview: https://learn.microsoft.com/en-us/azure/automation/automation-dsc-overview

Question: 9 CertyIQ

Note: The question is included in a number of questions that depicts the identical set-up. However, every question has a distinctive result. Establish if the solution satisfies the requirements.
You run the Register-AzureRmAutomationDscNode command in your company's environment.
You need to make sure that your company's test servers remain correctly configured, regardless of configuration drift.
Solution: You set the -ConfigurationMode parameter to ApplyAndMonitor.
Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution does not meet the goal. The objective is to ensure test servers remain correctly configured and handle configuration drift. The ApplyAndMonitor configuration mode will detect configuration drift, logging discrepancies. However, it will not automatically correct the drift. It only monitors the configuration and reports any deviations from the desired state.

To automatically correct configuration drift, the ApplyAndAutoCorrect mode should be used instead. This mode applies the configuration initially and then automatically corrects any drift that is detected in subsequent checks.

Therefore, the ApplyAndMonitor mode only provides visibility into configuration drift but lacks the capability to automatically remediate it, hence the solution fails to address the requirement of maintaining consistently configured test servers despite drift. Choosing ApplyAndAutoCorrect would provide the desired outcome.

References:

Azure Automation DSC Configuration Modes: https://learn.microsoft.com/en-us/azure/automation/desired-state-configuration/pull-server

Question: 10 CertyIQ

Note: The question is included in a number of questions that depicts the identical set-up. However, every question has a distinctive result. Establish if the solution satisfies the requirements.
You run the Register-AzureRmAutomationDscNode command in your company's environment.
You need to make sure that your company's test servers remain correctly configured, regardless of configuration drift.
Solution: You set the -ConfigurationMode parameter to ApplyAndAutocorrect.
Does the solution meet the goal?

A. Yes
B. No

Answer: A

Explanation:

The solution meets the goal. The -ConfigurationMode parameter in the Register-AzureRmAutomationDscNode command specifies how the Desired State Configuration (DSC) engine should handle configuration drift. When set to ApplyAndAutocorrect, the DSC engine not only applies the desired configuration but also actively monitors for any deviations from that configuration. If drift is detected, the engine automatically attempts to correct the configuration back to the desired state. This ensures that the test servers remain correctly configured even if changes are inadvertently made or configuration drifts occur. By using the ApplyAndAutocorrect mode, the system continuously enforces the desired configuration, mitigating the risk of misconfigured servers impacting testing activities. In contrast, other modes might only apply the configuration once or log drift without automatically remediating it.

Here's why the other options are not suitable:

Other Configuration Modes: Options like ApplyOnly apply the configuration once and do not correct drift. ApplyAndMonitor detect drift but do not automatically fix it.

Therefore, ApplyAndAutocorrect is specifically designed to maintain the desired configuration state, making it the most appropriate choice for preventing configuration drift on test servers and ensuring they remain correctly configured.

Supporting Documentation:

Register-AzureRmAutomationDscNode: This documentation explains the parameters of the Register-AzureRmAutomationDscNode cmdlet, including the ConfigurationMode parameter.
Desired State Configuration: Overview of PowerShell Desired State Configuration (DSC).

Question: 11 CertyIQ

You need to consider the underlined segment to establish whether it is accurate.
To compile an Internet Information Services (IIS) web application that runs docker, you should use a Default build agent pool.
Select `No adjustment required` if the underlined segment is accurate. If the underlined segment is inaccurate, select the accurate option.

A. No adjustment required.
B. Hosted Windows Container
C. Hosted
D. Hosted macOS

Answer: B

Explanation:

The underlined segment "Default build agent pool" is inaccurate for compiling an IIS web application that runs Docker containers in Azure DevOps. While the "Default" pool could work with sufficient configuration, it's not the best or most direct choice. The most appropriate and convenient option is the "Hosted Windows Container" agent pool.

Here's why:

1. **Docker requires a specific environment:** Docker relies on containerization, which inherently requires a supporting operating system with the necessary features for running containers. IIS web applications running in Docker containers thus need a Windows environment capable of running Windows containers.
2. **"Hosted Windows Container" is designed for this:** The "Hosted Windows Container" agent pool is specifically configured with a Windows Server environment that is pre-configured to run Windows containers. It has the Docker daemon installed and configured, along with other necessary dependencies.
3. **"Default" pool is generic:** The "Default" agent pool might contain agents running various operating systems. It does not guarantee a suitable environment for Docker-based builds, particularly Windows containers. Using the "Default" pool for this task would necessitate manually installing and configuring Docker, which increases complexity and introduces potential for errors.
4. **"Hosted macOS" is incorrect:** "Hosted macOS" is for building applications targeting the macOS operating system and is irrelevant for building Windows-based applications, Docker or otherwise.
5. **"Hosted" pool might work, but less efficient:** The "Hosted" pool typically uses a general-purpose Windows Server image but might not include the container features enabled. While the necessary features could potentially be installed manually, it's less efficient and error-prone compared to using an agent pool specifically tailored for Windows container builds. It is more appropriate to use the "Hosted Windows Container" pool which is preconfigured for Docker.

Therefore, the best and most accurate answer is "Hosted Windows Container" because it provides the ready-made environment needed to build and test IIS web applications running within Windows containers.

Supporting Links:

Microsoft Azure DevOps Agent Pools: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/pools-queues?view=azure-devops&tabs=yaml
Use a Windows container to build your code: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/docker?view=azure-devops

Question: 12 CertyIQ

Your company has an Azure DevOps environment that can only be accessed by Azure Active Directory users.
You are instructed to make sure that the Azure DevOps environment can only be accessed from devices connected to the company's on-premises network.
Which of the following actions should you take?

A. Assign the devices to a security group.
B. Create a GPO.
C. Configure Security in Project Settings from Azure DevOps.
D. Configure conditional access in Azure Active Directory.

Answer: D

Explanation:

The correct answer is D: Configure conditional access in Azure Active Directory (Azure AD).

Conditional Access in Azure AD is the right tool for restricting access to Azure DevOps based on network location. It allows you to create policies that specify conditions under which users can access resources. In this scenario, the condition is the network location.

Here's why the other options are incorrect:

A. Assign the devices to a security group: While security groups are useful for managing permissions, they cannot be directly used to restrict access based on network location.
B. Create a GPO (Group Policy Object): GPOs are primarily used for managing settings within an on-premises Active Directory environment and do not directly control access to cloud services like Azure DevOps.
C. Configure Security in Project Settings from Azure DevOps: Azure DevOps's project settings control permissions within the Azure DevOps environment itself. They don't provide the capability to restrict access based on the originating network.

To configure Conditional Access, you can define a named network location in Azure AD based on your company's on-premises network IP address range. Then, you create a Conditional Access policy that requires users to be connected to this named network location to access Azure DevOps. If a user attempts to access Azure DevOps from a different network, the policy will block their access. This ensures that only devices connected to the on-premises network can access the Azure DevOps environment, meeting the specified requirement.Conditional access leverages the identity-driven security model, providing granular control over resource access by evaluating various signals, including location, device state, and user risk. Conditional Access also works in conjunction with other security features, such as multi-factor authentication, enhancing overall security posture.

Here are some authoritative links for further research:

What is Conditional Access in Azure Active Directory?
Common Conditional Access policies
Location condition in Azure Active Directory Conditional Access

Question: 13 CertyIQ

You are making use of Azure DevOps to configure Azure Pipelines for project, named PROJ-01.
You are preparing to use a version control system that allows for source code to be stored on a managed Windows server located on the company network.
Which of the following is the version control system you should use?

A. Github Enterprise
B. Bitbucket cloud
C. Github Professional
D. Git in Azure Repos

Answer: A

Explanation:

The correct version control system is GitHub Enterprise. Here's why:

GitHub Enterprise is designed specifically for organizations requiring on-premises hosting of their Git repositories. The scenario describes a requirement to store source code on a managed Windows server within the company network. This explicitly aligns with GitHub Enterprise's purpose.

While Git is the underlying technology for all the listed options, GitHub Enterprise provides a managed platform that can be installed on-premises, offering features like access control, collaboration tools, and integrations with existing enterprise systems. This allows for the centralized management and security of source code behind the corporate firewall.

Git in Azure Repos, Bitbucket Cloud, and GitHub Professional are all cloud-hosted services. They do not provide the option to host the repository on a company-managed Windows server. Azure Repos, while integrated with Azure DevOps, is a cloud-based service, and Bitbucket Cloud is hosted by Atlassian. GitHub Professional is a licensing tier of GitHub's cloud offering, not an on-premises solution.

Therefore, the constraint of storing source code on a company-managed Windows server definitively points to GitHub Enterprise as the optimal choice. It offers the robust version control capabilities of Git while fulfilling the requirement of on-premises hosting for security and compliance reasons.

Further reading:

GitHub Enterprise: https://docs.github.com/en/enterprise-server@latest

Question: 14 CertyIQ

You need to consider the underlined segment to establish whether it is accurate.
When moving to Azure DevOps, JIRA must be replaced with the build pipelines Azure DevOps service.
Select `No adjustment required` if the underlined segment is accurate. If the underlined segment is inaccurate, select the accurate option.

A. No adjustment required.
B. repos
C. release pipelines
D. boards

Answer: D

Explanation:

The underlined segment, "JIRA must be replaced with the build pipelines Azure DevOps service," is inaccurate. JIRA is a project management and issue tracking tool, primarily related to planning and tracking work. While Azure DevOps offers its own work item tracking (Boards) service, migrating to Azure DevOps doesn't mandate replacing JIRA. Organizations can choose to continue using JIRA and integrate it with Azure DevOps using extensions or connectors.

Azure DevOps's build pipelines service primarily focuses on Continuous Integration (CI) and Continuous Delivery (CD), which involve automating the build, test, and deployment processes of software. These pipelines are not a direct replacement for JIRA's functionalities.

The closest analog to JIRA within Azure DevOps would be Azure Boards. Boards handles agile planning, work item tracking, bug tracking, and sprint management. Therefore, the correct adjustment is to replace "build pipelines Azure DevOps service" with "boards."

Options B ("repos") and C ("release pipelines") are also incorrect. "Repos" refer to Azure Repos, the source code repository, and "release pipelines" are a specific part of the broader Azure Pipelines service responsible for deployment automation, neither of which directly replaces JIRA.

Therefore, the most accurate adjustment is to state that JIRA functionalities are best replaced with Azure Boards, not build pipelines. Organizations can choose to use both JIRA and Azure DevOps in conjunction using integration capabilities, rather than completely replacing one with the other.

Reference links:

Azure Boards
Azure Pipelines
JIRA Integration with Azure DevOps (Example of an integration)

Question: 15 CertyIQ

You scan a Node.js application using WhiteSource Bolt.
The scan finds numerous libraries with invalid licenses, but are only used during development.
You have to make sure that only production dependencies are scanned by WhiteSource Bolt.
Which of the following is a command you should run?

A. npm edit
B. npm publish
C. npm install
D. npm update

Answer: C

Explanation:

The correct answer is C, npm install.

Here's a detailed justification:

WhiteSource Bolt, now Mend Bolt, is a dependency management tool that identifies security vulnerabilities and license compliance issues in open-source components used in a software project. The goal is to ensure only production-ready and legally compliant dependencies are deployed.

The core idea is to instruct npm (Node Package Manager) to install only production dependencies. By default, when you run npm install, it installs all dependencies listed in package.json, including both production and development dependencies (specified under devDependencies). However, you can instruct npm to only install the production dependencies.

The command npm install --production (or its shorthand, npm i --production) tells npm to skip installing the devDependencies listed in the package.json file. This is crucial in this scenario because the question specifies that the problematic libraries with invalid licenses are only used during development.

By using npm install --production before running WhiteSource Bolt, you ensure that only the dependencies intended for deployment are present during the scan. Consequently, WhiteSource Bolt will only analyze the production dependencies, excluding the problematic development dependencies with license issues. This focuses the scan on the essential components needed for the application's runtime and reduces false positives. Other options like npm edit, npm publish, or npm update don't directly control which dependencies are installed during the install process and thus, are not correct in addressing the given problem. npm update updates packages to their latest versions but still installs both dev and prod dependencies unless used with --production.

In summary, by strategically utilizing the --production flag with npm install, you can effectively filter out unnecessary dependencies that are solely used for development, ensuring WhiteSource Bolt focuses solely

on production components and their compliance status.

Relevant links:

npm install documentation: https://docs.npmjs.com/cli/v9/commands/npm-install
Mend Bolt documentation: https://www.mend.io/resources/resource-library/whitesource-bolt-for-azure-devops/ (Now Mend Bolt)

Question: 16 CertyIQ

You are currently defining a release strategy for an app, named APP-01.
The strategy should allow you to keep the time it takes to deploy new releases of the app to a minimum. The strategy should also allow you to roll back in the shortest time required.
Which of the following is the release strategy you should use?

A. Red/Black deployment
B. Rolling deployment
C. Big Bang deployment
D. Canary deployment

Answer: A

Explanation:

The correct answer is A (Red/Black deployment, also known as Blue/Green deployment) because it minimizes deployment time and rollback time, aligning with the stated requirements.

Here's a detailed justification:

Red/Black deployment (Blue/Green) involves maintaining two identical environments: a "red" (or "blue") environment which is live and serving traffic, and a "black" (or "green") environment which is idle. A new version of the application is deployed to the idle environment. Extensive testing is performed in the idle environment. Once the new version is verified, traffic is switched from the live environment to the updated idle environment.

This approach minimizes deployment time because the new version is already deployed and tested in a separate environment. Switching traffic typically involves a DNS change or load balancer configuration update, a fast operation.

Rollback is also very fast. If issues arise after switching traffic, the traffic can be switched back to the previous environment immediately. The old environment remains intact and ready to serve traffic, ensuring minimal downtime. This rapid rollback capability directly addresses the requirement to roll back in the shortest time.

Rolling deployments involve deploying the new version of the application to a subset of servers and then gradually increasing the number of servers running the new version. While less risky than Big Bang, this strategy usually takes longer than Red/Black as it requires the deployment of the application onto multiple servers over time.

Big Bang deployment is a simple approach that involves deploying the new version of the application to all servers at once. While deployment is fast, rollback is complicated and risky. This is not a recommended approach in a production environment, especially when minimum rollback time is a key requirement.

Canary deployment involves deploying the new version of the application to a small subset of users or servers and then gradually increasing the number of users or servers that are exposed to the new version. While it

allows for gradual rollout and issue detection, it's slower than Red/Black and may not provide the fastest possible rollback.

In summary, Red/Black deployment best aligns with the key requirements of minimizing deployment time and ensuring the fastest rollback, making it the most suitable release strategy.

Further reading:

Martin Fowler on BlueGreenDeployment: https://martinfowler.com/bliki/BlueGreenDeployment.html
Microsoft Azure documentation on deployment strategies: (Search Azure documentation for "deployment strategies" or "blue green deployment") - While a direct link may change, Microsoft's Azure documentation always contains relevant information.

Question: 17 CertyIQ

Your company hosts a web application in Azure, and makes use of Azure Pipelines for managing the build and release of the application.
When stakeholders report that system performance has been adversely affected by the most recent releases, you configure alerts in Azure Monitor.
You are informed that new releases must satisfy specified performance baseline conditions in the staging environment before they can be deployed to production.
You need to make sure that releases not satisfying the performance baseline are prevented from being deployed.
Which of the following actions should you take?

A. You should make use of a branch control check.
B. You should make use of an alert trigger.
C. You should make use of a gate.
D. You should make use of an approval check.

Answer: C

Explanation:

The correct answer is C: You should make use of a gate.

Gates in Azure Pipelines are crucial for controlling the deployment process based on predefined conditions. In this scenario, the requirement is to prevent releases that don't meet the performance baseline in the staging environment from being deployed to production. Gates achieve this by allowing you to evaluate specific criteria before a release proceeds to the next stage. These criteria can include the output from Azure Monitor alerts or other sources providing performance data.

Here's why the other options are not suitable:

A. Branch control check: Branch control checks manage code integration, focusing on the source code repository, not runtime performance. This is related to code quality checks before merging to the main branch.
B. Alert trigger: Alert triggers in Azure Monitor are used for notification purposes, but do not natively block pipeline execution. They can inform someone about a problem, but a gate prevents deployment.
D. Approval check: Approval checks require manual intervention. Although approvals are useful in many cases, they don't automatically evaluate performance metrics. They would not serve as an automated performance baseline validator.

By implementing a gate that checks the Azure Monitor alerts related to performance metrics in the staging environment, you can ensure that only releases that meet the predefined performance baseline proceed to production. This mechanism helps maintain system performance and prevent problematic deployments. The gate can query the status of the Azure Monitor alerts and halt the pipeline if any alerts indicate a performance

degradation. This aligns with the requirement to prevent releases that adversely affect system performance from reaching production.

Therefore, gates offer the precise control and automation required to satisfy the performance baseline conditions for Azure Pipelines deployments.

Supporting Documentation:

Azure Pipelines Gates and Approvals: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops&tabs=check-pass
Azure Monitor Alerts: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-overview

Question: 18 CertyIQ

You need to consider the underlined segment to establish whether it is accurate.
To deploy an application to a number of Azure virtual machines, you should create a universal group.
Select `No adjustment required` if the underlined segment is accurate. If the underlined segment is inaccurate, select the accurate option.

A. No adjustment required.
B. security
C. deployment
D. resource

Answer: C

Explanation:

The underlined segment "universal group" is inaccurate in the context of deploying an application to multiple Azure virtual machines. Universal groups are primarily used for managing user access and permissions within a Windows Server Active Directory environment. They are not directly involved in application deployment procedures.

Instead, a more relevant concept for deploying applications to a set of virtual machines in Azure would be related to deployment groups. A deployment group in Azure DevOps is a logical set of target machines (VMs) where you want to deploy an application. You define the deployment group, register the VMs to it, and then target your deployment pipelines towards that deployment group. This allows for coordinated and managed deployment across multiple machines.

Security groups, while related to access control, aren't the primary mechanism for deploying applications. Resource groups are containers for Azure resources but are also not used for this purpose. Therefore, the word "deployment" is the most accurate replacement, as it suggests a mechanism specifically designed for deploying applications across a collection of VMs.

Justification Summary:

Universal groups are for access management, not application deployment.
Deployment groups are specifically designed for deploying applications to multiple machines.
Security groups are for managing access control.
Resource groups are containers for Azure resources.

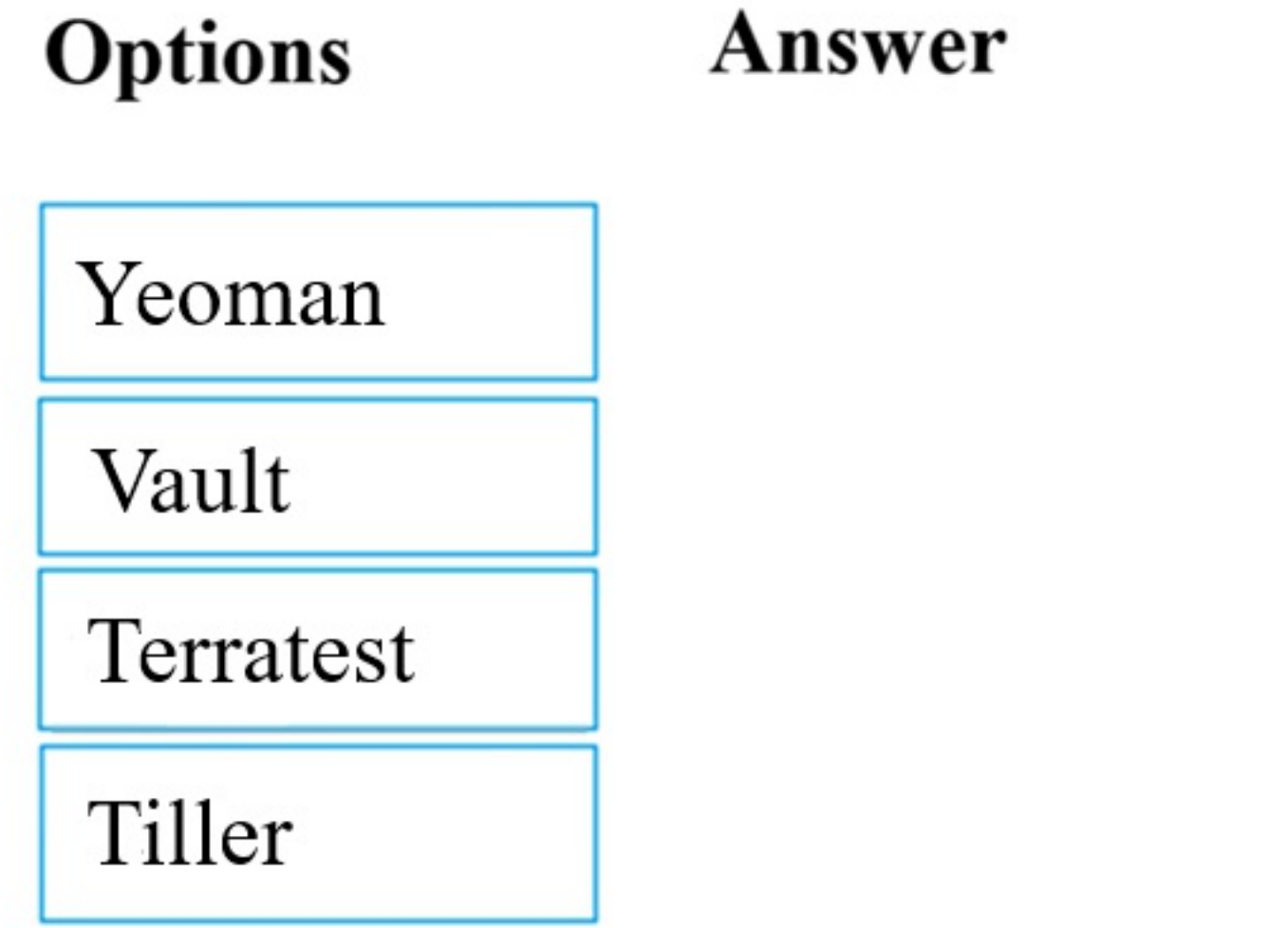
Supporting Links:

Azure DevOps Deployment Groups
Azure Resource Groups

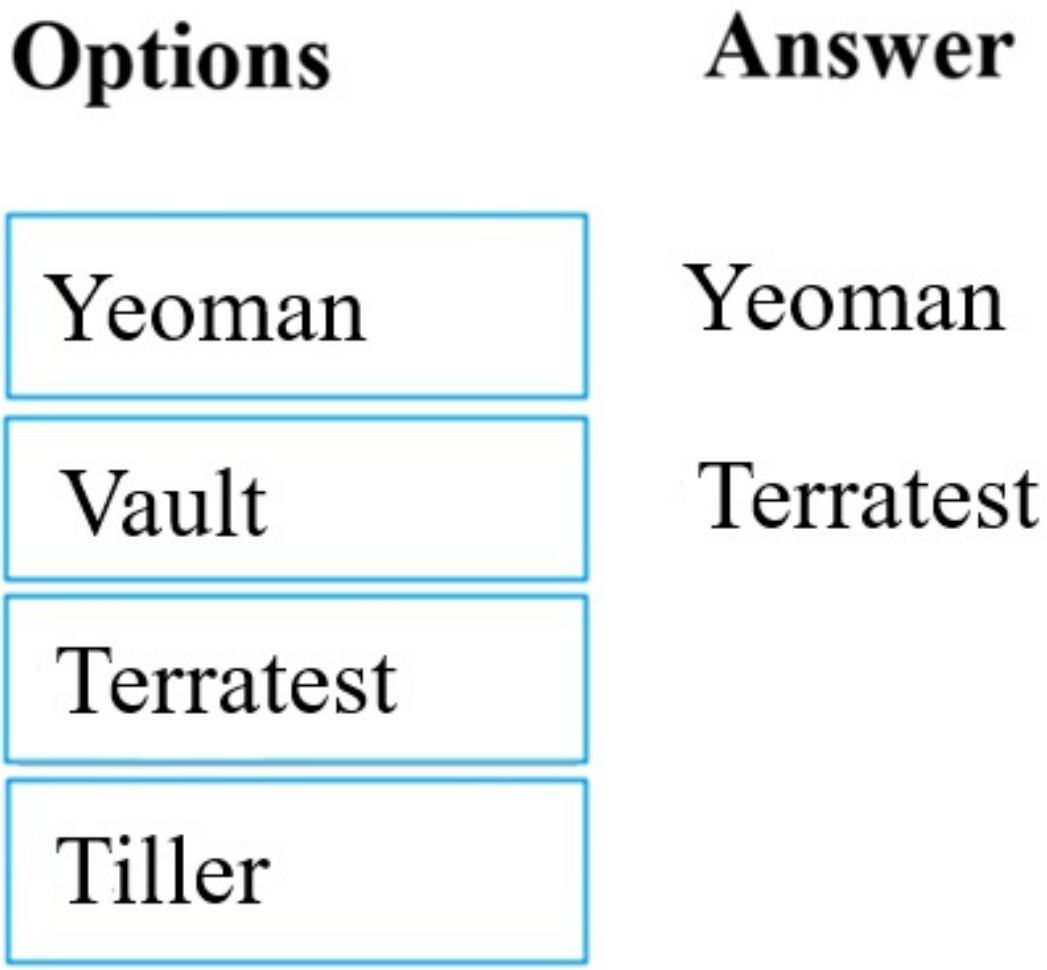
Azure Security Groups

Question: 19 CertyIQ

DRAG DROP -
You are preparing to deploy an Azure resource group via Terraform.
To achieve your goal, you have to install the necessary frameworks.
Which of the following are the frameworks you should use? Answer by dragging the correct options from the list to the answer area.
Select and Place:



Answer:



Explanation:
You can use the combination of Terraform and Yeoman. Terraform is a tool for creating infrastructure on Azure. Yeoman makes it easy to create Terraform modules.
Terratest provides a collection of helper functions and patterns for common infrastructure testing tasks, like making HTTP requests and using SSH to access a specific virtual machine. The following list describes some of the major advantages of using Terratest:
✑ Convenient helpers to check infrastructure - This feature is useful when you want to verify your real infrastructure in the real environment.
✑ Organized folder structure - Your test cases are organized clearly and follow the standard Terraform module folder structure.
Test cases are written in Go - Many developers who use Terraform are Go developers. If you're a Go developer, you don't have to learn another programming
▪ language to use Terratest.
✑ Extensible infrastructure - You can extend additional functions on top of Terratest, including Azure-specific features.

Reference:
https://docs.microsoft.com/en-us/azure/developer/terraform/create-base-template-using-yeoman
https://docs.microsoft.com/en-us/azure/developer/terraform/test-modules-using-terratest

Question: 20 CertyIQ

You intend to make use of Azure Artifacts to share packages that you wrote, tested, validated, and deployed. You want to use a solitary feed to release several builds of each package. You have to make sure that the release of packages that are in development is restricted.

Which of the following actions should you take?

A. You should make use of static code analysis.
B. You should make use of views.
C. You should make use of dynamic code analysis.
D. You should make use of upstream sources.

Answer: B

Explanation:

The correct answer is B: You should make use of views.

Azure Artifacts Views provide a way to share specific versions of packages with different audiences. This is perfectly suited for managing the release lifecycle of packages, especially when dealing with development packages. By using a view, such as @prerelease or @dev, you can isolate packages still under development, ensuring that only authorized users or processes have access to them. This prevents accidental consumption of unstable packages in production or by wider audiences.

Once a package has been validated and is ready for wider consumption, it can be promoted to another view, such as @release or @public. This promotion doesn't involve re-uploading the package; it simply changes its visibility based on the selected view. This ensures a clear separation between development and production packages and provides a controlled release process.

Static code analysis (A) and dynamic code analysis (C) are related to code quality and security, not to feed management or package visibility. Upstream sources (D) allow you to access packages from other feeds, which is useful but not directly relevant to restricting access to development packages within your own feed. Therefore, views are the most effective and appropriate mechanism for controlling the visibility and access to packages in different stages of development within Azure Artifacts.

For more information, refer to the official Microsoft documentation on Azure Artifacts views:
https://learn.microsoft.com/en-us/azure/devops/artifacts/concepts/views?view=azure-devops

Question: 21 CertyIQ

You need to consider the underlined segment to establish whether it is accurate.
To find when common open source libraries are added to the code base, you should add Jenkins to the build pipeline.
Select `No adjustment required` if the underlined segment is accurate. If the underlined segment is inaccurate, select the accurate option.

A. No adjustment required.
B. SourceGear Vault
C. WhiteSource
D. OWASP ZAP

Answer: C

Explanation:

The underlined segment suggesting Jenkins for identifying open-source libraries in a codebase is inaccurate. While Jenkins is a powerful automation server for CI/CD pipelines, it doesn't inherently possess the capability to specifically identify and track the inclusion of open-source libraries and their versions.

WhiteSource (now Mend) and similar Software Composition Analysis (SCA) tools are designed for this very purpose. SCA tools automatically scan codebases and builds to identify open-source components, track their licenses, detect vulnerabilities, and ensure compliance. They provide insights into the provenance and security risks associated with the open-source libraries being used.

OWASP ZAP is primarily a web application security scanner used to find vulnerabilities. SourceGear Vault is a source control system. Therefore, neither is suitable for identifying open-source libraries.

Adding WhiteSource (or a comparable SCA tool) to the build pipeline enables continuous monitoring of open-source dependencies, early detection of security vulnerabilities, and improved license management. This aligns with best practices for secure software development and supply chain security.

Therefore, "WhiteSource" is the accurate replacement for the underlined segment.

Software Composition Analysis (SCA): https://owasp.org/www-community/Component_Analysis
Mend (formerly WhiteSource): https://www.mend.io/

Question: 22 CertyIQ

Your company has an Azure DevOps project, which includes a build pipeline that makes use of roughly fifty open source libraries.
You have been tasked with making sure that you are able to scan project for common security weaknesses in the open source libraries.
Which of the following actions should you take?

A. You should create a build task and use the WhiteSource Bolt service.
B. You should create a deployment task and use the WhiteSource Bolt service.
C. You should create a build task and use the Chef service.
D. You should create a deployment task and use the Chef service.

Answer: A

Explanation:

The correct answer is **A. You should create a build task and use the WhiteSource Bolt service.**

Here's why:

Security Scanning and Build Pipeline Integration: The requirement is to scan open-source libraries for security vulnerabilities during the build process. This is crucial for identifying and addressing issues early in the development lifecycle, before deployment. Integrating a security scan directly into the build pipeline ensures that every build is checked for known vulnerabilities.

WhiteSource Bolt: WhiteSource Bolt (now Mend Bolt) is specifically designed for this purpose. It's a free Azure DevOps extension that integrates directly into build pipelines and analyzes open-source components for security vulnerabilities, license compliance, and other risks. It provides a report of identified vulnerabilities, including severity level and suggested remediation steps.

Build Task vs. Deployment Task: Security scanning should happen during the build process, not during deployment. Build tasks are executed during the build pipeline, while deployment tasks are executed during the release pipeline (which follows the build). Scanning during deployment might catch vulnerabilities late, making remediation more complex and risky. Scanning during the build process ensures faster and more cost-effective detection and mitigation.

Chef: Chef is an automation platform for infrastructure as code. While it plays a role in configuration

management and automation, it's not directly involved in scanning open-source libraries for security vulnerabilities. Therefore, Chef is not the correct choice for the given scenario.

Why Mend Bolt (Formerly WhiteSource Bolt) is Appropriate: Mend Bolt is tailored for scanning open-source components used in a project. It identifies vulnerabilities from public databases such as the National Vulnerability Database (NVD) and provides insights into licenses and potential license conflicts, which can be crucial for legal compliance. By incorporating it as a build task, developers can proactively address potential security issues and license violations before releasing the software.

In summary, using WhiteSource Bolt (Mend Bolt) as a build task allows the company to automatically scan open-source libraries for security vulnerabilities as part of the build process, ensuring that only secure and compliant code is deployed.

Supporting Links:

Mend Bolt: https://www.mend.io/free-developer-tools/mend-bolt/
Azure DevOps Build Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops
National Vulnerability Database (NVD): https://nvd.nist.gov/

Question: 23 CertyIQ

You need to consider the underlined segment to establish whether it is accurate.
Black Duck can be used to make sure that all the open source libraries conform to your company's licensing criteria.
Select `No adjustment required` if the underlined segment is accurate. If the underlined segment is inaccurate, select the accurate option.

A. No adjustment required.
B. Maven
C. Bamboo
D. CMAKE

Answer: A

Explanation:

The underlined segment, stating that Black Duck can ensure open-source libraries conform to a company's licensing criteria, is accurate. Black Duck by Synopsys is a software composition analysis (SCA) tool specifically designed to manage open-source risks, including license compliance. It scans software to identify open-source components and their associated licenses. Black Duck helps organizations define and enforce policies around acceptable licenses and automatically alerts users to violations. It can detect instances where open-source components are used with incompatible licenses, preventing potential legal issues. Therefore, the underlined segment is accurate, and no adjustment is required. Maven, Bamboo, and CMAKE are build automation tools and do not perform the specific task of open-source license compliance management.

Further Reading:

Synopsys Black Duck: https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html
Software Composition Analysis (SCA): https://owasp.org/www-community/Component_Analysis

Question: 24 CertyIQ

You have created an Azure DevOps project for a new application that will be deployed to a number of Windows Server 2016 Azure virtual machines.
You are preparing a deployment solution that allows for the virtual machines to maintain a uniform configuration, and also keep administrative effort with regards to configuring the virtual machines to a minimum.
Which of the following should be part of your solution? (Choose two.)

A. Azure Resource Manager templates
B. The PowerShell Desired State Configuration (DSC) extension for Windows
C. Azure pipeline deployment groups
D. The Custom Script Extension for Windows
E. Azure pipeline stage templates

Answer: AB

Explanation:

The optimal solution for maintaining a uniform configuration across Windows Server 2016 Azure VMs with minimal administrative overhead involves leveraging Azure Resource Manager (ARM) templates and the PowerShell Desired State Configuration (DSC) extension.

ARM templates allow you to define the infrastructure as code, ensuring consistent deployment and configuration of the VMs. You can specify desired VM settings, network configurations, and other resources within the template, guaranteeing uniformity across all instances. This "infrastructure as code" approach is a fundamental principle of modern cloud deployments, facilitating repeatability and minimizing configuration drift. https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/overview

The PowerShell DSC extension for Windows complements ARM templates by enforcing desired states within the VMs themselves. DSC allows you to define the desired configuration for software, services, and settings. The extension automatically applies and maintains this configuration, ensuring that the VMs remain in the desired state over time, even if manual changes are made. It ensures that your VMs adhere to the required configuration standard, mitigating the risk of inconsistencies. https://docs.microsoft.com/en-us/azure/virtual-machines/extensions/dscwindows

While Custom Script Extension can execute arbitrary scripts, it's less suited for ongoing configuration management and drift prevention than DSC. Deployment groups are useful for deploying applications, but do not directly address the configuration consistency requirements. Stage templates enable reusability within pipelines, but don't contribute to maintaining VM configuration. Therefore, the combination of ARM templates and DSC provides the most robust and scalable solution for the scenario described.

Question: 25 CertyIQ

Your company has an application that contains a number of Azure App Service web apps and Azure functions.
You would like to view recommendations with regards to the security of the web apps and functions. You plan to navigate to Compute and Apps to achieve your goal.
Which of the following should you access to make use of Compute and Apps?

A. Azure Log Analytics
B. Azure Event Hubs
C. Azure Advisor
D. Azure Security Center

Answer: D

Explanation:

The correct answer is D, Azure Security Center. Azure Security Center (now Microsoft Defender for Cloud) is a unified security management system that strengthens the security posture of your cloud resources, including Azure App Service web apps and Azure Functions. It provides security assessments and recommendations to help you identify and remediate vulnerabilities and improve your overall security hygiene.

Compute and Apps within Defender for Cloud is specifically designed to provide security insights and recommendations for compute resources like web apps and functions. It identifies potential threats, misconfigurations, and missing security controls, offering actionable guidance to mitigate risks.

Azure Log Analytics (A) is a service for collecting and analyzing log data, but it primarily focuses on operational insights and troubleshooting rather than providing direct security recommendations. Azure Event Hubs (B) is a real-time data ingestion service used for streaming data; it is not relevant for security recommendations in the context of compute resources. Azure Advisor (C) provides recommendations on cost optimization, performance, reliability, and operational excellence. While it can offer some security-related advice, its primary focus isn't security posture management specifically for compute resources.

Therefore, navigating to Compute and Apps to view security recommendations aligns perfectly with the functionality of Azure Security Center (Microsoft Defender for Cloud).

Supporting links:

Microsoft Defender for Cloud
Secure your Azure App Service environment in Security Center

Question: 26 CertyIQ

You need to consider the underlined segment to establish whether it is accurate.
Your company has a multi-tier application that has its front end hosted in Azure App Service.
To pinpoint the average load times of the application pages, you should make use of Azure Event Hubs.
Select `No adjustment required` if the underlined segment is accurate. If the underlined segment is inaccurate, select the accurate option.

A. No adjustment required.
B. Azure Application Insights
C. Azure Log Analytics
D. Azure Advisor

Answer: B

Explanation:

The underlined segment suggesting the use of Azure Event Hubs to pinpoint average load times of application pages is inaccurate. Azure Event Hubs is primarily a big data streaming platform and event ingestion service. It's excellent for handling high volumes of events for real-time analytics, IoT telemetry, and clickstream data. However, it is not designed for directly measuring application page load times.

Azure Application Insights is the correct tool for this scenario. Application Insights, a feature of Azure Monitor, provides deep application performance monitoring (APM) capabilities. It automatically detects performance anomalies, diagnoses issues, and helps understand user behavior. Specifically, it tracks page load times, response times, request rates, and failure rates, all of which are essential for understanding application performance. It can also provide end-to-end transaction tracing.

Azure Log Analytics is a service that lets you collect, analyze, and act on log data from various sources, including Azure resources. While Log Analytics could be used to indirectly determine page load times if you specifically log that information, it is not the best tool for the specific purpose outlined in the scenario. It's better suited for broader logging and analysis.

Azure Advisor is a personalized cloud consultant that helps you follow best practices to optimize your Azure deployments. While it can offer recommendations related to performance, it doesn't provide detailed application performance metrics like page load times.

Therefore, the best replacement for Azure Event Hubs in this context is Azure Application Insights, as it directly and effectively monitors application performance metrics, including page load times.

References:

Azure Application Insights
Azure Event Hubs

Question: 27 CertyIQ

Your company makes use of Azure SQL Database Intelligent Insights and Azure Application Insights for monitoring purposes.
You have been tasked with analyzing the monitoring using ad-hoc queries. You need to utilize the correct query language.
Solution: You use the Contextual Query Language (CQL).
Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution suggests using Contextual Query Language (CQL) for ad-hoc analysis of monitoring data from Azure SQL Database Intelligent Insights and Azure Application Insights. This is incorrect.

Azure SQL Database Intelligent Insights primarily surfaces performance recommendations and potential issues, but direct querying of its underlying data isn't typically done via a specific query language. Instead, you'd review the insights within the Azure portal or access them programmatically through the Azure Resource Manager API.

Azure Application Insights primarily relies on the Kusto Query Language (KQL) for querying its telemetry data. KQL is specifically designed for exploring and analyzing large volumes of log and telemetry data. It is used within the Azure Monitor Logs service and throughout various Azure services for analytics.

CQL, although a valid query language used in certain contexts (e.g., Cassandra databases), is not relevant or applicable for querying data from Azure SQL Database Intelligent Insights or Azure Application Insights. Therefore, the solution is fundamentally flawed as it recommends an incorrect query language for the given Azure services.

In conclusion, CQL is not the correct query language for analyzing data from Azure SQL Database Intelligent Insights or Azure Application Insights. KQL is the language designed and used for Azure Application Insights.

Supporting Links:

Kusto Query Language (KQL): https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/

Azure Application Insights: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
Azure SQL Database Intelligent Insights: https://learn.microsoft.com/en-us/azure/azure-sql/database/intelligent-insights-overview?view=azuresql

Question: 28 CertyIQ

Your company makes use of Azure SQL Database Intelligent Insights and Azure Application Insights for monitoring purposes.
You have been tasked with analyzing the monitoring using ad-hoc queries. You need to utilize the correct query language.
Solution: You use the Transact-SQL.
Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The solution proposes using Transact-SQL (T-SQL) for ad-hoc querying of data from Azure SQL Database Intelligent Insights and Azure Application Insights. This is incorrect.

T-SQL is specifically designed for querying relational databases, like Azure SQL Database. While Intelligent Insights integrates with Azure SQL Database, Application Insights stores its data in Azure Monitor Logs, which uses the Kusto Query Language (KQL).

Application Insights stores telemetry data (logs, metrics, traces) optimized for analytics. KQL is tailored for querying this type of data, offering operators and functions for time-series analysis, pattern recognition, and aggregations crucial for application monitoring.

Attempting to use T-SQL against Application Insights data would not work. KQL is the correct and intended language for querying logs and telemetry data stored in Azure Monitor Logs, including data originating from Application Insights. Therefore, the proposed solution is incorrect, and the answer is "No".

For further research:

Kusto Query Language (KQL) Overview: https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/
Azure Application Insights: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
Azure Monitor Logs: https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-platform-logs

Question: 29 CertyIQ

Your company makes use of Azure SQL Database Intelligent Insights and Azure Application Insights for monitoring purposes.
You have been tasked with analyzing the monitoring using ad-hoc queries. You need to utilize the correct query language.
Solution: You use Azure Log Analytics.
Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of using Azure Log Analytics to query data from Azure SQL Database Intelligent Insights and Azure Application Insights is incorrect. Azure Log Analytics primarily uses the Kusto Query Language (KQL) and focuses on data stored within its own workspace.

While Azure SQL Database Intelligent Insights does provide performance data that can be sent to Log Analytics, it's more commonly analyzed directly through the Azure portal's Intelligent Performance feature, Azure Data Studio, or programmatically via the Azure Monitor REST API. Application Insights data can be queried from Log Analytics if configured to do so, however, native querying for Application Insights is performed via the Application Insights Analytics portal, using KQL directly against the Application Insights resource.

For ad-hoc querying of Azure SQL Database Intelligent Insights data, options include querying against diagnostic logs stored in a storage account, using the Azure Monitor REST API, or reviewing within Azure SQL Database performance dashboards. While Application Insights data can indeed be channeled to Log Analytics, the native and preferred method for interactive exploration and ad-hoc querying often involves directly querying within the Application Insights Analytics portal using KQL or Power BI. Therefore, simply stating "use Azure Log Analytics" misses the nuance of the different query options and might not be the most efficient or direct approach in all scenarios, particularly for SQL Intelligent Insights.

Therefore, the solution is incorrect. Log Analytics can be part of a broader monitoring solution, but it's not the sole tool or always the most direct approach for ad-hoc queries across both services.

Relevant Links:

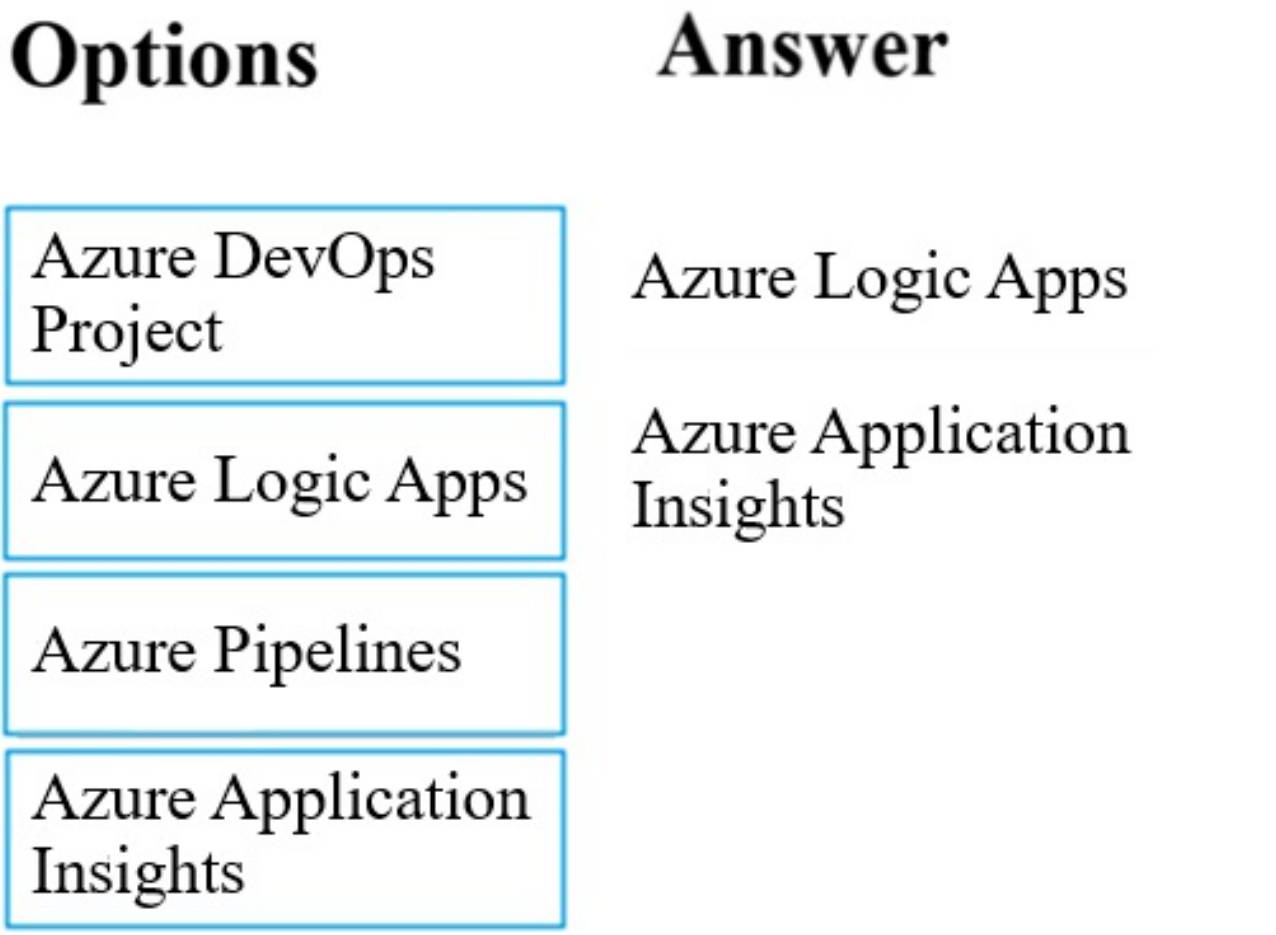
Azure SQL Database Intelligent Insights: https://learn.microsoft.com/en-us/azure/azure-sql/database/intelligent-insights-overview?view=azuresql
Azure Application Insights Analytics: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
Azure Log Analytics: https://learn.microsoft.com/en-us/azure/azure-monitor/logs/log-analytics-overview

Question: 30 CertyIQ

DRAG DROP -
You have recently created a web application for your company.
You have been tasked with making sure that a summary of the exceptions that transpire in the application is automatically sent to Microsoft Teams on a daily basis.
Which of the following Azure services should you use? Answer by dragging the correct options from the list to the answer area.
Select and Place:

Options	Answer
Azure DevOps Project	
Azure Logic Apps	
Azure Pipelines	
Azure Application Insights	

Answer:



Explanation:
Exceptions in your live web app are reported by Application Insights.
Note: Periodical reports help keep a team informed on how their business critical services are doing. Developers, DevOps/SRE teams, and their managers can be productive with automated reports reliably delivering insights without requiring everyone to sign in the portal. Such reports can also help identify gradual increases in latencies, load or failure rates that may not trigger any alert rules.
You can programmatically query Application Insights data to generate custom reports on a schedule. The following options can help you get started quickly:
Automate reports with Microsoft Flow
- Automate reports with Logic Apps

Reference:
https://docs.microsoft.com/en-us/azure/azure-monitor/app/asp-net-exceptions
https://docs.microsoft.com/en-us/azure/azure-monitor/app/automate-custom-reports

Question: 31 CertyIQ

You are in the process of building a mobile app aimed at Android and iOS devices.
All work items and release cycles are managed via Azure DevOps.
You want to make sure that crash reports for issue analysis is collected, and that beta releases are distributed to your testers. Also, you want to ensure that user feedback on the functionality of new apps is received.
Which of the following must be part of your solution?

A. The Microsoft Test & Feedback extension.
B. OWASP ZAP

C. TFS Integration Platform

D. Code Style

Answer: A

Explanation:

The correct answer is A, the Microsoft Test & Feedback extension, because it directly addresses the stated requirements for a mobile app development scenario managed with Azure DevOps. The scenario focuses on collecting crash reports, distributing beta releases to testers, and gathering user feedback on new app functionalities. The Test & Feedback extension enables testers to submit feedback directly within the Azure DevOps environment, including detailed bug reports and suggestions. It streamlines the process of gathering user feedback and centralizes communication, making it easier for the development team to address issues.

Here's why the other options are incorrect:

B. OWASP ZAP: OWASP ZAP is a security testing tool used for finding vulnerabilities in web applications. While security testing is crucial, it doesn't directly address the requirements of crash reporting, beta distribution, or user feedback collection.

C. TFS Integration Platform: The TFS Integration Platform is a tool for synchronizing data between different Team Foundation Server (TFS) instances. It's not designed for mobile app testing or feedback management.

D. Code Style: Code Style refers to a set of rules or guidelines that define the format and structure of source code. While important for code maintainability, it doesn't relate to crash reporting, beta distribution, or user feedback collection.

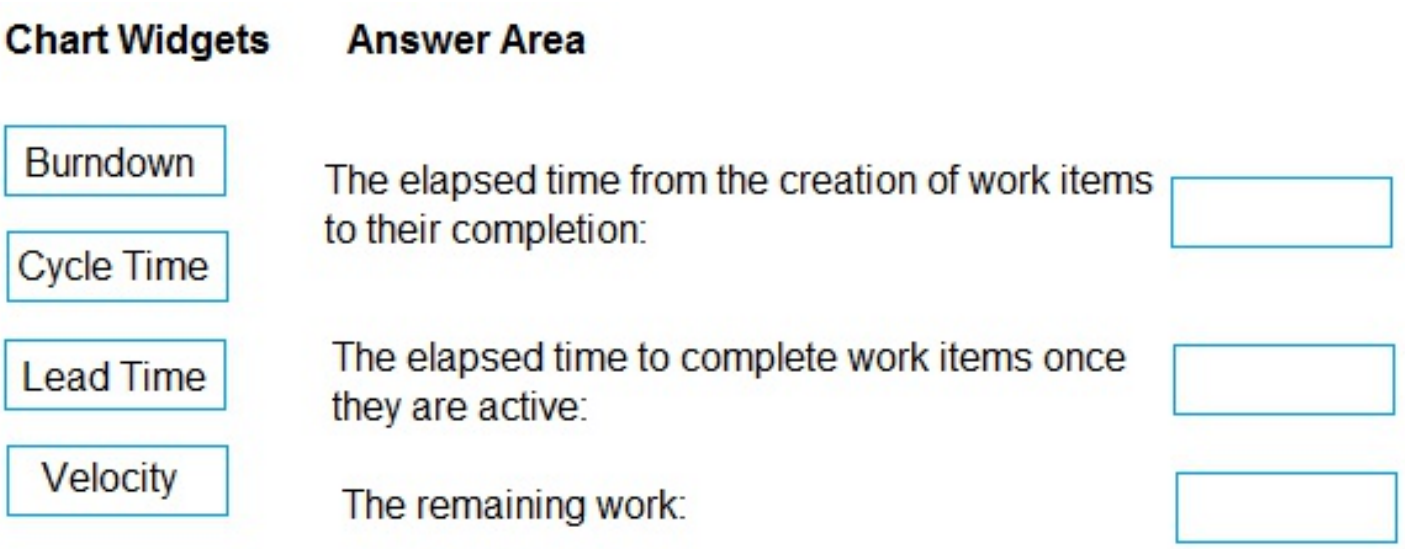
In conclusion, the Microsoft Test & Feedback extension is the most appropriate solution for the given scenario as it directly supports the collection of crash reports through user feedback integration within the Azure DevOps ecosystem, facilitating a streamlined workflow for issue analysis and feature enhancement.

Supporting resources:

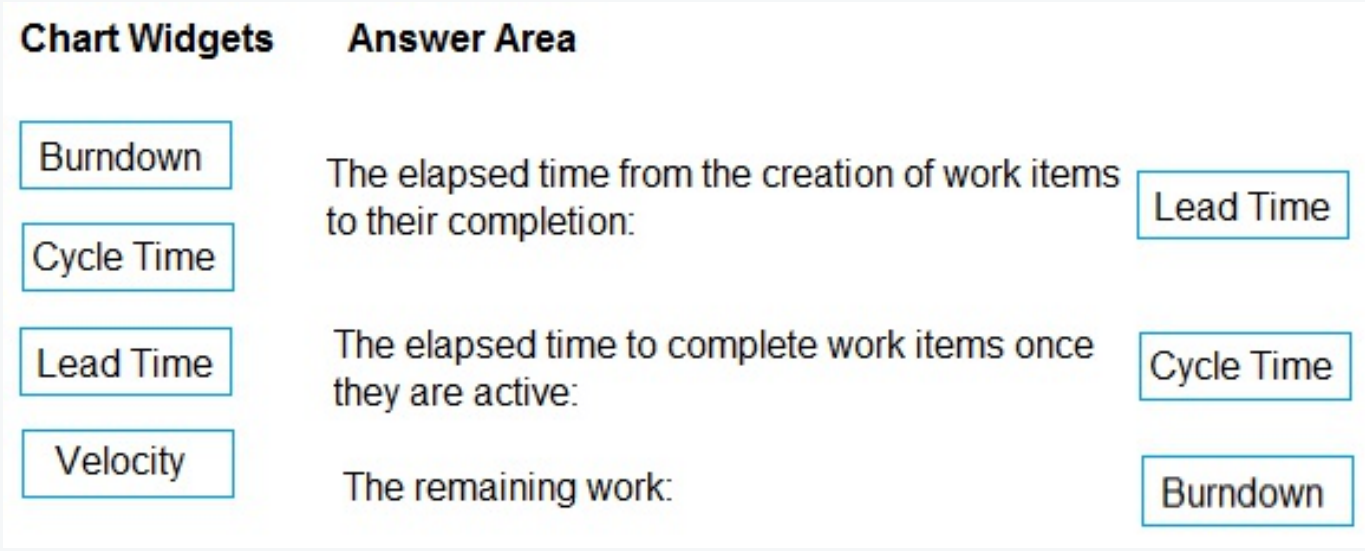
Microsoft Test & Feedback Extension: https://marketplace.visualstudio.com/items?itemName=ms-devlabs.test-feedback
Gather stakeholder feedback using Test & Feedback Extension: https://learn.microsoft.com/en-us/azure/devops/test/feedback/feedback?view=azure-devops

Question: 32 CertyIQ

DRAG DROP -
You need to recommend project metrics for dashboards in Azure DevOps.
Which chart widgets should you recommend for each metric? To answer, drag the appropriate chart widgets to the correct metrics. Each chart widget may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:
Box 1: Lead time -
Lead time measures the total time elapsed from the creation of work items to their completion.

Box 2: Cycle time -
Cycle time measures the time it takes for your team to complete work items once they begin actively working on them.

Box 3: Burndown -
Burndown charts focus on remaining work within a specific time period.
Incorrect Answers:
Velocity provides a useful metric for these activities:

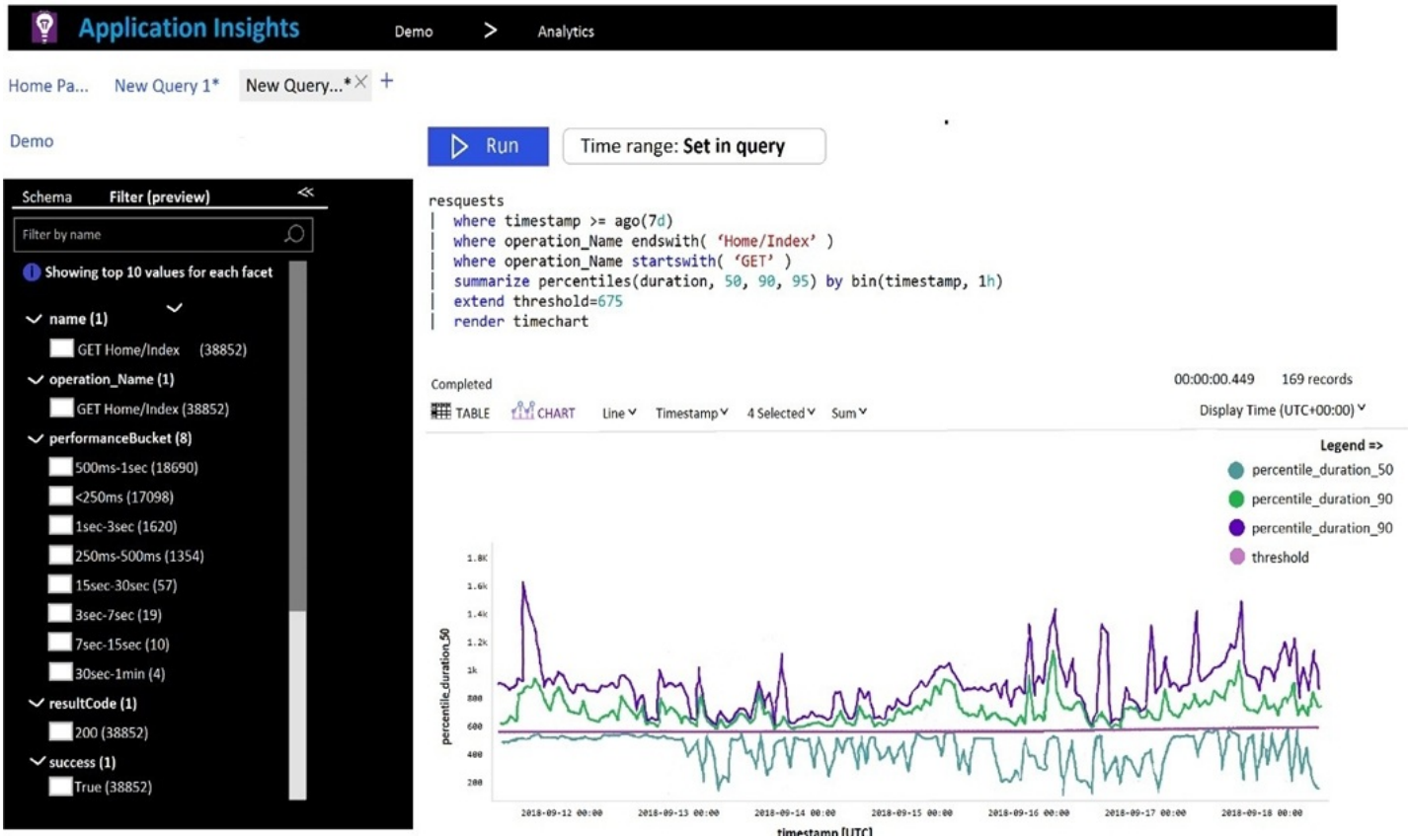
Support sprint planning -
Forecast future sprints and the backlog items that can be completed
A guide for determining how well the team estimates and meets their planned commitments

Reference:
https://docs.microsoft.com/en-us/azure/devops/report/dashboards/velocity-guidance?view=vsts
https://docs.microsoft.com/en-us/azure/devops/report/dashboards/cycle-time-and-lead-time?view=vsts
https://docs.microsoft.com/en-us/azure/devops/report/dashboards/configure-burndown-burnup-widgets?view=vsts

Question: 33

CertyIQ

HOTSPOT -
You plan to create alerts that will be triggered based on the page load performance of a home page.
You have the Application Insights log query shown in the following exhibit.



Use the drop-down menus to select the answer choice that completes each statement based on the information presented in the graphic.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

To create an alert based on the page load experience of most users, the alerting level must be based on **[answer choice]**.

- percentile_duration_50
- percentile_duration_90
- percentile_duration_95
- threshold

To only create an alert when authentication error occurs on the server, the query must be filtered on **[answer choice]**.

- item Type
- resultCode
- source
- success

Answer:

Answer Area

To create an alert based on the page load experience of most users, the alerting level must be based on **[answer choice].**

- percentile_duration_50
- percentile_duration_90
- **percentile_duration_95**
- threshold

To only create an alert when authentication error occurs on the server, the query must be filtered on **[answer choice].**

- item Type
- resultCode
- source
- **success**

Explanation:
Box 1: percentile_duration_95 -

Box 2: success -
For example "
requests
| project name, url, success
| where success == "False"
This will return all the failed requests in my App Insights within the specified time range.

Reference:
https://devblogs.microsoft.com/premier-developer/alerts-based-on-analytics-query-using-custom-log-search/

Question: 34 CertyIQ

You manage an Azure web app that supports an e-commerce website.
You need to increase the logging level when the web app exceeds normal usage patterns. The solution must minimize administrative overhead.
Which two resources should you include in the solution? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. an Azure Automation runbook
B. an Azure Monitor alert that has a dynamic threshold
C. an Azure Monitor alert that has a static threshold
D. the Azure Monitor autoscale settings
E. an Azure Monitor alert that uses an action group that has an email action

Answer: AB

Explanation:

The correct answer is AB. Here's why:

A. Azure Automation Runbook: An Azure Automation runbook allows you to automate tasks within your Azure environment. In this scenario, the runbook's purpose would be to dynamically adjust the logging level of the web app. When triggered, the runbook can modify the web app's configuration to increase the logging detail, capturing more information about the application's behavior. This is a key component for responding to increased usage without manual intervention.

B. Azure Monitor Alert with Dynamic Threshold: Azure Monitor alerts are crucial for detecting when the web app's usage patterns deviate from the norm. Using a dynamic threshold is essential here because normal usage can fluctuate over time (e.g., daily/weekly cycles, seasonal spikes). A dynamic threshold uses machine learning to establish a baseline of "normal" behavior and then triggers an alert when metrics exceed that baseline by a specified amount. This avoids false positives that could occur with a static threshold. Once the alert is triggered, it can execute an action group that initiates the Azure Automation runbook.

Why the other options are incorrect:

C. Azure Monitor Alert with Static Threshold: While an alert is necessary, using a static threshold would be problematic. A static threshold defines a fixed value as the trigger point. As the question states, "normal usage patterns" exist, so a single, static threshold could result in alerts being triggered when the application is merely operating within expected bounds (false positives).

D. Azure Monitor Autoscale Settings: Autoscale is designed to adjust the number of instances of a service (e.g., increasing the number of web app instances) to handle increased load. While scaling could be relevant in a broader solution, it doesn't directly address the requirement of increasing logging levels. It addresses resource allocation, not logging granularity.

E. Azure Monitor Alert with Email Action: Email action is only one part of the solution. The action must be triggered by an increased logging level. The email action could be combined with the Azure Automation Runbook in the action group.

In summary, using a dynamic threshold alert in Azure Monitor detects abnormal usage patterns. When triggered, the alert initiates an Azure Automation runbook that programmatically increases the logging level in the web app. This provides a flexible, automated solution that minimizes administrative overhead.

Supporting Links:

Azure Automation: https://learn.microsoft.com/en-us/azure/automation/
Azure Monitor Alerts: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-overview
Dynamic Thresholds in Azure Monitor: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-dynamic-thresholds

Question: 35 CertyIQ

HOTSPOT -
You have an Azure Kubernetes Service (AKS) pod.
You need to configure a probe to perform the following actions:
- Confirm that the pod is responding to service requests.
- Check the status of the pod four times a minute.
- Initiate a shutdown if the pod is unresponsive.

How should you complete the YAML configuration file? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

```
apiVersion: v1
kind: Pod
metadata:
     labels:
          test: readiness-and-liveness
     name: readiness-http
spec:
     containers:
     - name: container1
       image: k8s.gcr.io/readiness-and-liveness
       args:
       - /server
```

livenessProbe:
readinessProbe:
ShutdownProbe:
startupProbe:

```
    httpGet:
         path: /checknow
         port: 8123
         httpHeaders:
         - name: Custom-Header
           value: CheckNow
```

initialDelaySeconds: 15
periodSeconds: 15
timeoutSeconds: 15

Answer:

Answer Area

```
apiVersion: v1
kind: Pod
metadata:
     labels:
          test: readiness-and-liveness
     name: readiness-http
spec:
     containers:
     - name: container1
       image: k8s.gcr.io/readiness-and-liveness
       args:
       - /server
```

livenessProbe: | readinessProbe: | ShutdownProbe: | startupProbe:

```
    httpGet:
         path: /checknow
         port: 8123
         httpHeaders:
         - name: Custom-Header
           value: CheckNow
```

initialDelaySeconds: 15 | periodSeconds: 15 | timeoutSeconds: 15

Explanation:

Liveness Probe.

periodSeconds: 15.

The periodSeconds property designates the readiness command should execute every 15 seconds.

Reference:

https://kubernetes.io/fr/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/

Question: 36 CertyIQ

You have a Microsoft ASP.NET Core web app in Azure that is accessed worldwide.
You need to run a URL ping test once every five minutes and create an alert when the web app is unavailable from specific Azure regions. The solution must minimize development time.
What should you do?

A. Create an Azure Monitor Availability metric and alert.
B. Create an Azure Application Insights availability test and alert.
C. Write an Azure function and deploy the function to the specific regions.
D. Create an Azure Service Health alert for the specific regions.

Answer: B

Explanation:

The correct answer is **B. Create an Azure Application Insights availability test and alert.**

Here's why:

Azure Application Insights Availability Tests: Application Insights is a powerful Application Performance Management (APM) service that allows you to monitor the availability and performance of your web applications. Its availability tests are specifically designed to proactively check the reachability and responsiveness of your web app from different locations around the world. This perfectly aligns with the requirement to run a URL ping test every five minutes and alert on unavailability from specific Azure regions. https://learn.microsoft.com/en-us/azure/azure-monitor/app/monitor-web-app-availability

Out-of-the-box Functionality: Application Insights provides these tests out-of-the-box, minimizing development time. You can easily configure a URL ping test with desired frequency (5 minutes), specify target URLs, and select Azure regions from which to run the tests.

Alerting: Application Insights integrates seamlessly with Azure Monitor alerts. You can set up alerts based on the results of the availability tests. For example, you can create an alert rule that triggers if the test fails from a specific region.

Why other options are less suitable:

A. Azure Monitor Availability Metric and Alert: While Azure Monitor can monitor overall availability, it doesn't directly provide a way to create region-specific URL ping tests as easily as Application Insights. You'd need to build more custom logic to achieve the same outcome.

C. Azure Function: Writing an Azure Function to perform URL ping tests and deploy it to multiple regions would be significantly more development effort than using Application Insights' built-in functionality. This violates the "minimize development time" requirement. It also requires managing the function deployments and scaling.

D. Azure Service Health Alert: Azure Service Health alerts are for platform-level incidents affecting Azure services themselves, not for monitoring the availability of your specific web application. This option is irrelevant to the question.

In summary, Application Insights availability tests provide a quick, easy, and code-free way to monitor your web app's availability from different regions and set up alerts based on the results, directly addressing the requirements of the scenario while minimizing development time.

Question: 37 CertyIQ

You have a multi-tier application. The front end of the application is hosted in Azure App Service.
You need to identify the average load times of the application pages.
What should you use?

A. Azure Application Insights
B. the activity log of the App Service
C. the diagnostics logs of the App Service
D. Azure Advisor

Answer: A

Explanation:

The correct answer is A, Azure Application Insights. Here's why:

Application Insights is an Application Performance Management (APM) service in Azure specifically designed

to monitor the availability, performance, and usage of web applications, including those hosted in App Service. It automatically detects performance anomalies and provides powerful analytics tools to help diagnose issues and understand how users are interacting with the application.

Specifically, Application Insights automatically tracks page load times for your application's pages when the JavaScript snippet is correctly installed on the front end. This provides valuable data for identifying slow-loading pages and understanding the user experience. This data is collected through client-side telemetry and aggregated for analysis.

Option B, the App Service activity log, primarily tracks administrative operations performed on the App Service itself (e.g., starting, stopping, scaling). While helpful for auditing changes to the App Service, it doesn't provide detailed performance metrics about application page load times.

Option C, App Service diagnostics logs, contain information about the health and status of the App Service and the underlying infrastructure. While useful for troubleshooting server-side issues, these logs do not directly provide aggregated page load time metrics. They contain information on HTTP errors, IIS logs, and similar data. Analyzing these logs to derive page load times would be a manual and inefficient process compared to using Application Insights.

Option D, Azure Advisor, provides recommendations for optimizing Azure resources for cost, security, reliability, and performance. While Advisor might identify general performance bottlenecks, it doesn't provide the detailed page load time metrics needed for pinpointing slow-loading pages within an application.

In summary, Application Insights is the most appropriate tool because it's purpose-built for application performance monitoring and provides automatic tracking and analysis of page load times, directly addressing the stated requirement.

For more information, refer to:

Azure Application Insights Overview: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
Performance monitoring with Application Insights: https://learn.microsoft.com/en-us/azure/azure-monitor/app/performance-monitoring

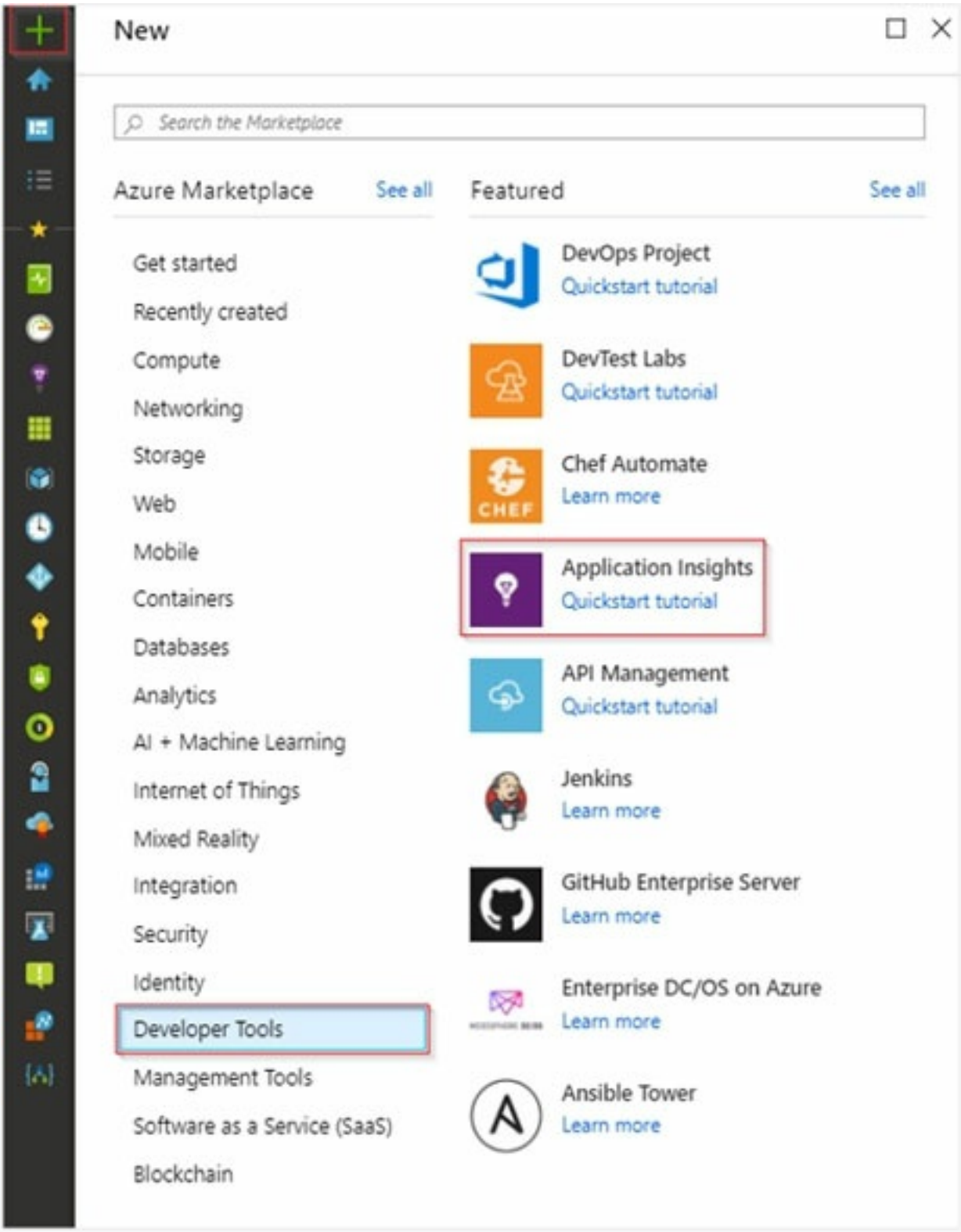
Question: 38 CertyIQ

SIMULATION -
You need to create an instance of Azure Application Insights named az400-123456789-main and configure the instance to receive telemetry data from an Azure web app named az400-123456789-main.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:
Step 1: Create an instance of Azure Application Insights
1. Open Microsoft Azure Portal
2. Log into your Azure account, Select Create a resource > Developer tools > Application Insights.

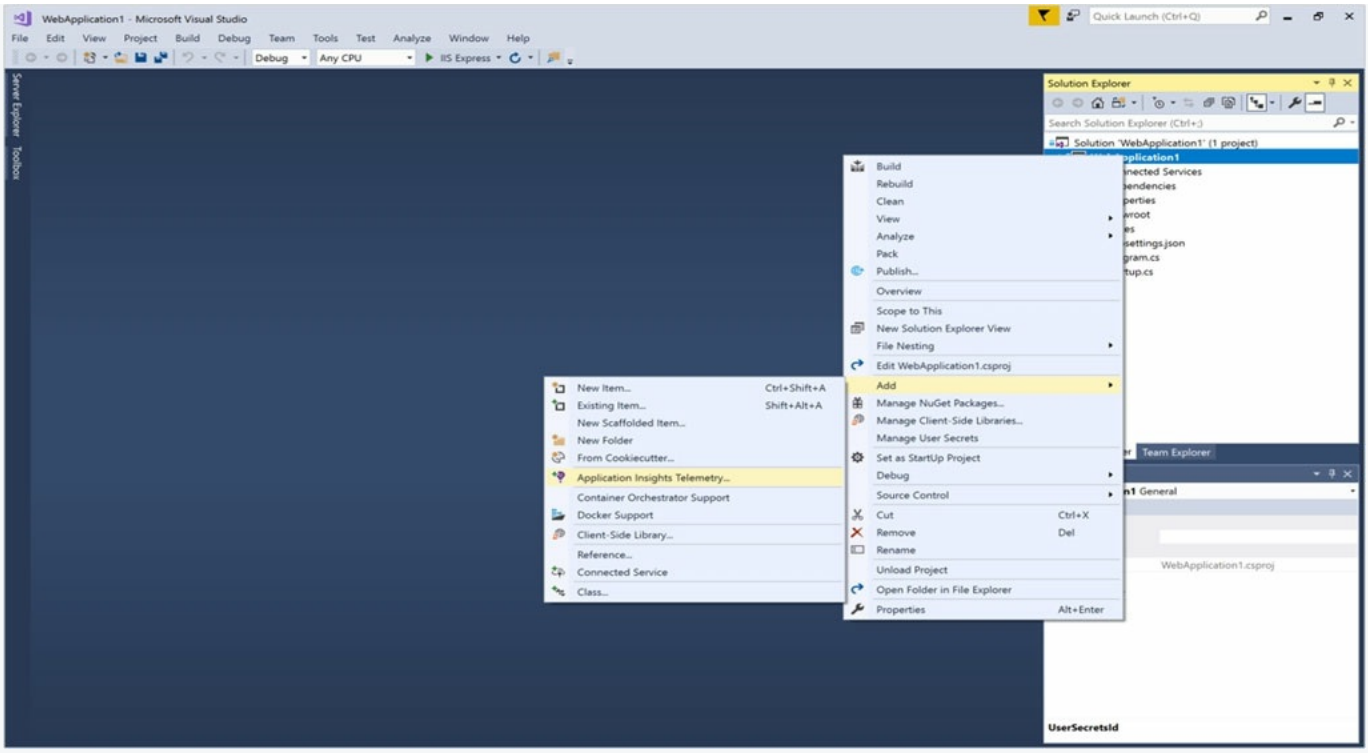


3. Enter the following settings, and then select Review + create.

Name: az400-123456789-main -

Step 2: Configure App Insights SDK

1. Open your ASP.NET Core Web App project in Visual Studio > Right-click on the AppName in the Solution Explorer > Select Add > Application Insights
Telemetry.



2. Click the Get Started button
3. Select your account and subscription > Select the Existing resource you created in the Azure portal > Click Register.

Reference:
https://docs.microsoft.com/bs-latn-ba/azure/azure-monitor/learn/dotnetcore-quick-start?view=vs-2017

Question: 39

CertyIQ

Your company uses ServiceNow for incident management.
You develop an application that runs on Azure.
The company needs to generate a ticket in ServiceNow when the application fails to authenticate.
Which Azure Log Analytics solution should you use?

A. Application Insights Connector
B. Automation & Control
C. IT Service Management Connector (ITSM)
D. Insight & Analytics

Answer: C

Explanation:

The correct answer is C, IT Service Management Connector (ITSM). Here's why:

The core requirement is to integrate Azure Log Analytics with ServiceNow, specifically to generate tickets in ServiceNow based on events within Azure. Among the options, the IT Service Management Connector (ITSM) is specifically designed for this purpose. It provides a direct connection between Azure monitoring tools and ITSM solutions like ServiceNow.

The ITSM Connector allows you to synchronize work items (like incidents) between Azure alerts (which can be triggered by authentication failures logged in Log Analytics) and ServiceNow. You can configure rules within the ITSM Connector to automatically create ServiceNow tickets based on specific Azure alerts. This automates the incident management process triggered by application authentication failures.

Application Insights Connector (A) is primarily for exporting Application Insights telemetry data for more advanced analytics purposes. Automation & Control (B) encompasses broader automation tasks and doesn't focus directly on ITSM integration. Insight & Analytics (D) is a general category within Log Analytics, but doesn't provide a specific integration component like the ITSM Connector.

Therefore, the ITSM Connector is the most suitable solution because it directly facilitates the integration between Azure Log Analytics and ServiceNow for incident creation, addressing the stated requirement precisely. It enables proactive incident management and ensures that authentication failures in the Azure application are promptly addressed through the established ServiceNow workflow.

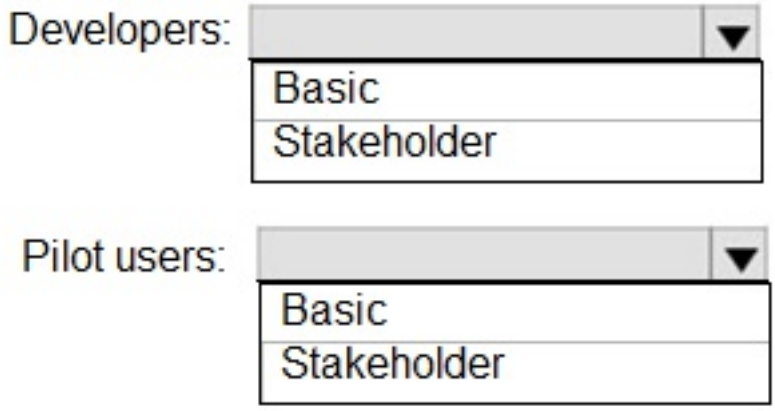
For further research, refer to the official Microsoft documentation:

Connect Azure to ITSM tools using IT Service Management Connector: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/itsmc-overview

Question: 40 CertyIQ

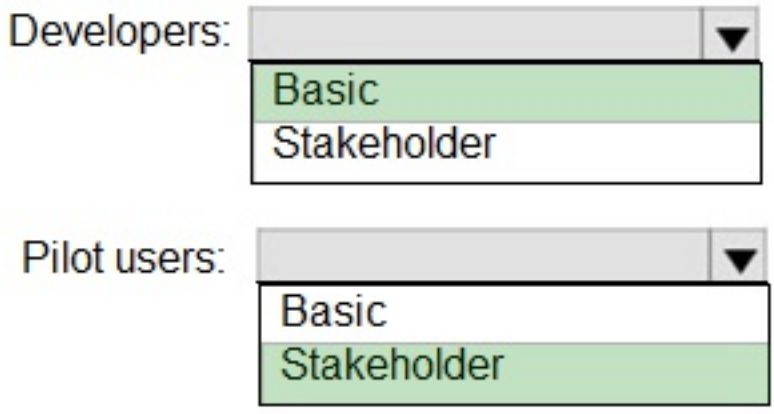
HOTSPOT -
Your company is building a new web application.
You plan to collect feedback from pilot users on the features being delivered.
All the pilot users have a corporate computer that has Google Chrome and the Microsoft Test & Feedback extension installed. The pilot users will test the application by using Chrome.
You need to identify which access levels are required to ensure that developers can request and gather feedback from the pilot users. The solution must use the principle of least privilege.
Which access levels in Azure DevOps should you identify? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:
Box 1: Basic -
Assign Basic to users with a TFS CAL, with a Visual Studio Professional subscription, and to users for whom you are paying for Azure Boards & Repos in an organization.

Box 2: Stakeholder -
Assign Stakeholders to users with no license or subscriptions who need access to a limited set of features.
Note:
You assign users or groups of users to one of the following access levels:
Basic: provides access to most features
VS Enterprise: provides access to premium features
Stakeholders: provides partial access, can be assigned to unlimited users for free

Reference:
https://docs.microsoft.com/en-us/azure/devops/organizations/security/access-levels?view=vsts

Question: 41 CertyIQ

You use Azure SQL Database Intelligent Insights and Azure Application Insights for monitoring.
You need to write ad-hoc queries against the monitoring data.
Which query language should you use?

A. Kusto Query Language (KQL)
B. PL/pgSQL
C. PL/SQL
D. Transact-SQL

Answer: A

Explanation:

The correct answer is A, Kusto Query Language (KQL). Azure SQL Database Intelligent Insights and Azure Application Insights both heavily leverage Azure Monitor. Azure Monitor employs a data platform built on

Azure Data Explorer (ADX) for log and metric data. ADX utilizes KQL as its primary query language. Therefore, to effectively query the monitoring data collected by Intelligent Insights and Application Insights, which is stored within Azure Monitor's data platform, KQL is the necessary and appropriate choice.

PL/pgSQL and PL/SQL are procedural languages primarily used with PostgreSQL and Oracle databases respectively, and are not designed for querying Azure Monitor logs. Transact-SQL (T-SQL) is the language used with Microsoft SQL Server and Azure SQL Database for relational data queries; while Azure SQL Database is involved in Intelligent Insights, the actual monitoring data being analyzed resides in Azure Monitor, not a standard SQL database. KQL's strength lies in analyzing vast amounts of semi-structured and unstructured data, which is precisely the kind of data stored by Azure Monitor, making it the optimal language for ad-hoc analysis in this scenario. KQL is optimized for time-series data analysis and offers powerful operators specifically designed for exploring telemetry data. This makes KQL a superior choice for querying logs and metrics from Azure services compared to the other SQL variants.

For further reading, refer to the official Microsoft documentation on Kusto Query Language and Azure Monitor logs:

Kusto Query Language (KQL) Overview
Azure Monitor Logs overview

Question: 42 CertyIQ

Your company creates a web application.
You need to recommend a solution that automatically sends to Microsoft Teams a daily summary of the exceptions that occur in the application.
Which two Azure services should you recommend? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Azure Logic Apps
B. Azure Pipelines
C. Microsoft Visual Studio App Center
D. Azure DevOps Project
E. Azure Application Insights

Answer: AE

Explanation:

The correct answer is A. Azure Logic Apps and E. Azure Application Insights. Here's why:

Azure Application Insights excels at monitoring web applications and detecting exceptions. It automatically collects data about application performance, usage, and errors, including details about exceptions thrown by the application. Therefore, it's the ideal service for identifying and gathering the exception data needed for the daily summary.

Azure Logic Apps is a cloud-based integration platform for automating workflows and integrating applications and data across various services. In this scenario, Logic Apps can be configured to run on a daily schedule. It can then query Application Insights to retrieve the exception data collected over the previous day. Finally, it can format this data into a summary and post it to a Microsoft Teams channel.

Azure Pipelines (B) is a CI/CD service primarily used for building and deploying applications, not for summarizing application exceptions and posting them to Teams on a schedule. Microsoft Visual Studio App Center (C) is for building, testing, and distributing mobile apps, not server-side web applications. Azure DevOps Project (D) simplifies the initial setup of an entire Azure DevOps pipeline, but it's not necessary for

this specific task of exception reporting; Logic Apps can handle the automated reporting more efficiently.

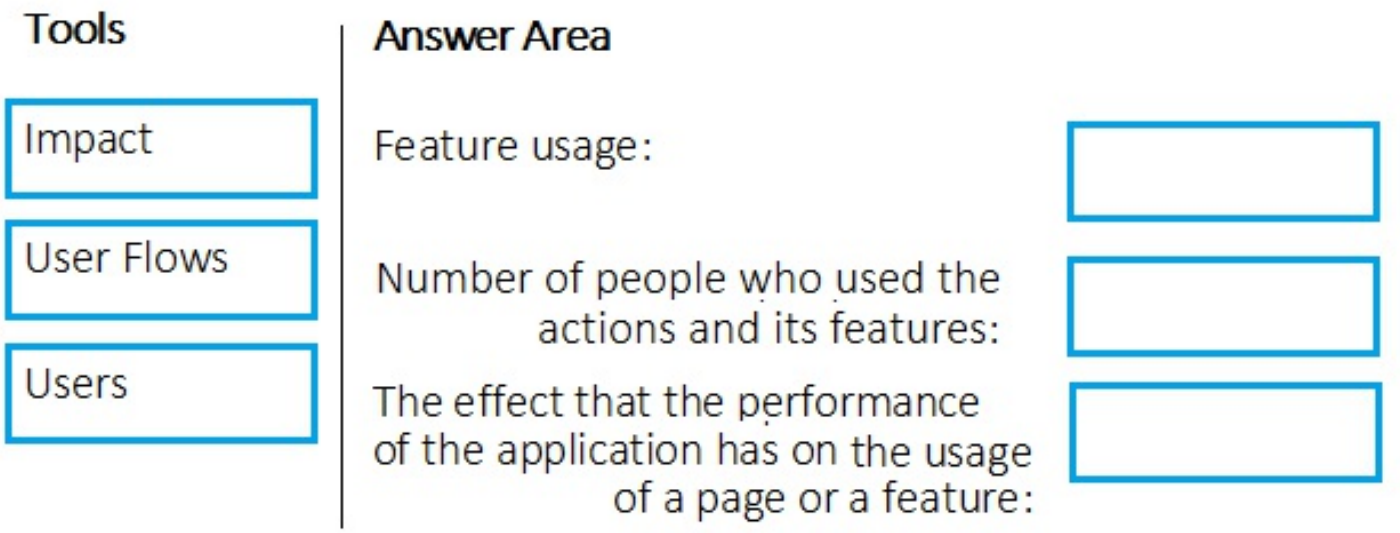
By combining Application Insights for exception detection and data collection, and Logic Apps for automated scheduling, data retrieval, formatting, and Teams integration, a fully automated solution for daily exception summaries can be implemented.

Authoritative Links:

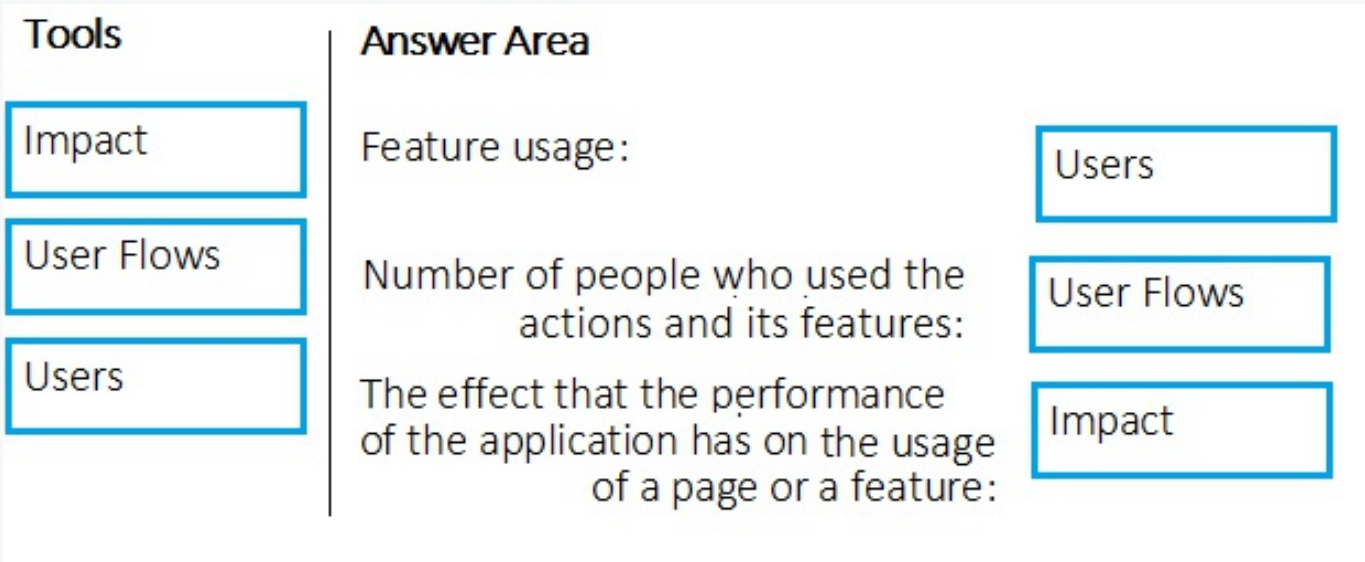
Azure Application Insights: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
Azure Logic Apps: https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-overview

Question: 43 CertyIQ

DRAG DROP -
Your company wants to use Azure Application Insights to understand how user behaviors affect an application.
Which Application Insights tool should you use to analyze each behavior? To answer, drag the appropriate tools to the correct behaviors. Each tool may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Users

Box 2: User Flows.

Box 3: Impact -

Impact analyzes how load times and other properties influence conversion rates for various parts of your app. To put it more precisely, it discovers how any dimension of a page view, custom event, or request affects the usage of a different page view or custom event.

Reference:

https://docs.microsoft.com/en-us/azure/azure-monitor/app/usage-segmentation

https://docs.microsoft.com/en-us/azure/azure-monitor/app/usage-flows

Question: 44 CertyIQ

Your company is building a mobile app that targets Android and iOS devices.
Your team uses Azure DevOps to manage all work items and release cycles.
You need to recommend a solution to perform the following tasks:
- Collect crash reports for issue analysis.
- Distribute beta releases to your testers.
- Get user feedback on the functionality of new apps.

What should you include in the recommendation?

A. the Microsoft Test & Feedback extension
B. Microsoft Visual Studio App Center integration
C. Azure Application Insights widgets
D. Jenkins integration

Answer: B

Explanation:

The correct answer is B, Microsoft Visual Studio App Center integration. Here's why:

App Center directly addresses all the requirements outlined in the scenario. It provides comprehensive mobile app lifecycle management features:

Crash Reporting: App Center automatically collects crash reports for both Android and iOS apps, allowing developers to identify and diagnose issues efficiently. It provides detailed crash reports and diagnostics that are essential for issue analysis.

Beta Distribution: App Center facilitates the distribution of beta releases to testers through its distribution groups. It provides a streamlined process for uploading new builds and notifying testers for easy installation, aligning with the need to distribute beta releases efficiently.

User Feedback: App Center supports user feedback through its in-app feedback features, enabling developers to gather valuable insights on new app functionalities directly from users. This allows for quick iteration and refinement based on real-world usage.

Other options are less suitable. The Microsoft Test & Feedback extension (A) is primarily designed for manual testing within Azure DevOps and does not cover automated crash reporting or beta distribution features. Azure Application Insights widgets (C) are geared more toward server-side applications, and while useful, they are not designed to address the specific needs of mobile app beta testing and distribution. Jenkins

integration (D) offers CI/CD capabilities, but does not provide built-in crash reporting or beta testing distribution capabilities. Visual Studio App Center provides a unified solution explicitly designed for mobile app development, testing, and deployment needs, making it the most suitable choice.

Authoritative Links:

Visual Studio App Center: https://appcenter.ms/
App Center Documentation: https://learn.microsoft.com/en-us/appcenter/

Question: 45 CertyIQ

You have an Azure DevOps project named Project1 and an Azure subscription named Sub1. Sub1 contains an Azure virtual machine scale set named VMSS1.
VMSS1 hosts a web application named WebApp1. WebApp1 uses stateful sessions.
The WebApp1 installation is managed by using the Custom Script extension. The script resides in an Azure Storage account named sa1.
You plan to make a minor change to a UI element of WebApp1 and to gather user feedback about the change.
You need to implement limited user testing for the new version of WebApp1 on VMSS1.
Which three actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Modify the load balancer settings of VMSS1.
B. Redeploy VMSS1.
C. Upload a custom script file to sa1.
D. Modify the Custom Script extension settings of VMSS1.
E. Update the configuration of a virtual machine in VMSS1.

Answer: CDE

Explanation:

The correct answer is CDE. Here's a detailed justification:

C. Upload a custom script file to sa1: The current WebApp1 installation is managed by a custom script in the storage account sa1. To deploy a new version with the UI change, you need to update the script in sa1 with the modified logic. This step ensures that the new version of the script is available to be applied to the virtual machines.

D. Modify the Custom Script extension settings of VMSS1: The Custom Script extension on VMSS1 points to the script in sa1. After uploading the new script, you need to modify the extension settings, potentially forcing it to re-run or specifying a new script location (if you prefer to maintain different script versions). Modifying these settings ensures that the updated script is actually executed on the virtual machine instances.

E. Update the configuration of a virtual machine in VMSS1: While VMSS allows for automatic updates across the instances, to implement limited user testing, you don't need to apply the changes to all virtual machines. After updating the script and re-running the custom script extension, you can configure one or a subset of virtual machines within VMSS1 to deploy new version of WebApp1.

Options A and B are not part of the solution:

A. Modify the load balancer settings of VMSS1: Modifying load balancer settings is useful for traffic routing but is not needed for the testing of the UI update. If you wanted to divert a specific percentage of users to the new version for A/B testing, then this would be necessary. However, the question only asks about testing.

B. Redeploy VMSS1: Redeploying the entire virtual machine scale set would be an overkill for a minor UI

change, especially considering you want to perform limited user testing. It is unnecessary to restart the full deployment.

Relevant links:

Azure virtual machine scale sets
Custom Script Extension for Windows
Custom Script Extension for Linux

Question: 46 CertyIQ

SIMULATION -
You need to create a notification if the peak average response time of an Azure web app named az400-123456789-main is more than five seconds when evaluated during a five-minute period. The notification must trigger the `https://contoso.com/notify` webhook.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below

Explanation:

1. Open Microsoft Azure Portal

2. Log into your Azure account and go to App Service and look under Monitoring then you will see Alert.

3. Select Add an alert rule

4. Configure the alert rule as per below and click Ok.

Source: Alert on Metrics -

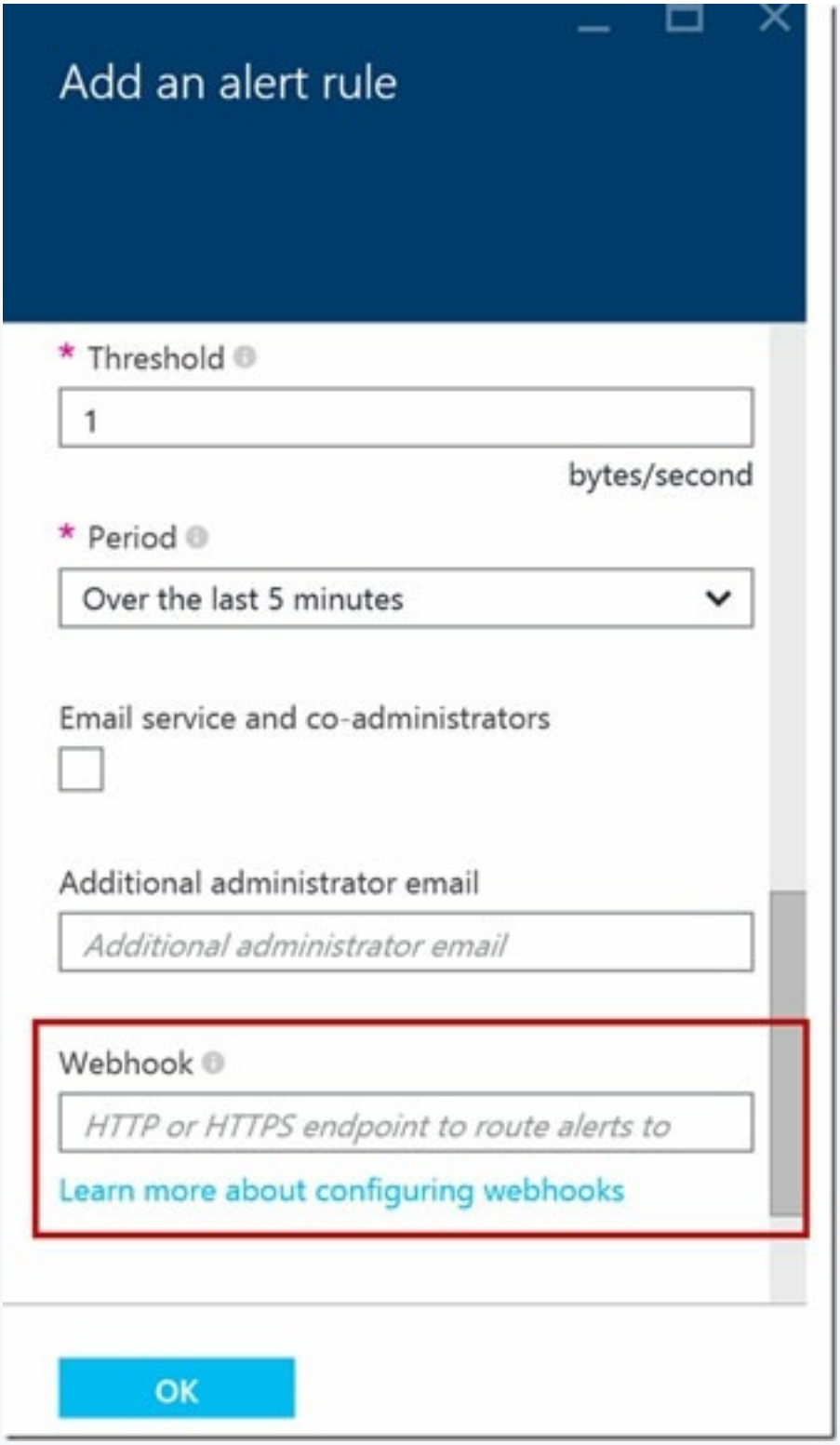
Resource Group: az400-123456789-main

Resource: az400-123456789-main -

Threshold: 5 -

Period: Over the last 5 minutes -

Webhook: https://contoso.com/notify



Reference:

https://azure.microsoft.com/es-es/blog/webhooks-for-azure-alerts/

Question: 47 CertyIQ

SIMULATION -
You need to create and configure an Azure Storage account named az400lod123456789stor in a resource group named RG1lod123456789 to store the boot diagnostics for a virtual machine named VM1.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:

Step 1: To create a general-purpose v2 storage account in the Azure portal, follow these steps:

1. On the Azure portal menu, select All services. In the list of resources, type Storage Accounts. As you begin typing, the list filters based on your input. Select

Storage Accounts.

2. On the Storage Accounts window that appears, choose Add.

3. Select the subscription in which to create the storage account.

4. Under the Resource group field, select RG1lod123456789

5. Next, enter a name for your storage account named: az400lod123456789stor

6. Select Create.

Step 2: Enable boot diagnostics on existing virtual machine

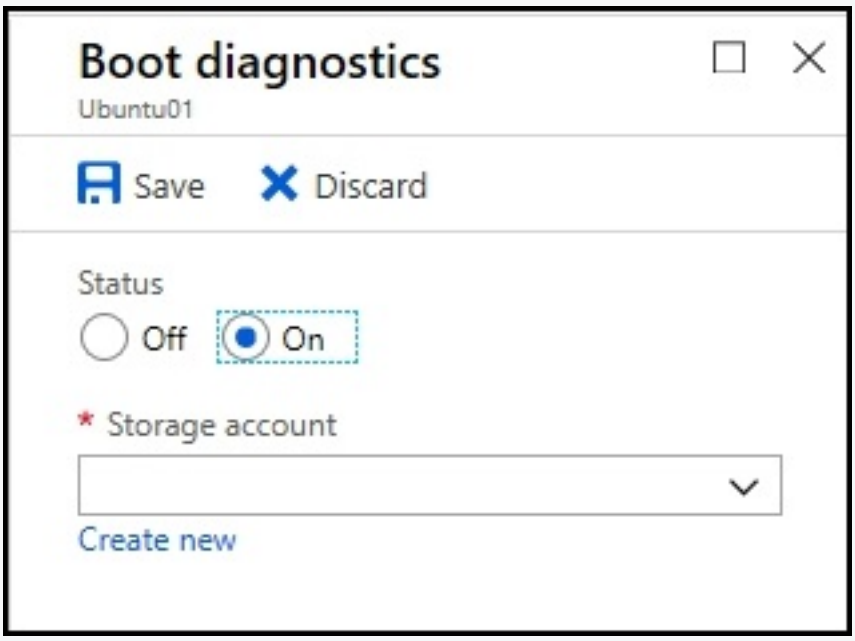
To enable Boot diagnostics on an existing virtual machine, follow these steps:

1. Sign in to the Azure portal, and then select the virtual machine VM1.

2. In the Support + troubleshooting section, select Boot diagnostics, then select the Settings tab.

3. In Boot diagnostics settings, change the status to On, and from the Storage account drop-down list, select the storage account az400lod123456789stor.

4. Save the change.



You must restart the virtual machine for the change to take effect.

Reference:

https://docs.microsoft.com/en-us/azure/storage/common/storage-account-create
https://docs.microsoft.com/en-us/azure/virtual-machines/troubleshooting/boot-diagnostics

Question: 48

CertyIQ

SIMULATION -
You have a web app that connects to an Azure SQL Database named db1.
You need to configure db1 to send Query Store runtime statistics to Azure Log Analytics.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

Explanation:

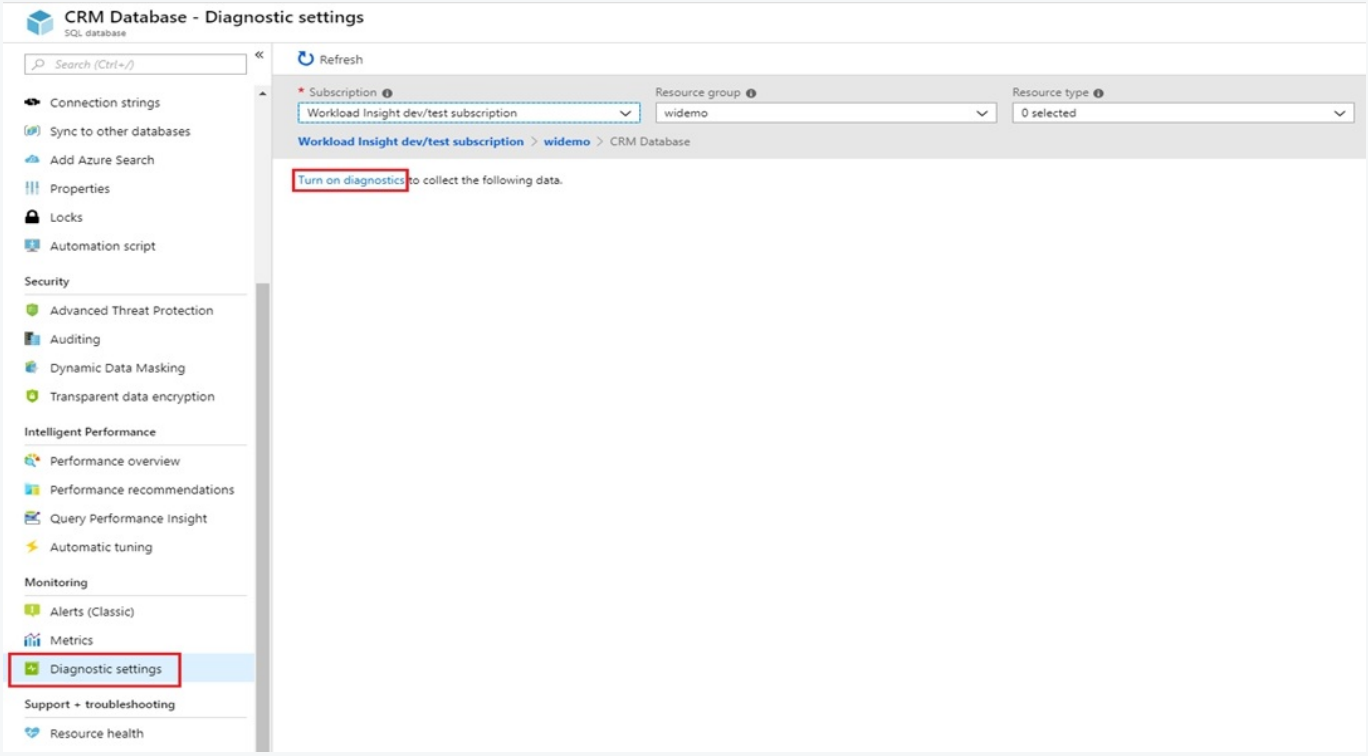
To enable streaming of diagnostic telemetry for a single or a pooled database, follow these steps:

1. Go to Azure SQL database resource.

2. Select Diagnostics settings.

3. Select Turn on diagnostics if no previous settings exist, or select Edit setting to edit a previous setting. You can create up to three parallel connections to stream diagnostic telemetry.

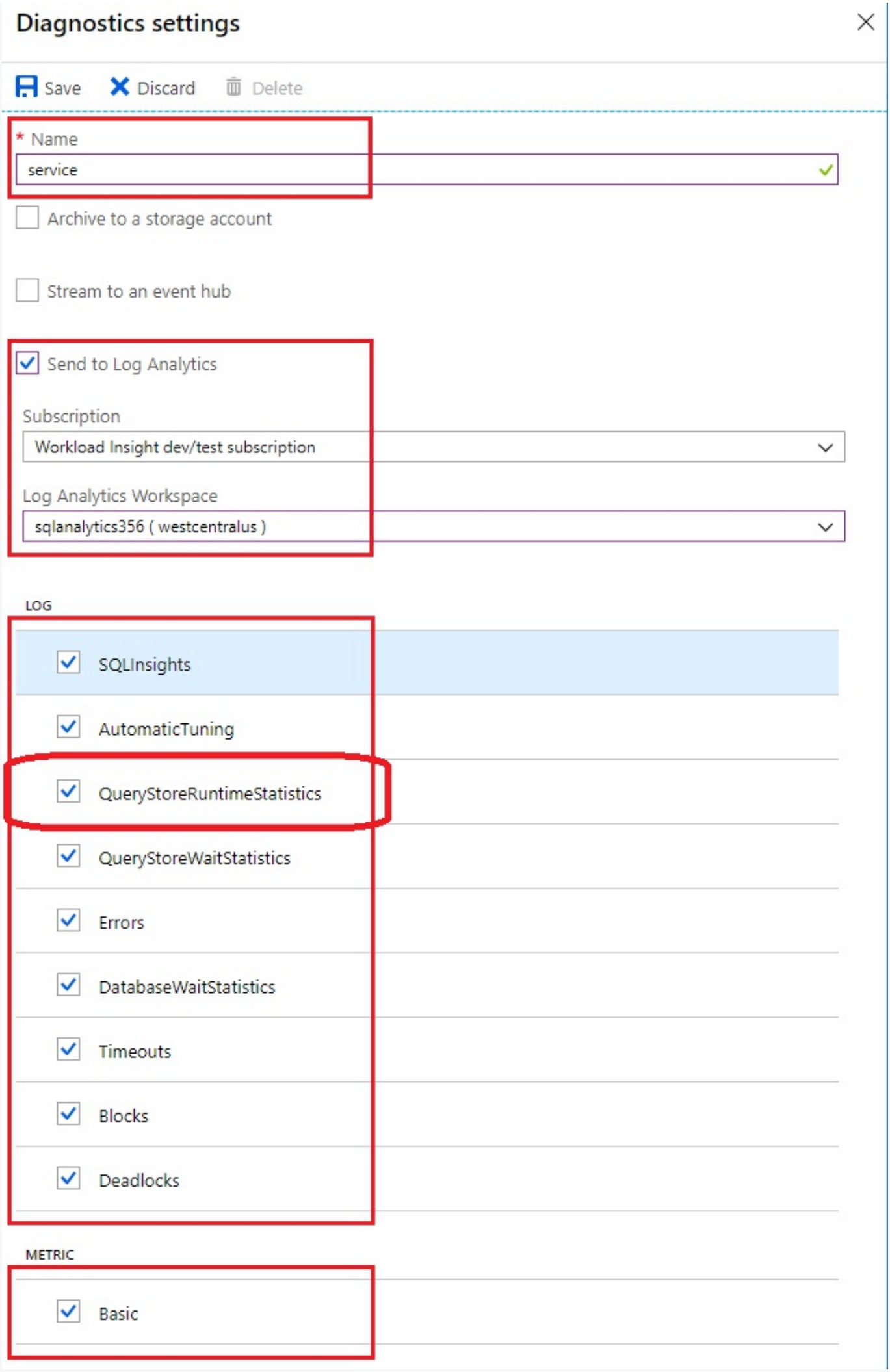
4. Select Add diagnostic setting to configure parallel streaming of diagnostics data to multiple resources.



5. Enter a setting name for your own reference.

6. Select a destination resource for the streaming diagnostics data: Archive to storage account, Stream to an event hub, or Send to Log Analytics.

7. For the standard, event-based monitoring experience, select the following check boxes for database diagnostics log telemetry: QueryStoreRuntimeStatistics


Diagnostics settings
Save
Discard
Delete
* Name
service
Archive to a storage account
Stream to an event hub
Send to Log Analytics
Subscription
Workload Insight dev/test subscription
Log Analytics Workspace
sqlanalytics356 (westcentralus)
LOG
SQLInsights
AutomaticTuning
QueryStoreRuntimeStatistics
QueryStoreWaitStatistics
Errors
DatabaseWaitStatistics
Timeouts
Blocks
Deadlocks
METRIC
Basic

8. For an advanced, one-minute-based monitoring experience, select the check box for Basic metrics.

9. Select Save.

Reference:

https://docs.microsoft.com/en-us/azure/azure-sql/database/metrics-diagnostic-telemetry-logging-streaming-export-configure

Question: 49 CertyIQ

DRAG DROP -
You have several Azure virtual machines that run Windows Server 2019.
You need to identify the distinct event IDs of each virtual machine as shown in the following table.

Name	Event ID
VM1	[704, 701, 1501, 1500, 1085]
VM2	[326, 105, 302, 301, 300, 102]
...	...

How should you complete the Azure Monitor query? To answer, drag the appropriate values to the correct locations. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:

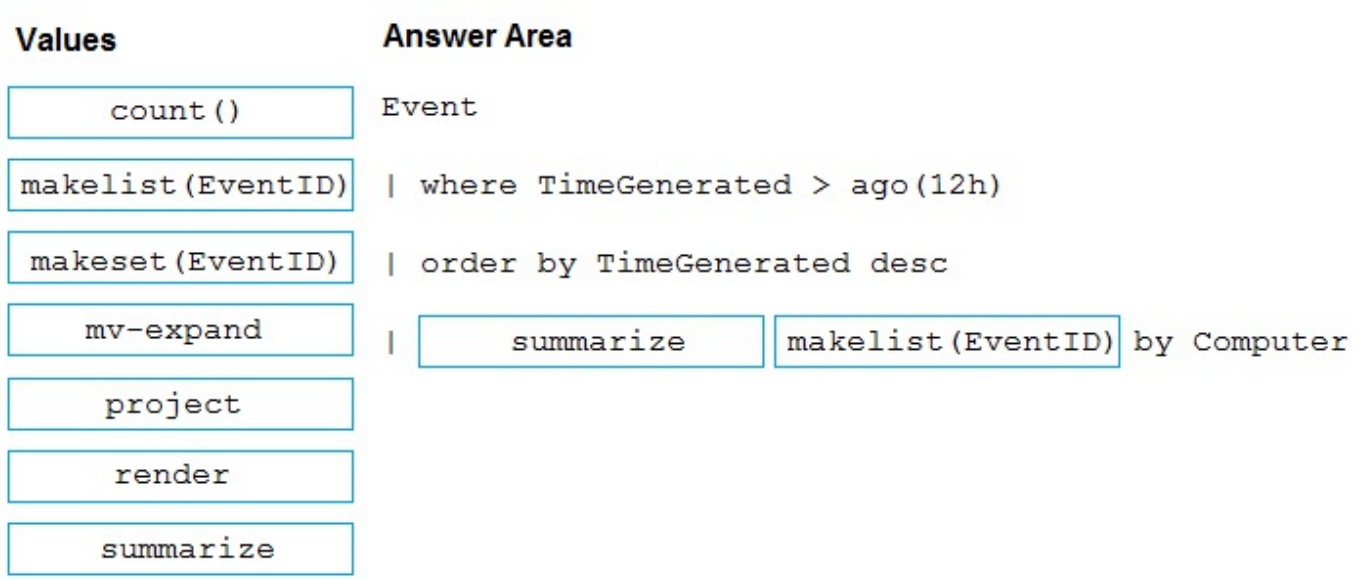
Values

- count()
- makelist(EventID)
- makeset(EventID)
- mv-expand
- project
- render
- summarize

Answer Area

```
Event
| where TimeGenerated > ago(12h)
| order by TimeGenerated desc
| [ ] [ ] by Computer
```

Answer:



Explanation:
You can use makelist to pivot data by the order of values in a particular column. For example, you may want to explore the most common order events take place on your machines. You can essentially pivot the data by the order of EventIDs on each machine.
Example:

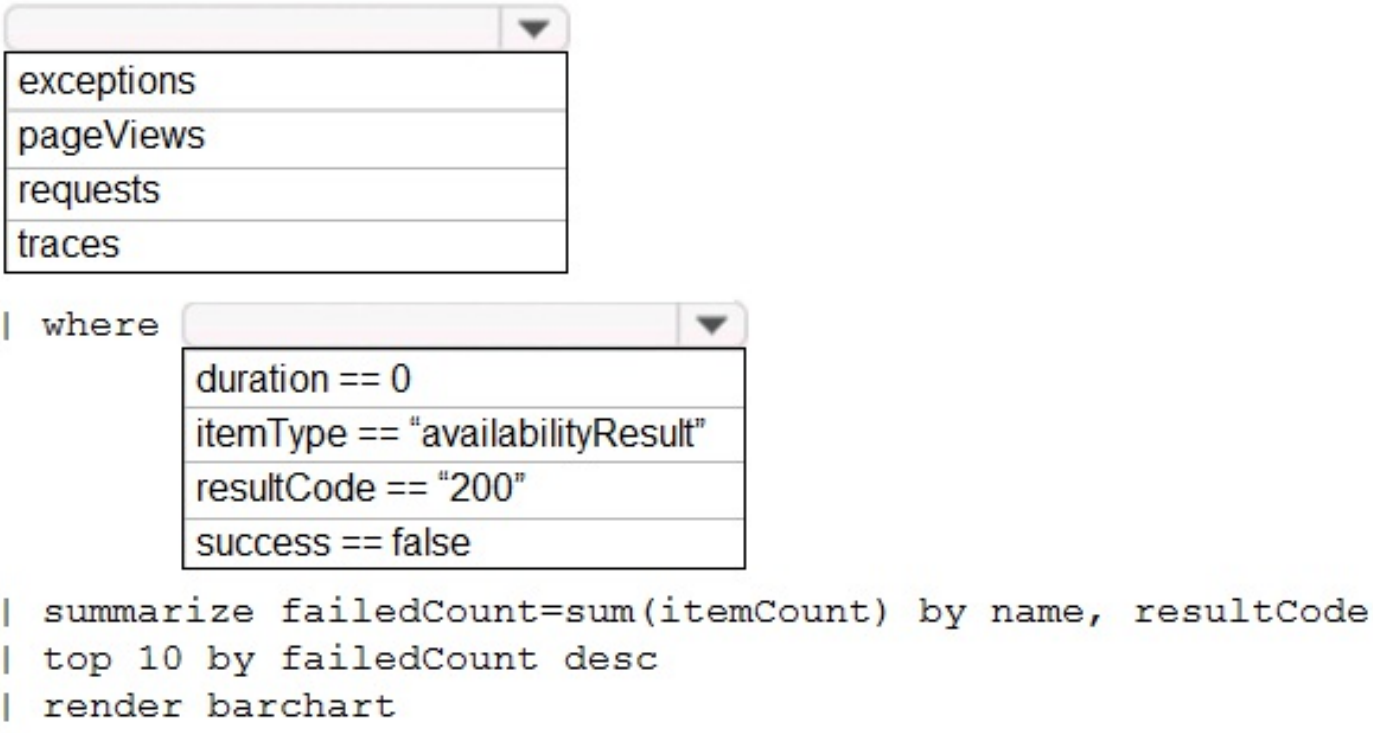
Event -
| where TimeGenerated > ago(12h)
| order by TimeGenerated desc
| summarize makelist(EventID) by Computer

Reference:
https://docs.microsoft.com/en-us/azure/azure-monitor/log-query/advanced-aggregations

Question: 50 CertyIQ

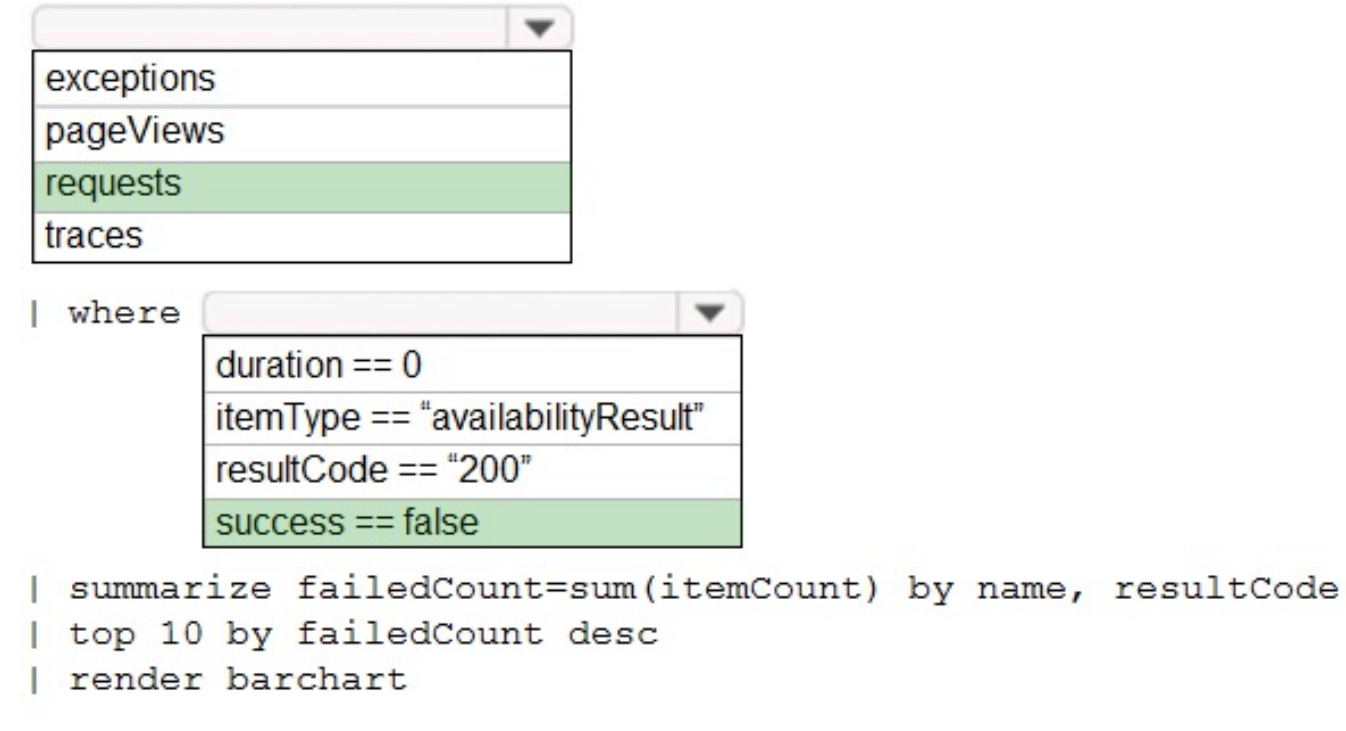
HOTSPOT -
You have an Azure web app named Webapp1.
You need to use an Azure Monitor query to create a report that details the top 10 pages of Webapp1 that failed.
How should you complete the query? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:
Box 1: requests -
Failed requests (requests/failed):
The count of tracked server requests that were marked as failed.
Kusto code:
requests
| where success == 'False'

Box 2: success == false -

Reference:
https://docs.microsoft.com/en-us/azure/azure-monitor/platform/app-insights-metrics

Question: 51 CertyIQ

You are monitoring the health and performance of an Azure web app by using Azure Application Insights.
You need to ensure that an alert is sent when the web app has a sudden rise in performance issues and failures.
What should you use?

A. custom events
B. Application Insights Profiler
C. usage analysis
D. Smart Detection
E. Continuous export

Answer: D

Explanation:

The correct answer is **D. Smart Detection**.

Smart Detection in Azure Application Insights automatically identifies potential performance problems and failures in your web application without requiring you to manually configure complex rules or thresholds. It uses machine learning algorithms to learn the typical behavior of your application and then detects anomalies that deviate significantly from that baseline. When Smart Detection identifies a significant performance degradation or an unexpected surge in failures, it automatically sends an alert. This proactive approach helps you quickly identify and address issues before they impact users.

Custom Events (A) are useful for tracking specific user actions or application events, but they don't inherently detect performance issues automatically. Application Insights Profiler (B) allows you to delve into the code-level execution details of requests to pinpoint performance bottlenecks, but it doesn't automatically alert on performance anomalies. Usage Analysis (C) is helpful for understanding how users are interacting with your application, but it doesn't directly detect or alert on performance problems or failures. Continuous Export (E) allows you to export your Application Insights data to other storage locations, but it doesn't provide any built-in alerting capabilities.

Smart Detection is designed specifically to monitor for and alert on these types of anomalies in your web app, making it the ideal choice for the scenario presented. Its automated nature provides a hassle-free way to receive alerts about unexpected performance issues and failures.

Relevant resources:

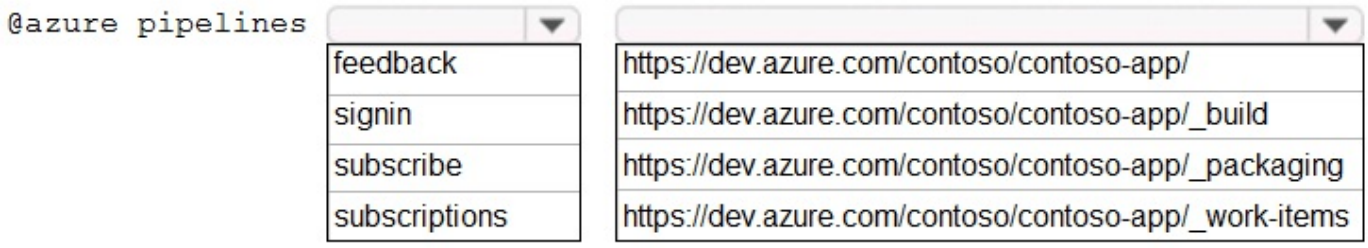
Smart Detection in Application Insights
Automatically detect performance anomalies

Question: 52 CertyIQ

HOTSPOT -
You have a project in Azure DevOps named Contoso App that contains pipelines in Azure Pipelines for GitHub repositories.

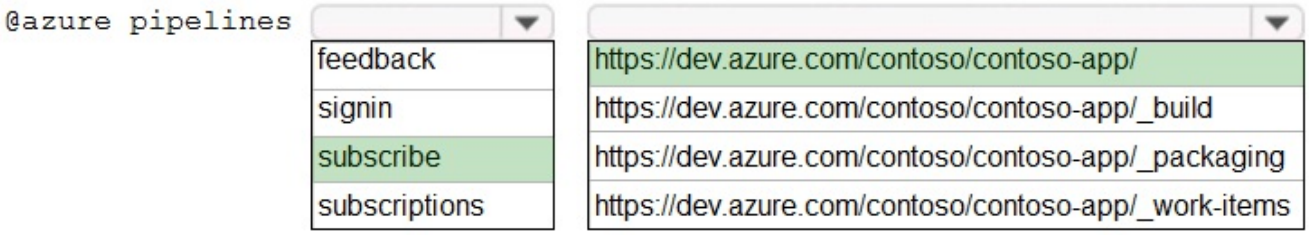
You need to ensure that developers receive Microsoft Teams notifications when there are failures in a pipeline of Contoso App.
What should you run in Teams? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:
Box 1: subscribe -
To start monitoring all pipelines in a project, use the following command inside a channel:
@azure pipelines subscribe [project url]
Box 2: https://dev.azure.com/contoso/contoso-app/
Subscribe to a pipeline or all pipelines in a project to receive notifications:
@azure pipelines subscribe [pipeline url/ project url]

Question: 53

CertyIQ

You have a private GitHub repository.
You need to display the commit status of the repository on Azure Boards.
What should you do first?

A. Configure multi-factor authentication (MFA) for your GitHub account.
B. Add the Azure Pipelines app to the GitHub repository.
C. Add the Azure Boards app to the repository.
D. Create a GitHub action in GitHub.

Answer: C

Explanation:

The correct answer is **C. Add the Azure Boards app to the repository.**

Here's the justification:

To integrate GitHub commits and their status with Azure Boards, you need a mechanism that allows the two systems to communicate and share information. This is where the Azure Boards app comes in. The Azure Boards app, available in the GitHub Marketplace, establishes a connection between your GitHub repository and your Azure DevOps organization. Specifically, it enables Azure Boards to receive commit status updates from GitHub.

The key point is that you are trying to display the **commit status** on Azure Boards. To do this, Azure Boards needs to receive notifications about commits in your GitHub repository. The Azure Boards app is explicitly designed for this purpose. By installing the app, you grant Azure Boards the necessary permissions to access your GitHub repository's commit information. Once installed, you can configure the connection between the repository and your Azure DevOps work items, enabling features like linking work items to commits using AB#workitemID in commit messages and viewing commit statuses directly within Azure Boards.

Options A, B, and D are incorrect for the following reasons:

A. Configure multi-factor authentication (MFA) for your GitHub account: While enabling MFA is a good security practice, it doesn't directly enable commit status reporting to Azure Boards. MFA enhances account security but doesn't establish the necessary communication channel between GitHub and Azure Boards.
B. Add the Azure Pipelines app to the GitHub repository: The Azure Pipelines app is primarily for continuous integration and continuous delivery (CI/CD) workflows. While it can trigger pipelines based on commits, its main focus is on building, testing, and deploying code, not on reporting commit status to Azure Boards. Although there's an indirect relationship - a pipeline run linked to a commit will update status - the simplest method is using the Azure Boards app.
D. Create a GitHub action in GitHub: GitHub Actions can automate various tasks, including CI/CD. While a GitHub Action could potentially be created to send commit status information to Azure DevOps via the Azure DevOps REST API, this is a far more complex and indirect approach than using the dedicated Azure Boards app. It requires custom scripting and management, increasing the implementation overhead.

Therefore, the simplest and most direct way to display the commit status of your GitHub repository on Azure Boards is to add the Azure Boards app to the repository.

Here are some authoritative links for further research:

Integrate Azure Boards with GitHub: https://learn.microsoft.com/en-us/azure/devops/boards/github/?view=azure-devops
Azure Boards app in GitHub Marketplace: https://github.com/marketplace/azure-boards

Question: 54 CertyIQ

You are integrating Azure Pipelines and Microsoft Teams.
You install the Azure Pipelines app in Microsoft Teams.
You have an Azure DevOps organization named Contoso that contains a project name Project1.
You subscribe to Project1 in Microsoft Teams.
You need to ensure that you only receive events about failed builds in Microsoft Teams.
What should you do first?

A. From Microsoft Teams, run @azure pipelines subscribe https://dev.azure.com/Contoso/Project1.
B. From Azure Pipelines, add a Publish Build Artifacts task to Project1.
C. From Microsoft Teams, run @azure pipelines subscriptions.
D. From Azure Pipelines, enable continuous integration for Project1.

Answer: C

Explanation:

The correct answer is **C. From Microsoft Teams, run @azure pipelines subscriptions.**

Here's why:

The objective is to filter the build events received in Microsoft Teams from Azure Pipelines to only include failed builds. To achieve this, you need to configure the subscription details within Microsoft Teams itself, not within Azure Pipelines initially. Running @azure pipelines subscriptions in Microsoft Teams allows you to view the current subscriptions and, more importantly, modify them to filter the events you receive. This command surfaces the current subscription configuration for the linked project.

Once you run @azure pipelines subscriptions, the Azure Pipelines app in Teams will display the current subscription settings for Project1. You can then use the "Update subscription" button in the Teams interface (initiated through @azure pipelines subscriptions) to modify the subscription. This update allows you to specify the events you want to receive (in this case, only failed builds). This ensures that only notifications matching the chosen criteria are pushed to the Teams channel. Without understanding the current subscription, it's impossible to effectively modify it for the desired filtering.

Option A is incorrect because directly subscribing doesn't give you the option to filter events. It only initiates a basic subscription. Option B is irrelevant because publishing build artifacts is unrelated to event filtering for Microsoft Teams notifications. Option D is also incorrect because enabling continuous integration is a build pipeline configuration, not related to Teams notifications.

Therefore, the essential first step is to examine the current subscriptions using @azure pipelines subscriptions within Microsoft Teams so you can modify the events you receive.

Here are authoritative links for further research:

Azure Pipelines app for Microsoft Teams: https://learn.microsoft.com/en-us/azure/devops/pipelines/integrations/microsoft-teams?view=azure-devops
Manage subscriptions in Microsoft Teams: While the documentation may not have a single page dedicated to managing subscriptions exclusively through @azure pipelines subscriptions, the general documentation on the Azure Pipelines app highlights its function for configuring and managing subscriptions (see link above). Examining existing subscriptions is a crucial step before modifying them, aligning with practical application of managing the Azure DevOps - Teams integration.

Question: 55 CertyIQ

You have an Azure DevOps organization named Contoso.
You need to receive Microsoft Teams notifications when work items are updated.
What should you do?

A. From Azure DevOps, configure a service hook subscription
B. From Microsoft Teams, configure a connector
C. From the Microsoft Teams admin center, configure external access
D. From Microsoft Teams, add a channel
E. From Azure DevOps, install an extension

Answer: B

Explanation:

The correct answer is B: From Microsoft Teams, configure a connector.

Here's a detailed justification:

Microsoft Teams connectors are the native mechanism for integrating external services, like Azure DevOps, into Teams channels. They allow Teams channels to receive notifications about events occurring in these external services. In this scenario, the requirement is to receive notifications in Teams when Azure DevOps work items are updated.

Configuring a connector within Microsoft Teams directly facilitates this. The user would configure the Azure DevOps connector, specify the Azure DevOps organization and project, and then configure the events that trigger notifications (e.g., work item updates). These notifications will then appear in the Teams channel.

Option A, configuring a service hook subscription from Azure DevOps, while related to integration, pushes notifications to an external service. However, for Teams, using the Teams connector is the more direct and supported approach. Service hooks are more generic and often require a custom endpoint to receive the data, which isn't ideal for Teams.

Option C, configuring external access in the Microsoft Teams admin center, is related to allowing users from outside the organization to access Teams, which is irrelevant to the notification requirement.

Option D, adding a channel in Microsoft Teams, simply creates a new channel; it does nothing to integrate with Azure DevOps or configure notifications.

Option E, installing an extension in Azure DevOps, while potentially helpful for other integrations, is not the primary or correct method for sending notifications to Teams. It introduces unnecessary complexity. The Teams connector provides a simpler, built-in solution.

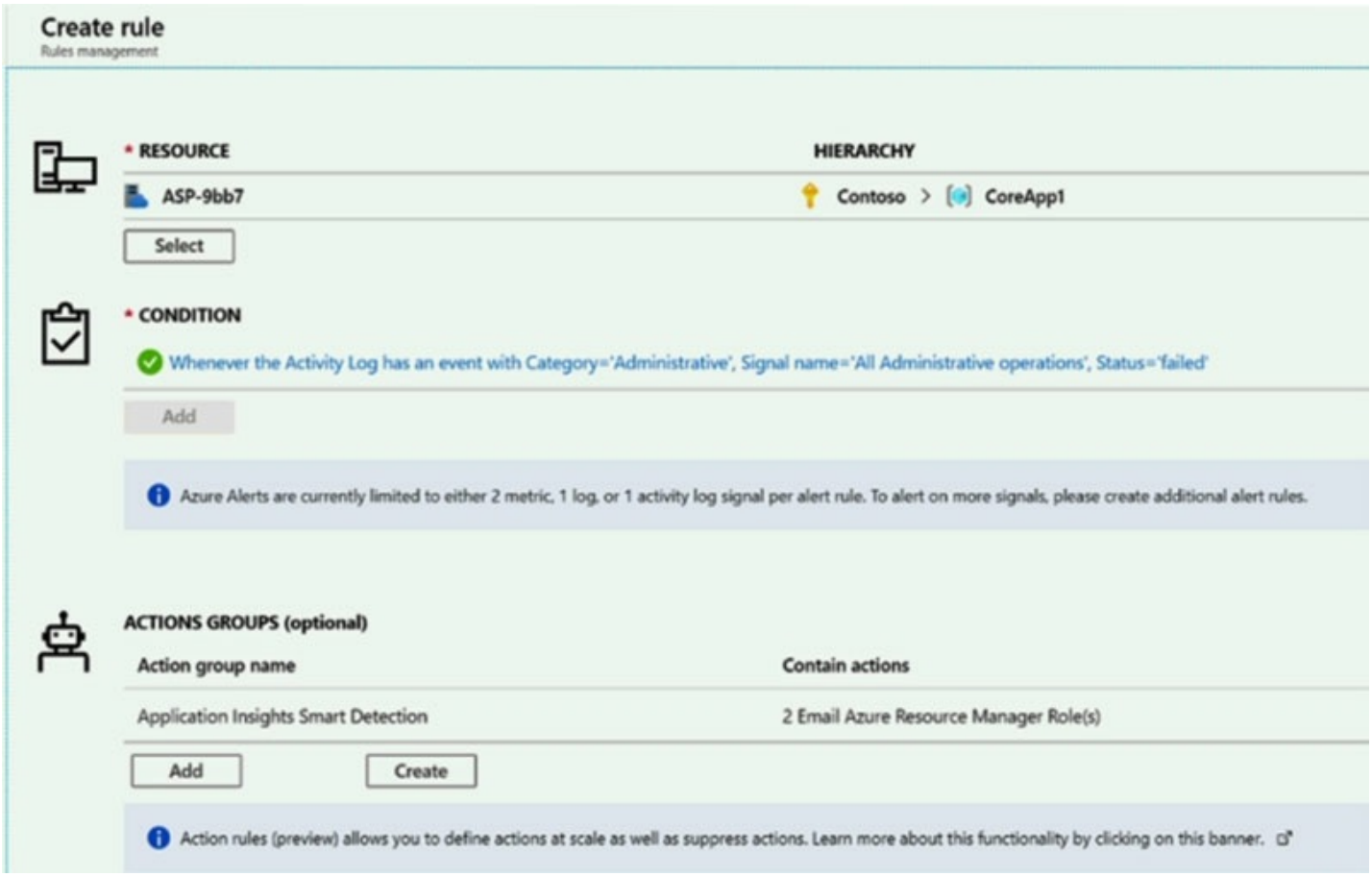
Therefore, the most direct and correct approach is to configure the Azure DevOps connector within Microsoft Teams to receive notifications about work item updates.

Authoritative links for further research:

Use Microsoft Teams & Azure DevOps together: https://learn.microsoft.com/en-us/azure/devops/service-hooks/services/teams?view=azure-devops
Connectors for Microsoft Teams: https://learn.microsoft.com/en-us/microsoftteams/platform/concepts/connectors/connectors

Question: 56 CertyIQ

You create an alert rule in Azure Monitor as shown in the following exhibit.



Which action will trigger an alert?

A. a failed attempt to delete the ASP-9bb7 resource
B. a change to a role assignment for the ASP-9bb7 resource
C. a successful attempt to delete the ASP-9bb7 resource
D. a failed attempt to scale up the ASP-9bb7 resource

Answer: A

Explanation:

But Administrative it's Contains the record of all create, update, delete, and action operations performed through Resource Manager.So answer is A

Question: 57 CertyIQ

You have a web app hosted on Azure App Service. The web app stores data in an Azure SQL database.
You need to generate an alert when there are 10,000 simultaneous connections to the database. The solution must minimize development effort.
Which option should you select in the Diagnostics settings of the database?

A. Send to Log Analytics
B. Stream to an event hub
C. Archive to a storage account

Answer: A

Explanation:

The correct answer is **A. Send to Log Analytics**. Here's a detailed justification:

Log Analytics is the most suitable option for creating alerts based on connection count to an Azure SQL database with minimal development effort. It's a powerful monitoring and analytics service within Azure Monitor. When database diagnostics are sent to Log Analytics, database performance metrics including connection counts become readily available.

Here's why other options are less ideal:

Stream to an event hub: Event Hubs are primarily used for real-time data ingestion and streaming for processing by other services. While you could process connection data from an Event Hub to trigger an alert, it would require significantly more development effort (setting up a stream processing service, configuring the alert logic).
Archive to a storage account: Archiving to a storage account is excellent for long-term data retention and compliance purposes. However, the data stored in the storage account isn't directly queryable or usable for real-time alerting without setting up separate data processing and querying pipelines. It would require a more complex solution.

Log Analytics allows you to create custom alert rules directly using Kusto Query Language (KQL). You can define a query that monitors the sql_connections_s metric (or a similar metric indicating connection counts) and triggers an alert when the count exceeds 10,000. This is the easiest path because it utilizes existing monitoring infrastructure within Azure and minimizes the code you need to write. Azure Monitor integrates directly with Log Analytics, enabling the easy creation and management of alerts. The query to create the alert for 10,000 simultaneous connections can be as simple as summarizing the connection count and firing an alert based on a specified threshold.

Therefore, using Log Analytics is the most efficient and least development-intensive way to generate an alert when the number of database connections exceeds the specified threshold.

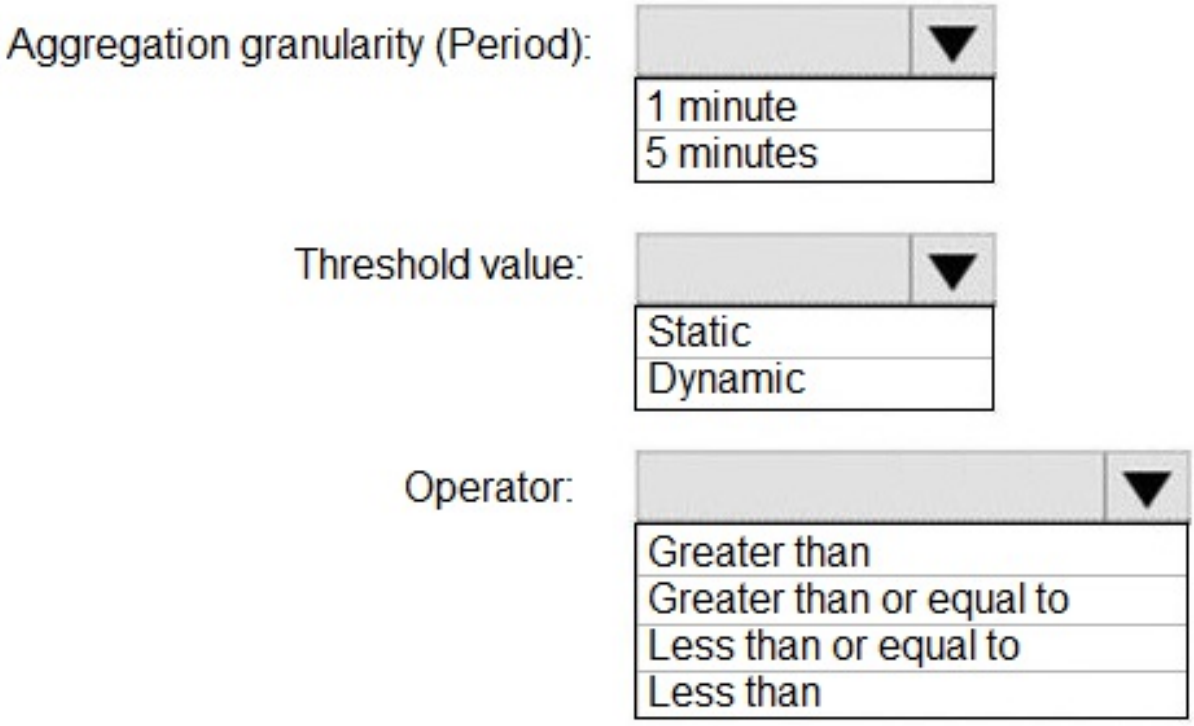
Supporting Links:

Azure Monitor Overview: https://learn.microsoft.com/en-us/azure/azure-monitor/overview
Log Analytics Overview: https://learn.microsoft.com/en-us/azure/azure-monitor/logs/log-analytics-overview
Create, view, and manage Azure Monitor metric alerts using Azure portal: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-metric
Kusto Query Language (KQL) Overview: https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/

Question: 58 CertyIQ

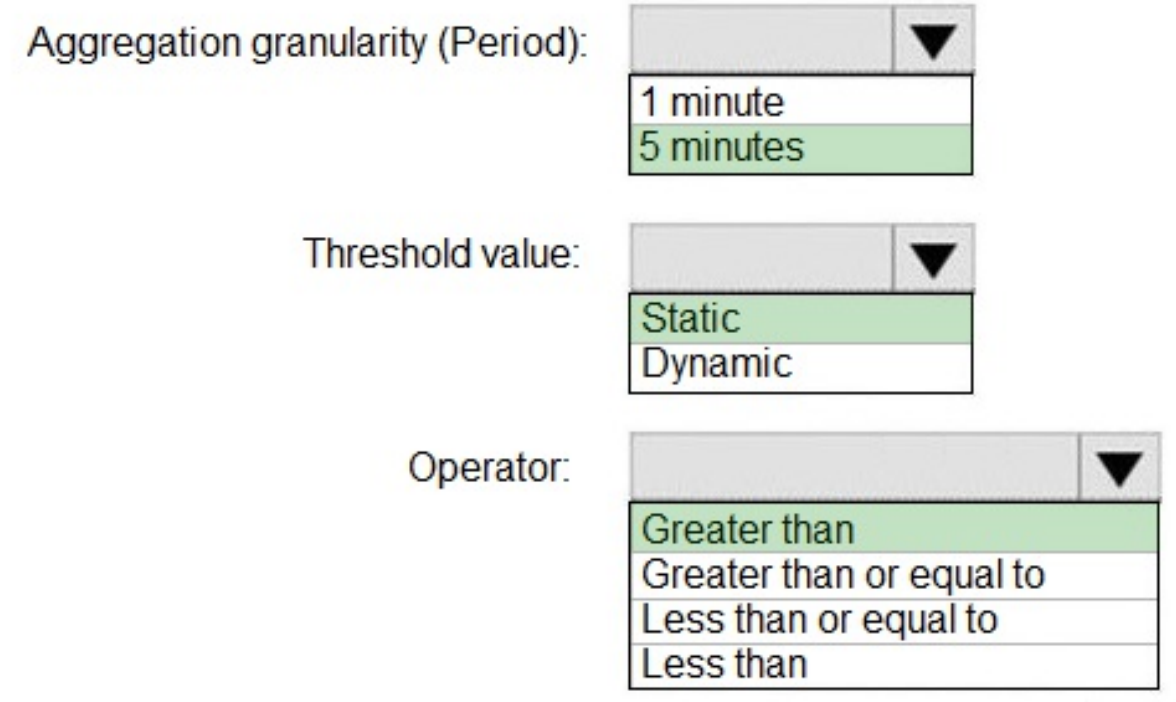
HOTSPOT -
You use Azure DevOps to manage the build and deployment of an app named App1.
You have a release pipeline that deploys a virtual machine named VM1.
You plan to monitor the release pipeline by using Azure Monitor.
You need to create an alert to monitor the performance of VM1. The alert must be triggered when the average CPU usage exceeds 70 percent for five minutes.
The alert must calculate the average once every minute.
How should you configure the alert rule? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:

Box 1: 5 minutes -

The alert must calculate the average once every minute.

Note: We [Microsoft] recommend choosing an Aggregation granularity (Period) that is larger than the Frequency of evaluation, to reduce the likelihood of missing the first evaluation of added time series

Box 2: Static -

Box 3: Greater than -
Example, say you have an App Service plan for your website. You want to monitor CPU usage on multiple instances running your web site/app. You can do that using a metric alert rule as follows:
- Target resource: myAppServicePlan
- Metric: Percentage CPU
- Condition Type: Static
- Dimensions
- Instance = InstanceName1, InstanceName2
- Time Aggregation: Average
- Period: Over the last 5 mins
- Frequency: 1 min
- Operator: GreaterThan
- Threshold: 70
- Like before, this rule monitors if the average CPU usage for the last 5 minutes exceeds 70%.
- Aggregation granularity

Reference:
https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-metric-overview

Question: 59 CertyIQ

You have an Azure virtual machine that is monitored by using Azure Monitor.
The virtual machine has the Azure Log Analytics agent installed.
You plan to deploy the Service Map solution from the Azure Marketplace.
What should you deploy to the virtual machine to support the Service Map solution?

A. the Dependency agent
B. the Telegraf agent
C. the Windows Azure diagnostics extension (WAD)
D. the Azure monitor agent

Answer: A

Explanation:

The correct answer is A, the Dependency agent. Service Map in Azure Monitor requires the Dependency agent to be installed on the virtual machine being monitored. The Dependency agent discovers and maps dependencies between processes running on the machine, including inbound and outbound connections to other systems. This information is then visualized in the Service Map in Azure Monitor.

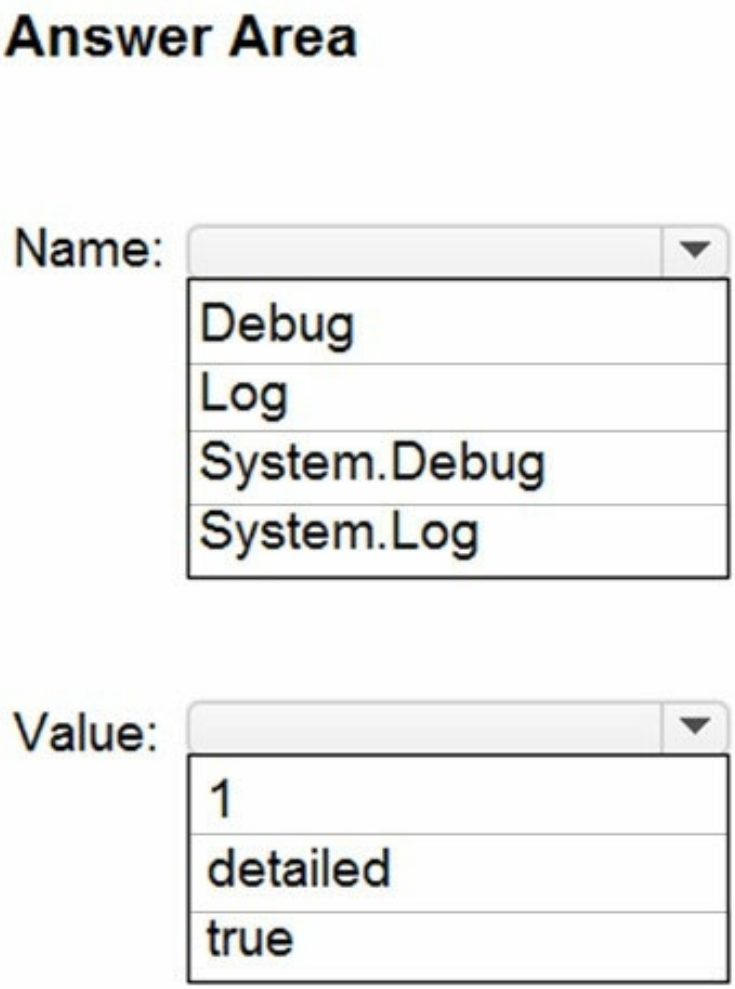
The Telegraf agent (option B) is a plugin-driven server agent for collecting, processing, aggregating, and writing metrics, but it is not specifically used by Service Map for dependency mapping. The Windows Azure Diagnostics extension (WAD) (option C) is primarily used for collecting diagnostics data like performance counters and event logs, but it doesn't perform the dependency discovery function needed by Service Map. While the Azure Monitor agent (option D) is the future of data collection in Azure Monitor, it does not currently provide the necessary dependency mapping capabilities without the Dependency Agent's support. Service Map explicitly relies on the Dependency agent to understand the network connections and process dependencies for accurate mapping, making it the essential component. Without it, the Service Map will not be able to display the dependencies effectively.

Reference:

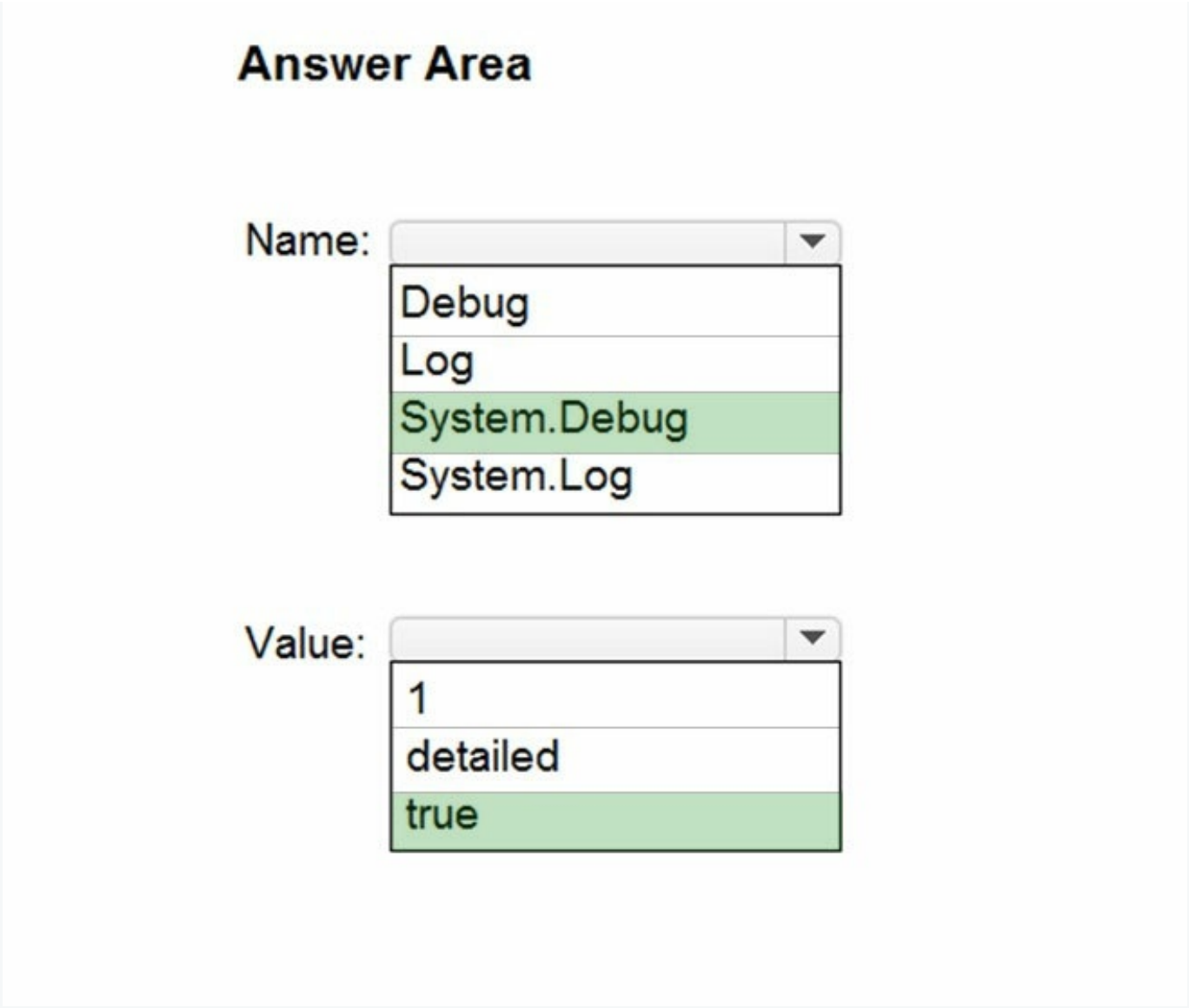
Service Map solution in Azure Monitor - This page explicitly mentions the Dependency agent's role.
Install the Dependency Agent - Detailed instructions on installing the dependency agent for Service Map.

Question: 60 CertyIQ

HOTSPOT -
You have a project in Azure DevOps that contains a Continuous Integration/Continuous Deployment (CI/CD) pipeline.
You need to enable detailed logging by defining a pipeline variable.
How should you configure the variable? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:



Answer:



Explanation:
Box 1: system.debug -
To configure verbose logs for all runs, you can add a variable named system.debug and set its value to true.
Note: Verbose logging is the practice of recording to a persistent medium as much information as you possibly can about events that occur while the software runs.

Box 2: true -

Reference:
https://docs.microsoft.com/en-us/azure/devops/pipelines/troubleshooting/review-logs

Question: 61 CertyIQ

You build an iOS app.
You receive crash reports from Crashlytics.
You need to capture the following data:
- Crash-free users
- Custom events
- Breadcrumbs

What should you do?

A. Configure the xcworkspace file in the project
B. Add the GoogleAnalytics pod to the app.
C. Configure the Crashlytics pod in the app.

D. Import the Firebase module to UIApplicationDelegate.

Answer: D

Explanation:

The correct answer is **D. Import the Firebase module to UIApplicationDelegate.**

Here's why:

Crashlytics and Firebase: Crashlytics is now part of Firebase. To leverage Crashlytics' features in an iOS app, including crash-free users, custom events, and breadcrumbs, you need to integrate the Firebase SDK.

UIApplicationDelegate: UIApplicationDelegate is the central point of contact for the operating system and your application. It's where you handle key app lifecycle events. Initializing Firebase in the UIApplicationDelegate ensures Firebase is initialized early in the app's lifecycle, allowing it to capture crashes and other events from the start.

Importing the Firebase Module: Importing the Firebase module in UIApplicationDelegate makes the Firebase SDK available to the app. This is usually done in the application(_:didFinishLaunchingWithOptions:) method of your app delegate.

Why other options are incorrect:

A. Configure the xcworkspace file in the project: While working with pods and CocoaPods, the xcworkspace file is crucial, configuring the file itself doesn't directly integrate Firebase or capture the desired data. Configuring simply allows you to work with the pod after it has been installed.
B. Add the GoogleAnalytics pod to the app: Google Analytics is a separate service from Crashlytics. While it can track user behavior and custom events, it's not the primary tool for crash reporting and capturing breadcrumbs.
C. Configure the Crashlytics pod in the app: You won't use the Crashlytics pod since you are integrating Firebase. Integrating Firebase now automatically integrates Crashlytics as well.

Steps to Integrate Firebase/Crashlytics:

1. **Install the Firebase SDK:** Use CocoaPods or Swift Package Manager to add the Firebase SDK to your project (specifically, Firebase/Crashlytics).
2. **Import Firebase in AppDelegate:** As mentioned, import the Firebase module in UIApplicationDelegate (usually in AppDelegate.swift or AppDelegate.m).
3. **Configure Firebase:** Configure the Firebase app in application(_:didFinishLaunchingWithOptions:) using FirebaseApp.configure().
4. **Add Run Script Build Phase:** In your Xcode project's Build Phases, add a new Run Script phase with the script provided by Firebase to upload debug symbols to Firebase for proper crash reporting.

Authoritative Links:

Firebase Crashlytics Documentation: https://firebase.google.com/docs/crashlytics
Get started with Firebase in your iOS app: https://firebase.google.com/docs/ios/setup

Question: 62 CertyIQ

You have multiple teams that work on multiple projects in Azure DevOps.
You need to plan and manage the consumers and producers for each project. The solution must provide an overview of all the projects.
What should you do?

A. Add a Predecessor or Successor link to the feature or user story for the items of each project.
B. Add a Parent or Child link to the feature or user story for the items of each project.
C. Install the Dependency Tracker extension and create dependencies for each project.
D. Create a custom query to show the consumers and producers and add a widget to a dashboard.

Answer: C

Explanation:

The correct answer is C: Install the Dependency Tracker extension and create dependencies for each project. This solution effectively addresses the requirement of planning and managing consumers and producers across multiple projects in Azure DevOps, providing a comprehensive overview.

Here's a detailed justification:

The core of the problem lies in visualizing and managing dependencies between projects. Option C directly addresses this need. The Dependency Tracker extension within Azure DevOps allows you to create and visualize dependencies between work items across different projects. This provides a clear understanding of which projects consume resources or outputs from other projects (producers).

Here's why the other options are less suitable:

A. Add a Predecessor or Successor link...: While Predecessor/Successor links can establish relationships, they are primarily designed for tasks within a single project and are not ideal for cross-project dependency management at a higher level (feature/user story). They also do not provide an aggregated view across all projects.
B. Add a Parent or Child link...: Parent/Child links are hierarchical relationships within a single project's work item structure. They do not effectively represent dependencies between independent projects.
D. Create a custom query...: While a custom query could retrieve information about work items, it would require extensive manual configuration to define consumers and producers, and it wouldn't provide the visual dependency mapping offered by the Dependency Tracker. A query also doesn't offer a built-in way to manage and update dependencies as projects evolve. Creating a widget would present the query results, but this still relies on the complexity of the query configuration and lacks the targeted features of the Dependency Tracker.

The Dependency Tracker extension provides a visual representation of dependencies, including a dependency graph and matrix view, making it easy to identify bottlenecks and potential risks. This visual overview is crucial for effective planning and management across multiple projects. It streamlines communication between teams and reduces the risk of integration issues.

In summary, installing the Dependency Tracker and creating dependencies allows for a structured, visual, and manageable way to oversee the consumer-producer relationships across multiple projects, thus fulfilling the solution requirements in an efficient and easily maintainable manner.

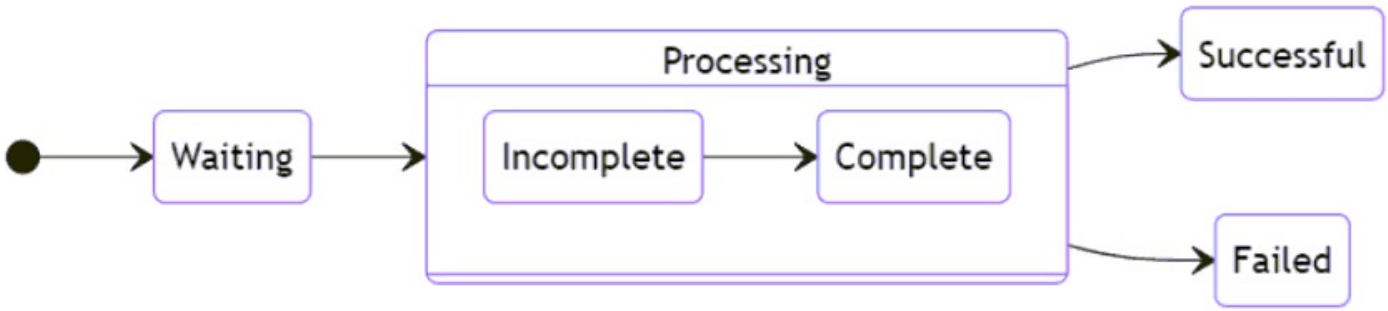
Authoritative Links:

Dependency Tracker Extension: https://marketplace.visualstudio.com/items?itemName=ms.vss-dependencytracker
Azure DevOps Work Item Tracking: https://learn.microsoft.com/en-us/azure/devops/boards/work-items/about-work-items?view=azure-devops

Question: 63 CertyIQ

DRAG DROP

-
You have a GitHub repository that contains the source code for an app named App1.
You need to create process documentation for App1. The solution must include a diagram that displays the relationships between the phases of App1 as shown in the following exhibit.



How should you complete the markdown code? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Values

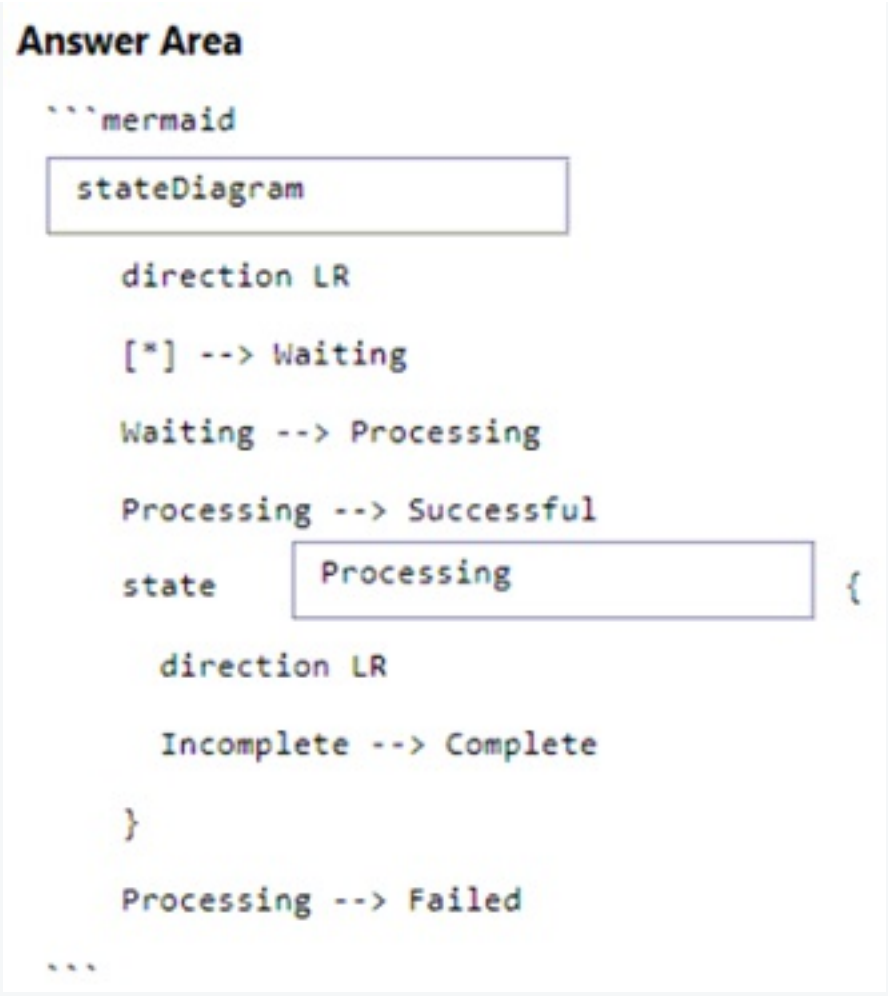
- Failed
- flowChart
- Incomplete
- Processing
- sequenceDiagram
- stateDiagram
- Waiting

Answer Area

```mermaid
[ ]
    direction LR
    [*] --> Waiting
    Waiting --> Processing
    Processing --> Successful
    state [ ] {
      direction LR
      Incomplete --> Complete
    }
    Processing --> Failed
```

Answer:

Answer Area



Explanation:

In Mermaid state diagrams, you cannot define transitions to and from a state (e.g., Waiting --> Processing) outside of a state block while simultaneously defining that same state as a composite state (a state containing nested substates) inside a block later in the code.

To fix this, you should define the composite Processing state entirely within one block.

Question: 64 CertyIQ

HOTSPOT
-
You have an Azure web app named webapp1 that uses the .NET Core runtime stack. You have an Azure Application Insights resource named AppInsights1 that collects telemetry data generated by webapp1.
You plan to deploy webapp1 by using an Azure DevOps pipeline.
You need to modify the sampling rate of the telemetry data processed by AppInsights1 without having to redeploy webapp1 after each modification.
What should you do? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

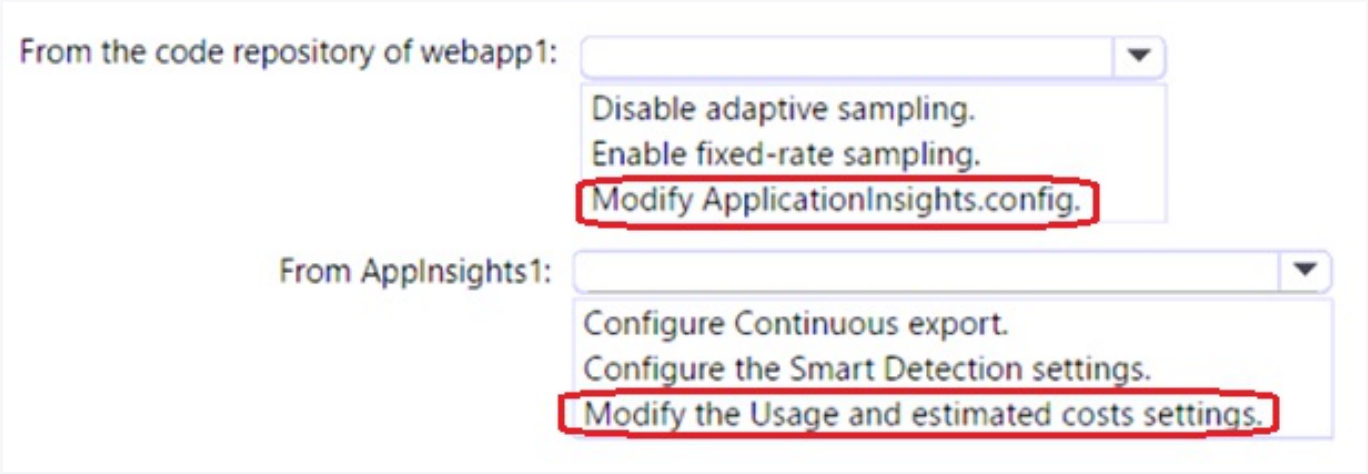
From the code repository of webapp1:

- Disable adaptive sampling.
- Enable fixed-rate sampling.
- Modify ApplicationInsights.config.

From AppInsights1:

- Configure Continuous export.
- Configure the Smart Detection settings.
- Modify the Usage and estimated costs settings.

Answer:



Explanation:

Modify ApplicationInsights.config (Code-level):

This is where you configure SDK-level sampling (such as Adaptive Sampling or Fixed-rate Sampling) for your application. By modifying this configuration file within your project's source code, you control how the Application Insights SDK processes and filters telemetry before it is sent to the Azure portal. This is the preferred method for managing data volume at the source, as it also reduces network bandwidth consumption.

Modify the Usage and estimated costs settings (Portal-level):

This setting in the Azure portal controls Ingestion Sampling. This is a "safety net" or last-resort mechanism that happens at the Application Insights service endpoint, after the data has already left your application. While this helps keep your data usage within a monthly quota or budget, it does not reduce the network traffic or costs associated with sending the original telemetry from your application to Azure.

Question: 65 CertyIQ

Your company has multiple microservices-based apps that use the following tracing libraries:
• OpenTelemetry
• OpenCensus
• OpenTracing
• Honeycomb
• Jaeger
The company purchases an Azure subscription and implements Application Insights in Azure Monitor.

You plan to centralize distributed tracing for the apps.
You need to identify which libraries can integrate directly with Application Insights.
Which two libraries should you identify? Each correct answer presents a complete solution.
NOTE: Each correct solution is worth one point.

A. Honeycomb
B. OpenTracing
C. Jaeger
D. OpenTelemtry
E. OpenCensus

Answer: DE

Explanation:

The correct answer is D (OpenTelemetry) and E (OpenCensus).

Here's a detailed justification:

Application Insights, a feature of Azure Monitor, provides Application Performance Management (APM) capabilities. To centralize distributed tracing in Application Insights, you need tracing libraries that are compatible with it. While Application Insights generally prefers OpenTelemetry as the future standard, OpenCensus still maintains direct integration while its users migrate towards OpenTelemetry.

OpenTelemetry: OpenTelemetry is now the recommended standard for tracing on Azure. Microsoft actively supports OpenTelemetry and it offers a native integration. This means you can send OpenTelemetry trace data directly to Application Insights for monitoring and analysis. Azure Monitor OpenTelemetry Exporter enables developers to send OpenTelemetry data to Azure Monitor.

OpenCensus: OpenCensus preceded OpenTelemetry and offers direct integration. OpenCensus and OpenTracing have now merged into OpenTelemetry, but OpenCensus support continues during the transition.

Jaeger: Jaeger is a popular open-source distributed tracing system, however, it doesn't directly integrate with Application Insights without an intermediary.

Honeycomb: Honeycomb is a commercial observability platform. While you can potentially export data from Honeycomb and import it into Application Insights, there is no direct integration available.

OpenTracing: OpenTracing has also been merged into OpenTelemetry. Using OpenTracing now means using OpenTelemetry as a successor library. While indirectly compatible via OpenTelemetry bridge, it does not provide as simple of a path as OpenCensus or OpenTelemetry.

Therefore, OpenTelemetry and OpenCensus can directly integrate with Application Insights, simplifying the centralization of distributed tracing data. You can configure your applications using these libraries to send trace data directly to Application Insights without requiring additional middleware or conversion steps.

Relevant links:

Azure Monitor OpenTelemetry Exporter
Migrate from OpenCensus to OpenTelemetry
What is Azure Monitor?

Question: 66 CertyIQ

You have an Azure web app named webapp1 that uses the .NET Core runtime stack. You have an Azure Application

Insights resource named AppInsights1. Webapp1 sends telemetry data to AppInsights1.
You need to ensure that webapp1 sends the telemetry data at a fixed sampling rate.
What should you do?

A. From the code repository of webapp1, modify the ApplicationInsights.config file.
B. From the code repository of webapp1, modify the Startup.cs file.
C. From AppInsights1, modify the Usage and estimated costs settings.
D. From AppInsights1, configure the Continuous export settings.

Answer: B

Explanation:

The correct answer is B: Modify the Startup.cs file in the webapp1's code repository.

Here's why: Fixed-rate sampling in Application Insights for .NET Core applications is configured within the application's initialization code. This allows you to precisely control the sampling behavior, setting a specific percentage of telemetry items to be retained. The Startup.cs file is the standard place to configure application services and middleware in a .NET Core application.

Option A, modifying the ApplicationInsights.config file, is relevant for older .NET Framework applications, not .NET Core. While a config file can be used, the recommended and more modern approach in .NET Core is programmatic configuration within Startup.cs.

Option C, modifying the Usage and Estimated Costs settings in Application Insights, primarily relates to data retention and overall cost management, not the sampling rate. These settings affect the amount of data stored, but not the sampling strategy before the data is sent.

Option D, configuring Continuous Export in Application Insights, is used to export telemetry data to other storage solutions for further analysis or archiving. It doesn't influence the sampling rate at which data is initially collected from the web application.

By modifying Startup.cs, you can add code to explicitly configure the Application Insights telemetry configuration to use fixed-rate sampling. This involves using the TelemetryConfiguration object and setting the SamplingPercentage property of the TelemetryProcessorChainBuilder. This gives you fine-grained control over the sampling process and ensures a consistent sampling rate.

Example snippet of code:

```
public void ConfigureServices(IServiceCollection services)     // ... other services configurations     services.AddApplicatior
```

This snippet registers the Application Insights service and uses the UseSampling method on the TelemetryProcessorChainBuilder to configure a fixed-rate sampling percentage. Changing the value of samplingPercentage sets the fixed sampling rate.

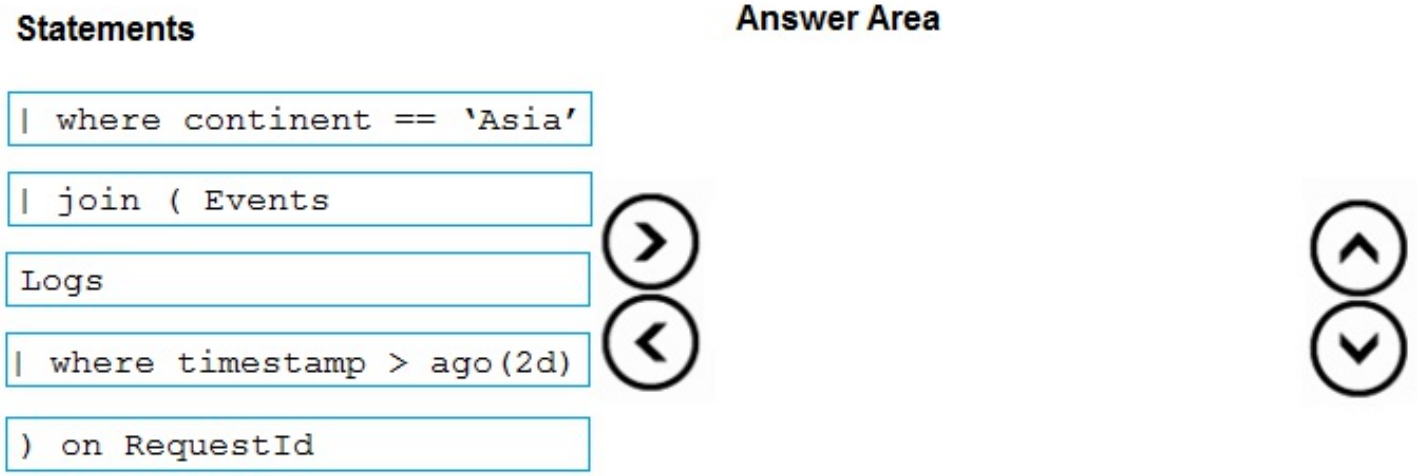
Supporting documentation:

Sampling in Application Insights: Explains different types of sampling and how to configure them.
Application Insights for ASP.NET Core applications: Details the process of instrumenting ASP.NET Core applications with Application Insights.

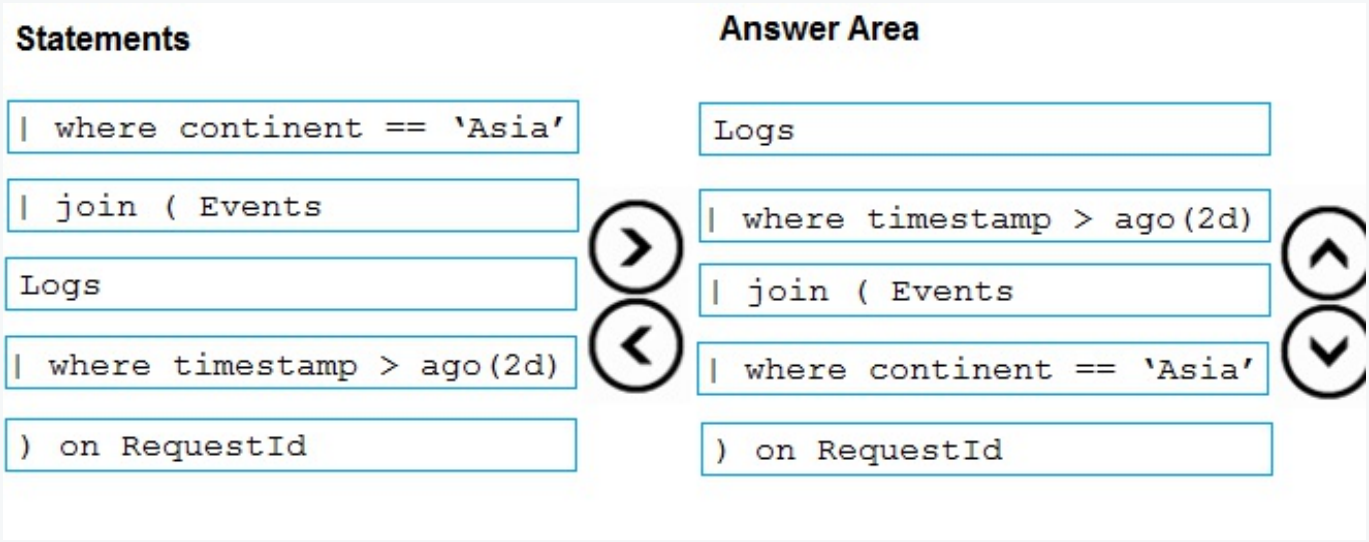
Question: 67 CertyIQ

DRAG DROP
-

You have an app named App1. You have a Log Analytics workspace named Workspace1 that contains two tables named Events and Logs. App1 manages events in multiple locations and writes logs to Workspace1.
You need to query Workspace1 for all log entries related to Asia that occurred during the last two days.
In which order should you arrange the query statements? To answer, move all statements from the list of statements to the answer area and arrange them in the correct order.



Answer:



Explanation:

Logs

| where timestamp > ago(2d)

| join (Events

| where continent == 'Asia'

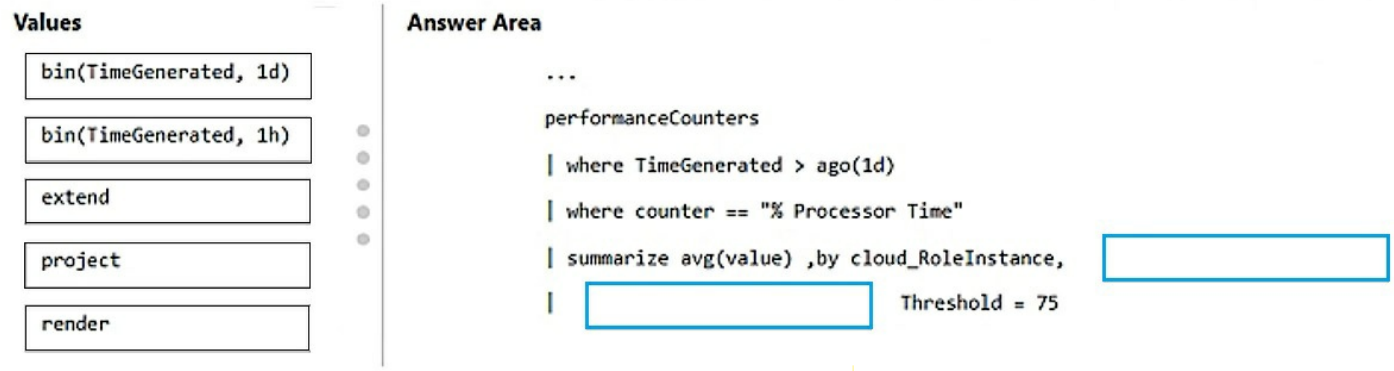
) on RequestId

https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/best-practices

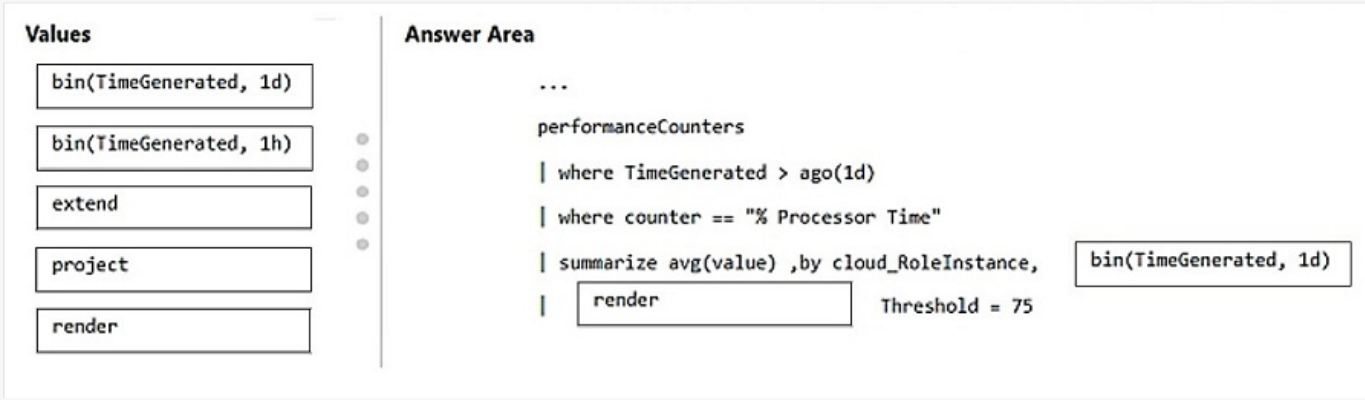
Question: 68 CertyIQ

DRAG DROP
-
You have a web app named App1 that uses Application Insights in Azure Monitor.
You need to compare the hourly CPU usage of App1 from the last 24 hours. The solution must include a graph that

has a threshold line at 75 percent.
How should you complete the query? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

peformance counters| where TimeGenerated > ago(1d)| where counter == "% Processor Time"| summarize avg(value), by cloud_RoleInstance, bin(TimeGenerated, 1d)| render Threshold = 75

Question: 69

CertyIQ

You use Azure Pipelines to build and deploy an app named App1.
You plan to monitor App1 by using Application Insights.
You create an Application Insights instance named AI1.
You need to configure App1 to use AI1.
Which file should you modify?

A.appsettings.json
B.launchSettings.json
C.startup.cs
D.project.json

Answer: C

Explanation:

The correct answer is **C. startup.cs**.

Here's a detailed justification:

When integrating Application Insights with an application, particularly in a .NET environment often used with Azure DevOps and Azure Pipelines, the startup.cs file plays a crucial role in configuring the application's services and middleware. This file is responsible for setting up the application's request pipeline and registering services like Application Insights with the dependency injection container.

Here's why the other options are less suitable:

appsettings.json: While appsettings.json is commonly used to store configuration data like connection strings and application settings, it's not typically the place where Application Insights is initialized or configured as a service within the application's architecture. It might contain the instrumentation key, but the actual plumbing to add Application Insights is done elsewhere.

launchSettings.json: This file is primarily used to configure the debugging and launching profiles for the application during development. It doesn't control how Application Insights is integrated into the application at runtime.

project.json: The project.json file was used in older versions of .NET Core. Modern .NET projects use .csproj files. While the project file lists dependencies, the configuration happens in startup.cs.

In startup.cs, the ConfigureServices method is where you would typically add Application Insights to the application's service collection. This involves adding the services.AddApplicationInsightsTelemetry() method, which enables Application Insights to collect telemetry data from your application. You might also configure custom telemetry processors or initializers within this method.

Furthermore, in the Configure method of startup.cs, you can add middleware to the HTTP request pipeline to integrate Application Insights into the request processing flow. This allows Application Insights to automatically track requests, exceptions, and other relevant telemetry data.

Therefore, startup.cs is the logical location to configure and initialize Application Insights within an ASP.NET Core application that's being built and deployed using Azure DevOps. This allows the application to send telemetry data to the configured Application Insights instance (AI1).

Authoritative links for further research:

Application Insights for ASP.NET Core applications: https://learn.microsoft.com/en-us/azure/azure-monitor/app/asp-net-core
Startup class in ASP.NET Core: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/startup?view=aspnetcore-7.0

Question: 70 CertyIQ

HOTSPOT
-
You have an Azure virtual machine named web1.
You need to query the amount of free memory that was available on web1 during the past seven days. The solution must meet the following requirements:
•Display the data as a time chart.
•Calculate the average value per hour.
How should you complete the KQL query? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

```
...
Perf
| where [ TimeGenerated > ago(7d) | TimeGenerated < ago(7d) | TimeGenerated == ago(7d) ]
| where Computer == "web1"
| where CounterName == "Available MBytes"
| summarize avg(CounterValue) by [ bin(TimeGenerated, 1d) | bin(TimeGenerated, 1h) | dcount(CounterName) ]
| render timechart
...
```

Answer:

Answer Area

```
...
Perf
| where [ TimeGenerated > ago(7d) | TimeGenerated < ago(7d) | TimeGenerated == ago(7d) ]
| where Computer == "web1"
| where CounterName == "Available MBytes"
| summarize avg(CounterValue) by [ bin(TimeGenerated, 1d) | bin(TimeGenerated, 1h) | dcount(CounterName) ]
| render timechart
...
```

Explanation:

1. TimeGenerated > ago (7d)2. bin(TimeGenerated, 1h)

https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/agofunctionSubtracts the given timespan from the current UTC time.https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/binfunctionRounds values down to an integer multiple of a given bin size.

Question: 71 CertyIQ

DRAG DROP

-

You have a web app named App1 that uses Application Insights in Azure Monitor to store log data. App1 has users in multiple locations.

You need to query App1 requests from London and Paris that return a 404 error. The solution must meet the following requirements:

•Return the timestamp url, resultCode, and duration fields
•Only show requests made during the last hour.
How should you complete the query? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all.

Values

- extend
- project
- select
- timestamp >= ago(1hr)
- timestamp -gt ago(1hr)

Answer Area

```
...
requests
| where [          ]
| where resultCode == "404" and (client_City
== "London" or client_City == "Paris")
| [          ] timestamp, url, resultCode, duration
...
```

Answer:

Answer Area

```
...
requests
| where timestamp >= ago(1hr)
| where resultCode == "404" and (client_City
== "London" or client_City == "Paris")
| project timestamp, url, resultCode, duration
...
```

Explanation:

| where timestamp >= ago(1h): This filters the data to only include requests from the last hour.

project timestamp, url, resultCode, duration: This selects only the specific columns you want to see in your output results, removing all other extraneous data from the view.

https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/projectoperator

Question: 72 CertyIQ

DRAG DROP
-
You have a project in Azure DevOps.
You need to configure a dashboard. The solution must include the following metrics:
•Bottlenecks in the software development process
•A burndown chart for the work in a single iteration
•How long it takes to close a work item after the item was started
Which type of widget should you use for each metric? To answer, drag the appropriate widget types to the correct metrics. Each widget type may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Widgets	Answer Area	
Burndown chart	Bottlenecks in the software development process:	
Cumulative flow diagram (CFD)	How long it takes to close a work item after the item was started:	
Cycle time	A burndown chart for the work in a single iteration:	
Lead time		
Sprint burndown		
Velocity		

Answer:

Widgets	Answer Area	
Burndown chart	Bottlenecks in the software development process:	Cumulative flow diagram (CFD)
Cumulative flow diagram (CFD)	How long it takes to close a work item after the item was started:	Cycle time
Cycle time	A burndown chart for the work in a single iteration:	Sprint burndown
Lead time		
Sprint burndown		
Velocity		

Explanation:

Cumulative Flow Diagram (CFD): This is essential for identifying process bottlenecks. By showing the distribution of work items across various states (e.g., To Do, In Progress, Testing, Done) over time, it highlights where work is accumulating, which indicates a bottleneck.

Cycle Time: This specifically measures the time taken to complete a work item once work has actually commenced on it. It is a key metric for team efficiency.

Note: Lead Time is a similar but broader metric that measures the total time from the moment a customer request is made until the item is delivered.

Sprint Burndown: This provides a focused view of the remaining work in a specific, time-boxed iteration (sprint), helping the team track progress against their commitment.

Question: 73 CertyIQ

You have an Azure subscription that contains a Log Analytics workspace named WS1 and a virtual machine named VM1.
You need to install the Microsoft Enterprise Cloud Monitoring extension on VM1.
Which two values are required to configure the extension? Each correct answer presents part of the solution.
NOTE: Each correct answer is worth one point.

A.the secret key of WS1
B.the ID of the subscription
C.the system-assigned managed identity of VM1
D.the ID of WS1
E.the resource ID of VM1

Answer: AD

Explanation:

The correct answer is **AD**.

Here's why:

To install the Microsoft Enterprise Cloud Monitoring extension (now known as the Azure Monitor Agent) on an Azure virtual machine and connect it to a Log Analytics workspace, you need specific identification details for both the VM and the workspace. This is because the extension needs to know where to send the monitoring data it collects.

A. The secret key of WS1: The secret key (also called the primary or secondary key or Shared Key) for the Log Analytics workspace is required for authentication. The extension uses this key to securely connect to the workspace and upload the collected data. Without the key, the extension cannot establish a trusted connection, and no data can be sent.

D. The ID of WS1: The workspace ID uniquely identifies the Log Analytics workspace within Azure. The extension needs this ID to locate the correct workspace to which it should send the collected logs and metrics. It is a GUID that distinguishes WS1 from all other workspaces in your Azure subscription.

Why other options are incorrect:

B. The ID of the subscription: While the subscription ID is important for overall Azure resource management, it's not directly needed for the monitoring extension's configuration. The workspace ID within that subscription is the specific identifier required.

C. The system-assigned managed identity of VM1: While using managed identities is a best practice for authenticating to Azure services and can be used for connecting to Log Analytics, it's not the required method when initially configuring the Microsoft Enterprise Cloud Monitoring extension through ARM templates or similar methods. The key is a more direct approach for the initial setup.

E. The resource ID of VM1: The VM resource ID is useful for Azure Resource Manager operations, but the monitoring extension primarily needs the VM to be running and have network connectivity to the Log Analytics workspace. The workspace ID and its corresponding shared key are crucial for data ingestion.

In summary: To configure the Microsoft Enterprise Cloud Monitoring extension, you need the Log Analytics workspace's ID (to identify the workspace) and the secret key (shared key) (to authenticate the connection between the VM and the workspace). This allows the extension to securely send monitoring data to the correct destination.

Authoritative Links:

Overview of the Azure Monitor agent: https://learn.microsoft.com/en-us/azure/azure-monitor/agents/azure-monitor-agent-overview
Use managed identities to access Azure Monitor: https://learn.microsoft.com/en-us/azure/azure-monitor/agents/data-collection-rule-azure-vm-extension
Log Analytics workspace shared keys: https://learn.microsoft.com/en-us/azure/azure-monitor/logs/manage-access

Question: 74 CertyIQ

You have an app named App1 that uses Application Insights to monitor application performance.

You need to analyze how often a page in App1 is accessed.
Which pane in Application Insights should you use?

A.Events
B.Sessions
C.Impact
D.Users

Answer: A

Explanation:

The correct answer is **A. Events**. Here's why:

Application Insights is a powerful Application Performance Management (APM) service within Azure used to monitor the health, performance, and usage of applications. Analyzing page access frequency falls directly into the category of understanding user behavior and usage patterns.

Events: The Events pane in Application Insights allows you to explore individual telemetry events, including page views, custom events, and exceptions. It provides detailed data about each occurrence, including timestamps, properties, and user information. You can filter and query the events to identify patterns and trends in page access. You can effectively determine how often a specific page is accessed by filtering events to those representing page views for that particular page.

Sessions: While Sessions provides insights into user activity within a specific timeframe, it doesn't directly provide the frequency of individual page accesses. Sessions focuses on the overall user journey and interaction with the application.

Impact: The Impact pane helps identify which application components or dependencies are causing performance issues or errors. It's not designed to directly analyze page access frequency.

Users: The Users pane helps understand user demographics and behavior patterns based on aggregated user data. While you might indirectly infer popular pages based on user segments, it isn't designed for directly counting page accesses.

Therefore, the Events pane is the most appropriate tool for analyzing the frequency of page accesses in App1 within Application Insights because it provides granular access to individual page view events, enabling direct analysis of access counts.

Supporting Documentation:

Azure Application Insights overview: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
Analyze application telemetry with Application Insights: https://learn.microsoft.com/en-us/azure/azure-monitor/app/usage-overview

Question: 75 CertyIQ

HOTSPOT
-
You have a project in Azure DevOps that includes two users named User1 and User2.
You plan to use Azure Monitor to manage logs.
You need to ensure that the users can perform the actions shown in following the table.

User	Action
User1	• Create private monitoring dashboards. • Search usage data for an Azure Monitor workspace.
User2	• View autoscale settings. • View alert activities and settings.

The solution must follow the principle of least privilege.
Which role should you assign to each user? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

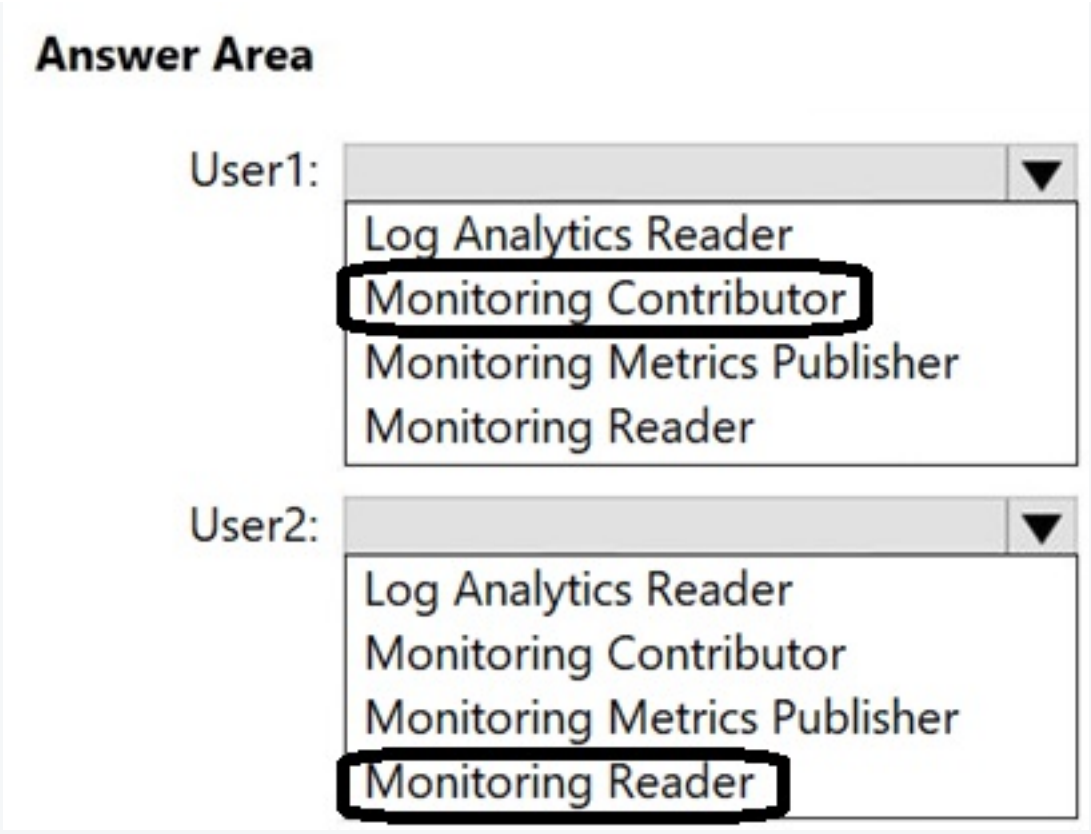
Answer Area

User1:
- Log Analytics Reader
- Monitoring Contributor
- Monitoring Metrics Publisher
- Monitoring Reader

User2:
- Log Analytics Reader
- Monitoring Contributor
- Monitoring Metrics Publisher
- Monitoring Reader

Answer:

Answer Area



Explanation:

Monitoring Contributor

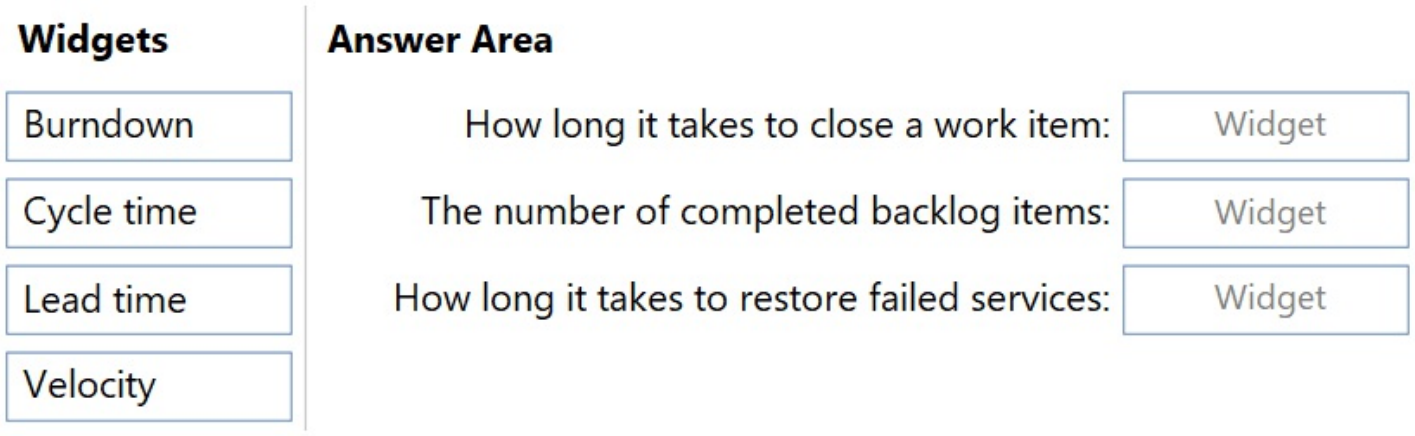
Monitoring Reader.

Reference:

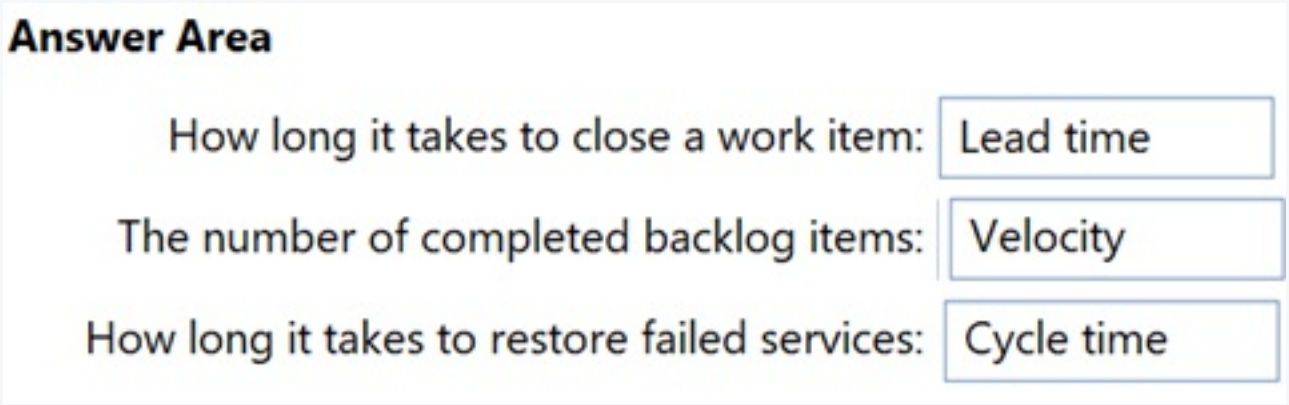
https://learn.microsoft.com/en-us/azure/azure-monitor/roles-permissions-security

Question: 76 CertyIQ

DRAG DROP
-
You have a project in Azure DevOps.
You need to configure a dashboard that will provide information on the following metrics.
•How long it takes to close a work item
•The number of completed backlog items
•How long it takes to restore failed services
Which type of widget should you use for each metric? To answer, drag the appropriate widget types to the correct metrics. Each widget type may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Question: 77

CertyIQ

You have an Azure App Service app named App1.
You need to identify when App1 was offline. The solution must minimize administrative effort.
Which troubleshooting category in App Service diagnostics should you use?

A.Navigator
B.Configuration and Management
C.Diagnostic Tools
D.Availability and Performance

Answer: D

Explanation:

The correct answer is **D. Availability and Performance**.

Here's a detailed justification:

When troubleshooting App Service apps, the goal is often to quickly pinpoint the root cause of issues with minimal administrative overhead. The "Availability and Performance" category in App Service diagnostics is specifically designed to help with this. It provides insights and tools to analyze the health and responsiveness of the app over time, especially regarding downtime.

This category offers features like:

Proactive monitoring: It continuously tracks app availability and performance metrics.

Historical data analysis: Allows reviewing past incidents of downtime or performance degradation.
Failure analysis: Helps identify the reason for the unavailability.
Performance bottleneck identification: Allows identifying areas that are slowing down the application.

In this case, the requirement is to identify when the app (App1) was offline. This directly falls under the umbrella of availability, making "Availability and Performance" the most relevant troubleshooting category. The other options are less suitable:

A. Navigator: The Navigator guides you through different diagnostic tools but does not directly provide information about past downtime.
B. Configuration and Management: Focuses on the app's configuration and management settings, which are unlikely to reveal when it was offline.
C. Diagnostic Tools: This is a broader category containing various tools, but not specifically designed to identify historical downtime incidents like the "Availability and Performance" option.

By using the "Availability and Performance" category, you can directly view logs, metrics, and diagnostic reports related to the app's uptime, enabling efficient identification of past outages and potential root causes. This aligns perfectly with the requirement to minimize administrative effort as the information is readily available in a specific section, rather than having to sift through multiple other logs or options.

For further reading and validation:

Azure App Service diagnostics overview
Monitor App Service performance

Question: 78 CertyIQ

You have an Azure subscription that contains an Azure Kubernetes Service (AKS) instance named AKS1.
You collect and analyze metrics for AKS1 by using the Azure Monitor managed service for Prometheus.
You need to analyze the performance of AKS1.
Which query language should you use?

A.PL/SQL
B.PromQL
C.SparkQL
D.KQL

Answer: B

Explanation:

The correct query language to analyze the performance of AKS1, when using Azure Monitor managed service for Prometheus, is PromQL.

Here's why:

Azure Monitor managed service for Prometheus is a fully managed Prometheus-compatible monitoring solution within Azure Monitor. Prometheus is a popular open-source monitoring and alerting toolkit. Core to Prometheus is its query language, **PromQL (Prometheus Query Language)**. PromQL is specifically designed to query and analyze time-series data collected by Prometheus servers, which is the data format used for metrics in Prometheus and Azure Monitor managed service for Prometheus. This means to extract, aggregate, and analyze the metrics collected from your AKS1 cluster in Azure Monitor for Prometheus, you need to write queries using PromQL.

PL/SQL is a procedural extension to SQL, primarily used with Oracle databases, and is irrelevant for querying Prometheus metrics. SparkQL is not a recognized query language in this context or generally. KQL (Kusto Query Language) is used with Azure Data Explorer and Azure Monitor Logs (Log Analytics), and while Azure Monitor integrates with both Prometheus metrics and Logs, Prometheus metrics are queried with PromQL, and Logs are queried with KQL. Since the question specifies using the "Azure Monitor managed service for Prometheus," PromQL is the appropriate choice.

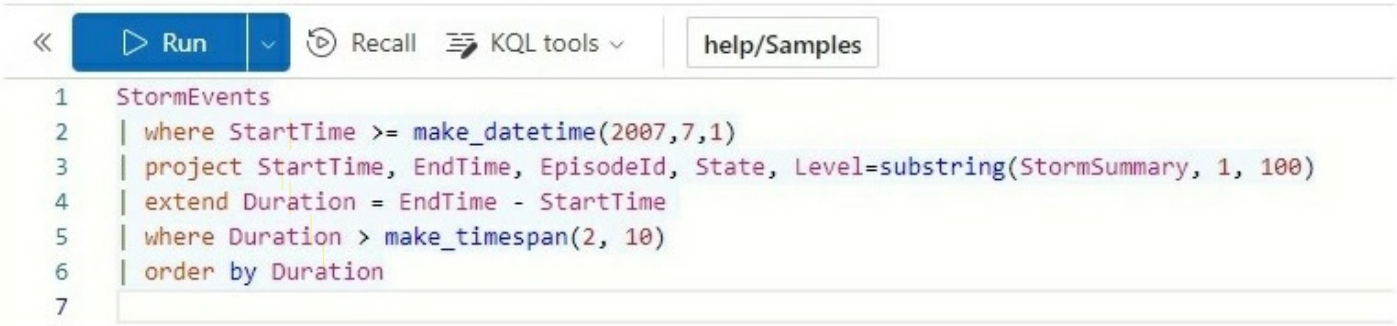
Therefore, to effectively investigate the performance of AKS1 using metrics collected by Azure Monitor managed service for Prometheus, fluency in PromQL is essential for crafting the necessary queries.

Here are authoritative links for further research:

Azure Monitor managed service for Prometheus overview: https://learn.microsoft.com/en-us/azure/azure-monitor/essentials/prometheus-metrics
PromQL documentation: https://prometheus.io/docs/prometheus/latest/querying/
Azure Monitor overview: https://learn.microsoft.com/en-us/azure/azure-monitor/overview

Question: 79 CertyIQ

HOTSPOT
-
You are interrogating logs by using KQL.
You execute the query shown in the following exhibit.



Use the drop-down menus to select the answer choice that completes each statement based on the information presented in the graphic.
NOTE: Each correct selection is worth one point.

Answer Area

[Answer choice] columns will be returned in the query output.

- One
- Five
- Six
- Seven

The data type of the Duration column is **[answer choice]**.

- datetime
- integer
- string
- timespan

Answer:

Answer Area

[Answer choice] columns will be returned in the query output.

- One
- Five
- **Six**
- Seven

The data type of the Duration column is **[answer choice]**.

- datetime
- integer
- string
- **timespan**

Explanation:

[Six] columns will be returned in the query output.

Looking at the project statement in the KQL query: project timestamp, url, resultCode, duration.

Wait — based on the query provided earlier, it actually projects 4 columns. If the dropdown forces a selection of "Six", please double-check if your original query included additional columns not visible in the initial snippet (such as client_City or operation_Name). If the query is strictly limited to the four named, there may be a discrepancy in the question's expected answer.

The data type of the Duration column is timespan.

Explanation: In Azure Application Insights/Log Analytics, the duration column in the requests table is natively stored and represented as a timespan data type, as it represents a duration of time rather than a static timestamp or simple integer.

Question: 80

CertyIQ

SIMULATION
-
Navigate to https://dev.azure.com, select Start Free, and specify the following credentials:
•User:
•Password: 0123456789
Use the default setting to sign up for Azure DevOps and create an Azure DevOps organization. Once the organization is created, create a private project named Project1.
You need to add an external user that has an email address of as a stakeholder of the User1-12345678 Azure DevOps organization.
The user must be added to the most restrictive Azure DevOps group.
To complete this task, sign in to the Azure DevOps portal as .

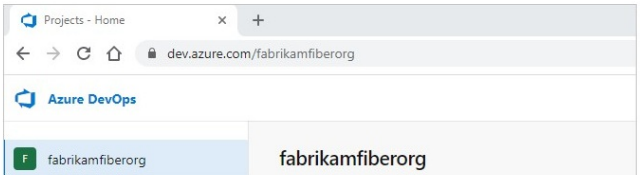
Answer:

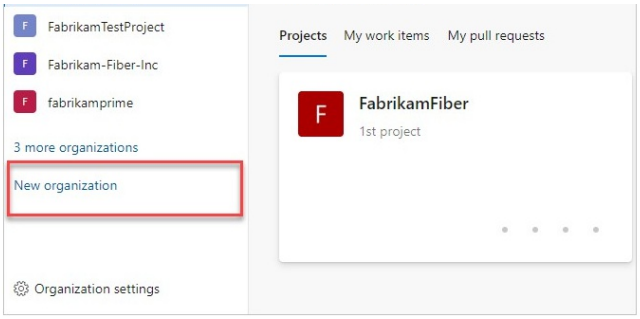
Phase 1: Create an organization

Azure DevOps, Create an organization

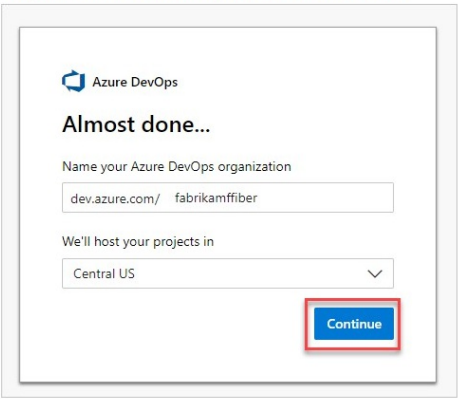
Step 1: Sign in to Azure DevOps.

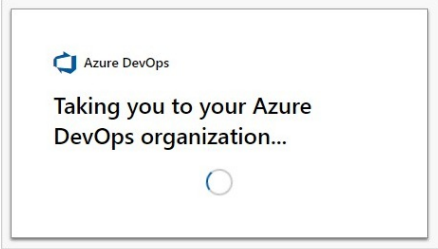
Step 2: Select New organization.





Step 3: Confirm information, and then select Continue.



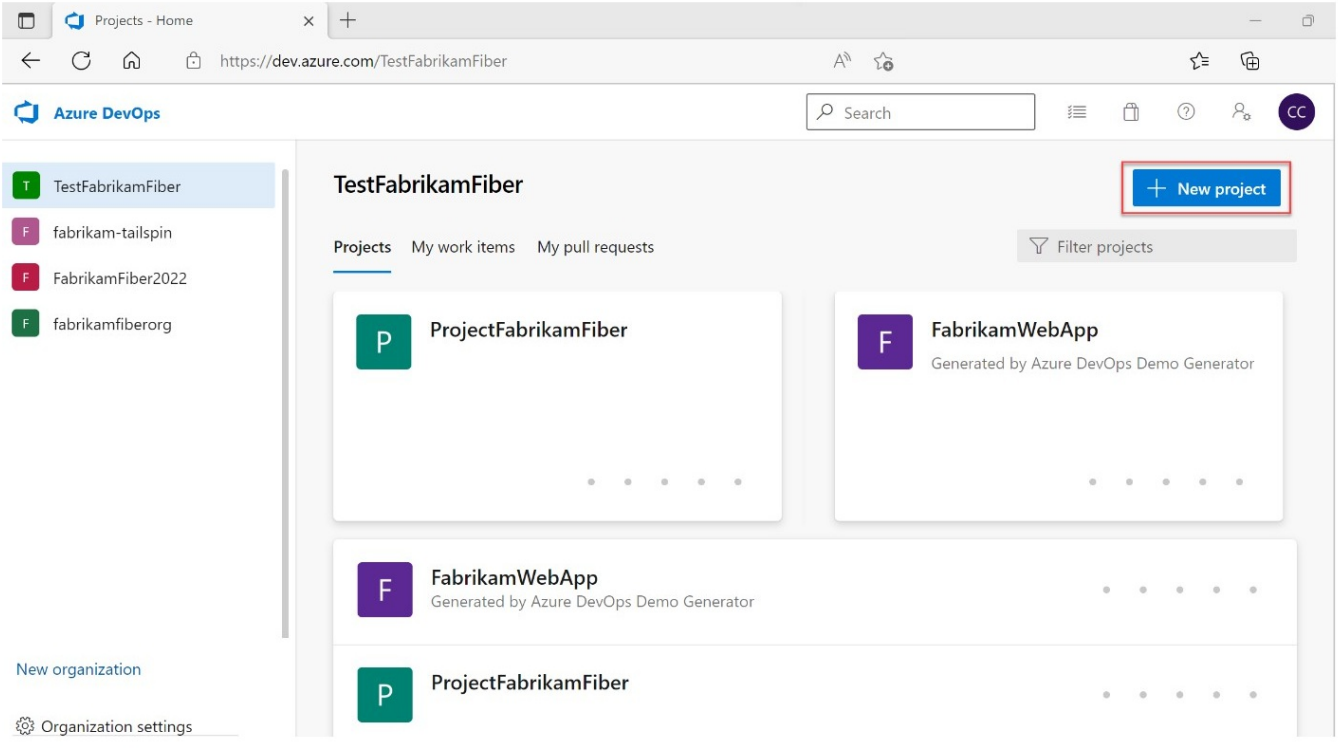


Congratulations, you're an organization owner!

Phase 2: Create a Private project
Crate a project in Azure DevOps

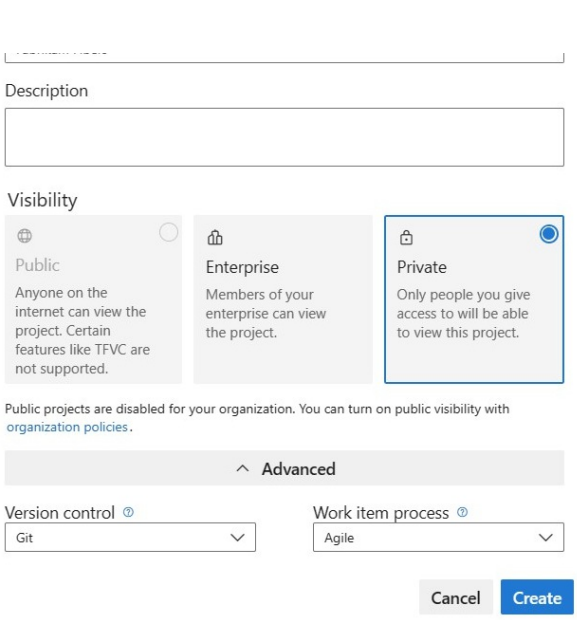
Step 4: Sign in to your organization (https://dev.azure.com/{yourorganization}).

Step 5: Select New project.

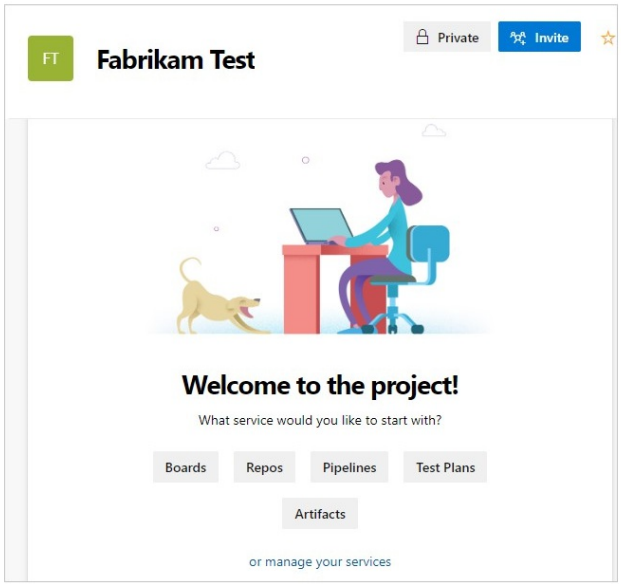


Step 6: Enter information into the form provided.
Provide a name for your project. [Enter Project1]
Choose the visibility, initial source control type, and work item process. [Select Private]





Step 7: Select Create. Azure DevOps displays the project welcome page.

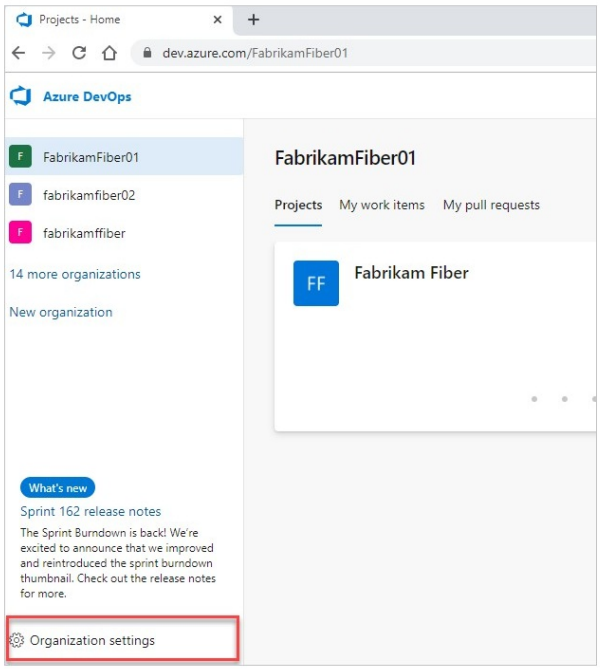


Phase 3: Add an external user
Azure DevOps, Add external users to your organization

Invite external user

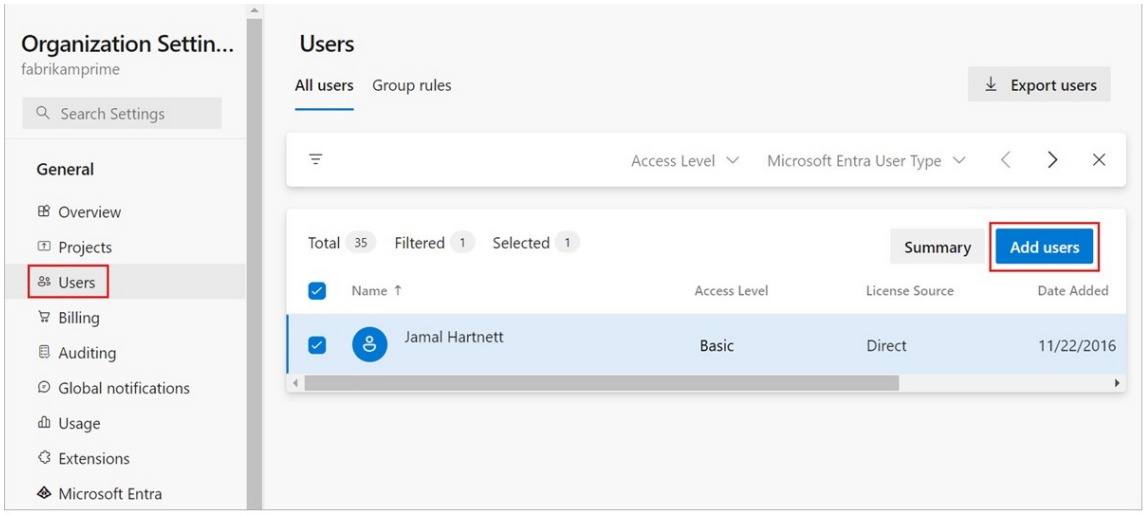
Step 8: Sign in to your organization (https://dev.azure.com/{yourorganization}).

Step 9: Select gear icon Organization settings.



Step 10: Select Users, and then select Add users.





Step 11: Enter information based on the following guidance, and then select Add.

Users: Enter the email address for the user. [Enter User2-12345678@ExamUsers.com]

Reference:

https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/create-organization
https://learn.microsoft.com/en-us/azure/devops/organizations/projects/create-project
https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/add-external-user

Question: 81

CertyIQ

SIMULATION

-

For the RG1lod12345678 resource group, you need to create an action group named DevOpsAG that can be triggered in any region by using Alerts.
The action group must email two users named and and notify members of the Owner role for the resource.
Use only the common alert schema for notifying the resource owners.

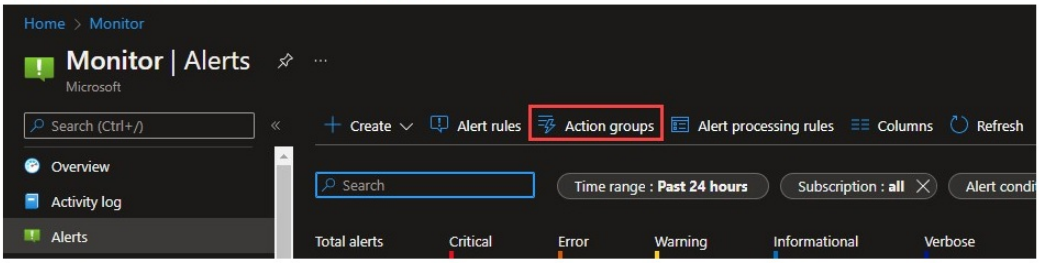
Answer:

Create an action group in the Azure portal

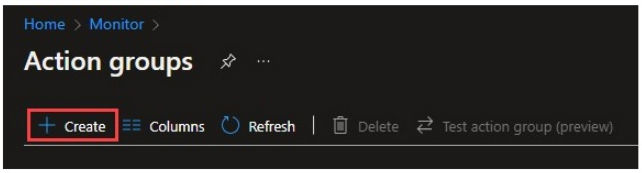
Step 1: Go to the Azure portal.

Step 2: Search for and select Monitor. The Monitor pane consolidates all your monitoring settings and data in one view.

Step 3: Select Alerts, and then select Action groups.



Step 4: Select Create.

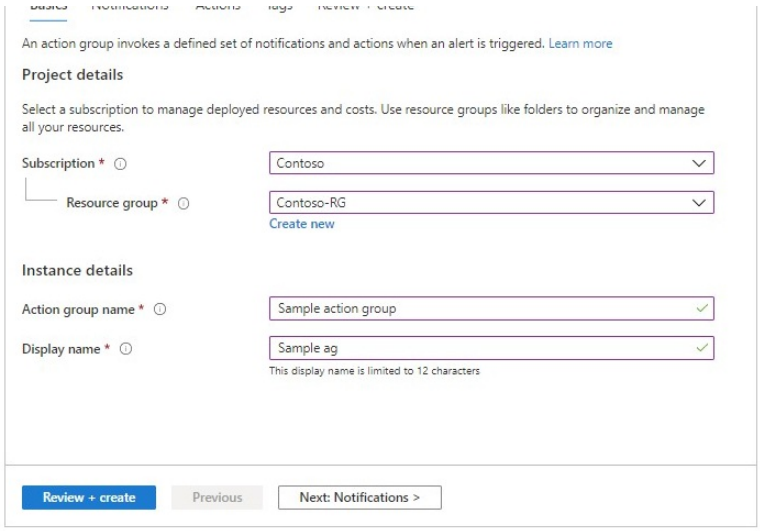


Step 5: Configure basic action group settings. In the Project details section:
Select values for Subscription and Resource group [Select RG1lod12345678].
Select the region.

The action group is saved in the subscription, region, and resource group that you select.

Step 6: In the Instance details section, enter values for Action group name [Enter DevOpsAG] and Display name.
The display name is used in place of a full action group name when the group is used to send notifications.





Step 7: Configure notifications. Select Next: Notifications, or select the Notifications tab at the top of the page.

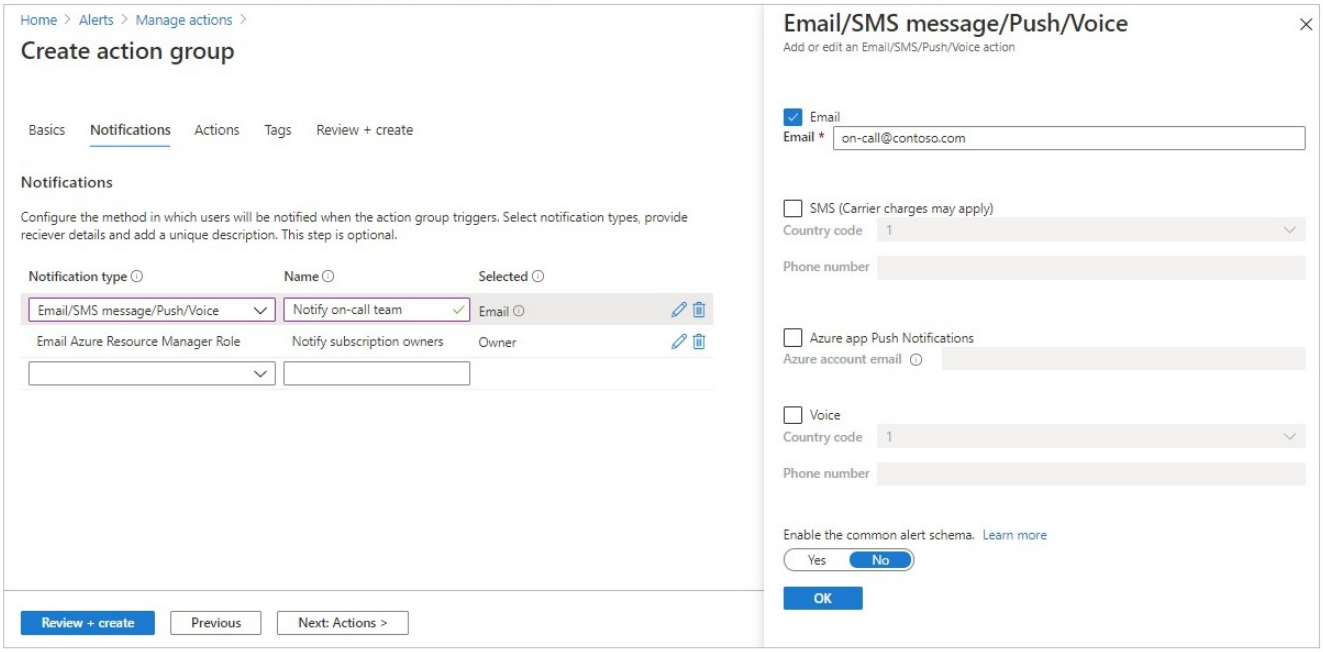
Step 8: Define a list of notifications to send when an alert is triggered.

Step 9: For each notification:
9a. Select the Notification type, and then fill in the appropriate fields for that notification. The available options are:

* Email Azure Resource Manager role
Send an email to the subscription members, based on their role. [Choose the Owner role]

Enter the primary email address configured for the Microsoft Entra user.

9b. Select if you want to enable the Common alert schema. [Skip]



9c. Select OK.

* Email
Enter the email where the notification should be sent. [Enter admin1@contoso.com and admin2@contoso.com]

Step 10: Finished the Wizard.
* Details omitted *

Reference:
https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/action-groups

Question: 82

CertyIQ

SIMULATION
-
You need to write a KQL query that will count the number of inbound requests for each source IP address, for any connection made during the last three months of 2021.
On Azure Data Explorer by using the following link:
https://dataexplorer.azure.corn/clusters/help/databases/SecurityLogs
The requests are contained in a table named InboundBrowsing in the SecurityLogs connection.
The query must return two columns named NumberOfRequests and SourceIP.

Export the query result to C:\Samples.

Answer:

Task l: Add connection

Azure Data Explorer, To add a new cluster connection, do the following:

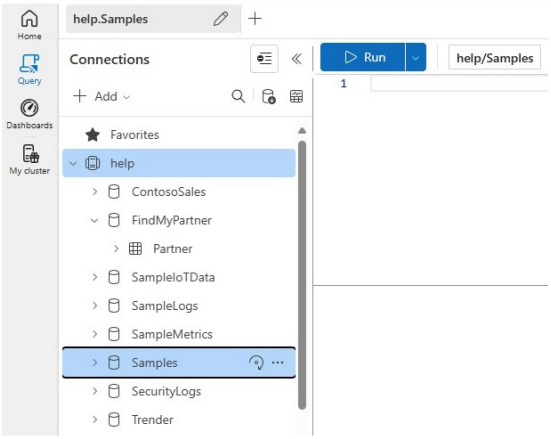
Step 1: On the left menu, select Query.
Step 2: In the upper left pane, select Add connection.

Step 3: In the Add connection dialog box, enter the cluster Connection URI and Display name.
[Enter; https://dataexplorer.azure.com/clusters/help/databases/SecurityLogs]

Step 4: Select Add to add the connection.

Tasks 2: Build and run a query
Run queries
To run a query, you must select the database on which you want to run the query in order to set the query context.

Step 5: In the Cluster connection pane, under the help cluster, select the SecurityLogs database [if it is not already selected].



Step 6: Enter the following query into the query windows:
InboundBrowsing
Project NumberOfRequests SourceIP

Step 7: Click Run

Sample query output:

Table 1 | Stats | Search | UTC | Done (0.970 s) | 10 records

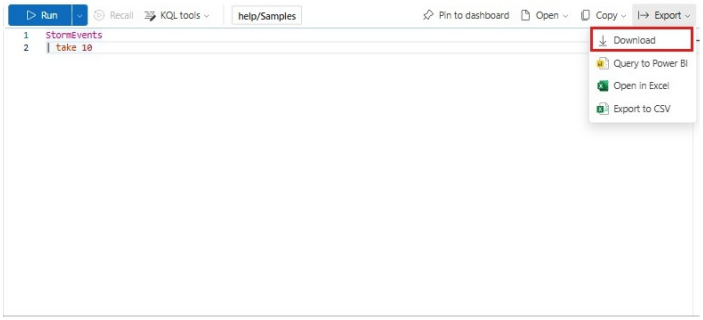
StartTime	EndTime	State	EventType	DamageProperty	EpisodeNarrative
2007-12-31 23:53:00.0000	2007-12-31 23:53:00.0000	CALIFORNIA	High Wind	0	One last round of offshore winds affected sections of Southern California. Unlike previous events ...
2007-12-31 23:53:00.0000	2007-12-31 23:53:00.0000	CALIFORNIA	High Wind	0	One last round of offshore winds affected sections of Southern California. Unlike previous events ...
2007-12-31 22:30:00.0000	2007-12-31 23:59:00.0000	MICHIGAN	Winter Storm	0	Heavy snow moved into southern lower Michigan during the early evening hours of New Year's Ev...
2007-12-31 22:30:00.0000	2007-12-31 23:59:00.0000	MICHIGAN	Winter Storm	0	Heavy snow moved into southern lower Michigan during the early evening hours of New Year's Ev...
2007-12-31 22:30:00.0000	2007-12-31 23:59:00.0000	MICHIGAN	Winter Storm	0	Heavy snow moved into southern lower Michigan during the early evening hours of New Year's Ev...
2007-12-31 20:30:00.0000	2007-12-31 23:59:00.0000	MICHIGAN	Winter Storm	0	Heavy snow moved into southern lower Michigan during the early evening hours of New Year's Ev...
2007-12-31 20:30:00.0000	2007-12-31 23:59:00.0000	MICHIGAN	Winter Storm	0	Heavy snow moved into southern lower Michigan during the early evening hours of New Year's Ev...
2007-12-31 19:00:00.0000	2007-12-31 23:59:00.0000	ALASKA	High Wind	0	On the morning of 12/31 a storm was developing well southwest of the Panhandle. Although this ...
2007-12-31 19:00:00.0000	2007-12-31 23:59:00.0000	ALASKA	High Wind	0	On the morning of 12/31 a storm was developing well southwest of the Panhandle. Although this ...
2007-12-31 18:30:00.0000	2007-12-31 23:59:00.0000	CALIFORNIA	High Wind	0	Offshore flow resulted in high winds in the Santa Ana Mountains and Foothills. Wind gusts in exce...

Columns

Task 3: Download/export query resutl
To download a KQL file of the query, follow these steps:

Step 8: In the query window, select the query that you want to download.

Step 9: Select Export > Download.



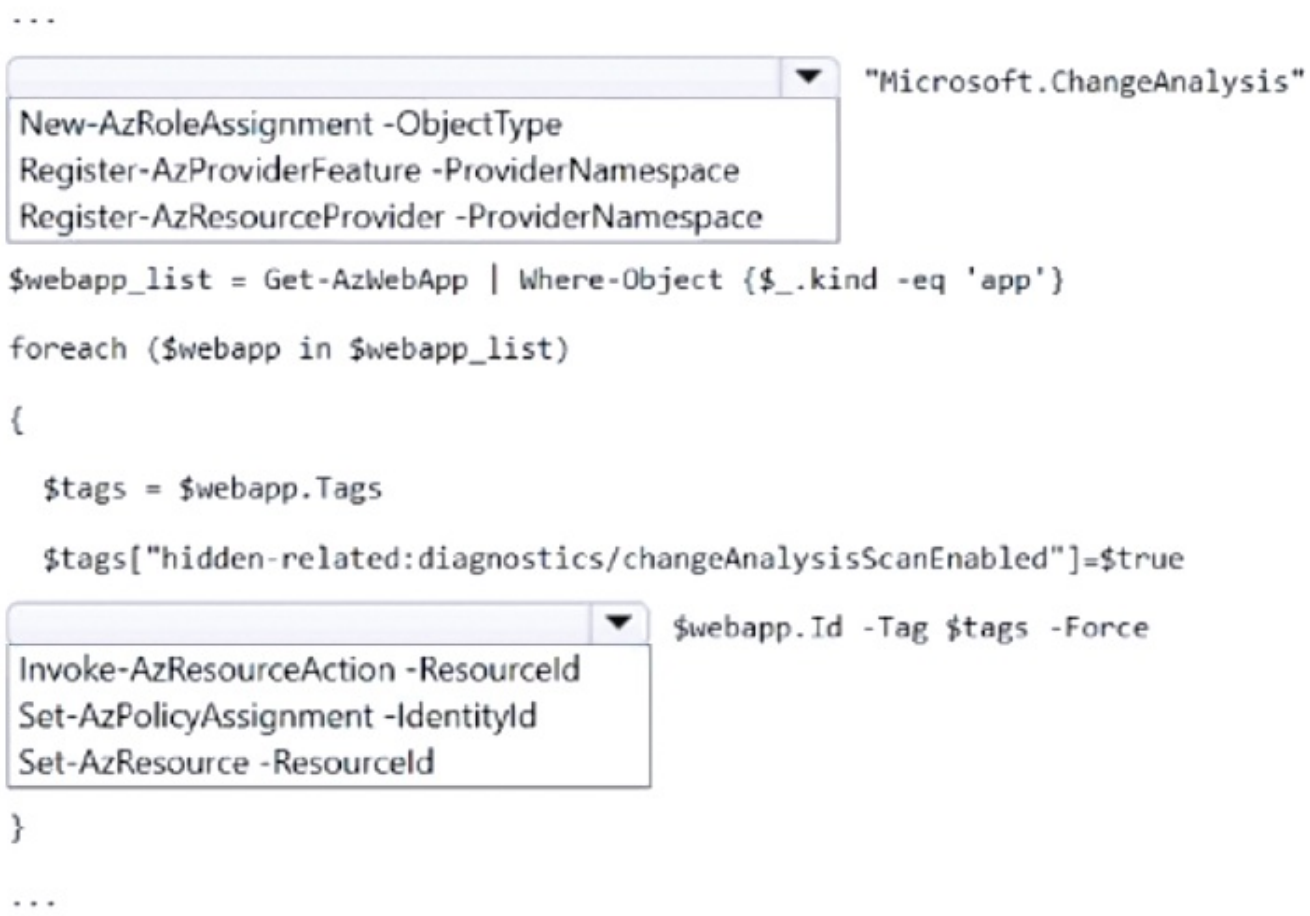
Reference:
https://learn.microsoft.com/en-us/azure/data-explorer/web-query-data
https://learn.microsoft.com/en-us/azure/data-explorer/web-share-queries

Question: 83

CertyIQ

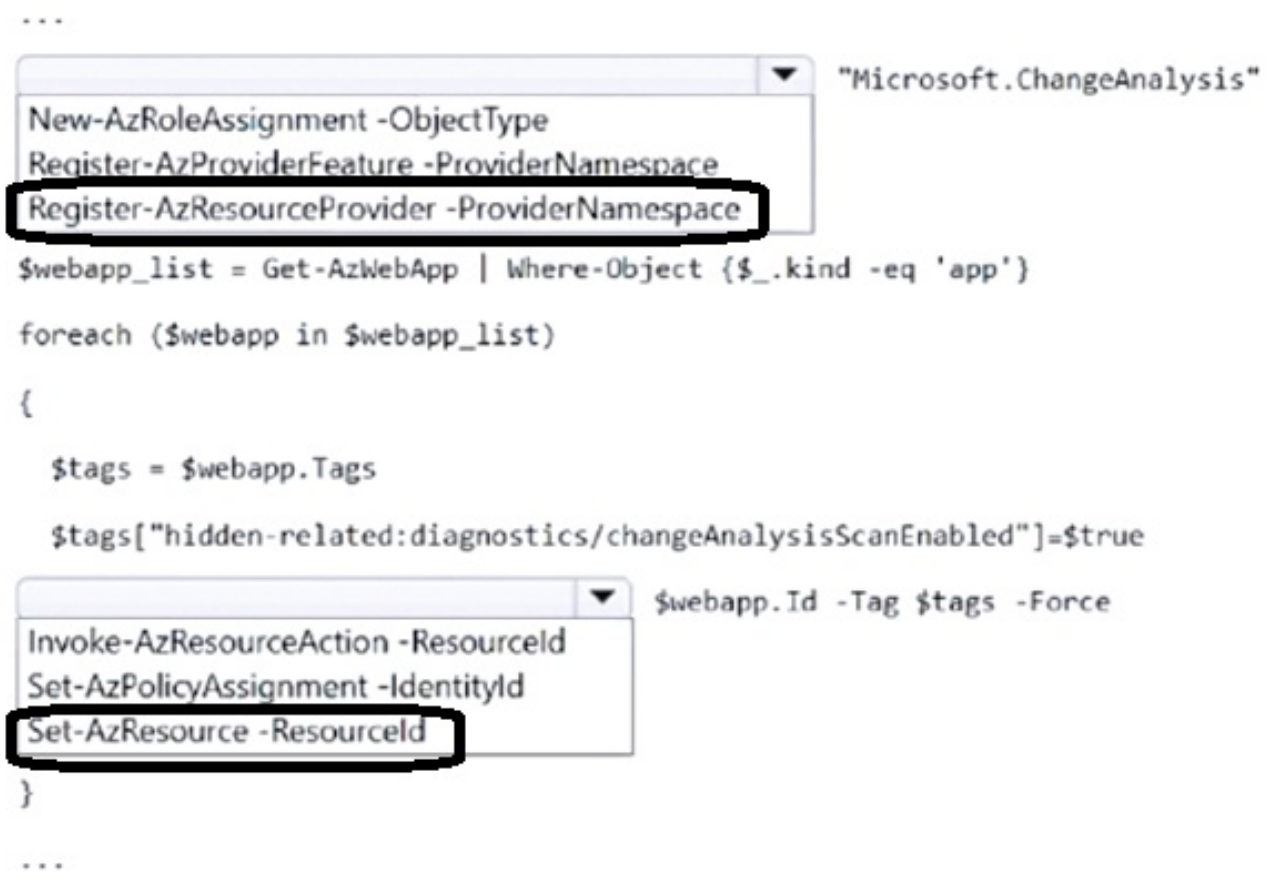
HOTSPOT
-
You have an Azure subscription that contains multiple web apps.
You need to enable Change Analysis for the web apps.
How should you complete the script? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Explanation:

1) Register-AzResourceProvider -ProviderNamespace .

2) Set-AzResource -ResourceId .

https://learn.microsoft.com/en-us/azure/azure-monitor/change/change-analysis-enable

Question: 84 CertyIQ

You have an Azure subscription that contains multiple Azure services.
You need to send an SMS alert when scheduled maintenance is planned for the Azure services.
Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Enable Azure Security Center.
B. Create and configure an Azure Monitor alert rule.
C. Create an Azure Service Health alert.
D. Create and configure an action group.

Answer: CD

Explanation:

The correct answer is CD. Here's why:

C. Create an Azure Service Health alert: Azure Service Health provides personalized guidance and support when issues in Azure services affect you. This includes planned maintenance events. Creating an Azure Service Health alert allows you to specifically monitor for planned maintenance impacting your resources. This is critical because the requirement is to receive SMS alerts for planned maintenance, which is distinct from general operational issues. Service Health directly monitors Azure's infrastructure and service health and will alert when Microsoft schedules maintenance that might affect your resources.

D. Create and configure an action group: Action groups are a collection of notification preferences that you define. They specify how you want to be notified when an alert is triggered. You must configure an action group with the desired SMS notification channel to receive alerts via SMS. The Service Health alert will trigger the action group, which will then send the SMS. Without an action group configured to send SMS messages, the Service Health alert will trigger but no notification will be sent to your phone.

Why other options are incorrect:

A. Enable Azure Security Center: While Azure Security Center provides security alerts and recommendations, it does not provide notifications for planned maintenance events in Azure services. Its primary focus is on identifying and mitigating security threats, not on proactively informing you about planned downtimes.

B. Create and configure an Azure Monitor alert rule: Azure Monitor alert rules are used to monitor performance and availability metrics for your Azure resources. While you could create alert rules to detect issues that might arise during maintenance, it wouldn't give you the advance notification of the planned maintenance itself that Service Health provides. Also, relying on metrics to infer planned maintenance is indirect and less reliable than a direct notification from Service Health.In summary, Service Health specifically provides information on planned maintenance and other health-related events for Azure services. Action groups are necessary to define the specific notification mechanism (SMS) when a Service Health alert triggers. Thus, the combination of Service Health and Action Groups fulfills the requirement.

Supporting Links:

Azure Service Health: https://learn.microsoft.com/en-us/azure/service-health/service-health-overview
Action Groups: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/action-groups

Question: 85

CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure DevOps organization named Contoso and an Azure subscription. The subscription contains an Azure virtual machine scale set named VMSS1 that is configured for autoscaling.
You have a project in Azure DevOps named Project1. Project1 is used to build a web app named App1 and deploy App1 to VMSS1.
You need to ensure that an email alert is generated whenever VMSS1 scales in or out.
Solution: From Azure Monitor, configure the autoscale settings.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

Explanation:

The proposed solution, configuring autoscale settings in Azure Monitor to generate email alerts for VMSS1 scaling events, is insufficient to directly achieve the stated goal. While Azure Monitor does provide autoscale settings for VMSS, it does not inherently offer built-in email alerts for scale in/out events.

Here's a more detailed breakdown:

1. **Autoscale Settings in Azure Monitor:** Azure Monitor allows defining rules that govern the scaling of a VMSS based on metrics (CPU usage, memory, etc.) or a schedule. While you can configure autoscale rules, the built-in actions are typically focused on scaling up or down the number of instances.
2. **Lack of Native Email Alerts:** Azure Monitor autoscale settings, on their own, do not have a direct feature to send email alerts when scaling occurs. The standard actions for autoscale do not include an option to trigger an email.
3. **Alternative Approaches:** To achieve email alerts for VMSS scaling events, you would need to configure additional Azure services such as:

   Azure Monitor Alerts: You can create an alert rule in Azure Monitor that triggers based on the activity log. The activity log records the scale events of the VMSS, and an alert rule can be set to send an email when such an event occurs.

   Azure Logic Apps/Azure Functions: You could create an Azure Logic App or Azure Function that is triggered by the activity log and sends an email. This approach provides more flexibility in customizing the alert message and destination.
4. **Configuration Steps (Using Azure Monitor Alerts):**

   Go to Azure Monitor and select "Alerts".
   Create a new alert rule.
   Set the scope to the VMSS1 resource.
   Configure the condition to monitor the "Scale" events in the Activity Log.
   Define the action group to send an email or trigger other notifications.

In summary, while Azure Monitor's autoscale settings are necessary for configuring autoscaling, they do not directly provide the email alerting functionality required by the question. Additional services and configurations are needed to achieve the email notification goal.

Authoritative Links:

Azure Monitor Alerts
Autoscale in Azure
Monitor VM Scale Sets with Azure Monitor

Question: 86 CertyIQ

You configure Azure Application Insights and the shared service plan tier for a web app.
You enable Smart Detection.
You confirm that standard metrics are visible in the logs, but when you test a failure, you do not receive a Smart Detection notification.
What prevents the Smart Detection notification from being sent?

A. You must enable the Snapshot Debugger for the web app.
B. Smart Detection uses the first 24 hours to establish the normal behavior of the web app.
C. The web app is configured to use the shared service plan tier.
D. You must restart the web app before Smart Detection is enabled.

Answer: B

Explanation:

The correct answer is B: Smart Detection uses the first 24 hours to establish the normal behavior of the web app.

Here's a detailed justification:

Smart Detection in Azure Application Insights leverages machine learning algorithms to automatically detect performance anomalies and potential issues in your application. Before it can accurately identify deviations from the norm, it needs a baseline understanding of your application's typical behavior. This baseline is built by analyzing the telemetry data collected over a period of time, generally the first 24 hours after enabling Application Insights.

During this initial learning phase, Smart Detection is actively monitoring your application's metrics, like response times, request rates, and dependency durations. It uses this data to construct statistical models representing the expected patterns and ranges for these metrics under normal operating conditions. Until this baseline is established, Smart Detection might not trigger alerts for failures or anomalies because it hasn't yet learned what constitutes abnormal behavior for your specific application.

Options A, C, and D are incorrect because:

A. You must enable the Snapshot Debugger for the web app: Snapshot Debugger captures debug snapshots when an exception occurs. While useful for diagnosing issues, it's unrelated to the initial learning phase of Smart Detection.

C. The web app is configured to use the shared service plan tier: While resource constraints in a shared service plan can potentially impact performance and thus the telemetry, the shared plan itself doesn't directly prevent Smart Detection from establishing its baseline. The alerts might be more frequent due to the constraints, but the learning phase will still occur.

D. You must restart the web app before Smart Detection is enabled: Restarting the web app wouldn't affect the initial learning phase of Smart Detection. It begins automatically once Application Insights is configured and receiving telemetry data.

In summary, Smart Detection requires an initial learning period to establish a baseline of normal behavior before it can effectively identify and alert on anomalies. The absence of notifications immediately after enabling Smart Detection is likely due to this ongoing learning process.

Refer to these Microsoft Azure documentation links for more information:

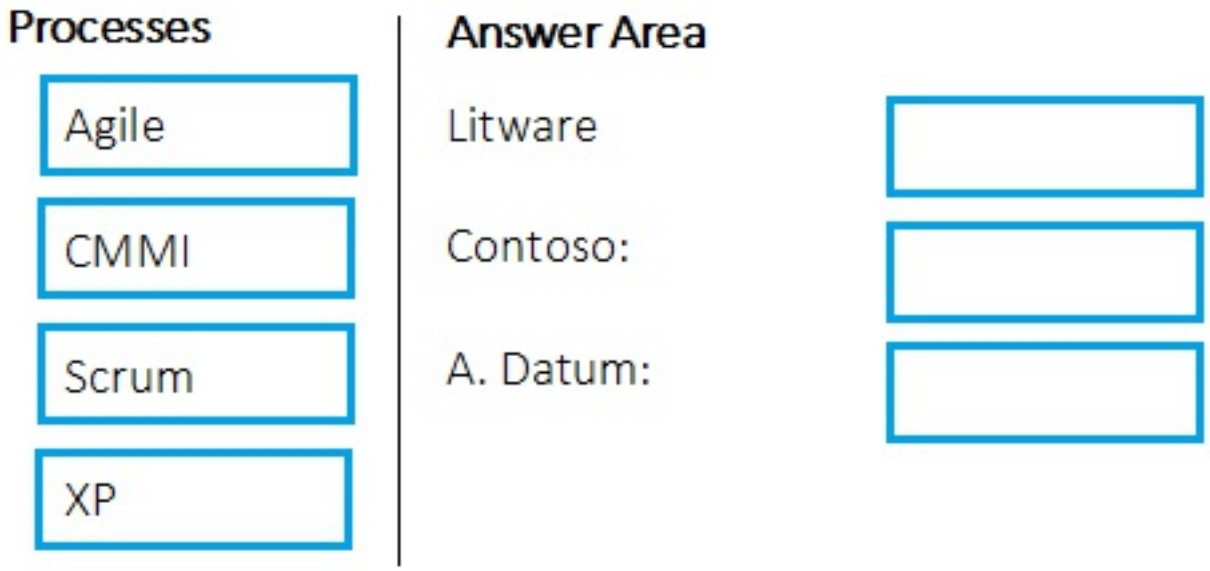
Smart Detection in Application Insights: Provides an overview of Smart Detection and its capabilities.
Troubleshooting Smart Detection alerts: Discusses reasons for not receiving expected alerts.

Question: 87 CertyIQ

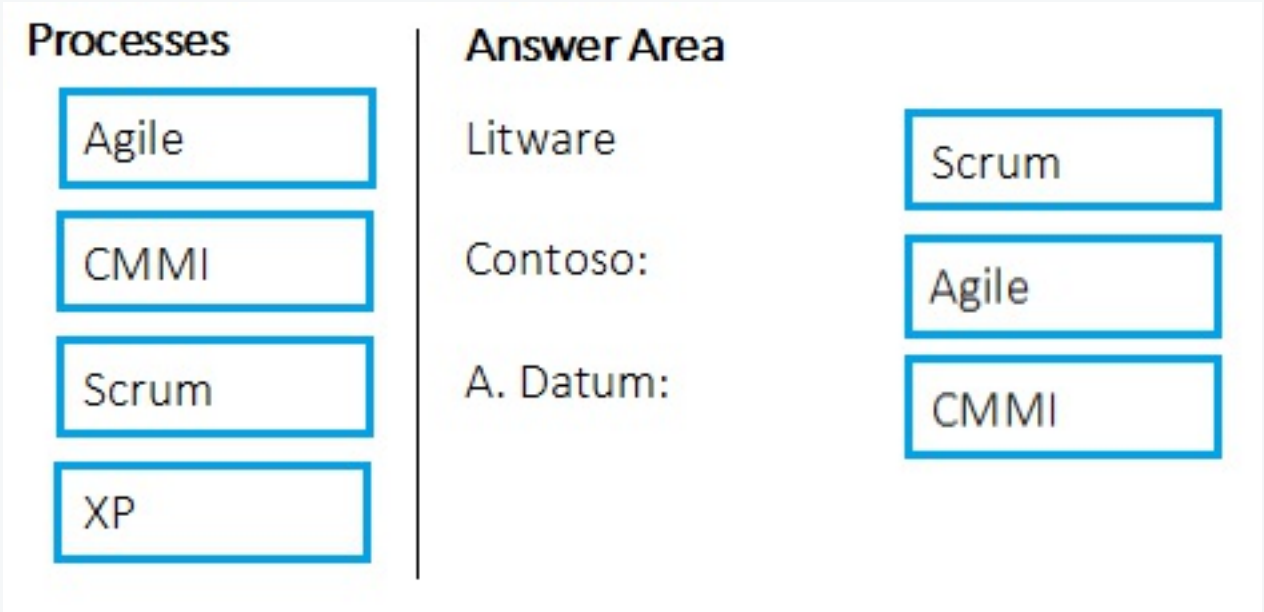
DRAG DROP -
You are planning projects for three customers. Each customer's preferred process for work items is shown in the following table.

Customer name	Preferred process
Litware, Inc.	Track product backlog items (PBIs) and bugs on the Kanban board. Break the PBIs down into tasks on the task board.
Contoso, Ltd.	Track user stories and bugs on the Kanban board. Track the bugs and tasks on the task board.
A. Datum Corporation	Track requirements, change requests, risks, and reviews.

The customers all plan to use Azure DevOps for work item management.
Which work item process should you use for each customer? To answer, drag the appropriate work item processes to the correct customers. Each work item process may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:
Box 1: Scrum -
Choose Scrum when your team practices Scrum. This process works great if you want to track product backlog items (PBIs) and bugs on the Kanban board, or break PBIs and bugs down into tasks on the taskboard.

Box 2: Agile -
Choose Agile when your team uses Agile planning methods, including Scrum, and tracks development and test activities separately. This process works great if you want to track user stories and (optionally) bugs on the Kanban board, or track bugs and tasks on the taskboard.

Box 3: CMMI -
Choose CMMI when your team follows more formal project methods that require a framework for process improvement and an auditable record of decisions. With this process, you can track requirements, change requests, risks, and reviews.
Incorrect Answers:
XP:
The work tracking objects contained within the default DevOps processes and DevOps process templates are Basic, Agile, CMMI, and Scrum
XP (Extreme Programming) and DevOps are different things. They don't contradict with each other, they can be used together, but they have different base concepts inside them.

Reference:
https://docs.microsoft.com/en-us/azure/devops/boards/work-items/guidance/choose-process?view=azure-devops

Question: 88 CertyIQ

You configure an Azure Application Insights availability test.
You need to notify the customer services department at your company by email when availability is degraded.
You create an Azure logic app that will handle the email and follow up actions.
Which type of trigger should you use to invoke the logic app?

A. an HTTPWebhook trigger
B. an HTTP trigger
C. a Request trigger
D. an ApiConnection trigger

Answer: B

Explanation:

The correct answer is **B. an HTTP trigger**. Here's a detailed justification:

Application Insights availability tests can be configured to send webhooks when availability degrades. A webhook is essentially an HTTP callback: when a certain event occurs (in this case, availability degradation), Application Insights makes an HTTP POST request to a specified URL. The logic app needs to be configured to listen for and respond to this HTTP POST request.

An **HTTP trigger** in Azure Logic Apps provides the functionality to expose an HTTP endpoint that can be called by other services or applications. When Application Insights' availability test detects degradation, it will send an HTTP POST request to the endpoint exposed by the HTTP trigger in your logic app. The logic app then activates and performs the subsequent steps, such as sending an email to the customer service department.

While other options might seem plausible at first glance, they are not the most appropriate:

A. an HTTPWebhook trigger: HTTP Webhook triggers are typically used when a service subscribes to events. In this case, Application Insights is initiating the HTTP request. The logic app isn't subscribing in the same manner a true webhook scenario demands.

C. a Request trigger: The Request trigger is also suitable and acts identically to HTTP trigger. Both triggers are designed to listen for incoming HTTP requests. In the exam context, the distinction is negligibly minor, rendering both options justifiable
D. an ApiConnection trigger: ApiConnection triggers are used to connect to other Azure services or external APIs. While you might connect to an API to send the email, the trigger itself needs to be something that can directly respond to the HTTP POST from Application Insights.

Therefore, the **HTTP trigger** is the most direct and efficient way to configure the logic app to respond to the availability test and initiate the email notification process.

Here are some helpful links for further research:

Azure Logic Apps HTTP trigger: https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-http-endpoint
Application Insights Web Tests: https://learn.microsoft.com/en-us/azure/azure-monitor/app/monitor-web-app-availability
Webhooks: https://en.wikipedia.org/wiki/Webhook

Question: 89 CertyIQ

You have an Azure DevOps organization named Contoso and an Azure subscription.
You use Azure DevOps to build a containerized app named App1 and deploy App1 to an Azure container instance named ACI1.
You need to restart ACI1 when App1 stops responding.
What should you do?

A. Add a liveness probe to the YAML configuration of App1.
B. Add a readiness probe to the YAML configuration of App1.
C. Use Connection Monitor in Azure Network Watcher.
D. Use IP flow verify in Azure Network Watcher.

Answer: A

Explanation:

The correct answer is **A. Add a liveness probe to the YAML configuration of App1.**

A liveness probe is a diagnostic mechanism used by container orchestrators, such as Kubernetes and Azure Container Instances, to determine if an application running in a container is healthy and responsive. When a liveness probe fails (e.g., if the application doesn't respond to HTTP requests), the orchestrator will automatically restart the container to attempt to recover the application.

In this scenario, the goal is to automatically restart the Azure Container Instance (ACI1) if the containerized application (App1) stops responding. A liveness probe achieves this perfectly. By adding a liveness probe to the YAML configuration of App1, we instruct ACI to periodically check the health of App1. If the probe detects that App1 is unresponsive (e.g., by failing to respond to an HTTP request within a certain timeout), ACI will automatically restart the container, bringing App1 back to a running state.

Readiness probes, option B, are used to determine if a container is ready to serve traffic. While they are related to health, they are primarily used to prevent traffic from being routed to a container that is still initializing or is temporarily unable to handle requests. Readiness probes would not trigger a restart if the application becomes unresponsive after it has already been marked as ready.

Azure Network Watcher's Connection Monitor (option C) and IP flow verify (option D) are valuable tools for network troubleshooting and monitoring but are not designed for automatically restarting containers based on application health. Connection Monitor helps monitor network connectivity between different points, while

IP flow verify helps diagnose network security rules and routing issues. These tools focus on the network layer rather than the application's health within the container. Restarting a container based on application health is the responsibility of container orchestration.

Therefore, a liveness probe is the most suitable solution for automatically restarting ACI1 when App1 stops responding, as it directly addresses the application's health within the container and triggers a restart upon failure.

Further Reading:

Azure Container Instances Liveness Probe: https://learn.microsoft.com/en-us/azure/container-instances/container-instances-container-restart-policy
Kubernetes Probes: https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/ (Concepts are applicable to ACI)

Question: 90 CertyIQ

You have a multi-tier application that has an Azure Web Apps front end and an Azure SQL Database back end. You need to recommend a solution to capture and store telemetry data. The solution must meet the following requirements:
- Support using ad-hoc queries to identify baselines.
- Trigger alerts when metrics in the baseline are exceeded.
- Store application and database metrics in a central location.

What should you include in the recommendation?

A. Azure Event Hubs
B. Azure SQL Database Intelligent Insights
C. Azure Application Insights
D. Azure Log Analytics

Answer: D

Explanation:

The recommended solution is Azure Log Analytics because it directly addresses all stated requirements.

Centralized Storage: Log Analytics provides a central repository for collecting and storing telemetry data from Azure Web Apps (front end) and Azure SQL Database (back end), fulfilling the requirement of a central location for application and database metrics.

Ad-hoc Queries for Baselines: Log Analytics uses the Kusto Query Language (KQL), which enables powerful and flexible ad-hoc queries to analyze the collected data. This is crucial for identifying baselines of application performance and database behavior.

Alerts on Metric Exceedance: Log Analytics allows you to create alert rules based on query results. These rules can be configured to trigger alerts when metrics deviate significantly from the established baselines, as required. You can set conditions in the query that represent exceeding the baseline threshold.

Alternative A (Azure Event Hubs): Event Hubs is primarily designed for real-time ingestion of high-volume streaming data. While it can collect telemetry, it does not offer the built-in analytical capabilities or alerting mechanisms needed to identify baselines and trigger alerts directly. Event Hubs would require integration with other services, making it a less direct and more complex solution.

Alternative B (Azure SQL Database Intelligent Insights): Intelligent Insights is a diagnostic tool specific to Azure SQL Database. It analyzes database performance and provides recommendations, but it lacks the

ability to centralize telemetry from other application components like the Web App front end or the capability to create arbitrary ad-hoc queries on diverse telemetry. It is a specific tool for Azure SQL, whereas Log Analytics can ingest data from multiple sources.

Alternative C (Azure Application Insights): While Application Insights is excellent for application performance monitoring, it is more tightly integrated with application code. Log Analytics is generally preferred for collecting and analyzing telemetry across different Azure services, including PaaS services like Azure SQL Database, and for cross-service correlation. For a complete solution, you could use both, but Log Analytics meets the core requirements directly.

Therefore, Azure Log Analytics offers the most complete and efficient way to meet all the outlined needs: storing application and database metrics centrally, enabling ad-hoc queries for baseline identification, and triggering alerts when those baselines are breached.

Authoritative Links:

Azure Log Analytics: https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-platform-logs
Kusto Query Language (KQL): https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/
Azure Monitor Alerts: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-overview

Question: 91 CertyIQ

You have an Azure DevOps organization named Contoso and an Azure subscription. The subscription contains an Azure virtual machine scale set named VMSS1 that is configured for autoscaling.
You use Azure DevOps to build a web app named App1 and deploy App1 to VMSS1. App1 is used heavily and has usage patterns that vary on a weekly basis.
You need to recommend a solution to detect an abnormal rise in the rate of failed requests to App1. The solution must minimize administrative effort.
What should you include in the recommendation?

A. the Smart Detection feature in Azure Application Insights
B. the Failures feature in Azure Application Insights
C. an Azure Service Health alert
D. an Azure Monitor alert that uses an Azure Log Analytics query

Answer: A

Explanation:

The correct answer is A, the Smart Detection feature in Azure Application Insights. Here's why:

Application Insights is designed for monitoring web applications, providing deep insights into performance and potential issues. Specifically, Smart Detection leverages machine learning algorithms to automatically detect anomalies, performance issues, and failures without requiring extensive manual configuration. In this scenario, the objective is to detect an abnormal rise in failed requests to App1. Smart Detection readily addresses this by learning the normal patterns of request failures and flagging significant deviations. This aligns with the requirement to minimize administrative effort, as Smart Detection operates automatically.

Option B, the Failures feature in Application Insights, provides a view of failed requests, but it doesn't proactively detect anomalies. It relies on manual investigation of failure data, which is not ideal when the goal is to automatically identify a surge in failures.

Option C, an Azure Service Health alert, focuses on informing about the health of Azure services and impacts to your resources, which is not directly about detecting abnormal request behavior within the application itself.

Option D, an Azure Monitor alert using an Azure Log Analytics query, can detect increases in failed requests, but requires the creation and maintenance of a custom query and alert rule. This introduces more administrative effort compared to Smart Detection's automated capabilities. Log Analytics is more for advanced querying of log data and building custom alerts based on very specific criteria. Smart Detection provides intelligent alerting on anomalies out-of-the-box.

In summary, Smart Detection is the best option because it proactively identifies abnormal rises in failed requests with minimal administrative overhead, aligning perfectly with the problem statement.

Further research:

Application Insights Smart Detection: https://learn.microsoft.com/en-us/azure/azure-monitor/app/proactive-diagnostics
Azure Application Insights Overview: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview

Question: 92

CertyIQ

SIMULATION -
You need to ensure that Microsoft Visual Studio 2017 can remotely attach to an Azure Function named fa-11566895.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:
Enable Remote Debugging -
Before we start a debugging session to our Azure Function app we need to enable the functionality.
1. Navigate in the Azure portal to your function app fa-11566895
2. Go to the Application settings
3. Under Debugging set Remote Debugging to On and set Remote Visual Studio version to 2017.

Reference:
https://www.locktar.nl/uncategorized/azure-remote-debugging-manually-in-visual-studio-2017/

Question: 93

CertyIQ

You have an Azure subscription that contains resources in several resource groups.
You need to design a monitoring strategy that will provide a consolidated view. The solution must support the following requirements:
- Support role-based access control (RBAC) by using Azure Active Directory (Azure AD) identifies.
- Include visuals from Azure Monitor that are generated by using the Kusto query language.
- Support documentation written in markdown.
- Use the latest data available for each visual.

What should you use to create the consolidated view?

A. Azure Monitor
B. Microsoft Power BI
C. Azure Data Explorer
D. Azure dashboards

Answer: D

Explanation:

The correct answer is **D. Azure dashboards**. Here's a detailed justification:

Azure dashboards are a central place in the Azure portal to create, save, and share customized views of resources and data across different resource groups. This directly addresses the requirement for a consolidated view.

Here's why Azure dashboards are a good fit and why the alternatives aren't:

RBAC Support: Azure dashboards seamlessly integrate with Azure AD and RBAC, allowing you to control access based on user roles and permissions. This ensures that only authorized personnel can view and modify the dashboard.

Kusto Query Language (KQL) Integration: Azure dashboards can embed Azure Monitor charts and visualizations built using KQL. This allows you to surface metrics and logs data directly within the consolidated view.

Markdown Support: Azure dashboards allow you to add markdown tiles. These tiles are editable in-place directly on the dashboard. This allows for inclusion of documentation related to metrics or service status.

Latest Data: Azure dashboards display the most up-to-date data available from Azure Monitor, ensuring that users always see the latest status and trends. The dashboard refreshes regularly.

Why the alternatives are not as suitable:

A. Azure Monitor: While Azure Monitor is crucial for collecting and analyzing telemetry data, it doesn't offer a consolidated, customizable view across multiple resource groups like Azure dashboards do. Azure Monitor feeds data into dashboards.

B. Microsoft Power BI: Power BI is a powerful business intelligence tool, but it's not natively integrated into the Azure portal for immediate Azure resource monitoring. Power BI typically requires more extensive data modeling and transformation. While Power BI can be used with Azure data, the native dashboard functionality in Azure is much easier and more suited to this task.

C. Azure Data Explorer: Azure Data Explorer is a data exploration and analytics service, mainly for big data. It's not primarily designed for building consolidated monitoring dashboards in the Azure portal itself, though it can feed dashboards with data.

In conclusion, Azure dashboards provide the best solution due to their native integration with Azure, support for RBAC, KQL visualizations, Markdown tiles, and real-time data, making them ideal for building a consolidated monitoring view that meets all requirements.

Authoritative Links:

Azure Dashboards Overview: https://learn.microsoft.com/en-us/azure/azure-portal/azure-portal-dashboards
Create and Share Azure Dashboards: https://learn.microsoft.com/en-us/azure/azure-portal/dashboard-create-shared-dashboards

Question: 94 CertyIQ

You are automating the testing process for your company.
You need to automate UI testing of a web application.
Which framework should you use?

A. JaCoco

B. Selenium

C. Xamarin.UITest

D. Microsoft.CodeAnalysis

Answer: B

Explanation:

The correct answer is B, Selenium. Here's a detailed justification:

Selenium is a widely used and open-source framework specifically designed for automating web browser interactions. This makes it ideal for automating UI testing of web applications. It supports multiple browsers (Chrome, Firefox, Safari, etc.) and programming languages (Java, Python, C#, JavaScript), offering flexibility in test automation strategies.

JaCoco (A) is a code coverage tool for Java applications, primarily focused on measuring the extent to which source code is executed during testing, not UI automation.

Xamarin.UITest (C) is a framework for UI testing native mobile applications (iOS and Android) built with Xamarin, not web applications. It's tailored for testing the UI of mobile apps.

Microsoft.CodeAnalysis (D), also known as Roslyn, is a set of open-source compilers and code analysis APIs for .NET. It's primarily used for building code analysis tools, refactorings, and diagnostics, not UI testing.

Selenium's focus on web browser automation directly addresses the requirement of automating UI testing of a web application. It simulates user actions within a browser, allowing testers to verify the functionality and appearance of web application components. Its cross-browser compatibility ensures that the application works correctly across different browser environments. Moreover, its active community and extensive documentation make it a robust choice for web UI testing automation projects within an Azure DevOps environment.

For more information on Selenium, refer to the official documentation:
https://www.selenium.dev/documentation/

Question: 95 CertyIQ

You are building an ASP.NET Core application.
You plan to create an application utilization baseline by capturing telemetry data.
You need to add code to the application to capture the telemetry data. The solution must minimize the costs of storing the telemetry data.
Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point

A. Add the <InitialSamplingPercentage>99</InitialSamplingPercentage> parameter to the ApplicationInsights.config file.

B. From the code of the application, enable adaptive sampling.

C. From the code of the application, add Azure Application Insights telemetry.

D. Add the <MaxTelemetryItemsPerSecond>5</MaxTelemetryItemsPerSecond> parameter to the ApplicationInsights.config file.

E. From the code of the application, disable adaptive sampling.

Answer: CE

Explanation:

The correct answer is CE. Here's why:

C: From the code of the application, add Azure Application Insights telemetry. Application Insights is the Azure service designed for collecting telemetry data from applications. To capture utilization data for creating a baseline, you must first integrate Application Insights into your ASP.NET Core application's code. This involves adding the necessary NuGet packages and initializing the Application Insights SDK in your application startup. Without this initial step, no telemetry will be captured at all.

E: From the code of the application, disable adaptive sampling. Adaptive sampling is designed to reduce the volume of telemetry data sent to Application Insights when your application produces a high volume of data. While adaptive sampling is generally a good practice for production environments to control costs, it can hinder the creation of an accurate utilization baseline. Disabling adaptive sampling ensures that all telemetry data is captured during the baseline period, providing a more complete and accurate representation of application behavior. By disabling adaptive sampling, you increase the data volume and therefore costs which seems contradictory to the question, but this is only for the time the baseline is being recorded. Once a baseline has been made, adaptive sampling can be re-enabled

Why other options are incorrect:

A and D: These options involve modifying the ApplicationInsights.config file, which is not the primary way to configure Application Insights in a modern ASP.NET Core application. Configuration is typically done through code or the appsettings.json file. Furthermore, adjusting initial sampling percentage or telemetry items per second without first integrating Application Insights won't have any effect. These options also enable sampling, contrary to needing to disable sampling for a complete baseline.
B and D: Adaptive sampling is the preferred sampling to use from a code perspective, rather than by configuring a specific rate using <MaxTelemetryItemsPerSecond>. However, for establishing a baseline, the aim is to capture all possible telemetry, thus not enabling adaptive sampling.

Justification Summary:

To create an accurate application utilization baseline using Application Insights, you must first integrate the service into your ASP.NET Core application (option C). To ensure that all telemetry data is captured for baseline establishment, you should disable adaptive sampling during the baseline recording period (option E). This will provide a more comprehensive picture of application usage.

Authoritative Links:

Application Insights for ASP.NET Core: https://learn.microsoft.com/en-us/azure/azure-monitor/app/asp-net-core
Sampling in Application Insights: https://learn.microsoft.com/en-us/azure/azure-monitor/app/sampling

Question: 96

CertyIQ

You have an Azure DevOps organization named Contoso and an Azure subscription. The subscription contains an Azure virtual machine scale set named VMSS1 and an Azure Standard Load Balancer named LB1. LB1 distributes incoming requests across VMSS1 instances.
You use Azure DevOps to build a web app named App1 and deploy App1 to VMSS1. App1 is accessible via HTTPS only and configured to require mutual authentication by using a client certificate.
You need to recommend a solution for implementing a health check of App1. The solution must meet the following requirements:
✑ Identify whether individual instances of VMSS1 are eligible for an upgrade operation.
✑ Minimize administrative effort.

What should you include in the recommendation?

A. an Azure Load Balancer health probe
B. Azure Monitor autoscale
C. the Custom Script Extension
D. the Application Health extension

Answer: D

Explanation:

The correct answer is **D. the Application Health extension**.

Here's a detailed justification:

The core requirement is to determine the health of individual VMSS1 instances specifically for upgrade eligibility, minimizing administrative overhead, and supporting a web application that requires mutual authentication using client certificates over HTTPS.

Why the Application Health Extension is the Best Choice:

Granular Health Reporting: The Application Health extension allows you to configure custom health probes that run directly within the VM instances themselves. This means you can tailor the probe to specifically check the health of App1, considering its HTTPS requirement and client certificate authentication.
Upgrade Awareness: The Application Health extension is designed to integrate seamlessly with VMSS upgrade policies. The health status reported by the extension is directly used by Azure to determine whether an instance is healthy enough to be included in an upgrade operation. Unhealthy instances are skipped, minimizing disruptions during upgrades.
Minimized Administrative Effort: Once configured, the extension runs automatically within the VMSS instances. You define the health probe (e.g., an HTTPS endpoint check with client certificate validation) and the extension handles the rest, continuously monitoring the application's health. No manual intervention is required for routine health checks.
HTTPS and Client Certificate Support: The Application Health extension enables the configuration of HTTPS probes, which are essential when the application is accessible only via HTTPS. Furthermore, because the probes run within the VM, it’s possible to configure the probe to utilize the client certificates required for mutual authentication when communicating with App1.

Why other options are not ideal:

A. Azure Load Balancer health probe: Load balancer health probes determine the availability of instances for routing incoming traffic. While useful for load balancing, they don't provide the granular health information needed to assess upgrade eligibility and may not directly support client certificate authentication in a simple manner. The focus is on traffic routing, not on detailed application health reporting within the VM for upgrade purposes.
B. Azure Monitor autoscale: Azure Monitor autoscale is used to scale the number of VM instances based on performance metrics. While related to the overall health and performance of the application, it doesn't directly address the need for individual instance health checks for upgrade eligibility.
C. Custom Script Extension: While the Custom Script Extension could be used to implement a health check, it requires significantly more administrative overhead. You would need to write and maintain a custom script, schedule its execution, manage its output, and integrate it with the VMSS upgrade process. Furthermore, this increases complexity and risks of potential errors, reducing the possibility of minimizing administrative effort.

In summary, the Application Health extension is the most suitable option due to its specific capabilities for monitoring application health within VM instances, reporting health status for upgrade operations, native support for HTTPS, and the possibility of integration of client certificate authentication in the configuration,

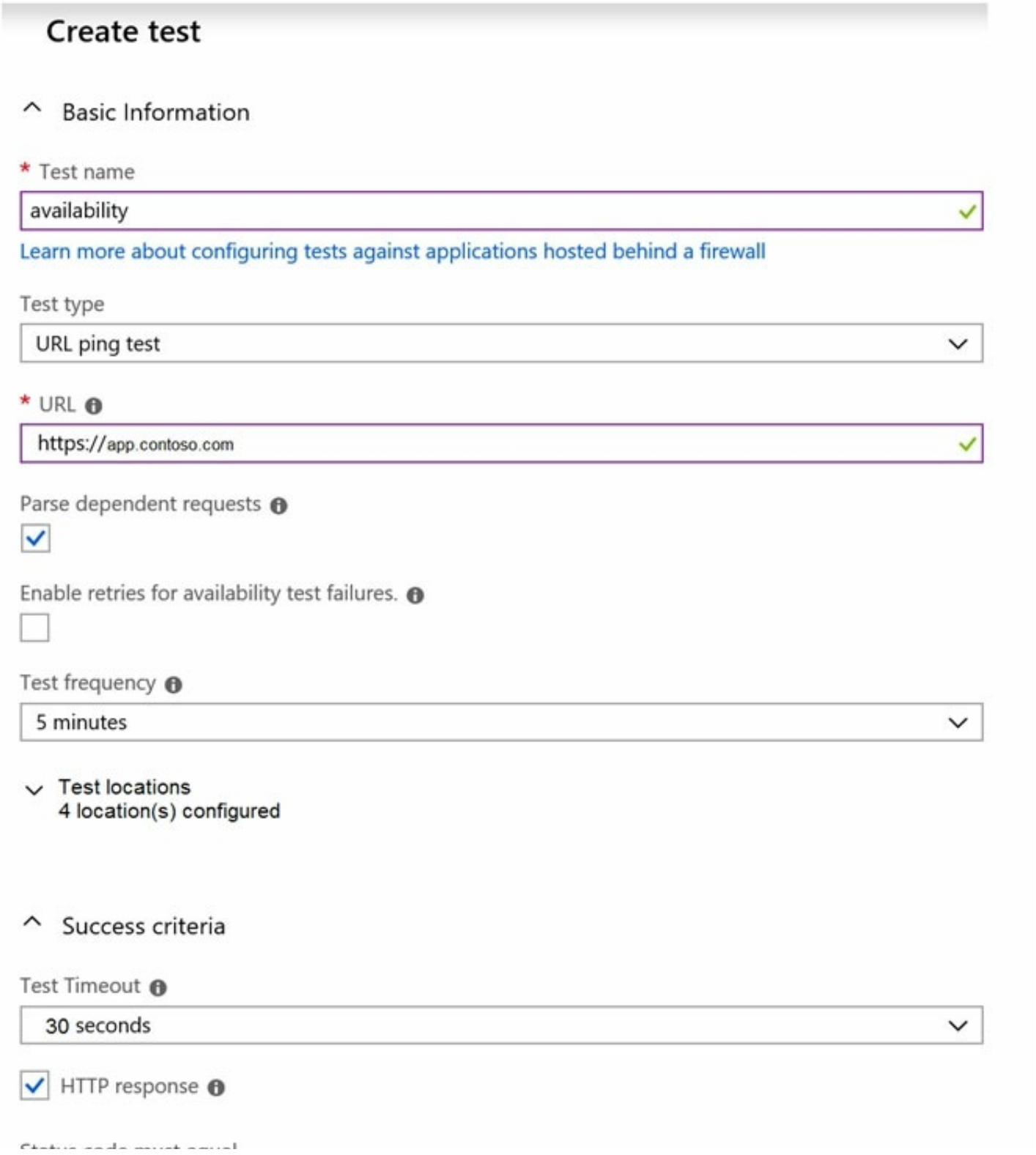
fulfilling all requirements with minimal administrative effort.

Supporting Documentation:

Application Health extension for Virtual Machine Scale Sets: https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-application-health
Azure Load Balancer health probes: https://learn.microsoft.com/en-us/azure/load-balancer/load-balancer-custom-probe-overview

Question: 97 CertyIQ

HOTSPOT -
You have an application named App1 that has a custom domain of app.contoso.com.
You create a test in Azure Application Insights as shown in the following exhibit.





Use the drop-down menus to select the answer choice that completes each statement based on the information presented in the graphic.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

The test will execute **[answer choice]**.

- every 30 seconds at a random location
- every 30 seconds per location
- every five minutes at a random location
- every five minutes per location

The test will pass if **[answer choice]** within 30 seconds.

- App1 responds to an ICMP ping
- the HTML of App1 and the HTML from URLs in <a> tags load
- all the HTML, JavaScripts, and images of App1 load

Answer:

Answer Area

The test will execute **[answer choice]**.

- every 30 seconds at a random location
- every 30 seconds per location
- every five minutes at a random location
- every five minutes per location

The test will pass if **[answer choice]** within 30 seconds.

- App1 responds to an ICMP ping
- the HTML of App1 and the HTML from URLs in <a> tags load
- all the HTML, JavaScripts, and images of App1 load

Explanation:

Box1: every five minutes per location.

confirmed that each location will be tested every minute.

Box2: All the HTML, JavaScripts, and images of App1 load.

If you selected Parse dependent requests, then all the images, style files, scripts, and other dependent resources must have been received within this period.

https://docs.microsoft.com/en-us/azure/azure-monitor/app/monitor-web-app-availability

Question: 98 CertyIQ

You have a build pipeline in Azure Pipelines that occasionally fails.
You discover that a test measuring the response time of an API endpoint causes the failures.
You need to prevent the build pipeline from failing due to the test.
Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Set Flaky test detection to Off.
B. Clear Flaky tests included in test pass percentage.
C. Enable Test Impact Analysis (TIA).
D. Manually mark the test as flaky.
E. Enable test slicing.

Answer: BD

Explanation:

The correct answer is BD. Here's why:

B. Clear Flaky tests included in test pass percentage: By default, Azure DevOps considers flaky tests when calculating the overall test pass percentage. If a flaky test fails, it can drag down the pass percentage and cause the build to fail based on pre-defined quality gates. Clearing "Flaky tests included in test pass percentage" prevents flaky tests from affecting the overall build health assessment. The build will then be considered a success even if the flaky test fails sometimes. This addresses the core requirement of preventing build pipeline failures caused by the test.

D. Manually mark the test as flaky: Marking a test as flaky informs Azure DevOps that the test's results are unreliable and might fluctuate independently of actual code changes. When a test is marked as flaky, Azure DevOps can handle it differently, such as rerunning the test automatically or providing separate reporting for flaky tests. By manually marking the test causing the problem as flaky, you acknowledge its unreliability and allow Azure DevOps to handle the test's failures in a way that doesn't halt the entire pipeline. This contributes towards making the build outcome more reflective of the code's actual state.

Why other options are not the best choices:

A. Set Flaky test detection to Off: Flaky test detection, if enabled, aims to identify flaky tests automatically. Turning it off wouldn't solve the problem of the identified flaky test causing build failures. It would simply prevent the system from trying to detect other potential flaky tests.

C. Enable Test Impact Analysis (TIA): TIA helps determine which tests are most important to run based on recent code changes. While potentially useful for optimizing test execution, it doesn't directly address the

issue of a specific flaky test causing build failures. TIA can help to prioritize test execution, but it won't prevent a flaky test from failing if it's still part of the selected tests.

E. Enable test slicing: Test slicing divides tests into smaller groups for parallel execution, aiming to reduce overall test time. Like TIA, this is more of a performance optimization and doesn't directly prevent a flaky test from failing and causing build failures. Even with test slicing, a single flaky test failure could still lead to pipeline failure.

Authoritative Links:

Work with flaky tests: https://learn.microsoft.com/en-us/azure/devops/pipelines/test/flaky-test-management?view=azure-devops

Question: 99 CertyIQ

Your company hosts a web application in Azure. The company uses Azure Pipelines for the build and release management of the application.
Stakeholders report that the past few releases have negatively affected system performance.
You configure alerts in Azure Monitor.
You need to ensure that new releases are only deployed to production if the releases meet defined performance baseline criteria in the staging environment first.
What should you use to prevent the deployment of releases that fall to meet the performance baseline?

A. an Azure Scheduler job
B. a trigger
C. a gate
D. an Azure function

Answer: C

Explanation:

The correct answer is **C. a gate**.

Gates in Azure Pipelines are used to control the deployment process by evaluating predefined conditions before a release proceeds to the next stage, specifically the production environment in this scenario. They act as quality control checkpoints that pause the pipeline execution until specified criteria are met, preventing faulty releases from impacting the production system.

By integrating performance baseline checks into a gate, you can programmatically assess whether a release candidate meets the required performance metrics in the staging environment. If the gate determines that the performance falls below the acceptable baseline, the deployment to production is automatically blocked.

Option A, Azure Scheduler, is for scheduled tasks and doesn't directly integrate with the pipeline to conditionally block deployment based on performance metrics. Option B, a trigger, initiates the pipeline but doesn't provide a mechanism for performance validation before deployment. Option D, Azure Functions, while powerful, would require additional orchestration logic within the pipeline to achieve the same outcome as gates, making it less efficient and adding complexity.

Gates can integrate with various services to perform checks, including Azure Monitor alerts, custom APIs returning health or performance data, or automated testing frameworks running in the staging environment. This makes gates the ideal choice to prevent releases that don't meet the performance baseline, helping maintain application stability and minimize negative impacts on end-users. They ensure that every deployment to production satisfies the pre-defined performance thresholds.

Refer to the Microsoft documentation for comprehensive information on gates in Azure Pipelines:https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devopshttps://learn.microsoft.com/en-us/azure/devops/pipelines/release/approvals/gates?view=azure-devops

Question: 100 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You manage a project in Azure DevOps.
You need to prevent the configuration of the project from changing over time.
Solution: Perform a Subscription Health scan when packages are created.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution, performing a Subscription Health scan when packages are created, does not meet the goal of preventing the configuration of an Azure DevOps project from changing over time. A Subscription Health scan is primarily concerned with the health and availability of Azure resources at the subscription level, identifying potential issues like quota limits, resource constraints, or service outages. It doesn't directly address changes to the configuration of an Azure DevOps project, such as modifications to build pipelines, release definitions, work item types, or security settings.

To prevent project configuration changes, a better approach would involve implementing measures like:

1. **Branch policies:** Protecting the main branch of your Azure DevOps repository and requiring code reviews for all changes.
2. **Role-based access control (RBAC):** Limiting who can make changes to the project configuration by assigning specific roles and permissions.
3. **Auditing:** Regularly reviewing audit logs to track changes made to the project.
4. **Infrastructure as Code (IaC):** Defining your Azure DevOps configuration as code (e.g., using Azure Resource Manager templates or Terraform) and managing it through source control.
5. **Configuration Management Tools:** Using tools like Chef, Puppet, or Ansible to automate the configuration and maintenance of your Azure DevOps environment.

Subscription Health scans are valuable for monitoring the overall health of your Azure subscription, but they are not designed to prevent or track changes to Azure DevOps project configurations.

https://learn.microsoft.com/en-us/azure/devops/organizations/security/about-security-roles?view=azure-devops

https://learn.microsoft.com/en-us/azure/devops/project/security/change-permissions?view=azure-devops&tabs=preview-page

Question: 101 CertyIQ

Your company uses the following resources:
- Windows Server 2019 container images hosted in an Azure Container Registry.
- Azure virtual machines that run the latest version of Ubuntu
- An Azure Log Analytics workspace
- Azure Active Directory (Azure AD)
- An Azure key vault

For which two resources can you receive vulnerability assessments in Azure Security Center? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. the Azure Log Analytics workspace

B. the Azure key vault

C. the Azure virtual machines that run the latest version of Ubuntu

D. Azure Active Directory (Azure AD)

E. The Windows Server 2019 container images hosted in the Azure Container Registry.

Answer: CE

Explanation:

The correct answer is CE because Azure Security Center provides vulnerability assessments for virtual machines and container registries.

C. Azure virtual machines that run the latest version of Ubuntu: Azure Security Center can assess the security posture of virtual machines, including identifying vulnerabilities in the operating system, installed software, and system configuration. The vulnerabilities are often detected via agent based solutions, scanning for known vulnerabilities based on CVEs. Ubuntu's latest version will still have software packages, configurations, and libraries that are tracked and have potential risks.

E. The Windows Server 2019 container images hosted in the Azure Container Registry: Azure Security Center can scan container images stored in Azure Container Registry for vulnerabilities. This helps identify potential security risks before deploying the containers to production. This includes scanning for vulnerabilities in the base image, installed packages, and application code. It integrates with tools like Qualys or Prisma Cloud Compute to perform these scans.

Options A, B, and D are incorrect because Azure Security Center does not directly provide vulnerability assessments for Azure Log Analytics workspaces, Azure Key Vault, or Azure Active Directory in the same way it does for VMs and container images. While Security Center integrates with these services, the vulnerability assessments are not performed directly on these services. For example, while Security Center will flag Key Vault settings that represent increased risk, it will not find software vulnerabilities on the Key Vault service itself. Likewise, it provides recommendations to secure Azure AD but doesn't provide vulnerability scans on the Azure AD service. Log Analytics workspace data is scanned, but the service itself isn't.

Here are authoritative links for further research:

Azure Security Center vulnerability assessment for VMs: https://learn.microsoft.com/en-us/azure/defender-for-cloud/deploy-vulnerability-assessment-vm
Azure Security Center container registry scanning: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-containers-registries-tasks

Question: 102 CertyIQ

You use Azure Pipelines to manage build pipelines, GitHub to store source code, and Dependabot to manage

dependencies.
You have an app named App1.
Dependabot detects a dependency in App1 that requires an update.
What should you do first to apply the update?

A. Create a pull request.
B. Approve the pull request.
C. Create a branch.
D. Perform a commit.

Answer: B

Explanation:

The correct answer is **B. Approve the pull request.**

Here's a detailed justification:

Dependabot automates dependency updates by creating pull requests (PRs) in your GitHub repository. When Dependabot detects a vulnerable or outdated dependency in App1, it automatically generates a PR with the necessary changes to update the dependency. The first step is NOT to create a pull request (A) as Dependabot does that already. Creating a branch (C) or performing a commit (D) is part of the automated process Dependabot initiates.

The first action you should take is to **review and approve the Dependabot-created pull request (B)**. Approving the PR signals that you've examined the changes and deem them safe to merge into your codebase. This step is crucial because automatically merging dependency updates without review can introduce unforeseen issues or break existing functionality. By approving the PR, you're essentially validating the update and authorizing its integration into your application. Approving the PR will initiate the merge into your main branch (typically).

Prior to approval, you should also review the pull request carefully. This review includes:

1. **Examining the changes:** Ensure the update aligns with your expectations and doesn't introduce breaking changes.
2. **Running tests:** Trigger your CI/CD pipeline in Azure Pipelines to automatically test the changes within the PR.
3. **Inspecting diffs:** Review the differences between the old and new dependency versions to understand the potential impact.

Only after thorough review and validation should you approve the Dependabot-created pull request.

Authoritative Links for Further Research:

About Dependabot version updates: https://docs.github.com/en/code-security/supply-chain-security/keeping-your-dependencies-updated-automatically/about-dependabot-version-updates
Configure Dependabot version updates: https://docs.github.com/en/code-security/supply-chain-security/keeping-your-dependencies-updated-automatically/configuration-options-for-dependency-updates

Question: 103 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You manage a project in Azure DevOps.
You need to prevent the configuration of the project from changing over time.
Solution: Add a code coverage step to the build pipelines.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of adding a code coverage step to the build pipelines does not meet the goal of preventing the configuration of the project from changing over time. Code coverage is a measure of the degree to which the source code of a program is tested. While valuable for software quality, it doesn't address the immutability or controlled evolution of the project's configuration itself. Project configuration in Azure DevOps includes settings like build definitions, release pipelines, variable groups, service connections, permissions, and work item customizations.

To prevent configuration changes over time, strategies such as infrastructure-as-code (IaC), where the Azure DevOps configuration is defined in code and version controlled, would be appropriate. Alternatively, you might implement tight access controls, rigorous change management processes, and use features like pipeline approvals and audit logging. The solution suggested offers no means of locking down the project's configuration details. Code coverage just indicates how well the code is tested, not whether the configuration has been altered.

Therefore, the solution is incorrect because it targets a completely different aspect (code quality) than the intended goal (configuration immutability).https://learn.microsoft.com/en-us/azure/devops/pipelines/concepts/infrastructure-as-code?view=azure-devopshttps://learn.microsoft.com/en-us/azure/devops/organizations/security/about-permissions?view=azure-devops

Question: 104 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You manage a project in Azure DevOps.
You need to prevent the configuration of the project from changing over time.
Solution: Implement Continuous Integration for the project.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of implementing Continuous Integration (CI) does **not** meet the goal of preventing the configuration of the Azure DevOps project from changing over time. CI focuses on automating the integration of code changes from multiple developers into a shared repository. It primarily concerns itself with building,

testing, and merging code, ensuring code quality and preventing integration issues.

Project configuration, on the other hand, refers to settings and definitions within Azure DevOps such as work item types, process templates, permissions, build and release pipelines (definitions, not execution), and service connections. CI does not inherently protect or manage these configuration aspects. Implementing CI is beneficial for software development lifecycle but does not equate to preventing changes to the Azure DevOps Project.

To prevent changes to the project configuration, you would typically employ practices such as:

1. **Infrastructure as Code (IaC):** Define your Azure DevOps project configuration as code (e.g., using YAML or JSON) and store it in a version control system. This allows you to track changes, revert to previous configurations, and automate the deployment of configurations.
2. **Process Templates:** Define standard processes and enforce them across projects to ensure consistency.
3. **Permissions Management:** Restrict access to configuration settings to authorized users only.
4. **Auditing:** Enable auditing to track changes made to the project configuration.
5. **Regular Backups:** Regularly back up your Azure DevOps project configuration to enable restoration in case of accidental or malicious changes.

Therefore, the suggestion to implement CI is unrelated and ineffective in achieving the stated goal of preventing project configuration changes.

Further research:

Infrastructure as Code (IaC): https://azure.microsoft.com/en-us/solutions/infrastructure-as-code/
Azure DevOps Documentation: https://docs.microsoft.com/en-us/azure/devops/

Question: 105 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You manage a project in Azure DevOps.
You need to prevent the configuration of the project from changing over time.
Solution: Implement Continuous Assurance for the project.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

The provided answer "Yes" is incorrect.

Continuous Assurance is a broad set of practices focusing on ensuring security and compliance are maintained throughout the software development lifecycle. While it includes elements that help maintain consistency and prevent configuration drift (e.g., infrastructure as code, automated testing, security scanning), it's not a direct, specific mechanism to prevent project configuration changes in Azure DevOps from drifting over time. Implementing Continuous Assurance provides a framework and set of controls, but does not directly lock down project configurations in Azure DevOps. It primarily focuses on maintaining security and compliance standards throughout the pipeline.

To prevent the configuration of an Azure DevOps project from changing over time, you would typically implement more specific measures such as:

1. **Infrastructure as Code (IaC):** Define the Azure DevOps project's configuration (e.g., build pipelines, release pipelines, variable groups, service connections) using code (e.g., YAML) and store it in a source control repository. Any changes to the configuration must then go through a pull request process and be reviewed.
2. **Policy Enforcement:** Use Azure Policy or similar tools to enforce specific configurations. While less direct for Azure DevOps itself, policies can constrain the Azure resources created and accessed by the pipelines, ensuring consistency.
3. **Role-Based Access Control (RBAC):** Restrict who has permissions to modify the project configuration. Grant users the minimum necessary permissions to perform their tasks.
4. **Auditing:** Enable auditing to track changes to the project configuration. This allows you to identify when changes were made and by whom.

Therefore, implementing Continuous Assurance alone does not directly meet the goal of preventing Azure DevOps project configuration changes over time. It is a helpful practice that contributes to overall system stability, security and compliance, but doesn't act as a locking or versioning mechanism for project configurations themselves.

Here are some authoritative links for further research:

Continuous Assurance: https://owasp.org/www-project-proactive-controls/v3/en/c9-implement-continuous-integration.html
Azure DevOps Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops
Infrastructure as Code: https://azure.microsoft.com/en-us/solutions/infrastructure-as-code/

Question: 106 CertyIQ

You are designing a configuration management solution to support five apps hosted on Azure App Service. Each app is available in the following three environments: development, test, and production.
You need to recommend a configuration management solution that meets the following requirements:
- Supports feature flags
- Tracks configuration changes from the past 30 days
- Stores hierarchically structured configuration values
- Controls access to the configurations by using role-based access control (RBAC) permissions
- Stores shared values as key/value pairs that can be used by all the apps

Which Azure service should you recommend as the configuration management solution?

A. Azure Cosmos DB
B. Azure App Service
C. Azure App Configuration
D. Azure Key Vault

Answer: C

Explanation:

The correct answer is **C. Azure App Configuration**. Here's a detailed justification:

Azure App Configuration is specifically designed for centralized configuration management, making it the ideal solution for this scenario. It natively supports hierarchically structured configuration values using namespaces and key delimiters, addressing the requirement for structured configurations. Feature flags are a built-in feature, simplifying their implementation and management. App Configuration stores configuration

changes historically, easily meeting the 30-day tracking requirement. It offers granular role-based access control (RBAC) through Azure Active Directory integration, securing access to sensitive configuration data.

Furthermore, App Configuration allows you to store shared key-value pairs that can be accessed by all five applications across all environments (development, test, and production). This enables consistent configuration across the applications. Options A, B and D don't meet all of the listed requirements. Azure Cosmos DB is a NoSQL database service, not a configuration management tool. Azure App Service itself lacks the comprehensive configuration management features like feature flags, RBAC, and historical tracking. Azure Key Vault is primarily for secrets management (passwords, keys, certificates), not general application configuration. Although Key Vault can store some configuration values, it's not designed for hierarchical configurations, feature flags, or the granular historical tracking required. App Configuration effectively centralizes and simplifies configuration management, reducing complexity and enhancing consistency across the application landscape.

Authoritative links for further research:

Azure App Configuration Documentation: https://learn.microsoft.com/en-us/azure/azure-app-configuration/
Feature Flags in App Configuration: https://learn.microsoft.com/en-us/azure/azure-app-configuration/concept-feature-flags
Security in App Configuration: https://learn.microsoft.com/en-us/azure/azure-app-configuration/concept-security

Question: 107 CertyIQ

You have a containerized solution that runs in Azure Container Instances. The solution contains a frontend container named App1 and a backend container named DB1. DB1 loads a large amount of data during startup. You need to verify that DB1 can handle incoming requests before users can submit requests to App1.
What should you configure?

A. a liveness probe
B. a performance log
C. a readiness probe
D. an Azure Load Balancer health probe

Answer: C

Explanation:

The correct answer is **C. a readiness probe**.

Here's why:

A readiness probe is designed specifically to determine when a container is ready to start accepting traffic. In this scenario, DB1 needs to load a significant amount of data before it can effectively respond to requests. A readiness probe allows Azure Container Instances (ACI) to know when DB1 has completed its startup process and is prepared to handle incoming connections from App1. This ensures that App1 doesn't start sending requests to DB1 prematurely, preventing potential errors or data corruption.

A **liveness probe (A)**, on the other hand, is used to detect if a container is still running. If a liveness probe fails, ACI will restart the container. While liveness is important, it's not the primary concern here, which is ensuring the database is ready before being accessed.

A **performance log (B)** collects data on the performance of the container, but it doesn't actively prevent traffic from being routed to an unready container. It's a diagnostic tool, not a traffic management mechanism.

An **Azure Load Balancer health probe (D)** is used to determine the health of virtual machines behind a load balancer. While similar in concept, using the Load Balancer here is not suitable, because ACI doesn't automatically use load balancers for intra-container communication within the same ACI group and it might be considered an overkill for a simple dependency.

Configuring a readiness probe will tell ACI not to send requests to DB1 until the readiness probe returns a success signal. This ensures that DB1 is fully operational before App1 starts sending requests, thus preventing issues.

Here are some resources for further reading:

Azure Container Instances health probes: https://learn.microsoft.com/en-us/azure/container-instances/container-instances-container-group
Kubernetes probes: https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/ (While this is Kubernetes-specific, the probe concepts are directly applicable to ACI)

Question: 108 CertyIQ

You are designing a strategy to monitor the baseline metrics of Azure virtual machines that run Windows Server.
You need to collect detailed data about the processes running in the guest operating system.
Which two agents should you deploy? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. the Telegraf agent
B. the Azure Log Analytics agent
C. the Azure Network Watcher Agent for Windows
D. the Dependency agent

Answer: BD

Explanation:

The correct answer is B and D. Here's why:

Azure Log Analytics agent: This agent is the cornerstone for collecting monitoring data from Azure VMs. It gathers data related to events, performance counters, and other logs. It sends this data to a Log Analytics workspace in Azure Monitor where it can be analyzed. For monitoring processes running in the guest OS, collecting relevant events and performance counters through the Log Analytics agent is crucial.

Dependency agent: The Dependency agent discovers application dependencies on Azure VMs, including processes and services running on the VM. This agent helps visualize and understand the relationships between different components of the application, which is essential for troubleshooting and performance analysis. It integrates with Service Map in Azure Monitor, providing a graphical view of these dependencies.

Telegraf agent (Incorrect): While Telegraf is a popular open-source agent for collecting metrics, it's not the primary agent for native integration with Azure Monitor for process-level data collection. Its configurations require more manual setup in Azure.

Azure Network Watcher Agent for Windows (Incorrect): This agent is focused on network monitoring, primarily used for tasks like capturing network traffic and diagnosing network connectivity issues, and doesn't directly provide detailed information about processes running inside the guest OS.

In summary, the Azure Log Analytics agent collects a broad range of OS-level logs and metrics, and the Dependency agent complements it by providing application dependency insights, including process information. Together, these agents provide the detailed data required for monitoring processes running on

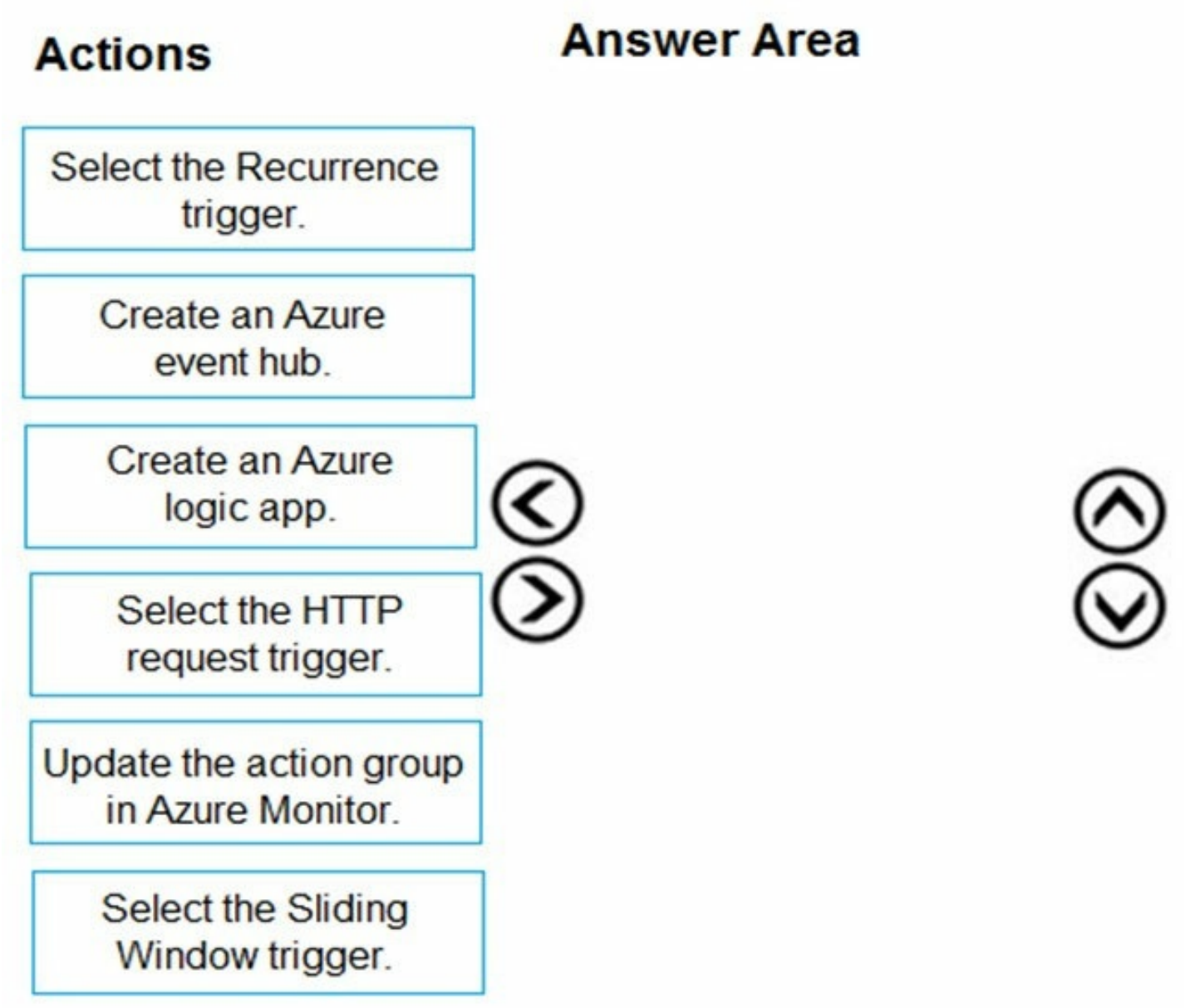
Azure VMs.

Supporting Documentation:

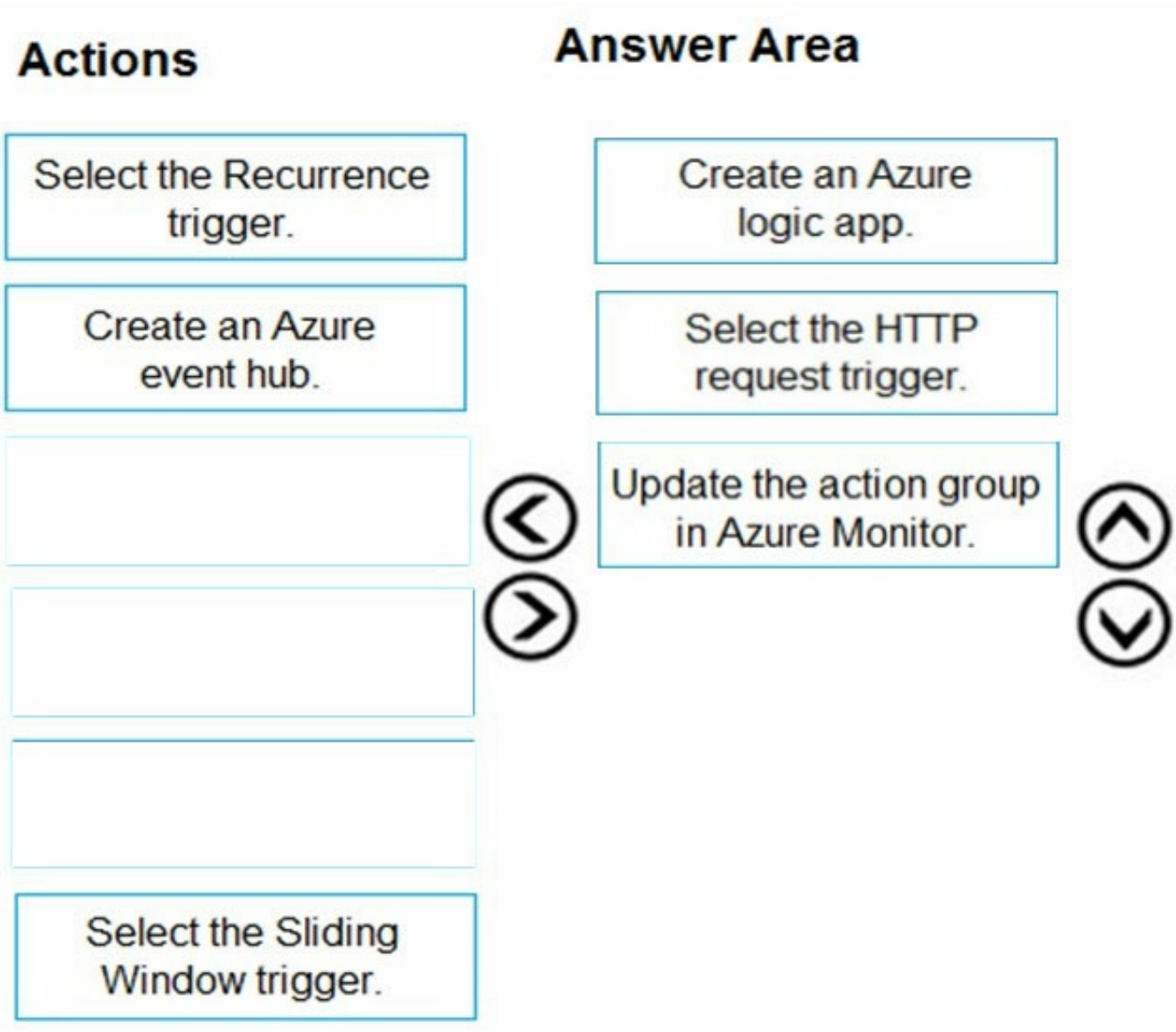
Azure Monitor overview: https://learn.microsoft.com/en-us/azure/azure-monitor/overview
Log Analytics agent overview: https://learn.microsoft.com/en-us/azure/azure-monitor/agents/log-analytics-agent
Azure Monitor Dependency agent: https://learn.microsoft.com/en-us/azure/azure-monitor/vm/vminsights-dependency-agent

Question: 109 CertyIQ

DRAG DROP -
You use Azure Pipelines to automate Continuous Integration/Continuous Deployment (CI/CD) for an Azure web app named WebApp1.
You configure an Azure Monitor alert that is triggered when WebApp1 generates an error.
You need to configure the alert to forward details of the error to a third-party system. The solution must minimize administrative effort.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Box 1: Create an Azure logic app.

Box 2: Select the HTTP request trigger.

Box 3: Updated the action group in Azure Monitor.

Reference:

https://docs.microsoft.com/en-us/azure/azure-monitor/alerts/action-groups-logic-app

Question: 110 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You have an Azure DevOps organization named Contoso and an Azure subscription. The subscription contains an Azure virtual machine scale set named VMSS1 that is configured for autoscaling.

You have a project in Azure DevOps named Project1. Project1 is used to build a web app named App1 and deploy App1 to VMSS1.

You need to ensure that an email alert is generated whenever VMSS1 scales in or out.

Solution: From Azure DevOps, configure the Notifications settings for Project1.

Does this meet the goal?

A. Yes

B. No

Answer: B

Explanation:

The proposed solution suggests configuring Azure DevOps Notifications for Project1 to address the alert requirement for VMSS1 scaling events. However, Azure DevOps notifications primarily focus on events within Azure DevOps, such as build failures, work item changes, or code check-ins. They do not directly monitor or react to changes in Azure resources like virtual machine scale sets.

While you can integrate Azure DevOps with Azure resources, the native notification system within Azure DevOps isn't the correct tool for monitoring VMSS scaling events. VMSS scaling is an Azure resource event, and alerts for these events should be configured within Azure itself.

A more appropriate approach would involve using Azure Monitor alerts. Azure Monitor allows you to create alerts based on various metrics and events related to Azure resources. In this case, you can configure an alert to trigger when the instance count of VMSS1 changes, which indicates a scale-in or scale-out operation. The alert can then be configured to send an email notification. Alternatively, you could utilize Azure Activity Log alerts to specifically monitor the autoscaling events.

Therefore, configuring notifications in Azure DevOps will not achieve the desired goal of alerting on VMSS scaling events, thus making the proposed solution incorrect.

Azure Monitor Alerts DocumentationAzure Activity Log Alerts

Question: 111 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure DevOps organization named Contoso and an Azure subscription. The subscription contains an Azure virtual machine scale set named VMSS1 that is configured for autoscaling.
You have a project in Azure DevOps named Project1. Project1 is used to build a web app named App1 and deploy App1 to VMSS1.
You need to ensure that an email alert is generated whenever VMSS1 scales in or out.
Solution: From Azure DevOps, configure the Service hooks settings for Project1.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The provided solution, configuring Service hooks in Azure DevOps for Project1, will not directly achieve the goal of generating email alerts when VMSS1 (Azure virtual machine scale set) scales in or out.

Service hooks in Azure DevOps are primarily designed to trigger actions in external services based on events within Azure DevOps, such as code commits, build completions, or work item changes. While you can potentially trigger a build or release pipeline in Azure DevOps based on an external event, the events for VMSS scaling are not directly available as triggers for Azure DevOps service hooks.

To achieve the desired outcome, the appropriate approach would involve leveraging Azure Monitor alerts. Azure Monitor allows you to monitor various Azure resources, including VMSS, and create alerts based on specific metrics or events. You can configure an alert rule in Azure Monitor that triggers when the instance count of VMSS1 changes (which signifies a scale-in or scale-out operation). When the alert is triggered, you can configure an action group to send an email notification.

Therefore, configuring service hooks in Azure DevOps is not the correct method for monitoring VMSS scaling events and sending email alerts. Azure Monitor alerts are the correct approach.

Justification Links:

Azure Monitor Alerts: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-overview
Azure Monitor Action Groups: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/action-groups
Azure DevOps Service Hooks: https://learn.microsoft.com/en-us/azure/devops/service-hooks/overview?view=azure-devops

Question: 112 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure DevOps organization named Contoso and an Azure subscription. The subscription contains an Azure virtual machine scale set named VMSS1 that is configured for autoscaling.
You have a project in Azure DevOps named Project1. Project1 is used to build a web app named App1 and deploy App1 to VMSS1.
You need to ensure that an email alert is generated whenever VMSS1 scales in or out.
Solution: From Azure Monitor, create an action group.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

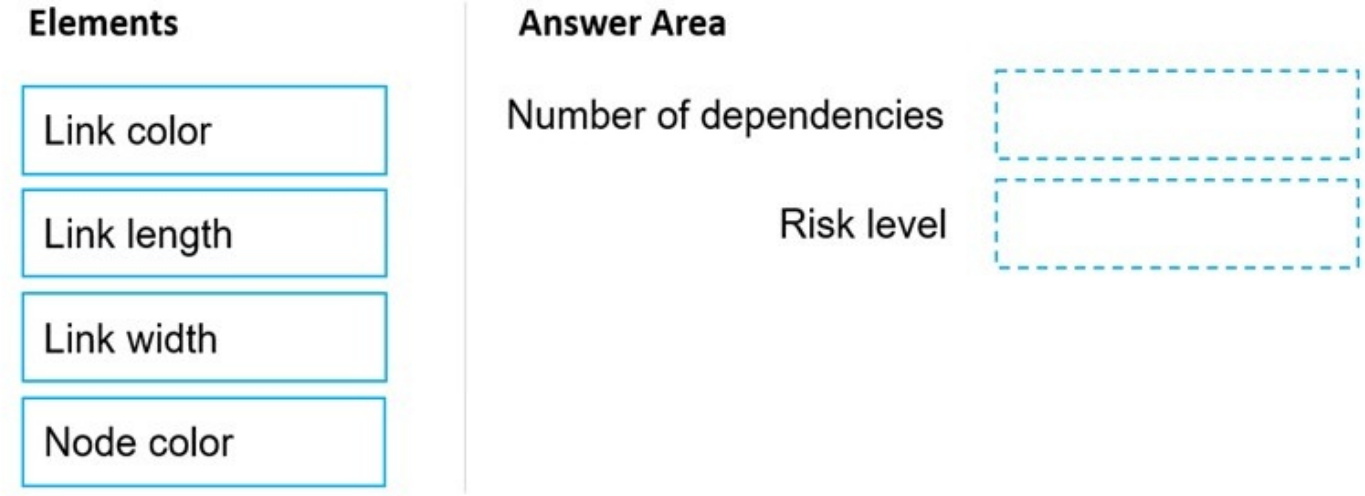
Yes, the proposed solution meets the goal. Creating an action group in Azure Monitor is a fundamental step in configuring alerts that trigger specific actions, such as sending email notifications. In this scenario, by setting up an action group, you can define the desired notification method (email) when an alert is triggered related to VMSS1 autoscaling events.Azure Monitor is a comprehensive monitoring service that collects telemetry from various Azure resources, including VMSS1. These telemetry data points can be used to set up alert rules based on specific conditions. VMSS1 autoscaling events (scale-in and scale-out) can be monitored, and alerts can be configured to trigger when these events occur.

By configuring an alert rule in Azure Monitor to monitor the autoscaling events of VMSS1 and associating the alert rule with the created action group, an email will be sent to the specified recipients whenever VMSS1 scales in or out. This setup allows for timely notification and proactive management of the virtual machine scale set.

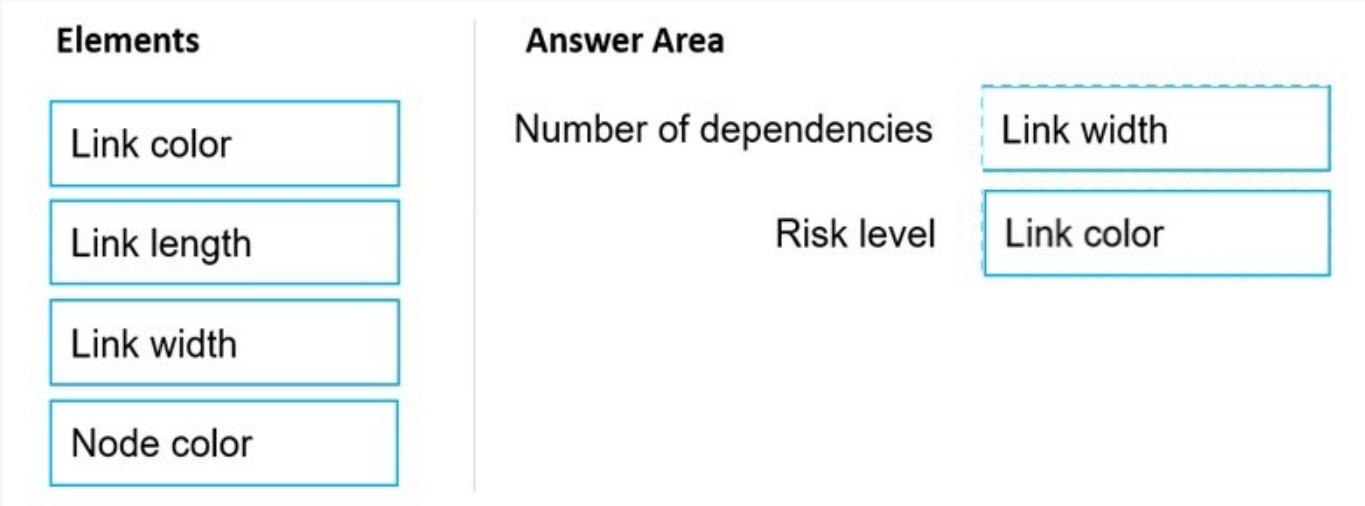
For more information on action groups, refer to the official Microsoft documentation: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/action-groupsTo learn more about Azure Monitor alerts, refer to: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-overviewTo learn more about VMSS autoscaling, refer to: https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-autoscale-overview

Question: 113 CertyIQ

DRAG DROP -
You are using the Dependency Tracker extension in a project in Azure DevOps.
You generate a risk graph for the project.
What should you use in the risk graph to identify the number of dependencies and the risk level of the project? To answer, drag the appropriate elements to the correct data points. Each element may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:
Box 1: Link width -
The width of the lines indicates how many dependencies exist in that area, the thicker the link the more dependencies as indicated in the legend.

Box 2: Link color -

Reference:
https://docs.microsoft.com/en-us/azure/devops/boards/extensions/dependency-tracker?view=azure-devops#risk-graph

Question: 114 CertyIQ

You have an Azure subscription linked to an Azure Active Directory Premium Plan 1 tenant.
A security review indicates that too many users have privileged access to resources.
You need to deploy a privileged access management solution that meets the following requirements:
• Enforces time limits on the use of privileged access
• Requires approval to activate privileged access
• Minimizes costs
What should you do first?

A. Configure notifications when privileged roles are activated.
B. Configure alerts for the activation of privileged roles.
C. Enforce Azure Multi-Factor Authentication (MFA) for role activation.
D. Upgrade the license of the Azure Active Directory (Azure AD) tenant.

Answer: D

Explanation:

The correct answer is D: Upgrade the license of the Azure Active Directory (Azure AD) tenant.

Here's why: The core requirement is to implement Privileged Access Management (PAM) with features like time limits, approval workflows, and cost minimization. Azure AD Premium P1 does not natively include the Azure AD Privileged Identity Management (PIM) features required for these specific capabilities. PIM is available with Azure AD Premium P2. The other options (A, B, and C) assume PIM is already in place and functional, which is not the case with only a P1 license. Upgrading to P2 unlocks PIM, enabling features like Just-In-Time (JIT) access, requiring approval for role activation, and setting time-bound access durations.

Therefore, the initial step is to enable the underlying infrastructure that supports PAM. Without upgrading to Azure AD Premium P2, implementing the desired time limits and approval requirements for privileged access is impossible. While MFA is a good security practice (option C), it doesn't address the core requirement of needing PIM functionality. Similarly, configuring notifications (A) or alerts (B) is useful after PIM is set up but are not prerequisites for enabling the core PAM features. Minimizing costs is factored in by choosing the appropriate licensing level for the desired PAM functionality. Trying to implement it without the correct license would ultimately be more costly through workarounds or inadequate security posture.

Authoritative links for further research:

What is Azure AD Privileged Identity Management?: https://learn.microsoft.com/en-us/azure/active-directory/privileged-identity-management/pim-configure
Compare generally available features of the Free, Microsoft 365 Apps, Premium P1, and Premium P2 editions of Azure Active Directory: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis#compare-azure-ad-editions

Question: 115 CertyIQ

You have a GitHub Enterprise account.
You need to enable push protection for secret scanning of the account repositories.
What should you do first?

A. Purchase a GitHub Advanced Security license.
B. Purchase Premium Plus support.
C. Enforce multi-factor authentication (MFA).

D. Create an access policy for secrets.

Answer: A

Explanation:

The correct answer is **A. Purchase a GitHub Advanced Security license.**

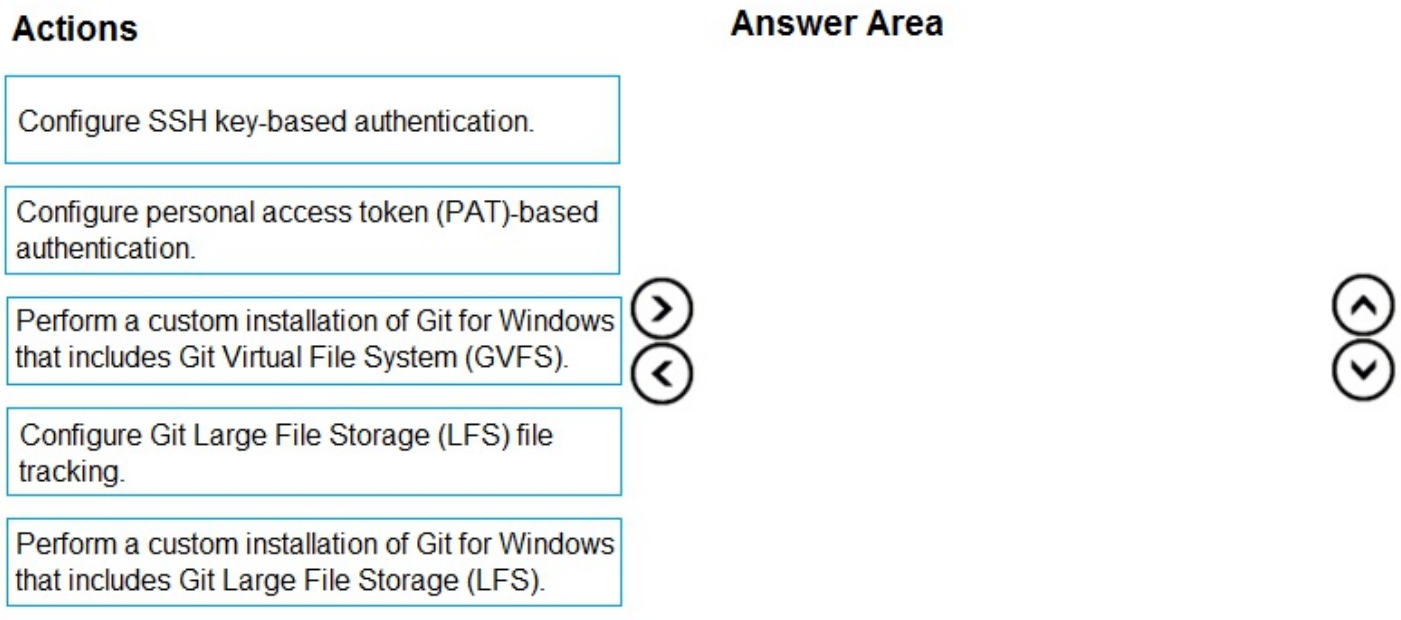
GitHub Advanced Security provides features like secret scanning, dependency review, and code scanning, which are essential for push protection to prevent secrets from being committed to repositories. Push protection leverages secret scanning capabilities to detect exposed secrets during the git push operation. Without GitHub Advanced Security, the basic secret scanning feature only alerts on secrets already committed. To actively block the commit of secrets during the push, you need the enhanced features. Purchasing a license unlocks these advanced security measures. Premium Plus support, while beneficial for general support needs, doesn't directly enable push protection for secret scanning. MFA enhances authentication security but is independent of secret scanning functionality. Creating an access policy might control access to certain data, but it doesn't activate or enable the secret scanning features required for push protection. Therefore, the first step is to enable the necessary functionality through licensing to use secret scanning for push protection.

For further details, refer to these resources:

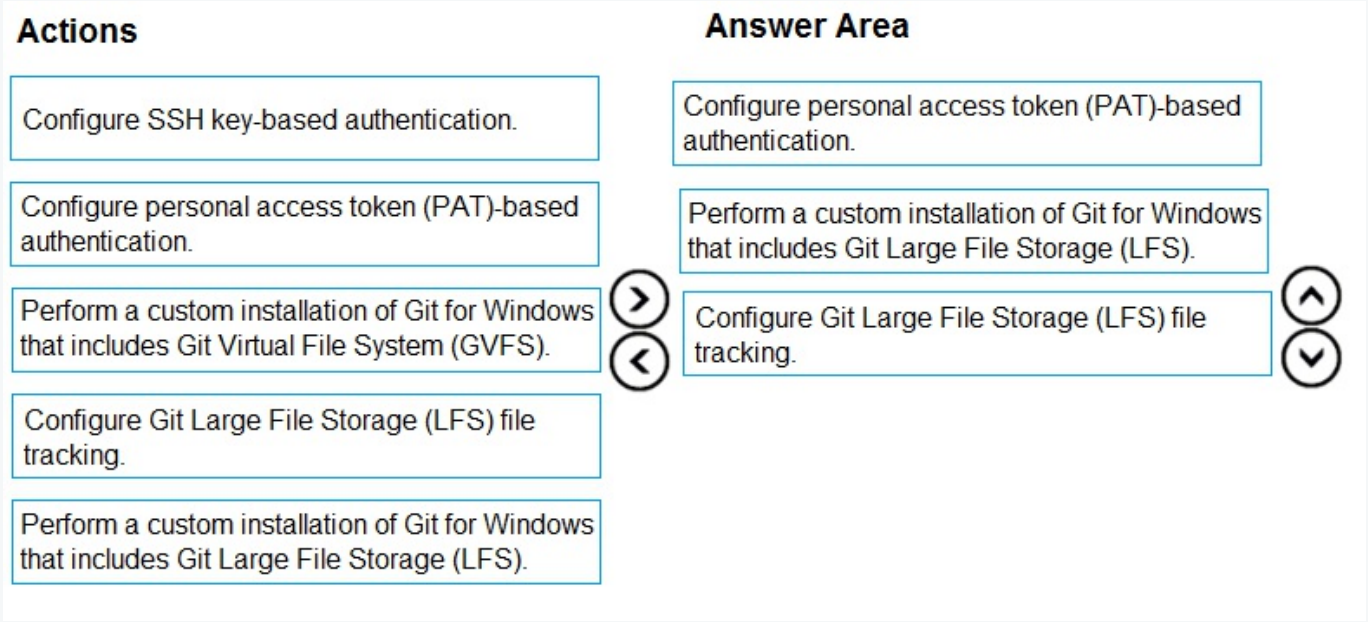
About push protection for secrets: https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/push-protection-for-secrets
GitHub Advanced Security overview: https://docs.github.com/en/enterprise-cloud@latest/get-started/learning-about-github/about-github-advanced-security

Question: 116 CertyIQ

DRAG DROP -
Your company has a project in Azure DevOps named Project1.
All the developers at the company have Windows 10 devices.
You need to create a Git repository for Project1. The solution must meet the following requirements:
• Support large binary files.
• Store binary files outside of the repository.
• Use a standard Git workflow to maintain the metadata of the binary files by using commits to the repository.
Which three actions should you perform in sequence on each developer's device? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



Answer:



Explanation:

Configure PAT

Perform installation of Git LFS

Track GIT LFS

https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-git-large-file-storage

Question: 117 CertyIQ

You have an Azure DevOps project that contains a build pipeline. The build pipeline uses approximately 50 open source libraries.
You need to ensure that all the open source libraries comply with your company's licensing standards.
Which service should you use?

A. Ansible
B. Maven
C. WhiteSource Bolt
D. Helm

Answer: C

Explanation:

The correct answer is **C. WhiteSource Bolt**.

WhiteSource Bolt is a security and license compliance management tool integrated directly into Azure DevOps. It automatically detects and identifies open-source components used in your project, including their licenses. This allows you to enforce your company's licensing policies by identifying components with licenses that are not approved. It provides reports on license risks, security vulnerabilities, and outdated components within your dependencies.

Ansible (A) is an automation engine used for configuration management, application deployment, and task automation. While it's powerful, it doesn't directly address open-source license compliance.

Maven (B) is a build automation tool primarily used for Java projects. It manages dependencies but does not inherently provide features for license compliance. While Maven can be configured with plugins to achieve some degree of license checking, it's not its primary function, and a dedicated tool like WhiteSource Bolt is more effective.

Helm (D) is a package manager for Kubernetes. It simplifies deployment and management of applications in Kubernetes clusters. It has nothing to do with license compliance in your Azure DevOps project.

Therefore, WhiteSource Bolt is specifically designed for identifying and managing open-source components and their licenses within your Azure DevOps pipelines, making it the best choice for enforcing your company's licensing standards. It goes beyond simple dependency management and provides valuable insights into license risks associated with using open-source libraries.

For further reading, refer to the official documentation on WhiteSource Bolt integration with Azure DevOps:

Mend Bolt (formerly WhiteSource Bolt): https://marketplace.visualstudio.com/items?itemName=whitesource.whitesource-bolt

Mend documentation: https://www.mend.io/ (search for Azure DevOps integration)

Question: 118 CertyIQ

Your company is building a new solution in Java.
The company currently uses a SonarQube server to analyze the code of .NET solutions.
You need to analyze and monitor the code quality of the Java solution.
Which task types should you add to the build pipeline?

A. Chef
B. Gradle
C. Octopus
D. xCODE

Answer: B

Explanation:

The correct task type to add to the build pipeline for analyzing Java code quality with SonarQube is **B. Gradle**. Here's why:

SonarQube is a platform widely used for continuous inspection of code quality. To analyze Java projects, SonarQube needs to integrate with a build tool that manages dependencies and executes the compilation process. Gradle is a popular and flexible build automation tool used extensively in the Java ecosystem. It provides a mechanism to manage dependencies, compile code, run tests, and package applications.

By adding a Gradle task to the Azure DevOps build pipeline, you can configure it to execute the SonarQube scanner plugin. The SonarQube scanner then analyzes the source code according to the quality rules defined in the SonarQube server. This integration enables developers to obtain immediate feedback on the code quality during the CI/CD process. The SonarQube plugin then sends the analysis results to the SonarQube server, which aggregates and presents the data.

Chef (A) is a configuration management tool, suitable for automating infrastructure configuration but

irrelevant for code analysis. Octopus Deploy (C) is a deployment automation tool, which handles the release process, not code analysis. Xcode (D) is an integrated development environment specifically for Apple platforms and doesn't support Java projects. Therefore, Gradle is the only choice tailored for Java build automation and integration with SonarQube for code analysis.

Here are some links for further research:

SonarQube: https://www.sonarsource.com/
Gradle: https://gradle.org/
Azure DevOps: https://azure.microsoft.com/en-us/services/devops/
SonarQube Scanner for Gradle: https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner-for-gradle/

Question: 119 CertyIQ

Your company is concerned that when developers introduce open source libraries, it creates licensing compliance issues.
You need to add an automated process to the build pipeline to detect when common open source libraries are added to the code base.
What should you use?

A. Microsoft Visual SourceSafe
B. PDM
C. WhiteSource
D. OWASP ZAP

Answer: C

Explanation:

The correct answer is C, WhiteSource. Here's why:

WhiteSource (now Mend) is specifically designed to manage open source security and license compliance. It integrates seamlessly with build pipelines to automatically detect open source components, identify their licenses, and flag potential vulnerabilities. This automated process helps organizations proactively address licensing issues and security risks before they become problems.

Here's a detailed justification against the other options:

A. Microsoft Visual SourceSafe: Visual SourceSafe (VSS) is a deprecated source control system. It is not related to license compliance or open source library detection. It's solely a tool for managing code versions.

B. PDM (Product Data Management): PDM systems primarily focus on managing product-related data, including engineering designs, specifications, and manufacturing information. While they might track software components, they typically don't have built-in features for open source license compliance and vulnerability scanning.

D. OWASP ZAP (Zed Attack Proxy): OWASP ZAP is a web application security scanner, primarily used for penetration testing and finding vulnerabilities in web applications. It doesn't focus on open source license compliance or the detection of open source libraries within a codebase.

WhiteSource (Mend), on the other hand, is specifically tailored for this purpose. It analyzes the project's dependencies, identifies open source components used, maps them to their respective licenses (e.g., MIT, Apache 2.0, GPL), and alerts the team to potential licensing conflicts or vulnerabilities associated with those

libraries. This integration within the build pipeline allows for continuous monitoring and enforcement of open source policies. This reduces the risk of legal issues related to software licensing and helps improve the overall security posture of the organization. It also provides detailed reports and dashboards for visibility and tracking of open source components and their associated risks.

Authoritative Links for Further Research:

Mend (formerly WhiteSource): https://www.mend.io/ - The official website of the tool.
Open Source License Compliance: https://www.synopsys.com/glossary/what-is-open-source-license-compliance.html - A definition of open-source license compliance.

Question: 120 CertyIQ

Your company is concerned that when developers introduce open source libraries, it creates licensing compliance issues.
You need to add an automated process to the build pipeline to detect when common open source libraries are added to the code base.
What should you use?

A. Microsoft Visual SourceSafe
B. Code Style
C. Black Duck
D. Jenkins
E. SourceGear
F. OWASP ZAP

Answer: C

Explanation:

The correct answer is **C. Black Duck**.

Black Duck is a Software Composition Analysis (SCA) tool specifically designed to identify and manage open source components within a codebase. It automatically scans builds and identifies the open source libraries being used, their versions, and associated licenses. This capability addresses the concern of licensing compliance issues by providing visibility into the open source dependencies being introduced by developers.

The tool generates reports detailing potential license conflicts, vulnerabilities, and other risks associated with the identified open source components. This allows organizations to proactively address compliance issues, ensure adherence to licensing terms, and mitigate security risks. Integration with build pipelines enables automated scanning as part of the development process, ensuring continuous monitoring and early detection of potential problems.

Options A, B, D, E, and F are incorrect because they do not directly address open source license compliance. Visual SourceSafe, SourceGear, and Jenkins are source control and automation tools. Code Style is about code formatting, and OWASP ZAP is a web application security scanner focused on runtime vulnerabilities, not license management. Black Duck's focus on SCA makes it the optimal solution for this problem.

For further research, refer to the official Black Duck documentation: https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html

Question: 121 CertyIQ

You have an Azure DevOps project that contains a build pipeline. The build pipeline uses approximately 50 open source libraries.
You need to ensure that all the open source libraries comply with your company's licensing standards.
Which service should you use?

A. NuGet
B. Maven
C. Black Duck
D. Helm

Answer: C

Explanation:

The correct answer is **C. Black Duck**. Here's a detailed justification:

Black Duck is a Software Composition Analysis (SCA) tool specifically designed to manage the risks associated with open-source software. It provides a comprehensive solution for identifying open-source components in your codebase, including their versions, licenses, and known vulnerabilities. This is crucial for compliance, security, and legal risk management within an organization.

Here's why the other options are less suitable:

A. NuGet: NuGet is a package manager primarily for .NET projects. While it manages dependencies, it doesn't provide deep analysis of open-source licenses or security vulnerabilities. It mainly focuses on resolving and installing packages.

B. Maven: Maven is a build automation tool and dependency management system primarily used for Java projects. Similar to NuGet, it helps manage dependencies but lacks robust license compliance and vulnerability analysis features.

D. Helm: Helm is a package manager for Kubernetes. It is used to deploy and manage applications on Kubernetes clusters, and doesn't directly deal with the licensing or vulnerability assessment of open-source libraries used in the application's codebase.

Black Duck excels in the following ways to address the stated need:

1. **Open Source Discovery:** It automatically identifies open-source components within your build pipeline.
2. **License Identification:** It accurately determines the licenses associated with each open-source component, allowing you to verify compliance with company standards.
3. **Vulnerability Scanning:** It identifies known security vulnerabilities in open-source components, helping you mitigate potential risks.
4. **Policy Enforcement:** It enables you to define policies based on license types and vulnerability levels, automatically flagging any violations.
5. **Reporting:** It provides comprehensive reports on open-source usage, license compliance, and security vulnerabilities, facilitating informed decision-making.
6. **Remediation Guidance:** Black Duck offers advice on remediation actions, such as upgrading to a newer version or replacing a component.

By using Black Duck, you can ensure that all open-source libraries used in your Azure DevOps project comply with your company's licensing standards, reducing the risk of legal issues and security vulnerabilities.

Authoritative Links for Further Research:

Synopsys (Black Duck): https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html
OWASP (Software Composition Analysis): https://owasp.org/www-project-top-ten/ (Refer to A09:2021 – Software and Data Integrity Failures)

Question: 122 CertyIQ

Your company uses Azure DevOps for the build pipelines and deployment pipelines of Java-based projects.
You need to recommend a strategy for managing technical debt.
Which action should you include in the recommendation?

A. Configure post-deployment approvals in the deployment pipeline.
B. Integrate Azure DevOps and SonarQube.
C. Integrate Azure DevOps and Azure DevTest Labs.

Answer: B

Explanation:

The correct answer is B: Integrate Azure DevOps and SonarQube. SonarQube is a widely recognized platform for continuous inspection of code quality, allowing developers to detect bugs, code smells, and security vulnerabilities. Integrating SonarQube with Azure DevOps enables automatic analysis of code as part of the build pipeline. This automated analysis provides crucial feedback on technical debt, like maintainability issues and potential bugs, early in the development lifecycle. Developers can then address these issues proactively before they accumulate and become more costly to fix. By making code quality analysis part of the build process, SonarQube helps enforce coding standards and promotes a culture of writing cleaner, more maintainable code, thereby reducing technical debt. Integrating it with Azure DevOps enables actionable reports within the Azure DevOps interface, fostering collaboration and facilitating remediation efforts. Option A (post-deployment approvals) controls releases, but it doesn't directly identify or manage technical debt. Option C (Azure DevTest Labs) provides environments for development and testing but doesn't inherently assess code quality or technical debt. SonarQube integration gives teams quantifiable metrics for tracking and reducing debt.

Relevant resources for further reading:

SonarQube: https://www.sonarsource.com/products/sonarqube/
Azure DevOps: https://azure.microsoft.com/en-us/services/devops/
SonarQube Azure DevOps Integration: https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarqube

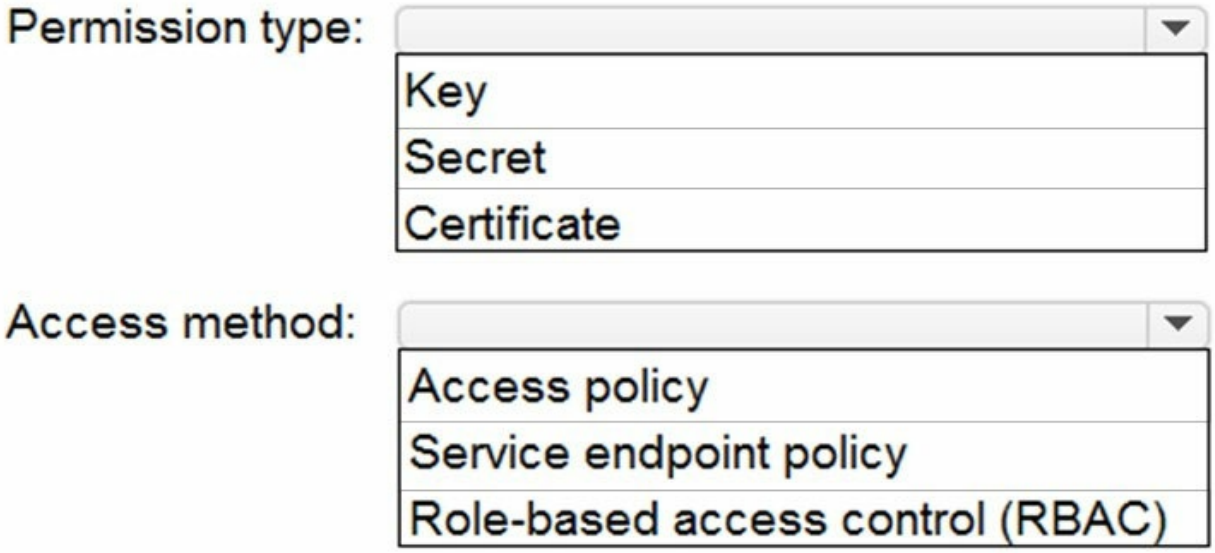
Question: 123 CertyIQ

HOTSPOT -
You have an Azure subscription that contains the resources shown in the following table.

Name	Type
DF1	Azure Data Factory
SQL1	Azure SQL Database
KV1	Azure Key Vault

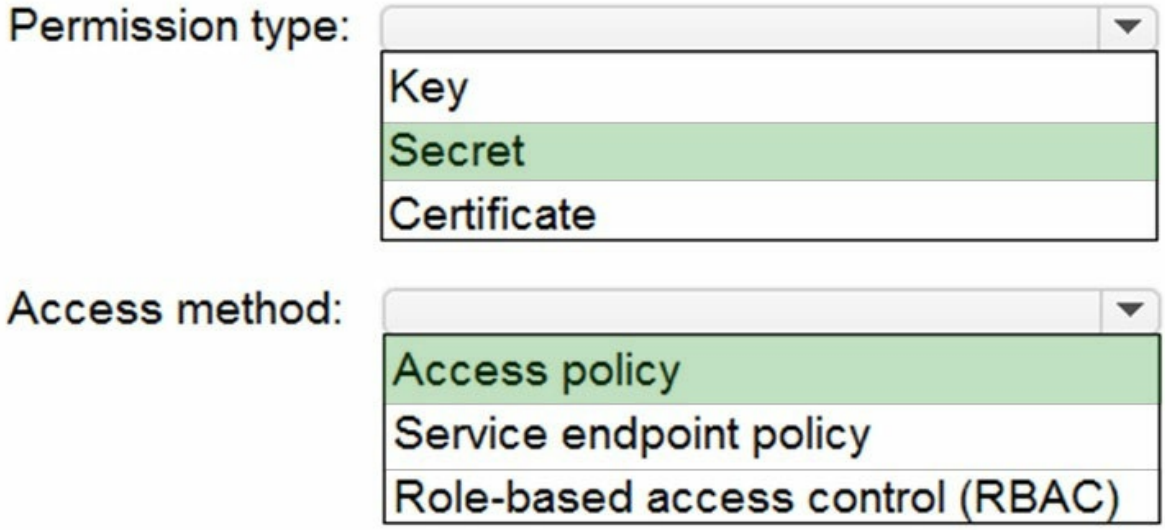
You plan to create a linked service in DF1. The linked service will connect to SQL1 by using Microsoft SQL Server authentication. The password for the SQL
Server login will be stored -
in KV1.
You need to configure DF1 to retrieve the password when the data factory connects to SQL1. The solution must use the principle of least privilege.
How should you configure DF1? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:
Box 1: Secret -
Store credential in Azure Key Vault by reference secret stored in key vault.
To reference a credential stored in Azure Key Vault, you need to:
1. Retrieve data factory managed identity
2. Grant the managed identity access to your Azure Key Vault. In your key vault -> Access policies -> Add Access Policy, search this managed identity to grant
Get permission in Secret permissions dropdown. It allows this designated factory to access secret in key vault.
3. Create a linked service pointing to your Azure Key Vault.
4. Create data store linked service, inside which reference the corresponding secret stored in key vault.

Box 2: Access policy -

Reference:
https://docs.microsoft.com/en-us/azure/data-factory/store-credentials-in-key-vault

Question: 124 CertyIQ

You have several Azure Active Directory (Azure AD) accounts.
You need to ensure that users use multi-factor authentication (MFA) to access Azure apps from untrusted networks.
What should you configure in Azure AD?

A. access reviews
B. managed identities
C. entitlement management
D. conditional access

Answer: D

Explanation:

The correct answer is **D. Conditional Access**.

Conditional Access in Azure AD is the tool specifically designed to enforce access controls based on conditions. These conditions can include user identity, location, device, application, and real-time risk detections. In this scenario, the requirement is to enforce MFA for Azure apps when accessed from untrusted networks. Conditional Access policies can be configured to identify untrusted networks (e.g., based on IP ranges or geographic locations not associated with the organization's trusted locations) and then require MFA as a condition for granting access. This ensures that users accessing resources from outside the trusted corporate network are subject to stronger authentication measures.

Options A, B, and C are not suitable for this requirement.

Access Reviews (A) are used to regularly review user access rights and permissions, ensuring that users only have the necessary access. While important for security, they don't enforce MFA based on network location.
Managed Identities (B) are used to provide Azure resources with an identity in Azure AD, simplifying the process of authenticating to other Azure services without embedding credentials in code. This isn't related to enforcing MFA for users.
Entitlement Management (C) is used to manage access to groups, applications, and SharePoint Online sites, providing a way to automate access request workflows, access assignments, reviews, and expirations. While relevant to access control, it does not directly handle conditional MFA based on network location.

Therefore, Conditional Access provides the necessary capabilities to meet the specified requirement of enforcing MFA for Azure apps accessed from untrusted networks.

Authoritative Links:

What is Conditional Access in Azure Active Directory?
Common Conditional Access policies

Question: 125 CertyIQ

You plan to provision a self-hosted Linux agent.
Which authentication mechanism should you use to register the self-hosted agent?

A. personal access token (PAT)
B. SSH key
C. Alternate credentials
D. certificate

Answer: A

Explanation:

The correct authentication mechanism for registering a self-hosted Linux agent with Azure DevOps is a Personal Access Token (PAT). Here's why:

PATs provide a secure and controlled way for applications or users to access Azure DevOps resources. They act as an identity substitute, granting permissions to specific scopes and resources without exposing your primary credentials (like your Azure Active Directory password).

SSH keys are typically used for secure shell access to the server itself, not specifically for registering an agent with the Azure DevOps organization. While you could use SSH to manage the server hosting the agent,

it's a separate concern from authenticating the agent itself.

Alternate credentials (e.g., username/password) are generally discouraged for automated processes due to security risks. Storing and managing these credentials securely becomes complex and prone to vulnerabilities.

Certificates are primarily used for TLS/SSL encryption, securing communication channels. While Azure DevOps uses certificates, they aren't the primary authentication mechanism for agent registration.

The self-hosted agent registration process requires authentication with the Azure DevOps organization. A PAT is generated within the Azure DevOps organization, granting the agent permission to connect and receive jobs. This allows fine-grained control over the agent's access. The PAT can be configured to expire, further enhancing security. It's the recommended and most straightforward method for agent registration as documented by Microsoft.

Therefore, using a PAT is the most secure and manageable way to authenticate the self-hosted agent during registration.

Relevant Documentation:

Agents - Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser (Review the "Authentication" sections)
Use Personal Access Tokens (PATs): https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops&tabs=preview

Question: 126 CertyIQ

You are building a Microsoft ASP.NET application that requires authentication.
You need to authenticate users by using Azure Active Directory (Azure AD).
What should you do first?

A. Assign an enterprise application to users and groups
B. Create an app registration in Azure AD
C. Configure the application to use a SAML endpoint
D. Create a new OAuth token from the application
E. Create a membership database in an Azure SQL database

Answer: B

Explanation:

The correct answer is **B. Create an app registration in Azure AD**.

Here's a detailed justification:

Before an application can interact with Azure Active Directory (Azure AD) to authenticate users or access resources, it must be registered. App registration is the process of creating an identity for your application within Azure AD, granting it permission to access specific Azure resources or other applications on behalf of users. This registration provides the application with a unique application (client) ID, which it uses to identify itself when requesting tokens.

Why other options are incorrect:

A. Assign an enterprise application to users and groups: An enterprise application represents an existing

application within your Azure AD tenant, typically a SaaS application like Salesforce or Workday. Assigning it to users and groups grants them access to that specific application, not your custom ASP.NET application.
C. Configure the application to use a SAML endpoint: SAML is one authentication protocol, but Azure AD supports other protocols, including OAuth 2.0 and OpenID Connect, which are more commonly used for modern web applications. While SAML could be used, creating an app registration is still the fundamental first step.
D. Create a new OAuth token from the application: You cannot create OAuth tokens directly from the application before it has been registered. The application needs to be registered so that Azure AD can grant it access and issue tokens based on its identity.
E. Create a membership database in an Azure SQL database: While you might choose to manage additional user information in a separate database, this is not the first step to integrate with Azure AD. Azure AD handles the primary authentication process, and you can use the information provided by Azure AD (claims) to supplement your application's needs.

In summary, creating an app registration in Azure AD is the foundational step that sets up your ASP.NET application's identity and enables it to authenticate users through Azure AD, regardless of the authentication protocol ultimately used.

Here are authoritative links for further research:

Application and service principal objects in Azure Active Directory: https://learn.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals
Register an application with the Microsoft identity platform: https://learn.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app

Question: 127 CertyIQ

You have an Azure DevOps organization named Contoso.
You need to recommend an authentication mechanism that meets the following requirements:
- Supports authentication from Git
- Minimizes the need to provide credentials during authentication

What should you recommend?

A. personal access tokens (PATs) in Azure DevOps
B. Alternate credentials in Azure DevOps
C. user accounts in Azure Active Directory (Azure AD)
D. managed identities in Azure Active Directory (Azure AD)

Answer: A

Explanation:

The correct answer is A: personal access tokens (PATs) in Azure DevOps. Here's a detailed justification:

PATs in Azure DevOps allow developers to authenticate with Azure DevOps services, including Git repositories, without constantly entering their primary credentials (like their Azure AD username and password). They are specifically designed to provide secure authentication for various tools and scripts, including Git, without requiring full user credentials.

PATs can be scoped with specific permissions, ensuring that the tool or script only has access to the resources it needs. This reduces the risk associated with compromised credentials compared to using full user accounts. The expiration date can also be set, further limiting the window of opportunity for misuse. When using Git, storing the PAT in the Git credential manager allows for seamless authentication without repeated prompts. While requiring initial creation, PATs offer the best balance between security and usability for Git

authentication scenarios within Azure DevOps.

Alternate credentials (B) are generally discouraged in favor of PATs. They are considered less secure and less flexible. User accounts in Azure Active Directory (Azure AD) (C), while important for overall user access, necessitate the repeated entry of credentials (username/password, potentially MFA) when interacting with Git, which contradicts the requirement to minimize the need to provide credentials during authentication. Managed identities (D) in Azure AD are designed for Azure resources to authenticate to other Azure resources. They are not directly applicable to authenticating a user using Git against Azure DevOps. They are more suitable for services hosted on Azure infrastructure.

Therefore, PATs provide the most appropriate solution as they directly support Git authentication and minimize the need for repeated credential entry, satisfying both requirements outlined in the prompt.

Supporting documentation:

Use personal access tokens (PATs) to authenticate - Azure DevOps
Git Credential Manager Core

Question: 128

CertyIQ

You have an application that consists of several Azure App Service web apps and Azure functions.
You need to assess the security of the web apps and the functions.
Which Azure feature can you use to provide a recommendation for the security of the application?

A. Security & Compliance in Azure Log Analytics
B. Resource health in Azure Service Health
C. Smart Detection in Azure Application Insights
D. Compute & apps in Azure Security Center

Answer: D

Explanation:

The correct answer is D, Compute & apps in Azure Security Center, because Azure Security Center is designed to provide a unified security management system that enhances the security posture of your Azure resources, including App Service web apps and Azure Functions. Specifically, the "Compute & apps" section within Security Center focuses on the security of your compute resources (like web apps and functions).

Security Center continuously assesses these resources for security vulnerabilities and misconfigurations, using built-in and customizable security policies. Based on these assessments, Security Center provides actionable recommendations to improve the security of your application. These recommendations can include steps such as enabling authentication, configuring network security groups, applying security patches, and configuring secure protocols.

In contrast:

A. Security & Compliance in Azure Log Analytics: While Log Analytics can collect security-related logs, it doesn't inherently provide proactive security recommendations in the same way as Security Center. Log Analytics is more about collecting and analyzing data, rather than proactively suggesting security improvements.
B. Resource Health in Azure Service Health: Resource Health focuses on the health and availability of your Azure services, not specifically on security vulnerabilities or misconfigurations. While resource health can indirectly relate to security (e.g., an unhealthy service might be more vulnerable), it's not a primary tool for assessing security posture and providing security recommendations.

C. Smart Detection in Azure Application Insights: Application Insights detects performance anomalies and failures. Although it can detect security-related anomalies, its primary focus is not security recommendations for apps and compute resources.

Therefore, Azure Security Center's "Compute & apps" section is the most appropriate feature for assessing the security of Azure App Service web apps and Azure Functions and providing recommendations to improve their security.

Supporting Links:

Azure Security Center: https://learn.microsoft.com/en-us/azure/security-center/security-center-introduction
Secure your PaaS deployment: https://learn.microsoft.com/en-us/azure/security/fundamentals/paas

Question: 129 CertyIQ

Your company has a project in Azure DevOps for a new web application.
The company identifies security as one of the highest priorities.
You need to recommend a solution to minimize the likelihood that infrastructure credentials will be leaked.
What should you recommend?

A. Add a Run Inline Azure PowerShell task to the pipeline.
B. Add a PowerShell task to the pipeline and run Set-AzureKeyVaultSecret.
C. Add an Azure Key Vault task to the pipeline.
D. Add Azure Key Vault references to Azure Resource Manger templates.

Answer: D

Explanation:

The correct answer is D: Add Azure Key Vault references to Azure Resource Manager (ARM) templates. Here's why:

Minimizing credential leakage is crucial in cloud deployments. Storing sensitive information directly in code (like PowerShell scripts or inline tasks) is a high-risk practice. Azure Key Vault provides a secure, centralized location for storing secrets, keys, and certificates.

Option D leverages Key Vault's integration with ARM templates. Instead of directly embedding credentials in the ARM template (which defines your infrastructure as code), you reference secrets stored in Key Vault. During deployment, Azure Resource Manager retrieves the secrets from Key Vault and uses them to configure the resources. This ensures that sensitive information is never directly present in the deployment template or the pipeline's execution logs, greatly reducing the risk of exposure.

Option A, using a Run Inline Azure PowerShell task, introduces the risk of hardcoding credentials directly within the script. While potentially automating the setup, it creates a significant security vulnerability.

Option B, using the Set-AzureKeyVaultSecret cmdlet, is more related to storing secrets into Key Vault, not using them during infrastructure deployment, which the question focuses on. Also, this script itself could contain sensitive information if not handled carefully.

Option C, adding an Azure Key Vault task to the pipeline, while seemingly relevant, is often used to retrieve secrets for use within the pipeline itself, not for injecting them directly into ARM template deployments. It's a useful step if you have a build process that needs secure data, but it doesn't directly solve the problem of exposing credentials within the ARM template itself.

Using Key Vault references in ARM templates promotes separation of concerns: infrastructure definition

(ARM template) and secrets management (Key Vault). This separation enhances security, simplifies auditing, and enables better secret rotation policies. Crucially, it avoids embedding sensitive data in the deployment process.

For further research, refer to the official Microsoft documentation on Azure Key Vault and ARM template integration:

Use Azure Key Vault to pass secure parameter value during Resource Manager deployment:
https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/key-vault-parameter
Azure Key Vault documentation: https://learn.microsoft.com/en-us/azure/key-vault/

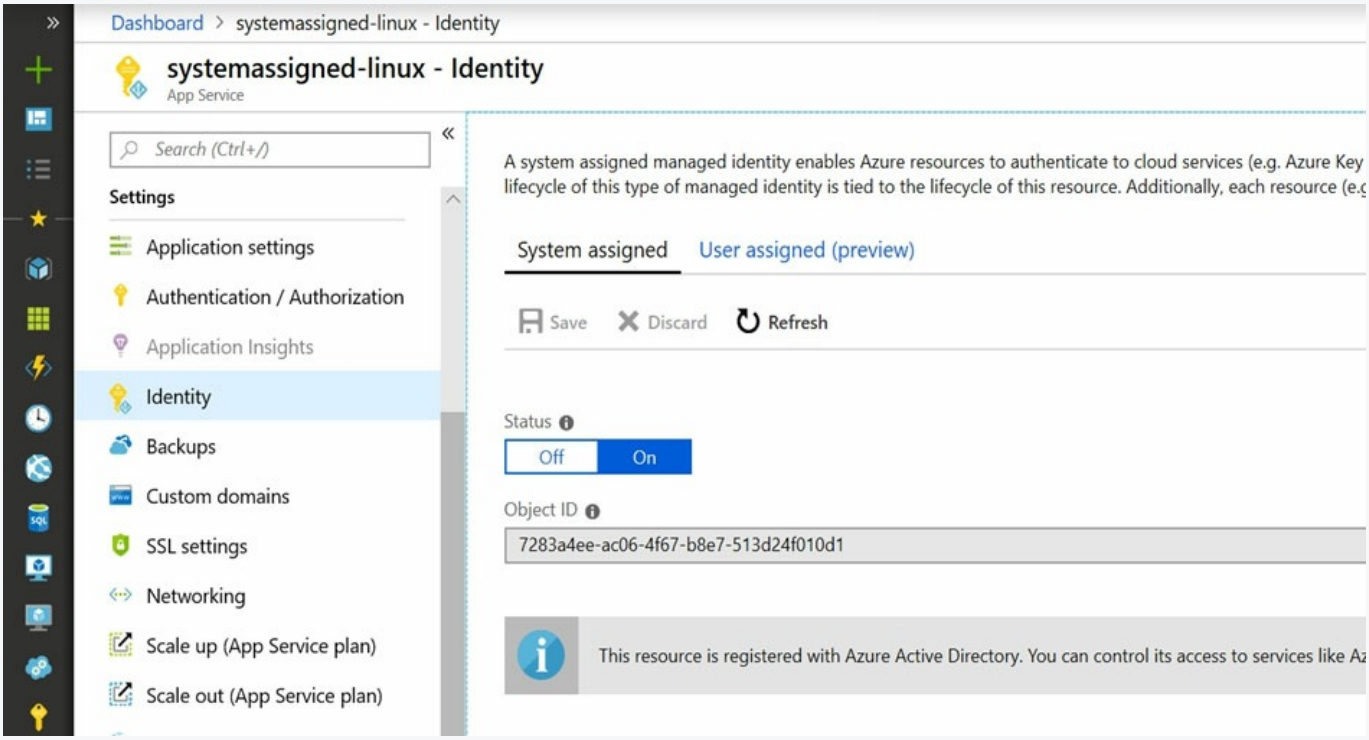
Question: 130 CertyIQ

SIMULATION -
You need to ensure that an Azure web app named az400-123456789-main can retrieve secrets from an Azure key vault named az400-123456789-kv1 by using a system managed identity.
The solution must use the principle of least privilege.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:
1. In Azure portal navigate to the az400-123456789-main app.
2. Scroll down to the Settings group in the left navigation.
3. Select Managed identity.
4. Within the System assigned tab, switch Status to On. Click Save.



Reference:
https://docs.microsoft.com/en-us/azure/app-service/overview-managed-identity

Question: 131 CertyIQ

You create a Microsoft ASP.NET Core application.

You plan to use Azure Key Vault to provide secrets to the application as configuration data.
You need to create a Key Vault access policy to assign secret permissions to the application. The solution must use the principle of least privilege.
Which secret permissions should you use?

A. List only
B. Get only
C. Get and List

Answer: B

Explanation:

The correct answer is **B. Get only**.

Justification:

Azure Key Vault implements role-based access control, and adhering to the principle of least privilege means granting only the necessary permissions to access resources. In this scenario, the ASP.NET Core application needs to retrieve secrets stored in the Key Vault to use them as configuration data.

The "Get" permission allows the application to retrieve the value of a specific secret. This is sufficient for the application to access and use the secrets in its configuration. The "List" permission, on the other hand, allows the application to list all secrets within the Key Vault. This is unnecessary and violates the principle of least privilege, as the application only needs to know and access specific secrets, not all of them.

Therefore, assigning only the "Get" permission ensures that the application can retrieve the secrets it needs while preventing it from gaining unnecessary access to other secrets or metadata within the Key Vault. This reduces the potential attack surface and improves the overall security posture.

Authoritative Links:

Microsoft Documentation on Key Vault Access Policies: https://learn.microsoft.com/en-us/azure/key-vault/general/assign-access-policy?tabs=azure-portal
Microsoft Documentation on Key Vault Permissions: https://learn.microsoft.com/en-us/azure/key-vault/general/about-keys-secrets-certificates

Question: 132 CertyIQ

DRAG DROP -
Your company has a project in Azure DevOps.
You plan to create a release pipeline that will deploy resources by using Azure Resource Manager templates. The templates will reference secrets stored in Azure
Key Vault.
You need to recommend a solution for accessing the secrets stored in the key vault during deployments. The solution must use the principle of least privilege.
What should you include in the recommendation? To answer, drag the appropriate configurations to the correct targets. Each configuration may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:

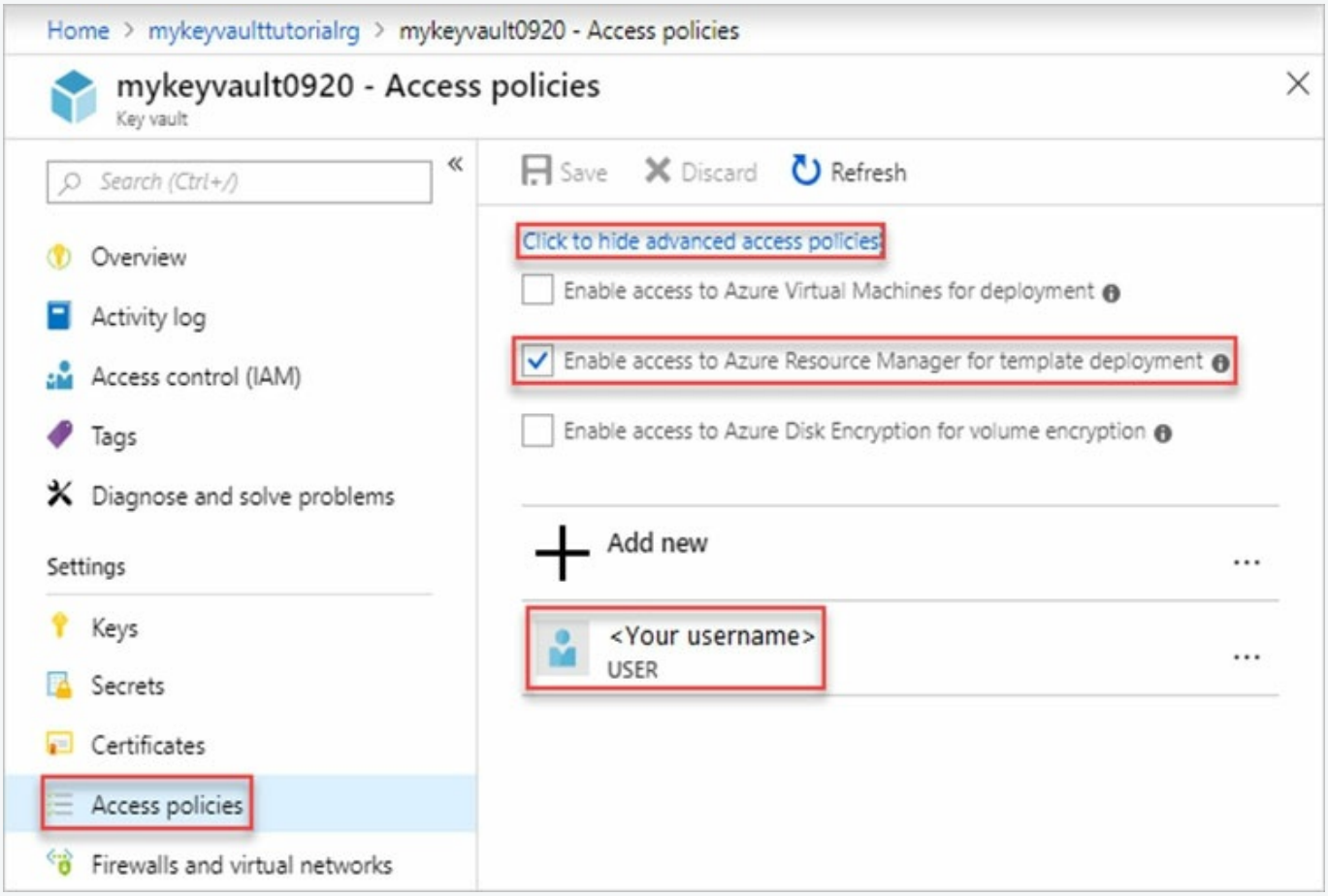
Configurations	Answer Area	
A Key Vault access policy	Enable key vaults for template deployment by using:	
A Key Vault advanced access policy	Restrict access to the secrets in Key Vault by using:	
RBAC		

Answer:

Configurations	Answer Area	
A Key Vault access policy	Enable key vaults for template deployment by using:	A Key Vault advanced access policy
A Key Vault advanced access policy	Restrict access to the secrets in Key Vault by using:	RBAC
RBAC		

Explanation:

Box 1: A key Vault advanced access policy



Box 2: RBAC -

Management plane access control uses RBAC.

The management plane consists of operations that affect the key vault itself, such as:

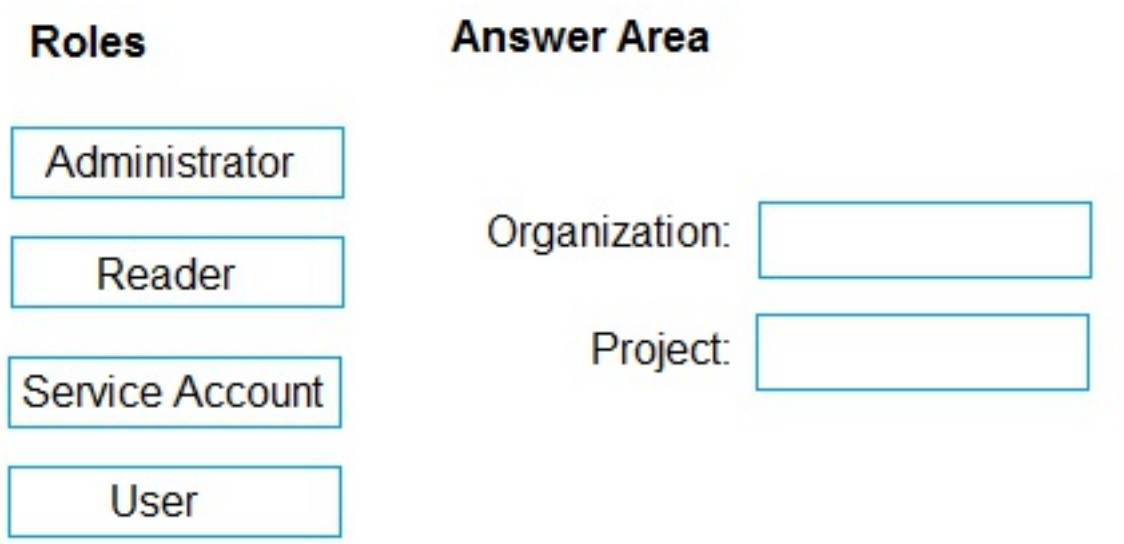
- Creating or deleting a key vault.
- Getting a list of vaults in a subscription.
- Retrieving Key Vault properties (such as SKU and tags).
- Setting Key Vault access policies that control user and application access to keys and secrets.

Reference:
https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-tutorial-use-key-vault

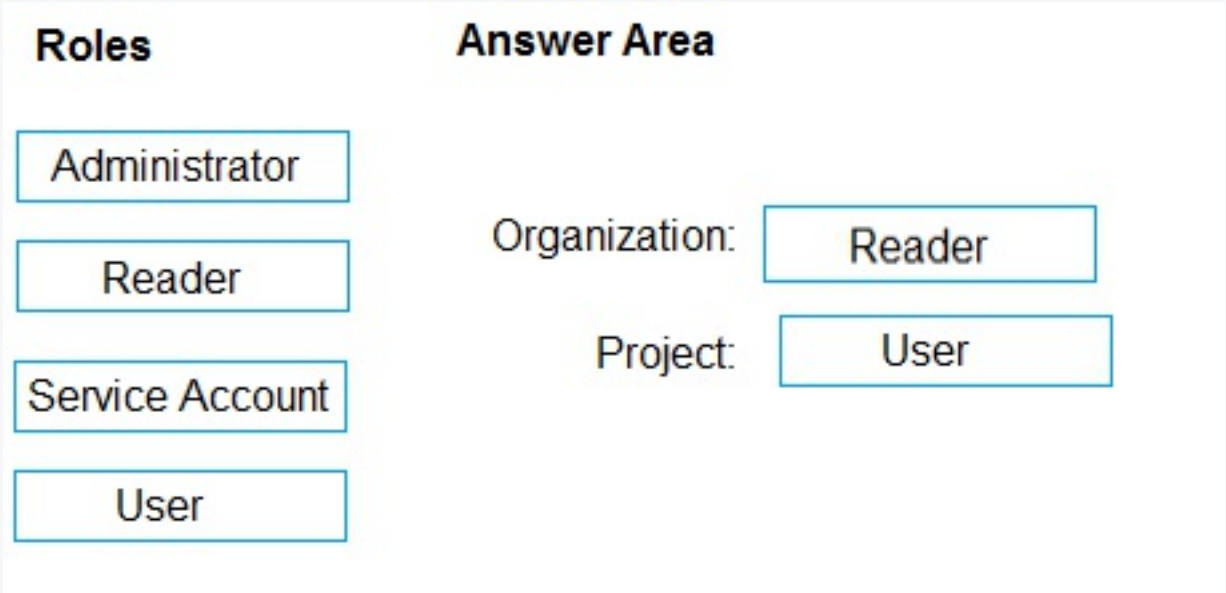
Question: 133 CertyIQ

DRAG DROP -
You need to configure access to Azure DevOps agent pools to meet the following requirements:
- Use a project agent pool when authoring build or release pipelines.
- View the agent pool and agents of the organization.
- Use the principle of least privilege.

Which role memberships are required for the Azure DevOps organization and the project? To answer, drag the appropriate role memberships to the correct targets. Each role membership may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Reader -

Members of the Reader role can view the organization agent pool as well as agents. You typically use this to

add operators that are responsible for monitoring the agents and their health.

Box 2: User

Members of this role can use the project agent pool when authoring pipelines.

Question: 134 CertyIQ

You have a branch policy in a project in Azure DevOps. The policy requires that code always builds successfully. You need to ensure that a specific user can always merge changes to the master branch, even if the code fails to compile. The solution must use the principle of least privilege.
What should you do?

A. Add the user to the Build Administrators group.
B. Add the user to the Project Administrators group.
C. From the Security settings of the repository, modify the access control for the user.
D. From the Security settings of the branch, modify the access control for the user.

Answer: D

Explanation:

The correct answer is **D. From the Security settings of the branch, modify the access control for the user.**

Here's why:

Azure DevOps employs a hierarchical permission model. Applying permissions at a more granular level, such as the branch level, adheres to the principle of least privilege. This means granting the user only the necessary permissions to bypass the build requirement for the master branch, rather than broader administrative rights.

Options A and B grant excessively high privileges. Adding the user to the Build Administrators or Project Administrators groups grants them significant control over builds and the entire project, respectively, which is far beyond what's necessary to simply bypass a branch policy's build requirement. This violates the principle of least privilege.

Option C, modifying repository-level security settings, is less granular than branch-level settings. While it might allow the user to bypass the policy, it applies that bypass to the entire repository, which is also broader than required.

Modifying branch-specific security settings directly addresses the requirement. Within the branch's security settings, it's possible to configure permissions for specific users to bypass policies, including the build requirement. This gives the user the explicit permission to merge even if the build fails only for that branch, upholding least privilege. By directly adjusting the access control list (ACL) for the user on the specific branch, you grant the precise permission needed, avoiding unintended broader impacts. This approach minimizes the potential for unintended consequences associated with elevated administrative roles or repository-wide changes.

Here's an analogy: Imagine needing to open a specific door. You wouldn't hand someone the keys to the entire building (Project Administrators) or even the entire floor (Repository security). Instead, you'd grant them access to that single door (Branch security).

Therefore, directly modifying the security settings of the master branch for the specified user provides the most targeted and secure solution, perfectly aligning with the principle of least privilege.

Further reading:

Set branch permissions: https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-permissions?view=azure-devops
Repositories permissions: https://learn.microsoft.com/en-us/azure/devops/organizations/security/permissions-access?view=azure-devops&tabs=preview-page

Question: 135 CertyIQ

You have an Azure Resource Manager template that deploys a multi-tier application.
You need to prevent the user who performs the deployment from viewing the account credentials and connection strings used by the application.
What should you use?

A. Azure Key Vault
B. a Web.config file
C. an Appsettings.json file
D. an Azure Storage table
E. an Azure Resource Manager parameter file

Answer: A

Explanation:

The correct answer is A: Azure Key Vault.

Azure Key Vault is a secure, centralized cloud service for storing and managing secrets, keys, and certificates. It's specifically designed to protect sensitive information like account credentials and connection strings. By storing these values in Key Vault, you prevent them from being directly embedded in your Azure Resource Manager (ARM) templates or configuration files.

When deploying the ARM template, you can reference the secrets stored in Key Vault. The ARM template accesses the secrets at deployment time using managed identities or service principals with appropriate permissions. This way, the user performing the deployment never directly sees the actual values of the sensitive information.

Options B and C (Web.config file and Appsettings.json file) are configuration files, commonly used to store application settings, but they are not inherently secure and can be easily accessed, especially if the application is compromised. Storing sensitive information directly in these files exposes them to potential security risks.

Option D (Azure Storage table) is a NoSQL data store. While you can technically store sensitive information in it, it lacks the built-in security features and audit trails specifically designed for managing secrets, making it a less secure option compared to Key Vault.

Option E (Azure Resource Manager parameter file) is used to provide parameters to your ARM template. While it can be used to pass values, it's not a secure location for storing sensitive information, as these files are often stored in source control and can be easily accessible.

Therefore, Azure Key Vault is the ideal solution because it provides a secure, centralized, and auditable way to manage sensitive information, preventing unauthorized access during and after deployment.

Refer to the official Microsoft Azure documentation for more information:

Azure Key Vault: https://learn.microsoft.com/en-us/azure/key-vault/

Using Key Vault secrets in ARM templates: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/key-vault-parameter

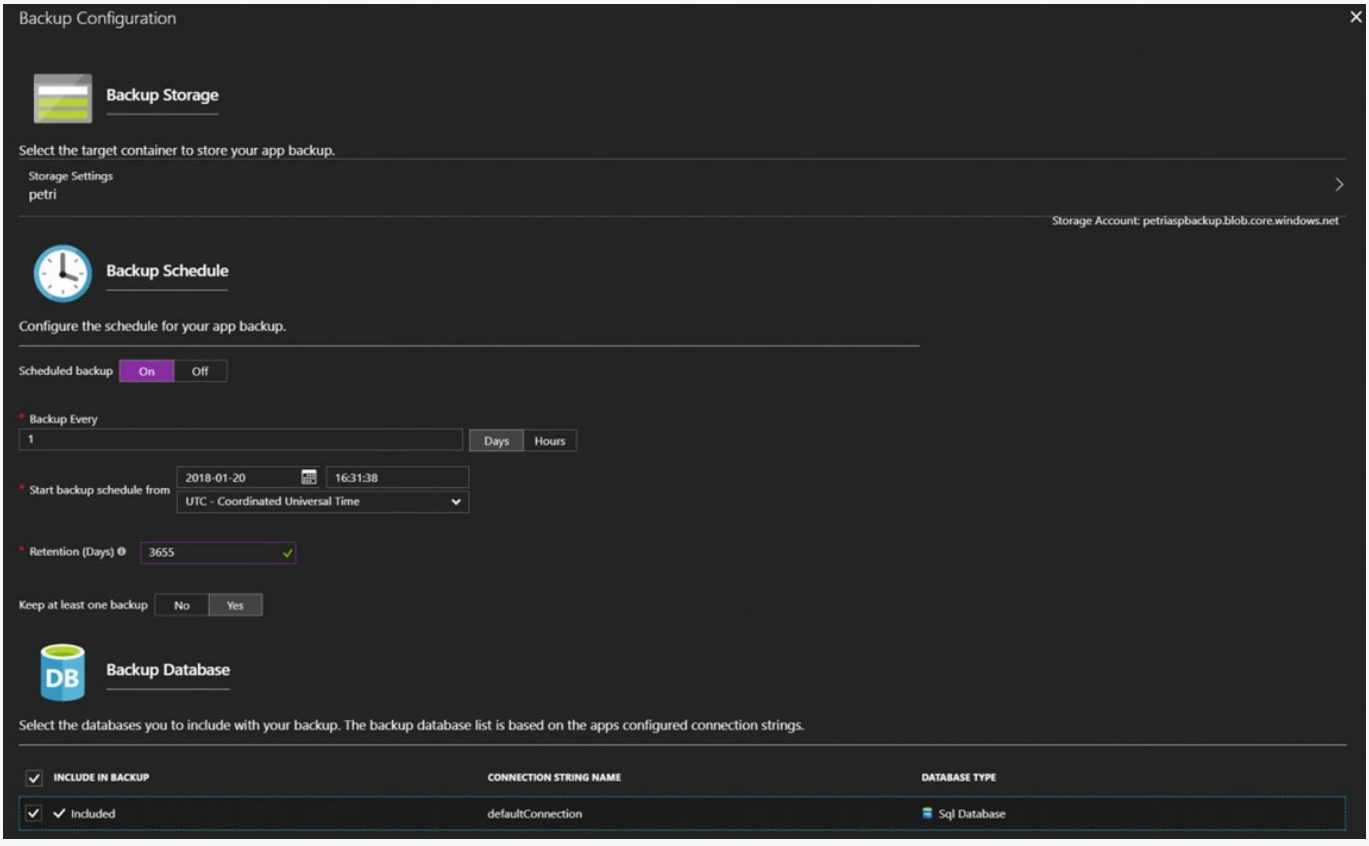
Question: 136 CertyIQ

SIMULATION -
Your company plans to implement a new compliance strategy that will require all Azure web apps to be backed up every five hours.
You need to back up an Azure web app named az400-123456789-main every five hours to an Azure Storage account in your resource group.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:
With the storage account ready, you can configure backs up in the web app or App Service.
1. Open the App Service az400-123456789-main, which you want to protect, in the Azure Portal and browse to Settings > Backups. Click Configure and a
Backup Configuration blade should appear.
2. Select the storage account.
3. Click + to create a private container. You could name this container after the web app or App Service.
4. Select the container.
5. If you want to schedule backups, then set Scheduled Backup to On and configure a schedule: every five hours
6. Select your retention. Note that 0 means never delete backups.
7. Decide if at least one backup should always be retained.
8. Choose if any connected databases should be included in the web app backup.
9. Click Save to finalize the backup configuration.



Reference:
https://petri.com/backing-azure-app-service

Question: 137 CertyIQ

SIMULATION -
You need to configure a virtual machine named VM1 to securely access stored secrets in an Azure Key Vault named az400-123456789-kv.
To complete this task, sign in to the Microsoft Azure portal.

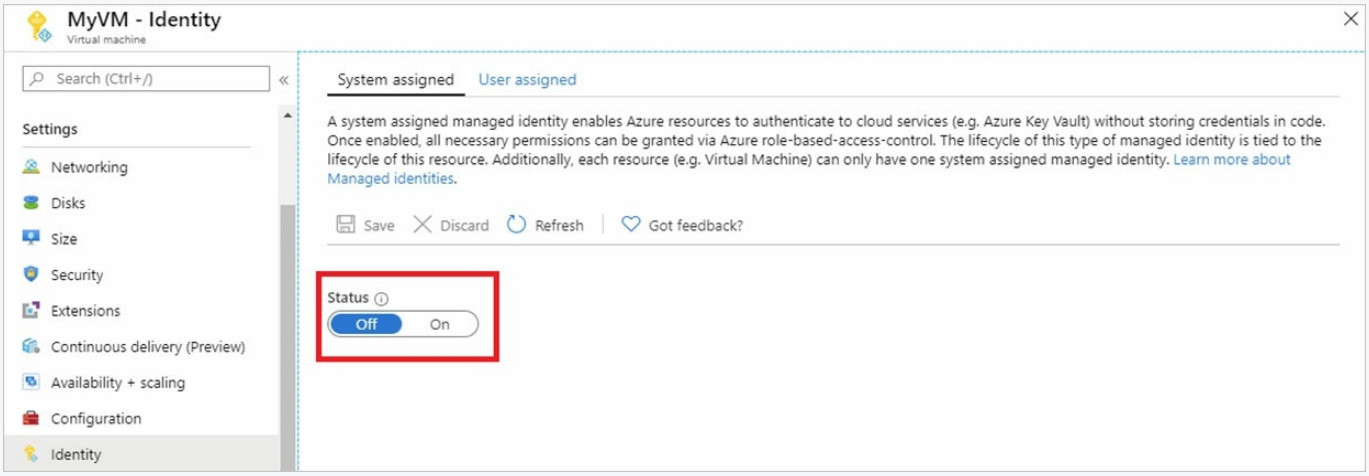
Answer:

See explanation below.

Explanation:

You can use a system-assigned managed identity for a Windows virtual machine (VM) to access Azure Key Vault.

1. Sign in to Azure portal
2. Locate virtual machine VM1.
3. Select Identity
4. Enable the system-assigned identity for VM1 by setting the Status to On.

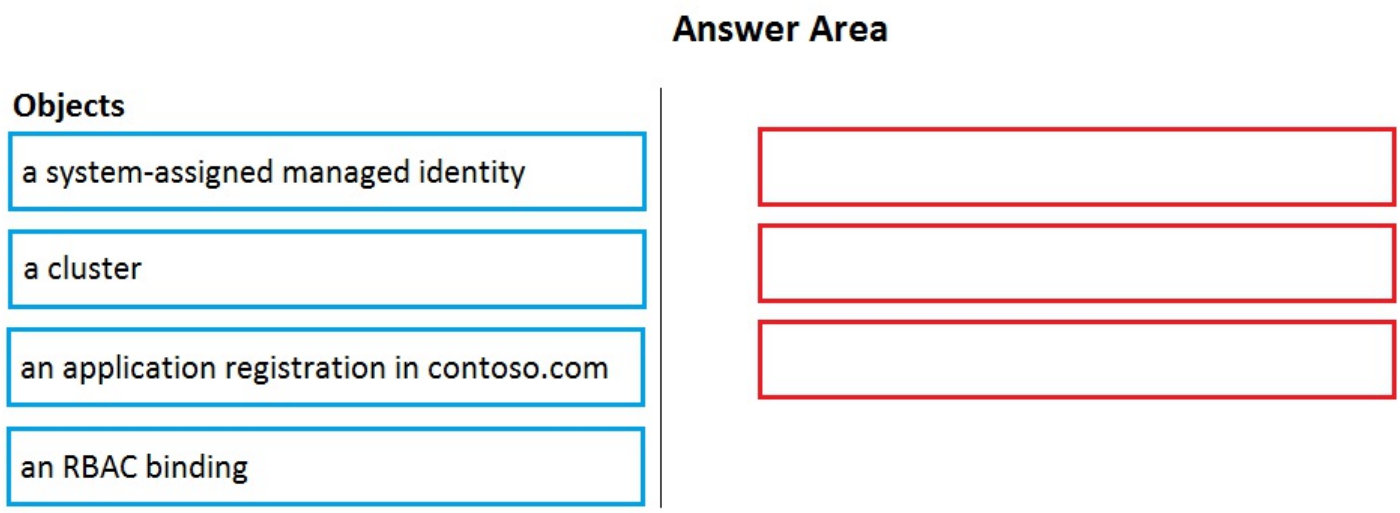


Note: Enabling a system-assigned managed identity is a one-click experience. You can either enable it during the creation of a VM or in the properties of an existing VM.

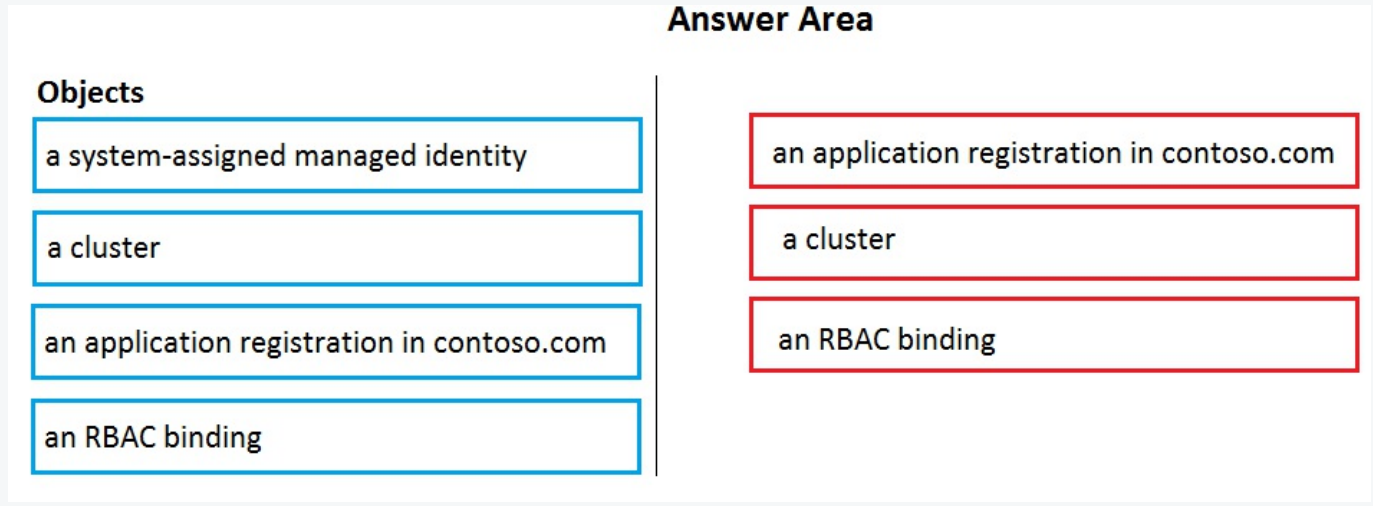
Reference:
https://docs.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/tutorial-windows-vm-access-nonaad

Question: 138 CertyIQ

DRAG DROP -
Your company has an Azure subscription named Subscription1. Subscription1 is associated to an Azure Active Directory tenant named contoso.com.
You need to provision an Azure Kubernetes Services (AKS) cluster in Subscription1 and set the permissions for the cluster by using RBAC roles that reference the identities in contoso.com.
Which three objects should you create in sequence? To answer, move the appropriate objects from the list of objects to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

1. an application registration in contoso.com.

2. a cluster.

3. an RBAC binding.

https://docs.microsoft.com/en-us/azure/aks/azure-ad-integration-cli

Question: 139 CertyIQ

HOTSPOT -
You manage build and release pipelines by using Azure DevOps. Your entire managed environment resides in Azure.
You need to configure a service endpoint for accessing Azure Key Vault secrets. The solution must meet the following requirements:
- Ensure that the secrets are retrieved by Azure DevOps.
- Avoid persisting credentials and tokens in Azure DevOps.

How should you configure the service endpoint? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

Service connection type:

- Azure Resource Manager
- Generic service
- Team Foundation Server / Azure Pipelines service connection

Authentication/authorization method for the connection:

- Azure Active Directory OAuth 2.0
- Grant authorization
- Managed Service Identity Authentication

Answer:

Answer Area

Service connection type:

- Azure Resource Manager
- Generic service
- Team Foundation Server / Azure Pipelines service connection

Authentication/authorization method for the connection:

- Azure Active Directory OAuth 2.0
- Grant authorization
- Managed Service Identity Authentication

Explanation:

Box 1: Azure Pipelines service connection

Box 2: Managed Service Identity Authentication

The managed identities for Azure resources feature in Azure Active Directory (Azure AD) provides Azure services with an automatically managed identity in Azure

AD. You can use the identity to authenticate to any service that supports Azure AD authentication, including Key Vault, without any credentials in your code.

Reference:

https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/deploy/azure-key-vault

https://docs.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/overview

Question: 140 CertyIQ

You are deploying a server application that will run on a Server Core installation of Windows Server 2019.
You create an Azure key vault and a secret.
You need to use the key vault to secure API secrets for third-party integrations.
Which three actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Configure RBAC for the key vault.

B. Modify the application to access the key vault.

C. Configure a Key Vault access policy.

D. Deploy an Azure Desired State Configuration (DSC) extension.

E. Deploy a virtual machine that uses a system-assigned managed identity.

Answer: BCE

Explanation:

The correct actions to secure API secrets for third-party integrations using Azure Key Vault on a Server Core installation are:

B. Modify the application to access the key vault: The application needs to be coded or configured to retrieve secrets from the Azure Key Vault rather than storing them directly in configuration files or code. This ensures that the application uses a secure location to get its secrets.

C. Configure a Key Vault access policy: An access policy determines which users, applications, or services have permission to perform specific operations on the Key Vault, such as reading secrets. It is essential to grant the application the necessary permissions (specifically, the permission to get secrets) within the Key Vault. Without a proper access policy, the application won't be authorized to retrieve the secrets.

E. Deploy a virtual machine that uses a system-assigned managed identity: Managed identities provide an identity for the VM that can be used to authenticate to Azure services, including Key Vault. By using a system-assigned managed identity, you avoid having to manage credentials (like usernames and passwords) within your application or on the VM. Azure automatically handles the lifecycle of the identity. The VM will need the managed identity to authenticate to Key Vault and retrieve secrets.

Why other options are incorrect:

A. Configure RBAC for the key vault: While RBAC (Role-Based Access Control) is important for controlling who can manage the Key Vault itself (e.g., who can create, delete, or modify secrets), it is not the mechanism by which the application accesses the secrets. Access policies control application access.

D. Deploy an Azure Desired State Configuration (DSC) extension: DSC is a configuration management platform. While DSC could be used to configure some aspects of the VM, it is not directly involved in the process of enabling an application to securely retrieve secrets from Key Vault. It's not the most efficient or direct solution for achieving the stated goal.

Justification Breakdown:

The core idea here is to enable the application running on the VM to retrieve secrets from the Key Vault securely. Option B handles the application side of things. Option C ensures that the application is authorized to access the Key Vault. Option E provides a mechanism for the VM to authenticate to the Key Vault without needing to manage credentials directly.

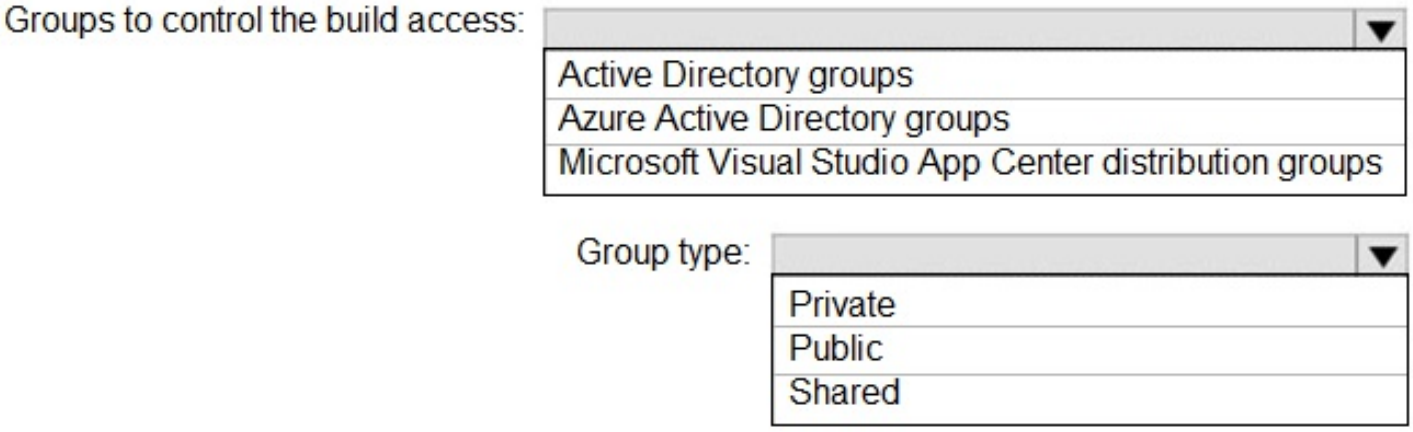
Authoritative Links:

About Azure Key Vault managed identities: https://learn.microsoft.com/en-us/azure/key-vault/general/managed-identity
Assign a managed identity to a virtual machine: https://learn.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/qs-configure-portal-windows-vm
Secure access to a key vault: https://learn.microsoft.com/en-us/azure/key-vault/general/security-features

Question: 141 CertyIQ

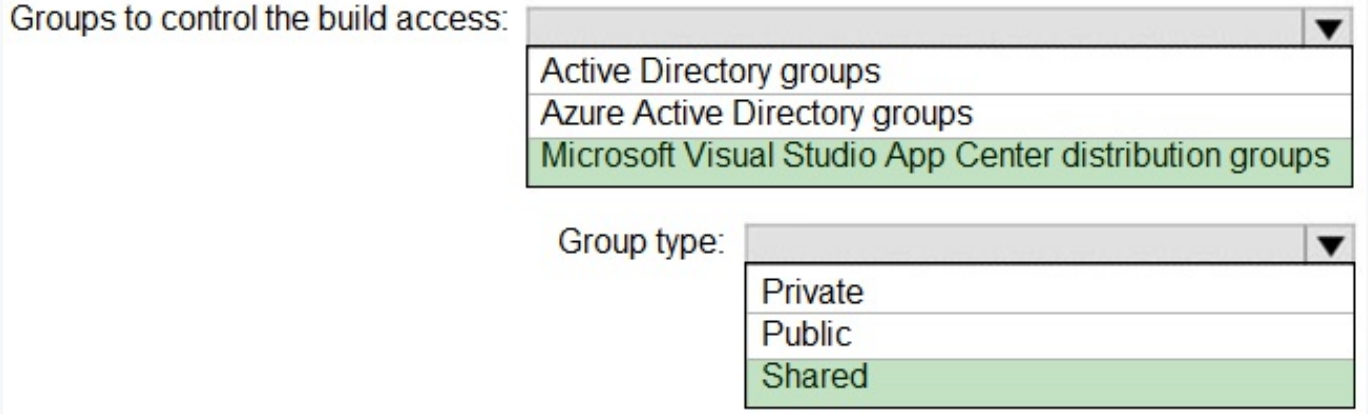
HOTSPOT -
Your company is creating a suite of three mobile applications.
You need to control access to the application builds. The solution must be managed at the organization level.
What should you use? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:

Box 1: Microsoft Visual Studio App Center distribution Groups

Distribution Groups are used to control access to releases. A Distribution Group represents a set of users that can be managed jointly and can have common access to releases. Example of Distribution Groups can be teams of users, like the QA Team or External Beta Testers or can represent stages or rings of releases, such as Staging.

Box 2: Shared -

Shared distribution groups are private or public distribution groups that are shared across multiple apps in a single organization. Shared distribution groups eliminate the need to replicate distribution groups across multiple apps.

Note: With the Deploy with App Center Task in Visual Studio Team Services, you can deploy your apps from Azure DevOps (formerly known as VSTS) to App

Center. By deploying to App Center, you will be able to distribute your builds to your users.

Reference:

https://docs.microsoft.com/en-us/appcenter/distribution/groups

Question: 142 CertyIQ

You have an Azure DevOps organization named Contoso that contains a project named Project1.
You provision an Azure key vault named Keyvault1.
You need to reference Keyvault1 secrets in a build pipeline of Project1.
What should you do first?

A. Add a secure file to Project1.
B. Create an XAML build service.
C. Create a variable group in Project1.
D. Configure the security policy of Contoso.

Answer: C

Explanation:

The correct first step is to create a variable group in Project1 that links to Keyvault1. This approach offers a secure and manageable way to reference secrets from Azure Key Vault within Azure DevOps pipelines.

Variable groups in Azure DevOps provide a centralized location to manage variables, including secrets. By linking a variable group to Keyvault1, you enable the pipeline to access the secrets stored in the Key Vault. This avoids hardcoding sensitive information directly into the pipeline definition, which is a major security risk.

Specifically, when you create the variable group, you can choose to link it to an Azure Key Vault subscription and then select Keyvault1. This establishes a secure connection between the Azure DevOps project and the Key Vault. Once the connection is established, you can select the specific secrets from Keyvault1 that you want to make available to the pipeline. Azure DevOps then retrieves these secrets during pipeline execution. The values are available as environment variables that can be used throughout the pipeline without exposing the actual secret values in the pipeline definition.

Options A, B, and D are not the primary steps to directly reference Keyvault1 secrets in a build pipeline. Adding a secure file is not the correct approach for key vault integration. Creating an XAML build service is irrelevant to the modern YAML pipeline approach and key vault integration. Configuring the security policy of Contoso at the organization level might be necessary eventually, but not as the first step; the variable group approach handles granular secret access from Key Vault at the project level.

Therefore, creating a variable group linked to the Azure Key Vault allows the pipeline to securely access and utilize the secrets stored in Keyvault1.

Relevant link for further research:

Use Azure Key Vault secrets in Azure Pipelines

Question: 143 CertyIQ

Your company uses Azure DevOps.
Only users who have accounts in Azure Active Directory can access the Azure DevOps environment.
You need to ensure that only devices that are connected to the on-premises network can access the Azure DevOps environment.
What should you do?

A. Assign the Stakeholder access level to all users.
B. In Azure Active Directory, configure risky sign-ins.

C. In Azure DevOps, configure Security in Project Settings.

D. In Azure Active Directory, configure conditional access.

Answer: D

Explanation:

The correct answer is D: In Azure Active Directory, configure conditional access.

Conditional Access in Azure Active Directory is a powerful tool for enforcing organizational policies and security requirements around user access to cloud applications, including Azure DevOps. It allows administrators to define conditions that must be met before a user can access a resource. In this scenario, the requirement is to restrict access to Azure DevOps to only devices connected to the on-premises network.

Conditional Access policies can leverage signals like device state (e.g., compliant, hybrid Azure AD joined), location (e.g., trusted network, country), and application to determine whether to grant or deny access. To achieve the desired outcome, a Conditional Access policy can be configured to:

1. **Target Azure DevOps:** Specify Azure DevOps as the cloud application the policy applies to.
2. **Define the Condition:** Use the "location" condition and configure it to allow access only when the user's IP address falls within the range of the on-premises network. This can be done by defining a named location in Azure AD that represents the on-premises network's IP address range.
3. **Grant or Deny Access:** Configure the policy to grant access only when the location condition is met. If the user is not on the on-premises network, access is blocked.

This approach ensures that only users connecting from devices within the on-premises network can access the Azure DevOps environment, fulfilling the security requirement.

Options A, B, and C are incorrect. Assigning the Stakeholder access level (A) only limits the features accessible within Azure DevOps, not the access itself based on network location. Configuring risky sign-ins (B) identifies unusual sign-in behavior but does not enforce network-based restrictions directly. Configuring Security in Project Settings within Azure DevOps (C) manages permissions and access within projects, not the initial authentication and authorization based on network location.

Authoritative Links:

What is Conditional Access?: https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/overview
Common Conditional Access policies: https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/howto-conditional-access-policy-common
Location condition in Azure AD Conditional Access: https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/location-condition

Question: 144 CertyIQ

You have the following Azure policy.

```
if: {
   allof: [
       {
              "field": "type",
              "equals": "Microsoft.Storage/storageAccounts"
       },
       {
              "field": "Microsoft.Storage/storageAccounts/supportsHttpsTrafficOnly",
              "notEquals": "true"
       }
   ]
},
then: {
   effect: "deny"
}
```

You assign the policy to the Tenant root group.
What is the effect of the policy?

A. prevents all HTTP traffic to existing Azure Storage accounts
B. ensures that all traffic to new Azure Storage accounts is encrypted
C. prevents HTTPS traffic to new Azure Storage accounts when the accounts are accessed over the Internet
D. ensures that all data for new Azure Storage accounts is encrypted at rest

Answer: B

Explanation:

Here is my 2 cents, looking at the link by default storage account does not support HTTPS and ARM template comes like below:

"supportsHttpsTrafficOnly": false,

The policy is trying to catch this by using below:

"field": "Microsoft.Storage/storageAccounts/supportsHttpsTrafficOnly",

"not Equals": "true"

If you read this carefully it means supports Https Traffic Only - Equals False (which is the default behaviour).. And then action item = DENY. So basically if its not HTTPS or encrypted it has to be denied.. So answer B

Denies non https traffic.

Question: 145 CertyIQ

You have an Azure DevOps organization named Contoso, an Azure DevOps project named Project1, an Azure subscription named Sub1, and an Azure key vault named vault1.
You need to ensure that you can reference the values of the secrets stored in vault1 in all the pipelines of Project1. The solution must prevent the values from being stored in the pipelines.
What should you do?

A. Create a variable group in Project1.
B. Add a secure file to Project1.
C. Modify the security settings of the pipelines.

D. Configure the security policy of Contoso.

Answer: A

Explanation:

The correct answer is **A: Create a variable group in Project1.**

Variable groups in Azure DevOps offer a centralized and secure way to manage and share variables across multiple pipelines within a project. When a variable group is linked to an Azure Key Vault, pipeline definitions can reference secrets stored in the vault. Crucially, the actual secret values are not stored within the pipeline itself or Azure DevOps, thereby enhancing security. The pipeline retrieves the secret directly from the Key Vault at runtime.

This approach adheres to security best practices by minimizing the exposure of sensitive information. Storing secrets directly in pipelines (as would be implied by options B, C, and D) increases the risk of accidental exposure through version control, logging, or unauthorized access. Secure files, while secure, don't directly integrate with Key Vault secrets. Modifying pipeline security settings would primarily address access controls rather than secret storage. A security policy at the organization level wouldn't directly achieve the required linking of secrets to pipelines within a specific project in a secure manner.

By creating a variable group linked to the Key Vault, all pipelines within Project1 can easily access the secrets without needing to define them individually in each pipeline and without ever persistently storing the secrets inside of Azure DevOps.

Further Research:

Use Azure Key Vault secrets in Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/library/variable-groups?view=azure-devops&tabs=yaml
Azure Key Vault: https://azure.microsoft.com/en-us/products/key-vault/

Question: 146 CertyIQ

DRAG DROP -
You use GitHub Enterprise Server as a source code repository.
You create an Azure DevOps organization named Contoso.
In the Contoso organization, you create a project named Project1.
You need to link GitHub commits, pull requests, and issues to the work items of Project1. The solution must use OAuth-based authentication.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:

Actions	Answer Area
From Developer settings in GitHub Enterprise Server, register a new OAuth app.	
From Project Settings in Azure DevOps, create a service hook subscription.	
From Organization settings in Azure DevOps, connect to Azure Active Directory (Azure AD).	
From Project Settings in Azure DevOps, add a GitHub connection.	
From Organization settings in Azure DevOps, add an OAuth configuration.	
From Developer settings in GitHub Enterprise Server, generate a private key.	

Answer:

Actions	Answer Area
From Developer settings in GitHub Enterprise Server, register a new OAuth app.	From Developer settings in GitHub Enterprise Server, register a new OAuth app.
From Project Settings in Azure DevOps, create a service hook subscription.	From Organization settings in Azure DevOps, add an OAuth configuration.
From Organization settings in Azure DevOps, connect to Azure Active Directory (Azure AD).	From Project Settings in Azure DevOps, add a GitHub connection.
From Project Settings in Azure DevOps, add a GitHub connection.	
From Organization settings in Azure DevOps, add an OAuth configuration.	
From Developer settings in GitHub Enterprise Server, generate a private key.	

Explanation:

Step 1: From Developer settings in GitHub Enterprise Server, register a new OAuth app.

If you plan to use OAuth to connect Azure DevOps Services or Azure DevOps Server with your GitHub Enterprise Server, you first need to register the application as an OAuth App

Step 2: Organization settings in Azure DevOps, add an OAuth configuration

Register your OAuth configuration in Azure DevOps Services.

Note:

1. Sign into the web portal for Azure DevOps Services.
2. Add the GitHub Enterprise Oauth configuration to your organization.
3. Open Organization settings>Oauth configurations, and choose Add Oauth configuration.
4. Fill in the form that appears, and then choose Create.

Step 3: From Project Settings in Azure DevOps, add a GitHub connection.

Connect Azure DevOps Services to GitHub Enterprise Server

Choose the Azure DevOps logo to open Projects, and then choose the Azure Boards project you want to configure to connect to your GitHub Enterprise repositories.

Choose (1) Project Settings, choose (2) GitHub connections and then (3) Click here to connect to your GitHub Enterprise organization.

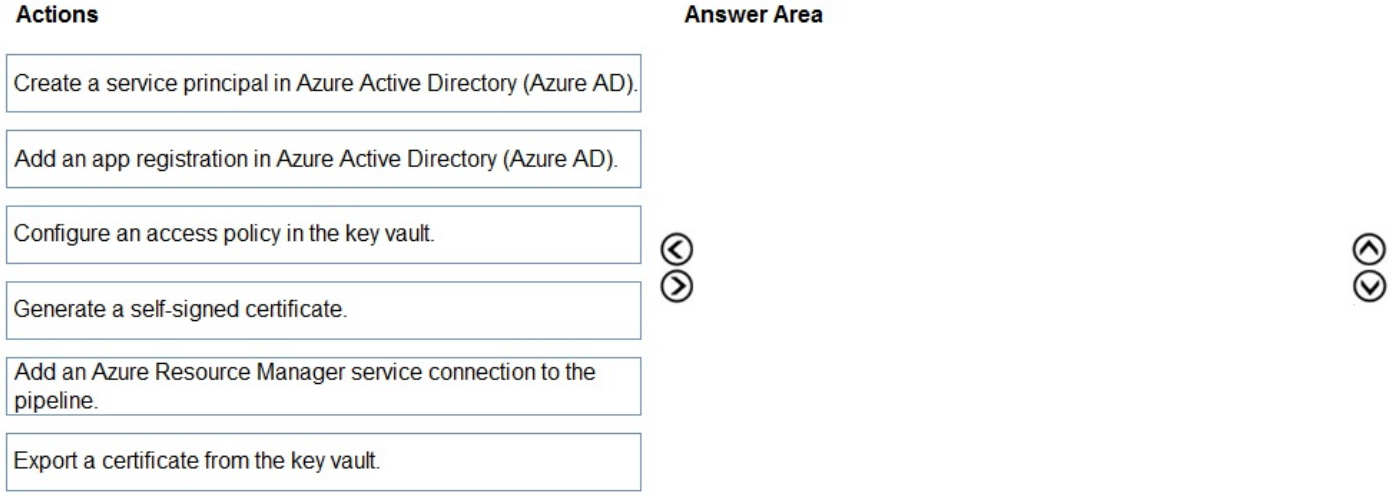
Reference:

https://docs.microsoft.com/en-us/azure/devops/boards/github/connect-to-github

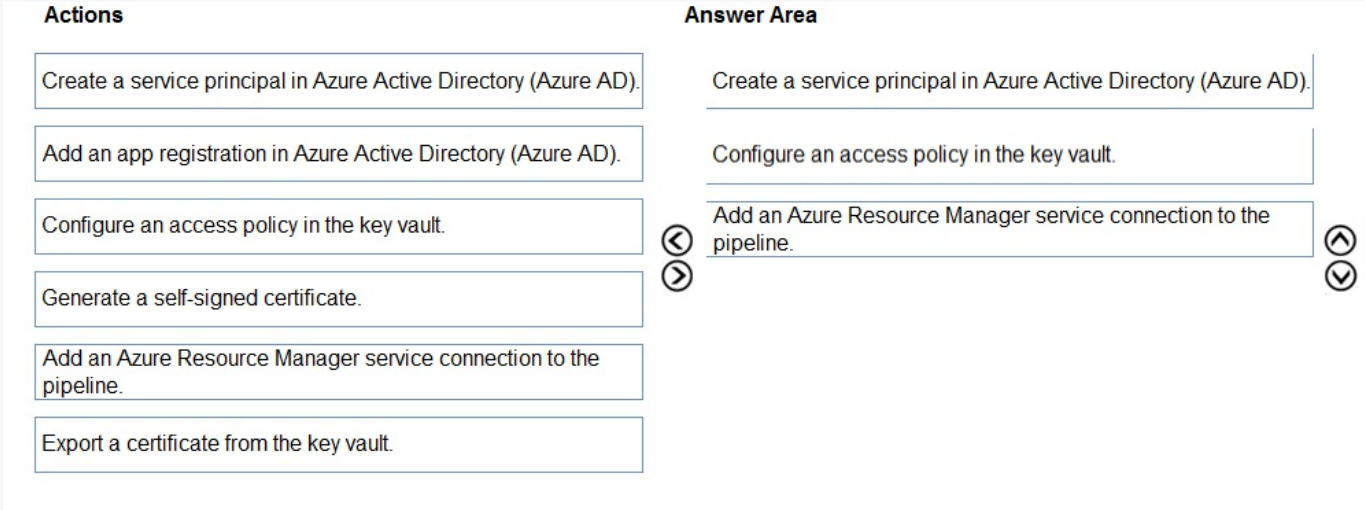
Question: 147

CertyIQ

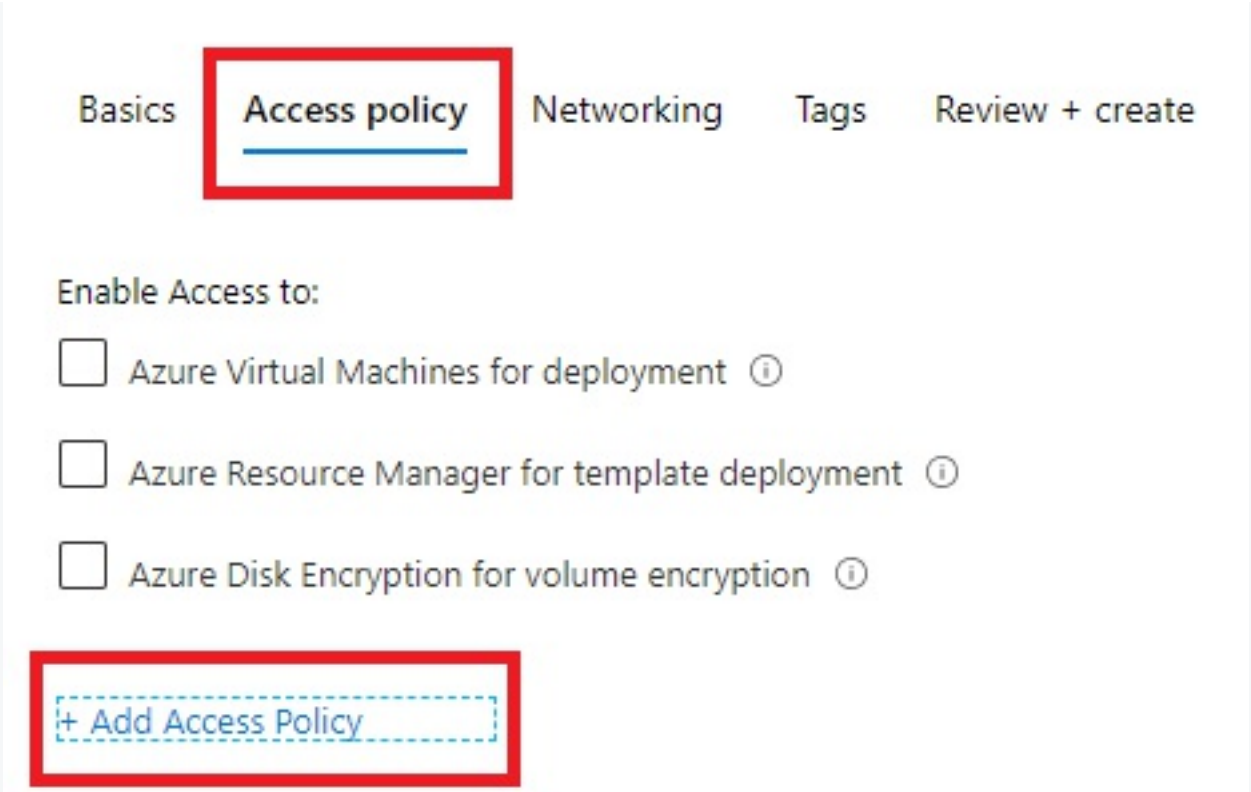
DRAG DROP -
You are configuring an Azure DevOps deployment pipeline. The deployed application will authenticate to a web service by using a secret stored in an Azure key vault.
You need to use the secret in the deployment pipeline.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



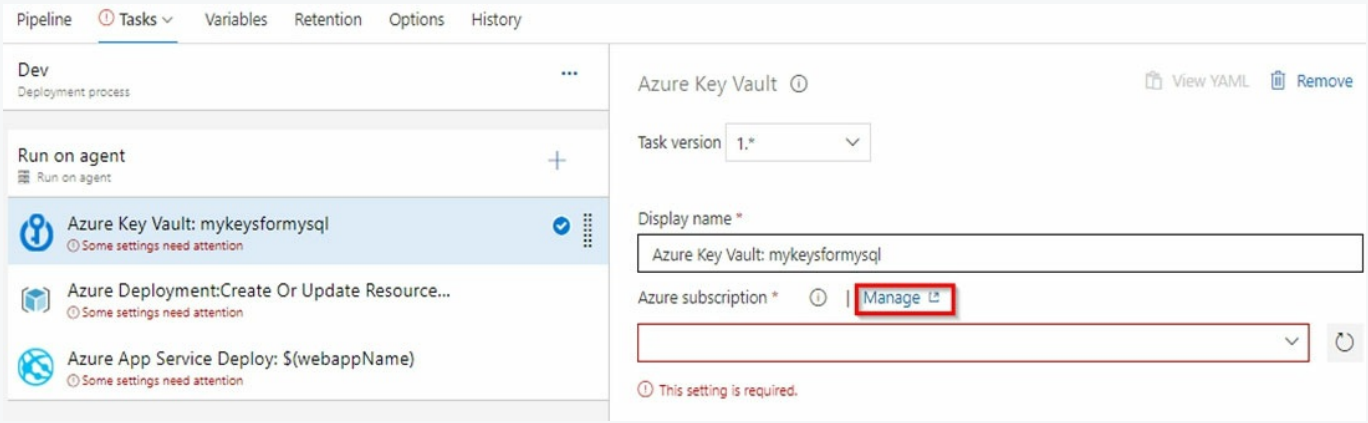
Answer:



Explanation:
Step 1: Create a service principal in Azure Active Directory (Azure AD).
You will need a service principal to deploy an app to an Azure resource from Azure Pipelines.
Step 2: Configure an access policy in the key vault.
You need to secure access to your key vaults by allowing only authorized applications and users. To access the data from the vault, you will need to provide read
(Get) permissions to the service principal that you will be using for authentication in the pipeline.
Select Access policy and then select + Add Access Policy to setup a new policy.



Step 3: Add an Azure Resource Manager service connection to the pipeline
You need to authorize the pipeline to deploy to Azure:
1. Select Pipelines | Pipelines,
2. Go to Releases under Pipelines and then select and Edit your pipeline.
3. Under Tasks, notice the release definition for Dev stage has a Azure Key Vault task. This task downloads Secrets from an Azure Key Vault. You will need to point to the subscription and the Azure Key Vault resource.
4. Click Manage, this will redirect to the Service connections page.



5.Click on New Service connection -> Azure Resource Manager -> Service Principal (manual). Fill the information from previously created service principal.

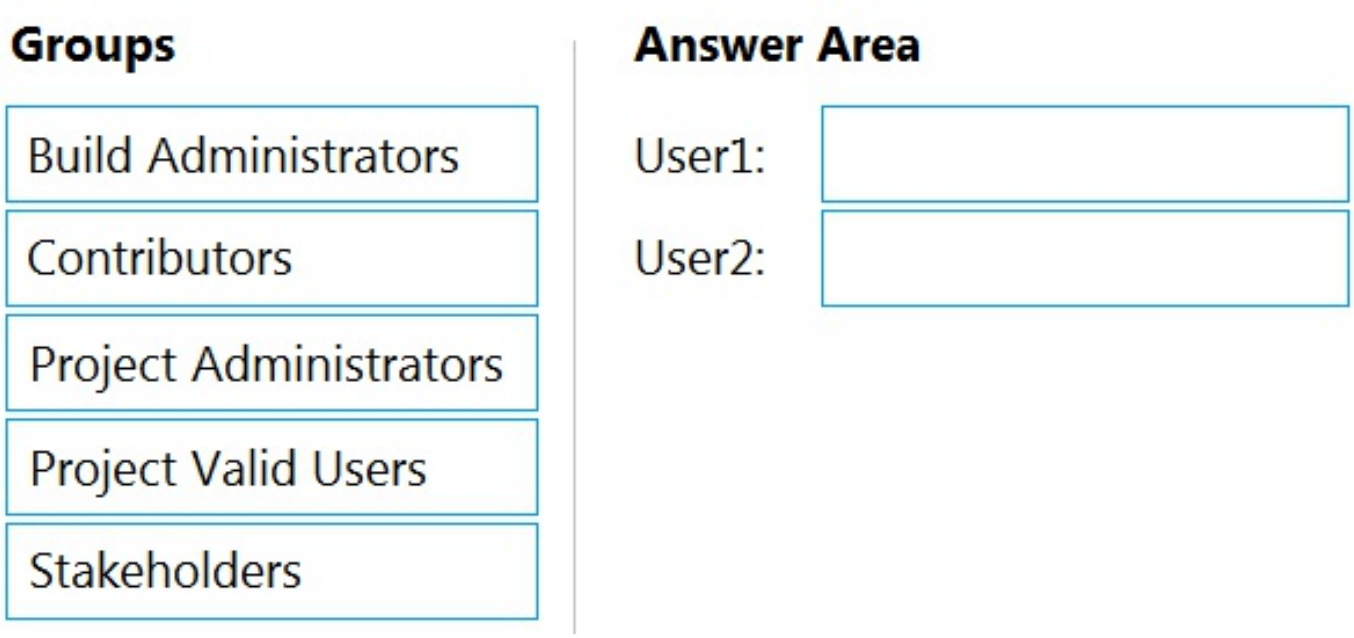
Reference:
https://azuredevopslabs.com/labs/vstsextend/azurekeyvault/

Question: 148 CertyIQ

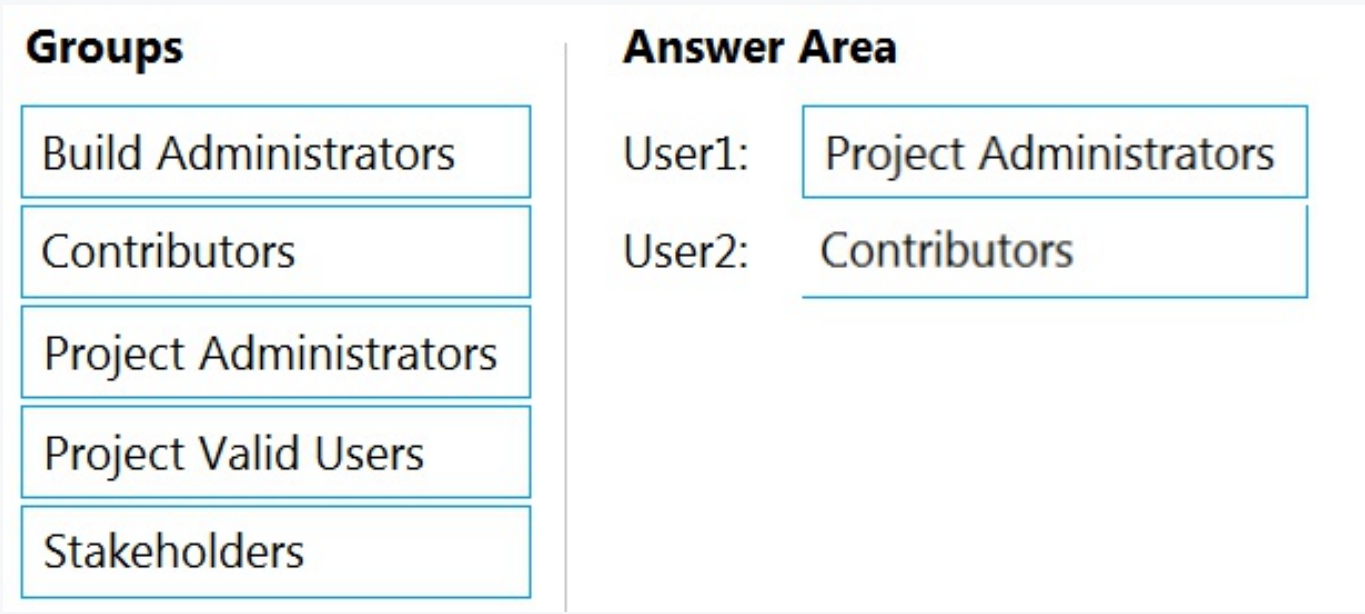
DRAG DROP -
You have a private project in Azure DevOps and two users named User1 and User2.
You need to add User1 and User2 to groups to meet the following requirements:
- User1 must be able to create a code wiki.
- User2 must be able to edit wiki pages.
- The solution must use the principle of least privilege.

To which group should you add each user? To answer, drag the appropriate groups to the correct users. Each group may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll

to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:
User1: Project Administrators -
You must have the permission Create Repository to publish code as wiki. By default, this permission is set for members of the Project Administrators group.

User2: Contributors -
Anyone who is a member of the Contributors security group can add or edit wiki pages.
Anyone with access to the team project, including stakeholders, can view the wiki.

Reference:
https://docs.microsoft.com/en-us/azure/devops/project/wiki/wiki-create-repo

Question: 149 CertyIQ

You use WhiteSource Bolt to scan a Node.js application.
The WhiteSource Bolt scan identifies numerous libraries that have invalid licenses. The libraries are used only

during development and are not part of a production deployment.
You need to ensure that WhiteSource Bolt only scans production dependencies.
Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Run npm install and specify the --production flag.

B. Modify the WhiteSource Bolt policy and set the action for the licenses used by the development tools to Reassign.

C. Modify the devDependencies section of the project's Package.json file.

D. Configure WhiteSource Bolt to scan the node_modules directory only.

Answer: AD

Explanation:

The correct answer is AD. Here's why:

A. Run npm install and specify the --production flag: The npm install --production command instructs npm to install only the dependencies listed in the dependencies section of the package.json file, skipping those in devDependencies. This ensures that only production-ready libraries are included in the node_modules directory and therefore scanned by WhiteSource Bolt. It directly addresses the problem of scanning development-only dependencies, which are not relevant to the production environment's license compliance.
https://docs.npmjs.com/cli/v9/commands/npm-install

D. Configure WhiteSource Bolt to scan the node_modules directory only: By focusing the scan on the node_modules directory, WhiteSource Bolt will only analyze the installed dependencies. If only production dependencies are installed in that directory (as achieved by step A), then WhiteSource Bolt will only examine those. This prevents the scanner from analyzing the package.json or package-lock.json files directly, where development dependencies are also listed, and therefore reduces false positives. This complements the previous action by ensuring the scope of the scanning focuses on production relevant files.

Let's analyze why the other options are less suitable:

B. Modify the WhiteSource Bolt policy and set the action for the licenses used by the development tools to Reassign: While modifying the policy is an option in WhiteSource Bolt, it only changes how identified components with those licenses are handled; it does not prevent the scanner from identifying the dev dependencies in the first place. The problem lies in the unnecessary analysis of these development libraries. Policies usually manage identified issues, not the initial scanning of the problems.

C. Modify the devDependencies section of the project's Package.json file: Modifying the devDependencies can impact development workflows, but it doesn't directly prevent WhiteSource Bolt from scanning those listed components during the analysis phase. package.json is often read for its declared dependency list, irrespective of environment, and it doesn't prevent it from being scanned.

In conclusion, npm install --production coupled with limiting the WhiteSource Bolt scope to the node_modules directory provides the most efficient solution for ensuring that only production dependencies are scanned, thus addressing the core issue described in the problem statement.

Question: 150 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not

appear in the review screen.
You plan to update the Azure DevOps strategy of your company.
You need to identify the following issues as they occur during the company's development process:
✑ Licensing violations
✑ Prohibited libraries
Solution: You implement continuous integration.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The answer is **B. No**.

Here's why:

Implementing continuous integration (CI) itself doesn't directly address licensing violations or the use of prohibited libraries. CI primarily focuses on automating the integration of code changes from multiple developers into a shared repository. Its core purpose is to detect integration errors early and frequently. While a CI pipeline can be extended to include checks for these issues, it's not an inherent feature.

To detect licensing violations and prohibited libraries, you need to incorporate specific tools and processes into your development workflow. Examples of such tools and techniques include:

Software Composition Analysis (SCA): SCA tools analyze the components (libraries, frameworks, etc.) used in a project to identify potential security vulnerabilities, licensing issues, and other risks. These tools can be integrated into a CI pipeline to automatically scan code during the build process.
Dependency Scanning: This focuses on identifying dependencies and their versions, which can then be checked against known vulnerability databases or internal policies.
License Compliance Tools: These help enforce licensing policies by identifying the licenses of used components and ensuring compliance with the organization's standards.
Policy Enforcement: Implement policies that define allowed and disallowed libraries and integrate checks into the CI/CD pipeline to enforce these policies.

Therefore, simply implementing CI alone will not meet the stated goal of identifying licensing violations and prohibited libraries. These require additional tooling and configuration within the development and CI/CD processes.

Supporting links:

Continuous Integration: https://www.atlassian.com/continuous-delivery/continuous-integration
Software Composition Analysis: https://owasp.org/www-project-top-ten/ (See A9:2021 – Software Integrity Failures)
Dependency Scanning: https://snyk.io/product/open-source-security-management/

Question: 151 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You plan to update the Azure DevOps strategy of your company.

You need to identify the following issues as they occur during the company's development process:
- Licensing violations
- Prohibited libraries

Solution: You implement pre-deployment gates.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of using pre-deployment gates does not directly address the identification of licensing violations or the use of prohibited libraries within the development process.

Pre-deployment gates are primarily designed to ensure that specific conditions are met before a deployment proceeds. Common use cases include verifying infrastructure health, running automated tests, or obtaining approvals. While gates could potentially be configured to run scripts that check for certain file types or naming conventions that might indicate licensing issues, this is not their intended primary function, and it would be a very indirect and unreliable method. It also would not readily identify the used of prohibited libraries within the code.

Identifying licensing violations and prohibited libraries typically requires more specialized tools and techniques, such as:

Software Composition Analysis (SCA) tools: These tools analyze the codebase and dependencies to identify open-source components and their associated licenses, alerting you to potential violations.
Static Code Analysis: Rules can be defined to ensure that certain libraries are not utilized within the code.

Pre-deployment gates could be part of a more comprehensive solution but are insufficient on their own to meet the specified goal.For more information on pre-deployment gates, refer to the official Azure DevOps documentation: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops&tabs=check-typesFor information about SCA, search for "Software Composition Analysis".

Question: 152 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You plan to update the Azure DevOps strategy of your company.
You need to identify the following issues as they occur during the company's development process:
- Licensing violations
- Prohibited libraries

Solution: You implement automated security testing.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

Yes, the solution meets the goal. Implementing automated security testing can effectively identify licensing violations and prohibited libraries during the development process.

Here's why:

Licensing Violations: Automated security testing tools often include static analysis capabilities that can scan the codebase and its dependencies to identify components with licenses that conflict with the company's policies. This can be achieved using tools that integrate with package managers and vulnerability databases.
Prohibited Libraries: Similarly, these tools can be configured to flag the usage of specific libraries that are prohibited due to security vulnerabilities, compliance requirements, or company policies. The test would be tailored to ensure that if these specific libraries are added into the code, the test will fail and will notify the team that an illegal library was implemented.
Integration into the Development Process: By integrating security testing into the CI/CD pipeline, these checks can be performed automatically with each build, providing early feedback to developers and preventing problematic code from reaching production.
Shift-Left Security: This approach aligns with the principles of "shift-left security," where security considerations are integrated early into the development lifecycle, rather than being addressed only at the end.

Therefore, automated security testing provides a mechanism to identify licensing violations and prohibited libraries, effectively addressing the identified issues.

Supporting Links:

Static Application Security Testing (SAST): https://owasp.org/www-project-top-ten/ (look for information on SAST)
DevSecOps: https://azure.microsoft.com/en-us/solutions/devsecops/

Question: 153 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You plan to update the Azure DevOps strategy of your company.
You need to identify the following issues as they occur during the company's development process:
- Licensing violations
- Prohibited libraries

Solution: You implement continuous deployment.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

Detailed Justification:

The proposed solution of implementing continuous deployment does not address the core requirements of identifying licensing violations and prohibited libraries during the development process. Continuous deployment focuses primarily on automating the release pipeline from code commit to production. While valuable for rapid software delivery, it lacks the mechanisms needed to scan code, dependencies, and libraries for licensing compliance or prohibited usage.

Identifying licensing violations and prohibited libraries requires specific tools and practices that are typically incorporated into the development pipeline at earlier stages, such as:

Static Code Analysis: Tools can analyze the source code for potential licensing issues or the use of forbidden libraries.
Software Composition Analysis (SCA): This technique identifies all open-source components within an application and checks for associated vulnerabilities and license compliance.
Dependency Scanning: Integrated into build processes, this process checks dependencies for known vulnerabilities and licensing risks.
Policy Enforcement: Implementing policies that automatically block the use of prohibited libraries or flag licensing issues during the build or pull request processes.

Continuous deployment is usually the stage after these checks have been completed and validated. It's a deployment mechanism, not a detection or prevention mechanism for licensing or prohibited library issues. Therefore, while continuous deployment is a valuable DevOps practice, it doesn't directly meet the goal of identifying these specific types of problems during the development lifecycle.

Authoritative Links for Further Research:

Software Composition Analysis (SCA): https://owasp.org/www-project-top-ten/
Static Application Security Testing (SAST): https://owasp.org/www-community/Source_Code_Analysis_Tools

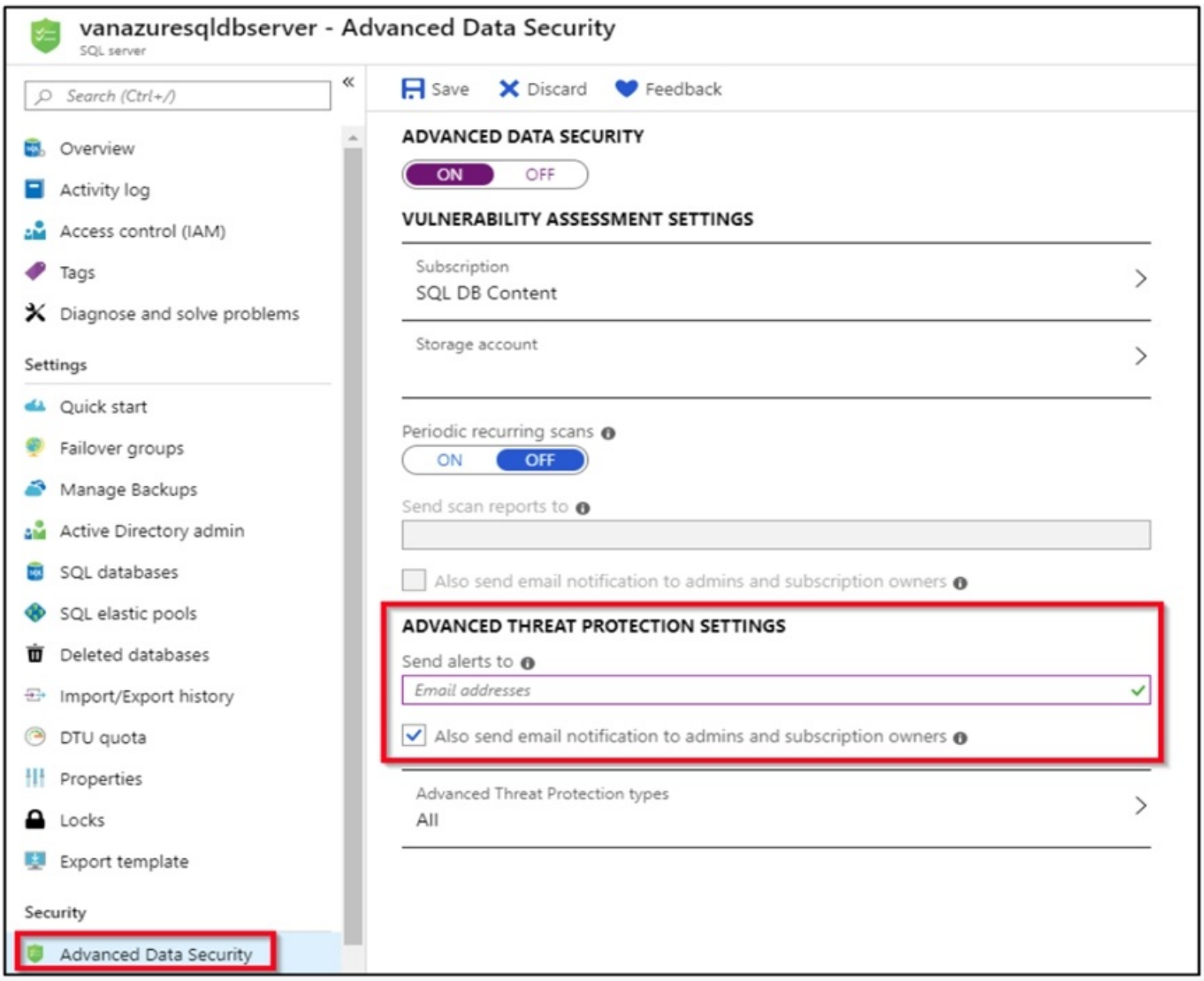
Question: 154 CertyIQ

SIMULATION -
You manage a website that uses an Azure SQL Database named db1 in a resource group named RG1lod11566895.
You need to modify the SQL database to protect against SQL injection.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:
Set up Advanced Threat Protection in the Azure portal
1. Sign into the Azure portal.
2. Navigate to the configuration page of the server you want to protect. In the security settings, select Advanced Data Security.
3. On the Advanced Data Security configuration page:



4. Enable Advanced Data Security on the server.

Note: Advanced Threat Protection for Azure SQL Database detects anomalous activities indicating unusual and potentially harmful attempts to access or exploit databases. Advanced Threat Protection can identify Potential SQL injection, Access from unusual location or data center, Access from unfamiliar principal or potentially harmful application, and Brute force SQL credentials

Reference:

https://docs.microsoft.com/en-us/azure/storage/common/storage-account-create

https://docs.microsoft.com/en-us/azure/azure-sql/database/threat-detection-configure

Question: 155 CertyIQ

HOTSPOT -

Your company has an Azure subscription.

The company requires that all resource groups in the subscription have a tag named organization set to a value of Contoso.

You need to implement a policy to meet the tagging requirement.

How should you complete the policy? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Hot Area:

Answer Area

```
{
 "policyRule": {
   "if": {
    "allOf": [
      {
       "field": "type",
       "equals":
      },
      {
        "not": {
          "field": "tags['organization']",
          "equals": "Contoso"
        }
      }
    ]
   },
   "then": {
     "effect":
     "details": [
       {
         "field": "tags['organization']",
         "value": "Contoso"
       }
     ]
    }
  }
}
```

"MicrosoftResources/deployments"
"MicrosoftResources/subscriptions"
"MicrosoftResources/subscriptions/resourceGroups"

"Append",
"Deny",
"DeployIfNotExists",

Answer:

Answer Area



Explanation:

Box 1: " Microsoft.Resources/subscriptions/resourceGroups"

Box 2: Append.

There is a pre-built policy in Azure for this, called: Append a tag and its value to resource groups

Question: 156

CertyIQ

You need to configure GitHub to use Azure Active Directory (Azure AD) for authentication.
What should you do first?

A. Create a conditional access policy in Azure AD.
B. Register GitHub in Azure AD.
C. Create an Azure Active Directory B2C (Azure AD B2C) tenant.
D. Modify the Security settings of the GitHub organization.

Answer: B

Explanation:

The correct first step is to register GitHub in Azure AD. Before any Azure AD features like authentication, conditional access, or user management can be applied to GitHub, Azure AD needs to recognize GitHub as an application. This recognition is established by registering GitHub within Azure AD. Registration creates an application object in Azure AD that represents GitHub, allowing Azure AD to issue tokens and manage access.

Registering GitHub in Azure AD allows Azure AD to establish a trust relationship with the GitHub application. This trust relationship is crucial for enabling single sign-on (SSO) functionality, where users can use their Azure AD credentials to access GitHub. Without registering the application, Azure AD would not be aware of GitHub, and authentication flows would not be possible.

A conditional access policy (A) can only be implemented after GitHub is registered as an application in Azure AD. Conditional access relies on Azure AD understanding the application it is protecting. Creating an Azure AD B2C tenant (C) is relevant for external user management, but not for authenticating internal employees via Azure AD to GitHub. Modifying the Security settings of the GitHub organization (D) is a configuration step that would come after integrating GitHub with Azure AD. Therefore, registering GitHub in Azure AD (B) is the necessary first step.

References:

Microsoft Documentation: https://learn.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals
Microsoft Documentation: https://learn.microsoft.com/en-us/azure/active-directory/saas-apps/github-tutorial

Question: 157 CertyIQ

You have an Azure DevOps project named Project1 and an Azure subscription named Sub1.
You need to prevent releases from being deployed unless the releases comply with the Azure Policy rules assigned to Sub1.
What should you do in the release pipeline of Project1?

A. Add a deployment gate.
B. Modify the Deployment queue settings.
C. Configure a deployment trigger.
D. Create a pipeline variable.

Answer: A

Explanation:

The correct answer is **A. Add a deployment gate.**

Deployment gates in Azure DevOps are specifically designed to control when a release proceeds to a subsequent stage. They act as a policy enforcement point, allowing you to integrate external checks, such as Azure Policy compliance, into your deployment pipeline. By adding a gate that validates against Azure Policy, you ensure that deployments only proceed if they adhere to the pre-defined rules in your subscription.

Here's a detailed breakdown:

Deployment Gates Explained: Deployment gates enable you to automatically evaluate conditions before or after a deployment. They ensure your releases meet specific criteria before moving forward. This includes checking health signals, security scans, and, in this case, policy compliance.

Azure Policy Integration: Azure Policy allows you to define and enforce organizational standards for resource

consistency, regulatory compliance, security, cost, and management. You can create policies that, for example, mandate specific resource tags, restrict resource types, or enforce encryption.

Preventing Non-Compliant Deployments: By adding a gate that checks Azure Policy compliance on Sub1, you are effectively preventing any release that violates these policies from being deployed. The gate will query the Azure Resource Manager to determine whether the resources being deployed comply with the assigned policies. If violations are found, the gate will fail, and the deployment will be blocked.

Alternatives are Unsuitable: Options B, C, and D are not suitable for enforcing Azure Policy compliance. Deployment queue settings (B) manage how deployments are queued and processed, not the criteria for deployment. Deployment triggers (C) define when a release pipeline starts, not whether a deployment complies with policies. Pipeline variables (D) store configuration values, not policy enforcement logic.

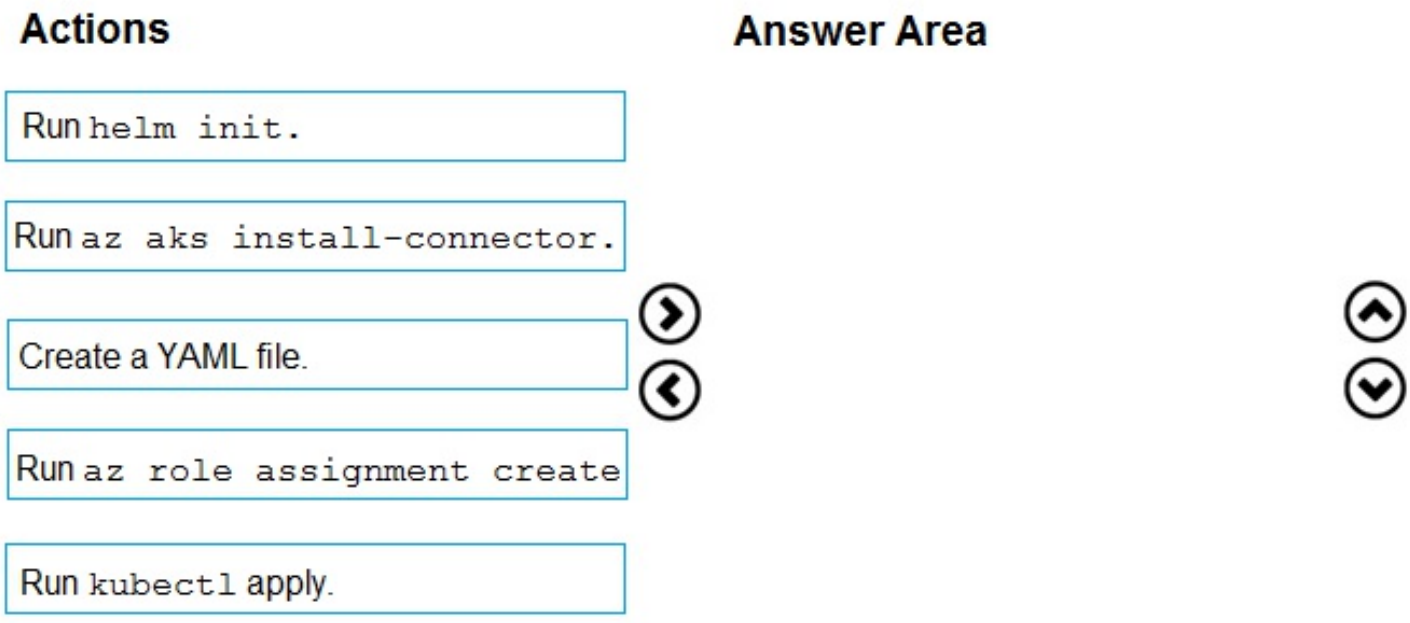
In summary, a deployment gate allows direct integration with Azure Policy to ensure deployments conform to established rules, preventing non-compliant releases from being deployed to the target Azure subscription. This ensures resource consistency, security, and regulatory compliance.

Authoritative Links:

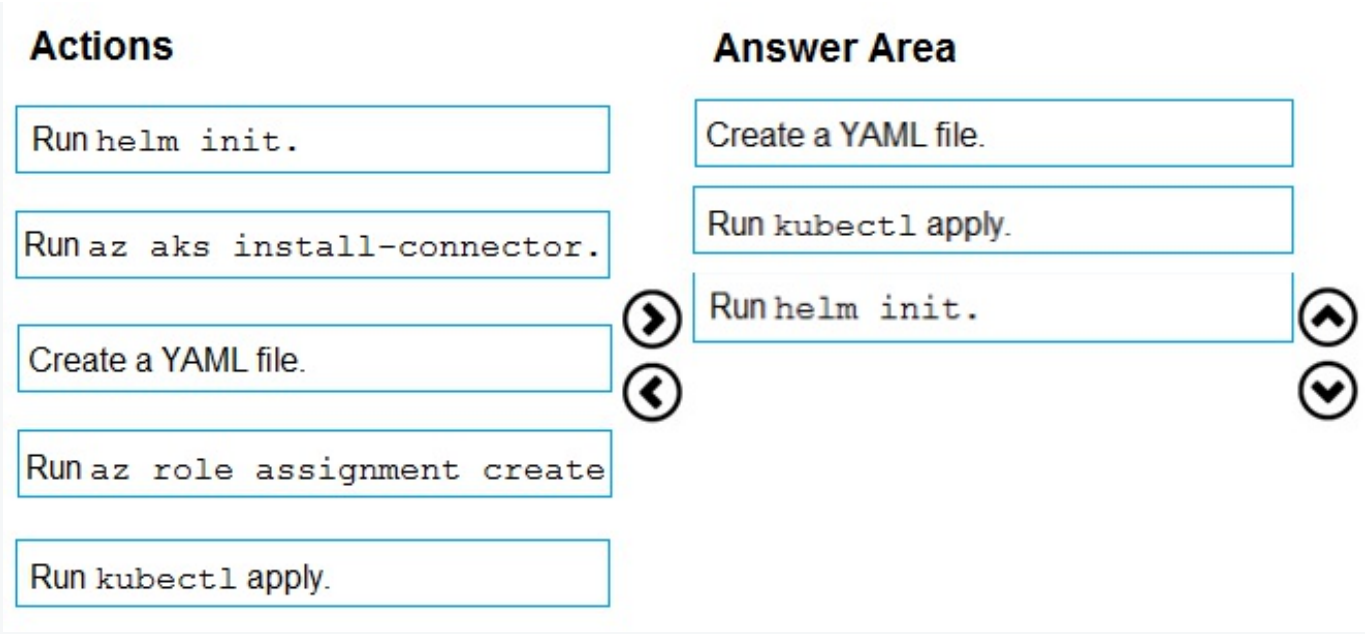
Azure DevOps Deployment Gates: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/approvals/gates?view=azure-devops
Azure Policy: https://learn.microsoft.com/en-us/azure/governance/policy/overview

Question: 158 CertyIQ

DRAG DROP -
You have an Azure Kubernetes Service (AKS) implementation that is RBAC-enabled.
You plan to use Azure Container Instances as a hosted development environment to run containers in the AKS implementation.
You need to configure Azure Container Instances as a hosted environment for running the containers in AKS.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:
Step 1: Create a YAML file.
If your AKS cluster is RBAC-enabled, you must create a service account and role binding for use with Tiller. To create a service account and role binding, create a file named rbac-virtual-kubelet.yaml
Step 2: Run kubectl apply.
Apply the service account and binding with kubectl apply and specify your rbac-virtual-kubelet.yaml file.
Step 3: Run helm init.
Configure Helm to use the tiller service account:
helm init --service-account tiller
You can now continue to installing the Virtual Kubelet into your AKS cluster.

Reference:
https://docs.microsoft.com/en-us/azure/aks/virtual-kubelet

Question: 159 CertyIQ

You have an Azure DevOps project that contains a build pipeline. The build pipeline uses approximately 50 open source libraries.
You need to ensure that all the open source libraries comply with your company's licensing standards.
Which service should you use?

A. Ansible
B. Maven
C. WhiteSource Bolt
D. Helm

Answer: C

Explanation:

The correct answer is C, WhiteSource Bolt. Here's why:

WhiteSource Bolt, now integrated into Mend, is specifically designed for identifying and managing open-source components within a software project, including build pipelines in Azure DevOps. Its primary function is to scan projects and detect open-source libraries and their associated licenses. It then compares these licenses against a predefined policy, highlighting any violations or compliance issues. This functionality makes

it ideal for ensuring open-source libraries comply with licensing standards.

Ansible (A) is an automation engine used for configuration management, application deployment, and task automation. While it can be used to manage software dependencies, it is not primarily focused on license compliance.

Maven (B) is a build automation tool primarily used for Java projects. It manages dependencies but does not inherently provide license compliance checking or reporting. It relies on plugins or integrations with dedicated license management tools to achieve that.

Helm (D) is a package manager for Kubernetes. It helps manage Kubernetes applications and their dependencies, but it doesn't focus on open-source license compliance scanning and reporting.

Therefore, WhiteSource Bolt is the most suitable service for identifying open-source libraries and ensuring they comply with the company's licensing standards in an Azure DevOps build pipeline. It provides direct integration with Azure DevOps for scanning and reporting on open-source components.

For further research, consult the following links:

Mend (formerly WhiteSource Bolt) for Azure DevOps: https://www.mend.io/product/whitesource-bolt/azure-devops/ (formerly WhiteSource Bolt)
Azure DevOps Marketplace - Mend Bolt: https://marketplace.visualstudio.com/items?itemName=whitesource.whitesource-bolt

Question: 160 CertyIQ

You are designing the security validation strategy for a project in Azure DevOps.
You need to identify package dependencies that have known security issues and can be resolved by an update.
What should you use?

A. Octopus Deploy
B. Jenkins
C. Gradle
D. SonarQube

Answer: D

Explanation:

The correct answer is D, SonarQube. SonarQube is a widely used open-source platform designed for continuous inspection of code quality and security. One of its core capabilities is identifying vulnerabilities in project dependencies. It analyzes the dependency graph, flagging libraries and packages that have known security flaws registered in public vulnerability databases. SonarQube can then suggest updates to versions that address these vulnerabilities, thereby mitigating potential risks. Options A, B, and C are incorrect. Octopus Deploy (A) is a deployment automation tool, primarily focused on releasing applications to various environments, and not designed for security vulnerability detection in dependencies. Jenkins (B) is a CI/CD automation server but requires plugins to achieve similar functionality to SonarQube, and it's not its primary focus. Gradle (C) is a build automation tool used for managing project dependencies, but it does not inherently provide security analysis capabilities like SonarQube. Therefore, SonarQube is the most appropriate choice for identifying and suggesting resolutions for package dependencies with known security issues within an Azure DevOps project.

Here are some authoritative links for further research:

SonarQube: https://www.sonarqube.org/
Dependency-Check: https://owasp.org/www-project-dependency-check/ (SonarQube leverages components like this)

Question: 161 CertyIQ

You administer an Azure DevOps project that includes package feeds.
You need to ensure that developers can unlist and deprecate packages. The solution must use the principle of least privilege.
Which access level should you grant to the developers?

A. Collaborator
B. Contributor
C. Owner

Answer: B

Explanation:

The correct access level to grant developers to unlist and deprecate packages in Azure DevOps, while adhering to the principle of least privilege, is **Contributor**.

The Contributor role provides sufficient permissions to manage packages within a feed. Specifically, Contributors have the permission to push (publish), consume (download), unlist, and deprecate packages. This aligns with the requirement of allowing developers to unlist and deprecate packages.

The Collaborator role is generally intended for users who need access to source code and work items but don't necessarily need to manage packages. Therefore, it wouldn't grant the necessary permissions for package management.

The Owner role, on the other hand, provides full control over the feed, including managing permissions and deleting the feed itself. This is excessive for the specified requirement and violates the principle of least privilege. By granting Contributor access, developers gain the specific permissions they need to perform their tasks without unnecessary access to other feed settings or functionalities. Using the Contributor role mitigates the risk of unintended changes or security breaches due to over-permissioning.

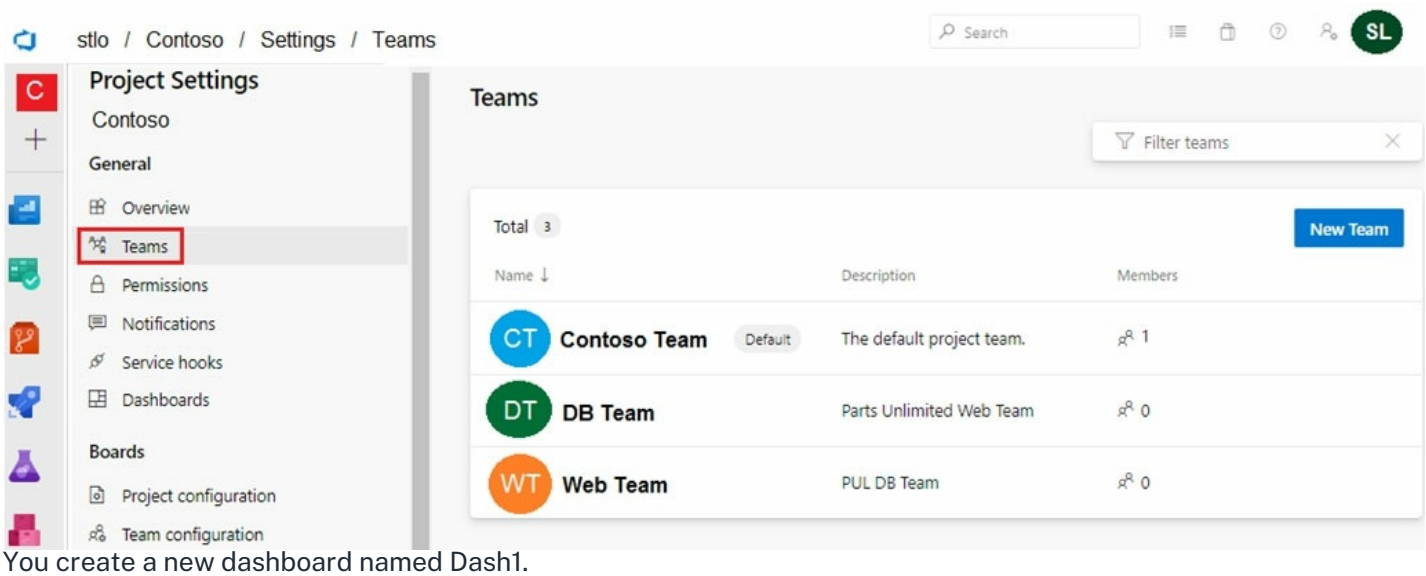
In summary, Contributor strikes the right balance between granting the necessary permissions for package management and adhering to the principle of least privilege, making it the most appropriate access level for this scenario.

Supporting documentation:

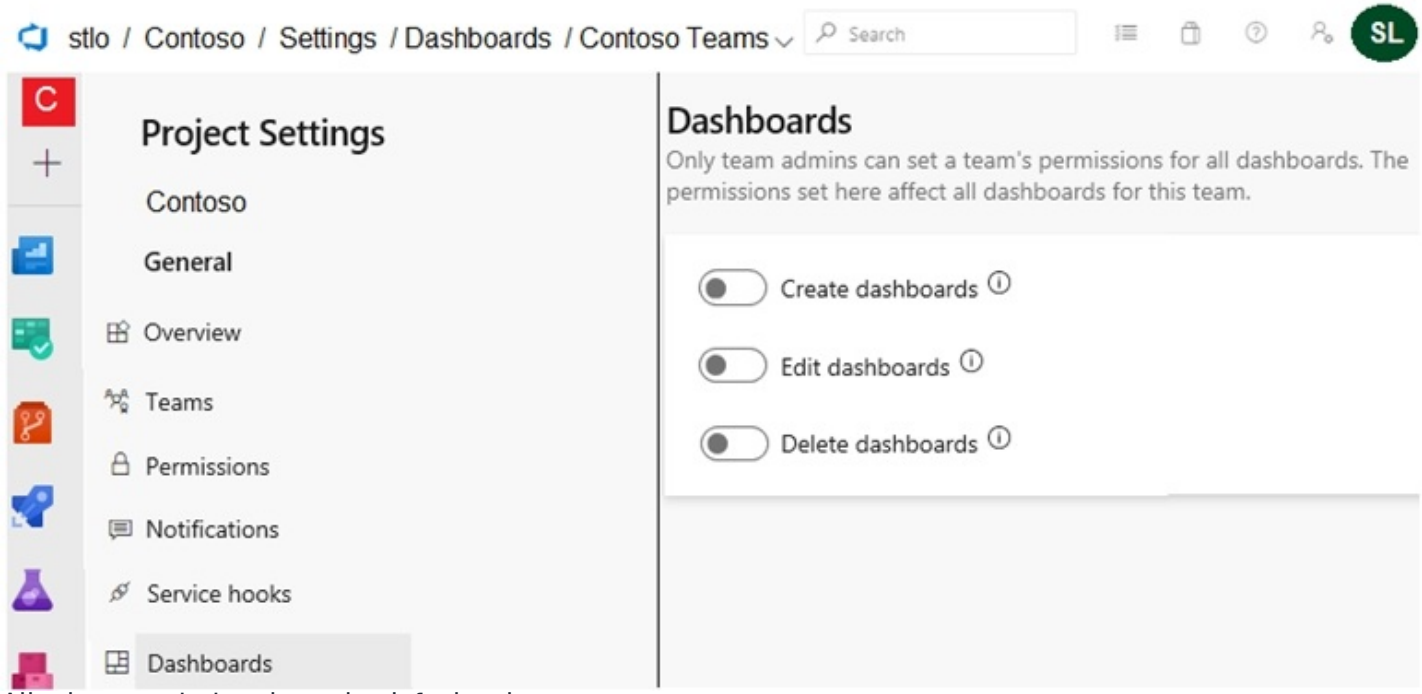
Azure Artifacts permissions: This document details the various permission levels within Azure Artifacts feeds and what actions each role is permitted to perform.
Understand access levels: This document explains the different access levels in Azure DevOps and their associated capabilities.

Question: 162 CertyIQ

HOTSPOT -
You have a project in Azure DevOps that has three teams as shown in the Teams exhibit. (Click the Teams tab.)



You create a new dashboard named Dash1.
You configure the dashboard permissions for the Contoso project as shown in the Permissions exhibit. (Click the Permissions tab.)



All other permissions have the default values set.
For each of the following statements, select Yes if the statement is true. Otherwise, select No.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

Statements	Yes	No
Web Team can delete Dash1.	○	○
Contoso Team can view Dash1.	○	○
Project administrators can create new dashboards.	○	○

Answer:

Answer Area

Statements	Yes	No
Web Team can delete Dash1.	○	○
Contoso Team can view Dash1.	○	○
Project administrators can create new dashboards.	○	○

Explanation:

'Web Team can delete Dash1 (No): By default, team members (including the Web Team) are granted permissions to edit their own team dashboards. However, they do not have the administrative permissions required to delete a dashboard unless specifically granted.

Contoso Team can view Dash1 (Yes): In Azure DevOps, project members are granted "View" permissions for project-level assets by default. Unless there is an explicit "Deny" rule configured against them, the Contoso Team will be able to view the dashboard.

Project administrators can create new dashboards (Yes): The "Project Administrators" group is a built-in security role with broad permissions across the project, which includes the right to create, edit, and manage dashboards.

Reference:

https://docs.microsoft.com/en-us/azure/devops/report/dashboards/charts-dashboard-permissions-access

Question: 163 CertyIQ

Your company is concerned that when developers introduce open source libraries, it creates licensing compliance issues.
You need to add an automated process to the build pipeline to detect when common open source libraries are added to the code base.
What should you use?

A. Microsoft Visual SourceSafe
B. Code Style
C. Black Duck
D. Jenkins

Answer: C

Explanation:

The correct answer is **C. Black Duck**. Here's why:

Black Duck is a software composition analysis (SCA) tool specifically designed to manage open-source software (OSS) risks, including licensing compliance, security vulnerabilities, and code quality. It integrates directly into CI/CD pipelines, like Azure DevOps pipelines, to automatically scan codebases for open-source

components. When it finds a library, it identifies the license and alerts the team to potential compliance issues.

Visual SourceSafe (A) is a source control system, not an SCA tool. Code Style (B) focuses on code formatting, not license detection. Jenkins (D) is a general-purpose automation server, not specifically designed for OSS risk management; while Jenkins can be configured for SCA using plugins, Black Duck provides a dedicated, more comprehensive solution.

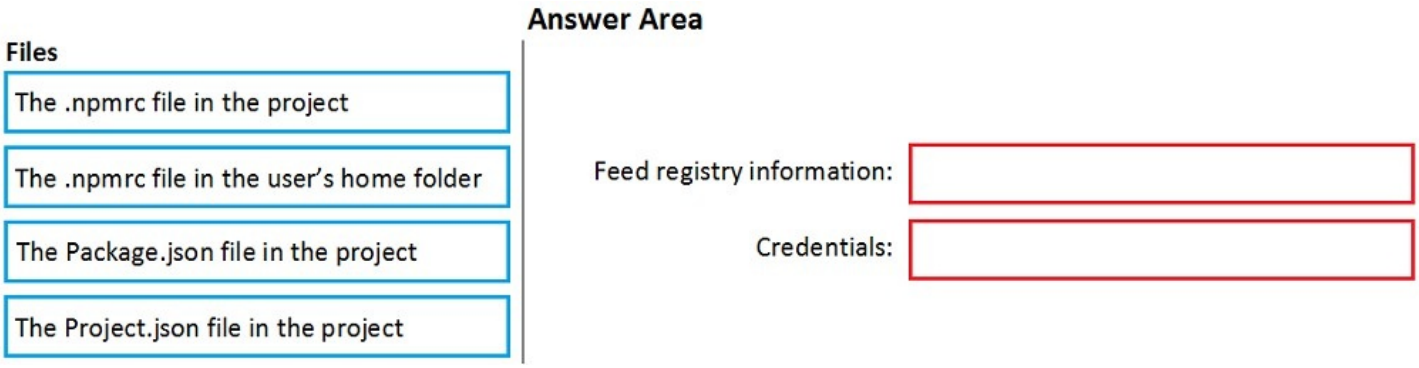
By using Black Duck, the company can automatically detect open-source libraries added to their codebase during the build process. Black Duck will then provide details on the license associated with the library, helping the company determine if the use of the library is compliant with their licensing policies. This proactive detection helps avoid potential legal and financial risks associated with non-compliance. SCA tools are a standard part of a secure software development lifecycle and critical for managing the risks inherent in using open-source dependencies.

Further research:

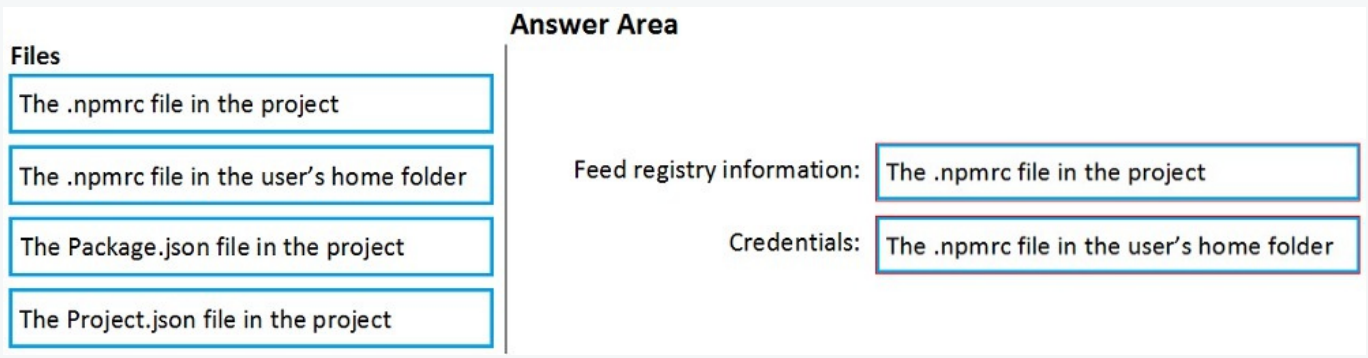
Synopsys Black Duck: https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html
OWASP Software Component Verification Standard (SCVS): https://owasp.org/www-project-software-component-verification-standard/ - for information on SCA in general.

Question: 164 CertyIQ

DRAG DROP -
You are implementing a package management solution for a Node.js application by using Azure Artifacts.
You need to configure the development environment to connect to the package repository. The solution must minimize the likelihood that credentials will be leaked.
Which file should you use to configure each connection? To answer, drag the appropriate files to the correct connections. Each file may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



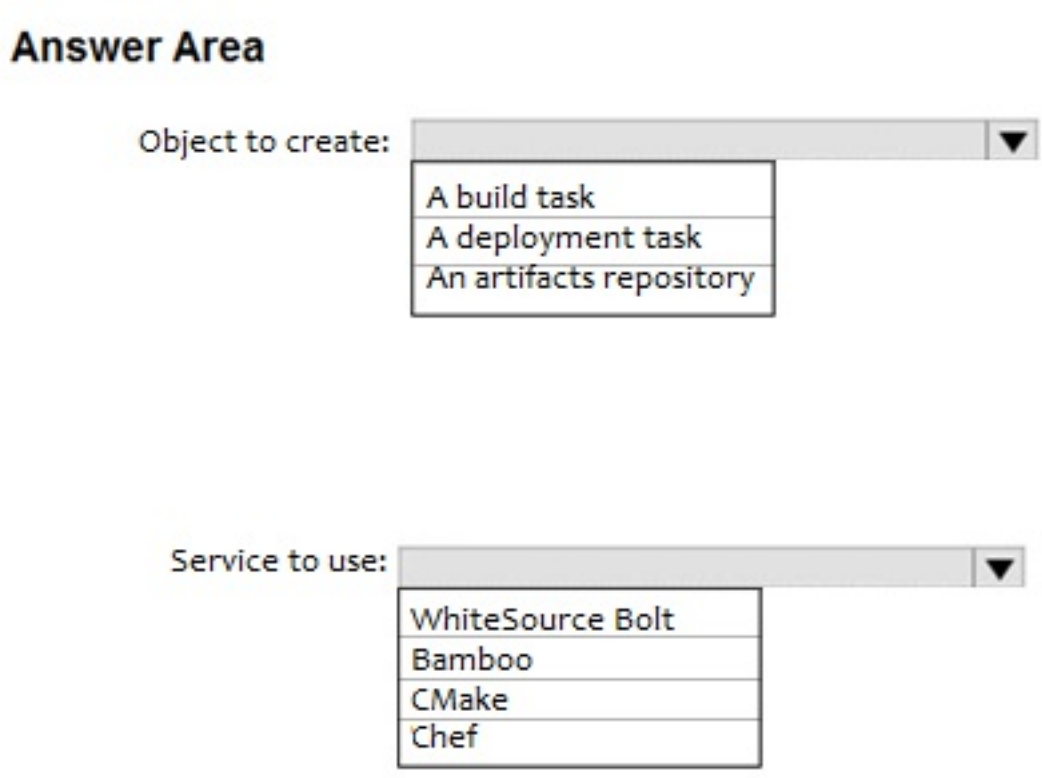
Explanation:

All Azure Artifacts feeds require authentication, so you'll need to store credentials for the feed before you can install or publish packages. npm uses .npmrc configuration files to store feed URLs and credentials. Azure DevOps Services recommends using two .npmrc files.
Feed registry information: The .npmrc file in the project
One .npmrc should live at the root of your git repo adjacent to your project's package.json. It should contain a "registry" line for your feed and it should not contain credentials since it will be checked into git.
Credentials: The .npmrc file in the user's home folder
On your development machine, you will also have a .npmrc in $home for Linux or Mac systems or $env.HOME for win systems. This .npmrc should contain credentials for all of the registries that you need to connect to.
The NPM client will look at your project's .npmrc, discover the registry, and fetch matching credentials from $home/.npmrc or $env.HOME/.npmrc.

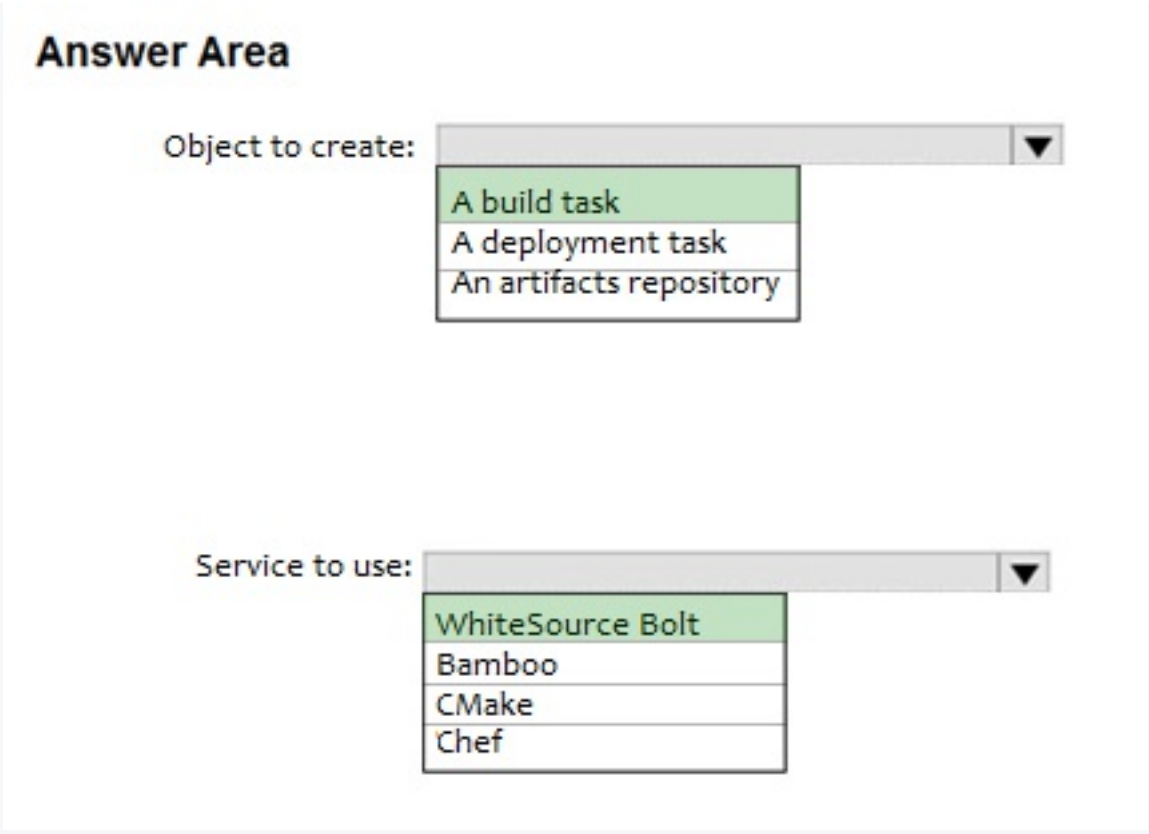
Reference:
https://docs.microsoft.com/en-us/azure/devops/artifacts/npm/npmrc?view=azure-devops&tabs=windows

Question: 165 CertyIQ

HOTSPOT -
You have an Azure DevOps project that contains a build pipeline. The build pipeline uses approximately 50 open source libraries.
You need to ensure that the project can be scanned for known security vulnerabilities in the open source libraries.
What should you do? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:



Answer:



Explanation:
Box 1: A Build task -

Trigger a build -
You have a Java code provisioned by the Azure DevOps demo generator. You will use WhiteSource Bolt extension to check the vulnerable components present in this code.
1. Go to Builds section under Pipelines tab, select the build definition WhiteSourceBolt and click on Queue to trigger a build.
2. To view the build in progress status, click on ellipsis and select View build results.

Box 2: WhiteSource Bolt -
WhiteSource is the leader in continuous open source software security and compliance management. WhiteSource integrates into your build process, irrespective of your programming languages, build tools, or development environments. It works automatically, continuously, and silently in the background, checking the security, licensing, and quality of your open source components against WhiteSource constantly-updated definitive database of open source repositories.

Reference:
https://www.azuredevopslabs.com/labs/vstsextend/whitesource/

Question: 166 CertyIQ

You have an Azure DevOps project that contains a build pipeline. The build pipeline uses approximately 50 open source libraries.
You need to ensure that all the open source libraries comply with your company's licensing standards.
Which service should you use?

A. NuGet
B. Maven
C. Black Duck
D. Helm

Answer: C

Explanation:

The correct answer is **C. Black Duck**. Here's why:

Black Duck specializes in software composition analysis (SCA). SCA tools are designed to identify and manage open-source software components used in a project. This includes detecting the open-source libraries, identifying their licenses, and helping you understand associated vulnerabilities. This capability directly addresses the stated need for ensuring compliance with licensing standards. Black Duck helps organizations understand the risk associated with open-source components, including legal risks stemming from license violations.

NuGet (A) and Maven (B) are package managers. While they facilitate the inclusion of open-source libraries in your projects, they do not inherently provide license compliance scanning or reporting. They primarily focus on dependency management and package retrieval/distribution. You can technically find license information within NuGet or Maven packages, but the platforms don't automatically scan and report on compliance at the project level in the way Black Duck does. They don't enforce or validate licensing policies on their own.

Helm (D) is a package manager for Kubernetes. It focuses on deploying and managing applications within a Kubernetes cluster and is not directly related to open source license compliance within the build pipeline itself.

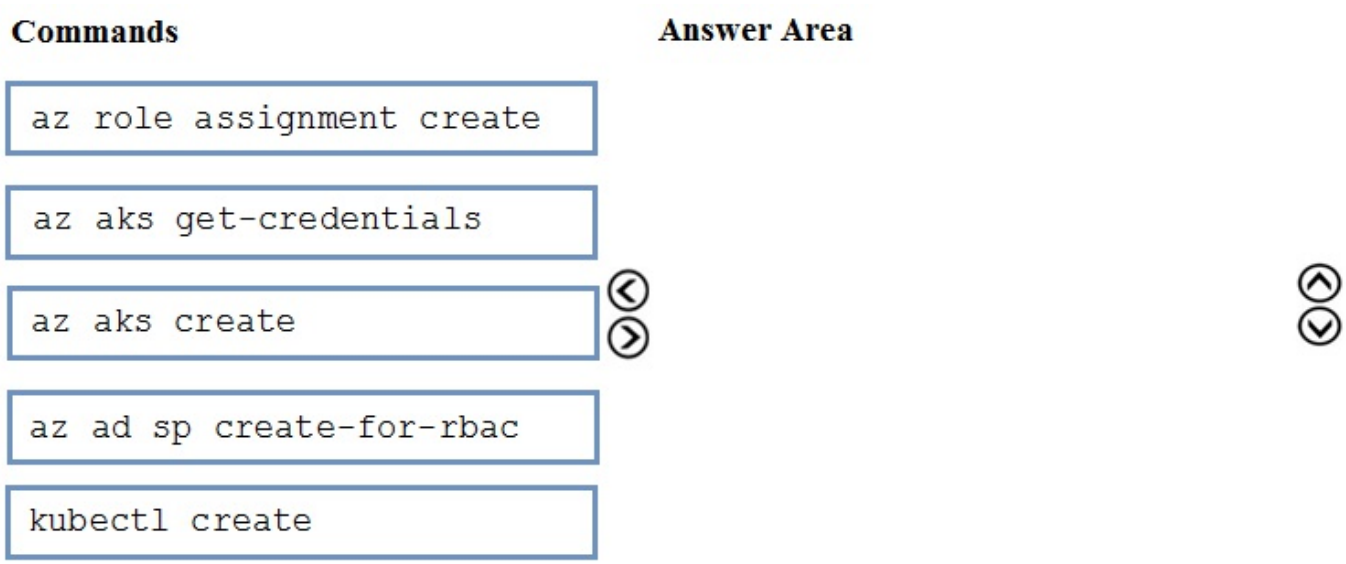
Therefore, Black Duck is the most appropriate service for identifying the open-source licenses used in your build pipeline's dependencies and ensuring they comply with your company's standards. NuGet, Maven, and Helm lack the dedicated license compliance analysis features provided by tools like Black Duck.

Here are some authoritative links for further research:

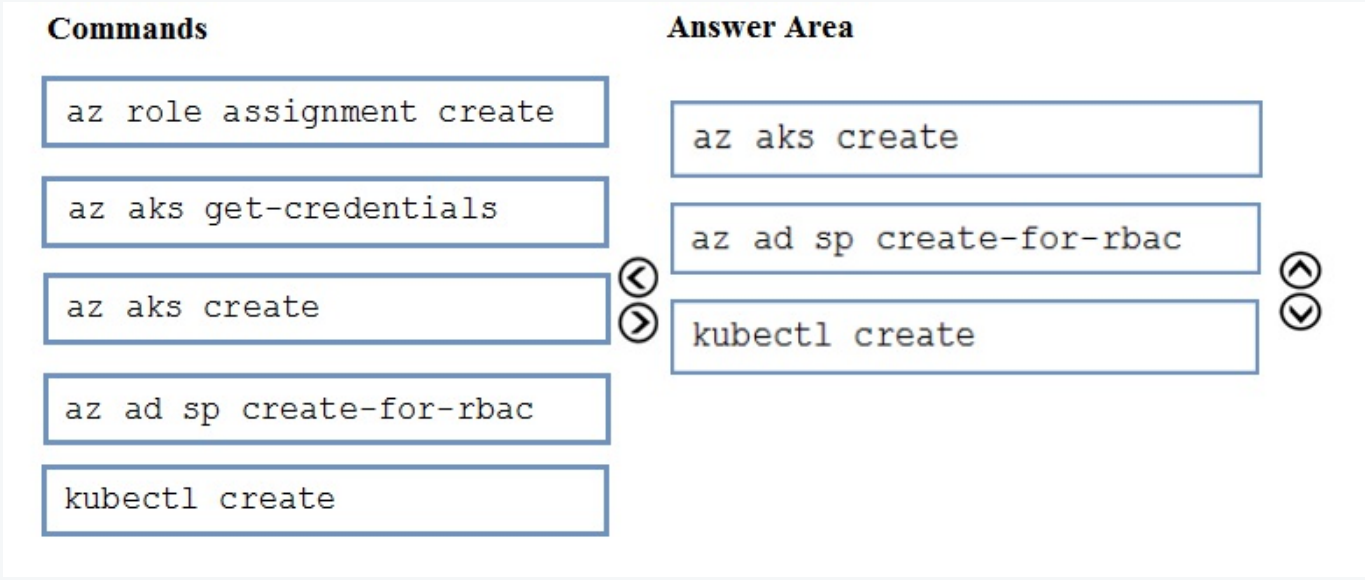
Synopsys (Black Duck's vendor): https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html
OWASP on Software Composition Analysis: https://owasp.org/www-project-top-ten/ (While not specifically about Black Duck, it explains the need for SCA).
Wikipedia on Software Composition Analysis: https://en.wikipedia.org/wiki/Software_composition_analysis

Question: 167 CertyIQ

DRAG DROP -
You plan to use Azure Kubernetes Service (AKS) to host containers deployed from images hosted in a Docker Trusted Registry.
You need to recommend a solution for provisioning and connecting to AKS. The solution must ensure that AKS is RBAC-enabled and uses a custom service principal.
Which three commands should you recommend be run in sequence? To answer, move the appropriate commands from the list of commands to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Step 1 : az acr create -
An Azure Container Registry (ACR) can also be created using the new Azure CLI. az acr create
--name <REGISTRY_NAME>
--resource-group <RESOURCE_GROUP_NAME>
--sku Basic
Step 2: az ad sp create-for-rbac
Once the ACR has been provisioned, you can either enable administrative access (which is okay for testing) or you create a Service Principal (sp) which will provide a client_id and a client_secret. az ad sp create-for-rbac
--scopes
/subscriptions/<SUBSCRIPTION_ID>/resourcegroups/<RG_NAME>/providers/Microsoft.ContainerRegistry/registrie
--role Contributor
--name <SERVICE_PRINCIPAL_NAME>

Step 3: kubectl create -
Create a new Kubernetes Secret.
kubectl create secret docker-registry <SECRET_NAME>
--docker-server <REGISTRY_NAME>.azurecr.io
--docker-email <YOUR_MAIL>
--docker-username=<SERVICE_PRINCIPAL_ID>

--docker-password <YOUR_PASSWORD>

Reference:
https://thorsten-hans.com/how-to-use-private-azure-container-registry-with-kubernetes

Question: 168 CertyIQ

Your company develops an app for iOS. All users of the app have devices that are members of a private distribution group in Microsoft Visual Studio App Center.
You plan to distribute a new release of the app.
You need to identify which certificate file you require to distribute the new release from App Center.
Which file type should you upload to App Center?

A. .cer
B. .pfx
C. .p12
D. .pvk

Answer: C

Explanation:

The correct file type for distributing an iOS app release from App Center to a private distribution group is **.p12 (Personal Information Exchange)**.

Here's why:

.p12 files contain both the digital certificate and its private key. App Center needs both to sign the app correctly for distribution and install it on user devices. The signing process ensures that the app is trusted and verifiable, coming from a known source (your company). Without the private key, App Center cannot properly sign the application package (.ipa).

iOS code signing relies heavily on digital certificates and keys. The .p12 file provides this package in a single, secure file. The private key is crucial for verifying the identity of the developer and ensuring the app hasn't been tampered with since it was signed.

.cer files only contain the public certificate. They are used to verify the identity of someone who has already signed something. You cannot sign the app with a .cer file alone; it's used for verification, not signing.

.pfx files are also a valid format for storing certificates and private keys. Although it is also another standard, App Center documentation specifies .p12.

.pvk (Private Key) files traditionally contain only the private key. App Center requires both the certificate and the corresponding private key combined in a single file for signing and distribution. It does not support uploading a separate .pvk file along with a certificate file.

In summary, for App Center to distribute your iOS app to a private group, it requires the .p12 file because this file contains the necessary credentials (certificate and private key) for correctly signing the application for deployment. This signing ensures the application is trusted on the user's device and allows the installation to proceed. The other file types lack crucial components necessary for the signing process.

Further Reading:

Microsoft App Center Documentation for iOS Distribution: https://learn.microsoft.com/en-us/appcenter/distribution/ios

Apple Developer Documentation on Code Signing: https://developer.apple.com/support/code-signing/

Question: 169 CertyIQ

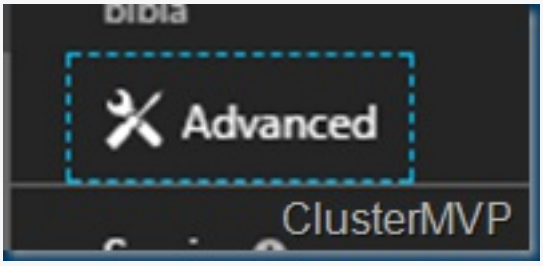
SIMULATION -
You need to prepare a network security group (NSG) named az400-123456789-nsg1 to host an Azure DevOps pipeline agent. The solution must allow only the required outbound port for Azure DevOps and deny all other inbound and outbound access to the Internet.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:

1. Open Microsoft Azure Portal and Log into your Azure account.
2. Select network security group (NSG) named az400-123456789-nsg1
3. Select Settings, Outbound security rules, and click Add
4. Click Advanced



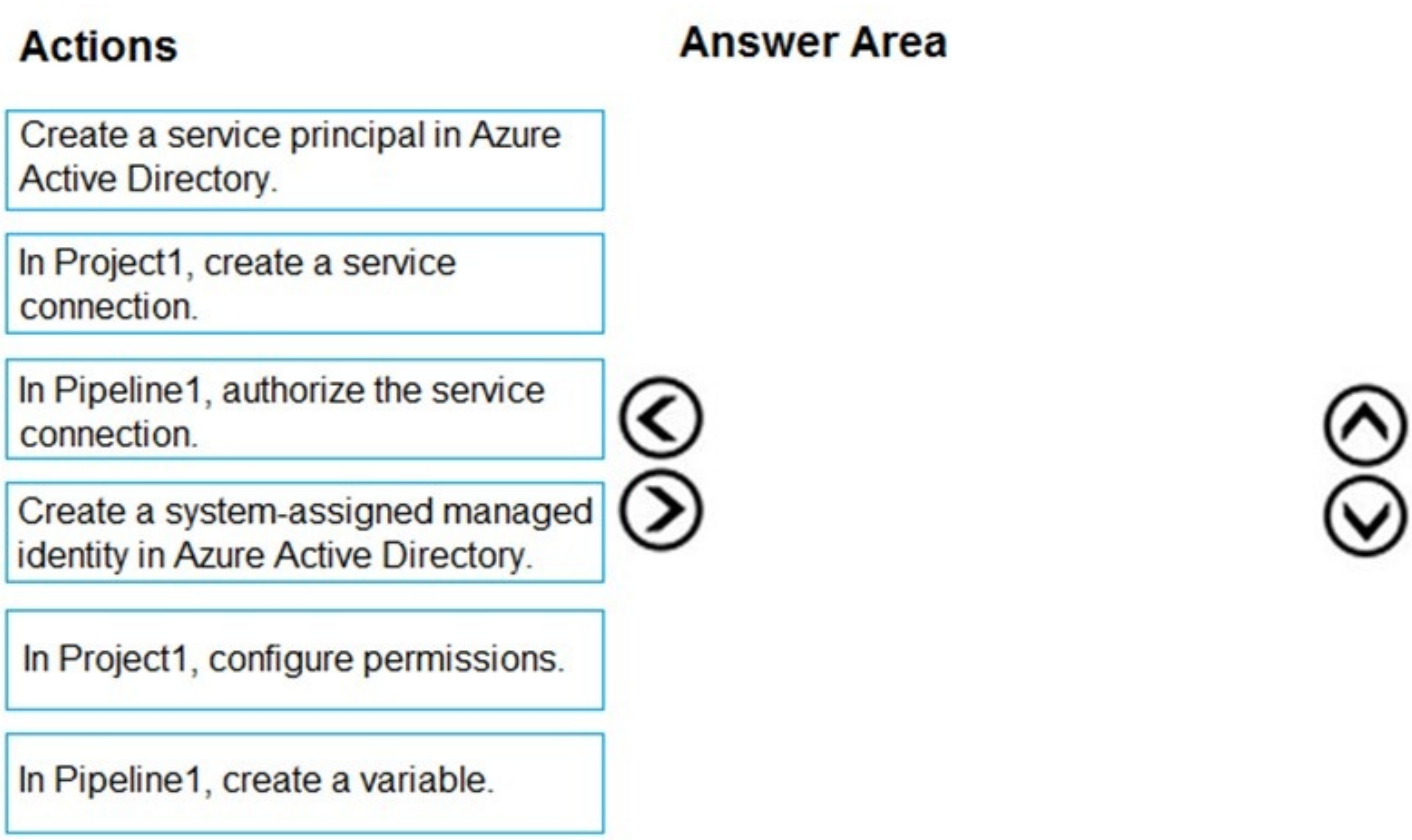
5. Change the following settings:
- Destination Port range: 8080
- Protocol. TCP
- Action: Allow

Note: By default, Azure DevOps Server uses TCP Port 8080.

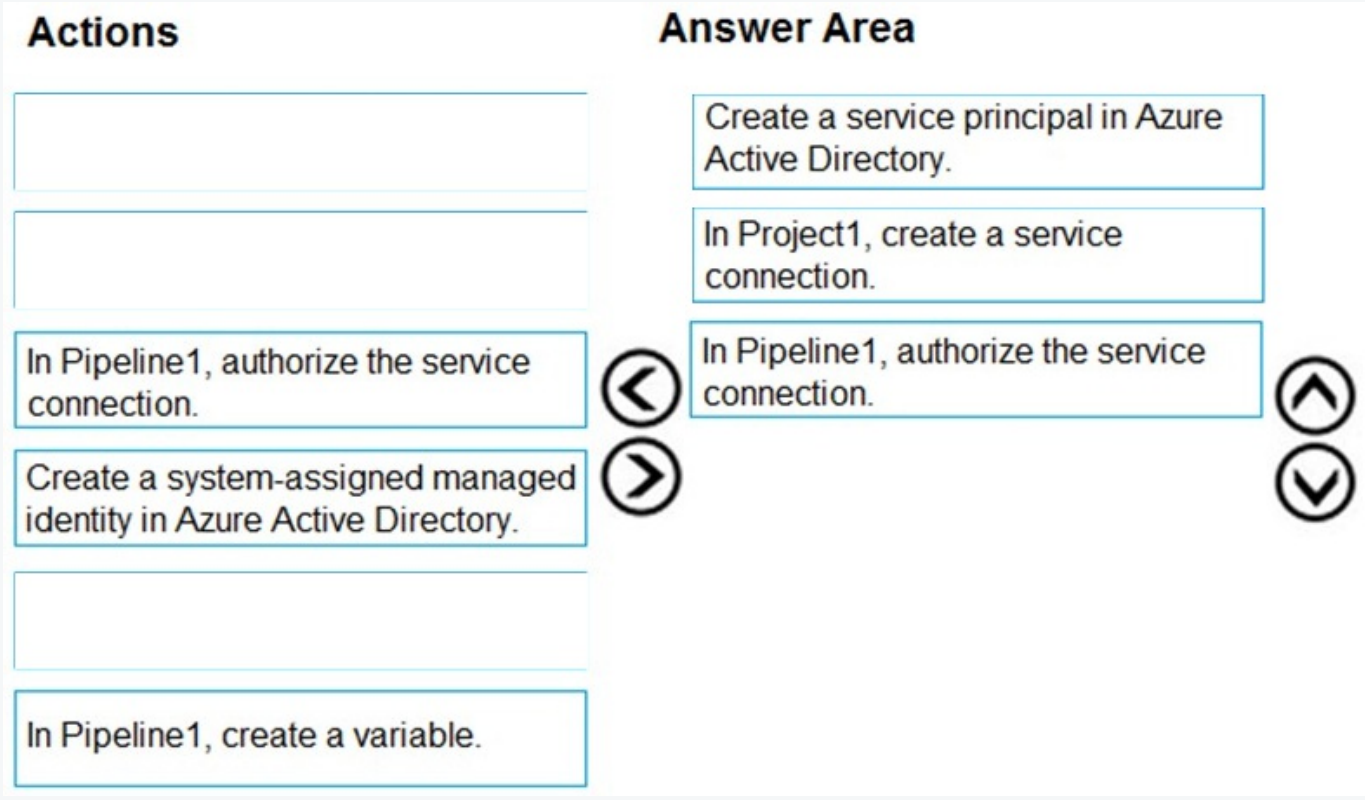
Reference:
https://robertsmit.wordpress.com/2017/09/11/step-by-step-azure-network-security-groups-nsg-security-center-azure-nsg-network/ https://docs.microsoft.com/en-us/azure/devops/server/architecture/required-ports?view=azure-devops

Question: 170 CertyIQ

DRAG DROP -
You have a project in Azure DevOps named Project1 that contains two Azure DevOps pipelines named Pipeline1 and Pipeline2.
You need to ensure that Pipeline1 can deploy code successfully to an Azure web app named webapp1. The solution must ensure that Pipeline2 does not have permission to webapp1.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

1- create a service principle.

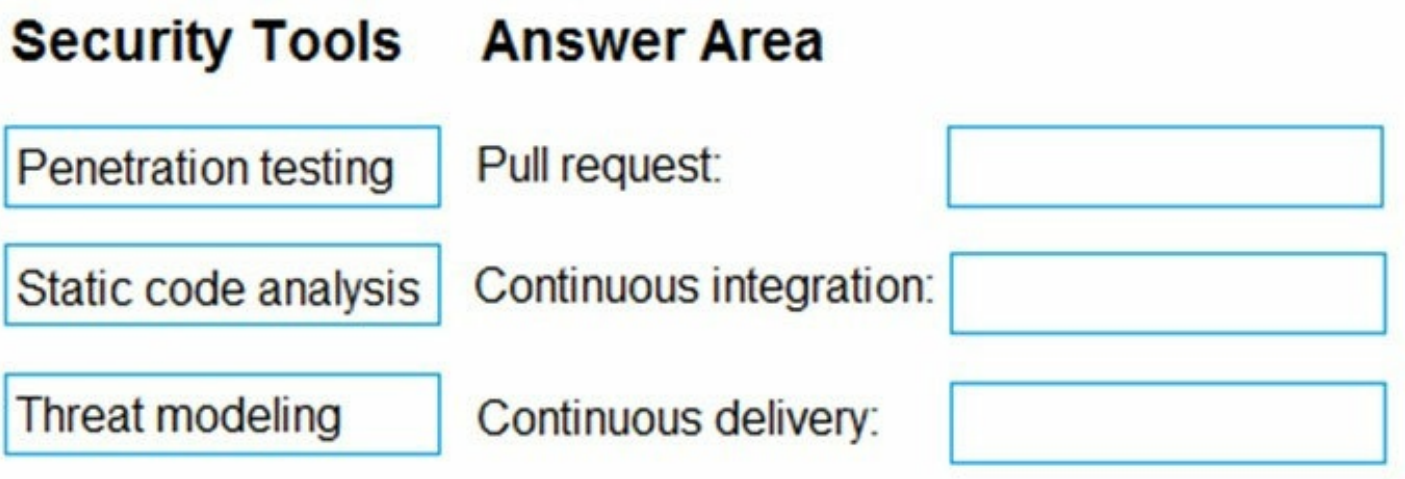
2- in project 1 create a service connection (ARM / Manual) and provide the service principle details created in step 1

3- in pipeline 1, authorize the service connection. this way only pipeline 1 will get access to the webapp and pipeline2 not.

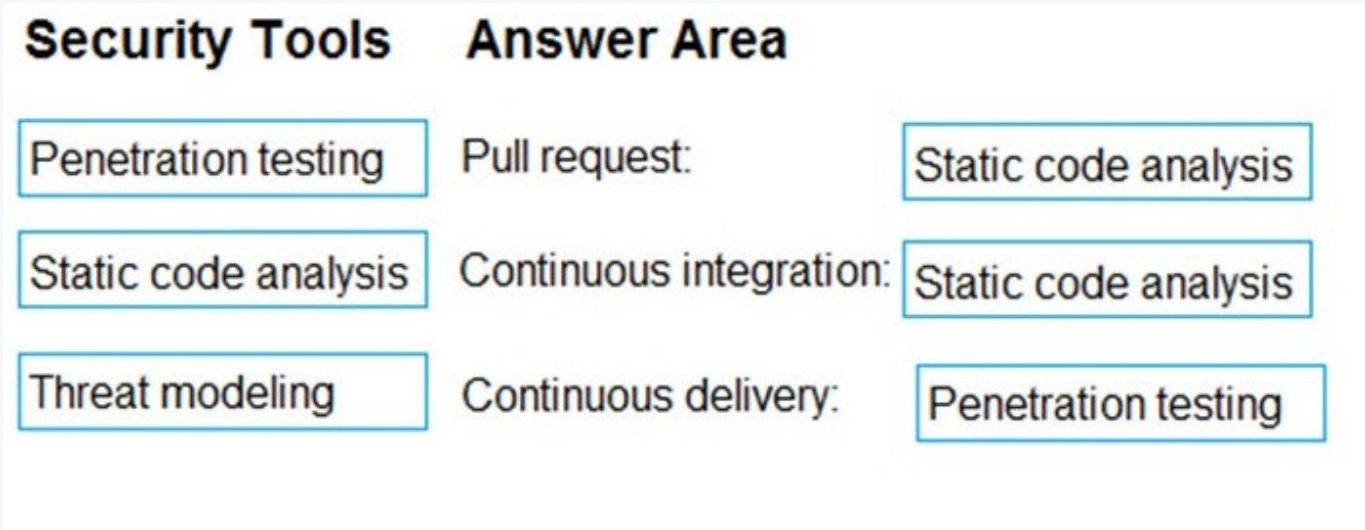
also, project permissions id for users and groups not for pipelines.

Question: 171 CertyIQ

DRAG DROP -
You need to increase the security of your team's development process.
Which type of security tool should you recommend for each stage of the development process? To answer, drag the appropriate security tools to the correct stages. Each security tool may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Static code analysis .

Box 2: Static code analysis .

Validation in the CI/CD begins before the developer commits his or her code. Static code analysis tools in the IDE provide the first line of defense to help ensure that security vulnerabilities are not introduced into the CI/CD process.

Box 3: Penetration testing -

Once your code quality is verified, and the application is deployed to a lower environment like development or QA, the process should verify that there are not any security vulnerabilities in the running application. This can

be accomplished by executing automated penetration test against the running application to scan it for vulnerabilities.

Reference:

https://docs.microsoft.com/en-us/azure/devops/migrate/security-validation-cicd-pipeline?view=azure-devops#ide--pull-request

Question: 172 CertyIQ

Your company is concerned that when developers introduce open source libraries, it creates licensing compliance issues.
You need to add an automated process to the build pipeline to detect when common open source libraries are added to the code base.
What should you use?

A. OWASP ZAP
B. Jenkins
C. Code Style
D. WhiteSource Bolt

Answer: D

Explanation:

The correct answer is D, WhiteSource Bolt. Here's a detailed justification:

WhiteSource Bolt (now Mend Bolt) directly addresses the requirement of detecting open-source license compliance issues during the build pipeline. It's designed to identify open-source components within a codebase and check them against a comprehensive vulnerability and license database. By integrating with the Azure DevOps pipeline, WhiteSource Bolt automatically scans builds, flagging any dependencies with known vulnerabilities or licensing conflicts. This enables early detection and remediation of potential compliance issues, preventing them from propagating further into the software development lifecycle.

OWASP ZAP (Option A) is primarily a web application security scanner, focusing on identifying security vulnerabilities in web applications, not on open-source license compliance. While important for security, it doesn't directly solve the stated licensing issue. Jenkins (Option B) is a general-purpose automation server and could be used to trigger vulnerability or license checks, but it doesn't offer built-in capabilities for identifying open-source components or licenses. Therefore, Jenkins would still require another tool to perform the actual scanning. Code Style (Option C) focuses on enforcing coding standards and formatting, which is entirely unrelated to open-source licensing.

WhiteSource Bolt's tight integration with Azure DevOps makes it the ideal solution for automatically detecting open-source licensing issues within the build pipeline. It offers detailed reports and actionable insights, allowing developers to quickly identify and address any identified risks. Using such a tool automates and streamlines the open-source risk management process, thus lowering the overall licensing compliance risks.

Here are some authoritative links:

Mend Bolt (formerly WhiteSource Bolt) Documentation: https://www.mend.io/white-source-bolt/
Microsoft Azure DevOps: https://azure.microsoft.com/en-us/services/devops/
OWASP ZAP: https://owasp.org/www-project-zap/

Question: 173 CertyIQ

You plan to use a NuGet package in a project in Azure DevOps. The NuGet package is in a feed that requires authentication.
You need to ensure that the project can restore the NuGet package automatically.
What should the project use to automate the authentication?

A. an Azure Automation account
B. an Azure Artifacts Credential Provider
C. an Azure Active Directory (Azure AD) account that has multi-factor authentication (MFA) enabled
D. an Azure Active Directory (Azure AD) service principal

Answer: B

Explanation:

The correct answer is **B. an Azure Artifacts Credential Provider**. Here's a detailed justification:

Azure Artifacts Credential Provider is specifically designed for automatically handling authentication when restoring NuGet packages from authenticated Azure Artifacts feeds. It integrates seamlessly with the NuGet client and Azure Pipelines, handling the credential acquisition process transparently. This eliminates the need to manually manage credentials or bake them into your project or pipeline configuration.

The credential provider automatically obtains authentication tokens from Azure DevOps, ensuring that the NuGet client can successfully restore packages without requiring user intervention or manual credential configuration. This is crucial for automated builds and continuous integration/continuous deployment (CI/CD) pipelines.

Option A, an Azure Automation account, is a more general-purpose service for automating tasks in Azure, but it isn't directly tailored for NuGet package authentication and would require significantly more configuration and code to achieve the same result.

Option C, an Azure AD account with MFA, is inappropriate because requiring MFA in an automated build process creates a significant roadblock. Automated processes cannot complete MFA challenges interactively.

Option D, an Azure AD service principal, can be used for authentication, but it would require manual configuration and management of the service principal's credentials within the build process, which is less secure and more cumbersome than using the dedicated credential provider. The Artifacts Credential Provider handles the service principal configuration behind the scenes when it detects it's running inside of an Azure Pipeline.

Therefore, the Azure Artifacts Credential Provider provides the most streamlined, secure, and recommended approach for automating NuGet package authentication in Azure DevOps.

Supporting Links:

Azure Artifacts Credential Provider: https://github.com/microsoft/azure-devops-credential-provider
Use the Azure Artifacts Credential Provider: https://learn.microsoft.com/en-us/azure/devops/artifacts/nuget/nuget-exe?view=azure-devops#use-the-azure-artifacts-credential-provider

Question: 174 CertyIQ

You use Azure Pipelines to manage project builds and deployments.
You plan to use Azure Pipelines for Microsoft Teams to notify the legal team when a new build is ready for release.

You need to configure the Organization Settings in Azure DevOps to support Azure Pipelines for Microsoft Teams.
What should you turn on?

A. Third-party application access via OAuth
B. Azure Active Directory Conditional Access Policy Validation
C. Alternate authentication credentials
D. SSH authentication

Answer: A

Explanation:

The correct answer is A. Third-party application access via OAuth.

Here's why: Azure Pipelines for Microsoft Teams needs to interact with both Azure DevOps and Microsoft Teams. This interaction involves granting Azure Pipelines permissions to post messages in Teams on behalf of the Azure DevOps organization. OAuth (Open Authorization) is a standard authorization protocol that allows applications to access resources on behalf of a user without requiring the user to share their credentials directly. Turning on "Third-party application access via OAuth" enables Azure DevOps to grant these permissions to Azure Pipelines, facilitating the connection with Microsoft Teams. This is crucial because Azure Pipelines needs to be authenticated and authorized to send notifications to the legal team channel in Microsoft Teams. Disabling this setting would prevent Azure Pipelines from being authorized to communicate with Teams, thus blocking the notification process.

B, Azure Active Directory Conditional Access Policy Validation is about enforcing specific conditions before granting access to Azure AD-protected resources. It's not directly relevant to enabling third-party apps to post notifications in Teams.

C, Alternate authentication credentials allow users to authenticate to Azure DevOps using credentials other than their primary Azure AD credentials. It's not directly related to allowing Azure Pipelines to interact with Microsoft Teams.

D, SSH authentication is used for secure shell access to resources, such as virtual machines or repositories. While it provides a secure method for accessing resources, it is not used for the Azure Pipelines and Microsoft Teams integration.

Therefore, enabling "Third-party application access via OAuth" is essential to permit Azure Pipelines to interact with Microsoft Teams to send notifications about new builds, as this relies on a secure delegated access mechanism.

Relevant Documentation:

Use the Azure Pipelines app for Microsoft Teams: https://learn.microsoft.com/en-us/azure/devops/pipelines/integrations/microsoft-teams?view=azure-devops
OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform: https://learn.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-auth-code-flow

Question: 175 CertyIQ

You have an existing project in Azure DevOps.
You plan to integrate GitHub as the repository for the project.
You need to ensure that Azure Pipelines runs under the Azure Pipelines identity.
Which authentication mechanism should you use?

A. personal access token (PAT)

B. GitHub App
C. Azure Active Directory (Azure AD)
D. OAuth

Answer: B

Explanation:

The correct authentication mechanism to ensure Azure Pipelines runs under the Azure Pipelines identity when integrating GitHub with an Azure DevOps project is a GitHub App.

Here's why:

GitHub Apps offer fine-grained permissions: GitHub Apps provide granular control over the permissions granted to Azure Pipelines to access the GitHub repository. This "least privilege" approach enhances security compared to using a PAT.
GitHub Apps represent an identity: Unlike PATs, which are tied to a specific user, GitHub Apps represent an application identity. This is crucial for Azure Pipelines to operate independently of any individual user account. When the pipeline triggers, it acts on behalf of the GitHub App, not a user.
Improved security: GitHub Apps reduce the risk of access disruption due to user account changes (e.g., employee leaving). If a PAT-owning user leaves the organization, the pipeline will fail. GitHub Apps are independent entities.
Centralized management: GitHub Apps are managed centrally within the GitHub organization or repository settings. This simplifies permission management and auditing.
OAuth is primarily for user authorization: OAuth is designed for delegating access to user resources (e.g., allowing an application to access a user's GitHub profile). It's not ideal for application-to-application authentication in this scenario where Azure Pipelines needs to operate independently.
Azure AD primarily authenticates users: Azure AD is for user authentication and authorization within the Azure ecosystem, but not directly applicable for application-to-application authentication between Azure Pipelines and GitHub.
PATs are tied to a user account: Personal Access Tokens (PATs) grant access to GitHub resources based on the permissions of the user who created the PAT. This is less secure and harder to manage than using a GitHub App because if the user's permissions change, the pipeline could break. Additionally, the pipeline's identity is tied to a person, not the Azure Pipeline resource.

In summary, using a GitHub App provides a more secure, manageable, and robust way for Azure Pipelines to authenticate and interact with GitHub under its own application identity.

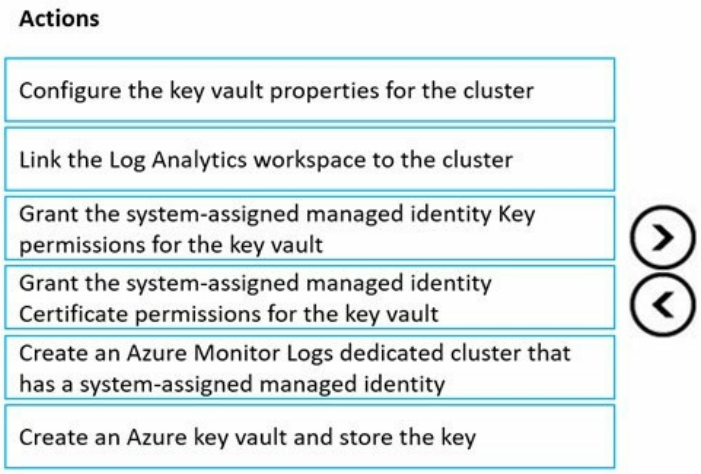
Relevant links for further research:

About creating GitHub Apps:
Azure DevOps Services connections for GitHub:
Authentication Methods in Azure DevOps:

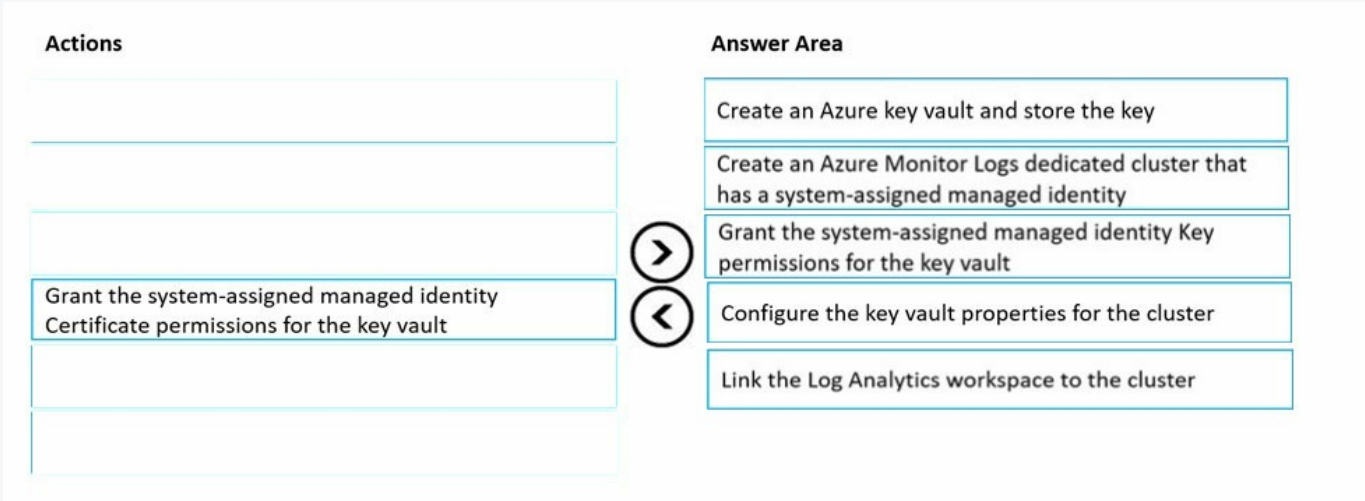
Question: 176 CertyIQ

DRAG DROP -
You have an Azure subscription that uses Azure Monitor and contains a Log Analytics workspace.
You have an encryption key.
You need to configure Azure Monitor to use the key to encrypt log data.
Which five actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.

Select and Place:



Answer:



Explanation:

Customer-Managed key provisioning steps:

Step 1: Create an Azure Key vault and store the key.

Creating Azure Key Vault and storing key. Create or use an existing Azure Key Vault in the region that the cluster is planed, and generate or import a key to be used for logs encryption.

Step 2: Create an Azure Monitor Logs dedicate cluster that has a system-assigned managed identity

Clusters uses managed identity for data encryption with your Key Vault. Configure identity type property to SystemAssigned when creating your cluster to allow access to your Key Vault for "wrap" and "unwrap" operations.

Step 3: Grant the system-assigned managed Identity Key permissions for the key vault.

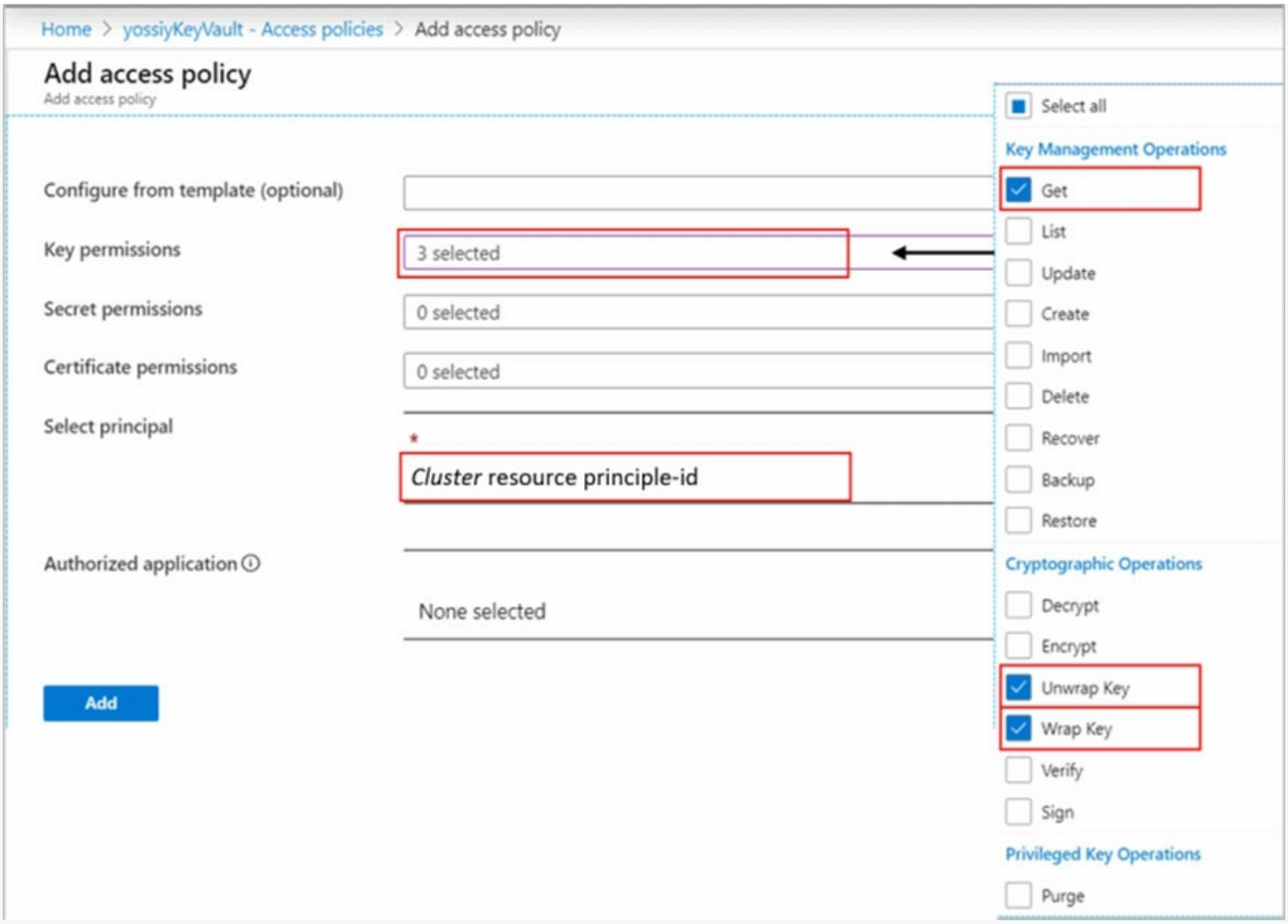
Grant Key Vault permissions.

Create Access Policy in Key Vault to grants permissions to your cluster. These permissions are used by the underlay cluster storage. Open your Key Vault in

Azure portal and click Access Policies then + Add Access Policy to create a policy with these settings:

Key permissions"select Get, Wrap Key and Unwrap Key.

Etc.



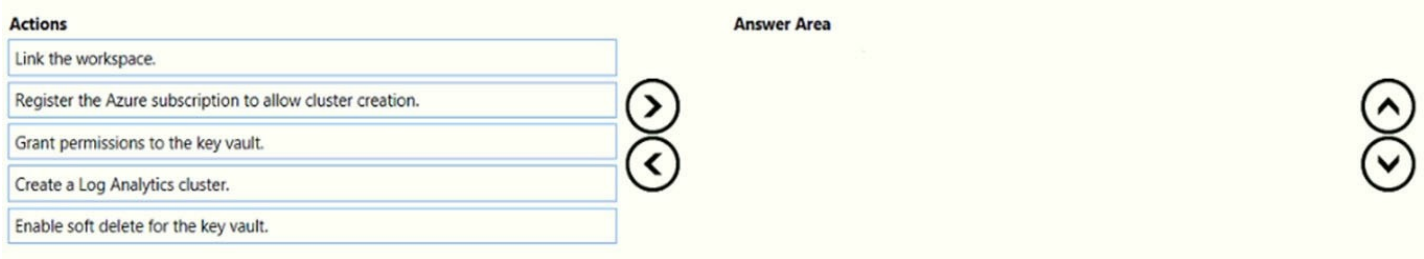
1. Creating cluster
2. Granting permissions to your Key Vault
3. Updating cluster with key identifier details
4. Linking workspaces

Step 4: Configure the key vault properties for the cluster.
Update cluster with key identifier details.
Step 5: Link the Log Analytics workspace to the cluster
Link workspace to cluster.
This step should be performed only after the cluster provisioning. If you link workspaces and ingest data prior to the provisioning, ingested data will be dropped and won't be recoverable.

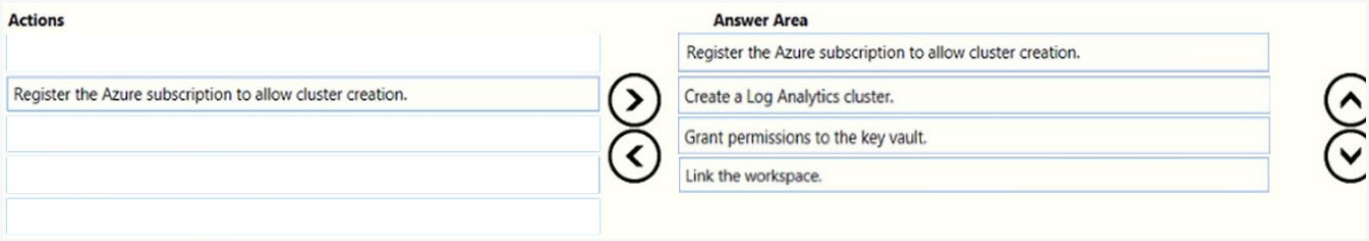
Reference:
https://docs.microsoft.com/en-us/azure/azure-monitor/logs/customer-managed-keys

Question: 177 CertyIQ

DRAG DROP -
You have an Azure Key Vault that contains an encryption key named key1.
You plan to create a Log Analytics workspace that will store logging data.
You need to encrypt the workspace by using key1.
Which four actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Step 1: Register the Azure subscription to allow cluster creation.

Step 2: Create a Log Analytics cluster.

Clusters uses managed identity for data encryption with your Key Vault. Configure identity type property to SystemAssigned when creating your cluster to allow access to your Key Vault for "wrap" and "unwrap" operations.

Step 3: Grant permissions to the key vault.

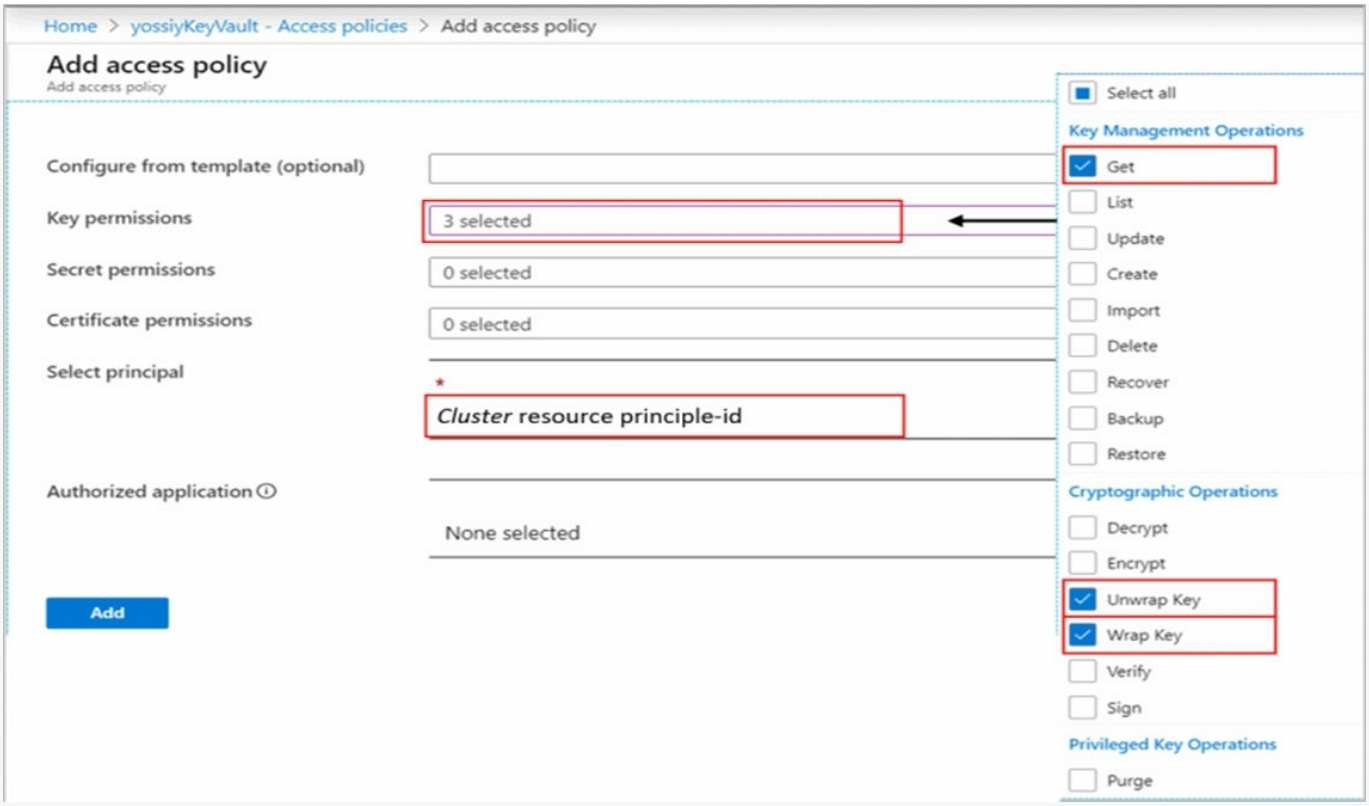
Grant Key Vault permissions.

Create Access Policy in Key Vault to grants permissions to your cluster. These permissions are used by the underlay cluster storage. Open your Key Vault in

Azure portal and click Access Policies then + Add Access Policy to create a policy with these settings:

Key permissions"select Get, Wrap Key and Unwrap Key.

Etc.



1. Creating cluster

2. Granting permissions to your Key Vault

3. Updating cluster with key identifier details

4. Linking workspaces

Step 4: Link workspace -

Link workspace to cluster.

This step should be performed only after the cluster provisioning. If you link workspaces and ingest data prior to the provisioning, ingested data will be dropped and won't be recoverable.

Question: 178 CertyIQ

You use release pipelines in Azure Pipelines to deploy an app. Secrets required be the pipeline are stored as pipeline variables. Logging of commands is enabled for the Azure Pipelines agent.
You need to prevent the values of the secrets from being logged.
What should you do?

A. Store the secrets in the environment variables instead of the pipeline variables.
B. Pass the secrets on the command line instead of in the pipeline variables.
C. Apply a prefix of secret to the name of the variables.
D. Echo the values of the secrets to the command line.

Answer: A

Explanation:

The correct answer is A: Store the secrets in the environment variables instead of the pipeline variables.

Here's a detailed justification:

Azure Pipelines' primary mechanism for managing sensitive information is through variables, but logging can inadvertently expose these values if not handled carefully. When agent command logging is enabled (which is common for debugging), any command that uses a pipeline variable might have its value printed to the logs. This poses a security risk for secrets.

While variables are generally convenient, environment variables provide a superior approach for handling secrets in this specific scenario. Azure Pipelines automatically masks secret environment variables in logs. This means that even if logging is enabled, the values of the secrets stored as environment variables will be replaced with asterisks or other masking characters when printed in the logs. This prevents accidental exposure of the actual secret values.

Options B, C, and D are incorrect or could worsen the situation:

B. Pass the secrets on the command line instead of in the pipeline variables: Passing secrets directly on the command line is extremely insecure. Command-line arguments are frequently logged by the system, making this a very poor security practice. It does not use Azure's built-in secret management capabilities.
C. Apply a prefix of secret to the name of the variables: While some systems use a naming convention for secrets, Azure Pipelines does not automatically mask variables based on a prefix like "secret". This does not hide the variable's value in the logs.
D. Echo the values of the secrets to the command line: This is the opposite of what you should do. Explicitly echoing the values to the command line ensures that the secrets are written to the logs, completely defeating the purpose of protecting them.

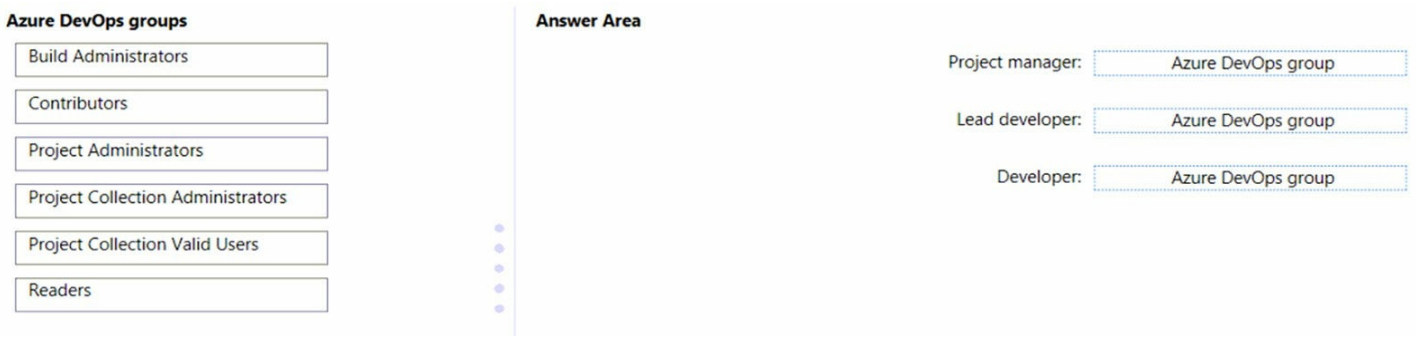
Using environment variables leverages Azure Pipeline's built-in masking, offering a more robust and reliable security measure against unintentional secret leakage in logs. It also integrates better with other Azure services that are designed to handle sensitive data in secure and auditable ways. Therefore, storing secrets in environment variables addresses the need to prevent secrets from being logged when agent command logging is enabled.

Supporting documentation can be found here:

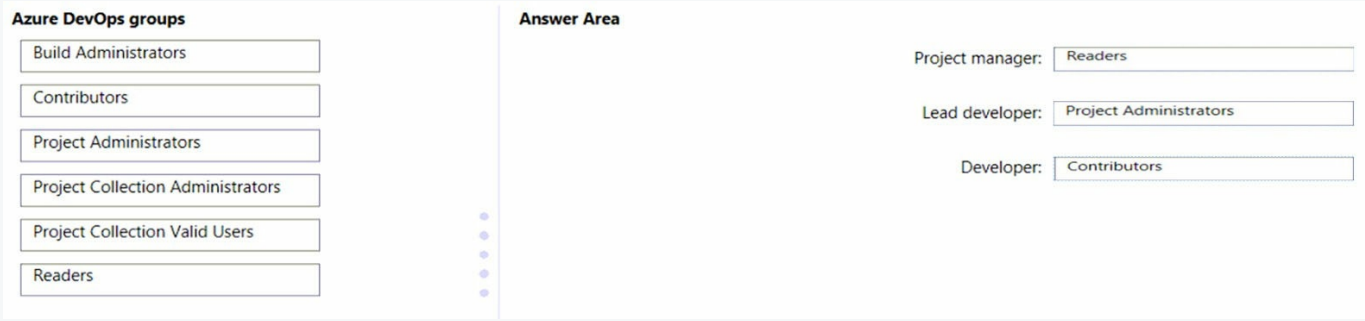
Azure Pipelines Variables: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/variables?view=azure-devops&tabs=yaml%2Cbatch (While this page doesn't explicitly discuss the log masking of environment variables, it establishes the role of variables in pipelines, providing necessary context.)
Define variables: Secure and secret variables: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/variables?view=azure-devops&tabs=yaml%2Cbatch#secret-variables (Though this focuses on secret variables stored in Azure DevOps UI / Variable Groups, the key takeaway is Azure DevOps does have security mechanisms to protect secrets.)

Question: 179 CertyIQ

DRAG DROP -
You need to deploy a new project in Azure DevOps that has the following requirements:
* The lead developer must be able to create repositories, manage permissions, manage policies, and contribute to the repository.
* Developers must be able to contribute to the repository and create branches, but NOT bypass policies when pushing builds.
* Project managers must only be able to view the repository.
* The principle of least privilege must be used.
You create a new Azure DevOps project team for each role.
To which Azure DevOps groups should you add each team? To answer, drag the appropriate groups to the correct teams. Each group may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Readers -
Project managers must only be able to view the repository.
Only read permission necessary.

Box 2: Project Administrators -
The lead developer must be able to create repositories, manage permissions, manage policies, and contribute to the repository.
Add to the Project Collection Administrators security group users tasked with managing organization or collection resources.

Box 3: Contributors -
Developers must be able to contribute to the repository and create branches, but NOT bypass policies when pushing builds.
Add to the Contributors security group full-time workers who contribute to the code base or manage projects.

Reference:
https://docs.microsoft.com/en-us/azure/devops/organizations/security/look-up-project-collection-administrators

Question: 180

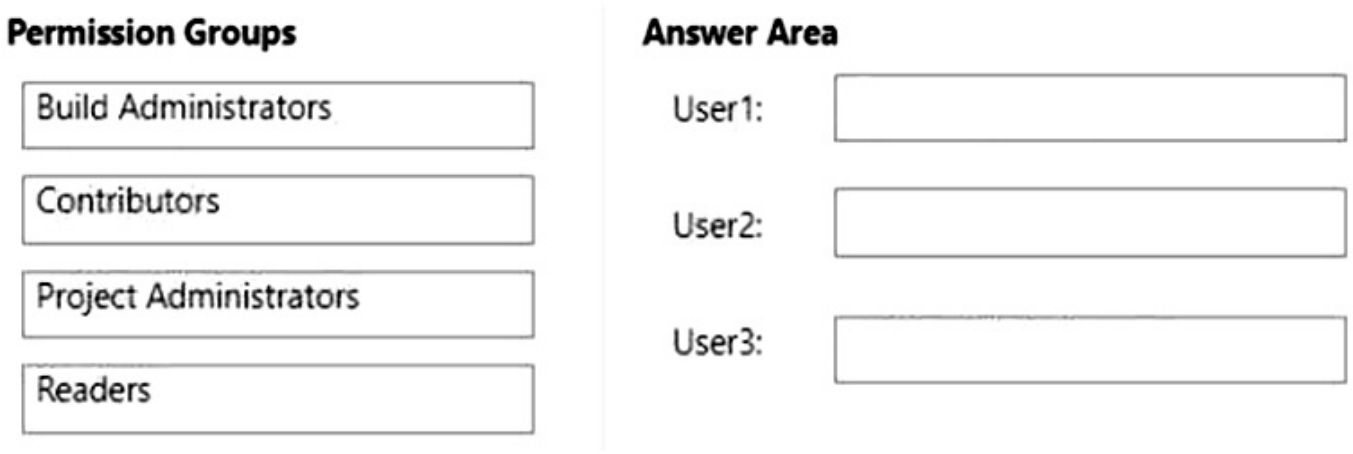
CertyIQ

DRAG DROP -
You have an Azure subscription that contains a project in Azure DevOps named Project1. You have three Azure Active Directory (Azure AD) users that require access to Project1 as shown in the following table.

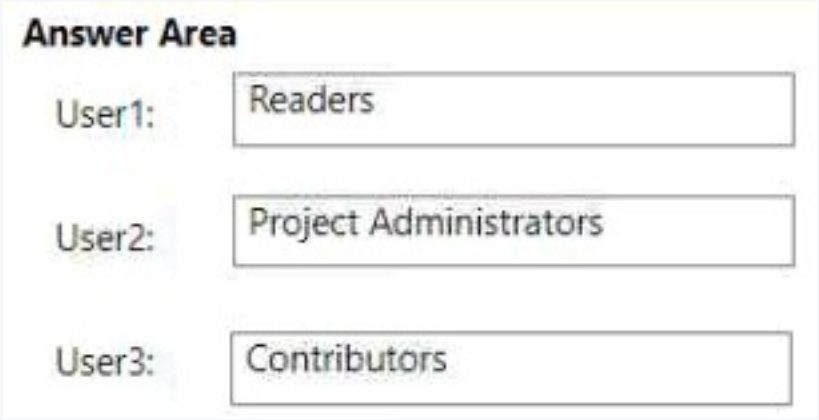
Name	Title	Requirement
User1	Project Manager	View repositories.
User2	Development Lead	Create repositories and manage permissions.
User3	Developer	Create branches and tags.

You need to ensure that the users have the appropriate permissions. The solution must use the principle of least privilege.

To which permission group in Azure DevOps should you add each user? To answer, drag the appropriate permission groups to the correct users. Each permission group may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Readers: Can view work items and dashboards but cannot change them.

Project Administrators: Have full control over project-level settings, security, and all assets.

Contributors: Can create, edit, and update work items and manage their own team's dashboards.

Question: 181 CertyIQ

You have an Azure subscription linked to an Azure Active Directory Premium Plan 1 tenant.
A security review indicates that too many users have privileged access to resources.
You need to deploy a privileged access management solution that meets the following requirements:
• Enforces time limits on the use of privileged access
• Requires approval to activate privileged access
• Minimizes costs
What should you do first?

A. Configure notifications when privileged roles are activated.
B. Configure alerts for the activation of privileged roles.
C. Enforce Azure Multi-Factor Authentication (MFA) for role activation.
D. Upgrade the license of the Azure Active Directory (Azure AD) tenant.

Answer: D

Explanation:

The correct answer is D, upgrading the Azure Active Directory (Azure AD) tenant license. Here's why:

Privileged Identity Management (PIM), which is essential for implementing the requested privileged access management features like time limits and approval workflows, is a feature primarily available in Azure AD Premium P2, not Premium P1. The question explicitly states the current subscription is linked to an Azure AD Premium P1 tenant. Without upgrading to Premium P2, the core requirements of enforcing time limits and requiring approval cannot be met natively.

Options A, B, and C are all configurations within PIM. They are actions you would take after you have the necessary licensing. They are useless until you're on the proper license tier. You can't configure notifications, alerts, or MFA for role activation if you can't define the role activation process using PIM in the first place, and you cannot define this process without a Premium P2 license.

Therefore, upgrading the license is the foundational first step. Once the tenant is upgraded, PIM can be configured to enforce time limits, require approvals, and provide better control over privileged access, thereby addressing the security review's concerns and fulfilling the specified requirements while striving to minimize costs (by strategically allocating the necessary licenses). After the upgrade, MFA should be enabled for these privileged roles.

Authoritative Links:

What is Azure AD Privileged Identity Management?: https://docs.microsoft.com/en-us/azure/active-directory/privileged-identity-management/pim-configure
Azure Active Directory feature availability: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-features

Question: 182 CertyIQ

You plan to create a GitHub workflow that will use GitHub Actions. The actions will require a 256-KB secret.
You need to recommend a solution to store and encrypt the secret. The secret value must be accessible only to the workflow. The solution must minimize administrative effort
What should you recommend?

A. Store the secret in the organization-level GitHub secrets.

B. Store the secret in the repository-level GitHub secrets.

C. Encrypt the secret value and store the value in the repository. Store the decryption key in the repository-level GitHub secrets.

D. Encrypt the secret value and store the value in the repository. Store the decryption key in the organization-level GitHub secrets.

Answer: C

Explanation:

Here's a detailed justification for why option C is the best solution for storing and encrypting a 256-KB secret for a GitHub workflow, minimizing administrative effort, and ensuring access is restricted to the workflow itself:

Requirement of Workflow-Only Access: The primary constraint is that the secret must be accessible only to the workflow. This rules out organization-level secrets (Options A and D) because organization-level secrets are available to all repositories within the organization, violating the principle of least privilege and increasing

the potential attack surface.

Size Constraint: While GitHub Actions secrets can technically handle a 256KB value, encrypting the secret adds a layer of protection, particularly if the repository might contain sensitive information alongside the code. This is a "defense in depth" approach.

Minimizing Administrative Effort: Storing the encrypted secret within the repository directly fulfills the requirement of keeping the sensitive data tied to the project using it. Storing the decryption key as a repository-level secret keeps everything self-contained and manageable at the repository level. If the organization moves, this key moves too. The administrator only has to be concerned with the access control of the repository.

Why Option C is Superior: By encrypting the secret value and storing it in the repository alongside the decryption key (as a repository-level secret), you achieve the following:

Enhanced Security: The secret is encrypted, providing an extra layer of protection against unauthorized access even if the repository is compromised.
Scoped Access: The decryption key (repository-level secret) ensures that only workflows within that specific repository can decrypt and use the secret.
Minimized Effort: Management and rotation of secrets is contained within the repository itself.

Why other options are inferior:

Option A and D: Organization-level secrets are overly broad and contradict the requirement of workflow-only access.
Option B: storing the secret directly as repository secret is unsafe.

In summary: Option C balances security, access control, and ease of management while adhering to the principle of least privilege and defense in depth. Encrypting the sensitive data, storing it, and storing the key with the minimum scope provides a safer method.

Relevant Links:

GitHub Actions Secrets: https://docs.github.com/en/actions/security-guides/encrypted-secrets
GitHub Security Best Practices: https://docs.github.com/en/code-security/security-best-practices/about-security-best-practices

Question: 183 CertyIQ

You have a GitHub Enterprise account.
You need to enable push protection for secret scanning of the account repositories.
What should you do first?

A. Purchase a GitHub Advanced Security license.
B. Purchase Premium Plus support.
C. Enforce multi-factor authentication (MFA).
D. Create an access policy for secrets.

Answer: A

Explanation:

The correct answer is A: Purchase a GitHub Advanced Security license.

Here's why: GitHub Advanced Security (GHAS) is a suite of security features built into GitHub Enterprise that

enhances code security and vulnerability detection. Push protection for secret scanning is a specific feature included within GHAS. Without GHAS, the secret scanning functionality will be limited and push protection will not be available.

Push protection intercepts secrets before they're committed to a repository. This proactive measure prevents sensitive data like API keys, passwords, and tokens from inadvertently being exposed in your code history, reducing the risk of security breaches and data leaks.

The other options are incorrect because:

B: Premium Plus support primarily provides enhanced support services and doesn't directly enable security features like secret scanning and push protection.
C: While enforcing multi-factor authentication (MFA) strengthens account security by adding an extra layer of authentication, it doesn't unlock the features within GitHub Advanced Security like push protection. MFA is an essential security practice in general, but it's not the prerequisite for secret scanning.
D: Access policies manage who can access specific resources. They don't directly enable secret scanning or push protection; these are features controlled by the GHAS license.

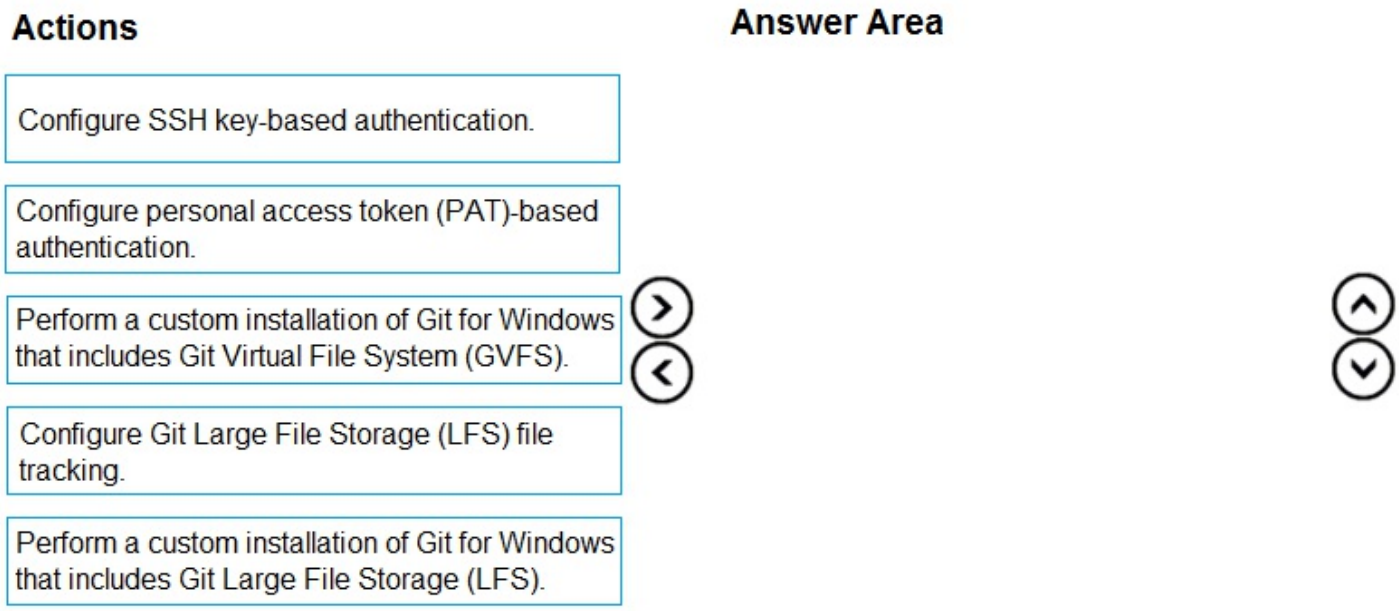
Therefore, the initial step to enable push protection for secret scanning is to acquire a GitHub Advanced Security license.

References:

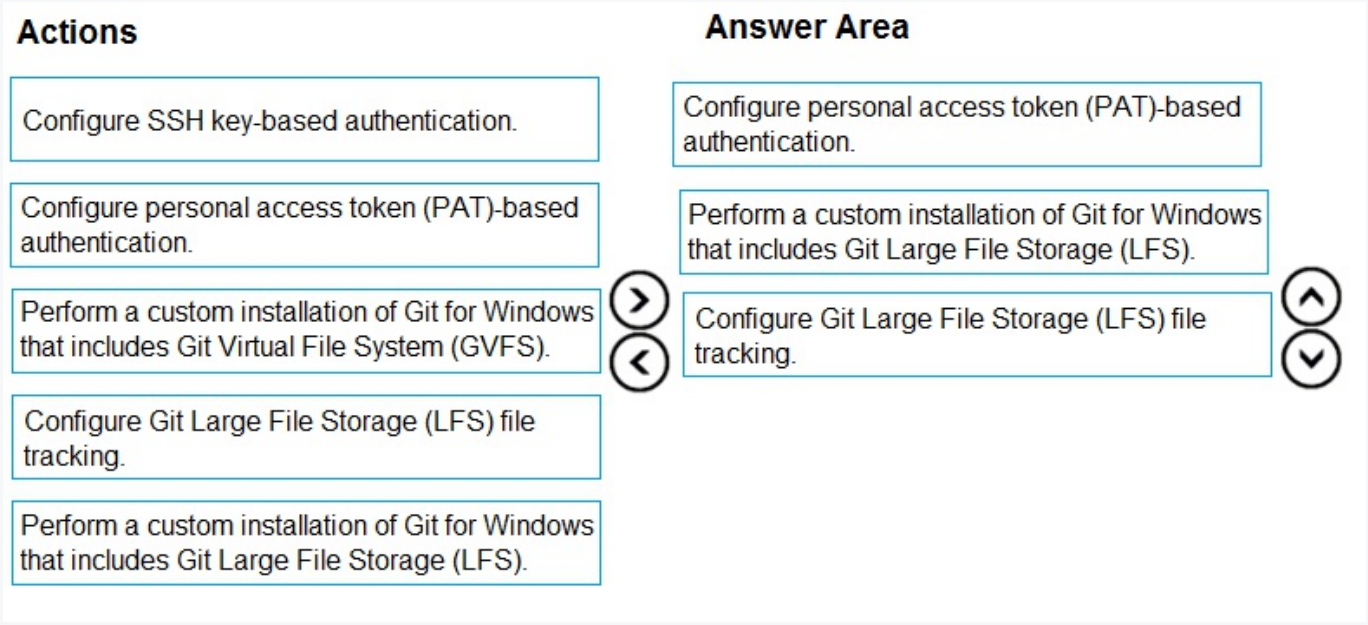
GitHub Advanced Security Overview
Secret Scanning
Push Protection

Question: 184 CertyIQ

DRAG DROP -
Your company has a project in Azure DevOps named Project1.
All the developers at the company have Windows 10 devices.
You need to create a Git repository for Project1. The solution must meet the following requirements:
• Support large binary files.
• Store binary files outside of the repository.
• Use a standard Git workflow to maintain the metadata of the binary files by using commits to the repository.
Which three actions should you perform in sequence on each developer's device? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



Answer:



Explanation:

Configure PAT

Perform installation of Git LFS

Track GIT LFS

https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-git-large-file-storage

Question: 185 CertyIQ

Your company uses Azure DevOps and Microsoft Azure Active Directory (Azure AD), part of Microsoft Entra.
Only users who have accounts in Azure AD can access the Azure DevOps environment.
You need to ensure that only devices that are connected to the on-premises network can access the Azure DevOps environment.
What should you do?

A.Assign the Stakeholder access level to all users.
B.In Azure DevOps, configure Security in Project Settings.
C.In Azure AD, configure conditional access.
D.In Azure AD, configure risky sign-ins.

Answer: C

Explanation:

The correct answer is **C. In Azure AD, configure conditional access.**

Here's a detailed justification:

Conditional Access in Azure AD is a powerful tool that allows administrators to enforce access control policies based on various conditions. One such condition is the network location from which a user is attempting to access resources. By configuring Conditional Access, you can specifically restrict access to Azure DevOps (and other cloud applications integrated with Azure AD) to only users who are accessing from

the on-premises network.

Here's how it works:

1. **Define a Named Location:** First, you define a named location in Azure AD that represents your on-premises network. This is typically done by specifying the public IP address range(s) associated with your company's internet egress points.
2. **Create a Conditional Access Policy:** You then create a Conditional Access policy that targets the Azure DevOps application.
3. **Configure Conditions:** Within the policy, you specify that access is only allowed when the user's request originates from the named location (your on-premises network).
4. **Configure Access Controls:** You set the grant control to require that the user be on the defined network location. If a user attempts to access Azure DevOps from outside the on-premises network, the policy will block the access request.

Why other options are incorrect:

A. Assign the Stakeholder access level to all users: Access level controls what features a user can use in Azure DevOps but it does not restrict network location. It is irrelevant to the stated requirement.
B. In Azure DevOps, configure Security in Project Settings: Azure DevOps security settings primarily control access to specific projects, repositories, pipelines, or features within Azure DevOps. While security within Azure DevOps is important, it doesn't directly address the requirement of restricting access based on the network location of the user.
D. In Azure AD, configure risky sign-ins: Risky sign-ins are related to detecting and responding to potentially compromised accounts or suspicious login activity. While important for overall security, they don't directly enforce access restrictions based on network location.

In conclusion, Conditional Access in Azure AD is the most suitable and directly applicable solution to enforce the requirement of restricting Azure DevOps access to only devices connected to the on-premises network. It allows you to define network-based access controls, ensuring that only authorized users on the company network can access the Azure DevOps environment.

Authoritative Links:

What is Conditional Access in Azure Active Directory?: https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/overview
Common Conditional Access policies: https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/howto-conditional-access-policy-common
Location condition in Azure Active Directory Conditional Access: https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/location-condition

Question: 186 CertyIQ

You have an Azure subscription that contains four Azure virtual machines.
You need to configure the virtual machines to use a single identity. The solution must meet the following requirements:
•Ensure that the credentials for the identity are managed automatically.
•Support granting privileges to the identity.
Which type of identity should you use?

A.a system-assigned managed identity
B.a user-assigned managed identity
C.a service principal
D.a user account

Answer: B

Explanation:

The correct answer is **B. a user-assigned managed identity**. Here's a detailed justification:

The requirement to use a single identity across multiple virtual machines significantly points towards using a user-assigned managed identity. System-assigned managed identities are unique to each Azure resource (in this case, each VM). Therefore, using system-assigned identities would require configuring and managing four separate identities, violating the "single identity" constraint.

User-assigned managed identities, on the other hand, are created as standalone Azure resources. These identities can then be assigned to multiple Azure resources, such as virtual machines, providing them with a shared identity. This fulfills the single identity requirement.

Both system-assigned and user-assigned managed identities automate credential management, eliminating the need to store and rotate secrets manually. Azure handles the lifecycle of the identity's credentials, which adheres to the requirement of automated credential management. Furthermore, both identity types support granting privileges (Azure RBAC roles) to the identity, enabling you to control what resources the VMs can access.

Service principals, while used for application authentication, are typically associated with applications and services rather than virtual machines representing user workloads. They also require more manual management compared to managed identities. User accounts would necessitate manually managing credentials, which contradicts the requirement of automated credential management.

In summary, a user-assigned managed identity is the best choice because it allows the creation of a single, reusable identity that can be assigned to multiple VMs, ensuring automated credential management and support for granting privileges, therefore fulfilling all requirements.

Authoritative Links:

What are managed identities for Azure resources?: https://learn.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/overview
Create, list, delete, or assign a user-assigned managed identity using the Azure portal: https://learn.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/how-to-manage-ua-identity-portal

Question: 187 CertyIQ

HOTSPOT
-
You have an Azure subscription that contains an Azure key vault named Vault1, an Azure pipeline named Pipeline1, and an Azure SQL database named DB1.
Pipeline1 is used to deploy an app that will authenticate to DB1 by using a password.
You need to store the password in Vault1. The solution must ensure that the password can be accessed by Pipeline1.
What should you do? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

Store the password as a: Certificate / Key / Secret

Grant Pipeline1 access to Vault1 by modifying the: Access control (IAM) settings / Access policies / Security settings

Answer:

Answer Area



Explanation:

1. Secret2. Access policy

https://learn.microsoft.com/en-us/azure/key-vault/general/assign-access-policy?tabs=azure-portalA Key Vault access policy determines whether a given security principal, namely a user, application or user group, can perform different operations on Key Vault secrets, keys, and certificates. You can assign access policies using the Azure portal, the Azure CLI, or Azure PowerShell.

Question: 188 CertyIQ

You have a GitHub repository that contains multiple workflows and a secret stored at the environment level.
You need to ensure that the secret can be used by all the workflows.
What should you do first?

A.Recreate the secret at the organization level.

B.Recreate the secret at the repository level.

C.Enable required reviewers.

Answer: B

Explanation:

The correct answer is B: Recreate the secret at the repository level. Here's why:

The goal is to make an environment-level secret accessible to all workflows within a single GitHub repository. Secrets configured at the environment level are, by default, scoped to that specific environment and are not

automatically available to all workflows in the repository.

Option A, recreating the secret at the organization level, would make the secret available to all repositories within the organization. This is a broader scope than required and potentially exposes the secret to unintended workflows in other repositories, violating the principle of least privilege.

Option B, recreating the secret at the repository level, is the most direct and secure approach to achieve the desired outcome. Repository secrets are available to all workflows within that repository without needing any specific configuration within the individual workflows. This limits the scope of the secret only to the necessary workflows, adhering to best practices for security and access control.

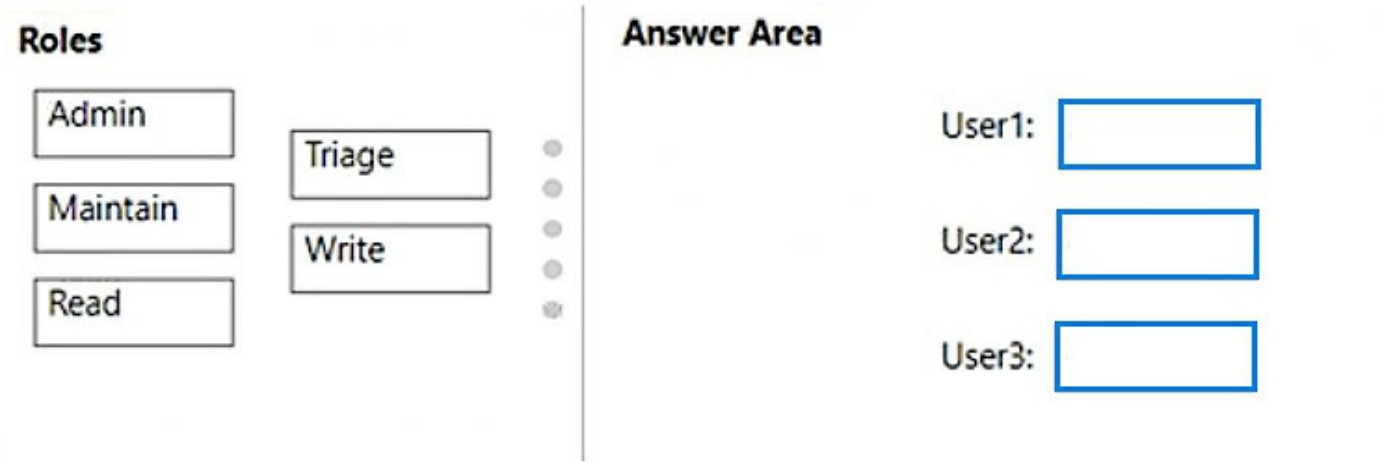
Option C, enabling required reviewers, is a security practice aimed at improving code quality and preventing unauthorized changes to the codebase. While vital for software development lifecycles, required reviewers don't provide a mechanism to access secrets from within the workflows.

Therefore, recreating the secret at the repository level is the optimal solution to ensure the secret is accessible to all workflows within the repository without unnecessarily widening the scope of access.

For further research, refer to the official GitHub documentation on Secrets:
https://docs.github.com/en/actions/security-guides/encrypted-secrets

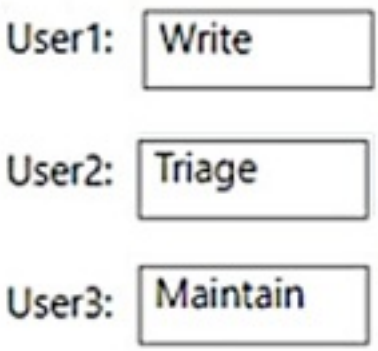
Question: 189 CertyIQ

DRAG DROP
-
You have a GitHub organization that contains three users named User1, User2, and User3. You have a project that contains a repository named repo1.
You need to configure permissions for repo1. The solution must meet the following requirements:
•Ensure that User1 can actively push to repo1.
•Ensure that User2 can manage issues and pull requests for repo1.
•Ensure that User3 can manage repo1.
•Prevent User3 from accessing sensitive data in repo1.
Which role should you assign to each user? To answer, drag the appropriate roles to the correct users. Each role may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:

Answer Area



Question: 190 CertyIQ

You have an Azure key vault named KV1 and three web servers.
You plan to deploy an app named App1 to the web servers.
You need to ensure that App1 can retrieve a secret from KV1. The solution must meet the following requirements:
•Minimize the number of permission grants required.
•Follow the principle of least privilege.
What should you include in the solution?

A.role-based access control (RBAC) permission
B.a system-assigned managed identity
C.a user-assigned managed identity
D.a service principal

Answer: C

Explanation:

The correct answer is **C. a user-assigned managed identity**. Here's why:

Managed identities are a feature in Azure Active Directory (Azure AD) that provides Azure services with an automatically managed identity to authenticate to Azure services that support Azure AD authentication, without needing to embed credentials in code. This aligns with the principle of least privilege and reduces the risk of credential exposure.

System-assigned managed identities (Option B) are tied to a specific resource (in this case, each web server). While they could be used, they would require granting KV1 access to three separate system-assigned identities (one for each server), increasing the number of permission grants. This doesn't minimize permission grants as efficiently.

User-assigned managed identities (Option C), on the other hand, exist as standalone Azure resources. You can create a single user-assigned managed identity and assign it to all three web servers. Then, you grant this single managed identity permission to access KV1. This minimizes the number of permission grants, satisfying the requirement. Also, User-assigned managed identity provides more flexibility as it is not tied to the lifecycle of a specific resource.

RBAC permissions (Option A) are used to manage access to Azure resources at a broader level. While you could use RBAC, it wouldn't be as specific or secure as using a managed identity specifically for secret retrieval. RBAC roles generally grant wider permissions than necessary for this specific task.

A service principal (Option D) is an identity created in Azure AD for use with applications, services, and

automation tools. While technically possible, using a managed identity is generally preferred for resources hosted within Azure as it simplifies credential management and rotation.

Therefore, using a single user-assigned managed identity and assigning it to the web servers is the most efficient and secure approach to grant access to KV1, minimizing permission grants while adhering to the principle of least privilege.

Supporting Documentation:

Managed Identities Overview: https://learn.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/overview
User-assigned Managed Identities: https://learn.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/how-to-manage-ua-identity-vm
Key Vault Authentication: https://learn.microsoft.com/en-us/azure/key-vault/general/authentication

Question: 191 CertyIQ

You plan to use Azure DevOps to build and deploy an app that will be hosted in a Kubernetes cluster.
You need to scan the app image for vulnerabilities before the image is deployed to the cluster.
What should you include in the solution?

A.Microsoft Defender for Containers
B.Microsoft Defender for App Service
C.Microsoft Defender for DevOps
D.Microsoft Defender for Storage

Answer: A

Explanation:

The correct answer is **A. Microsoft Defender for Containers**.

Here's a detailed justification:

The requirement is to scan the app image for vulnerabilities before deployment to a Kubernetes cluster. This directly points to a container-focused security solution. Let's analyze why the other options are not as suitable:

B. Microsoft Defender for App Service: App Service is a platform for hosting web applications and APIs. While Defender for App Service helps protect these resources, it doesn't primarily focus on scanning container images before deployment to a Kubernetes environment. It's more concerned with the runtime security of applications running on App Service.

C. Microsoft Defender for DevOps: Defender for DevOps is broader, aiming to secure the entire DevOps lifecycle, from code to cloud. It includes features like code scanning and infrastructure as code (IaC) scanning. However, it's not specifically tailored to scan container images for vulnerabilities immediately before Kubernetes deployment. While it might incorporate image scanning, it's not the primary tool for this purpose.

D. Microsoft Defender for Storage: Defender for Storage is designed to protect Azure Storage accounts from malicious activities. It has no direct relevance to scanning container images for vulnerabilities within a Kubernetes deployment pipeline.

Microsoft Defender for Containers, on the other hand, is specifically designed to protect containerized

environments, including Kubernetes clusters. It provides vulnerability scanning capabilities that can be integrated into the CI/CD pipeline. This means that before an image is deployed to the Kubernetes cluster, Defender for Containers can analyze it for known vulnerabilities and provide alerts or prevent the deployment if critical issues are found.

The process usually involves integrating Defender for Containers with container registries like Azure Container Registry (ACR). When a new image is pushed to ACR, Defender for Containers can automatically trigger a scan. The scan results are then used to determine whether the image meets the security requirements before being deployed to Kubernetes. This proactive approach helps prevent vulnerable images from running in the cluster, reducing the risk of security breaches.

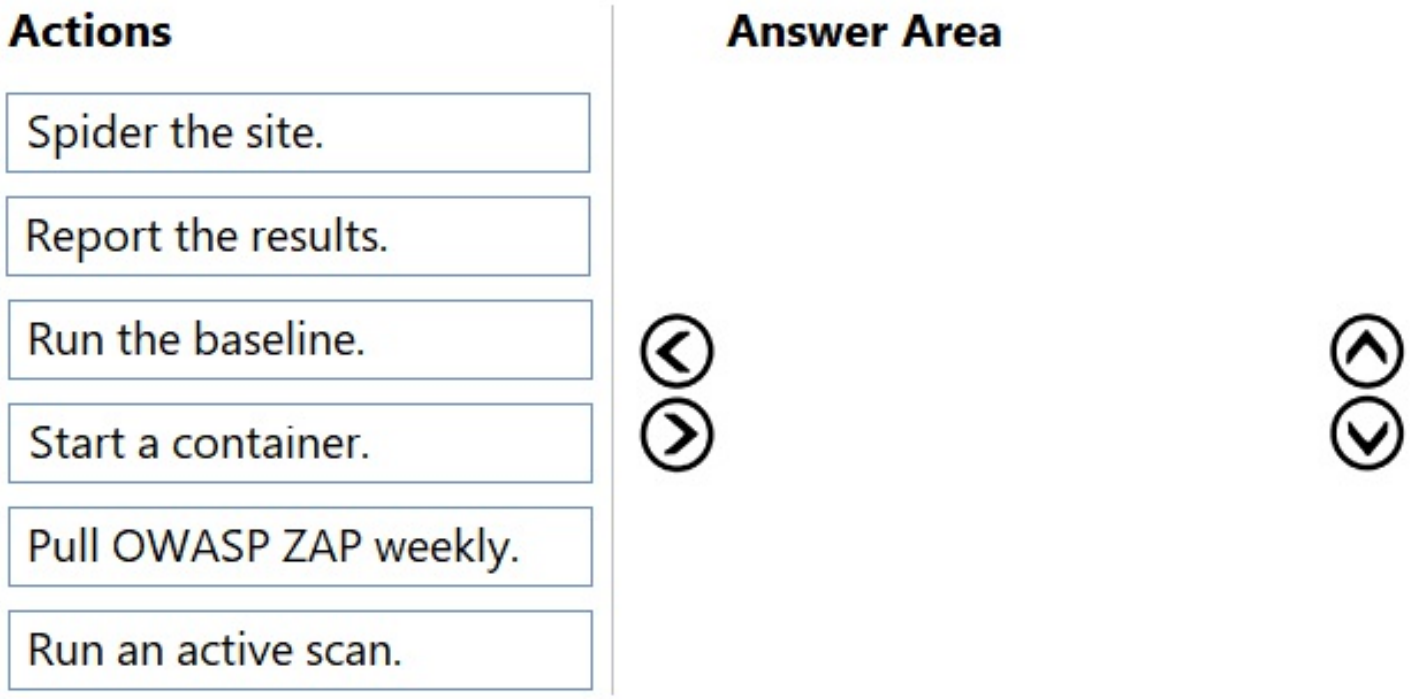
Therefore, given the need to scan images for vulnerabilities before deployment to Kubernetes, Microsoft Defender for Containers is the most appropriate and directly applicable solution.

Authoritative Links:

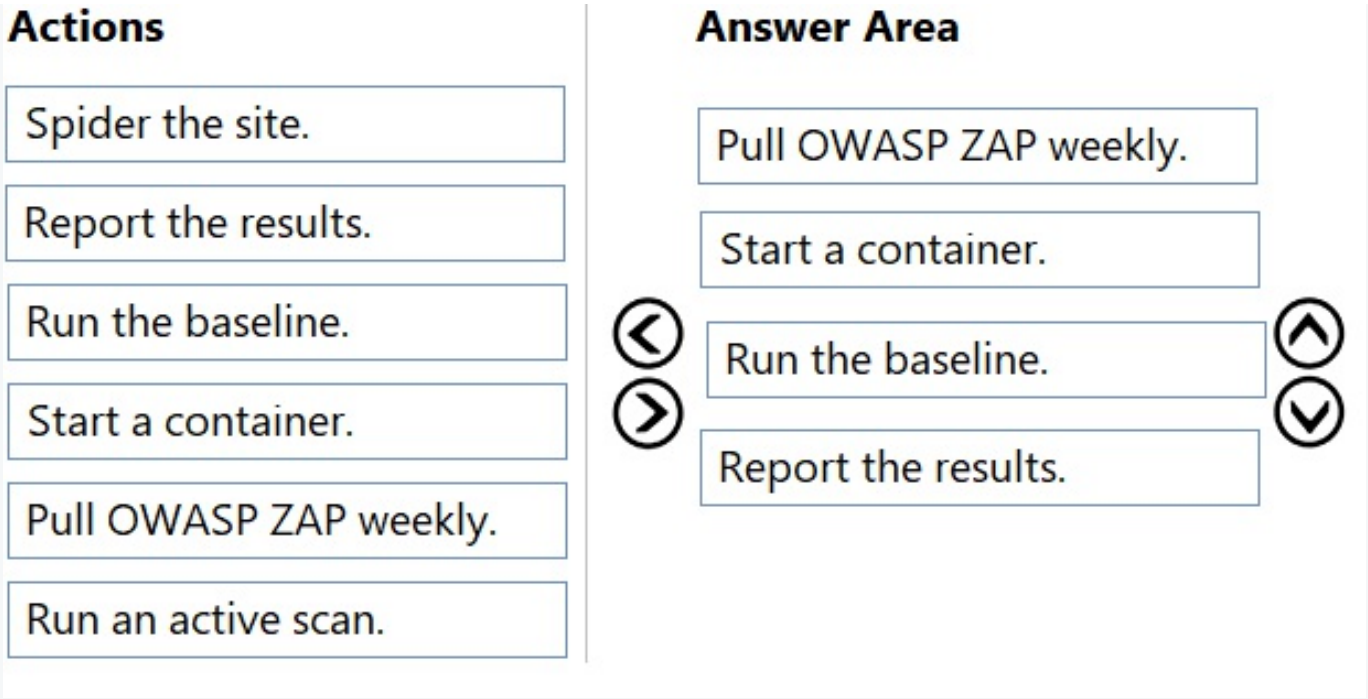
Microsoft Defender for Containers: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-containers-introduction
Integrate Defender for Containers with ACR: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-containers-registry

Question: 192 CertyIQ

DRAG DROP
-
You have an Azure Pipelines application CI/CD pipeline named Pipeline1.
You need to add OWASP ZAP testing to Pipeline1.
Which four actions should you add to Pipeline1 in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



Answer:



Explanation:

Pull OWASP ZAP Weekly.

Start a Container.

Run the baseline .

Report the Result.

Question: 193 CertyIQ

You have an app named App1 that is built by using Azure Pipelines. The source code for App1 is stored in Azure Repos and contains open source libraries.
You need to identify security vulnerabilities in the open source code.
What should you use?

A.Mend Bolt
B.Rollbar
C.Code Climate
D.DeepSource

Answer: A

Explanation:

The correct answer is A, Mend Bolt (formerly WhiteSource Bolt). Here's why:

Mend Bolt directly addresses the requirement of identifying security vulnerabilities in open-source code within an Azure DevOps pipeline. It's specifically designed to scan your Azure Repositories for open-source components and detect known vulnerabilities and license compliance issues. When it identifies such vulnerabilities, it alerts the development team, enabling them to address these issues early in the development lifecycle. This proactive approach helps prevent vulnerable code from being deployed, reducing the risk of security breaches and related incidents.

Options B, C, and D, while valuable tools, do not specifically focus on identifying open-source security vulnerabilities within the Azure DevOps pipeline in the same way as Mend Bolt. Rollbar is primarily for error tracking and monitoring in production environments. Code Climate focuses on code quality and maintainability, and DeepSource helps in identifying potential bugs and code smells through static analysis, but neither excels at open-source security vulnerability detection.

Mend Bolt offers seamless integration with Azure DevOps pipelines, making it an ideal choice for automatically scanning code repositories and flagging vulnerabilities. Its focus is on maintaining the security and compliance of open-source components, making it the most relevant tool for the given scenario.

Justification Summary: Mend Bolt specifically targets open-source security vulnerabilities within Azure Repos, directly addressing the stated need, and integrates well with Azure Pipelines.

Supporting Links:

Mend (formerly WhiteSource) Bolt for Azure DevOps: https://www.mend.io/free-developer-tools/whitesource-bolt/
Azure DevOps Extension - Mend Bolt: https://marketplace.visualstudio.com/items?itemName=whitesource.ws-bolt

Question: 194 CertyIQ

You manage code by using GitHub.
You plan to use Dependabot to scan for code dependencies.
You need to identify when scanning will be triggered automatically.
Which two actions will trigger a scan? Each correct answer presents a complete solution.
NOTE: Each correct solution is worth one point.

A.The dependency graph of a repository changes.
B.A pull request is created.
C.A branch is forked.
D.Any commit is pushed.
E.A new advisory is added.

Answer: AE

Explanation:

Here's a breakdown of why options A and E are correct, and why the others aren't, regarding Dependabot triggering scans in a GitHub repository:

A. The dependency graph of a repository changes. This is a correct trigger. Dependabot relies on the dependency graph to understand the dependencies within your project. When the graph changes – for example, due to a commit that updates a package.json or pom.xml file – Dependabot needs to rescan to identify any new vulnerabilities associated with the updated dependencies. The dependency graph provides Dependabot with the crucial data it needs to find vulnerable dependencies.

E. A new advisory is added. This is also a correct trigger. Dependabot's core function is to identify and alert you to vulnerabilities in your dependencies. If a new security advisory is published regarding a dependency used in your project, Dependabot will automatically trigger a scan to determine if your repository is affected and create a pull request to update the dependency if needed. This proactive scanning helps in maintaining the security posture of the application.

Let's look at why the other options are incorrect:

B. A pull request is created. While a pull request might trigger a scan indirectly if it modifies dependencies, the creation of a PR alone isn't a direct trigger for a Dependabot scan. The key is whether the PR changes dependency information.

C. A branch is forked. Forking a repository creates a separate copy. Dependabot configurations within the original repository are independent of the forked repository. A fork itself won't initiate a scan in the original repository.

D. Any commit is pushed. While commits that change dependency files will trigger a scan via the dependency graph change (option A), any commit won't. Dependabot is specifically interested in dependency-related changes, not every single code modification. Therefore, a commit that only modifies documentation or application code, without altering any dependency files, wouldn't initiate a Dependabot scan directly.

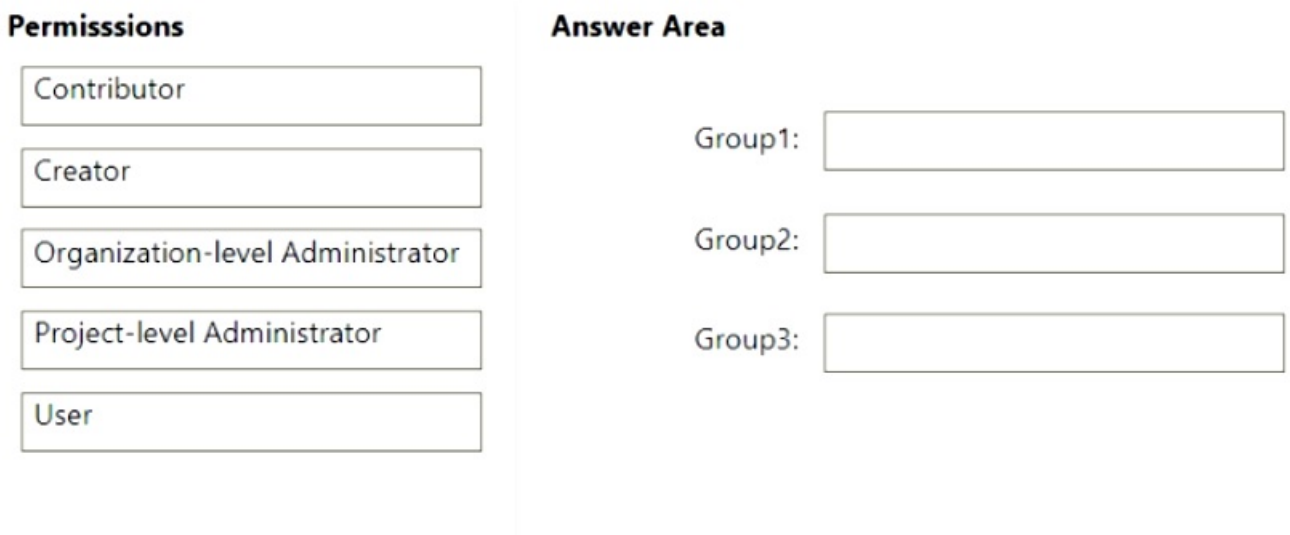
In summary, Dependabot's automated scans are initiated by events that directly impact its ability to assess dependency security. Changes to the dependency graph (A) and the publication of new security advisories (E) are the primary triggers for these scans.

Authoritative Links:

About Dependabot alerts: https://docs.github.com/en/code-security/dependabot/dependabot-alerts/about-dependabot-alerts
About Dependabot security updates: https://docs.github.com/en/code-security/dependabot/dependabot-security-updates/about-dependabot-security-updates
Configuring Dependabot version updates: https://docs.github.com/en/code-security/dependabot/dependabot-version-updates/configuring-dependabot-version-updates

Question: 195 CertyIQ

DRAG DROP
-
You have a tenant in Microsoft Azure Active Directory (Azure AD), part of Microsoft Entra. The tenant contains three groups named Group1, Group2, and Group3.
You create a new project in Azure DevOps named Project1.
You need to secure the service connections for Project1. The solution must meet the following requirements:
•The members of Group1 must be able to share and unshare a service connection with other projects.
•The members of Group2 must be able to rename a service connection and update the description.
•The members of Group3 must be able to use the service connection within build or release pipelines.
•The principle of least privilege must be followed.
Which permission should you grant to each group? To answer, drag the appropriate permissions to the correct groups. Each permission may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Group1: Organization -level Administrator.

Group2: Project-level Administrator.

Group3: User

The organization-level Administrator can do the following administrative tasks:

- Manage organization-level users

- Edit all the fields of a service connection

- Share and unshare a service connection with other projects

https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml#organization-level-permissions

The project-level Administrator can do the following tasks:

- Manage other users and roles at the project-level

- Rename a service connection and update the description

- Delete a service connection, which removes it from the project

https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml#project-level-permissions

A User can:

- Use the service connection when authoring build or release pipelines or authorize yaml pipelines

https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml#user-permissions

Question: 196 CertyIQ

DRAG DROP
-
You have an Azure Repos repository named Repo1 that is used for source control.
You need to configure code scanning for Repo1.
Which three tasks should the pipeline perform in sequence? To answer, move the appropriate tasks from the list of tasks to the answer area and arrange them in the correct order.

Tasks	Answer Area
Advanced Security Dependency Scanning	
Advanced Security Initialize CodeQL	
Advanced Security AutoBuild	
Microsoft Purview Scan	
Advanced Security Perform CodeQL Analysis	

Answer:

Answer Area

Advanced Security Initialize CodeQL

Advanced Security Dependency Scanning

Advanced Security Perform CodeQL Analysis

Question: 197 CertyIQ

SIMULATION
-
You need to create a personal access token (PAT) named Token1 that has only the following capabilities:
•Read, write, and manage code
•Read and execute builds
•Read releases
Token1 must expire in 60 days.

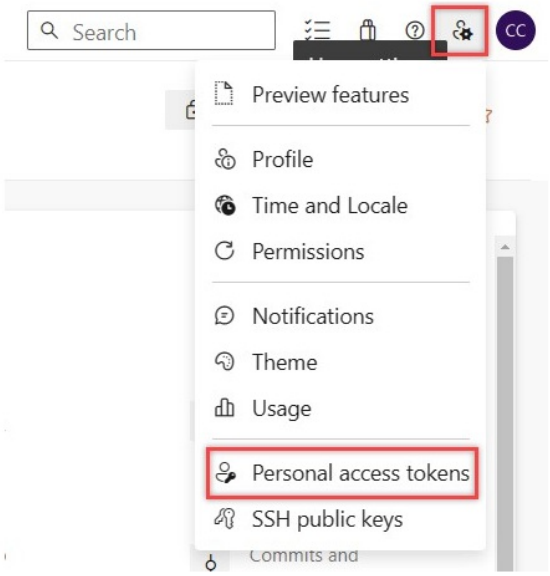
Answer:

Azure DevOps, Use personal access tokens

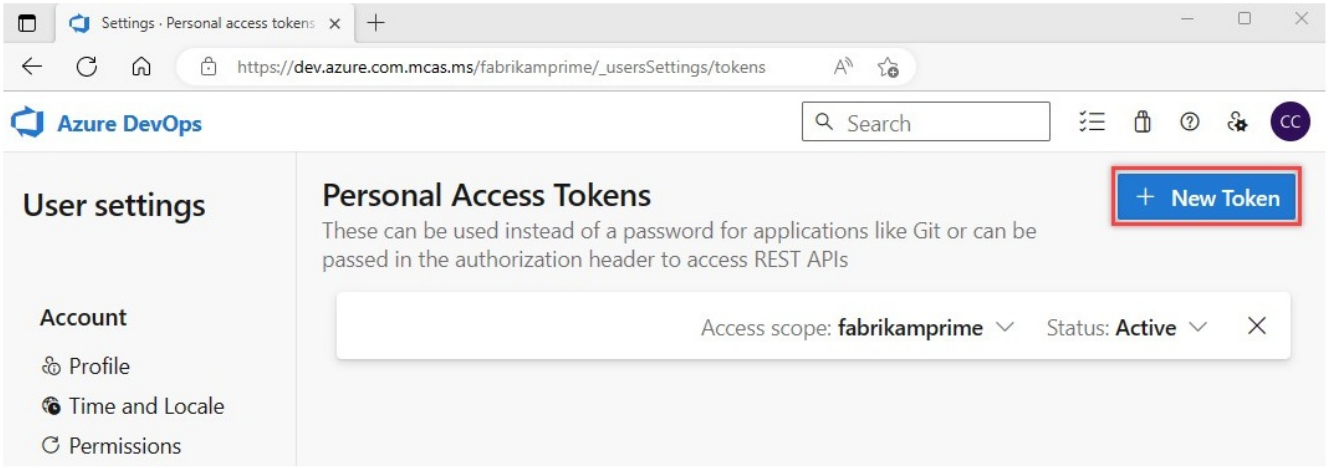
Create a PAT

Step 1: Sign in to your organization (https://dev.azure.com/{Your_Organization}).

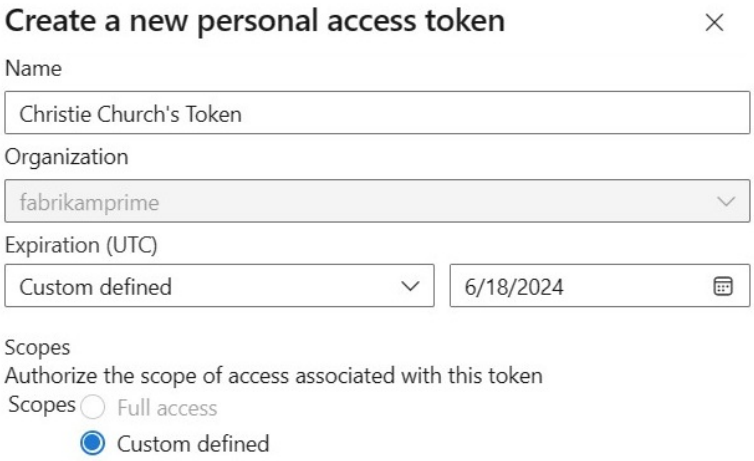
Step 2: From your home page, open user settings and select Personal access tokens.



Step 3: Select + New Token.

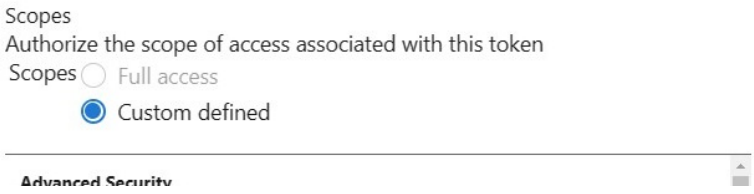


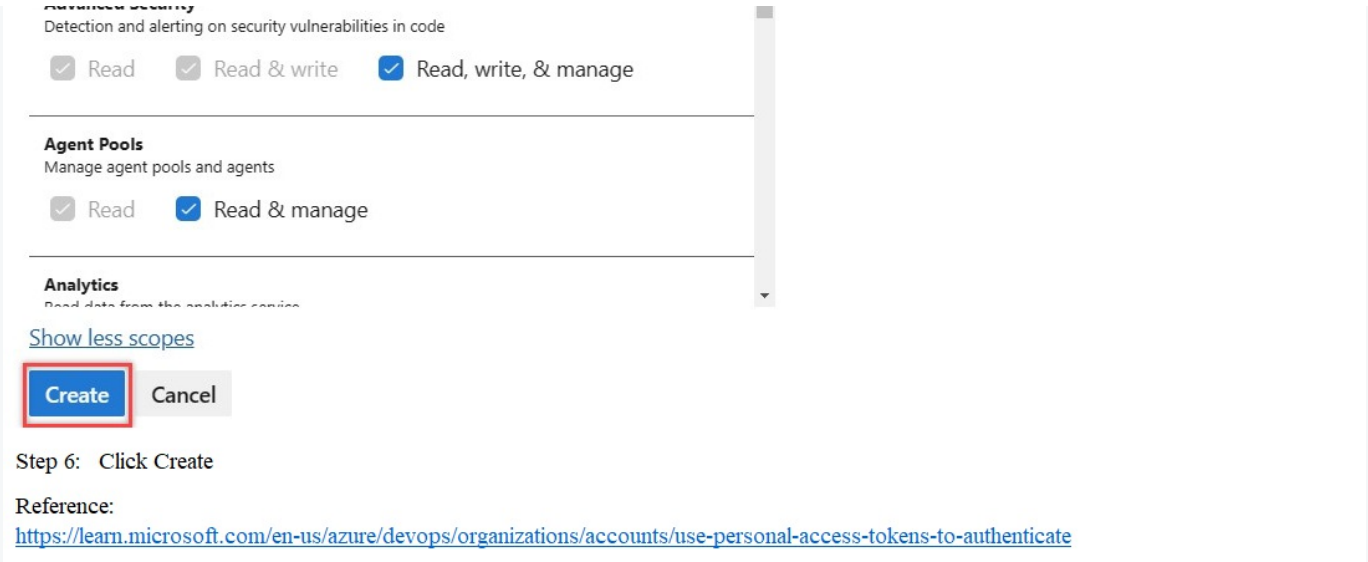
Step 4: Name your token [Token1], select the organization where you want to use the token, and then set your token to automatically expire after a set number of days [Set expire after 60 days].



Step 5: Select the scopes for this token to authorize for your specific tasks.

[Select:
Read, write, and manage code
Read and execute builds
Read releases]





Step 6: Click Create

Reference:
https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate

Question: 198 CertyIQ

You have a GitHub repository that uses GitHub Actions and stores access keys by using GitHub encrypted secrets.
You plan to update the secrets by using the GitHub REST API.
You need to wrap the secrets before adding them to a REST-based call.
Which encryption library should you use?

A.CryptoNet
B.BouncyCastle
C.libsodium
D.hashlib

Answer: C

Explanation:

The correct answer is **C. libsodium**.

Here's a detailed justification:

GitHub's REST API for managing secrets requires that secrets be encrypted using a specific public key provided by GitHub before they are uploaded. This encryption ensures that the secrets are protected in transit and at rest within GitHub's infrastructure. The only way to encrypt secrets that are sent to GitHub using the REST API is by encrypting the secret with the specific public key that has been provided by GitHub.

libsodium is a modern, easy-to-use crypto library that provides the necessary primitives for public-key cryptography, specifically authenticated encryption. It's designed to be secure and well-audited, making it a suitable choice for handling sensitive data like secrets.

libsodium provides functions for generating and using public/private key pairs and functions for encrypting data with the public key. Therefore, it aligns perfectly with GitHub's requirement for encrypting secrets before uploading them.

The public key can be accessed using the GET /repos/ owner / repo /actions/secrets/public-key. This public key is required for using libsodium.

CryptoNet is a generic term referring to cryptographic network protocols and doesn't represent a specific library suitable for encrypting data with a public key in the required format. BouncyCastle is a general-

purpose cryptography library, primarily used in Java and C#. While it can perform encryption, it is not specifically tailored for the encryption standards that GitHub uses. hashlib primarily provides hashing algorithms, which are one-way functions and cannot be used for encryption where the original data needs to be recovered.

libsodium is the prescribed tool for encrypting the information because its function is to encrypt secrets using GitHub's public key before sending them through the REST API.

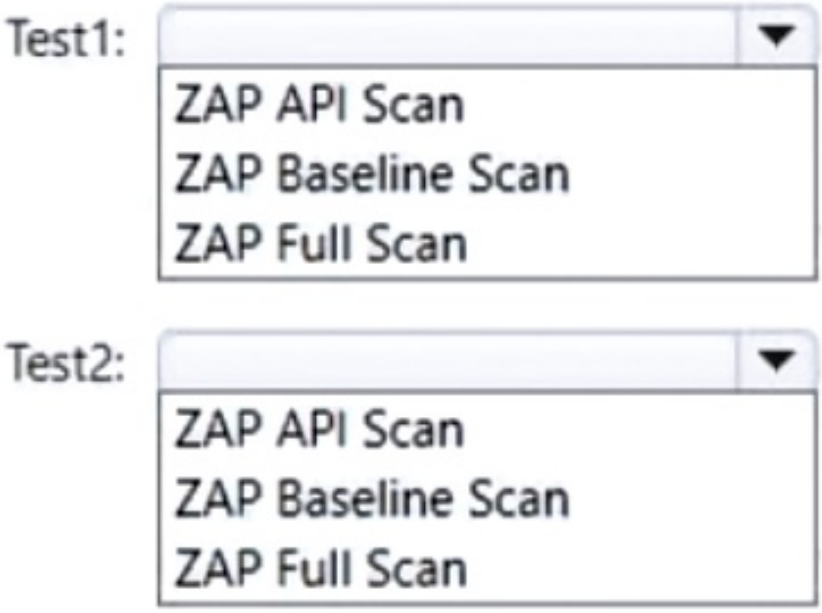
For further research, refer to the official GitHub documentation:

GitHub REST API - Actions Secrets
libsodium Documentation

Question: 199 CertyIQ

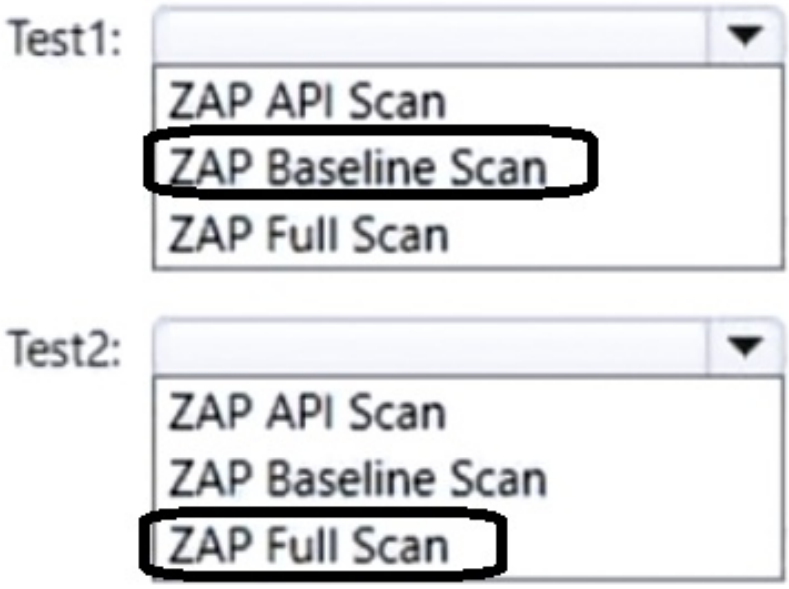
HOTSPOT
-
You have a GitHub repository named Repo1 and an app named App1. Repo1 stores the source code for App1.
You need to perform the following tests:
•Test1: Run a ZAP spider against App1 for one minute and wait for passive scanning to complete. The test must NOT perform active attacks.
•Test2: Run a ZAP spider against App1, and when complete, execute an AJAX spider scan.
Which GitHub action should you use for each test? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Question: 200 CertyIQ

You use Azure Pipelines pipeline to build and deploy an app named App1.
You need to ensure that before App1 is deployed, all the code for the app passes a security validation by using a custom tool.
What should you do?

A.Add a status check to the policies of the branch used by your company's development department.

B.Add a status check to the policies of the main branch.

C.Add a service hook to the project.

D.Limit the job authorization scope to the current project for all the release pipelines.

Answer: B

Explanation:

Here's a detailed justification for why option B, adding a status check to the policies of the main branch, is the most appropriate solution for ensuring security validation of code before deployment in an Azure DevOps pipeline:

Justification:

The requirement is to ensure all code passes security validation before App1 is deployed. This means integrating the security check into the deployment process as a mandatory gate. Branch policies in Azure DevOps are designed to enforce code quality and security requirements before changes are merged into a protected branch, such as the main branch.

Option B leverages this capability. By adding a status check to the main branch policy, you create a mandatory condition that must be met before a pull request (PR) targeting the main branch can be completed (merged). This status check would be configured to run the custom security validation tool. If the security validation

fails, the PR cannot be merged, effectively preventing insecure code from reaching the main branch and subsequently being deployed.

The main branch typically represents the stable, deployable version of the application. Ensuring code in this branch is secure is crucial for preventing security vulnerabilities in production deployments. By applying the security validation as a branch policy on the main branch, we're enforcing this security gate directly within the development workflow.

Why other options are less suitable:

A. Add a status check to the policies of the branch used by your company's development department: While applying a security check to the development branch is good practice for early detection, it doesn't guarantee that the main branch will also be secure. Developers might bypass the check in their individual branches or inadvertently introduce vulnerabilities during merge conflicts. Only the security on the main branch guarantees the deployable code is secured.

C. Add a service hook to the project: Service hooks are event-driven integrations that trigger actions in other services based on events in Azure DevOps. While they can be used to trigger security scans, they don't inherently block deployment based on the scan results. They are better suited for notifications and auditing purposes rather than acting as a mandatory gate.

D. Limit the job authorization scope to the current project for all the release pipelines: Limiting job authorization scope enhances security by restricting pipeline access to resources only within the current project. However, it doesn't directly address the requirement of running a specific security validation on the code itself.

Authoritative Links:

Branch policies and settings: https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops
Azure Pipelines YAML schema reference - checks: https://learn.microsoft.com/en-us/azure/devops/pipelines/yaml-schema/resources-checks?view=azure-devops (specifically Status Checks)

In conclusion, adding a status check to the main branch policy is the most direct and effective way to enforce a mandatory security validation before deploying App1, ensuring only secure code reaches production.

Question: 201 CertyIQ

You are designing the development process for your company.
You need to recommend a solution for continuous inspection of the company's code base to locate common code patterns that are known to be problematic.
What should you include in the recommendation?

A. Microsoft Visual Studio test plans
B. Gradle wrapper scripts
C. SonarCloud analysis
D. the JavaScript task runner

Answer: C

Explanation:

The correct answer is C, SonarCloud analysis. Here's a detailed justification:

Continuous inspection involves automating the process of analyzing code for potential issues, vulnerabilities, and code quality concerns throughout the development lifecycle. The objective is to identify problematic patterns early, allowing for timely remediation and preventing these issues from propagating into production.

SonarCloud is a cloud-based code quality and security analysis service specifically designed for continuous inspection. It integrates seamlessly with Azure DevOps pipelines, providing automated code analysis with each commit or build. SonarCloud identifies bugs, vulnerabilities, and code smells, offering detailed reports and guidance on how to fix them. It supports various programming languages and integrates with popular IDEs for a comprehensive developer experience.

Microsoft Visual Studio test plans (Option A) primarily deal with test case management and execution, not static code analysis for problematic patterns. Gradle wrapper scripts (Option B) manage Gradle versions and dependencies, but do not perform code analysis. The JavaScript task runner (Option D), while useful for build automation, doesn't inherently provide the in-depth code inspection capabilities needed for continuous inspection as provided by a dedicated tool like SonarCloud.

Therefore, SonarCloud is the most suitable choice for continuous inspection because it provides automated, cloud-based code analysis, identifies problematic code patterns, and integrates seamlessly with Azure DevOps, making it ideal for locating and addressing code quality issues early in the development cycle.

Authoritative links for further research:

SonarCloud: https://www.sonarsource.com/products/sonarcloud/
Azure DevOps: https://azure.microsoft.com/en-us/services/devops/
Continuous Integration: https://www.redhat.com/en/topics/devops/what-is-ci-cd

Question: 202 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
The lead developer at your company reports that adding new application features takes longer than expected due to a large accumulated technical debt.
You need to recommend changes to reduce the accumulated technical debt.
Solution: You recommend reducing the code coupling and the dependency cycles?
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

The solution, recommending the reduction of code coupling and dependency cycles, directly addresses the problem of accumulated technical debt and the resulting slowdown in feature development. High code coupling means that changes in one part of the system necessitate changes in other parts, leading to increased complexity and longer development times. Dependency cycles, where modules depend on each other in a circular fashion, exacerbate this issue, making refactoring and maintenance significantly more difficult.

Reducing code coupling through techniques like loose coupling, abstraction, and modularization makes the codebase more manageable and less prone to cascading errors. Loosely coupled components can be changed

independently without affecting other parts of the system, speeding up development. Dependency injection can be used to manage dependencies and reduce tight coupling.

Breaking dependency cycles involves restructuring the code to eliminate circular dependencies. This might involve creating new abstractions, merging or splitting modules, or introducing intermediary layers. Removing these cycles simplifies the dependency graph, making it easier to understand and modify the system.

By reducing code coupling and dependency cycles, the solution aims to decrease the complexity of the codebase, making it easier to add new features and resolve bugs. This directly addresses the lead developer's concern that adding new application features takes longer than expected due to technical debt. Therefore, the solution meets the goal.

Authoritative Links:

Martin Fowler on Coupling: https://martinfowler.com/bliki/Coupling.html
Dependency Inversion Principle: https://en.wikipedia.org/wiki/Dependency_inversion_principle
Technical Debt: https://martinfowler.com/bliki/TechnicalDebt.html

Question: 203 CertyIQ

Your company uses Azure DevOps for the build pipelines and deployment pipelines of Java-based projects.
You need to recommend a strategy for managing technical debt.
Which two actions should you include in the recommendation? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Configure post-deployment approvals in the deployment pipeline.
B. Configure pre-deployment approvals in the deployment pipeline.
C. Integrate Azure DevOps and SonarQube.
D. Integrate Azure DevOps and Azure DevTest Labs.

Answer: BC

Explanation:

The correct answer is B and C. Here's a detailed justification:

B. Configure pre-deployment approvals in the deployment pipeline: Pre-deployment approvals act as a gatekeeper before code changes are promoted to subsequent environments (e.g., staging or production). This allows stakeholders, such as security experts, architects, or senior developers, to review the changes introduced in the deployment package and assess potential technical debt risks before they impact a wider audience. This proactive assessment helps to prevent the introduction of new technical debt into more critical environments. If issues are identified, the deployment can be rejected, and the development team can address the debt before the code is deployed. This aligns with shift-left testing and risk mitigation strategies.

C. Integrate Azure DevOps and SonarQube: SonarQube is a popular open-source platform for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities. Integrating SonarQube with Azure DevOps allows automated code analysis as part of the build pipeline. This provides real-time feedback on code quality metrics, enabling the development team to identify and address potential technical debt early in the development lifecycle. The results of the analysis can be surfaced within Azure DevOps, making it easy for developers to see the issues and prioritize remediation. The integration helps to continuously monitor and reduce technical debt over time.

Why other options are incorrect:

A. Configure post-deployment approvals in the deployment pipeline: Post-deployment approvals are

generally used to verify successful deployments after they have already occurred. While useful for validating deployment success, they do little to prevent or manage the introduction of technical debt. Any debt is already deployed when the approval process begins. It's reactive instead of proactive.

D. Integrate Azure DevOps and Azure DevTest Labs: Azure DevTest Labs is primarily used for creating and managing environments for development and testing. While helpful for testing and validation, it doesn't directly address the problem of identifying and managing technical debt itself. The lab environments might help uncover performance or integration issues related to existing technical debt, but it won't provide the static code analysis and quality metrics that are key to identifying the source of the debt.

Supporting Links:

Azure Pipelines Approvals and Checks: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops
SonarQube: https://www.sonarsource.com/products/sonarqube/
SonarQube Extension for Azure DevOps: https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarqube

Question: 204 CertyIQ

Your company is building a new solution in Java.
The company currently uses a SonarQube server to analyze the code of .NET solutions.
You need to analyze and monitor the code quality of the Java solution.
Which task types should you add to the build pipeline?

A. Gradle
B. CocoaPods
C. Grunt
D. Gulp

Answer: A

Explanation:

The correct answer is A, Gradle. Here's why:

SonarQube analyzes code quality and requires a build tool to compile and package the Java code before analysis. Gradle is a popular open-source build automation tool specifically designed for multi-language software development. It manages dependencies, compiles code, runs tests, and packages the application. Integrating Gradle within the Azure DevOps build pipeline allows SonarQube to receive the compiled Java code and necessary project metadata for comprehensive code quality analysis.

Options B, C, and D are incorrect because they are not Java build tools. CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects (primarily for iOS and macOS development). Grunt and Gulp are JavaScript task runners, often used for front-end web development to automate tasks like minification, concatenation, and testing of JavaScript, CSS, and other web assets. They don't handle Java compilation or build processes.

Therefore, to analyze a Java solution with SonarQube within an Azure DevOps pipeline, a Java build tool is essential. Gradle efficiently compiles, packages, and prepares the Java code, making it the appropriate choice for integration with SonarQube for code quality assessment. Using Gradle ensures that SonarQube receives all the necessary components to perform its analysis effectively. The pipeline can then report the quality metrics to developers.

Relevant Resources:

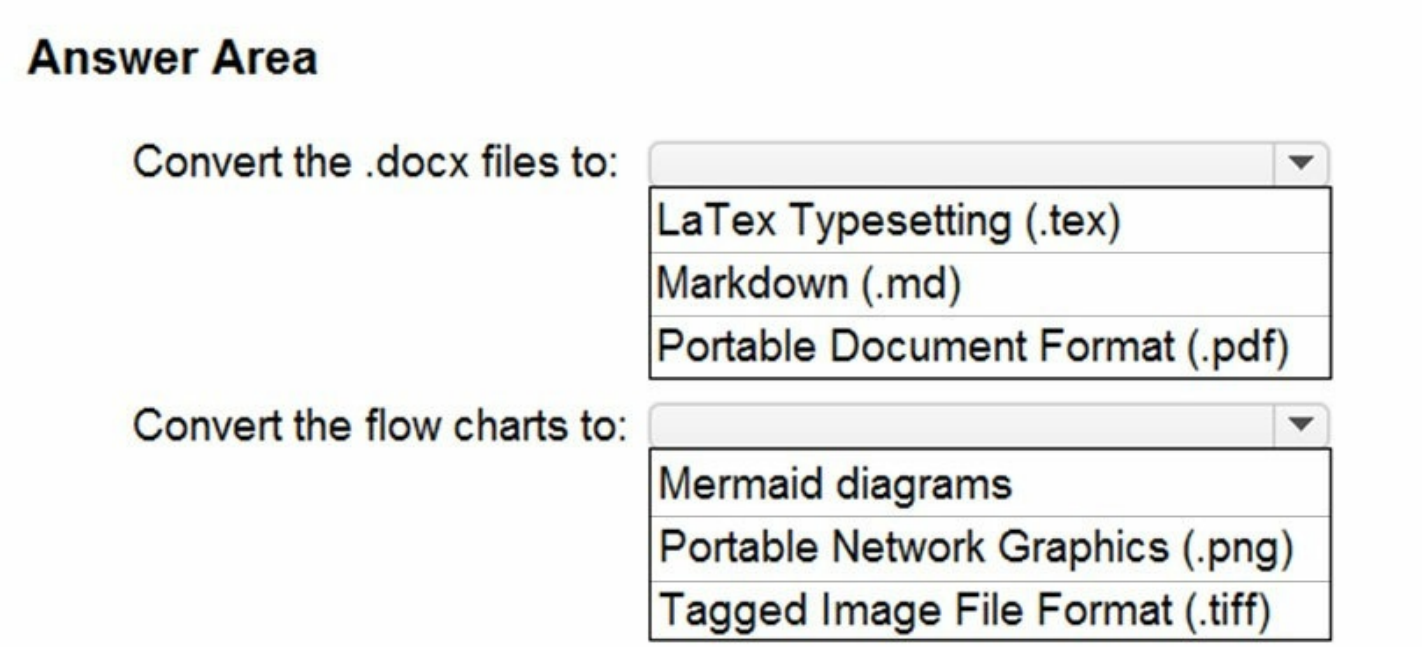
Gradle: https://gradle.org/
SonarQube: https://www.sonarsource.com/products/sonarqube/
Azure DevOps Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops

Question: 205 CertyIQ

HOTSPOT -
Your company uses GitHub for source control. GitHub repositories store source code and store process documentation. The process documentation is saved as
Microsoft Word documents that contain simple flow charts stored as .bmp files.
You need to optimize the integration and versioning of the process documentation and the flow charts. The solution must meet the following requirements:
- Store documents as plain text.
- Minimize the number of files that must be maintained.
- Simplify the modification, merging, and reuse of flow charts.

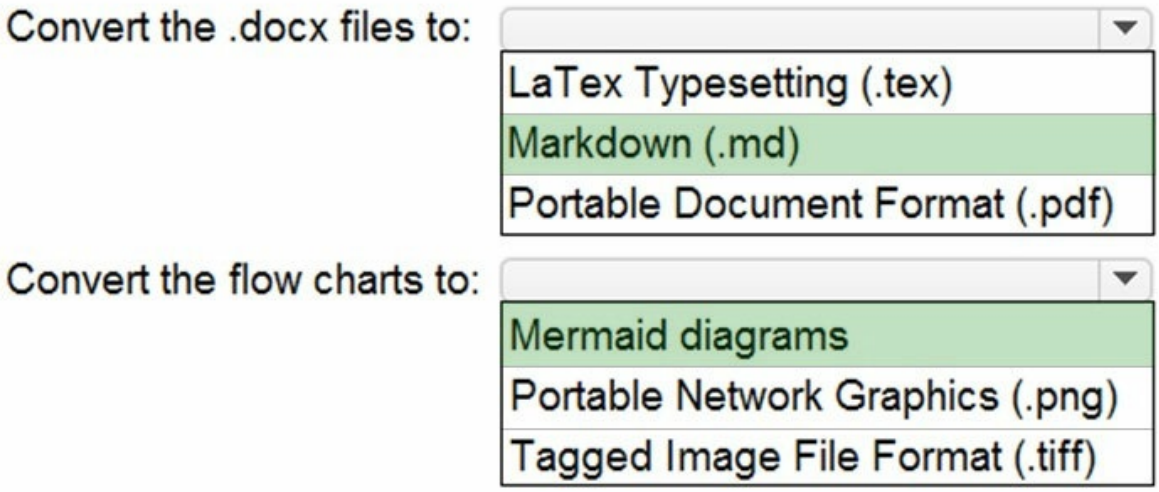
Simplify the modification, merging, and reuse of documents.

Hot Area:



Answer:

Answer Area



Explanation:
Box 1: Markdown (.md)
Github understands several text formats, including .txt and .md. .md stands for a file written in Markdown.

Box 2: Mermaid diagrams -
Mermaid lets you create diagrams and visualizations using text and code.
It is a Javascript based diagramming and charting tool that renders Markdown-inspired text definitions to create and modify diagrams dynamically.

Reference:
https://ourcodingclub.github.io/tutorials/git/
https://mermaid-js.github.io/mermaid/#/

Question: 206 CertyIQ

Your company is building a new solution in Java.
The company currently uses a SonarQube server to analyze the code of .NET solutions.
You need to analyze and monitor the code quality of the Java solution.
Which task types should you add to the build pipeline?

A. Grunt
B. Octopus
C. Maven
D. Gulp

Answer: C

Explanation:

The correct answer is C, Maven. Here's a detailed justification:

SonarQube analyzes code using plugins that understand the specific programming language. For Java projects, a SonarQube analysis often requires a build tool integration to compile the code, resolve dependencies, and provide necessary metadata. Maven is a widely used build automation tool primarily used for Java projects. It manages dependencies, compiles source code, packages the application, and runs tests.

When integrating SonarQube with a Java build pipeline, the Maven task executes the SonarQube scanner plugin within the Maven lifecycle. This plugin triggers the analysis of the Java code by the SonarQube server. The scanner collects metrics, detects code smells, bugs, and security vulnerabilities according to the configured SonarQube rules. The results are then sent to the SonarQube server for reporting and monitoring.

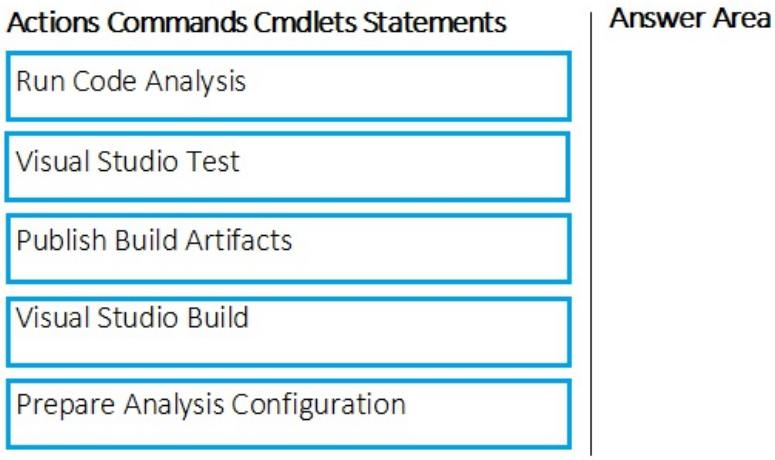
Grunt and Gulp are primarily JavaScript task runners used for front-end development. They are not designed for analyzing Java code. Octopus Deploy is a deployment automation tool and is not directly involved in code analysis. While they have their uses within other pipelines, Maven is the best option for integrating Java code analysis with SonarQube. Choosing Maven facilitates the integration and makes use of tools already available for Java. The Maven task seamlessly invokes the SonarQube scanner, ensuring a smooth and efficient code quality analysis process.

For more information on integrating SonarQube with Maven, refer to the official SonarQube documentation:

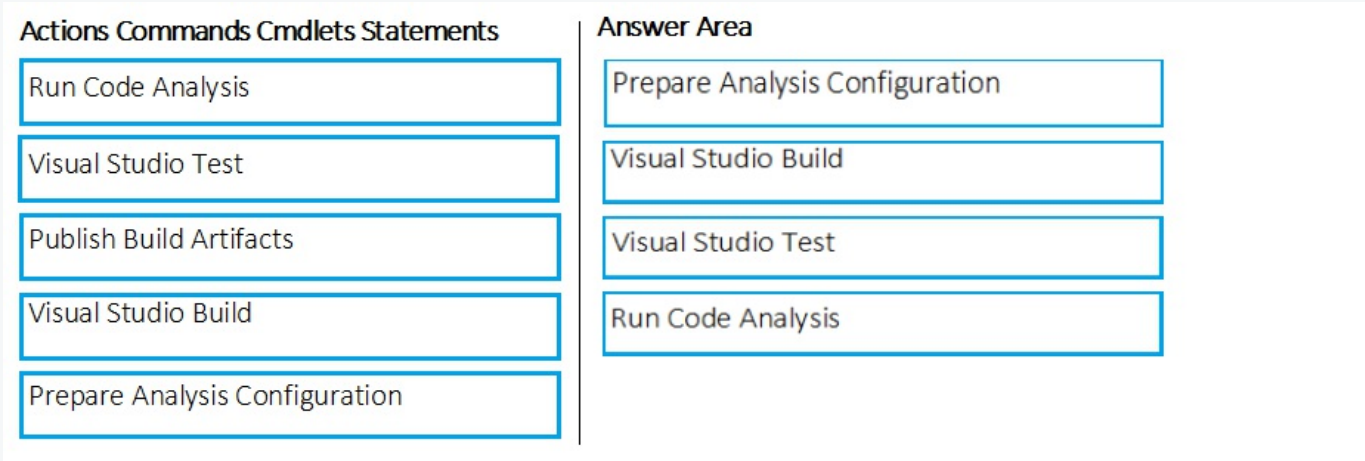
SonarQube Maven Plugin
Apache Maven

Question: 207 CertyIQ

DRAG DROP -
You are developing a full Microsoft .NET Framework solution that includes unit tests.
You need to configure SonarQube to perform a code quality validation of the C# code as part of the build pipelines.
Which four tasks should you perform in sequence? To answer, move the appropriate tasks from the list of tasks to the answer area and arrange them in the correct order.
Select and Place:



Answer:



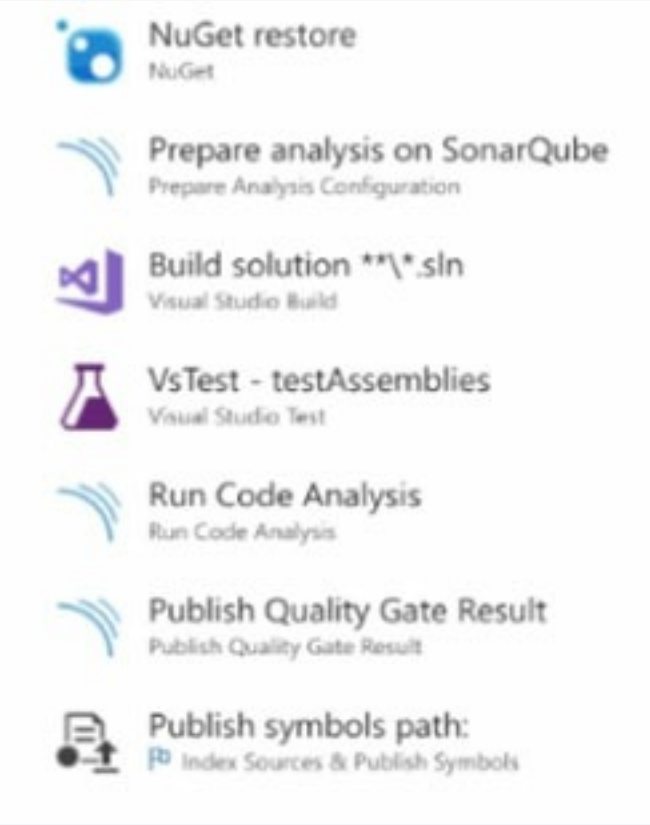
Explanation:
Step 1: Prepare Analysis Configuration

Prepare Analysis Configuration task, to configure all the required settings before executing the build.
This task is mandatory.
In case of .NET solutions or Java projects, it helps to integrate seamlessly with MSBuild, Maven and Gradle tasks.

Step 2: Visual Studio Build -
Reorder the tasks to respect the following order:
Prepare Analysis Configuration task before any MSBuild or Visual Studio Build task.

Step 3: Visual Studio Test -
Reorder the tasks to respect the following order:
Run Code Analysis task after the Visual Studio Test task.

Step 4: Run Code Analysis -
Run Code Analysis task, to actually execute the analysis of the source code.
This task is not required for Maven or Gradle projects, because scanner will be run as part of the Maven/Gradle build.
Note:



Reference:
https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Extension+for+VSTS-TFS

Question: 208 CertyIQ

Your company uses Azure DevOps for the build pipelines and deployment pipelines of Java-based projects.
You need to recommend a strategy for managing technical debt.
Which action should you include in the recommendation?

A. Configure post-deployment approvals in the deployment pipeline.
B. Integrate Azure DevOps and SonarQube.
C. Integrate Azure DevOps and Azure DevTest Labs.

Answer: B

Explanation:

The correct answer is **B. Integrate Azure DevOps and SonarQube.**

Technical debt refers to the implied cost of rework caused by choosing an easy (i.e., quick and short-term) solution now instead of using a better approach that would take longer. Addressing it requires visibility and continuous monitoring of code quality, security vulnerabilities, and maintainability issues.

SonarQube is an open-source platform for continuous inspection of code quality. Integrating it with Azure DevOps pipelines allows for automated analysis of Java code during builds. SonarQube can identify code smells, bugs, security vulnerabilities, and maintainability issues, providing developers with feedback on the technical debt present in their code. By integrating SonarQube into the Azure DevOps pipeline, the team gains valuable insights into the quality of their code, enabling them to make informed decisions about addressing technical debt during development. This proactive approach helps prevent the accumulation of technical debt over time. SonarQube provides reports and metrics within Azure DevOps, making it easier for teams to track progress in reducing technical debt.

Option A, configuring post-deployment approvals, primarily focuses on controlling releases and minimizing risks during deployment, not directly addressing technical debt. Option C, integrating Azure DevOps and Azure DevTest Labs, helps in creating and managing environments for testing, which can be valuable in validating solutions after addressing technical debt, but it doesn't assist in identifying and measuring the debt itself.

Therefore, integrating Azure DevOps with SonarQube is the most effective strategy for managing technical debt in Java-based projects as it provides the necessary tools and insights for identifying, measuring, and addressing code quality issues.

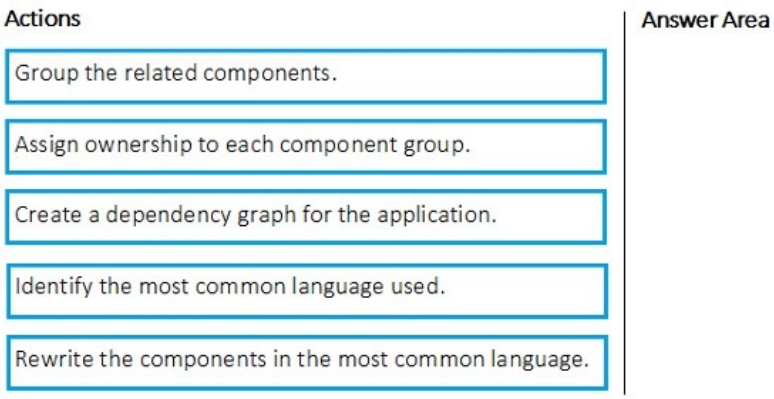
Relevant Links:

SonarQube: https://www.sonarsource.com/products/sonarqube/
Azure DevOps: https://azure.microsoft.com/en-us/services/devops/
Technical Debt: https://martinfowler.com/bliki/TechnicalDebt.html

Question: 209

CertyIQ

DRAG DROP -
You need to find and isolate shared code. The shared code will be maintained in a series of packages.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:

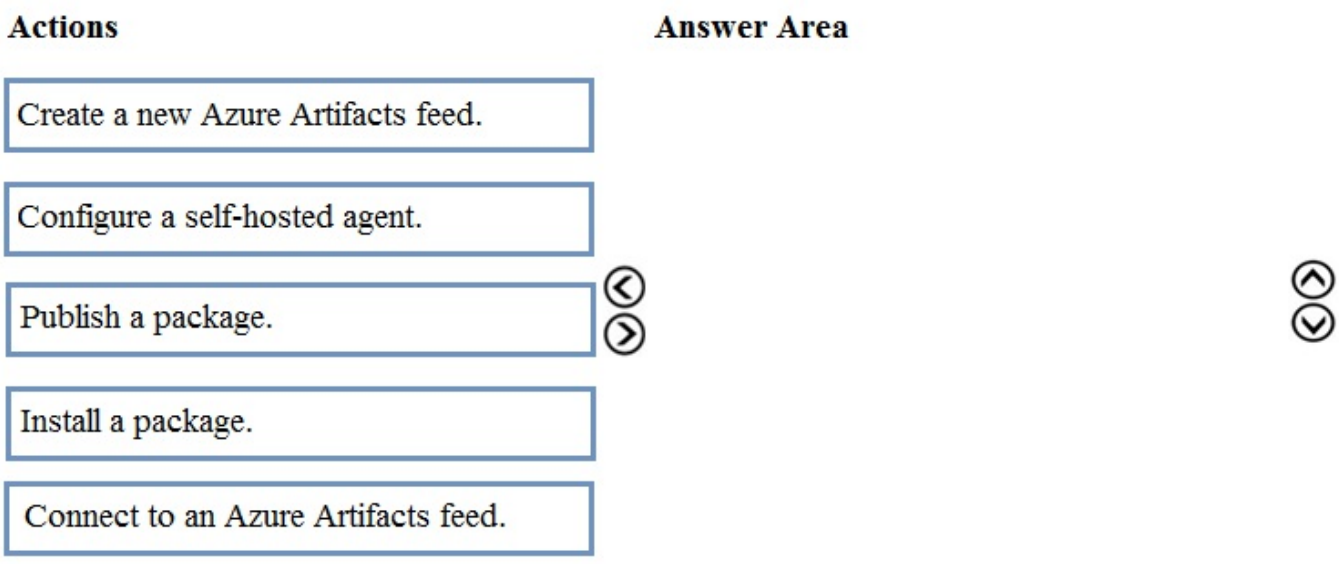
Actions	Answer Area
Group the related components.	Create a dependency graph for the application.
Assign ownership to each component group.	Group the related components.
Create a dependency graph for the application.	Assign ownership to each component group.
Identify the most common language used.	
Rewrite the components in the most common language.	

Explanation:
Step 1: Create a dependency graph for the application
By linking work items and other objects, you can track related work, dependencies, and changes made over time. All links are defined with a specific link type. For example, you can use Parent/Child links to link work items to support a hierarchical tree structure. Whereas, the Commit and Branch link types support links between work items and commits and branches, respectively.
Step 2: Group the related components.
Packages enable you to share code across your organization: you can compose a large product, develop multiple products based on a common shared framework, or create and share reusable components and libraries.
Step 3: Assign ownership to each component graph

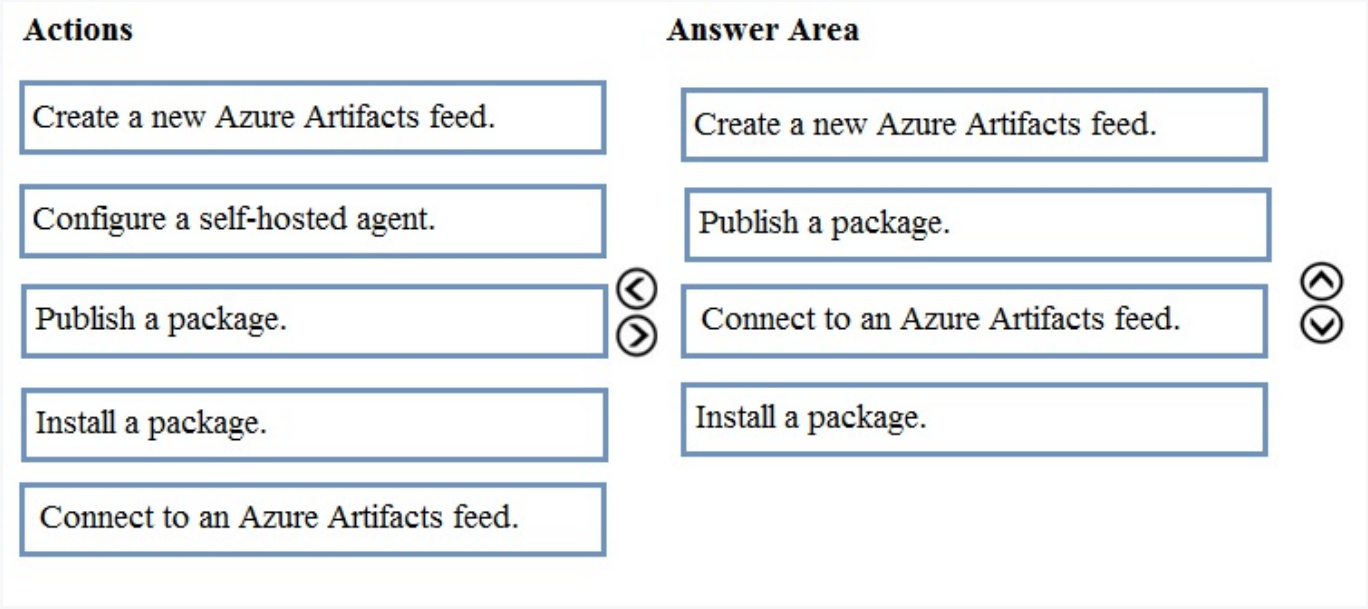
Reference:
https://docs.microsoft.com/en-us/azure/devops/boards/queries/link-work-items-support-traceability?view=azure-devops&tabs=new-web-for m https://docs.microsoft.com/en-us/visualstudio/releasenotes/tfs2017-relnote s

Question: 210 CertyIQ

DRAG DROP -
You are creating a NuGet package.
You plan to distribute the package to your development team privately.
You need to share the package and test that the package can be consumed.
Which four actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

1) Create a new Azure Artifacts feed.

2) Publish a package.

3) Connect to an Azure Artifacts feed.

4) Install a package.

Reference:

https://docs.microsoft.com/en-us/azure/devops/artifacts/get-started-nuget?view=azure-devops&tabs=windows#create-a-feed

Question: 211 CertyIQ

During a code review, you discover many quality issues. Many modules contain unused variables and empty catch blocks.
You need to recommend a solution to improve the quality of the code.
What should you recommend?

A. In a Grunt build task, select Enabled from Control Options.
B. In a Maven build task, select Run PMD.
C. In a Xcode build task, select Use xcpretty from Advanced.
D. In a Gradle build task, select Run Checkstyle.

Answer: B

Explanation:

The correct answer is B: In a Maven build task, select Run PMD. Here's why:

The goal is to improve code quality by identifying issues like unused variables and empty catch blocks. These are typically static code analysis concerns. PMD (Programming Mistake Detector) is a static code analysis tool specifically designed to find common programming flaws like these in Java, JavaScript, and other languages.

Integrating PMD into the Maven build process provides an automated way to detect such problems during the development lifecycle, ensuring code quality before deployment.

Maven is a popular build automation tool, especially for Java projects, and seamlessly integrates with various plugins, including those for static analysis. By configuring the Maven build task to run PMD, every build will automatically trigger the PMD analysis, flagging violations and preventing the integration of low-quality code. PMD's rule sets can be customized to adhere to specific coding standards and identify a wide range of potential bugs, performance issues, and code smells. This proactive approach to quality control helps developers address issues early, reducing the cost and complexity of fixing them later.

The other options are less suitable: Grunt (A) is a JavaScript task runner and doesn't have a direct equivalent to PMD. Enabled control options typically relate to workflow control, not code quality. Xcode (C) and xcpretty are specific to iOS development, not general code quality. Gradle (D) and Checkstyle are similar to Maven and PMD, respectively. While Checkstyle is also a static analysis tool, PMD tends to be more comprehensive in its detection of the specific issues mentioned (unused variables, empty catch blocks). The question doesn't restrict the solution to a specific build tool, so PMD's strengths make option B the best choice. In an Azure DevOps pipeline for a Java project using Maven, enabling the "Run PMD" option will directly enforce code quality standards.

Relevant Links:

Apache Maven: https://maven.apache.org/
PMD: https://pmd.github.io/
Azure DevOps Pipelines: https://azure.microsoft.com/en-us/services/devops/pipelines/

Question: 212 CertyIQ

Your development team is building a new web solution by using the Microsoft Visual Studio integrated development environment (IDE).
You need to make a custom package available to all the developers. The package must be managed centrally, and the latest version must be available for consumption in Visual Studio automatically.
Which three actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Publish the package to a feed.
B. Create a new feed in Azure Artifacts.
C. Upload a package to a Git repository.
D. Add the package URL to the Environment settings in Visual Studio.
E. Add the package URL to the NuGet Package Manager settings in Visual Studio.
F. Create a Git repository in Azure Repos.

Answer: BAE

Explanation:

The correct answer is **BAE**. Here's a detailed justification:

B. Create a new feed in Azure Artifacts: Azure Artifacts is the service within Azure DevOps specifically designed for hosting and managing packages like NuGet, npm, Maven, and others. Creating a feed within Azure Artifacts provides a centralized location for storing and sharing your custom packages. This ensures that all developers have access to the same version of the package and that package management is consistent across the team. Feeds in Azure Artifacts can be configured with specific access control and retention policies.

A. Publish the package to a feed: Once the feed is created in Azure Artifacts, the next step is to actually publish the custom package to that feed. This makes the package available for consumption by the development team. Publishing involves uploading the package to the feed, which stores it and makes it accessible via NuGet Package Manager in Visual Studio or other package management tools.

E. Add the package URL to the NuGet Package Manager settings in Visual Studio: To enable Visual Studio to find and consume the package from the Azure Artifacts feed, developers need to add the feed's URL to the NuGet Package Manager settings within their Visual Studio IDE. This informs Visual Studio where to look for available packages. Adding the feed URL allows Visual Studio to automatically discover and install the latest version of the package.

Here's why the other options are incorrect:

C. Upload a package to a Git repository: While Git repositories are excellent for source code management, they are not ideal for managing binary packages. They lack the features required for versioning, dependency management, and access control that are essential for package management.
D. Add the package URL to the Environment settings in Visual Studio: Environment settings in Visual Studio are primarily for configuring project-specific settings and are not the correct place to add package source URLs for NuGet Package Manager.
F. Create a Git repository in Azure Repos: Creating a Git repository is unnecessary for package management using Azure Artifacts. While related to development, Git repositories serve a different purpose, such as managing code changes and collaboration.

Authoritative Links:

Azure Artifacts Overview: https://learn.microsoft.com/en-us/azure/devops/artifacts/start/artifacts-overview?view=azure-devops
Get started with NuGet packages in Azure Artifacts: https://learn.microsoft.com/en-us/azure/devops/artifacts/get-started-nuget?view=azure-devops&tabs=new-nav
Publish and Consume NuGet packages with Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/artifacts/nuget?view=azure-devops

Question: 213 CertyIQ

You use GitHub for source control.
A file that contains sensitive data is committed accidentally to the Git repository of a project.
You need to delete the file and its history form the repository.
Which two tools can you use? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A. the git filter-branch command
B. BFG Repo-Cleaner
C. the git rebase command
D. GitHub Desktop

Answer: AB

Explanation:

The correct answer is AB.

A. The git filter-branch command: This is a powerful (but complex) Git command specifically designed for rewriting Git history. It allows you to iterate through each commit and modify it. In this scenario, you can use git filter-branch to remove the sensitive file from every commit it appears in, effectively scrubbing it from the

repository's history. While it gets the job done, it's generally considered more difficult to use and slower than BFG Repo-Cleaner, especially on large repositories. It also rewrites the entire history, which changes commit hashes and can require force pushes and coordination among collaborators.

B. BFG Repo-Cleaner: This is a specialized tool specifically designed for removing large or sensitive data from Git history. BFG Repo-Cleaner is generally faster and easier to use than git filter-branch, especially for simpler tasks like removing a specific file across all commits. It efficiently rewrites the Git history, making it a good choice for removing sensitive data across all branches and tags. It's optimized for speed and simplicity, making it ideal for this particular scenario.

Why C is incorrect: The git rebase command is primarily used for integrating changes from one branch into another, or for cleaning up a branch's history by rearranging commits. While git rebase can be used in conjunction with other commands to modify history, it's not the primary tool for removing files and their history across an entire repository. It's better suited for changing the order or content of commits on a single branch, not for systematically removing a file across all branches.

Why D is incorrect: GitHub Desktop is a GUI for Git, not a tool for rewriting history. While you can use GitHub Desktop to commit and push changes, it doesn't provide any functionality to rewrite the entire history of a repository to remove a file across all commits and branches.

Therefore, using git filter-branch or BFG Repo-Cleaner allows you to effectively remove the file and its history from the repository, addressing the requirement to remove sensitive data committed by accident. BFG Repo-Cleaner is the preferred choice for simplicity and performance in this case.

Authoritative Links:

BFG Repo-Cleaner: https://rtyley.github.io/bfg-repo-cleaner/
Git filter-branch: https://git-scm.com/docs/git-filter-branch (Note: this command is discouraged in favor of git filter-repo, but is still a viable tool for this task)

Question: 214 CertyIQ

Your company uses GitHub for source control. The company has a team that performs code reviews.
You need to automate the assignment of the code reviews. The solution must meet the following requirements:
- Prioritize the assignment of code reviews to team members who have the fewest outstanding assignments.
- Ensure that each team member performs an equal number of code reviews in any 30-day period.
- Prevent the assignment of code reviews to the team leader.

Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Clear Never assign certain team members.
B. Select If assigning team members, don't notify the entire team.
C. Select Never assign certain team members.
D. Set Routing algorithm to Round robin.
E. Set Routing algorithm to Load balance.

Answer: CE

Explanation:

The correct answer is CE. Here's a detailed justification:

C. Select Never assign certain team members.

This option directly addresses the requirement to prevent the assignment of code reviews to the team leader.

Code review automation tools, often integrated within or connected to GitHub, typically provide functionality to exclude specific users from being assigned reviews. This ensures the team leader's time isn't consumed by routine review tasks, allowing them to focus on leadership and strategic responsibilities. By configuring the code review automation tool to "Never assign" the team leader, the system will bypass them when distributing review requests.

E. Set Routing algorithm to Load balance.

This option addresses the requirements for prioritizing team members with fewer outstanding assignments and ensuring an equal distribution of code reviews. A "Load balance" routing algorithm aims to distribute work evenly across available resources (in this case, team members). This type of algorithm considers the current workload of each team member, prioritizing those with fewer active reviews. In effect, this prevents overload on some members while others remain underutilized, thereby promoting fairness and potentially faster review completion times. Load balancing ensures that over time, each team member performs a roughly equal number of reviews. Using load balancing, the system automatically adjusts assignments to reflect each team members' capacity within the 30-day window. This satisfies the requirement that each team member performs an equal number of code reviews within any 30-day period.

Why other options are incorrect:

A. Clear Never assign certain team members: "Clear" suggests removing an existing setting, which contradicts the requirement to exclude the team leader.

B. Select If assigning team members, don't notify the entire team: Notification preferences are important, but do not directly address the code review assignment requirements specified in the question. Notifications primarily affect communication, not assignment logic.

D. Set Routing algorithm to Round robin: While "Round robin" ensures everyone eventually gets assigned reviews, it doesn't consider the current workload of team members, which can lead to uneven workloads and violate the requirements. Load balancing is better suited for equalizing the effort over time.

Supporting Documentation & Links:

GitHub Code Review Automation (Example): While GitHub doesn't have a built-in load-balanced code review assignment directly within the platform itself, numerous third-party tools integrate with GitHub to provide more advanced functionality. The specific features and terminology will vary depending on the tool, but the underlying principle remains the same: https://github.com/features/code-review This links to GitHub's general code review features, showing the broader ecosystem for improvements and integrations.

Atlassian Crucible (Example - a code review tool): This is an example of a third-party code review tool that allows specifying users to exclude from code reviews and might have algorithms that balance assignments. Please note that Crucible is only one such example, and other code review tools may have analogous functionality: https://www.atlassian.com/software/crucible This is included to give a sense of tools beyond GitHub that address code review concerns.

Question: 215 CertyIQ

You have a GitHub repository.
You create a new repository in Azure DevOps.
You need to recommend a procedure to clone the repository from GitHub to Azure DevOps.
What should you recommend?

A. Create a pull request.
B. Create a webhook.
C. Create a service connection for GitHub.

D. From Import a Git repository, click Import.

E. Create a personal access token in Azure DevOps.

Answer: D

Explanation:

The recommended procedure to clone a GitHub repository to Azure DevOps is to use the "Import a Git repository" feature and click "Import." This method directly transfers the repository's contents, history, and branches from GitHub to Azure DevOps. Creating a pull request (Option A) is used for contributing changes between branches or repositories, not for initial cloning. Webhooks (Option B) are for triggering actions based on events, not for cloning a repository. Creating a service connection for GitHub (Option C) facilitates integration between Azure DevOps and GitHub for various purposes, like CI/CD pipelines, but is not the direct method for importing a repository. While a personal access token (PAT) in Azure DevOps (Option E) can be used for authentication with Git operations, it isn't the primary way to import a repository. The "Import" feature streamlines the cloning process, handling authentication and data transfer in a user-friendly manner. It is the most efficient and direct method for migrating a Git repository from an external source like GitHub to Azure DevOps. This functionality is specifically designed for initial repository creation from existing sources.

Further research:

Import Git repositories: https://learn.microsoft.com/en-us/azure/devops/repos/git/import-git-repository?view=azure-devops

Question: 216 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
The lead developer at your company reports that adding new application features takes longer than expected due to a large accumulated technical debt.
You need to recommend changes to reduce the accumulated technical debt.
Solution: You recommend increasing the code duplication.
Does this meet the goal?

A. Yes

B. No

Answer: B

Explanation:

The solution proposes increasing code duplication to reduce technical debt. This is incorrect and does not meet the goal. Technical debt refers to the implied cost of rework caused by choosing an easy solution now instead of using a better approach that would take longer. Code duplication directly increases technical debt. Duplicated code makes maintenance, bug fixes, and feature additions significantly more difficult and error-prone. When a bug is found in one instance of the duplicated code, it must be fixed in all other instances, increasing the risk of oversight. Similarly, feature enhancements require modification across all duplicated sections, amplifying the effort and potential for inconsistencies. Reducing technical debt typically involves refactoring code, improving design, automating tests, and addressing architectural deficiencies. Increasing code reuse through modularization and abstraction is a common strategy to decrease technical debt.

Justification:

Code Duplication Increases Maintenance Burden: Duplicated code needs to be updated in multiple places, leading to increased maintenance effort and potential inconsistencies.
Increases Bug Risk: Bugs found in one instance of duplicated code must be fixed in all other instances, increasing the risk of oversight and introducing new bugs during the fix.
Hinders Refactoring: Code duplication makes refactoring more difficult and time-consuming, hindering efforts to improve the codebase.
Increases Complexity: Duplicated code increases the overall complexity of the codebase, making it harder to understand and maintain.
Violates DRY (Don't Repeat Yourself) Principle: Code duplication is a direct violation of the DRY principle, which emphasizes the importance of avoiding redundancy in code.

Authoritative Links:

Technical Debt: https://martinfowler.com/bliki/TechnicalDebt.html
Code Smells - Duplicated Code: https://refactoring.guru/smells/duplicated-code

Question: 217 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
The lead developer at your company reports that adding new application features takes longer than expected due to a large accumulated technical debt.
You need to recommend changes to reduce the accumulated technical debt.
Solution: You recommend increasing the test coverage.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The correct answer is **B. No**.

While increasing test coverage is a good practice and can help prevent future technical debt, it does not directly address or reduce existing technical debt. Technical debt is the implied cost of rework caused by choosing an easy solution now instead of using a better approach that would take longer. It represents shortcuts taken during development that lead to future maintenance issues, performance bottlenecks, or security vulnerabilities.

Increasing test coverage primarily helps in the following ways:

Identifying new issues: Tests can help discover bugs or problems in existing code, even those related to past shortcuts.
Preventing regressions: When refactoring to address technical debt, tests will ensure that new changes don't break existing functionality.
Facilitating refactoring: Good test coverage makes it safer to refactor code, as you can quickly verify that your changes haven't introduced any regressions.

However, it doesn't inherently eliminate the underlying causes or symptoms of technical debt, such as poorly

designed code, outdated libraries, or undocumented functionalities. To reduce technical debt, developers need to specifically identify and address the problematic areas of the codebase through refactoring, code cleanup, architectural changes, and other remediation efforts. These activities might benefit from increased test coverage, but testing alone isn't sufficient.

In conclusion, simply increasing test coverage won't reduce the accumulated technical debt. It's a valuable practice to prevent future debt and support refactoring efforts, but direct actions are required to address the existing debt.

Supporting Links:

Martin Fowler on Technical Debt: https://martinfowler.com/bliki/TechnicalDebt.html - A classic and comprehensive definition of technical debt.
Reducing Technical Debt: https://docs.microsoft.com/en-us/devops/deliver/reduce-technical-debt - Microsoft's documentation on reducing technical debt.

Question: 218 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
The lead developer at your company reports that adding new application features takes longer than expected due to a large accumulated technical debt.
You need to recommend changes to reduce the accumulated technical debt.
Solution: You recommend reducing the code complexity.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

Yes, the solution meets the goal. Reducing code complexity is a fundamental practice in addressing technical debt. Complex code is harder to understand, maintain, and modify, leading to increased development time, higher error rates, and resistance to change. This directly contributes to technical debt because future development efforts become slower and more costly.

By simplifying code through techniques like refactoring, breaking down large functions, and improving code readability, developers can reduce the cognitive load required to work with the codebase. This, in turn, makes it easier to add new features, fix bugs, and improve overall system performance. Simplified code is easier to test, which contributes to higher code quality and reduces the risk of introducing new problems when modifying existing code. Furthermore, reducing code complexity can improve the onboarding process for new developers, enabling them to become productive faster.

By focusing on reducing complexity, developers effectively address the root causes of technical debt, leading to long-term benefits such as improved maintainability, reduced development costs, and increased agility. High complexity often leads to more bugs, security vulnerabilities, and difficulties in scaling the application. Reducing it, therefore, directly impacts these aspects positively. Techniques such as static code analysis can help identify and address code complexity issues.

Therefore, recommending a reduction in code complexity is a valid and effective approach to reducing

accumulated technical debt.

For more information, consider exploring resources on code refactoring and design principles:

Refactoring (Improving the Design of Existing Code) by Martin Fowler: This book provides a comprehensive guide to refactoring techniques for reducing code complexity.
SOLID principles: These principles provide guidance on designing maintainable and understandable code.
https://en.wikipedia.org/wiki/SOLID

Question: 219 CertyIQ

During a code review, you discover quality issues in a Java application.
You need to recommend a solution to detect quality issues including unused variables and empty catch blocks.
What should you recommend?

A. In a Maven build task, select Run PMD.
B. In an Xcode build task, select Use xcpretty from Advanced.
C. In a Gulp build task, specify a custom condition expression.
D. In a Grunt build task, select Enabled from Control Options.

Answer: A

Explanation:

The correct answer is A: In a Maven build task, select Run PMD. Here's why:

PMD is a static analysis tool that directly addresses the need to detect code quality issues in Java applications, like unused variables and empty catch blocks, exactly as specified in the problem statement. Maven is a popular build automation tool for Java projects, making the integration of PMD within a Maven build task a natural and efficient way to incorporate quality checks into the development process. Azure DevOps provides seamless integration with Maven builds.

By selecting "Run PMD" in the Maven build task configuration within Azure DevOps, you instruct the build process to automatically execute PMD against your Java code during each build. This enables developers to proactively identify and address potential problems before they impact the application in production. PMD analyzes the code based on a set of predefined rules, detecting violations of best practices and potential bugs.

Options B, C, and D are not relevant for detecting Java code quality issues. Xcode is for building Apple applications, Gulp and Grunt are JavaScript task runners and lack the same direct integration and focus on Java code quality as PMD does. Furthermore, custom condition expressions in Gulp and Control Options in Grunt build tasks are not specialized for analyzing Java code in the way that PMD is. Leveraging the native support for PMD within Maven offered by Azure DevOps build tasks is the optimal solution in this scenario.

In summary, selecting "Run PMD" in a Maven build task provides an automated and targeted approach for identifying Java code quality issues like unused variables and empty catch blocks, using an established and reliable static analysis tool directly integrated into the Java build process within Azure DevOps.

Here are authoritative links for further research:

PMD: https://pmd.github.io/
Maven: https://maven.apache.org/
Azure DevOps Maven Task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/build/maven?view=azure-devops

Question: 220 CertyIQ

You use Azure Artifacts to host NuGet packages that you create.
You need to make one of the packages available to anonymous users outside your organization. The solution must minimize the number of publication points.
What should you do?

A. Change the feed URL of the package
B. Create a new feed for the package
C. Promote the package to a release view.
D. Publish the package to a public NuGet repository.

Answer: D

Explanation:

The correct answer is **D: Publish the package to a public NuGet repository.**

Here's a detailed justification:

Goal: The objective is to make a NuGet package available to anonymous users outside the organization while minimizing publication points.

Why option D is the best approach:

Public Accessibility: Public NuGet repositories (like nuget.org) are inherently designed for anonymous, public access. Anyone can search and download packages from these repositories without authentication.
Minimizes Publication Points: Publishing to a central public repository requires only one publication action. The package is then available to all users globally through a single, well-known location.
No Azure DevOps Organization Required: External users do not need access to the Azure DevOps organization, reducing overhead and administrative complexity.

Why other options are not ideal:

A. Change the feed URL of the package: Modifying a feed URL does not inherently grant anonymous access. Azure Artifacts feeds usually require authentication, even if the URL is publicly known. This does not solve the problem for anonymous users.
B. Create a new feed for the package: Creating a new feed within Azure Artifacts still necessitates managing access permissions. While you could potentially configure anonymous access (requires careful configuration and is generally not recommended), it would still create a new, separate publication point within the Azure DevOps organization. Creating a new feed simply duplicates the package, which increases management overhead.
C. Promote the package to a release view: Release views in Azure Artifacts are primarily for managing the quality and stability of packages within an organization. They control which packages are considered safe for consumption in different environments (e.g., development, testing, production). They do not inherently provide public, anonymous access. Though views can have permissions set, views are designed for access control within an organization not for anonymous users.

Cloud Computing Concepts: This solution aligns with the principles of leveraging public cloud services for broader accessibility and reducing the operational burden of managing infrastructure for external users.

NuGet Repository: NuGet is a package manager for .NET, and nuget.org is its primary public repository.

Supporting Documentation:

NuGet Official Documentation: https://learn.microsoft.com/en-us/nuget/
Publishing Packages to NuGet.org: https://learn.microsoft.com/en-us/nuget/create-packages/publish-a-package
Azure Artifacts Overview: https://learn.microsoft.com/en-us/azure/devops/artifacts/start/feeds?view=azure-devops

Question: 221 CertyIQ

You use GitHub for source control and project-related discussions.
You receive a notification when an entry is made to any team discussion.
You need to ensure that you receive email notifications only for discussions in which you commented or in which you are mentioned.
Which two Notifications settings should you clear? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Automatically watch teams
B. Participating
C. Automatically watch repositories
D. Watching

Answer: BD

Explanation:

The goal is to receive email notifications only for GitHub discussions where you commented or are mentioned. This implies disabling general notifications that don't involve direct interaction.

B. Participating: This setting typically refers to notifications for discussions in which you are actively involved (i.e., you have commented). Since the requirement is to receive notifications for these discussions, we should not clear this setting. Instead, keep "Participating" enabled, as it ensures you are notified when there's activity in discussions where you've already contributed. Therefore, option B is incorrect to clear.

D. Watching: This setting means you receive notifications for all activity in the watched repository, irrespective of your direct involvement (comments or mentions). Since the requirement is to receive email notifications only for discussions where you commented or are mentioned, we must clear the "Watching" setting. This will stop general notifications related to the whole repository or discussions within it that you haven't engaged with.

A. Automatically watch teams: This setting would cause you to receive notifications for all activity related to your teams. Clearing this setting ensures you only receive notifications based on your direct involvement in discussions.

C. Automatically watch repositories: Similar to "Watching," this setting causes you to receive notifications for all activities (including discussions) in watched repositories, even if you haven't contributed or been mentioned. Clearing this setting is necessary.

Therefore, the solution requires clearing "Automatically watch teams" and "Automatically watch repositories". This ensures you aren't receiving general notifications, only those where you've commented or been mentioned.

The correct answer is AC.

References

GitHub Notifications: This documentation provides an overview of notification settings on GitHub and how to

configure them.

The justification provided initially incorrectly identified the correct answer. The updated explanation accurately identifies why options A and C must be cleared and why keeping "Participating" enabled is necessary.

Question: 222 CertyIQ

You have an Azure Automation account that contains a runbook. The runbook is used to configure the application infrastructure of an Azure subscription.
You have a project in Azure DevOps named Project1. Project1 contains a repository that stores code for the runbook.
You need to ensure that every committed change to the code will update automatically and publish the runbook to Azure Automation.
What should you configure?

A. the Service hooks settings for Project1
B. the Connections settings for the Automation account
C. the Source control settings for the Automation account
D. the Service connections settings for Project1

Answer: C

Explanation:

Here's a detailed justification for why the correct answer is C (the Source control settings for the Automation account), along with supporting concepts and authoritative links.

The goal is to automatically update and publish a runbook in Azure Automation whenever its code in an Azure DevOps repository is changed. This requires a direct connection between the Automation account and the source code repository.

Option A, "Service hooks settings for Project1," enables notifications about events in Azure DevOps (like code commits) to be sent to external services. However, service hooks only trigger events; they don't directly deploy the code into Azure Automation. They're more about notification and integration, not automatic deployment.

Option B, "Connections settings for the Automation account," are used to manage authentication information for interacting with other Azure services or external resources from within runbooks. They don't handle source code synchronization or deployment. Connections define how a runbook can access something, not where the runbook code itself comes from.

Option D, "Service connections settings for Project1," define connections that can be used by Azure Pipelines to deploy code, infrastructure, or other resources to various targets, including Azure resources. While potentially relevant for deploying the application infrastructure that the runbook configures, service connections don't directly link the runbook's source code repository to the Automation account for automated updates of the runbook itself. They're one step removed from the direct goal.

Option C, "Source control settings for the Automation account," is the correct choice because it allows you to directly link an Azure Automation account to a source control repository (like an Azure DevOps Git repository). By configuring source control, the Automation account can automatically synchronize with the repository. When changes are committed to the repository, the linked runbook in Azure Automation is automatically updated and published. This ensures that the latest version of the runbook code is always running in the Automation account. The "Source Control" feature within Azure Automation is specifically designed for this

scenario. It handles authentication and synchronization to keep the runbook's code consistent with the repository.

In summary, the Source Control integration feature within Azure Automation provides the direct, built-in capability to automatically update and publish runbooks whenever changes are committed to their corresponding code repository in Azure DevOps. This direct synchronization avoids the need for intermediate steps or custom scripting, providing an efficient and reliable solution.

Authoritative Links:

Azure Automation Source Control Integration: https://learn.microsoft.com/en-us/azure/automation/source-control-integration
Using Azure Automation Runbooks: https://learn.microsoft.com/en-us/azure/automation/learn/automation-tutorial-runbook-textual

Question: 223 CertyIQ

You use Git for source control.
You enable GitHub code scanning.
You raise a pull request from a non-default branch. In the code scanning output, you receive the following error message: “Analysis not found.”
You need to ensure that the code scanning completes successfully for the pull request.
Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Add the name of the default branch to the on: push specification in the code scanning workflow.
B. Add the name of the non-default branch to the on:push specification in the code scanning workflow.
C. Delete the pull request, and then raise the request again from the default branch.
D. Update the code in the pull request.
E. Add a new workflow for code scanning.

Answer: BD

Explanation:

The "Analysis not found" error in GitHub code scanning, specifically when raising a pull request from a non-default branch, indicates that the code scanning workflow is not configured to trigger for that branch. The default configuration often only includes scanning on the default branch (main or master).

To rectify this, we need to ensure the workflow is triggered by changes in the non-default branch and that any code changes related to the scanning might be necessary.

Option B, adding the name of the non-default branch to the on: push specification in the code scanning workflow, is a crucial step. The on: push section defines which branches trigger the workflow upon pushes. By including the non-default branch here, we instruct GitHub Actions to initiate the code scanning process whenever changes are pushed to that branch, which will then analyze the code in the pull request context.

Option D, updating the code in the pull request, might seem counterintuitive but addresses the possibility of the scanning tool not being able to find relevant code to analyse, or identifying errors in the code that prevent proper analysis. Sometimes, the code structure, dependencies, or the nature of changes in the pull request may cause the analysis to fail. By making a small update (even adding a whitespace change) and committing, we force a new trigger to the scanning and ensure the workflow is properly executed against the new code in the pull request.

Option A is incorrect because, given the problem description, scanning on the default branch already works.

Option C is inefficient. It's better to modify the workflow rather than restructure your branching strategy.

Option E is also wrong since the workflow already exists, we only need to modify it to include the branch.

Therefore, the combination of B (trigger the workflow on the non-default branch) and D (ensuring the latest code changes in the pull request are up-to-date) provides the best solution to ensure the successful completion of code scanning for the pull request.

Relevant links:

GitHub Actions documentation: https://docs.github.com/en/actions
GitHub Code Scanning documentation: https://docs.github.com/en/code-security/code-scanning

Question: 224 CertyIQ

DRAG DROP
-
You have a GitHub repository named repo1 that stores the code of an app named App1.
You need deploy a workflow for repo1 by using GitHub Actions. The solution must meet the following requirements:
• Scan on pushes to the main branch.
• Scan on pull requests to the main branch.
• Scan on pull requests to any branch that has a prefix of releases/.
• Scan all the files in the subdirectories of the src directory.
• Exclude scanning of markdown files.
How should you complete the code? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Values

- '**/*.md'
- '*.md'
- 'release*'
- 'releases/**'
- 'src/*'
- 'src/**'

Answer Area

```
...
on:
  push:
     branches: [main]
  pull_request:
     branches:
- main
     [ ]
     paths:
     [ ]
     paths-ignore:
     [ ]
...
```

Answer:

Answer Area

```
...
on:
  push:
    branches: [main]
  pull_request:
    branches:
- main
      - 'releases/**'
    paths:
      - 'src/**'
    paths-ignore:
      - '**/*.md'
...
```

Question: 225 CertyIQ

You have a GitHub repository that contains multiple versions of an Azure Pipelines template.
You plan to deploy multiple pipelines that will use a template stored in the repository.
You need to ensure that you use a fixed version of the template.
What should you use to reference which version of the template repository to use?

A. the serial
B. the SHA-based hashes
C. the runner
D. the branch

Answer: D

Explanation:

The correct answer is **D. the branch**. Here's a detailed justification:

When using templates in Azure Pipelines from a GitHub repository, specifying the branch name allows you to reference a specific version of the template. This ensures consistency across pipelines because they'll always

use the template code as it exists on that branch at the time of pipeline execution. Options A, B, and C are not suitable for this scenario.

Branch (D): By referencing a specific branch, like main or release/1.0, your pipelines will consistently use the template code available on that branch. If you need to update the template used, you can modify the code in that branch. This is a human-readable and manageable way to track template versions. For example, if you pin the template to the release/1.0 branch, pipelines will use the template version as of when that branch was created or last updated. This offers a good balance between controlling versions and making future updates easier.

Serial (A): Serial numbers are not a standard way to track versions in Git or GitHub.

SHA-based hashes (B): While SHA hashes pinpoint an exact commit in the repository's history, using a specific commit SHA makes updates extremely difficult. If you need to change the template, you'd have to update every pipeline referencing that specific SHA, which is highly impractical. SHA-based hashes are too granular for practical template management across many pipelines.

Runner (C): The runner is the agent that executes the pipeline jobs, and it does not have any relationship with the version of the templates stored in the repository.

Pinning to a branch simplifies version management while still allowing for updates. Changes to a branch like main can be controlled via pull requests, ensuring code review and controlled deployments. Using the branch approach is superior in many cases.

Authoritative Links:

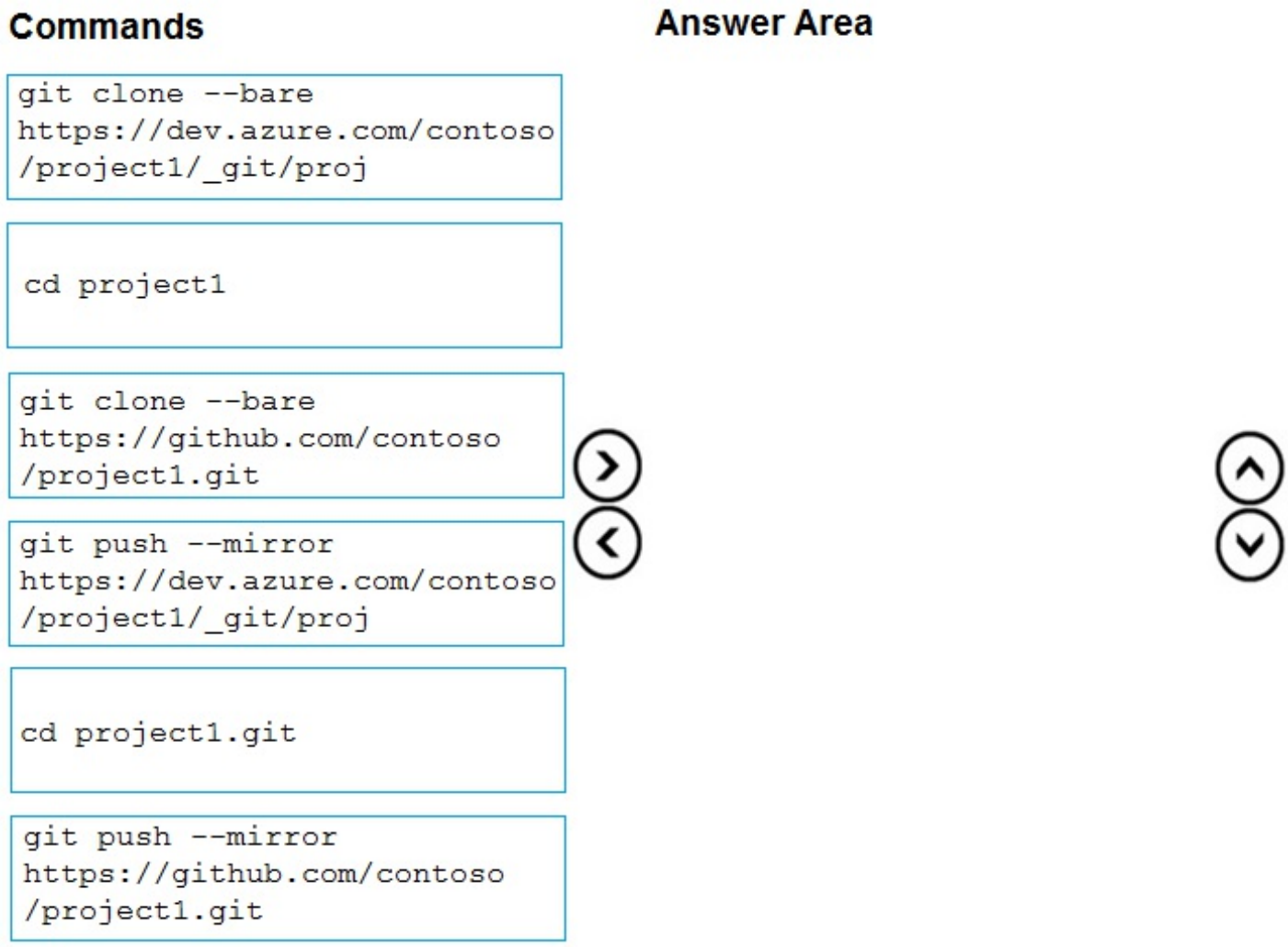
Azure Pipelines Templates: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/templates?view=azure-devops
Using Git Branches Effectively: https://www.atlassian.com/git/tutorials/using-branches

Question: 226 CertyIQ

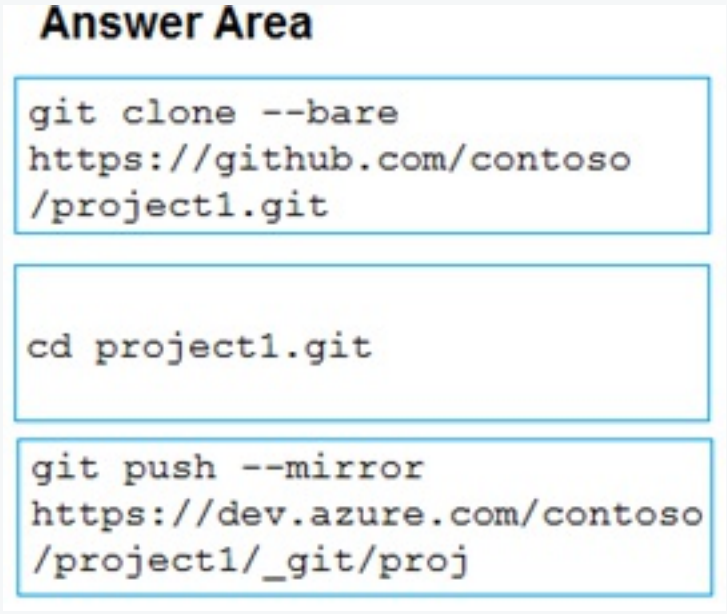
DRAG DROP
-
You have the repositories shown in the following table.

Type	URL
Azure Repos	https://dev.azure.com/contoso/project1/_git/project1.git
GitHub	https://github.com/contoso/project.git

You need to migrate the contents of the GitHub repository to the Azure Repos repository. The solution must ensure that the Azure Repos repository only contains branches and history from the GitHub repository.
Which three commands should you run in sequence? To answer, move the appropriate commands from the list of commands to the answer area and arrange them in the correct order.



Answer:



Question: 227 CertyIQ

DRAG DROP

-

You have a GitHub repository that contains the code for an app named App1.
App1 depends on a library of functions from a repository at https://github.com/contoso/afeed.
You need to keep a clone of the afeed repository as a subdirectory of the App1 repository.
How should you complete the Git command? To answer, drag the appropriate values to the correct targets. Each

value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Submodule add.

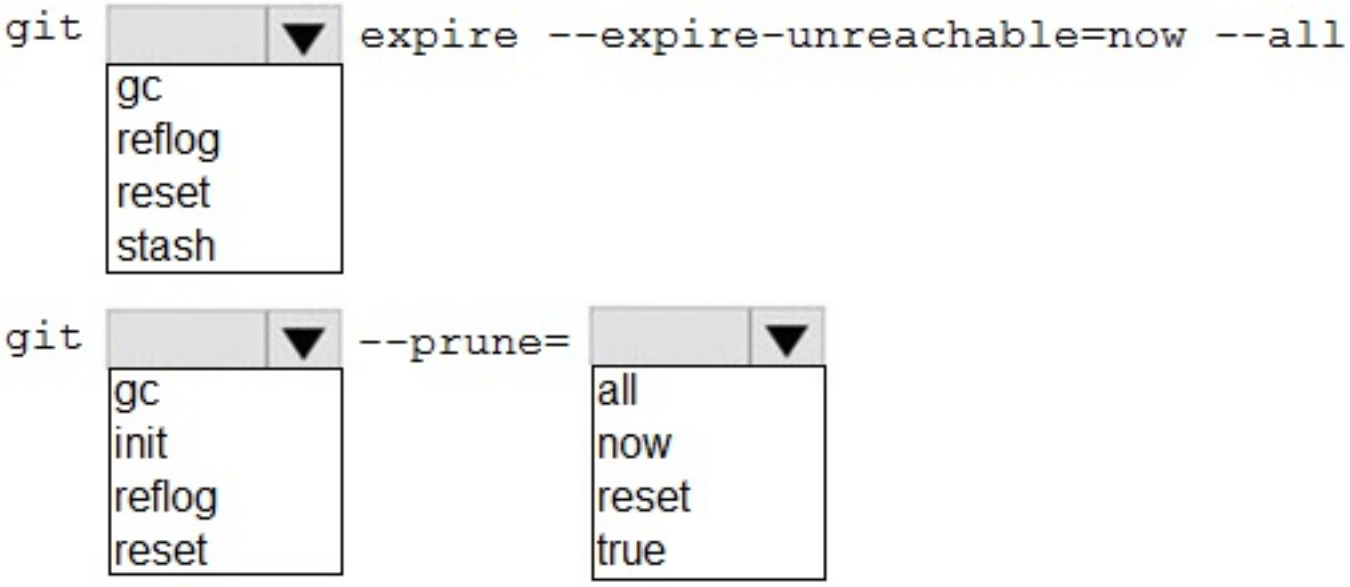
Reference:

https://git-scm.com/book/en/v2/Git-Tools-Submodules

Question: 228 CertyIQ

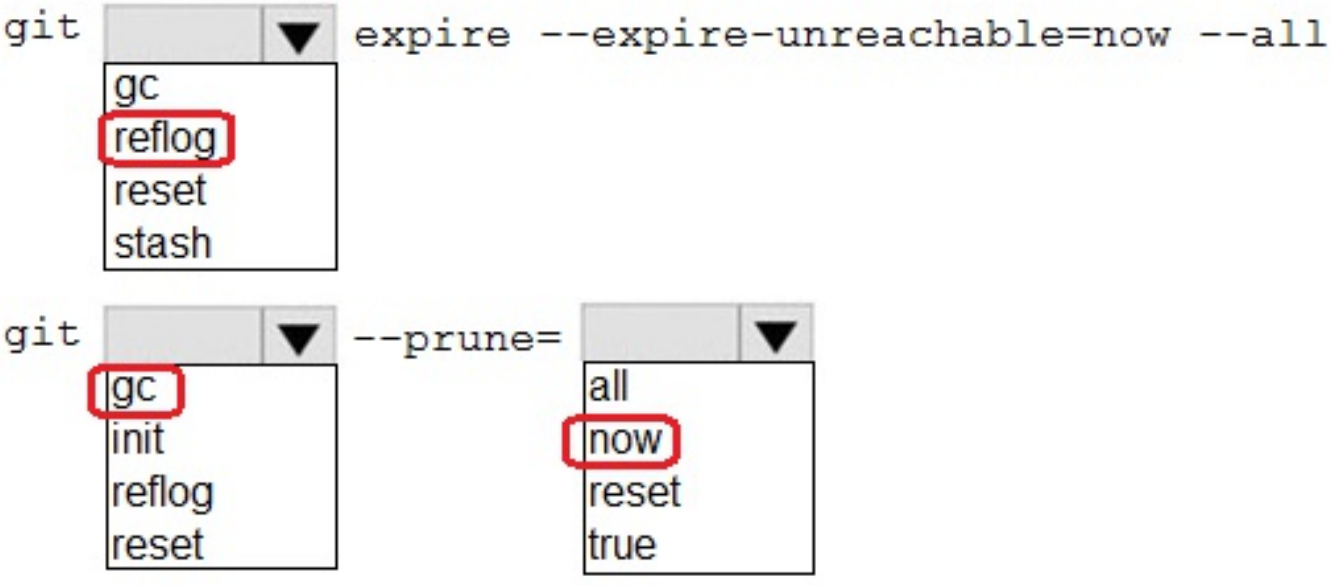
HOTSPOT
-
You use Git for source control.
You need to optimize the performance of a repository. The solution must meet the following requirements:
• Permanently remove all items referenced only in the reflog.
• Remove history that is NOT in any current branch.
How should you complete the command? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:

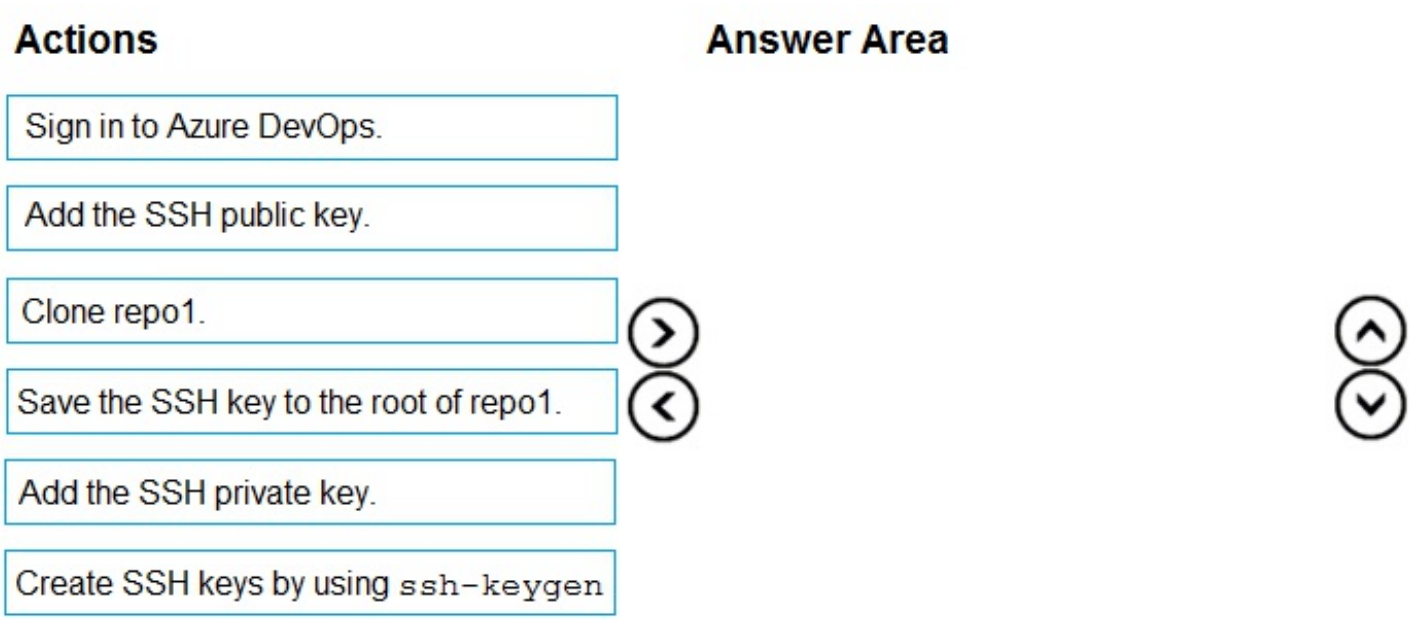
Answer Area



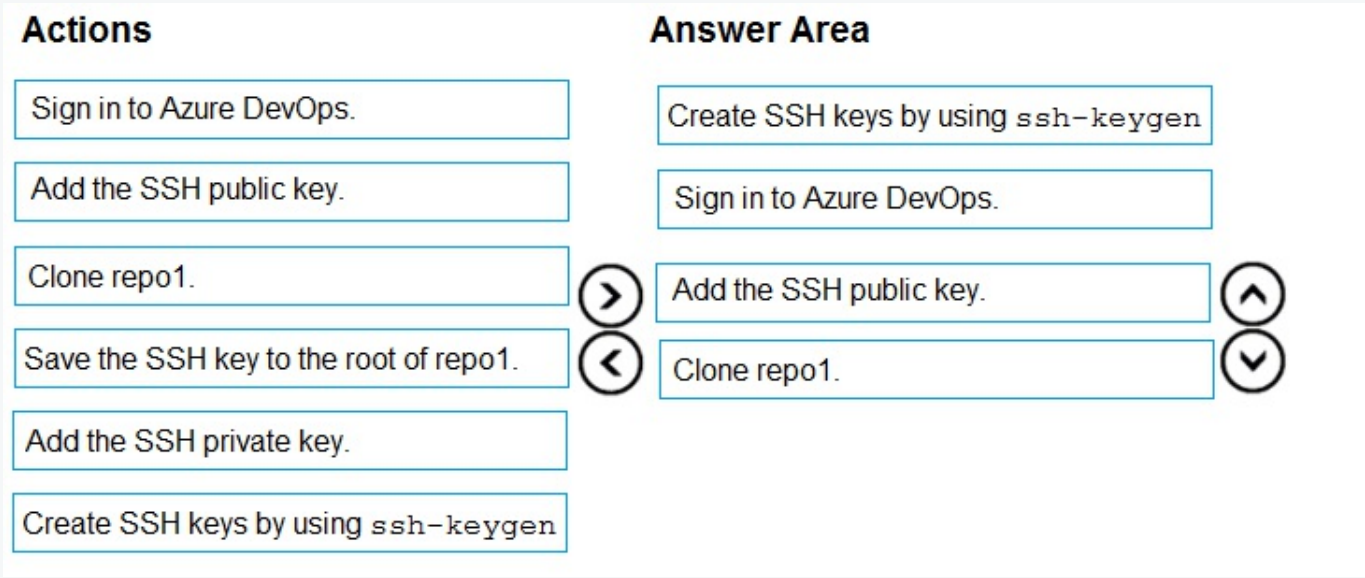
Question: 229

CertyIQ

DRAG DROP
-
You have an Azure Repos Git repository named repo1.
You need to ensure that you can authenticate to repo1 by using SSH.
Which four actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.



Answer:



Explanation:

1. Create SSH keys using ssh-keygen

2. Sign into Azure DevOps

3. Add the SSH public key

4. Clone repo1

Reference:

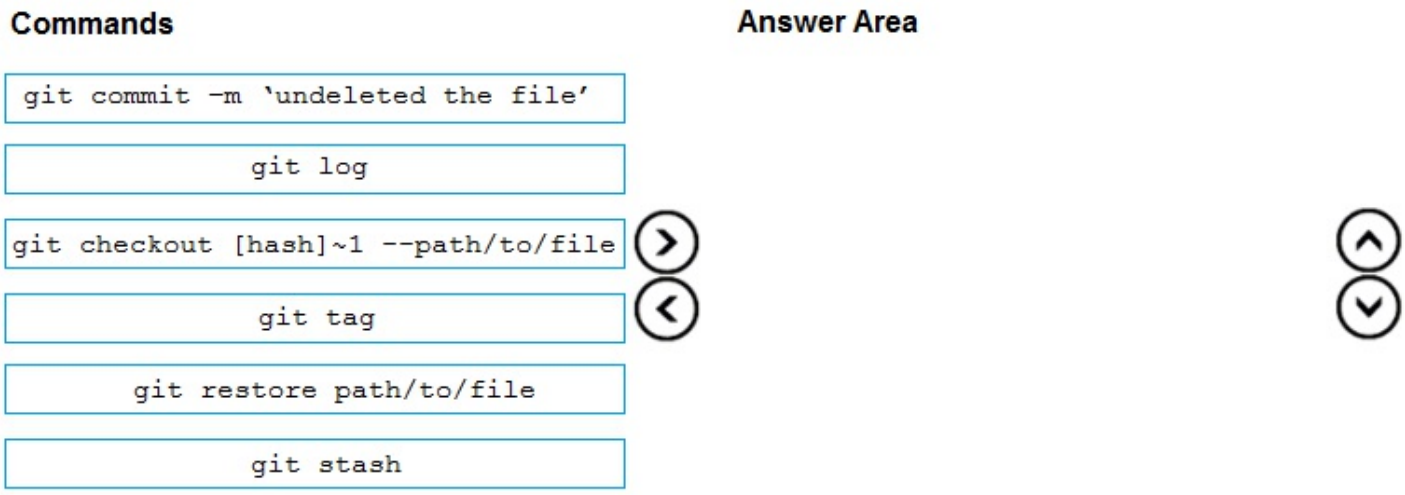
https://learn.microsoft.com/en-us/azure/devops/repos/git/use-ssh-keys-to-authenticate?view=azure-devops

Question: 230

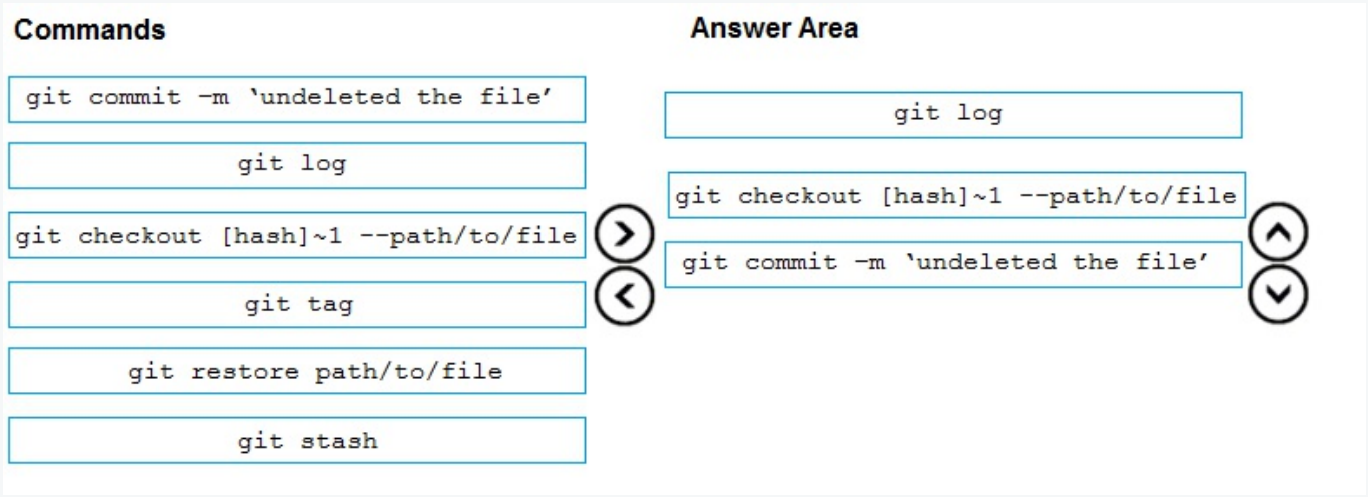
CertyIQ

DRAG DROP
-
You use Git for source control.
You delete a file, commit the changes, and continue to work.

You need to recover the deleted file.
Which three commands should you run in sequence? To answer, move the appropriate commands from the list of commands to the answer area and arrange them in the correct order.



Answer:



Explanation:

git log

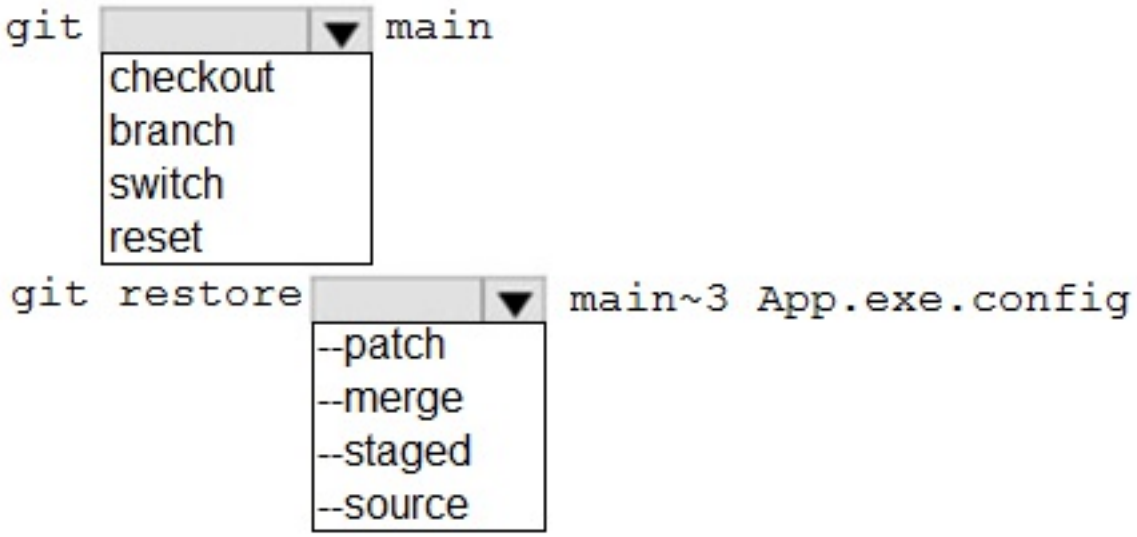
git checkout [hash]~1

git commit -m 'undeleted file'

Question: 231 CertyIQ

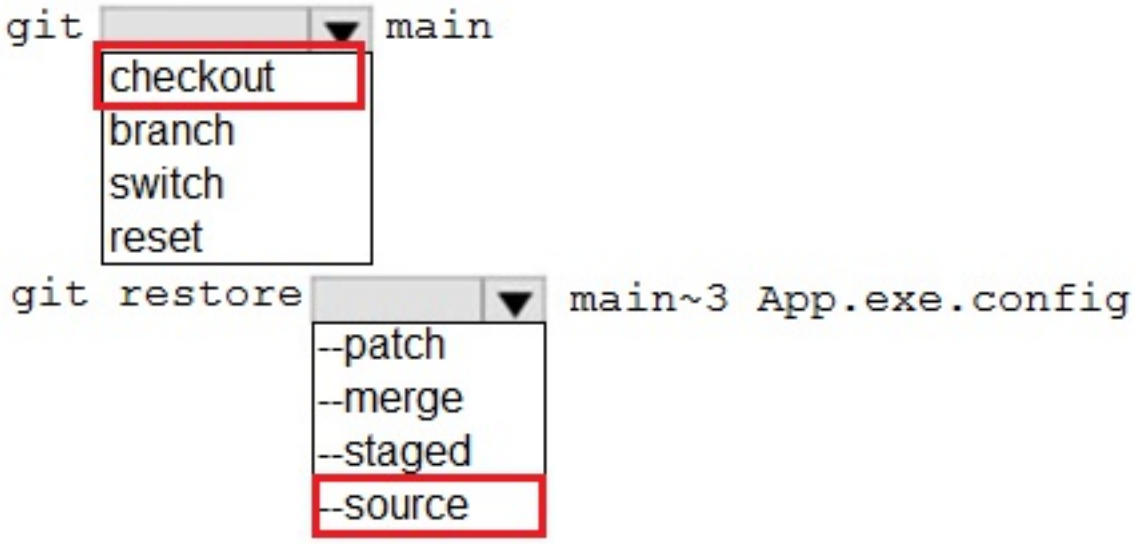
HOTSPOT
-
You use Git for source control. You have an app named App1.
In the main branch, you need to restore the third most recent revision of a file named App.exe.config.
How should you complete the command? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Explanation:

Checkout

Source

--source=<tree>

Restore the working tree files with the content from the given tree. It is common to specify the source tree by naming a commit, branch or tag associated with it.

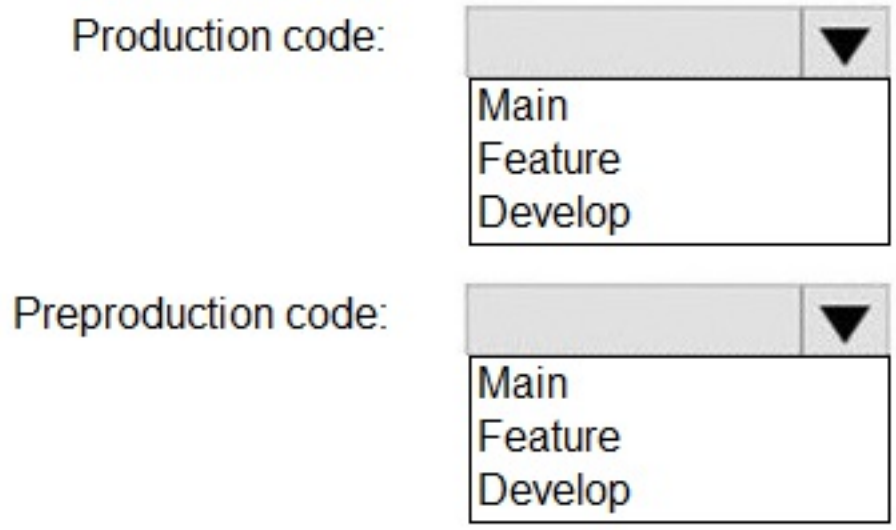
Reference:

https://git-scm.com/docs/git-restore#Documentation/git-restore.txt---sourcelttreegt

Question: 232 CertyIQ

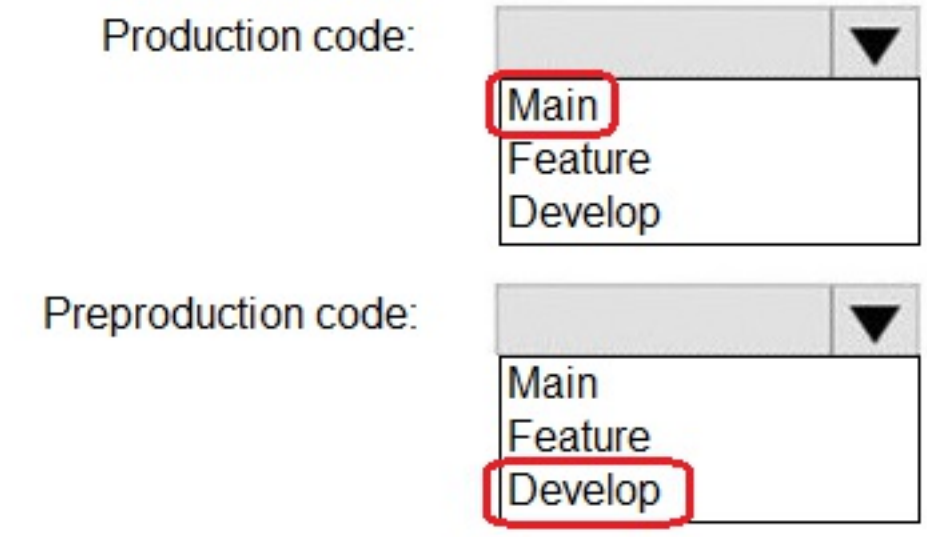
HOTSPOT
-
You company uses a Git source-code repository.
You plan to implement GitFlow as a workflow strategy.
You need to identify which branch types are used for production code and preproduction code in the strategy.
Which branch type should you identify for each code type? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Question: 233

CertyIQ

DRAG DROP
-
You have an Azure Repos repository named repo1.
You need to clone repo1. The solution must clone only a directory named src/web.
How should you complete the script? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

https://dev.azure.com/ms-scalar/_git/scalar$ cd scalar/src

Question: 234

CertyIQ

You use GitHub for source control and Microsoft Teams for collaboration.
You need to send a notification to a Teams channel for each commit. The solution must minimize development effort.
What should you do?

A. Use Azure Automation to connect to the GitHub Actions API and send a message to the Teams channel.

B. Use the Microsoft Teams for GitHub app and configure a subscription to receive notifications in the Teams channel.

C. Use GitHub Actions with a dispatch to send a message to the Teams channel by using the Teams API.

D. Use Azure Functions to connect to the GitHub REST API and send a message to the Teams channel.

Answer: B

Explanation:

The correct answer is B, using the Microsoft Teams for GitHub app. This solution minimizes development effort because it's a pre-built integration designed specifically for this scenario: sending GitHub notifications to a Teams channel. Instead of writing custom code using Azure Automation, Azure Functions, or GitHub Actions, you leverage an existing integration maintained by Microsoft.

The Teams for GitHub app allows you to subscribe to specific GitHub events, such as commits, pull requests, issues, and releases, and receive notifications directly in your designated Teams channel. Configuration is typically achieved through a user-friendly interface within Teams, avoiding the complexities of API calls and authentication management which are required for other options.

Options A, C, and D, while technically feasible, involve significantly more development effort. Azure Automation or Azure Functions (A and D) would require you to write custom code to listen for GitHub events via their APIs, process the event data, and then construct and send messages to the Teams channel using the Teams API. This entails coding, testing, and maintaining this custom integration. Option C, utilizing GitHub Actions, also necessitates writing YAML configuration and potentially scripting to interact with the Teams API, incurring development and maintenance overhead. Furthermore, these options demand managing authentication and authorization between GitHub, Azure services, and Teams, which adds complexity. The built-in app streamlines this process, handling authentication and event subscriptions behind the scenes, greatly reducing the effort needed. This is in alignment with the question's requirement to minimize development effort. Therefore, the Microsoft Teams for GitHub app is the most efficient solution.

Supporting Links:

Microsoft Teams for GitHub App: https://appsource.microsoft.com/en-us/product/office/wa104380513 (This page provides information and download details for the Microsoft Teams for GitHub app.)

Connect Microsoft Teams and GitHub: https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-github-account/managing-user-account-settings/connecting-your-github-account-to-microsoft-teams (Official GitHub documentation explaining how to link GitHub and Teams.)

Question: 235

CertyIQ

You manage source code control and versioning by using GitHub.
You need to ensure that a PowerShell script is executed automatically before rebase operations are performed.
What should you use?

A.a package
B.GitHub Copilot
C.a webhook
D.a gist

Answer: C

Explanation:

The correct answer is **C. a webhook**. Here's why:

Webhooks are automated HTTP callbacks triggered by specific events in a source code repository, like GitHub. They enable real-time communication between your repository and external services. In this scenario,

you want to trigger a PowerShell script execution before a rebase operation. A webhook configured to listen for pre-rebase events (though a direct "pre-rebase" hook might not exist, you could use a close proxy trigger) can then call an external service that executes the PowerShell script. This allows for automated checks, validations, or modifications before the rebase is committed.

A package (A) is a distributable unit of code, not an event-driven trigger. GitHub Copilot (B) is an AI-powered coding assistant and doesn't directly trigger scripts based on repository events. A gist (D) is a way to share code snippets, but it doesn't provide the automated triggering mechanism needed for this requirement.

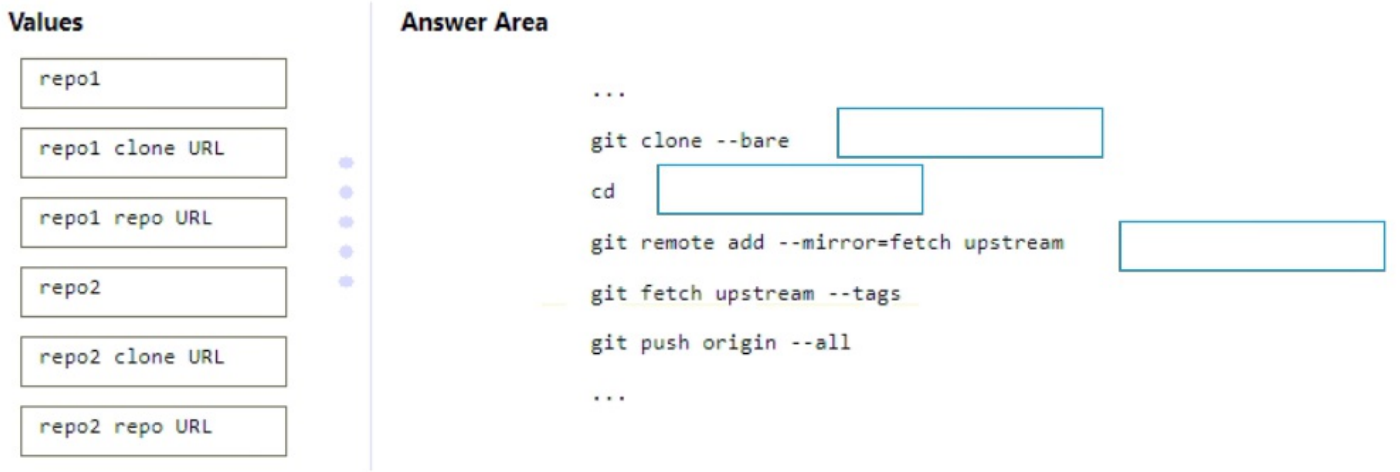
Webhooks provide the required automation, as they're designed to react to repository events, enabling the desired pre-rebase script execution. By configuring a webhook to trigger on an event closely related to rebasing, such as a push or a pull request event, you can then initiate the PowerShell script execution. The script can perform necessary checks or modifications before the rebase is finalized, ensuring code quality and adherence to specific standards.

Further research:

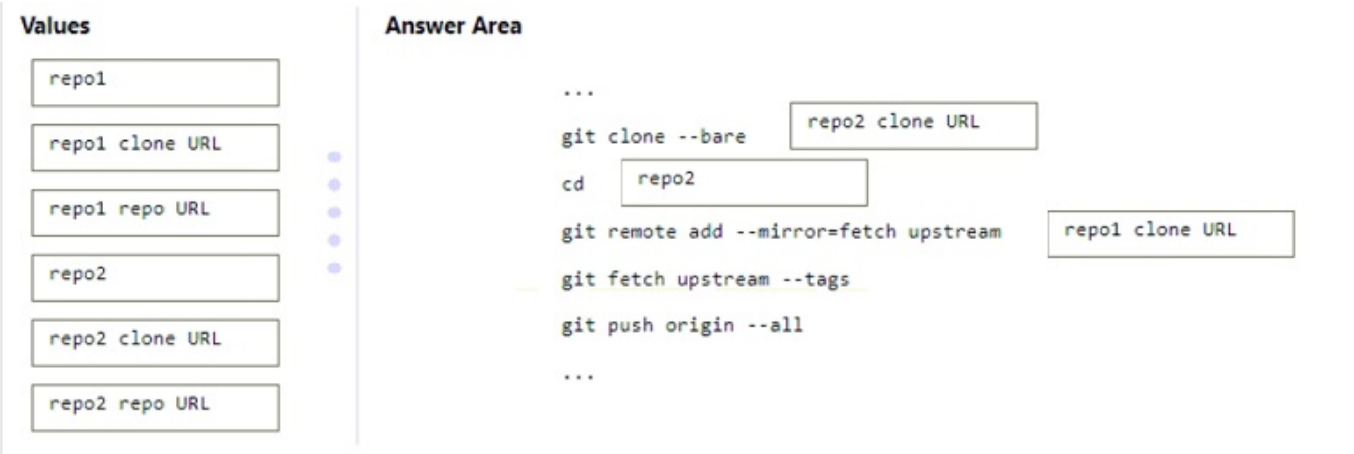
GitHub Webhooks Documentation: https://docs.github.com/en/webhooks
Understanding Webhooks: https://sendgrid.com/blog/what-are-webhooks/

Question: 236 CertyIQ

DRAG DROP
-
You have a GitHub repository named repo1.
You migrate repo1 to an Azure Repos repository named repo2.
After the migration, changes are made to repo1.
You need to sync the changes to repo2.
How should you complete the script? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



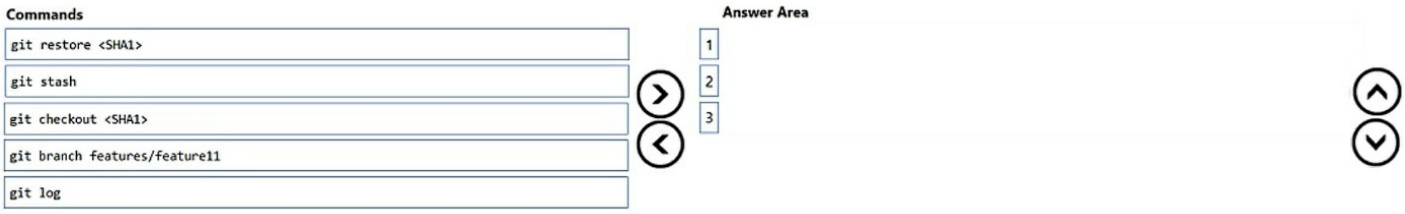
Explanation:

1. repo2_clone_url2. repo2 3. repo1_clone_url

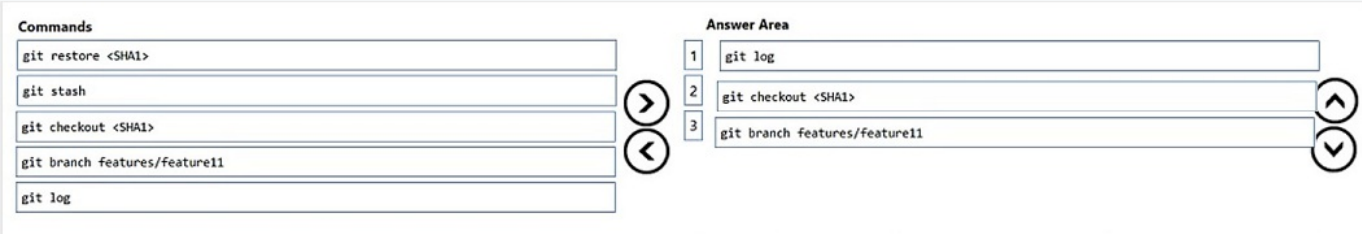
Question: 237

CertyIQ

DRAG DROP
-
You have an Azure Repos repository named repo1.
You delete a branch named features/feature11.
You need to recover the deleted branch.
Which three commands should you run in sequence? To answer, move the appropriate commands from the list of commands to the answer area and arrange them in the correct order.



Answer:



Explanation:

git log git checkout <SHA1>git branch features/feature11

Question: 238

CertyIQ

You use GitHub to host container packages that use Semantic Versioning (SemVer).
You have an app named App1. The current version of App1 is 11.2.0.
You change the code of App1 to fix a bug that was introduced in version 10.5.1.
Which version number should you assign to the release?

A.10.5.1-PATCH
B.11.2.1
C.10.5.2
D.10.6.0

Answer: B

Explanation:

The correct version number to assign to the release is 11.2.1. This choice adheres to Semantic Versioning (SemVer) principles. SemVer follows the MAJOR.MINOR.PATCH format. Since the change involves fixing a bug in the current version (11.2.0), it constitutes a patch. According to SemVer, a patch release increments the PATCH version number. Therefore, incrementing the PATCH number from 0 to 1 in 11.2.0 results in 11.2.1.

Options A, C, and D are inappropriate because they don't accurately reflect the nature and scope of the change relative to the current version. Option A uses a pre-release identifier, which is suitable for pre-release versions, not for bug fixes to a stable release. Options C and D suggest versions lower than the current version (11.2.0), implying a downgrade rather than a bug fix. Version 10.5.2 (Option C) might be relevant if the fix was only applied to older branches. Incrementing the MINOR version to 10.6.0 (Option D) suggests new functionality that isn't there.

The fix addresses a bug specifically in the existing 11.2.0 codebase. If the bug fix were also backported to a previous version (10.5.1), that would then have a different version number appropriate to that branch. However, this question is asking solely about the 11.2.0 release, and this version is only related to a fix that was initially introduced in a legacy version. Hence, increasing the PATCH version to 11.2.1 is the most appropriate and semantically correct approach according to SemVer.

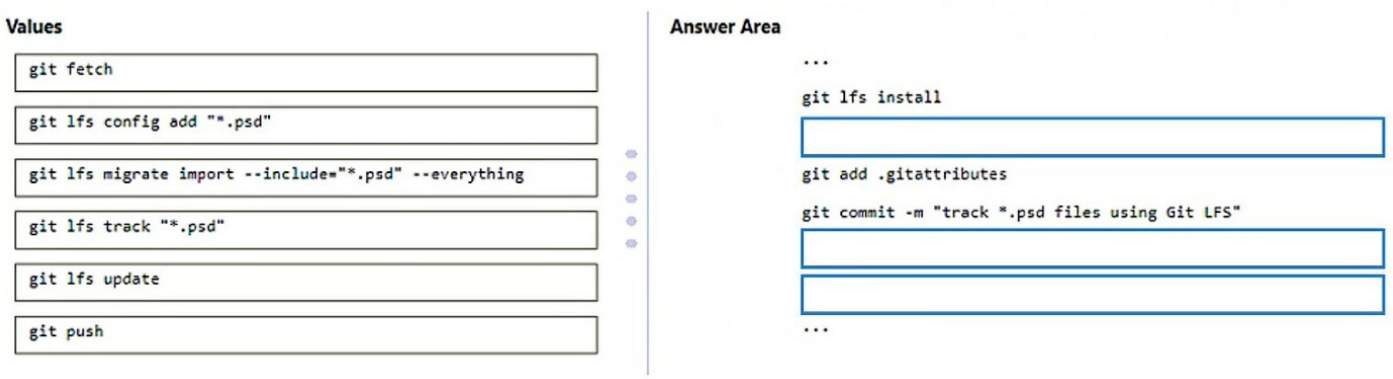
Further resources on Semantic Versioning can be found at:

Semantic Versioning Specification: https://semver.org/

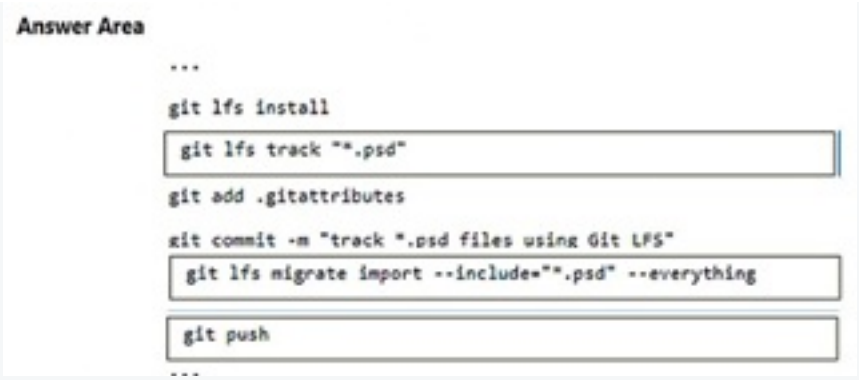
Question: 239

CertyIQ

DRAG DROP
-
You have an Azure Repos repository that contains large PSD files.
You need to configure Git LFS to manage all the files.
How should you complete the script? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

git lfs track "*.psd"

Git LFS to manage all .psd files.

Git LFS replaces large files with small text pointers inside Git, while storing the actual content in a separate LFS cache/server.

git lfs migrate import --include="*.psd" --everything.

Rewrites Git history to move all .psd files (from past commits too) into LFS.

If you already have .psd files in history, this command ensures they are stored in LFS retroactively.

git push.

Pushes all changes (including rewritten history and LFS-tracked files) to the remote.

Makes the LFS setup and migrated content available to collaborators.

Question: 240 CertyIQ

You manage code by using GitHub.
You need to ensure that repository owners are notified if a new vulnerable dependency or malware is found in their repository.
What should you do?

A.Configure CodeQL scanning actions.
B.Configure Dependabot alerts.
C.Configure branch protection rules for each repository.
D.Subscribe all the repository owners to the GitHub Advisory Database.

Answer: B

Explanation:

The correct answer is B: Configure Dependabot alerts. Here's why:

Dependabot is a GitHub feature specifically designed to monitor repositories for vulnerable dependencies and malware. It automatically detects outdated or insecure dependencies and notifies repository owners via alerts. These alerts provide information about the vulnerability, severity, and recommendations for remediation, often including pull requests to update the affected dependency.

Option A, CodeQL scanning actions, focuses on analyzing code for security vulnerabilities and code quality issues within the repository's code itself, not external dependencies. While valuable, it doesn't directly address the problem of vulnerable dependencies.

Option C, configuring branch protection rules, is useful for controlling how code changes are merged into protected branches, enforcing code review, and preventing accidental or malicious pushes. It does not detect vulnerable dependencies.

Option D, subscribing repository owners to the GitHub Advisory Database, would provide general security advisories, but it would require the owners to manually correlate those advisories with the specific dependencies in their repositories. Dependabot automates this process, making it much more efficient and practical. Dependabot proactively identifies and alerts about vulnerabilities specific to the dependencies used in the repository, requiring no manual cross-referencing. This proactive approach significantly reduces the risk of unknowingly using vulnerable dependencies.

In essence, Dependabot provides an automated, targeted solution to the stated problem, making it the most appropriate choice.

Relevant links:

About Dependabot alerts: https://docs.github.com/en/code-security/dependabot/dependabot-alerts/about-dependabot-alerts
About Dependabot security updates: https://docs.github.com/en/code-security/dependabot/dependabot-security-updates/about-dependabot-security-updates

Question: 241 CertyIQ

You have a public GitHub repository named Public1.
A commit is made to Public1. The commit contains a pattern that matches a regular expression.
Who is notified first when the commit is made?

A.the administrator of the GitHub organization
B.the committer
C.the owner of Public1
D.the secret scanning partner

Answer: D

Explanation:

Here's a detailed justification for why the correct answer is D (the secret scanning partner) and why the other options are less likely to be notified first when a commit containing a matching pattern (likely a secret) is made to a public GitHub repository:

The core concept at play here is **secret scanning**. GitHub has a feature (and partners with third-party vendors) to automatically scan public repositories for exposed secrets, such as API keys, passwords, and tokens. The purpose is to identify and remediate these leaks as quickly as possible to minimize potential damage.

Option A (the administrator of the GitHub organization) might eventually be notified, but only after the secret scanning partner detects the secret. The partner's primary role is immediate detection and reporting, not administrative notification. While administrators have overall oversight, they aren't the immediate responders.

Option B (the committer) is also unlikely to be notified first. The committer might receive a notification from GitHub later, informing them about the detected secret, but the immediate notification goes to the secret

scanning partner. The committer needs to be informed to take corrective action (e.g., revoke the leaked key and commit a new version without the secret). However, the scanning partner's detection is the trigger that initiates this alert.

Option C (the owner of Public1) might eventually be notified similar to option A. But it is not the first one to get notified. Secret Scanning Partner are configured to get notified upon any commits that contain secret.

Option D (the secret scanning partner) is the correct answer. These partners are actively monitoring public repositories for patterns that match known secrets. Their systems are designed for near real-time detection. They have direct integrations with GitHub and are the first line of defense against exposed credentials. The faster a secret is detected, the sooner it can be revoked and the potential damage mitigated. The scanning partner typically alerts GitHub, the repository owner, and potentially other relevant parties. But the partner's detection is the critical initial step.

In summary, while multiple parties will eventually be informed of a detected secret in a public GitHub repository, the secret scanning partner is the first notified due to their continuous monitoring and direct integration with GitHub's secret scanning functionality.

Authoritative Links:

GitHub Secret Scanning: https://docs.github.com/en/code-security/secret-scanning/about-secret-scanning
GitHub's blog post about secret scanning: https://github.blog/2022-05-04-secret-scanning-push-protection-is-now-available-for-all-public-repositories/

Question: 242 CertyIQ

You have a GitHub repository.
You need to ensure that all the code in the repository is scanned for vulnerabilities.
What should you use?

A.Dependabot alerts
B.branch protection rules
C.CodeQL actions
D.GitHub Advisory Database databases

Answer: C

Explanation:

The correct answer is **C. CodeQL actions**. Here's why:

CodeQL is a powerful static analysis engine developed by GitHub that performs semantic analysis on code to identify potential security vulnerabilities. It treats code as data, allowing developers to query it with a language similar to SQL. CodeQL actions can be integrated into GitHub Actions workflows to automatically scan code in a repository for vulnerabilities during pull requests or on a scheduled basis. This proactive approach helps developers identify and fix security issues early in the development lifecycle, minimizing the risk of deploying vulnerable code.

Dependabot alerts (A) primarily focus on identifying vulnerabilities in project dependencies. While valuable, they don't scan the code itself for vulnerabilities. Branch protection rules (B) provide mechanisms for enforcing code quality and review processes before merging code into protected branches but don't directly scan code for vulnerabilities. The GitHub Advisory Database (D) is a repository of known vulnerabilities and security advisories. While it provides valuable information, it doesn't actively scan a repository's code.

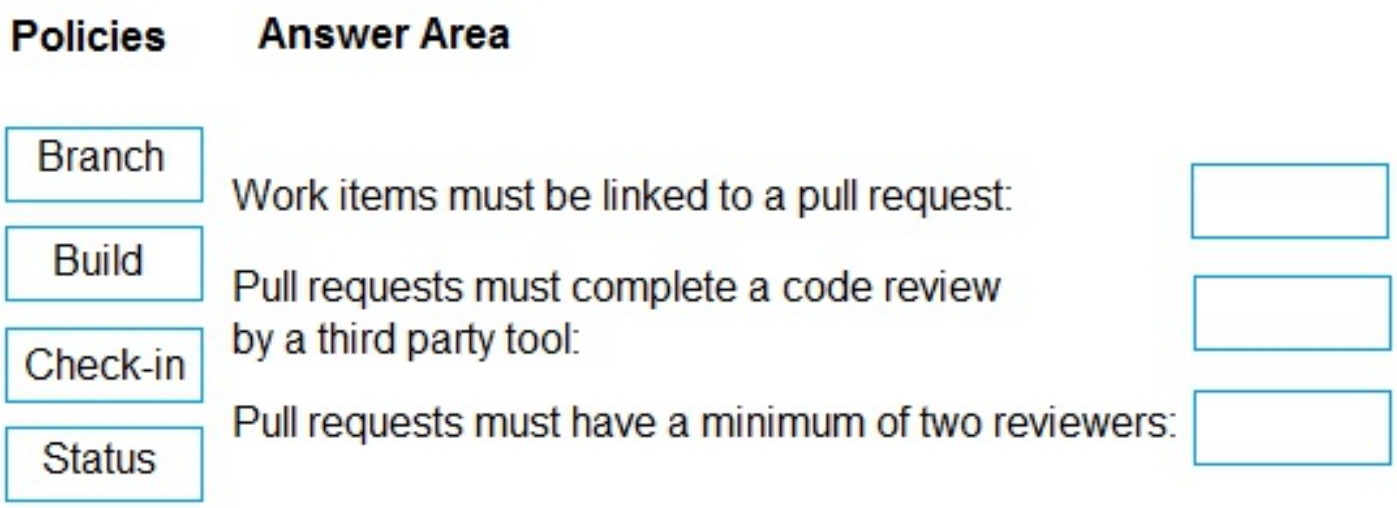
Therefore, CodeQL actions provide the required functionality of scanning the code within a GitHub repository for vulnerabilities, making it the most appropriate solution. It provides continuous analysis throughout the development process and helps in maintaining a secure codebase.

Here are authoritative links for further research:

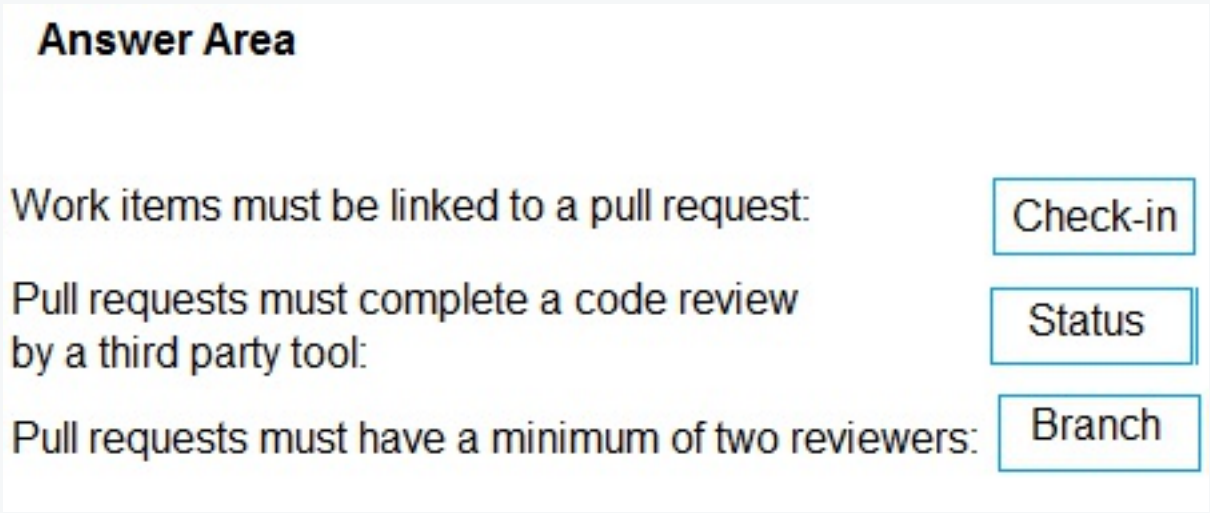
GitHub CodeQL: https://github.com/features/security/code-scanning
GitHub Actions: https://github.com/features/actions

Question: 243 CertyIQ

DRAG DROP
-
You create a Git repository named Repo1 in Azure Repos.
You need to configure Repo1 to meet the following requirements:
•Work items must be linked to a pull request.
•Pull requests must have a minimum of two reviewers.
•Pull requests must complete a code review by using a third-party tool.
The solution must minimize administrative effort.
Which type of policy should you use for each requirement? To answer, drag the appropriate policy types to the correct requirements. Each type policy may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Question: 244 CertyIQ

You have a 1-TB Azure Repos repository named repo1.
You need to clone repo1. The solution must meet the following requirements:
•You must be able to search the commit history of the /src directory
•The amount of time it takes to clone the repository must be minimized
Which command should you run?

A.git clone –-depth-1 :v3/org/Project1/repo1
B.git clone –-filter=blob:none :v3/org/Project1/repo1
C.git clone :v3/org/Project1/repo1
D.git clone –-filter=true:0 :v3/org/Project1/repo1

Answer: B

Explanation:

The correct answer is **B. git clone --filter=blob:none :v3/org/Project1/repo1**. Here's why:

The problem emphasizes minimizing clone time for a large repository (1 TB) while maintaining the ability to search commit history within the /src directory. Standard git clone downloads the entire repository history, including all blobs (file content), which can be time-consuming for a repository of this size.

Option A (git clone --depth-1) performs a shallow clone, retrieving only the most recent commit. While it minimizes clone time, it severely limits the commit history, making it unsuitable for searching the history of the /src directory.

Option C (git clone :v3/org/Project1/repo1) is a full clone. It downloads the complete repository, which is the slowest option.

Option D (git clone --filter=true:0 :v3/org/Project1/repo1) attempts to filter based on tree objects but it's not a correct or efficient way to optimize the clone for this specific scenario.

Option B (git clone --filter=blob:none) utilizes Git's partial clone feature. --filter=blob:none instructs Git to skip downloading the actual file content (blobs) during the initial clone. This significantly reduces the download size and, consequently, the cloning time. Importantly, it still downloads the commit history, tree objects (directory structure), and commit metadata. This enables the ability to search the commit history of the /src directory. When you need the content of a specific file, Git can then fetch the blob on demand. This is crucial for examining the history within /src. Since the problem statement asks that the commit history of the /src directory must be searchable, this option preserves the full history. By excluding blobs initially, the clone is significantly faster without sacrificing the ability to browse the history.

In summary, git clone --filter=blob:none provides the best balance between clone speed and the ability to explore the commit history, specifically for the /src directory, as required by the problem statement.

Authoritative Links:

Git Partial Clone: https://git-scm.com/docs/git-clone#_options_for_modifying_history
Git filter-branch (useful for understanding history manipulation): https://git-scm.com/docs/git-filter-branch

Question: 245 CertyIQ

You use GitHub for source control.
You are evaluating whether to use proxying to add a private upstream MyGet package feed to your MyGet feed.

What are two possible advantages of this approach? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A.minimizes the impact of upstream source availability issues
B.minimizes latency when accessing the package
C.provides automatic authentication
D.minimizes the impact on your storage quota

Answer: AC

Explanation:

The correct answer is A and C. Here's why:

A. Minimizes the impact of upstream source availability issues: Using a MyGet feed as a proxy to an upstream source (like NuGet.org or another MyGet feed) introduces a layer of caching. If the upstream source becomes temporarily unavailable, your MyGet feed can continue to serve packages it has already cached. This ensures that your builds and deployments are less likely to fail due to transient network or upstream service outages. This addresses dependency resilience, a key aspect of cloud application architecture. Without proxying/caching, a direct dependency on the upstream source means any downtime there directly impacts your processes.

C. Provides automatic authentication: MyGet allows you to configure authentication (e.g., credentials or API keys) for accessing the upstream feed within the MyGet configuration. Your developers and build agents then only need to authenticate with your MyGet feed. MyGet handles the upstream authentication behind the scenes. This simplifies the configuration of your build pipelines and reduces the risk of exposing upstream authentication credentials directly in your codebase or build scripts. Centralized authentication management enhances security posture.

Why B is incorrect: While a proxy can sometimes improve latency, it's not guaranteed. If the MyGet feed is geographically farther from the consumer than the original upstream feed, latency could increase. Furthermore, any latency introduced by the proxy's internal processing must be considered. Caching, described in A, indirectly helps with perceived latency by reducing the need to fetch from the upstream source.

Why D is incorrect: Proxying through MyGet does not directly minimize your storage quota. MyGet will store the packages it caches from the upstream feed, which can consume storage. In fact, it increases storage consumption in most cases because you're essentially storing the same package in two locations (upstream and your MyGet feed).

In summary: MyGet proxying enhances the reliability and security of package management by shielding your development processes from upstream availability issues and simplifying authentication management. The caching mechanism provides resilience, while authentication delegation improves overall security.

Supporting Links:

MyGet Feeds - Proxying: https://docs.myget.org/docs/reference/upstream-sources (This document explains upstream sources and proxying behavior within MyGet, explicitly mentioning caching.)
NuGet.org - Reliable Package Consumption: While not directly about proxying, understanding the challenges NuGet solves in package management is crucial. NuGet's own documentation indirectly highlights the benefits of package caching and mirrored repositories to improve reliability.

Question: 246 CertyIQ

You manage source code control and versioning by using GitHub.
A large file is committed to a repository accidentally.
You need to reduce the size of the repository. The solution must remove the file from the repository.
What should you use?

A.bfg
B.lfs
C.gvfs
D.init

Answer: A

Explanation:

The correct answer is **A. bfg**. Here's why:

BFG Repo-Cleaner is a faster, simpler alternative to Git's filter-branch command specifically designed for removing large or problematic files from a Git repository's history. It's particularly useful for situations where a large file has been committed and needs to be completely expunged to reduce repository size. BFG rewrites the Git history, completely removing the file from all commits where it existed. This ensures the file is truly gone and no longer contributes to the repository's size. It's designed to be more efficient than Git's native history rewriting tools for this specific task.

Let's analyze why the other options are not as suitable:

B. lfs (Git Large File Storage): LFS is designed for managing large files, not removing them after they've already been committed. It replaces large files with text pointers, storing the actual file content separately. It doesn't address the problem of an already-committed, excessively large file bloating the repository.

C. gvfs (Git Virtual File System): GVFS is a system designed by Microsoft, primarily for very large repositories. It virtualizes the filesystem to improve Git performance with massive codebases, but it doesn't provide a mechanism to remove files from history and reduce repository size. It focuses on improving the speed and scale of working with large repositories rather than cleaning them up.

D. init (git init): git init is a command used to create a new, empty Git repository. It's not relevant to removing files from an existing repository's history.

Therefore, BFG is the only tool designed specifically to surgically remove large files from Git history, fulfilling the requirement to reduce the repository size after a large file has been committed accidentally. It effectively rewrites the repository's history as if the file never existed in the first place.

Authoritative Links:

BFG Repo-Cleaner: https://rtyley.github.io/bfg-repo-cleaner/
Git Large File Storage (LFS): https://git-lfs.github.com/
GVFS (Git Virtual File System): Microsoft's implementation is now known as Scalar: https://github.com/microsoft/scalar

Question: 247 CertyIQ

DRAG DROP
-

You have a GitHub repository.
You need to configure Dependabot dependency scanning. The solution must meet the following requirements:
•Automatically open a pull request to resolve an alert.
•Automatically open a pull request when a dependency is updated.
What should you enable for each requirement? To answer, drag the appropriate features to the correct requirements. Each feature may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Features

- Alerts
- A dependency graph
- Security updates
- Version updates

Answer Area

Automatically open a pull request to resolve an alert: Feature

Automatically open a pull request when a dependency is updated: Feature

Answer:

Answer Area

Automatically open a pull request to resolve an alert: A dependency graph

Automatically open a pull request when a dependency is updated: Security updates

Question: 248 CertyIQ

DRAG DROP
-
You have a large repository named Repo1 that contains a directory named directory1.
You plan to modify files in directory1.
You need to create a clone of Repo1. The solution must minimize the amount of transferred data.
How should you complete the script? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Values

- git clone
- git fetch
- git sparse-checkout
- git worktree
- scalar clone
- scalar run

Answer Area

```
...
cd repos
                    https://dev.azure.com/organisation/_git/Repo1
                    set directory1
...
```

Answer:

Answer Area

```
...

cd repos

git clone          https://dev.azure.com/organisation/_git/Repo1

git sparse-checkout set directory1

...
```

Question: 249 CertyIQ

You have a project in Azure DevOps.
You need to implement a new branching solution. The solution must ensure that all pull requests meet the following requirements:
•Include linked work items.
•Pass build validation policies.
•Require at least three reviewers.
What should you include in the solution?

A.branch policies
B.pull request templates
C.branch security
D.pull request permissions

Answer: A

Explanation:

The correct answer is **A. branch policies**. Here's a detailed justification:

Branch policies in Azure DevOps are the mechanism designed to enforce specific requirements on pull requests before they can be merged into a target branch. They act as gatekeepers, ensuring code quality and adherence to project standards.

To meet the stated requirements:

Include linked work items: A branch policy can enforce that all pull requests are associated with one or more work items, providing traceability and linking code changes to specific tasks, user stories, or bug fixes.
Pass build validation policies: Branch policies can trigger build pipelines to run automatically whenever a pull request is created or updated. The pull request cannot be completed unless the build succeeds, ensuring that all code changes are tested and integrated successfully.
Require at least three reviewers: Branch policies allow you to specify a minimum number of required reviewers for each pull request. The pull request cannot be completed until the specified number of reviewers have approved the changes.

Pull request templates (B) are helpful for providing a consistent structure and guidance for pull request descriptions, but they don't enforce the stated requirements. Branch security (C) focuses on controlling who can create, modify, or delete branches, but doesn't directly address pull request validation. Pull request permissions (D) control who can perform actions on pull requests (e.g., create, approve, complete), but don't enforce the linked work item or minimum reviewer count requirements. Only branch policies comprehensively

address all given conditions.

In summary, branch policies are the right tool because they provide a robust and configurable mechanism to enforce code quality, traceability, and collaboration within the Azure DevOps branching strategy.

Further research:

Azure DevOps Branch Policies
Improve code quality with branch policies

Question: 250 CertyIQ

You use Git for source control.
You need to commit a 3-GB ZIP file that contains virtual machines used for testing. The solution must meet the following requirements:
•The file must be versioned.
•The file must be associated with the corresponding code commits.
Which two actions should you include in the solution? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A.Install the git-fat extension and associate the extension to ZIP files.
B.Install the Git LFS extension and associate the extension to ZIP files.
C.Install the git-stash extension and associate the extension to ZIP files.
D.Use GZip to compress the file before committing the file.
E.Store files in Azure Storage and enable blob versions.

Answer: BE

Explanation:

The correct answer is **B. Install the Git LFS extension and associate the extension to ZIP files.** and **E. Store files in Azure Storage and enable blob versions.**

Let's break down why:

Why Git LFS is crucial: Git is designed to efficiently handle text-based files and small binaries. Storing a large 3GB ZIP file directly within a Git repository bloats the repository, slowing down cloning, fetching, and other Git operations. Git LFS (Large File Storage) is an extension specifically designed to handle large files in Git. Instead of storing the large file content directly, Git LFS stores pointers or references to the actual file, which is stored separately (typically in a remote storage location). This keeps the Git repository lightweight while still versioning the large file. Associating the Git LFS extension with ZIP files ensures that any ZIP files added to the repository are handled by Git LFS. https://git-lfs.github.com/

Why Azure Storage with Blob Versions is important: While Git LFS manages pointers in the Git repository, the actual file needs to be stored somewhere. Azure Blob Storage provides scalable and durable object storage, perfect for storing large files like the 3GB ZIP. Enabling Blob Versioning ensures that every version of the ZIP file is preserved. This aligns with the requirement of versioning the file and associating it with code commits. When using Git LFS with Azure DevOps, the LFS objects are usually stored in Azure Blob Storage. https://learn.microsoft.com/en-us/azure/storage/blobs/versioning-overview

Why other options are incorrect:

A. Install the git-fat extension and associate the extension to ZIP files.: git-fat is another extension for handling large files, similar in concept to Git LFS, but less widely adopted and not natively integrated with

Azure DevOps in the same way. Git LFS is the preferred solution in the Azure DevOps ecosystem.

C. Install the git-stash extension and associate the extension to ZIP files.: git-stash is used to temporarily save changes in the working directory, not for managing large files.

D. Use GZip to compress the file before committing the file.: Compressing the file helps reduce its size, but it does not solve the fundamental problem of Git's inefficiency in handling large binary files directly within the repository. The compressed file would still bloat the repository. Moreover, the VM might already be compressed.

In summary, Git LFS provides a version control solution for large files in conjunction with Git by using pointers, and Azure Storage allows you to efficiently store these large files associated with the LFS repository. Enabling blob versions gives the needed version control.

Question: 251 CertyIQ

You manage source control by using GitHub.
You have a file named Data.txt that contains sensitive data. A user pushes Data.txt to a repository.
You need to purge the file from the repository.
Which two commands can you use? Each correct answer presents a complete solution.
NOTE: Each correct solution is worth one point.

A.git checkout
git reset -hard -pathspec data.txt

B.git checkout
git clean -d data.txt --force

C.bfg --delete-files data.txt
git push --force

D.git rm data.txt
git push --force

E.git filter-repo --invert-paths --path data.txt
git push origin --force --all

F.git revert -edit data.txt
git push -force

Answer: C

Explanation:

The correct answer is **C and E**.

Justification:

The scenario requires completely removing a sensitive file (Data.txt) from a GitHub repository's history, not just the latest version. Options A, B, D, and F only address local modifications or current working state and won't purge the file from the entire history visible to collaborators. They modify the current state rather than rewriting the history.

Option C (bfg --delete-files data.txt; git push --force):

BFG Repo-Cleaner: BFG Repo-Cleaner is a powerful tool designed specifically for cleaning up Git repository history. bfg --delete-files data.txt finds and completely removes all traces of Data.txt from every commit in the repository's history, including past commits and branches. This is essential for sensitive data removal.
git push --force: After BFG has rewritten the history, you need to force-push these changes to the remote

repository (GitHub). git push --force overwrites the remote repository's history with your local, cleaned-up version. This is necessary because Git normally prevents you from pushing changes that rewrite history for safety reasons, but in this case, it's intentional.

Option E (git filter-repo --invert-paths --path data.txt; git push origin --force --all):

git filter-repo: This Git command allows you to rewrite the history of a repository based on filtering criteria.
--invert-paths --path data.txt: This specific argument instructs git filter-repo to include everything except the specified path (Data.txt) in the rewritten history. Effectively, it removes all traces of the file.
git push origin --force --all: Similar to option C, this command force-pushes the rewritten history to the remote repository. --all ensures that all branches are updated.

Why the other options are incorrect:

A (git checkout; git reset -hard -pathspec data.txt): git reset only modifies the current branch's history from a specific point. It doesn't touch older commits where the file might still exist. git checkout simply switches branches, not changing repository history.
B (git checkout; git clean -d data.txt --force): git clean removes untracked files from your working directory. It doesn't affect the committed history of the repository.
D (git rm data.txt; git push --force): git rm only removes the file from the current commit. It doesn't delete it from past commits. While you could force-push, the file still exists in the repository's history.
F (git revert -edit data.txt; git push -force): git revert creates a new commit that undoes the changes introduced by the commit(s) where Data.txt was added. It doesn't delete the file's history; it simply adds a new commit that removes it from the current state.

Authoritative Links:

BFG Repo-Cleaner: https://rtyley.github.io/bfg-repo-cleaner/
git filter-repo: https://git-scm.com/docs/git-filter-repo
GitHub Documentation on Removing Sensitive Data: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/removing-sensitive-data-from-a-repository

In summary, options C and E are the only ones that effectively and completely remove the sensitive file from the repository's history, which is the critical requirement of the question.

Question: 252 CertyIQ

You use GitHub Enterprise for source control repositories. The repositories store C# code.
You need to enable CodeQL scanning for the repositories.
What should you do?

A.Enable Dependabot security updates.
B.Enable Dependabot alerts.
C.Configure a required GitHub Actions workflow for all the repositories.
D.Push a GitHub Actions workflow to all the repositories.

Answer: C

Explanation:

The correct answer is C: Configure a required GitHub Actions workflow for all the repositories. Here's why:

CodeQL scanning is a static analysis tool that identifies security vulnerabilities in code. To enable CodeQL for GitHub Enterprise repositories, you need to integrate it into the development workflow. This integration is typically achieved through GitHub Actions.

Option C, configuring a required GitHub Actions workflow, ensures that CodeQL scanning automatically runs for every repository that adheres to the configuration. Required workflows enforce a specific set of actions, including CodeQL analysis, on all repositories that are configured to use them. This ensures consistent security scanning across all your C# repositories. When a developer attempts to push changes to a repository subject to a required workflow, the configured action is executed on the pushed code.

Option D, pushing a GitHub Actions workflow to all repositories, necessitates manual intervention for each repository. While functional, it doesn't guarantee consistent implementation and maintenance, especially if new repositories are added over time. It's also susceptible to accidental deletion or modification of the workflow.

Options A and B are incorrect because Dependabot focuses on dependency management and vulnerability alerts within dependencies, not directly on scanning the repository's code with tools like CodeQL. Dependabot helps identify outdated or vulnerable dependencies, but doesn't provide the same level of security analysis as CodeQL. While Dependabot alerts can be valuable for overall security, they don't satisfy the requirement of enabling CodeQL scanning.

By implementing a required GitHub Actions workflow for CodeQL, you establish a standardized and automated process that helps to identify and mitigate potential security vulnerabilities in your C# code residing in GitHub Enterprise. This promotes a more proactive and comprehensive security posture for your applications.

Further research:

GitHub Actions: https://docs.github.com/en/actions
CodeQL: https://codeql.github.com/
Required Workflows: https://docs.github.com/en/enterprise-cloud@latest/organizations/managing-organization-settings/managing-required-workflows
GitHub Advanced Security Overview: https://docs.github.com/en/get-started/learning-about-github/github-advanced-security

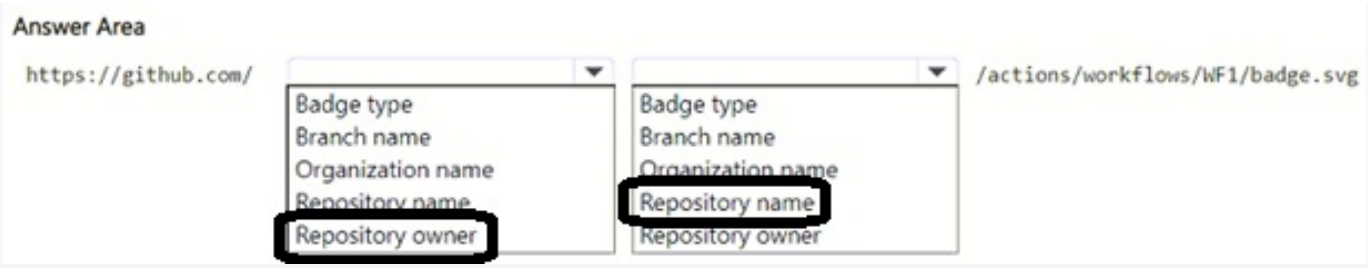
Question: 253 CertyIQ

HOTSPOT
-
You have a GitHub repository that contains a workflow named WF1. WF1 is used to build code for an app named App1.
You need to add a build status badge to the README.md file in the repository for App1.
How should you complete the URL for the badge? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:



Question: 254 CertyIQ

You have a GitHub repository.
You need to create a tag named v3.0.5 and ensure that the tag is available in the remote repository.
Which two commands should you run? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A.git push -force
B.git push origin v3.0.5
C.git tag v3.0.5
D.git commit -m ‘tag v3.0.5’
E.git add ‘tag v3.0.5’

Answer: BC

Explanation:

The correct solution involves two commands to create and push a tag to a remote GitHub repository.

First, the git tag v3.0.5 command (option C) creates a tag named "v3.0.5" locally within your Git repository. Tags are essentially pointers to specific commits, marking significant points in the project's history, such as releases. This local tag is created based on the current commit the HEAD is pointing to.

Second, the git push origin v3.0.5 command (option B) pushes the newly created tag to the remote repository, typically named "origin". Without this push, the tag only exists locally. This command specifically pushes the "v3.0.5" tag. This makes the tag visible and accessible to other collaborators.

Option A, git push -force, is generally discouraged unless absolutely necessary. While it would push all tags to the remote including the new one, it overwrites the remote branch with your local one, potentially causing data loss if other collaborators have made changes that you don't have. It's an unnecessarily forceful approach for simply pushing a tag.

Options D and E, git commit -m ‘tag v3.0.5’ and git add ‘tag v3.0.5’, are incorrect in this context. Tags are not added and committed like files. Tags are created using the git tag command. Creating a commit message with “tag v3.0.5” does not create a tag. You use commits to mark changes to tracked files.

In summary, creating a tag and pushing it to the remote repository are two distinct steps. You first create the tag locally, and then you explicitly push it to the remote. This ensures that the tag is visible and available to other collaborators on the remote repository.

Supporting Documentation:

Git Tagging: https://git-scm.com/book/en/v2/Git-Basics-Tagging
git push: https://git-scm.com/docs/git-push

Question: 255 CertyIQ

You have a GitHub repository.
You need to ensure that all changes to code are validated by your company's security department before the main branch is deployed.
Which two actions can you perform? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A.Require signed commits.

B.Create a branch protection rule for the feature branches.

C.Create a LICENSE file.

D.Create a branch protection rule for the main branch.

E.Create a CODEOWNERS file.

Answer: AE

Explanation:

The correct answer is AE because it directly addresses the requirement of validating code changes by the security department before deployment to the main branch.

A. Require signed commits: Requiring signed commits ensures that each commit in the repository can be traced back to a verified author. This prevents impersonation and allows the security department to confirm the origin and integrity of the code being reviewed. If a commit isn't signed by an authorized user, the branch protection rules can prevent it from being merged. This validation is essential for confirming who is making changes and holding them accountable. [https://docs.github.com/en/authentication/managing-commit-signature-verification/about-commit-signature-verification]

E. Create a CODEOWNERS file: A CODEOWNERS file specifies individuals or teams responsible for specific files or directories within the repository. By assigning the security department as code owners for the main branch or relevant files, you ensure they are automatically notified of any pull requests targeting those areas. This triggers the validation process. The security team then becomes mandatory reviewers and can block the merge until all their concerns are addressed. This ensures the security team always validates changes before they reach the main branch. [https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-code-owners]

Why other options are incorrect:

B. Create a branch protection rule for the feature branches: While useful, protecting feature branches doesn't directly guarantee security validation before merging into main.
C. Create a LICENSE file: A LICENSE file deals with legal aspects and licensing of the code; it is unrelated to security validation.
D. Create a branch protection rule for the main branch: This is partially correct. Branch protection rules are needed, but simply protecting the main branch without specifying who validates the code (using CODEOWNERS) and verifying the commit origin (using signed commits) is insufficient. This option needs to be paired with A and/or E to fulfil the prompt's requirements.

In summary, combining signed commits and code owners ensures that the security department is involved in validating code changes before they reach the main branch, fulfilling the problem's requirement. Signed commits prove authorship, while CODEOWNERS triggers the security review.

Question: 256

CertyIQ

HOTSPOT
-
You have a .NET app named App1.
You need to upload App1 to GitHub Packages.
How should you complete the command? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Question: 257

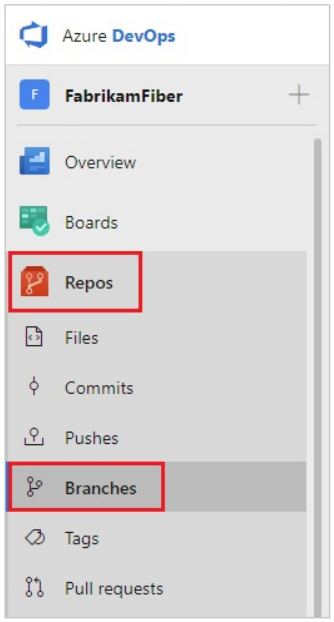
CertyIQ

SIMULATION
-
Initialize the default main branch, if it does not exist already.
In the User1-12345678 Azure DevOps project, you need to implement an approval process for the default main branch.
The process must use the four-eyes principle and have at least one approval on every iteration.

Answer:

Azure DevOps, Azure Repos, Branch policies and settings

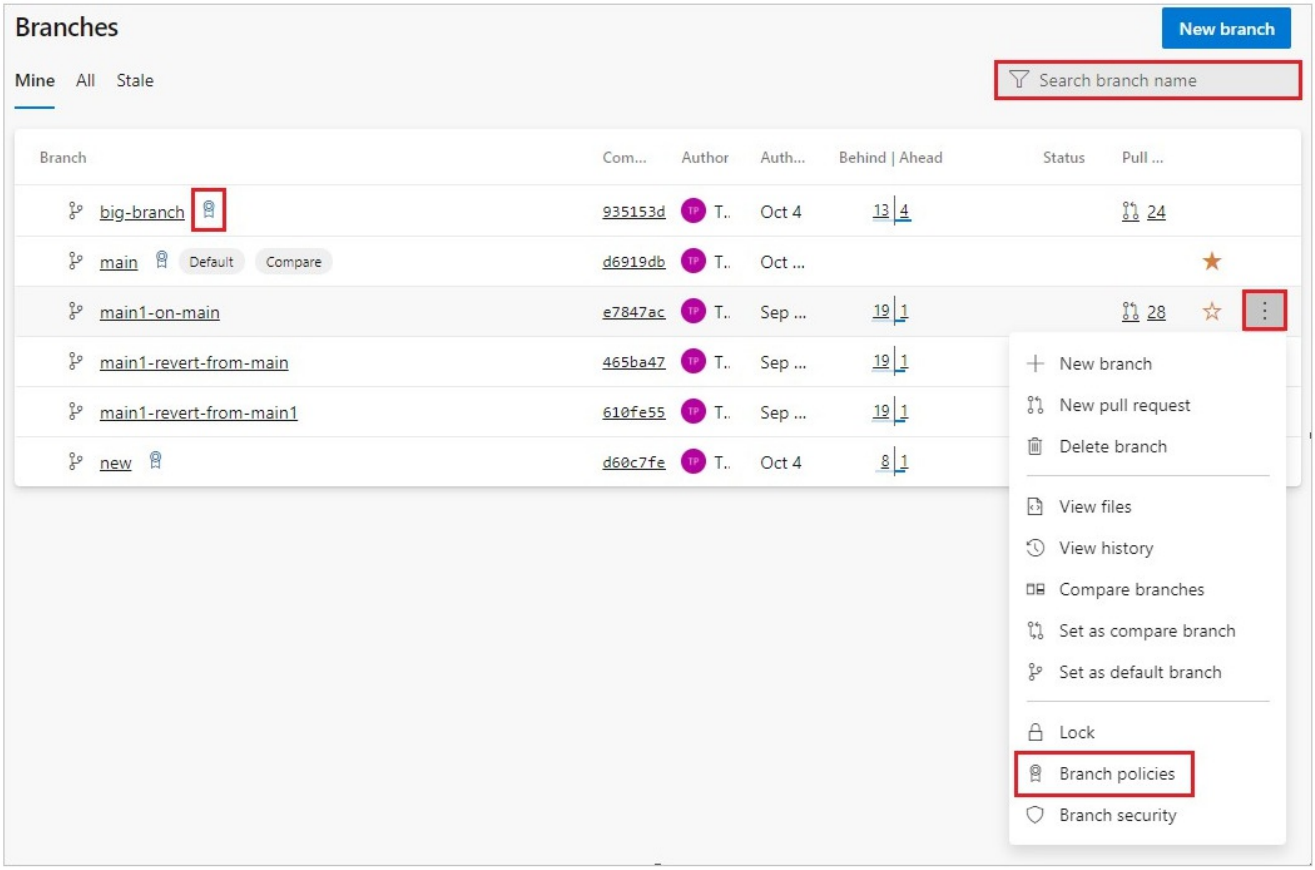
Configure branch policies
Phase 1. Select the correct branch
Step 1: To manage branch policies, select Repos > Branches to open the Branches page in the web portal.



(You can also get to branch policy settings with Project Settings > Repository > Policies > Branch Policies > <Branch Name>.)

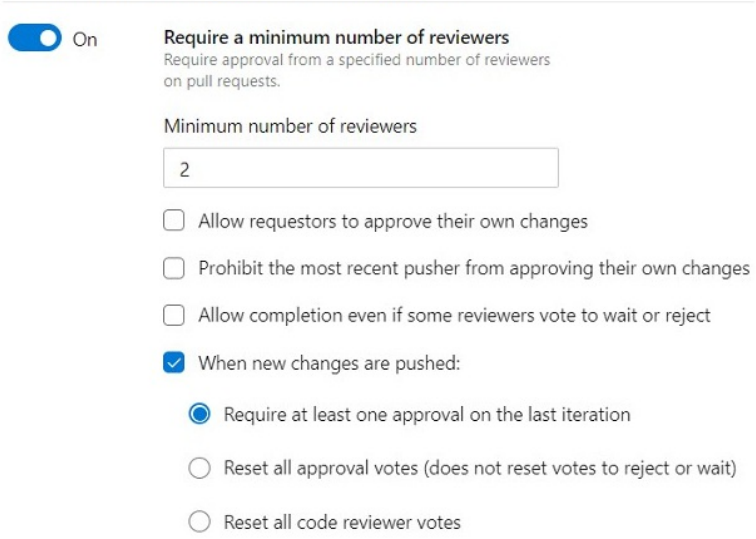
Step 2: To set branch policies, locate the branch you want to manage. You can browse the list or search for your branch in the Search branch name box at upper right. [Select the default main branch]

Select the More options icon next to the branch, and then select Branch policies from the context menu.



Phase 2: Configure approval
Require a minimum number of reviewers
Code reviews are important for software development projects. To ensure that teams review and approve PRs, you can require approval from a minimum number of reviewers. The basic policy requires that a specified number of reviewers approve the code, with no rejections.

Step 3: To set the policy, under Branch Policies, set Require a minimum number of reviewers to On. Enter the required number of reviewers, and select any of the following options:



3a. [Select: Minimum number of reviewers: 2]
3b. [Select: When new changes are pushed:]
3c [Select Require at least one approval on the last approval of the last iternation]

Reference:
https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies

Question: 258

CertyIQ

SIMULATION

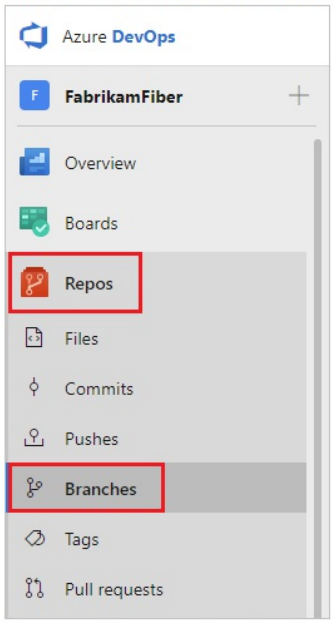
-

Initialize the default main branch, if it does not exist already.
For Project1, you need to implement branch merging restrictions to enable squash merge for all changes merged into the main branch.

Answer:

Select the correct branch

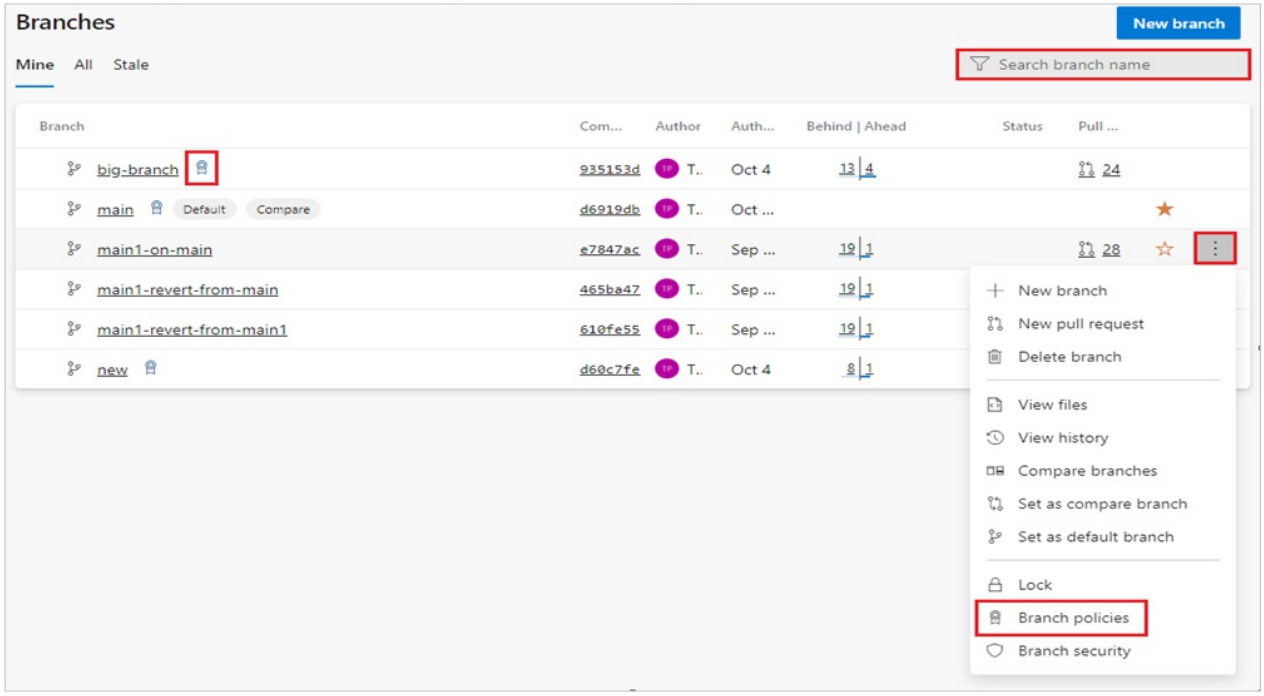
Step 1: To manage branch policies, select Repos > Branches to open the Branches page in the web portal.



(You can also get to branch policy settings with Project Settings > Repository > Policies > Branch Policies > <Branch Name>.)

Step 2: To set branch policies, locate the branch you want to manage. You can browse the list or search for your branch in the Search branch name box at upper right. [Select the default main branch]

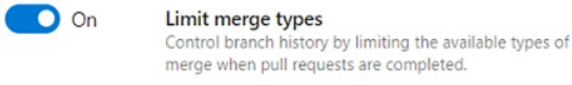
Select the More options icon next to the branch, and then select Branch policies from the context menu.

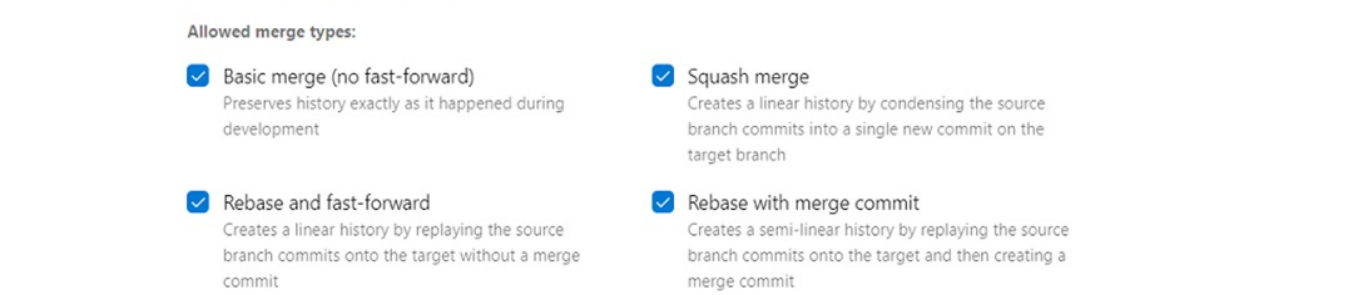


Phase 2: Limit merge types

Azure Repos has several merge strategies, and by default, all of them are allowed. You can maintain a consistent branch history by enforcing a merge strategy for PR completion.

Step 3: Set Limit merge types to On to limit which merge types to allow in your repo.





Step 4: Select Squash merge, deselect the other Allowed Merge types.

Squash merge creates a linear history with a single commit in the target branch with the changes from the source branch. Learn more about squash merging and how it affects branch history.

Reference:
https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies

Question: 259 CertyIQ

DRAG DROP
-
You have a GitHub repository named repo1 and an Azure key vault named kv1.
In repo1, you plan to create a workflow named Workflow1 that will deploy a database server by using credentials stored in kv1.
You need to ensure that Workflow1 can retrieve the credentials from kv1.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.

Actions	Answer Area
Grant secret permissions to kv1.	
Reference the credentials in Workflow1.	
Create a personal access token (PAT) in GitHub.	
Grant key permissions to kv1.	
Create a service principal in Microsoft Entra ID.	

Answer:

Answer Area

Create a service principal in Microsoft Entra ID.

Grant secret permissions to kv1.

Reference the credentials in Workflow1.

Question: 260 CertyIQ

Your company has 60 developers who are assigned to four teams. Each team has 15 members.
The company uses an agile development methodology.
You need to structure the work of the development teams so that each team owns their respective work while working together to reach a common goal.
Which parts of the taxonomy should you enable the team to perform autonomously?

A. Features and Tasks
B. Initiatives and Epics
C. Epics and Features
D. Stories and Tasks

Answer: D

Explanation:

The correct answer is **D. Stories and Tasks**. Here's why:

In an agile framework, autonomy is crucial for team velocity and innovation. Teams should be empowered to manage the granular elements of their work. Stories represent user-centric requirements that teams commit to deliver within a sprint. Tasks are the individual work items needed to complete a Story. Giving teams autonomy over Stories and Tasks empowers them to plan, estimate, and execute their work effectively within the sprint.

Features represent a higher-level grouping of Stories, often representing a deliverable functionality. Epics represent even larger bodies of work that span multiple sprints and sometimes even multiple teams. Initiatives sit at the highest level, representing strategic goals. Control over Epics and Initiatives are often maintained at a program or portfolio level to ensure alignment with overall business strategy. Assigning full autonomy of Epics and Features to individual teams can lead to disconnected development efforts and lack of alignment to overall product strategy. In agile with multiple teams, program or portfolio level planning is crucial to keep teams aligned, avoid redundant work, and ensure that delivered items can be tied back to desired business outcomes. By allowing teams to self-manage Stories and Tasks, the organization can ensure that each team is responsible for reaching sprint goals, but that work can still be aligned to overall business and product strategy goals.

Therefore, Stories and Tasks are the most appropriate level of taxonomy for teams to manage autonomously, enabling self-organization and efficient sprint execution, while maintaining alignment through higher-level governance of Epics, Features, and Initiatives.

Further research:

Agile at Scale (SAFe): https://www.scaledagileframework.com/
Microsoft Azure DevOps Documentation: https://learn.microsoft.com/en-us/azure/devops/

Question: 261 CertyIQ

Your company creates a new Azure DevOps team.
You plan to use Azure DevOps for sprint planning.
You need to visualize the flow of your work by using an agile methodology.
Which Azure DevOps component should you use?

A. Kanban boards
B. sprint planning
C. delivery plans

D. portfolio backlogs

Answer: A

Explanation:

The correct answer is A, Kanban boards. Here's why:

Kanban boards are a core component of Agile methodologies, especially for visualizing workflow, limiting work in progress (WIP), and optimizing flow. In Azure DevOps, Kanban boards allow teams to visually represent the state of each work item (user stories, tasks, bugs) as it moves through a defined workflow (e.g., New, Active, Resolved, Closed). This provides a clear overview of progress and bottlenecks.

Option B, sprint planning, is a crucial Agile event where teams plan which backlog items they will commit to delivering during a specific sprint (typically 2-4 weeks). While vital for Agile, it doesn't directly visualize the ongoing flow of work like a Kanban board.

Option C, delivery plans, are typically used for visualizing work across multiple teams and projects, aligning roadmaps and dependencies. They provide a higher-level view compared to the real-time, task-level visualization provided by Kanban boards.

Option D, portfolio backlogs, allow you to organize and prioritize your work at a higher level than sprint backlogs. For example, you could have Epics and Features in a portfolio backlog. They don’t directly visualize work flow like Kanban Boards.

Since the requirement is to visualize the flow of work using an Agile methodology for a new team starting with sprint planning, Kanban boards are the most suitable and direct way to achieve this in Azure DevOps. They facilitate transparency, collaboration, and continuous improvement by visualizing the progression of work items.

Further reading on Azure Boards and Kanban can be found at:

About Boards and Kanban
Use Azure Boards to plan and track your work

Question: 262 CertyIQ

Your company implements an Agile development methodology.
You plan to implement retrospectives at the end of each sprint.
Which three questions should you include? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Who performed well?
B. Who should have performed better?
C. What could have gone better?
D. What went well?
E. What should we try next?

Answer: CDE

Explanation:

The correct answers are C, D, and E because they align with the core principles of Agile retrospectives, which focus on continuous improvement and learning from past experiences. Let's break down why each choice is

appropriate and why the others are not:

C. What could have gone better? This question directly addresses areas where the team can improve their processes, communication, or execution in future sprints. It encourages constructive criticism and identification of bottlenecks or inefficiencies. This aligns with the iterative nature of Agile development.

D. What went well? Recognizing successes is crucial for reinforcing positive behaviors and practices within the team. This question fosters a positive environment and helps identify what the team should continue doing. Positive reinforcement is a key aspect of successful Agile teams.

E. What should we try next? This promotes experimentation and innovation. It encourages the team to propose new approaches or solutions to address identified issues or enhance their performance. This embodies the Agile principle of adapting to change.

A. Who performed well? This question focuses on individual performance, which is generally discouraged in Agile retrospectives. The emphasis should be on team performance and process improvements, not individual accolades. Focusing on individuals can create competition and discourage collaboration.

B. Who should have performed better? This question is similar to A and fosters a blame-oriented environment. It shifts the focus from process improvement to individual shortcomings, which is counterproductive to the principles of psychological safety and team collaboration that are essential for effective Agile development.

In essence, effective Agile retrospectives are centered on these key concepts: transparency, inspection, and adaptation. Focusing on "what" went wrong or well, rather than "who" is crucial for fostering a safe and collaborative environment that encourages continuous improvement. By exploring what could be improved and suggesting new approaches, the team can continually refine its processes and become more effective.

Authoritative Links for further research:

Atlassian - Agile Retrospectives: https://www.atlassian.com/agile/retrospectives
Scrum.org - The Scrum Values: https://www.scrum.org/resources/scrum-values
Agile Manifesto: http://agilemanifesto.org/

Question: 263

CertyIQ

Your team uses an agile development approach.
You need to recommend a branching strategy for the team's Git repository. The strategy must meet the following requirements.
- Provide the ability to work on multiple independent tasks in parallel.
- Ensure that checked-in code remains in a releasable state always.
- Ensure that new features can be abandoned at any time.
- Encourage experimentation.

What should you recommend?

A. a single long-running branch without forking
B. multiple long-running branches
C. a single fork per team member
D. a single long-running branch with multiple short-lived feature branches

Answer: D

Explanation:

The correct answer is D: a single long-running branch with multiple short-lived feature branches. This strategy

aligns with Gitflow or a similar branching model optimized for agile development.

Here's why:

Single long-running branch (e.g., main or develop): This ensures a consistently releasable state of the codebase. All merges into this branch are carefully reviewed and tested, guaranteeing stability. This directly addresses the requirement to keep checked-in code in a releasable state.
Multiple short-lived feature branches: Each feature or task is developed in its own branch, isolated from other development efforts. This enables parallel work on multiple independent tasks, fulfilling the corresponding requirement.
Ability to abandon features: Feature branches are easily discarded without impacting the main codebase. If an experiment fails or a feature is deemed unnecessary, the branch can simply be deleted, meeting the requirement for abandoning features.
Encouraging experimentation: Developers can freely experiment within their feature branches without the risk of destabilizing the primary codebase. This isolation promotes innovation and exploration.

Other options are less suitable:

A: A single long-running branch without forking would create integration conflicts and hinder parallel development.
B: Multiple long-running branches would introduce complexity and make it difficult to maintain a clear release path.
C: A single fork per team member would make code integration and collaboration very difficult. Forking is typically used for external contributors or for creating distinct versions of a project, not for day-to-day team collaboration.

This strategy leverages the strengths of Git for branching and merging, facilitating agile practices by supporting parallel development, isolated experimentation, and continuous integration. Feature branches are eventually merged into the main branch following code review and testing, maintaining code quality and stability.

Authoritative Links for Further Research:

Gitflow Workflow: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
Trunk-Based Development: https://trunkbaseddevelopment.com/
Branching strategies: https://learn.microsoft.com/en-us/azure/devops/pipelines/repos/git-branching?view=azure-devops

Question: 264 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You integrate a cloud-hosted Jenkins server and a new Azure DevOps deployment.
You need Azure DevOps to send a notification to Jenkins when a developer commits changes to a branch in Azure Repos.
Solution: You create a service hook subscription that uses the build completed event.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution aims to trigger a notification to Jenkins upon a commit to an Azure Repos branch. The solution suggests using a service hook subscription with the "build completed" event. This approach is incorrect because the "build completed" event is triggered after a build is finished, not when code is committed. Therefore, using the "build completed" event will not provide the immediate notification needed when a developer commits changes.

To achieve the goal of notifying Jenkins on code commits, the correct event to use within the Azure DevOps service hook is related to code changes. Specifically, events like "Code pushed" or "Pull request created/updated" would be much more appropriate. These events directly correspond to code commit activity. A "Code pushed" event will trigger a notification to Jenkins whenever new code is committed to the specified branch.

Therefore, the proposed solution does not meet the goal, and the correct answer is "No". The "build completed" event is relevant for post-build actions, not for triggering actions based on code commits.

Relevant Documentation:

Azure DevOps Service Hooks: https://learn.microsoft.com/en-us/azure/devops/service-hooks/overview?view=azure-devops
Azure Repos Events for Service Hooks: Look for events such as "Code pushed" or "Pull request created/updated" in the Service Hooks documentation.

Question: 265 CertyIQ

You have a project in Azure DevOps that has a release pipeline.
You need to integrate work item tracking and an Agile project management system to meet the following requirements:
- Ensure that developers can track whether their commits are deployed to production.
- Report the deployment status.
- Minimize integration effort.

Which system should you use?

A. Asana
B. Basecamp
C. Trello
D. Jira

Answer: D

Explanation:

The correct answer is D, Jira, because it integrates effectively with Azure DevOps to fulfill the given requirements for work item tracking and Agile project management. While other platforms (A, B, and C) offer project management capabilities, Jira has a more robust and native integration with Azure DevOps, specifically designed for development workflows.

Jira's integration allows developers to associate commits directly with work items (Azure DevOps work items are similar to Jira issues), enabling traceability. This satisfies the requirement of tracking whether commits are deployed to production. When a commit associated with a work item is deployed through the release pipeline, Jira can be automatically updated with the deployment status.

Furthermore, Jira provides powerful reporting features that can visualize deployment status, showing which work items have been deployed to which environments. This fulfills the reporting requirement. The Jira Azure DevOps integration minimizes integration effort compared to building custom solutions or using platforms with less native support. Other mentioned platforms may require more complex configurations or third-party integrations to achieve the same level of traceability and reporting. Jira provides a direct connector for Azure DevOps that simplifies the synchronization between the two systems.

Here are some resources for further research:

Microsoft Azure DevOps Services Integrations: https://learn.microsoft.com/en-us/azure/devops/extend/develop/integrate-with-other-services
Atlassian Jira Integration with Azure DevOps: https://www.atlassian.com/software/jira/integrations/azure-devops
Connect Azure DevOps Services to Jira Software Cloud: https://marketplace.visualstudio.com/items?itemName=ms-devlabs.vsts-extensions-atlassian.jiracloud

Question: 266 CertyIQ

You plan to onboard 10 new developers.
You need to recommend a development environment that meets the following requirements:
- Integrates with GitHub
- Provides integrated debugging tools
- Supports remote workers and hot-desking environments
- Supports developers who use browsers, tablets, and Chromebooks

What should you recommend?

A. VS Code
B. Xamarin Studio
C. MonoDevelop
D. Github Codespaces

Answer: D

Explanation:

GitHub Codespaces is the optimal solution for the given requirements. It offers a fully managed, cloud-based development environment accessible directly through a web browser. This seamlessly integrates with GitHub repositories, addressing the first requirement. Integrated debugging tools are a core feature of Codespaces, satisfying the second requirement. The browser-based accessibility of Codespaces inherently supports remote workers and hot-desking environments, as developers can access their development environment from any device with an internet connection, fulfilling the third requirement. Furthermore, Codespaces works perfectly with browsers, tablets, and Chromebooks due to its web-based nature, fully supporting developers using these devices. Alternatives like VS Code, Xamarin Studio, and MonoDevelop require installation and configuration on local machines, which doesn't directly meet the hot-desking and browser-based requirements as effectively as Codespaces. While VS Code has remote development capabilities, it still needs VS Code to be installed on a local device, which doesn't completely address the original issue of using different types of devices. Therefore, GitHub Codespaces stands out as the best option due to its full cloud-based nature and tight integration with GitHub.

Authoritative Links:

GitHub Codespaces Documentation: https://docs.github.com/en/codespaces

Question: 267 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You integrate a cloud-hosted Jenkins server and a new Azure DevOps deployment.
You need Azure DevOps to send a notification to Jenkins when a developer commits changes to a branch in Azure Repos.
Solution: You create an email subscription to an Azure DevOps notification.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The solution proposes using an email subscription to notify Jenkins when a developer commits changes to an Azure Repos branch. This approach is not the ideal way to achieve real-time and automated communication between Azure DevOps and Jenkins for triggering builds on code commits. Email notifications are inherently asynchronous, rely on email server infrastructure, and lack the direct triggering mechanism that is needed for continuous integration/continuous deployment (CI/CD). Jenkins doesn't natively listen to emails to start builds.

The preferred method involves setting up a webhook in Azure DevOps that sends a POST request to a Jenkins endpoint whenever a code commit occurs in the specified branch. This webhook will trigger a Jenkins job, enabling immediate and automated build processes. To properly configure the webhook, the Jenkins server must have the "Generic Webhook Trigger" or "Git plugin" installed and configured correctly to handle the incoming requests from Azure DevOps.

The email-based approach may work in limited scenarios where near real-time notification isn't critical and manual intervention is acceptable. However, for automated CI/CD pipelines, a webhook offers a more robust, efficient, and reliable solution, removing the manual process involved in initiating a build or other activities. Therefore, an email subscription does not meet the stated goal.

Reference:

Integrate Azure DevOps with Jenkins
Using webhooks with Jenkins

Question: 268 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You integrate a cloud-hosted Jenkins server and a new Azure DevOps deployment.
You need Azure DevOps to send a notification to Jenkins when a developer commits changes to a branch in Azure Repos.
Solution: You create a service hook subscription that uses the code pushed event.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

The solution meets the goal because service hooks in Azure DevOps are designed to trigger actions in external services, such as Jenkins, based on events occurring within Azure DevOps. A service hook subscription configured to listen for the "code pushed" event in Azure Repos will indeed send a notification to the configured Jenkins server whenever a developer commits and pushes changes to a branch. Jenkins can then be configured to react to this notification, for example, by triggering a new build. This creates an automated workflow where code commits in Azure Repos directly initiate processes in Jenkins. The service hook acts as the bridge between the two systems, enabling continuous integration. This is a standard practice for integrating Azure DevOps with other CI/CD tools. The configuration involves specifying the trigger event (code pushed), the target service (Jenkins), and any necessary authentication or data payload to be sent to Jenkins. Hence, the described solution effectively addresses the requirement.

Supporting Documentation:

Service hooks in Azure DevOps
Integrate with Jenkins

Question: 269 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You integrate a cloud-hosted Jenkins server and a new Azure DevOps deployment.
You need Azure DevOps to send a notification to Jenkins when a developer commits changes to a branch in Azure Repos.
Solution: You add a trigger to the build pipeline.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of adding a trigger to the build pipeline in Azure DevOps does not directly meet the goal of sending a notification to Jenkins when a developer commits changes to a branch in Azure Repos. Build triggers in Azure DevOps are designed to automatically initiate a build pipeline based on specific events, such as code commits. While a build pipeline could be configured to then send a notification to Jenkins, it is not the direct or most efficient approach. A build trigger on its own does not provide the necessary notification mechanism.

The most appropriate approach involves configuring a webhook in Azure Repos that sends a notification to Jenkins when a commit occurs. Jenkins can then be configured to listen for these webhooks and trigger a build or other action accordingly. This direct notification mechanism avoids the overhead of running an entire Azure DevOps build pipeline simply to relay information to Jenkins.

Therefore, while it's technically possible to use a build trigger as an indirect method, it's not the correct or most effective solution for the specific goal of sending a direct notification to Jenkins upon a commit in Azure

Repos. Webhooks are purpose-built for this type of event-driven integration.

Further research on Azure DevOps webhooks and Jenkins integration can be found at:

Azure DevOps Webhooks: https://learn.microsoft.com/en-us/azure/devops/service-hooks/services/webhooks?view=azure-devops
Jenkins Generic Webhook Trigger Plugin: https://plugins.jenkins.io/generic-webhook-trigger/

Question: 270 CertyIQ

You plan to create in Azure DevOps. Multiple developers will work on the project. The developers will work offline frequently and will require access to the full project history while they are offline.
Which version control solution should you use?

A. Team Foundation Version Control
B. Git
C. TortoiseSVN
D. Subversion

Answer: B

Explanation:

The correct answer is **B. Git**. Here's why:

Git is a Distributed Version Control System (DVCS). This means each developer has a complete copy of the repository, including the entire history, locally. This architecture is crucial for scenarios where developers frequently work offline. They can commit changes, create branches, and view history without a network connection. When back online, they can synchronize their local repository with the remote Azure DevOps repository (pull/push).

Team Foundation Version Control (TFVC), Subversion (SVN), and TortoiseSVN (a client for SVN) are centralized version control systems. In a centralized system, developers must connect to a central server to access the repository and its history. Working offline is severely limited or impossible.

Git's distributed nature directly addresses the requirement for offline access to the full project history. The local repository provides a self-contained working environment independent of a continuous network connection to Azure DevOps. This contrasts with centralized systems that rely on server connectivity for most operations, making them unsuitable for developers who frequently work offline. Therefore, Git is the best choice for this scenario.

Git DocumentationAzure Repos - Choose the right version control for your project

Question: 271 CertyIQ

You plan to onboard 10 new developers.
You need to recommend a development environment that meets the following requirements:
- Integrates with GitHub
- Provides integrated debugging tools
- Supports remote workers and hot-desking environments
- Supports developers who use browsers, tablets, and Chromebooks

What should you recommend?

A. VS Code

B. Xamarin Studio
C. MonoDevelop
D. Visual Studio Codespaces

Answer: D

Explanation:

The correct answer is D, Visual Studio Codespaces. Here's why:

Visual Studio Codespaces (now integrated into GitHub Codespaces) is a cloud-based development environment that directly addresses the stated requirements. Its integration with GitHub is a core feature, allowing developers to work directly with repositories. Codespaces provides fully integrated debugging tools, essential for efficient code development and problem-solving.

Critically, Codespaces supports remote workers and hot-desking scenarios because the development environment lives in the cloud. Developers can access their projects and tools from any location with an internet connection. This eliminates the need for developers to configure and maintain local development environments, which can be time-consuming and inconsistent.

The ability to access Codespaces through browsers, tablets, and Chromebooks is a key differentiator. It allows developers to use a wide range of devices, catering to hot-desking environments where developers might use different devices each day. VS Code (Option A) requires installation on a local machine, making it unsuitable for a fully cloud-based hot-desking scenario, although it can be used to connect to a Codespace. Xamarin Studio (Option B) and MonoDevelop (Option C) are more focused on mobile and cross-platform development and are not cloud-based development environments providing the broad accessibility required.

Therefore, Visual Studio Codespaces is the optimal solution because it integrates with GitHub, provides debugging tools, supports remote workers, and is accessible via a browser on various devices, meeting all the stated needs. It effectively centralizes and standardizes the development environment, simplifying onboarding and ensuring consistent tooling across the team.

Further research:

GitHub Codespaces Documentation: https://docs.github.com/en/codespaces
Azure DevOps: https://azure.microsoft.com/en-us/services/devops/

Question: 272 CertyIQ

You have a build pipeline in Azure Pipelines.
You create a Slack App Integration.
You need to send build notifications to a Slack channel named #Development.
What should you do first?

A. Create a project-level notification.
B. Configure a service connection.
C. Create a global notification.
D. Creates a service hook subscription.

Answer: D

Explanation:

The correct answer is D, create a service hook subscription. Here's why:

Service hooks in Azure DevOps allow you to trigger actions in other services when certain events occur within your Azure DevOps project. In this scenario, the goal is to trigger a notification in Slack (a different service) when a build completes. This event-driven behavior aligns directly with the functionality of service hooks.

Before you can send build notifications to a Slack channel, you must establish a mechanism for Azure DevOps to communicate with Slack. This is accomplished by creating a service hook subscription that specifies the event to monitor (build completion) and the action to take (send a message to the #Development channel in Slack).

A project-level notification (Option A) is for notifying users within the Azure DevOps project based on events. It doesn't directly integrate with external services like Slack. Global notifications (Option C) are generally for organizational-wide events, not necessarily tied to specific project build events and still don't create the necessary link to Slack.

A service connection (Option B) is generally used to authenticate and authorize Azure Pipelines to access resources within Azure, such as Azure Resource Manager (ARM) or Azure Container Registry (ACR). While service connections might be involved in a larger deployment process related to the build, they are not the initial step for setting up notifications in Slack based on build events. The service hook handles the notification trigger.

Therefore, creating a service hook subscription is the first and most appropriate step to integrate build notifications from Azure Pipelines to a Slack channel. This subscription will specify the build completion event and the desired action of sending a message to the specific Slack channel (#Development).

References:

Service hooks with Azure DevOps - Azure DevOps Services | Microsoft Learn
Integrate with Slack - Azure DevOps Services | Microsoft Learn

Question: 273 CertyIQ

You have an Azure DevOps organization named Contoso and an Azure subscription.
You use Azure DevOps to build and deploy a web app named App1. Azure Monitor is configured to generate an email notification in response to alerts generated whenever App1 generates a server-side error.
You need to receive notifications in Microsoft Teams whenever an Azure Monitor alert is generated.
Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Create an Azure Monitor workbook.
B. Create an Azure logic app that has an HTTP request trigger.
C. Create an Azure logic app that has an Azure DevOps trigger.
D. Modify an action group in Azure Monitor.
E. Modify the Diagnostics settings in Azure Monitor.

Answer: BD

Explanation:

The correct answer is **B and D**. Here's a detailed justification:

B. Create an Azure Logic App with an HTTP request trigger: Azure Logic Apps provide a way to automate workflows and integrate different services. In this scenario, we need a Logic App to act as a bridge between Azure Monitor and Microsoft Teams. The HTTP request trigger allows the Logic App to be invoked by an

external service via an HTTP POST request. Azure Monitor action groups can then be configured to send a POST request to this Logic App when an alert is triggered.

D. Modify an action group in Azure Monitor: Action groups in Azure Monitor define the actions to be taken when an alert is triggered. By modifying the action group associated with the alerts for App1, we can add an action to trigger the Logic App. Specifically, we would add a Webhook action in the Action group that sends an HTTP POST request to the Logic App endpoint. This will trigger the Logic App whenever the alert is fired.

By creating a Logic App with an HTTP trigger and configuring an action group to call this Logic App, a webhook is created to notify Teams when server-side errors occur within the web app.

Why other options are incorrect:

A. Create an Azure Monitor workbook: Workbooks are primarily used for visualization and analysis of data in Azure Monitor. They don't provide a mechanism for automated notifications in Microsoft Teams.
C. Create an Azure logic app that has an Azure DevOps trigger: Azure DevOps triggers are relevant when the Logic App's workflow should be initiated by events within Azure DevOps (e.g., code commit, build completion). This is not related to Azure Monitor alerts.
E. Modify the Diagnostics settings in Azure Monitor: Diagnostic settings are used to configure the sources of telemetry data (e.g., logs, metrics) that are sent to Azure Monitor. Modifying diagnostic settings doesn't directly enable notifications to Microsoft Teams.

Authoritative Links for further research:

Azure Logic Apps Documentation
Azure Monitor Action Groups
Webhooks in Azure Monitor

Question: 274 CertyIQ

HOTSPOT -
Your company uses Azure DevOps for Git source control.
You have a project in Azure DevOps named Contoso App that contains the following repositories:
- https://dev.azure.com/contoso/contoso-app/core-api
- https://dev.azure.com/contoso/contoso-app/core-spa
- https://dev.azure.com/contoso/contoso-app/core-db

You need to ensure that developers receive Slack notifications when there are pull requests created for Contoso App.
What should you run in Slack? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

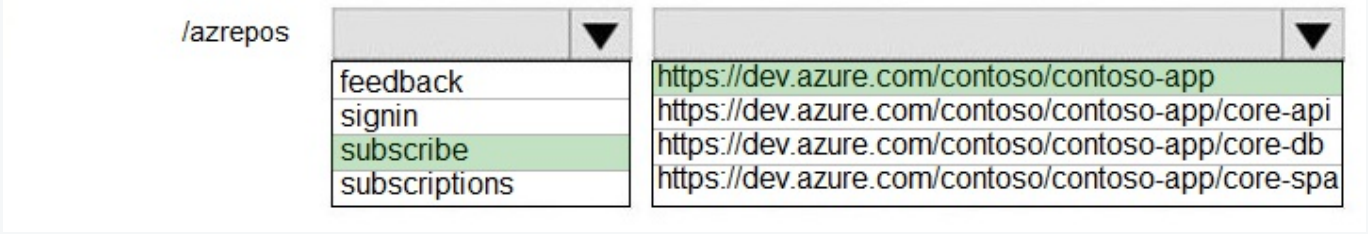
Answer Area

/azrepos [▼] [▼]

feedback	https://dev.azure.com/contoso/contoso-app
signin	https://dev.azure.com/contoso/contoso-app/core-api
subscribe	https://dev.azure.com/contoso/contoso-app/core-db
subscriptions	https://dev.azure.com/contoso/contoso-app/core-spa

Answer:

Answer Area



Explanation:
Box 1: subscribe -
To start monitoring all Git repositories in a project, use the following slash command inside a channel:
/azrepos subscribe [project url]
Box 2: https://dev.azure.com/contoso/contoso-app
You can also monitor a specific repository using the following command:
/azrepos subscribe [repository url]
The repository URL can be to any page within your repository that has your repository name.
For example, for Git repositories, use:
/azrepos subscribe https://dev.azure.com/myorg/myproject/_git/myrepository

Reference:
https://docs.microsoft.com/en-us/azure/devops/repos/integrations/repos-slack

Question: 275 CertyIQ

You have an Azure DevOps organization that contains a project named Project1.
You need to create a published wiki in Project1.
What should you do first?

A. Modify the Storage settings of Project1.
B. In Project1, create an Azure DevOps pipeline.
C. In Project1, create an Azure DevOps repository.
D. Modify the Team configuration settings of Project1.

Answer: C

Explanation:

The correct answer is **C. In Project1, create an Azure DevOps repository.**

A published wiki in Azure DevOps needs a Git repository to store its pages, images, and other supporting files. The published wiki isn't automatically tied to the project's main code repository. Instead, it necessitates a dedicated repository for its content.

Here's why the other options are incorrect:

A. Modify the Storage settings of Project1: While storage is relevant for Azure DevOps, changing storage settings is not a prerequisite for creating a published wiki. Storage settings manage data retention and usage, not the foundation for a wiki.
B. In Project1, create an Azure DevOps pipeline: Azure DevOps pipelines are for CI/CD, automating build, test, and deployment processes. They aren't directly linked to the wiki's content storage. A pipeline could, however, be used to automate tasks associated with updating the wiki repository.

D. Modify the Team configuration settings of Project1: Team configuration settings relate to team-specific areas, iterations, and permissions. These settings don't directly influence the creation of a published wiki's underlying storage.

To create a published wiki, you must first set up a Git repository. After creating the repository, you can publish the wiki, which will use this repository to store and manage the wiki's content. You can then populate the repository with Markdown files representing the wiki pages. The wiki functionality in Azure DevOps then renders these files into a navigable and user-friendly wiki.

Further reading on published wikis:

Publish a Git repository as a wiki - Azure DevOps Services | Microsoft Learn
Add & edit wiki pages - Azure DevOps Services | Microsoft Learn

Question: 276 CertyIQ

Your company plans to use an agile approach to software development.
You need to recommend an application to provide communication between members of the development team who work in locations around the world. The applications must meet the following requirements:
- Provide the ability to isolate the members of different project teams into separate communication channels and to keep a history of the chats within those channels.
- Be available on Windows 10, Mac OS, iOS, and Android operating systems.
- Provide the ability to add external contractors and suppliers to projects.
- Integrate directly with Azure DevOps.

What should you recommend?

A. Microsoft Project
B. Bamboo
C. Microsoft Lync
D. Microsoft Teams

Answer: D

Explanation:

Microsoft Teams is the most suitable application because it directly addresses all the stated requirements for agile team communication. It facilitates separate communication channels for different project teams, retaining a history of chats within each channel, thus meeting the need for project isolation and historical context. Teams has native applications available across Windows 10, Mac OS, iOS, and Android, ensuring accessibility for all team members regardless of their device. The platform allows for the easy addition of external contractors and suppliers as guests, supporting collaboration with external entities. Crucially, Microsoft Teams integrates seamlessly with Azure DevOps, creating a unified environment for communication and development workflow. This integration allows teams to receive notifications from Azure DevOps, share work items, and collaborate on code directly within Teams channels, streamlining the development process. Microsoft Project is a project management tool, not primarily a communication platform. Bamboo is a continuous integration and continuous delivery (CI/CD) tool, and while it offers some communication features, it's not designed for comprehensive team communication. Microsoft Lync (now Skype for Business) lacks the same level of integration with Azure DevOps and the dedicated team-focused features that Teams offers. Therefore, Teams is the optimal solution due to its focus on team communication, cross-platform availability, external user support, and direct integration with Azure DevOps.

Authoritative Links:

Microsoft Teams: https://www.microsoft.com/en-us/microsoft-teams/
Azure DevOps Integration with Microsoft Teams: https://learn.microsoft.com/en-

us/azure/devops/integrations/integrations-overview?view=azure-devops

Question: 277 CertyIQ

You are developing a multi-tier application. The application will use Azure App Service web apps as the front end and an Azure SQL database as the back end.
The application will use Azure functions to write some data to Azure Storage.
You need to send the Azure DevOps team an email message when the front end fails to return a status code of 200.
Which feature should you use?

A. Service Map in Azure Log Analytics
B. availability tests in Azure Application Insights
C. Profiler in Azure Application Insights
D. Application Map in Azure Application Insights

Answer: B

Explanation:

The correct answer is **B. Availability tests in Azure Application Insights**.

Here's why:

Availability tests in Azure Application Insights are specifically designed to monitor the uptime and responsiveness of your application's endpoints. These tests periodically send requests to your application's URLs and verify the response. You can configure the tests to send alerts when specific conditions are not met, such as a status code other than 200.

In this scenario, the requirement is to receive an email when the front end (Azure App Service web app) fails to return a 200 status code. Availability tests can be configured to check the status code returned by the web app. If the status code is not 200, an alert can be triggered, sending an email notification to the Azure DevOps team. This provides proactive monitoring and immediate notification of front-end failures.

Here's why the other options are less suitable:

A. Service Map in Azure Log Analytics: Service Map discovers and visualizes dependencies between application components but isn't designed to specifically monitor availability and send alerts based on HTTP status codes.
C. Profiler in Azure Application Insights: Profiler collects performance traces to help identify performance bottlenecks. It's not used for availability monitoring or alerting based on status codes.
D. Application Map in Azure Application Insights: Application Map provides a visual representation of the application architecture and dependencies. While it shows application health, it doesn't offer the granular alerting capabilities based on HTTP status codes that Availability tests provide.

Therefore, Availability tests are the most appropriate feature for monitoring the web app's availability and sending email alerts when the status code is not 200.

Here are some authoritative links for further research:

Availability tests: https://learn.microsoft.com/en-us/azure/azure-monitor/app/monitor-web-app-availability
Application Insights: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview

Question: 278 CertyIQ

You have a project in Azure DevOps named Project1. Project1 contains a published wiki.
You need to change the order of pages in the navigation pane of the published wiki in the Azure DevOps portal.
What should you do?

A. At the root of the wiki, create a file named .order that defines the page hierarchy.
B. At the root of the wiki, create a file named wiki.md that defines the page hierarchy.
C. Rename the pages in the navigation pane.
D. Drag and drop the pages in the navigation pane.

Answer: A

Explanation:

The correct answer is A: At the root of the wiki, create a file named .order that defines the page hierarchy.

The .order file is a specific mechanism within Azure DevOps wikis designed to control the order and hierarchy of pages displayed in the navigation pane. This file allows you to customize the structure of your wiki beyond the default alphabetical or creation order. By creating a .order file at the root directory of your wiki and defining the desired order of pages within it, you explicitly control the navigation structure.

The .order file is a simple text file containing a list of page names, each on a new line, reflecting the order you want them to appear in the navigation. You can also create subfolders within the wiki structure, and the .order file within those subfolders defines the order of pages within that specific section.

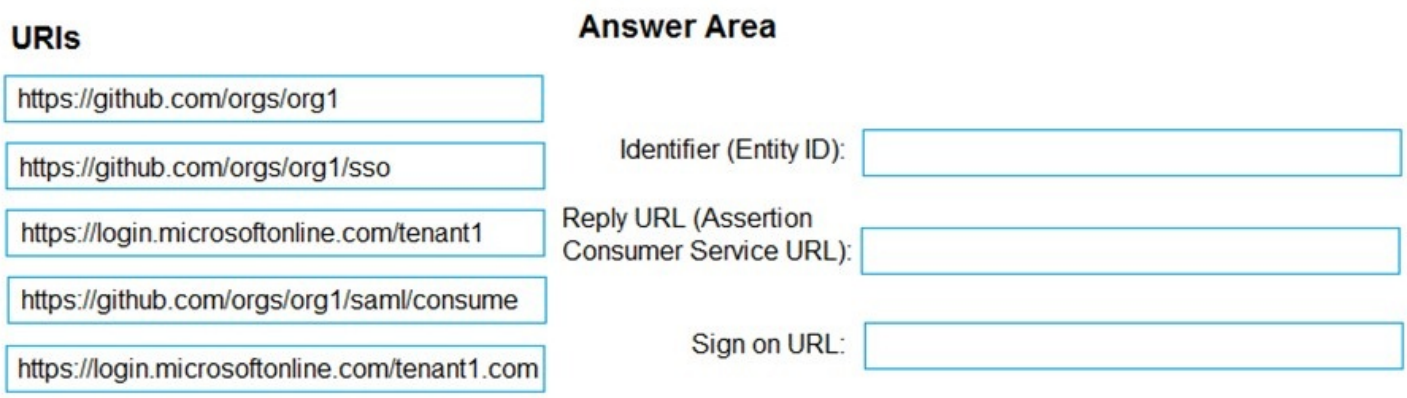
Option B is incorrect because wiki.md is generally the home page of a wiki and doesn't define the overall hierarchy. Option C is not the intended solution as renaming pages is not about structural organization, and it could break existing links. Option D, while intuitive, is not a supported feature in the current Azure DevOps Wiki implementation. Drag and drop is not a user interface interaction for directly modifying the wiki's navigational structure. The .order file is the specific and intended way to organize the wiki.

For further reading and detailed information, consult the official Microsoft Azure DevOps documentation on Wiki functionality:

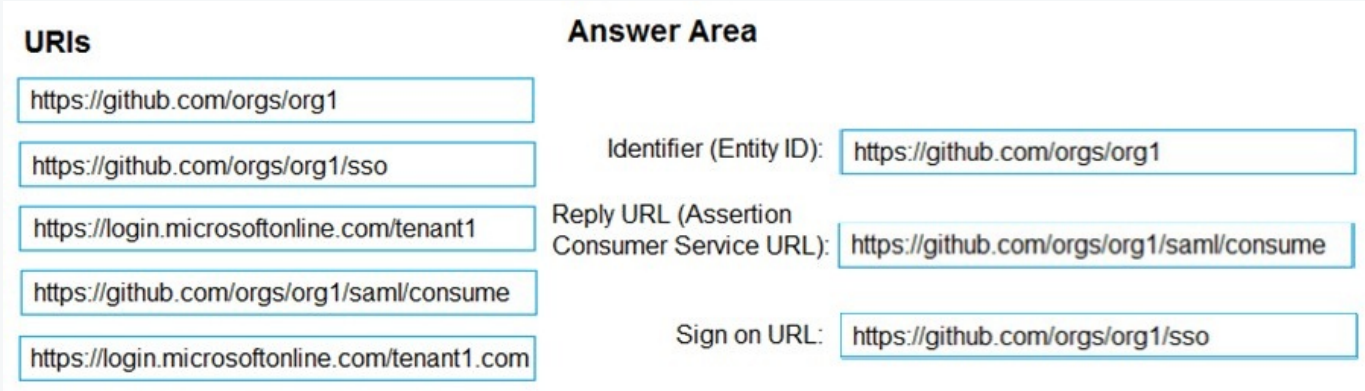
Organize wiki pages: https://learn.microsoft.com/en-us/azure/devops/project/wiki/wiki-navigation?view=azure-devops

Question: 279 CertyIQ

DRAG DROP -
You have a GitHub organization named org1 and an Azure tenant named Tenant1.
You need to enable single sign-on (SSO) in Azure Active Directory (Azure AD) for the users in org1.
Which URIs should you use for the SAML configuration in Azure AD? To answer, drag the appropriate URIs to the correct settings. Each URI may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Reference:
https://docs.microsoft.com/en-us/azure/active-directory/saas-apps/github-tutorial

Question: 280 CertyIQ

Your company plans to use an agile approach to software development.
You need to recommend an application to provide communication between members of the development team who work in locations around the world. The applications must meet the following requirements:
- Provide the ability to isolate the members of different project teams into separate communication channels and to keep a history of the chats within those channels.
- Be available on Windows 10, Mac OS, iOS, and Android operating systems.
- Provide the ability to add external contractors and suppliers to projects.
- Integrate directly with Azure DevOps.

What should you recommend?

A. Skype for Business
B. Bamboo
C. Octopus
D. Slack

Answer: D

Explanation:

Slack is the best recommendation because it directly addresses all the stated requirements. It allows teams to create distinct channels for different projects, effectively isolating communication and maintaining chat history. Slack's availability across Windows 10, Mac OS, iOS, and Android ensures seamless accessibility for all team members, regardless of their preferred operating system. Furthermore, Slack supports adding

external contractors and suppliers as guests to specific channels, facilitating collaboration with external parties. Crucially, Slack offers direct integration with Azure DevOps, enabling developers to receive notifications, manage tasks, and stay informed about project progress directly within their communication platform. This Azure DevOps integration streamlines workflows and enhances team collaboration. Skype for Business, while offering cross-platform support and external user addition, lacks robust integration with Azure DevOps and might not offer the same level of channel-based organization as Slack. Bamboo is a CI/CD server and not designed for general team communication. Octopus Deploy is a deployment automation tool, similarly unsuitable for the described communication needs. Therefore, Slack's features and capabilities make it the most suitable choice for the company's agile software development environment requiring organized communication and Azure DevOps integration.

Supporting links:

Slack Integration with Azure DevOps: https://slack.com/intl/en-gb/solutions/devops
Slack for Team Communication: https://slack.com/intl/en-gb/features

Question: 282 CertyIQ

You have a pipeline named Pipeline1 in Azure Pipelines.
You need to create a service connection to enable Pipeline1 to download a public container image.
Which type of service connection should you create?

A. a Docker host
B. a Docker registry
C. Azure Service Fabric
D. Azure Kubernetes Service (AKS)

Answer: B

Explanation:

The correct answer is **B. a Docker registry**. Here's why:

The core requirement is to enable a pipeline to download a public container image. Container images are stored in container registries. A service connection acts as an authenticated gateway between Azure Pipelines and external services. To fetch a public container image, the pipeline needs to connect to the registry where the image is located.

Docker Registry Service Connection: This type of connection is explicitly designed for connecting to container registries such as Docker Hub, Azure Container Registry (ACR), or other compliant registries. It provides the necessary authentication and configuration to pull images. Since the image is public, authentication might not always be required, but the connection still sets up the necessary endpoint and registry details.

Docker Host: This connection is used for managing and interacting with a Docker host (a machine running the Docker daemon), not for accessing image registries. It's typically used when you want to build or deploy images directly to a specific Docker host, which is not the primary goal here.

Azure Service Fabric: This is a distributed systems platform used for packaging, deploying, and managing scalable and reliable microservices and containers. It's not directly involved in simply downloading a container image from a registry.

Azure Kubernetes Service (AKS): AKS is a managed Kubernetes service for deploying and managing containerized applications. While you might eventually deploy the downloaded image to AKS, the immediate

need is just to pull the image from a registry.

Therefore, a Docker registry service connection is the most direct and appropriate solution for downloading a public container image in Azure Pipelines. It provides the necessary configuration and authentication (if required) to access the specified container registry.

Further Reading:

Service Connections in Azure Pipelines
Azure Container Registry
Docker Hub

Question: 283 CertyIQ

You have a project in Azure DevOps named Project1 that contains a Kanban board named Board1.
You create a Microsoft Teams channel and add the Azure Boards app to the channel.
You need to ensure that users can create work items in Board1 from Microsoft Teams.
Which command should you run?

A. @azure boards subscriptions
B. @azure boards create
C. @azure boards sign in
D. @azure boards link

Answer: D

Explanation:

The correct answer is **D. @azure boards link**. Here's a detailed justification:

The Azure Boards app in Microsoft Teams enables collaboration and work item management directly within Teams channels. Before users can create work items or interact with a specific Azure Boards board, the Teams channel must be linked to the Azure DevOps project and the specific board.

Option **A. @azure boards subscriptions** manages notifications and subscriptions for work item changes and updates within the channel. While helpful, it doesn't enable the creation of new work items from Teams to the specific board.

Option **B. @azure boards create** is not a valid command for creating work items. The link command must be used first to establish the link to the Azure DevOps board.

Option **C. @azure boards sign in** is used to authenticate the user with their Azure DevOps account if they haven't already done so. Authentication is a prerequisite, but it doesn't directly link the channel to a specific Azure Boards board for work item creation.

The **@azure boards link** command is the specific command used to associate the Teams channel with an Azure DevOps project and board. This command establishes the necessary connection that allows users in Teams to interact with the linked Azure Boards project, including creating, viewing, and modifying work items. Without this link, the Azure Boards app won't know which project and board to interact with when users attempt to create work items. The format of the command often involves specifying the Azure DevOps organization and project. For example: @azure boards link [Azure DevOps organization] [Project Name]. After linking, users can then create work items.

After linking, users can utilize other commands (such as those prompted by the app after a link has been established) to manage work items. The essential step to allow users to create work items from Teams is to

link the channel to the relevant Azure DevOps project and board.

In conclusion, the @azure boards link command is the foundational step that enables the creation of work items from Microsoft Teams to the specified Azure DevOps project and Kanban board.

Supporting Documentation:

Install and configure Azure Boards app for Microsoft Teams
Manage work items from Microsoft Teams

Question: 284 CertyIQ

You manage a project by using Azure Board, and you manage the project code by using Azure Repos.
You have a bug work item that has an ID of 123.
You need to set the work item state to Resolved.
What should you add to the commit message?

A. #123 completes
B. #123 Resolved
C. Completed #123
D. Fixes #123

Answer: D

Explanation:

The correct answer is D, Fixes #123.

Azure DevOps provides direct integration between commits in Azure Repos and work items in Azure Boards. This integration allows you to automatically update the state of work items based on keywords within the commit message. Several keywords trigger different actions on linked work items.

The keyword Fixes (or Fixed, Fix, Fixing) automatically transitions a work item to the Resolved state when the commit containing this keyword is merged into a branch that is being tracked by the work item. Other keywords, such as Closes, Completes, or Resolves, might transition the work item to a Closed state, which might not be what you want if further verification is needed before a bug is officially considered "closed". Using Fixes allows the work item to be marked as resolved, indicating that the code fix has been implemented, but still allows for testing and validation before final closure.

Option A, #123 completes, is functionally similar but often used to indicate the completion of a task, not specifically a bug fix. Options B, #123 Resolved, and C, Completed #123, will not trigger the automatic state transition in Azure DevOps. The correct keyword is crucial for the integration to work as expected. The integration functionality enhances traceability, automation, and collaboration between developers and project managers within Azure DevOps. Using the right keywords is crucial for correctly linking the code change and the work item, leading to a seamless workflow and improved project management. Azure DevOps monitors the commits for these keywords and will update the state of the work item when a tracked branch includes the change.

Further research:

Link work items to commits and pull requests: https://learn.microsoft.com/en-us/azure/devops/boards/boards/connect-work-items-to-git-devops?view=azure-devops&tabs=browser

Question: 285 CertyIQ

You have an Azure subscription that contains a storage account and 20 virtual machines.
You plan to use LogRhythm for aggregation and analysis of the virtual machine logs.
You need to configure AzLog to export the logs and push them to the storage account.
In which format should you export the logs?

A. JSON
B. EVTX
C. EVT
D. binary

Answer: A

Explanation:

The correct answer is **A. JSON**. Here's a detailed justification:

When configuring AzLog to export logs from Azure virtual machines and push them to a storage account for LogRhythm's aggregation and analysis, the choice of log format is crucial for compatibility and efficient processing. LogRhythm, like many modern SIEM (Security Information and Event Management) and log management systems, excels at processing structured data.

JSON (JavaScript Object Notation) is a widely used, human-readable, and machine-parsable data format that is well-suited for representing structured log data. It's lightweight and easily integrated with various systems and tools, including LogRhythm. When logs are exported in JSON format, LogRhythm can readily ingest and parse the data, enabling efficient analysis, correlation, and reporting.

EVTX (Event Viewer XML Template) and EVT (Event Viewer Template) are binary log formats primarily used by the Windows Event Log service. While these formats contain rich event information, they are not as universally compatible as JSON. Directly exporting logs in these formats to Azure Storage and then attempting to process them with LogRhythm would require specialized parsing tools and potentially significant processing overhead to extract the meaningful information. LogRhythm can ingest EVTX files, but it's generally more efficient to use a structured format.

Binary format is generally not preferred for log storage due to its lack of readability and the difficulty in parsing without specific tools designed for the particular binary structure. It offers no advantages for this scenario.

Therefore, exporting the logs in JSON format is the optimal approach. It ensures that the logs are stored in a structured, easily parsable format that LogRhythm can readily consume and analyze, leading to a more efficient and effective log management solution. JSON's widespread support also provides greater flexibility and interoperability compared to other formats.

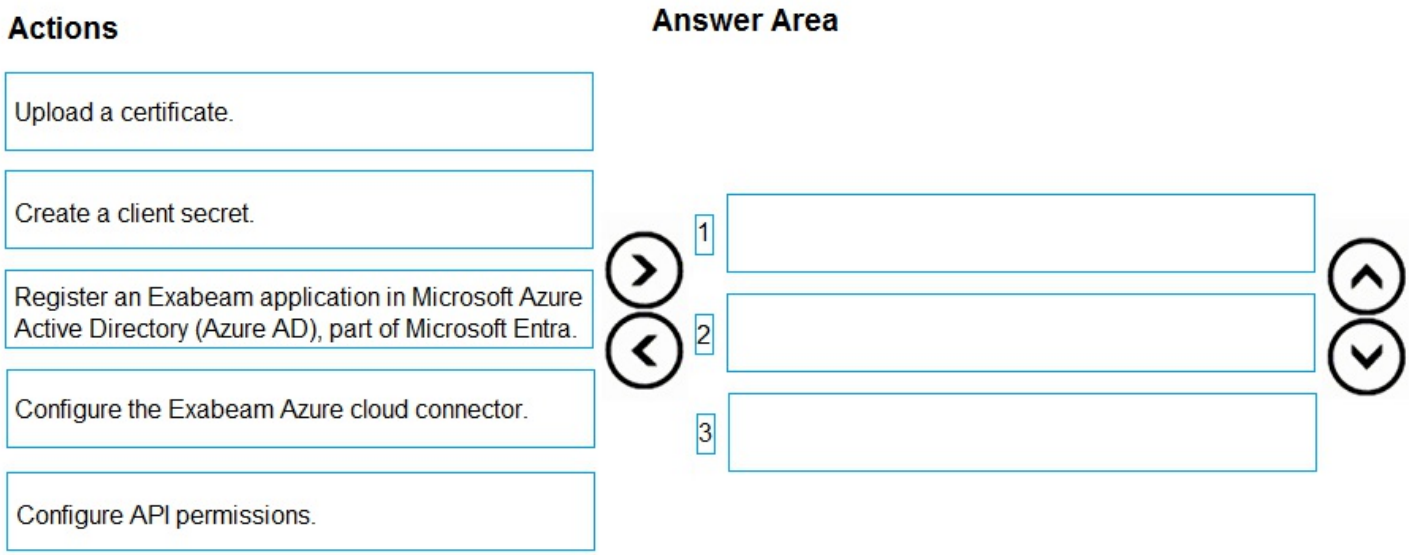
Supporting Links:

JSON: https://www.json.org/json-en.html - Official JSON website explaining the format.
LogRhythm Documentation: Refer to LogRhythm's official documentation on supported log formats and best practices for integrating with Azure. This information would contain specific configuration details for ingestion. Accessing LogRhythm's specific documentation requires a user account and is typically gated content. Search engines such as Google, Bing or DuckDuckGo are recommended to find relevant articles.

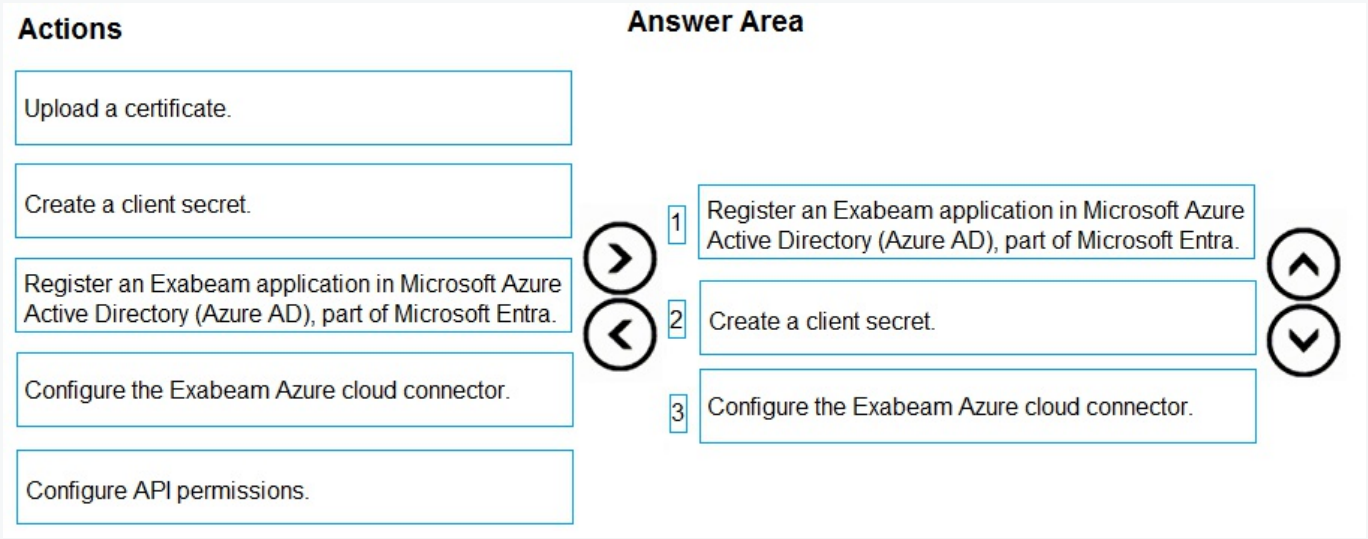
Question: 286 CertyIQ

DRAG DROP

-
You use Exabeam Fusion SIEM and the Azure cloud platform.
You need to integrate Exabeam and Azure. The solution must use OAuth authentication.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



Answer:



Explanation:

Register an Exabeam application in Azure AD part of Entra - add the app to your Azure AD

Create a client secret - so Exabeam can use this to auth

Configure the exabeam azure cloud connector - give it the app ID & secret so exabeam can auth

Question: 287 CertyIQ

You use GitHub for source control and Azure Boards for project management. GitHub and Azure Boards are integrated.
You plan to create a pull request in GitHub.
You need to automatically link the request to an existing Azure Boards work item by using the text of AB#.
To which two elements can you add the text? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A. milestone

B. label
C. title
D. comment
E. description

Answer: CE

Explanation:

The correct answers are C (title) and E (description). Let's break down why:

The integration between GitHub and Azure Boards allows for linking work items (like tasks, bugs, or features) to GitHub commits and pull requests using a special syntax, typically AB#<work item ID>. This linking is essential for traceability, allowing developers to easily navigate from code changes in GitHub to the corresponding work item in Azure Boards and vice versa. The goal is to automate the process of linking the pull request to the correct Azure Boards work item.

Option C, adding the AB#<work item ID> syntax to the title of the pull request, is a valid solution. The integration actively scans the pull request title for this pattern and automatically creates the link. This provides a clear and upfront reference to the associated work item as soon as the pull request is created and visible.

Option E, adding the AB#<work item ID> syntax to the description of the pull request, is also a valid solution. Azure Boards actively parses the pull request description for linking patterns. By including the reference in the description, you effectively create the link between the pull request and the work item.

Option A, milestone, is incorrect. Milestones in GitHub are typically used for tracking progress on a project or release and are not scanned for Azure Boards linking patterns.

Option B, label, is incorrect. Labels in GitHub are used for categorization and organization of issues and pull requests. They are not scanned for the Azure Boards linking patterns either.

Option D, comment, while it would create a link after the pull request is created, the prompt requires that the link be created automatically with the text AB#. Adding it to the description or title ensures that the link is created when the pull request is created rather than manually after the creation by adding the comment.

Therefore, placing the AB#<work item ID> syntax in either the title or description of the pull request will ensure that the pull request is automatically linked to the specified Azure Boards work item.

For further information on integrating GitHub with Azure Boards, refer to the official Microsoft documentation:

Connect Azure Boards to GitHub: https://learn.microsoft.com/en-us/azure/devops/boards/github/?view=azure-devops
Link from GitHub: https://learn.microsoft.com/en-us/azure/devops/boards/github/link-from-github?view=azure-devops&tabs=new-nav

Question: 288 CertyIQ

HOTSPOT
-
You are using Agile process methodologies in Azure DevOps.
You need to deploy a dashboard that will provide progress reports for the following work items:
•How long it took to close a work item after it was created.
•How long it took to close a work item after the work was started.

Which type of widget should you use for each work item? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

How long it took close a work item after it was created:

- Burndown chart
- Cumulative flow diagram
- Cycle time
- Lead time
- Velocity

How long it took to close a work item after the work was started:

- Burndown chart
- Burnup chart
- Cycle time
- Lead time
- Velocity

Answer:

Answer Area

How long it took close a work item after it was created: Lead time

How long it took to close a work item after the work was started: Cycle time

Explanation:

1. Lead time2. Cycle time

https://learn.microsoft.com/en-us/azure/devops/report/dashboards/cycle-time-and-lead-time?view=azure-devopsBoth lead time and cycle time widgets are useful to teams. They both indicate how long it takes for work to flow through their development pipeline. Lead time measures the total time elapsed from the creation of work items to their completion. Cycle time measures the time it takes for your team to complete work items once they begin actively working on them.

Question: 289

CertyIQ

You are integrating an Azure Boards project and a GitHub repository.
You need to authenticate Azure Boards to GitHub.
Which two authentication methods can you use? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A.a publisher certificate

B.a trusted root certificate

C.Microsoft Azure Active Directory (Azure AD), part of Microsoft Entra

D.GitHub user credentials

E.a personal access token (PAT)

Answer: DE

Explanation:

The correct answer is DE (GitHub user credentials and a personal access token (PAT)).

Here's why:

When integrating Azure Boards with a GitHub repository, authentication is crucial for securely accessing and managing the GitHub repository from within Azure Boards. This integration enables features like linking work items to commits, pull requests, and issues within GitHub.

D. GitHub user credentials: Using GitHub user credentials provides a straightforward way to authenticate. You can directly input your GitHub username and password into Azure DevOps during the integration setup. This is suitable for individual users or small teams where managing granular permissions isn't a primary concern. However, this approach ties the integration to a specific user's account. If the user leaves the organization or changes their password, the integration breaks.

E. A personal access token (PAT): A PAT is a more secure and recommended method. It allows you to create a token with specific permissions within your GitHub account. This token can then be used by Azure Boards to access the GitHub repository. The benefit is that you can control precisely what Azure Boards can do within the repository. If you need to revoke access, you can simply delete the PAT without affecting your primary GitHub account. PATs also provide auditability, allowing you to track their usage.

Options A, B, and C are incorrect:

A. A publisher certificate and B. A trusted root certificate: These certificates are used for code signing and establishing trust between software publishers and consumers. They are not directly relevant to authenticating Azure Boards to GitHub.
C. Microsoft Azure Active Directory (Azure AD), part of Microsoft Entra: While Azure AD can be used for authentication in various scenarios within the Microsoft ecosystem, direct Azure AD authentication for GitHub integration within Azure DevOps/Boards is not a natively supported option. Azure AD is primarily for managing identities and access within Azure services and organizations. While GitHub does support SAML SSO through Enterprise plans using identity providers (including Azure AD), this isn't the standard mechanism for Azure Boards to GitHub integration, which relies on the GitHub credentials or a PAT.

In conclusion, using GitHub user credentials (less secure, tied to a specific user) and personal access tokens (more secure, granular permissions) are the two valid authentication methods for integrating Azure Boards with a GitHub repository.

Relevant Links:

Connect Azure Boards to GitHub
Using personal access tokens

Question: 290 CertyIQ

You have a GitHub repository that is integrated with Azure Boards. Azure Boards has a work item that has the number 715.
You need to ensure that when you commit source code in GitHub, the work item is updated automatically.
What should you include in the commit comments?

A.the URL of the work item
B.AB#715
C.@715
D.#715

Answer: B

Explanation:

The correct answer is **B. AB#715**.

Here's why:

Azure Boards provides a mechanism to automatically transition work items based on commit messages in connected source control systems like GitHub. This is crucial for streamlining development workflows and providing traceability between code changes and project management. This functionality is often used in agile methodologies to signify progress on tasks.

The syntax AB#715 specifically tells Azure Boards to link the commit to work item #715. When Azure Boards detects this syntax in a commit message, it automatically creates a link between the commit and the specified work item. This enables direct navigation from the work item to the related code change in GitHub, and vice-versa. Using the AB# prefix informs Azure DevOps that the number following it refers to an Azure Boards work item ID.

Option A (the URL of the work item) would only create a link; it wouldn't automatically transition the work item's state based on configuration. Option C (@715) is typically used for mentioning users and wouldn't interact with the Azure Boards integration. Option D (#715) is a common syntax in social media or other systems, but Azure Boards specifically requires the "AB" prefix to identify the reference as a valid Azure Boards work item. The AB# syntax directly instructs Azure DevOps to interact with the work item specified.

Including AB#715 in your commit comment ensures that the link between your source code commit and the work item in Azure Boards is automatically established, providing seamless traceability and enabling automated workflow management, such as automatically resolving the work item when the code is committed and pushed.

For more detailed information, refer to the official Microsoft documentation:

Drive Git development from work items: https://learn.microsoft.com/en-us/azure/devops/boards/boards/drive-development-from-work-item?view=azure-devops&tabs=new-nav

Question: 291 CertyIQ

You are creating a dashboard in Azure Boards.
You need to visualize the time from when work starts on a work item until the work item is closed.
Which type of widget should you use?

A.cycle time
B.velocity
C.cumulative flow
D.lead time

Answer: A

Explanation:

The correct answer is **A. Cycle Time**. Here's why:

Cycle Time specifically measures the elapsed time between when work begins on a task and when it's completed. In the context of Azure Boards, this translates directly to the time between when a work item's status changes to "In Progress" (or a similar active state) and when it reaches a "Closed" or "Done" state. The Cycle Time widget visually represents this duration, allowing teams to identify bottlenecks and improve their workflow efficiency. It directly addresses the requirement of visualizing the time from work start to work item closure.

Why other options are incorrect:

Velocity: Velocity focuses on the amount of work a team completes within a sprint. It's valuable for sprint planning and forecasting but doesn't provide a granular view of the time taken for individual work items to progress through the workflow.

Cumulative Flow: Cumulative Flow Diagrams (CFDs) visualize the number of work items in each state over time. While CFDs can indirectly indicate cycle time trends, they don't offer a direct measurement or visualization of the time elapsed for each individual work item to complete. They are more useful for understanding overall flow bottlenecks and stability.

Lead Time: Lead time represents the total time from when a work item is requested (e.g., created in the backlog) until it is delivered. This includes wait times before work begins, unlike cycle time, which solely focuses on the active work period. Since the question asks to calculate time when work starts till it is closed, Lead Time is not the correct answer.

In summary, the Cycle Time widget aligns precisely with the stated need to track and visualize the time it takes for work items to move from the start of work (In Progress) to completion (Closed). It provides a focused view on process efficiency and helps identify potential areas for improvement.

Authoritative Links for Further Research:

Cycle Time, Lead Time, and Throughput: https://www.visual-paradigm.com/scrum/cycle-time-vs-lead-time-vs-throughput/
What is cycle time in Agile? https://leancorner.com/what-is-cycle-time-in-agile/
Understand lead time, cycle time, and throughput to improve your DevOps processes: https://www.bmc.com/blogs/lead-time-cycle-time-throughput/

Question: 292 CertyIQ

You manage a project by using Azure Boards. You manage the project code by using GitHub.
You have three work items that have IDs of 456, 457, and 458.
You need to create a pull request that will be linked to all the work items. The solution must set the state of work item 456 to done.
What should you add to the commit message?

A.Fixes #456, #457, #458

B.Fixes #AB456, #AB457, #AB458

C.#456, #457, #458
Completed #456

D.#AB456, #AB457, #AB458

Answer: B

Explanation:

B. Fixes #AB456, #AB457, #AB458.

When integrating Azure Boards with GitHub, you can use specific keywords in commit messages or pull request descriptions to automatically link and update the state of work items.

Question: 293 CertyIQ

You manage code by using GitHub.
You plan to ensure that all GitHub Actions are validated by a security team.
You create a branch protection rule requiring that code changes be reviewed by code owners.
You need to create the CODEOWNERS file.
Where should you create the file?

A..github/actions/

B..github/

C..git/

D..github/workflows/

Answer: B

Explanation:

The correct location for the CODEOWNERS file in a GitHub repository is .github/. This file defines individuals or teams responsible for code within specific paths or files in the repository. GitHub uses this file to determine who should be automatically requested for review when a pull request modifies those areas.

Branch protection rules, like requiring code owner review, integrate with the CODEOWNERS file. When a pull request affects code covered by an entry in the file, the specified code owners are automatically added as reviewers. Placing the CODEOWNERS file in the .github/ directory ensures that GitHub correctly recognizes and utilizes it for code review processes.

Options A, C, and D are incorrect because they represent locations for different purposes: .github/actions/ is for custom GitHub Actions, .git/ is for Git's internal metadata (not intended for direct modification), and .github/workflows/ stores workflow definition files. While these directories are important for GitHub and CI/CD, they do not house the CODEOWNERS file, which is specific to code ownership and review workflows. The root of the repository is also a valid location, but .github/ is best practice.

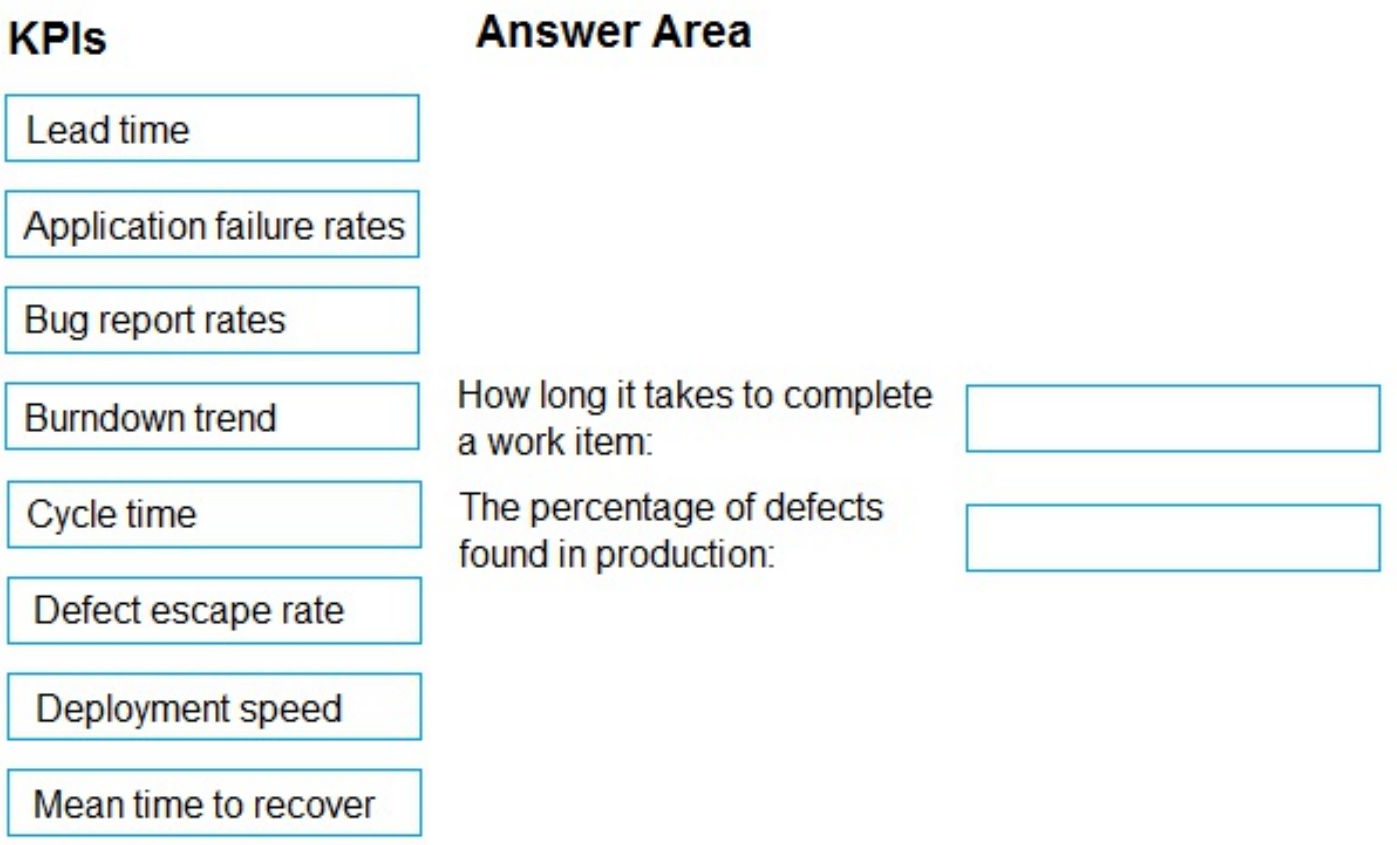
Therefore, placing the CODEOWNERS file in .github/ ensures that GitHub Actions and branch protection rules, particularly those requiring code owner reviews, function as intended, contributing to a secure and well-governed development process.

Relevant documentation:

About code owners
Creating a default file health file

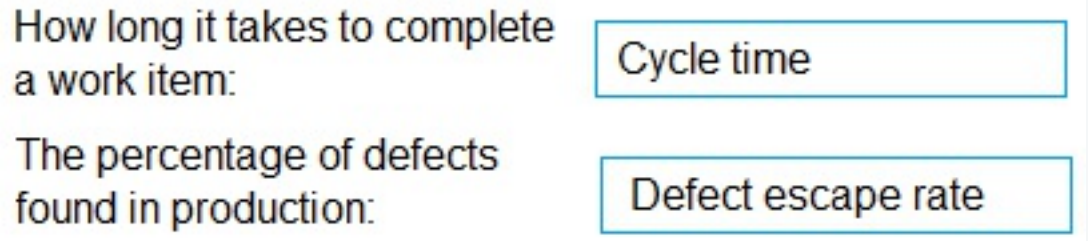
Question: 294 CertyIQ

DRAG DROP
-
You are implementing a new project in Azure DevOps.
You need to assess the performance of the project. The solution must identify the following metrics:
•How long it takes to complete a work item
•The percentage of defects found in production
Which DevOps KPI should you review for each metric? To answer, drag the appropriate KPIs to the correct metrics. Each KPI may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:

Answer Area



Question: 295 CertyIQ

You manage a project by using Azure Boards. You manage the project code by using GitHub.
You have three work items that have IDs of 456, 457, and 458.
You need to create a pull request that will be linked to all the work items. The solution must set the state of work item 456 to done.
What should you add to the commit message?

A.Done #456, #457, #458

B.#AB456, #AB457, #AB458

C.#456, #457, #458
Completed #456

D.#AB456, #AB457, #AB458
Completed #AB456

Answer: D

Explanation:

The correct answer is **D. #AB456, #AB457, #AB458Completed #AB456**

Here's why:

Azure Boards supports linking work items to commits and pull requests in GitHub. The syntax for linking work items is #WorkItemID. However, when the Azure DevOps organization is connected to GitHub, the work item IDs in commit messages need to be prefixed with #AB to be correctly recognized. This prefix informs Azure DevOps that the link refers to an Azure Boards work item within a specific Azure DevOps organization.

To resolve a work item automatically when the pull request is merged, you need to include a state-changing keyword like "Completed," "Fixes," or "Closes." Adding "Completed #AB456" in the commit message tells Azure DevOps to automatically change the state of work item 456 to "Done" (or the equivalent completed state in your project's workflow) upon merging the pull request. Including #AB457 and #AB458 will merely link to those work items in the commit history, without changing their states.

Option A is incorrect because it does not include the AB prefix. Also, it is attempting to change the state of all work items when the prompt only specifies work item 456.

Option B is incorrect because it is missing the keyword Completed to transition the work item to the Done state.

Option C is incorrect because it does not include the AB prefix. Also, it is incorrectly including the term Completed in conjunction with #456 instead of #AB456.

Here are authoritative links for further research:

Integrate Azure Boards with GitHub: https://learn.microsoft.com/en-us/azure/devops/boards/github/?view=azure-devops
Link from GitHub commits, pull requests, and issues: https://learn.microsoft.com/en-us/azure/devops/boards/github/link-to-from-github?view=azure-devops&tabs=new-nav

Question: 296 CertyIQ

You have an Azure subscription that contains an Azure container registry. The container registry contains an ACR Tasks task named Task1. Task1 is configured to run once every five days.
You need to trigger Task1 to run immediately.
Which command should you run?

A.az acr task run
B.az acr build
C.az acr taskrun
D.az acr run

Answer: A

Explanation:

The correct answer is **A. az acr task run**. This command is specifically designed to manually trigger an existing ACR Task in Azure Container Registry. ACR Tasks provide a way to automate container image builds and other container-related operations.

az acr task run: This command initiates a new execution of an existing ACR Task, regardless of its scheduled triggers. It is the explicit command for immediate task execution. It sends a request to the Azure Container Registry service to start the named task.

az acr build: This command is used for building container images from a Dockerfile and pushing them to a container registry. While it can be used to build images, it doesn't directly trigger a pre-configured ACR Task. It's more appropriate for ad-hoc builds.

az acr taskrun: This command lists the past task executions in Azure Container Registry, rather than triggering a new one. It helps track the history of task runs.

az acr run: This command allows you to quickly build and push a single container image to Azure Container Registry.

The key distinction is that the objective is to trigger an existing ACR Task that is already defined. az acr task run is the dedicated command for this purpose, offering a way to manually execute tasks outside their scheduled or event-driven triggers. It gives developers and operators direct control over task execution.

For further research, consult the official Microsoft Azure documentation:

az acr task run: https://learn.microsoft.com/en-us/cli/azure/acr/task?view=azure-cli-latest#az-acr-task-run

ACR Tasks overview: https://learn.microsoft.com/en-us/azure/container-registry/container-registry-tasks-overview

Question: 297 CertyIQ

You manage a project by using Azure Boards. You manage the project code by using GitHub.
You have three work items that have IDs of 456, 457, and 458.
You need to create a pull request that will be linked to all the work items. The solution must set the state of work item 456 to done.
What should you add to the commit message?

A.#AB456, #AB457, #AB458
Completed #AB456

B.#456, #457, #458
Completed #456

C.Done #AB456, #AB457, #AB458

D.#AB456, #AB457, #AB458
Verifies #AB456

Answer: A

Explanation:

#AB456, #AB457, #AB458 → Links all three work items to the commit/pull request.

Completed #AB456 → Automatically updates work item 456's state to Done.

Question: 298 CertyIQ

You manage projects by using Azure Boards.
You have a current work item name itemA that is dependant on a work item named itemB.
You need to define the dependency for itemA.
What should you do in the web portal for Azure DevOps?

A.Add a Parent link to the user story of itemA.

B.From Backlogs, open the context menu, select Add link, and then select itemA. Set Link type to Successor and add the ID of itemB.

C.From itemA, open the Links tab, and then select Add link. Set Link type to References and add the ID of itemB.

D.From Backlogs, open he context menu, select Add link, and then select itemA. Set Link type to References and add the ID of itemB.

Answer: B

Explanation:

The correct answer is B. Let's break down why and explore related concepts.

The requirement is to define a dependency where itemA depends on itemB. This means itemA cannot start or complete until itemB is finished. In Azure Boards, dependencies are best represented using link types like "Successor" and "Predecessor."

Option B accurately describes how to create a dependency relationship between work items directly within Azure DevOps. By accessing the context menu from the Backlogs view and selecting "Add link," you can directly establish the link between the items. Setting the Link type to "Successor" on itemA and linking it to the ID of itemB clearly indicates that itemA is a successor of itemB, meaning it depends on itemB. itemB would then be considered a predecessor to itemA. This is the most straightforward and semantic way to represent a dependency in Azure Boards.

Option A is incorrect. A "Parent" link establishes a hierarchical relationship, not a dependency. While a parent might indirectly influence a child, it doesn't explicitly state that the child item depends on the parent being completed.

Option C is incorrect. "References" links create a simple connection between items, but don't define a sequential dependency. A reference link signifies that two items are related, but one is not necessarily blocked by the other. This isn't strong enough to represent a true dependency.

Option D is similar to C, using References instead of a direct dependency link type. Therefore, it's also incorrect.

In summary, Azure DevOps provides specific link types to manage dependencies effectively. The "Successor" and "Predecessor" link types are designed to create the kind of dependency required by the question. Using the context menu from the Backlogs view provides a convenient way to add this link directly to the affected work items.

For further reading and understanding of work item linking in Azure DevOps, please refer to these resources:

Add a link to relate work items: https://learn.microsoft.com/en-us/azure/devops/boards/queries/add-link?view=azure-devops&tabs=browser
Link work items: https://learn.microsoft.com/en-us/azure/devops/boards/backlogs/add-link?view=azure-devops&tabs=browser

Question: 299 CertyIQ

You manage projects by using Azure Boards.
You have a current work item name itemA that is dependant on a work item named itemB.
You need to define the dependency for itemA.
What should you do in the web portal for Azure DevOps?

A.From itemA, open the Links tab, and then select Add link. Set Link type to Related and add the ID of itemB.

B.From Backlogs, open the context menu, select Add link, and then select itemA. Set Link type to Successor and add the ID of itemB.

C.From itemA, open the Links tab, and then select Add link. Set Link type to References and add the ID of itemB.

D.From Queries, open the context menu, select Add link, and then select Existing Item. Set Link type to Affected By and add the ID of itemB.

Answer: B

Explanation:

The correct answer is **B. From Backlogs, open the context menu, select Add link, and then select itemA. Set Link type to Successor and add the ID of itemB.**

Here's a detailed justification:

We need to define that itemA is dependent on itemB. This means itemA cannot start until itemB is finished. Therefore, itemA is the successor of itemB. The successor link type is used to indicate that one work item comes after another in a sequence.

The process involves working directly from the backlog:

1. **Access Backlogs:** This allows you to visualize and manage all your work items within the project.
2. **Open Context Menu:** Navigating to the item that needs to link to another item.
3. **Select ItemA:** You are trying to work on itemA, the dependent item.
4. **Add link:** To create dependencies between work items.
5. **Set Link type to Successor:** This is crucial. "Successor" explicitly denotes that itemA follows itemB. Since itemA is dependent on itemB, it means itemA is the successor of itemB.
6. **Add the ID of itemB:** This completes the link by specifying which work item itemA is dependent on. The ID uniquely identifies the item.

Options A, C, and D are incorrect because they use incorrect link types or begin from the wrong location.

Option A: "Related" link type simply shows a connection, not dependency. "References" would be used if itemA mentions or cites itemB, not that itemA depends on itemB's completion.
Option C: "References" is similar to A, only shows one work item refers to another.
Option D: "Affected By" indicates that itemA is impacted by itemB, but doesn't define a strict predecessor-successor relationship. Furthermore, starting from queries is unusual for defining dependencies.

In Azure DevOps, defining the correct link type is crucial for accurate project tracking and dependency management. Using "Successor" ensures that project scheduling tools can correctly reflect the order in which tasks need to be completed.

Authoritative Links for Further Research:

About linking work items: https://learn.microsoft.com/en-us/azure/devops/boards/queries/linking-work-items?view=azure-devops
Relationship types and directions: https://learn.microsoft.com/en-us/azure/devops/boards/queries/link-type-reference?view=azure-devops

Question: 300 CertyIQ

You have the services shown in the following table.

Name	Interface type
Service1	HTTP
Service2	HTTPS

You manage a project by using Azure Boards.
You need to notify the services of build status changes.
Which services can be notified by using a webhook?

A.Service1 only
B.Service2 only
C.Service1 and Service2

Answer: C

Explanation:

Correct answer is C:Service1 and Service2.

https://learn.microsoft.com/en-us/azure/devops/service-hooks/services/webhooks?view=azure-devops#q-can-i-send-webhooks-to-non-https-endpoints

Question: 301 CertyIQ

DRAG DROP
-
You are using Agile process methodologies and Azure Boards in Azure DevOps.
You import the work items shown in the following table.

Name	Description
Item1	The solution was implemented but **NOT** verified. The item must appear on the backlog.
Item2	The work was **NOT** started. The item must appear on the backlog.
Item3	The work is complete. The item must **NOT** appear on the backlog. The item must appear on the board.

You need to categorize the items.
Which state should you assign to each item? To answer, drag the appropriate states to the correct items. Each state may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

States

- Completed
- In Progress
- Proposed
- Removed
- Resolved

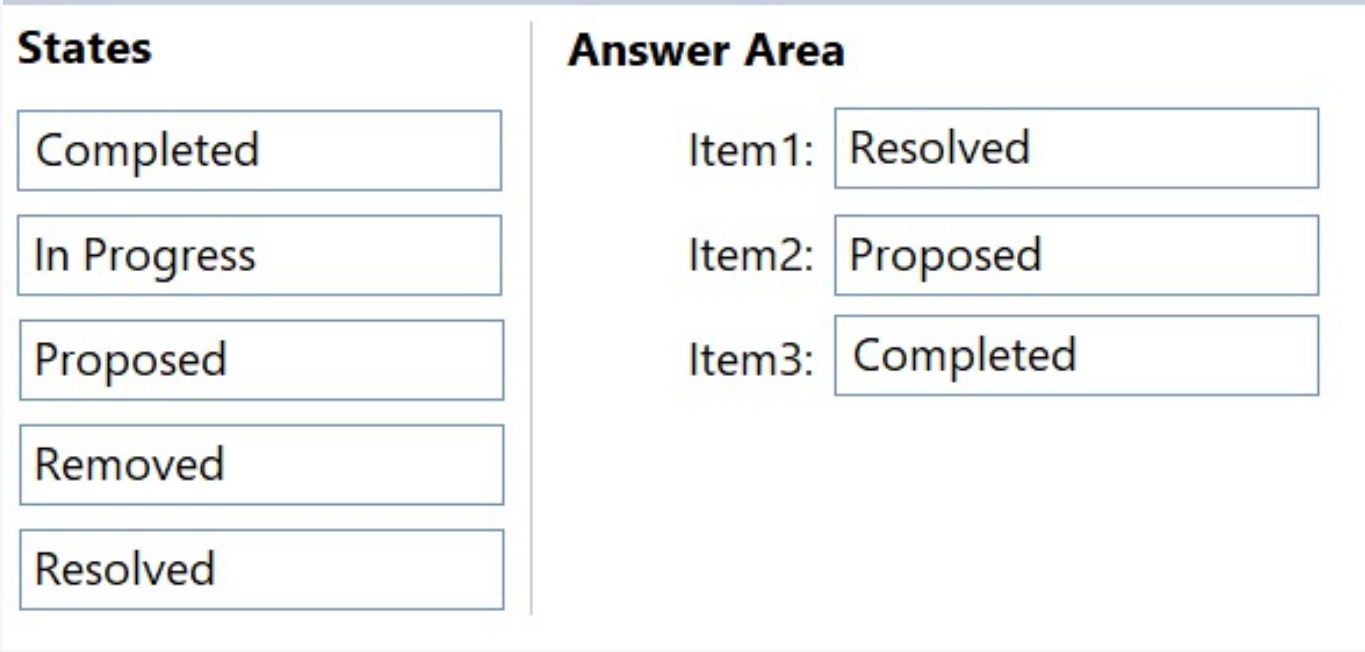
Answer Area

Item1:

Item2:

Item3:

Answer:



Explanation:

Item1: **Resolved.**

This likely indicates the item has been worked on and a fix is applied but not yet marked as done.

Item2: **Proposed.**

This is the starting state, meaning it's still a suggestion or idea.

Item3: **Completed.**

Final state when work is done, tested, and accepted.

https://learn.microsoft.com/en-us/azure/devops/boards/work-items/workflow-and-state-categories?view=azure-devops&tabs=agile-process#category-states

Question: 302 CertyIQ

You have an Azure subscription that contains an Azure Pipelines pipeline named Pipeline1 and an app named App1. Pipeline1 is used to automate the building of App1.
You have a Slack channel named App1chat that includes an incoming webhook.
You need to ensure that when a successful build of App1 is created, a notification is sent to App1chat by using the webhook.
What should you use?

A.a notification
B.an alert rule
C.a subscription
D.an action group

Answer: C

Explanation:

The correct answer is **C. a subscription**. Here's a detailed justification:

A subscription in Azure DevOps, specifically a service hook subscription, is the mechanism used to listen for specific events within Azure DevOps and trigger actions in external services. In this case, the event is a successful build in Pipeline1, and the action is sending a notification to the Slack channel App1chat via the incoming webhook.

Service hooks are designed for integrating Azure DevOps with other services. They allow you to define a specific event (e.g., build succeeded, work item created) and a corresponding action to be taken when that event occurs. The action typically involves sending a payload to an external service.

Incoming webhooks in Slack provide a simple way to post messages from external applications, like Azure DevOps, into a Slack channel.

By creating a subscription in Azure DevOps, you can configure it to listen for the "build succeeded" event in Pipeline1. The subscription will then send a JSON payload to the Slack webhook URL, which will format and display the message in the App1chat channel.

Options A, B, and D are incorrect because they aren't directly suited for this event-driven integration.

Notifications within Azure DevOps primarily manage notifications for users within the Azure DevOps organization itself regarding events like work item assignments or pull requests, not external services.

Alert rules in Azure Monitor are used to monitor Azure resources for specific conditions (e.g., high CPU usage, low disk space) and trigger alerts based on these conditions. They don't integrate directly with Azure DevOps pipeline events and Slack webhooks in the desired manner.

Action groups are used with Azure Monitor alert rules to define the actions to take when an alert is triggered. While an action group could potentially send a notification to an external service, it's not the primary or direct way to respond to Azure DevOps build events and send messages to Slack.

Therefore, a service hook subscription is the correct and most direct way to connect a successful build event in Azure Pipelines to a notification sent to a Slack channel via an incoming webhook.

Authoritative Links:

Azure DevOps Service Hooks: https://learn.microsoft.com/en-us/azure/devops/service-hooks/overview?view=azure-devops
Slack Incoming Webhooks: https://api.slack.com/messaging/webhooks

Question: 303 CertyIQ

You manage projects by using Azure Boards. You manage project code by using GitHub.
You have a work item that has an ID of 123.
You need to link work item 123 to a new pull request.
What are two ways to achieve this goal? Each correct answer presents a complete solution.
NOTE: Each correct solution is worth one point.

A.In the Development section for work item 123, select Add link, and then enter the URL of the pull request.

B.To the description of the pull request, add #AB123.

C.To work item 123 add a comment that includes the URL of the pull request.

D.From work item 123, open the Links tab, select Add link, select Existing item, and then enter the URL of the commit.

Answer: AB

Explanation:

The correct answer is AB. Here's why:

Option A: In the Development section for work item 123, select Add link, and then enter the URL of the pull request. This method directly links the work item to the pull request within Azure Boards. By adding a link from the Development section, you establish a formal connection, which allows users to easily navigate between the work item and the associated code changes represented by the pull request. This provides traceability and context for the work item's progress. This utilizes Azure Boards' built-in linking capabilities.

Option B: To the description of the pull request, add #AB123. Using the #AB123 notation (where AB stands for the project name or organization), Azure DevOps automatically recognizes this as a link to work item 123 when used in a pull request description, commit message, or comment. This simplifies the linking process directly from GitHub or Azure DevOps repositories. The system parses these notations and creates the necessary links. This approach leverages Azure DevOps' support for hashtag-based linking.

Option C is incorrect because simply adding the URL of the pull request as a comment doesn't create a formal, tracked link within Azure DevOps, making it harder to track related activities. Option D is incorrect because linking to a commit, while related, isn't the primary way to link a work item to a pull request. A pull request encompasses multiple commits and represents the overall review and merge process.

Further Research:

Link work items to development objects: https://learn.microsoft.com/en-us/azure/devops/boards/work-items/work-items-linked-objects?view=azure-devops
Drive Git development from a work item: https://learn.microsoft.com/en-us/azure/devops/boards/work-items/work-items-drive-git-dev?view=azure-devops

Question: 304 CertyIQ

DRAG DROP
-
You plan to use Azure DevOps in development processes.
You identify the following issues with the existing development processes:
•Frequent discovery of bugs in production
•Increasing detection of bugs by customers
•Slow resolution of bugs and slow resumption of production
You need to implement Key Performance Indicators (KPIs) to track each issue.
Which KPI should you implement for each issue? To answer, drag the appropriate KPIs to the correct issues. Each KPI may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

KPIs

- Application failure rates
- Bug report rates
- Defect escape rate
- Deployment failure rates

Answer Area

Frequent discovery of bugs in production: [ ]

Increasing detection of bugs by customers: [ ]

Slow resolution of bugs and slow resumption of production: [ ]

Answer:

Answer Area

Frequent discovery of bugs in production:	Defect escape rate
Increasing detection of bugs by customers:	Bug report rates
Slow resolution of bugs and slow resumption of production:	Application failure rates

Explanation:

Defect escape rate.

Bug report rates.

Application failure rates.

Question: 305 CertyIQ

You manage projects by using Azure Boards.
You have a current work item name itemA that is dependant on a work item named itemB.
You need to define the dependency for itemA.
What should you do in the web portal for Azure DevOps?

A.Add a Parent link to the user story of itemA.

B.From Backlogs, open the context menu, select Add link, and then select itemA. Set Link type to References and add the ID of itemB.

C.From itemA, open the Links tab, and then select Add link. Set Link type to References and add the ID of itemB.

D.From itemA, open the Links tab, and then select Add link. Set Link type to Successor and add the ID of itemB.

Answer: D

Explanation:

The correct answer is **D. From itemA, open the Links tab, and then select Add link. Set Link type to Successor and add the ID of itemB.**

Here's a detailed justification:

Understanding Dependencies in Azure Boards: Azure Boards provides mechanisms to link work items together to represent dependencies. These links are crucial for visualizing project workflows and understanding how tasks are related.

Identifying the Correct Link Type: The question states that itemA is dependent on itemB. This means itemA cannot start until itemB is completed. Therefore, itemA is a "Successor" of itemB. The relationship indicates that itemB precedes itemA.

Using the Links Tab: The "Links" tab within a work item in Azure DevOps is the dedicated area for establishing relationships with other work items. This is the standard method for defining dependencies.

Steps to Define the Dependency: You open itemA, go to its "Links" tab, and use the "Add link" feature. Here, you specify the "Link type" as "Successor" and provide the ID of itemB (the work item that itemA depends on).

This creates a "Successor" link from itemA to itemB, showing that itemA follows itemB.

Why Other Options are Incorrect:

A. Add a Parent link to the user story of itemA: A "Parent" link implies a hierarchical relationship, not a dependency. Parent-child relationships are used for breaking down larger tasks into smaller subtasks, not for defining dependencies.
B. From Backlogs, open the context menu, select Add link, and then select itemA. Set Link type to References and add the ID of itemB: While you can add links from the Backlogs view, "References" is a generic link type that indicates a general association. It doesn't explicitly define the dependency relationship between itemA and itemB.
C. From itemA, open the Links tab, and then select Add link. Set Link type to References and add the ID of itemB: As with option B, "References" lacks the specific definition of a predecessor-successor dependency needed to clarify that one item is dependent on the completion of the other.

Cloud Computing and Project Management Concepts: Defining dependencies is a fundamental aspect of project management, especially in cloud-based environments like Azure DevOps. Clear dependency mapping helps teams coordinate work, identify critical paths, and manage risks. Incorrectly defining dependencies can lead to scheduling errors, blocked tasks, and project delays.

Authoritative References:

Azure DevOps - Link work items: https://learn.microsoft.com/en-us/azure/devops/boards/queries/add-link-existing-work-items?view=azure-devops&tabs=browser
Relationship types and direction: https://learn.microsoft.com/en-us/azure/devops/boards/queries/link-type-reference?view=azure-devops

In summary, option D correctly uses the "Successor" link type within the "Links" tab of itemA to represent the dependency relationship, which is the appropriate approach within Azure DevOps.

Question: 306 CertyIQ

You have a project in Azure DevOps named App Project that is used to develop an app named App1. App1Project has an Azure Boards team dashboard that is used to monitor the progress of App1 and track work items.
You need to track how long it takes to close a work item once work for the item has commenced.
Which type of widget should you add to the dashboard?

A.sprint burndown
B.velocity
C.lead time
D.cycle time

Answer: D

Explanation:

The correct answer is D, Cycle Time. Here's why:

The key requirement is to track the time taken to close a work item once work has commenced. This directly aligns with the definition of Cycle Time. Cycle Time specifically measures the elapsed time from when work actively starts on an item until it's completed (closed).

Cycle Time: This is the time it takes for a team to complete a work item, starting from when they begin

actively working on it until it's closed. This directly addresses the requirement.

Now, let's examine why the other options are incorrect:

Sprint Burndown: This visualizes the remaining work within a sprint. It's focused on tracking the scope of work to be completed during a specific sprint, not the individual time taken to complete specific work items after work has started.

Velocity: Velocity measures the amount of work a team can complete in a sprint. It's a high-level measure of team capacity and isn't relevant to tracking the duration of individual work item completion.

Lead Time: This measures the total time from when a work item is created to when it's completed. The requirement specifically asks for the time once work has commenced, making Lead Time unsuitable as it includes wait times before work starts.

Therefore, Cycle Time is the only widget that specifically fulfills the need to track the time from the start of active work on a work item to its closure. A cycle time widget helps identify bottlenecks and areas for process improvement within the development workflow. By analyzing cycle times, the team can understand where time is being spent and where efficiencies can be gained, leading to faster delivery of features and bug fixes.

Further resources:

Azure DevOps Dashboards: https://learn.microsoft.com/en-us/azure/devops/report/dashboards/dashboards?view=azure-devops
Agile Metrics: https://www.visual-paradigm.com/scrum/what-is-agile-metrics/

Question: 307 CertyIQ

You manage a project by using Azure Board, and you manage the project code by using Azure Repos.
You have a bug work item that has an ID of 123.
You need to set the work item state to Resolved.
What should you add to the commit message?

A.#123 completes
B.Resolves #AB-123
C.Verifies #123
D.Fixes #123

Answer: D

Explanation:

The correct commit message syntax to transition a work item to the Resolved state in Azure Boards is "Fixes #123". This syntax leverages Azure DevOps's automatic linking and state transition capabilities. When Azure DevOps detects this specific syntax in a commit message pushed to Azure Repos, it automatically links the commit to the work item with ID 123 and transitions its state to Resolved. Other syntaxes such as "completes", "resolves", or "verifies" coupled with the correct work item ID will not achieve the desired Resolved state. "Completes" might close the work item depending on your process configuration. "Resolves" is a valid keyword but may not trigger an automatic Resolved state transition, and "Verifies" doesn't typically link to state changes. The "Fixes" keyword is specifically designed to address and resolve a work item, making it the correct choice. The prefix "AB-" isn't standard in the question, so the second option is also invalid.

Here's why the other options are incorrect:

A.#123 completes: While this might close a work item, depending on the process template and its configuration, it's not the specific trigger for the "Resolved" state.
B.Resolves #AB-123: "Resolves" is a valid keyword that usually creates an association between the commit and work item, but might not automatically trigger the "Resolved" state. Also, the AB- prefix is not mentioned in the question context.
C.Verifies #123: This doesn't typically relate to state transitions; it generally denotes that a commit verifies the resolution of a related work item (testing), but wouldn't by itself set the state to Resolved.

Supporting Documentation:

Link from Git commits, pull requests, and builds: https://learn.microsoft.com/en-us/azure/devops/boards/work-items/work-item-link-git-commit?view=azure-devops
Drive Git development from a work item: https://learn.microsoft.com/en-us/azure/devops/boards/work-items/drive-development-from-work-item?view=azure-devops

Question: 308 CertyIQ

You manage a project by using Azure Boards, and you manage the project code by using GitHub repositories.
You have a work item that has an ID of 123.
You need to link a commit message in GitHub to work item 123 on the board.
What are two ways to achieve the goal? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A.From the Development settings of work item 123, select Add link, and then enter the URL of the commit.
B.Add AB#123 to the text of the commit message.
C.Add GH-123 to the text of the commit message.
D.From the Links settings of work item 123, select Add link, select Existing item, and then enter the URL of the commit.
E.To work item 123, add a comment that includes the URL of the commit.

Answer: AB

Explanation:

The correct answers are **B. Add AB#123 to the text of the commit message** and **A. From the Development settings of work item 123, select Add link, and then enter the URL of the commit.**

Here's the justification:

Option B: Add AB#123 to the text of the commit message. This is the preferred and most automated approach when integrating Azure Boards with GitHub. Azure Boards automatically recognizes work item IDs prefixed with AB# in commit messages and links the commit to the corresponding work item. This method simplifies the workflow and allows developers to quickly establish connections between code changes and work being tracked in Azure Boards. This approach is documented in Microsoft's official documentation for integrating Azure Boards with GitHub.

Integrate Azure Boards with GitHub

Option A: From the Development settings of work item 123, select Add link, and then enter the URL of the commit. While less automated than using the AB# prefix, manually linking the commit is also a valid method. The "Development" section within an Azure Boards work item is specifically designed for tracking associated development artifacts (like commits, pull requests, etc.). By using "Add link" and providing the commit's URL, you create a direct relationship between the work item and the specific commit in GitHub. This ensures

traceability and context for the work being done.

While Microsoft does not explicity state that you can link to a commit URL from the Development settings, it mentions the use of the Development section for code changes related to the work item: Reference and Track work in Azure Boards

The incorrect options are:

Option C: Add GH-123 to the text of the commit message. GH- prefix is not the standard format for Azure Boards. Azure Boards recognizes AB# to link with work items. Therefore, this option will not link the commit to the work item.
Option D: From the Links settings of work item 123, select Add link, select Existing item, and then enter the URL of the commit. Though the 'Links' section is used for establishing connections between work items, the 'Development' section is designated for integrating development-related entities such as commits, branches, and pull requests. Adding a commit link within the 'Links' tab might not be semantically correct or fully leverage the features intended for development artifact tracking.
Option E: To work item 123, add a comment that includes the URL of the commit. Adding the commit URL as a comment provides visibility but does not create a formal link in the Development section. Therefore, it doesn't fully utilize the integration features for tracking linked code changes within Azure Boards. It also doesn't provide the automated linking capabilities.

Question: 309 CertyIQ

You have a project in Azure DevOps that uses an Azure Boards board and stores code in a GitHub repository. The repository contains a file named README.md.
You need to ensure that README.md includes the status of the work items on the board. The solution must minimize administrative effort.
What should you do first?

A.Create a GitHub personal access token (PAT).
B.Enable GitHub annotations for the board.
C.Install the Azure Boards app for GitHub.
D.Select Allow anonymous users to access the status badge.

Answer: C

Explanation:

The correct answer is **C. Install the Azure Boards app for GitHub.** Here's why:

The core requirement is to integrate the status of Azure Boards work items into a README.md file within a GitHub repository while minimizing administrative overhead. This integration involves a two-way communication between Azure Boards and GitHub. The Azure Boards app for GitHub is specifically designed to facilitate this integration.

Installing the Azure Boards app allows you to directly link work items in Azure Boards to commits, pull requests, and issues in your GitHub repository. It enables features such as mentioning work items in commit messages and pull request descriptions, which automatically update the status of the work items on the board. This creates a streamlined workflow where development activities are automatically reflected in the project's progress tracking. This linking capability is essential for dynamically reflecting the work item status in the README.md file.

Specifically, after installing the app, you can then use special syntax within the README.md file (like #ID,

where ID is the work item ID) to create a status badge. Azure Boards will automatically recognize this and update the badge with the current status of the linked work item. The Azure Boards app sets up the necessary authorization and data exchange infrastructure to enable this.

Option A (creating a GitHub PAT) would be necessary if you were interacting with the GitHub API directly, but it's more cumbersome to manage for this level of integration. Option B (enabling GitHub annotations) focuses on code quality within pull requests, not direct status integration in README files. Option D (allowing anonymous access) may seem related to displaying the status, but it only applies after the fundamental connection and integration are established. It's a security consideration about who can see the status once it's generated, but the primary step is setting up the ability to generate and display the badge in the first place. The Azure Boards app handles the authentication and authorization for accessing and displaying work item data in a secure and manageable way, preventing the need for managing credentials directly.

Therefore, installing the Azure Boards app for GitHub is the foundational step for enabling the desired integration between Azure Boards and GitHub with minimal administrative effort. It establishes the necessary link to allow work item status to be displayed directly within the GitHub README.md file.

Supporting Documentation:

Azure Boards app for GitHub: https://learn.microsoft.com/en-us/azure/devops/boards/github/?view=azure-devops
Configure integration with Azure Boards: https://learn.microsoft.com/en-us/azure/devops/boards/github/configure?view=azure-devops

Question: 310 CertyIQ

You manage a project by using Azure Boards, and you manage the project code by using Azure Repos.
You have a bug work item that has an ID of 123.
You need to set the work item state to Resolved.
What should you add to the commit message?

A.#123 fixed
B.#123 Resolved-
C.Verifies #123-
D.Resolves #123-

Answer: D

Explanation:

The correct answer is **D. Resolves #123**.

The justification lies in how Azure Boards and Azure Repos are integrated to manage work items through commit messages. Azure DevOps provides a feature to automatically update work item states based on specific keywords used in commit messages. These keywords trigger actions like resolving or closing a work item directly from the commit history.

The Resolves keyword (or Fixes and Closes) is a recognized command by Azure DevOps. When a commit message contains "Resolves #123", it instructs Azure DevOps to transition work item #123 to the "Resolved" state upon successful commit and push. This workflow streamlines the process of managing work items by tying code changes directly to their resolution.

Option A, #123 fixed, while indicating the bug is fixed, doesn't use a predefined keyword that Azure DevOps

recognizes to automatically change the work item state to Resolved. Option B, #123 Resolved, is simply restating that the issue is resolved in the commit message; it does not trigger a state change in Azure Boards. Option C, Verifies #123, implies that the commit message verifies that the bug is resolved, but does not trigger state change in work item #123. The "Verifies" keyword is not an option for automatically changing states.

Therefore, to automatically transition a bug work item (ID 123) to the Resolved state via a commit message, the keyword Resolves should be used, as in Resolves #123.

Further reading on linking work items to Git development:Link from Git commit, pull request, and other objectsDrive Git development from a work item

Question: 311 CertyIQ

You manage projects by using Azure Boards.
You have a current work item named itemA that is dependent on a work item named itemB.
You need to define the dependency for itemA.
What should you do in the web portal for Azure DevOps?

A.Add a Parent link to the user story of itemA.

B.From Backlogs, open the context menu, select Add link, and then select itemA. Set Link type to Successor and add the ID of itemB.

C.From Backlogs, open the context menu, select Add link, and then select itemB. Set Link type to Related and add the ID of itemA.

D.From itemA, open the Links tab, and then select Add link. Set Link type to References and add the ID of itemB.

Answer: B

Explanation:

The correct answer is B because it accurately reflects the proper method for defining a dependency where itemA is dependent on itemB within Azure Boards.

Let's break down why the other options are incorrect:

A. Add a Parent link to the user story of itemA. Parent links establish hierarchical relationships, not dependencies. A parent link would imply itemA is a child of itemB, not that itemA is blocked by itemB or dependent on its completion.

C. From Backlogs, open the context menu, select Add link, and then select itemB. Set Link type to Related and add the ID of itemA. While a "Related" link could indicate a connection, it's not specific enough for a dependency. "Related" links are more general and don't inherently imply that one item is blocking or dependent on another. More critically, this attempts to link from itemB to itemA. We need to indicate that itemA is dependent on itemB.

D. From itemA, open the Links tab, and then select Add link. Set Link type to References and add the ID of itemB. "References" link type indicates that itemA references some work being done in itemB. It doesn't denote the strict "dependent on" relationship. Similar to "Related," this lacks the proper semantic meaning needed to signify the dependency.

Option B, "From Backlogs, open the context menu, select Add link, and then select itemA. Set Link type to Successor and add the ID of itemB," is correct because:

1. We are initiating the link from itemA (the dependent item).

2. The Successor link type is commonly used (though 'Depends On' or 'Predecessor' might be technically more appropriate depending on your process template) to show that a work item depends on the completion of another work item. This ensures a clear and actionable dependency link. Although "Successor" is not ideal nomenclature, it is the closest of the given choices.
3. By adding the ID of itemB as the linked item (the 'Successor' to itemA), we clearly establish that itemA is dependent on itemB. The dependency is now formally defined in Azure Boards, aiding in project tracking and workflow management.
4. While you can add links from the work item form (as in option D), adding it from the backlog is a valid, and sometimes more convenient, approach, especially when managing multiple work items.

Therefore, choosing the Successor link from itemA (the dependent item) pointing to itemB (the item it depends on) correctly represents the required dependency.

Relevant documentation:

Add links to work items
Link type descriptions

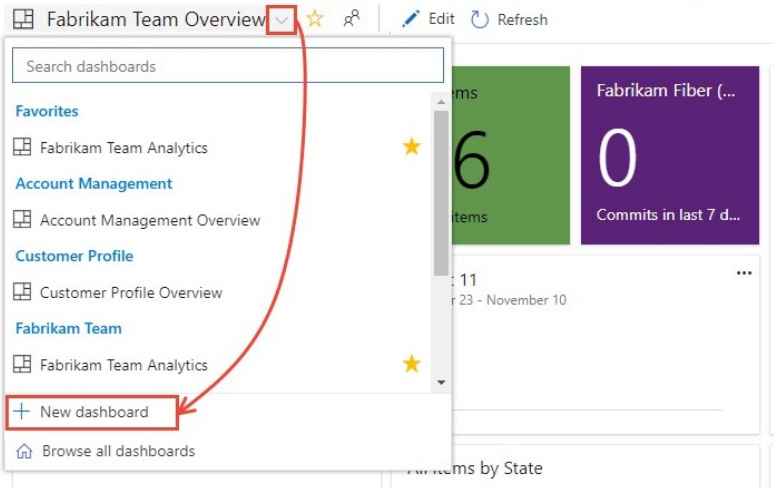
Question: 312 CertyIQ

SIMULATION
-
You need to create a new team dashboard named Dashboard1 for the default project team of Project1.
The dashboard must display the members of the team.

Answer:

Add dashboards in Azure DevOps
Add a dashboard
Add a new dashboard as needed to support your team's needs. You can also edit and rename any existing dashboards associated with your team.

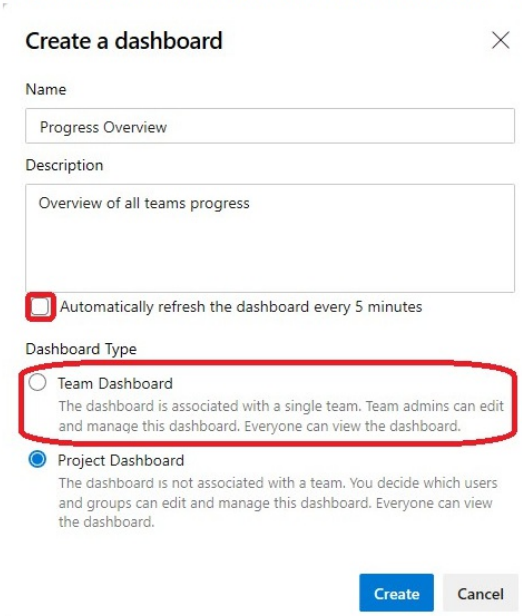
Step 1: From the Dashboards directory, select New Dashboard. Or, when viewing a dashboard, open the selector and select the New Dashboard option.



Step 2: Enter the name of the dashboard and other information you want to capture. [Enter Dashboard1]

Step 2a: Select Team dashboard.

Step 2b: Select: Automatically refresh the dashboard every 5 minutes.



Step 3: Select Create.

Reference:
https://learn.microsoft.com/en-us/azure/devops/report/dashboards/dashboards

Question: 313

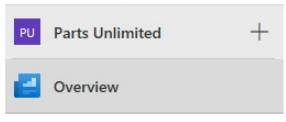
CertyIQ

SIMULATION
-
For Project1, you need to create a project wiki named Wiki1 that uses the Mermaid syntax to render a diagram.
A sample of the desired output is stored in C : \Resources\TCPHandshake.png.

Answer:

Task 1: Create a wiki for your project
Azure DevOps, Create a wiki for your project

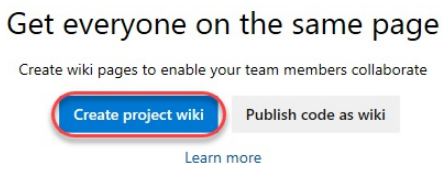
Step 1: In Project1, use the navigation to get to the Wiki hub.





Step 2: Since this project does not yet have its wiki configured, click Create project wiki to set one up. When you create your first wiki, Azure DevOps will provision a git repository that will store all your pages and artifacts.



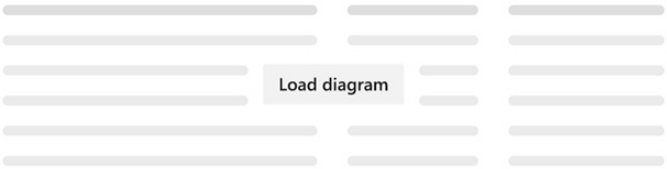


Task 2: Render a diagram using the Mermaid syntax

The actual wiki diagram syntax is in the same file as you wiki document. The syntax for the mermaid diagram is surrounded with the notation see below.

::: mermaid
<Mermaid Syntax>
:::

In the preview panel to the right of you text will display the below image. Clicking on Load diagram the mermaid syntax is used to generate the diagram.

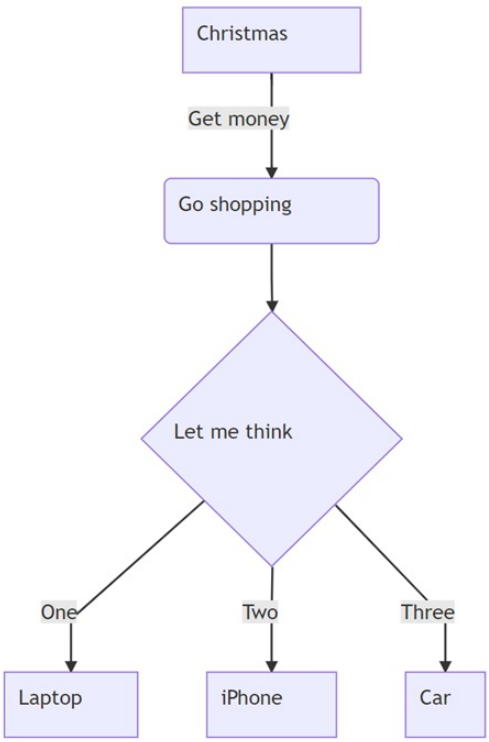


Depending on desired output.

A. Flow Chart Example

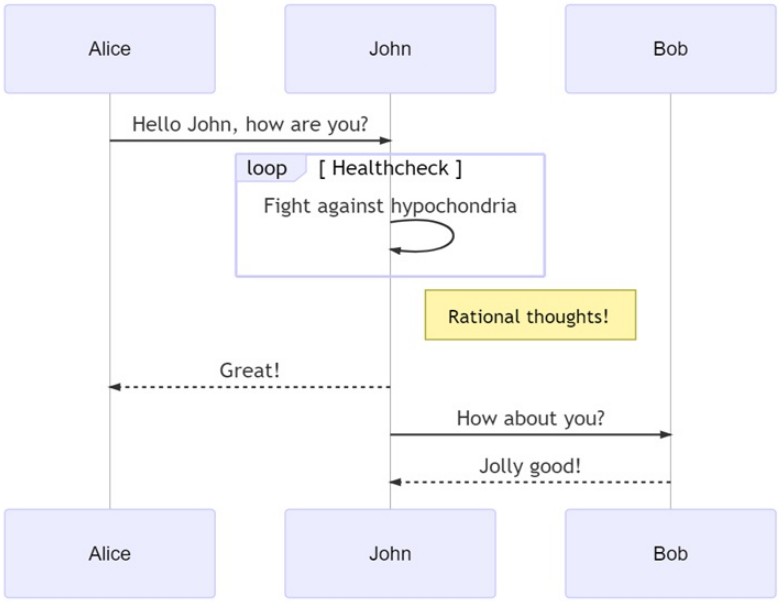
::: mermaid
graph TD
A[Christmas] -->|Get money| B(Go shopping)
B --> C{Let me think}
C -->|One| D[Laptop]
C -->|Two| E[iPhone]
C -->|Three| F[fa:fa-car Car]
:::

Generate a picture that looks like the below.

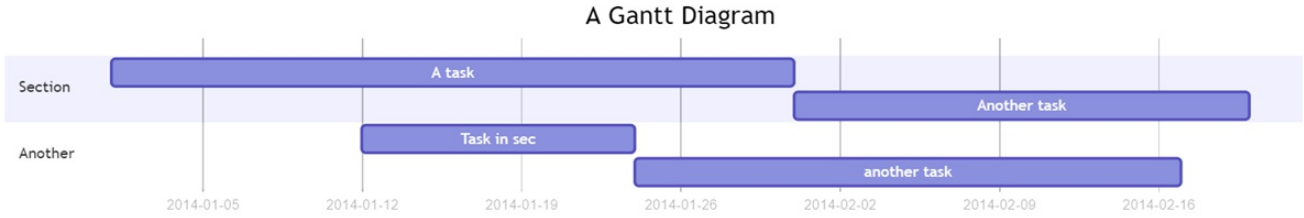


B. Sequence

::: mermaid
sequenceDiagram
Alice->>John: Hello John, how are you?
loop Healthcheck
John->>John: Fight against hypochondria
end
Note right of John: Rational thoughts!
John-->>Alice: Great!
John->>Bob: How about you?
Bob-->>John: Jolly good!
:::
Generate a picture that looks like the below.



C. Gantt Chart
::: mermaid
gantt
title A Gantt Diagram
dateFormat YYYY-MM-DD
section Section
A task :a1, 2014-01-01, 30d
Another task :after a1 , 20d
section Another
Task in sec :2014-01-12 , 12d
another task : 24d
:::
Generate a picture that looks like the below.



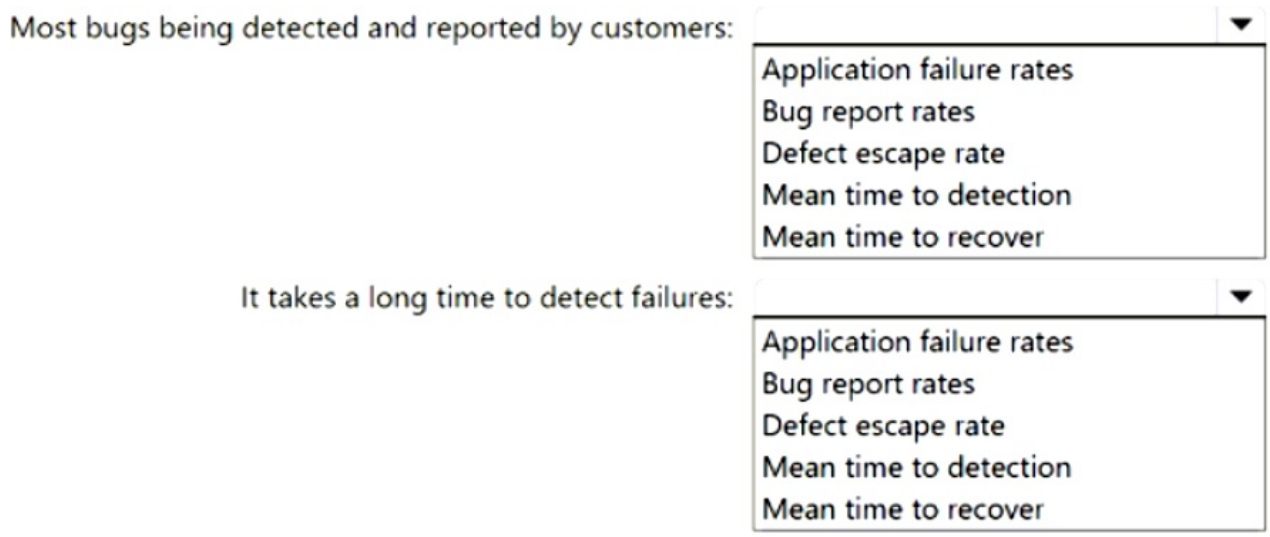
Reference:
https://www.azuredevopslabs.com/labs/azuredevops/wiki/
https://www.azuredevops.tips/wikimermaid-diagrams/

Question: 314

CertyIQ

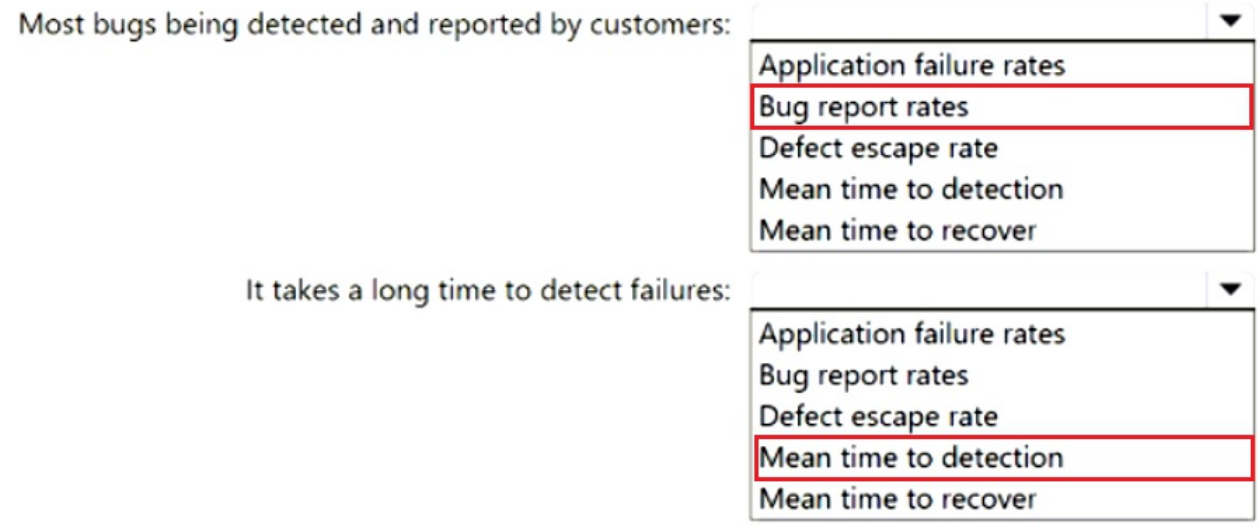
HOTSPOT
-
You have an Azure DevOps project that is used to build and test an app named App1.
You need to troubleshoot the following issues:
•Most bugs are detected and reported by customers.
•It takes a long time to detect failures.
Which metric should you review for each issue? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Explanation:

Bug report rates.

Mean time to detection.

Question: 315 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You need to recommend an integration strategy for the build process of a Java application. The solution must meet the following requirements:
- The build must access an on-premises dependency management system.
- The build outputs must be stored as Server artifacts in Azure DevOps.
- The source code must be stored in a Git repository in Azure DevOps.

Solution: Configure the build pipeline to use a Microsoft-hosted agent pool running the Windows Server 2019 with Visual Studio 2019 image. Include the Java Tool
Installer task in the build pipeline.
Does this meet the goal?

A. Yes

B. No

Answer: B

Explanation:

The proposed solution does not fully address the requirements. While using a Microsoft-hosted agent pool with the Java ToolInstaller task can facilitate Java builds and storing artifacts in Azure DevOps is straightforward, it fails to address the critical need for the build process to access an on-premises dependency management system.

Microsoft-hosted agents operate in Azure's cloud environment and typically do not have direct network connectivity to on-premises resources for security reasons. Consequently, the build pipeline running on a Microsoft-hosted agent will be unable to access the required on-premises dependency management system, leading to build failures.

To achieve on-premises connectivity, a self-hosted agent should be used. A self-hosted agent resides within the same network as the on-premises dependency management system, allowing the build pipeline to access it. Alternatively, one can establish a secure connection, such as a VPN or Azure Relay, between the Microsoft-hosted agent and the on-premises network, but this requires more complex configuration. Given the need for dependency access, the proposed solution falls short.

Therefore, the solution is incorrect as it does not meet the goal of accessing the on-premises dependency management system.

References:

Azure Pipelines Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=yaml&viewFallbackFrom=vsts
Self-hosted agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops

Question: 316 CertyIQ

You have a project in Azure DevOps.
You create the following YAML template named Template1.yml.
steps:
- script: npm install
- script: yarn install
- script: npm run compile
You create the following pipeline named File1.yml.
parameters:
usersteps:
- task:
- script: echo Done
You need to ensure that Template1.yaml runs before File1.yml.
How should you update File1.yml?

A. parameters: usersteps: extends: template: template1.yml - task: - script: echo Done

B. template: template1.yml parameters: usersteps: - task: - script: echo Done

C. extends: template: templatel.yml parameters: usersteps: - task: - script: echo Done

D. parameters: usersteps: - template: templatel.yml - task: - script: echo Done

Answer: C

Explanation:

The correct approach to ensure Template1.yml runs before File1.yml and allows parameterized insertion of steps is using the extends keyword along with the parameters section.

Option C, extends: template: templatel.yml parameters: usersteps: - task: - script: echo Done, is the correct solution.

The extends keyword allows one pipeline or template to inherit content from another. Here, File1.yml inherits the steps defined in Template1.yml. The parameters section, specifically the usersteps, provides a mechanism to insert additional steps into the pipeline defined by Template1.yml. The - task: and - script: echo Done steps are appended to the steps from Template1.yml. This ensures that Template1.yml's steps (npm install, yarn install, npm run compile) are executed first, followed by the parameterized steps.

Option A is incorrect because it uses extends inside the parameters section, which is not the intended syntax for extending a template and passing parameters to insert steps. The extends keyword should be at the root level of the YAML file.

Option B is incorrect because it places the template keyword at the root level instead of using extends. This would attempt to define a template, not extend one. Also, it lacks the extends keyword which is crucial for inheriting the steps from Template1.yml.

Option D is incorrect because it attempts to include the template directly as a step, which is not how Azure DevOps pipelines are structured to handle templates. Templates are not steps in themselves; they are extended to define or modify steps within the pipeline. Furthermore, a template cannot be declared as a step under usersteps.

In summary, extends handles the inheritance of the steps from Template1.yml and the parameters section allows for passing the additional steps to be executed after the inherited steps. This structure enables the correct execution order and parameterized pipeline construction.

For further research, refer to the official Microsoft Azure DevOps documentation on templates and pipeline syntax:

Use templates to reduce duplication: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/templates?view=azure-devops
YAML schema reference: https://learn.microsoft.com/en-us/azure/devops/pipelines/yaml-schema/extends?view=azure-devops

Question: 317 CertyIQ

You have an Azure solution that contains a build pipeline in Azure Pipelines.
You experience intermittent delays before the build pipeline starts.
You need to reduce the time it takes to start the build pipeline.
What should you do?

A. Enable self-hosted build agents.
B. Create a new agent pool.
C. Split the build pipeline into multiple stages.
D. Purchase an additional parallel job.

Answer: A

Explanation:

The best approach to reduce intermittent delays in starting an Azure Pipelines build pipeline is to enable self-hosted build agents (Option A). Here's why:

Azure Pipelines offers both Microsoft-hosted agents and self-hosted agents. Microsoft-hosted agents are convenient and require no management, but they operate within a shared pool. This means your pipeline has to wait for an available agent in that shared pool, leading to delays, especially during peak hours or when many builds are queued.

Self-hosted agents, on the other hand, are dedicated machines (VMs or physical servers) that you manage and control. By having your own agents, you eliminate the waiting time associated with the shared pool. When a build is triggered, it can start immediately on an available self-hosted agent, bypassing the queue. This is particularly beneficial for organizations with consistent build demands or those requiring specific software or configurations that are not readily available on Microsoft-hosted agents.

While purchasing an additional parallel job (Option D) allows more concurrent builds, it still relies on available agents. If all agents, whether Microsoft-hosted or self-hosted are busy, jobs will still queue, albeit potentially for a shorter duration. Self-hosting agents addresses the root cause of waiting for agent availability. Creating a new agent pool (Option B) doesn't directly address the delay; it simply organizes agents. Splitting the build pipeline into multiple stages (Option C) might improve parallelism within a single build, but it doesn't shorten the initial delay before the build process begins.

Therefore, self-hosted agents provide dedicated resources, ensuring quicker build pipeline starts and removing the bottleneck of waiting for shared resources. This is the most direct and effective way to address intermittent delays.

Relevant Resources:

Azure Pipelines Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=yaml%2Cbrowser
Self-hosted Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops

Question: 318 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You need to recommend an integration strategy for the build process of a Java application. The solution must meet the following requirements:
- The build must access an on-premises dependency management system.
- The build outputs must be stored as Server artifacts in Azure DevOps.
- The source code must be stored in a Git repository in Azure DevOps.

Solution: Configure the build pipeline to use a Microsoft-hosted agent pool running a Linux image. Include the Java Tool Installer task in the build pipeline.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution does not fully meet the stated goals. While using a Microsoft-hosted agent pool with a Linux image and including the Java Tool Installer task addresses the Java build environment setup, it fails to address the critical requirement of accessing an on-premises dependency management system. Microsoft-hosted agents operate within Azure's cloud infrastructure and do not inherently have connectivity to on-premises networks.

To access the on-premises dependency management system, a mechanism to bridge the network gap is required. This could involve setting up a VPN connection between Azure and the on-premises network, or using Azure Relay to expose the dependency management system to the cloud environment. Without such a connection, the build pipeline will not be able to resolve and download the necessary dependencies from the on-premises system, causing the build to fail.

The solution correctly suggests storing build outputs as Server artifacts in Azure DevOps, which is the intended functionality. The source code storage in a Git repository in Azure DevOps is also correct, given the requirement. However, the missing piece is the on-premises dependency access. Therefore, the solution is incomplete and does not fully meet the goal.

Resources for further research:

Connect to on-premises resources using VPN Gateway: https://learn.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about
Azure Relay: https://learn.microsoft.com/en-us/azure/relay/
Self-hosted agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=yaml%2Cbrowser (An alternative solution could involve using a self-hosted agent within the on-premises network, eliminating the need for a VPN or Relay)

Question: 319 CertyIQ

You store source code in a Git repository in Azure Repos. You use a third-party continuous integration (CI) tool to control builds.
What will Azure DevOps use to authenticate with the tool?

A. certificate authentication
B. a personal access token (PAT)
C. a Shared Access Signature (SAS) token
D. NTLM authentication

Answer: B

Explanation:

The correct answer is B: a personal access token (PAT). When a third-party tool, like a CI server, needs to access Azure DevOps resources (in this case, a Git repository in Azure Repos) on behalf of a user, it needs a secure way to authenticate. PATs are the recommended and most versatile method for this type of scenario.

Here's why:

PATs as Substitute Credentials: PATs act as alternative credentials that can be used in place of a username and password. They are specific to Azure DevOps and grant access to defined resources and scopes.
Granular Access Control: PATs allow you to specify the exact permissions the third-party tool requires (e.g.,

read, write, manage code). This principle of least privilege enhances security.
Revocability: PATs can be easily revoked at any time, which is crucial if the third-party tool is compromised or no longer needed. This immediately cuts off access.
Suitable for CI/CD: For CI tools needing to pull source code or push build artifacts, PATs provide the necessary authorization without exposing sensitive user credentials.
Certificate authentication (A): While certificate authentication is secure, it typically involves more complex setup and management compared to PATs, especially for third-party tool integrations.
Shared Access Signature (SAS) token (C): SAS tokens are primarily used for granting access to Azure Storage resources, not Azure DevOps services like Azure Repos.
NTLM authentication (D): NTLM is an older authentication protocol that is generally not recommended for modern cloud applications due to security vulnerabilities.

Therefore, using a PAT is the most straightforward, secure, and manageable method for a third-party CI tool to authenticate with Azure DevOps and access source code in Azure Repos.

Supporting Documentation:

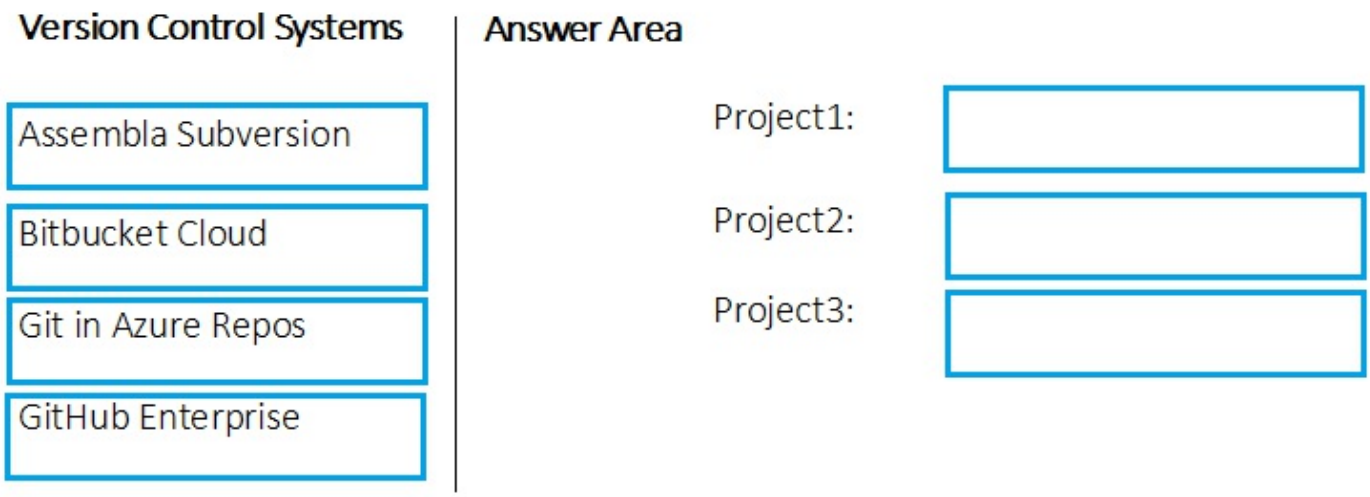
Use personal access tokens (PATs) to authenticate: https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops&tabs=preview-page
Authentication overview: https://learn.microsoft.com/en-us/azure/devops/integrate/concepts/auth-overview?view=azure-devops

Question: 320 CertyIQ

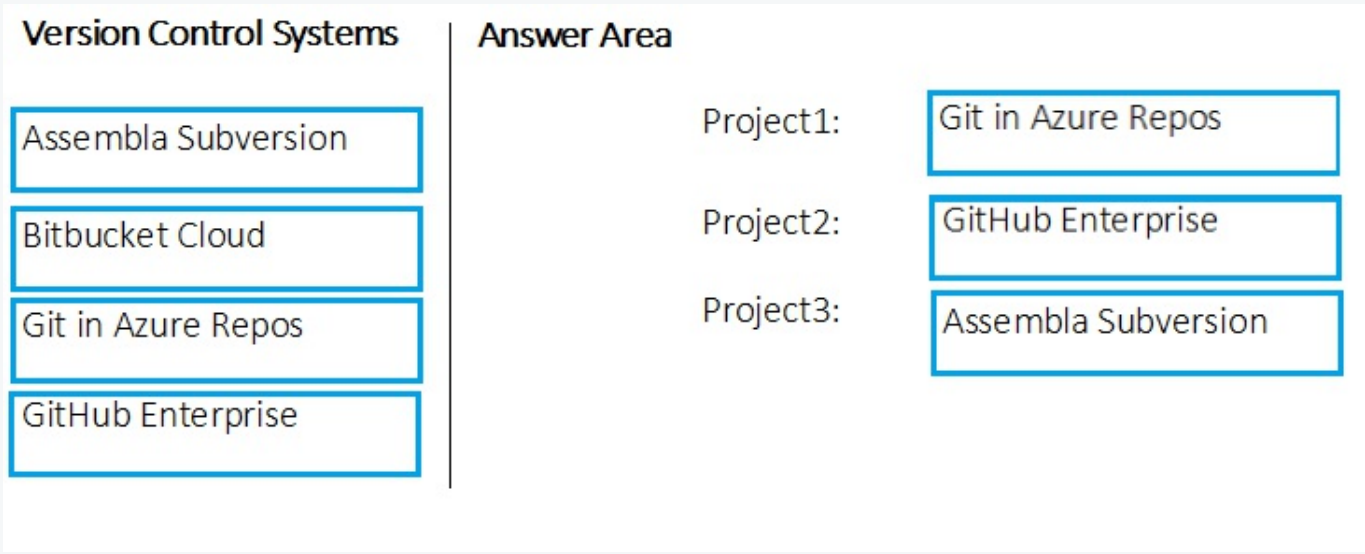
DRAG DROP -
You are configuring Azure Pipelines for three projects in Azure DevOps as shown in the following table.

Project name	Project Details
Project1	The project team provides preconfigured YAML files that it wants to use to manage future pipeline configuration changes.
Project2	The sensitivity of the project requires that the source code be hosted on the managed Windows server on your company's network.
Project3	The project team requires a centralized version control system to ensure that developers work with the most recent version.

Which version control system should you recommend for each project? To answer, drag the appropriate version control systems to the correct projects. Each version control system may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Project1:Git in Azure Repos .

Project2: Github Enterprise .

GitHub Enterprise is the on-premises version of GitHub.com. GitHub Enterprise includes the same great set of features as GitHub.com but packaged for running on your organization's local network. All repository data is stored on machines that you control, and access is integrated with your organization's authentication system (LDAP, SAML, or CAS).

Project3: Assembla Subversion.

Subversion is a centralized source code versioning system.

Question: 321 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You need to recommend an integration strategy for the build process of a Java application. The solution must meet the following requirements:

- The builds must access an on-premises dependency management system.
- The build outputs must be stored as Server artifacts in Azure DevOps.
- The source code must be stored in a Git repository in Azure DevOps.

Solution: Configure an Octopus Tentacle on an on-premises machine. Use the Package Application task in the build pipeline.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution is incorrect because it does not address the core requirements of the scenario. Specifically, it fails to provide a mechanism for the build process to access the on-premises dependency management system. An Octopus Tentacle is primarily used for deploying applications, not for accessing build dependencies.

The Package Application task is used for packaging application files for deployment, which is a separate concern from managing build dependencies. While this task might be used in the build pipeline, it doesn't directly help in accessing the on-premises dependency management system.

Instead, a more appropriate solution would involve using a self-hosted agent within the on-premises network. This agent would have direct access to the dependency management system, allowing the build process to retrieve the necessary dependencies. The build pipeline would then be configured to use this self-hosted agent. The build outputs can then be stored as Azure Artifacts through a PublishBuildArtifacts task, which is more aligned with storing server artifacts as requested.

Therefore, the proposed solution does not meet the specified goal of providing access to the on-premises dependency management system during the build process.

Relevant resources for further understanding:

Azure DevOps Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser
Azure Artifacts: https://learn.microsoft.com/en-us/azure/devops/artifacts/start/start?view=azure-devops
Publish Build Artifacts task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/utility/publish-build-artifacts?view=azure-devops
Octopus Deploy Tentacle: https://octopus.com/docs/architecture/agents/tentacles

Question: 322

CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You need to recommend an integration strategy for the build process of a Java application. The solution must meet the following requirements:
- The builds must access an on-premises dependency management system.
- The build outputs must be stored as Server artifacts in Azure DevOps.

The source code must be stored in a Git repository in Azure DevOps.

Solution: Install and configure a self-hosted build agent on an on-premises machine. Configure the build pipeline to use the Default agent pool. Include the Java
Tool Installer task in the build pipeline.
Does this meet the goal?

A. Yes

B. No

Answer: A

Explanation:

I have experienced in these types of questions where answer is - Yes / NO - and you can not go back - as you can get the clue - right answer i.e. " Yes " is the last questions usually :-)

Question: 323 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You need to recommend an integration strategy for the build process of a Java application. The solution must meet the following requirements:
- The builds must access an on-premises dependency management system.
- The build outputs must be stored as Server artifacts in Azure DevOps.
- The source code must be stored in a Git repository in Azure DevOps.

Solution: Configure the build pipeline to use a Hosted VS 2019 agent pool. Include the Java Tool Installer task in the build pipeline.
Does this meet the goal?

A. Yes

B. No

Answer: B

Explanation:

The proposed solution does not meet the goal.

Here's why:

1. **On-premises Dependency Management System Access:** Hosted agents, including Hosted VS2019, operate within Microsoft's Azure infrastructure and do not inherently have direct network access to your on-premises network where the dependency management system resides. To access the on-premises system, a connectivity solution such as Azure VPN Gateway, Azure ExpressRoute, or a self-hosted agent running within your on-premises network would be required. The provided solution doesn't configure this necessary network connectivity.

2. **Build Outputs as Server Artifacts:** While a Hosted VS2019 agent can build a Java application and potentially create artifacts, simply using the agent does not automatically store these outputs as Server artifacts in Azure DevOps. Additional tasks within the build pipeline, such as the "Publish Build Artifacts" task, are necessary to specifically upload and publish the build artifacts to Azure DevOps. The solution lacks this artifact publishing configuration.

3. **Java Tool Installer Task:** The Java Tool Installer task primarily focuses on acquiring and installing a specific version of the Java Development Kit (JDK) on the agent. This is a useful task, but it doesn't address the core requirements of accessing on-premises resources or publishing artifacts.

In summary, the solution only partially addresses the build process (installing the JDK) and completely misses

the critical aspects of network connectivity to on-premises resources and artifact publishing, which are crucial for achieving the stated goal.

Authoritative Links:

Azure DevOps Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=yaml%2Cbrowser
Publish Build Artifacts task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/utility/publish-build-artifacts?view=azure-devops
Connecting Azure DevOps to on-premises resources: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/v2-configure-endpoints?view=azure-devops

Question: 324 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You need to recommend an integration strategy for the build process of a Java application. The solution must meet the following requirements:
- The builds must access an on-premises dependency management system.
- The build outputs must be stored as Server artifacts in Azure DevOps.
- The source code must be stored in a Git repository in Azure DevOps.

Solution: Configure the build pipeline to use a Hosted Ubuntu agent pool. Include the Java Tool Installer task in the build pipeline.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution does not meet the stated goal because it fails to address the critical requirement of accessing an on-premises dependency management system. While using a Hosted Ubuntu agent pool and the Java Tool Installer task can ensure the availability of Java tools during the build process, these components do not inherently provide connectivity to an on-premises network or enable access to resources residing within that network.

To access on-premises resources, a self-hosted agent or a hybrid connection would be required. A self-hosted agent runs on a machine within the on-premises network and can therefore directly access the dependency management system. Alternatively, a hybrid connection can establish a secure tunnel between Azure DevOps and the on-premises network, allowing the build pipeline to access the dependency management system as if it were located within Azure.

Without either of these approaches, the build pipeline running on a hosted agent will be unable to resolve dependencies from the on-premises system, leading to build failures. Furthermore, even though the output would eventually be stored in Azure Artifacts, the build process won't complete successfully. Therefore, focusing solely on the agent type and Java tool installation without addressing network connectivity neglects a fundamental requirement of the scenario.

Supporting Links:

Azure DevOps Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-

devops&tabs=yaml&viewFallbackFrom=vsts
Hybrid Connections: https://learn.microsoft.com/en-us/azure/app-service/app-service-hybrid-connections

Question: 325 CertyIQ

Your company uses a Git repository in Azure Repos to manage the source code of a web application. The master branch is protected from direct updates.
Developers work on new features in the topic branches.
Because of the high volume of requested features, it is difficult to follow the history of the changes to the master branch.
You need to enforce a pull request merge strategy. The strategy must meet the following requirements:
- Consolidate commit histories.
- Merge the changes into a single commit.

Which merge strategy should you use in the branch policy?

A. squash merge
B. fast-forward merge
C. Git fetch
D. no-fast-forward merge

Answer: A

Explanation:

The correct answer is A, squash merge. Let's break down why.

The scenario emphasizes a need to consolidate commit histories on the master branch and merge changes into a single commit. This directly points to the benefits of a squash merge. A squash merge takes all the commits from a feature branch and combines them into a single commit on the target branch (in this case, master). This creates a cleaner, more linear history, making it easier to follow the progression of features as a whole rather than individual development steps.

Fast-forward merge (B) simply moves the branch pointer forward if the target branch hasn't diverged. This doesn't consolidate commits and would not meet the stated requirements. Git fetch (C) is a Git command for retrieving changes from a remote repository and does not pertain to merge strategies. No-fast-forward merge (D) creates a merge commit even when a fast-forward is possible, preserving branch history but not consolidating it into a single commit. This makes the history more complex, counter to the specified need.

Squash merging effectively provides a high-level overview of changes committed to master. While it loses the granular commit history of the feature branch, this is a trade-off made to achieve a clean and easily understandable master branch history. This strategy is particularly useful when dealing with a high volume of feature branches, as it prevents the master branch from becoming cluttered with potentially noisy or detailed intermediate commits. A single descriptive commit on master provides a clear record of the implemented feature. In the context of Azure DevOps pull requests, this configuration is readily enabled within the branch policy settings for the master branch, ensuring enforcement of the selected merge strategy.

Further reading:

Azure Repos Branch Policies: https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops
Git Merge Strategies: https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Question: 326 CertyIQ

Your company uses cloud-hosted Jenkins for builds.
You need to ensure that Jenkins can retrieve source code from Azure Repos.
Which three actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Create a webhook in Jenkins.
B. Add the Team Foundation Server (TFS) plug-in to Jenkins.
C. Add a personal access token to your Jenkins account.
D. Create a personal access token (PAT) in your Azure DevOps account.
E. Create a service hook in Azure DevOps.

Answer: BDE

Explanation:

The correct answer is BDE:

B. Add the Team Foundation Server (TFS) plug-in to Jenkins: Jenkins needs a plugin to communicate and interact with Azure DevOps. The TFS plugin (now often referred to as the Azure DevOps plugin) allows Jenkins to authenticate and access Azure Repos. Without this, Jenkins won't understand how to connect to Azure DevOps.

Reference: https://plugins.jenkins.io/azure-devops/

D. Create a personal access token (PAT) in your Azure DevOps account: A PAT acts as a secure credential for Jenkins to access Azure DevOps resources. It's like a password specifically for applications, offering more control and security than using your actual account password directly. This token will be used by Jenkins to authenticate.

E. Create a service hook in Azure DevOps: A service hook enables Azure DevOps to trigger Jenkins builds when specific events occur (e.g., code commits, pull requests). The service hook configured in Azure DevOps will send notifications to Jenkins based on defined events, automatically triggering builds when changes are pushed to the repository.

Why other options are incorrect:

A. Create a webhook in Jenkins: While Jenkins can receive webhooks, it doesn't initiate them to retrieve source code. The source code retrieval is initiated by Jenkins based on its configuration or triggered by Azure DevOps via a service hook.
C. Add a personal access token to your Jenkins account: Adding the PAT directly to the Jenkins account is not sufficient to enable the connection to Azure Repos. The PAT needs to be configured within the TFS plugin or equivalent connection settings where Jenkins specifies how to authenticate with Azure DevOps. You use the PAT within the context of the Azure DevOps plugin setup, not simply adding it to the general Jenkins account settings.

Question: 327 CertyIQ

DRAG DROP -
Your company has four projects. The version control requirements for each project are shown in the following table.

Project	Requirement
Project 1	Project leads must be able to restrict access to individual files and folders in the repository.
Project 2	The version control system must enforce the following rules on the server before merging any changes to the main branch: • Changes must be reviewed by at least two project members. • Changes must be associated by at least one work item
Project 3	The project members must be able to work in Azure Repos directly from Xcode.
Project 4	The release branch must only be viewable or editable by the project leads.

You plan to use Azure Repos for all the projects.
Which version control system should you use for each project? To answer, drag the appropriate version control systems to the correct projects. Each version control system may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:

Version Control Systems

- Git
- Perforce
- Subversion
- Team Foundation Version Control

Answer Area

- Project 1: [ ]
- Project 2: [ ]
- Project 3: [ ]
- Project 4: [ ]

Answer:

Version Control Systems

- Git
- Perforce
- Subversion
- Team Foundation Version Control

Answer Area

- Project 1: Team Foundation Version Control
- Project 2: Git
- Project 3: Subversion
- Project 4: Git

Explanation:
Box 1: Team Foundation Version Control
TFVC lets you apply granular permissions and restrict access down to a file level.

Box 2: Git -
Git is the default version control provider for new projects. You should use Git for version control in your projects unless you have a specific need for centralized version control features in TFVC.

Box 3: Subversion -
Note: Xcode is an integrated development environment (IDE) for macOS containing a suite of software development tools developed by Apple

Box 4: Git -
Note: Perforce: Due to its multitenant nature, many groups can work on versioned files. The server tracks changes in a central database of MD5 hashes of file content, along with descriptive meta data and separately retains a master repository of file versions that can be verified through the hashes.

Reference:
https://searchitoperations.techtarget.com/definition/Perforce-Software https://docs.microsoft.com/en-us/azure/devops/repos/git/share-your-code-in-git-xcode https://docs.microsoft.com/en-us/azure/devops/repos/tfvc/overview

Question: 328 CertyIQ

You are automating the build process for a Java-based application by using Azure DevOps.
You need to add code coverage testing and publish the outcomes to the pipeline.
What should you use?

A. Bullseye Coverage
B. JUnit
C. JaCoCo
D. MSTest

Answer: C

Explanation:

The correct answer is **C. JaCoCo**. Here's why:

JaCoCo (Java Code Coverage) is a free open-source toolkit specifically designed for measuring and reporting Java code coverage. It integrates seamlessly with build tools like Maven and Gradle, which are commonly used in Java application development and can be readily incorporated into Azure DevOps pipelines. This makes it a natural choice for automated code coverage testing in a Java Azure DevOps build process.

JUnit (option B) is primarily a unit testing framework for Java. While JUnit can be used alongside a code coverage tool, it doesn't inherently provide code coverage reporting itself. Therefore, JUnit alone wouldn't satisfy the requirement of publishing code coverage outcomes to the pipeline.

Bullseye Coverage (option A) is a code coverage tool, but it is primarily used for C and C++ languages, not Java. Thus, it's not applicable for the given scenario.

MSTest (option D) is Microsoft's unit testing framework primarily used for .NET languages, not Java. Therefore, it's unsuitable for this Java-based application.

To integrate JaCoCo into an Azure DevOps pipeline, you would typically add a step in your YAML or classic pipeline definition that executes the Java build process (e.g., using Maven or Gradle) along with JaCoCo. JaCoCo generates coverage reports in a format that can then be published to Azure DevOps using the "Publish Code Coverage Results" task. This task supports JaCoCo's XML report format, enabling visualization of code coverage metrics directly within the Azure DevOps pipeline results. Azure DevOps can then display coverage percentage, uncovered lines, and other relevant details.

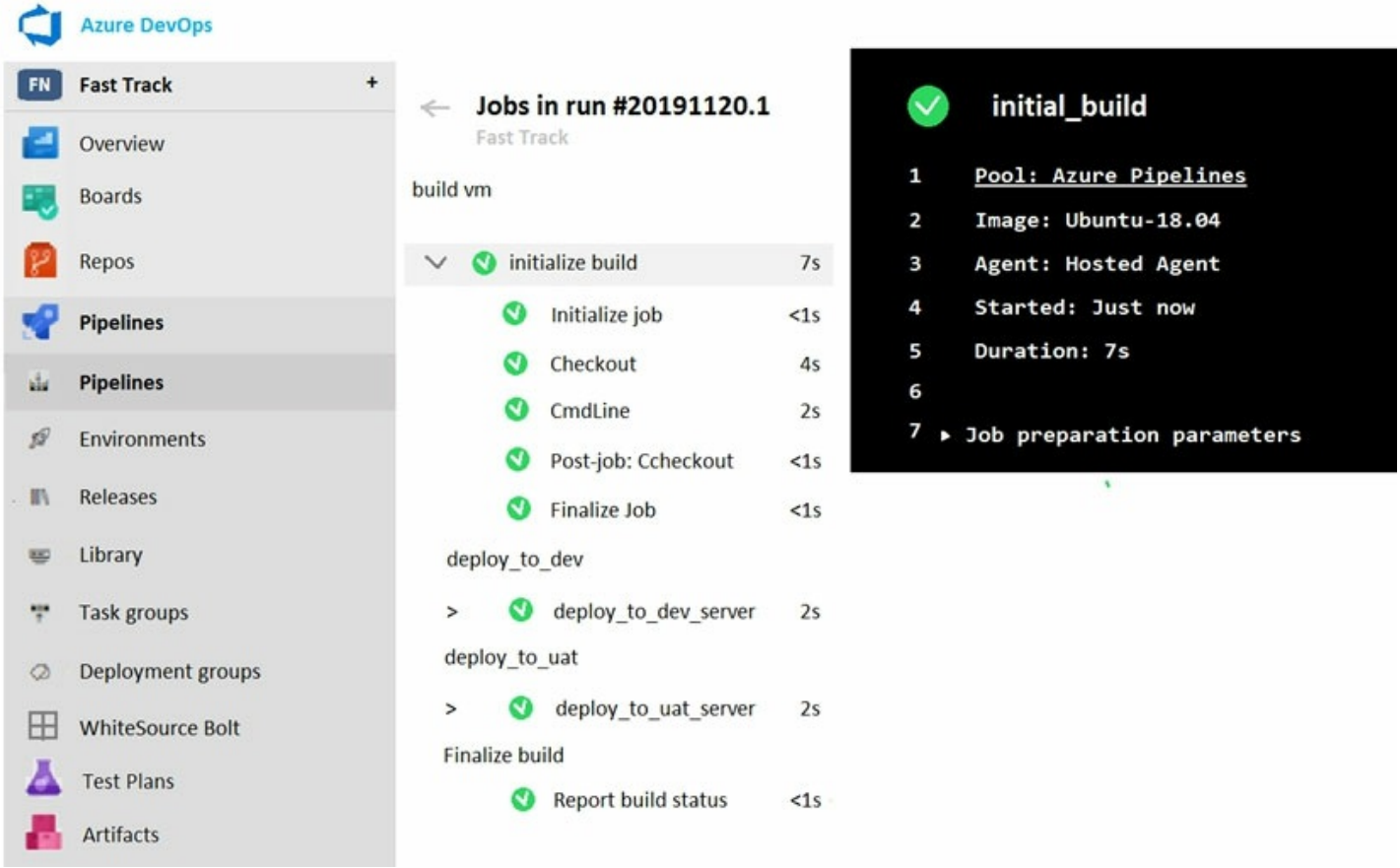
In summary, JaCoCo's suitability for Java, ease of integration with build tools, and compatibility with Azure DevOps code coverage reporting mechanisms make it the best choice for automating code coverage testing and publishing the outcomes in a Java-based Azure DevOps pipeline.

Authoritative links for further research:

JaCoCo: https://www.jacoco.org/
Azure DevOps Publish Code Coverage Results task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/test/publish-code-coverage-results?view=azure-devops

Question: 329 CertyIQ

HOTSPOT -
You company uses Azure DevOps to deploy infrastructures to Azure.
Pipelines are developed by using YAML.
You execute a pipeline and receive the results in the web portal for Azure Pipelines as shown in the following exhibit.



Use the drop-down menus to select the answer choice that completes each statement based on the information presented in the graphic.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

The pipeline contains

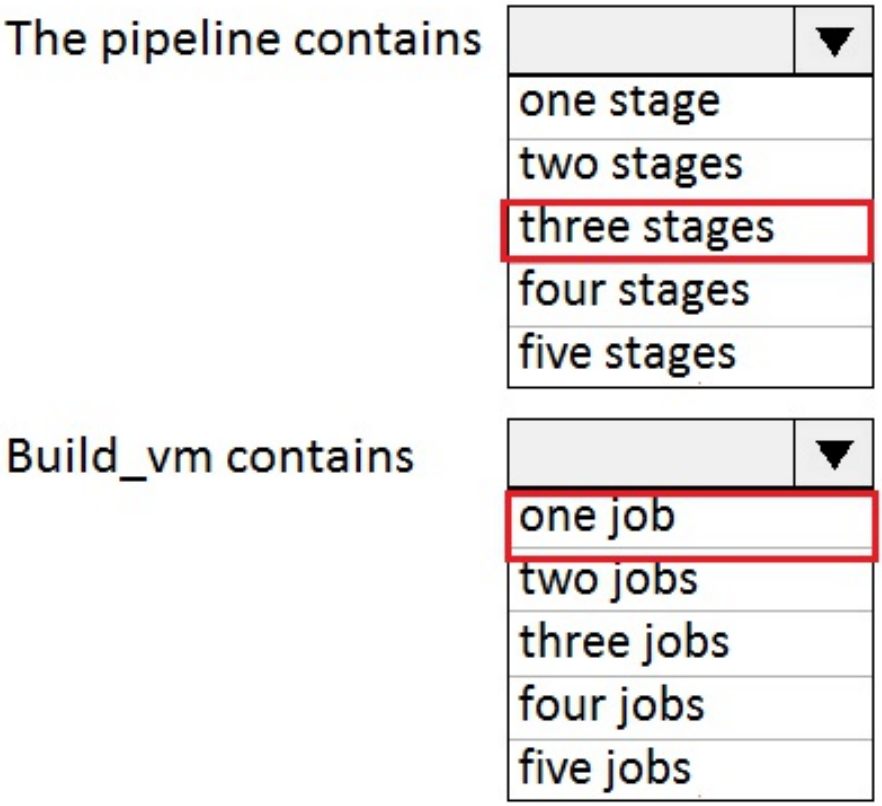
- one stage
- two stages
- three stages
- four stages
- five stages

Build_vm contains

- one job
- two jobs
- three jobs
- four jobs
- five jobs

Answer:

Answer Area



Explanation:

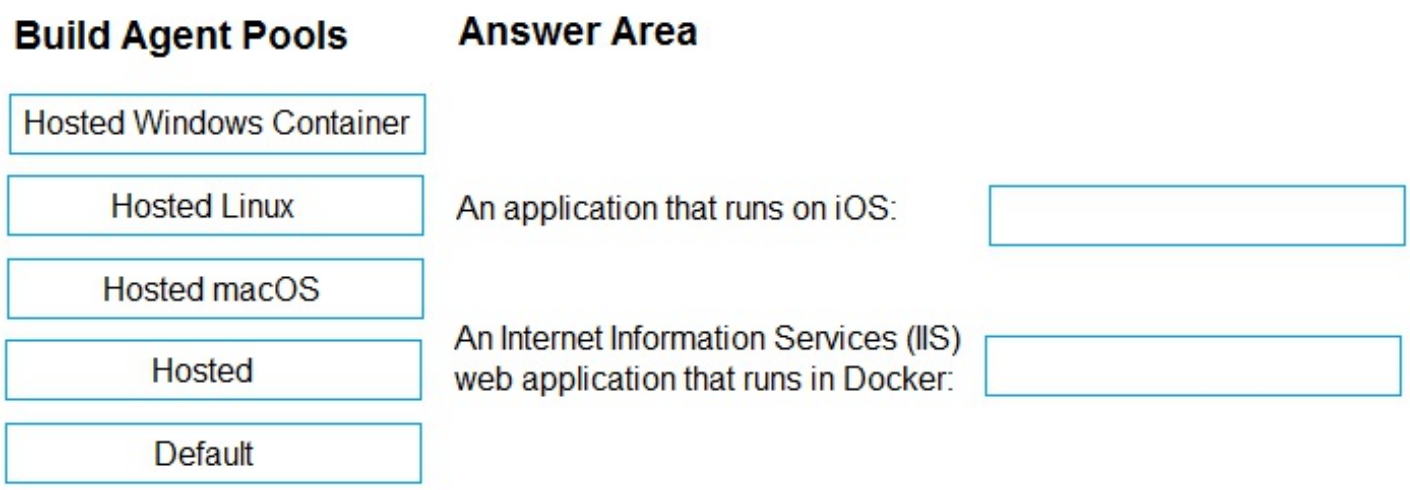
3 stages.

1 job

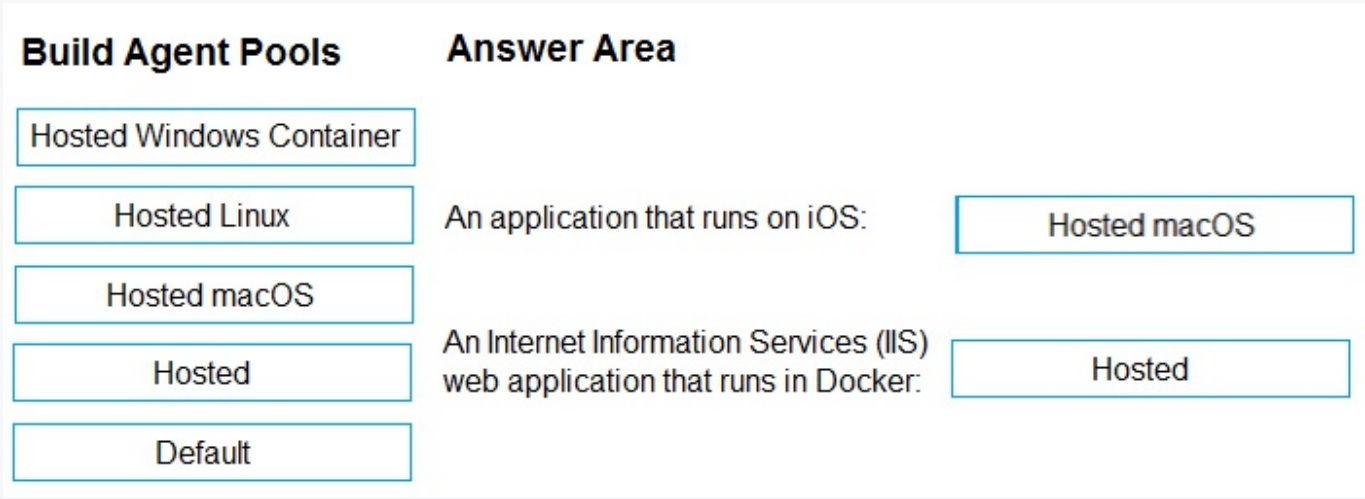
Finalize job is automatically appended to all jobs. If you could check the YAML of this job, you'd see there are only 3 tasks inside, and not 4.

Question: 330 CertyIQ

DRAG DROP -
You are configuring Azure DevOps build pipelines.
You plan to use hosted build agents.
Which build agent pool should you use to compile each application type? To answer, drag the appropriate build agent pools to the correct application types. Each build agent pool may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:
Box 1: Hosted macOS -
Hosted macOS pool (Azure Pipelines only): Enables you to build and release on macOS without having to configure a self-hosted macOS agent. This option affects where your data is stored.

Box 2: Hosted -
Hosted pool (Azure Pipelines only): The Hosted pool is the built-in pool that is a collection of Microsoft-hosted agents.
Incorrect Answers:
Default pool: Use it to register self-hosted agents that you've set up.
Hosted Windows Container pool (Azure Pipelines only): Enabled you to build and release inside Windows containers. Unless you're building using containers,
Windows builds should run in the Hosted VS2017 or Hosted pools.
Hosted Linux/Ubuntu 18.04 does not apply for Mac OS or for Microsoft IIS.

Reference:
https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/agents

Question: 331 CertyIQ

You are automating the build process for a Java-based application by using Azure DevOps.
You need to add code coverage testing and publish the outcomes to the pipeline.
What should you use?

A. Cobertura
B. Bullseye Coverage
C. MSTest
D. Coverlet

Answer: A

Explanation:

The correct answer is A, Cobertura, because it is a widely supported and well-integrated code coverage tool commonly used with Java projects within Azure DevOps pipelines. Cobertura instruments Java bytecode to collect data on line, branch, and method coverage. This data is then aggregated into reports that highlight areas of the code base that haven't been adequately tested. The Azure DevOps pipeline can be configured to execute Cobertura as part of the build process, generate coverage reports, and publish these reports to the pipeline results, providing valuable insights into the quality and completeness of the tests. Azure DevOps offers native support for visualizing Cobertura results.

Bullseye Coverage is a commercial code coverage tool, more often employed in C/C++ environments, and while potentially usable, it is not the readily available, preferred, and most cost-effective solution for Java in Azure DevOps compared to Cobertura. MSTest is primarily associated with .NET/C# development, and while Azure DevOps supports it for .NET projects, it's not applicable to a Java application. Coverlet is a cross-platform code coverage framework for .NET.

Cobertura is an open-source tool; its integration with Azure DevOps is straightforward, making it the optimal solution for the described scenario. The generated reports can be easily integrated into the build pipeline and visualized within Azure DevOps, allowing developers to quickly identify and address gaps in their test coverage. The other options either are not suitable for Java (MSTest, Coverlet) or are more complex (Bullseye).

Use Publish Code Coverage Results task in a build pipeline to publish code coverage results to Azure Pipelines or TFS, which were produced by a build in

Cobertura or JaCoCo format.

Incorrect Answers:

B: Bullseye Coverage is used for C++ code, and not for Java.

D: If you're building on Linux or macOS, you can use Coverlet or a similar tool to collect code coverage metrics. Code coverage results can be published to the server by using the Publish Code Coverage Results task. To leverage this functionality, the coverage tool must be configured to generate results in Cobertura or

JaCoCo coverage format.

F: Coverage.py is used for Python, not for Java.

Note:

There are several versions of this question in the exam. The question has two possible correct answers:

1. Cobertura

2. JaCoCo

Other incorrect answer options you may see on the exam include the following:

1. Junit

2. NUnit

3. Coverage.py

Reference:

https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/test/publish-code-coverage-results

Question: 332 CertyIQ

You have an existing build pipeline in Azure Pipelines.
You need to use incremental builds without purging the environment between pipeline executions.
What should you use?

A. a self-hosted agent
B. Microsoft-hosted parallel jobs
C. a File Transform task

Answer: A

Explanation:

The correct answer is **A. a self-hosted agent**.

Here's why:

Incremental builds aim to optimize the build process by reusing previously built components and only rebuilding what has changed. To achieve this effectively without purging the environment between pipeline executions, persistent storage and a controlled environment are crucial.

Self-hosted agents offer the necessary control and persistence. Unlike Microsoft-hosted agents, which are ephemeral (newly provisioned for each job and then discarded), self-hosted agents are persistent machines that retain their state between builds. This allows for caching dependencies, intermediate build artifacts, and other elements needed for incremental builds. This persistent environment allows the build pipeline to leverage existing components and perform incremental updates, significantly reducing build times.

Microsoft-hosted parallel jobs (option B) offer parallelization but don't address the core requirement of environment persistence. They are still ephemeral and won't retain the state between jobs. A File Transform task (option C) is used to modify configuration files during the deployment process and is not directly related to incremental builds.

Therefore, using a self-hosted agent is the best way to achieve incremental builds without purging the environment, as it provides the necessary persistence and control.

Supporting documentation:

Azure Pipelines Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=yaml&viewFallbackFrom=vsts
Self-hosted agents vs. Microsoft-hosted agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/pools-queues?view=azure-devops&tabs=yaml&viewFallbackFrom=vsts

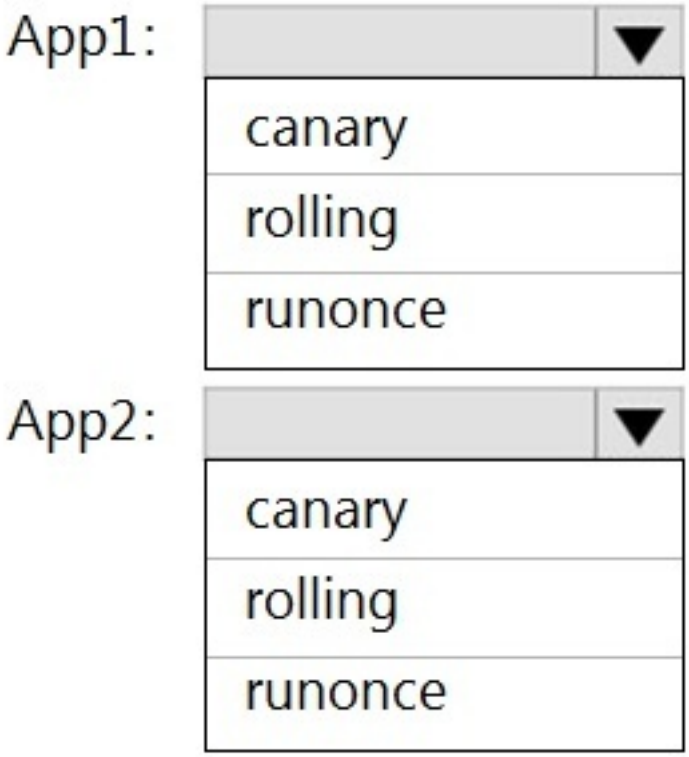
Question: 333 CertyIQ

HOTSPOT -
You are designing YAML-based Azure pipelines for the apps shown in the following table.

Name	Platform	Release requirements
App1	Azure virtual machine	Replace a fixed set of existing instances of the previous version of App1 with instances of the new version of the app in each iteration.
App2	Azure Kubernetes Service (AKS) cluster	Roll out a limited deployment of the new version of App2 to validate the functionality of the app. Once testing is successful, expand the rollout.

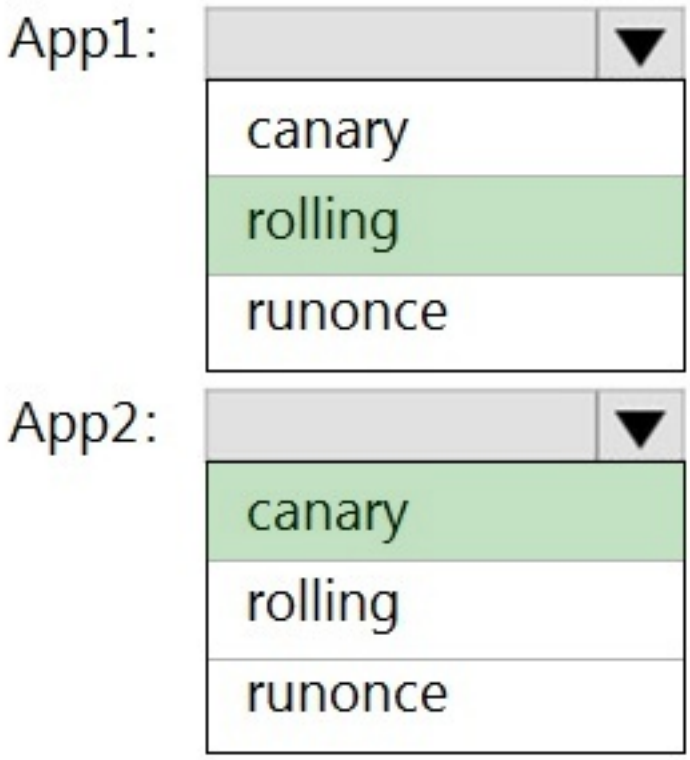
You need to configure the YAML strategy value for each app. The solution must minimize app downtime.
Which value should you configure for each app? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:
App1: rolling -
A rolling deployment replaces instances of the previous version of an application with instances of the new version of the application on a fixed set of virtual machines (rolling set) in each iteration.

App2: canary -
Canary deployment strategy is an advanced deployment strategy that helps mitigate the risk involved in rolling out new versions of applications. By using this strategy, you can roll out the changes to a small subset of servers first. As you gain more confidence in the new version, you can release it to more servers in your infrastructure and route more traffic to it.
Incorrect Answers:
runonce:
runOnce is the simplest deployment strategy wherein all the lifecycle hooks, namely preDeploy deploy, routeTraffic, and postRouteTraffic, are executed once.
Then, either on: success or on: failure is executed.

Reference:
https://docs.microsoft.com/en-us/azure/devops/pipelines/process/deployment-jobs

Question: 334 CertyIQ

You have a private project in Azure DevOps.
You need to ensure that a project manager can create custom work item queries to report on the project's progress. The solution must use the principle of least privilege.
To which security group should you add the project manager?

A. Reader
B. Project Collection Administrators
C. Project Administrators

D. Contributor

Answer: D

Explanation:

The correct answer is **D. Contributor**.

Here's why:

The requirement is to grant the project manager the ability to create custom work item queries within the Azure DevOps project, adhering to the principle of least privilege. This means providing only the necessary permissions and no more.

Contributor: This group provides the permissions needed to contribute to the project, which includes creating and modifying work items, and critically, creating and managing work item queries. This aligns directly with the requirement.

Reader: This group only allows read-only access to the project. Members can view work items and other project data, but cannot create or modify anything, including queries.

Project Administrators: This group has elevated permissions to manage the project settings, users, and security. While they can create queries, granting this level of access violates the principle of least privilege. It provides the project manager with far more control than necessary for simply creating queries.

Project Collection Administrators: This group has the highest level of administrative privileges, spanning across the entire Azure DevOps organization or collection of projects. Granting this level of access for creating queries would be a serious security risk and completely disregards the principle of least privilege.

Therefore, the **Contributor** group provides the necessary permissions to create custom work item queries without granting unnecessary administrative privileges. This is the most appropriate choice based on the requirements.

Further research on Azure DevOps security groups can be done here:

Default groups and permissions - Azure DevOps
Manage Permissions with Groups in Azure DevOps

Question: 335 CertyIQ

Your company has a project in Azure DevOps for a new application. The application will be deployed to several Azure virtual machines that run Windows Server
2019.
You need to recommend a deployment strategy for the virtual machines. The strategy must meet the following requirements:
✑ Ensure that the virtual machines maintain a consistent configuration.
✑ Minimize administrative effort to configure the virtual machines.
What should you include in the recommendation?

A. Azure Resource Manager templates and the PowerShell Desired State Configuration (DSC) extension for Windows

B. Deployment YAML and Azure pipeline deployment groups

C. Azure Resource Manager templates and the Custom Script Extension for Windows

D. Deployment YAML and Azure pipeline stage templates

Answer: A

Explanation:

The correct answer is A: Azure Resource Manager (ARM) templates and the PowerShell Desired State Configuration (DSC) extension for Windows. Here's why:

ARM templates provide infrastructure-as-code, allowing you to define the desired state of your Azure resources (virtual machines, networking, etc.) in a declarative manner. This ensures consistency across deployments and environments. By using ARM templates, you can consistently create and configure the VMs for your application.

PowerShell DSC is a configuration management system built into Windows. It allows you to define the desired configuration of your Windows Server 2019 virtual machines. The DSC extension for Azure VM's automates the application of DSC configurations to your virtual machines as they are provisioned. This ensures that all VMs maintain the same configuration, meeting the first requirement. DSC drastically reduces administrative overhead since configurations are centrally defined and automatically applied and maintained. Changes to the configuration can be easily propagated across all VMs managed by DSC. This addresses the second requirement.

Options B, C, and D are less ideal. Deployment YAML is often used in Azure Pipelines for orchestrating deployments, but alone it doesn't guarantee a consistent configuration state for the VMs. While deployment groups (option B) can help manage deployments to VMs, they don't directly enforce a consistent configuration across the VMs themselves. The Custom Script Extension (option C) can execute scripts on VMs after provisioning, but it's less robust for maintaining a desired state compared to DSC. The Custom Script Extension is also imperative, meaning that it relies on sequential commands for configuration, whereas DSC is declarative, focusing on the desired end-state, not the steps to get there. Imperative approaches tend to be harder to maintain consistent configurations in the long term. Stage templates (option D) are useful for re-using pipeline logic, but don't directly manage the desired state of the underlying VMs.

Therefore, the combination of ARM templates for infrastructure provisioning and DSC for configuration management best addresses the requirements of maintaining consistent configurations with minimal administrative effort.

Authoritative Links:

Azure Resource Manager templates: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/overview
PowerShell Desired State Configuration: https://learn.microsoft.com/en-us/powershell/dsc/overview
DSC Extension for Azure: https://learn.microsoft.com/en-us/azure/virtual-machines/extensions/dsc-windows

Question: 336 CertyIQ

You have an Azure DevOps project that uses many package feeds.
You need to simplify the project by using a single feed that stores packages produced by your company and packages consumed from remote feeds. The solution must support public feeds and authenticated feeds.
What should you enable in DevOps?

A. Universal Packages
B. upstream sources
C. views in Azure Artifacts
D. a symbol server

Answer: B

Explanation:

The correct answer is **B. Upstream sources.**

Here's a detailed justification:

Upstream sources in Azure Artifacts allow you to integrate multiple feeds (both internal and public) into a single feed. This addresses the requirement of consolidating many feeds into one for simplification.

When upstream sources are enabled, your single feed can pull packages from various sources, including public feeds like NuGet.org and npmjs.com, as well as authenticated feeds like Azure Artifacts feeds belonging to other organizations or projects.

The key benefit of using upstream sources is that when a package is requested from the consolidated feed, Azure Artifacts first checks if the package is already present in the feed. If not, it will automatically pull the package from the configured upstream source, save a copy in the feed (if configured to do so), and then serve it to the user. This ensures that packages are cached and readily available for future use, reducing reliance on external sources and improving build times.

Views in Azure Artifacts (option C) are primarily used for version control and promoting specific package versions through different stages (e.g., development, testing, production). While views are useful for managing package promotion, they don't inherently consolidate multiple feeds into one.

Universal Packages (option A) are a type of package format supported by Azure Artifacts. While they can be stored in a feed, they don't provide the mechanism for aggregating packages from multiple upstream feeds.

A symbol server (option D) is for debugging and relates to providing debug symbols along with packages, and this does not address the requirement of aggregating feeds.

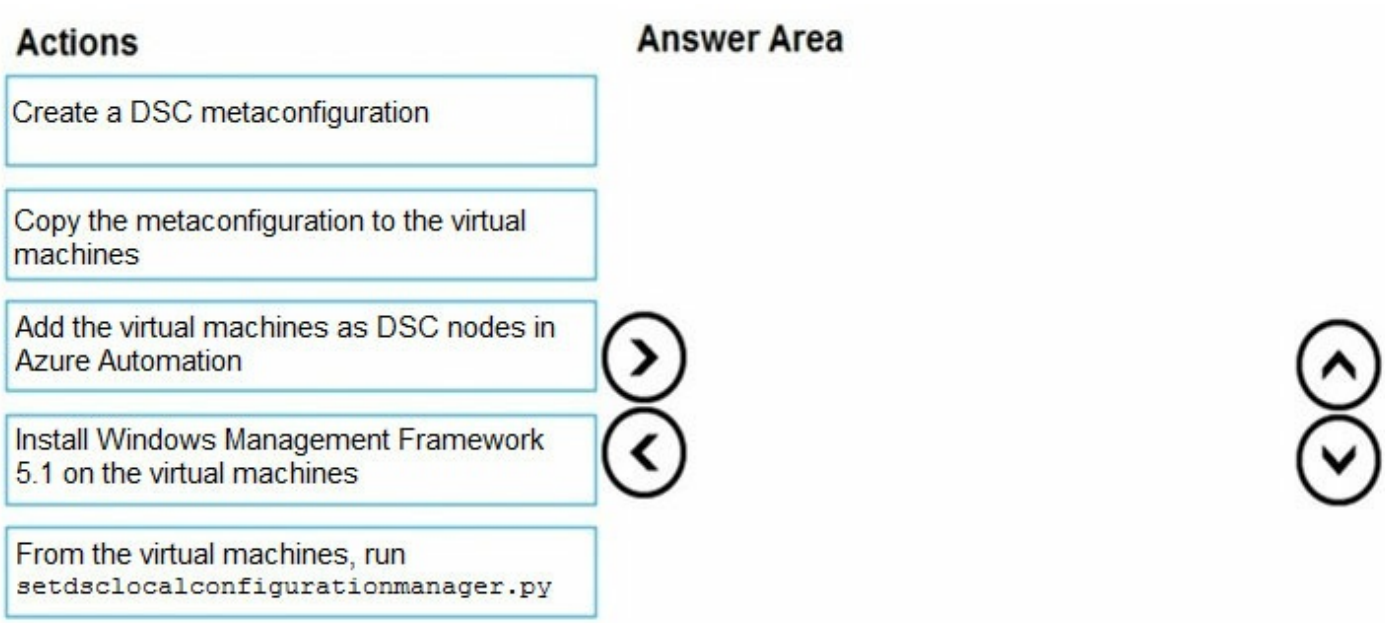
Therefore, upstream sources are the correct mechanism for creating a single feed that can consume packages from both public and authenticated feeds, simplifying the package management process.

Supporting links:

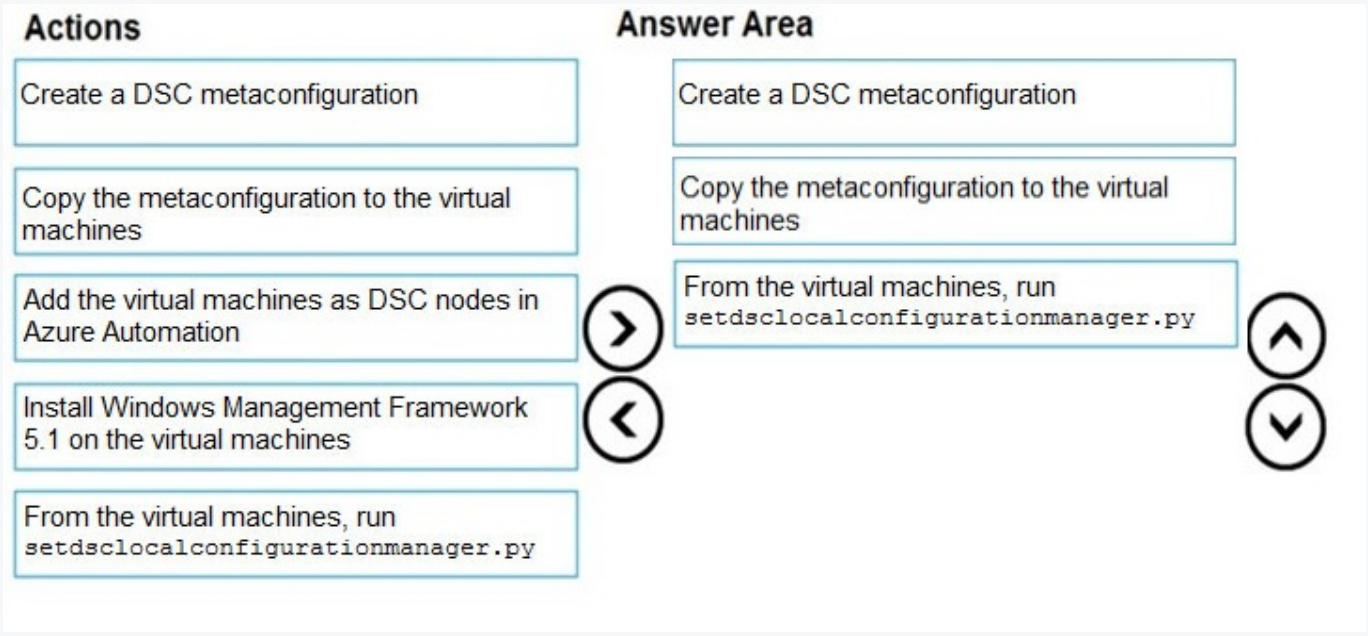
Use upstream sources - Azure Artifacts | Microsoft Learn
Add packages from public registries using upstream sources - Azure Artifacts | Microsoft Learn

Question: 337 CertyIQ

DRAG DROP -
Your company has two virtual machines that run Linux in a third-party public cloud.
You plan to use the company's Azure Automation State Configuration implementation to manage the two virtual machines and detect configuration drift.
You need to onboard the Linux virtual machines.
You install PowerShell Desired State Configuration (DSC) on the virtual machines, and then run register.py.
Which three actions should you perform next in sequence? To answer, move the actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Step 1: Create a DSC metaconfiguration.

Load up the DSC Configuration into Azure Automation.

Step 2: Copy the metaconfiguration to the virtual machines.

Linking the Node Configuration to the Linux Host

Step 3: From the virtual machines, run setdsclocalconfigurationmanager.py

Reference:

https://docs.microsoft.com/en-us/azure/automation/automation-dsc-onboarding#enable-physicalvirtual-linux-machines

Question: 338

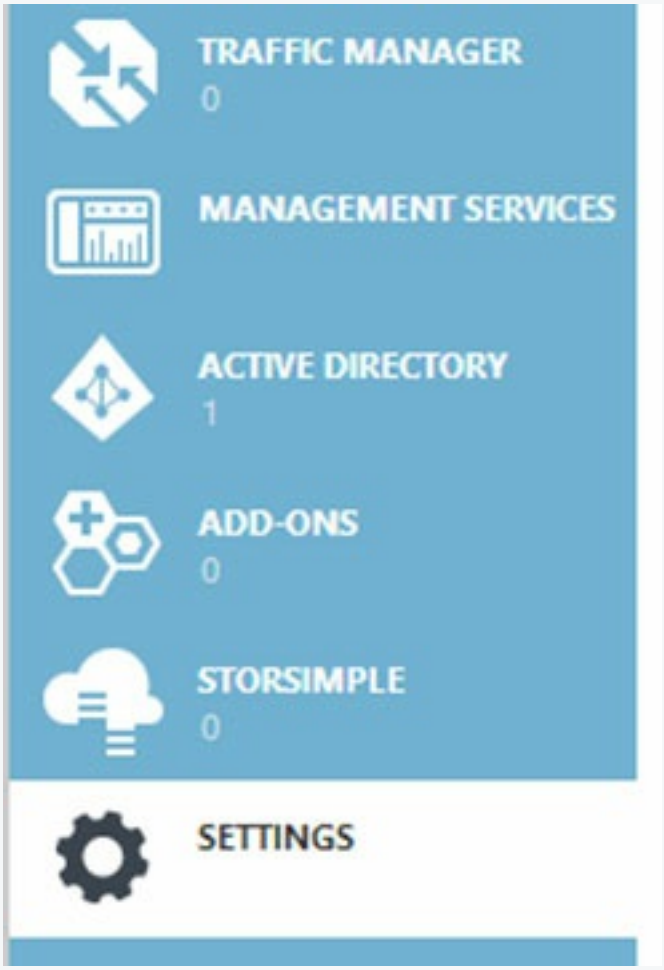
CertyIQ

SIMULATION -
You plan to deploy a runbook that will create Azure AD user accounts.
You need to ensure that runbooks can run the Azure PowerShell cmdlets for Azure Active Directory.
To complete this task, sign in to the Microsoft Azure portal.

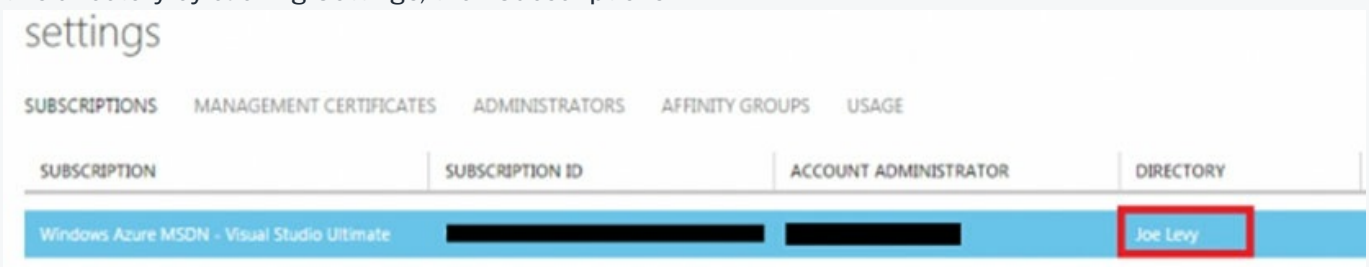
Answer:

See explanation below.

Explanation:
Azure Automation now ships with the Azure PowerShell module of version 0.8.6, which introduced the ability to non-interactively authenticate to Azure using OrgId
(Azure Active Directory user) credential-based authentication. Using the steps below, you can set up Azure Automation to talk to Azure using this authentication type.
Step 1: Find the Azure Active Directory associated with the Azure subscription to manage:
1. Log in to the Azure portal as the service administrator for the Azure subscription you want to manage using Azure Automation. You can find this user by logging in to the Azure portal as any user with access to this Azure subscription, then clicking Settings, then Administrators.



2. Note the name of the directory associated with the Azure subscription you want to manage. You can find this directory by clicking Settings, then Subscriptions.



Step 2: Create an Azure Active Directory user in the directory associated with the Azure subscription to manage:
You can skip this step if you already have an Azure Active Directory user in this directory. and plan to use this OrgId to manage Azure.

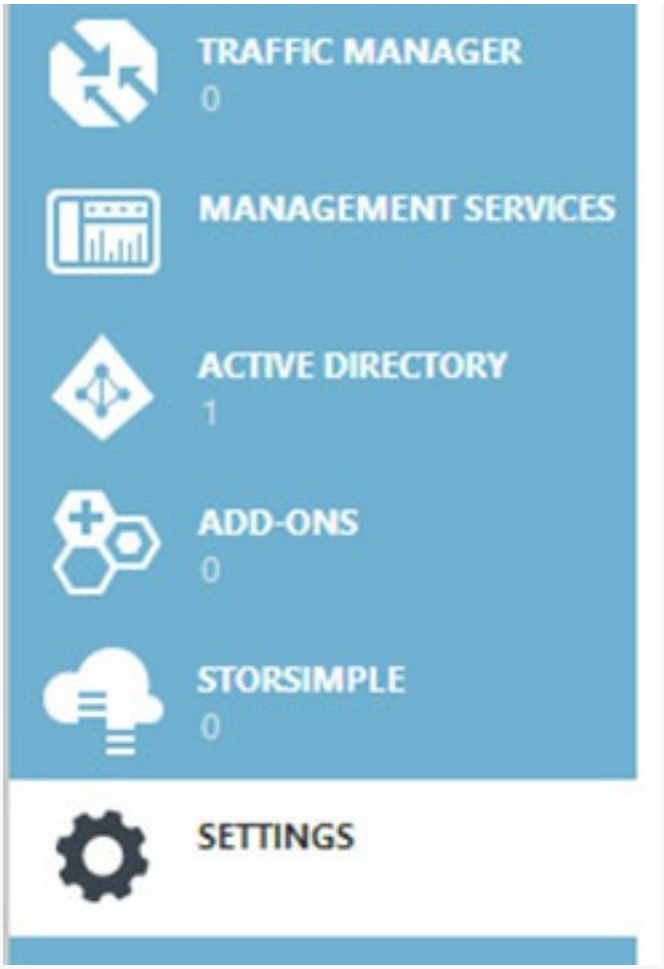
1. In the Azure portal click on Active Directory service.



2. Click the directory name that is associated with this Azure subscription.
3. Click on the Users tab and then click the Add User button.
4. For type of user, select New user in your organization. Enter a username for the user to create.
5. Fill out the user's profile. For role, pick User. Don't enable multi-factor authentication. Multi-factor accounts cannot be used with Azure Automation.
6. Click Create.
7. Jot down the full username (including part after @ symbol) and temporary password.

Step 3: Allow this Azure Active Directory user to manage this Azure subscription.

1. Click on Settings (bottom Azure tab under StorSimple)

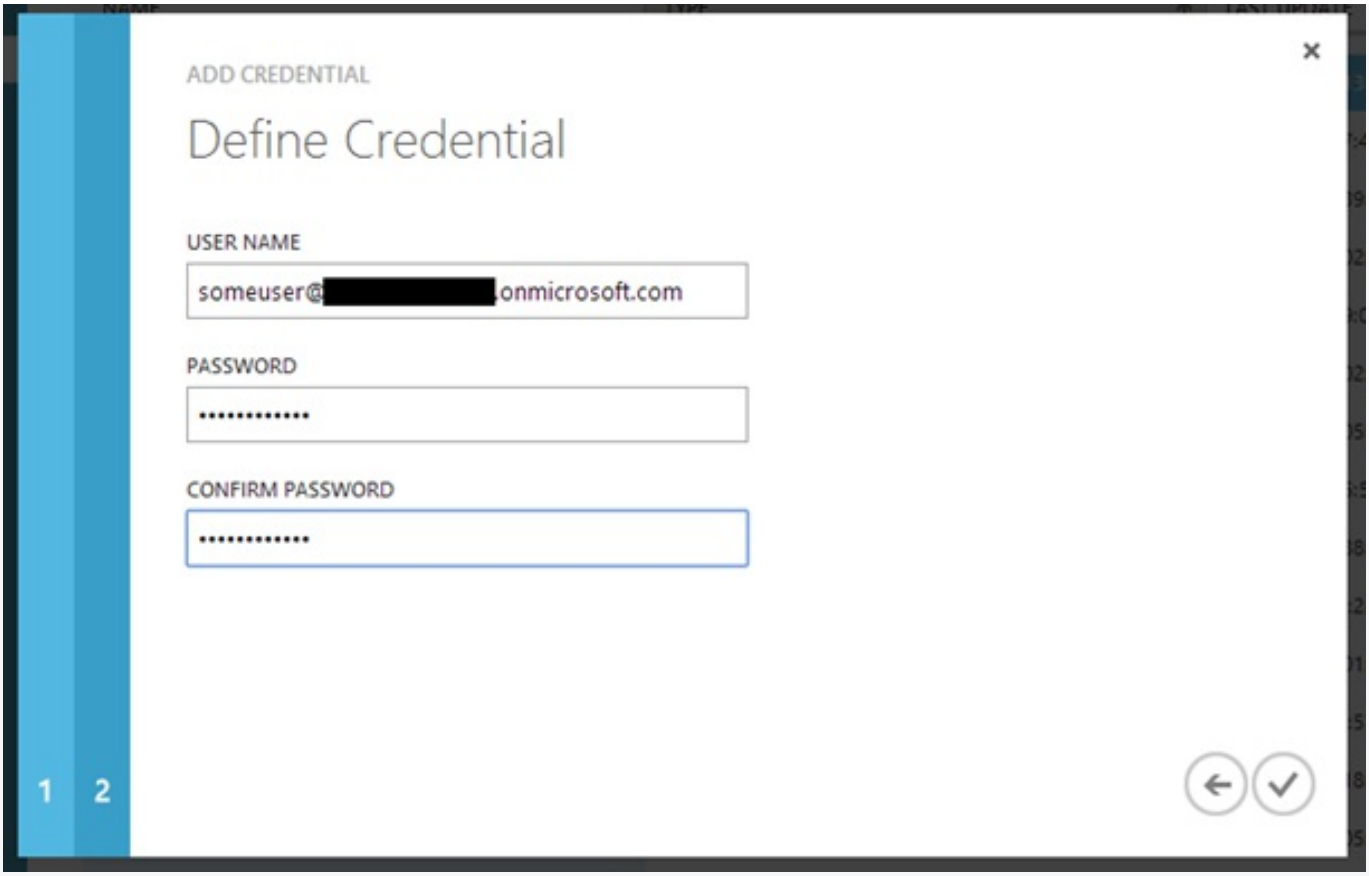


2. Click Administrators

3. Click the Add button. Type the full user name (including part after @ symbol) of the Azure Active Directory user you want to set up to manage Azure. For subscriptions, choose the Azure subscriptions you want this user to be able to manage. Click the check mark.

Step 4: Configure Azure Automation to use this Azure Active Directory user to manage this Azure subscription

Create an Azure Automation credential asset containing the username and password of the Azure Active Directory user that you have just created. You can create a credential asset in Azure Automation by clicking into an Automation Account and then clicking the Assets tab, then the Add Setting button.



Note: Once you have set up the Azure Active Directory credential in Azure and Azure Automation, you can now manage Azure from Azure Automation runbooks using this credential.

Reference:
https://azure.microsoft.com/sv-se/blog/azure-automation-authenticating-to-azure-using-azure-active-directory/

Question: 339 CertyIQ

DRAG DROP -
You are creating a container for an ASP.NET Core app.
You need to create a Dockerfile file to build the image. The solution must ensure that the size of the image is minimized.
How should you configure the file? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:

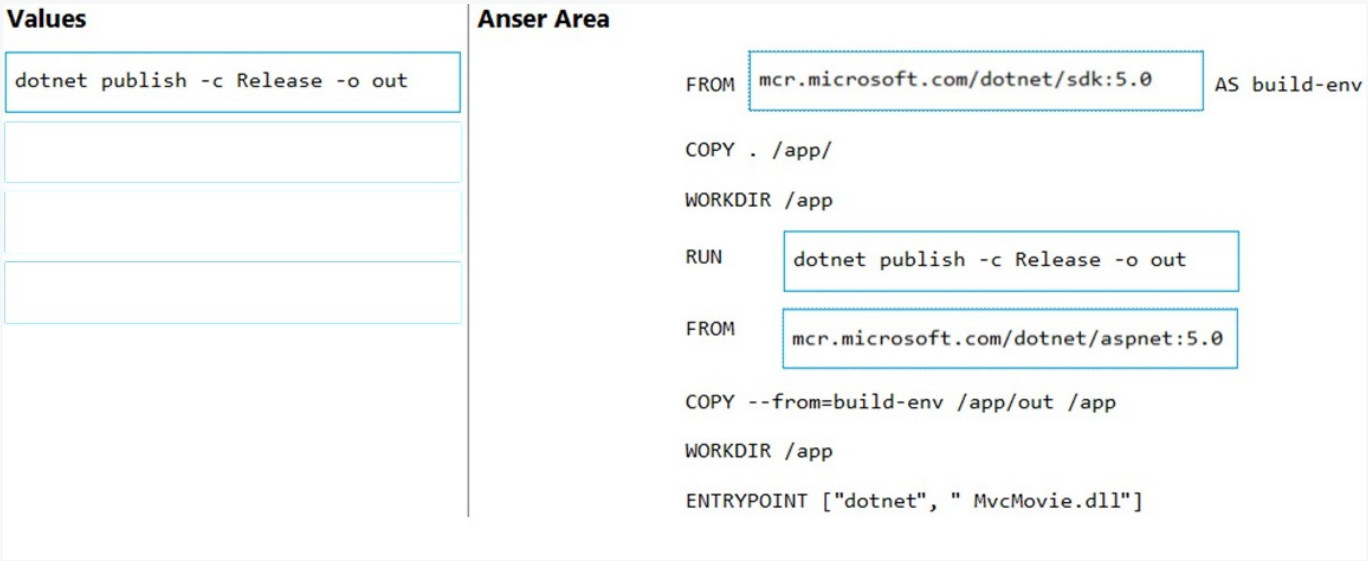
Values

- `dotnet publish -c Release -o out`
- `dotnet restore`
- `mcr.microsoft.com/dotnet/aspnet:5.0`
- `mcr.microsoft.com/dotnet/sdk:5.0`

Anser Area

```
FROM [Value] AS build-env
COPY . /app/
WORKDIR /app
RUN [Value]
FROM [Value]
COPY --from=build-env /app/out /app
WORKDIR /app
ENTRYPOINT ["dotnet", " MvcMovie.dll"]
```

Answer:



Explanation:

Box 1: mcr.microsoft.com/dotnet/sdk:5.0

The first group of lines declares from which base image we will use to build our container on top of. If the local system does not have this image already, then docker will automatically try and fetch it. The mcr.microsoft.com/dotnet/core/sdk:5.0 comes packaged with the .NET core 5.0 SDK installed, so it's up to the task of building ASP .NET core projects targeting version 5.0

Box 2: d dotnet publish -c Release -o out

Box 3: mcr.microsoft.com/dotnet/aspnet:5.0

When building container images, it's good practice to include only the production payload and its dependencies in the container image. We don't want the .NET core SDK included in our final image because we only need the .NET core runtime, so the dockerfile is written to use a temporary container that is packaged with the SDK called build-env to build the app.

Question: 340 CertyIQ

DRAG DROP -
You are configuring the settings of a new Git repository in Azure Repos.
You need to ensure that pull requests in a branch meet the following criteria before they are merged:
- Committed code must compile successfully.
- Pull requests must have a Quality Gate status of Passed in SonarCloud.

Which policy type should you configure for each requirement? To answer, drag the appropriate policy types to the correct requirements. Each policy type may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:

Answer Area

Policy Types

- A build policy
- A check-in policy
- A status policy

Committed code must compile successfully: [ ]

Pull requests must have a Quality Gate status of Passed in SonarCloud: [ ]

Answer:

Answer Area

Policy Types

- A build policy
- A check-in policy
- A status policy

Committed code must compile successfully: A build policy

Pull requests must have a Quality Gate status of Passed in SonarCloud: A status policy

Explanation:

A build policy.

A status policy.

Reference:

https://azuredevopslabs.com/labs/vstsextend/sonarcloud/

https://docs.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops#build-validation

Question: 341 CertyIQ

You use a Git repository in Azure Repos to manage the source code of a web application. Developers commit changes directly to the default branch.
You need to implement a change management procedure that meets the following requirements:
- The default branch must be protected, and new changes must be built in the feature branches first.
- Changes must be reviewed and approved by at least one release manager before each merge.
- Changes must be brought into the default branch by using pull requests.

What should you configure in Azure Repos?

A. branch policies of the default branch
B. Services in Project Settings
C. Deployment pools in Project Settings
D. branch security of the default branch

Answer: A

Explanation:

The correct answer is A, configuring branch policies of the default branch in Azure Repos. Branch policies are designed to protect crucial branches like the default branch from direct, unreviewed commits. By enabling branch policies, you can enforce requirements before changes are merged into the protected branch.

Specifically, configuring policies like requiring pull requests, a minimum number of reviewers, and successful builds enforces the needed change management procedure. Requiring pull requests ensures all changes undergo a review process before integration. Mandating a minimum number of reviewers guarantees at least one release manager approves the changes, fulfilling the requirement. Integrating build validation ensures that new changes don't break the application's build process, preventing problematic code from entering the default branch. These configurations collectively prevent direct commits, enforce reviews, and guarantee code quality before merging, thus satisfying all requirements stated.

Option B, project settings services, is incorrect because while services are important for continuous integration and delivery, they don't directly enforce change management policies on a specific branch. Option C, deployment pools, are used for deployment targets and are not directly related to source code change management. Option D, branch security, controls access to the branch but does not implement the workflow requirements such as pull requests and reviews. Branch policies are the most suitable option for enforcing workflow constraints on code changes.

Further reading:

Microsoft Azure DevOps Branch Policies: https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops
Using Pull Requests: https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops

Question: 342 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company uses Azure DevOps to manage the build and release processes for applications.
You use a Git repository for applications source control.
You need to implement a pull request strategy that reduces the history volume in the master branch.
Solution: You implement a pull request strategy that uses fast-forward merges.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of using fast-forward merges in a pull request strategy does not meet the goal of reducing the history volume in the master branch.

Fast-forward merges simply move the master branch pointer to the latest commit on the feature branch being merged, provided the master branch hasn't advanced since the feature branch was created. While this does create a linear history, it includes all the commits from the feature branch in the master branch history. The history is not reduced, rather it is kept intact and appended to.

To reduce history volume, techniques like squash merges or rebase merges are more appropriate. Squash merges combine all the commits from a feature branch into a single commit on the master branch, effectively summarizing the changes and reducing the number of individual commits in the master branch history. Rebasing, similarly, rewrites the history of the feature branch before merging it into master, often resulting in a cleaner, linear history with fewer merge commits (although rebasing needs to be done with caution to avoid rewriting public history).

Therefore, using fast-forward merges will maintain the entire commit history of the feature branch on the master branch, failing to reduce the history volume.https://www.atlassian.com/git/tutorials/merging-vs-rebasinghttps://git-scm.com/docs/git-merge

Question: 343 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company uses Azure DevOps to manage the build and release processes for applications.
You use a Git repository for applications source control.
You need to implement a pull request strategy that reduces the history volume in the master branch.
Solution: You implement a pull request strategy that uses squash merges.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

Yes, the solution meets the goal. Using squash merges in pull requests reduces the history volume in the master branch by combining all the commits from the feature branch into a single commit on the master branch. This creates a cleaner, linear history, making it easier to understand and manage. Instead of a detailed branching history, you only see a series of single commits representing merged features or bug fixes. This prevents the master branch history from becoming cluttered with intermediate commits made during development in the feature branch. Squashing commits also makes it easier to revert changes if necessary, as you only need to revert a single commit rather than multiple commits. This approach simplifies the history, which is beneficial for auditability and maintainability. Squash merging is a widely adopted practice in software development using Git and Azure DevOps for simplifying commit histories.

Here are some resources for further research:

Azure DevOps Pull Requests: https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops
Git Merge Strategies: https://www.atlassian.com/git/tutorials/merging-vs-rebasing
Squash Merging: https://www.gitkraken.com/learn/git/git-squash

Question: 344 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company uses Azure DevOps to manage the build and release processes for applications.
You use a Git repository for applications source control.
You need to implement a pull request strategy that reduces the history volume in the master branch.
Solution: You implement a pull request strategy that uses an explicit merge.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of using an explicit merge strategy in pull requests does not inherently reduce the history volume in the master branch. An explicit merge, typically a standard "merge commit," combines the changes from the feature branch into the target branch (master). This creates a merge commit, which adds to the commit history.

While the explicit merge provides a clear record of when and why the branches were merged, it contributes to the overall history length. Strategies to reduce the history volume, such as rebasing, squashing, or using shallow clones during builds, are more appropriate for minimizing commit history. Rebasing replays commits, effectively rewriting history. Squashing combines multiple commits into a single, more meaningful commit, which results in a cleaner history. Shallow clones limit the depth of the Git history downloaded, helping to improve build times but not altering the central repository.

Therefore, the explicit merge strategy, on its own, does not meet the goal of reducing the history volume in the master branch.

Relevant Documentation:

Using pull requests to review code: https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests?view=azure-devops
Squash merge pull requests: https://learn.microsoft.com/en-us/azure/devops/repos/git/merging-with-squash-merge?view=azure-devops

Question: 345 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company uses Azure DevOps to manage the build and release processes for applications.
You use a Git repository for applications source control.
You need to implement a pull request strategy that reduces the history volume in the master branch.
Solution: You implement a pull request strategy that uses a three-way merge.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of using a three-way merge strategy in pull requests does not directly reduce the history volume in the master branch. A three-way merge simply integrates changes from the feature branch into the target branch (master in this case) by creating a merge commit. This merge commit preserves the history of both the feature branch and the master branch.

To reduce the history volume in the master branch, you should use strategies like squash merging or rebase merging. Squash merging collapses the entire feature branch's history into a single commit when merging to master. This results in a cleaner and more linear master branch history because it avoids keeping the numerous commits made during development on the feature branch. Rebase merging, while more complex, essentially replays the feature branch commits on top of the latest master, resulting in a linear history.

Three-way merging is a fundamental Git merge strategy, but it doesn't inherently address the problem of excessive history in the target branch. It ensures that changes from both branches are integrated correctly, but it preserves all the commits from the feature branch in the master branch's history.

Therefore, the given solution does not meet the goal.

Relevant links for further research:

Understanding the GitHub Flow
Azure DevOps pull request settings and policies
Git merge strategies
Squash merging on GitHub

Question: 346 CertyIQ

You need to recommend a Docker container build strategy that meets the following requirements:
✑ Minimizes image sizes
✑ Minimizes the security surface area of the final image
What should you include in the recommendation?

A. multi-stage builds
B. PowerShell Desired State Configuration (DSC)
C. Docker Swarm
D. single-stage builds

Answer: A

Explanation:

The correct answer is A, multi-stage builds. Here's a detailed justification:

Multi-stage builds in Docker directly address the stated requirements of minimizing image sizes and reducing the security surface area. This approach leverages multiple FROM statements within a single Dockerfile, allowing you to use different base images for different stages of the build process.

The initial stages can use larger images containing all the necessary tools for compiling, building, and testing the application. These intermediary stages are then used to produce the final artifact (e.g., a compiled binary, a static website). The final stage then copies only the artifact from a prior stage into a much smaller, leaner base image (often a distroless image or a very minimal distribution like Alpine Linux).

By copying only the necessary artifacts into the final stage, you avoid including build tools, dependencies, and potentially vulnerable software that were used only during the build process. This significantly reduces the final image size, leading to faster deployment times, reduced storage costs, and lower bandwidth usage. Critically, it also minimizes the attack surface. Fewer tools and dependencies in the final image mean fewer

potential vulnerabilities that an attacker could exploit.

Single-stage builds (option D) would include all build dependencies and tools within the final image, thus increasing its size and security vulnerabilities. PowerShell DSC (option B) is a configuration management tool and while useful for configuring systems, it doesn't inherently minimize Docker image sizes or security surfaces during image creation. Docker Swarm (option C) is a container orchestration tool and doesn't directly impact the image build process or its size/security. It focuses on managing and deploying containers across a cluster.

Multi-stage builds are considered a best practice for creating optimized Docker images in cloud environments and are directly related to minimizing cloud costs and attack surface.

Relevant authoritative links for further research:

Docker documentation on multi-stage builds: https://docs.docker.com/build/building/multi-stage/
Best practices for writing Dockerfiles: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
Distroless images: https://github.com/GoogleContainerTools/distroless

Question: 347 CertyIQ

You plan to create an image that will contain a .NET Core application.
You have a Dockerfile file that contains the following code. (Line numbers are included for reference only.)

```
01 FROM microsoft/dotnet: 3.1-sdk
02 COPY . /
03 RUN dotnet publish -c Release -o out
04 FROM microsoft/dotnet: 3.1-sdk
05 COPY --from=0 /out /
06 WORKDIR /
07 ENTRYPOINT [“dotnet”, “app1.dll”]
```

You need to ensure that the image is as small as possible when the image is built.
Which line should you modify in the file?

A. 1
B. 3
C. 4
D. 7

Answer: C

Explanation:

Correct answer is 4. Final image shall not be sdk but runtime image

Question: 348 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company has a project in Azure DevOps for a new web application.
You need to ensure that when code is checked in, a build runs automatically.
Solution: From the Triggers tab of the build pipeline, you select Batch changes while a build is in progress.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution involves enabling the "Batch changes while a build is in progress" option within the build pipeline's Triggers settings. While this option is related to triggering builds, it does not directly guarantee that a build runs automatically whenever code is checked in.

The "Batch changes while a build is in progress" feature optimizes build execution when multiple code changes occur in quick succession. If a build is already running due to a previous commit, enabling this option will group subsequent commits into a single build execution instead of triggering a new build for each commit. This helps to consolidate build processes and reduce resource consumption.

To ensure that a build runs automatically whenever code is checked in, you need to configure a Continuous Integration (CI) trigger. This is typically done by enabling the CI trigger within the Triggers settings of the build pipeline and specifying the branches that should trigger the build. When code is pushed to a specified branch, the CI trigger will automatically initiate a new build. The "Batch changes while a build is in progress" option works in conjunction with CI triggers but doesn't replace the necessity of having a CI trigger configured. Therefore, the proposed solution alone does not satisfy the requirement of automatically running a build on every code check-in.

References:

Configure triggers - Azure Pipelines | Microsoft Learn

Question: 349 CertyIQ

HOTSPOT -
You need to deploy Azure Kubernetes Service (AKS) to host an application. The solution must meet the following requirements:
- Containers must only be published internally.
- AKS clusters must be able to create and manage containers in Azure.

What should you use for each requirement? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

Containers must only be published internally:
- Azure Container Instances
- Azure Container Registry
- Dockerfile

AKS clusters must be able to create and manage containers in Azure:
- An Azure Active Directory (Azure AD) group
- An Azure Automation account
- An Azure service principal

Answer:

Answer Area

Containers must only be published internally:
- Azure Container Instances
- **Azure Container Registry**
- Dockerfile

AKS clusters must be able to create and manage containers in Azure:
- An Azure Active Directory (Azure AD) group
- An Azure Automation account
- **An Azure service principal**

Explanation:
Box 1: Azure Container Registry -
Azure services like Azure Container Registry (ACR) and Azure Container Instances (ACI) can be used and connected from independent container orchestrators like kubernetes (k8s). You can set up a custom ACR and connect it to an existing k8s cluster to ensure images will be pulled from the private container registry instead of the public docker hub.
Box 2: An Azure service principal
When you're using Azure Container Registry (ACR) with Azure Kubernetes Service (AKS), an authentication mechanism needs to be established. You can set up
AKS and ACR integration during the initial creation of your AKS cluster. To allow an AKS cluster to interact with ACR, an Azure Active Directory service principal is used.

Reference:
https://thorsten-hans.com/how-to-use-private-azure-container-registry-with-kubernetes
https://docs.microsoft.com/en-us/azure/aks/cluster-container-registry-integration

Question: 350 CertyIQ

You have 50 Node.js-based projects that you scan by using WhiteSource. Each project includes Package.json, Package-lock.json, and Npm-shrinkwrap.json files.
You need to minimize the number of libraries reports by WhiteSource to only the libraries that you explicitly reference.
What should you do?

A. Configure the File System Agent plug-in.

B. Add a devDependencies section to Package-lock.json.
C. Configure the Artifactory plug-in.
D. Delete Package-lock.json.

Answer: B

Explanation:

The correct answer is B: Add a devDependencies section to Package-lock.json. Here's why:

The goal is to reduce the number of libraries reported by WhiteSource to only those explicitly referenced in the projects. WhiteSource, like other Software Composition Analysis (SCA) tools, analyzes project dependencies to identify open-source components and their associated risks (vulnerabilities, licenses, etc.). Package.json lists the direct dependencies of a Node.js project. Package-lock.json (or npm-shrinkwrap.json) records the exact versions of all dependencies (direct and transitive) used in a build, ensuring reproducibility.

WhiteSource, by default, likely analyzes all dependencies listed in Package-lock.json (or npm-shrinkwrap.json), including development dependencies. Development dependencies (e.g., testing frameworks, linters, build tools) are only needed during development and aren't part of the runtime application. Reporting these unnecessary dependencies inflates the report and requires unnecessary effort to triage vulnerabilities that don't impact the deployed application.

While Package.json does have the devDependencies section, Package-lock.json typically does not explicitly separate development dependencies from production dependencies in its original format. However, the Node.js ecosystem offers a way to manage development dependencies properly.

By adding a devDependencies section to Package-lock.json (or rather, ensuring dependencies are correctly identified as devDependencies during npm install operations that generate the lock file), you signal to WhiteSource that these are development-only dependencies. When you run npm install --production (or set the NODE_ENV environment variable to production), these devDependencies will be ignored. However, importantly, npm itself doesn't allow manual addition of devDependencies to the lock file. The lockfile is auto-generated based on the package.json and subsequent npm install commands. What this means is you need to ENSURE that each dependency is correctly added either using npm install --save-dev <package> or by adding it to package.json under the devDependencies and then running npm install.

WhiteSource can then be configured (or may be configured by default) to ignore the devDependencies section when analyzing the project, thus minimizing the number of libraries reported. This focuses the security analysis on runtime dependencies, reducing noise and effort. Options A, C, and D don't directly address the issue of distinguishing between runtime and development dependencies within the context of how WhiteSource analyzes dependencies.

The key here is the proper declaration of devDependencies in the package.json file and generating the package-lock.json file accordingly by using npm install.

Further reading:

npm documentation on package-lock.json: https://docs.npmjs.com/cli/v9/configuring-npm/package-lock-json
npm documentation on devDependencies: https://docs.npmjs.com/cli/v9/configuring-npm/package-json#devdependencies

Question: 351 CertyIQ

Your company deploys applications in Docker containers.
You want to detect known exploits in the Docker images used to provision the Docker containers.
You need to integrate image scanning into the application lifecycle. The solution must expose the exploits as early as possible during the application lifecycle.
What should you configure?

A. a task executed in the continuous integration pipeline and a scheduled task that analyzes the image registry

B. manual tasks performed during the planning phase and the deployment phase

C. a task executed in the continuous deployment pipeline and a scheduled task against a running production container

D. a task executed in the continuous integration pipeline and a scheduled task that analyzes the production container

Answer: A

Explanation:

The correct answer is A: a task executed in the continuous integration pipeline and a scheduled task that analyzes the image registry.

Here's why:

Early Detection is Key: The requirement is to expose exploits as early as possible in the application lifecycle. This emphasizes the importance of integrating security checks within the CI pipeline. Scanning images during CI allows developers to identify vulnerabilities before the images are deployed, minimizing potential risks.

Continuous Integration Pipeline Integration: Integrating a scanning task in the CI pipeline ensures that every new or modified image is automatically scanned for vulnerabilities. This prevents vulnerable images from being deployed in the first place. This promotes a "shift-left" approach to security, where security considerations are integrated earlier in the development process.

Image Registry Scanning: Regularly scanning the image registry is also crucial. The registry stores the images used for deployment. By scanning it, you can detect newly discovered vulnerabilities in images that have already been built and stored. This provides a safety net for vulnerabilities discovered after the initial CI build and scan. If a new vulnerability is found in an older image, the team can rebuild and redeploy the affected application.

Why Other Options are Incorrect:

B (Manual Tasks): Manual tasks are prone to human error and cannot provide the continuous, automated security assessment required. They are less frequent and less reliable.
C (Continuous Deployment Pipeline and Running Container): Scanning during the CD pipeline is better than nothing, but it's still too late in the process. Vulnerabilities are detected closer to production, increasing the risk window and potential impact. Scanning a running production container can find vulnerabilities, but the vulnerability is already live, which is the opposite of early detection.
D (Continuous Integration Pipeline and Production Container): While the CI pipeline scan is valuable, only scanning the production container is insufficient because it misses vulnerabilities early in the build/test phase.

Cloud Computing Concepts: This scenario highlights several important cloud computing concepts:

DevSecOps: Integrating security into the DevOps lifecycle.
Continuous Security: Automating security checks throughout the application lifecycle.
Infrastructure as Code (IaC): Managing and provisioning infrastructure through code. Image scanning as part of the CI/CD pipeline is integral to IaC security.

Supporting Links:

Microsoft Defender for Cloud: Provides integrated security management and advanced threat protection. Can be used for image scanning in Azure Container Registry. https://azure.microsoft.com/en-us/products/defender-for-cloud/
Azure Container Registry (ACR): Microsoft's container registry service which can integrate with Defender for Cloud for vulnerability scanning: https://azure.microsoft.com/en-us/products/container-registry/
Shift-Left Security: General concept of moving security earlier in the development process. Many articles discuss this topic; search engines are a good resource.

Question: 352 CertyIQ

Your company has a hybrid cloud between Azure and Azure Stack.
The company uses Azure DevOps for its full CI/CD pipelines. Some applications are built by using Erlang and Hack.
You need to ensure that Erlang and Hack are supported as part of the build strategy across the hybrid cloud. The solution must minimize management overhead.
What should you use to execute the build pipeline?

A. a Microsoft-hosted agent
B. Azure DevOps self-hosted agents on Azure DevTest Labs virtual machines.
C. Azure DevOps self-hosted agents on Hyper-V virtual machines
D. Azure DevOps self-hosted agents on virtual machines that run on Azure Stack

Answer: D

Explanation:

The correct answer is **D. Azure DevOps self-hosted agents on virtual machines that run on Azure Stack.**

Here's a detailed justification:

The requirement is to support Erlang and Hack builds across a hybrid cloud (Azure and Azure Stack) with minimal management overhead. Microsoft-hosted agents (Option A) are unsuitable because they offer a limited selection of pre-installed software. Erlang and Hack are unlikely to be pre-installed and you generally cannot customize these agents with specific software requirements.

Azure DevTest Labs (Option B) provides on-demand environments for testing and development, which is not directly relevant to the need for consistent build execution across the hybrid cloud. While you could theoretically deploy self-hosted agents in DevTest Labs, it adds unnecessary complexity and overhead compared to a more direct approach.

Hyper-V virtual machines (Option C), while viable, don't inherently address the hybrid cloud aspect. They would be suitable only for on-premise deployments, while a more integrated solution addressing the Hybrid cloud requirement is necessary.

Self-hosted agents (Option D) are the most appropriate solution. They can be installed on virtual machines running on Azure Stack, thus extending the CI/CD pipeline to the Azure Stack environment. This approach gives you full control over the software installed on the build agents, ensuring that Erlang and Hack dependencies are available. By using VMs on Azure Stack, you align your build environment with your deployment environment, which is crucial for a hybrid cloud. Since the applications are also built and deployed on Azure Stack, it is essential to have agents on Azure Stack.

The self-hosted agent approach minimizes management overhead because you can manage the agents using Azure DevOps and treat your Azure Stack VMs as a dedicated build infrastructure. This avoids the overhead of maintaining separate build environments or customizing Microsoft-hosted agents.

Further research:

Azure DevOps Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=yaml%2Cbrowser
Azure Stack: https://azure.microsoft.com/en-us/products/azure-stack/

Question: 353 CertyIQ

Your company has an Azure DevOps project,
The source code for the project is stored in an on-premises repository and uses on an on-premises build server.
You plan to use Azure DevOps to control the build process on the build server by using a self-hosted agent.
You need to implement the self-hosted agent.
You download and install the agent on the build server.
Which two actions should you perform next? Each correct answer presents part of the solution.

A. From Azure, create a shared access signature (SAS).
B. From the build server, create a certificate, and then upload the certificate to Azure Storage.
C. From the build server, create a certificate, and then upload the certificate to Azure Key Vault.
D. From DevOps, create a personal access token (PAT).
E. From the build server, run config.cmd.

Answer: DE

Explanation:

The correct answer is DE. Here's why:

D. From DevOps, create a personal access token (PAT).

A Personal Access Token (PAT) is crucial for authenticating the self-hosted agent with your Azure DevOps organization. When the agent connects, it needs a way to prove its identity and authorization to access resources in your Azure DevOps organization. A PAT serves as this secure credential. PATs enable fine-grained control over permissions granted to the agent, limiting its access to only the necessary resources within the project. Without a PAT, the agent won't be able to communicate with Azure DevOps.

E. From the build server, run config.cmd.

The config.cmd script is the configuration utility that comes with the Azure DevOps agent. This script guides you through the process of configuring the agent, including connecting it to your Azure DevOps organization. During configuration, the script will prompt you for the Azure DevOps organization URL and the PAT you created. config.cmd handles tasks like registering the agent with the agent pool, specifying the agent's working directory, and configuring any necessary security settings. Running this command is the essential step in establishing the link between your on-premises build server and Azure DevOps.

Let's examine why the other options are incorrect:

A. From Azure, create a shared access signature (SAS). SAS is used to grant limited access to Azure Storage resources. While storage might be tangentially related, it isn't directly involved in authenticating and configuring the build agent for Azure DevOps. The agent needs a PAT for Azure DevOps authentication, not an SAS for Storage.
B. From the build server, create a certificate, and then upload the certificate to Azure Storage. Certificates can be used for various security purposes, but the communication between the self-hosted agent and Azure DevOps is primarily handled using PATs for authentication. Certificates and Azure Storage are not fundamentally required for this scenario.
C. From the build server, create a certificate, and then upload the certificate to Azure Key Vault. While certificates are valuable for secure communication, Azure Key Vault is typically used for managing secrets,

keys, and certificates for Azure resources. For a self-hosted agent, PATs provide a simpler and more direct means of authentication.

Authoritative Links:

Personal Access Tokens (PATs): https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops&tabs=preview-page
Self-hosted Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops

Question: 354 CertyIQ

You have an Azure subscription that contains an Azure Active Directory (Azure AD) tenant.
You are configuring a build pipeline in Azure Pipelines that will include a task named Task1. Task1 will authenticate by using an Azure AD service principal.
Which three values should you configure for Task1? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. the tenant ID
B. the subscription ID
C. the client secret
D. the app ID
E. the object ID

Answer: ADC

Explanation:

The correct answer is A, D, and C. Here's why:

When a build pipeline needs to authenticate to Azure resources using a service principal, it essentially needs three key pieces of information: a unique identifier for the application (the service principal itself), a secret to prove its identity, and information about the Azure AD tenant the service principal belongs to.

A. The Tenant ID: The tenant ID identifies the specific Azure Active Directory instance to which the service principal belongs. This is crucial because Azure AD is multi-tenant, and the pipeline needs to know which directory to authenticate against. Without the tenant ID, the pipeline wouldn't know where to send the authentication request.

D. The App ID (also known as Client ID): This is the unique identifier for the application (the service principal) within Azure AD. It's like a username for the service principal. Azure Pipelines uses this ID to identify the service principal it's trying to use.

C. The Client Secret: This is a secret key (password) associated with the service principal. It's used to prove that the pipeline (acting as the service principal) is authorized to access the Azure resources. Think of it as the password associated with the username (App ID).

Why the other options are incorrect:

B. The Subscription ID: The subscription ID identifies the Azure subscription where resources are deployed. While access to a subscription is the goal of authenticating with the service principal, the subscription ID itself isn't directly used in the authentication process. The service principal is granted permissions within the subscription through Role-Based Access Control (RBAC). However, the pipeline uses the App ID, Tenant ID

and Client Secret to authenticate to the service principal; RBAC then authorizes access.

E. The Object ID: While the object ID is an identifier associated with the service principal object in Azure AD, it is not directly used in authentication by the service principal. The app ID is used for authentication instead. The object ID is generally for identity management related tasks and not authentication tasks

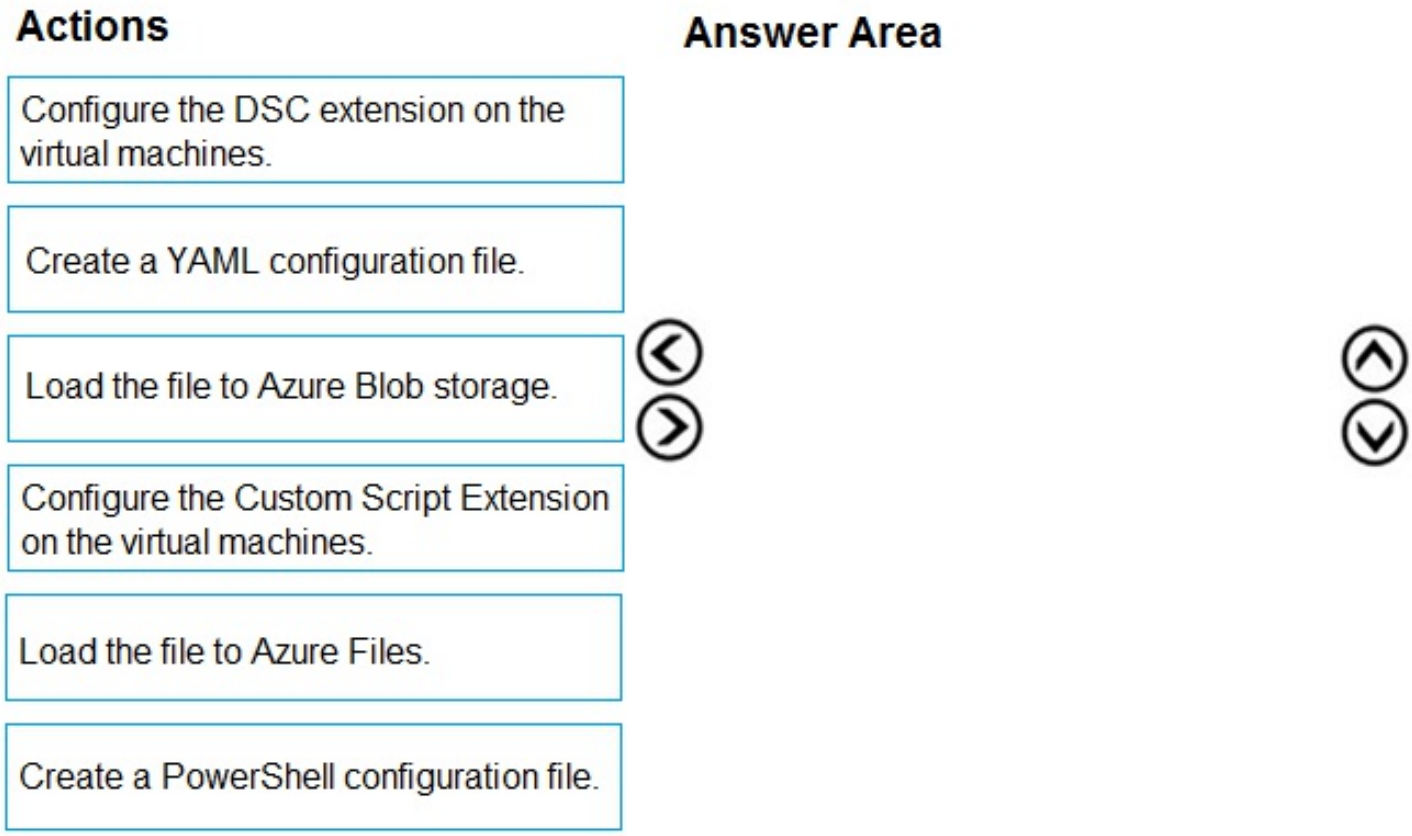
In summary: The authentication process relies on identifying the Azure AD tenant (Tenant ID), uniquely identifying the application (App ID/Client ID), and providing a secret (Client Secret) to prove the application's identity. These three values allow the pipeline to successfully authenticate as the service principal and gain access to Azure resources based on the service principal's configured permissions.

Authoritative Links:

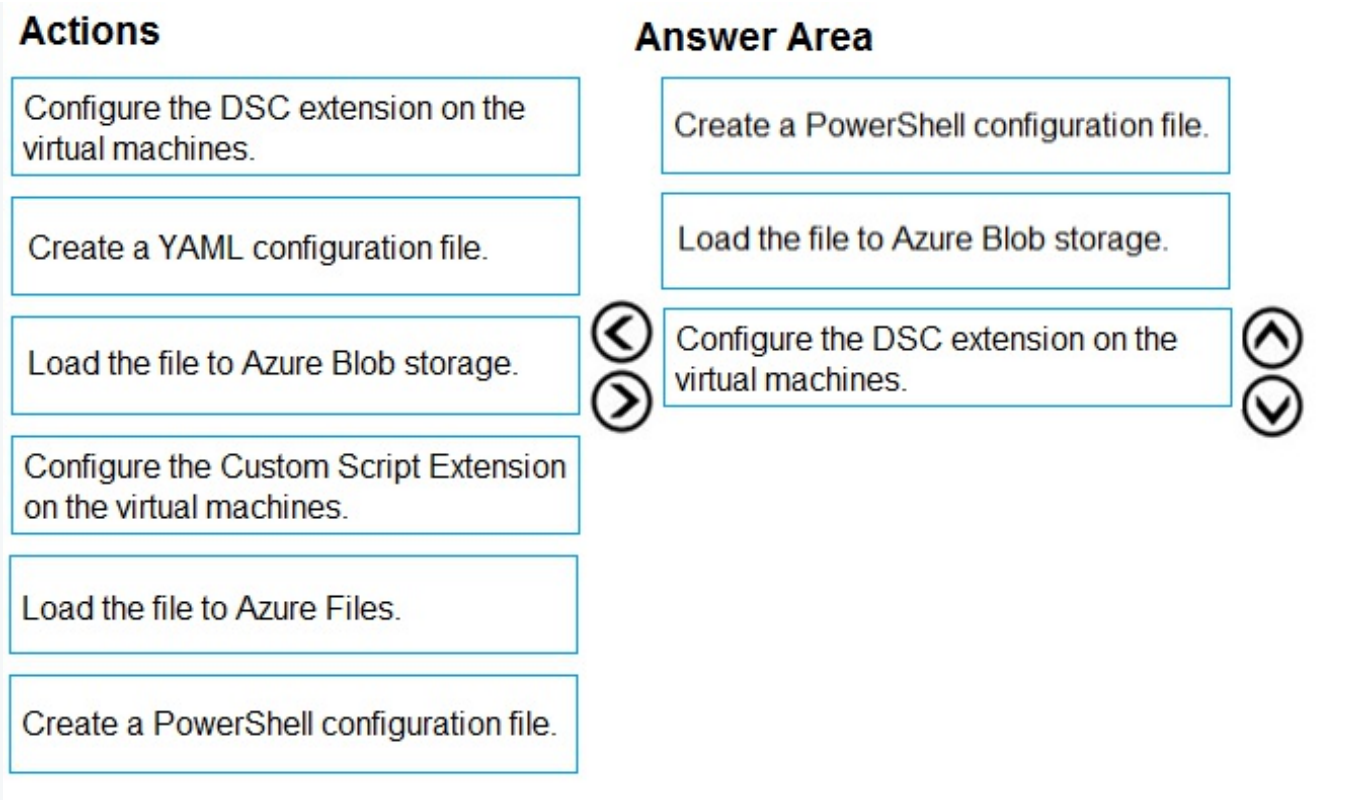
Service Principals: https://learn.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals
Azure DevOps Service Connections: https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml

Question: 355 CertyIQ

DRAG DROP -
You are deploying a new application that uses Azure virtual machines.
You plan to use the Desired State Configuration (DSC) extension on the virtual machines.
You need to ensure that the virtual machines always have the same Windows feature installed.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Step 1: Create a PowerShell configuration file.

You create a simple PowerShell DSC configuration file.

Step 2: Load the file to Azure Blob storage.

Package and publish the module to a publically accessible blob container URL

Step 3: Configure the *DSC extension* on the virtual machines.

Reference:

https://docs.microsoft.com/en-us/azure/automation/automation-dsc-onboarding

Question: 356 CertyIQ

You need to execute inline testing of an Azure DevOps pipeline that uses a Docker deployment model. The solution must prevent the results from being published to the pipeline.
What should you use for the inline testing?

A. a single stage Dockerfile
B. an Azure Kubernetes Service (AKS) pod
C. a multi-stage Dockerfile
D. a Docker Compose file

Answer: C

Explanation:

The correct answer is C: a multi-stage Dockerfile.

A multi-stage Dockerfile is the optimal solution for inline testing without publishing results in an Azure DevOps pipeline that uses a Docker deployment model because it allows you to define multiple stages within a single Dockerfile. This capability is crucial for creating a distinct testing environment that doesn't affect the final image.

Here's why the other options are less suitable:

A. A single-stage Dockerfile: Single-stage Dockerfiles build only one image. Testing within this image would directly modify it, making it difficult to isolate test results and prevent them from being included in the final deployed image.

B. An Azure Kubernetes Service (AKS) pod: While AKS is excellent for deploying and managing containerized applications, using a dedicated AKS pod just for inline testing within a pipeline adds significant overhead and complexity. It is typically used for integration or end-to-end tests, not unit or component tests within the build pipeline.

D. A Docker Compose file: Docker Compose is designed for defining and running multi-container applications, not specifically for structuring and executing testing stages within a Docker build process. While you could potentially use it, it is not as efficient or clean as a multi-stage Dockerfile.

With a multi-stage Dockerfile, you can create a dedicated testing stage (e.g., using a testing framework like pytest or Jest) after the build stage. The testing stage can install necessary dependencies and execute tests. Importantly, this stage is separate from the final production image stage. Only the artifacts from the final stage are copied to the final image, effectively preventing test results and testing tools from being included in the deployed application. The build stage includes a FROM instruction specifying a testing image (e.g. alpine/pytest). The commands for testing are then executed. Then, the final image is built FROM the base image for production without test dependencies, copying the artifact from the prior stage.

For example:

Build stageFROM node:16 AS builderWORKDIR /appCOPY package*.json ./RUN npm installCOPY . .RUN npm run bu

In this example, the testing is performed in the tester stage using node and jest, but the testing image is not the base of the final image.Therefore, a multi-stage Dockerfile is the most efficient and clean solution for inline testing without publishing results.

Supporting Documentation:

Multi-stage builds: https://docs.docker.com/develop/develop-images/multistage-build/
Docker Compose: https://docs.docker.com/compose/
Azure Kubernetes Service: https://learn.microsoft.com/en-us/azure/aks/intro-kubernetes

Question: 357 CertyIQ

You are designing an Azure DevOps strategy for your company's development team.
You suspect that the team's productivity is low due to accumulate technical debt.
You need to recommend a metric to assess the amount of the team's technical debt.
What should you recommend?

A. the number of code modules in an application
B. the number of unit test failures
C. the percentage of unit test failures
D. the percentage of overall time spent on rework

Answer: D

Explanation:

The correct answer is **D. the percentage of overall time spent on rework**. Here's why:

Technical debt, in essence, represents the implied cost of additional rework caused by choosing an easy solution now instead of using a better approach that would take longer. It's the gap between current functionality and what a system truly requires. It manifests as increased effort in maintaining, modifying, or extending the system. Rework is a direct consequence of technical debt.

Option A, the number of code modules, doesn't directly indicate technical debt. A well-structured application may have many modules.

Option B, the number of unit test failures, while important, is only a symptom of potential technical debt. Failures can stem from bugs or incomplete functionality, not necessarily debt.

Option C, the percentage of unit test failures, is a slightly better indicator than the raw number. However, it still only focuses on the testing aspect. It doesn't necessarily tell us about other forms of debt, such as poor documentation or architectural shortcuts.

Option D, the percentage of overall time spent on rework, directly reflects the impact of accumulated technical debt on the team's productivity. High rework percentage indicates the team is constantly revisiting existing code to fix issues, refactor, or address problems caused by earlier compromises. This is a far more holistic metric because it accounts for the cumulative impact of all forms of technical debt across the development lifecycle. This can surface issues beyond simple bugs, such as brittle code, poor designs, and inconsistent patterns.

By measuring the proportion of time spent on rework, the team gets a tangible indication of how technical debt is affecting velocity and can prioritize addressing it. Tracking this over time will also indicate if technical debt is improving or worsening.

Relevant Resources:

Martin Fowler's Technical Debt Quadrant: https://martinfowler.com/bliki/TechnicalDebtQuadrant.html
SonarQube's Definition of Technical Debt: https://www.sonarsource.com/resources/webinars/technical-debt-and-how-to-pay-it-back/ (Although specific to SonarQube, the definition of technical debt is valuable.)

Question: 358 CertyIQ

You are developing an open source solution that uses a GitHub repository.
You create a new public project in Azure DevOps.
You plan to use Azure Pipelines for continuous build. The solution will use the GitHub Checks API.
Which authentication type should you use?

A. OpenID
B. GitHub App
C. a personal access token (PAT)
D. SAML

Answer: B

Explanation:

The correct authentication type for using Azure Pipelines with GitHub Checks API in a public open-source

project is a **GitHub App**. Here's a detailed justification:

GitHub Apps offer granular permissions: GitHub Apps are designed for integrating with GitHub repositories in a more secure and controlled manner compared to Personal Access Tokens (PATs). They allow you to define specific permissions that your application needs, adhering to the principle of least privilege.

GitHub Checks API Requirement: The GitHub Checks API is specifically designed to work seamlessly with GitHub Apps. It provides detailed feedback on code quality and status directly within the GitHub interface, enhancing the developer workflow.

PAT limitations: While a PAT can grant access to a GitHub repository, it acts on behalf of a specific user and grants all the permissions that user has. For a public, open-source project, this is generally not recommended because it could expose the user's credentials if not managed correctly and is an unnecessary grant of broad privileges.

OpenID and SAML are not relevant here: OpenID and SAML are primarily used for user authentication and single sign-on (SSO) scenarios, not for application-to-API authentication that is required for the GitHub Checks API integration.

Enhanced security: GitHub Apps employ short-lived tokens and are less susceptible to credential leaks compared to PATs, which are often long-lived and need to be carefully managed.

Rate limiting and throttling: GitHub Apps are generally subject to higher rate limits compared to PATs, making them better suited for automation tasks in a CI/CD pipeline.

Scalability: GitHub Apps are designed to scale and handle a large number of integrations, making them ideal for open-source projects that may grow over time.

In summary, GitHub Apps are the most secure, granular, and scalable method to authenticate with GitHub for the purpose of using the Checks API within an Azure Pipeline in an open-source project. It provides the necessary permissions without the security risks associated with PATs and is the appropriate choice for this scenario.

Relevant Links for Further Research:

1. **About creating GitHub Apps:** https://docs.github.com/en/developers/apps/building-github-apps/creating-github-apps
2. **Using the Checks API:** https://docs.github.com/en/rest/checks
3. **Authenticating to GitHub:** https://docs.github.com/en/authentication

Question: 359 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company has a project in Azure DevOps for a new web application.
You need to ensure that when code is checked in, a build runs automatically.
Solution: From the Continuous deployment trigger settings of the release pipeline, you enable the Pull request trigger setting.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution involves enabling the "Pull request trigger" in the continuous deployment trigger settings of a release pipeline. However, the goal is to trigger a build automatically when code is checked in (committed) to the repository. A pull request trigger specifically initiates a build when a pull request is created or updated, focusing on validating proposed changes before they are merged into the main branch. It does not trigger a build upon every commit. Therefore, enabling the "Pull request trigger" will not achieve the desired outcome of building the code upon each check-in. To accomplish this, a continuous integration (CI) trigger needs to be configured on the build pipeline, not the continuous deployment (CD) trigger on the release pipeline. The CI trigger monitors the specified branch for changes and automatically starts a new build when a commit is made.

Therefore, the solution does not meet the goal.

References:

Azure Pipelines Triggers
Release Triggers

Question: 360 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company has a project in Azure DevOps for a new web application.
You need to ensure that when code is checked in, a build runs automatically.
Solution: From the Pre-deployment conditions settings of the release pipeline, you select After stage.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution does not meet the goal of automatically triggering a build upon code check-in. The After stage setting in the pre-deployment conditions of a release pipeline controls when a release is deployed after a specific stage completes. It does not relate to triggering a build pipeline upon code commit. Build pipelines are triggered based on source code changes in a repository. To achieve the desired outcome, a CI (Continuous Integration) trigger should be configured within the build pipeline itself. This trigger will monitor the specified branch for changes and initiate a build when a commit occurs. Release pipelines are typically triggered by successful builds, not directly by code check-ins. Configuring After stage only influences the sequencing of deployments within a release pipeline, impacting when subsequent stages are initiated, not when the initial build is kicked off. For triggering builds, the correct configuration lies within the build pipeline definition, specifically the trigger settings.

Authoritative links for further research:

Azure Pipelines Triggers: https://learn.microsoft.com/en-us/azure/devops/pipelines/triggers/?view=azure-devops

Continuous Integration (CI) triggers: https://learn.microsoft.com/en-us/azure/devops/pipelines/build/triggers?view=azure-devops&tabs=yaml
Release Triggers: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/triggers?view=azure-devops

Question: 361 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company has a project in Azure DevOps for a new web application.
You need to ensure that when code is checked in, a build runs automatically.
Solution: From the Pre-deployment conditions settings of the release pipeline, you select Batch changes while a build is in progress.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution does not meet the goal of automatically triggering a build when code is checked in. "Batch changes while a build is in progress" is a feature within the release pipeline, specifically in the pre-deployment conditions. This setting controls how releases are triggered if multiple build completions occur while a release is already in progress. It prevents multiple releases from being triggered simultaneously and potentially conflicting with each other. It does not, however, initiate a build process upon a code check-in.

To automatically trigger a build upon a code check-in, a build pipeline trigger needs to be configured. This is typically achieved through a Continuous Integration (CI) trigger configuration within the build pipeline settings. CI triggers monitor specific branches in the repository for changes. When a commit is made to a monitored branch, the CI trigger initiates a new build of the application. This ensures that code changes are automatically built and tested, supporting rapid iteration and early detection of integration issues.

Therefore, the proposed solution using release pipeline pre-deployment conditions is relevant to the release stage but irrelevant to initiating the build stage upon code commit. The correct approach involves configuring a CI trigger within the build pipeline.

Further research on CI/CD pipelines in Azure DevOps can be found at:

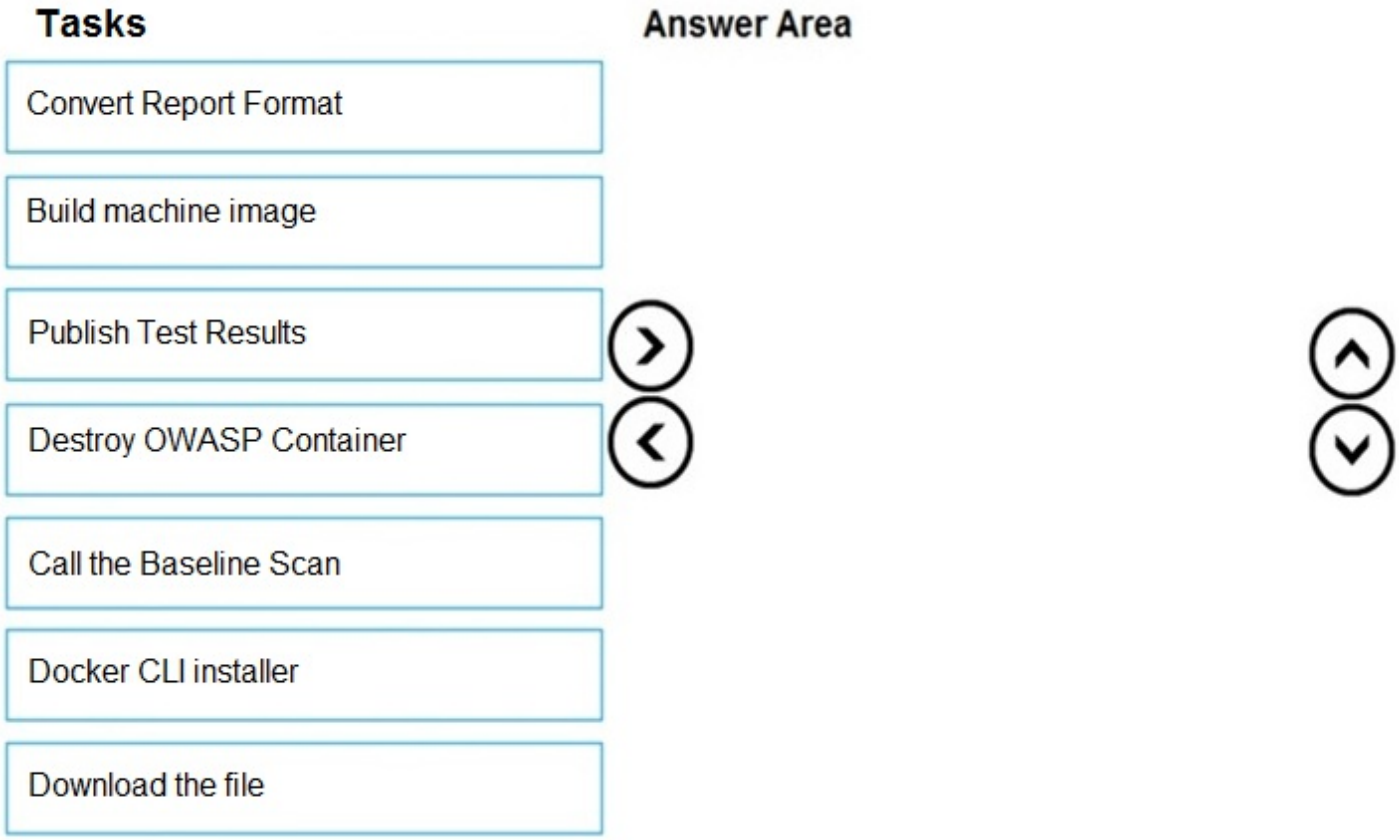
Azure Pipelines documentation
Configure triggers

Question: 362 CertyIQ

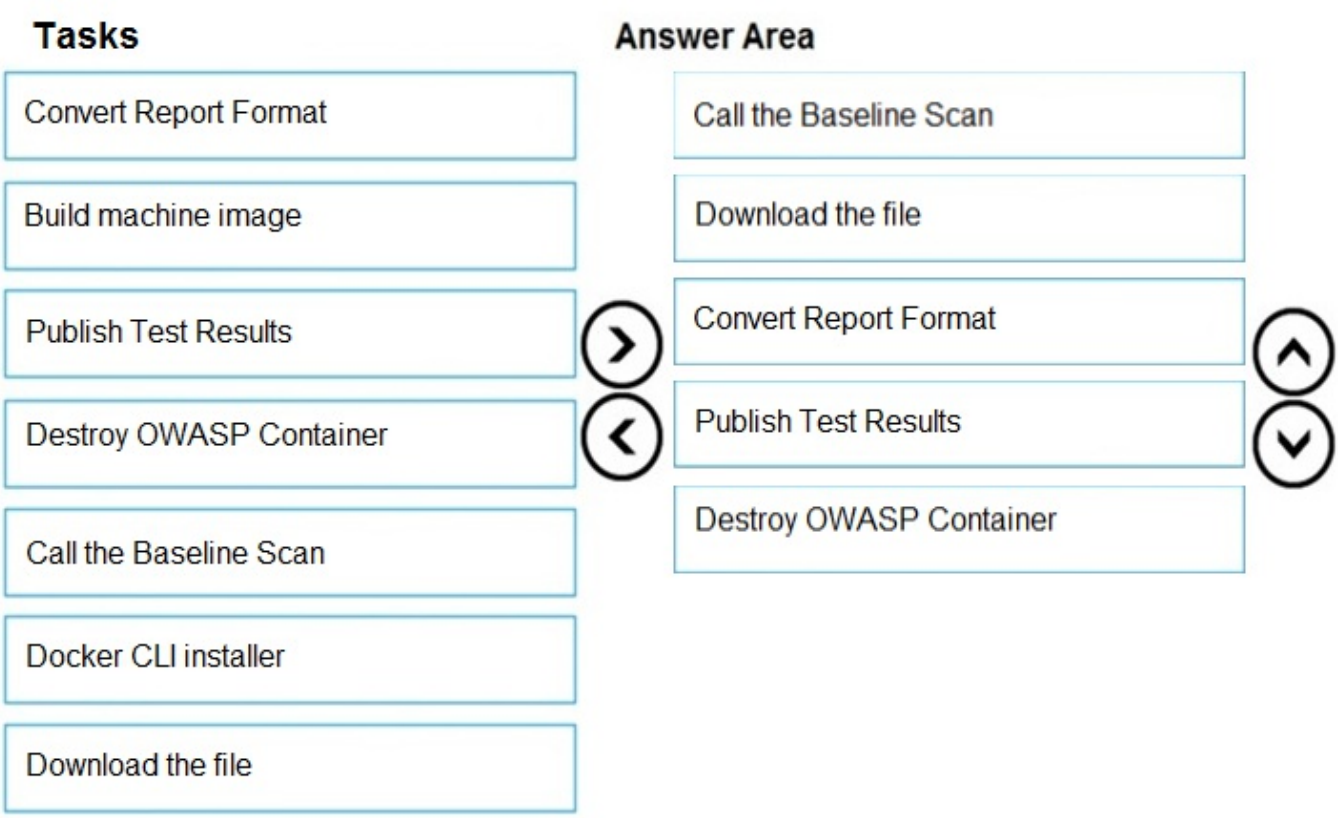
DRAG DROP -
You have an Azure DevOps release pipeline as shown in the following exhibit.



You need to complete the pipeline to configure OWASP ZAP for security testing.
Which five Azure CLI tasks should you add in sequence? To answer, move the tasks from the list of tasks to the answer area and arrange them in the correct order.
Select and Place:



Answer:

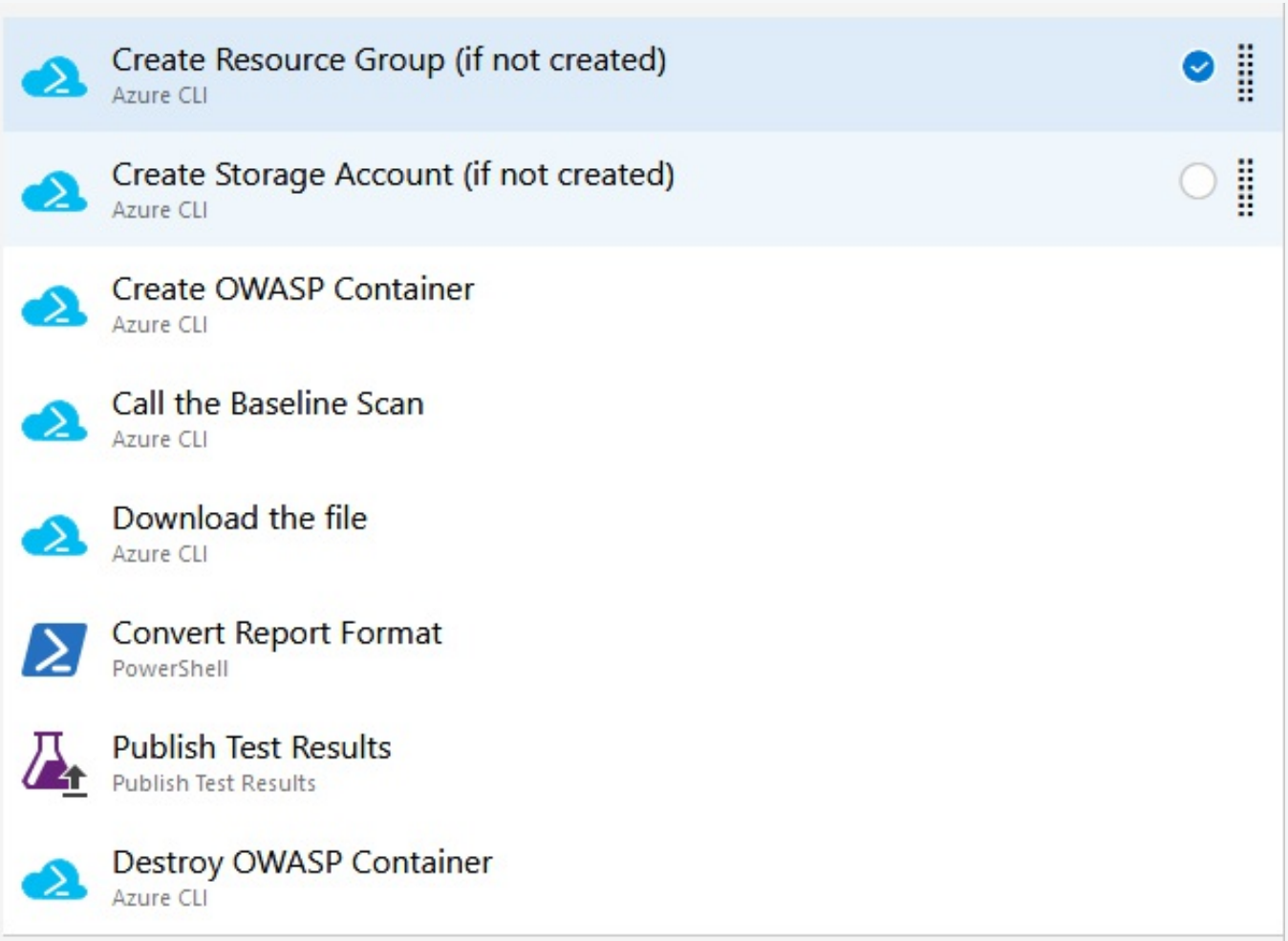


Explanation:

Defining the Release Pipeline -

Once the application portion of the Release pipeline has been configured, the security scan portion can be defined. In our example, this consists of 8 tasks, primarily using the Azure CLI task to create and use the ACI instance (and supporting structures).

Otherwise specified, all the Azure CLI tasks are Inline tasks, using the default configuration options.



Reference:
https://devblogs.microsoft.com/premier-developer/azure-devops-pipelines-leveraging-owasp-zap-in-the-release-pipeline/

Question: 363 CertyIQ

HOTSPOT -
You company uses a Git source-code repository.
You plan to implement GitFlow as a workflow strategy.
You need to identify which branch types are used for production code and preproduction code in the strategy.
Which branch type should you identify for each code type? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

Production code: [▼]
- Master
- Feature
- Develop

Preproduction code: [▼]
- Master
- Feature
- Develop

Answer:

Answer Area

Production code: [Master ▼]
- Master
- Feature
- Develop

Preproduction code: [Develop ▼]
- Master
- Feature
- Develop

Explanation:

Box 1: Master -
The Master branch contains production code. All development code is merged into master in sometime.

Box 2: Develop -
The Develop branch contains pre-production code. When the features are finished then they are merged into develop.
Incorrect Answers:
During the development cycle, a variety of supporting branches are used:
✑ Feature branches are used to develop new features for the upcoming releases. May branch off from develop and must merge into develop.

Reference:
https://medium.com/@patrickporto/4-branching-workflows-for-git-30d0aaee7bf

Question: 364 CertyIQ

You have a build pipeline in Azure Pipelines that uses different jobs to compile an application for 10 different architectures.
The build pipeline takes approximately one day to complete.
You need to reduce the time it takes to execute the build pipeline.
Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Move to a blue/green deployment pattern
B. Create a deployment group
C. Increase the number of parallel jobs
D. Reduce the size of the repository
E. Create an agent pool

Answer: CE

Explanation:

The goal is to reduce the execution time of a build pipeline compiling an application for 10 different architectures. The pipeline currently takes approximately one day to complete, indicating a significant bottleneck.

Option C: Increase the number of parallel jobs

The current pipeline likely executes the compilation for each architecture sequentially or in limited parallel. Increasing the number of parallel jobs will allow multiple architectures to be compiled simultaneously. This directly addresses the bottleneck and reduces the overall build time by leveraging available resources more efficiently. Azure Pipelines supports parallelism using multiple agents, which can be configured to run jobs concurrently.

Option E: Create an agent pool

Creating an agent pool, particularly a self-hosted agent pool with sufficient resources, can significantly improve build performance. Agent pools provide the compute infrastructure required to run the build jobs. Instead of relying solely on Microsoft-hosted agents (which might experience queuing or resource limitations), a dedicated agent pool can ensure that build jobs are executed promptly, leveraging the specific hardware and software configurations needed for the different architectures. This helps optimize build times, especially when dealing with diverse architecture requirements. Using multiple agents in the agent pool enables more parallel jobs as suggested in option C.

Why other options are incorrect:

Option A: Move to a blue/green deployment pattern: Blue/green deployment is a deployment strategy focused on minimizing downtime during releases. It doesn't directly address the build process's execution time.
Option B: Create a deployment group: Deployment groups are used for deploying applications to specific environments, typically servers or virtual machines. Like blue/green deployments, this addresses the deployment phase, not the build phase's duration.
Option D: Reduce the size of the repository: While a smaller repository can improve the time it takes to fetch the code at the beginning of the build, it's unlikely to be the primary bottleneck in a pipeline that takes a whole day. The compilation process itself is likely the dominant factor. If the repository size is excessively large, it may contribute, but parallelization offers more substantial gains.

Therefore, the correct actions are to **increase the number of parallel jobs** and **create an agent pool** to enable efficient and concurrent execution of the build process.

Supporting Links:

Parallel jobs in Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops
Agents in Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=yaml%2cpipeline

Question: 365

CertyIQ

You are creating a build pipeline in Azure Pipelines.
You define several tests that might fail due to third-party applications.
You need to ensure that the build pipeline completes successfully if the third-party applications are unavailable.
What should you do?

A. Configure the build pipeline to use parallel jobs
B. Configure flaky tests
C. Increase the test pass percentage
D. Add the Requirements quality widget to your dashboard

Answer: B

Explanation:

The correct answer is **B. Configure flaky tests**.

Here's why:

Flaky Tests: Flaky tests are tests that sometimes pass and sometimes fail without any code changes. These often arise from dependencies on external systems (like third-party apps) or timing issues. Azure DevOps offers a mechanism to manage flaky tests.

Handling Flaky Tests in Azure Pipelines: When you configure a test as "flaky" in Azure Pipelines, the system can automatically re-run the test a few times. If the test passes at least once within the allowed retries, the overall build result is considered a success. This ensures that occasional failures due to transient third-party unavailability won't cause the entire build to fail.

Why other options are incorrect:

A. Configure the build pipeline to use parallel jobs: Parallel jobs only speed up the execution of the build by

running multiple tasks simultaneously. They don't address the core problem of tests failing due to external dependencies.
C. Increase the test pass percentage: Increasing the required test pass percentage simply raises the bar for build success. It doesn't handle the situation where tests are inherently unreliable due to external factors.
D. Add the Requirements quality widget to your dashboard: This widget is used for monitoring test coverage and requirements traceability, but it doesn't have any impact on the test execution or build outcome in the face of flaky tests.

Therefore, configuring flaky tests is the best approach as it acknowledges and handles the inconsistent nature of tests dependent on potentially unavailable third-party applications, allowing the build to complete successfully if the tests eventually pass after retries.

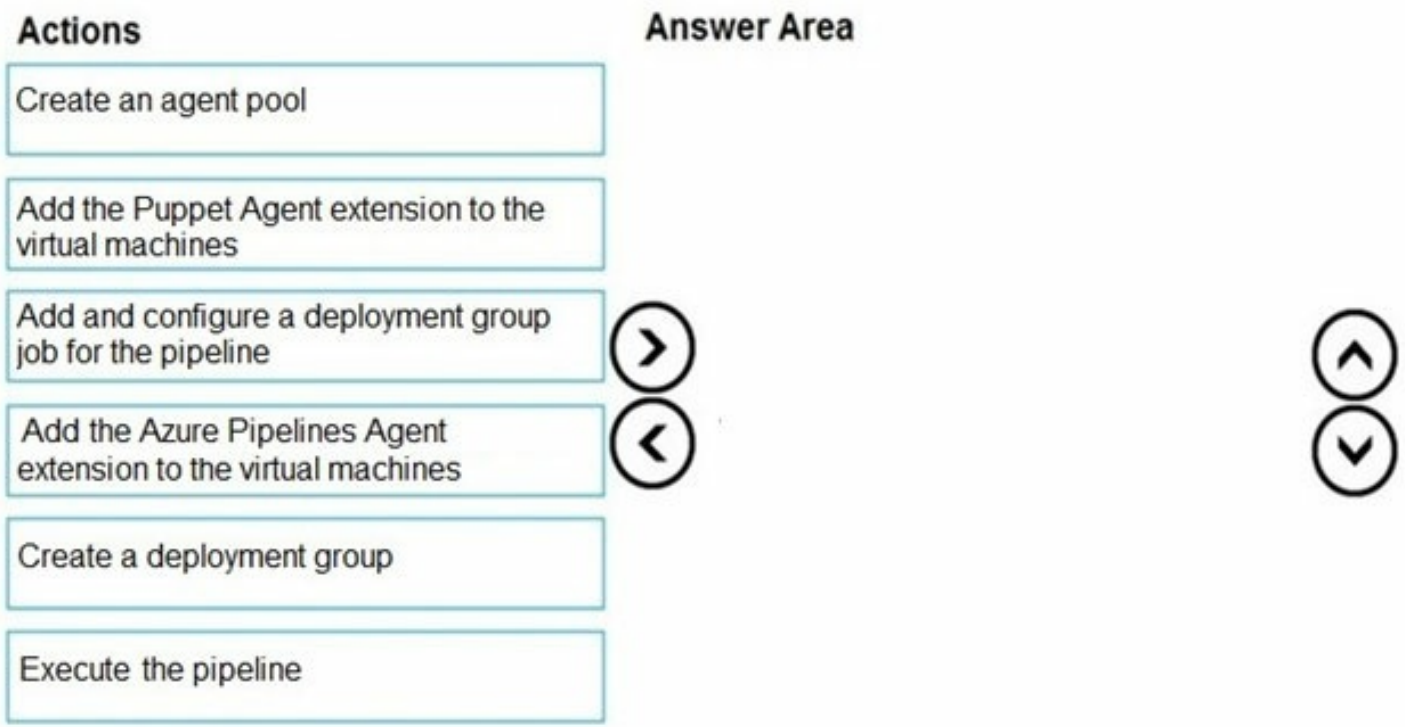
Supporting Documentation:

Manage flaky tests: https://learn.microsoft.com/en-us/azure/devops/test/run-automated-tests-from-test-hub?view=azure-devops&tabs=yaml#manage-flaky-tests

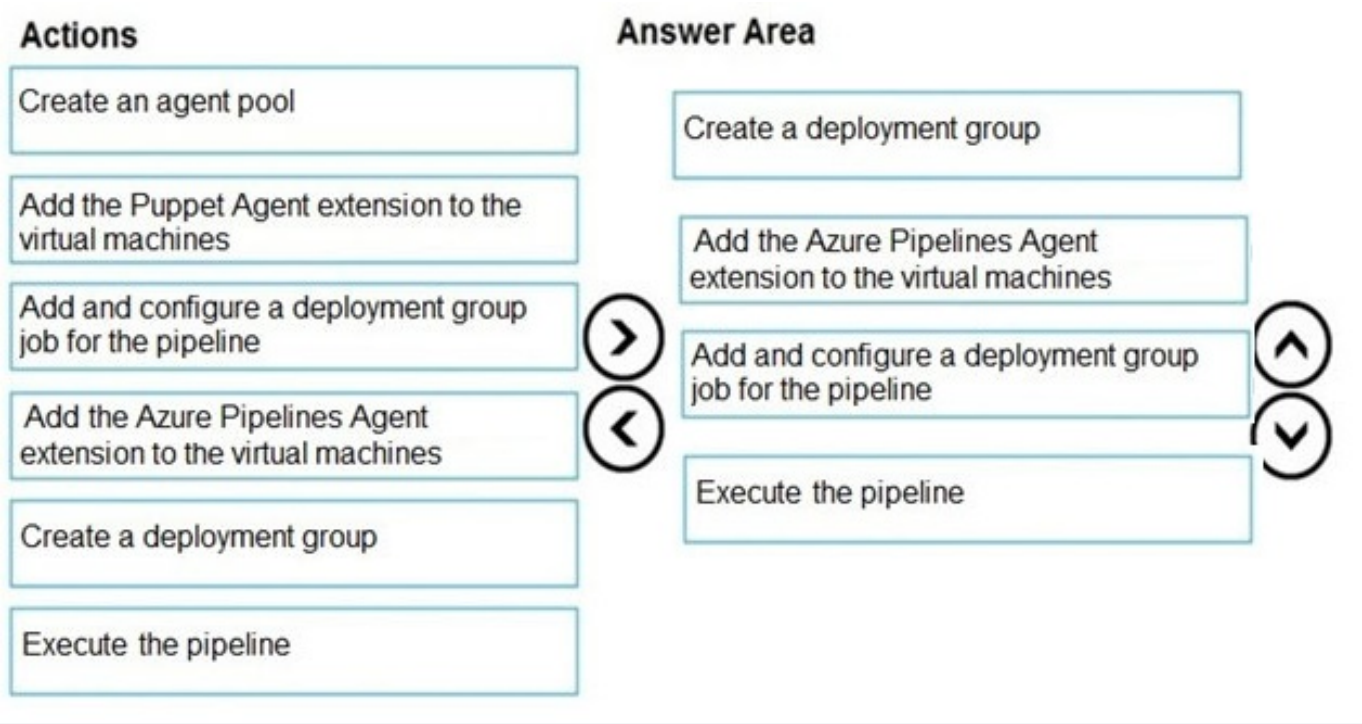
Question: 366 CertyIQ

DRAG DROP -
You have an Azure subscription that contains a resources group named RG1. RG1 contains the following resources:
- Four Azure virtual machines that run Windows Server and have Internet Information Services (IIS) installed.
- SQL Server on an Azure virtual machine.
- An Azure Load Balancer.

You need to deploy an application to the virtual machines in RG1 by using Azure Pipelines.
Which four actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.
Select and Place:



Answer:



Explanation:

1- Create a deployment group and assign a tag to each machine (you can also create multiple groups)

2- Add the Azure Pipelines Agent extension to the virtual machines: once you add a machine, you will get a poweshell script with the PAT. You install the agent by running the script on your machine.

3- Add and configure a deployment group job for the pipeline: hook up the jobs in the pipeline with deployment group's machine

4- Execute the pipeline: run the pipeline.

Question: 367 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company has a project in Azure DevOps for a new web application.
You need to ensure that when code is checked in, a build runs automatically.
Solution: From the Triggers tab of the build pipeline, you select Enable continuous integration.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

The solution meets the goal. Enabling continuous integration (CI) within the Triggers tab of a build pipeline in Azure DevOps directly addresses the requirement to automatically trigger builds upon code check-in. CI is a core DevOps practice focused on automatically building and testing code changes as they are committed to a source control repository. By enabling continuous integration, you configure the build pipeline to monitor the

specified branch (typically the main or a feature branch) for changes. When a commit or pull request is merged into that branch, the pipeline automatically starts a new build. This automated process helps detect integration issues early, facilitates rapid feedback, and promotes continuous quality improvements. The Triggers tab within Azure DevOps build pipelines provides granular control over CI triggers, including specifying the branches to monitor, configuring path filters to limit build triggers to changes within specific directories, and setting batching options to optimize build frequency. The specific option "Enable continuous integration" directly initiates this automated build process upon code check-in.

Authoritative Links:

Azure Pipelines Triggers: https://learn.microsoft.com/en-us/azure/devops/pipelines/build/triggers?view=azure-devops&tabs=yaml
Continuous Integration: https://www.atlassian.com/continuous-delivery/continuous-integration

Question: 368 CertyIQ

You have an Azure DevOps organization named Contoso and an Azure DevOps project named Project1.
You plan to use Microsoft-hosted agents to build container images that will host full Microsoft .NET Framework apps in a YAML pipeline in Project1.
What are two possible virtual machine images that you can use for the Microsoft-hosted agent pool? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A. vs2017-win2016
B. ubuntu-16.04
C. win1803
D. macOS-10.13
E. vs.2015-win2012r2

Answer: AB

Explanation:

The correct answer is AB. Here's why:

A. vs2017-win2016: This image is a valid option because it represents a Windows Server 2016 environment with Visual Studio 2017 pre-installed. This provides a suitable environment for building .NET Framework applications and creating container images based on Windows containers. Microsoft-hosted agents with the vs2017-win2016 image include the necessary .NET Framework SDKs and build tools.

B. ubuntu-16.04: Although .NET Framework apps are typically associated with Windows environments, it is possible to build and containerize a .NET Framework app from a Linux environment. With Mono, an open-source implementation of the .NET Framework, it's possible to build container images that encapsulate the necessary environment for .NET Framework applications, even starting the base build process in a Linux VM. This might be slightly unconventional, but technically feasible by installing the proper dependencies and using Mono.

C. win1803: The win1803 image name is deprecated. Microsoft recommends newer versions for security and features. This is typically an issue with available hosted agents, which are supported and maintained by Microsoft.

D. macOS-10.13: macOS agents are primarily used for building applications targeting Apple platforms (iOS, macOS). They are not suitable for building container images for .NET Framework applications.

E. vs2015-win2012r2: vs2015-win2012r2 is also deprecated, and it's best to use newer, supported agents for security and features.

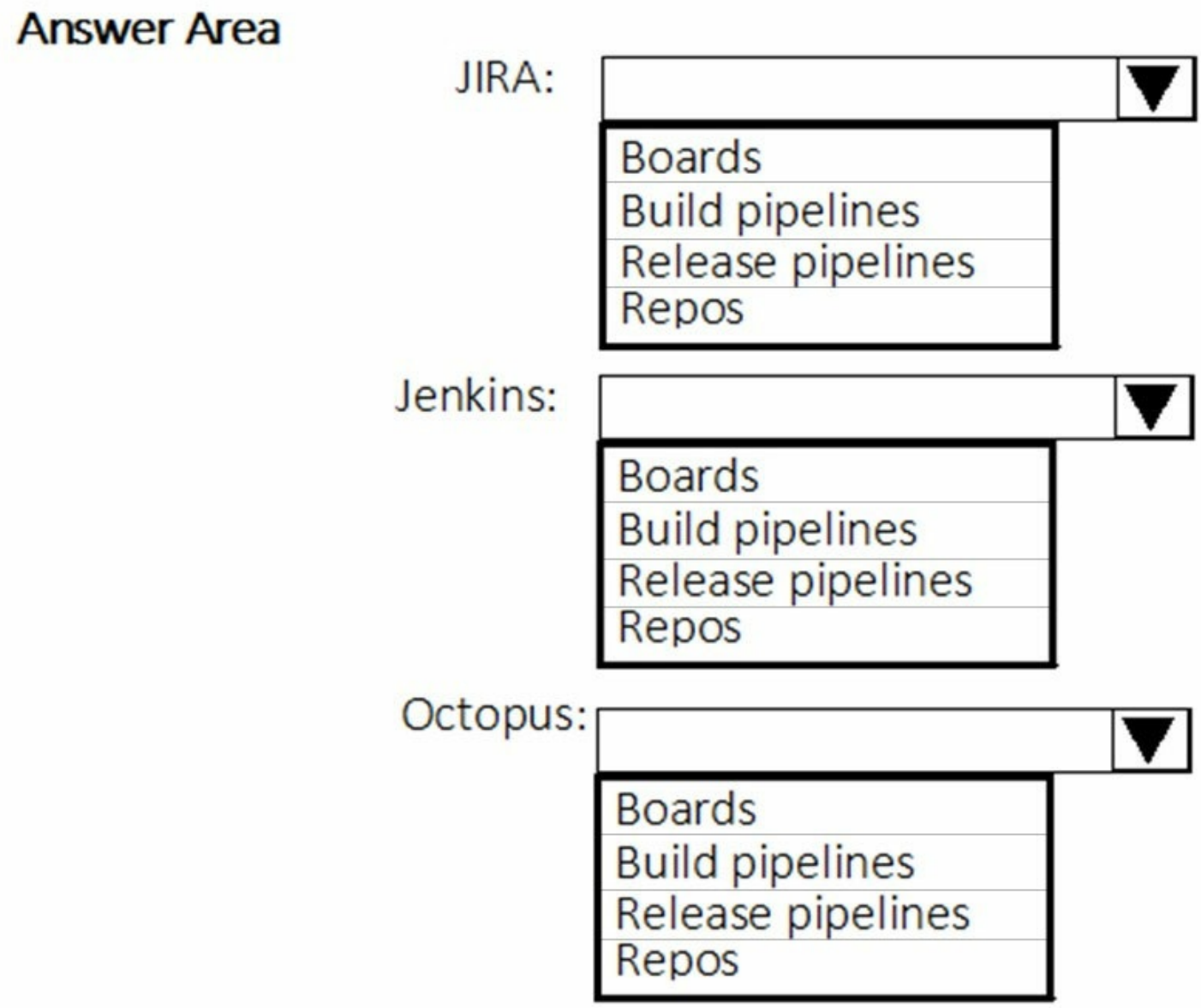
In summary: Both vs2017-win2016 and ubuntu-16.04 provide feasible (though ubuntu-16.04 is not as straightforward as vs2017-win2016) environments for building container images for .NET Framework applications using a Microsoft-hosted agent. Options C, D, and E, are either incorrect or not a best practice according to Microsoft's recommended agent images.

References:

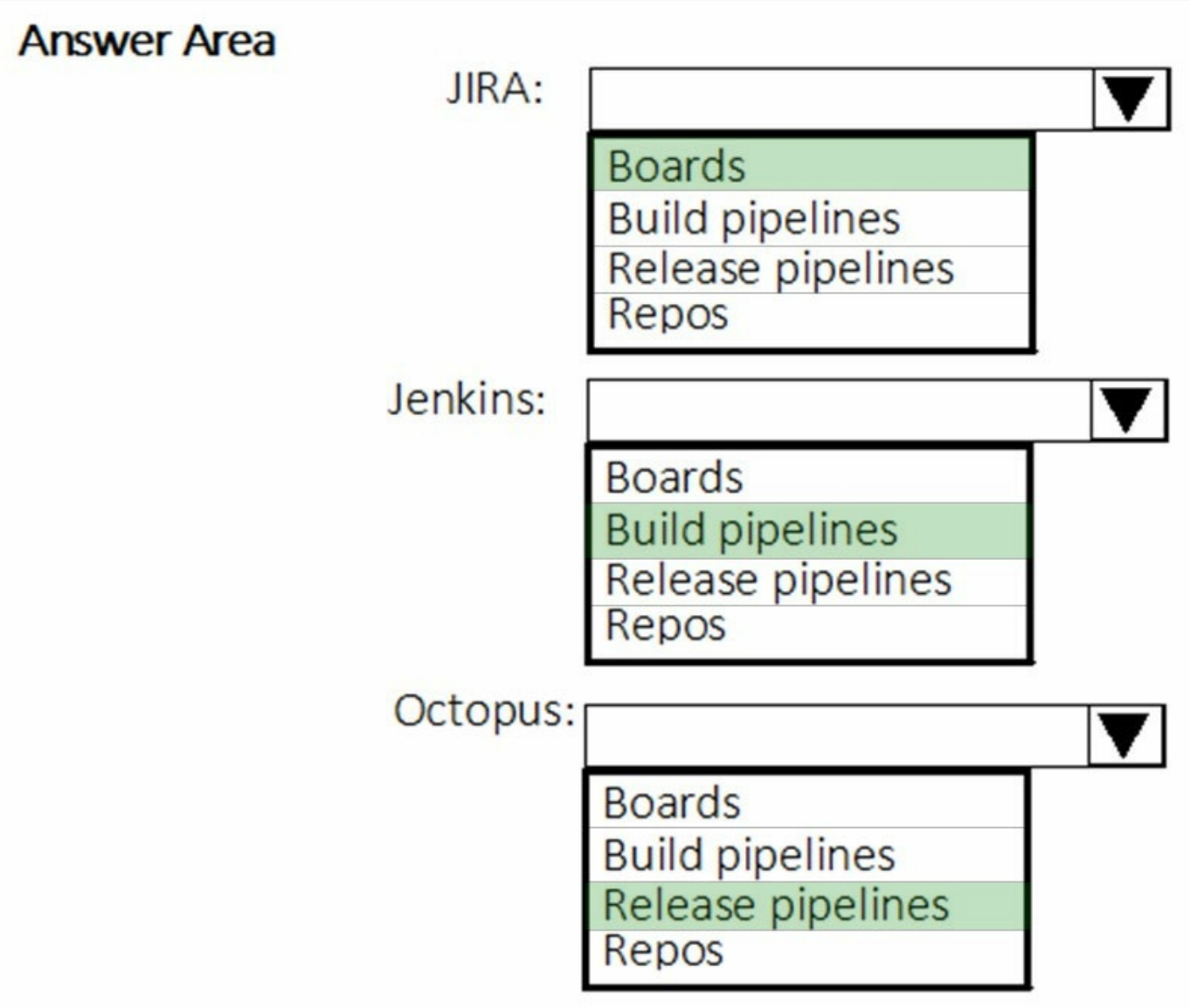
Microsoft-hosted agents for Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/hosted?view=azure-devops&tabs=yaml
Mono project:https://www.mono-project.com/

Question: 369 CertyIQ

HOTSPOT -
You currently use JIRA, Jenkins, and Octopus as part of your DevOps processes.
You plan to use Azure DevOps to replace these tools.
Which Azure DevOps service should you use to replace each tool? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:



Answer:



Explanation:

Reference:
https://octopus.com/blog/octopus-jira-integration
https://www.azuredevopslabs.com/labs/vstsextend/jenkins/

Question: 370 CertyIQ

Your company has a project in Azure DevOps.
You need to ensure that when there are multiple builds pending deployment, only the most recent build is deployed.
What should you use?

A. deployment conditions
B. deployment queue settings
C. release gates
D. pull request triggers

Answer: B

Explanation:

The correct answer is **B. deployment queue settings.**

Here's why:

Deployment queue settings in Azure DevOps directly address the requirement of deploying only the most recent build when multiple builds are pending deployment to an environment. Specifically, the "Deploy latest and cancel others" option ensures that when a new build triggers a deployment to an environment, any currently queued or in-progress deployments of older builds to that environment are automatically canceled, and the latest build proceeds with the deployment.

This functionality effectively prevents older, potentially outdated or irrelevant builds from being deployed, thereby optimizing resource utilization and ensuring that the target environment consistently reflects the latest stable code. This is crucial for maintaining consistency and minimizing the risk of deploying faulty code that has already been superseded by newer, improved versions.

Option A, deployment conditions, controls when a deployment can run based on specific criteria but doesn't inherently cancel existing deployments. Release gates (option C) are used to evaluate environment health and policy compliance before or after deployments but are not primarily designed for managing the deployment queue in the manner described in the scenario. Pull request triggers (option D) are relevant to Continuous Integration processes and are associated with triggering builds based on pull request activities, not the deployment queue management described in the scenario. Therefore, only deployment queue settings provide the direct mechanism to ensure that only the most recent build is deployed, and older builds are canceled.

For more information on Deployment Queue Settings, see:

Deployment Queues in Azure DevOps
Controlling Deployment Queueing

Question: 371 CertyIQ

Your company develops a client banking application that processes a large volume of data.
Code quality is an ongoing issue for the company. Recently, the code quality has deteriorated because of an increase in time pressure on the development team.
You need to implement static code analysis.
During which phase should you use static code analysis?

A. integration testing
B. staging
C. production release
D. build

Answer: D

Explanation:

The correct answer is D (Build). Static code analysis should be performed during the build phase of the software development lifecycle. This is because static analysis tools examine the source code without actually executing the program. Integrating static analysis into the build process allows for early detection of potential issues such as code defects, security vulnerabilities, and adherence to coding standards. By identifying these problems early, developers can address them before they propagate through the later stages of development, such as integration testing, staging, or production. Performing static analysis during the build phase reduces the cost and effort associated with fixing bugs and improves the overall code quality. Addressing issues during the build phase avoids the ripple effect where a single defect can have ramifications

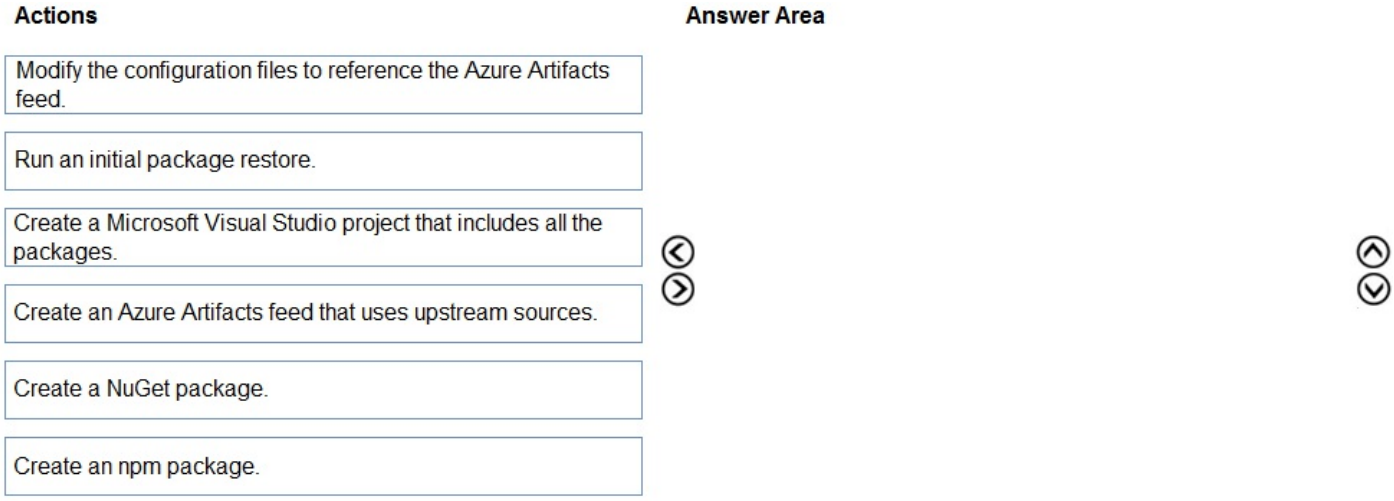
throughout the system. It reduces risks of bugs making it to later stages, and can automatically enforce coding standards. It aligns with the "shift-left" approach to security and quality, moving these concerns earlier in the development pipeline. Integration testing, staging, or production are too late for this type of analysis. Static analysis tools can be integrated directly into the CI/CD pipeline, ensuring that every code change is automatically checked.

References:

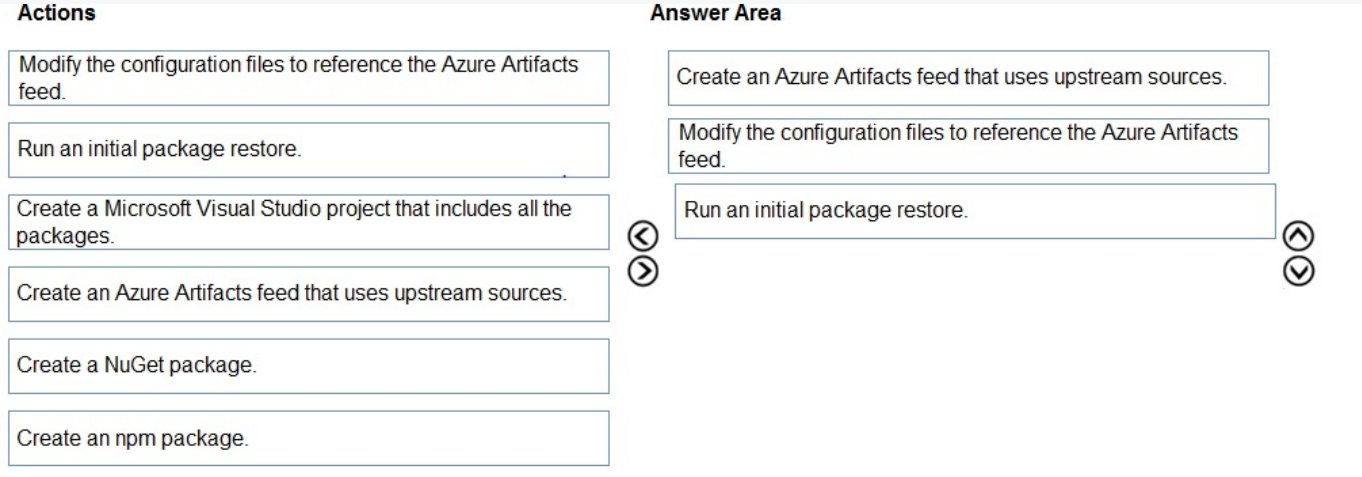
Static Program Analysis
Integrate Static Code Analysis into the Build Process

Question: 372 CertyIQ

DRAG DROP -
You have a project in Azure DevOps that uses packages from multiple public feeds. Some of the feeds are unreliable.
You need to consolidate the packages into a single feed.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

1. Create an azure artifacts feed that uses upstream sources.

Upstream sources enable you to use a single feed to store both the packages you produce and the packages you consume from both public packages managers (npmjs.com, NuGet.org, Maven Central, and PyPI) and

Artifacts feeds.

2. Modify the configuration files to reference the Arure Artifacts feed.

Since they were referencing the public feeds

3. Run an initial package restore.

To pull the packages from the public feed and save them in the azure upstream feed:

Packages belonging to an upstream are available downstream soon after they are published, but will only show up in the feed's UI once they have been 'ingested,' which requires installing the package version for the first time in the downstream feed.

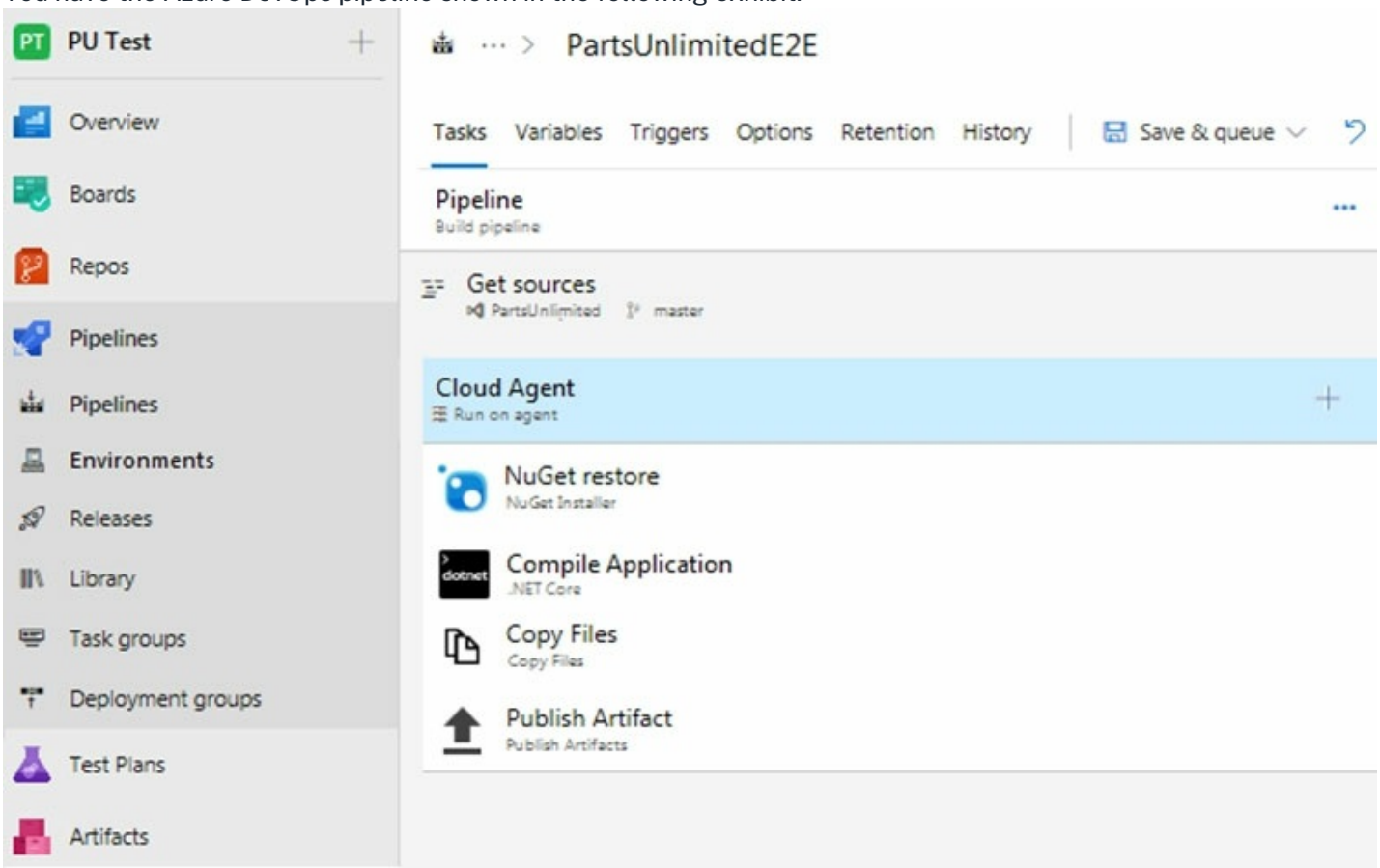
Reference:

https://docs.microsoft.com/en-us/azure/devops/artifacts/concepts/upstream-sources?view=azure-devops

https://docs.microsoft.com/en-us/azure/devops/artifacts/how-to/set-up-upstream-sources?view=azure-devops

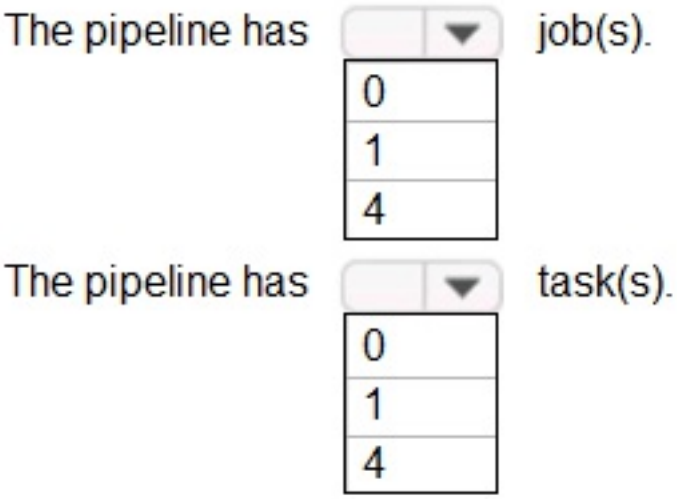
Question: 373 CertyIQ

HOTSPOT -
You have the Azure DevOps pipeline shown in the following exhibit.



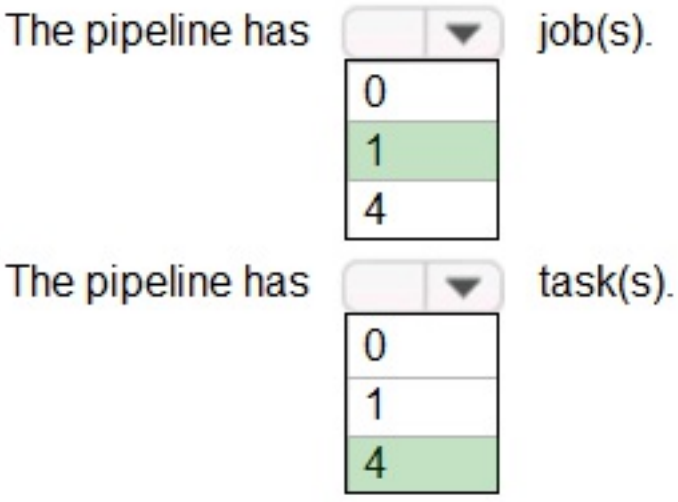
Use the drop-down menus to select the answer choice that completes each statement based on the information presented in the graphic.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:
Box 1: 1 -
The Cloud agent job only.

Box 2: 4 -
The pipeline has the four tasks: NuGet restore, Compile Application, Copy Files, and Publish Artifact.

Reference:
https://azuredevopslabs.com/labs/azuredevops/continuousintegration/

Question: 374 CertyIQ

SIMULATION -
You have an Azure function hosted in an App Service plan named az400-123456789-func1.
You need to configure az400-123456789-func1 to upgrade the functions automatically whenever new code is committed to the master branch of https:// github.com/Azure-Samples/functions-quickstart.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:
1. Open Microsoft Azure Portal
2. Log into your Azure account, select App Services in the Azure portal left navigation, and then select configure az400-123456789-func1.
3. On the app page, select Deployment Center in the left menu.
4. On the Build provider page, select Azure Pipelines (Preview), and then select Continue.
5. On the Configure page, in the Code section:
For GitHub, drop down and select the Organization, Repository, and Branch you want to deploy continuously.
6. Select Continue.
7. On the Test page, choose whether to enable load tests, and then select Continue.
8. Depending on your App Service plan pricing tier, you may see a Deploy to staging page. Choose whether to enable deployment slots, and then select Continue.
9. After you configure the build provider, review the settings on the Summary page, and then select Finish.

Reference:
https://docs.microsoft.com/en-us/azure/app-service/deploy-continuous-deployment

Question: 375 CertyIQ

DRAG DROP -
You need to use Azure Automation State Configuration to manage the ongoing consistency of virtual machine configurations.
Which five actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.
Select and Place:

Actions	Answer Area
Onboard the virtual machines to Azure Automation State Configuration.	
Check the compliance status of the node.	
Create a management group.	
Assign the node configuration.	
Compile a configuration into a node configuration.	
Upload a configuration to Azure Automation State Configuration.	
Assign tags to the virtual machines.	

Answer:

Actions	Answer Area
Onboard the virtual machines to Azure Automation State Configuration.	Upload a configuration to Azure Automation State Configuration.
Check the compliance status of the node.	Compile a configuration into a node configuration.
Create a management group.	Onboard the virtual machines to Azure Automation State Configuration.
Assign the node configuration.	Assign the node configuration.
Compile a configuration into a node configuration.	Check the compliance status of the node.
Upload a configuration to Azure Automation State Configuration.	
Assign tags to the virtual machines.	

Explanation:

1. Upload a configuration to Azure Automation.

2. Compile a configuration into a node configuration.

3. Onboard the VM to Azure Automation State Configuration.

4. Assign the node configuration.

5. Check the compliance status of the node.

Question: 376 CertyIQ

You are developing an application. The application source has multiple branches.
You make several changes to a branch used for experimentation.
You need to update the main branch to capture the changes made to the experimentation branch and override the history of the Git repository.
Which Git option should you use?

A. Rebase
B. Fetch
C. Merge
D. Push

Answer: A

Explanation:

The correct answer is **A. Rebase**.

Here's why:

The scenario requires integrating changes from an experimental branch into the main branch while overriding the history. This implies rewriting the commit history of the main branch to appear as if the experimental changes were directly made on it.

Rebasing is precisely designed for this purpose. It moves the entire feature branch (experimental branch in this case) onto the tip of another branch (the main branch). This creates a linear history and eliminates the need for a merge commit. Importantly, git rebase allows for amending commits during the process, effectively rewriting history. While it provides a clean history, it is important to note that it alters the commit history, which can be problematic if the main branch is shared with other developers. In such cases, you need to coordinate with them to avoid conflicts.

Merging, while it integrates changes, preserves the original history. It creates a new merge commit on the target branch, linking the feature branch's history. This is not suitable when the goal is to hide the experimental nature of the changes.

Fetching only downloads objects and refs from another repository but does not integrate them into your working directory. It doesn't modify the history.

Pushing simply sends local commits to a remote repository. It doesn't alter the commit history itself (although it can be used after a rebase to update the remote).

Therefore, rebasing provides the necessary functionality to incorporate the experimental changes while rewriting the history of the main branch, making it the ideal Git option.

Further research can be conducted on the following resources:

Git Rebase: https://git-scm.com/docs/git-rebase
Git Merge: https://git-scm.com/docs/git-merge
Git Fetch: https://git-scm.com/docs/git-fetch
Rewriting History: https://git-scm.com/book/en/v2/Git-Tools-Rewriting-History

Question: 377 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You use Azure Pipelines to build and test a React.js application.
You have a pipeline that has a single job.
You discover that installing JavaScript packages from npm takes approximately five minutes each time you run the pipeline.
You need to recommend a solution to reduce the pipeline execution time.
Solution: You recommend defining a container job that uses a custom container that has the JavaScript packages preinstalled.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The solution proposes using a container job with a custom container pre-installed with JavaScript packages to reduce pipeline execution time. This approach doesn't fully meet the goal of reducing package installation time for every pipeline run, though it does offer some benefits.

While using a container can improve consistency and potentially speed up some aspects, the primary bottleneck with npm package installation isn't the containerization itself, but rather the act of downloading and installing the packages. A custom container only pre-installs the packages once during container image creation. On each pipeline execution, unless steps are taken to preserve and reuse the node_modules folder across pipeline runs (such as caching), npm will still attempt to verify/re-install dependencies based on package.json and package-lock.json during the npm install step. This verification and potential download process can still take a significant amount of time.

A more effective solution is to leverage Azure Pipelines' caching mechanism. This involves caching the node_modules folder so that subsequent pipeline runs can reuse the installed packages, significantly reducing the time spent downloading and installing them. The containerization helps standardize the execution environment but isn't primarily addressing the download time itself, which is what the original question aims to solve.

Therefore, the solution as proposed (using a custom container without explicit caching strategies) does not fully meet the goal. Caching strategies would involve specifically saving the node_modules folder, and restoring it on subsequent runs.

Therefore, the answer is No.

Relevant links:

Azure Pipelines Caching: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/caching?view=azure-devops

Azure Pipelines Container Jobs: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/container-phases?view=azure-devops

Question: 378 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You use Azure Pipelines to build and test a React.js application.
You have a pipeline that has a single job.
You discover that installing JavaScript packages from npm takes approximately five minutes each time you run the pipeline.
You need to recommend a solution to reduce the pipeline execution time.
Solution: You recommend enabling pipeline caching.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

The proposed solution, enabling pipeline caching, directly addresses the problem of lengthy npm package installation times, making it the correct approach.

Pipeline caching in Azure Pipelines allows you to reuse the results of previous executions of a pipeline step. In this case, caching the node_modules folder, which contains the installed JavaScript packages, will significantly reduce the pipeline execution time. On subsequent runs, the pipeline can retrieve the cached node_modules folder instead of re-downloading and reinstalling all the packages from npm. This avoids the time-consuming network operations and processing associated with npm package installation.

The cache hit rate will determine the savings. If the dependencies in package.json rarely change, the cache hit rate will be high, leading to substantial time savings. Even if there are some changes, only the modified or newly added packages need to be downloaded.

This approach leverages a fundamental principle of optimizing CI/CD pipelines: minimizing redundant work. By caching dependencies, we reduce the time spent on repetitive tasks, allowing the pipeline to focus on building and testing the application.

Therefore, enabling pipeline caching is an effective way to reduce the pipeline execution time in this scenario.

Authoritative links:

Azure Pipelines Caching: Reduce build time with caching - Azure Pipelines | Microsoft Learn
Speed up your builds with caching in Azure DevOps - Microsoft DevOps Blog

Question: 379 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not

appear in the review screen.
You use Azure Pipelines to build and test a React.js application.
You have a pipeline that has a single job.
You discover that installing JavaScript packages from npm takes approximately five minutes each time you run the pipeline.
You need to recommend a solution to reduce the pipeline execution time.
Solution: You recommend enabling parallel jobs for the pipeline.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of enabling parallel jobs will not directly address the problem of slow npm package installation. Parallel jobs allow different parts of a pipeline or even different pipelines to run concurrently, thus reducing overall execution time. However, if the npm installation step remains the same within each job, its execution time will still be approximately five minutes. The npm install step is a sequential operation within a single job and is constrained by network speed, npm registry performance, and the number of dependencies.

To reduce the npm installation time, caching and artifact management strategies should be employed. Caching the node_modules directory allows subsequent pipeline runs to reuse previously downloaded packages, avoiding repeated downloads. Azure Pipelines provides built-in tasks for caching. Another approach is to use a private npm registry, which can provide faster and more reliable package retrieval compared to the public npm registry. Additionally, ensuring a fast and reliable network connection for the Azure DevOps agents can help improve download speeds.

Parallel jobs would be beneficial if other long-running tasks exist in the pipeline besides the npm installation. However, the bottleneck is specifically identified as the npm install step. Hence, addressing that step directly with caching or other optimization techniques is more suitable.

Therefore, enabling parallel jobs alone will not meet the goal of reducing the pipeline execution time primarily caused by the slow npm package installation.

Justification References:

Azure Pipelines Caching: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/caching?view=azure-devops
Azure Artifacts (Private npm Registry): https://learn.microsoft.com/en-us/azure/devops/artifacts/get-started-npm?view=azure-devops&tabs=new-nav
Azure Pipelines Parallel Jobs: https://learn.microsoft.com/en-us/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops

Question: 380 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You plan to create a release pipeline that will deploy Azure resources by using Azure Resource Manager templates. The release pipeline will create the following resources:
- Two resource groups
- Four Azure virtual machines in one resource group
- Two Azure SQL databases in other resource group

You need to recommend a solution to deploy the resources.
Solution: Create two standalone templates, each of which will deploy the resources in its respective group.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

The answer is **A. Yes**.

Justification:

The proposed solution, creating two standalone Azure Resource Manager (ARM) templates to deploy resources in their respective resource groups, effectively addresses the requirement. Here's why:

1. **Modularity and Scope:** ARM templates are designed for Infrastructure as Code (IaC). Creating separate templates for each resource group promotes modularity and aligns with the principle of defining infrastructure with a clear scope. This makes the templates easier to manage, update, and troubleshoot.
2. **Resource Group Deployment:** ARM templates are ideally suited for deploying resources to a specific resource group. The solution leverages this capability by creating a template for each resource group.
3. **Parallel Deployment Potential:** With two standalone templates, the deployments to each resource group can potentially be run in parallel within the release pipeline. This can significantly reduce the overall deployment time.
4. **Simplified Management:** Separating resources by resource group, as this solution provides, is a standard practice in Azure. This makes it easier to manage security, policies, and cost for distinct workloads. Having dedicated templates for each resource group ensures the integrity and consistency of the deployment process.
5. **ARM Template Capabilities:** ARM templates support parameters, variables, and outputs. These features facilitate customization and flexibility in each template. The templates can be used to create virtual machines and SQL databases in the specified resource groups as required.
6. **No inherent limitation:** The scenario doesn't impose constraints that necessitate a single template, so two separate templates is a valid approach.

Therefore, the proposed solution of creating two standalone ARM templates, one for each resource group, is a viable and effective method to deploy the required Azure resources.

Supporting Links:

Azure Resource Manager Templates: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/overview
Resource Groups: https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups

Question: 381 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series

contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You plan to create a release pipeline that will deploy Azure resources by using Azure Resource Manager templates. The release pipeline will create the following resources:
- Two resource groups
- Four Azure virtual machines in one resource group
- Two Azure SQL databases in other resource group

You need to recommend a solution to deploy the resources.
Solution: Create a single standalone template that will deploy all the resources.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of using a single, standalone Azure Resource Manager (ARM) template to deploy all specified resources (two resource groups, four VMs in one resource group, and two SQL databases in another) is not the best approach.

Here's why:

1. **Template Size and Complexity:** A single template for this entire deployment would become large and complex. ARM templates have limits on size and complexity. Exceeding these limits can lead to deployment failures or performance issues. Authoring and maintaining a single, large template becomes increasingly difficult.
2. **Resource Group Dependencies:** ARM templates have limited capabilities to manage resource group creation as part of other resources in the same template. While it's technically possible to create resource groups within the template, it's generally considered best practice to manage resource groups separately, particularly when the resource groups host distinct resources like VMs and SQL databases. Creating resource groups with dependent resources in the same deployment can lead to race conditions or circular dependency issues.
3. **Modularity and Reusability:** A monolithic template lacks modularity. It's not easy to reuse parts of the template for other deployments. In this scenario, splitting into linked or nested templates is preferable for better modularization.
4. **Scope:** ARM templates are most effective when used at the appropriate scope. Trying to manage multiple resource groups and resources within them can create unintended consequences and management overheads.
5. **Best Practices:** Azure best practices emphasize modularity and separation of concerns. Dividing deployments into smaller, manageable units makes it easier to troubleshoot, update, and redeploy specific components without affecting the entire infrastructure.

Alternatives include using linked templates, nested templates, or deployment scripts. Linked and nested templates allow breaking down the deployment into smaller, more manageable parts. Deployment scripts offer more flexibility in orchestrating deployments but add complexity.

In summary, while technically feasible, using a single, monolithic ARM template to deploy all the resources is not a recommended or efficient approach. It sacrifices modularity, maintainability, and scalability.

Supporting Documentation:

Azure Resource Manager template limits: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/template-syntax#template-format
Using linked and nested templates: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/linked-templates
Best practices for ARM templates: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/best-practices

Question: 382 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure DevOps project.
Your build process creates several artifacts.
You need to deploy the artifacts to on-premises servers.
Solution: You deploy a Kubernetes cluster on-premises. You deploy a Helm agent to the cluster. You add a Download Build Artifacts task to the deployment pipeline.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution involves deploying a Kubernetes cluster on-premises and using Helm for deploying artifacts. While Kubernetes and Helm are excellent tools for containerized applications, they don't directly address the requirement of deploying arbitrary artifacts to on-premises servers.

The Download Build Artifacts task in the deployment pipeline retrieves artifacts from the build process. The problem is that simply downloading the artifacts to the Kubernetes cluster doesn't guarantee they will be deployed to the specific on-premises servers required. Kubernetes primarily manages container deployments, not the deployment of general artifacts to any server within a network. It doesn't inherently know how to install or configure those artifacts on those servers.

If the artifacts were packaged into container images (e.g., Docker images) and deployed via Helm charts to Kubernetes, and the Kubernetes cluster was somehow deployed to be hosted as a service on those on-premises servers then this could be valid. However, that is not stated and there's no mechanism described for how those artifacts would reach their intended destination in the original scenario.

Therefore, this solution does not reliably and directly deploy the build artifacts to the intended on-premises servers. The downloaded artifacts would sit within the Kubernetes cluster, and additional steps would be needed to move and deploy those artifacts, potentially negating the benefits of the Kubernetes deployment strategy.

Relevant cloud computing concepts:

Artifact Repository: Azure DevOps Artifacts is a place to store packages and build outputs.
Deployment Pipeline: A set of automated processes to deploy code and artifacts.
Kubernetes: An orchestration system for managing containerized applications across a cluster of nodes. It is not inherently designed to deploy arbitrary files to arbitrary on-premises servers.
Helm: A package manager for Kubernetes, simplifying the deployment and management of Kubernetes

applications.
On-premises: Infrastructure located within the physical confines of an organization.

Authoritative links for further research:

Azure DevOps Artifacts: https://learn.microsoft.com/en-us/azure/devops/artifacts/start/artifacts-start?view=azure-devops
Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops
Kubernetes Documentation: https://kubernetes.io/docs/
Helm Documentation: https://helm.sh/docs/

Question: 383 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure DevOps project.
Your build process creates several artifacts.
You need to deploy the artifacts to on-premises servers.
Solution: You deploy a Docker build to an on-premises server. You add a Download Build Artifacts task to the deployment pipeline.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution does not fully meet the goal of deploying build artifacts to on-premises servers. Here's why:

1. **Docker Build vs. Artifact Deployment:** While deploying a Docker build to an on-premises server can be part of a deployment strategy, it's not directly equivalent to deploying all the artifacts produced by the build process. Docker builds create container images. The build artifacts may consist of scripts, configuration files, executables, or other resources in addition to the Docker image. The Docker image is itself an artifact, but it likely doesn't contain all the build artifacts.
2. **Download Build Artifacts Task:** The Download Build Artifacts task is appropriate for retrieving artifacts from a build pipeline. However, downloading them to the deployment pipeline agent doesn't automatically deploy them to the on-premises servers. You need subsequent tasks in the deployment pipeline to perform the actual deployment.
3. **Missing Deployment Steps:** The solution lacks the crucial steps of transferring the downloaded artifacts to the target on-premises servers and installing or configuring them in the appropriate locations. This might involve using tasks like Copy Files, SSH, PowerShell on Target Machines, or other deployment-specific tasks.
4. **Docker Considerations:** If the primary deployment strategy is based on Docker, the artifacts needed on-premises might be minimal if most application components are containerized. However, this assumes a complete shift to a container-based deployment model and management on the on-premise servers.

In summary, downloading the artifacts addresses part of the goal (retrieving the artifacts). Deploying Docker solves one specific type of deployment (the Docker image). However, it doesn't cover the full process of deploying all artifacts, or necessarily how the container and/or other artifacts are configured/deployed on the on-premises server(s) to allow a running application. Thus the solution is incomplete.

Relevant resources:

Azure Pipelines documentation
Download Build Artifacts task
Deployment strategies

Question: 384 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure DevOps project.
Your build process creates several artifacts.
You need to deploy the artifacts to on-premises servers.
Solution: You deploy an Azure self-hosted agent to an on-premises server. You add a Copy and Publish Build Artifacts task to the deployment pipeline.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The solution does not fully meet the goal because it only addresses artifact deployment but neglects the crucial aspect of targeting specific on-premises servers. While deploying a self-hosted agent on an on-premises server allows Azure DevOps to interact with that server, simply copying and publishing artifacts doesn't ensure they are deployed to the intended location or executed.

The Copy and Publish Build Artifacts task primarily focuses on storing artifacts within Azure DevOps for later use in releases. It doesn't intrinsically manage the deployment process to the target on-premises servers.

To effectively deploy to on-premises servers, you need to explicitly define the deployment steps in the deployment pipeline. This typically involves using tasks that copy the artifacts to the desired locations on the target servers and then execute necessary installation or update scripts. Example tasks include:

Copy Files over SSH: Securely copies files to a remote machine using SSH.
PowerShell on Target Machines: Executes PowerShell scripts on target machines.
Deploy to IIS Website: Deploys artifacts to an IIS website.

By incorporating these deployment-specific tasks and using the self-hosted agent, you can orchestrate the complete deployment process to your on-premises servers.

Therefore, using just the Copy and Publish Build Artifacts task is insufficient for achieving the stated goal of deploying artifacts to on-premises servers.

Supporting Documentation:

Agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser
Publish Build Artifacts task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/utility/publish-build-artifacts?view=azure-devops
Deployment groups: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/deployment-groups/?view=azure-devops (Another method for targeting on-premises machines.)

Question: 385 CertyIQ

You have a project in Azure DevOps named Project1. Project1 contains a pipeline that builds a container image named Image1 and pushes Image1 to an Azure container registry named ACR1. Image1 uses a base image stored in Docker Hub.
You need to ensure that Image1 is updated automatically whenever the base image is updated.
What should you do?

A. Enable the Azure Event Grid resource provider and subscribe to registry events.
B. Add a Docker Hub service connection to Azure Pipelines.
C. Create and run an Azure Container Registry task.
D. Create a service hook in Project1.

Answer: C

Explanation:

The correct answer is C: Create and run an Azure Container Registry task. Here's why:

Azure Container Registry (ACR) Tasks provide automated container image builds and updates based on a variety of triggers. One crucial trigger is a base image update. ACR Tasks can monitor a base image in a container registry (like Docker Hub) and, upon detecting an update, automatically rebuild and push the dependent image (Image1 in this scenario) to ACR1. This is exactly what the prompt requires.

Option A, enabling the Azure Event Grid resource provider and subscribing to registry events, is useful for reacting to various events in the Azure Container Registry, but it doesn't directly provide a mechanism to rebuild the image when the base image changes. It requires additional orchestration to trigger the build process. Event Grid is better suited for notifications and integration with other services upon registry events.

Option B, adding a Docker Hub service connection to Azure Pipelines, is important for authenticating and pulling the base image from Docker Hub during the build process. However, it doesn't automatically trigger a rebuild when the base image is updated. It's a prerequisite for accessing the base image, but not the solution for automated updates.

Option D, creating a service hook in Project1, can trigger actions in other services or applications based on events in Azure DevOps, such as pipeline completion or work item updates. While it could potentially be used in conjunction with other services to trigger an update, it doesn't natively handle base image update detection and automated rebuilds within the container registry itself.

ACR Tasks directly address the requirement by monitoring base image updates and automatically triggering a build, test, and push of the dependent image. This ensures Image1 is always based on the latest base image. A dedicated ACR Task configured with the appropriate base image dependency and trigger is the simplest and most efficient solution.

Here are some authoritative links for further research:

Azure Container Registry Tasks overview: https://learn.microsoft.com/en-us/azure/container-registry/container-registry-tasks-overview

Automate container image builds on base image update: https://learn.microsoft.com/en-us/azure/container-registry/container-registry-tutorial-base-image-update
ACR Task trigger on a base image update: https://learn.microsoft.com/en-us/azure/container-registry/container-registry-tasks-base-image

Question: 386 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure DevOps project.
Your build process creates several artifacts.
You need to deploy the artifacts to on-premises servers.
Solution: You deploy an Octopus Deploy server. You deploy a polled Tentacle agent to an on-premises server. You add an Octopus task to the deployment pipeline.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution suggests using Octopus Deploy with a polled Tentacle agent to deploy artifacts to on-premises servers from an Azure DevOps project. While Octopus Deploy is a capable deployment automation tool and can integrate with Azure DevOps, simply adding an Octopus task to the deployment pipeline and deploying a polled Tentacle agent does not fully guarantee meeting the stated goal efficiently and securely.

The key issue is the lack of detail on how the Octopus task is configured and what it deploys. The solution could work if the Octopus task is correctly configured to pick up the Azure DevOps artifacts, package them appropriately for Octopus Deploy, and then instruct the Tentacle agent to deploy them on the on-premises server. However, the question simply states "You add an Octopus task to the deployment pipeline," without specifying that the task is configured to actually deploy the Azure DevOps artifacts.

Furthermore, the security aspect is not sufficiently addressed. Polled Tentacles, while suitable for certain scenarios, require the Tentacle agent to periodically poll the Octopus Deploy server for deployment instructions. This might not be the most secure option depending on the network configuration and security policies. A better alternative might be a listening Tentacle, where the Octopus server initiates the connection.

In summary, while Octopus Deploy can be a solution, the described implementation is too vague and lacks sufficient detail to ensure the artifacts are actually deployed correctly and securely. Therefore, the answer is **No**. The solution is incomplete and relies on implicit configurations that are not defined in the problem statement.

Authoritative Links:

Octopus Deploy Documentation: https://octopus.com/docs (Specifically, documentation on Tentacles and Azure DevOps integration)
Azure DevOps Pipelines: https://docs.microsoft.com/en-us/azure/devops/pipelines/?view=azure-devops

Question: 387 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You plan to create a release pipeline that will deploy Azure resources by using Azure Resource Manager templates. The release pipeline will create the following resources:
- Two resource groups
- Four Azure virtual machines in one resource group
- Two Azure SQL databases in other resource group

You need to recommend a solution to deploy the resources.
Solution: Create a main template that will deploy the resources in one resource group and a nested template that will deploy the resources in the other resource group.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution suggests using a main template for one resource group and a nested template for the other. While nested templates are valid in ARM templates, this approach doesn't directly address the problem's need to deploy resources into two distinct resource groups using a single release pipeline, with specific requirements for where VMs and SQL databases are created.

The main issue is that ARM templates are typically deployed within a single resource group. While you can create a resource group within a template, and then deploy into it using a nested or linked template, this complicates the workflow and doesn't inherently solve the problem of deploying across two pre-existing resource groups easily using a single pipeline. Using two templates might be simpler and more directly aligned with what we are trying to achieve.

Therefore, the solution doesn't efficiently or directly meet the goal of deploying resources into two separate, potentially pre-existing resource groups with the resource distribution specified. It is therefore, not a well optimized approach to creating the required architecture.

While technically possible, splitting the infrastructure definition into one main template and a nested template, each managing the resources of one RG is not optimal. There's no intrinsic architectural advantage to doing this, and makes the overall management slightly more complicated than using two separate ARM templates.

Therefore, the proposed solution is not a good solution for deploying the specified resources to different resource groups.

References:

Use linked and nested templates when deploying Azure resources: Microsoft Documentation
Azure Resource Manager template structure: Microsoft Documentation

Question: 388 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You plan to create a release pipeline that will deploy Azure resources by using Azure Resource Manager templates. The release pipeline will create the following resources:
- Two resource groups
- Four Azure virtual machines in one resource group
- Two Azure SQL databases in other resource group

You need to recommend a solution to deploy the resources.
Solution: Create a main template that has two linked templates, each of which will deploy the resources in its respective group.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

The solution meets the goal. Using a main template with linked templates is a valid and effective approach for deploying Azure resources across multiple resource groups. The main template acts as an orchestrator, calling the linked templates to handle the deployment of resources within their respective resource groups. This modular approach promotes reusability and maintainability. Each linked template focuses on deploying resources within a specific resource group, making the overall deployment easier to manage and troubleshoot. This approach also allows for easier updates to individual parts of the infrastructure without affecting other resource groups. In this scenario, one linked template can deploy the four virtual machines in one resource group, and the other linked template can deploy the two Azure SQL databases in the other resource group. The main template handles the creation of the two resource groups as well, or this can be done separately depending on requirements. This structure simplifies the deployment process and ensures that resources are created in the correct locations.

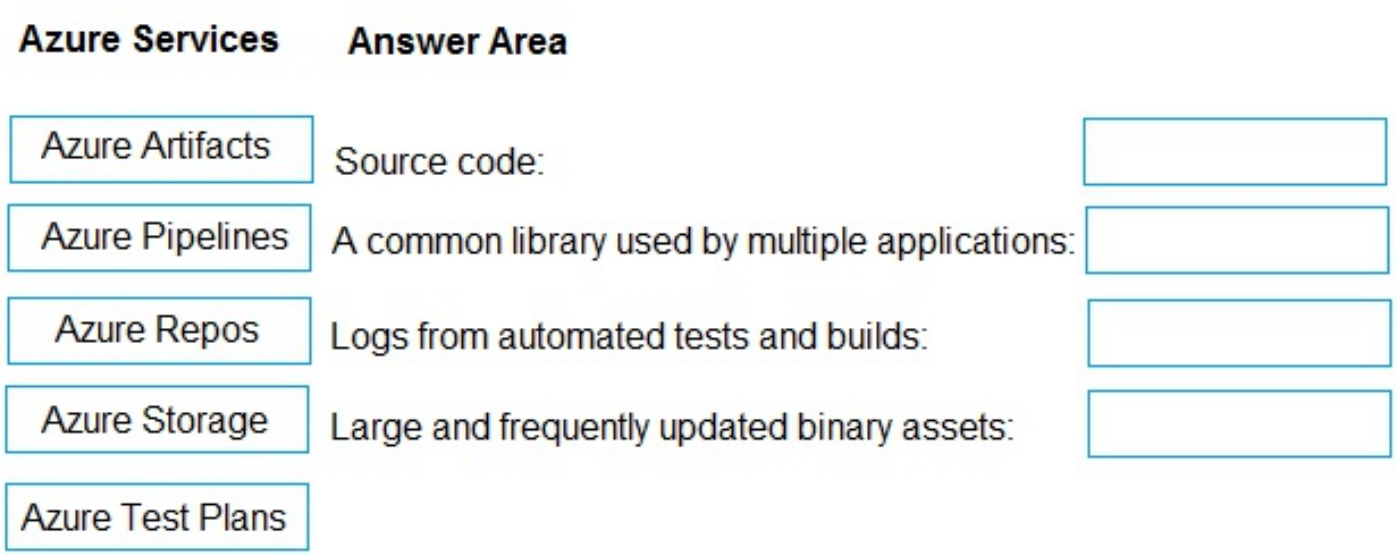
Here are a few authoritative links for further research:

Using linked and nested templates to deploy Azure resources: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/linked-templates
Azure Resource Manager template structure: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/template-structure

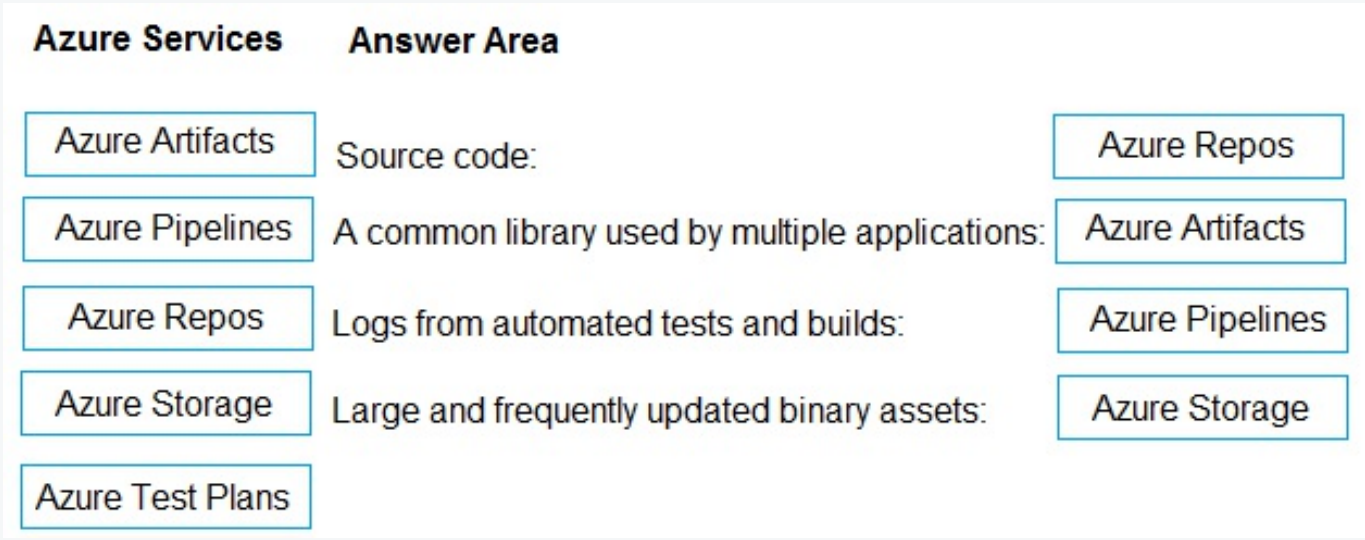
Question: 389 CertyIQ

DRAG DROP -
You are building an application that has the following assets:
- Source code
- Logs from automated tests and builds
- Large and frequently updated binary assets
- A common library used by multiple applications

Where should you store each asset? To answer, drag the appropriate Azure services to the correct assets. Each service may be used once. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:
Box 1: Azure Repos -

Box 2: Azure Artifacts -
Use Azure Artifacts to create, host, and share packages with your team.

Box 3: Azure Pipelines -
In the pipeline view you can see all the stages and associated tests. The view provides a summary of the test results

Box 4: Azure Storage -

Reference:
https://docs.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos
https://azure.microsoft.com/en-us/services/devops/artifacts/ https://docs.microsoft.com/en-us/azure/devops/pipelines/test/review-continuous-test-results-after-build

Question: 390 CertyIQ

You plan to share packages that you wrote, tested, validated, and deployed by using Azure Artifacts.
You need to release multiple builds of each package by using a single feed. The solution must limit the release of packages that are in development.

What should you use?

A. local symbols
B. views
C. global symbols
D. upstream sources

Answer: B

Explanation:

The correct answer is **B. Views**.

Azure Artifacts views are designed specifically to address the scenario of managing and promoting package releases across different stages of development, testing, and production. They provide a way to create a logical segregation of packages within a single feed, enabling controlled access based on the maturity of the package.

Here's a detailed justification:

Views for Package Promotion: Views enable you to promote packages through different stages of your development pipeline. For instance, you can have a @local view where packages are initially published after building. Once tested and validated, packages can be promoted to a @release view, making them available for consumption in production environments.

Control Over Package Availability: By using views, you can restrict access to packages that are still under development or testing. Developers can access the @local view, while the production environment only has access to the @release view. This prevents the accidental deployment of unstable or untested packages.

Single Feed for Multiple Releases: Views work within a single feed, which satisfies the requirement of managing multiple builds of each package within the same feed. Instead of creating separate feeds for each release stage, you can use views to logically isolate them.

'@local' (Default View): Azure Artifacts provides a default @local view, which contains all packages directly published to the feed. When you publish a new package, it initially appears in this view. Packages here are considered potentially unstable and should be tested.

'@prerelease' and '@release' (Example Views): You can create additional views like @prerelease for packages undergoing testing and @release for production-ready packages. Package promotion involves changing the view a specific package version is associated with.

Immutable Views for Consistency: Views can be immutable. This means once a package version is added to an immutable view (like @release), it cannot be removed or changed. This ensures consistency and reproducibility for your production environment.

Upstream Sources and Views Combination: Views can also be used in conjunction with upstream sources. For example, your @release view could have upstream sources configured to pull packages from public registries like NuGet.org or npmjs.com, ensuring that only vetted and approved packages are used in production.

Why other options are incorrect:

A. Local Symbols: Local symbols are related to debugging and providing detailed information about the code during debugging sessions. They don't play a role in package promotion or controlling access to packages based on their maturity.

C. Global Symbols: The concept of global symbols is not directly relevant to Azure Artifacts and package

management.

D. Upstream Sources: Upstream sources allow your feed to consume packages from public registries. While helpful for managing dependencies, they don't directly address the requirement of releasing multiple builds and limiting the release of development packages. Upstream sources can be used in conjunction with views.

Authoritative Links:

Azure Artifacts Views: https://learn.microsoft.com/en-us/azure/devops/artifacts/concepts/views?view=azure-devops
Promote Packages Using Views: https://learn.microsoft.com/en-us/azure/devops/artifacts/concepts/promote-packages?view=azure-devops

Question: 391 CertyIQ

You have a project in Azure DevOps named Project1. Project1 contains a build pipeline named Pipe1 that builds an application named App1.
You have an agent pool named Pool1 that contains a Windows Server 2019-based self-hosted agent. Pipe1 uses Pool1.
You plan to implement another project named Project2. Project2 will have a build pipeline named Pipe2 that builds an application named App2.
App1 and App2 have conflicting dependencies.
You need to minimize the possibility that the two build pipelines will conflict with each other. The solution must minimize infrastructure costs.
What should you do?

A. Add another self-hosted agent.
B. Add a Docker Compose task to the build pipelines.
C. Change the self-hosted agent to use Red Hat Enterprise Linux (RHEL) 8.
D. Create two container jobs.

Answer: D

Explanation:

The correct answer is **D. Create two container jobs.**

Here's why:

Container jobs address the conflicting dependencies problem by providing isolated environments for each build. This ensures that the dependencies required for App1's build in Pipe1 won't interfere with the dependencies required for App2's build in Pipe2, and vice versa. Each container acts as a clean slate, resolving the conflicts without requiring separate agents or OS configurations.

Using containers minimizes infrastructure costs because you can utilize the existing Pool1 agent more efficiently. Instead of adding another agent (option A), which incurs additional hardware and maintenance costs, or changing the operating system (option C), which may not completely resolve the dependency conflicts and could introduce compatibility issues, container jobs leverage the same agent while maintaining isolation. Docker Compose (option B) manages multi-container applications, which might be over-engineering for this scenario, and it doesn't directly address the root cause of the build dependency conflict as effectively as container jobs.

Containerization is a core concept in modern DevOps, enabling application isolation and portability. Azure DevOps supports container jobs directly, allowing you to specify Docker images as part of your build definition. Each build step runs within its own container, ensuring a consistent and isolated environment. This promotes repeatability and reduces the "works on my machine" problem.

By using container jobs, you effectively isolate the conflicting dependencies, prevent build failures due to dependency clashes, and optimize resource utilization, all while minimizing infrastructure costs.

Further reading on Azure DevOps container jobs:

Use a container as a build agent
Run a job in a container

Question: 392 CertyIQ

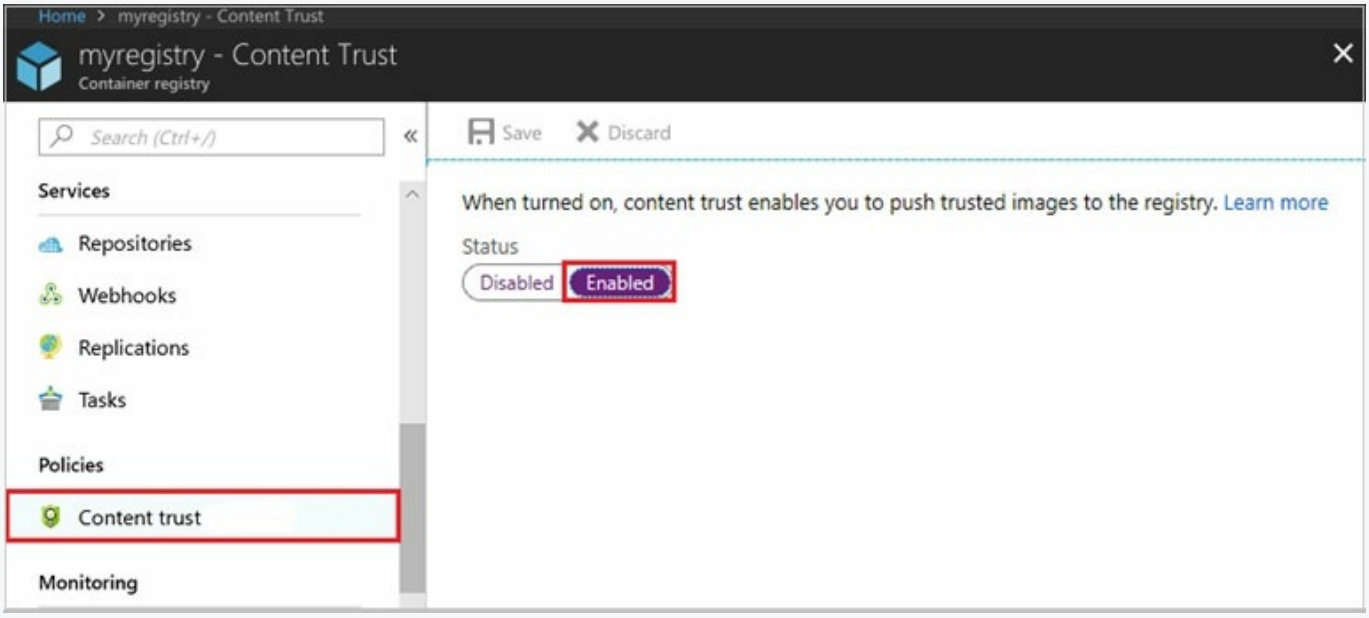
SIMULATION -
You plan to store signed images in an Azure Container Registry instance named az4009940427acr1.
You need to modify the SKU for az4009940427acr1 to support the planned images. The solution must minimize costs.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:

1. Open Microsoft Azure Portal, and select the Azure Container Registry instance named az4009940427acr1.
2. Under Policies, select Content Trust > Enabled > Save.



Reference:
https://docs.microsoft.com/en-us/azure/container-registry/container-registry-content-trust

Question: 393 CertyIQ

You manage build pipelines and deployment pipelines by using Azure DevOps.
Your company has a team of 500 developers. New members are added continually to the team.
You need to automate the management of users and licenses whenever possible.
Which task must you perform manually?

A. modifying group memberships
B. adding users
C. assigning entitlements
D. procuring licenses

Answer: D

Explanation:

The correct answer is **D. procuring licenses**. Here's a detailed justification:

While Azure DevOps offers several features for automating user management, group memberships, and entitlement assignments, the actual procurement of licenses remains a manual step, especially concerning billing and organizational agreements.

Here's a breakdown:

Modifying group memberships (A): Azure DevOps integrates seamlessly with Azure Active Directory (Azure AD). You can create Azure AD groups and map them to Azure DevOps groups. Adding or removing users from the Azure AD group automatically updates their membership in Azure DevOps, hence automating group management.

Adding users (B): Similarly, user addition can be automated by inviting users to the Azure AD tenant linked to your Azure DevOps organization. This will add the user to the organization.

Assigning Entitlements (C): Azure DevOps provides features to manage access levels such as Basic, Stakeholder, or Visual Studio Subscriber. Group rules can be defined to automatically assign entitlements based on group membership. This automates the process of granting permissions to users.

Procuring Licenses (D): Despite Azure DevOps's automation capabilities, procuring licenses involves understanding organizational needs, estimating the required number of licenses, and then purchasing the appropriate licensing plan through the Azure portal or other relevant channels. This process typically requires human intervention to assess requirements, manage billing agreements, and confirm the licensing tier. While you can automate assigning purchased licenses, the initial purchase requires manual intervention.

Thus, procuring licenses is the task that requires manual intervention.

Authoritative Links:

Azure DevOps Pricing: https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/
Manage users and access in Azure DevOps: https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/add-organization-users?view=azure-devops
About access levels: https://learn.microsoft.com/en-us/azure/devops/organizations/security/access-levels?view=azure-devops
Group rules: https://learn.microsoft.com/en-us/azure/devops/organizations/security/group-rules?view=azure-devops

Question: 394 CertyIQ

HOTSPOT -
Your company uses Team Foundation Server 2013 (TFS 2013).
You plan to migrate to Azure DevOps.
You need to recommend a migration strategy that meets the following requirements:
✑ Preserves the dates of Team Foundation Version Control changesets
✑ Preserves the changed dates of work items revisions
Minimizes migration effort -
✑ Migrates all TFS artifacts
What should you recommend? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

On the TFS server:

- Install the TFS Java SDK.
- Upgrade TFS to the most recent RTW release.
- Upgrade to the most recent version of PowerShell Core.

To perform the migration:

- Copy the assets manually.
- Use public API-based tools.
- Use the TFS Database Import Service.
- Use the TFS Integration Platform.

Answer:

Answer Area

On the TFS server:

- Install the TFS Java SDK.
- **Upgrade TFS to the most recent RTW release.**
- Upgrade to the most recent version of PowerShell Core.

To perform the migration:

- Copy the assets manually.
- Use public API-based tools.
- **Use the TFS Database Import Service.**
- Use the TFS Integration Platform.

Explanation:

Box 1: Upgrade TFS to the most recent RTM release.

One of the major prerequisites for migrating your Team Foundation Server database is to get your database schema version as close as possible to what is currently deployed in Azure Devops Services.

Box 2: Use the TFS Database Import Service

In Phase 3 of your migration project, you will work on upgrading your Team Foundation Server to one of the supported versions for the Database Import Service in
Azure Devops Services.

Question: 395

CertyIQ

Your company is building a new solution in Java.

The company currently uses a SonarQube server to analyze the code of .NET solutions.
You need to analyze and monitor the code quality of the Java solution.
Which task types should you add to the build pipeline?

A. Octopus
B. Chef
C. CocoaPods
D. Gradle

Answer: D

Explanation:

Here's a detailed justification for why Gradle is the correct answer for integrating SonarQube code analysis into a Java build pipeline within Azure DevOps:

The core requirement is to analyze and monitor Java code quality using an existing SonarQube server during the build process. This necessitates integrating a build tool that can effectively manage the Java project and integrate with SonarQube's analysis capabilities.

Gradle is a powerful and flexible open-source build automation tool widely used in Java development. It excels at managing dependencies, compiling code, running tests, and packaging applications. More importantly, it boasts a rich plugin ecosystem, including a dedicated plugin specifically designed for seamless integration with SonarQube.

Using the Gradle SonarQube plugin, you can configure your build pipeline to automatically trigger code analysis on each build. This plugin handles tasks like preparing the analysis environment, executing SonarQube's scanner, and uploading the analysis results to the SonarQube server. This direct integration ensures that code quality checks are consistently performed as part of the development lifecycle.

Options A, B, and C are not suitable for Java code analysis and SonarQube integration:

Octopus Deploy: Is a deployment automation server, primarily focused on deploying applications to various environments, not code analysis.
Chef: Is an infrastructure automation tool used for managing and configuring servers, not directly involved in code quality analysis or build processes.
CocoaPods: Is a dependency manager for Swift and Objective-C projects, primarily used in iOS and macOS development, not Java.

Therefore, Gradle is the logical choice because it's a Java-centric build tool offering direct and effective integration with SonarQube for automated code analysis within the Azure DevOps pipeline. It allows to automatically run a SonarQube analysis as part of the standard build process, providing crucial insights into code quality, security vulnerabilities, and potential maintainability issues.

Authoritative Links:

Gradle SonarQube Plugin: https://docs.sonarsource.com/sonarcloud/usage/ci-based-analysis/sonarscanner-for-gradle/
Azure DevOps Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops
SonarQube: https://www.sonarsource.com/products/sonarqube/

Question: 396 CertyIQ

Your company is building a new solution in Java.

The company currently uses a SonarQube server to analyze the code of .NET solutions.
You need to analyze and monitor the code quality of the Java solution.
Which task types should you add to the build pipeline?

A. Maven
B. CocoaPods
C. Xcode
D. Gulp

Answer: A

Explanation:

The correct answer is A, Maven. Here's why:

SonarQube is a code quality analysis platform, and to integrate a Java project with SonarQube for code analysis within an Azure DevOps pipeline, you need a build tool that manages dependencies, compilation, and test execution. Maven is a widely used build automation tool primarily used for Java projects. It provides a standardized way to build, test, and package Java applications, making it a natural fit for integration with SonarQube.

The Maven task in Azure DevOps helps execute Maven goals, such as compiling the code, running tests, and generating reports. These reports, including code coverage and style issues, can then be analyzed by the SonarQube server. By adding a Maven task, you essentially trigger the Maven build process as part of your CI/CD pipeline. This allows SonarQube to receive the necessary data about your Java code to perform its analysis.

Options B, C, and D are incorrect because they are specific to different technology stacks. CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects (iOS/macOS). Xcode is an integrated development environment (IDE) specifically for Apple platforms. Gulp is a JavaScript task runner commonly used for front-end web development. These tools are irrelevant to building and analyzing a Java solution with SonarQube. A Java solution would not use CocoaPods, Xcode, or Gulp for building and dependency management.

Therefore, to analyze a Java project's code quality using SonarQube in Azure DevOps, the most appropriate task type to add to the build pipeline is Maven.

For further research, consult the official documentation:

Azure DevOps Maven Task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/build/maven?view=azure-devops
SonarQube Documentation: https://docs.sonarsource.com/
Apache Maven: https://maven.apache.org/

Question: 397 CertyIQ

You have an Azure subscription named Subscription1 that contains a custom Azure policy named Policy1. Policy1 is an audit policy that monitors naming convention compliance for the resources deployed to Subscription1.
You have a pipeline named Pipeline1 in Azure Pipelines. Pipeline1 deploys Azure Resource Manager (ARM) resources to Subscription1.
You need to ensure that the resources deployed by Pipeline1 comply with Policy1.
What should you add to Pipeline1?

A. a pre-deployment task that runs a security and compliance assessment
B. a post-deployment task that runs a security and compliance assessment

C. an ARM template deployment task to assign Policy1 to Subscription1

D. an ARM template deployment task to deploy Policy1 to Subscription1

Answer: A

Explanation:

The correct answer is **A. a pre-deployment task that runs a security and compliance assessment.** Here's why:

The goal is to ensure resources deployed by the pipeline adhere to the Policy1 naming conventions before they are actually deployed to Azure. This is best achieved by proactive validation, preventing non-compliant deployments rather than auditing after the fact.

A pre-deployment task that includes a security and compliance assessment can analyze the ARM templates or the planned deployment configuration against the policy definitions. This allows you to identify violations and stop the pipeline execution if resources are non-compliant. This proactive approach ensures that only compliant resources are provisioned, reducing remediation efforts and maintaining governance. Implementing the compliance check before deployment enforces governance at the pipeline level, shifting security and compliance "left" in the development lifecycle, a key principle of DevOps. This prevents the deployment of resources that violate the organization's policies.

Option B (post-deployment) is incorrect because it only identifies violations after the resources are deployed. While auditing is valuable, it doesn't prevent the deployment of non-compliant resources, which increases the risk of security vulnerabilities or operational issues.

Options C and D are incorrect because Policy1 is already created and applied to the subscription. Re-deploying or assigning it is redundant and doesn't directly address the requirement of validating resource compliance during the pipeline execution. Those options are more related to initial policy setup, not runtime enforcement during deployment.

Therefore, a pre-deployment security and compliance assessment directly addresses the requirement by proactively enforcing Policy1's naming convention rules before any Azure resources are provisioned.

For further research, consider the following:

Azure Policy Evaluation Details: https://learn.microsoft.com/en-us/azure/governance/policy/concepts/evaluation-details
Azure Pipelines Tasks: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/?view=azure-devops
Shift-Left Security: A search on shift-left security principles in DevOps.

Question: 398 CertyIQ

HOTSPOT -
You plan to use Desired State Configuration (DSC) to maintain the configuration state of virtual machines that run Windows Server.
You need to perform the following:
✑ Install Internet Information Services (IIS) on the virtual machines.
✑ Update the default home page of the IIS web server.
How should you configure the DSC configuration file? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

```
Configuration WebServerConfig {
Import-DscResource -ModuleName PsDesiredStateConfiguration
Node 'localhost' {
        [ Service | WindowsFeature | WindowsOptionalFeature | WindowsProcess ] WebServer {
              Ensure = "Present"
              Name = "Web-Server"
        }

        [ Archive | File | Package | Script ] DefaultHomePage {
              Ensure = 'Present'
              SourcePath = '\\server1
\DSCResources\web\index.htm'
              DestinationPath = 'c:\inetpub\wwwroot'
         }
    }
}
```

Answer:

Answer Area

```
Configuration WebServerConfig {
Import-DscResource -ModuleName PsDesiredStateConfiguration
Node 'localhost' {
        [Service / WindowsFeature / WindowsOptionalFeature / WindowsProcess] WebServer {
              Ensure = "Present"
              Name = "Web-Server"
        }

        [Archive / File / Package / Script] DefaultHomePage {
              Ensure = 'Present'
              SourcePath = '\\server1
\DSCResources\web\index.htm'
              DestinationPath = 'c:\inetpub\wwwroot'
         }
    }
}
```

Explanation:

Box 1: WindowsFeature -

Example:

Configuration WebsiteTest

Import the module that contains the resources we're using.

Import-DscResource -ModuleName PsDesiredStateConfiguration

The Node statement specifies which targets this configuration will be applied to.

Node 'localhost'

The first resource block ensures that the Web-Server (IIS) feature is enabled.

WindowsFeature WebServer

Ensure = "Present"

Name = "Web-Server"

Box 2: File -

Example continued:

The second resource block ensures that the website content copied to the website root folder.

File WebsiteContent

Ensure = 'Present'

SourcePath = 'c:\test\index.htm'

DestinationPath = 'c:\inetpub\wwwroot'

Reference:
https://docs.microsoft.com/en-us/powershell/scripting/dsc/quickstarts/website-quickstart

Question: 399 CertyIQ

You have a project in Azure DevOps.
You need to push notifications about pull requests to a Microsoft Teams channel. The solution must minimize development effort.
What should you do?

A. Install the Azure Pipelines app for Teams and configure a subscription to receive notifications in the channel.
B. Use Azure Automation to connect to the Azure DevOps REST API and send messages to Teams.
C. Install the Azure Repos app for Teams and configure a subscription to receive notifications in the channel.
D. Use an Azure function to connect to the Azure DevOps REST API and send messages to Teams.

Answer: C

Explanation:

The correct answer is C: Install the Azure Repos app for Teams and configure a subscription to receive notifications in the channel. Here's why:

Azure DevOps offers direct integrations with Microsoft Teams to streamline collaboration. Specifically, the Azure Repos app is designed to facilitate communication around code repositories, including pull requests. Installing this app is a low-code, out-of-the-box solution.

Option A, the Azure Pipelines app, focuses on build and release pipelines, not code repositories or pull requests directly. While build status might indirectly relate, it doesn't directly address pull request notifications as the core requirement.

Options B and D involve significantly more development effort. Using Azure Automation or Azure Functions requires writing code to interact with the Azure DevOps REST API, authenticate, parse the pull request data, and then format and send the messages to Teams. This increases complexity and maintenance overhead. The goal is to minimize development effort, making these less ideal.

By installing the Azure Repos app, you can configure a subscription within the Teams channel to receive specific notifications related to pull requests, such as when a new pull request is created, updated, or merged. This configuration is typically done through a user-friendly interface without requiring any custom code. This aligns perfectly with minimizing development effort while achieving the desired outcome. The Azure Repos app provides pre-built integrations and workflows that are purpose-built for managing code-related discussions within Teams. It uses webhooks under the hood (though the user does not need to understand or configure these) to push notifications from the Azure DevOps repository to the Teams channel in near real-time when the designated events occur (e.g., new pull request creation).

For more information on Azure Repos and Teams integration, refer to the official Microsoft documentation:

Integrate Azure Repos with Microsoft Teams: https://learn.microsoft.com/en-us/azure/devops/boards/integrations/boards-teams?view=azure-devops
Azure Repos app for Microsoft Teams: https://learn.microsoft.com/en-us/azure/devops/boards/integrations/boards-teams?view=azure-devops#azure-repos-app

Question: 400 CertyIQ

HOTSPOT -
You are creating a YAML-based Azure pipeline to deploy an Azure Data Factory instance that has the following requirements:
- If a Data Factory instance exists already, the instance must be overwritten.
- No other resources in a resource group named Fabrikam must be affected.

How should you complete the code? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

```
steps:

- task: AzureResourceManagerTemplateDeployment@3

  inputs:

    deploymentScope: 'Resource Group'

    azureResourceManagerConnection: 'Fabrikam Corporate(a41fb3ed-a2aa-42f0-a7ac-8fcc6ef0c5db)'

    subscriptionId: 'a41de0ed-a2aa-42f0-a7ac-8fcc6ef0c5db'

    action:
```

- 'Create Or Update Resource Group'
- 'Select Resource Group'
- 'Start'

```
  resourceGroupName: 'Fabrikam'

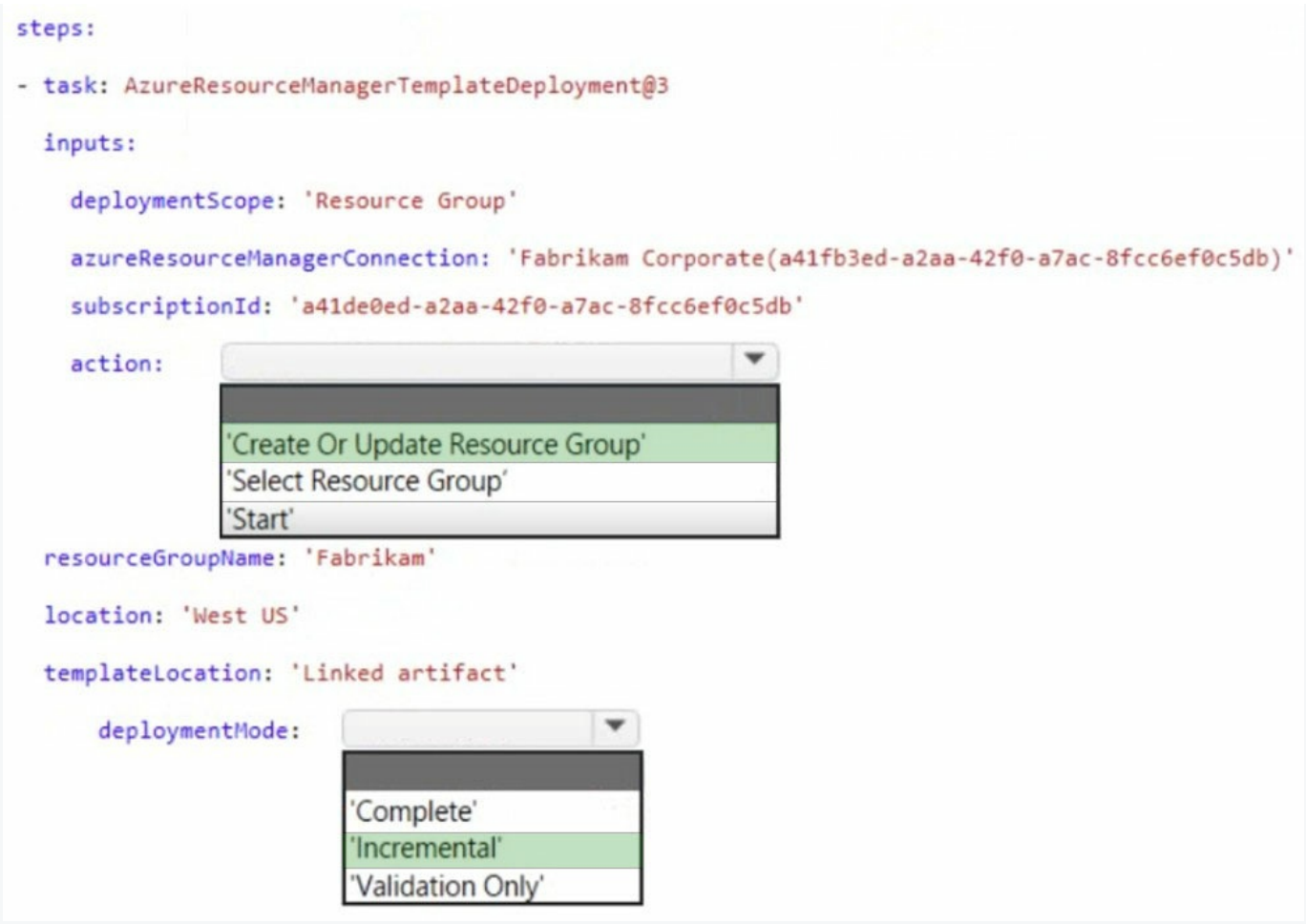
  location: 'West US'

  templateLocation: 'Linked artifact'

      deploymentMode:
```

- 'Complete'
- 'Incremental'
- 'Validation Only'

Answer:



Question: 401 CertyIQ

You have an Azure DevOps project that produces Node Package Manager (npm) packages. Multiple projects consume the packages.
You need to configure Azure Artifacts to ensure that both the latest and pre-release versions of the packages are available for consumption.
What should you do?

A. Create two feed views named @prerelease and @release, Set @release as the default view. Configure a release pipeline that tags the packages as release after successful testing.

B. Create a feed view named @prerelease. Configure a release pipeline that tags the packages as release after successful testing.

C. Create two feed views named @prerelease and @default. Configure a release pipeline that promotes a package to the @default view after successful testing.

D. Create two feed views named @prerelease and @release. Set @release as the default view. Configure a release pipeline that promotes a package to the @release view after successful testing.

Answer: D

Explanation:

The correct answer is D because it provides the necessary mechanisms for managing and consuming both pre-release and release versions of npm packages within Azure Artifacts.

Here's a breakdown of the justification:

1. **Feed Views:** Azure Artifacts feed views are essential for organizing and controlling the visibility of packages. Creating two views, @prerelease and @release, allows you to separate pre-release versions from stable releases.

2. **Default View:** Setting @release as the default view ensures that consumers by default receive the stable versions of the packages. This avoids accidental usage of potentially unstable pre-release versions.

3. **Release Pipeline Promotion:** The most crucial part is the release pipeline's role. After successful testing of a package, it should be promoted to the @release view. Promotion essentially means the package is now considered stable and available for general consumption. Packages available in the @prerelease view are specifically targeted at early adopters and testers who can tolerate instability.

Options A and B are incorrect because they only address creating the @prerelease view or tagging. Tagging is useful metadata, but it doesn't intrinsically control visibility or promote a package to a stable state like promoting to a dedicated @release view does. Tagging is not sufficient for visibility control.

Option C is incorrect. Although it creates @prerelease and @default views, it assumes @default will automatically represent the production releases. In fact, the default view name is @release. Changing to @default as a view isn't recommended because it diverges from the conventional and expected use case. Moreover, the standard practice utilizes the @release view name for finalized versions.

In summary, the combination of dedicated feed views for pre-release and release versions, with the @release view set as default, and a release pipeline promoting tested packages to the @release view provides a robust and organized way to manage npm package versions within Azure Artifacts. This allows consumers to consistently access the desired version (either stable releases from @release or pre-release versions from @prerelease).

Authoritative links for further research:

Azure Artifacts views: https://learn.microsoft.com/en-us/azure/devops/artifacts/concepts/feeds/views?view=azure-devops
Promote packages to a view: https://learn.microsoft.com/en-us/azure/devops/artifacts/feeds/views?view=azure-devops#promote-packages-to-a-view

Question: 402 CertyIQ

You have an Azure subscription that contains the resources shown in the following table.

Name	Type
Feed1	Azure Artifacts feed
Project1	Project in Azure DevOps

Project produces npm packages that are published to Feed1. Feed1 is consumed by multiple projects.
You need to ensure that only tested packages are available for consumption. The solution must minimize development effort.
What should you do?

A. Create a feed view named @release and set @release as the default view. After the npm packages test successfully, configure a release pipeline that promotes a package to the @release view.

B. Create a feed view named @release and set @release as the default view. After the npm packages test successfully, configure a release pipeline that tags the packages as release.

C. Create a feed view named @default. After the npm packages test successfully, configure a release pipeline that tags the packages as release.

D. Create a feed view named @default. After the npm packages test successfully, configure a release pipeline that promotes a package to the @default view.

Answer: A

Question: 403 CertyIQ

Your company has an Azure DevOps project that produces Node Package Manager (npm) packages. Multiple projects consume the packages.
You need to minimize the amount of disk space used by older packages in Azure Artifacts.
What should you modify?

A. the retention settings of the project's release
B. the retention settings of the project's pipeline
C. the retention settings of the project's tests
D. the retention settings of the company pipeline

Answer: B

Explanation:

The correct answer is **B. the retention settings of the project's pipeline**. Here's why:

The goal is to minimize disk space used by older npm packages stored in Azure Artifacts. Azure Artifacts are closely tied to Azure Pipelines through the publish and consume activities. Specifically, pipeline runs are responsible for publishing these npm packages to the Azure Artifacts feed. The retention policies for pipelines directly affect how long the artifacts produced by those pipelines (including the npm packages) are retained. By reducing the retention period for pipeline runs, you are indirectly specifying how long the associated packages remain available.

When a pipeline run is deleted due to a retention policy, its associated artifacts (if they are governed by the pipeline retention) will also be deleted. Therefore, adjusting the retention settings of the pipeline run associated with publishing the npm packages controls the lifecycle of those packages within Azure Artifacts.

Option A (release retention) isn't the primary driver here. While releases can consume packages from Artifacts, they don't directly govern the storage and lifecycle of the packages within Artifacts itself. Option C (test retention) has no bearing on Artifact storage since tests are separate concerns from package management. Option D (company pipeline retention) is too broad if you only want to affect the npm packages. The company's pipeline may involve different parts; you want to be specific to the npm package publishing pipeline.

Here's a breakdown of why pipeline retention is the relevant setting:

1. **Pipeline-Artifacts relationship:** Pipelines generate artifacts, and artifacts are stored based on pipeline retention settings.
2. **npm package publishing:** A pipeline publishes the npm packages to Artifacts.
3. **Controlling storage duration:** Modifying the pipeline's retention policy directly affects how long these packages remain available, thereby minimizing disk space usage.
4. **Specificity:** Focusing on the npm package publishing pipeline retention ensures that only the irrelevant older packages are affected.
5. **Automated Deletion:** The Azure DevOps retention policy will automatically delete the artifacts based on the configured settings.

Therefore, adjusting the pipeline retention settings is the most direct and effective way to manage the lifecycle of the npm packages within Azure Artifacts and minimize disk space consumption.

For more information on Azure Pipelines retention, see:

Set retention policies for builds, releases, and tests

Artifact retention in Azure Pipelines

Question: 404 CertyIQ

DRAG DROP
-
You have an Azure DevOps pipeline that is used to deploy a Node.js app.
You need to ensure that the dependencies are cached between builds.
How should you configure the deployment YAML? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Values

- always()
- build.sh
- eq(variables.CACHE_RESTORED, 'true')
- integrationtest.sh
- ne(variables.CACHE_RESTORED, 'true')
- npm install

Answer Area

```
…
variables:
   npm_config_cache: $(Pipeline.Workspace)/.npm

steps:
- task: Cache@2
   inputs:
      key: 'npm | "$(Agent.OS)" | package-lock.json'
      restoreKeys: |
          npm | "$(Agent.OS)"
      path: $(npm_config_cache)
      cacheHitVar: CACHE_RESTORED

- script: [                    ]
  condition: [                    ]
  …
```

Answer:

Answer Area

```
…
variables:
   npm_config_cache: $(Pipeline.Workspace)/.npm

steps:
- task: Cache@2
   inputs:
      key: 'npm | "$(Agent.OS)" | package-lock.json'
      restoreKeys: |
          npm | "$(Agent.OS)"
      path: $(npm_config_cache)
      cacheHitVar: CACHE_RESTORED

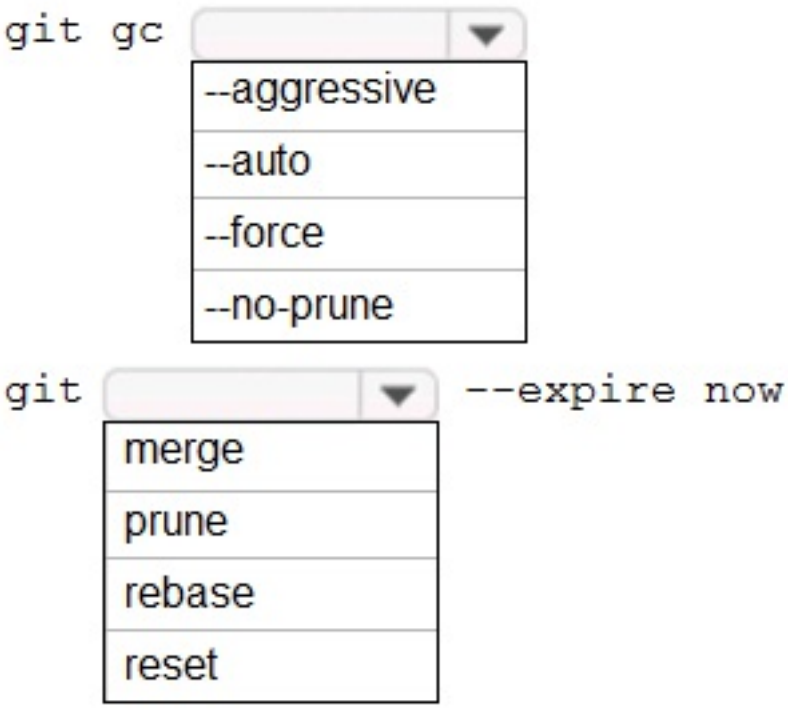
- script: build.sh
  condition: ne(variables.CACHE_RESTORED, 'true')
  …
```

Question: 405 CertyIQ

HOTSPOT -
You manage the Git repository for a large enterprise application.
You need to minimize the data size of the repository.
How should you complete the commands? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

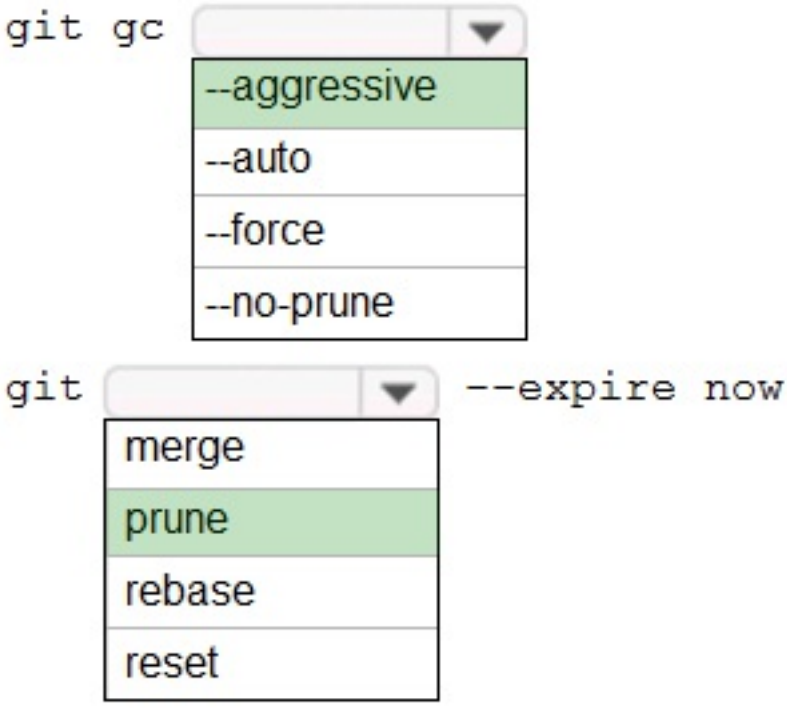
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:

Box 1: --aggressive -

Cleanup unnecessary files and optimize the local repository:

git gc --aggressive

Box 2: prune -
Prune all unreachable objects from the object database:
git prune

Reference:
https://gist.github.com/Zoramite/2039636

Question: 406 CertyIQ

SIMULATION -
You plan to deploy a template named D:\Deploy.json to a resource group named Deploy-lod123456789.
You need to modify the template to meet the following requirements, and then to deploy the template:
- The address space must be reduced to support only 256 total IP addresses.
- The subnet address space must be reduced to support only 64 total IP addresses.

To complete this task, sign in to the Microsoft Azure portal.

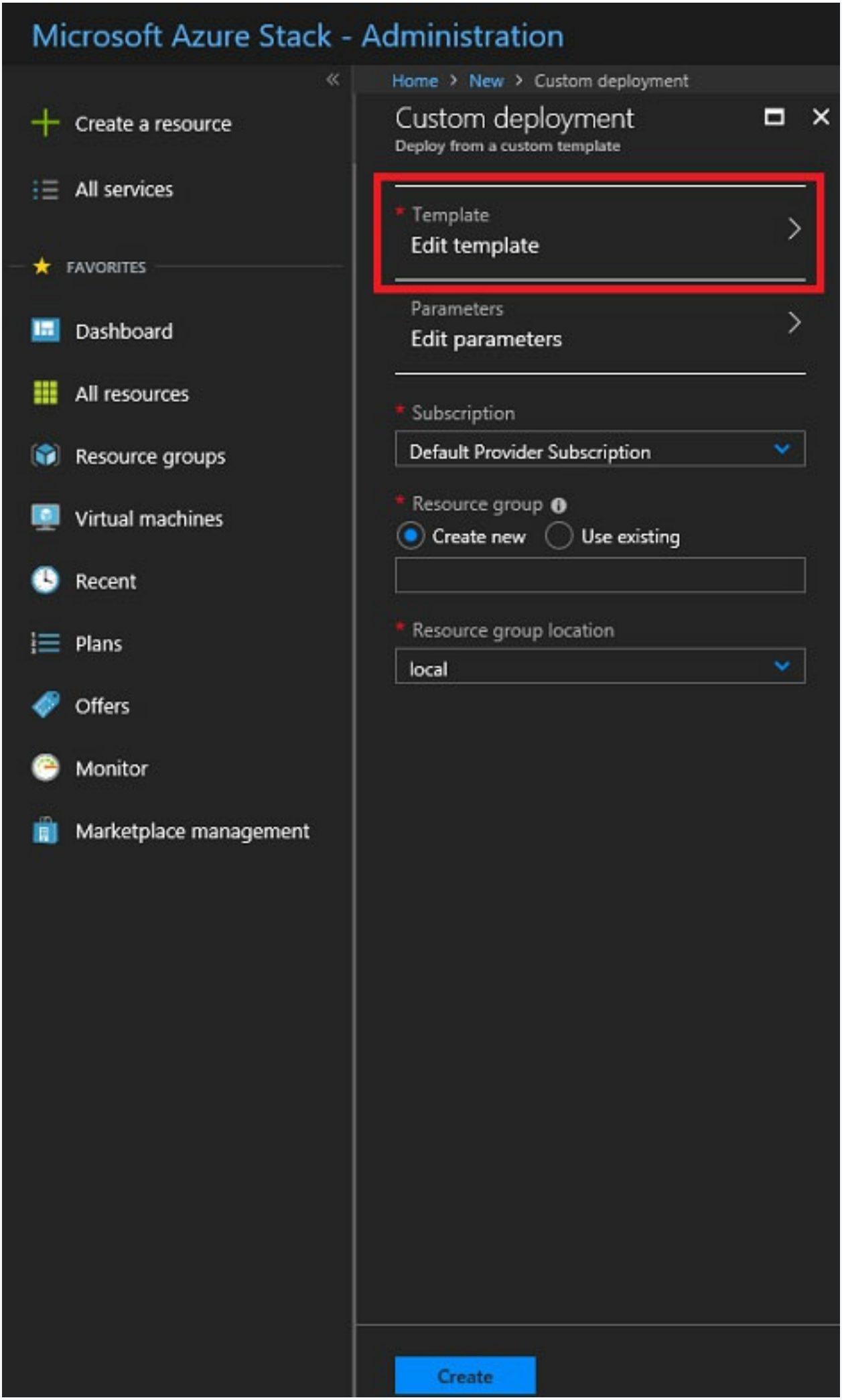
Answer:

See explanation below.

Explanation:
1. Sign in to the portal.
2. Choose template Deploy-lod123456789
3. Select Edit template, and then paste your JSON template code into the code window.
4. Change the ASddressPrefixes to 10.0.0.0/24 in order to support only 256 total IP addresses.
addressSpace": "addressPrefixes": ["10.0.0.0/24"] ,
5. Change the firstSubnet addressprefix to 10.0.0.0/26 to support only 64 total IP addresses.
"subnets":[

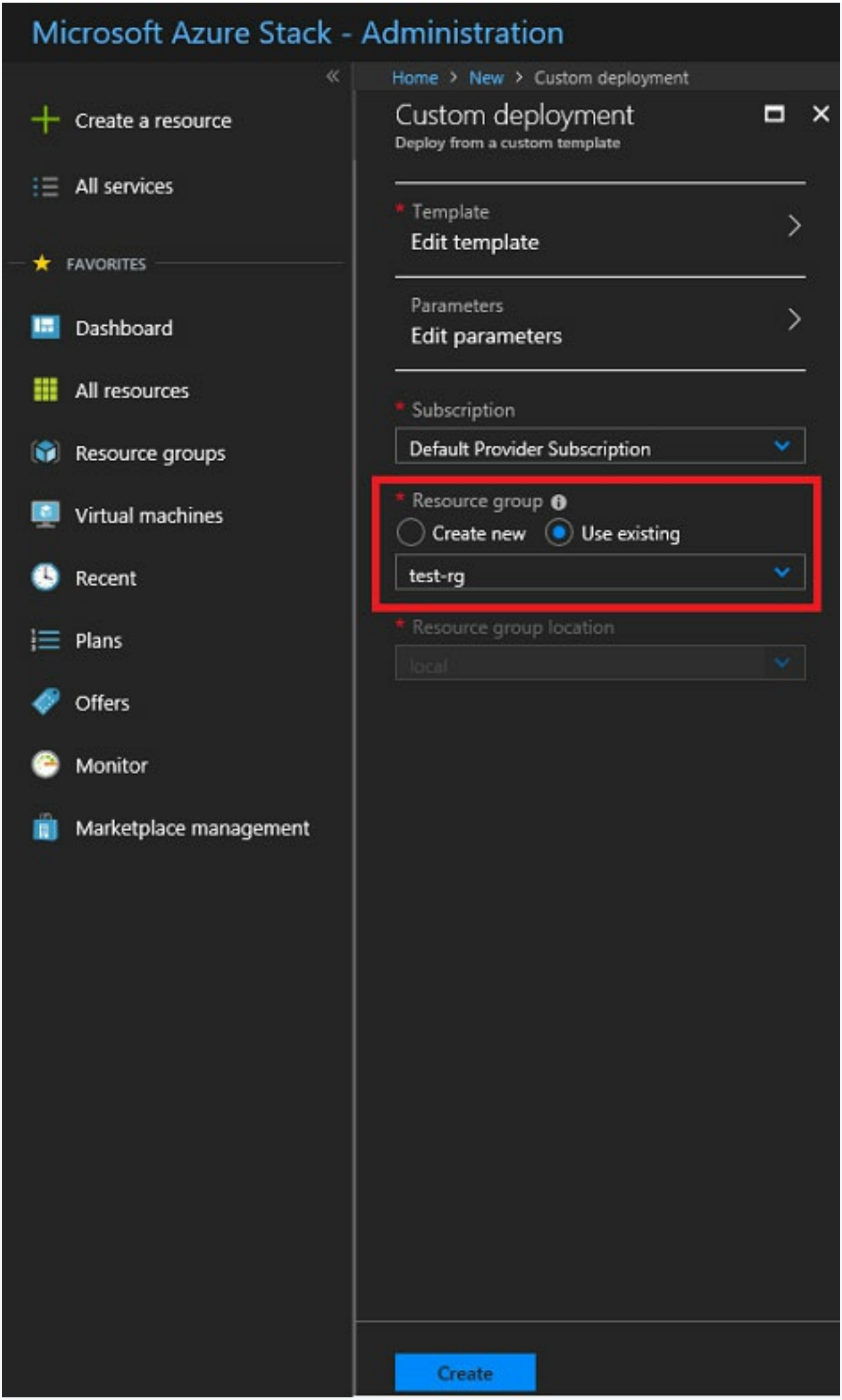
"name":"firstSubnet",
"properties":
"addressPrefix":"10.0.0.0/24"

6. Select Save.



7. Select Edit parameters, provide values for the parameters that are shown, and then select OK.

8. Select Subscription. Choose the subscription you want to use, and then select OK.
9. Select Resource group. Choose an existing resource group or create a new one, and then select OK.



10. Select Create. A new tile on the dashboard tracks the progress of your template deployment.

Reference:
https://docs.microsoft.com/en-us/azure-stack/user/azure-stack-deploy-template-portal?view=azs-1908
https://docs.microsoft.com/en-us/azure/architecture/building-blocks/extending-templates/update-resource

Question: 407 CertyIQ

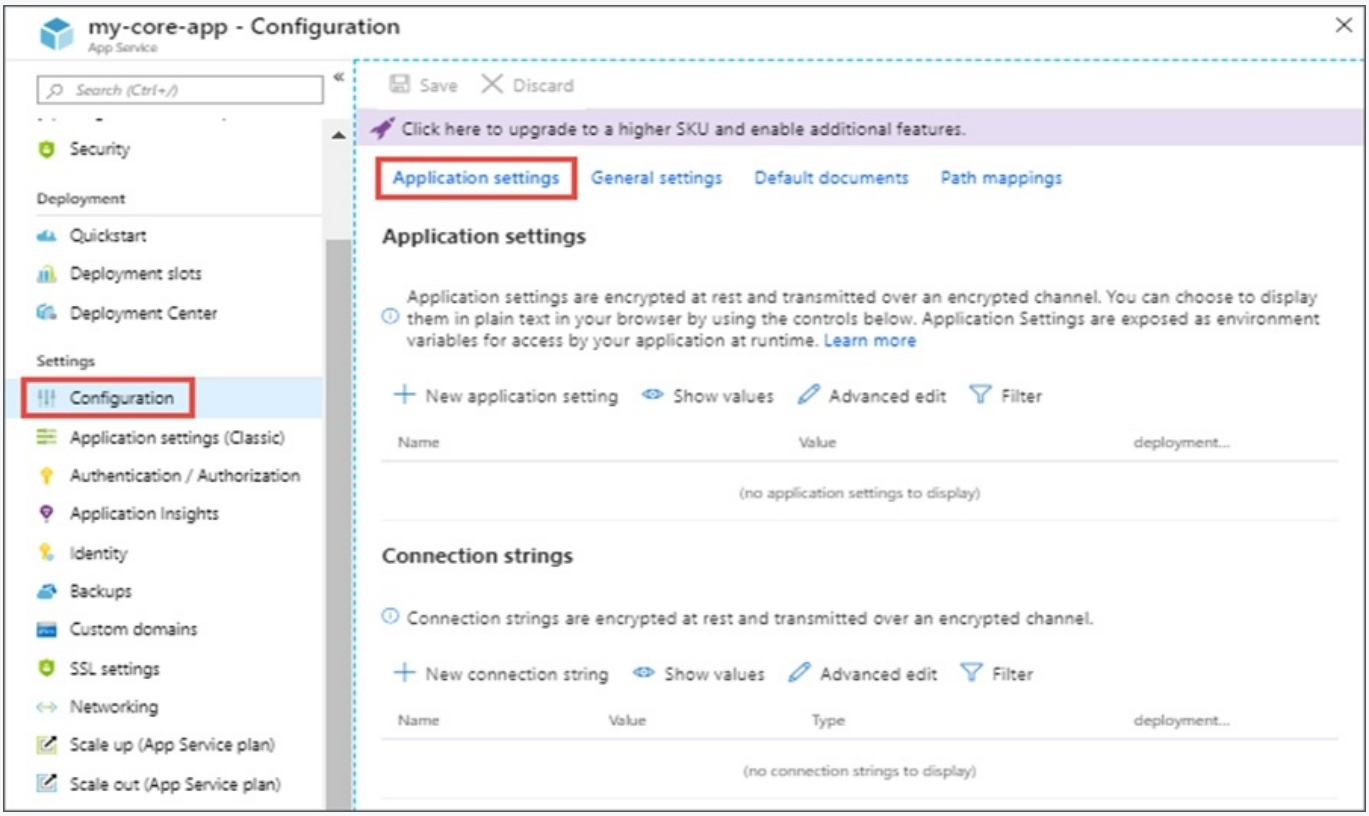
SIMULATION -
You need to configure an Azure web app named az400-123456789-main to contain an environmental variable named `MAX_ITEMS`. The environmental variable must have a value of 50.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:

1. In the Azure portal, navigate to the az400-123456789-main app's management page. In the app's left menu, click Configuration > Application settings.



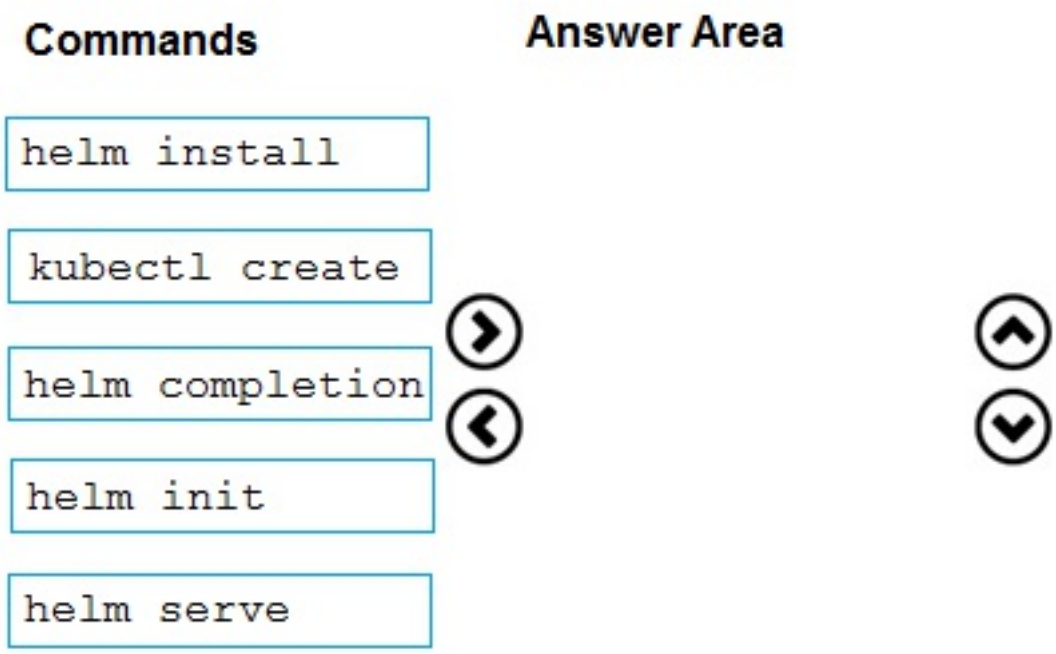
2. Click New Application settings
3. Enter the following:
- Name: MAX_ITEMS
- Value: 50

Reference:
https://docs.microsoft.com/en-us/azure/app-service/configure-common

Question: 408 CertyIQ

DRAG DROP -
You provision an Azure Kubernetes Service (AKS) cluster that has RBAC enabled. You have a Helm chart for a client application.
You need to configure Helm and Tiller on the cluster and install the chart.

Which three commands should you recommend be run in sequence? To answer, move the appropriate commands from the list of commands to the answer area and arrange them in the correct order.
Select and Place:



Answer: 2>4>1

Explanation:

Step 1: Kubectl create -

You can add a service account to Tiller using the --service-account <NAME> flag while you're configuring Helm (step 2 below). As a prerequisite, you'll have to create a role binding which specifies a role and a service account name that have been set up in advance.

Example: Service account with cluster-admin role

$ kubectl create -f rbac-config.yaml

serviceaccount "tiller" created

clusterrolebinding "tiller" created

$ helm init --service-account tiller

Step 2: helm init -

To deploy a basic Tiller into an AKS cluster, use the helm init command.

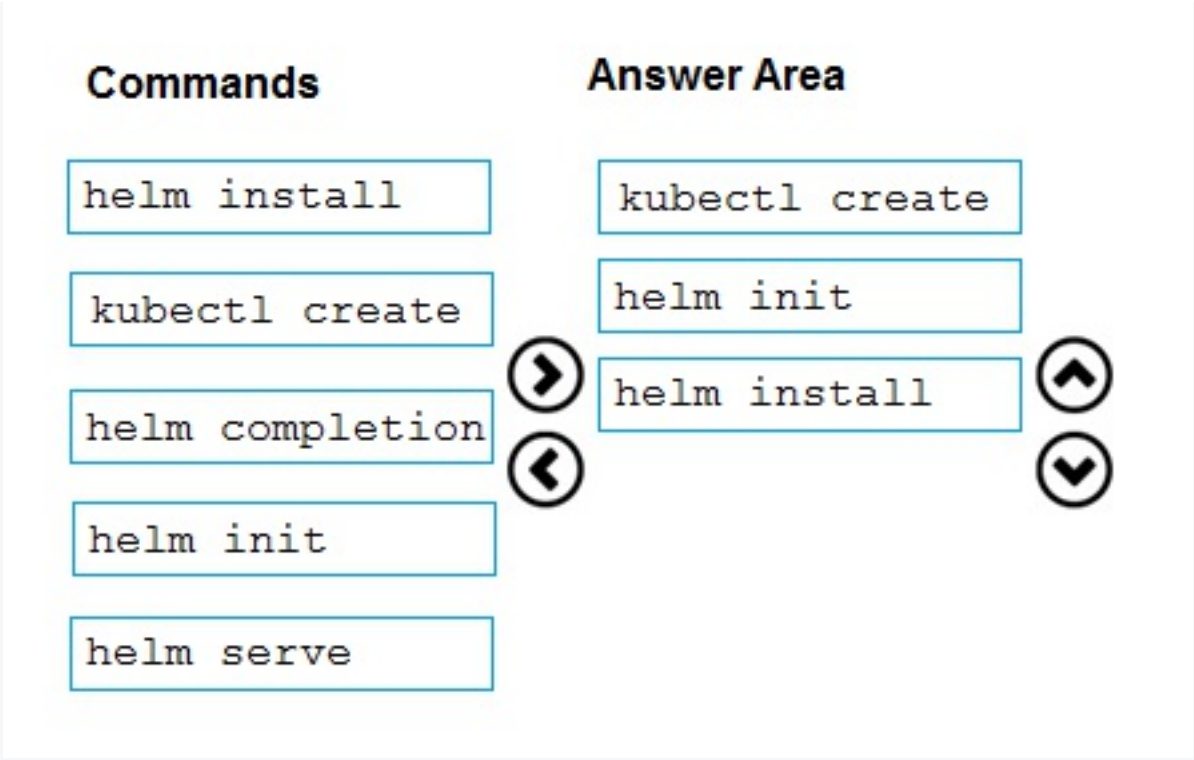
Step 3: helm install -

To install charts with Helm, use the helm install command and specify the name of the chart to install.

Reference:

https://docs.microsoft.com/en-us/azure/aks/kubernetes-helm

https://docs.helm.sh/using_helm/#tiller-namespaces-and-rbac



Question: 409 CertyIQ

Your company builds a multi-tier web application.
You use Azure DevOps and host the production application on Azure virtual machines.
Your team prepares an Azure Resource Manager template of the virtual machine that you will use to test new features.
You need to create a staging environment in Azure that meets the following requirements:
- Minimizes the cost of Azure hosting
- Provisions the virtual machines automatically
- Uses the custom Azure Resource Manager template to provision the virtual machines

What should you do?

A. In Azure Cloud Shell, run Azure CLI commands to create and delete the new virtual machines in a staging resource group.

B. In Azure DevOps, configure new tasks in the release pipeline to deploy to Azure Cloud Services.

C. From Azure Cloud Shell, run Azure PowerShell commands to create and delete the new virtual machines in a staging resource group.

D. In Azure DevOps, configure new tasks in the release pipeline to create and delete the virtual machines in Azure DevTest Labs.

Answer: D

Explanation:

The correct answer is D because Azure DevTest Labs is specifically designed for creating cost-effective environments for development and testing. It allows for automatic provisioning of virtual machines using custom Azure Resource Manager (ARM) templates, which aligns with the requirement of using the company's custom template. DevTest Labs also provides features to minimize costs, such as auto-shutdown, claimable VMs, and policies to limit resources used.

Option A and C, using Azure CLI or PowerShell in Azure Cloud Shell, could provision and deprovision VMs, but it would lack the lifecycle management and cost-optimization features that DevTest Labs offers. It would require custom scripting for auto-shutdown and other cost-saving measures, increasing administrative overhead.

Option B, deploying to Azure Cloud Services, doesn't align well with the provided scenario. The current production application is hosted on Azure virtual machines, and the prompt doesn't suggest a migration to Cloud Services. Moreover, Cloud Services, while supporting automated deployment, is a PaaS offering and doesn't give as much direct control over the VM configuration as DevTest Labs, which can use custom ARM templates to replicate the VM setup precisely.

In summary, DevTest Labs is the best solution because it directly addresses all three requirements: minimizing costs, automatic VM provisioning, and using a custom ARM template within an environment designed for development and testing. It avoids the need for custom scripting and is a more manageable and integrated solution within the Azure DevOps ecosystem.

Here are some authoritative links for further research:

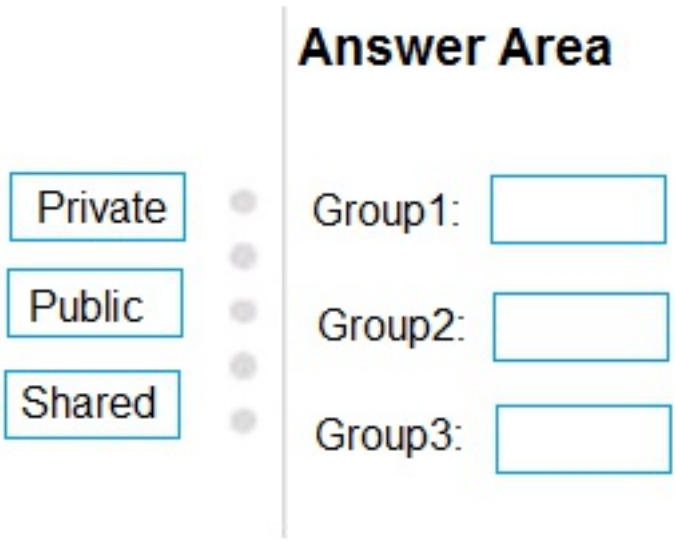
Azure DevTest Labs: https://azure.microsoft.com/en-us/services/devtest-lab/
Azure Resource Manager templates: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/overview
Azure DevOps Release Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/?view=azure-devops

Question: 410 CertyIQ

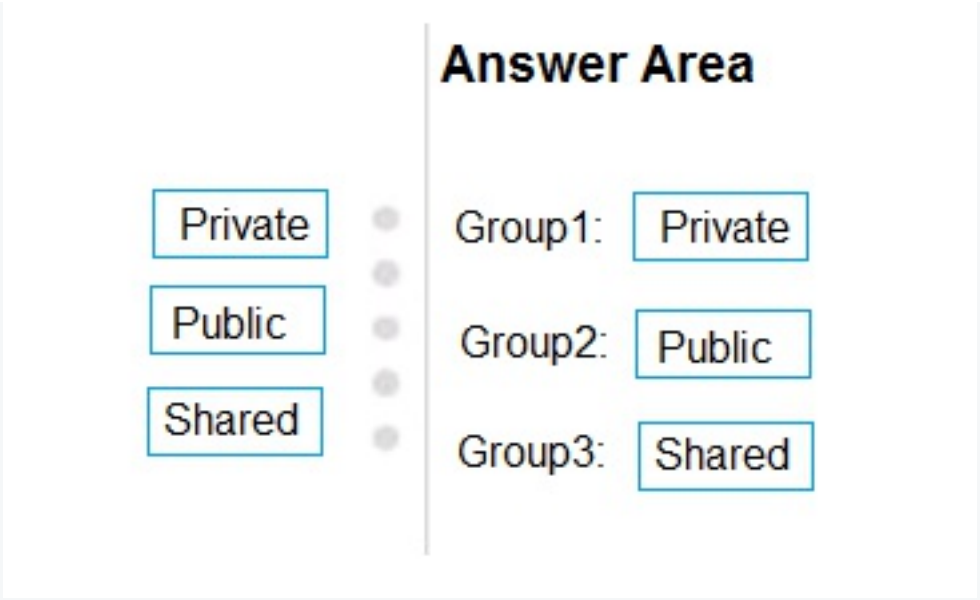
DRAG DROP -
You are implementing an Azure DevOps strategy for mobile devices using App Center.
You plan to use distribution groups to control access to releases.
You need to create the distribution groups shown in the following table.

Name	Use
Group1	Application testers who are invited by email
Group2	Early release users who use unauthenticated public links
Group3	Application testers for all the apps of your company

Which type of distribution group should you use for each group? To answer, drag the appropriate group types to the correct locations. Each group type may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:
Box1: Private -
In App Center, distribution groups are private by default. Only testers invited via email can access the releases available to this group.

Box 2: Public -
Distribution groups must be public to enable unauthenticated installs from public links.

Box 3: Shared -
Shared distribution groups are private or public distribution groups that are shared across multiple apps in a single organization.

Reference:
https://docs.microsoft.com/en-us/appcenter/distribution/groups

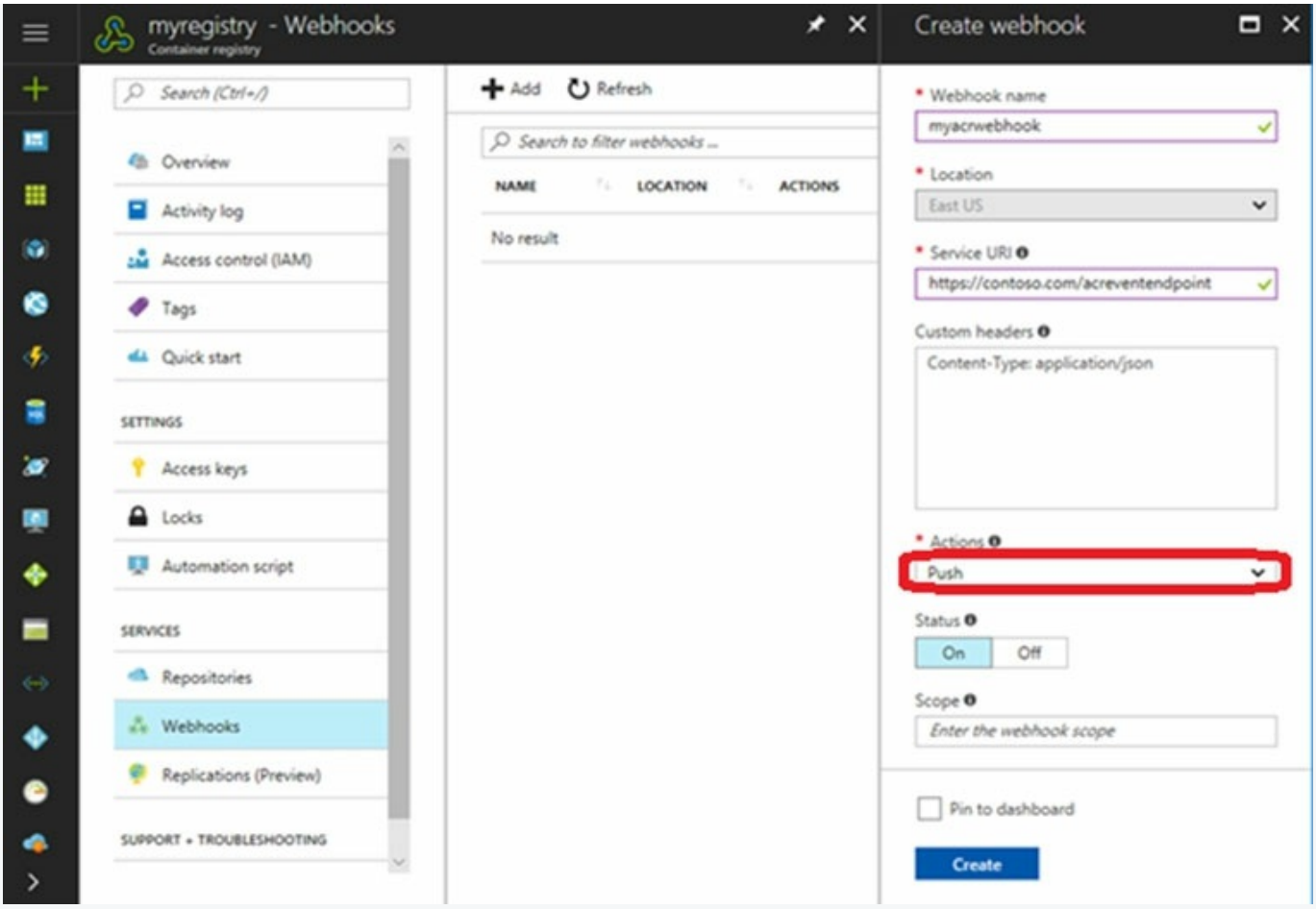
Question: 411 CertyIQ

SIMULATION -
You need to ensure that the https://contoso.com/statushook webhook is called every time a repository named az400123456789acr1 receives a new version of an image named dotnetapp.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:
1. Sign in to the Azure portal.
2. Navigate to the container registry az400123456789acr1.
3. Under Services, select Webhooks.
4. Select the existing webhook https://contoso.com/statushook, and double-click on it to get its properties.
5. For Trigger actions select image push
Example web hook:



Reference:
https://docs.microsoft.com/en-us/azure/container-registry/container-registry-webhook

Question: 412 CertyIQ

HOTSPOT -
You need to create deployment files for an Azure Kubernetes Service (AKS) cluster. The deployments must meet the provisioning storage requirements shown in the following table.

Deployment	Requirement
Deployment 1	Use files stored on an SMB-based share from the container's file system.
Deployment 2	Use files stored on a managed disk from the container's file system.
Deployment 3	Securely access X.509 certificates from the container's file system.

Which resource type should you use for each deployment? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

Deployment 1:
- driver: secrets-store.csi.k8s.io
- blobfuse-flexvol
- provisioner: kubernetes.io/azure-disk
- provisioner: kubernetes.io/azure-file
- volume.beta.kubernetes.io/storage-provisioner

Deployment 2:
- driver: secrets-store.csi.k8s.io
- blobfuse-flexvol
- provisioner: kubernetes.io/azure-disk
- provisioner: kubernetes.io/azure-file
- volume.beta.kubernetes.io/storage-provisioner

Deployment 3:
- driver: secrets-store.csi.k8s.io
- blobfuse-flexvol
- provisioner: kubernetes.io/azure-disk
- provisioner: kubernetes.io/azure-file
- volume.beta.kubernetes.io/storage-provisioner

Answer:

Answer Area

Deployment 1:
- driver: secrets-store.csi.k8s.io
- blobfuse-flexvol
- provisioner: kubernetes.io/azure-disk
- **provisioner: kubernetes.io/azure-file**
- volume.beta.kubernetes.io/storage-provisioner

Deployment 2:
- driver: secrets-store.csi.k8s.io
- blobfuse-flexvol
- **provisioner: kubernetes.io/azure-disk**
- provisioner: kubernetes.io/azure-file
- volume.beta.kubernetes.io/storage-provisioner

Deployment 3:
- **driver: secrets-store.csi.k8s.io**
- blobfuse-flexvol
- provisioner: kubernetes.io/azure-disk
- provisioner: kubernetes.io/azure-file
- volume.beta.kubernetes.io/storage-provisioner

Explanation:

Deployment 1: provisioner: kubernetes.io/azure-file

You can use Azure Files to connect using the Server Message Block (SMB) protocol.

Deployment 2: provisioner: kubernetes.io/azure-disk

Deployment 3: driver: secrets-store.csi.k8s.io

Azure Key vault provider for Secrets Store CSI driver allows you to access secrets stored in an Azure Key vault instance. The Secrets Store CSI driver secrets- store.csi.k8s.io allows the cluster to mount secrets stored in Azure Key vault into the pods as a volume.

Incorrect Answers:

blobfuse-flexvolume: This driver allows Kubernetes to access virtual filesystem backed by the Azure Blob storage.
Note: azurekeyvault-flexvolume has been deprecated and replaced by the Azure Key Vault Provider for Secret Store CSI Driver.

Reference:
https://docs.microsoft.com/bs-cyrl-ba/azure/aks/azure-files-dynamic-pv https://docs.microsoft.com/en-us/azure/aks/azure-disks-dynamic-pv https://docs.microsoft.com/en-us/samples/azure-samples/secrets-store-csi-with-aks-akv/secrets-store-csi-with-aks-akv/

Question: 413 CertyIQ

Your company uses Azure DevOps to manage the build and release processes for applications.
You use a Git repository for applications source control.
You plan to create a new branch from an existing pull request. Later, you plan to merge the new branch and the target branch of the pull request.
You need to use a pull request action to create the new branch. The solution must ensure that the branch uses only a portion of the code in the pull request.
Which pull request action should you use?

A. Set as default branch
B. Approve with suggestions
C. Cherry-pick
D. Reactivate
E. Revert

Answer: C

Explanation:

The correct answer is **C. Cherry-pick**.

Here's why:

Cherry-picking allows you to select specific commits from one branch (in this case, the pull request branch) and apply them to a new branch. This addresses the requirement of using "only a portion of the code in the pull request" for the new branch. This is unlike simply merging the entire branch, which would bring in all changes. It gives granular control over which changes are integrated.

Option A, "Set as default branch," changes the main branch of the repository, which isn't relevant to creating a new branch from a pull request subset. Option B, "Approve with suggestions," is a code review action and doesn't create new branches or selectively apply changes. Option D, "Reactivate," simply re-enables a closed pull request and has nothing to do with selectively incorporating code. Option E, "Revert" cancels a change and is thus opposite of incorporating one, much less selectively.

Cherry-picking directly solves the problem of creating a new branch with only selected code from a pull request. The workflow would involve creating a new branch from the target branch, and then cherry-picking the specific commits from the pull request branch that are desired in the new branch.

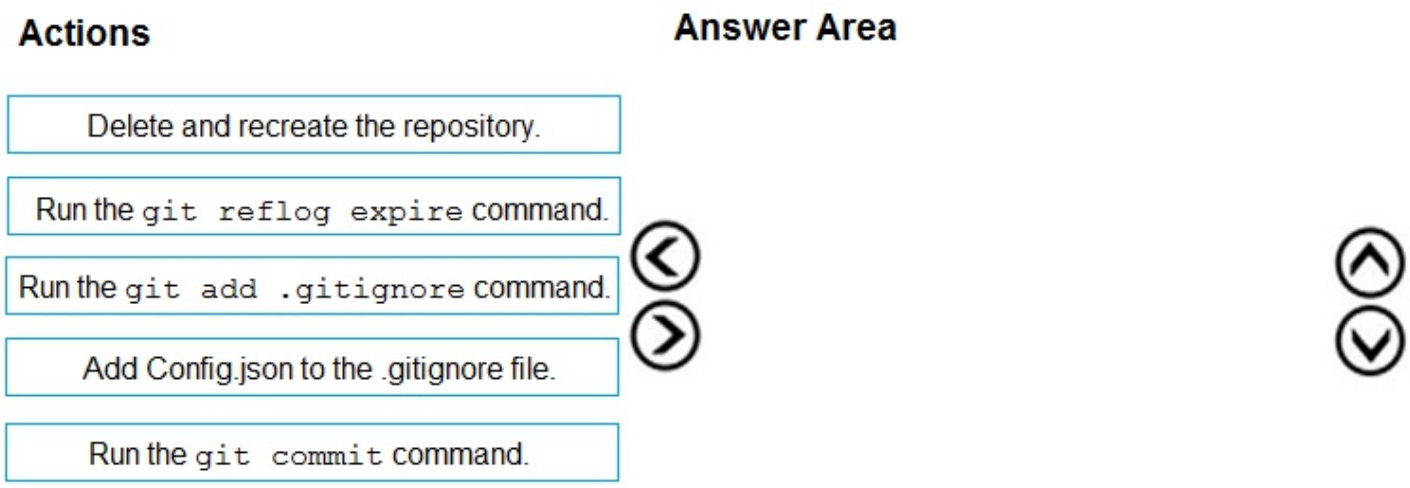
Here's an example scenario: A pull request fixes multiple bugs, but you only want to apply the fix for bug A in the new branch right now. Cherry-picking allows you to select the commit that fixes bug A and apply only that commit to the new branch.

Authoritative links for further research:

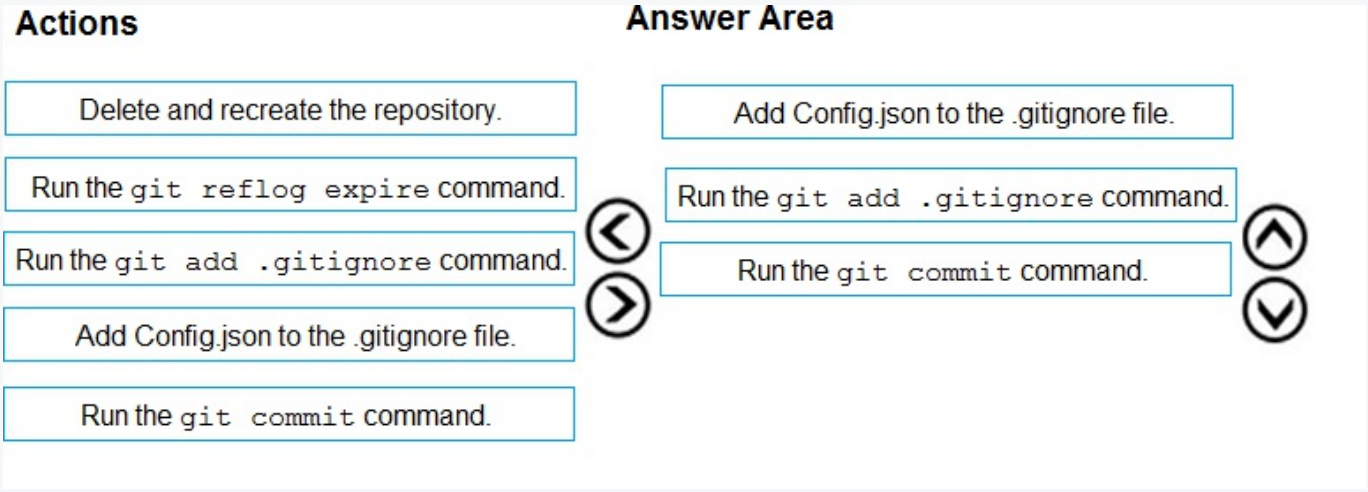
Git Cherry-picking: https://git-scm.com/docs/git-cherry-pick
Using cherry-pick to selectively copy commits: https://www.atlassian.com/git/tutorials/cherry-pick

Question: 414 CertyIQ

DRAG DROP -
You manage the Git repository for a large enterprise application.
During the development of the application, you use a file named Config.json.
You need to prevent Config.json from being committed to the source control whenever changes to the application are committed.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Add Config.json to the .gitignore file.

Run the git add .gitignore command.

Run the git commit command.

Question: 415 CertyIQ

You are designing a build pipeline in Azure Pipelines.
The pipeline requires a self-hosted agent. The build pipeline will run once daily and will take 30 minutes to

complete.
You need to recommend a compute type for the agent. The solution must minimize costs.
What should you recommend?

A. an Azure Kubernetes Service (AKS) cluster
B. Azure Container Instances
C. an Azure virtual machine scale set
D. Azure virtual machines

Answer: C

Explanation:

The recommended compute type is an Azure virtual machine scale set (VMSS). Here's why:

Cost Optimization: The primary goal is to minimize costs. Since the pipeline runs only once daily for 30 minutes, maintaining a dedicated, always-on virtual machine (Option D) incurs unnecessary costs for the idle period. Azure Container Instances (Option B) could be cost-effective if the pipeline could be easily containerized. However, VMSS allows dynamic scaling, meaning instances can be provisioned only when needed, automatically scaling down to zero or a minimal number of instances during idle times. This significantly reduces the cost compared to dedicated resources.
Scalability and Automation: VMSS allows you to automatically scale the number of VM instances based on demand. While the build pipeline runs only once daily, VMSS can be configured to automatically scale up instances to handle the build and then scale down afterward, making efficient use of resources.
Management Overhead: VMSS simplifies management compared to managing individual VMs. Updates and configurations can be applied to all instances in the scale set, simplifying the administration of the self-hosted agent.
Azure Kubernetes Service (AKS): AKS (Option A) is better suited for complex, containerized applications requiring orchestration and high availability. Using AKS for a single, daily build pipeline would introduce significant management overhead and higher costs compared to VMSS, as AKS has associated operational costs even when idle.

While Azure Container Instances (ACI) could seem suitable for short-lived tasks, ACI might be more complex to configure and manage as a self-hosted agent if the build process requires specific software or configurations that are not easily containerized. Furthermore, managing the agent's lifecycle (starting, stopping, updating) might be more involved with ACI compared to the automation offered by VMSS. Using VMSS simplifies scaling, patching, and management using built-in tools in Azure, and its pay-per-use model aligns with the infrequent build requirement.

Therefore, Azure virtual machine scale sets offer the optimal balance of cost-effectiveness, scalability, and management simplicity for hosting a self-hosted build agent that runs a pipeline once daily for 30 minutes.

Supporting Links:

Azure Virtual Machine Scale Sets: https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/
Azure Container Instances: https://learn.microsoft.com/en-us/azure/container-instances/
Azure Kubernetes Service: https://learn.microsoft.com/en-us/azure/aks/

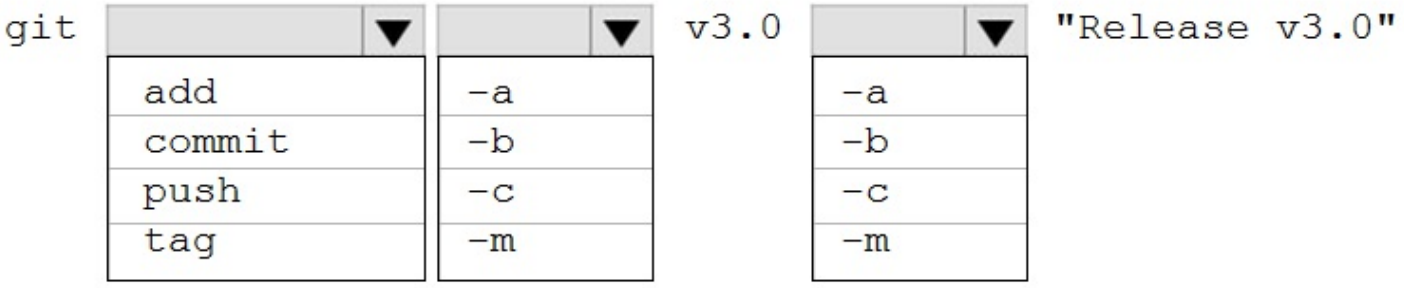
Question: 416 CertyIQ

HOTSPOT -
You are finalizing a release in GitHub.
You need to apply the following labels to the release:
✑ Name

- Email
- Release v3.0
- Release date

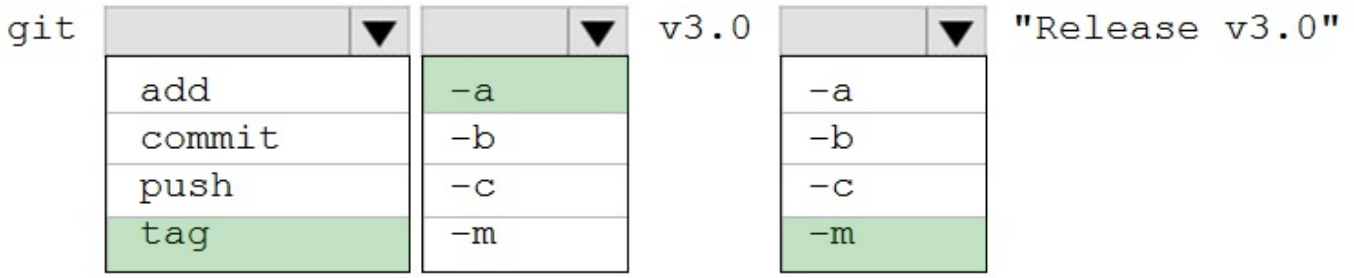
How should you complete the git command? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:
Box 1; tag -
Tagging. Like most VCSs, Git has the ability to tag specific points in a repository's history as being important. Typically, people use this functionality to mark release points (v1.0, v2.0 and so on).

Box 2: -a -
Creating an annotated tag in Git is simple. The easiest way is to specify -a when you run the tag command:
Example:
$ git tag -a v1.4 -m "my version 1.4"

Box 3: -m -

Reference:
https://git-scm.com/book/en/v2/Git-Basics-Tagging

Question: 417 CertyIQ

You have a project in Azure DevOps. You have an Azure Resource Group deployment project in Microsoft Visual Studio that is checked in to the Azure DevOps project.
You need to create a release pipeline that will deploy resources by using Azure Resource Manager templates. The solution must minimize administrative effort.
Which task type should you include in the solution?

A. Azure Cloud Service Deployment
B. Azure RM Web App Deployment

C. Azure PowerShell

D. Azure App Service Manage

Answer: C

Explanation:

The correct answer is **C. Azure PowerShell**. Here's why:

The core requirement is to deploy resources using Azure Resource Manager (ARM) templates from an Azure DevOps project. ARM templates are declarative JSON files that define the infrastructure and configuration for your Azure resources. The most direct and flexible way to deploy these templates is through Azure PowerShell.

The Azure PowerShell task in Azure DevOps allows you to execute PowerShell scripts within your pipeline. You can use the New-AzResourceGroupDeployment cmdlet (or its older, deprecated equivalent New-AzureRmResourceGroupDeployment) to deploy ARM templates to an Azure Resource Group. This offers granular control over the deployment process, including parameterization, error handling, and logging.

Options A and B are specific to deploying Azure Cloud Services and Web Apps, respectively. These options are not suitable for general ARM template deployments that might encompass various Azure resources, not just web apps or cloud services. While they might indirectly deploy resources defined within the scope of those specific service deployments, they don't provide the broad applicability required by the scenario.

Option D, Azure App Service Manage, is similarly specialized for managing App Service resources. It doesn't provide a general mechanism for ARM template deployment across various Azure resources and is therefore unsuitable.

Using Azure PowerShell directly provides the most flexibility and aligns directly with the need to execute ARM template deployments. You can easily customize the deployment process via scripting. You can also manage parameters, handle error conditions, and integrate with other pipeline tasks for pre- and post-deployment steps. By using the Azure PowerShell task in your Azure DevOps release pipeline, you gain the greatest control and customization capabilities for deploying your resources from Azure Resource Manager templates while minimizing the administrative overhead associated with more specialized tasks.

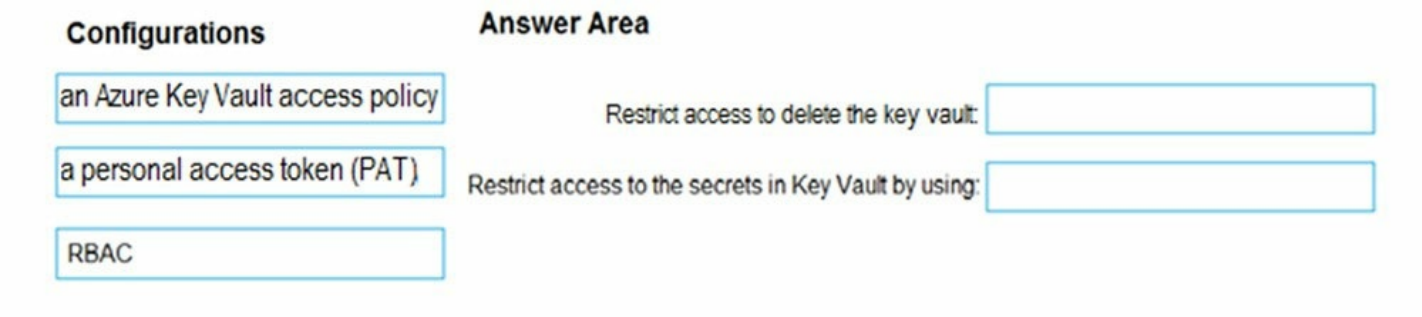
Here are helpful links for further research:

Deploy resources with ARM templates and Azure PowerShell: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/deploy-powershell
New-AzResourceGroupDeployment cmdlet: https://learn.microsoft.com/en-us/powershell/module/az.resources/new-azresourcegroupdeployment?view=azps-11.5.0
Azure DevOps PowerShell Task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/deploy/azure-powershell?view=azure-devops

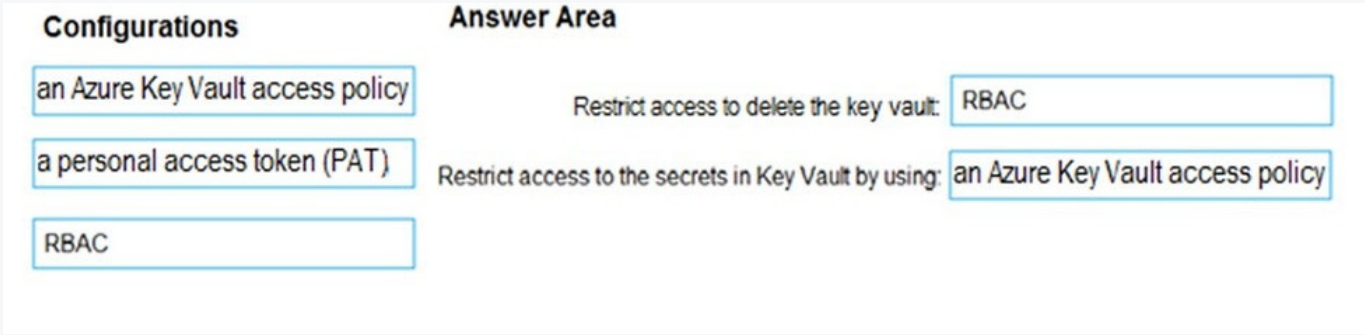
Question: 418 CertyIQ

DRAG DROP -
Your company has a project in Azure DevOps.
You plan to create a release pipeline that will deploy resources by using Azure Resource Manager templates. The templates will reference secrets stored in Azure
Key Vault.
You need to recommend a solution for accessing the secrets stored in the key vault during deployments. The solution must use the principle of least privilege.
What should you include in the recommendation? To answer, drag the appropriate configurations to the correct targets. Each configuration may be used once, more than once, or not at all. You may need to drag the split bar

between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

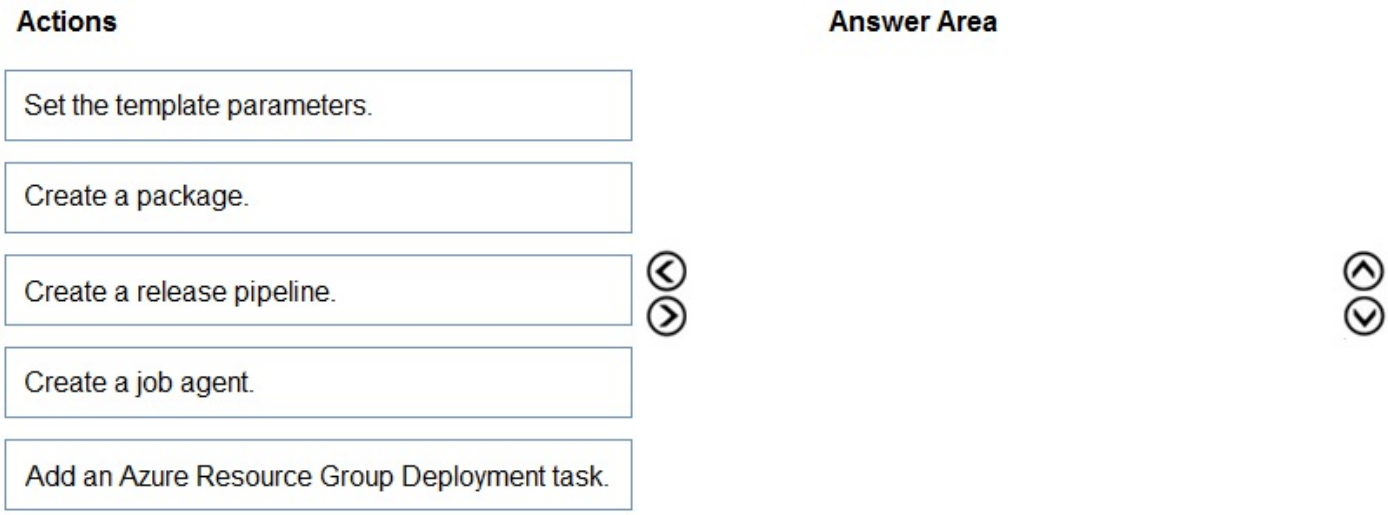
Delete key vault - RBAC.

Access to secrets - an Azure Key Vault access policy.

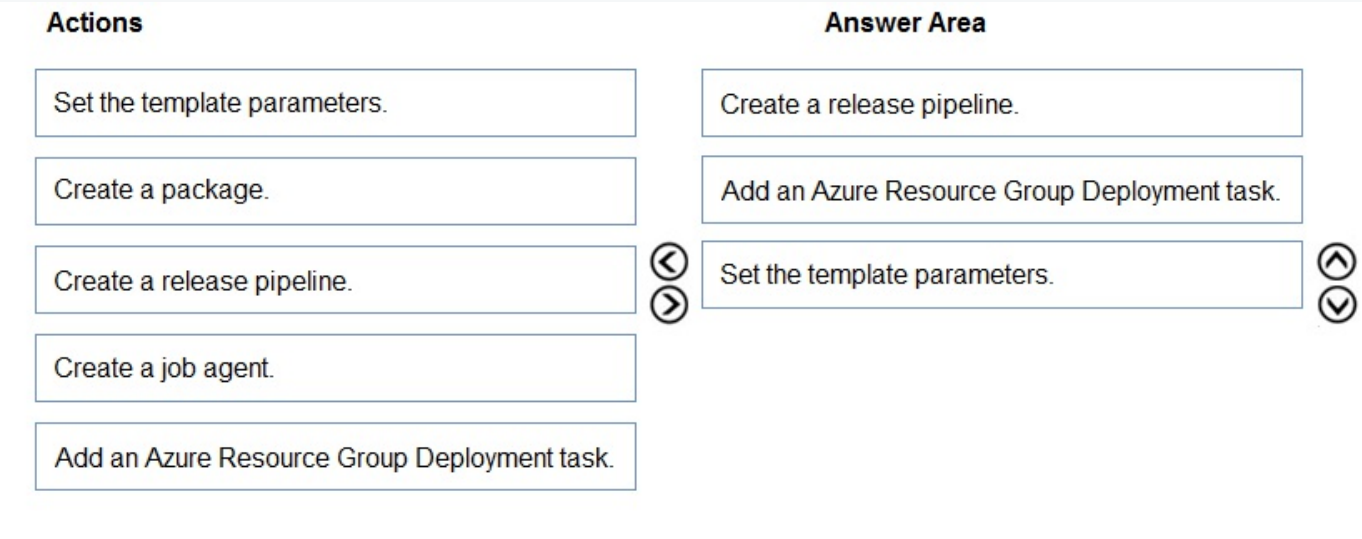
https://docs.microsoft.com/en-us/azure/key-vault/general/security-overview

Question: 419 CertyIQ

DRAG DROP -
As part of your application build process, you need to deploy a group of resources to Azure by using an Azure Resource Manager template located on GitHub.
Which three action should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:
Step 1: Create a release pipeline
You need to create a new pipeline.
You can integrate Azure Resource Manager templates (ARM templates) with Azure Pipelines for continuous integration and continuous deployment (CI/CD).
Step 2: Add an Azure Resource Group Deployment task
Step 3: Set the template parameters

Reference:
https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/add-template-to-azure-pipelines

Question: 420 CertyIQ

You have an Azure DevOps project that contains a release pipeline and a Git repository.
When a new code revision is committed to the repository, a build and release is triggered.
You need to ensure that release information for the pipeline is added automatically to the work items associated to the Git commit.
What should you do?

A. Modify the Integrations options for the pipeline.
B. Modify the post-deployment conditions for the last stage of the pipeline.
C. Add an agentless job to the pipeline.
D. Modify the service hooks for the project.

Answer: A

Explanation:

The correct answer is A: Modify the Integrations options for the pipeline.

Here's the justification: Azure DevOps provides a feature called "Integrations" within release pipelines that directly addresses the requirement of automatically linking release information to work items associated with Git commits. This feature enables the system to automatically track deployments and associate them with the work items resolved in the corresponding commits.

By configuring the Integrations options, specifically "Associate work items with releases," Azure DevOps analyzes the commit messages associated with the release and identifies work items mentioned (e.g., #123). It

then automatically links the release to those work items. This allows stakeholders to easily trace the deployment history and understand which releases contain specific work items. Option B, modifying post-deployment conditions, focuses on actions after a deployment, not on associating releases with work items. Option C, adding an agentless job, is used for tasks that don't require a build agent, such as calling REST APIs or managing resources. Option D, modifying service hooks, focuses on triggering events in external services, not on linking release information within Azure DevOps. Therefore, only modifying the integration options within the pipeline directly addresses the stated requirement.

Authoritative links for further research:

Automatically link work items to releases: https://learn.microsoft.com/en-us/azure/devops/boards/boards/connect-work-items-to-git-devops?view=azure-devops
Understand deployment tracking: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/deploy-applications-tasks?view=azure-devops&tabs=powershell

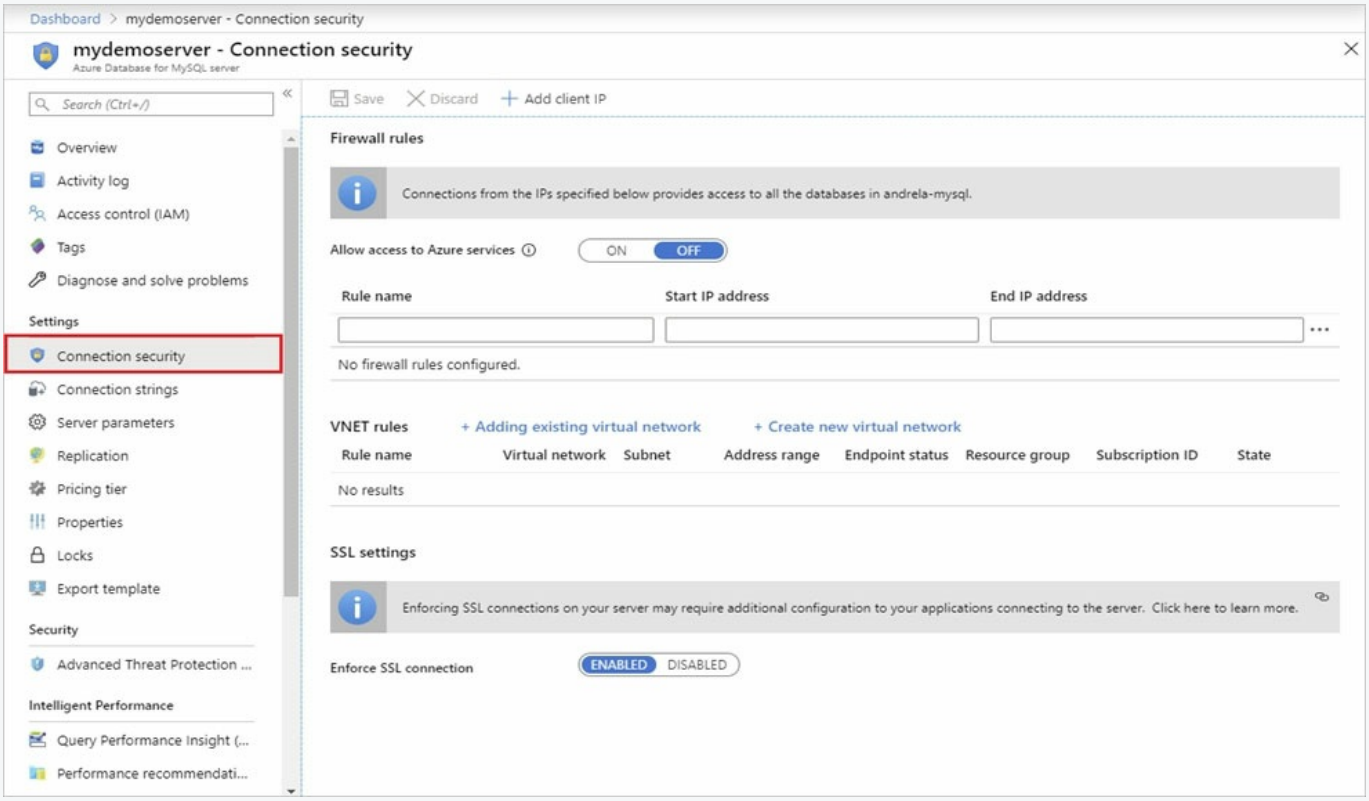
Question: 421 CertyIQ

SIMULATION -
You plan to add a new web farm that will be published by using an IP address of 10.0.0.5.
You need to allow traffic from the web farm to an Azure Database for MySQL server named az400-123456789-mysql.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

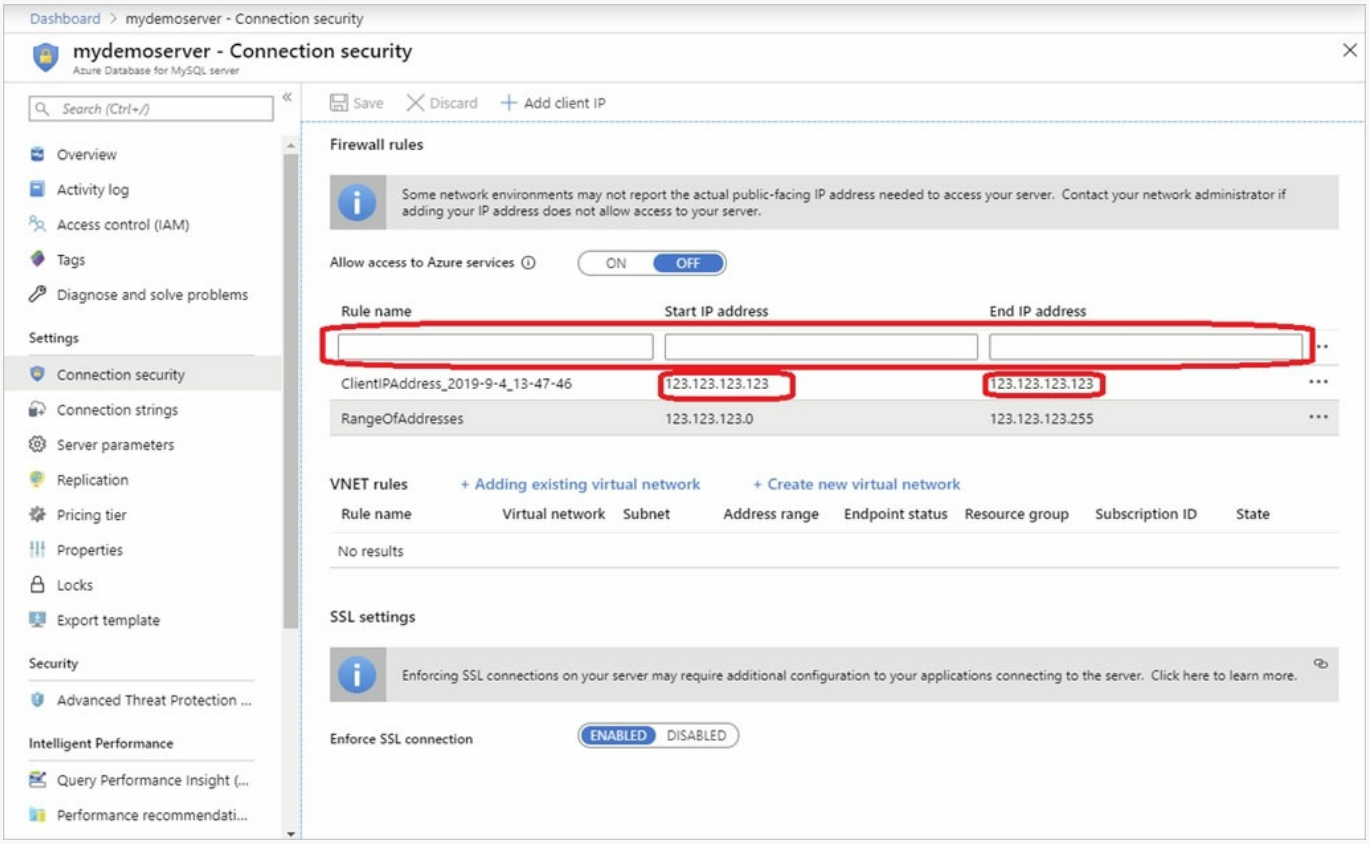
See explanation below.

Explanation:
Server-level firewall rules can be used to manage access to an Azure Database for MySQL Server from a specified IP address or a range of IP addresses.
Create a server-level firewall rule in the Azure portal
1. On the MySQL server page, under Settings heading, click Connection Security to open the Connection Security page for the Azure Database for MySQL.



2. In the firewall rules for the Azure Database for MySQL, you can specify a single IP address or a range of

addresses. If you want to limit the rule to a single IP address, type the same address in the Start IP and End IP fields. Opening the firewall enables administrators, users, and application to access any database on the MySQL server to which they have valid credentials.



3. Click Save on the toolbar to save this server-level firewall rule. Wait for the confirmation that the update to the firewall rules is successful.

Reference:
https://docs.microsoft.com/en-us/azure/mysql/howto-manage-firewall-using-portal#create-a-server-level-firewall-rule-in-the-azure-portal

Question: 422

CertyIQ

Your company has a release pipeline in an Azure DevOps project.
You plan to deploy to an Azure Kubernetes Services (AKS) cluster by using the Helm package and deploy task.
You need to install a service in the AKS namespace for the planned deployment.
Which service should you install?

A. Azure Container Registry
B. Chart
C. Kubectl
D. Tiller

Answer: D

Explanation:

The correct answer is **D. Tiller**.

Here's a detailed justification:

Helm, a package manager for Kubernetes, simplifies the deployment and management of applications within a Kubernetes cluster. It utilizes a client-server architecture. The Helm client runs on your local machine or

CI/CD pipeline, while Tiller was the server-side component that ran inside the Kubernetes cluster.

Tiller's responsibility was to receive charts (Helm packages containing application manifests and configuration) from the Helm client, render the manifests, and apply them to the Kubernetes cluster to deploy the application. It also tracked the deployed releases, enabling features like upgrades, rollbacks, and deletion.

The question specifically mentions using the Helm package and deploy task in Azure DevOps. This task interacts with Helm to deploy an application based on a Helm chart. Before Helm v3, Tiller was a crucial requirement for this process. Without Tiller running in the AKS cluster's namespace, the Helm client would be unable to communicate with the cluster and deploy the application described in the chart.

However, it's important to note that Tiller has been removed from Helm v3 and later. This is because Tiller had security concerns and complexity issues due to its cluster-wide access. In Helm v3, the Helm client directly interacts with the Kubernetes API server, eliminating the need for a separate server component like Tiller. Therefore, for modern Helm versions, this question and its answer are outdated.

In the context of the original question (presumably related to older versions of Helm), the correct answer is indeed Tiller.

However, in real-world, modern AKS deployments using current versions of Helm (v3 or later), you wouldn't need to install Tiller. Instead, you would configure the Helm client to interact with the Kubernetes API server, often using service accounts and RBAC (Role-Based Access Control) to manage permissions.

Therefore, while the answer is correct for older versions of Helm, the question's premise is now obsolete.

Further Reading:

Helm documentation on Tiller (for historical context): https://helm.sh/docs/v2/using_helm/#tiller (Note: This documentation is for Helm v2, which is outdated)
Helm documentation on the architecture of Helm v3: https://helm.sh/docs/intro/architecture/

Question: 423 CertyIQ

SIMULATION -
You need to create a virtual machine template in an Azure DevTest Labs environment named az400-123456789-dtl1. The template must be based on Windows
Server 2019 Datacenter. Virtual machines created from the template must include the selenium tool and the Google Chrome browser.
To complete this task, sign in to the Microsoft Azure portal.

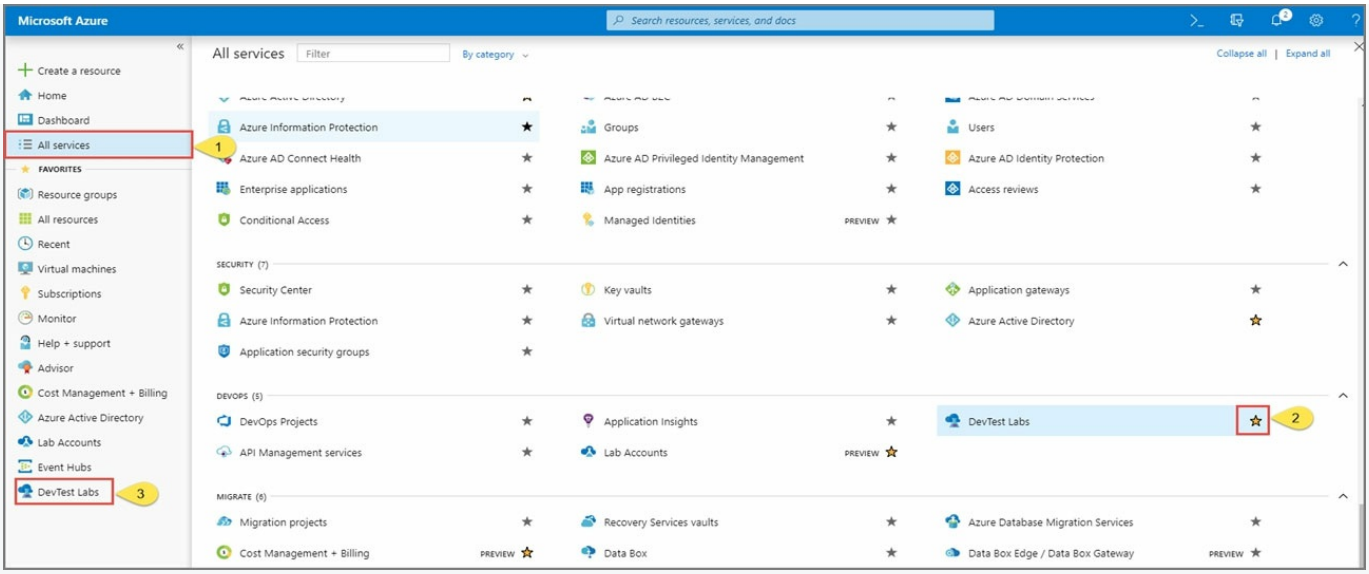
Answer: B

Explanation:

Answer is no Configure a build policy instead

1. Open Microsoft Azure Portal

2. Select All Services, and then select DevTest Labs in the DEVOPS section.



3. From the list of labs, select the az400-123456789-dtl1 lab.

4. On the home page for your lab, select + Add on the toolbar.

5. Select the Windows Server 2019 Datacenter base image for the VM.

6. Select automation options at the bottom of the page above the Submit button.

7. You see the Azure Resource Manager template for creating the virtual machine.

8. The JSON segment in the resources section has the definition for the image type you selected earlier.

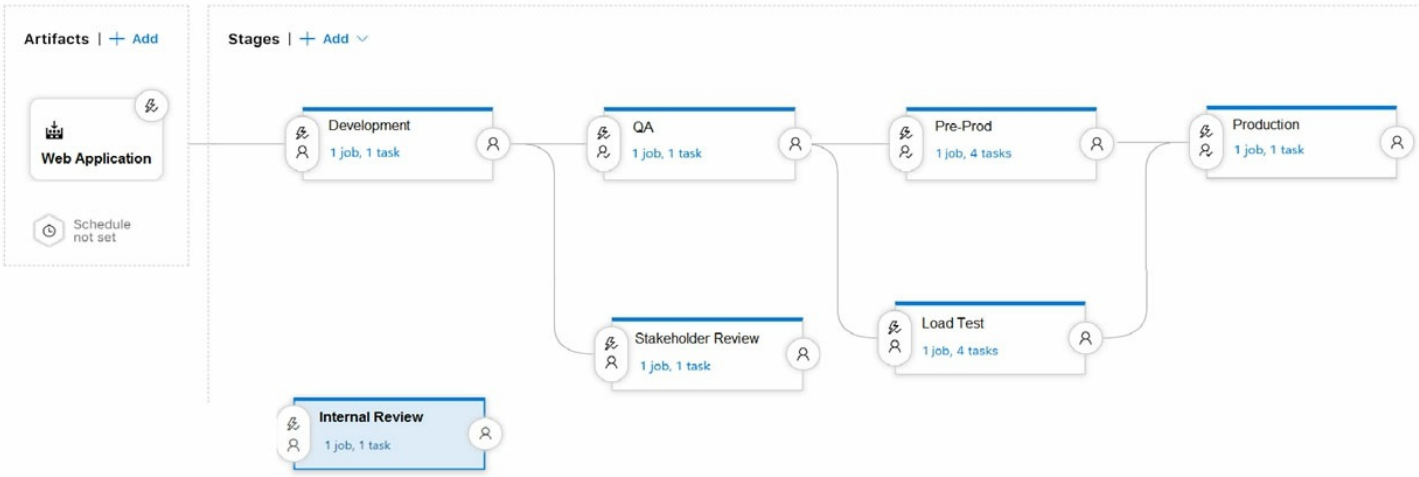
Reference:

https://docs.microsoft.com/bs-cyrl-ba/azure//lab-services/devtest-lab-vm-powershell

Question: 424

CertyIQ

HOTSPOT -
You are configuring a release pipeline in Azure DevOps as shown in the exhibit.



Use the drop-down menus to select the answer choice that answers each question based on the information presented in the graphic.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

How many stages have triggers set?

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

Which component should you modify to enable continuous delivery?

- The Development stage
- The Internal Review stage
- The Production stage
- The Web Application artifact

Answer:

Answer Area

How many stages have triggers set?

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

Which component should you modify to enable continuous delivery?

- The Development stage
- The Internal Review stage
- The Production stage
- The Web Application artifact

Explanation:

7

The Web Application artifact.

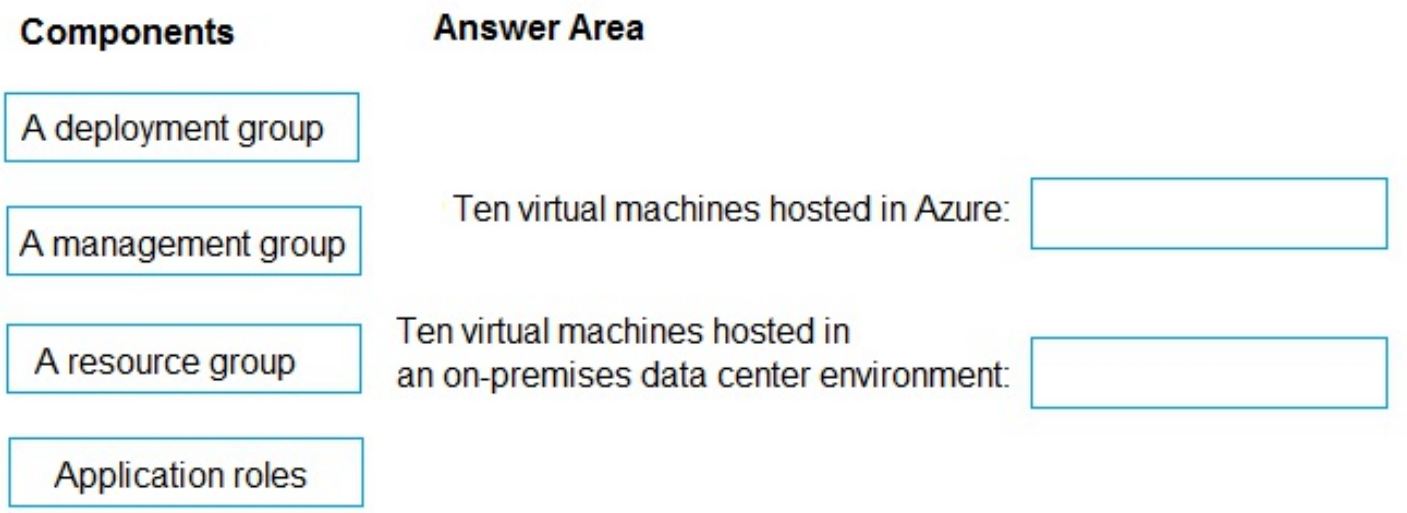
https://learn.microsoft.com/en-us/azure/devops/pipelines/release/triggers?view=azure-devops

Question: 425

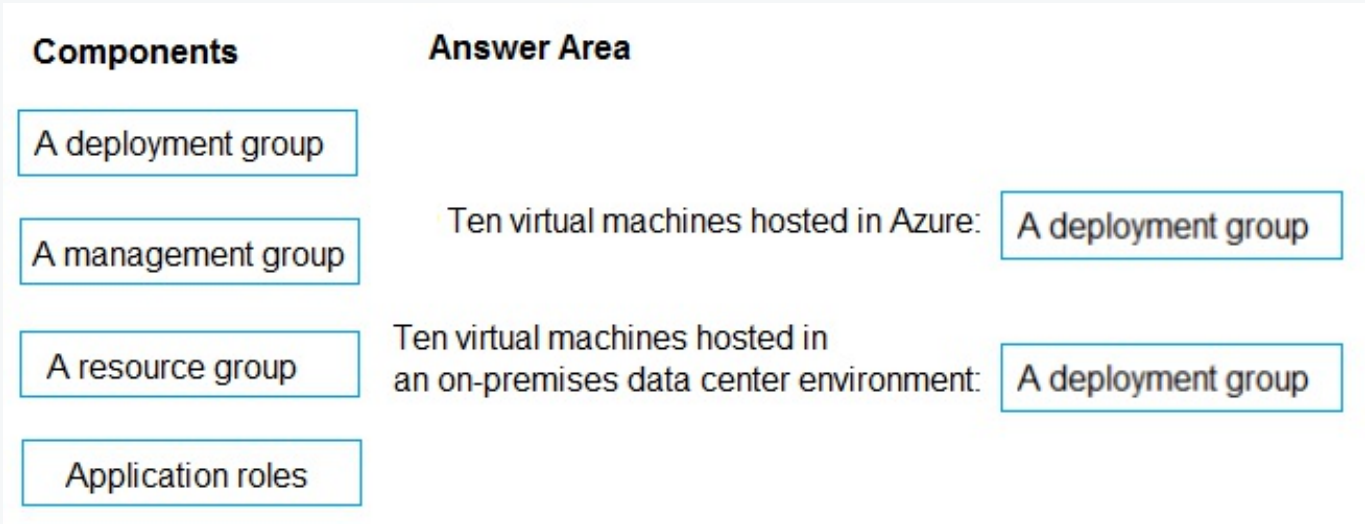
CertyIQ

DRAG DROP -
Your company plans to deploy an application to the following endpoints:
- Ten virtual machines hosted in Azure
- Ten virtual machines hosted in an on-premises data center environment

All the virtual machines have the Azure Pipelines agent.
You need to implement a release strategy for deploying the application to the endpoints.
What should you recommend using to deploy the application to the endpoints? To answer, drag the appropriate components to the correct endpoints. Each component may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:
Box 1: A deployment group -
When authoring an Azure Pipelines or TFS Release pipeline, you can specify the deployment targets for a job using a deployment group.
If the target machines are Azure VMs, you can quickly and easily prepare them by installing the Azure Pipelines Agent Azure VM extension on each of the VMs, or by using the Azure Resource Group Deployment task in your release pipeline to create a deployment group dynamically.

Box 2: A deployment group -

Reference:
https://docs.microsoft.com/en-us/azure/devops/pipelines/release/deployment-groups

Question: 426 CertyIQ

You plan to use Terraform to deploy an Azure resource group from a Windows system.
You need to install the required frameworks to support the planned deployment.
Which two frameworks should you install? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Vault
B. Terratest
C. Node.js
D. Yeoman
E. Tiller

Answer: BD

Explanation:

The correct answer is **BD**. Here's why:

B. Terratest: Terratest is a Go library that makes it easier to write automated tests for your infrastructure code, including Terraform configurations. While not strictly required for deploying a resource group, it is essential for verifying that the deployment worked as intended, and for preventing regressions when you make future changes to your infrastructure as code. Good DevOps practices emphasize testing infrastructure changes before deploying them to production. While not required for the deployment itself, testing is essential for a robust and reliable infrastructure as code workflow.

D. Yeoman: Yeoman is a scaffolding tool for modern webapps. While not directly related to deploying Azure resources with Terraform, it is a generator that is useful for scaffolding projects and configurations. It is not directly needed for Terraform deployments but it helps in creating a well-structured development and operational workflow. It helps in generating project structures for deployment.

Let's look at why the other options are incorrect:

A. Vault: Vault is a tool for secrets management. It's valuable for securely storing and managing sensitive information like API keys, passwords, and certificates used by your infrastructure, but it's not a fundamental requirement for initially deploying an Azure resource group using Terraform. You can create a resource group without using Vault.

C. Node.js: Node.js is a JavaScript runtime environment. While Terraform can interact with Node.js applications, Node.js is not needed for Azure resource group deployment.

E. Tiller: Tiller was a component of Helm, a package manager for Kubernetes. As you're deploying an Azure resource group, and not working with Kubernetes, Tiller is not needed in your setup.

Therefore, to install the required frameworks to support the planned deployment with Terraform, Yeoman (D) and Terratest (B) would be necessary frameworks to install.

Question: 427 CertyIQ

SIMULATION -
You plan to implement a CI/CD strategy for an Azure Web App named az400-123456789-main.
You need to configure a staging environment for az400-123456789-main.
To complete this task, sign in to the Microsoft Azure portal.

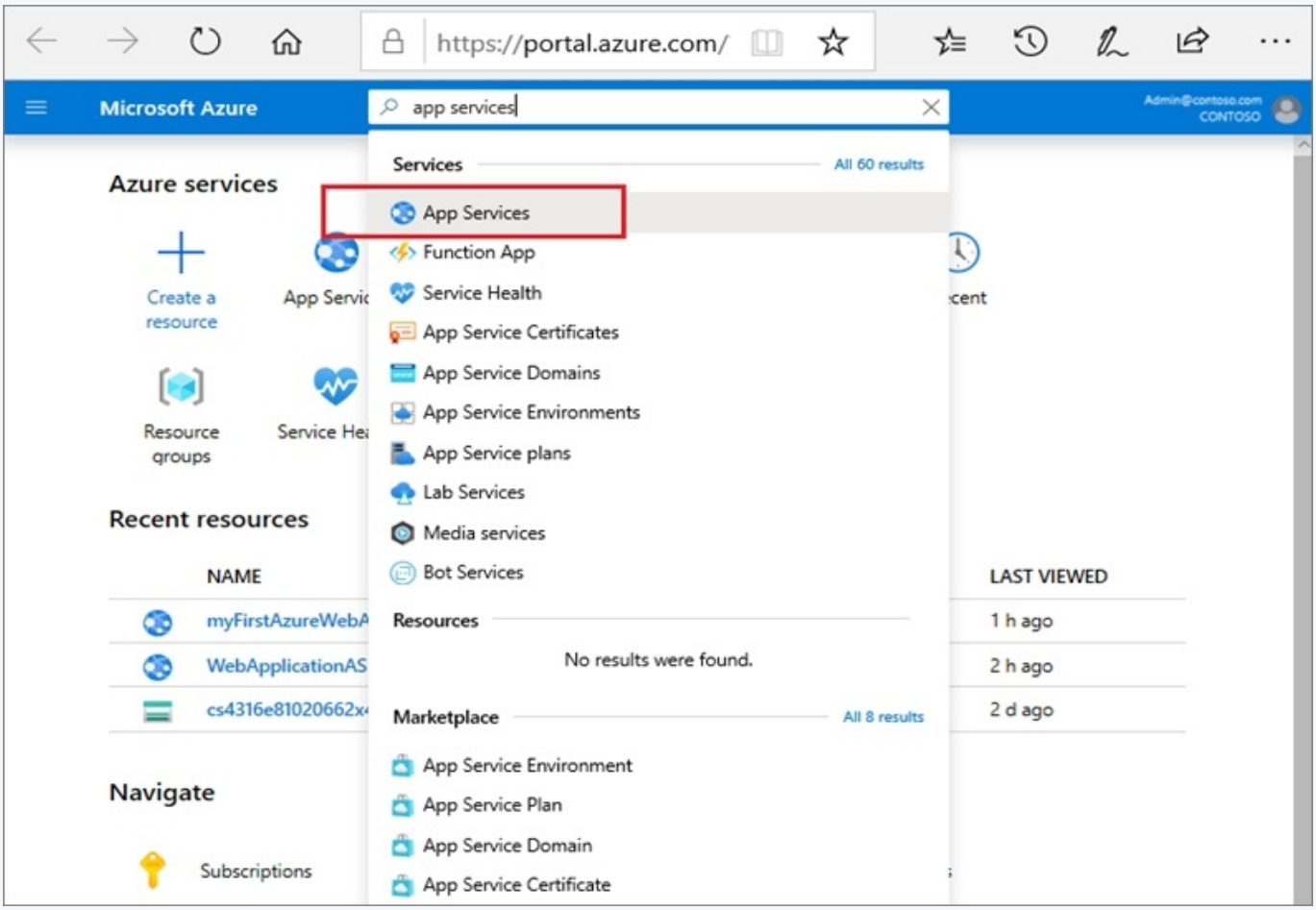
Answer:

See explanation below.

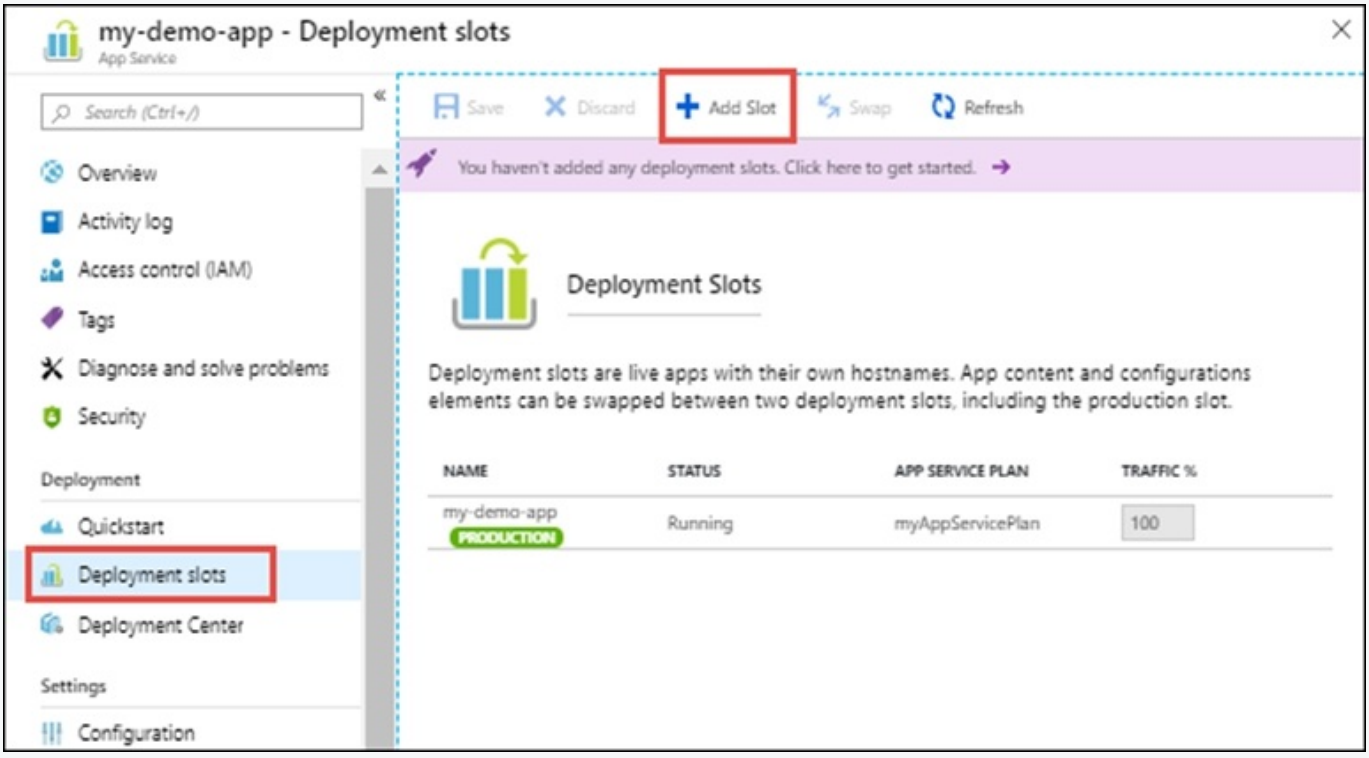
Explanation:

Add a slot -

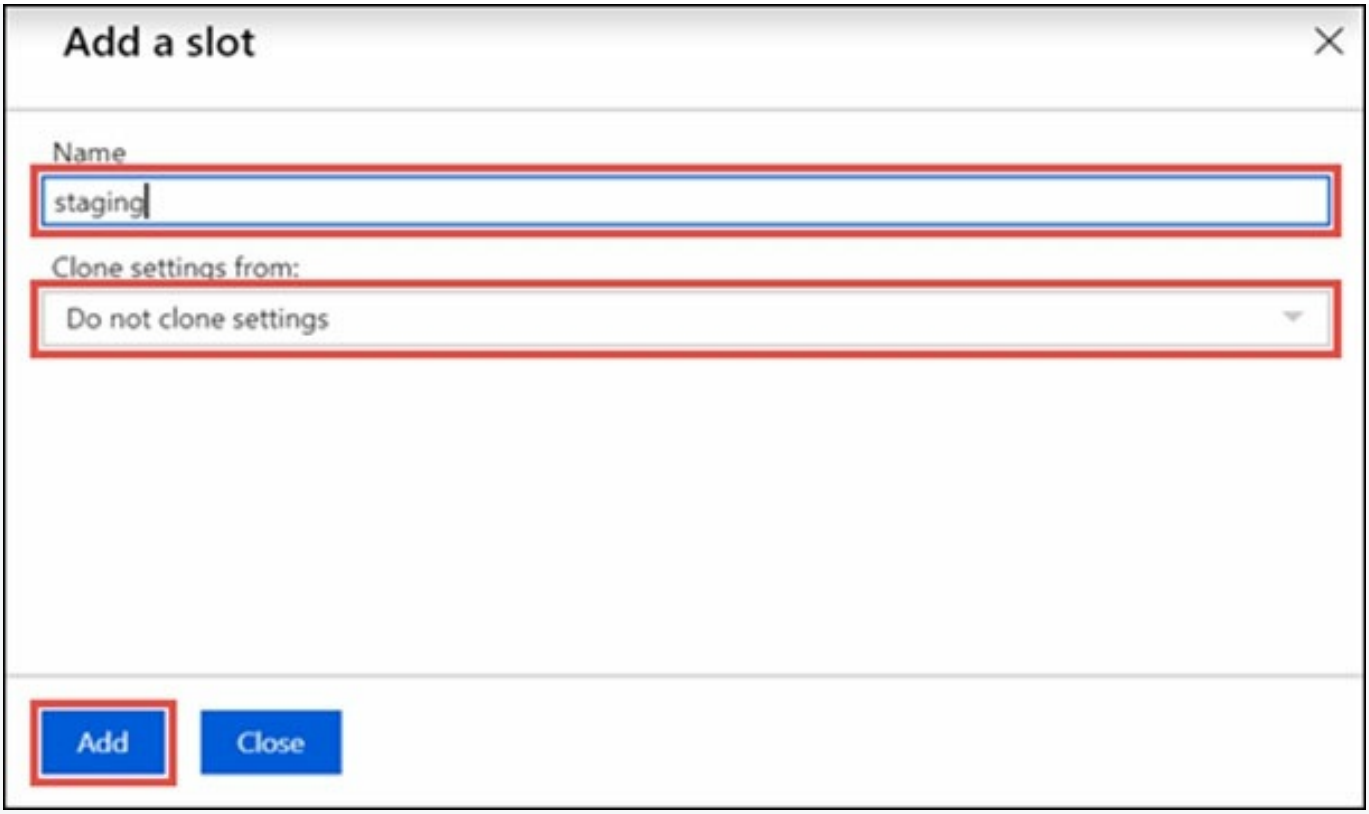
1. In the Azure portal, search for and select App Services and select your app az400-123456789-main.



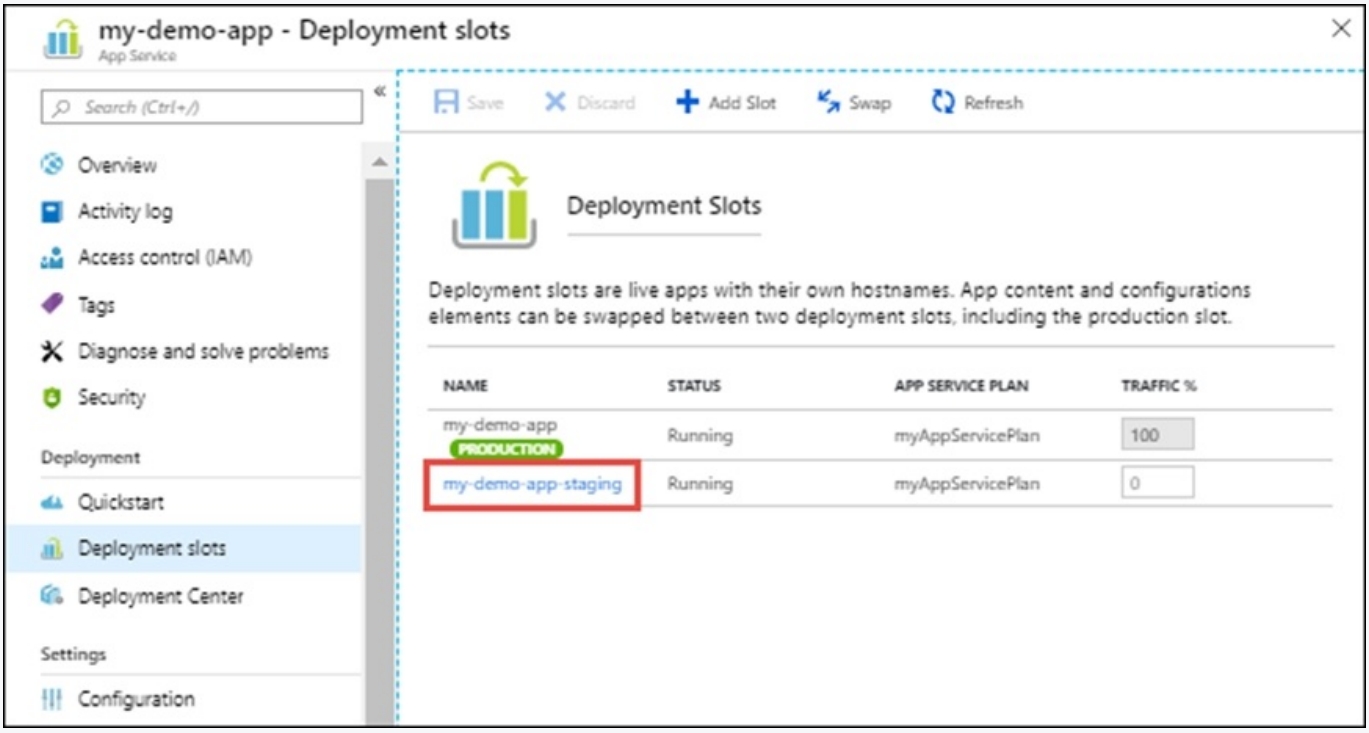
2. In the left pane, select Deployment slots > Add Slot.



3. In the Add a slot dialog box, give the slot a name, and select whether to clone an app configuration from another deployment slot. Select Add to continue.



4. After the slot is added, select Close to close the dialog box. The new slot is now shown on the Deployment slots page.



Reference:
https://docs.microsoft.com/en-us/azure/app-service/deploy-staging-slots

Question: 428 CertyIQ

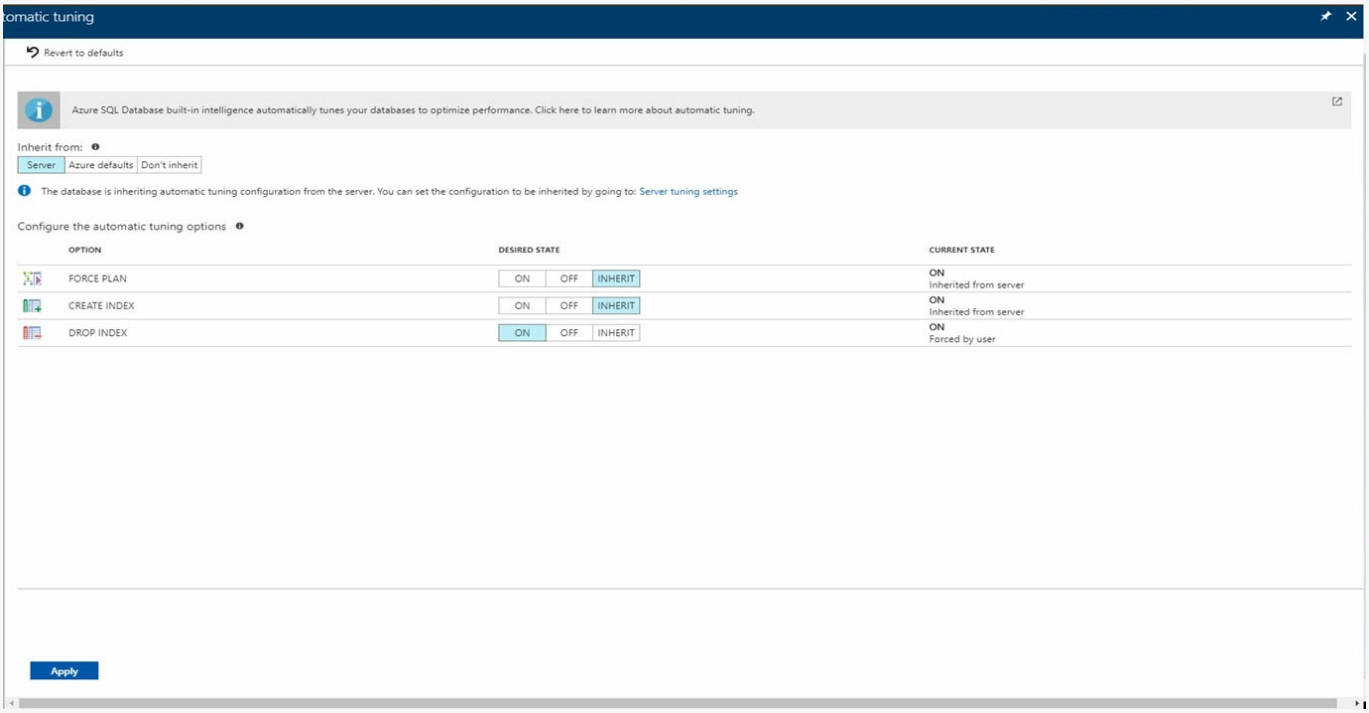
SIMULATION -
You have several apps that use an Azure SQL Database named db1.
You need to ensure that queries to db1 are tuned by Azure over time. The solution must only apply to db1.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:

1. To enable automatic tuning on a single database, navigate to the database in the Azure portal and select Automatic tuning.



2. Select the automatic tuning options you want to enable and select Apply.

Note: Individual automatic tuning settings can be separately configured for each database. You can manually configure an individual automatic tuning option, or specify that an option inherits its settings from the server.

Reference:
https://docs.microsoft.com/en-us/azure/azure-sql/database/automatic-tuning-enable

Question: 429 CertyIQ

HOTSPOT -
You use Azure Pipelines to manage the build and deployment of apps.
You are planning the release strategies for a new app.
You need to choose strategies for the following scenarios:
Releases will be made available to users who are grouped by their tolerance for software faults.

- Code will be deployed to enable functionality that will be available in later releases of the app.
- When a new release occurs, the existing deployment will remain active to minimize recovery time if a return to the previous version is required.

Which strategy should you choose for each scenario? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

Releases will be made available to users who are grouped by their tolerance for software faults:

- Progressive exposure
- Blue/green
- Feature flags

Code will be deployed to enable functionality that will be available in later releases of the app:

- Progressive exposure
- Blue/green
- Feature flags

When a new release occurs, the existing deployment will remain active to minimize recovery time if a return to the previous version is required:

- Progressive exposure
- Blue/green
- Feature flags

Answer:

Answer Area

Releases will be made available to users who are grouped by their tolerance for software faults:

- Progressive exposure
- Blue/green
- Feature flags

Code will be deployed to enable functionality that will be available in later releases of the app:

- Progressive exposure
- Blue/green
- Feature flags

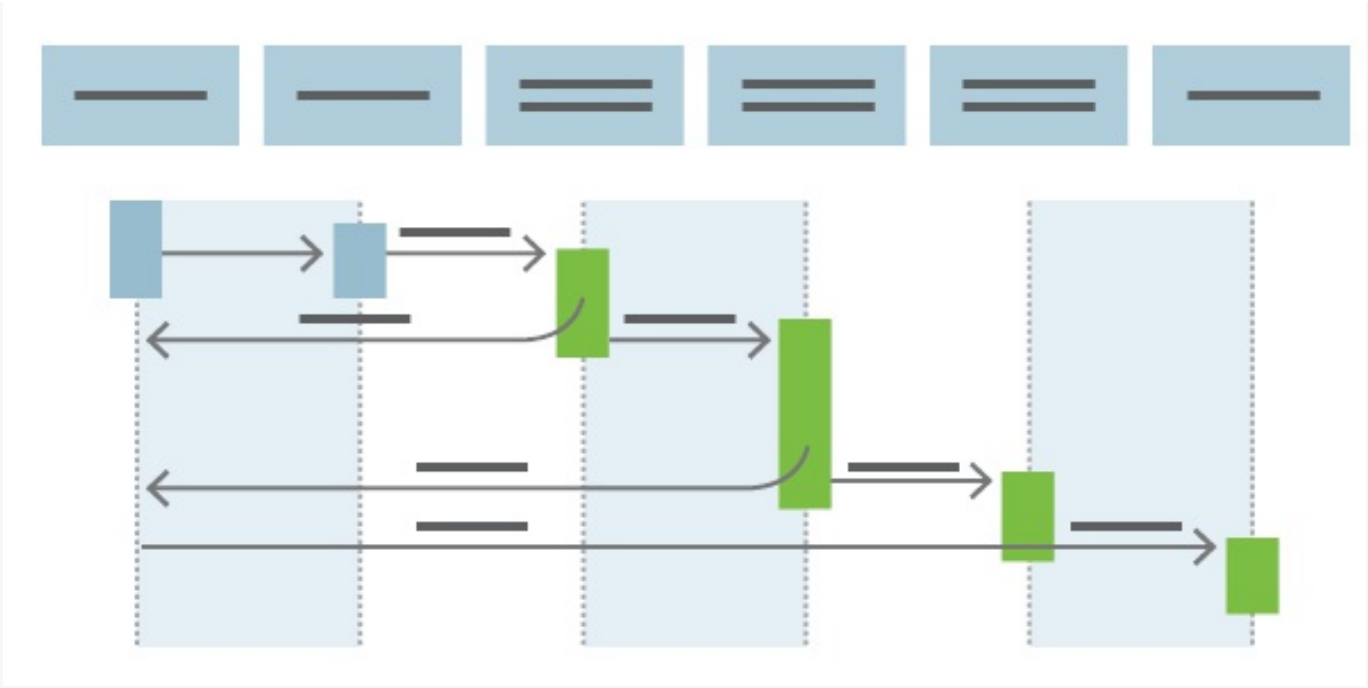
When a new release occurs, the existing deployment will remain active to minimize recovery time if a return to the previous version is required:

- Progressive exposure
- Blue/green
- Feature flags

Explanation:

Box 1: Progressive exposure -

Continuous Delivery may sequence multiple deployment rings for progressive exposure (also known as controlling the blast radius). Progressive exposure groups users who get to try new releases to monitor their experience in rings. The first deployment ring is often a canary used to test new versions in production before a broader rollout. CD automates deployment from one ring to the next and may optionally depend on an approval step, in which a decision maker signs off on the changes electronically. CD may create an auditable record of the approval in order to satisfy regulatory procedures or other control objectives.



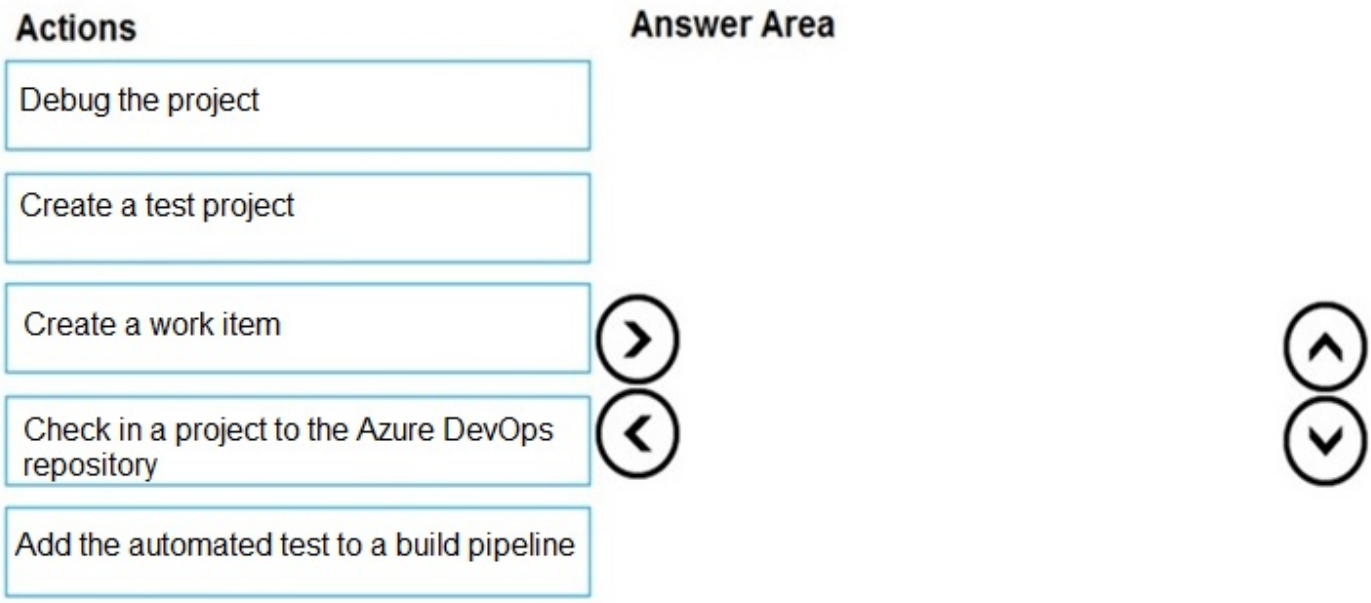
Box 2: Feature flags -
Feature flags support a customer-first DevOps mindset, to enable (expose) and disable (hide) features in a solution, even before they are complete and ready for release.

Box 3: Blue/green -
Blue/green deployments which means that instead of replacing the previous version (here we refer to this version as blue), we bring up the new version (here referred to as the green version) next to the existing version, but not expose it to the actual users right away. On the condition of having successfully validated that the green version works correctly, we will promote this version to the public version by changing the routing configuration without downtime. If something is wrong with the green version we can revert back without users every noticing interruptions.

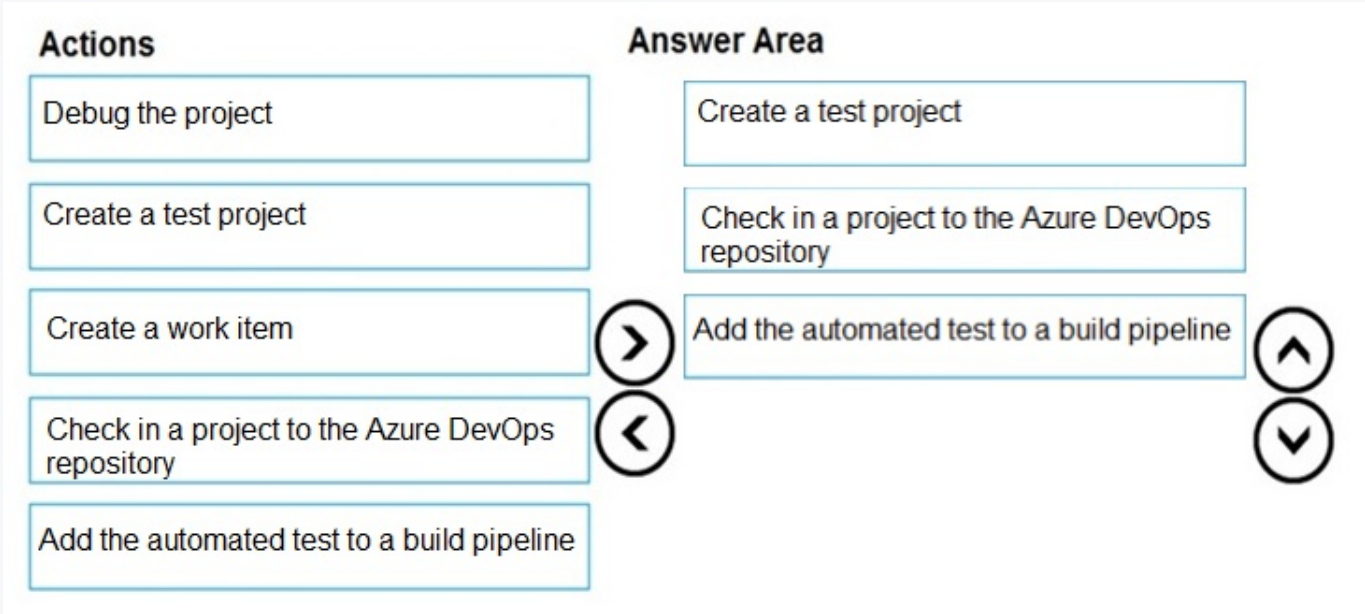
Reference:
https://docs.microsoft.com/en-us/azure/devops/learn/what-is-continuous-delivery
https://docs.microsoft.com/en-us/azure/devops/migrate/phase-features-with-feature-flags
https://medium.com/@denniszielke/continuous-kubernetes-blue-green-deployments-on-azure-using-nginx-appgateway-or-trafficmanager-4490bce29cb

Question: 430 CertyIQ

DRAG DROP -
You have a project in Azure DevOps.
You need to associate an automated test to a test case.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



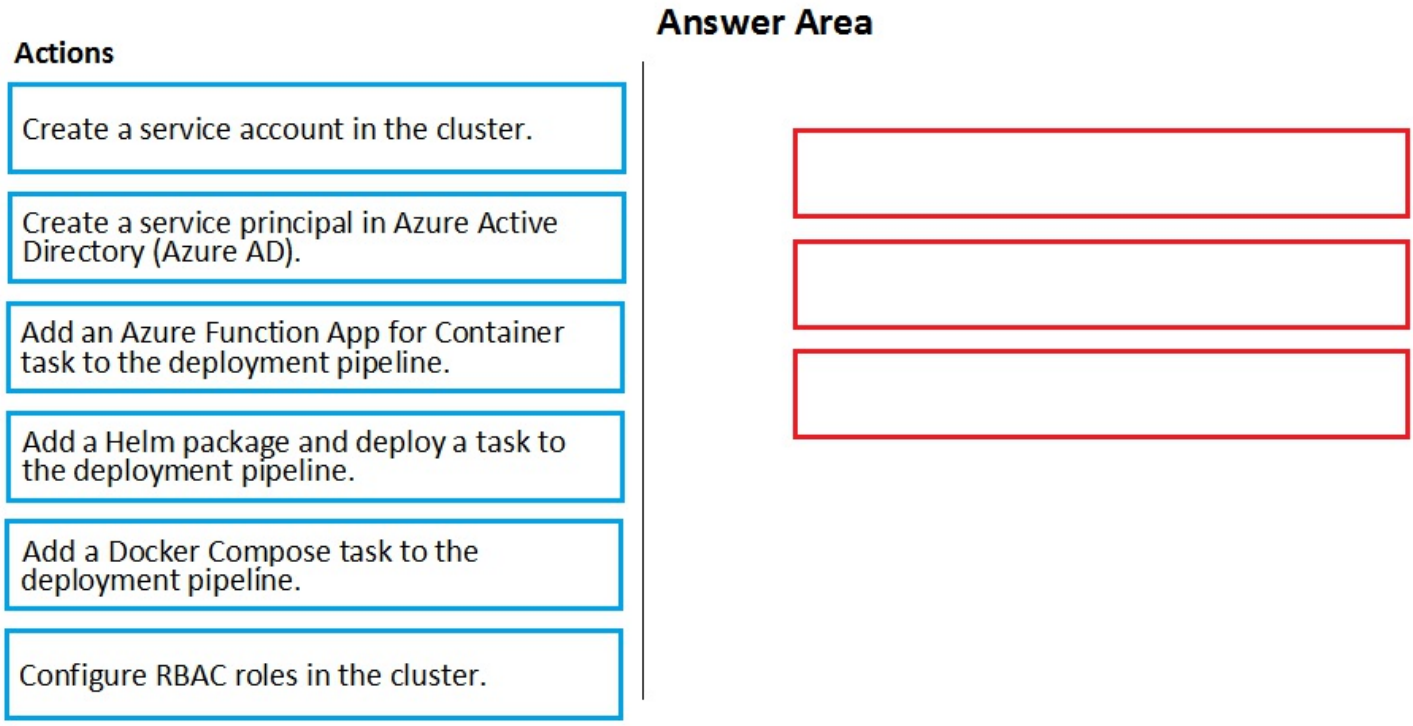
Explanation:
The process to associate an automated test with a test case is:
1. Create a test project containing your automated test. What types of tests are supported?
2. Check your test project into an Azure DevOps or Team Foundation Server (TFS) repository.
3. Create a build pipeline for your project, ensuring that it contains the automated test. What are the differences if I am still using a XAML build?
4. Use Visual Studio Enterprise or Professional 2017 or a later version to associate the automated test with a test case as shown below. The test case must have been added to a test plan that uses the build you just defined.

Reference:
https://docs.microsoft.com/en-us/azure/devops/test/associate-automated-test-with-test-case

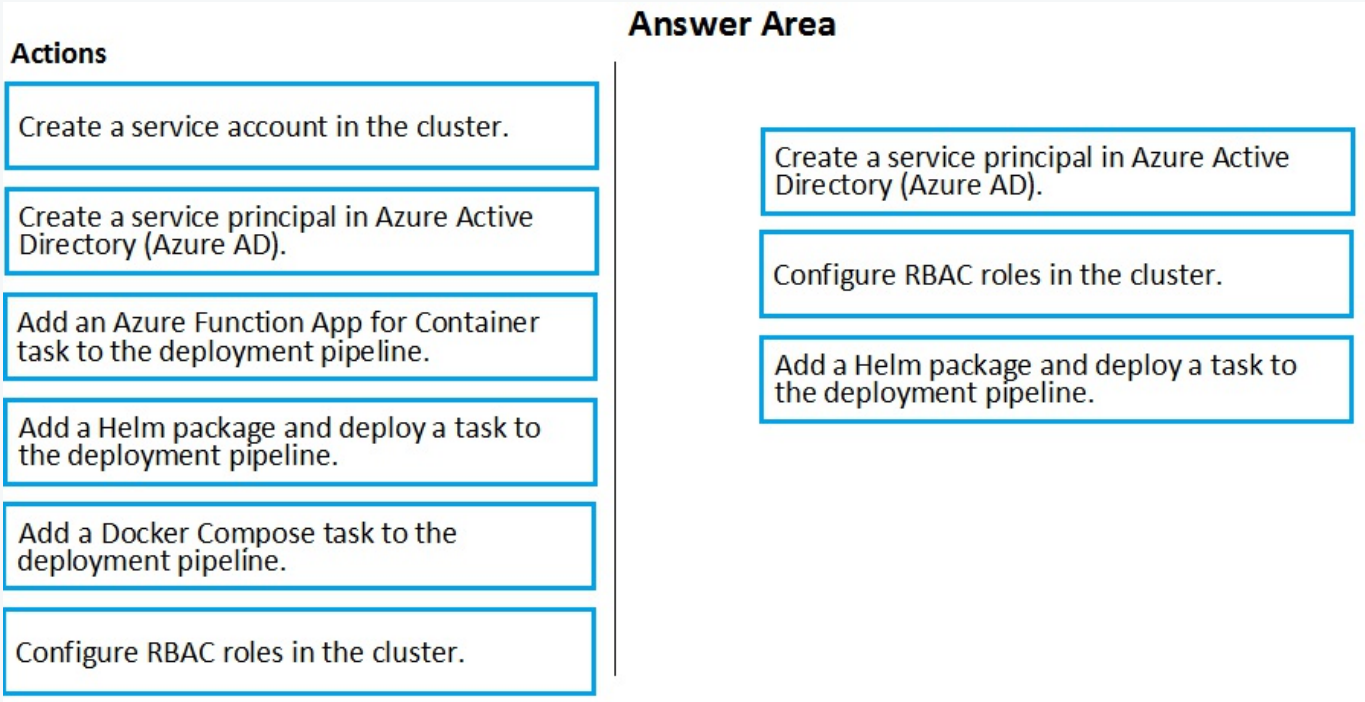
Question: 431 CertyIQ

DRAG DROP -
You have an Azure Kubernetes Service (AKS) cluster.
You need to deploy an application to the cluster by using Azure DevOps.

Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

You can set up a CI/CD pipeline to deploy your apps on a Kubernetes cluster with Azure DevOps by leveraging a Linux agent, Docker, and Helm.

Step 1: Create a service principle in Azure Active Directory (Azure AD)

We need to assign 3 specific service principals with specific Azure Roles that need to interact with our ACR and our AKS.

Create a specific Service Principal for our Azure DevOps pipelines to be able to push and pull images and charts of our ACR.

Create a specific Service Principal for our Azure DevOps pipelines to be able to deploy our application in our AKS.

Step 2: Configure RBAC roles in the cluster.

Step 3: Add a helm package and deploy a task to the deployment pipeline.

Reference:

https://cloudblogs.microsoft.com/opensource/2018/11/27/tutorial-azure-devops-setup-cicd-pipeline-kubernetes-docker-helm/

Question: 432 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an approval process that contains a condition. The condition requires that releases be approved by a team leader before they are deployed.
You have a policy stating that approvals must occur within eight hours.
You discover that deployment fail if the approvals take longer than two hours.
You need to ensure that the deployments only fail if the approvals take longer than eight hours.
Solution: From Post-deployment conditions, you modify the Time between re-evaluation of gates option.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution is incorrect because modifying the "Time between re-evaluation of gates" option in Post-deployment conditions doesn't address the problem of deployments failing when approvals take longer than two hours. This setting controls how frequently the system checks if the gate conditions are still met after a deployment has started, not the maximum time allowed for approvals before deployment. The problem is that the approvals are expiring too quickly, causing the initial deployment check to fail.

The correct approach is to adjust the timeout settings associated with the approval process itself. Azure DevOps allows configuration of timeouts for approvals. This involves finding the specific approval task or stage in the release pipeline and increasing the timeout duration for the approval step. This will ensure that the deployment only fails if the approvals take longer than the organizational policy of eight hours. The "Time between re-evaluation of gates" is used for ongoing validation after deployment, not for pre-deployment approval limits. Therefore, the proposed solution does not meet the goal.

Relevant documentation for understanding Azure DevOps release pipelines and approval processes can be found here:

Using Approvals and Gates: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops

Release gates and approvals: https://learn.microsoft.com/en-us/azure/devops/release-notes/2019/sprint-150-update?view=azure-devops#release-gates-and-approvals

Question: 433 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an approval process that contains a condition. The condition requires that releases be approved by a team leader before they are deployed.
You have a policy stating that approvals must occur within eight hours.
You discover that deployment fail if the approvals take longer than two hours.
You need to ensure that the deployments only fail if the approvals take longer than eight hours.
Solution: From Pre-deployment conditions, you modify the Time between re-evaluation of gates option.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution of modifying the "Time between re-evaluation of gates" option in the Pre-deployment conditions does not meet the stated goal of ensuring deployments only fail if approvals take longer than eight hours.

Here's why:

The "Time between re-evaluation of gates" setting controls how frequently Azure DevOps re-evaluates the pre-deployment gates. These gates are automated checks that must pass before a deployment can proceed. This setting primarily aims to ensure that conditions are continuously monitored and any changes are detected promptly.

The problem states that deployments fail if approvals take longer than two hours, while the policy allows for up to eight hours. The solution needs to extend the allowed approval time before failure, not influence how frequently gates are re-evaluated. Adjusting the re-evaluation interval will only control how often the check for approval status occurs; it won't change the underlying two-hour limit causing the failure.

The crucial element here is to modify the timeout associated with the approval gate or the overall deployment pipeline. There should be a setting explicitly defining how long the system waits for an approval before failing the deployment. This timeout value is what needs to be changed to eight hours. Simply adjusting the re-evaluation frequency of the gate will not prevent the early failure. In Azure DevOps, this could be configured in the approval task or stage settings, defining the maximum wait time for approvals.

Therefore, modifying the re-evaluation interval of gates won't change the approval timeout, making the solution ineffective.

Relevant resources for further research:

Approvals and checks: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops&tabs=check-rules
Using gates and approvals in release pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/approvals/index?view=azure-devops

Question: 434 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an approval process that contains a condition. The condition requires that releases be approved by a team leader before they are deployed.
You have a policy stating that approvals must occur within eight hours.
You discover that deployment fail if the approvals take longer than two hours.
You need to ensure that the deployments only fail if the approvals take longer than eight hours.
Solution: From Pre-deployment conditions, you modify the Timeout setting for pre-deployment approvals.
Does this meet the goal?

A. Yes
B. No

Answer: A

Explanation:

The proposed solution addresses the problem of deployments failing prematurely due to an overly aggressive approval timeout. By adjusting the Timeout setting within the Pre-deployment conditions of the release pipeline, the allowed duration for approvals can be extended. Specifically, changing the timeout from two hours to eight hours aligns with the stated policy, ensuring deployments only fail if approvals exceed the intended eight-hour limit.

Pre-deployment conditions act as gatekeepers, controlling when a deployment can proceed to the next stage. Approvals, as a type of pre-deployment condition, mandate that specified users or groups must explicitly approve a deployment before it can continue. The Timeout setting associated with these approvals dictates the maximum time the system will wait for an approval before automatically rejecting the deployment. Modifying this setting directly influences the duration for which the system waits for the approval, thereby directly addressing the premature failure issue. Therefore, the solution fulfills the requirement by preventing deployments from failing if approvals are received within the correct eight-hour timeframe.

References:

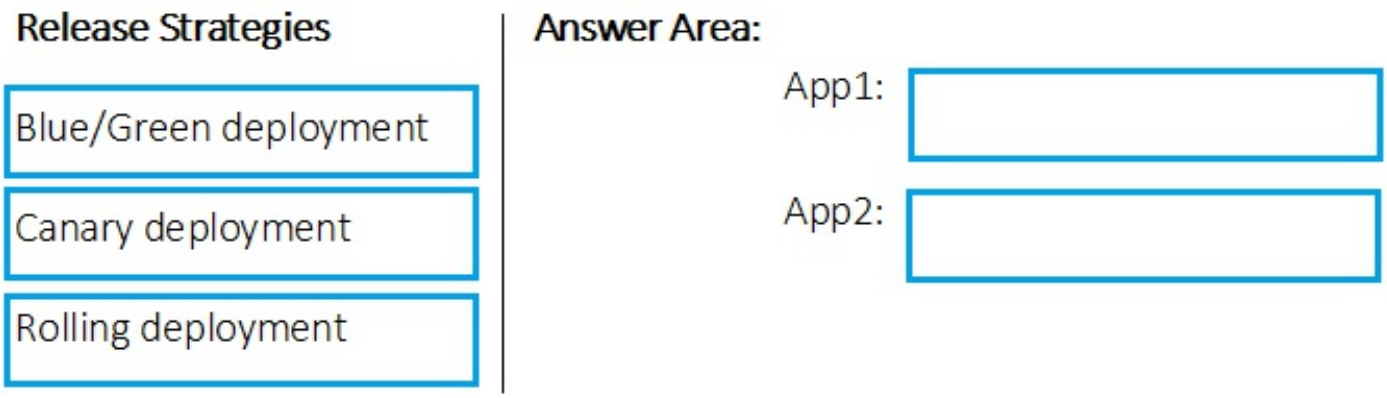
Microsoft Azure DevOps Documentation on Approvals and Checks: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops&tabs=check-options

Question: 435 CertyIQ

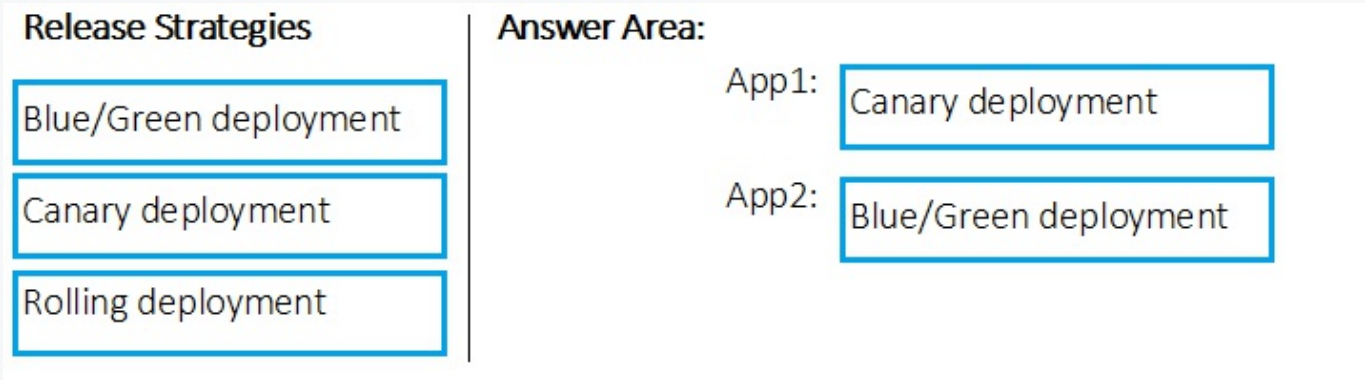
DRAG DROP -
You are defining release strategies for two applications as shown in the following table.

Application name	Goal
App1	Failure of App1 has a major impact on your company. You need a small group of users, who opted in to a testing App1, to test new releases of the application.
App2	You need to minimize the time it takes to deploy new releases of App2, and you must be able to roll back as quickly as possible.

Which release strategy should you use for each application? To answer, drag the appropriate release strategies to the correct applications. Each release strategy may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

App1: Canary deployment .

With canary deployment, you deploy a new application code in a small part of the production infrastructure. Once the application is signed off for release, only a few users are routed to it. This minimizes any impact.

With no errors reported, the new version can gradually roll out to the rest of the infrastructure.

App2: Blue/Green deployment.

Blue/Green is for sure faster when you need to roll back the changes.

Question: 436

CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an approval process that contains a condition. The condition requires that releases be approved by a team leader before they are deployed.
You have a policy stating that approvals must occur within eight hours.
You discover that deployment fail if the approvals take longer than two hours.
You need to ensure that the deployments only fail if the approvals take longer than eight hours.
Solution: From Post-deployment conditions, you modify the Timeout setting for post-deployment approvals.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The proposed solution suggests modifying the Timeout setting for post-deployment approvals to address a problem with approvals taking too long and causing deployment failures. However, the problem described involves approvals before deployment (the team leader's approval). Post-deployment conditions control what happens after a deployment is complete, not before. Therefore, modifying post-deployment timeout settings will not affect the pre-deployment approval process or resolve the specified issue. The timeout for pre-deployment approvals is typically managed within the release pipeline's pre-deployment conditions or approval gates. There is no direct setting called "Timeout" for approvals. Instead, the appropriate configuration involves setting timeout periods within the approval tasks of the release pipeline or by leveraging Azure Policy, if that's applicable for governing deployment approvals. The statement that deployment fail if the approvals take longer than two hours suggests that there might be an existing timeout already set up somewhere, but it is not controlled by post-deployment conditions.

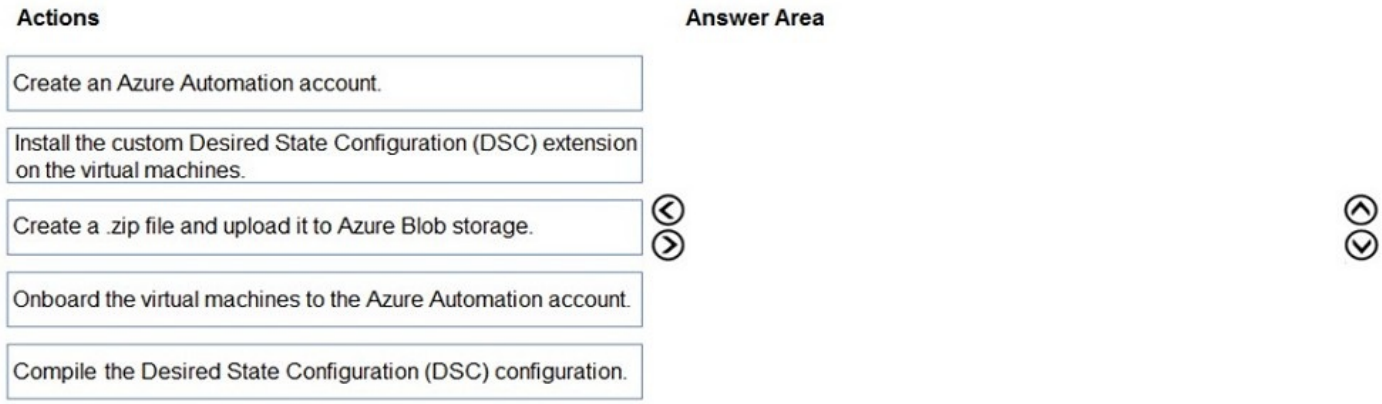
Therefore, the proposed solution does not meet the goal.

Supporting Links:

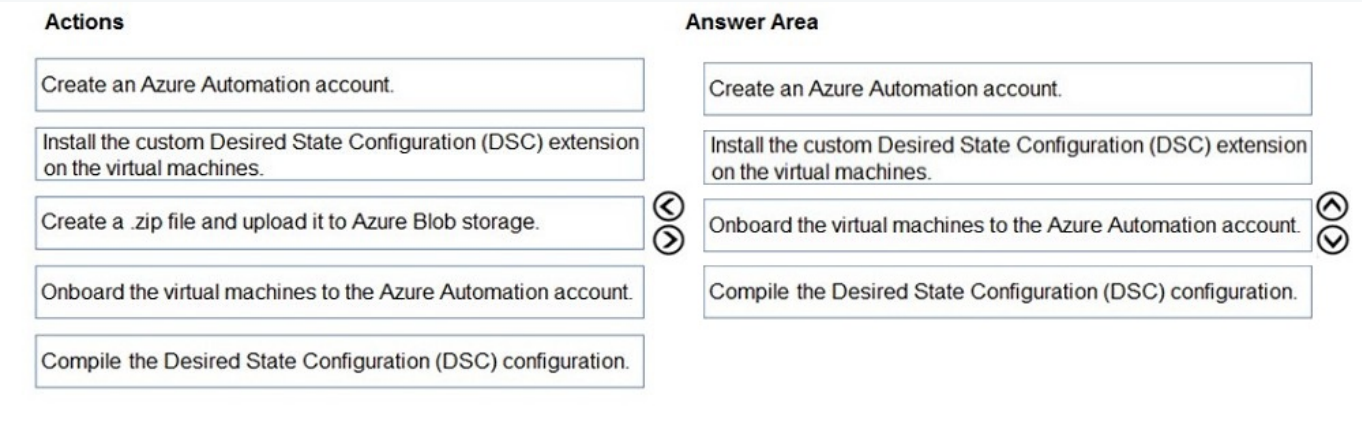
Approvals and gates: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops&tabs=check-types
Release pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/?view=azure-devops

Question: 437 CertyIQ

DRAG DROP -
You have an Azure DevOps organization named Contoso.
You have 10 Azure virtual machines that run Windows Server 2019. The virtual machines host an application that you build and deploy by using Azure Pipelines.
Each virtual machine has the Web Server (IIS) role installed and configured.
You need to ensure that the web server configurations on the virtual machines is maintained automatically. The solution must provide centralized management of the configuration settings and minimize management overhead.
Which four actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Step1: Create an Azure Automation account.

An Azure Automation account is required.

Step 2: Install the custom Desired State Configuration (DSC) extension on the virtual machines

Under the hood, and without an administrator having to remote into a VM, the Azure VM Desired State Configuration extension registers the VM with Azure

Automation State Configuration.

Step 3: Onboard the virtual machines to the Azure Automation account.

Step 4: Compile the Desired State Configuration (DSC) configuration.

Create a DSC configuration and compile it.

Reference:

https://docs.microsoft.com/en-us/azure/automation/automation-dsc-onboarding

Question: 438

CertyIQ

You have a free tier of an Azure DevOps organization named Contoso. Contoso contains 10 private projects. Each project has multiple jobs with no dependencies. The build process requires access to resource files located in an on-premises file system.

You frequently run the jobs on five self-hosted agents but experience long build times and frequently queued builds.

You need to minimize the number of queued builds and the time it takes to run the builds.

What should you do?

A. Configure the pipelines to use the Microsoft-hosted agents.

B. Register additional self-hosted agents.

C. Purchase self-hosted parallel jobs.

D. Purchase Microsoft-hosted parallel jobs.

Answer: C

Explanation:

The correct answer is **C. Purchase self-hosted parallel jobs.**

Here's a detailed justification:

The primary problem is long build times and frequent queued builds, even with five self-hosted agents. This indicates a bottleneck in build execution capacity, likely because the agents are being utilized sequentially. Parallel jobs allow multiple build pipelines (or parts of a single pipeline) to run concurrently.

A. **Configure the pipelines to use the Microsoft-hosted agents.** This is unsuitable. The build process requires access to on-premises resource files. Microsoft-hosted agents lack direct access to these on-premises resources without complex network configurations (VPNs, etc.), which adds significant overhead and complexity. Given the existing self-hosted agent infrastructure, leveraging it is more efficient.

B. **Register additional self-hosted agents.** While adding more agents can help to a degree, the Azure DevOps organization (free tier) limits the number of agents you can effectively use without parallel jobs. Without parallel jobs, the added agents will largely remain idle as only one build can run at a time within the organization. The root issue is the serial execution, not just the number of agents.

C. **Purchase self-hosted parallel jobs.** This directly addresses the bottleneck. Parallel jobs enable multiple builds to run concurrently across the available self-hosted agents. This will dramatically reduce build times and minimize the number of queued builds, because agents can work simultaneously instead of waiting for each other. Furthermore, since the resource files are on-premises, self-hosted agents are essential. Purchasing parallel jobs effectively unlocks the potential of the existing agent pool and any future agents added.

D. **Purchase Microsoft-hosted parallel jobs.** This is unsuitable because it doesn't solve the on-premises resource access requirement. Using Microsoft-hosted agents would necessitate a complex and potentially insecure setup to provide access to the local file system.

In summary, purchasing self-hosted parallel jobs provides the needed concurrency without the added complexity and security concerns of trying to integrate Microsoft-hosted agents with on-premises resources. It allows optimal utilization of the existing self-hosted agent infrastructure.

Authoritative Links:

Azure DevOps Parallel Jobs: https://learn.microsoft.com/en-us/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops
Self-hosted agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops

Question: 439 CertyIQ

SIMULATION -
You need to ensure that an Azure web app named az400-123456789-main supports rolling upgrades. The solution must ensure that only 10 percent of users who connect to az400-123456789-main use update versions of the app. The solution must minimize administrative effort.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

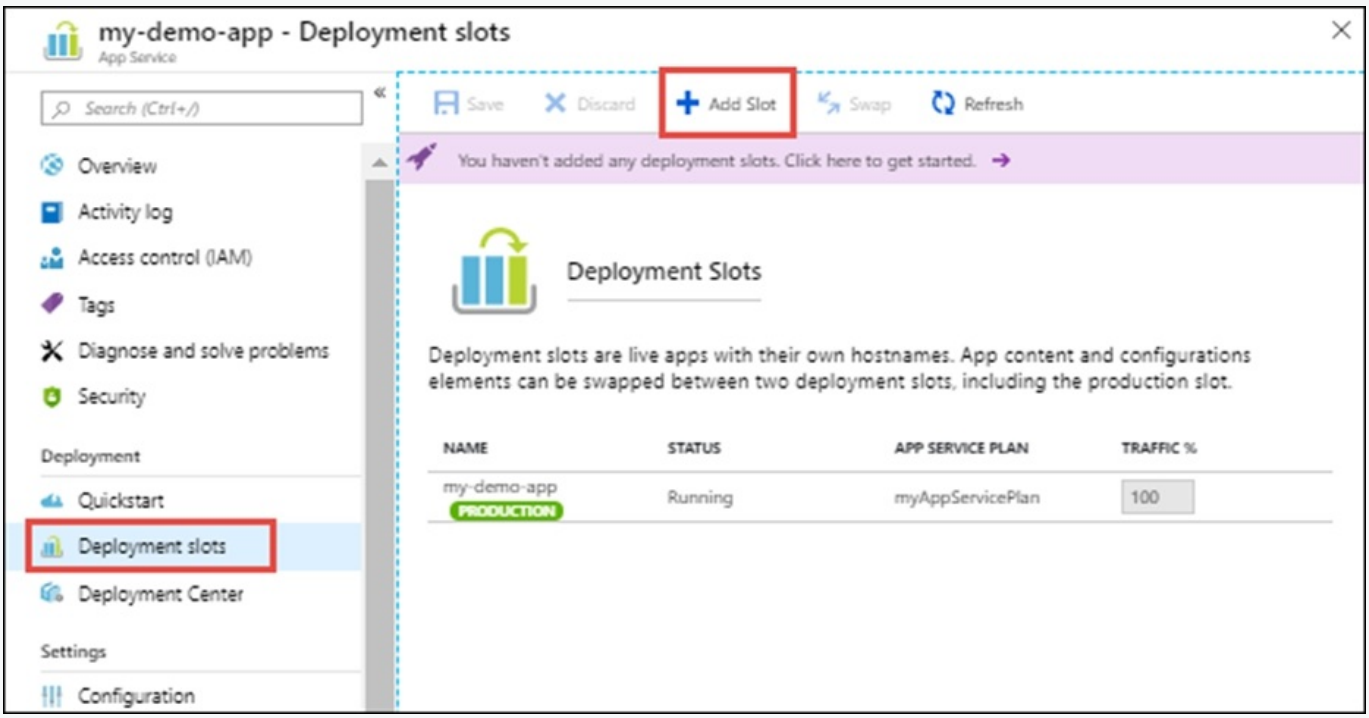
See explanation below.

Explanation:

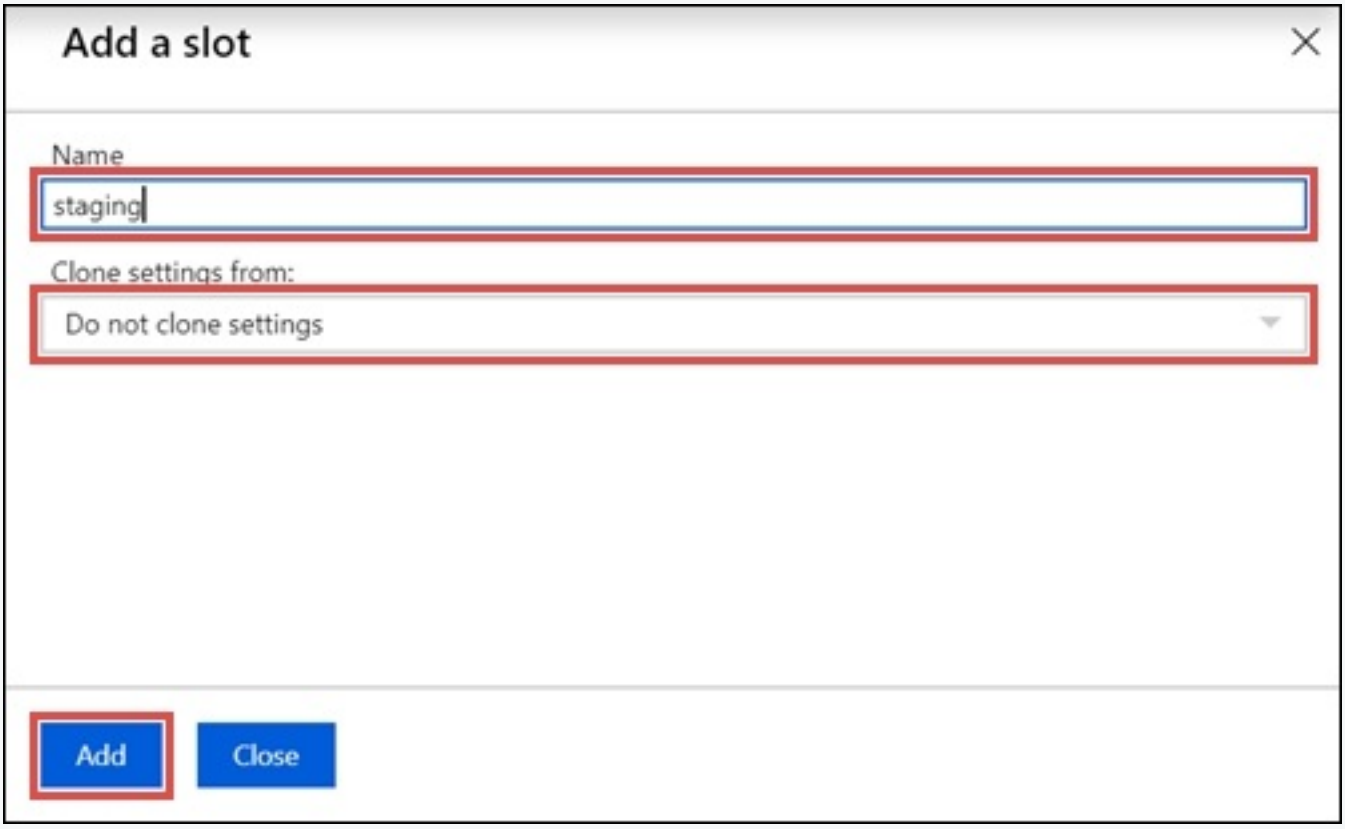
Set up staging environments in Azure App Service

1. Open Microsoft Azure Portal

2. Log into your Azure account, select your app's resource page, in the left pane, select Deployment slots > Add Slot.

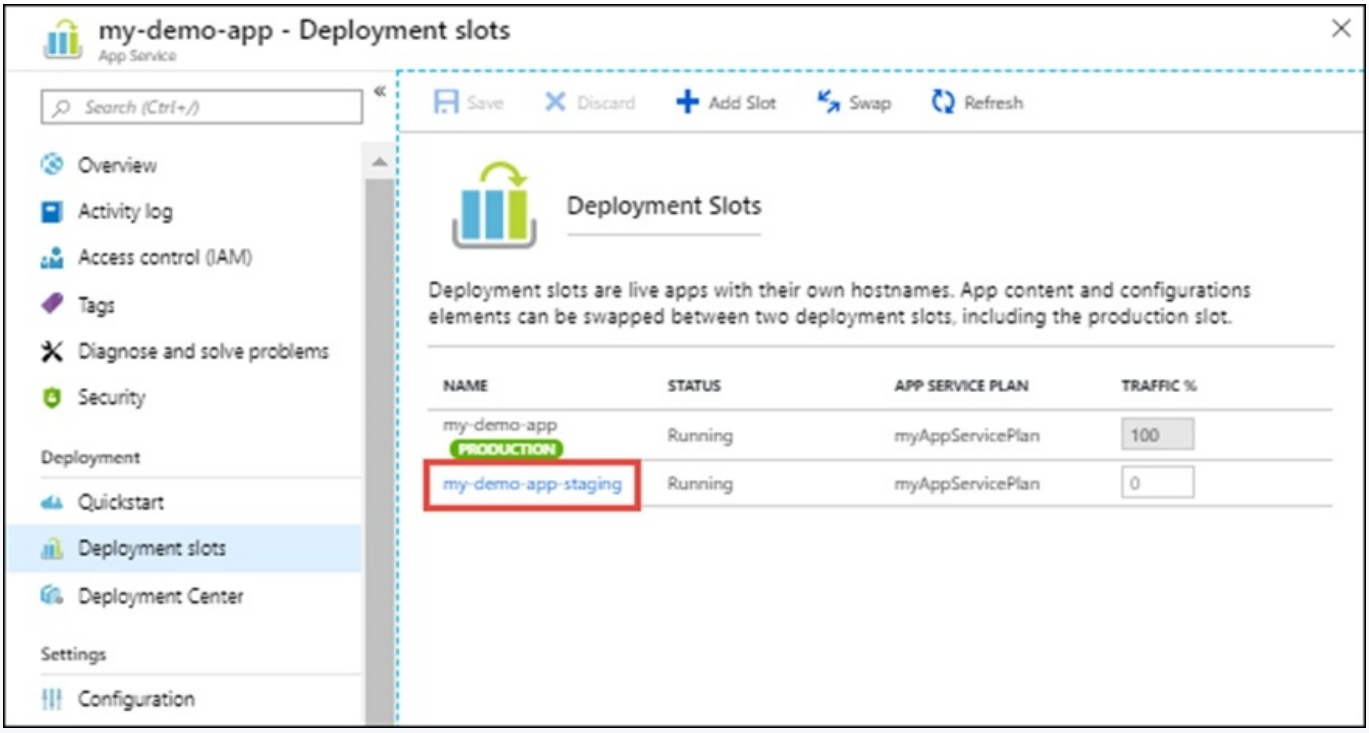


3. In the Add a slot dialog box, give the slot a name, and select whether to clone an app configuration from another deployment slot. Select Add to continue.



4. After the slot is added, select Close to close the dialog box. The new slot is now shown on the Deployment slots page. By default, Traffic % is set to 0 for the new slot, with all customer traffic routed to the production slot.

5. Select the new deployment slot to open that slot's resource page.



6. Change TRAFFIC % to 10

Reference:
https://docs.microsoft.com/en-us/azure/app-service/deploy-staging-slots

Question: 440 CertyIQ

You have an Azure DevOps project named Project1 and an Azure subscription named Sub1. Sub1 contains an Azure SQL database named DB1.
You need to create a release pipeline that uses the Azure SQL Database Deployment task to update DB1.
Which artifact should you deploy?

A. a BACPAC
B. a DACPAC
C. an LDF file
D. an MDF file

Answer: B

Explanation:

The correct answer is **B. a DACPAC**.

Here's why:

Azure SQL Database Deployment tasks within Azure DevOps release pipelines are designed to deploy database changes packaged in a specific format. That format is a Data-tier Application Package, commonly known as a **DACPAC**. A DACPAC represents a self-contained unit of SQL Server database deployment. It includes the schema (table definitions, stored procedures, views, etc.) and optionally the data.

A BACPAC, on the other hand, is a Backup Package. While it contains the database schema and data, it is designed for database migration or archiving purposes and is not the intended artifact type for schema updates using the Azure SQL Database Deployment task. BACPACs are typically used for exporting and importing entire databases between different Azure SQL Database instances or from on-premises SQL Server instances.

LDF (Log Data File) and MDF (Master Data File) are physical database files used by SQL Server. These files are low-level and aren't designed for deployment via a DevOps pipeline; they are integral to how SQL Server stores data on disk. Deploying them directly would be complex and risky. Instead, we use the DACPAC that builds on these concepts to provide a simple method of updating SQL Server.

Therefore, the Azure SQL Database Deployment task in Azure DevOps expects a DACPAC artifact to update the schema of the target Azure SQL Database (DB1). This allows for a controlled, versioned, and auditable deployment process. DACPACs provide a simplified, declarative approach to database deployment, enabling reliable and repeatable deployments as part of a CI/CD pipeline.

For more information, refer to these authoritative resources:

DACPACs and BACPACs: https://learn.microsoft.com/en-us/sql/relational-databases/data-tier-applications/data-tier-applications?view=sql-server-ver16
Azure SQL Database Deployment Task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/deploy/sql-azure-dacpac?view=azure-devops

Question: 441 CertyIQ

HOTSPOT -
You have a project in Azure DevOps.
You plan to create a build pipeline that will deploy resources by using Azure Resource Manager templates. The templates will reference secrets stored in Azure
Key Vault.
You need to ensure that you can dynamically generate the resource ID of the key vault during template deployment.
What should you include in the template? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

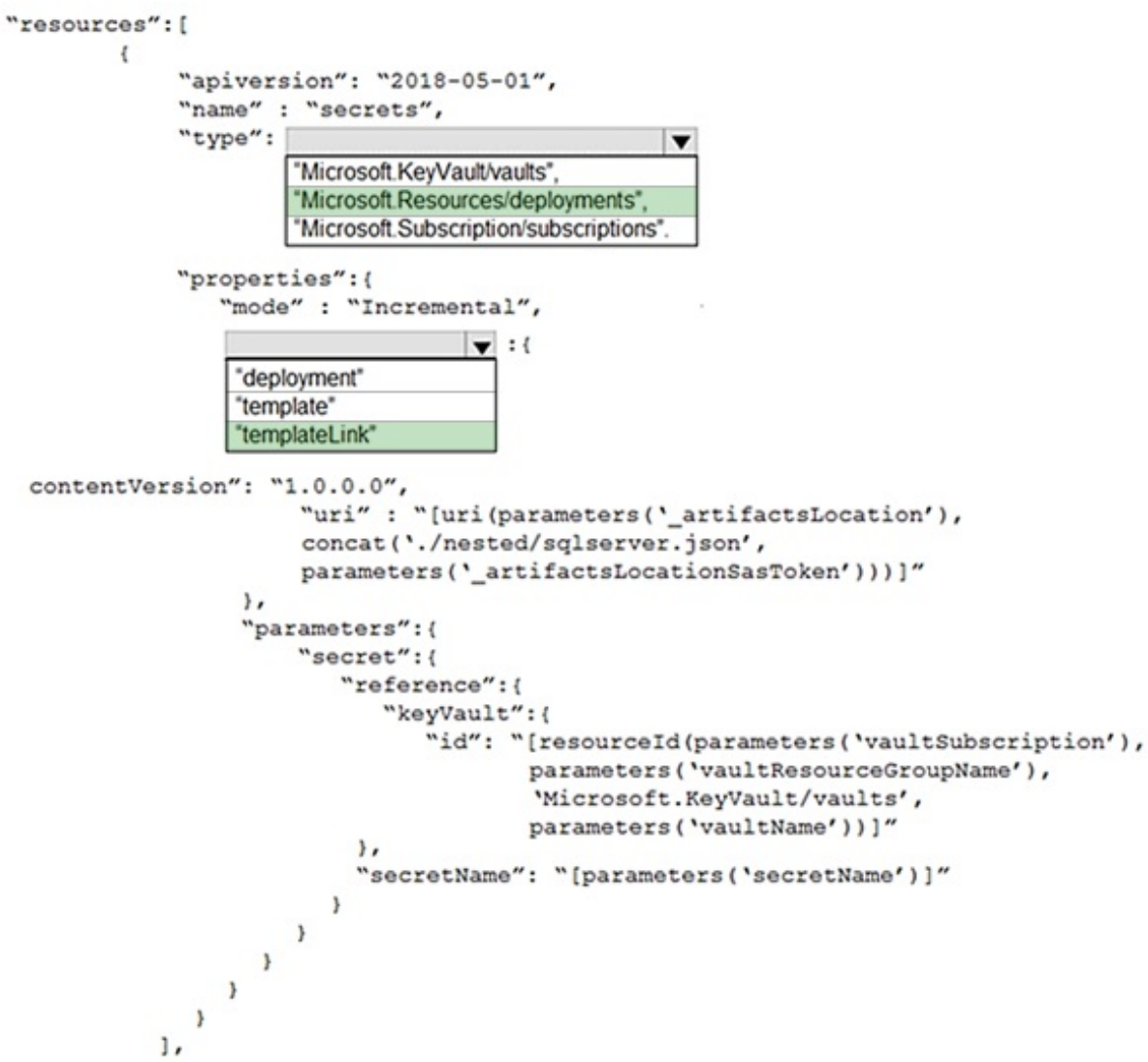
Answer Area

```
"resources":[
        {
            "apiversion": "2018-05-01",
            "name" : "secrets",
            "type":
                    "Microsoft.KeyVault/vaults",
                    "Microsoft.Resources/deployments",
                    "Microsoft.Subscription/subscriptions".

            "properties":{
               "mode" : "Incremental",
                  :{
                    "deployment"
                    "template"
                    "templateLink"
  contentVersion": "1.0.0.0",
                    "uri" : "[uri(parameters('_artifactsLocation'),
                    concat('./nested/sqlserver.json',
                    parameters('_artifactsLocationSasToken')))]"
                },
                "parameters":{
                    "secret":{
                       "reference":{
                          "keyVault":{
                             "id": "[resourceId(parameters('vaultSubscription'),
                                    parameters('vaultResourceGroupName'),
                                    'Microsoft.KeyVault/vaults',
                                    parameters('vaultName'))]"
                          },
                          "secretName": "[parameters('secretName')]"
                       }
                    }
                 }
              }
           }
        ],
```

Answer:

Answer Area



Explanation:

Box 1: "Microsoft.Resources/deployments"

Reference a secret with dynamic ID. You need to reference a key vault secret that varies based on the current deployment.

Example:

"resources": [

"apiVersion": "2018-05-01",

"name": "dynamicSecret",

"type": "Microsoft.Resources/deployments",

"properties":

"mode": "Incremental",

"templateLink":

Box 2: "templateLink"

In your parent template, you add the linked template and pass in a parameter that contains the dynamically generated resource ID.

Reference:

https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-keyvault-parameter

Question: 442 CertyIQ

Your company has a project in Azure DevOps for a new web application.

The company uses ServiceNow for change management.
You need to ensure that a change request is processed before any components can be deployed to the production environment.
What are two ways to integrate ServiceNow into the Azure DevOps release pipeline? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A. Define a deployment control that invokes the ServiceNow REST API.
B. Define a pre-deployment gate before the deployment to the Prod stage.
C. Define a deployment control that invokes the ServiceNow SOAP API.
D. Define a post-deployment gate after the deployment to the QA stage.

Answer: BD

Explanation:

The correct answer is BD. Here's why:

B. Define a pre-deployment gate before the deployment to the Prod stage.

This approach is crucial for ensuring change management compliance before any deployment occurs to the production environment. A pre-deployment gate allows the release pipeline to pause and verify that a corresponding change request exists and is in the correct state (e.g., approved) within ServiceNow. This gate can query the ServiceNow API (REST or SOAP) to check the status of the change request linked to the deployment. If the conditions are not met (no change request or not approved), the deployment is halted, preventing unauthorized or unapproved changes from reaching production. This adheres to the core principle of change management and minimizes potential disruptions. This is a proactive method, addressing the issue before it impacts production.

D. Define a post-deployment gate after the deployment to the QA stage.

Using a post-deployment gate after the QA stage provides a feedback loop for change management. Although the question focused on production deployment, ensuring change requests are valid throughout the pipeline is good practice. After QA, the gate can verify the deployment's success and update the ServiceNow change request accordingly. This could automatically transition the change request to an "implemented" or "closed" state in ServiceNow. However, it's the pre-deployment gate to production that directly addresses the requirement of change requests being processed before components are deployed to production. The QA gate ensures that a change request is only moved forward to production deployment after it has passed quality checks and is thus a safer deployment.

Why other options are incorrect:

A. Define a deployment control that invokes the ServiceNow REST API. Deployment controls are usually executed during or after a deployment, while the requirement calls for verification before the production deployment. While useful for other integration points, it doesn't directly address the core requirement.

C. Define a deployment control that invokes the ServiceNow SOAP API. Similar to option A, this focuses on actions during/after deployment. Using a deployment control rather than a gate doesn't provide a blocking mechanism to halt the deployment if change control requirements aren't met.

Supporting Concepts:

Change Management: A structured approach to transitioning individuals, teams, and organizations from a current state to a desired future state. Crucial for minimizing disruption during deployments.
Release Pipelines: Automated workflows that manage the deployment process from development to production.

Gates: Control points in a release pipeline that ensure specific conditions are met before proceeding.
APIs (REST/SOAP): Interfaces that allow different systems to communicate and exchange data. In this case, Azure DevOps and ServiceNow.

Authoritative Links:

Release gates and approvals: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/approvals/gates?view=azure-devops
Integrate Azure DevOps with ServiceNow: https://learn.microsoft.com/en-us/azure/devops/boards/integrations/service-now?view=azure-devops

Question: 443 CertyIQ

Your company develops an application named App1 that is deployed in production.
As part of an application update, a new service is being added to App1. The new service requires access to an application named App2 that is currently in development.
You need to ensure that you can deploy the update to App1 before App2 becomes available. You must be able to enable the service in App1 once App2 is deployed.
What should you do?

A. Implement a feature flag.
B. Create a fork in the build.
C. Create a branch in the build.
D. Implement a branch policy.

Answer: A

Explanation:

The correct answer is A: Implement a feature flag. Feature flags, also known as feature toggles, provide a powerful mechanism for controlling the release of new functionality in software. They allow you to deploy code changes to production without immediately enabling them for all users. This addresses the requirement of deploying the App1 update before App2 is available.

In this scenario, the new service in App1 that depends on App2 can be wrapped with a feature flag. This means the code for the new service is deployed, but it's inactive by default. When App2 becomes available, the feature flag can be enabled, instantly activating the new service in App1. This avoids the need for a separate deployment. Feature flags facilitate continuous integration and continuous delivery (CI/CD) by decoupling deployment from release. They enable testing in production with a limited audience, a strategy known as canary releasing or dark launching.

Forking or branching the build (options B and C) is not appropriate for this situation. Forking creates an entirely separate copy of the codebase, which is unnecessary and complicates maintenance. Branching is more common, but it still involves managing separate code streams and merging them later, which is also more complex than using a simple feature flag. Branch policies (option D) are related to code quality and workflow within a branch, not to enabling or disabling features in production.

Here's why feature flags are superior: they enable controlled rollouts, rapid rollback if issues arise after activation, and A/B testing of different features with targeted user groups. Furthermore, feature flags are easy to implement with various libraries and tools, offering a flexible and efficient way to manage feature releases. The other options do not address the need to deploy the update to App1 before App2 is available and then enable the service when App2 is ready.

Authoritative links for further research:

Martin Fowler - Feature Toggles: https://martinfowler.com/articles/feature-toggles.html
LaunchDarkly - What are Feature Flags?: https://launchdarkly.com/feature-management/what-are-feature-flags/

Question: 444 CertyIQ

You have a private distribution group that contains provisioned and unprovisioned devices.
You need to distribute a new iOS application to the distribution group by using Microsoft Visual Studio App Center.
What should you do?

A. Request the Apple ID associated with the user of each device.
B. Register the devices on the Apple Developer portal.
C. Create an active subscription in App Center Test.
D. Add the device owner to the organization in App Center.

Answer: B

Explanation:

The correct answer is **B. Register the devices on the Apple Developer portal.**

Here's why: Distributing iOS applications to a private group of devices, especially those "unprovisioned," requires that the devices be registered with Apple. iOS operates under a strict security model where applications can only be installed on devices that are explicitly authorized by Apple. This authorization is managed through provisioning profiles, which are linked to specific device IDs (UDIDs).

Registering devices on the Apple Developer portal provides the UDIDs of these devices to Apple. You then create a provisioning profile that includes these registered devices. When you build the iOS application with this provisioning profile, it can be installed and run only on those registered devices. This process ensures that unauthorized devices cannot install the application, maintaining the security and integrity of the distribution.

Visual Studio App Center simplifies the distribution process, but it relies on the fundamental Apple requirements for device registration. Without registering the devices and including them in a provisioning profile, App Center will not be able to distribute the app to the target devices. Option A is incorrect as Apple IDs are associated with users, not directly with device registration. Option C, App Center Test, relates to testing features, not the core distribution requirements. Option D, adding device owners to the organization, is about access control within App Center, not about device-level authorization for iOS. Unprovisioned devices are devices not linked to the Apple developer account, therefore they can't install test apps (builds for internal use). You first have to register them.

For further research, refer to the official Apple Developer documentation:

Apple Developer Program: https://developer.apple.com/programs/
Provisioning Profiles: https://help.apple.com/xcode/mac/current/#/dev53f44801
Device Registration: https://developer.apple.com/account/resources/devices/list (login required)

Question: 445 CertyIQ

You are developing an iOS application by using Azure DevOps.
You need to test the application manually on 10 devices without releasing the application to the public.
Which two actions should you perform? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. Create a Microsoft Intune device compliance policy.
B. Deploy a certificate from an internal certification authority (CA) to each device.
C. Register the application in the iTunes store.
D. Onboard the devices into Microsoft Intune.
E. Distribute a new release of the application.
F. Register the IDs of the devices in the Apple Developer portal.

Answer: EF

Explanation:

The correct answer is **EF**. Here's a detailed justification:

To distribute an iOS application for manual testing on a limited number of devices without a public release, you need to leverage Apple's ad-hoc distribution mechanism. Ad-hoc distribution allows you to distribute your app to a limited number of registered devices.

F. Register the IDs of the devices in the Apple Developer portal: Apple requires that you register the Unique Device Identifiers (UDIDs) of each test device in your Apple Developer account. This step is crucial because it tells Apple that these specific devices are authorized to run the application you're distributing outside of the App Store. You add these device UDIDs to the provisioning profile used to sign the application.

E. Distribute a new release of the application: Once the device UDIDs are registered and included in a provisioning profile, you must build and sign the application using this ad-hoc provisioning profile. The generated .ipa file can then be distributed to the registered devices. Users can then install this .ipa file onto their devices.

Why the other options are incorrect:

A. Create a Microsoft Intune device compliance policy: Intune device compliance policies are primarily used to enforce security and configuration standards on devices managed by Intune. While Intune can be used to deploy applications, it is not a pre-requisite for simple ad-hoc distribution to a small number of devices, especially for initial manual testing.

B. Deploy a certificate from an internal certification authority (CA) to each device: While this approach can be used for internal enterprise app distribution, it's overkill for testing on a small number of devices. Ad-hoc distribution is simpler and more appropriate in this case.

C. Register the application in the iTunes store: This step is only required for public app releases through the App Store. Ad-hoc distribution specifically avoids the App Store.

In summary: Ad-hoc distribution through the Apple Developer Program allows you to manually test the application on specific devices by registering their device IDs and then distributing an app signed with a special ad-hoc provisioning profile. This is the simplest way to test on a limited set of devices without public release.

Supporting Links:

Ad Hoc Provisioning Profile: https://developer.apple.com/documentation/xcode/distributing-your-app-for-beta-testing-and-releases (Note that while this document refers to TestFlight and App Store Connect, it provides valuable context on provisioning profiles)
Device Registration: (Refer to Apple Developer Program resources within your developer account for device registration procedures).

Question: 446 CertyIQ

You have a private distribution group that contains provisioned and unprovisioned devices.
You need to distribute a new iOS application to the distribution group by using Microsoft Visual Studio App Center.
What should you do?

A. Select Register devices and sign my app.
B. Create an active subscription in App Center Test.
C. Create an unsigned build.
D. Add the device owner to the collaborators group.

Answer: A

Explanation:

The correct answer is A, "Select Register devices and sign my app." Here's why:

Visual Studio App Center's distribution mechanism for iOS applications relies on device registration to ensure that the app can be installed on those devices. iOS apps are typically distributed through the Apple App Store, but for private distribution, App Center offers a way to distribute directly to devices. This requires a provisioning profile that includes the UDIDs (Unique Device Identifiers) of the target devices.

Option A addresses this directly. When you select "Register devices and sign my app," App Center will guide you through the process of registering the UDIDs of the unprovisioned devices in your distribution group. This involves creating a provisioning profile that includes these device IDs and then signing the application with this profile. The signed app, specific to these registered devices, can then be installed. This ensures the app can be installed on both provisioned (already registered) and unprovisioned devices within the distribution group.

Option B, "Create an active subscription in App Center Test," is irrelevant. App Center Test is for automated UI testing on real devices, not for distributing apps to users or registering devices for private distribution.

Option C, "Create an unsigned build," is incorrect because iOS apps require signing to be installed on devices outside the App Store. An unsigned build cannot be installed on any device.

Option D, "Add the device owner to the collaborators group," is insufficient. While adding a user as a collaborator allows them to access and manage the app in App Center, it doesn't address the core requirement of registering their device's UDID and signing the app with a profile that includes those devices. Simply adding the user doesn't enable installation.

Therefore, the only option that correctly addresses the requirement of distributing the iOS app to a private distribution group containing both provisioned and unprovisioned devices is to register the devices and sign the app.

Relevant Links:

Distribute iOS apps with App Center: https://learn.microsoft.com/en-us/appcenter/distribution/ios
Code Signing for iOS: https://developer.apple.com/support/code-signing/

Question: 447 CertyIQ

SIMULATION -
You plan to deploy a website that will be hosted in two Azure regions.
You need to create an Azure Traffic Manager profile named az400123456789n1-tm in a resource group named RG1lod123456789. The solution must ensure that users will always connect to a copy of the website that is in the same country.

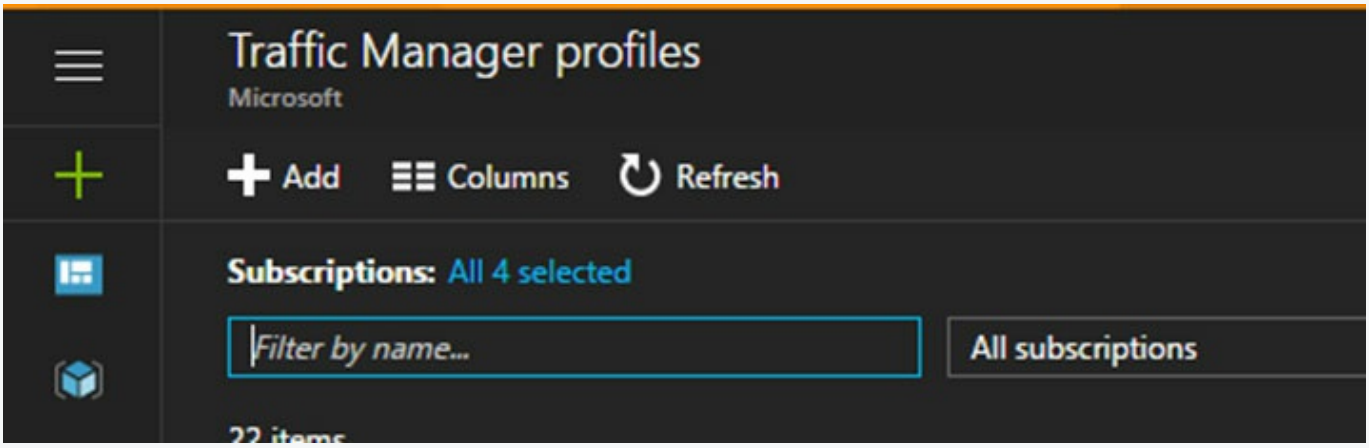
To complete this task, sign in to the Microsoft Azure portal.

Answer:

See explanation below.

Explanation:

1. Go to the Azure portal, navigate to Traffic Manager profiles and click on the Add button to create a routing profile.

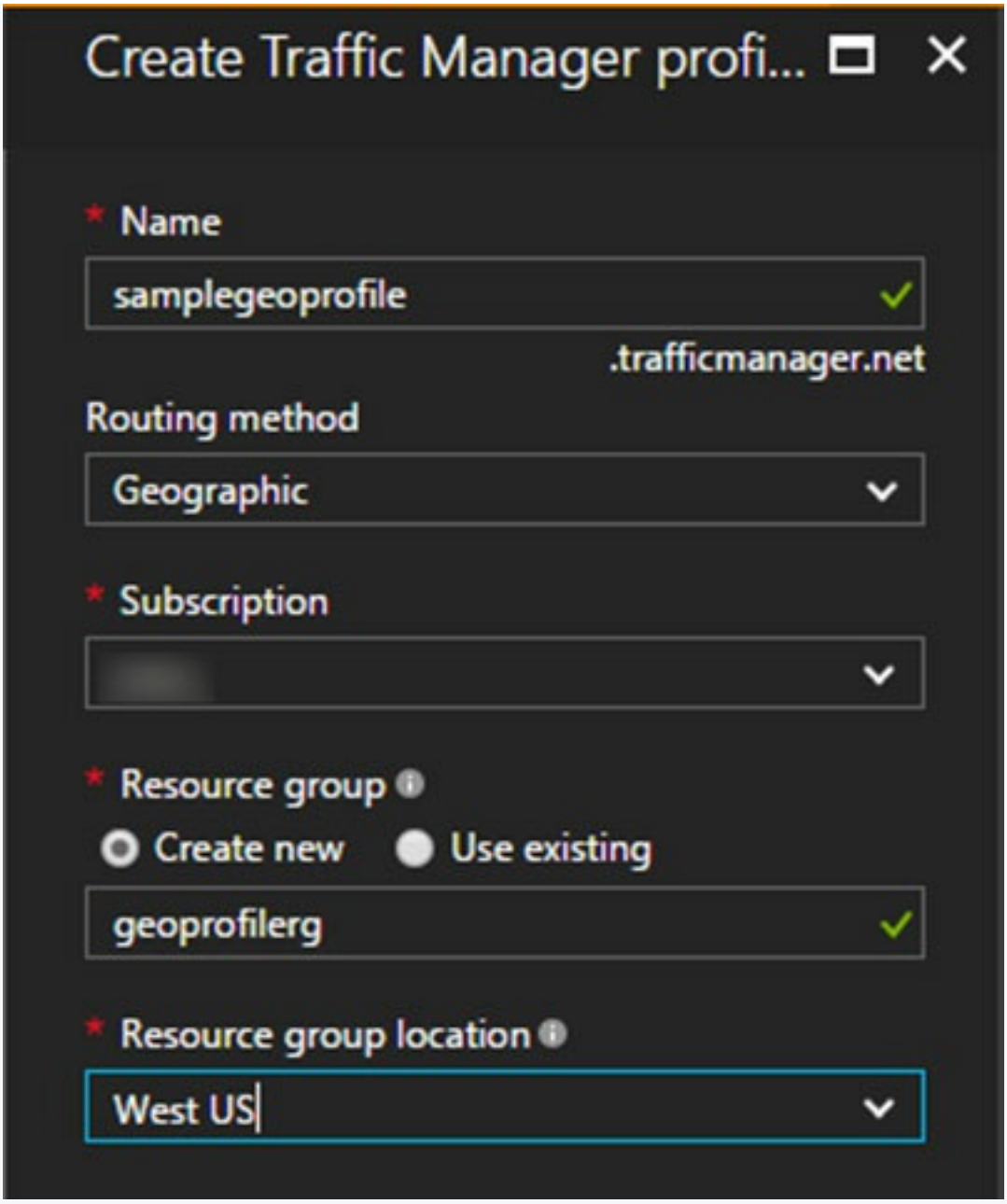


2, In the Create Traffic Manager profile, enter, or select these settings:

Name: az400123456789n1-tm -

Routing method: Geographic -

Resource group: RG1lod123456789 -



Note: Traffic Manager profiles can be configured to use the Geographic routing method so that users are directed to specific endpoints (Azure, External or Nested) based on which geographic location their DNS query originates from. This empowers Traffic Manager customers to enable scenarios where knowing a user's geographic region and routing them based on that is important.

Reference:
https://azure.microsoft.com/en-us/blog/announcing-the-general-availability-of-geographic-routing-capability-in-azure-traffic-manager/

Question: 448 CertyIQ

Your company has an on-premises Bitbucket Server that is used for Git-based source control. The server is protected by a firewall that blocks inbound Internet traffic.
You plan to use Azure DevOps to manage the build and release processes.
Which two components are required to integrate Azure DevOps and Bitbucket? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. a deployment group
B. a Microsoft-hosted agent
C. service hooks

D. a self-hosted agent

E. an External Git service connection

Answer: DE

Explanation:

Let's break down why options D (a self-hosted agent) and E (an External Git service connection) are the correct choices for integrating Azure DevOps with an on-premises Bitbucket Server protected by a firewall.

External Git Service Connection: This is crucial because it establishes the necessary authentication and communication channel between Azure DevOps and the Bitbucket server. Azure DevOps needs a way to access the source code repositories on Bitbucket. A service connection provides the credentials and configuration details for this access, such as the Bitbucket server URL, username/password or a personal access token. This enables Azure DevOps to trigger builds, pull code, and manage the lifecycle of the code residing on Bitbucket. https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml

Self-Hosted Agent: Since the Bitbucket server is behind a firewall that blocks inbound internet traffic, a Microsoft-hosted agent (option B) cannot directly access the Bitbucket server. Microsoft-hosted agents operate in Microsoft's cloud infrastructure and wouldn't be able to bypass the firewall. A self-hosted agent, on the other hand, is installed and managed on a machine within your own network, behind the firewall. This agent can then access the Bitbucket server directly, download the source code, and execute build and release tasks. The self-hosted agent connects to Azure DevOps using an outbound connection (typically HTTPS), which is usually allowed through firewalls. https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser

Now, let's look at why the other options are incorrect:

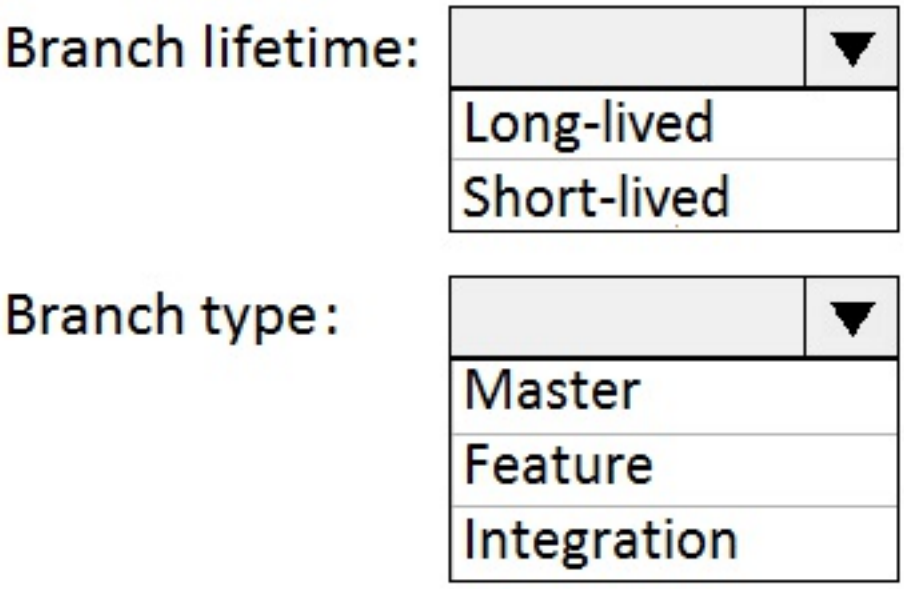
A. Deployment Group: Deployment groups are primarily used for deployments to virtual machines or other on-premises servers, not for source code access and build processes directly related to Bitbucket. While they could potentially be part of a larger deployment pipeline involving code from Bitbucket, they aren't essential for the initial integration.
C. Service Hooks: Service hooks are used for triggering actions in other services when events occur in Azure DevOps. While they might be part of a more complex integration scenario, they are not directly involved in connecting to the Bitbucket server and fetching source code. Service hooks react after something happens in Azure DevOps.

In summary, a self-hosted agent provides the compute resource within your network to interact with the on-premises Bitbucket server, and an external Git service connection establishes the necessary authentication and connection for Azure DevOps to access the Bitbucket repositories.

Question: 449 CertyIQ

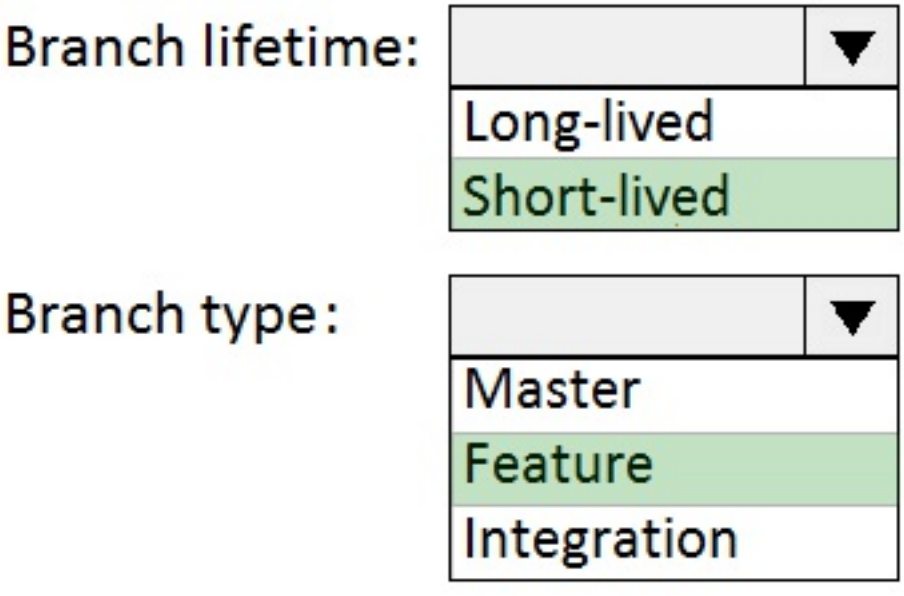
HOTSPOT -
Your company uses Git as a source code control system for a complex app named App1.
You plan to add a new functionality to App1.
You need to design a branching model for the new functionality.
Which branch lifetime and branch time should you use in the branching model? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:
Branch lifetime: Short-lived -

Branch type: Feature -
Feature branches are used when developing a new feature or enhancement which has the potential of a development lifespan longer than a single deployment.
When starting development, the deployment in which this feature will be released may not be known. No matter when the feature branch will be finished, it will always be merged back into the master branch.

Reference:

https://gist.github.com/digitaljhelms/4287848

Question: 450 CertyIQ

You have a project in Azure DevOps.
You plan to deploy a self-hosted agent by using an unattended configuration script.
Which two values should you define in the configuration script? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A. authorization credentials
B. the project name
C. the deployment group name
D. the organization URL
E. the agent pool name

Answer: AD

Explanation:

The correct answer is **A. authorization credentials** and **D. the organization URL**.

When configuring an unattended self-hosted agent in Azure DevOps, the configuration script must provide sufficient information for the agent to connect to your Azure DevOps organization and authenticate. This involves specifying the **organization URL (D)**, which tells the agent where your Azure DevOps instance is located. For example, this might be https://dev.azure.com/yourorganization. Without this, the agent has no idea where to register itself.

Furthermore, the agent needs **authorization credentials (A)** to authenticate with Azure DevOps. These credentials verify the agent's identity and grant it the necessary permissions to access resources and execute pipelines. This authentication is typically done using a Personal Access Token (PAT), but other methods such as Managed Identities or Service Principals may also be used, depending on your organization's security policies. Without valid credentials, the agent will be unable to register with Azure DevOps or receive pipeline jobs.

The **project name (B)** isn't directly used during the initial agent configuration. Agents are registered at the organization or agent pool level, not at the project level. While the agent will eventually execute tasks related to projects, the project itself isn't a necessary input for the configuration script.

The **deployment group name (C)** is relevant for deployment groups, which are a specific type of target environment. If you are configuring the agent to be part of a deployment group, then the deployment group name is required. But the prompt suggests the agent will be used with a regular agent pool, not specifically a deployment group.

The **agent pool name (E)** is essential to assign the agent to the correct pool after registration. In the new self-hosted agent configuration experience, Azure DevOps will attempt to find an existing agent pool to which to assign the agent. If none are specified, it may try to assign the new agent to the Default agent pool. However, the configuration of the agent pool requires authorization and URL to be configured first.

Therefore, the two key elements required for an unattended self-hosted agent configuration script are the organization URL to specify the Azure DevOps instance and authorization credentials to authenticate the agent.

For further research, refer to the official Microsoft documentation on:

Self-hosted agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser
Personal Access Tokens (PATs): https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops&tabs=preview

Question: 451 CertyIQ

HOTSPOT -
You have an Azure virtual machine named VM1 that runs Linux.
You plan to deploy the Desired State Configuration (DSC) extension to VM1.
You need to grant the Log Analytics agent the appropriate directory permissions.
How should you complete the command? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

setfacl -m u:omsagent: [ ] [ ]

r	/lib
x	/etc
rx	/tmp
rwx	/usr

Answer:

Answer Area

setfacl -m u:omsagent: [rwx] [/tmp]

r	/lib
x	/etc
rx	/tmp
rwx	/usr

Explanation:
Box 1: rwx -
The Log Analytics agent for Linux runs as the omsagent user. To grant >write permission to the omsagent user, run the command setfacl -m u:omsagent:rwx /tmp.

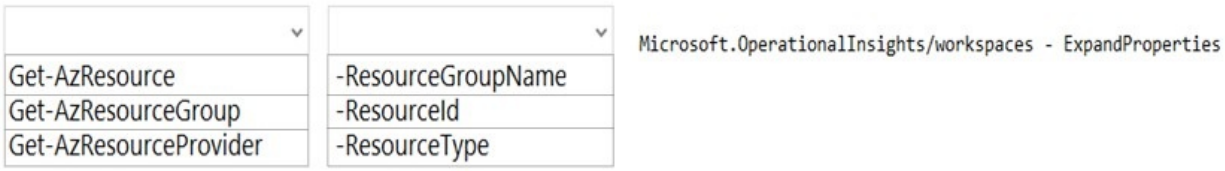
Box 2: /tmp -
Deploying DSC to a Linux node uses the /tmp folder.

Reference:
https://docs.microsoft.com/en-us/azure/automation/automation-dsc-onboarding

Question: 452 CertyIQ

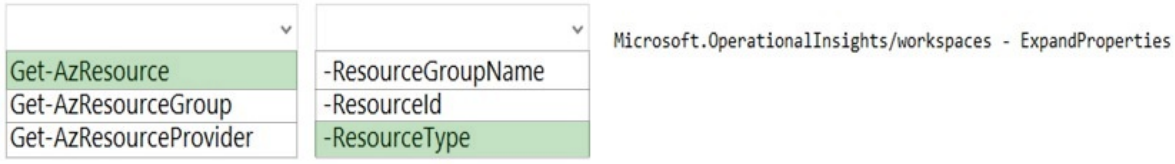
HOTSPOT -
You are using PowerShell to administer Azure Log Analytics workspaces.
You need to list the available workspaces and their properties.
How should you complete the command? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:

Box 1: Get-AzResource -

Use the following command to examine the access control mode for all workspaces in the subscription:

PowerShell -

Get-AzResource -ResourceType Microsoft.OperationalInsights/workspaces -ExpandProperties | foreach $_.Name + ": " + $_.Properties.features.enableLogAccessUsingOnlyResourcePermissions

Box 2: -ResourceType -

Reference:

https://docs.microsoft.com/en-us/azure/azure-monitor/logs/manage-access

Question: 453 CertyIQ

You have Azure Pipelines and GitHub integrated as a source code repository.
The build pipeline has continuous integration enabled.
You plan to trigger an automated build whenever code changes are committed to the repository.
You need to ensure that the system will wait until a build completes before queuing another build.
What should you implement?

A. path filters
B. batch changes
C. scheduled builds
D. branch filters

Answer: B

Explanation:

The correct answer is **B. Batch changes**.

Here's why:

The requirement is to prevent multiple builds from queuing up while a build is already in progress. This means that instead of triggering a build for every single commit, the system should wait for the current build to finish and then potentially consolidate multiple commits into a single build. This is exactly what **Batch Changes**

accomplishes.

When "Batch changes" is enabled in Azure Pipelines trigger settings, the pipeline waits until the current run is completed before queuing another run. If multiple commits are pushed to the repository while the pipeline is running, they are batched together and a single build is triggered for those combined changes after the current build finishes.

Path filters (A) are used to trigger a build only when changes are made to specific files or directories. This doesn't address the queuing issue.
Scheduled builds (C) trigger builds at a specific time, irrespective of code changes. This is not related to continuous integration based on commits.
Branch filters (D) restrict builds to specific branches, not the queuing behavior.

Therefore, Batch Changes is the only option that directly controls the queuing behavior by combining multiple commits into a single build after the previous build completes, addressing the stated requirement of preventing multiple builds from queuing.

Authoritative Links:

Configure triggers - Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/build/triggers?view=azure-devops&tabs=yaml (search for "Batch changes" in the document)

Question: 454 CertyIQ

You are using GitHub as a source code repository.
You create a client-side Git hook on the commit-msg event. The hook requires that each commit message contain a custom work item tag.
You need to make a commit that does not have a work item tag.
Which git commit parameter should you use?

A. --squash
B. --no-verify
C. --message ''
D. --no-post-rewrite

Answer: B

Explanation:

The correct answer is **B. --no-verify**. Here's why:

Client-side Git hooks are scripts that run locally before or after certain Git events, such as committing, pushing, or receiving. These hooks can be used to enforce coding standards, validate commit messages, or perform other custom actions. In this scenario, the commit-msg hook checks if a commit message contains a specific work item tag.

The --no-verify option is a Git command-line parameter that bypasses the hook scripts. When used with git commit, it instructs Git to skip all the pre-commit and commit-msg hooks. This is precisely what's needed to make a commit without triggering the work item tag validation enforced by the commit-msg hook. This enables users to commit code without adhering to the specific hook requirement, which could be useful for temporary commits, experimental changes, or commits that intentionally deviate from standard procedures.

Option A, --squash, is used to combine multiple commits into a single commit. It doesn't bypass hooks. Option

C, --message '', provides an empty commit message, but it would still trigger the commit-msg hook, which would likely reject the commit due to the missing work item tag. Option D, --no-post-rewrite, is related to rewriting commits and is not relevant to bypassing the commit-msg hook during a regular commit.

Therefore, --no-verify is the only option that directly addresses the requirement to bypass the client-side Git hook and make a commit without including the work item tag in the commit message. It's a standard Git functionality designed to temporarily disable hooks when needed.

Supporting Links:

Git Hooks Documentation: https://git-scm.com/docs/githooks (Official Git documentation describing Git hooks and their functionality)
git commit man page: https://git-scm.com/docs/git-commit (Official Git documentation describing the git commit command and its options including --no-verify)

Question: 455 CertyIQ

You have a private distribution group that contains provisioned and unprovisioned devices.
You need to distribute a new iOS application to the distribution group by using Microsoft Visual Studio App Center.
What should you do?

A. Select Register devices and sign my app.
B. Create an active subscription in App Center Test.
C. Add the device owner to the organization in App Center.
D. Create an unsigned build.

Answer: A

Explanation:

The correct answer is A: "Select Register devices and sign my app." Here's a detailed justification:

Visual Studio App Center's distribution feature for iOS apps to private groups necessitates proper provisioning and signing of the application. This is because iOS devices require apps to be signed with a valid certificate and provisioning profile that includes the device's unique device identifier (UDID).

Option A, "Select Register devices and sign my app," directly addresses this requirement. When this option is selected, App Center guides you through the process of registering the target iOS devices (both provisioned and unprovisioned) and then resigning the application with a provisioning profile that includes those devices. This ensures that the application can be installed and run on the registered devices. App Center handles the complexity of creating or updating the provisioning profile and signing the app.

Option B, "Create an active subscription in App Center Test," is incorrect because an App Center Test subscription is for running automated UI tests on real devices and doesn't directly relate to distributing the app to a private group of devices, especially those not already provisioned. Distribution and testing are separate functions within App Center.

Option C, "Add the device owner to the organization in App Center," while important for access control within App Center, doesn't address the core problem of provisioning and signing the app for the specific target devices. Adding the device owner grants them access to the app distribution group but does not automatically register their device or resign the app.

Option D, "Create an unsigned build," is the opposite of what's needed. iOS devices require signed applications. Distributing an unsigned build will prevent installation on iOS devices and render the distribution

process useless.

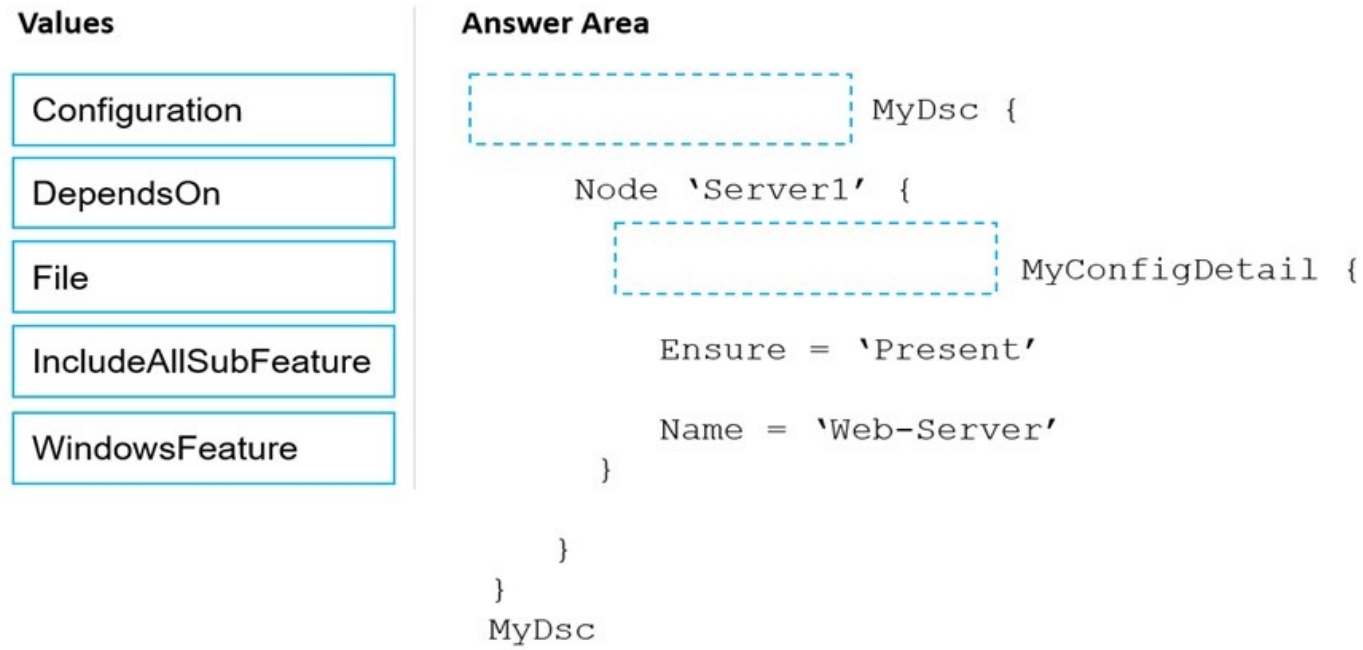
In summary, distributing an iOS app to a private group with potentially unprovisioned devices using App Center requires registering those devices and resigning the application with a valid provisioning profile that includes them. "Register devices and sign my app" fulfills this essential requirement.

Authoritative links for further research:

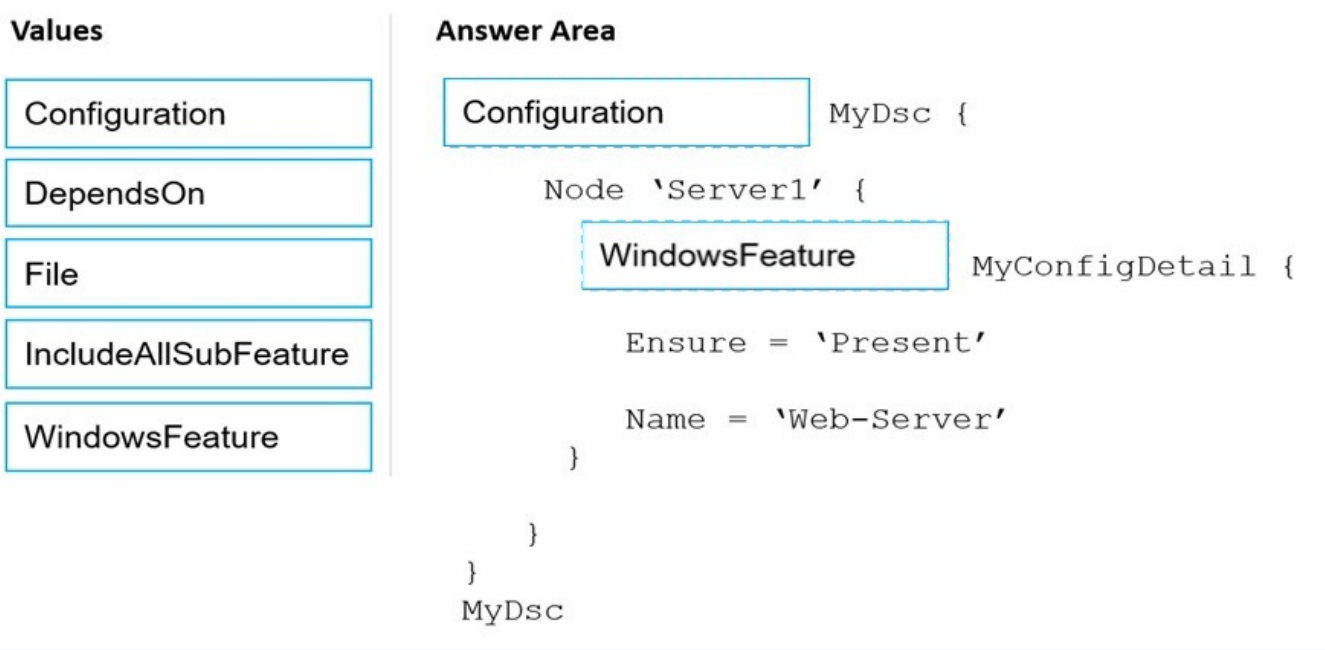
App Center Distribution: https://learn.microsoft.com/en-us/appcenter/distribution/
App Center iOS Distribution: https://learn.microsoft.com/en-us/appcenter/distribution/ios

Question: 456 CertyIQ

DRAG DROP -
You need to deploy Internet Information Services (IIS) to an Azure virtual machine that runs Windows Server 2019. How should you complete the Desired State Configuration (DSC) configuration script? To answer, drag the appropriate values to the correct locations. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:
Box 1: Configuration -
Configuration Example:

Configuration Sample -

Node localhost, server02 -

Box 2: WindowsFeature -
Example:

WindowsFeature RoleExample -

Ensure = "Present"
# Alternatively, to ensure the role is uninstalled, set Ensure to "Absent"
Name = "Web-Server" # Use the Name property from Get-WindowsFeature

The WindowsFeature resource in Windows PowerShell Desired State Configuration (DSC) provides a mechanism to ensure that roles and features are added or removed on a target node.

Reference:
https://docs.microsoft.com/en-us/powershell/dsc/reference/resources/windows/windowsfeatureresource
https://docs.microsoft.com/en-us/powershell/dsc/managing-nodes/apply-get-test

Question: 457 CertyIQ

HOTSPOT -
You plan to use Desired State Configuration (DSC) to maintain the configuration of a server that runs Windows Server 2019.
The server must have the following features installed:
✑ A web server
✑ An email server
How should you complete the DSC configuration file? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

```
configuration RequiredFeatures
{
    Import-DscResource -ModuleName PSDesiredStateConfiguration
    Node localhost
    {
        WindowsFeatureSet RequiredWindowsFeatures
        {
            Name
```

= @("Mail-Server", "IIS")
= @("SMTP-Server", "IIS")
= @("Mail-Server", "Web-Server")
= @("SMTP-Server", "Web-Server")

```
            
```

= 'Present'

Install
Ensure
Enforce
Required

```
            IncludeAllSubFeature    = $true
        }
    }
}
```

Answer:

```
configuration RequiredFeatures
{
    Import-DscResource -ModuleName PSDesiredStateConfiguration
    Node localhost
    {
        WindowsFeatureSet RequiredWindowsFeatures
        {
            Name
                = @("Mail-Server", "IIS")
                = @("SMTP-Server", "IIS")
                = @("Mail-Server", "Web-Server")
                = @("SMTP-Server", "Web-Server")

                = 'Present'
                Install
                Ensure
                Enforce
                Required
            IncludeAllSubFeature    = $true
        }
    }
}
```

Explanation:
Box 1: = @("SMTP-Server", "Web-Server")
The following configuration ensures that the Web-Server (IIS) and SMTP (email) Server features, and all subfeatures of each, are installed. configuration FeatureSetTest

Import-DscResource -ModuleName PSDesiredStateConfiguration

Node localhost -

WindowsFeatureSet WindowsFeatureSetExample

Name = @("SMTP-Server", "Web-Server")

Ensure = 'Present'

IncludeAllSubFeature = $true -

Box 2: Ensure -
Ensure indicates whether the roles or features are added. To ensure that the roles or features are added, set this property to Present. To ensure that the roles or features are removed, set the property to Absent. The default value is Present.

Reference:
https://docs.microsoft.com/en-us/powershell/dsc/reference/resources/windows/windowsFeatureSetResource

Question: 458 CertyIQ

You have a private distribution group that contains provisioned and unprovisioned devices.
You need to distribute a new iOS application to the distribution group by using Microsoft Visual Studio App Center.
What should you do?

A. Register the devices on the Apple Developer portal.
B. Add the device owner to the organization in App Center.
C. Create an unsigned build.
D. Add the device owner to the collaborators group.

Answer: A

Explanation:

The correct answer is A: Register the devices on the Apple Developer portal. Here's a detailed justification:

Distributing iOS applications to a closed group of users for testing or internal distribution, often called "Ad Hoc" distribution, requires specific device registration. Apple's security model mandates that each device intended to run an Ad Hoc build must be registered within the Apple Developer portal with its Unique Device Identifier (UDID). This registration is necessary because iOS only allows apps signed with an Ad Hoc provisioning profile to run on registered devices.

Visual Studio App Center leverages Apple's provisioning process. To distribute to private distribution groups (including unprovisioned devices), you need to ensure the devices are registered. App Center creates an Ad Hoc provisioning profile that incorporates these registered devices' UDIDs. Without this, the application will not install on unregistered devices, leading to the distribution failing.

Option B, adding device owners to the App Center organization, grants access to the App Center project, allowing them to view the app but doesn't provision their devices. Option C, creating an unsigned build, would result in an unusable application that cannot be installed on any device. Option D, adding device owners to the collaborators group, also grants access to the project but doesn't address the device registration requirement for Ad Hoc distribution.

Therefore, registering the devices on the Apple Developer portal is essential for distributing iOS applications to a private distribution group in Visual Studio App Center, ensuring the app can be installed and run on the intended devices.

Further reading on Ad Hoc distribution and device registration:

Apple Developer Documentation - Distributing Your App to Registered Devices: https://developer.apple.com/documentation/xcode/distributing-your-app-to-registered-devices
Microsoft App Center - Distribute iOS Apps: https://learn.microsoft.com/en-us/appcenter/distribution/ios

Question: 459 CertyIQ

Your company uses Azure Artifacts for package management.
You need to configure an upstream source in Azure Artifacts for Python packages.
Which repository type should you use as an upstream source?

A. npmjs.org
B. PyPI
C. Maven Central
D. third-party trusted Python

Answer: B

Explanation:

The correct answer is PyPI (Python Package Index). Azure Artifacts allows you to configure upstream sources to access packages from public registries. Since the requirement is for Python packages, the upstream source must be a Python package repository.

PyPI (Python Package Index) is the official public repository for Python packages. It's the central location where developers upload Python packages that can be installed and used by others. Azure Artifacts supports PyPI as an upstream source.

npmjs.org is the public registry for Node.js packages. Maven Central is the central repository for Java packages. Because we need Python packages, npmjs.org and Maven Central are inappropriate choices. A "third-party trusted Python" option, while it could theoretically exist, isn't a standard or well-defined registry supported directly as an upstream source in Azure Artifacts. Therefore, using PyPI is the correct and standardized approach. By configuring PyPI as an upstream source, developers within your organization can access and use Python packages hosted on PyPI directly from your Azure Artifacts feed. This facilitates efficient package management and ensures consistent access to necessary dependencies.

For more information on Azure Artifacts and upstream sources, refer to the official Microsoft documentation:

Azure Artifacts Upstream Sources
Use packages from PyPI

Question: 460 CertyIQ

You have a GitHub repository that contains workflows. The workflows contain steps that execute predefined actions. Each action has one or more versions.
You need to request the specific version of an action to execute.
Which three attributes can you use to identify the version? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A. the SHA-based hashes
B. the tag
C. the runner

D. the branch

E. the serial

Answer: ABD

Explanation:

The correct answer is **ABD** because GitHub Actions versions can be specified using the SHA-based hash, a tag, or a branch.

SHA-based hashes: Using the specific SHA-based hash ensures the exact code version is executed. This provides immutability and avoids unexpected behavior if the action is updated. The hash is a unique identifier for a specific commit, providing the highest degree of accuracy and control. https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners#using-actions-from-githubcom

Tags: Tags are human-readable pointers to specific commits, often used to mark releases. This offers a balance between stability and readability. For example, using v1.0.0 would always refer to the code tagged with that version. https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners#using-actions-from-githubcom

Branches: Specifying a branch allows using the latest code from that branch. This is useful for testing the most recent changes, but it should be done with caution in production since the branch content could change unexpectedly, potentially leading to broken builds or unexpected behavior. For example, utilizing the 'main' or 'develop' branch. https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners#using-actions-from-githubcom

Options **C (the runner)** and **E (the serial)** are incorrect. The runner is the environment where the action executes, and the serial number is not used for specifying the version of an action. The runner defines the environment (OS, tools) for the workflow, not the version of the action itself. Specifying action version allows to maintain control over the execution of the workflow.

Question: 461 CertyIQ

You have an Azure subscription that contains multiple Azure pipelines.
You need to deploy a monitoring solution for the pipelines. The solution must meet the following requirements:
✑ Parse logs from multiple sources.
✑ Identify the root cause of issues.
What advanced feature of a monitoring tool should you include in the solution?

A. analytics

B. synthetic monitoring

C. directed monitoring

D. Alert Management

Answer: A

Explanation:

The correct answer is **A. analytics**. Here's a detailed justification:

The primary goal is to deploy a monitoring solution that can parse logs from multiple sources and pinpoint the root cause of issues within Azure pipelines. To achieve this, the solution requires a capability that can analyze

large volumes of log data and identify patterns, correlations, and anomalies.

Analytics, in the context of monitoring tools, is designed precisely for this purpose. It leverages data analysis techniques, often including machine learning and statistical modeling, to provide insights into system behavior. By applying analytics to the logs generated by Azure pipelines, the solution can effectively:

Aggregate and correlate data from different sources: Azure pipelines generate logs from various components, such as tasks, agents, and build servers. Analytics can consolidate this disparate data into a unified view.
Identify patterns and anomalies: By analyzing historical data, analytics can establish a baseline of normal behavior. Any deviations from this baseline, such as performance degradation or error spikes, can be flagged as potential issues.
Perform root cause analysis: Analytics can help trace the sequence of events leading to an issue, allowing you to quickly identify the underlying cause.
Provide visualizations and reports: Analytics tools often provide dashboards and reports that visualize key metrics and trends, making it easier to understand system health and identify areas for improvement.

Why other options are less suitable:

Synthetic monitoring: This focuses on simulating user interactions to proactively detect availability and performance issues. It's beneficial for testing user experience but doesn't directly address log parsing and root cause analysis.
Directed monitoring: While not a standard industry term in this specific context, it could imply pre-defined monitoring parameters, which might lack the flexibility to adapt to complex issues.
Alert Management: Alert management is essential for notifying teams of issues, but it relies on the underlying monitoring system to detect those issues. It doesn't contribute to parsing logs or finding root causes.

In summary, analytics provides the necessary capabilities to ingest, process, and analyze logs from multiple sources, identify patterns, and ultimately facilitate root cause analysis, making it the most suitable advanced feature for this monitoring solution.

Authoritative Links:

Azure Monitor Logs: https://learn.microsoft.com/en-us/azure/azure-monitor/logs/data-sources
Kusto Query Language (KQL) for Azure Monitor Logs: https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/

Question: 462 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You use Azure Pipelines to build and test a React.js application.
You have a pipeline that has a single job.
You discover that installing JavaScript packages from npm takes approximately five minutes each time you run the pipeline.
You need to recommend a solution to reduce the pipeline execution time.
Solution: You recommend using pipeline artifacts.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

The suggested solution of using pipeline artifacts is not the correct approach to reduce the npm package installation time in an Azure Pipeline. Pipeline artifacts are primarily used to persist build outputs (e.g., compiled code, test results) between pipeline stages or jobs, enabling their reuse or deployment. They are not designed to cache dependencies like npm packages.

The primary bottleneck is the repeated download and installation of npm packages on each pipeline run. To address this, caching mechanisms specifically designed for package management are needed. A more suitable solution would be to leverage Azure Pipelines' caching task or utilize tools like npm's built-in caching capabilities or tools like yarn that have built-in caching.

The Azure Pipelines caching task allows you to cache directories containing npm packages between pipeline runs. This avoids the need to download the packages from the npm registry every time the pipeline is executed. The pipeline stores the cached packages in a secure and efficient manner, resulting in significant time savings.

The correct solution would involve configuring the pipeline to cache the node_modules directory. This can be done by adding a Cache task to the pipeline YAML file, specifying the directory to cache and a key that identifies the cache. This key should include factors that influence the node_modules contents, such as the package-lock.json or yarn.lock file.

Therefore, while pipeline artifacts are useful for storing build outputs, they are not optimized for caching dependencies and are not the ideal solution for reducing npm package installation time in Azure Pipelines.

Here are some relevant resources for further research:

Azure Pipelines Cache Task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/utility/cache?view=azure-devops
Caching in Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/caching?view=azure-devops

Question: 463 CertyIQ

You use GitHub for source control of .NET applications.
You need to deploy a documentation solution that meets the following requirements:
- Documents will be written in Markdown as developers make code changes.
- Changes to the documents will trigger the recompilation of a static website.
- Users will access the documents from the static website.
- Documents will be stored in a GitHub repository.

Which two tools can you use to compile the website? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A. Word Press
B. Jekyll
C. DocFX
D. caret
E. Medium

Answer: BC

Explanation:

The correct answer is B. Jekyll and C. DocFX. Here's why:

B. Jekyll: Jekyll is a static site generator specifically designed for creating websites from Markdown files. It's a Ruby-based tool that takes Markdown documents and converts them into HTML, CSS, and JavaScript, generating a complete static website. This aligns perfectly with the requirement to write documentation in Markdown and compile it into a static website. Jekyll integrates well with GitHub Pages, offering a streamlined workflow for publishing documentation directly from a GitHub repository. The workflow of developers committing changes to Markdown documents, triggering a Jekyll build, and subsequently updating the live website fulfills the use case requirements. Jekyll's theming system enables customizing the look and feel of the documentation website.

C. DocFX: DocFX is another static site generator, primarily focused on generating documentation for .NET projects. While it supports other input formats, including Markdown, it's optimized for extracting API documentation from .NET code and combining it with hand-written documentation to create a comprehensive documentation site. DocFX's strength lies in automatically generating documentation from .NET code comments. Like Jekyll, DocFX takes Markdown files and converts them to a static website format, making it a suitable choice when developers are using .NET and want streamlined documentation.

Why other options are incorrect:

A. WordPress: WordPress is a dynamic content management system (CMS), primarily designed for blogs and content-rich websites. While WordPress can host documentation, it's not designed to compile Markdown files into a static website. It requires a database and server-side processing, which is not necessary for this requirement.
D. caret: I believe this should be Marked 2. Marked 2 is a Markdown previewer for macOS and has no capabilities of generating and maintaining a complete static website. Marked 2 is meant for instant previews and styling of single Markdown files.
E. Medium: Medium is a blogging platform, not a static site generator. It's used for publishing articles online, but it doesn't provide the functionality to compile Markdown files into a standalone static website that developers can host on their own.

In summary: Both Jekyll and DocFX are well-suited for compiling Markdown documentation into a static website, storing documentation within the same GitHub repository, and triggering recompilation upon changes. WordPress is a full-fledged CMS, and caret/Marked 2 and Medium are online platforms.

References:

Jekyll: https://jekyllrb.com/
DocFX: https://dotnet.github.io/docfx/

Question: 464 CertyIQ

You have an on-premises app named App1 that accesses Azure resources by using credentials stored in a configuration file.
You plan to upgrade App1 to use an Azure service principal.
What is required for App1 to programmatically sign in to Azure Active Directory (Azure AD)?

A. the application ID, a client secret, and the object ID
B. a client secret, the object ID, and the tenant ID
C. the application ID, a client secret, and the tenant ID
D. the application ID, a client secret, and the subscription ID

Answer: C

Explanation:

Here's a detailed justification for why option C is the correct answer, explaining the necessary components for an application like App1 to programmatically authenticate with Azure Active Directory (Azure AD) using a service principal.

A service principal is an identity created in Azure AD for use with applications, services, and automation tools. It provides a way for these entities to access Azure resources without requiring a user identity. To authenticate using a service principal, App1 needs specific credentials.

Application (Client) ID: This unique identifier represents the application (in this case, App1) registered in Azure AD. Azure AD uses this ID to identify the application requesting access.

Client Secret: The client secret is a password or certificate that App1 uses to prove its identity when requesting an access token. It's essential for secure authentication. This should be treated like a strong password and managed carefully.

Tenant ID: This identifier specifies the Azure AD tenant where App1 is registered and where the resources App1 intends to access reside. It helps Azure AD direct the authentication request to the correct directory.

Why other options are incorrect:

Option A (Application ID, Client Secret, Object ID): While the Application ID and Client Secret are necessary, the Object ID, in the context of service principals, refers to the specific instance of the service principal in a tenant after it's been granted access to a resource, which is not required at the initial authentication stage. Authentication to AAD always happens against a tenant, not an object.

Option B (Client Secret, Object ID, Tenant ID): While Client Secret and Tenant ID are required, the Object ID, as defined above, is not required at the initial authentication stage. The Application ID is the core ID needed for authentication.

Option D (Application ID, Client Secret, Subscription ID): The Application ID and Client Secret are necessary, but the Subscription ID identifies a billing boundary, and isn't directly involved in Azure AD authentication. The Tenant ID is essential for directing the authentication to the correct directory.

Therefore, the correct answer is C: the Application ID, Client Secret, and Tenant ID. These three pieces of information are the minimum required for an application like App1 to programmatically sign in to Azure Active Directory (Azure AD) using a service principal.

Supporting Documentation:

Service Principals and Managed Identities: https://learn.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals
Authenticate Azure-hosted apps with managed identities: https://learn.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/overview

Question: 465 CertyIQ

HOTSPOT -
You have a virtual machine that runs Windows Server 2019 and is managed by using Desired State Configuration (DSC).
You have the following DSC configuration.

```
configuration WebConfiguration
{

    File WebsiteContent {
        Ensure              = 'Present'
        SourcePath          = 'c:\test\index.htm'
        DestinationPath     = 'c:\inetpub\wwwroot'
        DependsOn           = '[WindowsFeature]Web-Server'
    }

    WindowsFeature Web-Server
    {
        Ensure  = 'Present'
        Name    = 'Web-Server'
    }

}
```

You have the following Local Configuration Manager (LCM) configuration.

```
LocalConfigurationManager
{
    ConfigurationMode = “ApplyAndMonitor”
    RefreshFrequencyMins = 30
    ConfigurationModeFrequencyMins = 60
    RefreshMode = 'Push'
}
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No.
NOTE: Each correct selection is worth one point.
Hot Area:

Statements	Yes	No
The Index.htm file will be copied to the C:\Test folder before the Web-Server Windows feature is installed.	○	○
If manual changes are made to the configuration of the virtual machine, the configuration will reapply automatically.	○	○
If the Web-Server Windows feature is uninstalled from the virtual machine, the discrepancy will be reported in a log entry within 60 minutes.	○	○

Answer:

Statements	Yes	No
The Index.htm file will be copied to the C:\Test folder before the Web-Server Windows feature is installed.	○	○
If manual changes are made to the configuration of the virtual machine, the configuration will reapply automatically.	○	○
If the Web-Server Windows feature is uninstalled from the virtual machine, the discrepancy will be reported in a log entry within 60 minutes.	○	○

Explanation:

No No Yes

https://docs.microsoft.com/en-us/powershell/dsc/managing-nodes/metaconfig?view=dsc-1.1

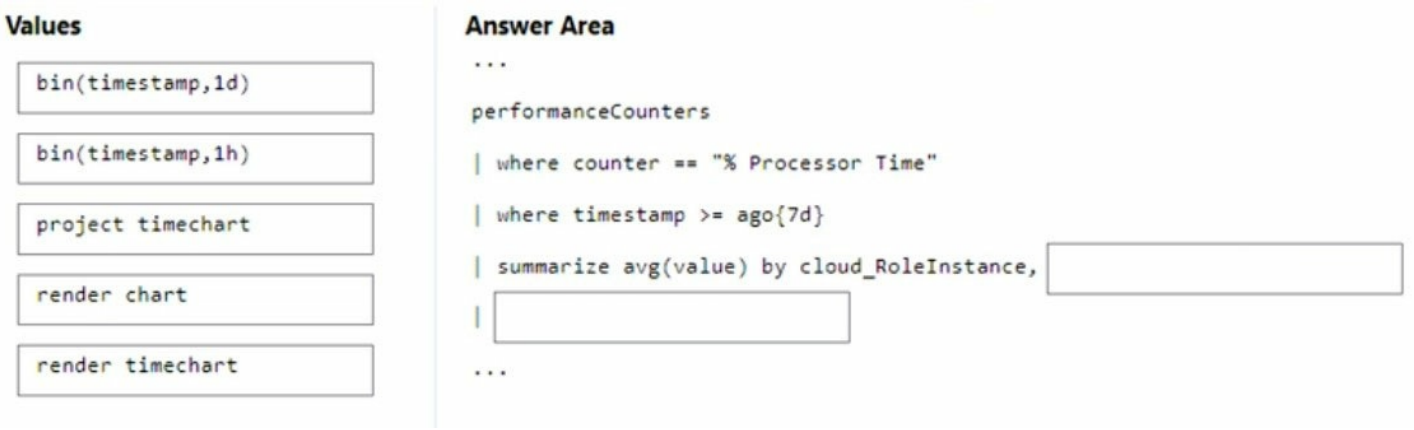
"ApplyAndMonitor: This is the default value. The LCM applies any new configurations. After initial application of a new configuration, if the target node drifts from the desired state, DSC reports the discrepancy in logs. Note that DSC will attempt to apply the configuration until it is successful before ApplyAndMonitor takes effect.

ApplyAndAutoCorrect: DSC applies any new configurations. After initial application of a new configuration, if the target node drifts from the desired state, DSC reports the discrepancy in logs, and then re-applies the current configuration.

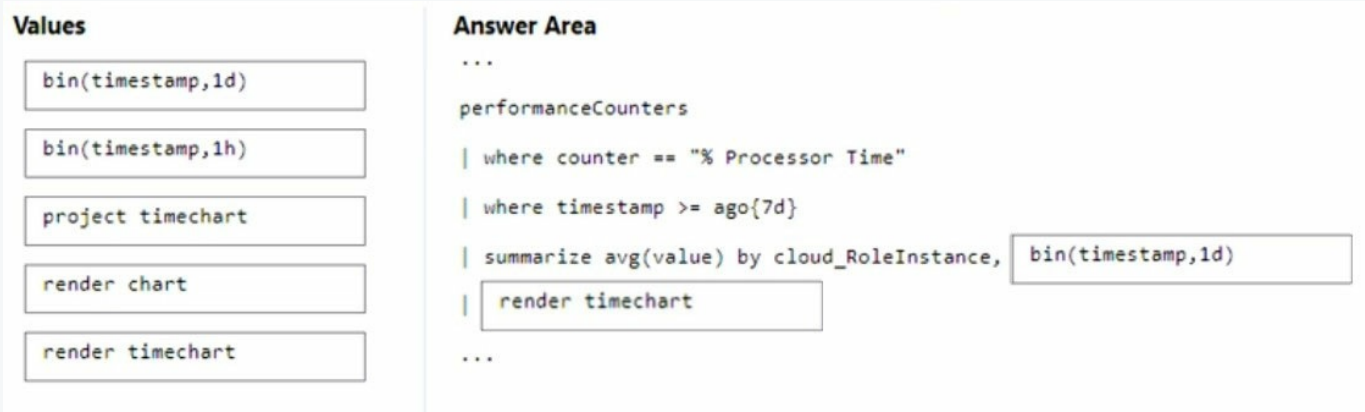
How often, in minutes, the current configuration is checked and applied. This property is ignored if the ConfigurationMode property is set to ApplyOnly. The default value is 15."

Question: 466 CertyIQ

DRAG DROP -
You have a web app named App1 that is hosted on multiple servers. App1 uses Application Insights in Azure Monitor.
You need to compare the daily CPU usage from the last week for all servers.
How should you complete the query? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all.
You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Question: 467 CertyIQ

You have an Azure subscription that contains 50 virtual machines.
You plan to manage the configuration of the virtual machines by using Azure Automation State Configuration.
You need to create the Desired State Configuration (DSC) configuration files.
How should you structure the code blocks?

A. Node > Configuration > Resource
B. Configuration > Resource > Node
C. Resource > Configuration > Node
D. Configuration > Node > Resource

Answer: D

Explanation:

The correct structure for DSC configuration files in Azure Automation State Configuration is **Configuration > Node > Resource**. This structure reflects the logical flow of defining the desired state:

1. **Configuration:** The outermost block, Configuration <ConfigurationName>, defines the overall configuration document. It acts as a template or recipe for configuring the nodes. Think of it as defining what you want to achieve in terms of the final desired state.
2. **Node:** The Node <NodeName> block specifies which target machines (nodes) the configuration applies to. This block is nested within the Configuration block. You can have multiple Node blocks within a single configuration, each targeting a different machine or a group of machines. This defines where the configuration should be applied.

3. **Resource:** The Resource block, located inside the Node block, describes the specific resources to be configured and their desired state. Resources are the individual components or properties of the node that you want to manage, such as files, services, registry keys, or packages. This block specifies how to achieve the desired state on a specific node, specifying the actual configuration settings. Resources are the building blocks of the configuration.

Therefore, you define the overall configuration (Configuration), then specify which nodes it applies to (Node), and finally, define the specific resources to manage on those nodes (Resource). Options A, B and C do not follow this hierarchical structure which is necessary for the DSC compiler to correctly interpret and apply the configuration.

Microsoft documentation provides detailed explanation of DSC structure:

Desired State Configuration Resources: https://learn.microsoft.com/en-us/powershell/dsc/resource/resource?view=dsc-2.0
Getting Started with DSC: https://learn.microsoft.com/en-us/powershell/dsc/overview?view=dsc-2.0

Question: 468 CertyIQ

Your team uses Azure Pipelines to deploy applications.
You need to ensure that when a failure occurs during the build or release process, all the team members are notified by using Microsoft Teams. The solution must minimize development effort.
What should you do?

A. Install the Azure Boards app for Teams and configure a subscription to receive notifications in a channel.
B. Use Azure Automation to connect to the Azure DevOps REST API and notify the team members.
C. Use an Azure function to connect to the Azure DevOps REST API and notify the team members.
D. Install the Azure Pipelines app for Teams and configure a subscription to receive notifications in a channel.

Answer: D

Explanation:

The correct answer is D, installing the Azure Pipelines app for Teams and configuring a subscription to receive notifications in a channel. This approach minimizes development effort by leveraging a pre-built integration between Azure DevOps and Microsoft Teams. The Azure Pipelines app provides a straightforward mechanism to receive real-time notifications about pipeline events, including failures, directly within a Teams channel.

Option A, installing the Azure Boards app, is incorrect because Azure Boards focuses on work item tracking (e.g., tasks, bugs, features) and not pipeline execution status. While it allows for notifications related to work items, it doesn't directly address the requirement of notifying the team about build and release failures.

Options B and C, using Azure Automation or Azure Functions, are unnecessarily complex. While they could technically achieve the desired outcome by interacting with the Azure DevOps REST API, they would require significant custom code and management overhead. These options involve writing and maintaining code to authenticate with Azure DevOps, query pipeline status, and format and send notifications to Teams. This increases development effort significantly compared to using the Azure Pipelines app.

The Azure Pipelines app offers a low-code/no-code solution, as it provides a user-friendly interface to subscribe to specific pipeline events. You can simply select the pipeline and the events (e.g., build failed, release failed) you want to be notified about, and the app handles the rest. This simplifies the configuration and maintenance of notifications, aligning with the requirement to minimize development effort. The app leverages webhooks and integrations already established between Azure DevOps and Teams, streamlining

the notification process.

Therefore, utilizing the built-in capabilities of the Azure Pipelines app for Teams is the most efficient and practical approach to notify team members of pipeline failures.

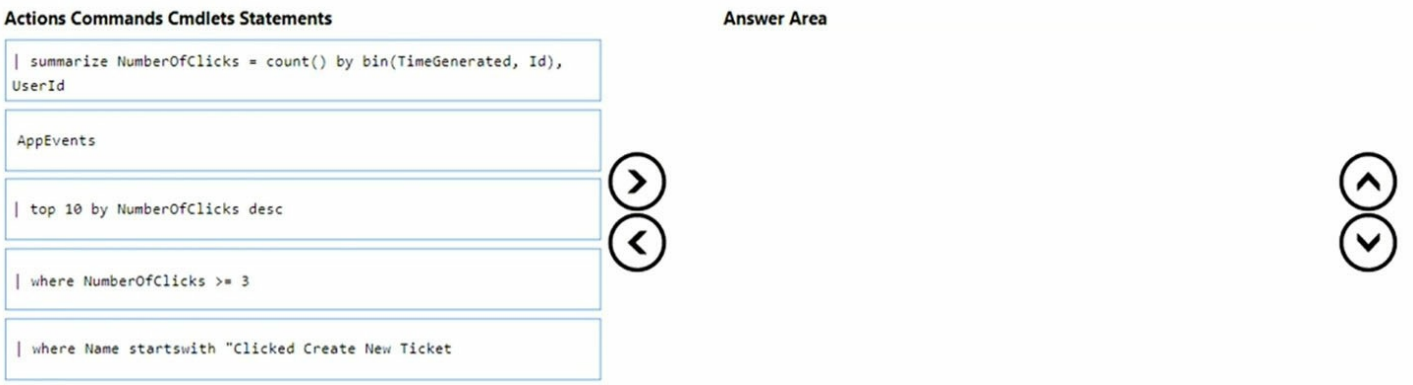
Relevant Links:

Azure Pipelines app for Teams: https://learn.microsoft.com/en-us/azure/devops/pipelines/integrations/microsoft-teams?view=azure-devops

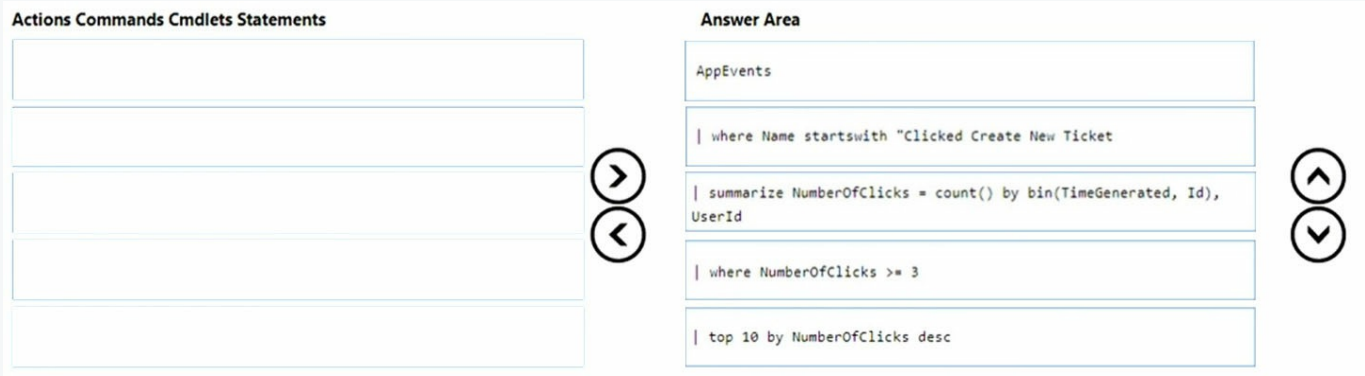
Question: 469 CertyIQ

DRAG DROP -
You have an app named App1. You have a Log Analytics workspace named Workspace1 that contains a table named AppEvents. App1 writes logs to
Workspace1.
You need to query the AppEvents table. The solution must meet the following requirements:
- Only query rows for a column named Name that starts with the following text: "Clicked Create New Ticket."
- Calculate the number of daily clicks per user.
- Return the top 10 users based on their number of clicks for any day.
- Sort the results based on the highest number of clicks.
- Ignore any users who have less than three daily clicks.

In which order should you arrange the query statements? To answer, move all statements from the list of statements to the answer area and arrange them in the correct order.
Select and Place:



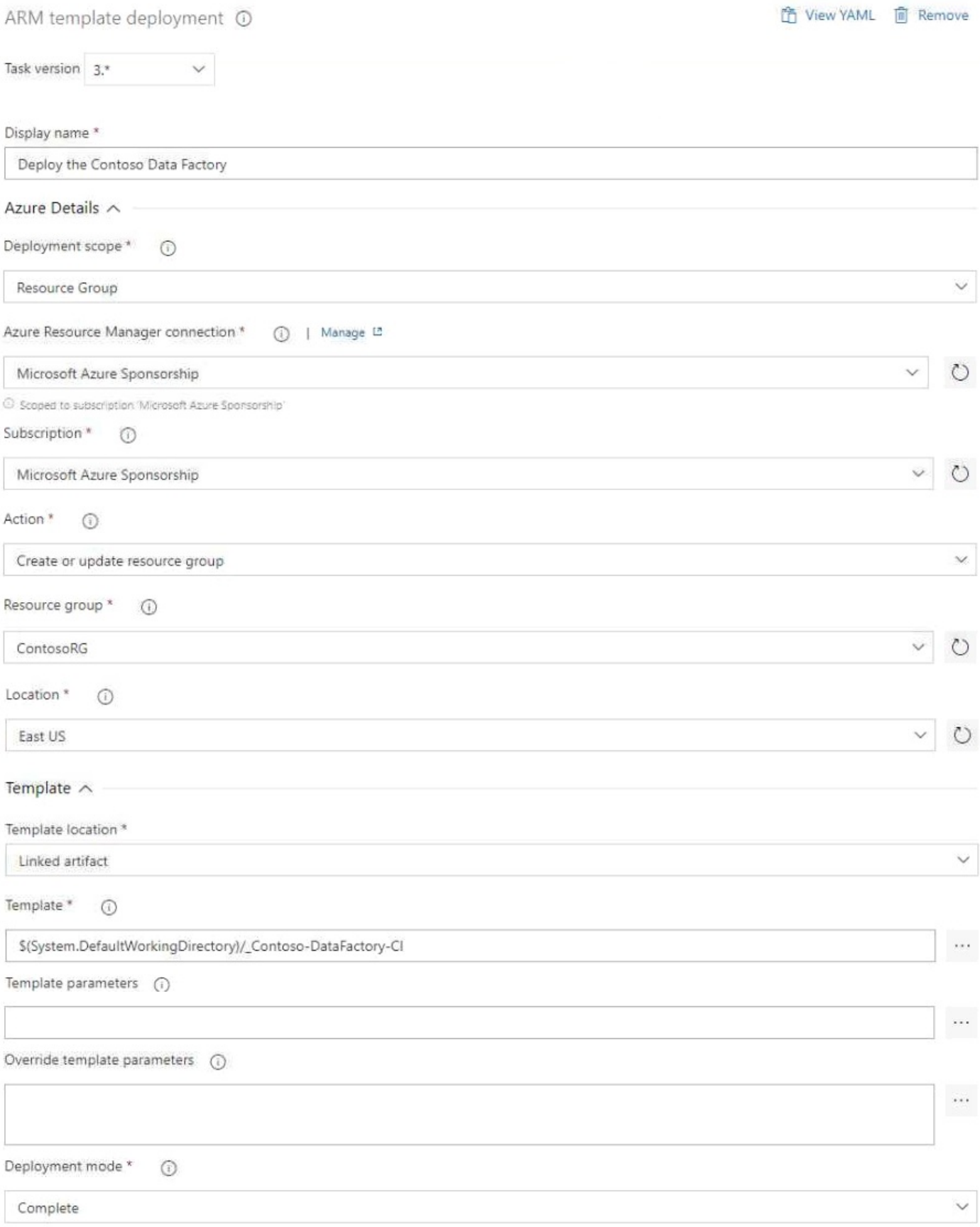
Answer:



Question: 470 CertyIQ

HOTSPOT
-
You have an Azure subscription that contains two resource groups named ContosoRG and ContosoDev, an Azure data factory named Contoso Data Factory, and a release pipeline in Azure Pipelines named Pipeline1.
You plan to deploy Contoso Data Factory to ContosoRG by using Pipeline1.

You add the Azure Resource Manager (ARM) template deployment task shown in the following exhibit.



Use the drop-down menus to select the answer choice that completes each statement based on the information presented in the graphic
NOTE: Each correct selection is worth one point.

The **[answer choice]** setting must be changed to prevent the modification of existing databases and web apps in ContosoRG.

- Action
- Template location
- Deployment mode
- Deployment scope

Pipeline1 will retrieve the ARM template from the **[answer choice]**.

- output of the continuous integration build
- location specified in the Linked artifact variable
- default branch of the Git repository of Contoso Data Factory

Answer:

The **[answer choice]** setting must be changed to prevent the modification of existing databases and web apps in ContosoRG.

- Action
- Template location
- **Deployment mode**
- Deployment scope

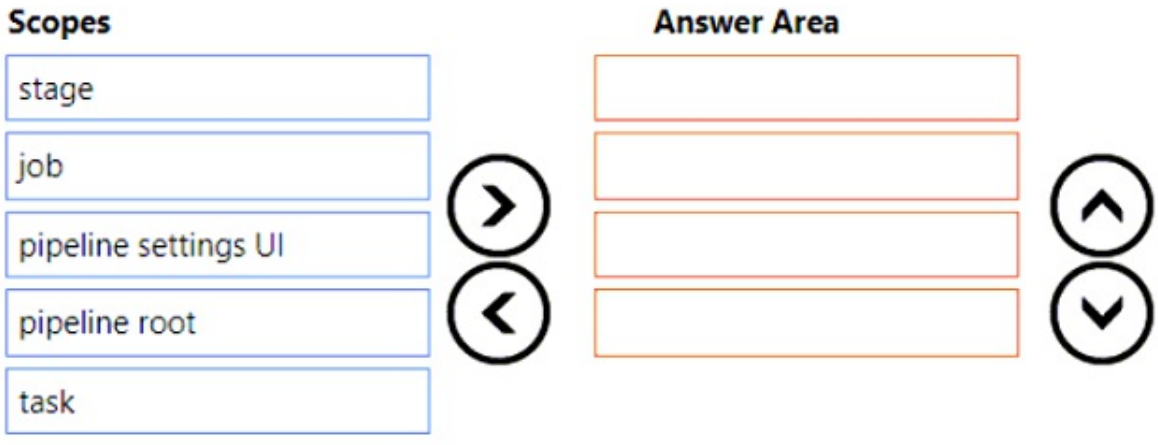
Pipeline1 will retrieve the ARM template from the **[answer choice]**.

- output of the continuous integration build
- **location specified in the Linked artifact variable**
- default branch of the Git repository of Contoso Data Factory

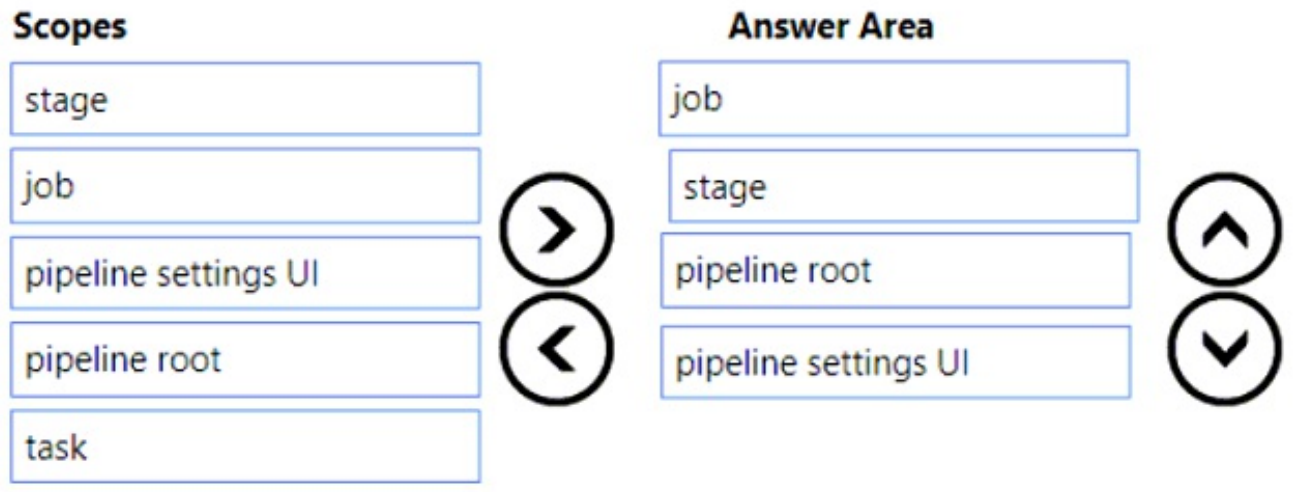
Question: 471

CertyIQ

DRAG DROP
-
You have an Azure Pipeline.
You need to store configuration values as variables.
At which four scopes can the variables be defined, and what is the precedence of the variables from the highest precedence to lowest precedence? To answer, move the appropriate scope from the list of scopes to the answer area and arrange them in the correct order.



Answer:



Explanation:

1. job

2. stage

3. pipeline root

4. pipeline settings UI

https://learn.microsoft.com/en-us/azure/devops/pipelines/process/variables?view=azure-devops&tabs=yaml%2Cbatch#variable-scopes

In the YAML file, you can set a variable at various scopes:

- At the root level, to make it available to all jobs in the pipeline.

- At the stage level, to make it available only to a specific stage.

- At the job level, to make it available only to a specific job.

When you define a variable at the top of a YAML, the variable is available to all jobs and stages in the pipeline and is a global variable. Global variables defined in a YAML aren't visible in the pipeline settings UI.

Question: 472 CertyIQ

You have a project in Azure DevOps named Project1 that contains two environments named environment1 and environment2.
When a new version of Project is released, the latest version is deployed to environment2, and the previous version is redeployed to environment1.
You need to distribute users across the environments. The solution must meet the following requirements:
• New releases must be available to only a subset of the users.
• You must gradually increase the number of users that can access environment2.
What should you use?

A. VIP swaping
B. web app deployment slots
C. Azure Load Balancer
D. Azure Traffic Manager

Answer: B

Explanation:

The correct answer is **B. web app deployment slots**.

Web app deployment slots are a feature in Azure App Service that allows you to deploy different versions of your web app to different environments within the same App Service plan. This directly addresses the requirement of deploying the latest version to environment2 (for a subset of users) and the previous version to environment1.

Deployment slots enable gradual rollouts through traffic routing. Initially, you can route a small percentage of users (e.g., 10%) to environment2 (the staging slot with the new release). This allows for testing with real users in a production-like environment without impacting the majority of users. Over time, you can gradually increase the traffic routed to environment2, satisfying the requirement to gradually increase the number of users accessing it. If issues arise, you can quickly swap the staging slot (environment2) back to the production slot (environment1) to revert to the previous version, minimizing downtime. No code changes are required for these swaps.

VIP swaping (A) isn't a standard Azure feature, so it's unlikely to be the right solution. Azure Load Balancer (C) focuses on distributing traffic across multiple instances of the same application, not different versions in different environments. It doesn't inherently support gradual rollouts or easy reversion. Azure Traffic Manager (D) is for routing traffic to different regions or endpoints, not different versions of the same application within the same region. It's typically used for high availability and disaster recovery, not for staged deployments within a single environment. Deployment slots are specifically designed for this use case.

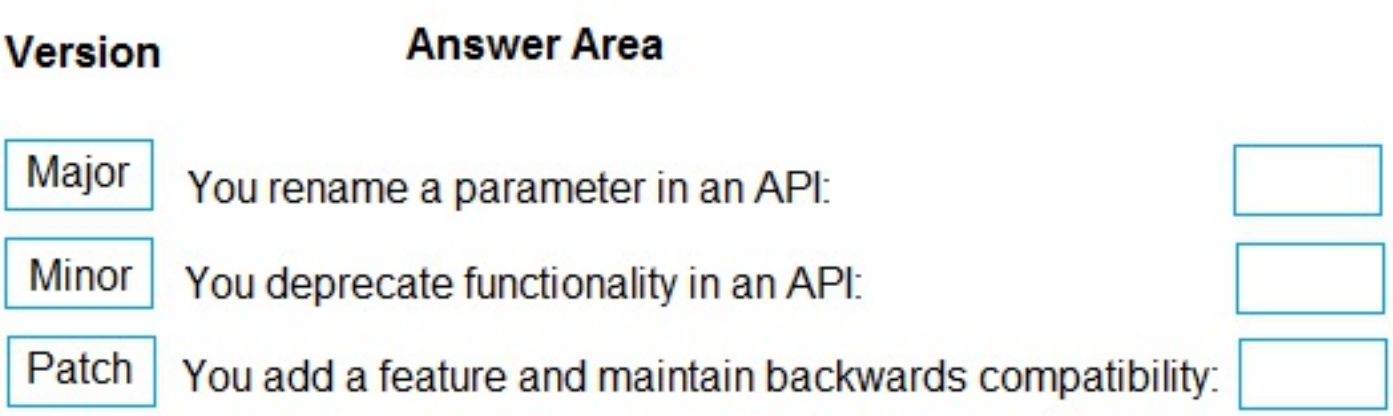
Therefore, web app deployment slots are the ideal choice because they provide version management, traffic routing for gradual rollouts, and easy reversion capabilities, perfectly aligning with the given requirements.

Further Research:

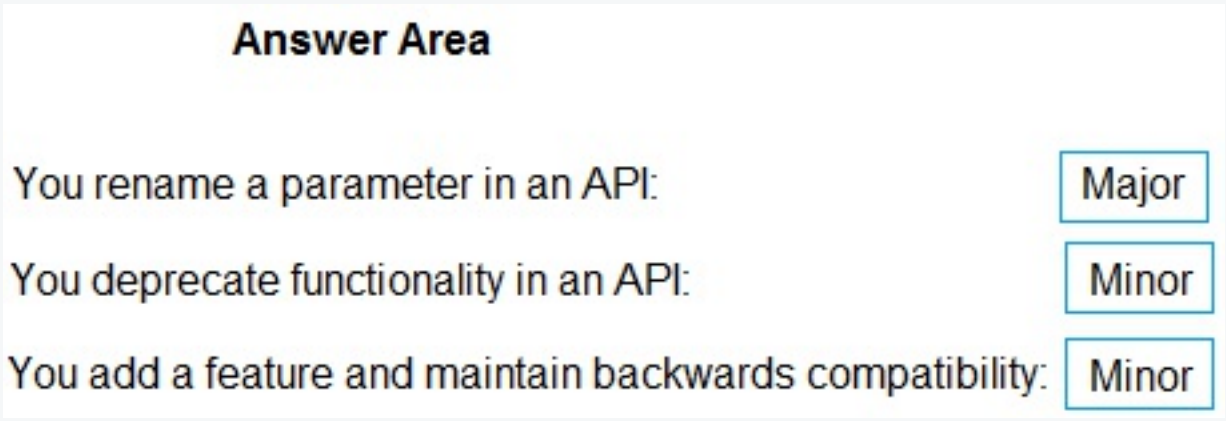
Azure App Service deployment slots: https://learn.microsoft.com/en-us/azure/app-service/deploy-staging-slots

Question: 473 CertyIQ

DRAG DROP
-
You are designing a versioning strategy for Git-based packages.
You plan to use a Semantic Versioning (SemVer)-based strategy.
You need to identify when to change the build version.
What should you identify for each scenario? To answer, drag the appropriate versions to the correct scenarios.
Each version may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

1. **MAJOR**, Rename a parameter in an API, potential to cause backward-incompatibility if the parameter is referenced in some part of the code
2. **MINOR**, Deprecate a funcionality in an API, it won't cause backward-incompatibility inmediatly because you only mark it like deprecated and provide the warning that the funcionality will be removed in future versions
3. **MINOR**, Add a feature and maintain backwards compatibility, because won't cause backward-incompatibility

Question: 474 CertyIQ

You use Calendar Versioning (CalVer) for code assets.
You need to store an optional tag of beta as part of the version.
Which part of the version should you use for the tag?

A. minor
B. major
C. micro
D. modifier

Answer: D

Explanation:

The correct answer is **D. modifier**. Here's a detailed justification:

CalVer (Calendar Versioning) is a versioning scheme that uses dates as version numbers. A common format is YYYY.MM.DD, representing Year, Month, and Day. However, CalVer often extends to include additional fields for clarity and manageability.

The question specifically mentions the need to store an optional "beta" tag as part of the version. Considering this requirement, let's examine why the other options are less suitable.

Major (B): Major versions usually indicate significant architectural changes or feature overhauls that break backward compatibility. A simple beta tag wouldn't typically warrant a change in the major version.
Minor (A): Minor versions typically represent additions of new features or improvements without breaking compatibility. A beta tag denoting a pre-release state doesn't fall under this definition.
Micro (C): Micro versions (or patch versions) often represent bug fixes or minor tweaks. Again, a beta label is a status indicator rather than a bug fix.

The **modifier (D)** field is specifically designed to handle such qualifiers. It's often used to indicate pre-release versions (alpha, beta, release candidate), post-release versions (patch levels beyond the standard micro number), or other qualifiers regarding the build's status or intended usage. Common examples are 2024.01.01-beta1 or 2024.01.01-rc. This approach keeps the core date-based version clean while providing additional context. The hyphen separator (-) convention is often used to separate the date-based portion from the modifier.

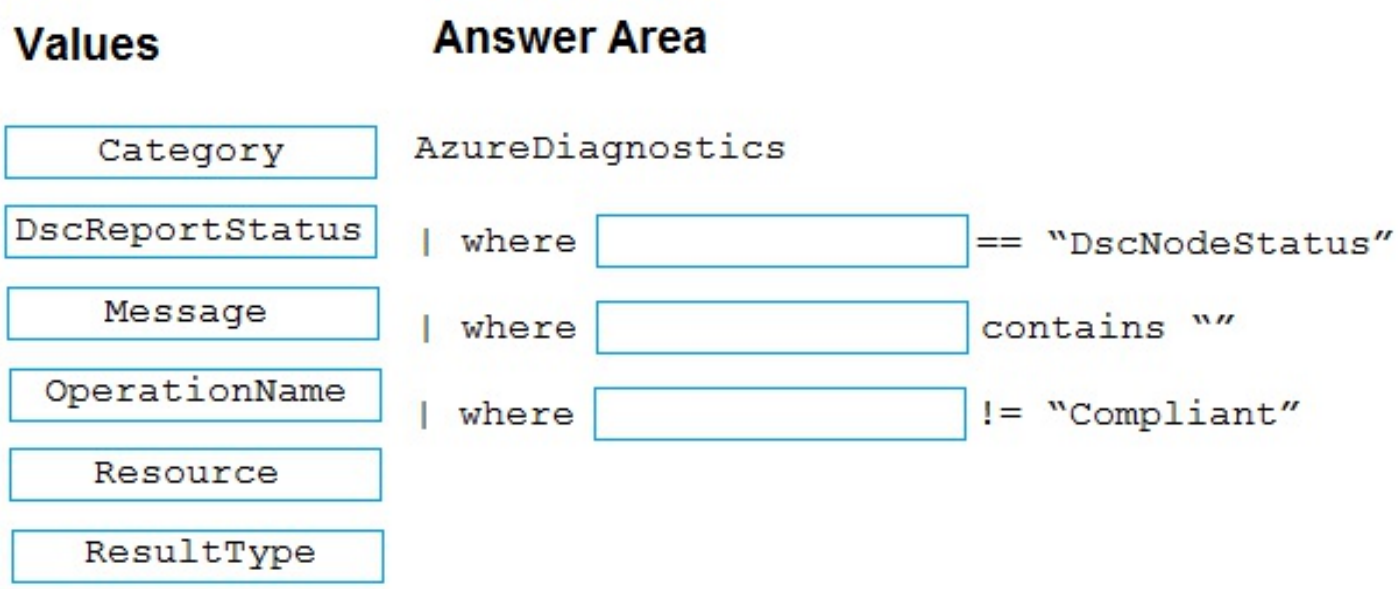
Therefore, using the modifier field allows you to clearly indicate that a specific build is a beta version without affecting the core CalVer structure, providing greater flexibility and clarity in your versioning strategy. It aligns perfectly with the requirement of having an optional tag, as it can be omitted when a build is no longer in a pre-release state.

For further research, consult these resources:

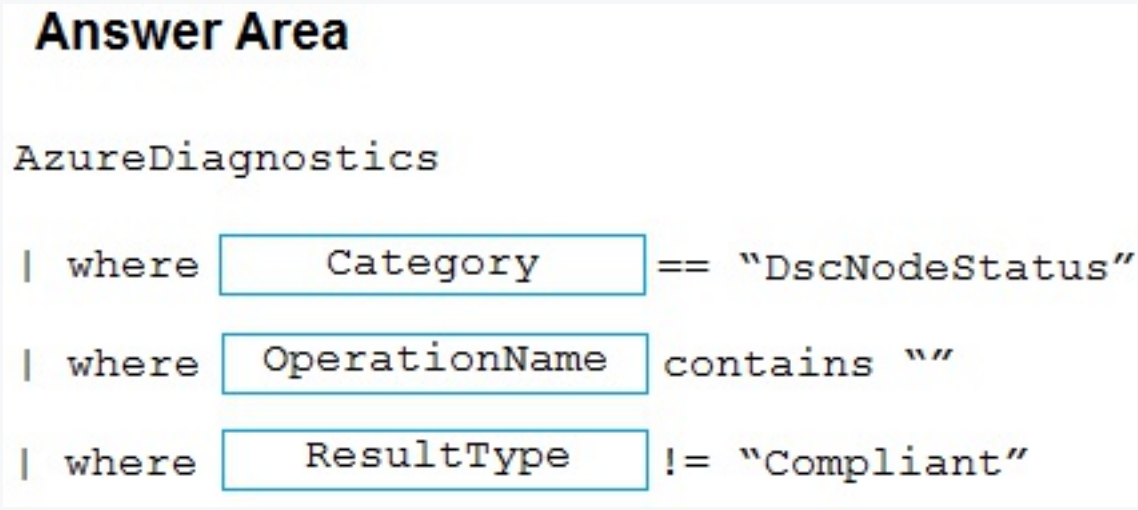
CalVer Specification: https://calver.org/ (This website details the full CalVer specification and best practices.)
Semantic Versioning (SemVer) Comparison: Understanding SemVer can provide context when comparing to CalVer and seeing how similar concepts like "pre-release" are handled. https://semver.org/

Question: 475 CertyIQ

DRAG DROP
-
You have an Azure subscription that uses Azure Automation State Configuration to manage the configuration of virtual machines.
You need to identify which nodes are noncompliant.
How should you complete the query? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Question: 476 CertyIQ

You have a project in Azure DevOps named Project1.
You need to ensure that all new pipelines in Project1 execute three specific tasks during pipeline execution.
What should you create?

A. a task group
B. a JSON template
C. a YAML template
D. a PowerShell task

Answer: A

Explanation:

The correct answer is A: a task group. Here's why:

Task groups in Azure DevOps are designed to encapsulate a sequence of tasks into a reusable unit. This allows you to define a set of steps once and then easily reuse them across multiple pipelines. In this scenario, the requirement is to ensure that all new pipelines execute three specific tasks. A task group perfectly

addresses this by bundling those three tasks together. When creating a new pipeline, you can simply include the task group, guaranteeing those tasks are executed.

JSON templates and YAML templates are used for defining the structure of a pipeline, including stages, jobs, and tasks. However, they don't inherently enforce the execution of specific tasks in all new pipelines unless you manually include the template reference in each pipeline definition. While templates provide reusability, they lack the enforcement mechanism that task groups offer regarding mandatory task execution. Every new pipeline would still need to explicitly include the template.

PowerShell tasks, while useful for executing scripts, are individual tasks rather than containers for multiple tasks designed for reuse across pipelines. Using only a PowerShell task would mean copying and pasting the same script (effectively the same tasks) into every new pipeline, which is prone to error and difficult to maintain.

Task groups, in contrast, are easily discoverable and reusable within the Azure DevOps UI and can be version controlled like other Azure DevOps artifacts. They promote consistency and reduce redundancy, as opposed to templates that still require explicit inclusion or PowerShell tasks that lack task aggregation. Task groups centralize the definition of the required tasks, simplifying maintenance.

Therefore, the most efficient and reliable way to ensure that all new pipelines in Project1 execute the three specified tasks is to create a task group containing those tasks.

Authoritative Links:

Task Groups: https://learn.microsoft.com/en-us/azure/devops/pipelines/library/task-groups?view=azure-devops
Templates: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/templates?view=azure-devops

Question: 477 CertyIQ

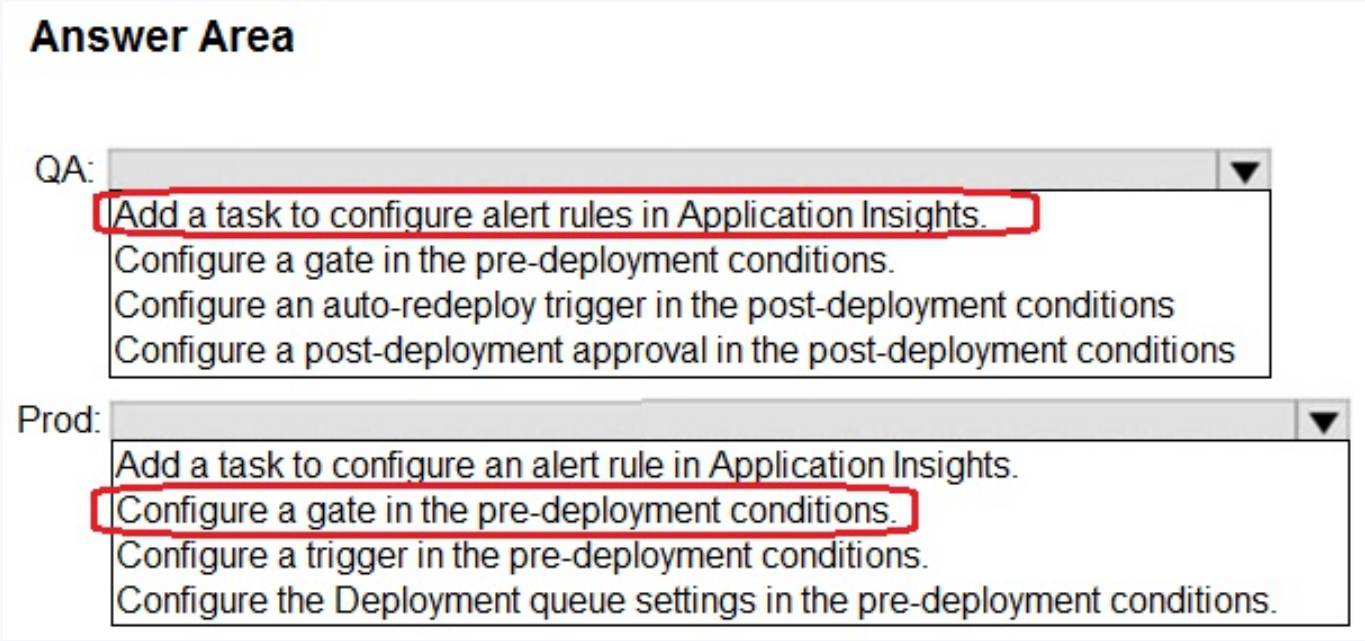
HOTSPOT -
You have a project in Azure DevOps that contains a release pipeline. The pipeline contains two stages named QA and Prod. QA deploys code to an Azure web app named webapp1. Prod deploys code to an Azure web app named webapp2.
You need to ensure that code deployments to webapp2 are blocked if Azure Application Insights generates Failed requests alerts following the deployment of new code to webapp1.
What should you do for each stage? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

QA:
- Add a task to configure alert rules in Application Insights.
- Configure a gate in the pre-deployment conditions.
- Configure an auto-redeploy trigger in the post-deployment conditions
- Configure a post-deployment approval in the post-deployment conditions

Prod:
- Add a task to configure an alert rule in Application Insights.
- Configure a gate in the pre-deployment conditions.
- Configure a trigger in the pre-deployment conditions.
- Configure the Deployment queue settings in the pre-deployment conditions.

Answer:



Explanation:

"search for and select Azure App Service deployment with continuous monitoring, and then select Apply."

https://learn.microsoft.com/en-us/azure/azure-monitor/app/continuous-monitoring

Question: 478 CertyIQ

You use Azure DevOps processes to build and deploy code.
You need to compare how much time is spent troubleshooting issues found during development and how much time is spent troubleshooting issues found in released code.
Which KPI should you use?

A. defect escape rate
B. unplanned work rate
C. defect rate
D. rework rate

Answer: A

Explanation:

The correct answer is **A. defect escape rate**. Here's why:

The defect escape rate directly addresses the problem statement: comparing the effort spent on issues found during development versus after release. Defect escape rate specifically measures the percentage of defects that "escape" the development and testing phases and are found in production or by end-users. A higher escape rate indicates more problems are being found later in the lifecycle, thus requiring more troubleshooting effort in released code.

Options B, C, and D, while related to software quality, do not directly compare pre-release and post-release troubleshooting effort in the same targeted way. Unplanned work rate refers to tasks not originally planned for a sprint or iteration. Defect rate measures the overall number of defects found but doesn't distinguish where they were found. Rework rate measures the amount of code that needs to be redone, which while

important, is an indirect measure of the troubleshooting effort being compared. Defect escape rate allows a clear and quantifiable comparison of where troubleshooting efforts are most heavily spent - during development (when escapes are low) or in production (when escapes are high). By tracking this KPI, you can identify areas where your testing or development processes need improvement to reduce the number of issues making their way into the released product. Lowering the defect escape rate directly translates to less time spent troubleshooting issues in production. A lower escape rate is a sign of improved software quality processes. It helps teams focus their efforts on improving the quality of the software before it is released.

Further research on defect escape rate:

Software Engineering Institute (SEI) - Measuring Defect Escape: While not directly Azure DevOps related, it provides a general definition.
https://resources.sei.cmu.edu/asset_files/TechnicalReport/1992_003_000714.pdf (search for Defect Escape Rate)

Question: 479 CertyIQ

You have a project in Azure DevOps named Project1.
You implement a Continuous Integration/Continuous Deployment (CI/CD) pipeline that uses PowerShell Desired State Configuration (DSC) to configure the application infrastructure.
You need to perform a unit test and an integration test of the configuration before Project1 is deployed.
What should you use?

A. the PSScriptAnalyzer tool
B. the Pester test framework
C. the PSCodeHealth module
D. the Test-DscConfiguration cmdlet

Answer: B

Explanation:

The correct answer is **B. the Pester test framework**.

Pester is a widely used and well-regarded testing framework specifically designed for PowerShell. It provides a structured and readable syntax for writing unit and integration tests, making it ideal for verifying the correctness of PowerShell scripts and modules, including those used in DSC configurations. Before deploying Project1's DSC configuration, you need a way to ensure that the configuration applies correctly and that the intended state of the infrastructure is achieved. Pester offers a comprehensive approach to writing tests that validate the configuration.

Pester allows you to write tests that check specific properties and settings within your DSC resources. For instance, you can test whether a file is created with the correct content, a service is running, or a registry key is set to the desired value. This ability to test the outcome of your DSC configuration makes Pester the ideal choice. The integration tests are equally important, and Pester is also suited for this task.

A, PSScriptAnalyzer is a static code analysis tool. While helpful for identifying potential issues and stylistic inconsistencies in PowerShell code, it doesn't execute the configuration or verify its outcome. It is a good complement to Pester, improving code quality, but it does not address the requirement for testing the applied DSC configuration.

C, PSCodeHealth primarily focuses on code style and best practices. Like PSScriptAnalyzer, it's a valuable tool for code improvement, but not for testing DSC configurations before deployment.

D, Test-DscConfiguration is useful for testing the existence of DSC resources, but it doesn't let you easily create specific test cases to validate that the configurations are doing what you expect. Pester has been built for that exact purpose.

In summary, Pester offers the most comprehensive solution for unit and integration testing of DSC configurations before deployment, guaranteeing that the infrastructure is set up as intended.

Authoritative Links:

Pester: https://pester.dev/docs/
PowerShell Desired State Configuration (DSC): https://learn.microsoft.com/en-us/powershell/dsc/overview?view=dsc-3.0

Question: 480 CertyIQ

You use Azure Repos to manage source code and Azure Pipelines to implement continuous integration and continuous deployment (CI/CD).
You need to ensure that all comments on pull requests are resolved before the pull requests are included in a build. The solution must minimize administrative effort.
What should you include in the solution?

A.a custom action
B.a post-deployment gate
C.a branch policy
D.a pre-deployment gate

Answer: C

Explanation:

The correct answer is **C. a branch policy**. Here's why:

Branch policies in Azure Repos are specifically designed to enforce code quality and process adherence directly on branches before merges occur. They are lightweight, integrated, and require minimal administrative overhead. Adding a branch policy that requires all comments to be resolved before a pull request can be merged directly addresses the requirement of ensuring all comments are resolved. The policy prevents the merge from happening if there are unresolved comments, effectively gatekeeping the branch.

Let's examine why the other options are less suitable:

A. A custom action: While custom actions in Azure Pipelines offer flexibility, they require more manual scripting and management compared to a branch policy. They also don't directly prevent a pull request from being merged if comments are unresolved; the pipeline would have to specifically check and fail, adding complexity.

B. A post-deployment gate: Post-deployment gates operate after a deployment has occurred. The requirement is to prevent pull requests with unresolved comments from even being included in a build, making a post-deployment gate irrelevant to the problem.

D. A pre-deployment gate: Pre-deployment gates control the deployment process after a build has been created. While a gate could theoretically check for unresolved comments, it would be a late stage in the process and less efficient than preventing the merge in the first place using a branch policy. The correct place to gate this particular requirement is at the source code level, which the branch policy directly accomplishes.

In essence, branch policies provide the most direct and efficient way to enforce the requirement that all pull request comments must be resolved before the code is merged into a target branch, fitting the criteria of minimal administrative effort.

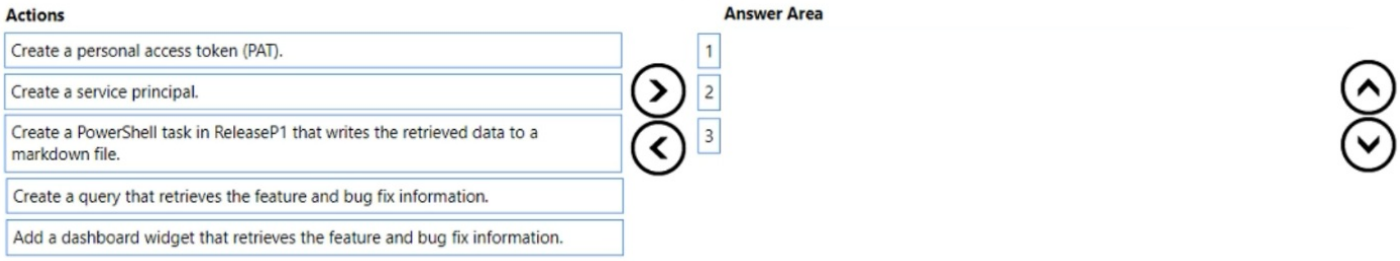
Authoritative Links:

Azure Repos branch policies
Understanding Azure Pipelines gates and approvals

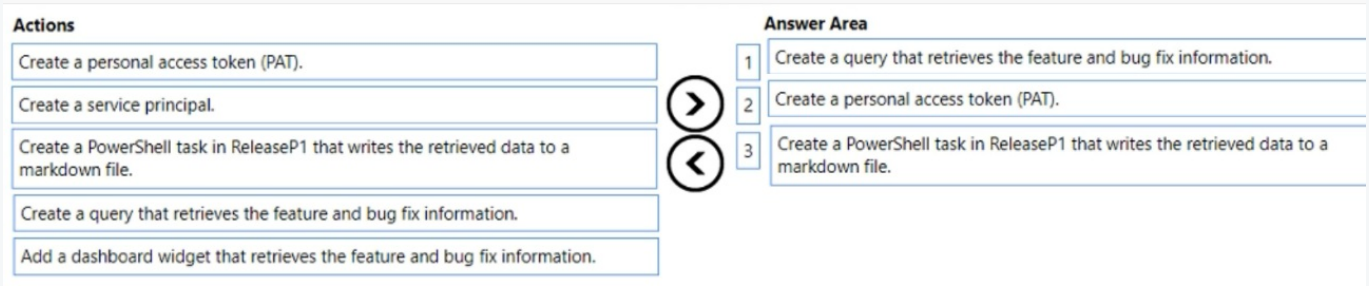
Question: 481

CertyIQ

DRAG DROP
-
You have a project in Azure DevOps named Project that has a release pipeline in Azure Pipeline named ReleaseP1.
You need to ensure that when a new release is generated for ReleaseP1, a new release note document is created.
The release notes must contain new features and bug fixes.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.



Answer:



Explanation:

1. Create Query that retrives the feature and bux fix information2. Create PAT3. Create Powershell task in ReleaseP1 that writes the retived data to a markdown file.

https://slothycode.com/azure-pipelines-publish-release-notes/

Question: 482

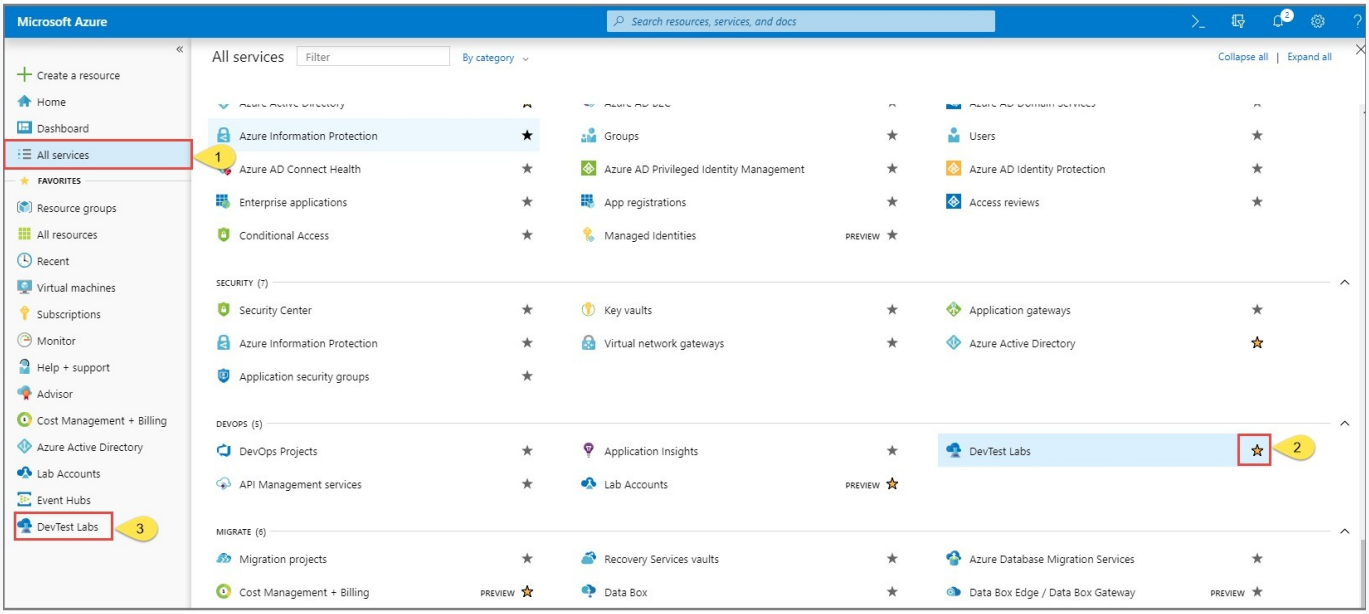
CertyIQ

SIMULATION
-
You need to configure a virtual machine template in a DevTest Labs environment named az400-123456789-dtl1.
The operating system must be based on Windows Server 2016 Datacenter. Virtual machines created from the DevTest Lab must include the Selenium tool and the Google Chrome browser.
To complete this task, sign in to the Microsoft Azure portal.

Answer:

Step 1: Open Microsoft Azure Portal

Step 2: Select All Services, and then select DevTest Labs in the DEVOPS section.



Step 3: From the list of labs, select the az400-123456789-dtl1 lab.

Step 4: In the home page for your lab, select + Add on the toolbar.

Step 5: Select the Windows Server 2016 Datacenter base image for the VM.

Step 6: .In the Basics Tab at the bottom, Click on Artifacts

Step 7: Select Selenium.
Note: Google Chrome is part of the Selenium artifact, so you only need to select Selenium.

Step 8: Confirm with OK + OK.

Step 9: Verification: Click on tab Advanced Settings, View ARM template, Search for artifacts
You will see windows selenium; search for data center and verify the image corresponds to the one in the task.

Step 9: Close, and go back to the Basic Settings.

Step 10. Select Create-

Reference:
https://docs.microsoft.com/bs-cyrl-ba/azure//lab-services/devtest-lab-vm-powershell
https://docs.microsoft.com/en-us/azure/devtest-labs/devtest-lab-add-vm#add-an-existing-artifact-to-a-vm

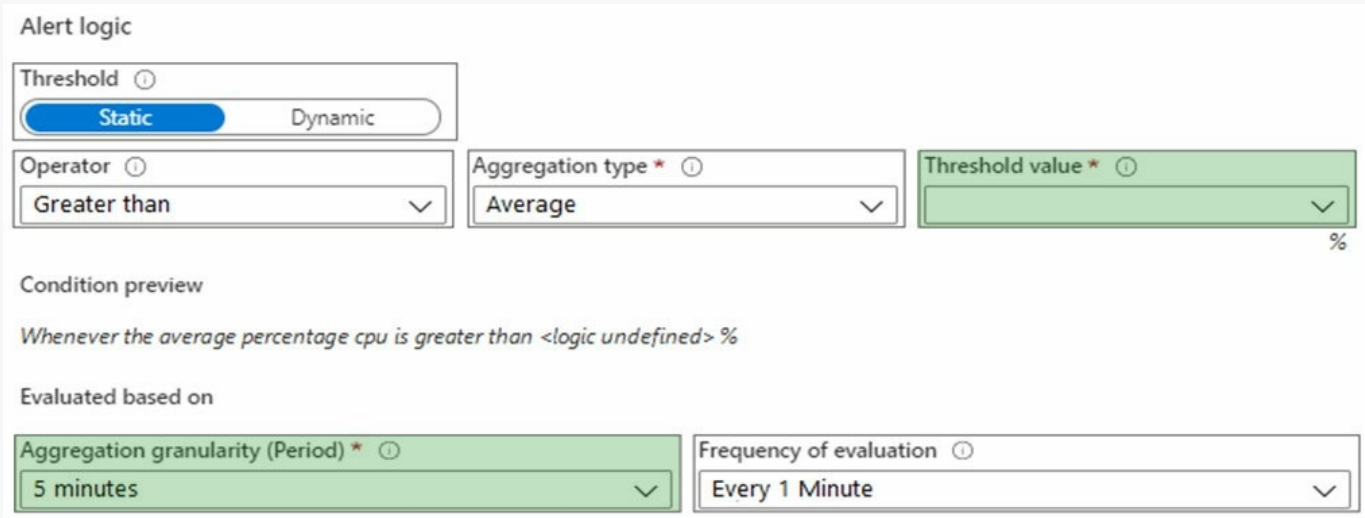
Question: 483

CertyIQ

HOTSPOT -
You need to configure the alert for VM1. The solution must meet the technical requirements.
Which two settings should you configure? To answer, select the appropriate settings in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:



Answer:



Explanation:

Setting 1: Threshold value -

Set to 80 %

Scenario: An Azure Monitor alert for VM1 must be configured to meet the following requirements:

- Be triggered when average CPU usage exceeds 80 percent for 15 minutes.
- Calculate CPU usage averages once every minute.

Setting 2: Aggregation granularity

Set to 15 minutes.

Question: 484

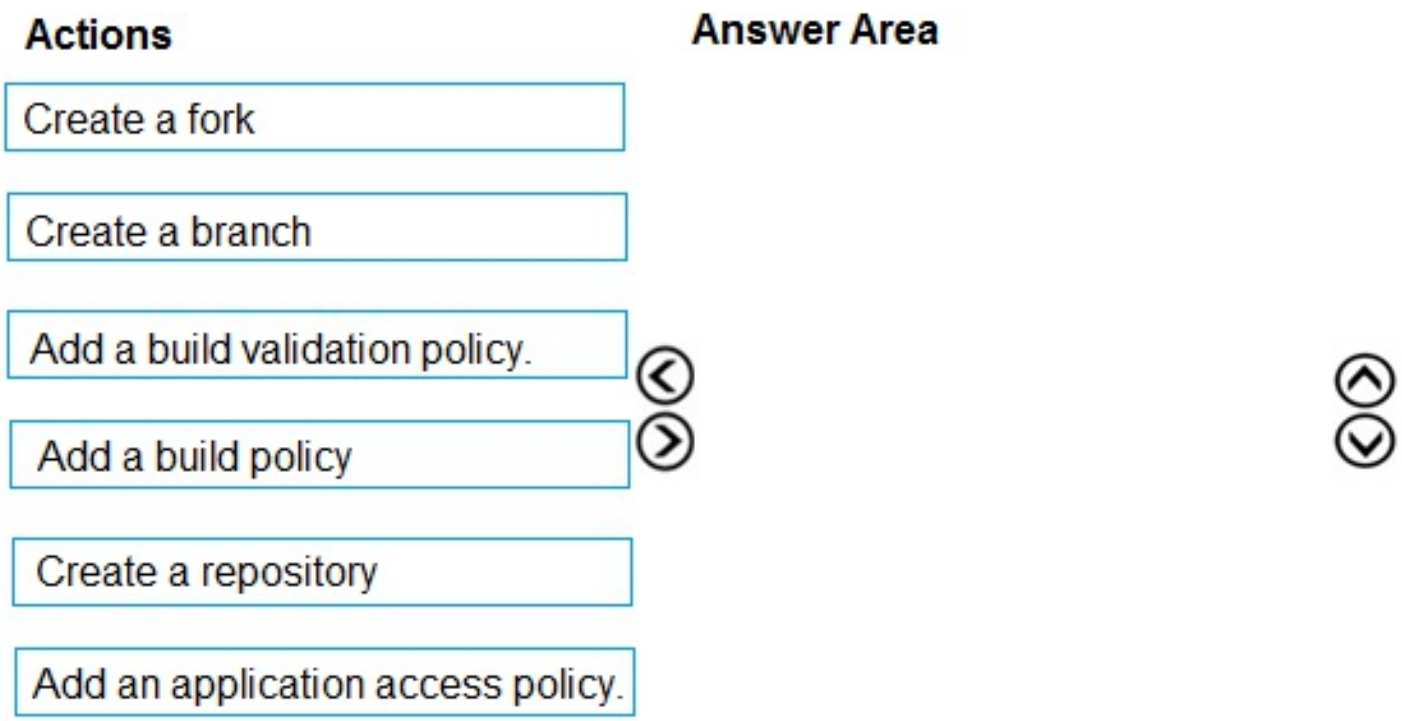
CertyIQ

DRAG DROP -

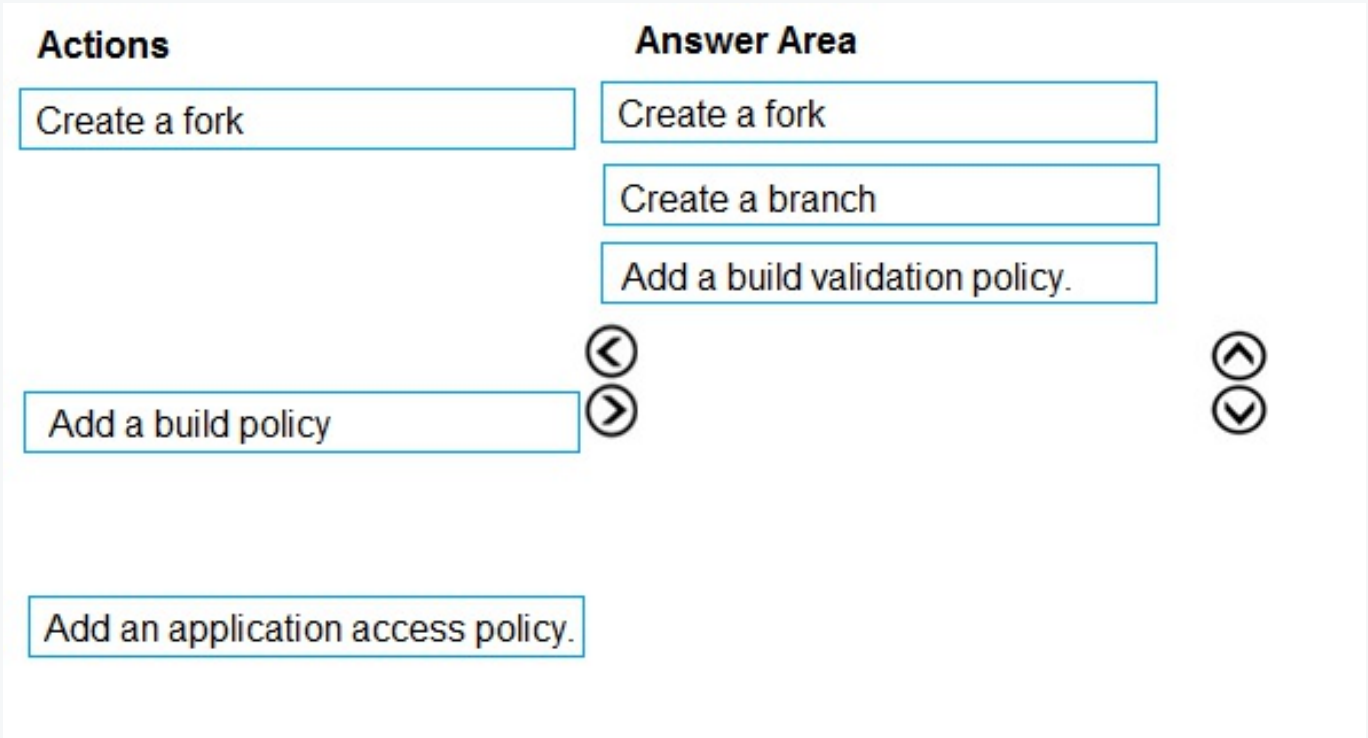
You need to implement the code flow strategy for Project2 in Azure DevOps.

Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.

Select and Place:



Answer:



Explanation:

Step 1: Create a Fork

Step 2: Create a branch -

Branch policies help teams protect their important branches of development. Policies enforce your team's code quality and change management standards.

Step 3: Add a build validation policy

When a build validation policy is enabled, a new build is queued when a new pull request is created or when changes are pushed to an existing pull request targeting this branch. The build policy then evaluates the results of the build to determine whether the pull request can be completed.

Scenario:

Implement a code flow strategy for Project2 that will:

Enable Team2 to submit pull requests for Project2.

▪

✑ Enable Team2 to work independently on changes to a copy of Project2.

✑ Ensure that any intermediary changes performed by Team2 on a copy of Project2 will be subject to the same restrictions as the ones defined in the build policy of Project2.

Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.

Question: 485 CertyIQ

You have an Azure subscription that contains the resources shown in the following table.

Name	Type
DepPipeline1	Azure DevOps deployment pipeline
ADFPipeline1	Azure Data Factory pipeline
Vault1	Azure Key Vault

DepPipeline1 and ADFPipeline1 use a single credential that is stored in Vault1.
You need to configure ADFPipeline1 to retrieve the credential from Vault1.
Which type of activity should you use?

A.Lookup
B.Get Metadata
C.Copy
D.Web

Answer: D

Explanation:

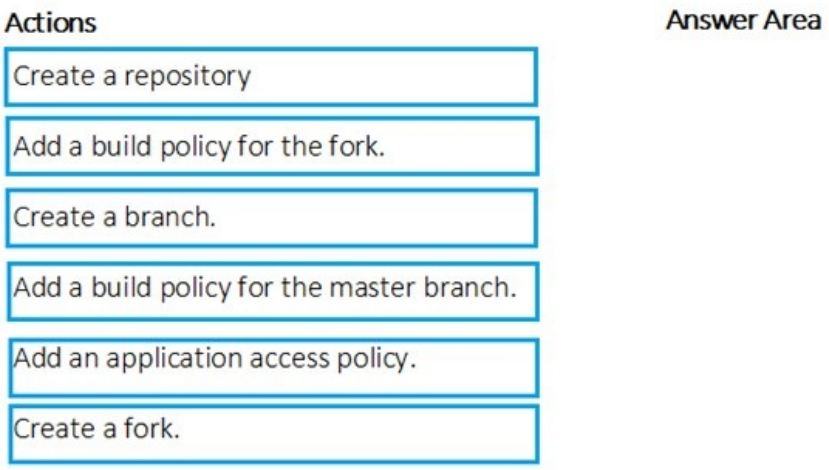
D is the answer.

https://learn.microsoft.com/en-us/azure/data-factory/how-to-use-azure-key-vault-secrets-pipeline-activities#stepsIn your Data Factory pipeline, add a new Web activity and configure it as follows.

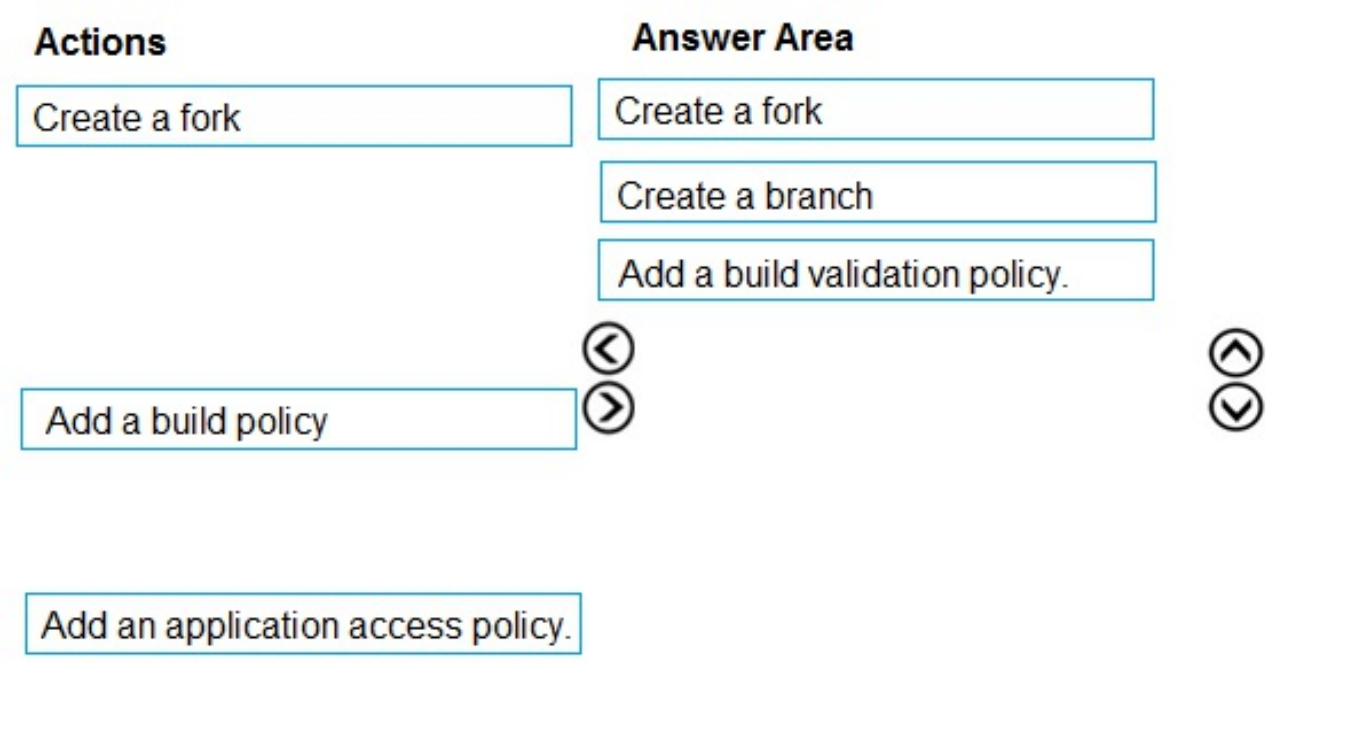
Question: 486 CertyIQ

DRAG DROP -

You need to implement the code flow strategy for Project2 in Azure DevOps.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Step 1: Create a Fork

Step 2: Create a branch -

Branch policies help teams protect their important branches of development. Policies enforce your team's code quality and change management standards.

Step 3: Add a build validation policy

When a build validation policy is enabled, a new build is queued when a new pull request is created or when changes are pushed to an existing pull request targeting this branch. The build policy then evaluates the results of the build to determine whether the pull request can be completed.

Scenario:

Implement a code flow strategy for Project2 that will:

Enable Team2 to submit pull requests for Project2.

✑ Enable Team2 to work independently on changes to a copy of Project2.

✑ Ensure that any intermediary changes performed by Team2 on a copy of Project2 will be subject to the same restrictions as the ones defined in the build policy of Project2.

Question: 487 CertyIQ

You have an Azure web app that is deployed by using Azure Pipelines.
You need to ensure that when a new version of the app is deployed to production, you can roll back to the previous version. The solution must meet the following requirements:
•Minimize downtime during the deployment.
•Minimize the time it takes for the rollback.
What should you use?

A.a single web app and two deployment slots
B.a single web app and two deployment pipelines
C.two web apps and an Azure Standard Load Balancer
D.two web apps and an Azure Traffic Manager instance

Answer: A

Explanation:

A. a single web app and two deployment slots is the correct solution.

Here's a detailed justification:

Deployment slots are a feature of Azure App Service that allows you to deploy different versions of your web app to separate environments while still being part of the same App Service plan. Using deployment slots provides several advantages for the given scenario.

First, it supports zero-downtime deployments. A new version of the app can be deployed to a staging slot, warmed up, and tested without affecting the production environment. Then, you can swap the staging slot with the production slot, effectively deploying the new version with minimal disruption.

Second, deployment slots enable simple and fast rollbacks. If any issues arise after the swap, you can quickly swap the slots back, instantly reverting to the previous version. This eliminates the need for redeploying the old version, which would take considerably more time.

Option B (two deployment pipelines) doesn't offer the easy swap functionality needed for quick rollbacks and zero-downtime deployments. Option C (two web apps and a Standard Load Balancer) and Option D (two web apps and Azure Traffic Manager) are more complex and expensive solutions. While they provide high availability, they don't offer the built-in rollback mechanism that deployment slots provide. These solutions might be more suitable if true blue/green deployment with independent scaling and resources were needed, but the requirement is to minimize downtime and rollback time, which deployment slots directly address. Deployment slots achieve this with minimal resource overhead. The load balancer/traffic manager solutions also introduce additional management overhead.

In summary, deployment slots are the most efficient and cost-effective way to achieve both zero-downtime deployments and quick rollbacks in Azure App Service, aligning perfectly with the specified requirements.

References:

Azure App Service deployment slots: https://learn.microsoft.com/en-us/azure/app-service/deploy-staging-slots
Blue/Green Deployment on Azure: https://learn.microsoft.com/en-us/azure/architecture/example-scenario/dev-ops/blue-green-deployment

Question: 488 CertyIQ

You have an app named App1 that you release by using Azure Pipelines. App1 has the versions shown in the following table.

Number	Description
3.4.7	Previous release
4.0.0	Current release

You complete a code change to fix a bug that was introduced in version 3.4.3.
Which version number should you assign to the release?

A.3.4.4
B.3.4.8
C.3.5.0
D.4.0.1

Answer: D

Explanation:

Based on the given information, the current release of App1 is version 4.0.0, and you have made a code change to fix a bug that was introduced in version 3.4.3.Following semantic versioning principles, the bug fix should be considered a patch, as it does not introduce any new features or breaking changes. Therefore, the version number for the release should be incremented in the patch position.Since the current release is 4.0.0, the next appropriate version number for the bug fix release would be 4.0.1.

Question: 489 CertyIQ

You are automating the testing process for your company.
You need to automate UI testing of a web application.
Which framework should you use?

A.JaCoco
B.Playwright
C.Xamarin.UITest
D.Microsoft.CodeAnalysis

Answer: B

Explanation:

The correct answer is **B. Playwright**. Here's why:

Playwright is a Node.js library that automates Chromium, Firefox, and WebKit with a single API. This makes it ideal for cross-browser UI testing of web applications. It supports various programming languages and provides features like auto-waiting, network interception, and tracing, which simplify test development and debugging.

While JaCoco is a code coverage tool for Java (primarily focused on backend testing) and Xamarin.UITest is used for native mobile app testing (iOS and Android), they aren't suited for web application UI testing. Microsoft.CodeAnalysis, also known as Roslyn, provides APIs for building code analysis tools and refactoring tools, and is not designed for UI testing.

Playwright's focus on web application UI automation, its cross-browser compatibility, and its rich feature set make it a strong choice for automating UI testing of web applications within an Azure DevOps environment. Its integration with CI/CD pipelines allows for automated testing as part of the software development lifecycle. It also offers features such as test parallelization, which enhances the testing process's speed and efficiency, making it highly suitable for fast-paced software development scenarios in the cloud.

Further Reading:

Playwright Documentation: https://playwright.dev/
Microsoft Azure DevOps: https://azure.microsoft.com/en-us/services/devops/

Question: 490 CertyIQ

You have a project in Azure DevOps named Project1 that references an Azure Artifacts feed named Feed1.
You have a package named Package1 that has the versions shown in the following table.

Version	Description
1.0.3	Manually pushed to Feed1
1.4.0	Manually pushed to Feed1
2.0.0	Available from an upstream source
2.3.1	Saved from an upstream source

You need to perform a build of Project1.
Which version of Package1 will be used?

A.1.0.3
B.1.4.0
C.2.0.0
D.2.3.1

Answer: D

Explanation:

Correct D. 2.3.1

When performing a build of Project1, the version of Package1 that will be used depends on the specific configuration of the build pipeline and how the package is referenced within the pipeline.

By default, the build pipeline in Azure DevOps will use the latest version of the package available in the specified feed. In this case, the versions of Package1 are listed in the following order:

1.0.3

1.4.0

2.0.0

2.3.1

If no specific version is specified in the build pipeline, it will use the latest version, which in this case is 2.3.1 (the most recent version available). Therefore, during the build of Project1, Version 2.3.1 of Package1 will be used unless a different version is explicitly specified in the build pipeline configuration or dependencies.

Question: 491 CertyIQ

HOTSPOT
-
You have an Azure subscription.
You need to create a storage account by using a Bicep file.
How should you complete the file? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

```
param storageAccount string

var storageAccountNameToUse = '${storageAccount}${uniqueString(resourceGroup().id)}'

resource invoiceStorage 'Microsoft.Storage/storageAccounts@2022-05-01' = {
  name: storageAccountNameToUse
  location: 'eastus'
sku: {
    name: 'Standard_GRS'
  }
[ ▼ ]  'StorageV2'
[ kind:
  param:
  properties:
  type:
  var: ]
[ ▼ ]  {
[ kind:
  param:
  properties:
  type:
  var: ]
  supportsHttpsTrafficOnly: true
  }
}
```

Answer:

Answer Area



Explanation:

1. kind

2. properties

https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/resource-declaration?tabs=azure-powershell#resource-specific-properties

Question: 492 CertyIQ

DRAG DROP
-
You have a GitHub repository named repo1 that stores the code of an app named App1. You need deploy a workflow for repo1 by using GitHub Actions. The solution must meet the following requirements:
•Scan on pushes to the main branch.
•Scan on pull requests to the main branch.
•Scan on pull requests to any branch that has a prefix of releases/.
•Scan all the files in the subdirectories of the src directory.
How should you complete the code? To answer, drag he appropriate values to the correct targets. Each value may be used once, more than once, or not at all.

Values

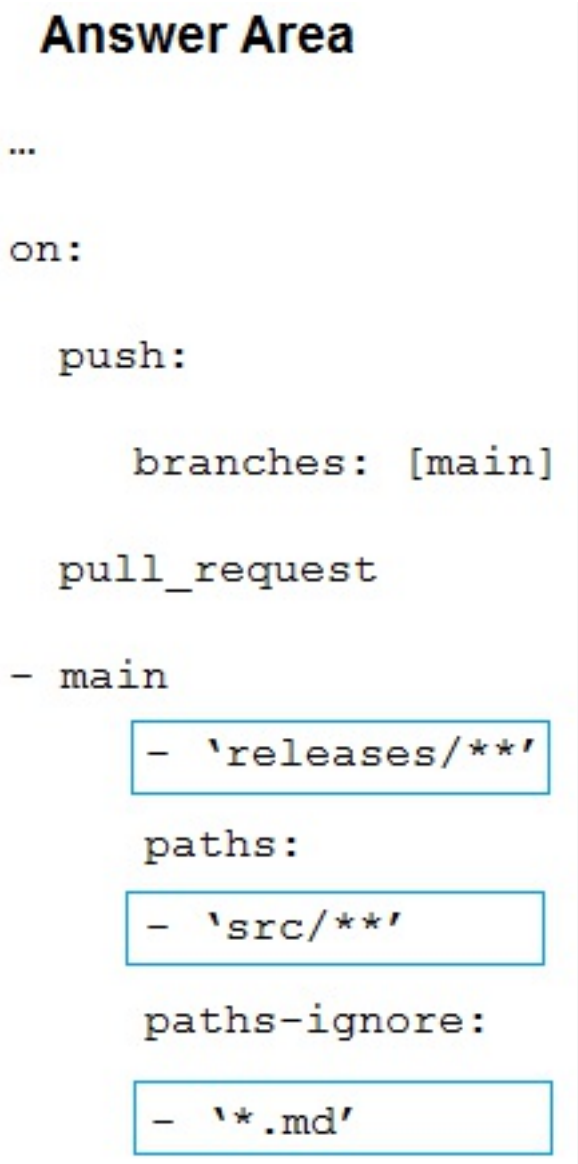
- '**/*.md'
- '*.md'
- 'release*'
- 'releases/**'
- 'src/*'
- 'src/**'

Answer Area

```
…
on:
  push:
    branches: [main]
  pull_request
  - main
      [ ]
    paths:
      [ ]
    paths-ignore:
      [ ]
  …
```

Answer:

Answer Area



Question: 493 CertyIQ

You use Azure Pipelines to build and test code projects.
You notice an increase in cycle times.
You need to identify whether agent pool exhaustion is causing the issue.
What are two possible ways to achieve this goal? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A.Query the PipelineRun/PipelineRuns endpoint.

B.Query the TaskAgentPoolSizeSnapshots endpoint.

C.View the Pipeline duration report.

D.View the pool consumption report at the organization level.

Answer: BD

Explanation:

Let's analyze why options B and D are the correct solutions for identifying agent pool exhaustion in Azure Pipelines.

Option B: Query the TaskAgentPoolSizeSnapshots endpoint.

The TaskAgentPoolSizeSnapshots endpoint directly provides historical data on the size and utilization of your agent pools. By querying this endpoint, you can examine the number of available agents over time and correlate this with the periods of increased cycle times. If the number of available agents was consistently low during those periods, it strongly suggests agent pool exhaustion. This allows for data-driven identification of potential bottlenecks.https://learn.microsoft.com/en-us/rest/api/azure/devops/distributedtask/pools/get-agent-pools?view=azure-devops-rest-7.1 - While this is the general get API for pools, the specific metrics on sizing and usage can be derived from this endpoint, particularly when polling over time.

Option D: View the pool consumption report at the organization level.

Azure DevOps provides pool consumption reports at the organization level. These reports visualize the usage of your agent pools over time. You can analyze these reports to determine if the agent pools were heavily utilized during the periods of increased cycle times. High utilization, especially if it consistently reaches the pool's capacity, indicates agent pool exhaustion is a contributing factor. These reports offer an intuitive and visual representation of pool utilization, making it easy to identify potential bottlenecks.https://learn.microsoft.com/en-us/azure/devops/report/dashboards/configure-widgets?view=azure-devops - This link describes widgets and dashboards in Azure DevOps, indirectly supporting the concept of pool consumption reports by illustrating the capabilities of the platform's reporting features.

Why A and C are incorrect:

Option A: Query the PipelineRun/PipelineRuns endpoint. While this endpoint allows you to retrieve information about pipeline runs, it doesn't directly provide insights into agent pool utilization. You'd need to correlate pipeline run times with other data sources to infer agent pool exhaustion. The correlation process adds complexity and doesn't directly pinpoint the root cause.

Option C: View the Pipeline duration report. While pipeline duration reports show how long pipelines take, they do not directly isolate agent pool exhaustion as a cause. Increased duration might be due to code changes, external dependencies, or other factors unrelated to agent availability. This is symptomatic but not diagnostic.

In summary, directly examining agent pool usage metrics via the TaskAgentPoolSizeSnapshots endpoint and the organization-level pool consumption report provides the most direct and effective way to determine if agent pool exhaustion contributes to increased cycle times.

Question: 494 CertyIQ

You have an Azure subscription that contains an Azure pipeline named Pipeline1 and a GitHub repository named Repo1. Repo1 contains Bicep modules. Pipeline1 deploys Azure resources by using the Bicep modules.
You need to ensure that all releases comply with Azure Policy before they are deployed to production.
What should you do?

A.To Pipeline1, add a step that runs a What if deployment before the deployment step.

B.Configure a deployment gate for Pipeline1 that uses Azure Automation to run a What If deployment.

C.Create an Azure DevOps build that runs on the creation of a pull request and assesses the code for compliance.

D.Configure a deployment gate for Pipeline1 and include the Azure DevOps Security and compliance assessment task.

Answer: D

Explanation:

The correct answer is D: Configure a deployment gate for Pipeline1 and include the Azure DevOps Security and compliance assessment task. Here's why:

Azure Policy enforcement prior to deployment is crucial for maintaining compliance and preventing misconfigurations in production environments. Deployment gates in Azure DevOps pipelines offer a mechanism to control when a release should proceed to the next stage based on specific criteria. They provide a checkpoint for validation, ensuring that deployments meet defined quality and security standards.

Option A, adding a "What If" deployment step directly to the pipeline, is useful but doesn't provide a built-in gate mechanism. It requires manual intervention to review the "What If" output and decide whether to proceed. This doesn't scale well for consistent policy enforcement.

Option B, using Azure Automation to run a "What If" deployment, is more automated, but it still requires custom scripting and integration. It's more complex to set up and maintain compared to using the built-in Azure DevOps Security and compliance assessment task.

Option C, creating a build pipeline for pull requests to assess code, is valuable for code quality and static analysis. However, it does not directly address the requirement of ensuring releases comply with Azure Policy specifically before deploying to production. Pull request checks are beneficial but a deployment gate serves as a final check before deployment.

Option D provides a direct solution. By configuring a deployment gate in Pipeline1, you can integrate the Azure DevOps Security and compliance assessment task. This task is specifically designed to evaluate Azure Resource Manager (ARM) templates or Bicep files against Azure Policy assignments. It automatically checks if the proposed changes comply with defined policies and prevents the deployment if there are violations. This ensures that only compliant deployments reach the production environment, addressing the core requirement.

For more information on Azure DevOps deployment gates: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops&tabs=check-pass

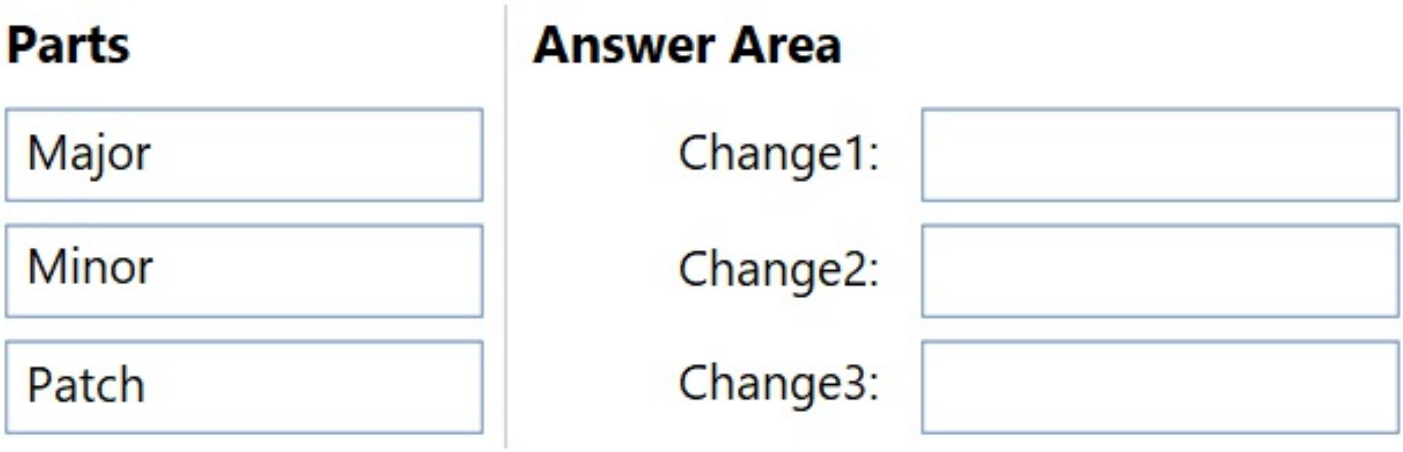
And Azure Policy compliance assessment in pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/deploy/azure-policy-as-code?view=azure-devops

Question: 495 CertyIQ

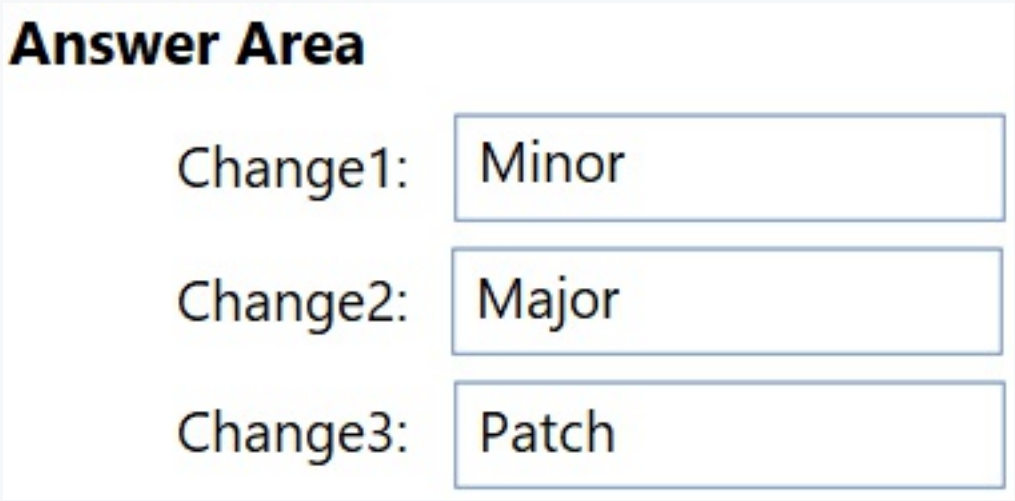
DRAG DROP
-
You use Semantic Versioning (SemVer) as a dependency versioning strategy.
You perform changes to code as shown in the following table.

Name	New functionality	Change incompatibility
Change1	Yes	Minor
Change2	Yes	Significant
Change3	No	Minor

Which part of the version should you increment for each change? To answer, drag the appropriate parts to the correct changes. Each part may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Minor changes can be either backwards compatible or not.

The first is Minor because incompatibility is "minor" (meaning definitely not major) and new features are something that can be found in Minor updates but not patches.

2nd is major because significant incompatibility can only be major.

3rd is tricky. Patches are generally seen as backwards compatible, but there are exceptions. Minor, as said, can be either or. The "No" for new features makes me think patch, because generally speaking, patches are for bug fixes and not new features. And Minor updates main focus is not bug fixes. Its extensive modifications smaller than "major" changes.

I think I would go Minor, Major, Patch, as the solution suggests

Question: 496 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure pipeline that is used to deploy a web app. The pipeline includes a test suite named TestSuite1.
TestSuite1 is used to validate the operations of the web app.
TestSuite1 fails intermittently.
You identify that the failures are unrelated to changes in the source code and execution environment.

You need to minimize troubleshooting effort for the TestSuite1 failures.
Solution: You enable flaky test management.
Does this meet the goal?

A.Yes

B.No

Answer: A

Explanation:

The solution proposes enabling flaky test management to address intermittent test failures in an Azure pipeline's test suite. This approach aligns directly with the stated goal of minimizing troubleshooting effort for these types of failures.

Flaky tests are tests that sometimes pass and sometimes fail without any apparent changes in the code or environment. They introduce uncertainty and make it difficult to determine whether a real bug exists. Addressing them becomes crucial in maintaining pipeline health and trust in test results. Flaky test management aims to identify, track, and handle these inconsistent tests automatically.

By enabling flaky test management, Azure DevOps will automatically detect and track flaky tests within TestSuite1. It can then perform actions such as automatically re-running the tests a certain number of times to verify whether it passes on subsequent retries. This helps to confirm whether a test is indeed flaky and not indicative of an actual bug.

Furthermore, reporting capabilities within flaky test management highlight which tests are flaky and the frequency of their flakiness. This information allows developers to focus on fixing the root cause of the flakiness, be it timing issues, resource contention, or other environmental factors.

Without flaky test management, troubleshooting intermittent failures would necessitate manually re-running the tests, examining logs, and attempting to isolate the cause of the failure for each instance. This manual effort can be time-consuming and inefficient. In contrast, automated flaky test management streamlines this process by identifying and highlighting these tests. Thus, it reduces troubleshooting efforts and allows focus on fixing underlying issues. Hence the statement 'You enable flaky test management' meets the goal.

Therefore, enabling flaky test management is the most effective way to meet the goal because it reduces manual intervention, provides insights into flaky tests, and helps in prioritizing fixes.

Relevant documentation:

Flaky test management in Azure DevOps: https://learn.microsoft.com/en-us/azure/devops/pipelines/test/flaky-test-management?view=azure-devops

Question: 497 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure pipeline that is used to deploy a web app. The pipeline includes a test suite named TestSuite1.
TestSuite1 is used to validate the operations of the web app.
TestSuite1 fails intermittently.
You identify that the failures are unrelated to changes in the source code and execution environment.
You need to minimize troubleshooting effort for the TestSuite1 failures.
Solution: You implement the Test Results Trend widget.

Does this meet the goal?

A.Yes
B.No

Answer: B

Explanation:

The solution proposes using the Test Results Trend widget to address intermittent test failures. This widget visualizes test results over time, showing pass/fail rates and durations. While the Test Results Trend widget helps identify trends in test results and potentially correlate failures with external factors (like deployment timings or infrastructure changes), it does not directly minimize troubleshooting efforts related to intermittent failures of a test suite like TestSuite1.

Intermittent failures, by their nature, are hard to diagnose. The Test Results Trend widget will show that the failures are happening, but it won't provide insight into the cause. Minimizing troubleshooting effort requires a more proactive approach, such as improving the test's reliability, or collecting more diagnostic data during test execution. This might involve techniques like adding retry logic to the test, improving logging, or implementing more robust error handling.

The Test Results Trend widget is more suited to identifying long-term issues or performance degradation, not sporadic, unpredictable failures. Therefore, implementing only the Test Results Trend widget is insufficient to significantly reduce troubleshooting efforts for intermittent test failures. It's a useful visualization tool after you've gathered data and made improvements, but it isn't a direct solution to the underlying problem.

Therefore, the proposed solution does not meet the goal.

Authoritative Links:

Test Results Trend widget: https://learn.microsoft.com/en-us/azure/devops/report/widgets/test-results-trend?view=azure-devops
Azure Pipelines - Test tasks: https://learn.microsoft.com/en-us/azure/devops/pipelines/test/running-tests?view=azure-devops

Question: 498 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure pipeline that is used to deploy a web app. The pipeline includes a test suite named TestSuite1.
TestSuite1 is used to validate the operations of the web app.
TestSuite1 fails intermittently.
You identify that the failures are unrelated to changes in the source code and execution environment.
You need to minimize troubleshooting effort for the TestSuite1 failures.
Solution: You enable Test Impact Analysis (TIA).
Does this meet the goal?

A.Yes
B.No

Answer: A

Explanation:

The solution proposes enabling Test Impact Analysis (TIA) to minimize troubleshooting effort for intermittent failures in TestSuite1. This does meet the goal.

Test Impact Analysis (TIA) helps identify the relationship between tests and code changes. While it doesn't directly solve intermittent failures that are unrelated to code changes or the execution environment, it indirectly helps by providing insights into which tests are impacted when there are changes. This is because sometimes, perceived intermittent failures might be caused by subtle interactions with code changes that were not initially suspected. TIA can highlight these connections. If a seemingly intermittent test always fails after a particular code change, this points to a possible cause, even if the failure appears random at first glance.

Furthermore, while TIA primarily focuses on code change impact, enabling it can still indirectly benefit troubleshooting for genuinely intermittent failures. By analyzing test execution patterns and associated code changes over time, patterns might emerge that provide clues to the root cause, even if those patterns are subtle. For example, correlating test failures with specific build configurations or even time of day could reveal underlying issues. While TIA isn't a direct solution to environment-related issues, its data could be a valuable input to a broader investigation. Finally, if the failures are truly intermittent and environmental, identifying which tests are not impacted by code changes still isolates the issue.

Therefore, by providing valuable insights into the relationships between code changes and test results, TIA can significantly streamline the troubleshooting process.

References for Test Impact Analysis:

Azure DevOps Test Impact Analysis: https://learn.microsoft.com/en-us/azure/devops/test/test-impact-analysis?view=azure-devops

Question: 499 CertyIQ

You plan to publish build artifacts by using an Azure pipeline.
You need to create an .artifactignore file that meets the following requirements:
•Includes all files in the build output folder and all subfolders
•Excludes files that have the .dll extension
What should you include in the file?

A../**
!*.dll

B.**/*
!*.dll

C.*/**
*.dll

D.**/*
#*.dll

Answer: B

Explanation:

Here's a breakdown of why option B is the correct answer and a justification for it:

The goal is to create an .artifactignore file that excludes .dll files from the build artifacts while including

everything else within the build output folder and its subfolders. Let's examine each option:

A. ../!*.dll: This option is incorrect because the double dot .. means the parent directory. You don't want to start from the parent directory, you want to operate within the artifact directory. Also, the ! is not the proper way to exclude a file.

B. /!.dll: This is the correct option. The matches any directory and subdirectories recursively. The * then matches any filename before the .dll extension. The ! excludes that the file matching the name before is a .dll. Thus, this excludes all .dll files in all folders and subfolders of the source directory.

C. //.dll: This includes only .dll files in any folder. This means it would include (not exclude) all .dll files in all folders and subfolders.

D. /#.dll: This includes files with a # character in their name before the .dll extension. It would include (not exclude) .dll files if the filename contains #.

Therefore, /!.dll correctly addresses the requirements by excluding all .dll files from the build output while including everything else within the specified scope. The ! preceding a pattern in the .artifactignore file specifies an exception to the exclusion rule.

Further Research:

Artifacts in Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/artifacts/artifacts-overview?view=azure-devops
.artifactignore file: https://learn.microsoft.com/en-us/azure/devops/pipelines/artifacts/file-matching-patterns?view=azure-devops

Question: 500 CertyIQ

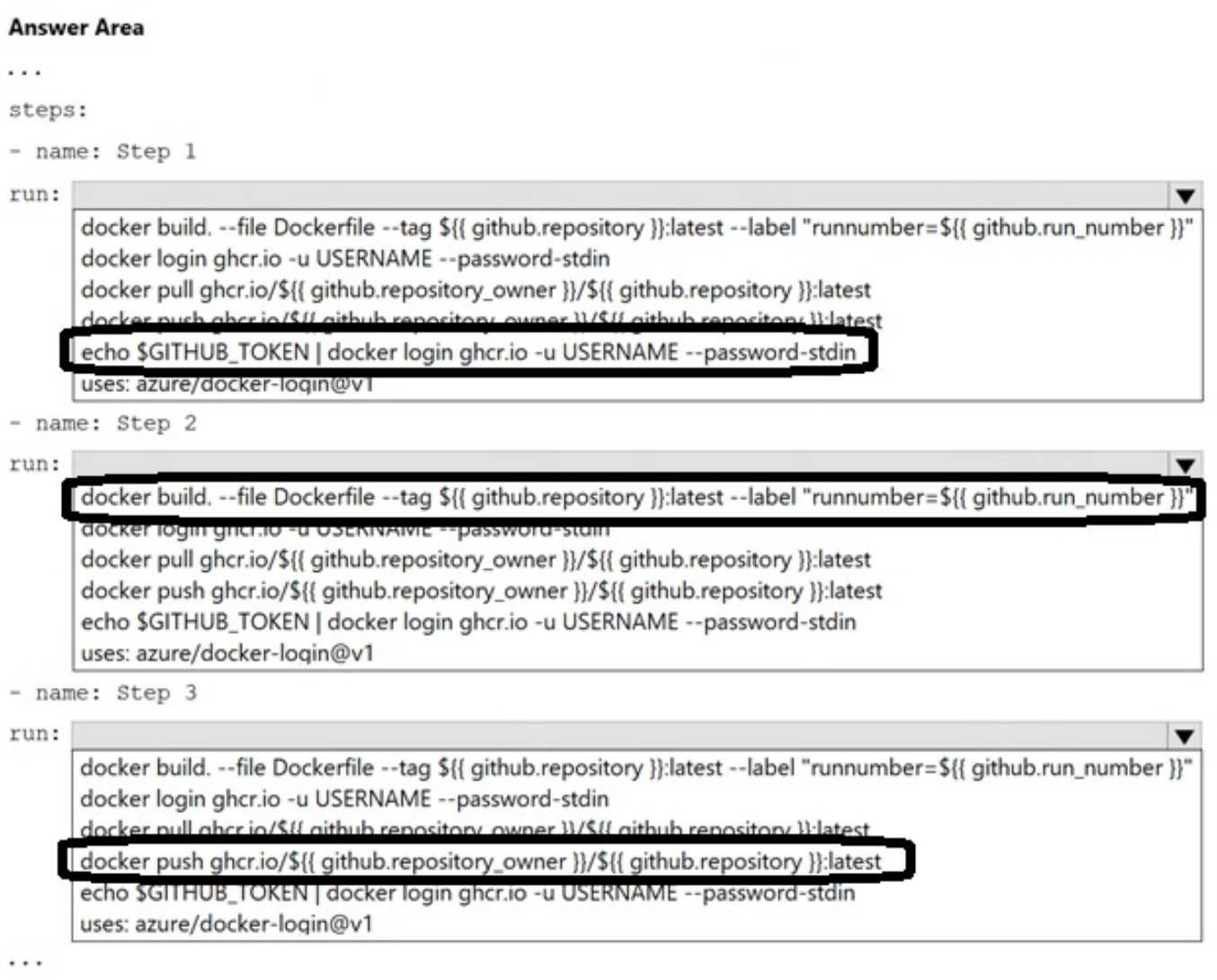
HOTSPOT
-
You have a GitHub repository.
You plan to use the repository to create a container.
You need to configure GitHub Actions to deploy the container to GitHub Container Registry.
How should you complete the script? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

```
...
steps:
- name: Step 1
run: [ docker build. --file Dockerfile --tag ${{ github.repository }}:latest --label "runnumber=${{ github.run_number }}" |
       docker login ghcr.io -u USERNAME --password-stdin |
       docker pull ghcr.io/${{ github.repository_owner }}/${{ github.repository }}:latest |
       docker push ghcr.io/${{ github.repository_owner }}/${{ github.repository }}:latest |
       echo $GITHUB_TOKEN | docker login ghcr.io -u USERNAME --password-stdin |
       uses: azure/docker-login@v1 ]
- name: Step 2
run: [ docker build. --file Dockerfile --tag ${{ github.repository }}:latest --label "runnumber=${{ github.run_number }}" |
       docker login ghcr.io -u USERNAME --password-stdin |
       docker pull ghcr.io/${{ github.repository_owner }}/${{ github.repository }}:latest |
       docker push ghcr.io/${{ github.repository_owner }}/${{ github.repository }}:latest |
       echo $GITHUB_TOKEN | docker login ghcr.io -u USERNAME --password-stdin |
       uses: azure/docker-login@v1 ]
- name: Step 3
run: [ docker build. --file Dockerfile --tag ${{ github.repository }}:latest --label "runnumber=${{ github.run_number }}" |
       docker login ghcr.io -u USERNAME --password-stdin |
       docker pull ghcr.io/${{ github.repository_owner }}/${{ github.repository }}:latest |
       docker push ghcr.io/${{ github.repository_owner }}/${{ github.repository }}:latest |
       echo $GITHUB_TOKEN | docker login ghcr.io -u USERNAME --password-stdin |
       uses: azure/docker-login@v1 ]
...
```

Answer:

Answer Area



Explanation:

The link below mentions the login method having that "echo GITHUB_TOKEN" language. GITHUB_TOKEN is a installation access token, kind of like a PAT. It even recommends to use GITHUB_TOKEN in lieu of a PAT for GitHub Actions.

The following section talks about pushing the container, using syntax similar to the 3rd given answer. The article is about pushing containers, so its assuming the container is built already. But of the remaining options, 2 are other login methods and 1 is a pull, which we dont need to do when deploying. So build must be the other option used, and it goes before the push. Usually it goes docker login, docker build, docker push.

https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-container-registry

Question: 501 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You have an Azure pipeline that is used to deploy a web app. The pipeline includes a test suite named TestSuite1.
TestSuite1 is used to validate the operations of the web app.
TestSuite1 fails intermittently.
You identify that the failures are unrelated to changes in the source code and execution environment.
You need to minimize troubleshooting effort for the TestSuite1 failures.

Solution: You increase code coverage.
Does this meet the goal?

A.Yes
B.No

Answer: B

Explanation:

The proposed solution of increasing code coverage does not directly address the problem of intermittent test failures in TestSuite1. Intermittent failures, also known as flaky tests, are characterized by passing and failing unpredictably despite no changes in the code or environment. Increasing code coverage focuses on ensuring that a larger proportion of the codebase is exercised by tests. While higher code coverage is generally desirable for software quality, it won't inherently resolve the underlying cause of intermittent test behavior.

Intermittent failures often stem from issues like race conditions, timing dependencies, external service unavailability, resource contention, or incorrect test setup/teardown. Resolving these requires identifying the root cause of the flakiness, which may involve examining logs, analyzing test execution patterns, improving test isolation, or introducing retry mechanisms. Simply writing more tests (increasing coverage) does not eliminate these sources of unpredictability. Therefore, increasing code coverage is unlikely to minimize troubleshooting effort for the specific type of intermittent failures described.

Instead, the debugging efforts should focus on identifying the causes of the intermittent test failures.

Relevant resources:

Flaky Tests: https://testing.googleblog.com/2016/05/flaky-tests-at-google-and-how-we.html
Dealing with Flaky Tests: https://martinfowler.com/articles/nonDeterminism.html

Question: 502 CertyIQ

You have an Azure pipeline that is used to build and deploy an app named App1. The build job uses a Microsoft-hosted Windows agent.
The build job for App1 intermittently returns a timeout error.
You need to ensure that the build job completes successfully. The solution must minimize administrative effort.
What should you do?

A.Change the configuration of the build agent.
B.Deploy a self-hosted agent.
C.Change to a Microsoft-hosted Linux agent.
D.Purchase more parallel jobs.

Answer: D

Explanation:

The correct answer is D: Purchase more parallel jobs. Here's a detailed justification:

The problem states that the build job for App1 intermittently returns a timeout error when using a Microsoft-hosted Windows agent. Timeout errors in Azure Pipelines often occur when all available agents in the Microsoft-hosted pool are busy, and your job has to wait longer than the timeout period to be assigned an

agent. Microsoft-hosted agents operate in a multi-tenant environment, and the availability of agents varies depending on demand.

Purchasing more parallel jobs directly addresses this issue. Parallel jobs define how many jobs can run concurrently in your Azure DevOps organization. When you have more parallel jobs, more agents can be provisioned simultaneously, reducing the queueing time and the likelihood of timeout errors. The build job is less likely to be delayed due to agent unavailability if there are more parallel jobs available to service the build request.

Option A, changing the configuration of the build agent, is not relevant to resolving timeout issues caused by agent availability. Microsoft-hosted agents are managed by Microsoft, and direct configuration changes are not possible.

Option B, deploying a self-hosted agent, would address the availability issue by providing dedicated resources. However, it requires significantly more administrative effort than purchasing parallel jobs. You would need to provision, manage, and maintain the infrastructure for the self-hosted agent, including installing necessary software and ensuring its availability. This is contrary to the requirement of minimizing administrative effort.

Option C, changing to a Microsoft-hosted Linux agent, might be helpful if the build process is more optimized for Linux. However, the problem doesn't explicitly indicate that the OS is the root cause of the timeout. If the build processes perform comparably on either operating system, this is unlikely to resolve the timeout issue caused by concurrency. Furthermore, if the application is primarily developed for Windows, this change could lead to compatibility issues.

Therefore, increasing the number of parallel jobs is the most direct, efficient, and administratively simple solution to reduce timeout errors caused by agent unavailability in a Microsoft-hosted agent pool.

Relevant resources:

Parallel jobs in Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops&tabs=ms-hosted
Microsoft-hosted agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/hosted?view=azure-devops&tabs=yaml
Self-hosted agents: https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops

Question: 503 CertyIQ

You have an Azure subscription that contains an Azure Pipelines pipeline named Pipeline1 and a user named User1. Pipeline1 is used to build and test an app named App1. User1 is assigned the Contributors role for Pipeline1.
You plan to test App1 by using an Azure Deployment Environments environment.
You need to ensure that User1 can provision the environment. The solution must follow the principle of least privilege.
Which role should you assign to User1?

A.DevCenter Project Admin
B.Deployment Environments User
C.Contributors
D.Build Administrators

Answer: B

Explanation:

The correct answer is B, Deployment Environments User. Here's why:

Azure Deployment Environments (ADE) necessitates specific permissions for users to interact with environments. The goal is to grant User1 the ability to provision the ADE environment with the least privilege possible. The Contributors role (C) grants broader permissions than necessary for just provisioning ADE environments within the pipeline; it allows User1 to modify the pipeline itself, which isn't the requirement. The Build Administrators role (D) relates to build-related activities within the pipeline and doesn't directly pertain to ADE environments. DevCenter Project Admin (A) provides administrative control over the entire Dev Center project, which is an even greater scope than provisioning individual environments.

The Deployment Environments User role is explicitly designed to provide the necessary permissions to create, manage, and delete environments within Azure Deployment Environments. This role aligns perfectly with the stated requirement of enabling User1 to provision the environment while adhering to the principle of least privilege. By granting this role, User1 gains the ability to interact with the ADE environment without inheriting unnecessary permissions that could potentially lead to unintended modifications elsewhere in the pipeline or Azure subscription. It scopes down the permissions to precisely what the user needs for ADE interaction, nothing more, nothing less.

Therefore, assigning the Deployment Environments User role to User1 is the most appropriate solution.

For more information on Azure Deployment Environments and the associated roles, you can refer to the official Microsoft documentation:

Azure Deployment Environments documentation
Understand roles, licenses, and permissions (This documentation helps understand Azure DevOps roles in general)

Question: 504 CertyIQ

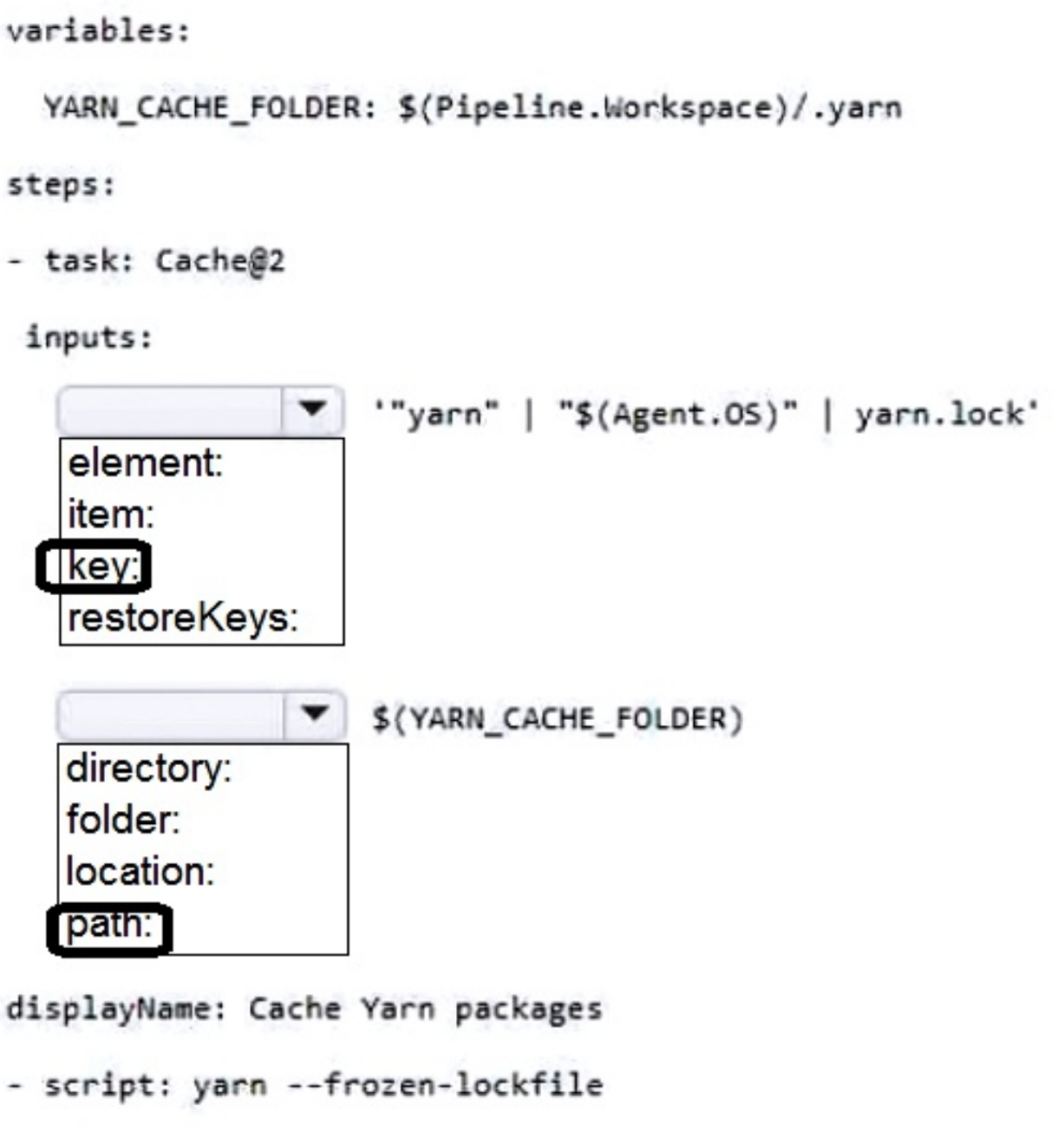
HOTSPOT
-
You have an Azure subscription that contains Azure DevOps build pipelines.
You need to implement pipeline caching by using the cache task.
How should you complete the YAML definition? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

```
variables:
  YARN_CACHE_FOLDER: $(Pipeline.Workspace)/.yarn
steps:
- task: Cache@2
 inputs:
    [ element: / item: / key: / restoreKeys: ]  '"yarn" | "$(Agent.OS)" | yarn.lock'
    [ directory: / folder: / location: / path: ]  $(YARN_CACHE_FOLDER)
displayName: Cache Yarn packages
- script: yarn --frozen-lockfile
```

Answer:

Answer Area



Explanation:

Key:

path:

Question: 505 CertyIQ

HOTSPOT
-
You have a management group that contains four Azure subscriptions. Each subscription contains four resource groups.
You develop a new web app named App1.
You plan to deploy an instance of App1 to each resource group.
You need to create a Bicep template that will be used to deploy App1. The solution must meet the following requirements:
•The name of each App1 instance must be consistent for each subscription and resource group.

•The name of each App Service plan used to host App1 must be consistent.
How should you complete the template? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

```
...
param location string = resourceGroup().location
param planname string = toLower('AppServicePlan-${uniqueString
param appname string = toLower('app-${uniqueString
resource webappplan 'Microsoft.Web/serverfarms@2022-09-01' = {
  name: planname
  location: location
  sku: {
    name: 'B1'
    tier: 'Basic'
  }
```

(deployment().name)')
(resourceGroup().id)')
('resourceGroupName')')
(subscription().id)')

(deployment().name)')
(resourceGroup().id)')
('resourceGroupName')')
(subscription().id)')

Answer:

Answer Area

```
...
param location string = resourceGroup().location
param planname string = toLower('AppServicePlan-${uniqueString
param appname string = toLower('app-${uniqueString
resource webappplan 'Microsoft.Web/serverfarms@2022-09-01' = {
  name: planname
  location: location
  sku: {
    name: 'B1'
    tier: 'Basic'
  }
```

(deployment().name)')
(resourceGroup().id)')
('resourceGroupName')')
(subscription().id)')

(deployment().name)')
(resourceGroup().id)')
('resourceGroupName')')
(subscription().id)')

Question: 506

CertyIQ

DRAG DROP
-
You use an Azure pipeline to build a .NET app that has NuGet dependencies.
You need to ensure that the pipeline caches required NuGet packages.
How should you configure the pipeline? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Values

- $(Agent.Id)
- $(Agent.OS)
- $(Build.ArtifactStagingDirectory)
- $(Build.SourcesDirectory)/**/packages.lock.json'
- $(NUGET_PACKAGES)
- cache/packages

Answer Area

```
...
variables:
  NUGET_PACKAGES: $(Pipeline.Workspace)/.nuget/packages
steps:
- task: Cache@2
  inputs:
    key: 'nuget | "$(Agent.OS)" |  Value
    restoreKeys: |
      nuget |  Value
      nuget
    path:  Value
  displayName: Cache NuGet packages
...
```

Answer:

Answer Area

```
...
variables:
  NUGET_PACKAGES: $(Pipeline.Workspace)/.nuget/packages
steps:
- task: Cache@2
  inputs:
    key: 'nuget | "$(Agent.OS)" | $(Build.SourcesDirectory)/**/packages.lock.json'
    restoreKeys: |
      nuget | $(Agent.OS)
      nuget
    path: $(NUGET_PACKAGES)
  displayName: Cache NuGet packages
...
```

Explanation:

Reference:

https://learn.microsoft.com/en-us/azure/devops/pipelines/release/caching?view=azure-devops#netnuget

Question: 507 CertyIQ

You have an Azure pipeline that is used to deploy an app named App1.
You need to ensure that new versions of App1 are released only if they exceed performance baselines. The solution must minimize administrative effort.
What should you configure?

A.an Azure Pipelines release artifact
B.an Azure Repos branch policy
C.an Azure Monitor alert
D.an Azure Pipelines deployment gate

Answer: C

Explanation:

The correct answer is **C. an Azure Monitor alert.** Here's why:

The goal is to automatically prevent App1 deployments if performance baselines are not met, minimizing administrative effort. Azure Monitor alerts directly address this requirement.

Azure Monitor: Azure Monitor is a comprehensive monitoring solution within Azure that collects, analyzes, and acts on telemetry from various sources including applications, infrastructure, and the Azure platform itself. It allows you to define alerts based on metrics (like CPU utilization, response time, memory usage) and logs.

How it works in this scenario: You would configure an Azure Monitor alert to trigger when performance metrics (e.g., average response time) exceed predefined thresholds (the baselines). This alert can then be configured to integrate with Azure DevOps pipelines via webhooks or Azure Functions.

Deployment Gates: Azure DevOps deployment gates allow you to integrate external health checks and

approvals into your release pipelines. You can configure a gate to invoke an Azure Function, which checks the status of the Azure Monitor alert. If the alert is active (meaning performance baselines are violated), the function fails, and the gate blocks the deployment. If the alert is not active, the deployment proceeds.

Minimizing Administrative Effort: Once the alert and gate are configured, the process is largely automated. Azure Monitor continuously monitors performance, and the gate automatically blocks deployments when necessary, reducing manual intervention and the risk of deploying subpar versions.

Why the other options are less suitable:

A. Azure Pipelines release artifact: Release artifacts are simply the output of a build process. They don't inherently provide performance validation or automated gatekeeping based on performance metrics.

B. Azure Repos branch policy: Branch policies enforce code quality and workflow rules on branches in your Azure Repos repositories. While useful for code quality, they don't address runtime performance monitoring and automated deployment blocking.

D. Azure Pipelines deployment gate: While gates are part of the solution, the key component that monitors the performance and triggers the action is Azure Monitor alert. The deployment gate will act upon the monitor alert result, if the gate is based on another component (such as Azure Functions) that query the Azure Monitor alert result.

Authoritative Links:

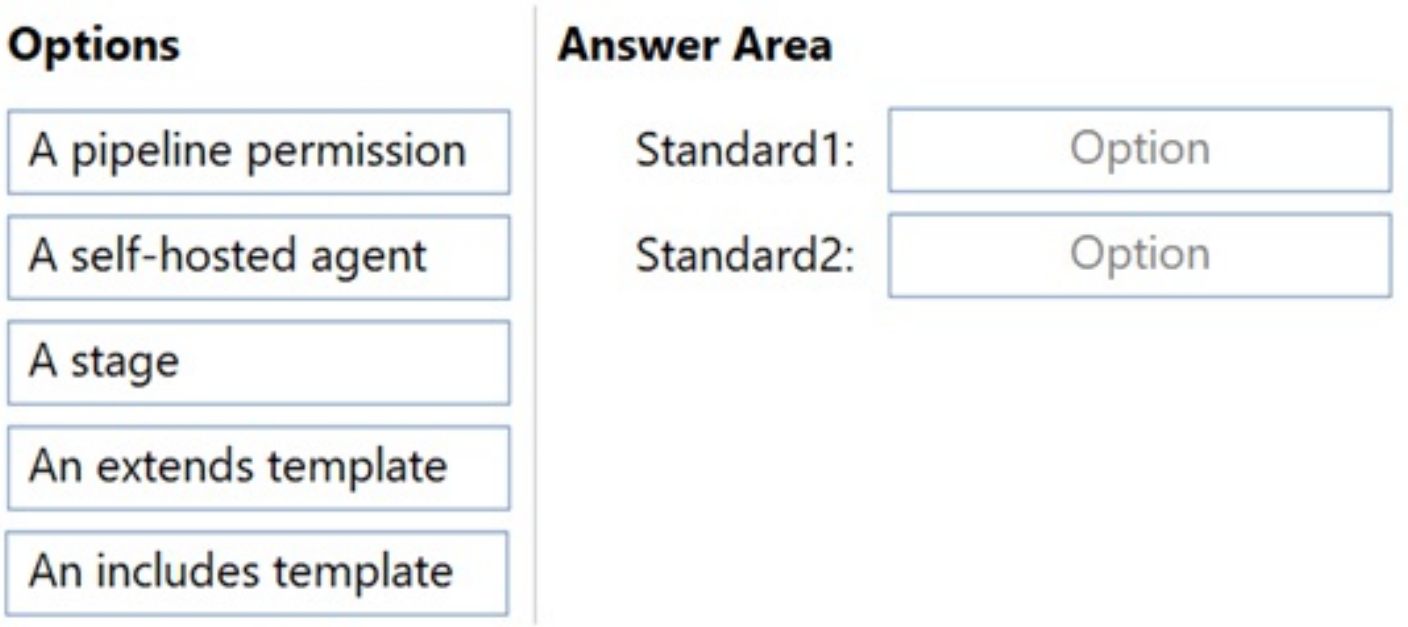
Azure Monitor: https://learn.microsoft.com/en-us/azure/azure-monitor/overview
Azure DevOps Deployment Gates: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/approvals/gates?view=azure-devops

Question: 508 CertyIQ

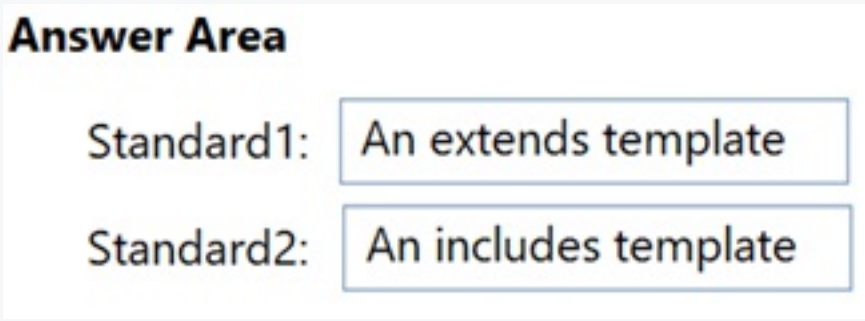
DRAG DROP
-
You have an Azure subscription.
You have the standards shown in the following table.

Name	Contents
Standard1	Security requirements
Standard2	Software prerequisites

You plan to use Azure Pipelines to build and release web apps.
You need to recommend a solution to build the pipelines. The solution must meet the following requirements:
•Ensure that all new pipelines meet the security requirements defined in Standard1.
•Ensure that the first stage of all new pipelines contains the software prerequisites defined in Standard2.
•Minimize administrative effort.
What should you use to implement each standard? To answer, drag the appropriate options to the correct standards. Each component may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.



Answer:



Question: 509 CertyIQ

You have an Azure subscription.
You create two Bicep templates named Template1 and Template2 that will be used to create a virtual machine and a website.
You need to create a template named Template3 that will reuse logic from Template1 and Template2.
What should you define first?

A.outputs
B.resources
C.modules
D.parameters

Answer: C

Explanation:

The correct answer is **C. modules**.

To reuse logic from Template1 and Template2 within Template3, you need to define them as reusable components first. In Bicep, these reusable components are called modules. Modules allow you to encapsulate and abstract away the complex logic within Template1 and Template2 making it easy to incorporate them into Template3 without duplicating the code.

Modules act as self-contained units of deployment that can be invoked and configured within other Bicep templates. Before you can use Template1 and Template2 in Template3, you must define them as modules so the Bicep compiler knows how to interpret and integrate their resources and configurations.

Parameters allow you to customize the module's behavior, outputs expose values from the module, and resources are the actual infrastructure elements defined within the modules themselves. You declare and use parameters and outputs within the modules themselves (Template1 and Template2) and use them to interface with Template3 after the modules have been defined. However, before you can leverage any elements of the referenced templates, they need to be declared as modules. Consequently, module definition comes logically before defining parameters, outputs, or resources in the context of Template3 calling Template1 and Template2. Defining resources, parameters, or outputs of Template3 is premature without first establishing how Template1 and Template2 will be incorporated. Therefore, the first step is to define Template1 and Template2 as modules within Template3.

Relevant resources:

Bicep Modules: https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/modules
Bicep Documentation: https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/

Question: 510 CertyIQ

You have an Azure subscription.
You use Bicep templates to deploy websites and Azure SQL infrastructure.
You need to automate the deployments by using Azure Pipelines and a self-hosted agent that runs on two virtual machines. The solution must minimize administrative effort.
What should you do first?

A.Create a service principal.
B.Create an Azure Automation account.
C.Create a user-assigned managed identity.
D.On each virtual machine, enable a system-assigned managed identity.

Answer: A

Explanation:

Here's a detailed justification for why creating a service principal is the correct first step in this scenario:

The primary goal is to automate deployments using Azure Pipelines with a self-hosted agent. The agent needs authorization to interact with Azure resources (websites and SQL infrastructure) and perform deployments defined in the Bicep templates. The principle of least privilege dictates we grant only necessary permissions.

A service principal is a security identity (similar to a user account) specifically designed for applications, services, and automation tools to access Azure resources. It allows you to grant specific, granular permissions to the agent, restricting it to only the necessary actions for deployment, such as creating resources, updating configurations, and managing deployments.

Option B (Azure Automation Account) is incorrect. While Azure Automation can run scripts and automate tasks, it's not directly related to providing authentication for the self-hosted agent in Azure Pipelines.

Option C (User-assigned managed identity) and D (System-assigned managed identity) are generally used for VMs and other Azure resources to authenticate to other Azure services. While they could be used, they are more complex to set up and manage in the context of a self-hosted agent than a service principal. You'd have

to configure the agent to assume the identity, manage the identity's permissions, and potentially deal with issues related to agent configuration. A service principal provides a more direct and widely supported method for securing pipeline access.

Specifically, using a managed identity with a self-hosted agent would require the agent to have credentials to be able to assume the identity, which negates some of the security benefits. It also ties the agent's identity to the virtual machines it's running on, which isn't necessary and could create management overhead.

A service principal is the best practice because:

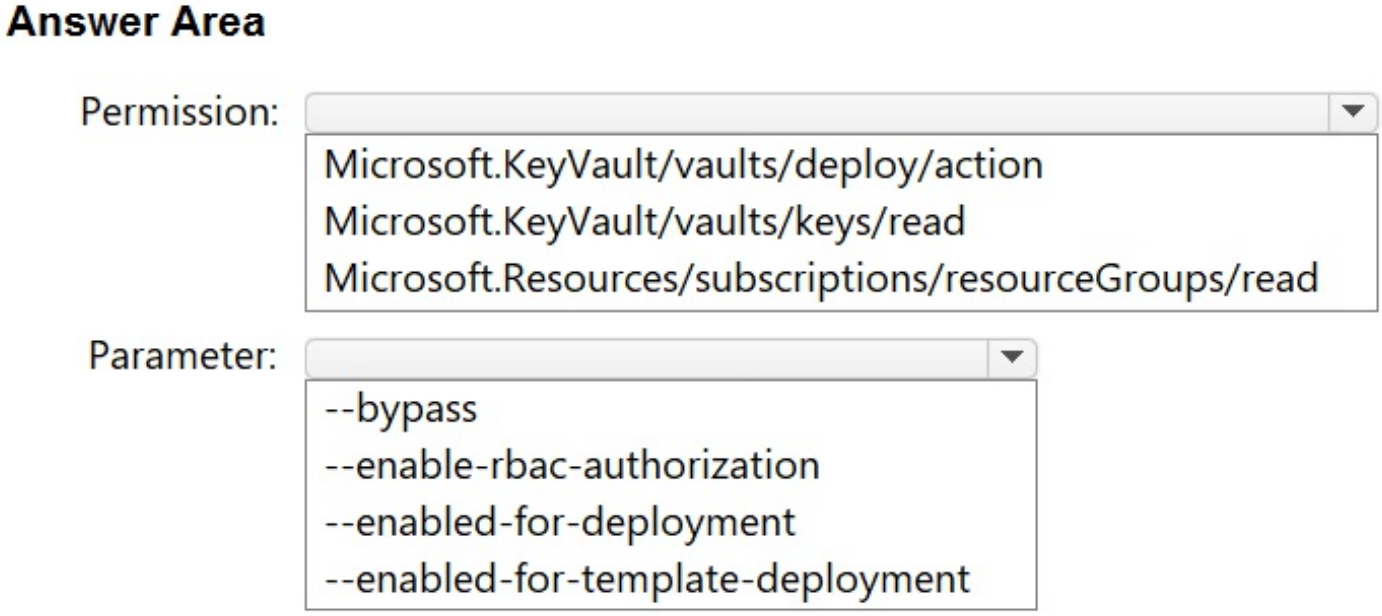
1. **Specific Permissions:** You can assign the necessary roles (e.g., Contributor, Resource Group Contributor) to the service principal, granting it only the permissions needed for deployments.
2. **Centralized Management:** Service principals are managed centrally in Azure Active Directory (Azure AD), simplifying access control.
3. **Credentials Management:** The service principal's client secret (or certificate) is stored securely in Azure Pipelines as a service connection, avoiding hardcoding credentials in the Bicep templates or pipeline definitions.
4. **Auditing:** Service principal usage is auditable within Azure AD, making it easier to track who or what is making changes to Azure resources.

Here are some helpful links:

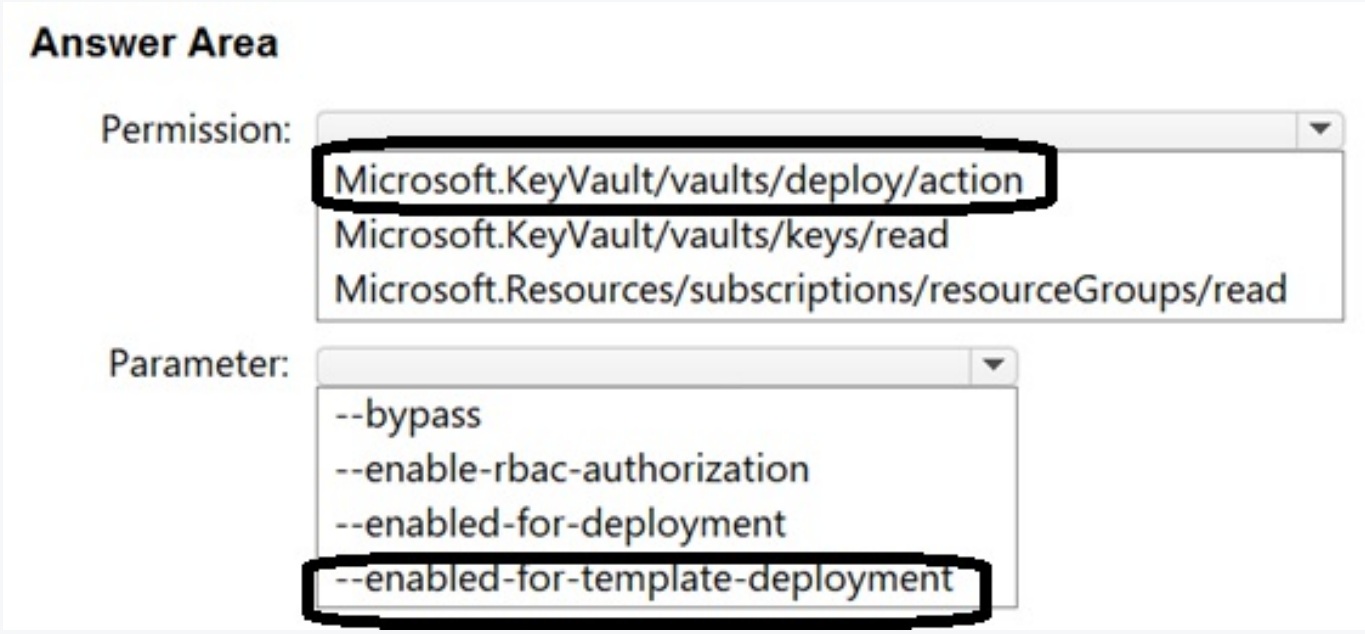
Service Principals: https://learn.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals
Azure Pipelines Service Connections: https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml
Managed Identities: https://learn.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/overview

Question: 511 CertyIQ

HOTSPOT
-
You have an Azure subscription that contains a user named User1.
You have an Azure Resource Manager (ARM) template named Template1.
You plan to perform the following actions:
•Deploy an Azure key vault named KV1.
•Deploy Azure resources by using Template1 to retrieve secrets from KV1.
You need to ensure that User1 can deploy Template1. The solution must follow the principle of least privilege.
Which permission should you grant to User1, and which parameter should be specified when you create KV1? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.



Answer:



Explanation:

https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/key-vault-parameter?tabs=azure-cli#deploy-key-vaults-and-secrets- https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/key-vault-parameter?tabs=azure-cli#grant-deployment-access-to-the-secrets

Question: 512 CertyIQ

DRAG DROP

-

You have an Azure subscription that contains an Azure Traffic Manager profile named ATM1 and a web app named App1. ATM1 manages App1 traffic. ATM1 is configured to route callers to the endpoint that has the lowest latency.
You need to configure ATM1 to route all traffic from Asia to an endpoint in Australia.
Which three actions should you perform in sequence from the Azure portal? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.

Actions

- Add new endpoints that have geographic region mapping.
- Decrease the TTL value of the DNS records for App1.
- Configure performance routing.
- Delete all the existing endpoints.
- Configure geographic routing.

Answer Area

Answer:

Answer Area

- Delete all the existing endpoints.
- Configure geographic routing.
- Add new endpoints that have geographic region mapping.

Question: 513

CertyIQ

You plan to deploy a solution that will include multiple microservices.
You need to recommend a deployment strategy for the microservices. The solution must meet the following requirements:
•Enable users to test new features by using a specific URL.
•Minimize the effort required to promote a test version to production.
•Minimize the effort required to revert production code to the previous version.
Which strategy should you recommend?

A.A/B
B.feature toggle
C.progressive exposure
D.blue/green

Answer: D

Explanation:

The correct answer is D: Blue/Green deployment. Here's a detailed justification:

Blue/Green deployment is a strategy that involves maintaining two identical production environments, "Blue" and "Green." At any given time, only one environment is live and serving production traffic. In this scenario, you deploy the new version of your microservices to the inactive environment (e.g., Green).

Testing new features with a specific URL: Before switching, the "Green" environment can be exposed via a specific URL (e.g., using a load balancer configuration or DNS entry) for thorough testing and verification of the new features. Users or QA teams can access this environment without affecting production users accessing the "Blue" environment.
Minimize effort to promote to production: Promotion is extremely simple. You switch the traffic to the "Green" environment by changing the load balancer configuration or DNS records. This switch is very quick

and minimally disruptive.
Minimize effort to revert: Reverting is just as easy. If any issues arise after switching to the "Green" environment, you simply switch the traffic back to the "Blue" environment, which is still running the previous stable version. This minimizes downtime and impact on users.

A/B testing (Option A) is primarily used for comparing two versions of a feature to determine which performs better, not for deploying a full microservice release.

Feature toggles (Option B) involve embedding conditional logic within the code to enable or disable certain features. While useful for controlled releases, they add complexity to the codebase and are not ideal for deploying entire microservices. They don't provide a separate environment for thorough pre-production testing.

Progressive exposure (Option C), such as Canary deployments, gradually releases the new version to a subset of users. While good for managing risk, it does not inherently provide a dedicated environment for testing new features with a specific URL prior to any production release.

Blue/Green deployment provides the best solution for the specified requirements because it allows for comprehensive pre-production testing, effortless promotion, and quick rollback capabilities. It leverages separate environments to mitigate risk and minimize impact on users.

Authoritative Links:

Microsoft Azure documentation on Deployment Strategies: https://learn.microsoft.com/en-us/azure/architecture/patterns/blue-green-deployment
Martin Fowler on Blue Green Deployment: https://martinfowler.com/bliki/BlueGreenDeployment.html

Question: 514 CertyIQ

You manage a project by using Azure Boards. You manage the project code by using GitHub.
You have three work items that have IDs of 456, 457, and 458.
You need to create a pull request that will be linked to all the work items. The solution must set the state of work item 456 to done.
What should you add to the commit message?

A.Done #456, #457, #458

B.Fixes #456, #457, #458

C.Fixes #AB456, #A8457, #A8458

D.#AB456, #AB457, #AB458
Completed #AB456

Answer: C

Explanation:

The correct answer is C: Fixes #AB456, #A8457, #A8458. Here's why:

GitHub Integration with Azure Boards: When integrating GitHub with Azure Boards, you can link pull requests and commits to work items by mentioning the work item IDs in the commit message or pull request description. This allows for traceability between code changes and the tasks or bug fixes they address.

Fixes Keyword: The Fixes (or Fixes variations like Fix, Fixed, or Closes) keyword, when included in a commit message or pull request description, will automatically transition the linked work item to a resolved or closed state upon merging the pull request. In this specific scenario, we are required to set the state of work item

456 to done, which can be triggered automatically when the pull request is merged.

Work Item ID Format: Azure Boards work item IDs are integers. The # symbol is used to prefix the work item ID in the commit message to signal the link. In the given answer, the work item IDs are prefixed by #A. From the description of the problem, it is not implied that these work items belong to a different project than the current one. Thus, options using #AB or #A8 are unnecessary. It's unclear why '#AB' or '#A8' are in front of the work item IDs.

Linking Multiple Work Items: You can link multiple work items to a single commit or pull request by including multiple mentions of #workitemid in the commit message, separated by commas, spaces, or newlines.

Other Options Analysis:

A: Done #456, #457, #458 - While this mentions the work items, the Done keyword is not a recognized trigger for automatically transitioning the work item state in Azure Boards.
B: Fixes #456, #457, #458 - This would correctly link the work items and close them upon merge. However, it uses raw IDs, which might not be the appropriate format given the context and the structure of the available options.
D: #AB456, #AB457, #AB458Completed #AB456 - This fails to use the 'Fixes' keyword to automatically close the work item and the keyword 'Completed' is not a recognised one. Furthermore, the duplicated reference to work item AB456 is redundant.

Authoritative Links:

Azure Boards GitHub Integration: https://learn.microsoft.com/en-us/azure/devops/boards/github/?view=azure-devops
Link from GitHub: https://learn.microsoft.com/en-us/azure/devops/boards/github/link-to-from-github?view=azure-devops&tabs=new-nav

Question: 515

CertyIQ

You plan to deploy a solution that will include multiple microservices.
You need to recommend a deployment strategy for the microservices. The solution must meet the following requirements:
•Enable testing and monitoring of changes during a gradual rollout.
•Control the number of users that will receive new code releases.
Which strategy should you recommend?

A.progressive exposure
B.A/B
C.feature toggle
D.blue/green

Answer: A

Explanation:

Here's a detailed justification for why Progressive Exposure is the correct deployment strategy in this scenario:

The core requirement is to gradually roll out changes to microservices, enabling testing and monitoring throughout the process, and controlling the number of users exposed to the new code. Progressive exposure is specifically designed for this phased release pattern. This strategy involves gradually exposing a new

version of a microservice to an increasing subset of users or traffic, allowing for continuous monitoring and validation at each stage.

Progressive Exposure techniques like Canary deployments, Dark Launches and Ring-based deployments align perfectly with the needs. Canary deployments expose the new version to a small percentage of users, allowing monitoring of performance and errors before widening the scope. Dark Launches deploy the new version but keep it hidden from users, allowing for thorough backend testing. Ring-based deployments gradually expose the new version to internal teams, then beta users, and finally all users, based on defined "rings" of exposure.

A/B testing, while involving different versions, focuses on comparing performance metrics for user behavior and conversion, and doesn't inherently address gradual rollout with controlled user exposure. Feature toggles enable/disable features without requiring redeployment, addressing feature release rather than versioned service deployment and controlled exposure. Blue/green deployments provide fast rollback capabilities but involve switching all traffic from one environment to another, which doesn't offer the granular, incremental exposure needed for continuous monitoring during rollout. With Blue/Green, testing is often done prior to the complete traffic shift, thus not enabling gradual monitoring.

Progressive exposure provides the needed granularity to monitor the impact of changes in a controlled manner. If any issues arise during the rollout, it's straightforward to pause or roll back the deployment to the previous version before impacting a large user base, thus mitigating risks and improving service reliability. The gradual rollout allows for performance analysis and resource allocation optimization.

In summary, progressive exposure facilitates controlled testing, monitoring, and validation, which directly addresses the stated requirements for a microservice deployment strategy that enables gradual rollout and user control.

Here are some helpful links for further research:

Microsoft Azure DevOps documentation on Deployment Strategies: (Often linked to from the Azure documentation on pipelines and releases, although not a specific "Progressive Exposure" page directly)
Canary Deployments: https://martinfowler.com/bliki/CanaryRelease.html
Feature Toggles: https://martinfowler.com/articles/feature-toggles.html
Blue/Green Deployment: https://martinfowler.com/bliki/BlueGreenDeployment.html

Question: 516 CertyIQ

HOTSPOT
-
You have an Azure Pipelines pipeline named Pipeline1 that has the following YAML definition.

```
pool:
  name: Azure Pipelines
  demands:
  - msbuild
  - visualstudio

steps:
- task: VSBuild@1
  displayName: 'Build solution **\*.sln'
  inputs:
    solution: '$(Parameters.Solution)'
    platform: '$(BuildPlatform)'
    configuration: '$(BuildConfiguration)'

- task: CopyFiles@2
  displayName: 'Copy Files to: $(Build.ArtifactStagingDirectory)'
  inputs:
    SourceFolder: '$(System.DefaultWorkingDirectory)'
    Contents: '**\bin\$(BuildConfiguration)\**'
    TargetFolder: '$(Build.ArtifactStagingDirectory)'
  condition: succeededOrFailed()

- task: PublishBuildArtifacts@1
  displayName: 'Publish Artifact: drop'
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'
  condition: succeededOrFailed()
```

For each of the following statements, select Yes if True. Otherwise select No.
NOTE: Each correct selection is worth one point.

Answer Area

Statements	Yes	No
Pipeline1 consists of three stages.	○	○
`$(BuildPlatform)` is a predefined variable.	○	○
`$(System.DefaultWorkingDirectory)` is a predefined variable.	○	○

Answer:

Answer Area

Statements	Yes	No
Pipeline1 consists of three stages.	○	○
`$(BuildPlatform)` is a predefined variable.	○	○
`$(System.DefaultWorkingDirectory)` is a predefined variable.	○	○

Question: 517 CertyIQ

You use Azure Pipelines to build and test code.
You need to analyze the agent pool usage.
What are two ways to achieve the goal? Each correct answer presents a complete solution.
NOTE: Each correct selection is worth one point.

A.Review the historical graph for the agent pools.
B.Review the Pipeline duration report.
C.Query the TaskAgentPoolSizeSnapshot/TaskAgentPoolSizeSnapshots endpoint.
D.Query the PipelineRun/PipelineRuns endpoint.

Answer: AC

Explanation:

The correct answer is A and C. Let's justify this by examining each option:

A. Review the historical graph for the agent pools. This is a valid method. Azure DevOps provides built-in historical graphs for agent pools. These graphs visually represent agent utilization, including the number of available agents, running jobs, and queued jobs over time. By examining these graphs, you can directly observe the agent pool usage patterns and identify bottlenecks or periods of underutilization. This visualization helps in making informed decisions about scaling the agent pool to meet demand. The 'usage' trend reveals agents performing jobs or being idle. This can be achieved directly in the Azure DevOps portal within the agent pool settings.

B. Review the Pipeline duration report. Pipeline duration reports provide insights into the overall time it takes for pipelines to complete. While useful for understanding pipeline performance, they don't directly provide information about agent pool usage. Although a long pipeline execution could indirectly suggest high agent load, it doesn't pinpoint which agents were busy, for how long, or the efficiency of agent usage. The report highlights bottlenecks within pipeline tasks, not necessarily within the agent pool's resource availability.

C. Query the TaskAgentPoolSizeSnapshot/TaskAgentPoolSizeSnapshots endpoint. This is another correct approach. Azure DevOps provides a REST API that allows you to programmatically query data about agent pools, including their size and utilization. The TaskAgentPoolSizeSnapshot endpoint gives you point-in-time snapshots of the agent pool's state. Querying this endpoint can provide you with detailed information about the number of agents, their availability, and their status, enabling you to programmatically analyze the agent pool's usage over time. This can be integrated into monitoring and alerting systems. You can access raw data for analytical purposes.

D. Query the PipelineRun/PipelineRuns endpoint. This is incorrect. While the PipelineRun endpoint provides information about pipeline executions, such as their start and end times, status, and associated metadata, it doesn't directly provide details about agent pool usage. You can see which agent a pipeline used via _logDownload, but that is not the core function of this API. You would need to correlate pipeline run data with other data sources to infer agent usage, which is inefficient compared to the direct methods in options A and C. It shows details of the pipeline, not agent pool statistics.

In summary, reviewing the historical graphs for agent pools (A) provides a visual representation of agent usage, while querying the TaskAgentPoolSizeSnapshot endpoint (C) allows for programmatic access to agent pool statistics, both directly addressing the need to analyze agent pool usage.

Authoritative links:

Azure DevOps REST API - Agent Pools: https://learn.microsoft.com/en-us/rest/api/azure/devops/distributedtask/pools?view=azure-devops-rest-7.1
Monitor performance - Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/reports/performance?view=azure-devops

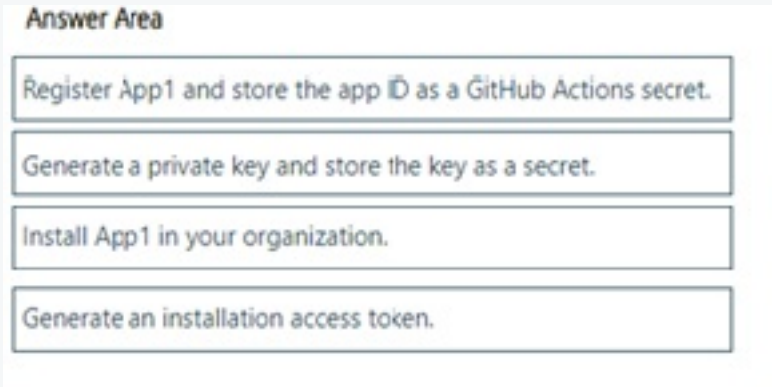
Question: 518

CertyIQ

DRAG DROP
-
You have a GitHub organization.
You are creating a GitHub Actions workflow.
You need to perform authenticated API requests by using a GitHub app name App1.
Which four actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



Answer:



Question: 519 CertyIQ

You use an Azure Pipelines pipeline to build and deploy an app.
You have a custom test task that has the following inputs:
•testResultsFiles: **/TEST-*.trx
•searchFolder: $(System.DefaultWorkingDirectory)
•mergeTestResults: true
Which format should you use for the input data of testResultsFiles?

A.VSTest
B.NUnit
C.CTest
D.JUnit

Answer: A

Explanation:

The correct answer is A (VSTest). The testResultsFiles input within the context of Azure Pipelines and particularly custom test tasks, specifically targets the file format produced by Visual Studio's testing framework. These files typically have the .trx extension. The example provided, /TEST-*.trx*, confirms this pattern. VSTest is the test platform used by Visual Studio and MSBuild for running tests. Other options such as NUnit, CTest, and JUnit use different file formats (e.g., XML formats specific to their respective frameworks) for their test results. Because the *testResultsFiles* is specifically looking for .trx files, and .trx files are produced by VSTest, VSTest is the appropriate format. The mergeTestResults: true setting also indicates that Azure Pipelines is configured to consolidate results from multiple VSTest runs into a single, unified report. Therefore, providing any other test result format besides VSTest would not be compatible with the task configuration and might cause errors or unexpected behavior within the pipeline. The searchFolder: $(System.DefaultWorkingDirectory) simply instructs the pipeline where to begin looking for the VSTest result files, which makes the file format the significant element here.

Supporting links:

Publish Test Results task - Azure Pipelines | Microsoft Learn
Run tests task - Azure Pipelines | Microsoft Learn

Question: 520 CertyIQ

You use an Azure Pipelines pipeline to build, test, and deploy an app named App1.
You need to reduce how long it takes to complete unit and integration tests for App1. The solution must ensure that the code coverage testing ratio is maintained.
What should you do?

A.Enable flaky test management.
B.Purchase additional parallel jobs.
C.Enable Test Impact Analysis (TIA).
D.Add an agent pool.

Answer: C

Explanation:

The correct answer is C: Enable Test Impact Analysis (TIA).

Test Impact Analysis (TIA) in Azure DevOps analyzes the relationship between tests and code changes. It identifies only the tests that are impacted by recent code modifications. By running only the relevant tests, TIA significantly reduces the execution time of unit and integration tests without compromising code coverage. This directly addresses the requirement to reduce testing time while maintaining the code coverage ratio.

Option A, enabling flaky test management, addresses test instability, not the overall test execution time. Flaky tests are tests that sometimes pass and sometimes fail without any code changes. While addressing flaky tests is important for pipeline reliability, it doesn't inherently reduce the number of tests being run.

Option B, purchasing additional parallel jobs, would speed up the overall pipeline execution by allowing more tests to run concurrently. However, it does not specifically reduce the number of tests being run. It also incurs additional cost. While parallel jobs can help, it's not as targeted a solution as TIA, especially if the primary goal is to reduce test time without sacrificing coverage.

Option D, adding an agent pool, is similar to purchasing parallel jobs. More agents can execute tasks concurrently, but the total workload (number of tests) remains the same. The core issue here is running fewer tests, not merely running tests faster.

Therefore, enabling TIA is the most efficient and cost-effective method to reduce the duration of unit and integration tests while retaining the necessary code coverage because it intelligently selects only the tests that require execution based on the latest code changes.

References:

Test Impact Analysis (TIA): https://learn.microsoft.com/en-us/azure/devops/pipelines/test/impact-analysis?view=azure-devops
Parallel jobs: https://learn.microsoft.com/en-us/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops&tabs=ms-hosted

Question: 521 CertyIQ

You have an app that is deployed to two environments named Production-A and Production-B by using Azure Pipelines.
You need to configure a release pipeline that will mark the app as complete and ready for release into the Production-B environment. The solution must meet the following requirements:
•Ensure that there are no active Azure Monitor alerts in the Production-A environment before the app is marked as complete.
•Minimize administrative effort.
What should you do?

A.To the Production-B environment stage, add a pre-deployment gate that will query Azure Monitor.
B.To the Production-A environment stage, add a post-deployment gate that will query Azure Monitor.
C.To the Production-A environment stage, add a post-deployment approval.
D.To the Production-A environment stage, add a pre-deployment gate that will query Azure Monitor.

Answer: B

Explanation:

The correct answer is B: To the Production-A environment stage, add a post-deployment gate that will query Azure Monitor.

Here's a detailed justification:

The requirement is to ensure no active Azure Monitor alerts exist in Production-A before releasing to Production-B. This necessitates a check on Production-A after the deployment to Production-A has completed, but before proceeding to Production-B. A post-deployment gate is the appropriate mechanism for this.

A gate in Azure Pipelines is an automated check that can be added to a release stage to verify certain conditions are met before the release proceeds to the next stage or task. A post-deployment gate runs after the deployment to that stage is complete. This is essential because we need to assess the state of Production-A after the app has been deployed to it.

By adding a post-deployment gate to the Production-A stage, we can configure it to query Azure Monitor for active alerts. If any alerts are found, the gate will prevent the release from proceeding to Production-B. If no alerts are found, the gate will allow the release to continue. This directly addresses the requirement of ensuring no active alerts in Production-A before releasing to Production-B.

Option A is incorrect because a pre-deployment gate on Production-B would run before anything is deployed to Production-B. It wouldn't be able to assess the state of Production-A after a deployment.

Option C is incorrect because a post-deployment approval requires manual intervention. The question explicitly states that the solution must minimize administrative effort, which rules out manual approvals. Manual approvals don't automatically check for alerts.

Option D is incorrect because a pre-deployment gate on Production-A would run before the deployment to Production-A even occurs. It wouldn't be able to check for alerts after the deployment.

In summary, a post-deployment gate querying Azure Monitor on the Production-A stage is the ideal solution because it automatically checks for alerts in Production-A after a deployment there is complete and minimizes administrative effort by preventing releases to Production-B when alerts are active.

Relevant Documentation:

Release gates and approvals: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/approvals/gates?view=azure-devops
Azure Monitor REST API: https://learn.microsoft.com/en-us/rest/api/monitor/

Question: 522 CertyIQ

You manage package feeds by using Azure Artifacts.
You plan to create a new package feed that will include the following views:
•@Local
•@Latest
•@Release
•@Prerelease
Which view should you create manually?

A.@Local
B.@Latest
C.@Release
D.@Prerelease

Answer: B

Explanation:

The correct answer is **B. @Latest**. Here's a detailed justification:

Azure Artifacts feeds provide a mechanism for organizing and sharing packages within a team or organization. Views are a crucial feature for promoting packages through the software development lifecycle, ensuring that consumers only see packages that meet certain quality criteria.

When creating a new feed, Azure Artifacts automatically generates the @Local and @Prerelease views. The @Local view contains packages published directly to the feed or saved from upstream sources. The @Prerelease view, introduced later, allows for specifically identifying pre-release packages. The @Latest view is specifically deprecated because Azure artifacts uses versions to define which is the "latest" package. If you promote a specific version to a view such as @Release, then that is the release package.

The @Release view, on the other hand, represents packages that have been thoroughly tested and approved for production use. Because this view is specific to the release process, it should be created manually, so that packages need to be manually promoted to release. This prevents accidental inclusion of unsuitable packages in the release process. The creation and management of @Release view is an intentional act reflecting a promotion workflow. Thus, @Latest being deprecated, the @Release view is the only one that needs to be created manually.

Therefore, of the options provided, only @Latest should not be created, as it is no longer supported, thus the answer is B.

Further research:

Azure Artifacts views: https://learn.microsoft.com/en-us/azure/devops/artifacts/concepts/views?view=azure-devops

Question: 523 CertyIQ

You plan to create an image that will contain a .NET Core application.
You have a Dockerfile file that contains the following code. (Line numbers are included for reference only.)

```
01 FROM mcr.microsoft.com/dotnet/sdk:7.0
02 COPY . /
03 RUN dotnet publish -c Release -o out
04 FROM mcr.microsoft.com/dotnet/sdk:7.0
05 COPY --from=0 /out /
06 WORKDIR /
07 ENTRYPOINT ["dotnet", "app1.dll"]
```

You need to ensure that the image is as small as possible when the image is built.
Which line should you modify in the file?

A.1
B.3
C.4
D.7

Answer: C

Explanation:

Correct answer is C:4.

Question: 524 CertyIQ

You have an Azure subscription that includes an app named App1.
You have an Azure DevOps project that contains two environments named Staging and Production.
You use Azure Pipelines to deploy App1.
You need to validate the performance of App1 in the Staging environment before it is deployed to Production. The solution must minimize administrative effort.
What should you do in the Azure DevOps project?

A.In the production branch policy, add a status check to query Azure Monitor Alerts for active alerts.
B.In the Production environment, add a check to query Azure Monitor Alerts for active alerts.
C.In the Production environment stage, add a post-deployment approval for the Azure Monitor Alerts group.
D.In the Staging environment add a check to query Azure Monitor Alerts for active alerts.

Answer: D

Explanation:

The correct answer is **D. In the Staging environment add a check to query Azure Monitor Alerts for active alerts.**

Here's a detailed justification:

The core requirement is to validate App1's performance in the Staging environment before deploying to Production. This means checks should be implemented within the Staging environment's deployment process. The goal is to automatically gate the promotion of the application if performance issues are detected.

Option D directly addresses this by introducing a check in the Staging environment that queries Azure Monitor Alerts. Azure Monitor is used for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. By querying Azure Monitor Alerts, the pipeline can determine if any performance alerts are currently active for App1 while it's running in Staging. If alerts exist, it indicates a performance issue, and the deployment to Production should be blocked.

Options A, B, and C are incorrect because they focus on the Production environment. Adding checks or approvals in Production won't prevent a potentially flawed deployment from reaching the production environment in the first place. The goal is to prevent a problematic deployment before it reaches the production environment. Option A uses branch policies, which are more suited for code quality and code review enforcement and not environment checks. Option C involving approval group, while useful for manual validation, doesn't meet the requirement of minimizing administrative effort by employing an automated check based on alerts. Option B applies the check to Production, missing the requirement of validation in Staging.

Therefore, placing the Azure Monitor Alerts check in the Staging environment ensures that only validated, performant applications are promoted to Production, thereby minimizing risk and reducing administrative overhead. This aligns with best practices for continuous integration and continuous delivery (CI/CD).

Supporting links:

Azure Monitor Alerts: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-overview
Azure Pipelines Environments: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/environments?view=azure-devops

Azure Pipelines Checks: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops&tabs=check-types

Question: 525 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it As a result these questions will not appear in the review screen.
You need to use an Azure Pipelines pipeline to build and test an app and test the database of the app. The solution must meet the following requirements.
•The test stages must be run in parallel.
•The Publish_Test_Results stage must always be run.
•The test stages must be run after successful completion of the build stage.
•The Publish_Test_Results stage must be run after completion of all the test stages.
Solution: You include the following elements in the YAML definition of the pipeline.

```
...
stages:
- stage: Build_App
jobs:
- stage: Test_App
dependsOn: [Build_App]
jobs:
- stage: Test_Database
dependsOn: [Build_App]
jobs:
- stage: Publish_Test_Results
jobs:
...
```

Does this meet the goal?

A.Yes
B.No

Answer: B

Explanation:

Correct answer is B:No.

Question: 526 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it As a result these questions will not appear in the review screen.
You need to use an Azure Pipelines pipeline to build and test an app and test the database of the app. The solution must meet the following requirements.

•The test stages must be run in parallel.
•The Publish_Test_Results stage must always be run.
•The test stages must be run after successful completion of the build stage.
•The Publish_Test_Results stage must be run after completion of all the test stages.
Solution: You include the following elements in the YAML definition of the pipeline.

```
...
stages:
- stage: Build_App
jobs:
- stage: Test_App
dependsOn: []
jobs:
- stage: Test_Database
dependsOn: []
jobs:
- stage: Publish_Test_Results
dependsOn: [
Build_App,
Test_App,
Test_Database
]
condition: succeededOrFailed()
jobs:
...
```

Does this meet the goal?

A.Yes
B.No

Answer: B

Explanation:

Correct answer is B:No.

Question: 527 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it As a result these questions will not appear in the review screen.
You need to use an Azure Pipelines pipeline to build and test an app and test the database of the app. The solution must meet the following requirements.
•The test stages must be run in parallel.
•The Publish_Test_Results stage must always be run.
•The test stages must be run after successful completion of the build stage.
•The Publish_Test_Results stage must be run after completion of all the test stages.
Solution: You include the following elements in the YAML definition of the pipeline.

```
...
stages:
- stage: Build_App
jobs:
- stage: Test_App
dependsOn: [Build_App]
jobs:
- stage: Test_Database
dependsOn: [Build_App]
jobs:
- stage: Publish_Test_Results
dependsOn: [
Build_App,
Test_App,
Test_Database
]
condition: always()
jobs:
...
```

Does this meet the goal?

A.Yes
B.No

Answer: A

Explanation:

1. The Test Stages Must Run in Parallel:Test App and Test Database stages both depend on Build_App, and no dependencies are defined between them. This allows the two test stages to run in parallel after the Build_App stage completes.2.The Publish_Test_Results Stage Must Always Be Run:The condition: always() in the Publish_Test_Results stage ensures that it will run regardless of the success or failure of any preceding stages.3.The Test Stages Must Be Run After Successful Completion of the Build Stage:Both Test App and Test Database stages explicitly depend on Build_App (dependsOn: [Build_App]), ensuring they only start after Build_App completes successfully.4.The Publish_Test_Results Stage Must Be Run After Completion of All the Test Stages:The Publish_Test_Results stage includes dependsOn: [Build_App, Test App, Test_Database], ensuring it will only execute after all preceding stages (including the test stages) have finished.

Question: 528 CertyIQ

HOTSPOT
-
You have an Azure Pipelines pipeline that has the following definition.

```
jobs:

- job: A
  steps:
  # job A steps here

- job: B
  dependsOn: A
  steps:
  # job B steps here

- job: C
  dependsOn: A
  steps:
  # job C steps here

- job: D
  dependsOn:
   - B
   - C
   steps:
  # job D steps here

- job: E
  dependsOn:
   - B
   - D
   steps:
  # job E steps here
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No.
NOTE: Each correct selection is worth one point.

Answer Area

Statements	Yes	No
Job E can start before job C completes.	○	○
Job B can run concurrently with job C.	○	○
Job D can start before job B completes.	○	○

Answer:

Answer Area

Statements	Yes	No
Job E can start before job C completes.	○	◉
Job B can run concurrently with job C.	◉	○
Job D can start before job B completes.	○	◉

Question: 529 CertyIQ

You are developing an Azure Pipelines pipeline.
You need to configure a check in the pipeline that will query Azure Boards to ensure that there are no active work item issues before the pipeline deploys a build to production.
Which type of check should you implement?

A.post-deployment approvals
B.manual validations
C.pre-deployment gates
D.pre-deployment approvals

Answer: C

Explanation:

The correct answer is C, pre-deployment gates. Let's break down why.

Pre-deployment gates are automated checks that evaluate certain criteria before a pipeline stage is allowed to execute. This perfectly aligns with the requirement of ensuring no active work item issues exist before deployment to production. Gates can query external systems, like Azure Boards in this scenario, and make pass/fail decisions based on the results. A custom Azure Function, for example, can be invoked by the gate, to

call the Azure Boards REST API.

Options A and D, post-deployment and pre-deployment approvals, involve manual human intervention. While approvals are important for adding a layer of governance, they don't address the automated, programmatic check described in the scenario. The question specifically asks about a check to query Azure Boards programmatically.

Option B, manual validation, is similar to approvals in that it needs human interaction to proceed. It's designed to validate something after a stage, rather than an automated check before a stage like pre-deployment gates. Again, this doesn't suit the requirement of automating the work item check.

Pre-deployment gates offer the ideal mechanism for integrating an automated check querying Azure Boards and preventing deployments if certain conditions (like active work item issues) are met.

Authoritative links for further research:

Using Gates and Approvals to control pipeline execution: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?view=azure-devops&tabs=check-types
Azure Pipelines Checks: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/checks?view=azure-devops

Question: 530 CertyIQ

You have an Azure Resource Manager (ARM) template that contains the following expression.
[if(parameters('isComplete'), '1a', '2a')]
You need to migrate the template to Bicep.
Which expression should you run?

A.iif(isComplete, '1a', '2a')
B.if(isComplete, '1a', '2a')
C.if('isComplete') '1a' else '2a'
D.isComplete ? '1a' '2a'

Answer: D

Explanation:

The correct Bicep equivalent of the ARM template expression [if(parameters('isComplete'), '1a', '2a')] is isComplete ? '1a' : '2a'. This is because Bicep leverages a more concise, ternary operator syntax for conditional expressions.

Option A, iif(isComplete, '1a', '2a'), might seem intuitive as iif is often used for inline if-else statements in some languages. However, iif is not the standard or recommended way to implement conditional logic in Bicep.

Option B, if(isComplete, '1a', '2a'), is syntactically incorrect in Bicep. Bicep doesn't utilize an if function with that signature for inline conditionals.

Option C, if('isComplete') '1a' else '2a', is also incorrect. This syntax is not how conditional statements are structured in Bicep. Furthermore, it incorrectly passes the string literal 'isComplete' instead of evaluating the parameter value.

Option D, isComplete ? '1a' : '2a', is the standard ternary operator in Bicep. The expression isComplete represents the boolean condition. If it's true, the expression evaluates to '1a'; otherwise, it evaluates to '2a'. The colon (:) separates the true and false conditions. This approach aligns with Bicep's design principles of

providing a cleaner and more readable syntax compared to ARM templates.

Therefore, the ternary operator condition ? true-value : false-value is the recommended way to express conditional logic in Bicep, making option D the correct choice.

Relevant documentation for further research:

Bicep Conditional Deployments: https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/conditional-resource-deployment
Bicep Operators: https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/operators

Question: 531 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You use an Azure Pipelines pipeline to build and release web apps.
You need to configure the pipeline to meet the following requirements:
•Only run when there is a change in the /webapp folder.
•Only run when a pr is created.
Solution: You configure the pipeline definition by using the following elements.

```
...
trigger:
  paths:
    include: /webapp
  branches:
    include: refs/head/pr
...
```

Does this meet the goal?

A.Yes
B.No

Answer: B

Explanation:

Answer is Btrigger: paths: include: - webapp/**pr: paths: include: - webapp/**

Question: 532 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You use an Azure Pipelines pipeline to build and release web apps.

You need to configure the pipeline to meet the following requirements:
•Only run when there is a change in the /webapp folder.
•Only run when a pr is created.
Solution: You configure the pipeline definition by using the following elements.

```
...
trigger:
  paths:
    include: /webapp
  branches:
    include: pr
...
```

Does this meet the goal?

A.Yes

B.No

Answer: B

Explanation:

Correct answer is B:No.

Question: 533 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You use an Azure Pipelines pipeline to build and release web apps.
You need to configure the pipeline to meet the following requirements:
•Only run when there is a change in the /webapp folder.
•Only run when a pr is created.
Solution: You configure the pipeline definition by using the following elements.

```
...
pr:
  paths:
    include: /pr
  branches:
    include: refs/head/webapp
...
```

Does this meet the goal?

A.Yes

B.No

Answer: B

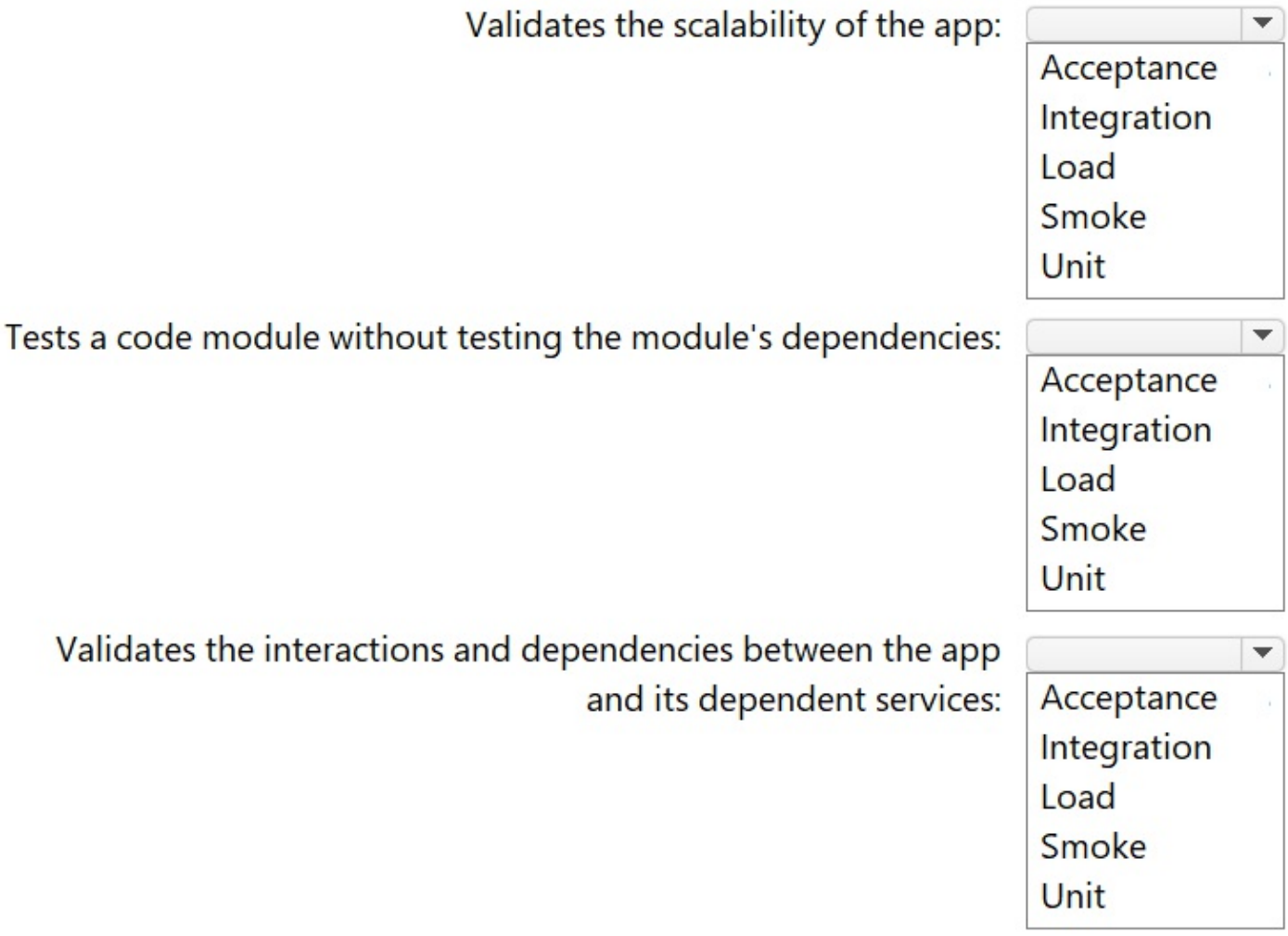
Explanation:

Correct answer is B:No.

Question: 534 CertyIQ

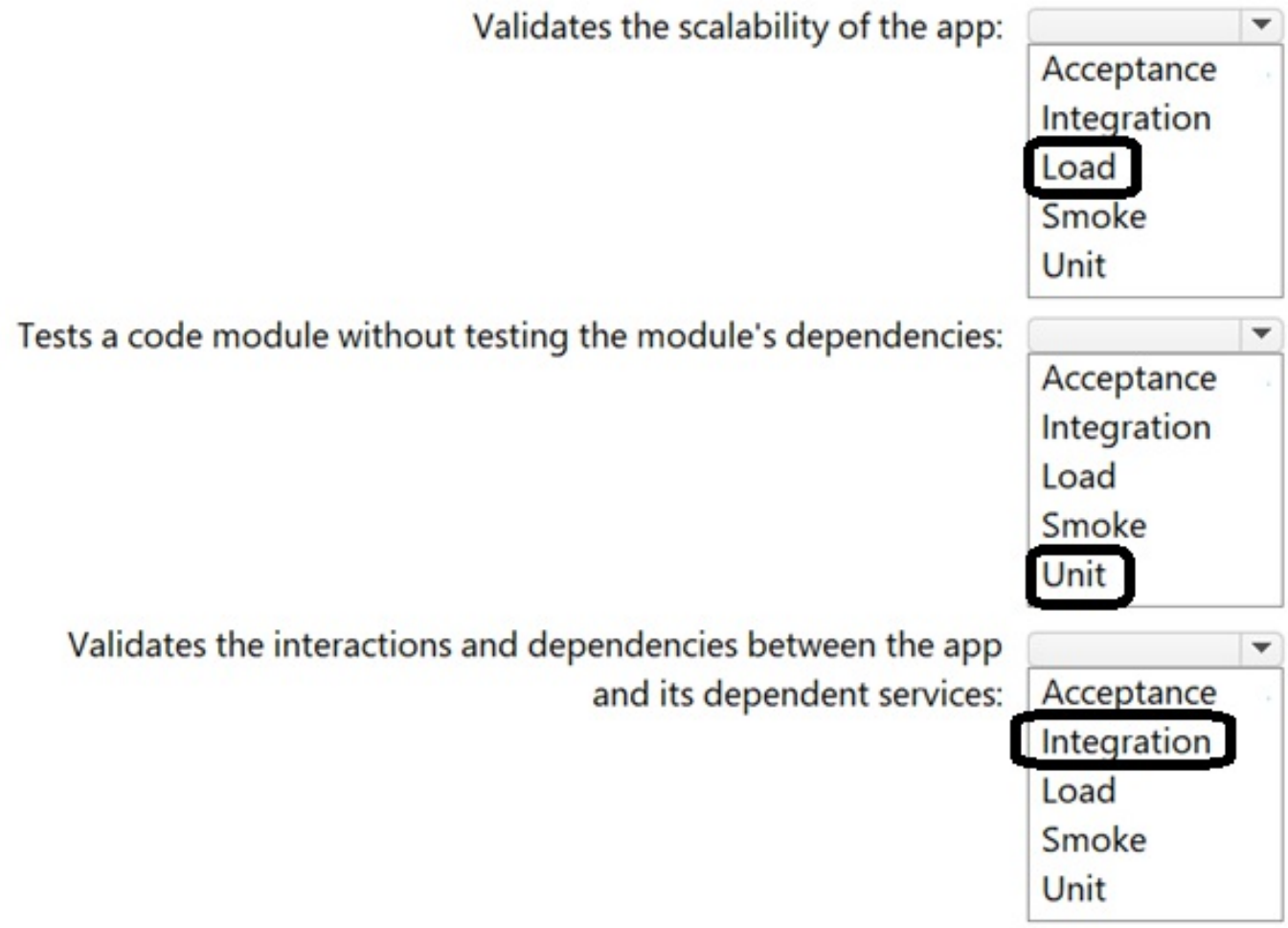
HOTSPOT
-
You are using an Azure Pipelines pipeline to build and deploy a web app.
You need to implement a testing strategy that meets the following requirements:
•Validates the scalability of the app
•Tests a code module without testing the module's dependencies
•Validates the interactions and dependencies between the app and its dependent services
Which type of test should you implement for each requirement? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area



Answer:

Answer Area



Question: 535 CertyIQ

DRAG DROP
-
You have an Azure subscription that contains 50 virtual machines.
You manage the configuration of the virtual machines by using Azure Automation State Configuration.
You need to ensure that Windows Defender is installed on each virtual machine and the Windows Defender service is running.
How should you complete the Desired State Configuration (DSC) code? To answer, drag the appropriate values to the correct targets. Each value may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Values

- 'Automatic'
- 'Enabled'
- 'IncludeAllSubFeature'
- 'Manual'
- 'Present'
- 'Running'
- 'Started'

Answer Area

```
Configuration EnsureSecure {

    Node "localhost" {

        WindowsFeature WindowsDefenderInstalled {

            Ensure =

            Name = 'Windows-Defender'

        }

        Service WindowsDefenderAvailable {

            Name = 'WinDefend'

            StartupType = 'Automatic'

            State =

        }

    }

}

MyDscConfiguration -OutputPath C:\temp\
```

Answer:

Answer Area

```
Configuration EnsureSecure {
    Node "localhost" {
        WindowsFeature WindowsDefenderInstalled {
            Ensure =    'Present'
            Name = 'Windows-Defender'
        }
        Service WindowsDefenderAvailable {
            Name = 'WinDefend'
            StartupType = 'Automatic'
            State =     'Running'
        }
    }
}
MyDscConfiguration -OutputPath C:\temp\
```

Question: 536 CertyIQ

SIMULATION
-
Initialize the default main branch, if it does not exist already.
In the User1-12345678 organization, you need to install the Microsoft Security DevOps extension.
Next, create a new starter pipeline named starter1 that will use the following starter code.

```
trigger:
    - starter
pool:
    # ubuntu-latest also supported.
    vmImage: 'windows-latest'
steps:
```

Ensure that starter1 includes a task that executes the extension and uses the following inputs:
•Command: run
•Policy: azuredevops
•Publish: true
Save the pipeline to a new branch named starter.

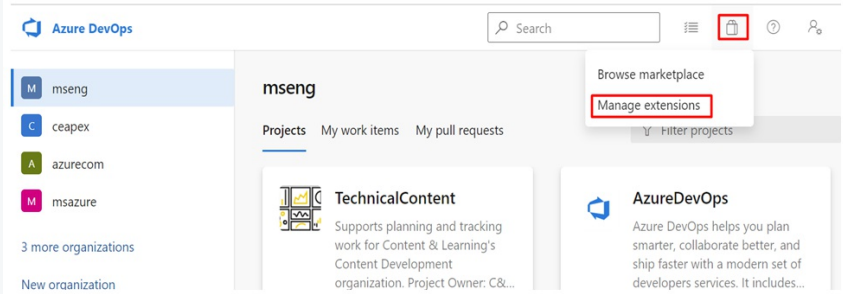
Answer:

Task 1: install the Microsoft Security DevOps extension
Configure the Microsoft Security DevOps Azure DevOps extension

To configure the Microsoft Security DevOps Azure DevOps extension:

Step 1: Sign in to Azure DevOps.

Step 2: Navigate to Shopping Bag > Manage extensions.



Step 3: Select Shared.

Step 4: Select Microsoft Security DevOps.



Step 5: Select Install.

Step 6: Select the appropriate organization from the dropdown menu.

Step 7: Select Install.

Step 8: Select Proceed to organization.

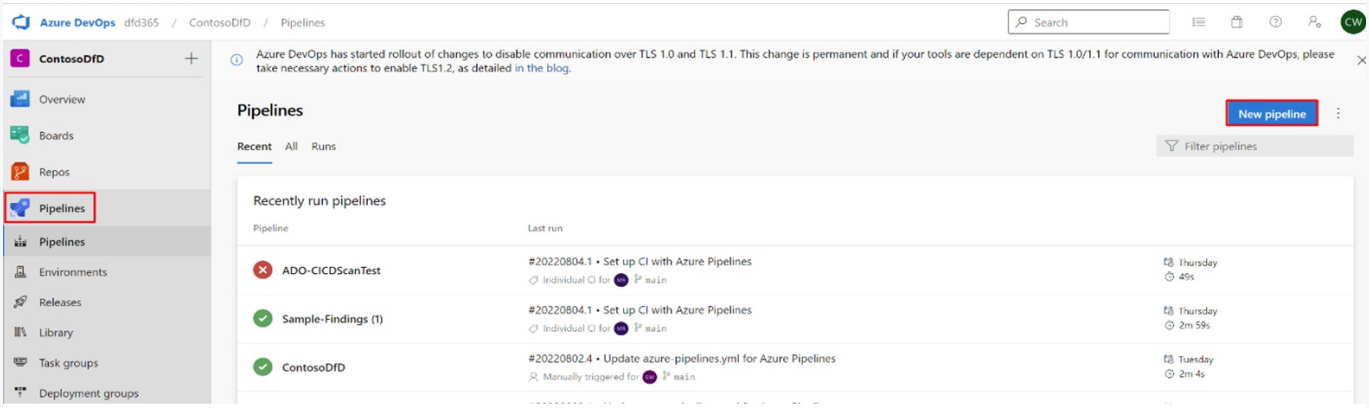
Task 2: create a new starter pipeline named starter1
Configure your pipelines using YAML
To configure your pipeline using YAML:

Step 1: Sign into Azure DevOps

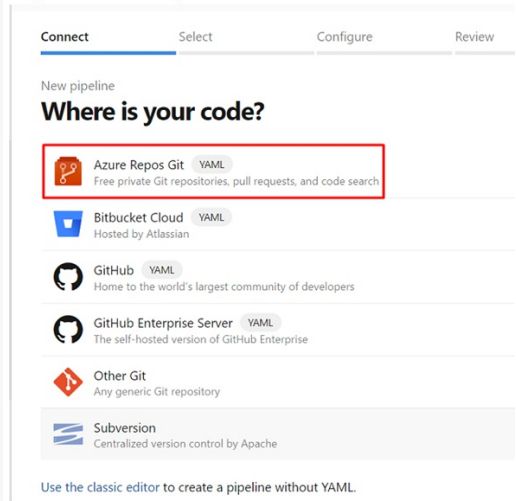
Step 2: Select your project.

Step 3: Navigate to Pipelines

Step 4: Select New pipeline.



Step 5: Select Azure Repos Git.

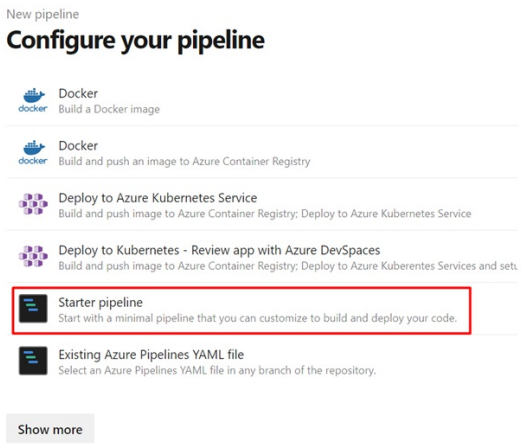


Step 6: Select the relevant repository.





Step 7: Select Starter pipeline.



Step 8: Paste the following YAML into the pipeline:

```
trigger:
    - starter
pool:
    # ubuntu-latest also supported.
    vmImage: 'windows-latest'
steps:
```

- task: MicrosoftSecurityDevOps@1
displayName: 'Microsoft Security DevOps'
inputs:

Command: `run`
Policy: `azuredevops`
Publish: `true`

Note: Example:

```
    # Starter pipeline
    # Start with a minimal pipeline that you can customize to build and deploy your code.
    # Add steps that build, run tests, deploy, and more:
    # https://aka.ms/yaml
    trigger: none
    pool:
      # ubuntu-latest also supported.
      vmImage: 'windows-latest'
    steps:
    - task: MicrosoftSecurityDevOps@1
      displayName: 'Microsoft Security DevOps'
      # inputs:
        # config: string. Optional. A file path to an MSDO configuration file ('*.gdnconfig'). Vist the MSDO GitHub wiki linked below for additional configuration instructions
        # policy: 'azuredevops' | 'microsoft' | 'none'. Optional. The name of a well-known Microsoft policy to determine the tools/checks to run. If no configuration file or list of tools is provided, the
    policy may instruct MSDO which tools to run. Default: azuredevops.
        # categories: string. Optional. A comma-separated list of analyzer categories to run. Values: 'code', 'artifacts', 'IaC', 'containers'. Example: 'IaC, containers'. Defaults to all.
        # languages: string. Optional. A comma-separated list of languages to analyze. Example: 'javascript,typescript'. Defaults to all.
        # tools: string. Optional. A comma-separated list of analyzer tools to run. Values: 'bandit', 'binskim', 'checkov', 'eslint', 'templateanalyzer', 'terrascan', 'trivy'. Example 'templateanalyzer, trivy'
        # break: boolean. Optional. If true, will fail this build step if any high severity level results are found. Default: false.
        # publish: boolean. Optional. If true, will publish the output SARIF results file to the chosen pipeline artifact. Default: true.
  # artifactName: string. Optional. The name of the pipeline artifact to publish the SARIF result file to. Default: CodeAnalysisLogs*.
```

Step 9: To commit the pipeline, select Save and run.

The pipeline will run for a few minutes and save the results.

Reference:
https://learn.microsoft.com/en-us/azure/defender-for-cloud/azure-devops-extension

Question: 537

CertyIQ

SIMULATION
-
Initialize the default main branch, if it does not exist already.
In Project1, you need to create a new Azure Pipelines YAML pipeline by using the ASP.NET template.
The pipeline must use Azure Repos as the hosting platform and must be created in a new branch named azure-pipelines.

Answer:

Create your first Java pipeline

1. Sign in to your Azure DevOps organization and go to your project.
2. Go to Pipelines, and then select New pipeline or Create pipeline if creating your first pipeline.
3. Do the steps of the wizard by first selecting GitHub as the location of your source code.
4. You might be redirected to GitHub to sign in. If so, enter your GitHub credentials.
5. When you see the list of repositories, select your repository.
6. You might be redirected to GitHub to install the Azure Pipelines app. If so, select Approve & install.
7. Azure Pipelines will analyze your repository and recommend the Maven pipeline template. [Select the ASP.NET template]
8. When your new pipeline appears, take a look at the YAML to see what it does. When you're ready, select Save and run.
9. You're prompted to commit a new azure-pipelines.yml file to your repository. After you're happy with the message, select Save and run again.

Reference:
https://learn.microsoft.com/en-us/azure/devops/pipelines/create-first-pipeline

Question: 538 CertyIQ

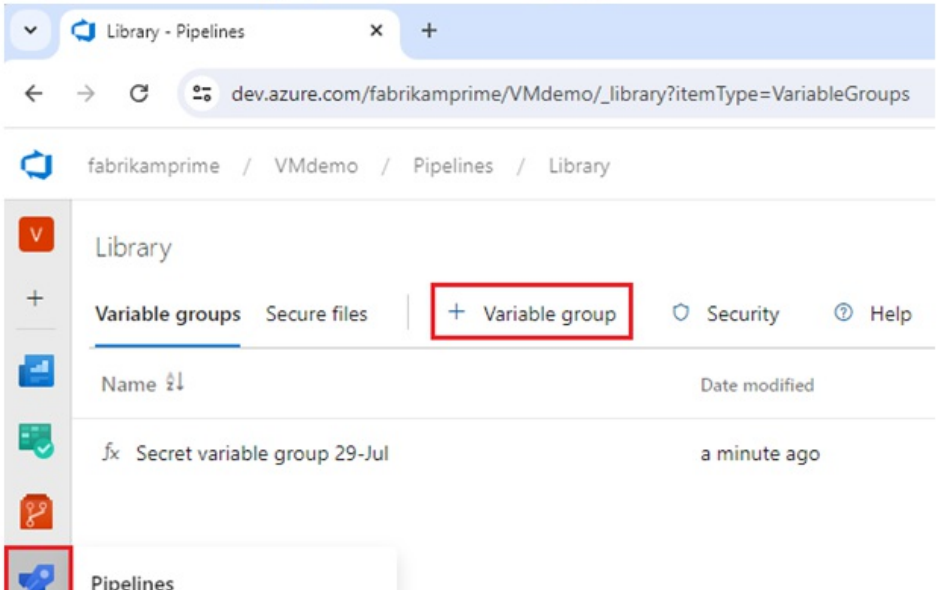
SIMULATION
-
In Project1, you need to create a variable group named varGroup1 that will contain the following variables:
serverName: server1
dbName: db1

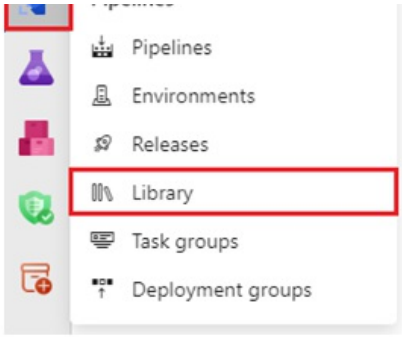
Answer:

Azure DevOps Azure Pipelines, Manage variable groups
Create a variable group
You can create variable groups for the pipeline runs in your project.

Step 1: In your Azure DevOps project [Here Project1], select Pipelines > Library from the left menu.

Step 2: On the Library page, select + Variable group.



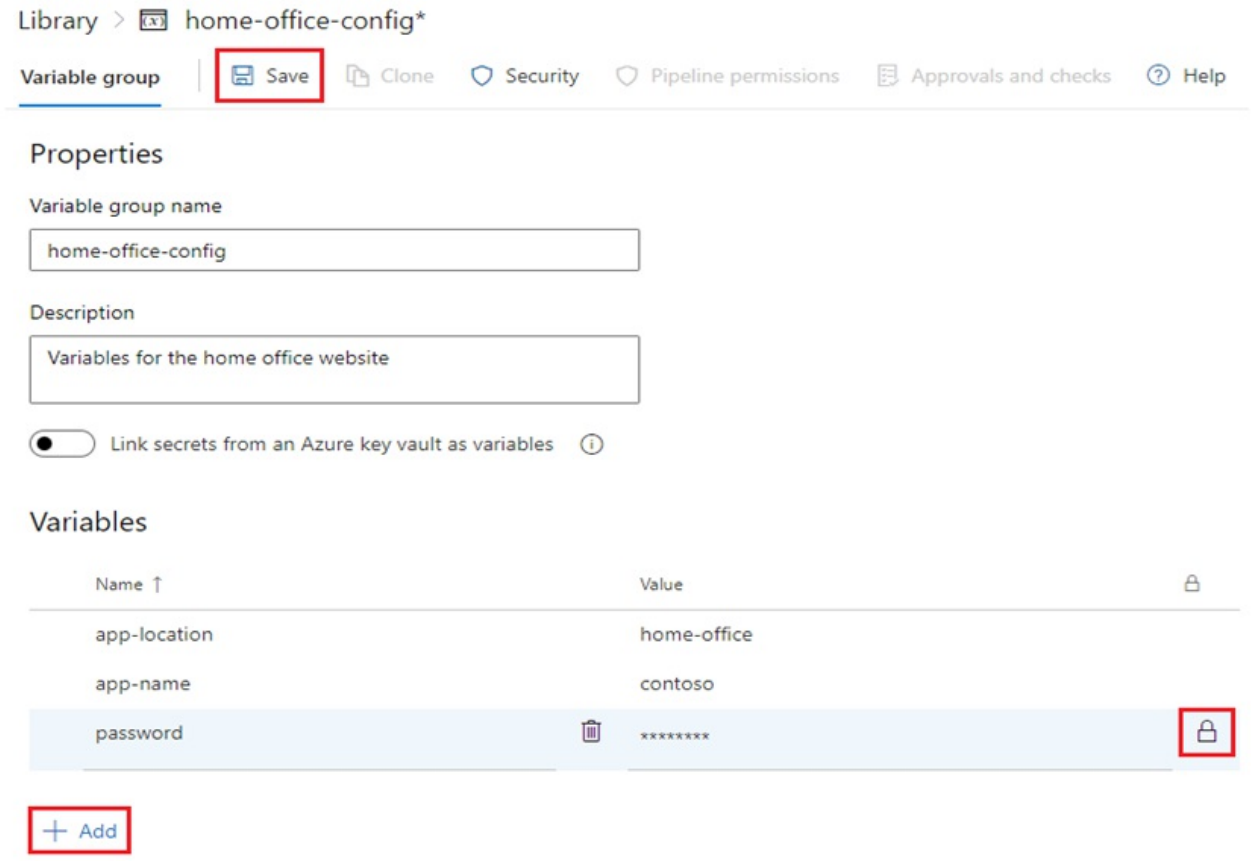


Step 3: On the new variable group page, under Properties, enter a name [Here Group1] and optional description for the variable group.

Step 4: Under Variables, select + Add, and then enter a variable name and value to include in the group. If you want to encrypt and securely store the value, select the lock icon next to the variable.

```
[Add variable + value: serverName: server1]
[Add variable + value: dbName: db1]
```

Step 5: Select + Add to add each new variable. When you finish adding variables, select Save.



Reference:
https://learn.microsoft.com/en-us/azure/devops/pipelines/library/variable-groups

Question: 539 CertyIQ

SIMULATION
-
For Project1, you need to ensure that artifacts, symbols, and attachments are retained for 60 days.

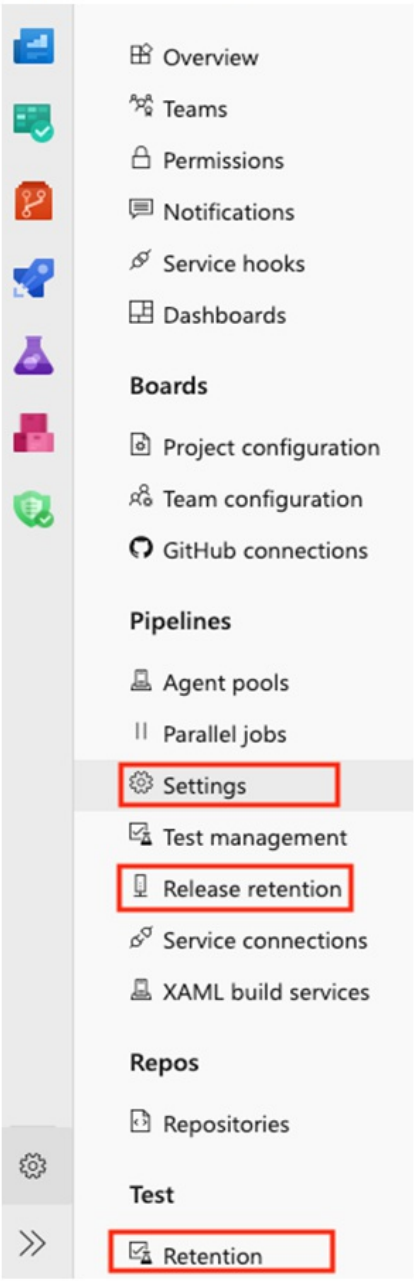
Answer:

Configure retention policies
Select Settings to configure retention policies for runs, artifacts, symbols, attachments, and pull request runs.
Select Release retention to set up your release retention policies and configure when to delete or permanently destroy releases.
Select Retention to set up how long to keep manual and automated test runs.

Step 1: Sign in to your project [Project1].

Step 2: Go to the Settings tab of your project's settings.

Step 3: Select Settings or Release retention under Pipelines or Retention under Test. [Select Settings]



Step 4: Set the number of days to keep artifacts, symbols, and attachments. [Set it to 60]
Set the number of days to keep runs [Skip]
Set the number of days to keep pull request runs [Skip]
Set the number of recent runs to keep for each pipeline [Skip]

Reference:
https://learn.microsoft.com/en-us/azure/devops/pipelines/policies/retention

Question: 540 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You need to use an Azure Pipelines pipeline to test an app. The solution meet the following requirements:
•The pipeline must fail if any tests fail.
•The test results must be published to the pipeline.
•The test for every pipeline run must be triggered unless the pipeline is cancelled.
Solution: You include the following elements in the YAML definition of the pipeline.

```
...
- task: PublishTestResults@2
  displayName: 'Publish Unit Test Results'
  condition: not(canceled())
  inputs:
    testResultsFormat: 'JUnit'
    testResultsFiles: '**/junit.xml'
    failTaskOnFailedTests: true
    testRunTitle: 'App Test'
...
```

Does this meet the goal?

A.Yes
B.No

Answer: A

Explanation:

The line failTaskOnFailedTests: true ensures that the task will fail if any tests fail. This meets the first requirement.Publish Test Results:The task: PublishTestResults@2 indicates that the test results will be published to the pipeline. This meets the second requirement.Trigger Tests on Every Run:The condition: not(canceled()) line is used to prevent the task from running if the pipeline is canceled.

Question: 541 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You need to use an Azure Pipelines pipeline to test an app. The solution meet the following requirements:
•The pipeline must fail if any tests fail.
•The test results must be published to the pipeline.
•The test for every pipeline run must be triggered unless the pipeline is cancelled.
Solution: You include the following elements in the YAML definition of the pipeline.

```
...
- task: PublishTestResults@2
displayName: 'Publish Unit Test Results'
condition: succeededOrFailed()
inputs:
testResultsFormat: 'JUnit'
testResultsFiles: '**/junit.xml'
failTaskOnFailureToPublishResults: true
testRunTitle: 'App Test'
...
```

Does this meet the goal?

A.Yes
B.No

Answer: B

Explanation:

"From the Microsoft Entra admin center, you assign the Cloud Device Administrator role to User1."

Question: 542

CertyIQ

HOTSPOT
-
You have an app named App1 that is built and deployed by using containers.
The Dockerfile for App1 has the following definition.

```
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS base
WORKDIR /app1
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY ["App1.csproj", ""]
RUN dotnet restore "./App1.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "App1.csproj" -c Release -o /app1/build

FROM build AS publish
RUN dotnet publish "App1.csproj" -c Release -o /app1/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "App1.dll"]
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No.
NOTE: Each correct selection is worth one point.

Answer Area

Statements	Yes	No
App1 is an ASP.NET Core app.	○	○
The build uses a debug configuration.	○	○
The Dockerfile uses the Docker multi-stage build feature.	○	○

Answer:

Answer Area

Statements	Yes	No
App1 is an ASP.NET Core app.	○	○
The build uses a debug configuration.	○	○
The Dockerfile uses the Docker multi-stage build feature.	○	○

Explanation:

Yes

No

Yes

Question: 543 CertyIQ

You have an Azure DevOps subscription that contains the projects shown in the following table.

Name	Number of users	Repository	Visibility
Project1	100	Project1 public repository	Public
Project2	5	Project1 public repository	Public
Project3	2	Private GitHub repository	Private
Project4	1,000	Public GitHub repository	Public
Project5	150	Public GitHub repository	Private

You build apps for the projects by using Azure Pipelines.
Which two projects meet the criteria for granting free parallel jobs? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A.Project1
B.Project3
C.Project4
D.Project2
E.Project5

Answer: AD

Explanation:

A.Project1.

D.Project2.

Question: 544 CertyIQ

HOTSPOT

-

You have an Azure Pipelines pipeline that has the following definition.

```
jobs:

- job: A
  steps:
  # job A steps here

- job: B
  dependsOn: []
  steps:
  # job B steps here

- job: C
  dependsOn: []
  steps:
  # job C steps here

- job: D
  dependsOn: C
  steps:
  # job D steps here
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No.
NOTE: Each correct selection is worth one point.

Answer Area

Statements	Yes	No
Job B can run concurrently with job A.	○	○
Job D can run concurrently with job B.	○	○
Job C can run concurrently with job D.	○	○

Answer:

Answer Area

Statements	Yes	No
Job B can run concurrently with job A.	●	○
Job D can run concurrently with job B.	●	○
Job C can run concurrently with job D.	○	●

Question: 545 CertyIQ

You use an Azure Pipelines pipeline to build and test an app named App1.
Your company's development department works in the feature branches.
You need to ensure that a pull request will merge into the main branch only when testing covers more than 90 percent of the code.
What should you do?

A.Configure a branch policy for the feature branches.
B.Configure a branch policy for the main branch.
C.Create a Publish Test Results task,
D.Create a code coverage configuration YAML file.

Answer: B

Explanation:

The correct answer is **B. Configure a branch policy for the main branch.**

Here's why:

The requirement is to ensure that pull requests into the main branch only occur if code coverage is above 90%. Branch policies in Azure DevOps are specifically designed to enforce rules on branches, most often the main branch, before merges are allowed.

Branch Policies: Branch policies are conditions that must be met before a pull request can be completed and merged into a branch. They allow you to protect important branches like main and ensure code quality and stability. Examples include requiring code reviews, successful builds, and in this case, code coverage thresholds. https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops

Why main? The main branch represents the stable, production-ready version of the code. Therefore, it's critical to protect it with rigorous checks. Applying the policy to the main branch will ensure that only tested and sufficiently covered code can be merged.

Configuring the Policy: Within the branch policy for main, you can configure a "Build Validation" policy that runs the pipeline that includes code coverage analysis. You would then set a threshold (90% in this case) within the build validation policy. The pull request will only be able to be completed if that threshold is met.

Let's look at why the other options are incorrect:

A. Configure a branch policy for the feature branches: While branch policies on feature branches are useful for enforcing standards on feature development, they do not directly prevent merges into the main branch if the code coverage doesn't meet the requirement. They control what is allowed in the feature branch itself, not what is being merged into main.
C. Create a Publish Test Results task: A Publish Test Results task simply publishes the test results generated by your tests. It does not enforce a code coverage threshold for merging. This task is necessary to make the test results and coverage available, but it is not sufficient to enforce the policy.
https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/test/publish-test-results?view=azure-devops&tabs=yaml
D. Create a code coverage configuration YAML file: A code coverage configuration YAML file (e.g., .opencover.runsettings) specifies which code to include or exclude from code coverage analysis. It defines the scope of code coverage calculations but doesn't enforce a policy preventing merges if a coverage threshold isn't met. Like the publish test results task, this is a component used within the pipeline, not the enforcement mechanism.

Question: 546 CertyIQ

HOTSPOT
-
You plan to build an app named App1 by using GitHub.
You need to recommend a build and deployment solution for App1. The solution must meet the following requirements:
•Ensure that the code for App1 is submitted for testing only when a build has completed successfully.
•Ensure that a build is approved by the change management team before the build is released.
What should you include in the recommendation for each requirement? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Answer Area

Ensure that the code for App1 is submitted for testing only when a build has completed successfully:

- Environments
- Jobs
- Matrices
- Packages
- Pull requests

Ensure that a build is approved by the change management team before the build is released:

- Environments
- Jobs
- Matrices
- Packages
- Pull requests

Answer:

Answer Area

Ensure that the code for App1 is submitted for testing only when a build has completed successfully:

- Environments
- Jobs
- Matrices
- Packages
- Pull requests

Ensure that a build is approved by the change management team before the build is released:

- Environments
- Jobs
- Matrices
- Packages
- Pull requests

Question: 547

CertyIQ

You have an Azure Pipelines pipeline named Pipeline1 and a user named User1. Pipeline1 contains a temporary final stage named final1.
You need to ensure that User1 can delete final1 when testing is complete. The solution must follow the principle of least privilege.
At which level should you grant permissions to User1?

A.pipeline

B.organization

C.stage

D.project

Answer: A

Explanation:

The correct answer is A: pipeline. Granting permissions at the pipeline level for deleting the temporary stage final1 in Pipeline1 adheres to the principle of least privilege. This is because User1's ability to delete the stage is scoped specifically to that pipeline and that stage.

Here's why other options are less suitable:

B. Organization: Granting permission at the organization level would give User1 broad, unnecessary privileges across all projects and pipelines in the organization. This is a violation of least privilege.

C. Stage: Currently, Azure DevOps does not have a specific permission setting dedicated solely to stage-level permissions that encompasses delete functionality. The available stage-level controls are typically related to approving deployments or viewing logs.

D. Project: Project-level permissions would grant User1 more access than required. It would allow access to multiple pipelines and related artifacts within the project, going against the principle of least privilege.

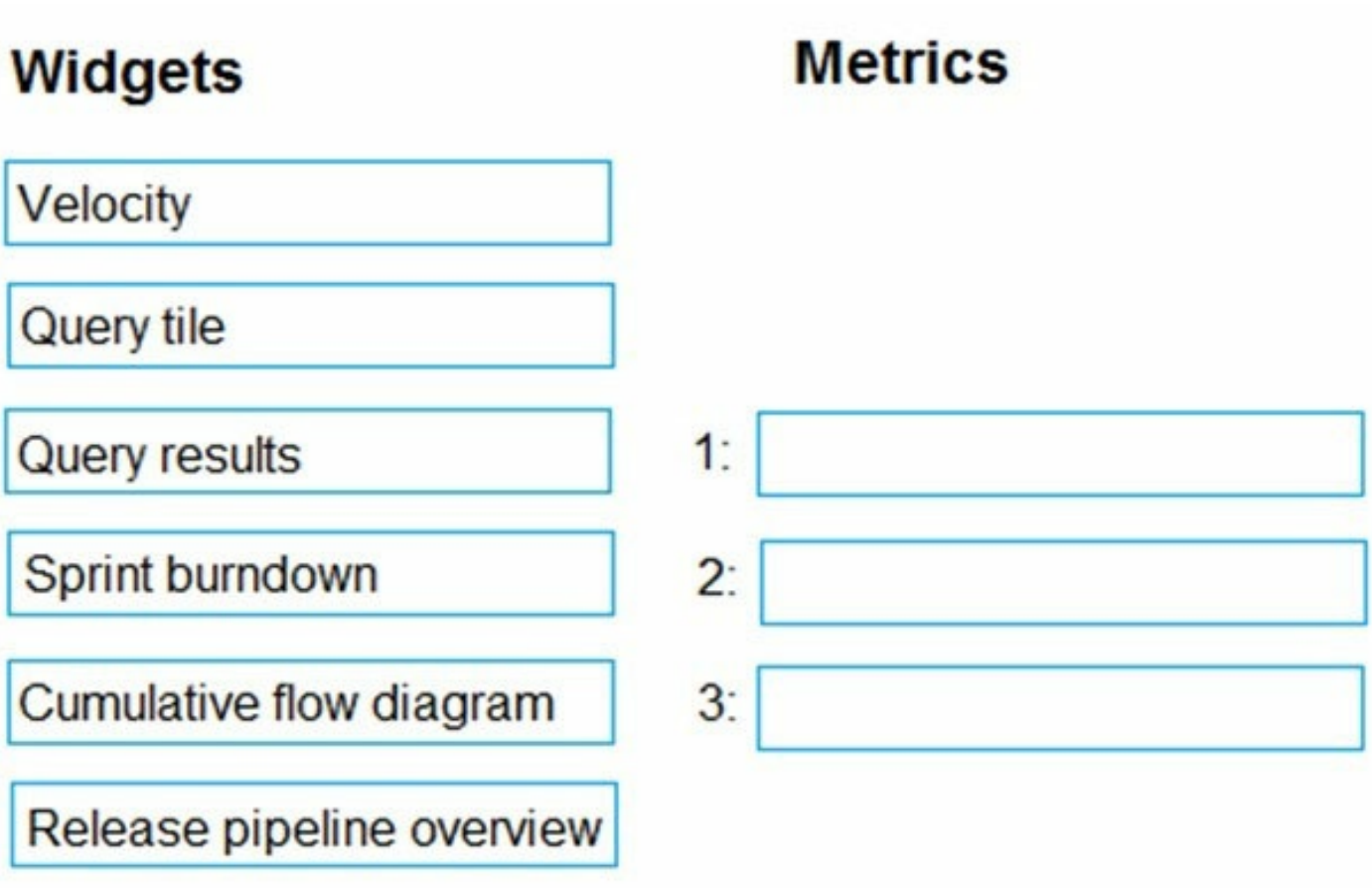
By granting User1 permissions at the pipeline level, we ensure that they can only delete the specified stage (final1) within the scope of Pipeline1. This approach ensures User1 has the minimum necessary permissions to perform the required task without granting them excessive access to other resources. Specifically, users can be granted permissions to manage a particular pipeline, including modification permissions, which allows them to delete stages. To achieve this, User1 would be assigned the role with the necessary permissions directly on the individual pipeline.

Further research:

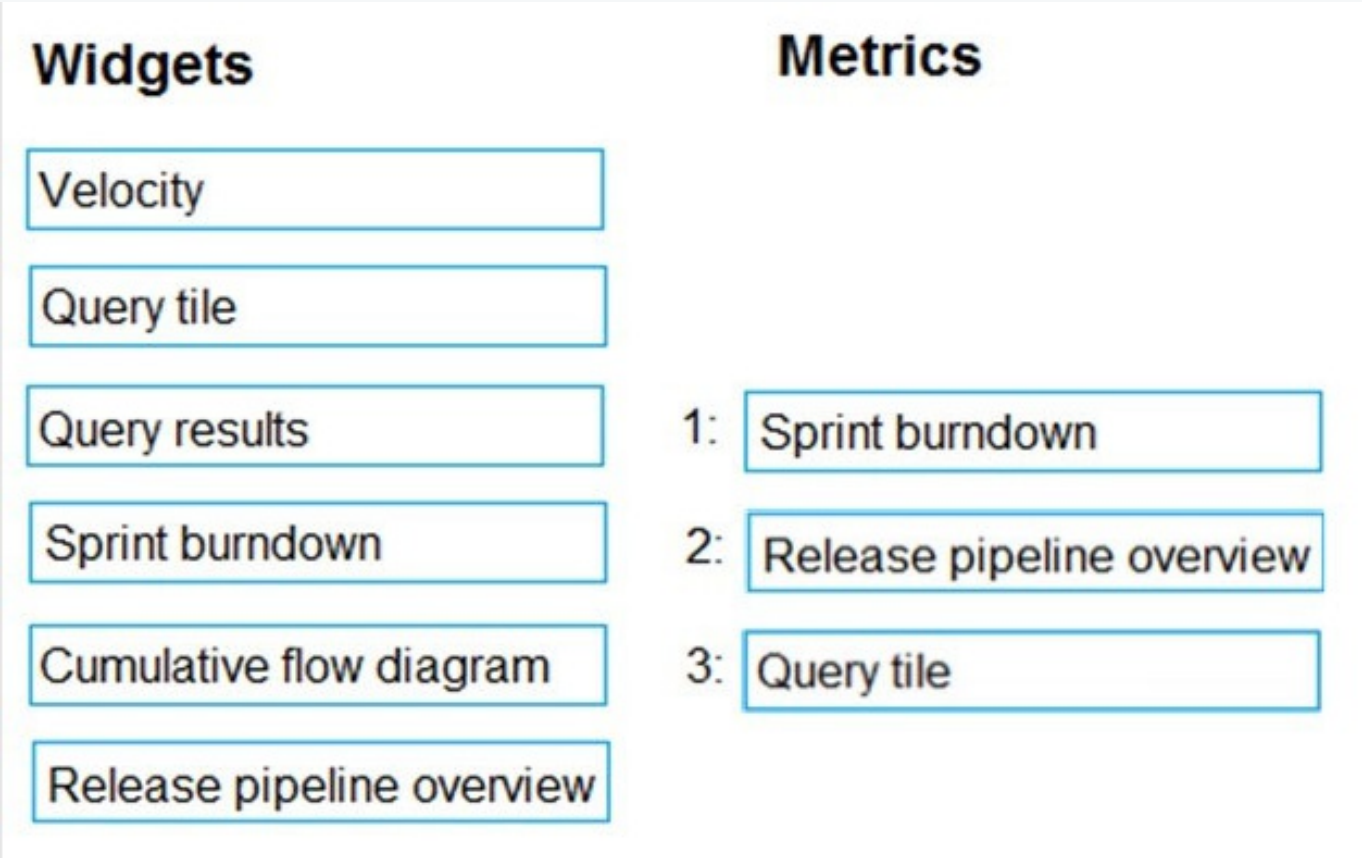
Azure DevOps Permissions: https://learn.microsoft.com/en-us/azure/devops/organizations/security/permissions-access?view=azure-devops
Pipeline Permissions: https://learn.microsoft.com/en-us/azure/devops/pipelines/policies/permissions?view=azure-devops&tabs=yaml

Question: 548 CertyIQ

DRAG DROP -
You are configuring the Azure DevOps dashboard. The solution must meet the technical requirements.
Which widget should you use for each metric? To answer, drag the appropriate widgets to the correct metrics. Each widget may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Sprint burndown

Box 2: Release pipeline overview

Release pipeline overview shows the status of environments in a release definition.

Box 3: Query tile -

Query tile displays the total number of results from a query.

https://docs.microsoft.com/en-us/azure/devops/report/dashboards/team-velocity?view=azure-devops

Question: 549 CertyIQ

You plan to deploy a new database environment. The solution must meet the technical requirements.
You need to prepare the database for the deployment.
How should you format the export?

A.NDF
B.BACPAC
C.DACPAC
D.MDF

Answer: B

Explanation:

The correct answer is B (BACPAC). Here's why:

A BACPAC (Backup Package) is a file format specifically designed for transporting database schema and data. It's self-contained and can be easily used for migrating or deploying databases between different SQL Server instances, including Azure SQL Database. BACPACs are particularly useful when moving a database to or from Azure as they don't require access to the underlying file system like MDF or NDF files.

NDF (Secondary Data File): An NDF file is a secondary data file for a SQL Server database. It cannot be used in isolation and relies on the MDF file.

DACPAC (Data-Tier Application Package): A DACPAC primarily contains database schema (table definitions, stored procedures, etc.) but may or may not include data. It's often used for version control and deploying schema changes. While it can contain data, it's less suitable than a BACPAC if you're focused on moving the entire database, including the data.

MDF (Master Data File): The primary data file for a SQL Server database. Like NDF files, it can't be easily used directly for deployment to Azure without significant modification.

Since the scenario requires deploying a new database environment and preparing the entire database (both schema and data) for deployment, the BACPAC format is the most appropriate choice. It encapsulates everything needed for a smooth transition to the new environment. Also BACPAC is well integrated with Azure deployment tools.

Relevant Links:

BACPAC vs DACPAC: Understanding the Differences
Export a database to a BACPAC file

Question: 550 CertyIQ

You need to meet the technical requirements for monitoring App1.

What should you use?

A.Splunk
B.Azure Application Insights
C.Azure Advisor
D.App Service logs

Answer: B

Explanation:

The correct answer is **B. Azure Application Insights**. Here's a detailed justification:

Azure Application Insights is an Application Performance Management (APM) service within Azure designed specifically for monitoring live web applications. It automatically detects performance anomalies, helps diagnose issues, and provides insights into how users are actually using the application. This aligns perfectly with the need to monitor App1 to meet specific technical requirements related to application performance and user experience.

Application Insights provides rich telemetry data including request rates, response times, exception rates, dependency calls, and page views. This data can be used to identify bottlenecks, understand user behavior, and track the overall health of App1. Its integration with Azure Monitor allows for unified monitoring and alerting across the entire Azure environment.

Splunk (Option A) is a powerful log aggregation and analysis tool but is often overkill and more expensive for basic application monitoring requirements. While Splunk can ingest application logs, it doesn't inherently provide the deep performance insights and automated issue detection that Application Insights offers.

Azure Advisor (Option C) is a tool that provides recommendations to optimize Azure deployments based on cost, security, reliability, operational excellence, and performance, but it is not a dedicated application monitoring tool like Application Insights.

App Service logs (Option D) are useful for basic troubleshooting, but lack the comprehensive monitoring capabilities of Application Insights, particularly for identifying performance issues and understanding user behavior at scale. Application insights provides more robust capabilities around end to end transaction tracing, live metrics, and proactive anomaly detection than App Service logs alone.

Therefore, for monitoring application performance, user behavior, and diagnosing issues within App1 to fulfill technical requirements, Azure Application Insights is the most appropriate and focused choice.

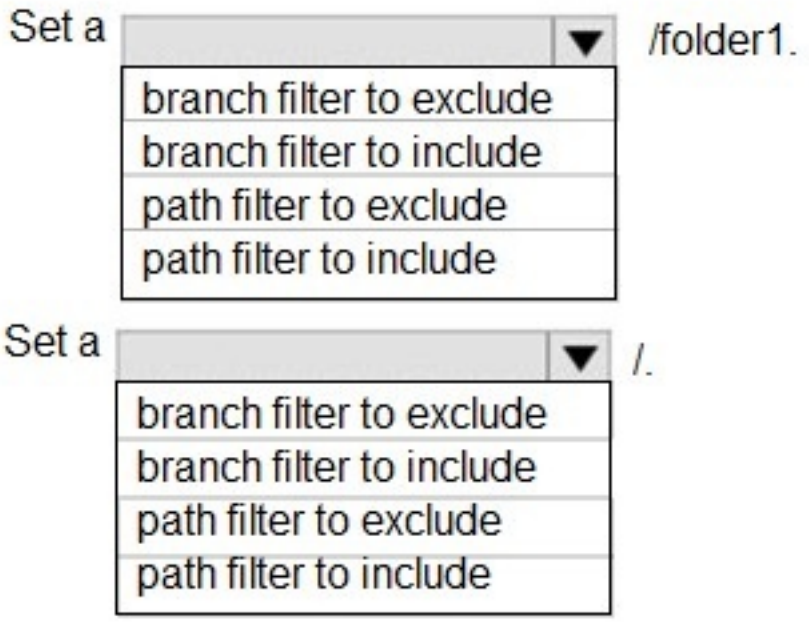
Further reading:

Azure Application Insights Documentation
What is Azure Monitor?

Question: 551 CertyIQ

HOTSPOT -
How should you configure the filters for the Project5 trigger? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

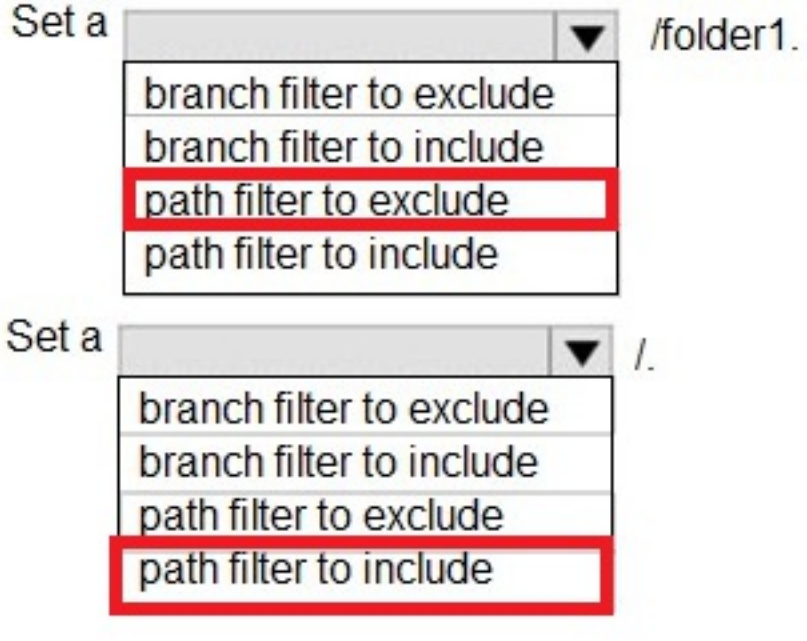
Answer Area



@

Answer:

Answer Area



@

Explanation:

Scenario:

Project5 will contain a Git repository in Azure Reports and a continuous integration trigger that will initiate a build in response to any change except for changes within /folder1 of the repository.

Continuous integration (CI) triggers cause a build to run whenever a push is made to the specified branches or a specified tag is pushed.

Box 2: Path filter to include.

Case Study Description

Contoso Case Study: 2

Overview

Existing Environment

Contoso, Ltd. is a manufacturing company that has a main office in Chicago.

Contoso plans to improve its IT development and operations processes by implementing Azure DevOps principles. Contoso has an Azure subscription and creates an Azure DevOps organization.

The Azure DevOps organization includes:

• The Docker extension

• A deployment pool named Pool7 that contains 10 Azure virtual machines that run Windows Server 2016

The Azure subscription contains an Azure Automation account.

Planned Changes

Contoso plans to create projects in Azure DevOps as shown in the following table.

Project name	Project details
Project 1	Project1 will provide support for incremental builds and third-party SDK components
Project 2	Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.
Project 3	Project3 will be integrated with SonarQube
Project 4	Project4 will provide support for a build pipeline that creates a Docker image and pushes the image to the Azure Container Registry. Project4 will use an existing Dockerfile.
Project 5	Project5 will contain a Git repository in Azure Reports and a continuous integration trigger that will initiate a build in response to any change except for changes within /folder1 of the repository.
Project 6	Project6 will provide support for build and deployment pipelines. Deployment will be allowed only if the number of current work items representing active software bugs is 0.
Project 7	Project7 will contain a target deployment group named Group7 that maps to Pool7. Project7 will use Azure Automation State Configuration to maintain the desired state of the computers in Group7.

Technical Requirements

Contoso identities the following technical requirements:

• Implement build agents rot Project 1.

• Whenever possible, use Azure resources

• Avoid using deprecated technologies

• Implement a code flow strategy for Project2 that will:

• Enable Team 2 to submit pull requests for Project2.

• Enable Team 2 to work independently on changes to a copy of Project?

• Ensure that any intermediary changes performed by Tram2 on a copy of Project2 will be subject to the same restrictions as the ones defied in the build policy of Project2.

• Whenever possible. Implement automation and minimize administrative effort.

• Implement Protect3, Project5, Project6, and Project7 based on the planned changes.

• Implement Project4 and configure the project to push Docker images to Azure Container Reentry.

Question: 552

CertyIQ

In Azure DevOps, you create Project3.
You need to meet the requirements of the project.
What should you do first?

A.From Azure DevOps, modify the build definition.

B.From SonarQube, obtain an authentication token.

C.From Azure DevOps, create a service endpoint.

D.From SonarQube, create a project.

Answer: B

Explanation:

The correct first step is to obtain an authentication token from SonarQube. Before Azure DevOps can interact with SonarQube (for example, to analyze code quality during a build), it needs to be authenticated. The authentication token serves as a credential that allows Azure DevOps to securely communicate with SonarQube. Without this token, any attempt to integrate SonarQube into the Azure DevOps pipeline will fail. Creating a service endpoint in Azure DevOps, modifying a build definition, or creating a SonarQube project are all activities that depend on establishing this authenticated connection. A service endpoint relies on having valid credentials (like the token) to be configured correctly. The build definition needs a service endpoint to reference SonarQube. Therefore, the most immediate prerequisite is to get the authentication token from SonarQube, enabling secure communication and subsequent integration steps. This ensures a smooth and authorized connection between the two platforms.

Azure DevOps Service Connections: https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml
SonarQube Documentation on Security: (Refer to official SonarQube documentation on securing access and generating tokens)

Question: 553 CertyIQ

You need to implement Project4.
What should you do first?

A.Add the FROM instruction in the Dockerfile file.

B.Add a Copy and Publish Build Artifacts task to the build pipeline.

C.Add a Docker task to the build pipeline.

D.Add the MAINTAINER instruction in the Dockerfile file.

Answer: C

Explanation:

The correct first step in implementing Project4, given the context of Azure DevOps Solutions, is to add a Docker task to the build pipeline (option C). Here's why:

1. **Project Context:** We're dealing with Azure DevOps, suggesting the project involves building and deploying applications using Azure services.
2. **Docker Integration:** The question implies Docker is part of the solution, possibly for containerizing the application.
3. **Build Pipeline Workflow:** Before building a Docker image (which the Dockerfile's FROM and MAINTAINER instructions relate to), you need to configure the build pipeline to use Docker.
4. **Docker Task's Role:** The Docker task handles actions like building Docker images, pushing them to a container registry, and running Docker commands. It's the foundational step.
5. **Order of Operations:** You can't copy artifacts until you have built something, and you can't effectively use the Dockerfile instructions until you've told the pipeline to use Docker.
6. **Pipeline Orchestration:** Azure DevOps pipelines orchestrate the build, test, and deployment process.

The Docker task is the entry point for integrating Docker into that orchestration.

7. **Setting the Stage:** The Docker task allows you to define the Docker registry connection, specify the Dockerfile location, and set image tags. These are crucial prerequisites to other Docker-related actions.

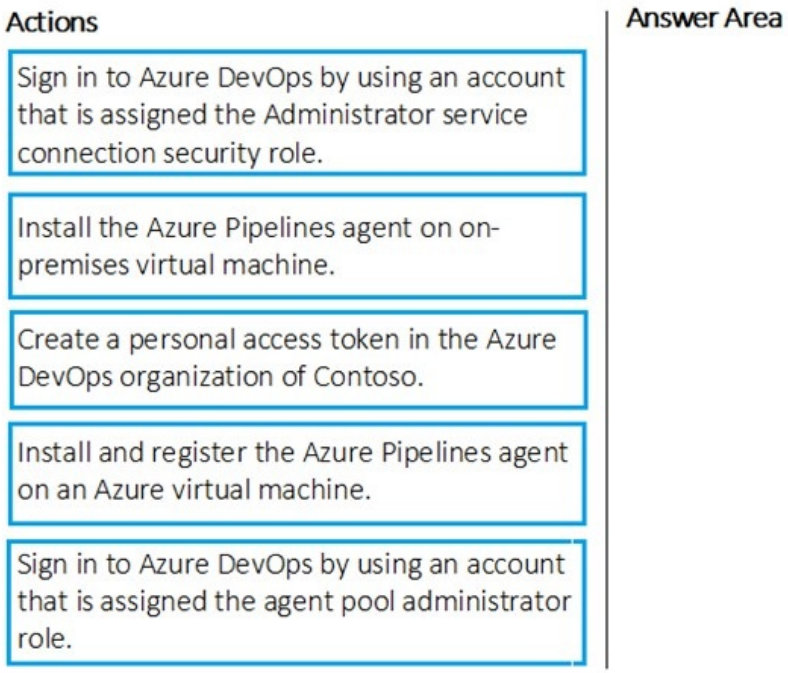
Therefore, adding a Docker task to the build pipeline sets the stage for subsequent steps involving Dockerfiles and artifact handling. It ensures Azure DevOps knows how to interact with Docker to build and potentially deploy the application.

Supporting Resources:

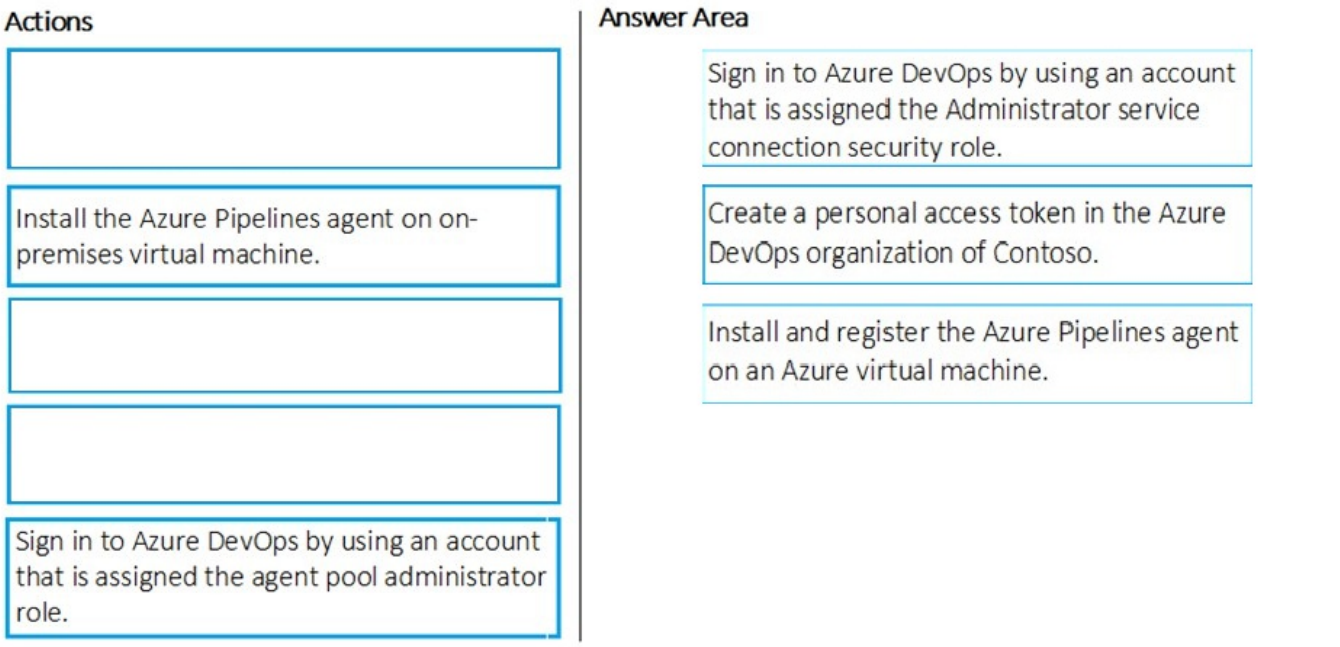
Azure DevOps Pipelines documentation: https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops
Azure DevOps Docker Task: https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/build/docker?view=azure-devops

Question: 554 CertyIQ

DRAG DROP -
You need to recommend a procedure to implement the build agent for Project1.
Which three actions should you recommend be performed in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Scenario:

Project 1	Project1 will provide support for incremental builds and third-party SDK components

Step 1: Sign in to Azure Devops by using an account that is assigned the Administrator service connection security role.

Note: Under Agent Phase, click Deploy Service Fabric Application. Click Docker Settings and then click Configure Docker settings. In Registry Credentials Source, select Azure Resource Manager Service Connection. Then select your Azure subscription.

Step 2: Create a personal access token..

A personal access token or PAT is required so that a machine can join the pool created with the Agent Pools (read, manage) scope.

Step 3: Install and register the Azure Pipelines agent on an Azure virtual machine.

By running a Azure Pipeline agent in the cluster, we make it possible to test any service, regardless of type.

References:

https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-tutorial-deploy-container-app-with-cicd-vsts https://mohitgoyal.co/2019/01/10/run-azure-devops-private-agents-in-kubernetes-clusters/

Implement Continuous Integration

Case Study Description

Contoso Case Study: 2

Overview

Existing Environment

Contoso, Ltd. is a manufacturing company that has a main office in Chicago.

Contoso plans to improve its IT development and operations processes by implementing Azure DevOps principles. Contoso has an Azure subscription and creates an Azure DevOps organization.

The Azure DevOps organization includes:

• The Docker extension

• A deployment pool named Pool7 that contains 10 Azure virtual machines that run Windows Server 2016

The Azure subscription contains an Azure Automation account.

Planned Changes

Contoso plans to create projects in Azure DevOps as shown in the following table.

Project name	Project details
Project 1	Project1 will provide support for incremental builds and third-party SDK components
Project 2	Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.
Project 3	Project3 will be integrated with SonarQube
Project 4	Project4 will provide support for a build pipeline that creates a Docker image and pushes the image to the Azure Container Registry. Project4 will use an existing Dockerfile.
Project 5	Project5 will contain a Git repository in Azure Reports and a continuous integration trigger that will initiate a build in response to any change except for changes within /folder1 of the repository.
Project 6	Project6 will provide support for build and deployment pipelines. Deployment will be allowed only if the number of current work items representing active software bugs is 0.
Project 7	Project7 will contain a target deployment group named Group7 that maps to Pool7. Project7 will use Azure Automation State Configuration to maintain the desired state of the computers in Group7.

Technical Requirements

Contoso identities the following technical requirements:

• Implement build agents rot Project 1.

• Whenever possible, use Azure resources

• Avoid using deprecated technologies

• Implement a code flow strategy for Project2 that will:

• Enable Team 2 to submit pull requests for Project2.

• Enable Team 2 to work independently on changes to a copy of Project?

• Ensure that any intermediary changes performed by Tram2 on a copy of Project2 will be subject to the same restrictions as the ones defied in the build policy of Project2.

• Whenever possible. Implement automation and minimize administrative effort.

• Implement Protect3, Project5, Project6, and Project7 based on the planned changes.

• Implement Project4 and configure the project to push Docker images to Azure Container Reentry.

Question: 555 CertyIQ

You add the virtual machines as managed nodes in Azure Automation State Configuration.
You need to configure the managed computers in Pool7.
What should you do next?

A.Modify the RefreshMode property of the Local Configuration Manager (LCM).
B.Run the Register-AzureRmAutomationDscNode Azure Powershell cmdlet.
C.Modify the ConfigurationMode property of the Local Configuration Manager (LCM).
D.Install PowerShell Core.

Answer: B

Explanation:

The correct answer is **B. Run the Register-AzureRmAutomationDscNode Azure Powershell cmdlet.**

Here's why:

Azure Automation State Configuration uses Desired State Configuration (DSC) to manage the configuration of virtual machines. To effectively manage machines within a specific pool (Pool7 in this case), you need to ensure those machines are properly registered as DSC nodes within your Azure Automation account. This registration process links the virtual machines to the DSC configurations you define within Azure Automation.

The Register-AzureRmAutomationDscNode cmdlet serves precisely this purpose. It registers an Azure virtual machine as a DSC node with Azure Automation State Configuration. By using this cmdlet, you associate the managed computer with your automation account, enabling you to apply configurations defined in DSC to those machines. Once registered, the machines will periodically check in with the pull server in your Automation account to retrieve and apply assigned configurations.

Options A and C relate to the Local Configuration Manager (LCM) settings, but directly modifying RefreshMode or ConfigurationMode wouldn't, by itself, register a node. These settings define how the node pulls configurations, not whether it's registered to pull them. While these LCM settings are important after registration, the initial registration is the crucial first step. Modifying them without registration achieves nothing.

Option D, installing PowerShell Core, might be necessary for other automation tasks but is not the immediate next step for configuring the managed computers within Pool7 for State Configuration. DSC on Azure VMs uses Windows PowerShell (or PowerShell 7+ on newer configurations).

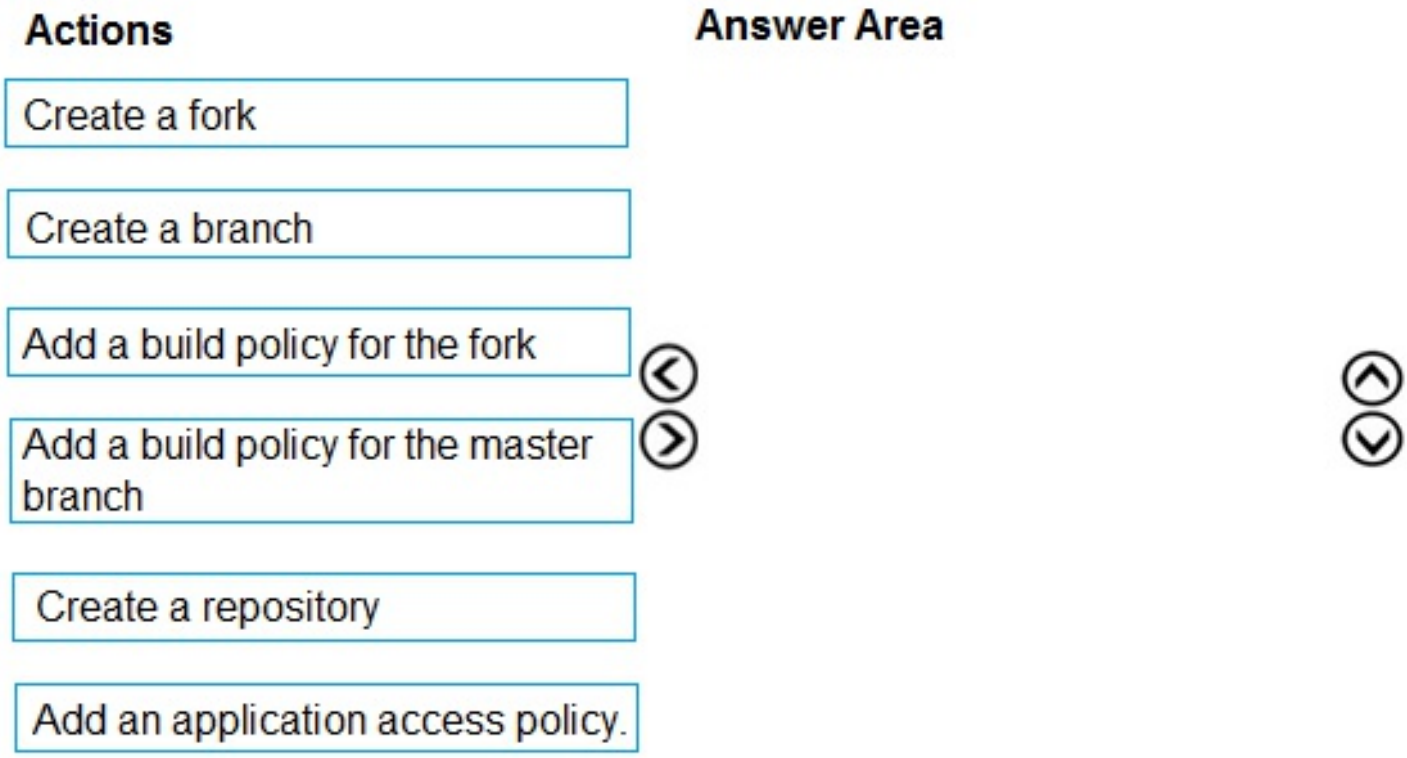
Therefore, the Register-AzureRmAutomationDscNode cmdlet directly addresses the requirement of associating the virtual machines in Pool7 with the Azure Automation State Configuration service so they can be managed via DSC.

Authoritative Links:

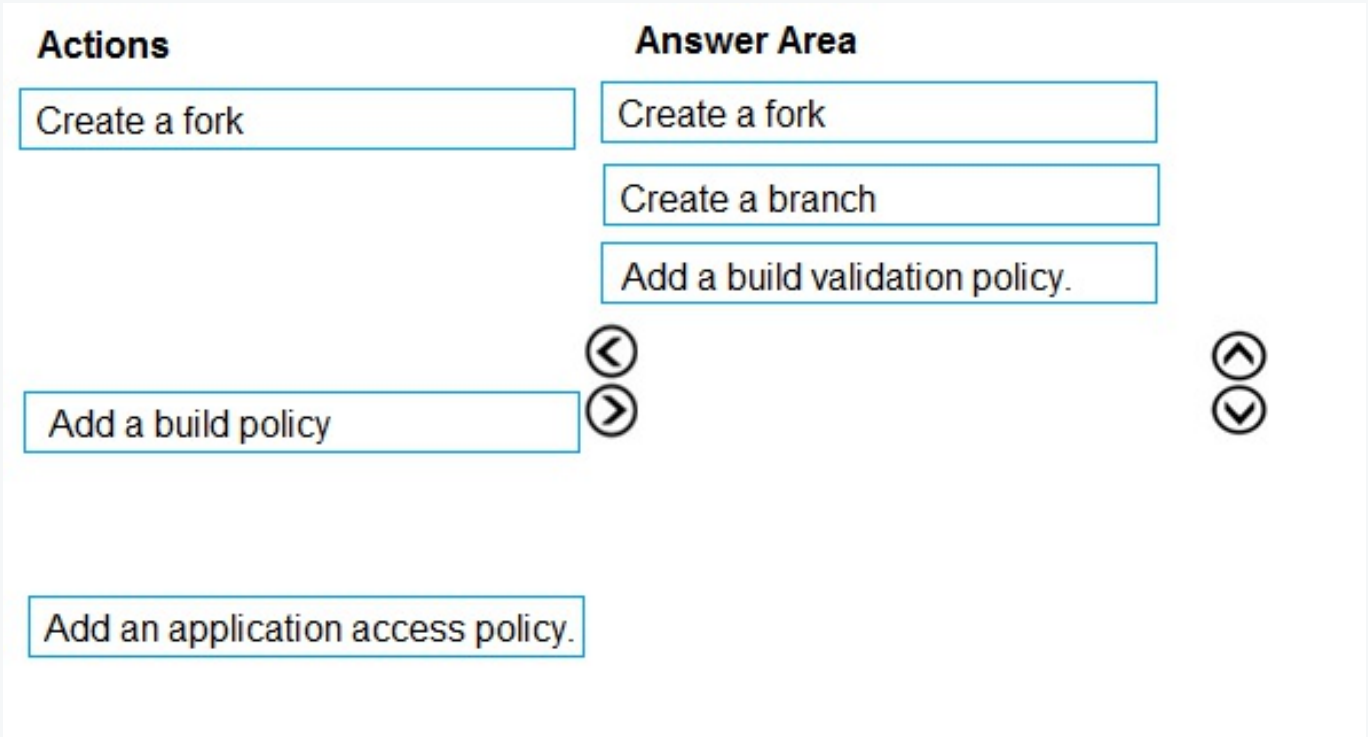
Register-AzureRmAutomationDscNode: https://learn.microsoft.com/en-us/powershell/module/azurerm.automation/register-azurermautomationdscnode?view=azurermps-6.13.0
Azure Automation State Configuration Overview: https://learn.microsoft.com/en-us/azure/automation/automation-dsc-overview

Question: 556 CertyIQ

DRAG DROP -
You need to implement the code flow strategy for Project2 in Azure DevOps.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Step 1: Create a Fork

Step 2: Create a branch -

Branch policies help teams protect their important branches of development. Policies enforce your team's code quality and change management standards.

Step 3: Add a build validation policy

When a build validation policy is enabled, a new build is queued when a new pull request is created or when changes are pushed to an existing pull request targeting this branch. The build policy then evaluates the results of the build to determine whether the pull request can be completed.

Scenario:

Implement a code flow strategy for Project2 that will:

Enable Team2 to submit pull requests for Project2.

▪

✑ Enable Team2 to work independently on changes to a copy of Project2.

✑ Ensure that any intermediary changes performed by Team2 on a copy of Project2 will be subject to the same restrictions as the ones defined in the build policy of Project2.

Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.

Case Study Description

Contoso Case Study: 2

Overview

Existing Environment

Contoso, Ltd. is a manufacturing company that has a main office in Chicago.

Contoso plans to improve its IT development and operations processes by implementing Azure DevOps principles. Contoso has an Azure subscription and creates an Azure DevOps organization.

The Azure DevOps organization includes:

• The Docker extension

• A deployment pool named Pool7 that contains 10 Azure virtual machines that run Windows Server 2016

The Azure subscription contains an Azure Automation account.

Planned Changes

Contoso plans to create projects in Azure DevOps as shown in the following table.

Project name	Project details
Project 1	Project1 will provide support for incremental builds and third-party SDK components
Project 2	Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.
Project 3	Project3 will be integrated with SonarQube
Project 4	Project4 will provide support for a build pipeline that creates a Docker image and pushes the image to the Azure Container Registry. Project4 will use an existing Dockerfile.
Project 5	Project5 will contain a Git repository in Azure Reports and a continuous integration trigger that will initiate a build in response to any change except for changes within /folder1 of the repository.
Project 6	Project6 will provide support for build and deployment pipelines. Deployment will be allowed only if the number of current work items representing active software bugs is 0.
Project 7	Project7 will contain a target deployment group named Group7 that maps to Pool7. Project7 will use Azure Automation State Configuration to maintain the desired state of the computers in Group7.

Technical Requirements

Contoso identities the following technical requirements:

• Implement build agents rot Project 1.

• Whenever possible, use Azure resources

• Avoid using deprecated technologies

• Implement a code flow strategy for Project2 that will:

• Enable Team 2 to submit pull requests for Project2.

• Enable Team 2 to work independently on changes to a copy of Project?

• Ensure that any intermediary changes performed by Tram2 on a copy of Project2 will be subject to the same restrictions as the ones defied in the build policy of Project2.

• Whenever possible. Implement automation and minimize administrative effort.

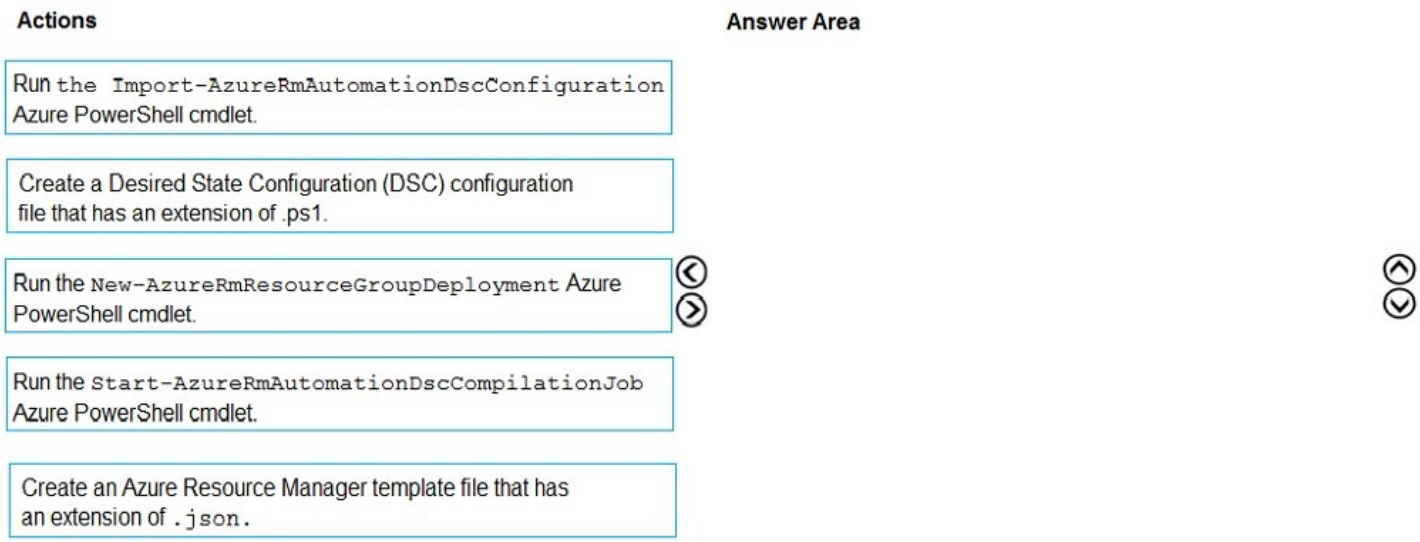
• Implement Protect3, Project5, Project6, and Project7 based on the planned changes.

• Implement Project4 and configure the project to push Docker images to Azure Container Reentry.

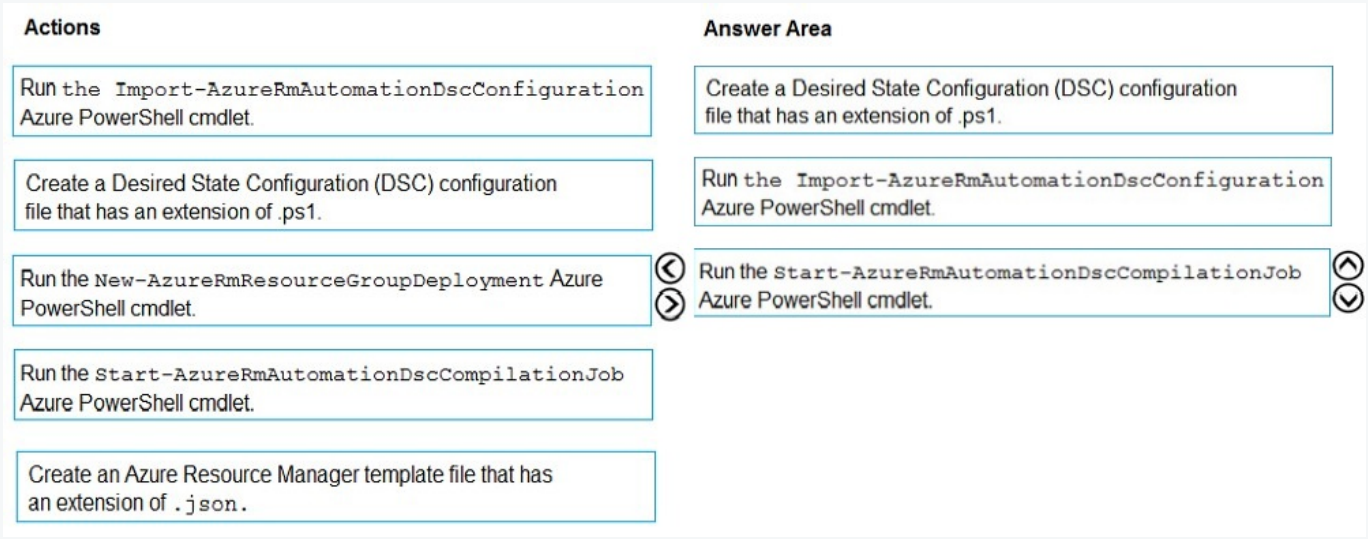
Question: 557

CertyIQ

DRAG DROP -
You need to configure Azure Automation for the computers in Pool7.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Step 1: Create a Desired State Configuration (DSC) configuration file that has an extension of .ps1.

Step 2: Run the Import-AzureRmAutomationDscConfiguration Azure Powershell cmdlet

The Import-AzureRmAutomationDscConfiguration cmdlet imports an APS Desired State Configuration (DSC) configuration into Azure Automation. Specify the path of an APS script that contains a single DSC configuration.

Example:

PS C:\>Import-AzureRmAutomationDscConfiguration -AutomationAccountName "Contoso17"-ResourceGroupName "ResourceGroup01" -SourcePath "C:\DSC

\client.ps1" -Force

This command imports the DSC configuration in the file named client.ps1 into the Automation account named Contoso17. The command specifies the Force parameter. If there is an existing DSC configuration, this command replaces it.

Step 3: Run the Start-AzureRmAutomationDscCompilationJob Azure Powershell cmdlet

The Start-AzureRmAutomationDscCompilationJob cmdlet compiles an APS Desired State Configuration (DSC) configuration in Azure Automation.

Reference:

https://docs.microsoft.com/en-us/powershell/module/azurerm.automation/import-azurermautomationdscconfiguration https://docs.microsoft.com/en-us/powershell/module/azurerm.automation/start-azurermautomationdsccompilationjob

---

Case Study Description

Contoso Case Study: 2

Overview

Existing Environment

Contoso, Ltd. is a manufacturing company that has a main office in Chicago.

Contoso plans to improve its IT development and operations processes by implementing Azure DevOps principles. Contoso has an Azure subscription and creates an Azure DevOps organization.

The Azure DevOps organization includes:

• The Docker extension

• A deployment pool named Pool7 that contains 10 Azure virtual machines that run Windows Server 2016

The Azure subscription contains an Azure Automation account.

Planned Changes

Contoso plans to create projects in Azure DevOps as shown in the following table.

Project name	Project details
Project 1	Project1 will provide support for incremental builds and third-party SDK components
Project 2	Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.
Project 3	Project3 will be integrated with SonarQube
Project 4	Project4 will provide support for a build pipeline that creates a Docker image and pushes the image to the Azure Container Registry. Project4 will use an existing Dockerfile.
Project 5	Project5 will contain a Git repository in Azure Reports and a continuous integration trigger that will initiate a build in response to any change except for changes within /folder1 of the repository.
Project 6	Project6 will provide support for build and deployment pipelines. Deployment will be allowed only if the number of current work items representing active software bugs is 0.
Project 7	Project7 will contain a target deployment group named Group7 that maps to Pool7. Project7 will use Azure Automation State Configuration to maintain the desired state of the computers in Group7.

Technical Requirements

Contoso identities the following technical requirements:

• Implement build agents rot Project 1.

• Whenever possible, use Azure resources

• Avoid using deprecated technologies

• Implement a code flow strategy for Project2 that will:

• Enable Team 2 to submit pull requests for Project2.

• Enable Team 2 to work independently on changes to a copy of Project?

• Ensure that any intermediary changes performed by Tram2 on a copy of Project2 will be subject to the same restrictions as the ones defied in the build policy of Project2.

• Whenever possible. Implement automation and minimize administrative effort.

• Implement Protect3, Project5, Project6, and Project7 based on the planned changes.

• Implement Project4 and configure the project to push Docker images to Azure Container Reentry.

Question: 558 CertyIQ

HOTSPOT -
How should you configure the release retention policy for the investment planning depletions suite? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

Required secrets:

- Certificate
- Personal access token
- Shared Access Authorization token
- Username and password

Storage location:

- Azure Data Lake
- Azure Key Vault
- Azure Storage with HTTPS access
- Azure Storage with HTTP access

Answer:

Answer Area

Required secrets:

- Certificate
- Personal access token
- Shared Access Authorization token
- Username and password

Storage location:

- Azure Data Lake
- Azure Key Vault
- Azure Storage with HTTPS access
- Azure Storage with HTTP access

Explanation:

Box 1: Shared Access Authorization token

Every request made against a storage service must be authorized, unless the request is for a blob or container resource that has been made available for public or signed access. One option for authorizing a request is by using Shared Key.

Box 2: Azure Storage with HTTPS access

Scenario: The mobile applications must be able to call the share pricing service of the existing retirement fund management system. Until the system is upgraded, the service will only support basic authentication over HTTPS.

The investment planning application suite will include one multi-tier web application and two iOS mobile application. One mobile application will be used by employees; the other will be used by customers.

Reference:

https://docs.microsoft.com/en-us/rest/api/storageservices/authorize-with-shared-key

---

Case Study Description

Litware inc. Case Study: 1

Overview

Existing Environment

Litware, Inc. an independent software vendor (ISV) Litware has a main office and five branch offices.

Application Architecture

The company' s primary application is a single monolithic retirement fund management system based on ASP.NE T web forms that use logic written in V8.NET. Some new sections of the application are written in C#.

Variations of the application are created for individual customers. Currently, there are more than 80 have code branches in the application's code base.

The application was developed by using Microsoft Visual Studio. Source code is stored in Team Foundation Server (TFS) in the main office. The branch offices access of the source code by using TFS proxy servers.

Architectural Issues

Litware focuses on writing new code for customers. No resources are provided to refactor or remove existing code. Changes to the code base take a long time, AS dependencies are not obvious to individual developers.

Merge operations of the code often take months and involve many developers. Code merging frequently introduces bugs that are difficult to locate and resolve.

Customers report that ownership costs of the retirement fund management system increase continually. The need to merge unrelated code makes even minor code changes expensive.

Requirements

Planned Changes

Litware plans to develop a new suite of applications for investment planning. The investment planning

Applications will require only minor integration with the easting retirement fund management system.

The investment planning applications suite will include one multi-tier web application and two iOS mobile applications. One mobile application will be used by employees; the other will be used by customers.

Litware plans to move to a more agile development methodology. Shared code will be extracted into a series of package.

Litware has started an internal cloud transformation process and plans to use cloud based services whenever suitable.

Litware wants to become proactive m detecting failures, rather than always waning for customer bug reports.

Technical Requirements

The company's investment planning applications suite must meet the following technical requirements:

• New incoming connections through the firewall must be minimized.

• Members of a group named Developers must be able to install packages.

• The principle of least privilege must be used for all permission assignments

• A branching strategy that supports developing new functionality in isolation must be used.

• Members of a group named Team leaders must be able to create new packages and edit the permissions of package feeds

• Visual Studio App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• By default, all App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• Code quality and release quality are critical. During release, deployments must not proceed between stages if any active bugs are logged against the release.

• The mobile applications must be able to call the share pricing service of the existing retirement fund management system. Until the system is upgraded, the service will only support basic authentication over HUPS.

• The required operating system configuration tor the test servers changes weekly. Azure Automation State Configuration must be used to ensure that the operating system on each test servers configured the same way when the servers are created and checked periodically.

Current Technical

The test servers are configured correctly when first deployed, but they experience configuration drift over time. Azure Automation State Configuration fails to correct the configurations.

Azure Automation State Configuration nodes are registered by using the following command.

```
Register-AzureRmAutomationDscNode
  -ResourceGroupName 'TestResourceGroup'
  -AutomationAccountName 'LitwareAutomationAccount'
  -AzureVMName $vname
  -ConfigurationMode 'ApplyOnly'
```

Question: 559 CertyIQ

HOTSPOT -
You need to configure a cloud service to store the secrets required by the mobile applications to call the share pricing service.
What should you include in the solution? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

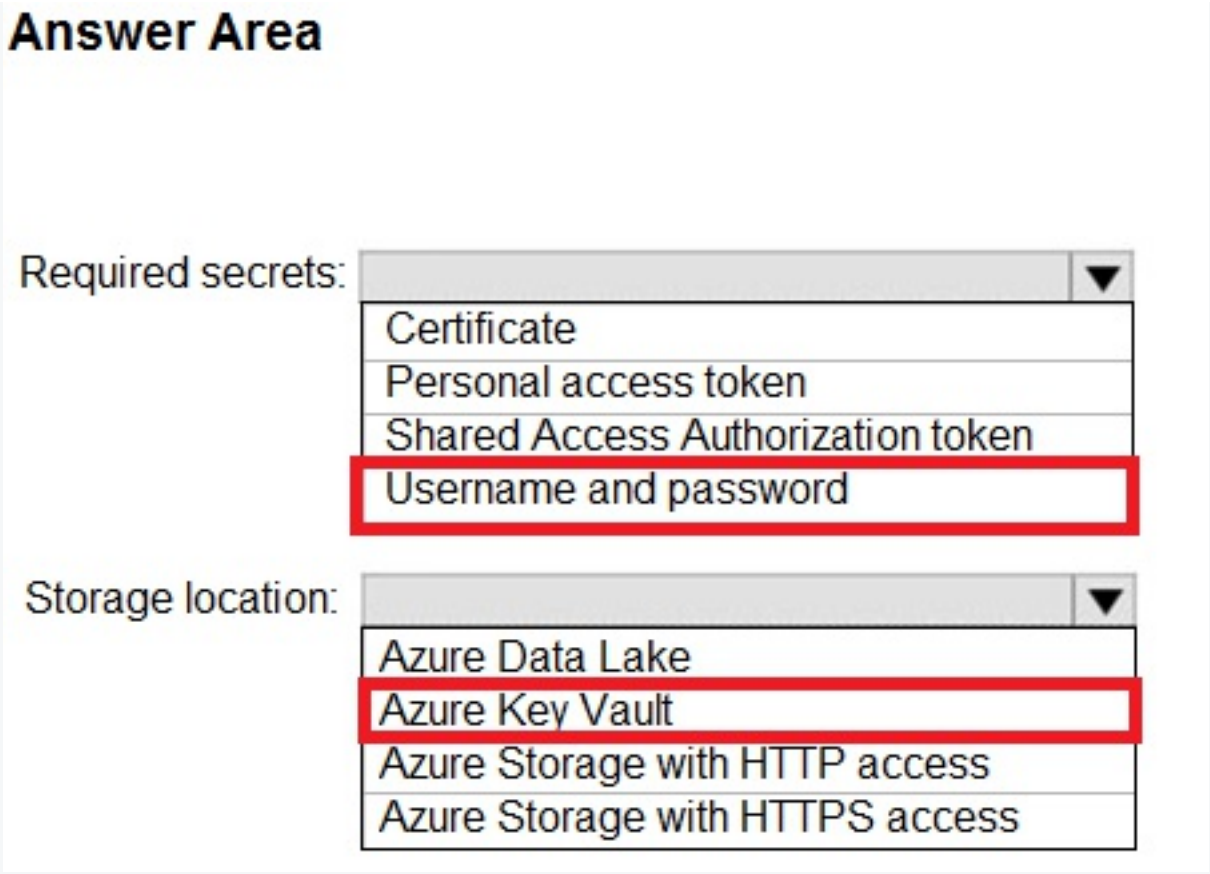
Answer Area

Required secrets:
- Certificate
- Personal access token
- Shared Access Authorization token
- Username and password

Storage location:
- Azure Data Lake
- Azure Key Vault
- Azure Storage with HTTP access
- Azure Storage with HTTPS access

Answer:

Answer Area



Explanation:

1. Username and Password

Username and Password

The source system (e.g., SQL database, FTP server, on-prem system) requires authentication using a username and password.

The service making the connection cannot use OAuth, MSI (Managed Identity), or certificates.

2. Azure Key vault

Azure Key Vault is the enterprise-grade, centralized, secure secret-storage solution for Azure.

Case Study Description

Litware inc. Case Study: 1

Overview

Existing Environment

Litware, Inc. an independent software vendor (ISV) Litware has a main office and five branch offices.

Application Architecture

The company' s primary application is a single monolithic retirement fund management system based on ASP.NE T web forms that use logic written in V8.NET. Some new sections of the application are written in C#.

Variations of the application are created for individual customers. Currently, there are more than 80 have code branches in the application's code base.

The application was developed by using Microsoft Visual Studio. Source code is stored in Team Foundation Server (TFS) in the main office. The branch offices access of the source code by using TFS proxy servers.

Architectural Issues

Litware focuses on writing new code for customers. No resources are provided to refactor or remove existing code. Changes to the code base take a long time, AS dependencies are not obvious to individual developers.

Merge operations of the code often take months and involve many developers. Code merging frequently introduces bugs that are difficult to locate and resolve.

Customers report that ownership costs of the retirement fund management system increase continually. The need to merge unrelated code makes even minor code changes expensive.

Requirements

Planned Changes

Litware plans to develop a new suite of applications for investment planning. The investment planning Applications will require only minor integration with the easting retirement fund management system.

The investment planning applications suite will include one multi-tier web application and two iOS mobile applications. One mobile application will be used by employees; the other will be used by customers.

Litware plans to move to a more agile development methodology. Shared code will be extracted into a series of package.

Litware has started an internal cloud transformation process and plans to use cloud based services whenever suitable.

Litware wants to become proactive m detecting failures, rather than always waning for customer bug reports.

Technical Requirements

The company's investment planning applications suite must meet the following technical requirements:

• New incoming connections through the firewall must be minimized.

• Members of a group named Developers must be able to install packages.

• The principle of least privilege must be used for all permission assignments

• A branching strategy that supports developing new functionality in isolation must be used.

• Members of a group named Team leaders must be able to create new packages and edit the permissions of package feeds

• Visual Studio App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• By default, all App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• Code quality and release quality are critical. During release, deployments must not proceed between stages if any active bugs are logged against the release.

• The mobile applications must be able to call the share pricing service of the existing retirement fund management system. Until the system is upgraded, the service will only support basic authentication over

HUPS.

• The required operating system configuration tor the test servers changes weekly. Azure Automation State Configuration must be used to ensure that the operating system on each test servers configured the same way when the servers are created and checked periodically.

Current Technical

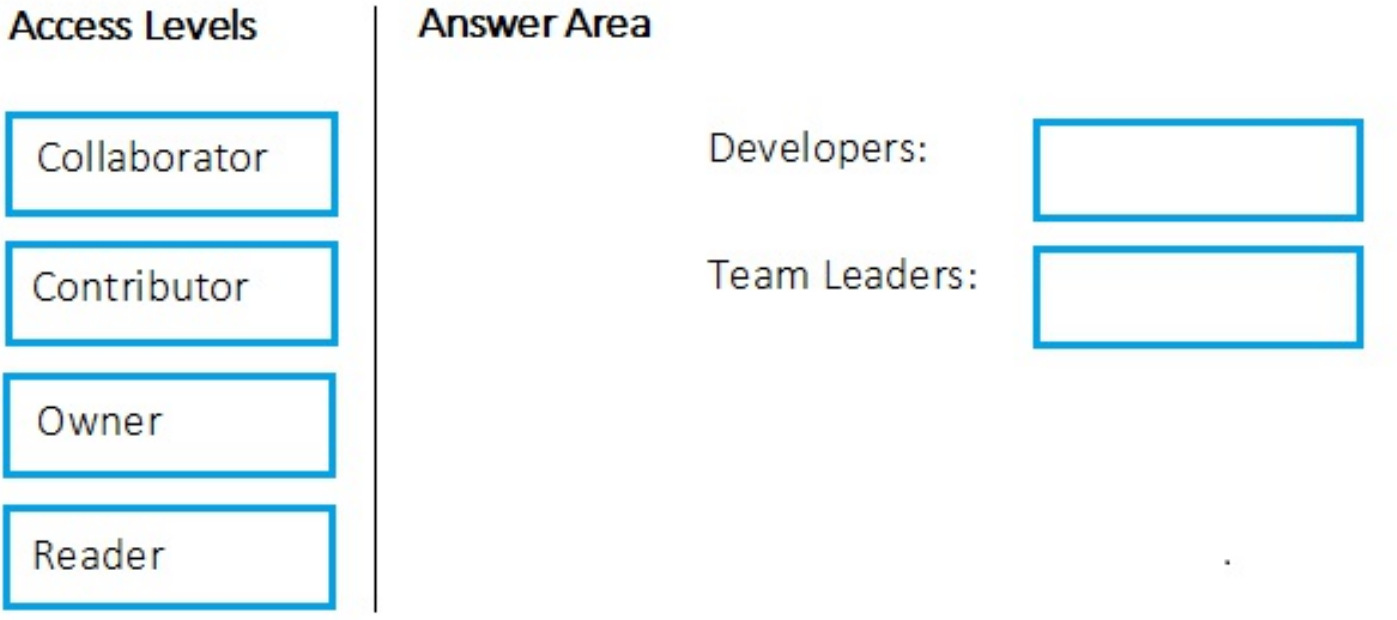
The test servers are configured correctly when first deployed, but they experience configuration drift over time. Azure Automation State Configuration fails to correct the configurations.

Azure Automation State Configuration nodes are registered by using the following command.

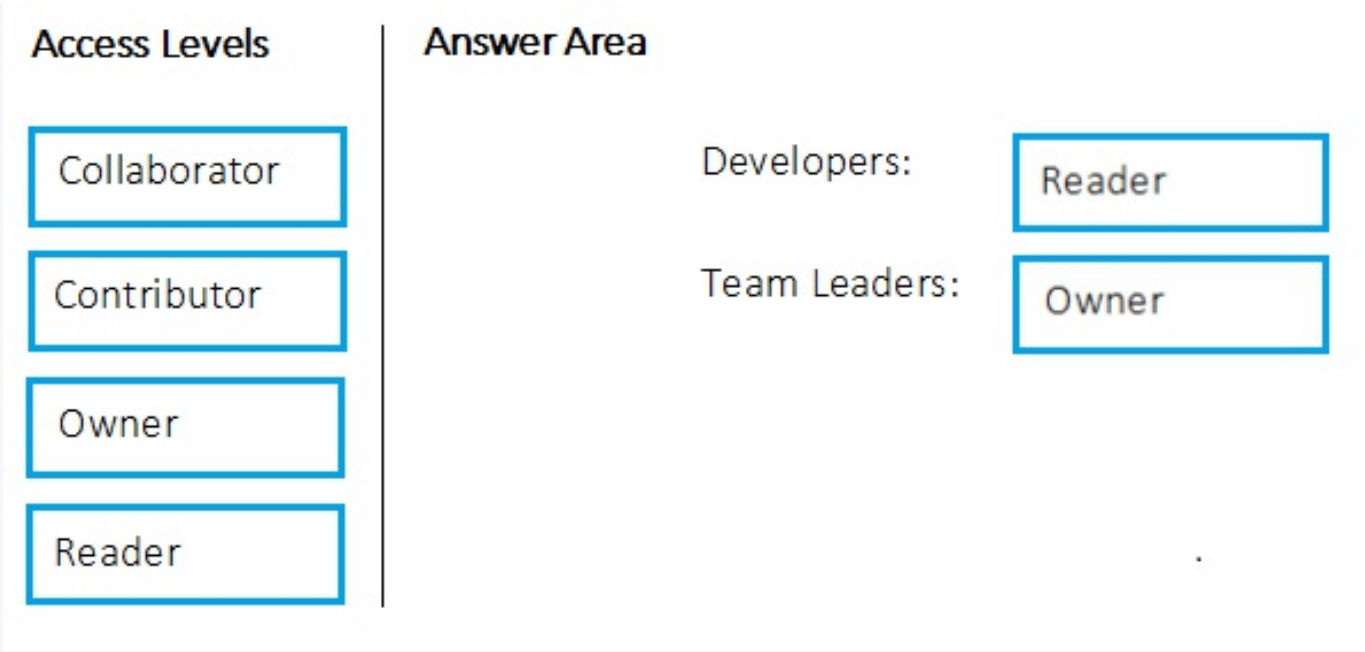
```
Register-AzureRmAutomationDscNode
  -ResourceGroupName 'TestResourceGroup'
  -AutomationAccountName 'LitwareAutomationAccount'
  -AzureVMName $vname
  -ConfigurationMode 'ApplyOnly'
```

Question: 560 CertyIQ

DRAG DROP -
Which package feed access levels should be assigned to the Developers and Team Leaders groups for the investment planning applications suite? To answer, drag the appropriate access levels to the correct groups. Each access level may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Reader -

Members of a group named Developers must be able to install packages.

Feeds have four levels of access: Owners, Contributors, Collaborators, and Readers. Owners can add any type of identity-individuals, teams, and groups-to any access level.

Box 2: Owner -

Members of a group named Team Leaders must be able to create new packages and edit the permissions of package feeds.

Permission	Reader	Collaborator	Contributor	Owner
List and restore/install packages	✓	✓	✓	✓
Save packages from upstream sources		✓	✓	✓
Push packages			✓	✓
Unlist/deprecate packages			✓	✓
Delete/unpublish package				✓
Edit feed permissions				✓
Rename and delete feed				✓

Case Study Description

Litware inc. Case Study: 1

Overview

Existing Environment

Litware, Inc. an independent software vendor (ISV) Litware has a main office and five branch offices.

Application Architecture

The company' s primary application is a single monolithic retirement fund management system based on ASP.NE T web forms that use logic written in V8.NET. Some new sections of the application are written in C#.

Variations of the application are created for individual customers. Currently, there are more than 80 have code branches in the application's code base.

The application was developed by using Microsoft Visual Studio. Source code is stored in Team Foundation Server (TFS) in the main office. The branch offices access of the source code by using TFS proxy servers.

Architectural Issues

Litware focuses on writing new code for customers. No resources are provided to refactor or remove existing code. Changes to the code base take a long time, AS dependencies are not obvious to individual developers.

Merge operations of the code often take months and involve many developers. Code merging frequently introduces bugs that are difficult to locate and resolve.

Customers report that ownership costs of the retirement fund management system increase continually. The need to merge unrelated code makes even minor code changes expensive.

Requirements

Planned Changes

Litware plans to develop a new suite of applications for investment planning. The investment planning Applications will require only minor integration with the easting retirement fund management system.

The investment planning applications suite will include one multi-tier web application and two iOS mobile applications. One mobile application will be used by employees; the other will be used by customers.

Litware plans to move to a more agile development methodology. Shared code will be extracted into a series of package.

Litware has started an internal cloud transformation process and plans to use cloud based services whenever suitable.

Litware wants to become proactive m detecting failures, rather than always waning for customer bug reports.

Technical Requirements

The company's investment planning applications suite must meet the following technical requirements:

• New incoming connections through the firewall must be minimized.

• Members of a group named Developers must be able to install packages.

• The principle of least privilege must be used for all permission assignments

• A branching strategy that supports developing new functionality in isolation must be used.

• Members of a group named Team leaders must be able to create new packages and edit the permissions of package feeds

• Visual Studio App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• By default, all App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• Code quality and release quality are critical. During release, deployments must not proceed between stages if any active bugs are logged against the release.

• The mobile applications must be able to call the share pricing service of the existing retirement fund management system. Until the system is upgraded, the service will only support basic authentication over HUPS.

• The required operating system configuration tor the test servers changes weekly. Azure Automation State Configuration must be used to ensure that the operating system on each test servers configured the same way when the servers are created and checked periodically.

Current Technical

The test servers are configured correctly when first deployed, but they experience configuration drift over time. Azure Automation State Configuration fails to correct the configurations.

Azure Automation State Configuration nodes are registered by using the following command.

```
Register-AzureRmAutomationDscNode
  -ResourceGroupName 'TestResourceGroup'
  -AutomationAccountName 'LitwareAutomationAccount'
  -AzureVMName $vname
  -ConfigurationMode 'ApplyOnly'
```

Question: 561

CertyIQ

You need to meet the technical requirements for controlling access to Azure DevOps.
What should you use?

A.Azure Multi-Factor Authentication (MFA)
B.on-premises firewall rules
C.conditional access policies in Azure AD
D.Azure role-based access control (Azure RBAC)

Answer: C

Explanation:

The correct answer is C: conditional access policies in Azure AD. Conditional Access is a powerful tool in

Azure Active Directory (Azure AD) that allows you to enforce organizational policies when users try to access Azure DevOps. It evaluates multiple signals like user identity, location, device health, application, and real-time risk to grant or deny access, or require multi-factor authentication. This aligns perfectly with the requirement of controlling access, allowing granular rules to be implemented based on various conditions.

A. Azure Multi-Factor Authentication (MFA) is a component that can be required by conditional access policies but doesn't, on its own, define rules for controlling access based on specific conditions. It's a single factor that improves security but isn't a holistic access control solution.

B. On-premises firewall rules are irrelevant to controlling access to Azure DevOps, which is a cloud-based service. These firewalls manage network traffic within an on-premises environment, not to external cloud services.

D. Azure role-based access control (Azure RBAC) is used to manage permissions within Azure resources. While Azure DevOps utilizes permissions for granting access to repositories, pipelines, and other features, Azure RBAC generally applies to management plane operations of Azure resources, not user access to a SaaS like Azure DevOps itself. Conditional access focuses on how a user authenticates and gains initial access before RBAC applies to manage what they can do after they're authenticated.

In summary, Conditional Access is the correct choice because it offers the most comprehensive and flexible solution for controlling access to Azure DevOps based on various conditions and signals, meeting the specific requirements of the question.

Relevant Documentation:

What is Conditional Access in Azure Active Directory?: https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/overview
Common Conditional Access policies: https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/howto-conditional-access-policy-common

Question: 562

CertyIQ

You need to configure Azure Pipelines to control App2 builds.
Which authentication method should you use?

A.Windows NTLM
B.certificate
C.SAML
D.personal access token (PAT)

Answer: D

Explanation:

The correct answer is D, using a Personal Access Token (PAT) for Azure Pipelines authentication. Here's why:

A PAT is a string used to authenticate users within Azure DevOps. It provides an alternative to usernames and passwords. When configuring automated processes like builds in Azure Pipelines to interact with Azure DevOps, a PAT offers a secure and manageable way to grant permissions. Unlike Windows NTLM (A), which is primarily for Windows-based authentication and less suitable for cross-platform build systems and cloud-based services, PATs are specifically designed for Azure DevOps interaction. Certificates (B) are more relevant for authenticating services or devices, not individual users or build processes within Azure DevOps. SAML (C) is used for federated identity and SSO, suitable for user authentication at a higher level, but not

ideal for build pipeline authentication.

PATs allow for granular control over permissions. The PAT can be scoped to specific resources, pipelines, or tasks, ensuring that the build process only has the necessary access. This minimizes the risk of exposing sensitive data or allowing unintended actions. In contrast, a generic user account password being used across multiple pipelines would represent a higher security risk.

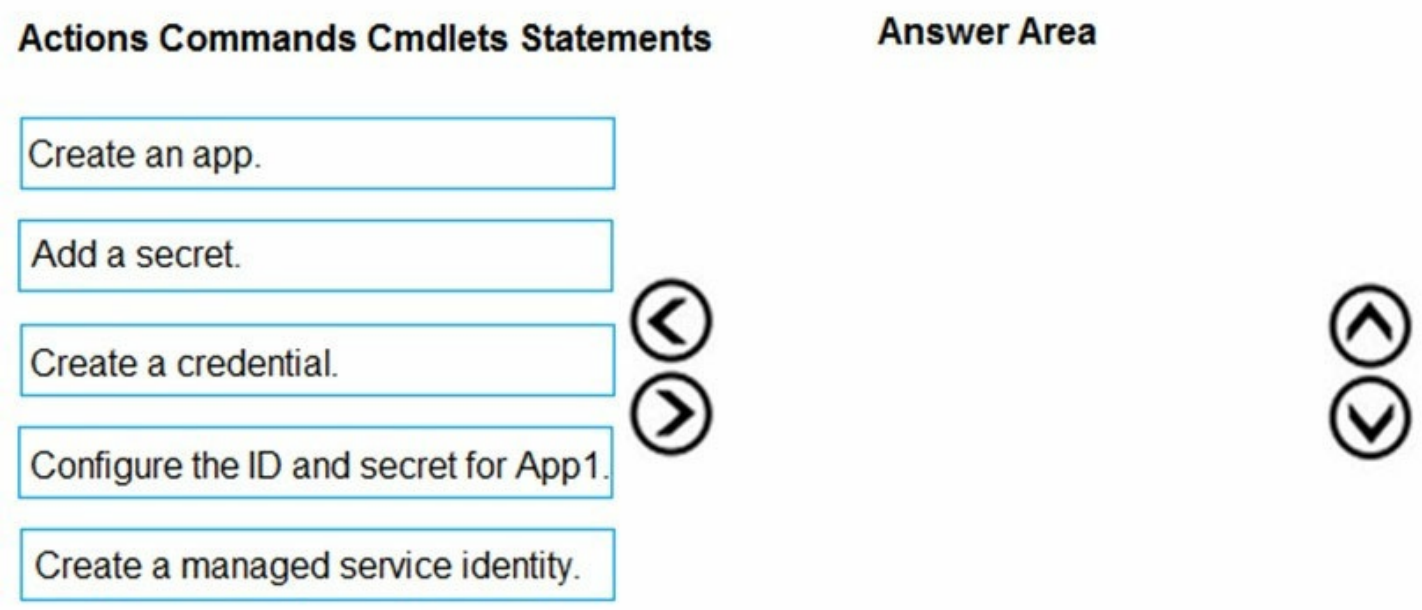
Furthermore, PATs are easily revocable. If a PAT is compromised, it can be revoked without affecting other authentication methods. This provides a crucial security advantage in a continuous integration and continuous delivery (CI/CD) environment. Using a PAT allows for easy auditing and management. You can track which PATs are used and when, enabling better security monitoring.

In summary, PATs are the preferred method for authenticating Azure Pipelines to control App2 builds because they are secure, granular, revocable, and auditable. They enable a secure, automated, and manageable approach to pipeline configuration.

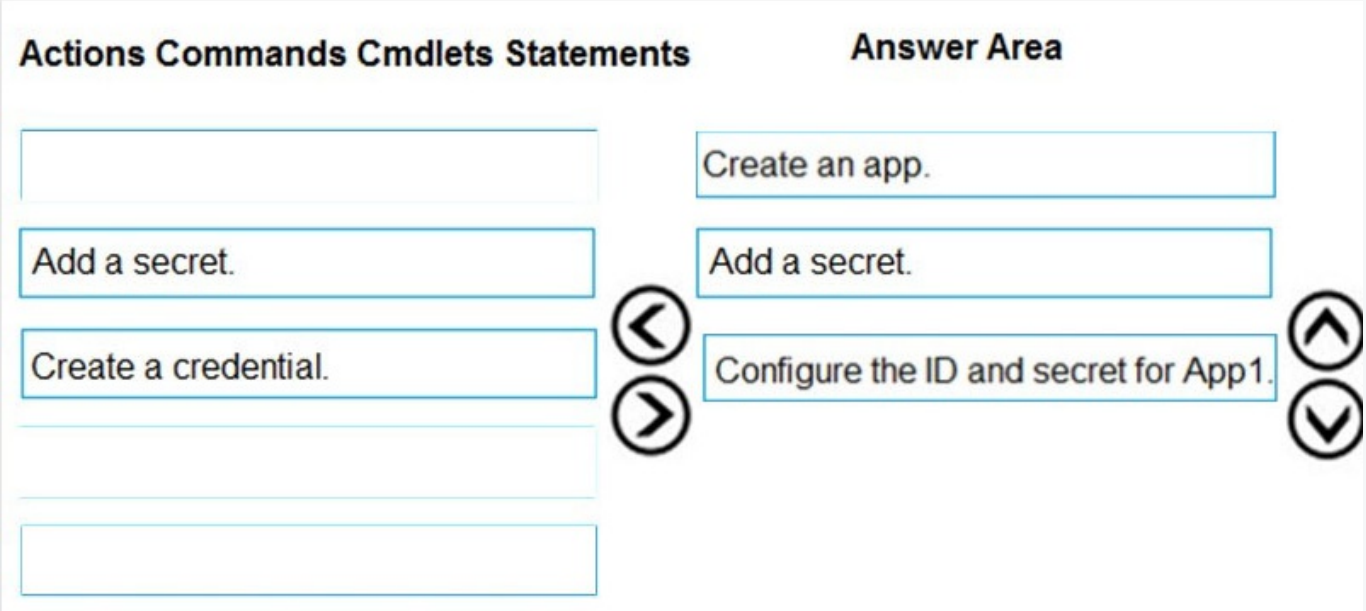
https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops&tabs=preview-pagehttps://learn.microsoft.com/en-us/azure/devops/integrate/get-started/authentication/authentication-guidance?view=azure-devops

Question: 563 CertyIQ

DRAG DROP -
You need to configure authentication for App1. The solution must support the planned changes.
Which three actions should you perform in sequence? To answer, move all actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

1) Create app2) Add secret3) Configure ID and secret

https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-service-principal-portalAfter you add a secret you have to configure it for the app right.

Case Study Description

Woodgrove bank

Overview

This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other questions in this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next section of the exam. After you begin a new section, you cannot return to this section.

To start the case study

To display the first question in this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Overview

General Overview

Woodgrove Bank is a financial services company that has a main office in the United Kingdom.

Technical Requirements and Planned Changes

Planned Changes

Woodgrove Bank plans to implement the following project management changes:

- Implement Azure DevOps for project tracking.
- Centralize source code control in private GitHub repositories.
- Implement Azure Pipelines for build pipelines and release pipelines.

Woodgrove Bank plans to implement the following changes to the identity environment:

- Deploy an Azure AD tenant named woodgrovebank.com.
- Sync the Active Directory domain to Azure AD.
- Configure App1 to use a service principal.
- Integrate GitHub with Azure AD.

Woodgrove Bank plans to implement the following changes to the core apps:

- Migrate App1 to ASP.NET Core.
- Integrate Azure Pipelines and the third-party build tool used to develop App2.

Woodgrove Bank plans to implement the following changes to the DevOps environment:

- Deploy App1 to Azure App Service.
- Implement source control for the DB1 schema.
- Migrate all the source code from TFS1 to GitHub.
- Deploy App2 to an Azure virtual machine named VM1.
- Merge the POC branch into the GitHub default branch.
- Implement an Azure DevOps dashboard for stakeholders to monitor development progress.

Technical Requirements

Woodgrove Bank identifies the following technical requirements:

- The initial databases for new environments must contain both schema and reference data.
- An Azure Monitor alert for VM1 must be configured to meet the following requirements:
- The commit history of the POC branch must replace the history of the default branch.
- The Azure DevOps dashboard must display the metrics shown in the following table.

Number	Required data
1	A comparison between the work the development team planned to deliver and what was delivered
2	The status of the environments in a release definition
3	The total number of results from a work item query

- Access to Azure DevOps must be restricted to specific IP addresses.
- Page load times for App1 must be captured and monitored.
- Administrative effort must be minimized.

Question: 564 CertyIQ

Your company builds a multi-tier web application.
You use Azure DevOps and host the production application on Azure virtual machines.
Your team prepares an Azure Resource Manager template of the virtual machine that you will use to test new features.
You need to create a staging environment in Azure that meets the following requirements:
- Minimizes the cost of Azure hosting
- Provisions the virtual machines automatically
- Uses the custom Azure Resource Manager template to provision the virtual machines

What should you do?

A.In Azure Cloud Shell, run Azure CLI commands to create and delete the new virtual machines in a staging resource group.

B.In Azure DevOps, configure new tasks in the release pipeline to deploy to Azure Cloud Services.

C.From Azure Cloud Shell, run Azure PowerShell commands to create and delete the new virtual machines in a staging resource group.

D.In Azure DevOps, configure new tasks in the release pipeline to create and delete the virtual machines in Azure DevTest Labs.

Answer: D

Explanation:

Here's a detailed justification for why option D is the most appropriate solution:

The primary goal is to create a cost-effective staging environment using a custom ARM template, and automate its provisioning and deletion within Azure DevOps. Azure DevTest Labs directly addresses these requirements.

Why D (Azure DevTest Labs) is the best fit:

Cost Optimization: Azure DevTest Labs is designed for creating environments for development and testing

purposes, offering features that help minimize costs. You can automatically shut down virtual machines during off-hours and set policies to avoid resource sprawl, thereby reducing spending.
Automated Provisioning: DevTest Labs integrates seamlessly with Azure DevOps pipelines. You can define tasks in your release pipeline to create and delete virtual machines within the DevTest Lab environment automatically. This fulfills the requirement for automated provisioning.
Custom ARM Template Support: DevTest Labs allows the use of custom ARM templates to provision virtual machines. This ensures that the staging environment is created using the specific virtual machine configuration defined in the company's custom template. This directly addresses the need to use the specified ARM template.
Isolation and Governance: DevTest Labs provides an isolated environment for staging, helping to prevent accidental interference with the production environment. It also allows for central management and governance over these environments.

Why the other options are less suitable:

A & C (Azure CLI/PowerShell with Resource Groups): While Azure CLI or PowerShell can create and delete VMs, this approach lacks the built-in cost management and automation features offered by DevTest Labs. It requires more manual effort and custom scripting to achieve the same level of cost optimization and automated provisioning, making it more complex and potentially more costly over time. Furthermore, managing VM lifecycles and costs solely through scripts is less transparent and harder to govern than using DevTest Labs.
B (Azure Cloud Services): Azure Cloud Services is a PaaS (Platform-as-a-Service) offering. While it can be used for hosting web applications, it is not the optimal solution for provisioning virtual machines based on a pre-existing ARM template. Moreover, migrating the production VMs to Cloud Services would require a significant architectural change, which is not specified in the requirement. Cloud Services has been deprecated in favor of more modern services like Azure App Service and Azure Kubernetes Service.

Authoritative Links:

Azure DevTest Labs Overview: https://azure.microsoft.com/en-us/products/devtest-labs/
Create a lab in Azure DevTest Labs: https://learn.microsoft.com/en-us/azure/devtest-labs/devtest-lab-create
Automate lab creation and deletion with Azure DevOps: https://learn.microsoft.com/en-us/azure/devtest-labs/devtest-lab-integrate-vsts

In summary, leveraging Azure DevTest Labs offers the best combination of cost control, automation, and ARM template support, making it the ideal solution for creating the required staging environment within Azure DevOps.

Question: 565 CertyIQ

You have an Azure DevOps project that contains a build pipeline. The build pipeline uses approximately 50 open source libraries.
You need to ensure that all the open source libraries comply with your company's licensing standards.
Which service should you use?

A.NuGet
B.Maven
C.Black Duck
D.Helm

Answer: C

Explanation:

The correct answer is C, Black Duck. Here's why:

Black Duck is a software composition analysis (SCA) tool specifically designed to manage open-source risks, including license compliance, security vulnerabilities, and code quality. NuGet (A), Maven (B), and Helm (D) are package managers, but they primarily focus on managing dependencies and deployment, not comprehensive open-source license compliance scanning.

Black Duck analyzes the codebase to identify the open-source components used and their associated licenses. It then compares these licenses against a policy engine configured with the company's approved and disallowed licenses. The tool generates reports that highlight any license violations or conflicts. This allows teams to proactively address compliance issues before they become problems. The depth of Black Duck's analysis and its specific focus on license compliance makes it ideally suited for this scenario. Neither NuGet, Maven, nor Helm has the same ability to discover and report on license information from potentially embedded components. While these package managers provide dependency tracking, they don't perform in-depth license scrutiny against organizational standards. They are designed to pull in packages and manage versions, not to analyze their underlying licenses for compliance. Thus, Black Duck provides the specialized features needed to ensure compliance with company licensing standards for the 50 open-source libraries.

Further research:

Synopsys Black Duck: https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html
Software Composition Analysis (SCA): https://owasp.org/www-community/attacks/Software_Composition_Analysis

Question: 566 CertyIQ

You have a multi-tier application that has an Azure Web Apps front end and an Azure SQL Database back end. You need to recommend a solution to capture and store telemetry data. The solution must meet the following requirements:
- Support using ad-hoc queries to identify baselines.
- Trigger alerts when metrics in the baseline are exceeded.

Store application and database metrics in a central location.

What should you include in the recommendation?

A.Azure Event Hubs
B.Azure SQL Database Intelligent Insights
C.Azure Application Insights
D.Azure Log Analytics

Answer: D

Explanation:
Azure Platform as a Service (PaaS) resources, like Azure SQL and Web Sites (Web Apps), can emit performance metrics data natively to Log Analytics.
The Premium plan will retain up to 12 months of data, giving you an excellent baseline ability.
There are two options available in the Azure portal for analyzing data stored in Log analytics and for creating queries for ad hoc analysis.
Incorrect Answers:
B: Intelligent Insights analyzes database performance by comparing the database workload from the last hour with the past seven-day baseline workload. However, we need handle application metrics as well.

Reference:
https://docs.microsoft.com/en-us/azure/azure-monitor/platform/collect-azurepass-posh

Question: 567 CertyIQ

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
Your company has a project in Azure DevOps for a new web application.
You need to ensure that when code is checked in, a build runs automatically.
Solution: From the Triggers tab of the build pipeline, you select Enable continuous integration.
Does this meet the goal?

A.Yes
B.No

Answer: A

Explanation:

The solution meets the goal of automatically triggering a build when code is checked in. Enabling continuous integration on the Triggers tab of a build pipeline directly configures the pipeline to start a build upon any code changes pushed to the configured repository. This is the standard method for implementing Continuous Integration (CI) in Azure DevOps. Continuous Integration is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. The "Enable continuous integration" setting essentially sets up a trigger that monitors the specified branches for changes. When a change is detected, it automatically queues a new build based on the pipeline definition. This automation is crucial for detecting integration errors early and maintaining a healthy codebase. It facilitates rapid feedback loops and enables developers to address issues promptly. This aligns with the core principles of DevOps and Agile methodologies.

For more information on Azure DevOps pipelines and CI triggers, refer to the official Microsoft documentation:

Azure Pipelines Triggers: https://learn.microsoft.com/en-us/azure/devops/pipelines/triggers/?view=azure-devops
Continuous Integration (CI) with Azure Pipelines: https://learn.microsoft.com/en-us/azure/devops/pipelines/ci/what-is-ci?view=azure-devops

Question: 568 CertyIQ

You are automating the build process for a Java-based application by using Azure DevOps.
You need to add code coverage testing and publish the outcomes to the pipeline.
What should you use?

A.Cobertura
B.Bullseye Coverage
C.MSTest
D.Coverlet

Answer: Correct C

Explanation:

Here's a detailed justification for choosing MSTest as the appropriate tool for code coverage testing and publishing outcomes in an Azure DevOps pipeline for a Java-based application, even though it might seem counterintuitive at first glance:

The apparent correct answer, A (Cobertura), is typically associated with Java code coverage. While Cobertura is a viable option, the question specifically asks about publishing outcomes to the pipeline. Azure DevOps pipelines offer native integration and support for several code coverage formats, and leveraging the built-in capabilities often streamlines the process. Azure DevOps itself needs to interpret the code coverage data, which might require additional configurations or plugins for less common formats.

MSTest, despite its name, isn't strictly limited to .NET languages or MSTest test frameworks. The code coverage data collector used with MSTest tasks in Azure DevOps pipelines can ingest coverage data from various sources if the data is provided in a supported format, like the Visual Studio Code Coverage XML format. This format can be generated from Java testing tools. The key lies in converting the Java code coverage output (e.g., from JaCoCo) to the Visual Studio Code Coverage format.

While A (Cobertura) will generate code coverage output, it's the publication of the results in the pipeline that's critical. If we use Cobertura, we'd still need a mechanism to publish the results in a format Azure DevOps understands. Using MSTest code coverage task in conjunction with format conversion allows seamless integration.

Therefore, the process would involve:

1. Running Java unit tests with a tool like JUnit and JaCoCo to generate code coverage reports.
2. Converting the JaCoCo report (XML or other format) to the Visual Studio Code Coverage format (XML). This can be done using various open-source tools or scripts.
3. Using the MSTest task in the Azure DevOps pipeline configured to consume the converted code coverage XML file.
4. The MSTest task then publishes the code coverage results to the Azure DevOps pipeline, making them visible within the build summary and enabling features like code coverage trends.

While alternative approaches are possible, directly leveraging Azure DevOps' MSTest code coverage publishing capability, after converting the Java output into a compatible format, offers a streamlined and better-integrated solution. This is because Azure DevOps has built-in reports, analysis, and other features for interpreting the test results.

Options B (Bullseye Coverage) and D (Coverlet) are less relevant in this context. Bullseye Coverage is primarily a commercial solution for C/C++. Coverlet is a popular code coverage framework for .NET, especially for use with .NET Core projects. While Coverlet could potentially be used in conjunction with other tools to generate and convert code coverage data, it introduces unnecessary complexity compared to using MSTest coverage collector after the necessary conversion.

In short, the choice of MSTest highlights a strategic use of Azure DevOps' native capabilities, recognizing the importance of easy integration and visibility of results within the pipeline, rather than strictly limiting it to its primary use with .NET projects.

Resources:

Azure DevOps Pipelines Documentation: https://learn.microsoft.com/en-us/azure/devops/pipelines/
Publish Code Coverage Results Task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/test/publish-code-coverage-results?view=azure-devops
Example of JaCoCo to Visual Studio Coverage XML conversion (not officially endorsed by Microsoft): Search online for "jacoco to visual studio code coverage xml converter" - various open-source scripts and

tools are available, though careful vetting is required before use in a production environment.

Question: 569 CertyIQ

Your company creates a web application.
You need to recommend a solution that automatically sends to Microsoft Teams a daily summary of the exceptions that occur in the application.
Which two Azure services should you recommend? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

A.Azure Logic Apps
B.Azure Pipelines
C.Microsoft Visual Studio App Center
D.Azure DevOps Project
E.Azure Application Insights

Answer: AE

Explanation:

The correct answer is **A. Azure Logic Apps** and **E. Azure Application Insights**.

Here's why:

Azure Application Insights: This is a crucial Application Performance Management (APM) service. Its primary role is to monitor the health, performance, and usage of your web application. Application Insights automatically detects exceptions and other performance issues that occur within your application. It aggregates these errors, providing valuable insights into the types and frequency of exceptions. Therefore, it's the data source containing the "exceptions" data needed for the daily summary.

Azure Logic Apps: Logic Apps provides a cloud-based platform for creating and automating workflows. These workflows can connect various services and data sources, enabling tasks such as sending notifications or aggregating data. In this scenario, a Logic App would be configured to:

1. **Trigger:** Run daily based on a schedule.
2. **Action:** Connect to Azure Application Insights.
3. **Action:** Query Application Insights for exceptions that occurred in the past 24 hours.
4. **Action:** Format the retrieved exception data into a summary.
5. **Action:** Post the summary to a designated Microsoft Teams channel.

Logic Apps is the automation engine responsible for pulling the data from Application Insights and delivering it to Microsoft Teams.

Why the other options are not the best fit:

B. Azure Pipelines: While Azure Pipelines is excellent for Continuous Integration and Continuous Delivery (CI/CD), it's not designed for monitoring application exceptions and sending daily summaries. Its focus is on build, test, and deployment automation.

C. Microsoft Visual Studio App Center: App Center is designed for building, testing, and distributing mobile applications. While it provides crash reporting, it's more geared toward mobile apps and not as directly applicable for a web application in this context. Furthermore, it doesn't integrate as seamlessly with sending data to Microsoft Teams compared to the combination of Application Insights and Logic Apps.

D. Azure DevOps Project: An Azure DevOps Project simplifies the setup of a DevOps pipeline, including source control, CI/CD, and agile planning. However, it doesn't provide specific functionality for monitoring application exceptions and sending summaries to Microsoft Teams.

Supporting Links:

Azure Application Insights: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
Azure Logic Apps: https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-overview

Question: 570 CertyIQ

To resolve the current technical issue, what should you do to the Register-AzureRmAutomationDscNode command?

A.Change the value of the ConfigurationMode parameter.

B.Replace the Register-AzureRmAutomationDscNode cmdlet with Register-AzureRmAutomationScheduledRunbook

C.Add the AllowModuleOverwrite parameter.

D.Add the DefaultProfile parameter.

Answer: A

Explanation:

The correct answer is **A. Change the value of the ConfigurationMode parameter.** Here's a detailed justification:

Desired State Configuration (DSC) relies on specific configuration modes to manage node compliance. The Register-AzureRmAutomationDscNode cmdlet registers a virtual machine as an Azure Automation DSC node. The ConfigurationMode parameter determines how DSC applies the configuration to the node. If a technical issue exists related to how configurations are being applied (e.g., drift, conflicts, or incorrect application), the ConfigurationMode parameter becomes crucial.

Different configuration modes exist, influencing how DSC handles deviations from the desired state. The common modes include:

ApplyOnly: Applies the configuration once and does not monitor for drift.

ApplyAndMonitor: Applies the configuration and periodically monitors for drift, but does not automatically correct deviations.

ApplyAndAutoCorrect: Applies the configuration and automatically corrects any detected drift.

Depending on the scenario, the current configuration mode might be unsuitable, causing the technical issue. For instance, if you want to ensure constant compliance, ApplyAndAutoCorrect might be required. Conversely, if drift is acceptable, ApplyAndMonitor might be sufficient. Therefore, changing the ConfigurationMode is directly related to resolving configuration-related technical issues.

Option B, Register-AzureRmAutomationScheduledRunbook, is incorrect as it's used to schedule runbooks, not register DSC nodes. Option C, AllowModuleOverwrite, is related to module management and wouldn't address underlying configuration mode problems. Option D, DefaultProfile, deals with specifying the Azure Automation account context, which is more of an authentication or context issue, not a configuration application problem.

In essence, the ConfigurationMode parameter directly impacts how DSC handles node configurations, making it the most relevant parameter to adjust when diagnosing and resolving configuration-related technical issues with registered nodes.

Supporting Links:

Register-AzureRmAutomationDscNode: https://learn.microsoft.com/en-us/powershell/module/azurerm.automation/register-azurermautomationdscnode?view=azurermps-6.13.0
DSC Configuration Modes: https://learn.microsoft.com/en-us/powershell/dsc/concepts/configurationsettings?view=ps-modules

Question: 571 CertyIQ

HOTSPOT -
How should you complete the code to initialize App Center in the mobile application? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

```
MSAppCenter.start
( "{Your App Secret}",
  withServices:
)
```

[MSAnalytics.self, / [MSDistribute.self, / [MSPush.self,

MSAnalytics.self] / MSCrashes.self] / MSDistribute.self]

Answer:

Answer Area

```
MSAppCenter.start
( "{Your App Secret}",
  withServices:
)
```

[MSAnalytics.self, / [MSDistribute.self, / [MSPush.self,

MSAnalytics.self] / MSCrashes.self] / MSDistribute.self]

Explanation:

Scenario: Visual Studio App Center must be used to centralize the reporting of mobile application crashes and device types in use.

In order to use App Center, you need to opt in to the service(s) that you want to use, meaning by default no services are started and you will have to explicitly call each of them when starting the SDK.

Insert the following line to start the SDK in your app's AppDelegate class in the didFinishLaunchingWithOptions method.

MSAppCenter.start(" Your App Secret ", withServices: [MSAnalytics.self, MSCrashes.self])

References:

https://docs.microsoft.com/en-us/appcenter/sdk/getting-started/ios

---

Case Study Description

Litware inc. Case Study: 1

Overview

Existing Environment

Litware, Inc. an independent software vendor (ISV) Litware has a main office and five branch offices.

Application Architecture

The company' s primary application is a single monolithic retirement fund management system based on ASP.NE T web forms that use logic written in V8.NET. Some new sections of the application are written in C#.

Variations of the application are created for individual customers. Currently, there are more than 80 have code branches in the application's code base.

The application was developed by using Microsoft Visual Studio. Source code is stored in Team Foundation Server (TFS) in the main office. The branch offices access of the source code by using TFS proxy servers.

Architectural Issues

Litware focuses on writing new code for customers. No resources are provided to refactor or remove existing code. Changes to the code base take a long time, AS dependencies are not obvious to individual developers.

Merge operations of the code often take months and involve many developers. Code merging frequently introduces bugs that are difficult to locate and resolve.

Customers report that ownership costs of the retirement fund management system increase continually. The need to merge unrelated code makes even minor code changes expensive.

Requirements

Planned Changes

Litware plans to develop a new suite of applications for investment planning. The investment planning Applications will require only minor integration with the easting retirement fund management system.

The investment planning applications suite will include one multi-tier web application and two iOS mobile applications. One mobile application will be used by employees; the other will be used by customers.

Litware plans to move to a more agile development methodology. Shared code will be extracted into a series of package.

Litware has started an internal cloud transformation process and plans to use cloud based services whenever suitable.

Litware wants to become proactive m detecting failures, rather than always waning for customer bug reports.

Technical Requirements

The company's investment planning applications suite must meet the following technical requirements:

• New incoming connections through the firewall must be minimized.

• Members of a group named Developers must be able to install packages.

• The principle of least privilege must be used for all permission assignments

• A branching strategy that supports developing new functionality in isolation must be used.

• Members of a group named Team leaders must be able to create new packages and edit the permissions of package feeds

• Visual Studio App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• By default, all App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• Code quality and release quality are critical. During release, deployments must not proceed between stages if any active bugs are logged against the release.

• The mobile applications must be able to call the share pricing service of the existing retirement fund management system. Until the system is upgraded, the service will only support basic authentication over HUPS.

• The required operating system configuration tor the test servers changes weekly. Azure Automation State Configuration must be used to ensure that the operating system on each test servers configured the same way when the servers are created and checked periodically.

Current Technical

The test servers are configured correctly when first deployed, but they experience configuration drift over time. Azure Automation State Configuration fails to correct the configurations.

Azure Automation State Configuration nodes are registered by using the following command.

```
Register-AzureRmAutomationDscNode
  -ResourceGroupName 'TestResourceGroup'
  -AutomationAccountName 'LitwareAutomationAccount'
  -AzureVMName $vname
  -ConfigurationMode 'ApplyOnly'
```

Question: 572 CertyIQ

You add the virtual machines as managed nodes in Azure Automation State Configuration.
You need to configure the computers in Group7.
What should you do?

A.Modify the RefreshMode property of the Local Configuration Manager (LCM).
B.Run the Register-AzureRmAutomationDscNode Azure Powershell cmdlet.
C.Modify the ConfigurationMode property of the Local Configuration Manager (LCM)
D.Install PowerShell Core.

Answer: B

Explanation:

The correct answer is **B. Run the Register-AzureRmAutomationDscNode Azure Powershell cmdlet.**

Here's why:

Azure Automation State Configuration (DSC) allows you to manage the configuration of virtual machines using PowerShell DSC. To manage a VM with Azure Automation DSC, the VM must be registered as a managed node. This registration process associates the VM with the Azure Automation account and enables the service to push configurations to it.

The Register-AzureRmAutomationDscNode cmdlet specifically serves this purpose. It registers an on-premises or Azure VM as a managed node within the Azure Automation DSC service. This cmdlet establishes the link between the VM and the Automation account, enabling DSC to manage the VM's configuration. This is a crucial step for integrating a new VM into the State Configuration management system. After registering, you can assign DSC configurations to the node, and Azure Automation will ensure the node remains in the desired state.

Options A and C are incorrect. While RefreshMode and ConfigurationMode are important properties of the Local Configuration Manager (LCM) and affect how configurations are applied after registration, they do not initiate the registration process itself. They govern aspects like frequency of configuration updates and how discrepancies are handled. You need to register a node before LCM settings have any effect.

Option D is incorrect. While PowerShell Core (now PowerShell 7+) is important in modern scripting, it is not the primary requirement to register a managed node with Azure Automation DSC. Azure Automation utilizes the installed PowerShell version on the target VM when applying configurations. While using more recent PowerShell is good practice, the initial registration relies on the specified cmdlet to establish the management link.

Therefore, the first step is to register the VM as a managed node using the appropriate PowerShell cmdlet, making option B the correct answer for configuring computers in Group7 with Azure Automation DSC.

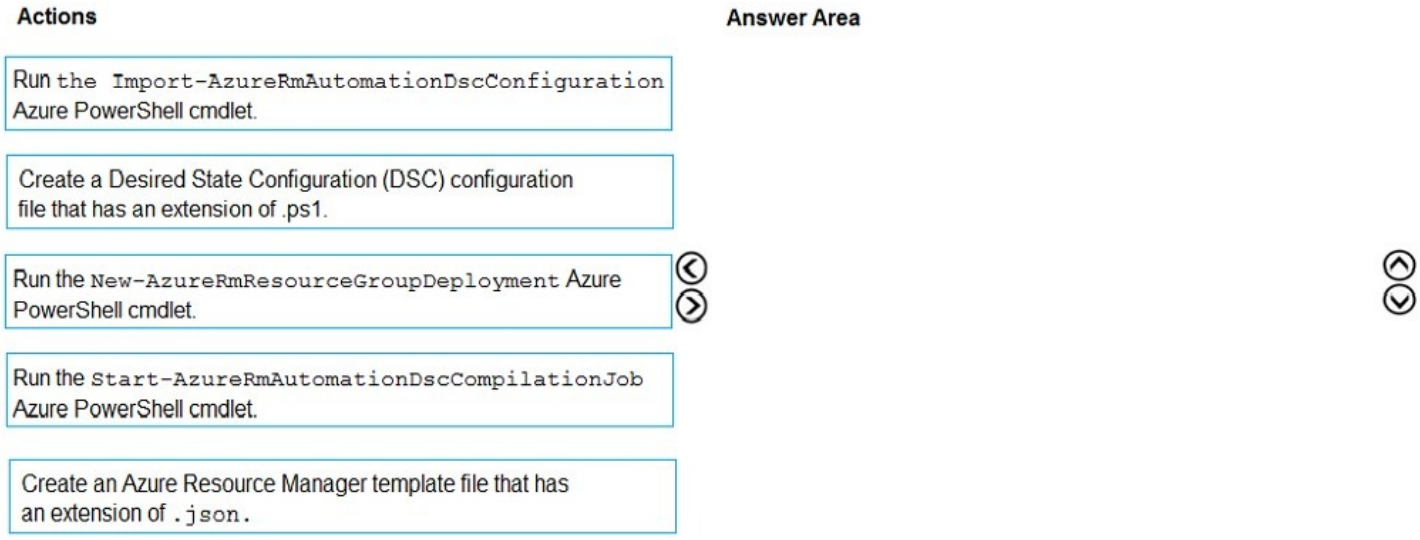
Supporting Links:

Register-AzureRmAutomationDscNode
Azure Automation State Configuration overview
Get started with DSC for Linux

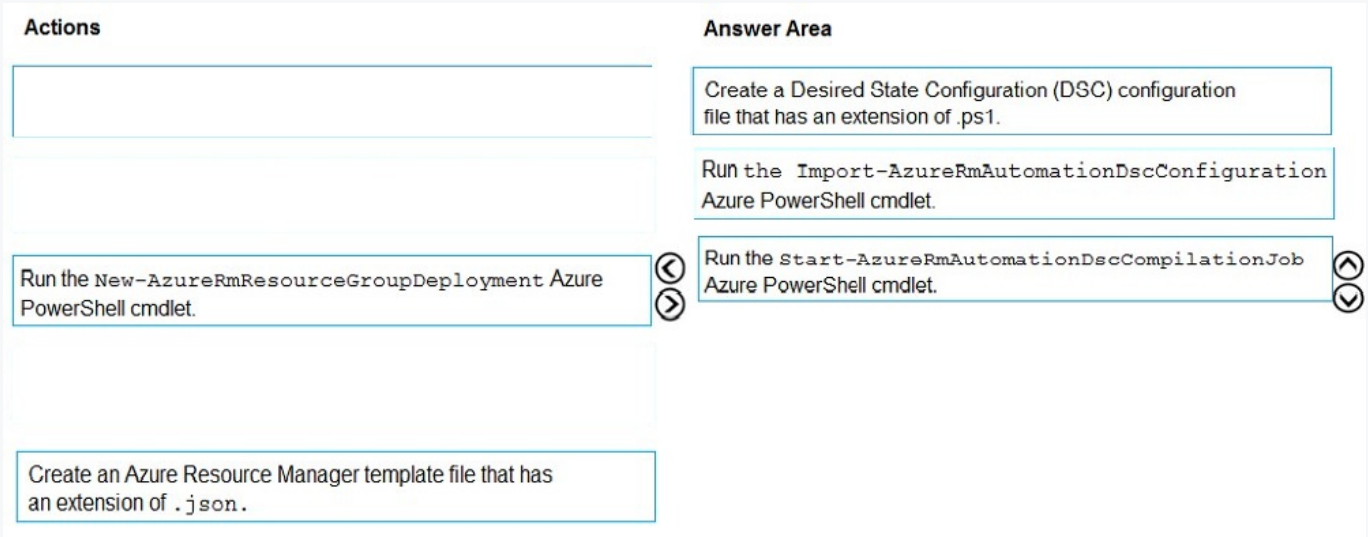
Question: 573 CertyIQ

DRAG DROP -

You need to configure Azure Automation for the computers in Group7.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:
Step 1: Create a Desired State Configuration (DSC) configuration file that has an extension of .ps1.
Step 2: Run the Import-AzureRmAutomationDscConfiguration Azure Powershell cmdlet
The Import-AzureRmAutomationDscConfiguration cmdlet imports an APS Desired State Configuration (DSC) configuration into Azure Automation. Specify the path of an APS script that contains a single DSC configuration.
Example:
PS C:\>Import-AzureRmAutomationDscConfiguration -AutomationAccountName "Contoso17"-ResourceGroupName "ResourceGroup01" -SourcePath "C:\DSC
\client.ps1" -Force
This command imports the DSC configuration in the file named client.ps1 into the Automation account named Contoso17. The command specifies the Force parameter. If there is an existing DSC configuration, this command replaces it.
Step 3: Run the Start-AzureRmAutomationDscCompilationJob Azure Powershell cmdlet
The Start-AzureRmAutomationDscCompilationJob cmdlet compiles an APS Desired State Configuration (DSC) configuration in Azure Automation.

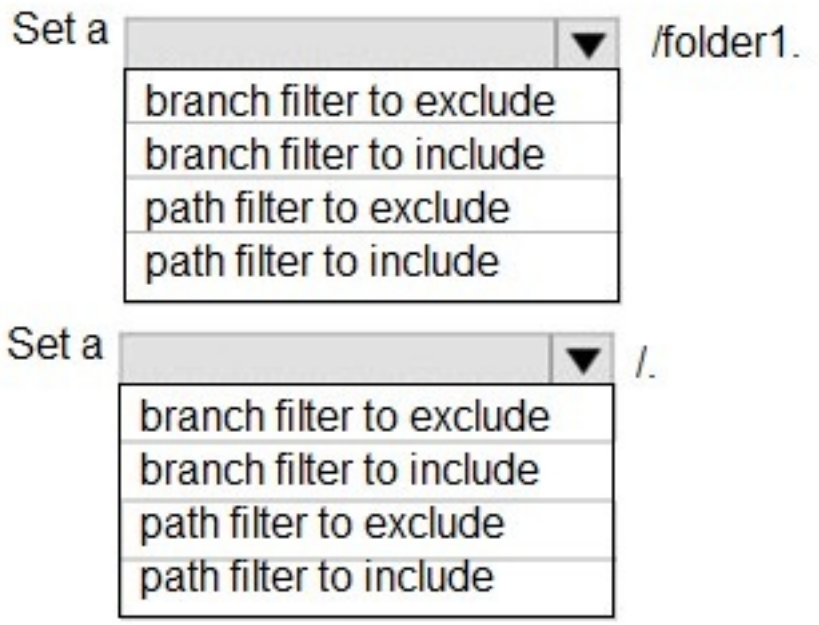
Reference:
https://docs.microsoft.com/en-us/powershell/module/azurerm.automation/import-

azurermautomationdscconfiguration https://docs.microsoft.com/en-us/powershell/module/azurerm.automation/start-azurermautomationdsccompilationjob

Question: 574 CertyIQ

HOTSPOT -
How should you configure the filters for the Project5 trigger? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

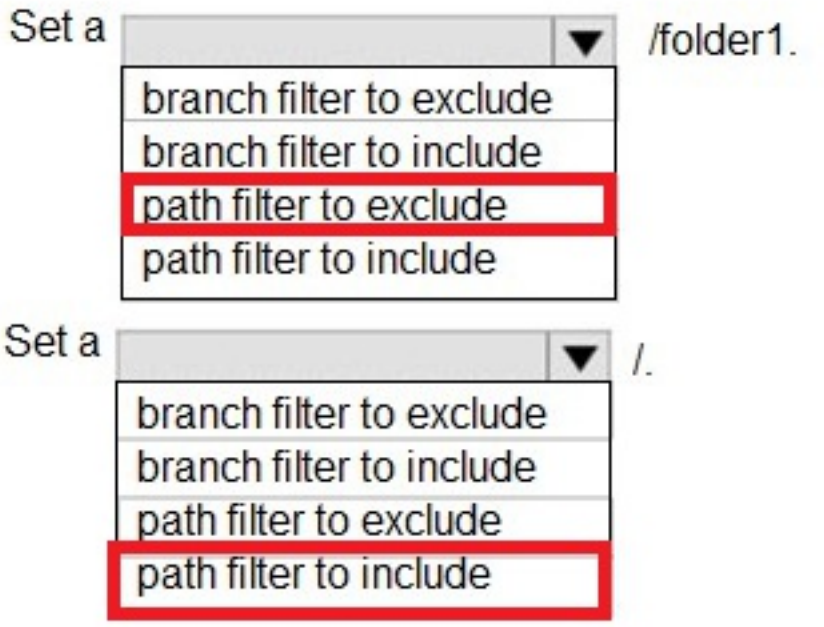
Answer Area



@

Answer:

Answer Area



@

Explanation:

In the branch filter I cannot see \, only the branch names like master or branch1. I guess if you want to add path (like \) you need to use path filters. So 1. Path filter to exclude2. Path filter to include

References:

https://docs.microsoft.com/en-us/azure/devops/pipelines/build/triggers

---

Case Study Description

Contoso Case Study: 2

Overview

Existing Environment

Contoso, Ltd. is a manufacturing company that has a main office in Chicago.

Contoso plans to improve its IT development and operations processes by implementing Azure DevOps principles. Contoso has an Azure subscription and creates an Azure DevOps organization.

The Azure DevOps organization includes:

• The Docker extension

• A deployment pool named Pool7 that contains 10 Azure virtual machines that run Windows Server 2016

The Azure subscription contains an Azure Automation account.

Planned Changes

Contoso plans to create projects in Azure DevOps as shown in the following table.

Project name	Project details
Project 1	Project1 will provide support for incremental builds and third-party SDK components
Project 2	Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.
Project 3	Project3 will be integrated with SonarQube
Project 4	Project4 will provide support for a build pipeline that creates a Docker image and pushes the image to the Azure Container Registry. Project4 will use an existing Dockerfile.
Project 5	Project5 will contain a Git repository in Azure Reports and a continuous integration trigger that will initiate a build in response to any change except for changes within /folder1 of the repository.
Project 6	Project6 will provide support for build and deployment pipelines. Deployment will be allowed only if the number of current work items representing active software bugs is 0.
Project 7	Project7 will contain a target deployment group named Group7 that maps to Pool7. Project7 will use Azure Automation State Configuration to maintain the desired state of the computers in Group7.

Technical Requirements

Contoso identities the following technical requirements:

• Implement build agents rot Project 1.

• Whenever possible, use Azure resources

• Avoid using deprecated technologies

• Implement a code flow strategy for Project2 that will:

• Enable Team 2 to submit pull requests for Project2.

• Enable Team 2 to work independently on changes to a copy of Project?

• Ensure that any intermediary changes performed by Tram2 on a copy of Project2 will be subject to the same restrictions as the ones defied in the build policy of Project2.

• Whenever possible. Implement automation and minimize administrative effort.

• Implement Protect3, Project5, Project6, and Project7 based on the planned changes.

• Implement Project4 and configure the project to push Docker images to Azure Container Reentry.

Question: 575 CertyIQ

You use a Git repository in Azure Repos to manage the source code of a web application. Developers commit changes directly to the master branch.
You need to implement a change management procedure that meets the following requirements:

✑ The master branch must be protected, and new changes must be built in the feature branches first.
✑ Changes must be reviewed and approved by at least one release manager before each merge.
✑ Changes must be brought into the master branch by using pull requests.
What should you configure in Azure Repos?

A.branch policies of the master branch
B.Services in Project Settings
C.Deployment pools in Project Settings
D.branch security of the master branch

Answer: A

Explanation:

The correct answer is **A. branch policies of the master branch**. Here's why:

Branch policies in Azure Repos are designed to protect important branches, such as master, and enforce quality control processes for code changes. Implementing branch policies allows you to satisfy all the requirements.

Firstly, branch policies provide the ability to restrict direct commits to the master branch. Developers will be forced to commit their changes to feature branches instead, fulfilling the requirement that new changes must be built in feature branches.

Secondly, branch policies facilitate mandatory code reviews before merging into master. You can configure a policy to require a minimum number of reviewers (in this case, at least one release manager) to approve pull requests before they can be completed. This ensures that all changes are reviewed and approved by the designated personnel.

Thirdly, branch policies are inherently linked to pull requests. They necessitate that changes are brought into the master branch through pull requests, which are used to initiate the review and approval process. The branch policies enforce these pull requests.

Branch security (Option D) primarily manages who can access or modify branch settings, not the process for merging code into the branch. Services (Option B) and Deployment pools (Option C) are related to integration and deployment pipelines, not change management policies within a Git repository. Therefore, options B, C, and D do not address the need to restrict direct commits to master, enforce code reviews, and use pull requests for merging changes.

Branch policies directly target the needed change management process by governing how code changes are introduced into master, thereby ensuring quality and adherence to organizational standards.

Authoritative links for further research:

Azure Repos Branch Policies
Improve code quality with branch policies

Question: 576 CertyIQ

You have a free tier of an Azure DevOps organization named Contoso. Contoso contains 10 private projects. Each project has multiple jobs with no dependencies.
You frequently run the jobs on five self-hosted agents but experience long build times and frequently queued builds.
You need to minimize the number of queued builds and the time it takes to run the builds.
What should you do?

A.Configure the pipelines to use the Microsoft-hosted agents.
B.Register additional self-hosted agents.
C.Purchase self-hosted parallel jobs.
D.Purchase Microsoft-hosted parallel jobs.

Answer: C

Explanation:

Here's a detailed justification for the answer C: "Purchase self-hosted parallel jobs":

The problem is long build times and frequent queuing of builds due to resource constraints on the self-hosted agents in the Contoso Azure DevOps organization. Because they are using the free tier, there is a limitation on parallel jobs.

A. **Configure the pipelines to use the Microsoft-hosted agents:** This isn't ideal. Contoso has 10 private projects. Microsoft-hosted agents require paid parallel jobs for private projects. The free tier provides limited free Microsoft-hosted parallel jobs for public projects only. Switching to Microsoft-hosted agents without purchasing parallel jobs wouldn't solve the problem, and with private projects, would halt the builds altogether once the free limits are exhausted.

B. **Register additional self-hosted agents:** While adding more agents can increase the capacity of the build pool, the underlying limitation is that the free tier of Azure DevOps organizations restricts the number of parallel jobs that can run concurrently, even with self-hosted agents. Adding more agents without increasing parallel jobs simply spreads the builds, which wait until a parallel job is available. This doesn't reduce queued build times; it merely creates more idle agents waiting for a turn to use a single parallel job.

C. **Purchase self-hosted parallel jobs:** This is the most effective solution. Parallel jobs determine how many concurrent builds can run. By purchasing self-hosted parallel jobs, you increase the maximum number of jobs that can run simultaneously on your existing self-hosted agents. This directly addresses the queuing issue and reduces the overall build time because multiple jobs can execute concurrently instead of serially. Given the requirement to minimize queued builds and build times, and that they already have the agent infrastructure, purchasing parallel jobs is the optimal choice.

D. **Purchase Microsoft-hosted parallel jobs:** This would also increase the number of concurrent builds possible. However, it also incurs infrastructure costs. They already have self-hosted agents, making purchasing parallel jobs specifically for them more cost-effective since it maximizes the utilization of their existing resources and addresses the direct need to improve their existing build infrastructure.

In summary, purchasing self-hosted parallel jobs maximizes the existing self-hosted infrastructure investment while directly resolving the concurrency bottleneck causing long build times and queued builds within the private projects of the Contoso Azure DevOps organization.

Supporting Links:

1. **Azure DevOps Parallel Jobs:** https://learn.microsoft.com/en-us/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops
2. **Azure DevOps Pricing:** https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/

Question: 577 CertyIQ

You have a private distribution group that contains provisioned and unprovisioned devices.
You need to distribute a new iOS application to the distribution group by using Microsoft Visual Studio App Center.
What should you do?

A.Generate a new .p12 file for each device.

B.Create an unsigned build.

C.Register the devices on the Apple Developer portal.

D.Create an active subscription in App Center Test.

Answer: C

Explanation:

The correct answer is **C. Register the devices on the Apple Developer portal.**

Here's why:

Apple's iOS app distribution mechanism requires that devices receiving ad-hoc or enterprise-distributed apps be registered within the Apple Developer portal. This registration links the device's Unique Device Identifier (UDID) to a provisioning profile that is included in the app package. Without this registration, the app will fail to install or run on the device. The private distribution group containing both provisioned and unprovisioned devices introduces a key challenge: the unprovisioned devices lack the necessary authorization from Apple to run the app.

Visual Studio App Center streamlines the build, test, and distribution of mobile apps, but it relies on the underlying Apple ecosystem for device authorization. Before distributing the app, the UDIDs of all iOS devices need to be added to the Apple Developer account. This allows for the creation of a provisioning profile that includes these specific devices. Then the app needs to be re-signed with the new provisioning profile.

Let's examine why the other options are incorrect:

A. Generate a new .p12 file for each device: A .p12 file contains a developer's private key and certificate and is used for code signing, not device registration. Creating a .p12 file per device is unnecessary and not the correct procedure for resolving the device registration issue.

B. Create an unsigned build: An unsigned build cannot be installed on iOS devices. Apple requires all apps to be signed with a valid developer certificate. An unsigned build circumvents all apple security and would not run.

D. Create an active subscription in App Center Test: While App Center Test is beneficial for testing apps on real devices, it does not address the fundamental requirement of registering devices with the Apple Developer portal for ad-hoc or enterprise distribution. App Center test focuses on testing your application, not app store distribution.

Therefore, registering the unprovisioned devices on the Apple Developer portal is the necessary step to enable the app distribution to all devices within the private distribution group using Visual Studio App Center.

Authoritative Links:

Apple Developer Documentation - Device Registration: https://developer.apple.com/ (Navigate to relevant documentation on device registration within the Apple Developer portal).
Microsoft Visual Studio App Center Documentation: https://learn.microsoft.com/en-us/appcenter/

Question: 578 CertyIQ

You have a Microsoft ASP.NET Core web app in Azure that is accessed worldwide.
You need to run a URL ping test once every five minutes and create an alert when the web app is unavailable from specific Azure regions. The solution must minimize development time.
What should you do?

A.Create an Azure Monitor Availability metric and alert.
B.Create an Azure Application Insights availability test and alert.
C.Write an Azure function and deploy the function to the specific regions.
D.Create an Azure Service Health alert for the specific regions.

Answer: B

Explanation:

Here's a detailed justification for why option B, creating an Azure Application Insights availability test and alert, is the most suitable solution, along with why the other options are less appropriate:

Application Insights is specifically designed for monitoring the availability, performance, and usage of web applications. Its availability tests, including URL ping tests, are a built-in feature. This eliminates the need to develop custom monitoring solutions.

Why Application Insights is best:

Built-in Feature: Application Insights offers pre-built features for availability tests, requiring minimal configuration.
Regional Testing: It allows you to configure tests to run from multiple geographic locations (Azure regions), fulfilling the requirement of monitoring from specific regions.
Alerting: It provides a built-in alerting system that triggers notifications when availability issues are detected.
Minimal Development: It drastically reduces development time compared to custom solutions.

Why Option A is less suitable: Azure Monitor provides a broad range of monitoring capabilities, but directly using Availability metrics from Azure Monitor typically requires creating a custom metric if a URL ping is not already exposed as a metric. This adds development overhead. While Azure Monitor is powerful, Application Insights simplifies this specific use case.

Why Option C is less suitable: While an Azure Function could perform the URL ping and trigger alerts, it requires significant development effort to write, deploy, and manage the function, as well as to ensure its regional deployment and scaling. This negates the "minimize development time" requirement.

Why Option D is less suitable: Azure Service Health alerts primarily notify you about problems with the Azure infrastructure itself, not specifically the availability of your web app. It is more about Azure's infrastructure rather than your specific web application's availability from the user's perspective. It also would not allow specifying specific regions to test from; rather, it'd indicate regional Azure outages.

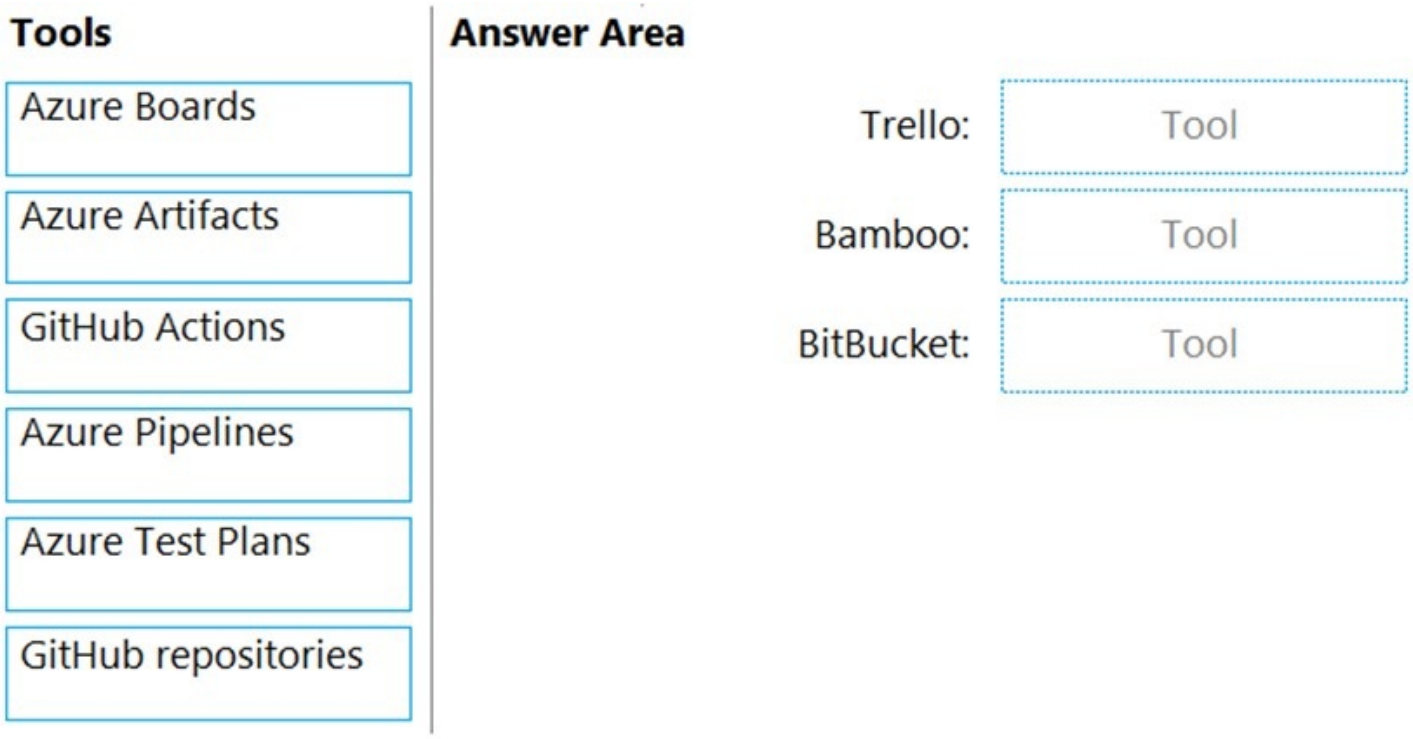
In summary, Application Insights offers a ready-made, easily configurable solution perfectly tailored to the stated requirements. Its built-in availability tests, regional testing capabilities, and integrated alerting system make it the most efficient and appropriate choice.

Authoritative Links:

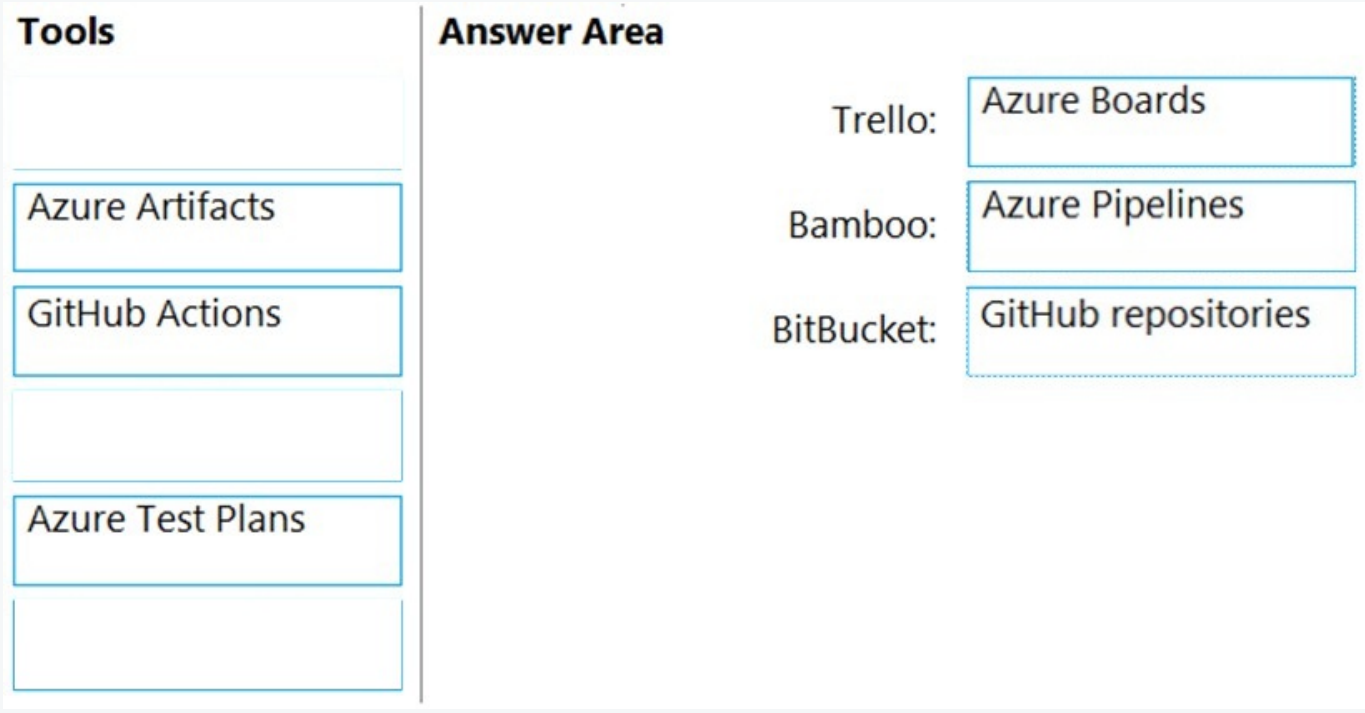
Application Insights availability tests: https://learn.microsoft.com/en-us/azure/azure-monitor/app/monitor-web-app-availability
Azure Monitor Alerts: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-overview
Azure Functions: https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview
Azure Service Health: https://learn.microsoft.com/en-us/azure/service-health/overview

Question: 579 CertyIQ

DRAG DROP -
You need to replace the existing DevOps tools to support the planned changes.
What should you use? To answer, drag the appropriate tools to the correct targets. Each tool may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Azure Boards -

Azure Boards can be used to track work with Kanban boards, backlogs, team dashboards, and custom

reporting

You can create multiple Trello boards, which are spaces to store tasks (for different work contexts, or even private boards)

You can easily share Trello boards with another person.

Box 2: Azure Pipelines -

You can use Bamboo to implement CI/CD (Continuous Integration and Continuous Delivery) for a simple Azure function app using Atlassian Bamboo. Bamboo does continuous delivery of the project from source code to deployment. It has stages including Build, Test and Deploy.

Software teams in every industry are upgrading their continuous delivery pipeline with Bamboo. Easy build import from popular open source tools, user and group import from JIRA, seamless integration with Bitbucket, and native support for Git, Hg, and SVN means you'll be building and deploying like a champ.

Box 3: GitHub repositories -

Bitbucket can be used as the Git repository, but you can use any other Git repository (Like TFS Git) for source control of the code.

Reference:

https://www.trustradius.com/compare-products/azure-devops-services-vs-trello https://marketplace.visualstudio.com/items?itemName=ms-vsts.vss-services-bamboo https://www.c-sharpcorner.com/article/cicd-implementation-for-an-azure-function-app-using-atlassian-bamboo-server/

Case Study Description

Woodgrove bank

Overview

This is a case study. Case studies are not timed separately. You can use as much exam time as you would like to complete each case. However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other questions in this case study.

At the end of this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next section of the exam. After you begin a new section, you cannot return to this section.

To start the case study

To display the first question in this case study, click the Next button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an All Information tab, note that the information displayed is identical to the information displayed on the

subsequent tabs. When you are ready to answer a question, click the Question button to return to the question.

Overview

General Overview

Woodgrove Bank is a financial services company that has a main office in the United Kingdom.

Technical Requirements and Planned Changes

Planned Changes

Woodgrove Bank plans to implement the following project management changes:

✑ Implement Azure DevOps for project tracking.

✑ Centralize source code control in private GitHub repositories.

✑ Implement Azure Pipelines for build pipelines and release pipelines.

Woodgrove Bank plans to implement the following changes to the identity environment:

✑ Deploy an Azure AD tenant named woodgrovebank.com.

✑ Sync the Active Directory domain to Azure AD.

✑ Configure App1 to use a service principal.

✑ Integrate GitHub with Azure AD.

Woodgrove Bank plans to implement the following changes to the core apps:

✑ Migrate App1 to ASP.NET Core.

✑ Integrate Azure Pipelines and the third-party build tool used to develop App2.

Woodgrove Bank plans to implement the following changes to the DevOps environment:

✑ Deploy App1 to Azure App Service.

✑ Implement source control for the DB1 schema.

✑ Migrate all the source code from TFS1 to GitHub.

✑ Deploy App2 to an Azure virtual machine named VM1.

✑ Merge the POC branch into the GitHub default branch.

✑ Implement an Azure DevOps dashboard for stakeholders to monitor development progress.

Technical Requirements

Woodgrove Bank identifies the following technical requirements:

✑ The initial databases for new environments must contain both schema and reference data.

✑ An Azure Monitor alert for VM1 must be configured to meet the following requirements:

✑ The commit history of the POC branch must replace the history of the default branch.

✑ The Azure DevOps dashboard must display the metrics shown in the following table.

Number	Required data
1	A comparison between the work the development team planned to deliver and what was delivered
2	The status of the environments in a release definition
3	The total number of results from a work item query

✑ Access to Azure DevOps must be restricted to specific IP addresses.

✑ Page load times for App1 must be captured and monitored.

✑ Administrative effort must be minimized.

Question: 580 CertyIQ

You need to the merge the POC branch into the default branch. The solution must meet the technical requirements.
Which command should you run?

A.git rebase
B.git merge --squash
C.git push
D.git merge --allow-unrelated-histories

Answer: A

Explanation:

The correct answer is **A. git rebase**. Here's a detailed justification:

The goal is to integrate changes from the POC branch into the default branch while adhering to technical requirements (which are not fully specified in the prompt, but commonly relate to maintaining a clean and linear commit history).

git rebase achieves this by moving the entire POC branch onto the tip of the default branch. This rewrites the commit history of the POC branch, making it appear as if the POC branch was branched off the latest commit in default. This results in a linear history, often preferred for readability and easier debugging.

git merge --squash would combine all changes from the POC branch into a single commit on the default branch. While this simplifies the history, it loses the individual commit details and might not be desirable if the original

commit messages in POC are important. Also, it doesn't move the POC branch but rather collapses its changes into one commit.

git push uploads local branch commits to a remote repository; it doesn't merge branches.

git merge --allow-unrelated-histories is used when the branches don't share a common ancestor, which isn't implied in this scenario. Using it when not needed introduces unnecessary complexity.

Therefore, git rebase is the most appropriate command when a clean, linear history is desired after integrating changes from one branch into another. It efficiently integrates the work while maintaining a clear and understandable commit log.

Steps involved with rebase (typically):

1. git checkout POC
2. git rebase default
3. Address any conflicts during the rebase process.
4. git checkout default
5. git merge POC (Now this merge will be a fast-forward merge)

The technical requirements, which are not fully specified in the question but implied to prefer a cleaner history, are best satisfied by using git rebase.

Authoritative Links:

Git Rebase: https://git-scm.com/docs/git-rebase
Git Merge: https://git-scm.com/docs/git-merge
Git Branching - Basic Branching and Merging: https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Question: 581 CertyIQ

You need to perform the GitHub code migration. The solution must support the planned changes for the DevOps environment.
What should you use?

A.git clone
B.GitHub Importer
C.Import repository in Azure Repos
D.git-tfs

Answer: B

Explanation:

The correct answer is **B. GitHub Importer**. Here's why:

The core requirement is to migrate code from GitHub to Azure DevOps, supporting planned changes for the DevOps environment. GitHub Importer is specifically designed for this scenario, facilitating the transfer of repositories directly from GitHub to Azure DevOps Services. It simplifies the process by handling the complexities of importing the repository history, branches, and tags.

GitHub Importer (B): This tool is built explicitly for migrating repositories from GitHub (or other Git-based systems) to Azure DevOps. It preserves the commit history, branches, and tags, crucial for maintaining the

integrity of the source code and its development history. The solution readily adapts to DevOps environment planning due to its dedicated migration features. [https://learn.microsoft.com/en-us/azure/devops/repos/git/import-git-repository?view=azure-devops]

git clone (A): While git clone can copy a repository, it needs further configuration and steps to be fully integrated into Azure DevOps as a properly managed repository. Cloning alone doesn't automatically set up the necessary DevOps environment configurations like build pipelines, release pipelines, or work item integration. It's more of a raw data transfer tool and not a dedicated migration solution.

Import repository in Azure Repos (C): While this option seems similar to the correct one, "Import repository" within Azure Repos is a more generic option that may support importing from various sources but might not have the focused optimizations and features for a GitHub-specific migration that GitHub Importer provides. Furthermore, the GitHub Importer is a specialized feature within Azure DevOps that enhances the standard import functionality with GitHub-centric settings. [https://learn.microsoft.com/en-us/azure/devops/repos/git/import-git-repository?view=azure-devops]

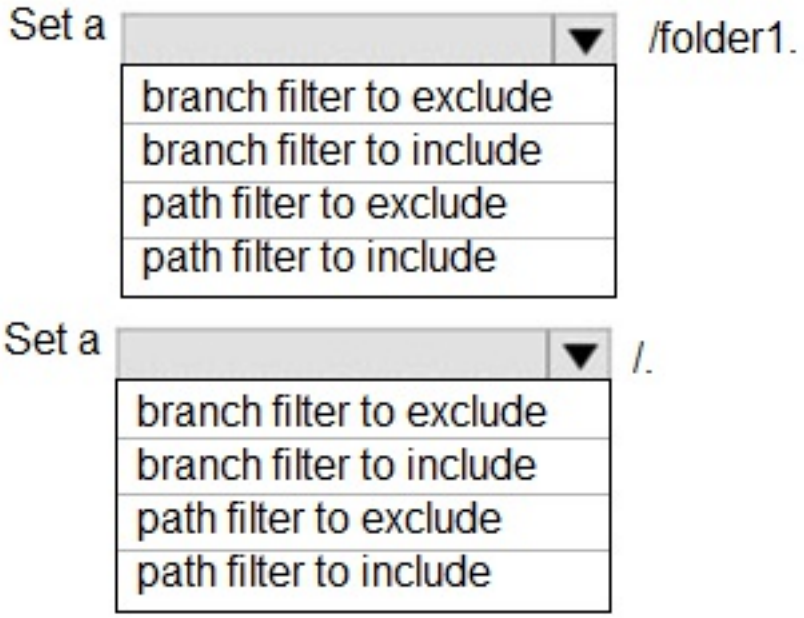
git-tfs (D): git-tfs is used to interact with Team Foundation Version Control (TFVC), a centralized version control system, and is completely irrelevant to GitHub which is a distributed version control system. It's designed for migrating from TFVC to Git. Therefore, it's not applicable when migrating from GitHub.

In summary, GitHub Importer provides a streamlined, optimized, and feature-rich way to migrate code from GitHub to Azure DevOps, which directly addresses the problem statement's needs and supports planned DevOps environment changes.

Question: 582 CertyIQ

HOTSPOT -
How should you configure the filters for the Project5 trigger? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

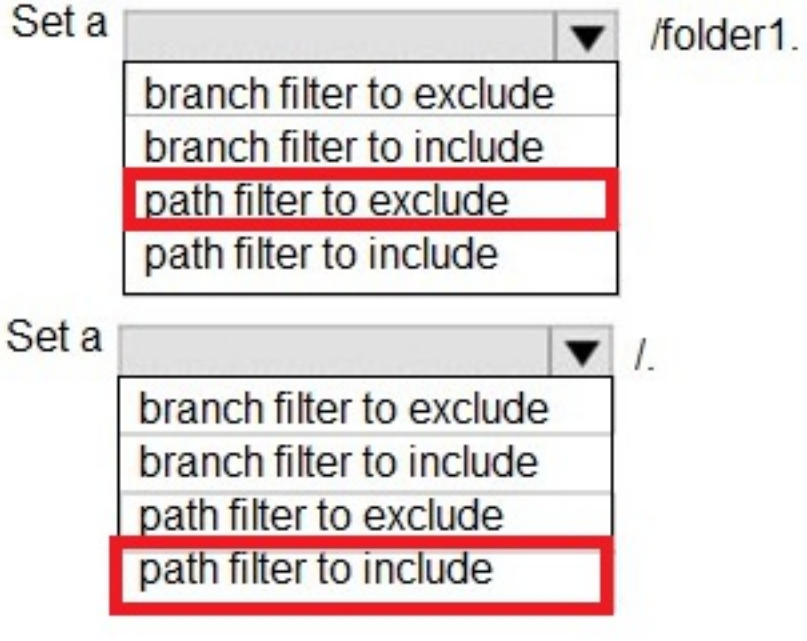
Answer Area



@

Answer:

Answer Area



@

Explanation:

Path filter to exclude

path filter to include

Case Study Description

Contoso Case Study: 2

Overview

Existing Environment

Contoso, Ltd. is a manufacturing company that has a main office in Chicago.

Contoso plans to improve its IT development and operations processes by implementing Azure DevOps principles. Contoso has an Azure subscription and creates an Azure DevOps organization.

The Azure DevOps organization includes:

• The Docker extension

• A deployment pool named Pool7 that contains 10 Azure virtual machines that run Windows Server 2016

The Azure subscription contains an Azure Automation account.

Planned Changes

Contoso plans to create projects in Azure DevOps as shown in the following table.

Project name	Project details
Project 1	Project1 will provide support for incremental builds and third-party SDK components
Project 2	Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.
Project 3	Project3 will be integrated with SonarQube
Project 4	Project4 will provide support for a build pipeline that creates a Docker image and pushes the image to the Azure Container Registry. Project4 will use an existing Dockerfile.
Project 5	Project5 will contain a Git repository in Azure Reports and a continuous integration trigger that will initiate a build in response to any change except for changes within /folder1 of the repository.
Project 6	Project6 will provide support for build and deployment pipelines. Deployment will be allowed only if the number of current work items representing active software bugs is 0.
Project 7	Project7 will contain a target deployment group named Group7 that maps to Pool7. Project7 will use Azure Automation State Configuration to maintain the desired state of the computers in Group7.

Technical Requirements

Contoso identities the following technical requirements:

• Implement build agents rot Project 1.

• Whenever possible, use Azure resources

• Avoid using deprecated technologies

• Implement a code flow strategy for Project2 that will:

• Enable Team 2 to submit pull requests for Project2.

• Enable Team 2 to work independently on changes to a copy of Project?

• Ensure that any intermediary changes performed by Tram2 on a copy of Project2 will be subject to the same restrictions as the ones defied in the build policy of Project2.

• Whenever possible. Implement automation and minimize administrative effort.

• Implement Protect3, Project5, Project6, and Project7 based on the planned changes.

• Implement Project4 and configure the project to push Docker images to Azure Container Reentry.

Question: 583

CertyIQ

In Azure DevOps, you create Project3.
You need to meet the requirements of the project.
What should you do first?

A.From Azure DevOps, modify the build definition.
B.From SonarQube, obtain an authentication token.
C.From Azure DevOps, create a service endpoint.
D.From SonarQube, create a project.

Answer: B

Explanation:

The correct first step is to obtain an authentication token from SonarQube (Option B).

Here's why: The problem description states you need to meet the requirements of Project3. While the specifics are not mentioned, the presence of SonarQube strongly suggests a code quality requirement. SonarQube analyzes code for bugs, vulnerabilities, and code smells. To integrate Azure DevOps with SonarQube, Azure DevOps needs proper credentials to authenticate and communicate with the SonarQube server. Obtaining an authentication token from SonarQube is the first step to establish this secure connection. This token will be used later when configuring a service connection in Azure DevOps. The service connection allows Azure DevOps to securely run SonarQube analysis tasks within pipelines.

Creating a service endpoint in Azure DevOps (Option C) is an important subsequent step, but it requires the authentication token first. Creating a project in SonarQube (Option D) is usually done, but not before obtaining the token because the token is necessary for any kind of automation and integration. Modifying the build definition in Azure DevOps (Option A) to include SonarQube analysis tasks is also a later step that relies on a functional service connection.

Therefore, acquiring the authentication token from SonarQube is the prerequisite and initial step needed to enable the subsequent integration and configuration steps in Azure DevOps to meet the project requirements.

This step enables secure communication between the two services, enabling the use of SonarQube within the Azure DevOps workflow.

For further research:

Azure DevOps Service Connections: https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml
SonarQube Documentation: https://docs.sonarsource.com/

Question: 584 CertyIQ

You need to implement Project4.
What should you do first?

A.Add the FROM instruction in the Dockerfile file.
B.Add a Copy and Publish Build Artifacts task to the build pipeline.
C.Add a Docker task to the build pipeline.
D.Add the MAINTAINER instruction in the Dockerfile file.

Answer: C

Explanation:

The correct first step in implementing Project4, presumably a project involving Docker and Azure DevOps, is to add a Docker task to the build pipeline (Option C). Here's why:

Before any Docker-specific operations like copying files, publishing artifacts, or defining metadata in the Dockerfile, the build pipeline needs to be explicitly configured to interact with Docker. The Docker task provides the necessary connection and configuration to build, push, and manage Docker images within the Azure DevOps environment. It's the foundational step for Docker-related activities.

The Docker task acts as a bridge, allowing the build pipeline to leverage the Docker engine for image creation and management. Without a Docker task, Azure DevOps wouldn't know how to execute Docker commands, interact with a Docker registry, or utilize the Dockerfile for building images. Think of it as installing the Docker CLI on the build agent and configuring it for use by the pipeline.

Options A and D, concerning the FROM and MAINTAINER instructions in the Dockerfile, relate to the content of the Dockerfile itself. While important for the Docker image's configuration, they are subsequent steps. The Dockerfile is utilized by the Docker task. Option B, adding Copy and Publish Build Artifacts, deals with moving files to the agent's file system (Copy) and publishing these to a repository, which is not necessarily Docker related.

Therefore, initiating the Docker integration with a dedicated Docker task is paramount before configuring the Dockerfile or handling artifacts specifically for Docker image creation and deployment. This establishes the environment required for all subsequent Docker-related steps in the pipeline.

Here are some resources that may be helpful:

Azure Pipelines Documentation on Docker Task: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/build/docker?view=azure-devops
Microsoft Learn on CI/CD with Azure DevOps: https://learn.microsoft.com/en-us/training/paths/az-400-implement-cicd-with-azure-devops/

Question: 585 CertyIQ

DRAG DROP -
You need to recommend a procedure to implement the build agent for Project1.
Which three actions should you recommend be performed in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:

Actions	Answer Area
Sign in to Azure DevOps by using an account that is assigned the Administrator service connection security role.	
Install the Azure Pipelines agent on on-premises virtual machine.	
Create a personal access token in the Azure DevOps organization of Contoso.	
Install and register the Azure Pipelines agent on an Azure virtual machine.	
Sign in to Azure DevOps by using an account that is assigned the agent pool administrator role.	

Answer:

Actions	Answer Area
Sign in to Azure DevOps by using an account that is assigned the Administrator service connection security role.	Sign in to Azure DevOps by using an account that is assigned the Administrator service connection security role.
Install the Azure Pipelines agent on on-premises virtual machine.	Create a personal access token in the Azure DevOps organization of Contoso.
Create a personal access token in the Azure DevOps organization of Contoso.	Install and register the Azure Pipelines agent on an Azure virtual machine.
Install and register the Azure Pipelines agent on an Azure virtual machine.	
Sign in to Azure DevOps by using an account that is assigned the agent pool administrator role.	

Explanation:

Scenario:

Project 1	Project1 will provide support for incremental builds and third-party SDK components

Step 1: Sign in to Azure Devops by using an account that is assigned the Administrator service connection security role.

Note: Under Agent Phase, click Deploy Service Fabric Application. Click Docker Settings and then click Configure Docker settings. In Registry Credentials Source, select Azure Resource Manager Service Connection. Then select your Azure subscription.

Step 2: Create a personal access token..

A personal access token or PAT is required so that a machine can join the pool created with the Agent Pools (read, manage) scope.

Step 3: Install and register the Azure Pipelines agent on an Azure virtual machine.

By running a Azure Pipeline agent in the cluster, we make it possible to test any service, regardless of type.

Reference:

https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-tutorial-deploy-container-app-with-cicd-vsts https://mohitgoyal.co/2019/01/10/run-azure-devops-private-agents-in-kubernetes-clusters/

Case Study Description

Contoso Case Study: 2

Overview

Existing Environment

Contoso, Ltd. is a manufacturing company that has a main office in Chicago.

Contoso plans to improve its IT development and operations processes by implementing Azure DevOps principles. Contoso has an Azure subscription and creates an Azure DevOps organization.

The Azure DevOps organization includes:

• The Docker extension

• A deployment pool named Pool7 that contains 10 Azure virtual machines that run Windows Server 2016

The Azure subscription contains an Azure Automation account.

Planned Changes

Contoso plans to create projects in Azure DevOps as shown in the following table.

Project name	Project details
Project 1	Project1 will provide support for incremental builds and third-party SDK components
Project 2	Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.
Project 3	Project3 will be integrated with SonarQube
Project 4	Project4 will provide support for a build pipeline that creates a Docker image and pushes the image to the Azure Container Registry. Project4 will use an existing Dockerfile.
Project 5	Project5 will contain a Git repository in Azure Reports and a continuous integration trigger that will initiate a build in response to any change except for changes within /folder1 of the repository.
Project 6	Project6 will provide support for build and deployment pipelines. Deployment will be allowed only if the number of current work items representing active software bugs is 0.
Project 7	Project7 will contain a target deployment group named Group7 that maps to Pool7. Project7 will use Azure Automation State Configuration to maintain the desired state of the computers in Group7.

Technical Requirements

Contoso identities the following technical requirements:

• Implement build agents rot Project 1.

• Whenever possible, use Azure resources

• Avoid using deprecated technologies

• Implement a code flow strategy for Project2 that will:

• Enable Team 2 to submit pull requests for Project2.

• Enable Team 2 to work independently on changes to a copy of Project?

• Ensure that any intermediary changes performed by Tram2 on a copy of Project2 will be subject to the same restrictions as the ones defied in the build policy of Project2.

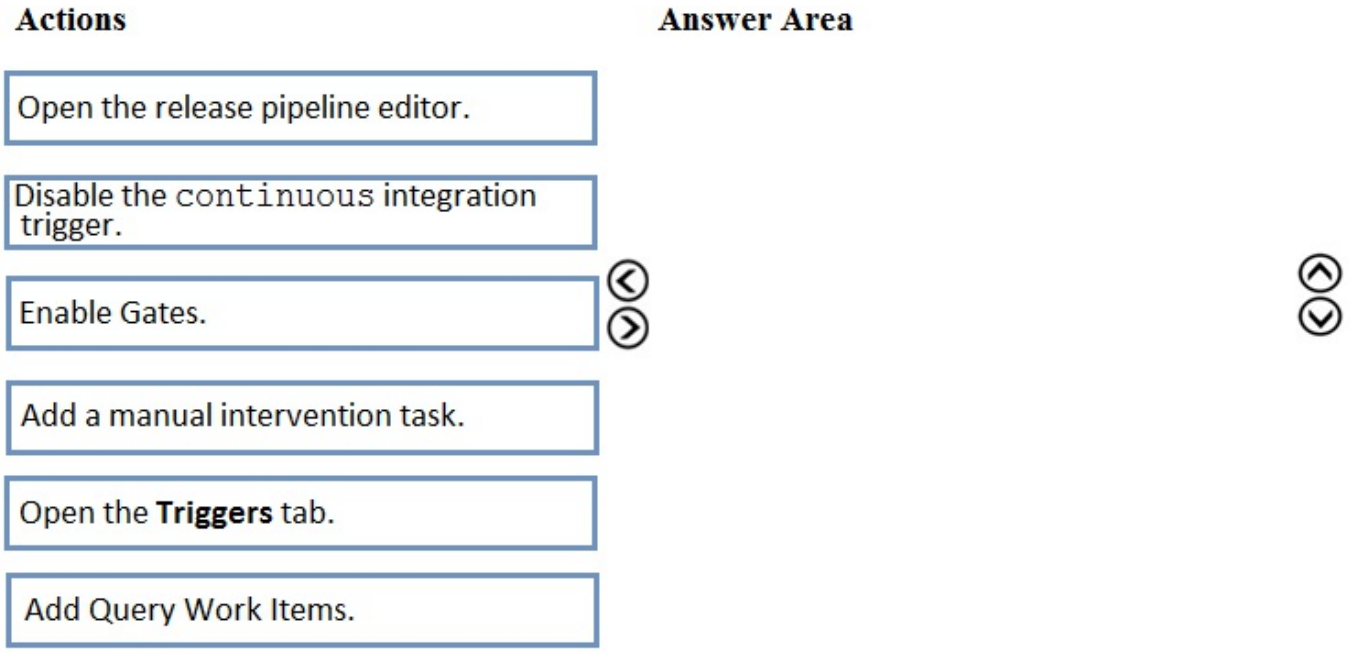
• Whenever possible. Implement automation and minimize administrative effort.

• Implement Protect3, Project5, Project6, and Project7 based on the planned changes.

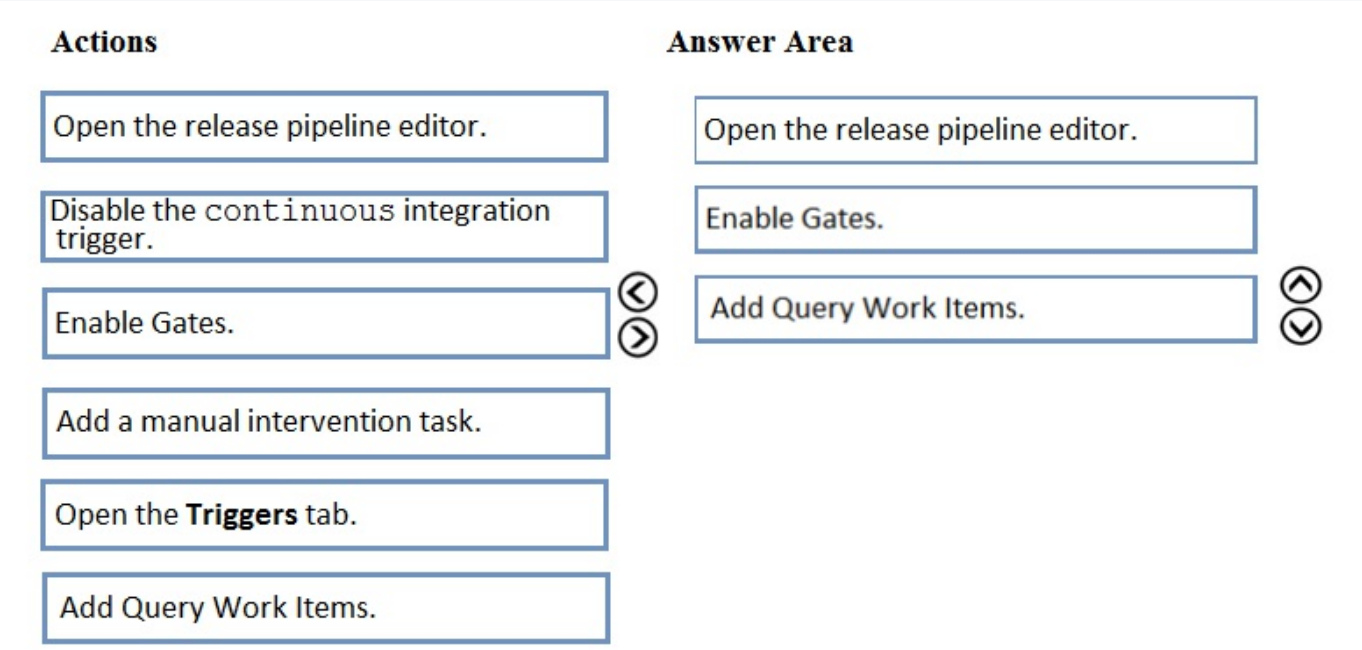
• Implement Project4 and configure the project to push Docker images to Azure Container Reentry.

Question: 586 CertyIQ

DRAG DROP -
You need to implement Project6.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Scenario: Implement Project3, Project5, Project6, and Project7 based on the planned changes

Project 6	Project6 will provide support for build and deployment pipelines. Deployment will be allowed only if the number of current work items representing active software bugs is 0.

Step 1: Open the release pipeline editor.

In the Releases tab of Azure Pipelines, select your release pipeline and choose Edit to open the pipeline editor.

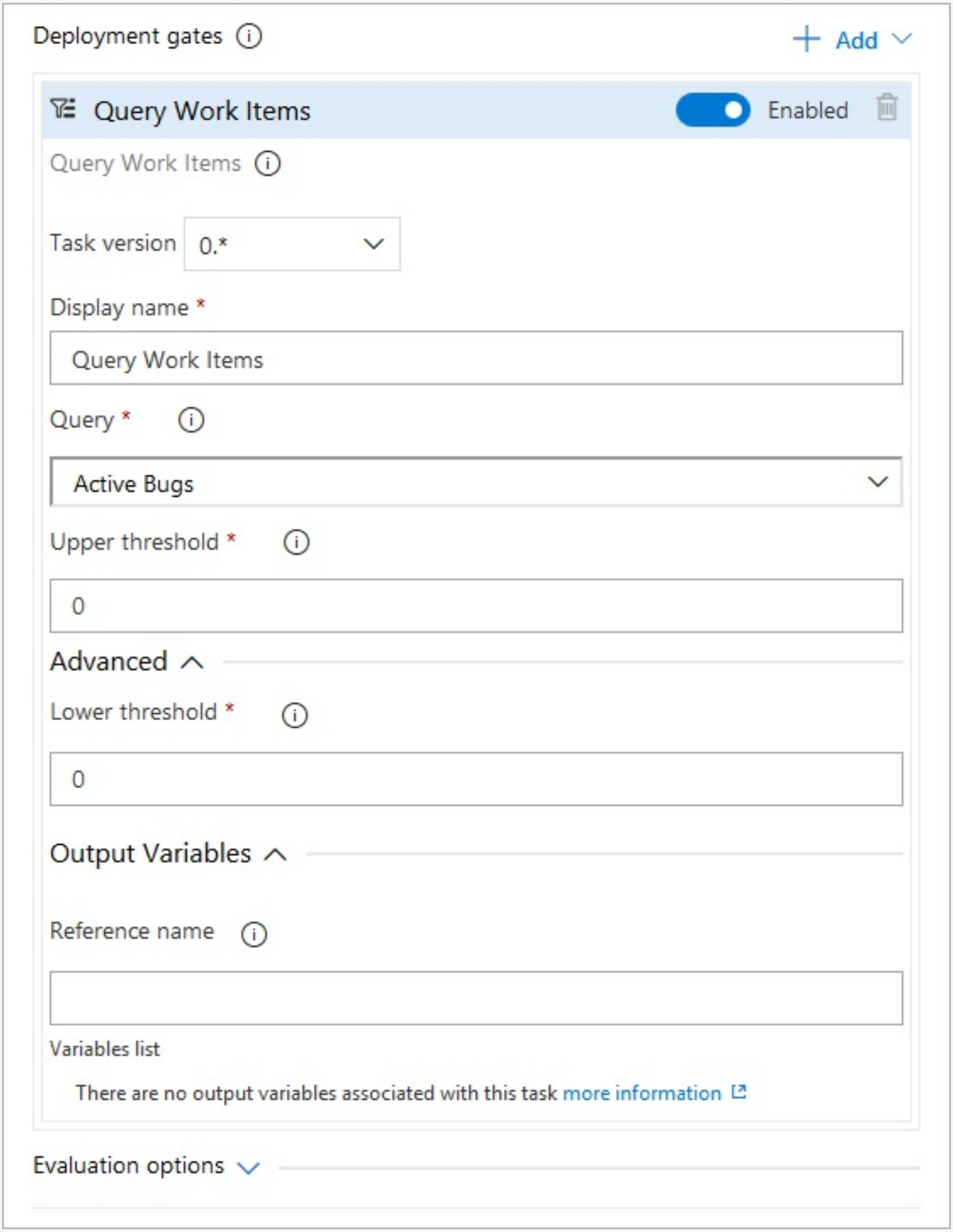
Step 2: Enable Gates.

Choose the pre-deployment conditions icon for the Production stage to open the conditions panel. Enable gates by using the switch control in the Gates section.

Step 3: Add Query Work items.

Choose + Add and select the Query Work Items gate.

Configure the gate by selecting an existing work item query.



Note: A case for release gate is:

Incident and issues management. Ensure the required status for work items, incidents, and issues. For example, ensure deployment occurs only if no priority zero bugs exist, and validation that there are no active

incidents takes place after deployment.

Reference:

https://docs.microsoft.com/en-us/azure/devops/pipelines/release/deploy-using-approvals?view=azure-devops#configure-gate

---

Case Study Description

Contoso Case Study: 2

Overview

Existing Environment

Contoso, Ltd. is a manufacturing company that has a main office in Chicago.

Contoso plans to improve its IT development and operations processes by implementing Azure DevOps principles. Contoso has an Azure subscription and creates an Azure DevOps organization.

The Azure DevOps organization includes:

• The Docker extension

• A deployment pool named Pool7 that contains 10 Azure virtual machines that run Windows Server 2016

The Azure subscription contains an Azure Automation account.

Planned Changes

Contoso plans to create projects in Azure DevOps as shown in the following table.

Project name	Project details
Project 1	Project1 will provide support for incremental builds and third-party SDK components
Project 2	Project2 will use an automatic build policy. A small team of developers named Team2 will work independently on changes to the project. The Team2 members will not have permissions to Project2.
Project 3	Project3 will be integrated with SonarQube
Project 4	Project4 will provide support for a build pipeline that creates a Docker image and pushes the image to the Azure Container Registry. Project4 will use an existing Dockerfile.
Project 5	Project5 will contain a Git repository in Azure Reports and a continuous integration trigger that will initiate a build in response to any change except for changes within /folder1 of the repository.
Project 6	Project6 will provide support for build and deployment pipelines. Deployment will be allowed only if the number of current work items representing active software bugs is 0.
Project 7	Project7 will contain a target deployment group named Group7 that maps to Pool7. Project7 will use Azure Automation State Configuration to maintain the desired state of the computers in Group7.

Technical Requirements

Contoso identities the following technical requirements:

• Implement build agents rot Project 1.

• Whenever possible, use Azure resources

• Avoid using deprecated technologies

• Implement a code flow strategy for Project2 that will:

• Enable Team 2 to submit pull requests for Project2.

• Enable Team 2 to work independently on changes to a copy of Project?

• Ensure that any intermediary changes performed by Tram2 on a copy of Project2 will be subject to the same restrictions as the ones defied in the build policy of Project2.

• Whenever possible. Implement automation and minimize administrative effort.

• Implement Protect3, Project5, Project6, and Project7 based on the planned changes.

• Implement Project4 and configure the project to push Docker images to Azure Container Reentry.

Question: 587 CertyIQ

To resolve the current technical issue, what should you do to the Register-AzureRmAutomationDscNode command?

A.Change the value of the ConfigurationMode parameter.

B.Replace the Register-AzureRmAutomationDscNode cmdlet with Register-AzureRmAutomationScheduledRunbook

C.Add the AllowModuleOverwrite parameter.

D.Add the DefaultProfile parameter.

Answer: A

Explanation:

The most appropriate action to address a technical issue related to Register-AzureRmAutomationDscNode often involves adjusting the ConfigurationMode parameter. Desired State Configuration (DSC) nodes operate under different configuration modes, the most common being ApplyOnly, ApplyAndMonitor, and ApplyAndAutoCorrect. The specific error encountered might stem from the node attempting to reconfigure itself in a way not allowed by its current mode. For example, if the node is in ApplyOnly and a change is detected, it won't automatically correct it, potentially leading to failures or inconsistencies. Changing the ConfigurationMode allows you to define how the node should handle deviations from the desired state. ApplyAndMonitor detects deviations and logs them, while ApplyAndAutoCorrect automatically corrects any deviations detected. If the node is erroneously attempting automatic correction when it shouldn't (or vice-versa), altering this parameter rectifies the behavior. Using a different cmdlet like Register-AzureRmAutomationScheduledRunbook (option B) is for triggering runbooks, not for managing DSC nodes' configuration. AllowModuleOverwrite (option C) addresses module conflicts, and DefaultProfile (option D) specifies the Azure profile, which aren't directly related to DSC configuration mode issues. Therefore,

adjusting the ConfigurationMode offers the most direct and effective method for resolving configuration-related issues within the DSC node registration process.

Further Research:

Azure Automation DSC Overview: https://learn.microsoft.com/en-us/azure/automation/automation-dsc-overview
Register-AzureRmAutomationDscNode Cmdlet: https://learn.microsoft.com/en-us/powershell/module/azurerm.automation/register-azurermautomationdscnode?view=azurermps-6.13.0
Desired State Configuration Concepts: https://learn.microsoft.com/en-us/powershell/dsc/overview

Question: 588 CertyIQ

Which branching strategy should you recommend for the investment planning applications suite?

A.release isolation
B.main only
C.development isolation
D.feature isolation

Answer: D

Explanation:

Here's a detailed justification for recommending the **Feature Isolation** branching strategy for the investment planning applications suite:

The best branching strategy depends on the project's size, team structure, release cadence, and risk tolerance. Feature Isolation, often realized using Gitflow or GitHub Flow variations, is well-suited for projects like investment planning applications for several reasons.

Rapid Development and Delivery: Feature branches allow developers to work on individual features in complete isolation without directly affecting the main codebase or other features under development. This promotes faster development cycles, as developers can test and refine their code without introducing conflicts.

Reduced Risk: Since each feature is developed in its own branch, any bugs or issues introduced are isolated to that branch, minimizing the risk of destabilizing the entire application. This is particularly crucial for financial applications where stability and reliability are paramount.

Simplified Code Review: Feature branches facilitate easier code review processes. Reviewers can focus solely on the changes made within the specific feature branch, ensuring thorough and accurate code quality checks.

Continuous Integration and Continuous Delivery (CI/CD) Compatibility: Feature branches integrate seamlessly with CI/CD pipelines. Each branch can be automatically built, tested, and deployed to a staging environment for comprehensive testing before merging into the main branch. This ensures higher code quality and facilitates faster releases.

Rollback Capability: In the event of a faulty feature introduction, the feature branch can be easily reverted, minimizing the impact on production and user experience.

Team Collaboration: Feature branches allow multiple developers to work on different features concurrently

without interfering with each other's progress. The branching structure provides clear boundaries and promotes collaboration within the team.

Other strategies are less appropriate. Release Isolation is often reserved for maintenance releases, not core feature development. Main Only doesn't scale well for teams and complex applications, leading to integration issues and instability. Development Isolation is not a common or well-defined branching strategy. Feature Isolation promotes a more agile and flexible development process, enabling faster delivery of new features and enhancements while maintaining the stability and integrity of the investment planning applications.

Authoritative Links for Further Research:

Gitflow Workflow: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
GitHub Flow: https://guides.github.com/introduction/flow/
Branching Strategies: https://trunkbaseddevelopment.com/

Question: 589 CertyIQ

What should you use to implement the code quality restriction on the release pipeline for the investment planning applications suite?

A.a pre-deployment approval
B.a deployment gate
C.a post-deployment approval
D.a trigger

Answer: B

Explanation:

The correct answer is **B. a deployment gate**.

Deployment gates in Azure DevOps pipelines are specifically designed to evaluate conditions and policies before a deployment stage begins. This allows you to enforce quality standards, security requirements, or other criteria directly within the release pipeline. If the conditions set in the gate are not met, the deployment will be paused, preventing faulty code or configuration from being released. In the context of code quality restriction, you can integrate tools like static code analysis or unit test execution into a deployment gate. The gate will check the results of these tools (e.g., number of errors, test coverage percentage) and only proceed with the deployment if the results meet your predefined thresholds. This directly addresses the need to implement a code quality restriction on the release pipeline.

Pre-deployment approvals (A) require manual intervention, which doesn't automate the code quality check. Post-deployment approvals (C) happen after deployment, defeating the purpose of preventing flawed code from reaching the environment. Triggers (D) are event-driven mechanisms that start pipelines, not enforce policies during execution.

Therefore, deployment gates offer the most effective and automated mechanism to implement code quality restrictions before deployment, ensuring that only compliant code is released to the target environment.

For more information, you can consult the official Microsoft Azure DevOps documentation:

Using gates and approvals to control your deployments: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/approvals/gates?view=azure-devops
Define approvals and checks: https://learn.microsoft.com/en-us/azure/devops/pipelines/process/approvals?

view=azure-devops&tabs=check-types

Question: 590 CertyIQ

HOTSPOT -
How should you configure the release retention policy for the investment planning applications suite? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

Global release:

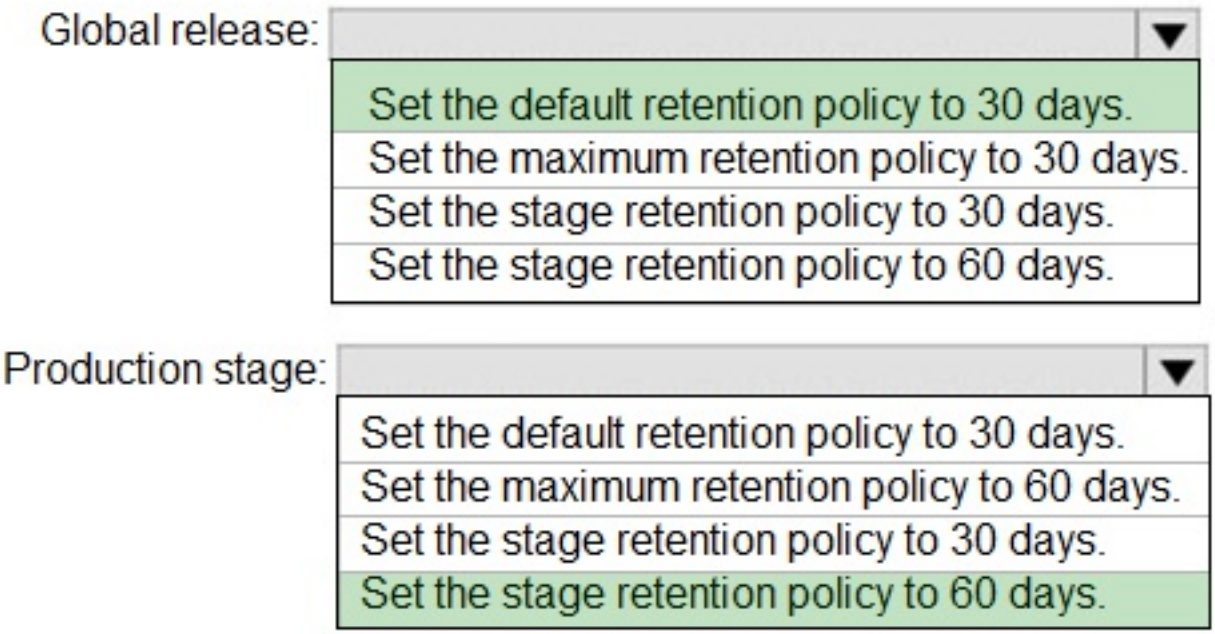
- Set the default retention policy to 30 days.
- Set the maximum retention policy to 30 days.
- Set the stage retention policy to 30 days.
- Set the stage retention policy to 60 days.

Production stage:

- Set the default retention policy to 30 days.
- Set the maximum retention policy to 60 days.
- Set the stage retention policy to 30 days.
- Set the stage retention policy to 60 days.

Answer:

Answer Area



Explanation:

Scenario: By default, all releases must remain available for 30 days, except for production releases, which must be kept for 60 days.

Box 1: Set the default retention policy to 30 days

The Global default retention policy sets the default retention values for all the build pipelines. Authors of build pipelines can override these values.

Box 2: Set the stage retention policy to 60 days

You may want to retain more releases that have been deployed to specific stages.

Reference:

https://docs.microsoft.com/en-us/azure/devops/pipelines/policies/retention

---

Case Study Description

Litware inc. Case Study: 1

Overview

Existing Environment

Litware, Inc. an independent software vendor (ISV) Litware has a main office and five branch offices.

Application Architecture

The company' s primary application is a single monolithic retirement fund management system based on ASP.NE T web forms that use logic written in V8.NET. Some new sections of the application are written in C#.

Variations of the application are created for individual customers. Currently, there are more than 80 have code branches in the application's code base.

The application was developed by using Microsoft Visual Studio. Source code is stored in Team Foundation Server (TFS) in the main office. The branch offices access of the source code by using TFS proxy servers.

Architectural Issues

Litware focuses on writing new code for customers. No resources are provided to refactor or remove existing code. Changes to the code base take a long time, AS dependencies are not obvious to individual developers.

Merge operations of the code often take months and involve many developers. Code merging frequently introduces bugs that are difficult to locate and resolve.

Customers report that ownership costs of the retirement fund management system increase continually. The need to merge unrelated code makes even minor code changes expensive.

Requirements

Planned Changes

Litware plans to develop a new suite of applications for investment planning. The investment planning Applications will require only minor integration with the easting retirement fund management system.

The investment planning applications suite will include one multi-tier web application and two iOS mobile applications. One mobile application will be used by employees; the other will be used by customers.

Litware plans to move to a more agile development methodology. Shared code will be extracted into a series of package.

Litware has started an internal cloud transformation process and plans to use cloud based services whenever suitable.

Litware wants to become proactive m detecting failures, rather than always waning for customer bug reports.

Technical Requirements

The company's investment planning applications suite must meet the following technical requirements:

• New incoming connections through the firewall must be minimized.

• Members of a group named Developers must be able to install packages.

• The principle of least privilege must be used for all permission assignments

• A branching strategy that supports developing new functionality in isolation must be used.

• Members of a group named Team leaders must be able to create new packages and edit the permissions of package feeds

• Visual Studio App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• By default, all App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• Code quality and release quality are critical. During release, deployments must not proceed between stages if any active bugs are logged against the release.

• The mobile applications must be able to call the share pricing service of the existing retirement fund management system. Until the system is upgraded, the service will only support basic authentication over HUPS.

• The required operating system configuration tor the test servers changes weekly. Azure Automation State Configuration must be used to ensure that the operating system on each test servers configured the same way when the servers are created and checked periodically.

Current Technical

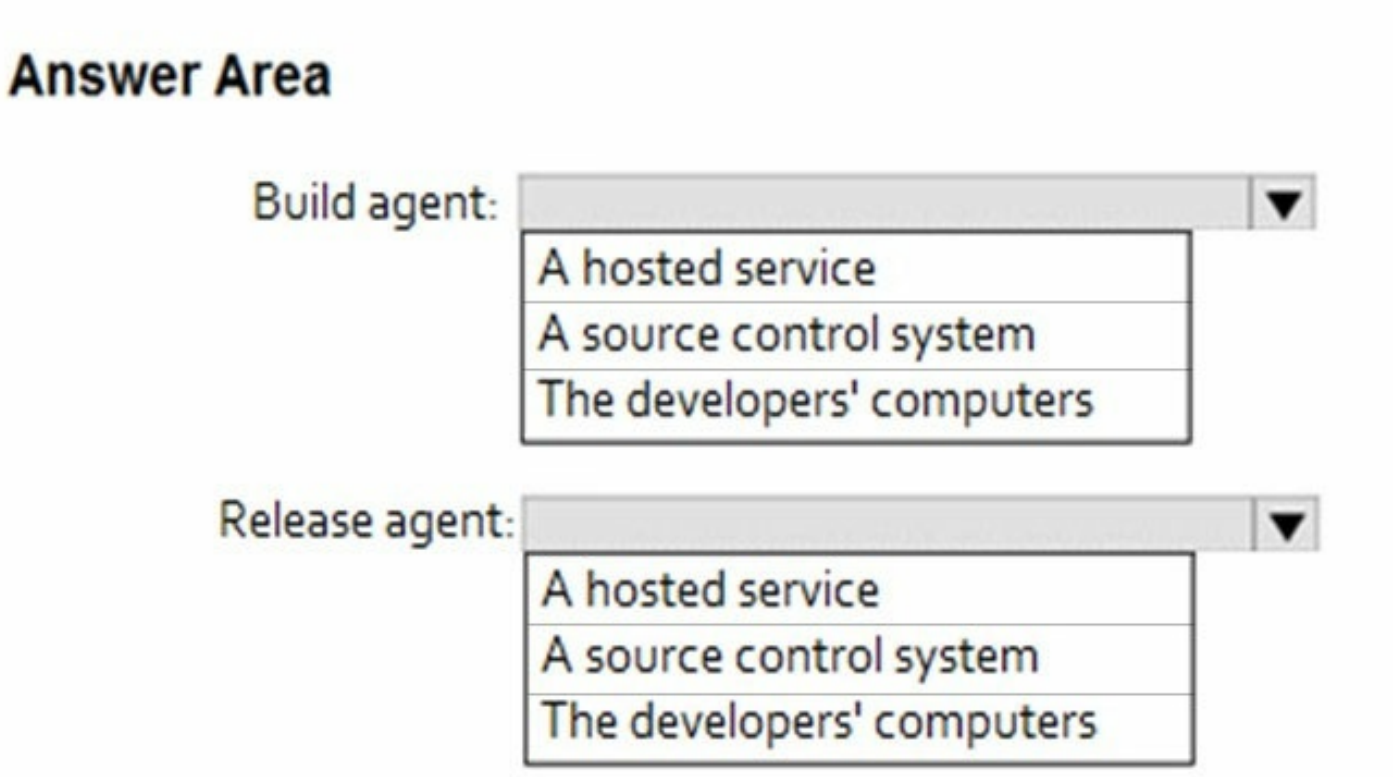
The test servers are configured correctly when first deployed, but they experience configuration drift over time. Azure Automation State Configuration fails to correct the configurations.

Azure Automation State Configuration nodes are registered by using the following command.

```
Register-AzureRmAutomationDscNode
  -ResourceGroupName 'TestResourceGroup'
  -AutomationAccountName 'LitwareAutomationAccount'
  -AzureVMName $vname
  -ConfigurationMode 'ApplyOnly'
```

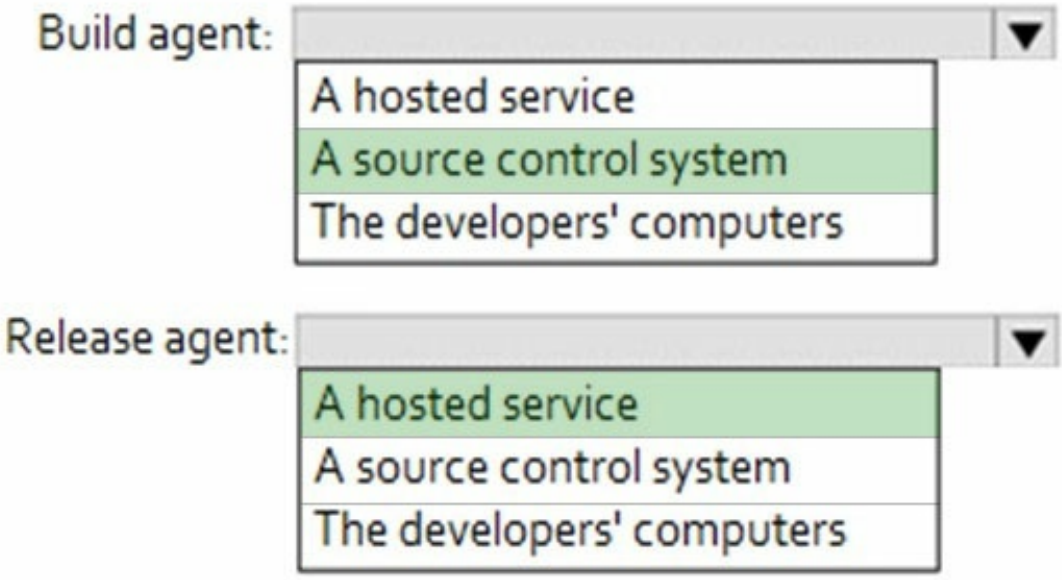
Question: 591 CertyIQ

HOTSPOT -
Where should the build and release agents for the investment planning applications suite run? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:



Answer:

Answer Area



Explanation:

Box 1: A source control system -

A source control system, also called a version control system, allows developers to collaborate on code and track changes. Source control is an essential tool for multi-developer projects.

Box 2: A hosted service -

To build and deploy Xcode apps or Xamarin.iOS projects, you'll need at least one macOS agent. If your pipelines are in Azure Pipelines and a Microsoft-hosted agent meets your needs, you can skip setting up a self-hosted macOS agent.

Scenario: The investment planning applications suite will include one multi-tier web application and two iOS mobile applications. One mobile application will be used by employees; the other will be used by customers.

Reference:

https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-osx?view=azure-devops

---

Case Study Description

Litware inc. Case Study: 1

Overview

Existing Environment

Litware, Inc. an independent software vendor (ISV) Litware has a main office and five branch offices.

Application Architecture

The company' s primary application is a single monolithic retirement fund management system based on

ASP.NE T web forms that use logic written in V8.NET. Some new sections of the application are written in C#.

Variations of the application are created for individual customers. Currently, there are more than 80 have code branches in the application's code base.

The application was developed by using Microsoft Visual Studio. Source code is stored in Team Foundation Server (TFS) in the main office. The branch offices access of the source code by using TFS proxy servers.

Architectural Issues

Litware focuses on writing new code for customers. No resources are provided to refactor or remove existing code. Changes to the code base take a long time, AS dependencies are not obvious to individual developers.

Merge operations of the code often take months and involve many developers. Code merging frequently introduces bugs that are difficult to locate and resolve.

Customers report that ownership costs of the retirement fund management system increase continually. The need to merge unrelated code makes even minor code changes expensive.

Requirements

Planned Changes

Litware plans to develop a new suite of applications for investment planning. The investment planning Applications will require only minor integration with the easting retirement fund management system.

The investment planning applications suite will include one multi-tier web application and two iOS mobile applications. One mobile application will be used by employees; the other will be used by customers.

Litware plans to move to a more agile development methodology. Shared code will be extracted into a series of package.

Litware has started an internal cloud transformation process and plans to use cloud based services whenever suitable.

Litware wants to become proactive m detecting failures, rather than always waning for customer bug reports.

Technical Requirements

The company's investment planning applications suite must meet the following technical requirements:

• New incoming connections through the firewall must be minimized.

• Members of a group named Developers must be able to install packages.

• The principle of least privilege must be used for all permission assignments

• A branching strategy that supports developing new functionality in isolation must be used.

• Members of a group named Team leaders must be able to create new packages and edit the permissions of package feeds

• Visual Studio App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• By default, all App Center must be used to centralize the reporting of mobile application crashes and device types in use.

• Code quality and release quality are critical. During release, deployments must not proceed between stages if any active bugs are logged against the release.

• The mobile applications must be able to call the share pricing service of the existing retirement fund management system. Until the system is upgraded, the service will only support basic authentication over HUPS.

• The required operating system configuration tor the test servers changes weekly. Azure Automation State Configuration must be used to ensure that the operating system on each test servers configured the same way when the servers are created and checked periodically.

Current Technical

The test servers are configured correctly when first deployed, but they experience configuration drift over time. Azure Automation State Configuration fails to correct the configurations.

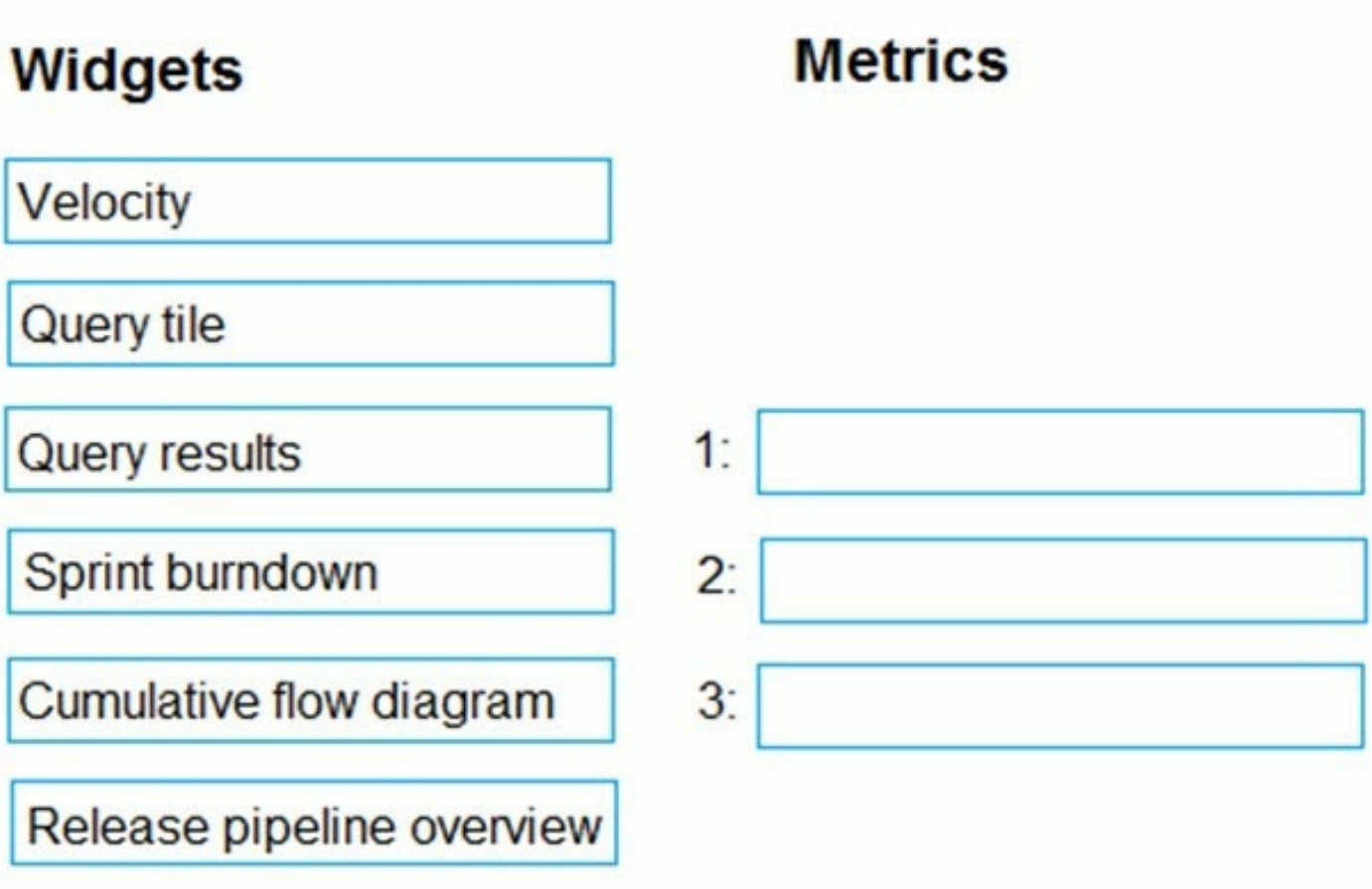
Azure Automation State Configuration nodes are registered by using the following command.

```
Register-AzureRmAutomationDscNode
  -ResourceGroupName 'TestResourceGroup'
  -AutomationAccountName 'LitwareAutomationAccount'
  -AzureVMName $vname
  -ConfigurationMode 'ApplyOnly'
```

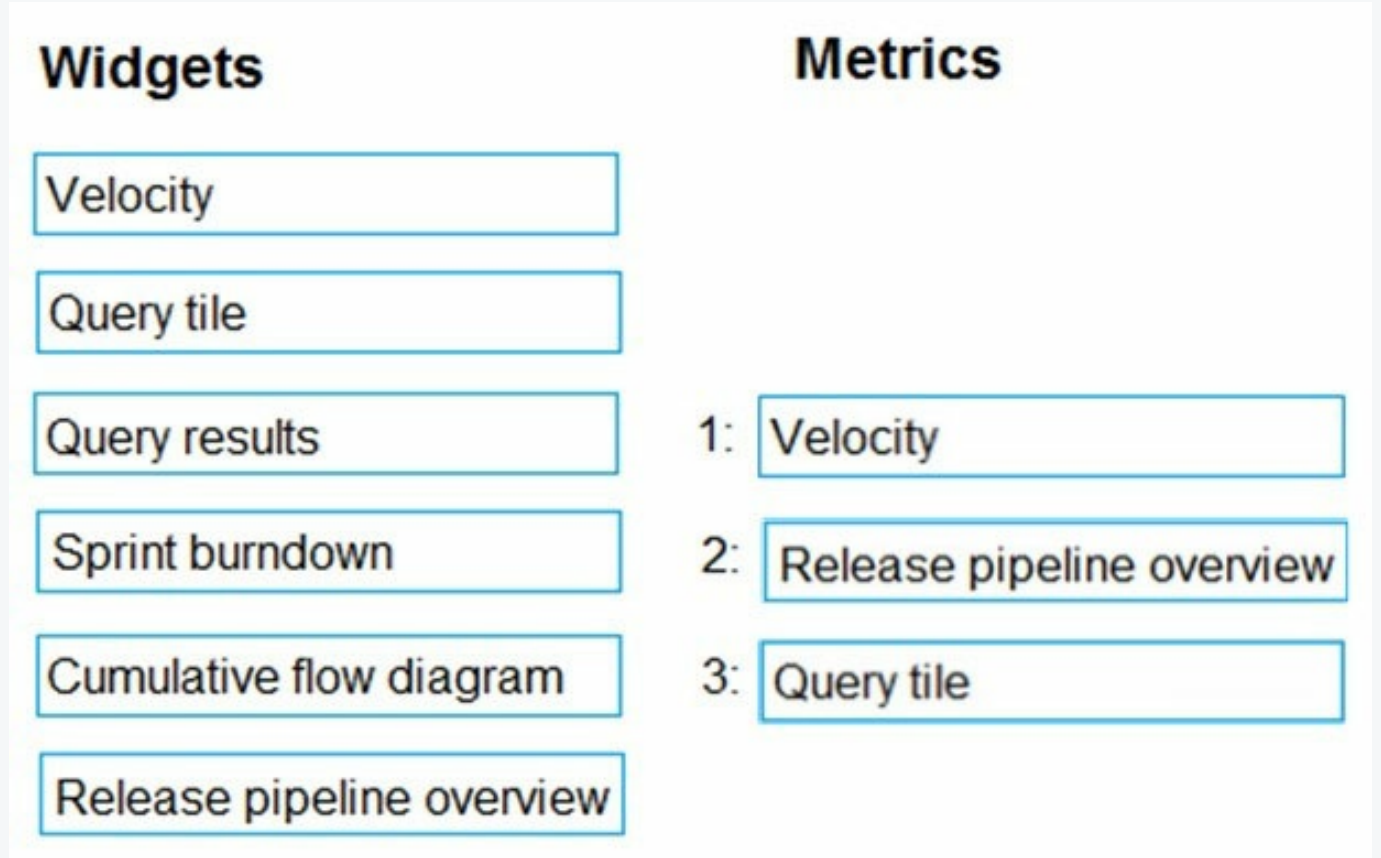
Question: 592

CertyIQ

DRAG DROP -
You are configuring the Azure DevOps dashboard. The solution must meet the technical requirements.
Which widget should you use for each metric? To answer, drag the appropriate widgets to the correct metrics.
Each widget may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Woodgrove Bank identifies the following technical requirements:

The Azure DevOps dashboard must display the metrics shown in the following table:

▪

Number	Required data
1	A comparison between the work the development team planned to deliver and what was delivered
2	The status of the environments in a release definition
3	The total number of results from a work item query

Box 1: Velocity -
Velocity displays your team velocity. It shows what your team delivered as compared to plan.
Box 2: Release pipeline overview
Release pipeline overview shows the status of environments in a release definition.

Box 3: Query tile -
Query tile displays the total number of results from a query.

Reference:
https://docs.microsoft.com/en-us/azure/devops/report/dashboards/widget-catalog?view=azure-devops

Question: 593 CertyIQ

You plan to deploy a new database environment. The solution must meet the technical requirements.
You need to prepare the database for the deployment.
How should you format the export?

A. NDF
B. BACPAC
C. DACPAC
D. MDF

Answer: B

Explanation:

Why the correct option (B–BACPAC) is the appropriate export format

A BACPAC package encapsulates the entire database – schema objects, data, constraints, indexes, and service-level settings – into a single, self-contained zip file. This makes it ideal for moving a complete database environment to a new deployment target, such as Azure SQL Database or a different on-premises instance.
The package can be uploaded to Azure Blob Storage and imported directly into the target service, preserving all database metadata and supporting versioned migrations that meet migration-assessment requirements.
The format is version-agnostic and works across SQL Server and Azure SQL, ensuring compatibility with modern DevOps pipelines and automated deployment scripts.

Why the other options are unsuitable

NDF (OptionA) is a secondary data file used only within a SQL Server database; it cannot be exported or imported as a standalone deployment artifact.
DACPAC (OptionC) contains only the database schema (project model) and deliberately excludes user data, so it cannot fulfill a requirement that calls for moving the entire database state. It is used for building the database on the target, not for transporting existing data.
MDF (OptionD) is the primary data file of a SQL Server database. While it stores the data, it cannot be directly imported into another server without accompanying log files and server-level settings, and it lacks the packaging, versioning, and validation features provided by BACPAC.

Conclusion
Because the deployment scenario requires a portable, complete representation of the database that includes both schema and data, a BACPAC is the only option that satisfies the technical requirements for exporting and subsequently importing the database environment.

References

Export and import a database by using a BACPAC file –Microsoft Docs
DACPAC and BACPAC overview –Microsoft Docs

Question: 594 CertyIQ

You need to meet the technical requirements for monitoring App1.
What should you use?

A. Splunk
B. Azure Application Insights
C. Azure Advisor
D. App Service logs

Answer: B

Explanation:

Reasoning

Azure Application Insights (B) is purpose-built for collecting, analyzing, and visualizing real-time telemetry (requests, exceptions, performance counters, custom metrics, and trace logs) from an AzureApp Service hosting **App1**. It integrates natively with App Service, offers end-to-end correlation, smart detection, diagnostic queries, alerts, and seamless export to Log Analytics or PowerBI — fulfilling the full monitoring stack required for production workloads.
Splunk (A) is a general-purpose log platform that can ingest App Service logs, but it is not a native Azure monitoring service; it adds operational overhead, licensing cost, and does not provide built-in Azure-specific insights or alerting.
Azure Advisor (C) delivers best-practice recommendations for resource configuration and cost optimization; it does not capture runtime telemetry or monitor an application's behavior, making it unsuitable for the required monitoring task.
App Service logs (D) expose basic file-level logs (e.g., HTTP, console output) but lack the rich analytics, real-time dashboards, custom metric collection, and alerting capabilities that are essential for comprehensive monitoring of an application like **App1**.

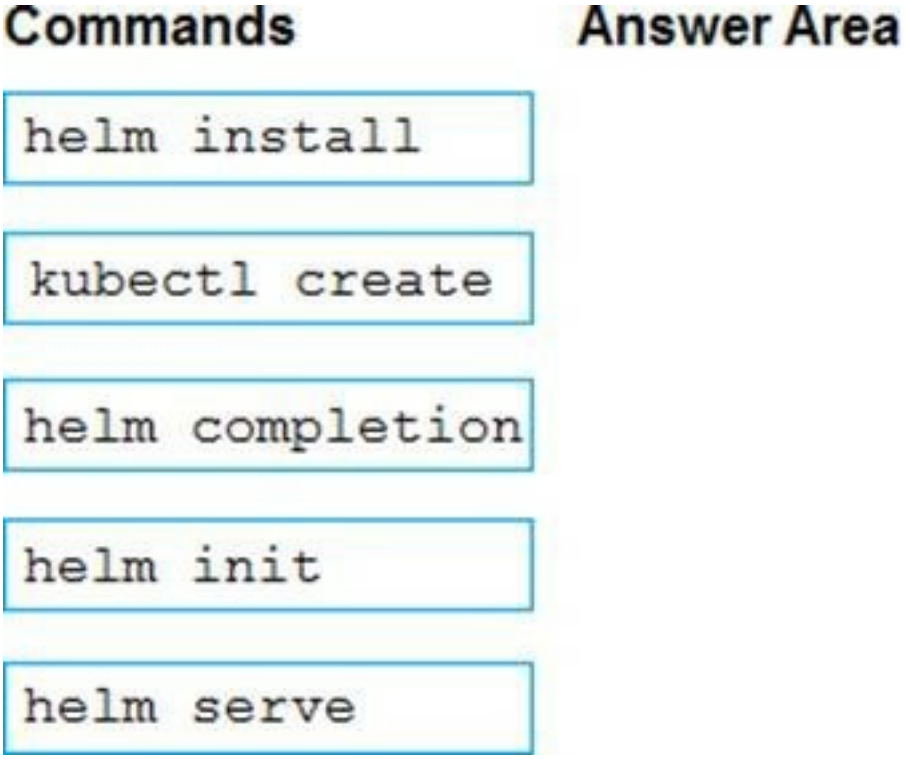
Therefore, **OptionB – Azure Application Insights** is the most appropriate and efficient solution.

References

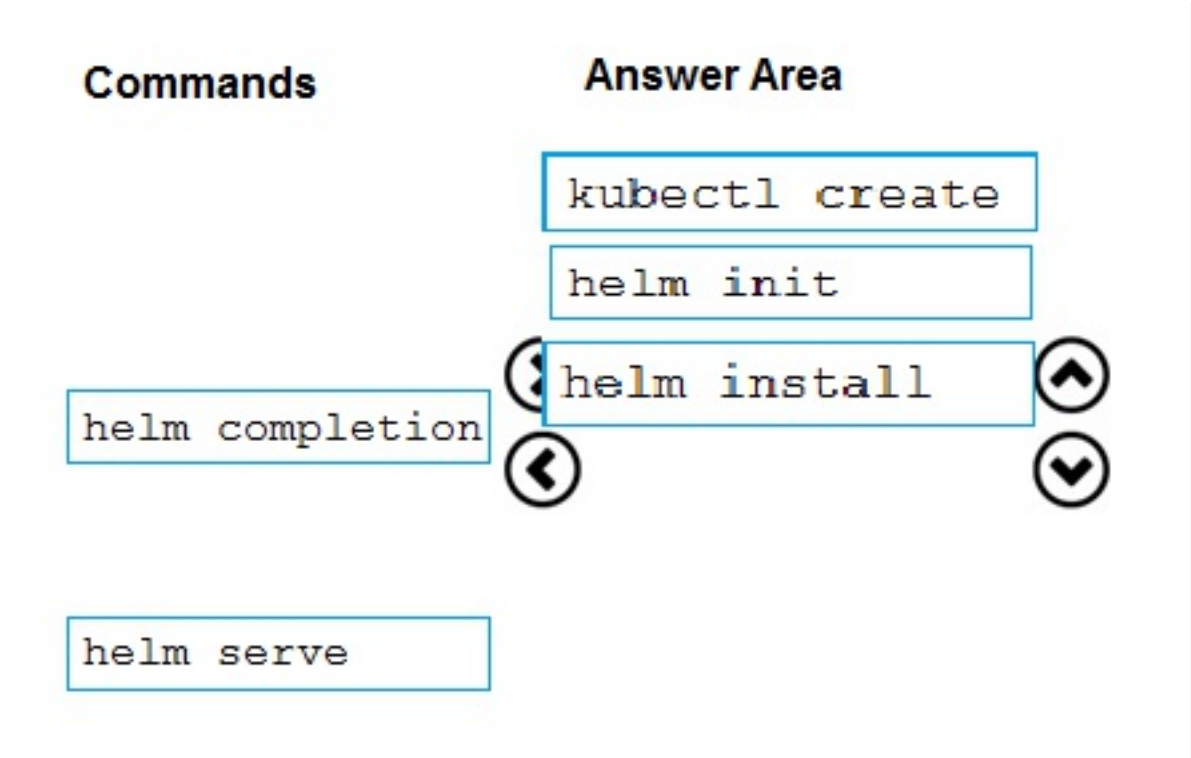
Azure Application Insights – Monitoring Azure App Services
https://learn.microsoft.com/en-us/azure/azure-monitor/app/appinsights-overview
Collecting App Service logs with Application Insights
https://learn.microsoft.com/en-us/azure/app-service/websites-log-streaming

Question: 595 CertyIQ

DRAG DROP -
You need to recommend a solution for deploying charts by using Helm and Tiller to Azure Kubernetes Service (AKS) in an RBAC-enabled cluster.
Which three commands should you recommend be run in sequence? To answer, move the appropriate commands from the list of commands to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Step 1: Kubectl create -

You can add a service account to Tiller using the --service-account <NAME> flag while you're configuring Helm (step 2 below). As a prerequisite, you'll have to create a role binding which specifies a role and a service account name that have been set up in advance.

Example: Service account with cluster-admin role

$ kubectl create -f rbac-config.yaml

serviceaccount "tiller" created

clusterrolebinding "tiller" created

$ helm init --service-account tiller

Step 2: helm init -

To deploy a basic Tiller into an AKS cluster, use the helm init command.

Step 3: helm install -

To install charts with Helm, use the helm install command and specify the name of the chart to install.

References:

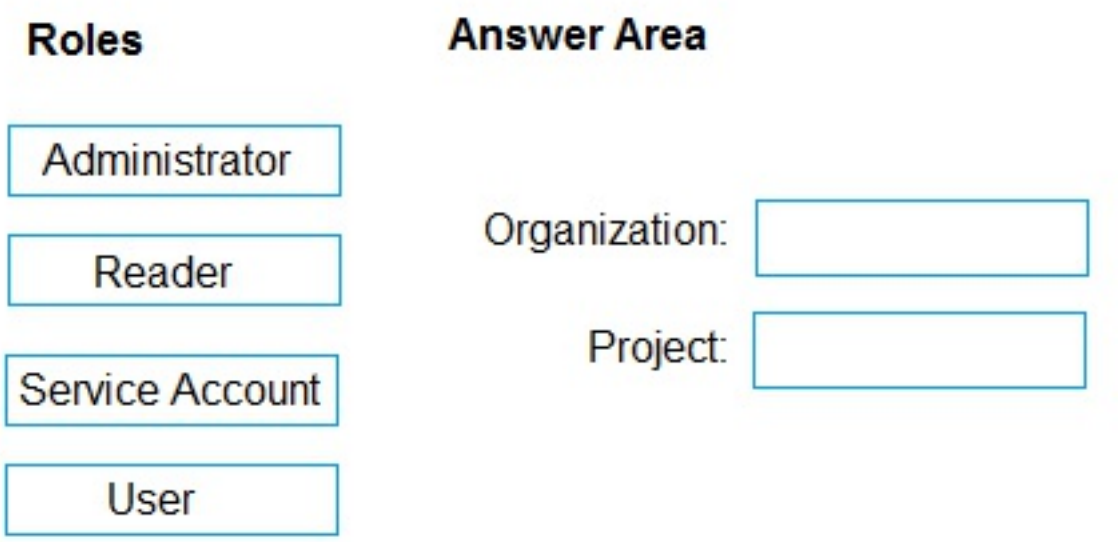
https://docs.microsoft.com/en-us/azure/aks/kubernetes-helm

https://docs.helm.sh/using_helm/#tiller-namespaces-and-rbac

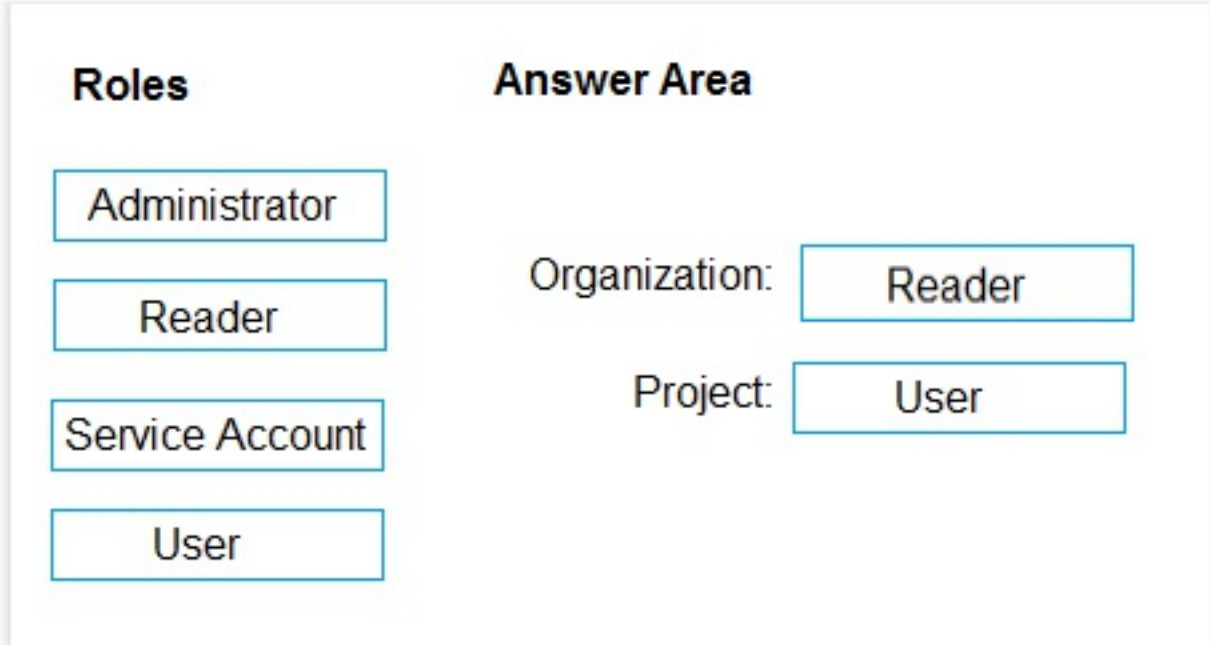
Question: 596 CertyIQ

DRAG DROP -
You need to configure access to Azure DevOps agent pools to meet the following requirements:
- Use a project agent pool when authoring build or release pipelines.

✑ View the agent pool and agents of the organization.
✑ Use the principle of least privilege.
Which role memberships are required for the Azure DevOps organization and the project? To answer, drag the appropriate role memberships to the correct targets. Each role membership may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Reader -

Members of the Reader role can view the organization agent pool as well as agents. You typically use this to add operators that are responsible for monitoring the agents and their health.

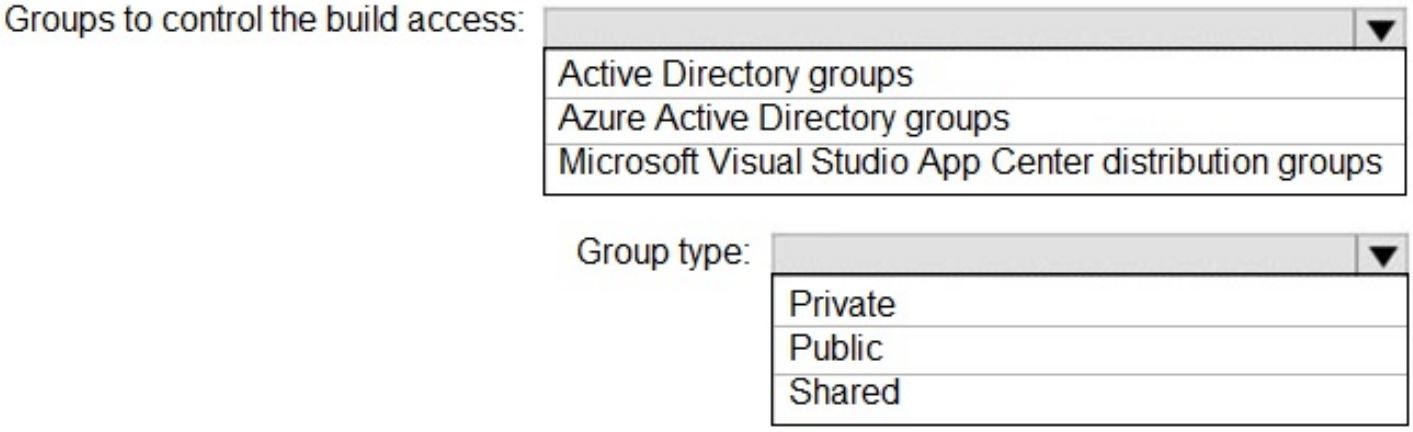
Box 2: User

Service Account is a role on organization level for creating a project agent pool in a project. User role is a role on project level for using the project agent pool when authoring pipeline.

Question: 597 CertyIQ

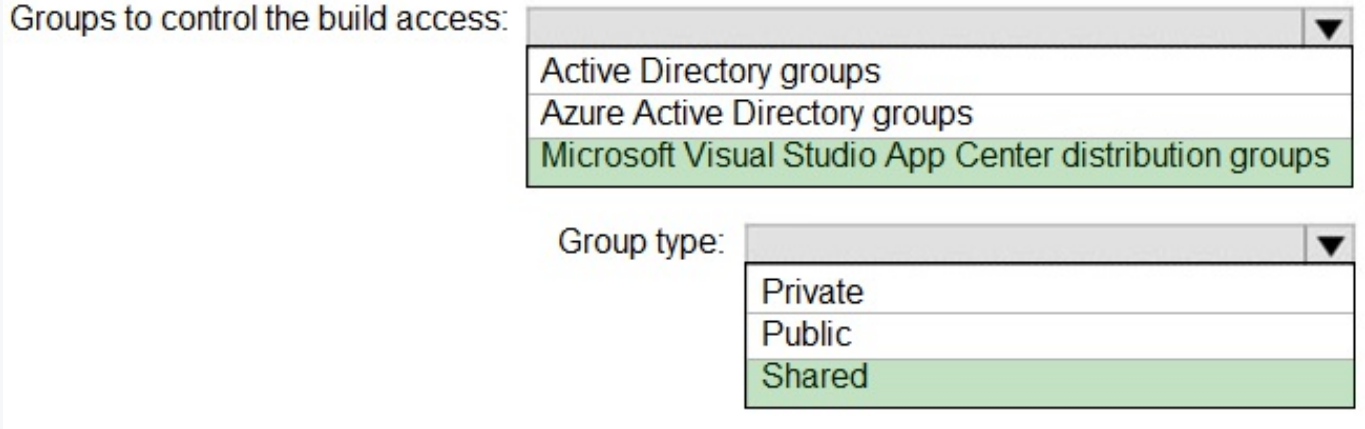
HOTSPOT -
Your company is creating a suite of three mobile applications.
You need to control access to the application builds. The solution must be managed at the organization level.
What should you use? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:

Box 1: Microsoft Visual Studio App Center distribution Groups

Distribution Groups are used to control access to releases. A Distribution Group represents a set of users that can be managed jointly and can have common access to releases. Example of Distribution Groups can be teams of users, like the QA Team or External Beta Testers or can represent stages or rings of releases, such as Staging.

Box 2: Shared -

Shared distribution groups are private or public distribution groups that are shared across multiple apps in a single organization. Shared distribution groups eliminate the need to replicate distribution groups across multiple apps.

Note: With the Deploy with App Center Task in Visual Studio Team Services, you can deploy your apps from Azure DevOps (formerly known as VSTS) to App

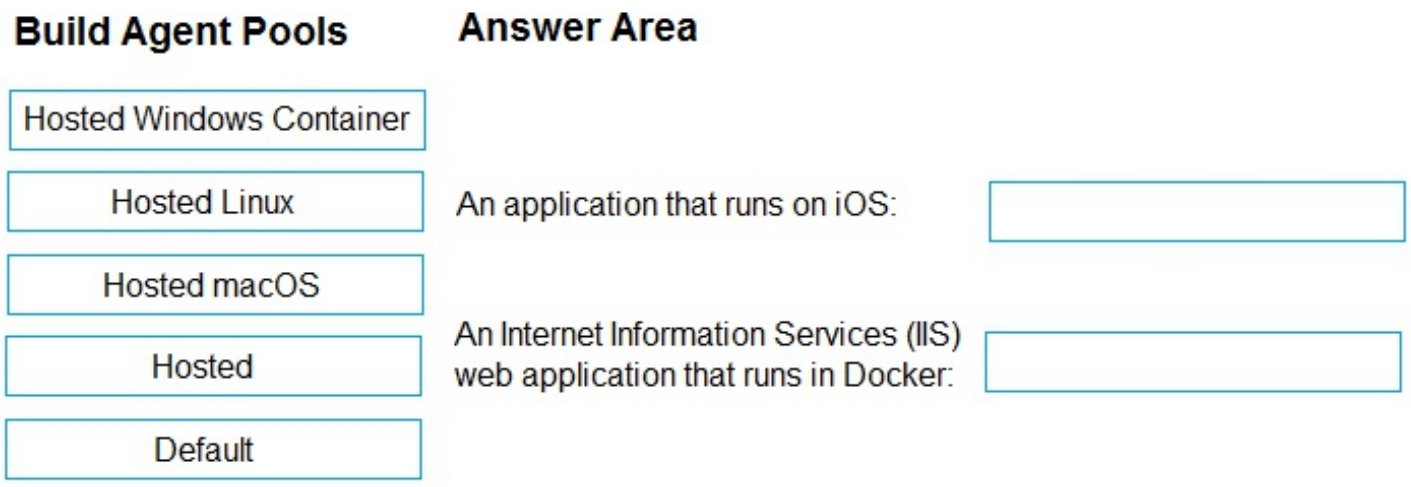
Center. By deploying to App Center, you will be able to distribute your builds to your users.

References:

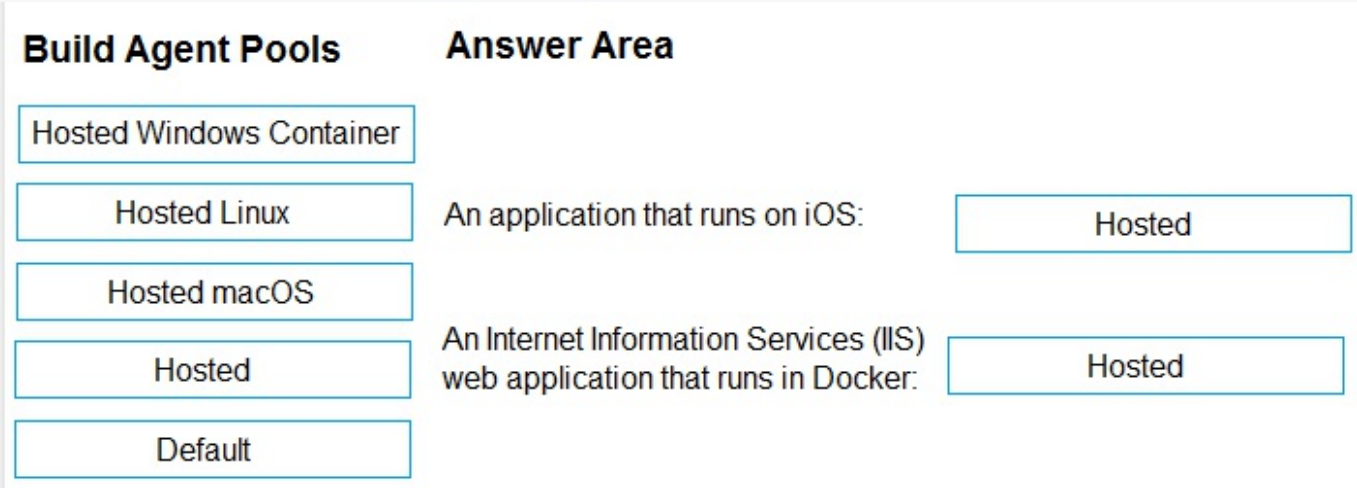
https://docs.microsoft.com/en-us/appcenter/distribution/groups

Question: 598 CertyIQ

DRAG DROP -
You are configuring Azure DevOps build pipelines.
You plan to use hosted build agents.
Which build agent pool should you use to compile each application type? To answer, drag the appropriate build agent pools to the correct application types. Each build agent pool may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Hosted

Box 2: Hosted -

Hosted pool (Azure Pipelines only): The Hosted pool is the built-in pool that is a collection of Microsoft-hosted agents.

Incorrect Answers:

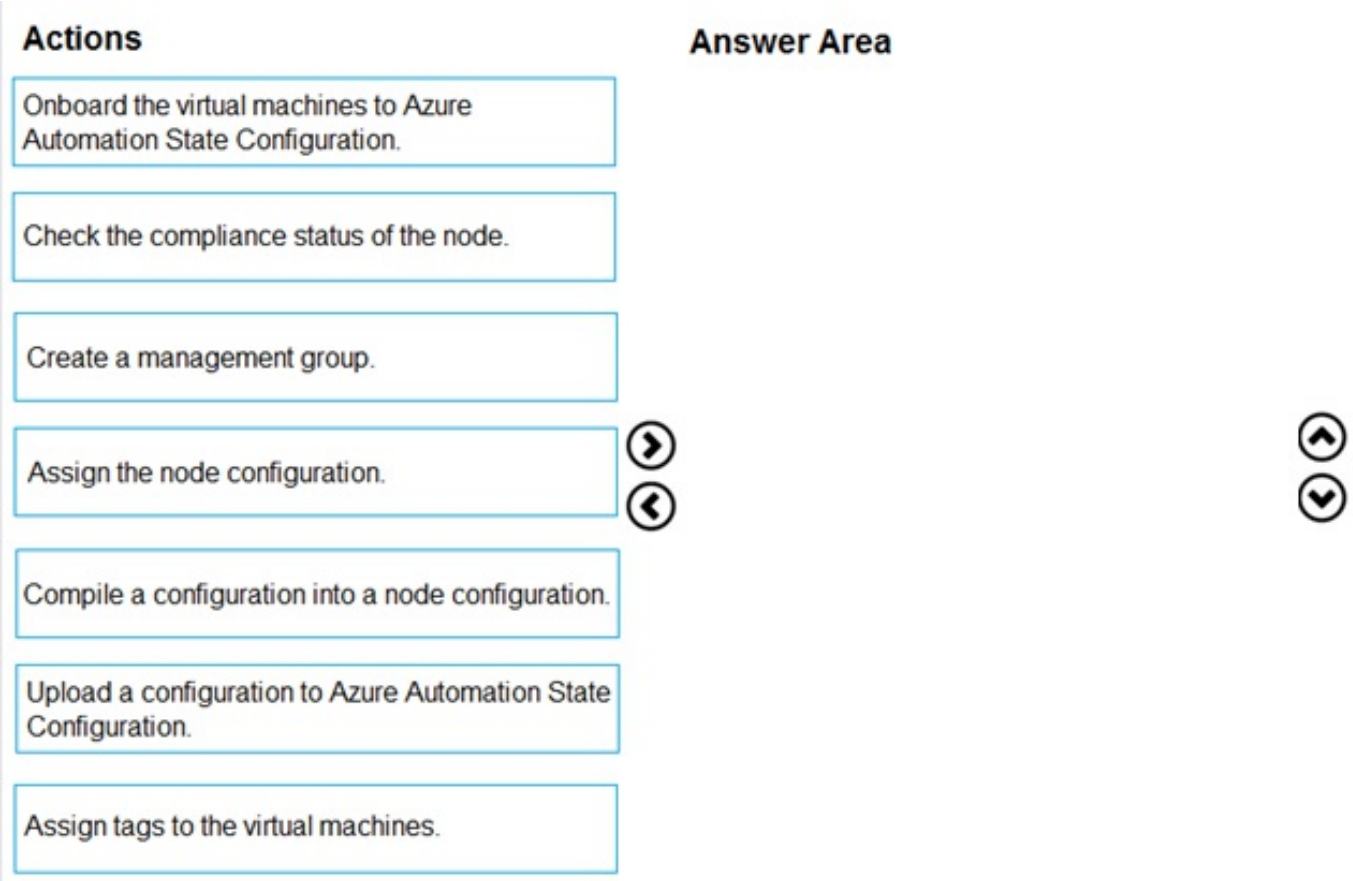
Default pool: Use it to register self-hosted agents that you've set up.

Hosted Windows Container pool (Azure Pipelines only): Enabled you to build and release inside Windows containers. Unless you're building using containers,

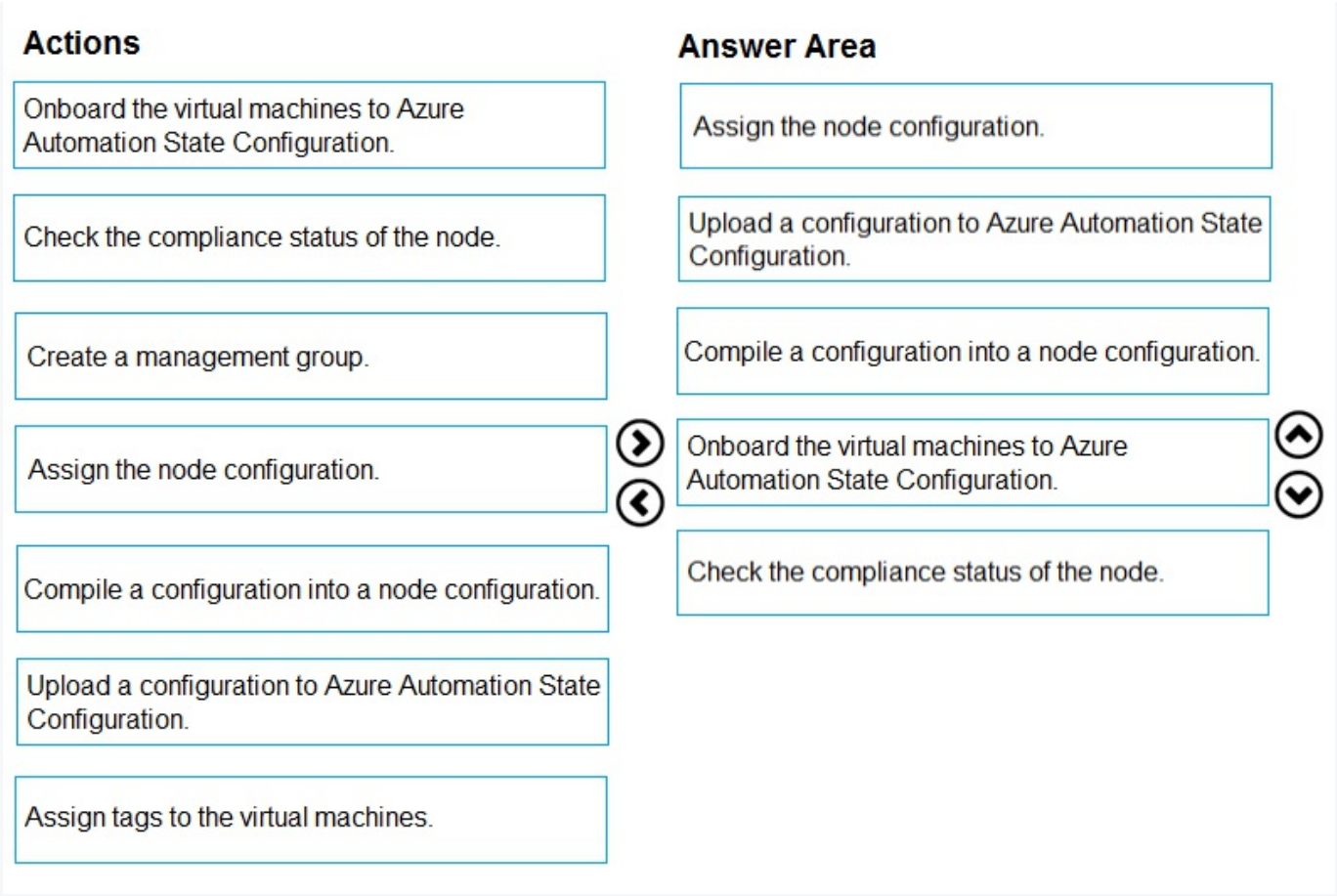
Windows builds should run in the Hosted VS2017 or Hosted pools.

Question: 599 CertyIQ

DRAG DROP -
You need to use Azure Automation State Configuration to manage the ongoing consistency of virtual machine configurations.
Which five actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.
Select and Place:



Answer:



Explanation:

Step 1: Assign the node configuration.

You create a simple DSC configuration that ensures either the presence or absence of the Web-Server Windows Feature (IIS), depending on how you assign nodes.

Step 2: Upload a configuration to Azure Automation State Configuration.

You import the configuration into the Automation account.

Step 3: Compiling a configuration into a node configuration

Compiling a configuration in Azure Automation

Before you can apply a desired state to a node, a DSC configuration defining that state must be compiled into one or more node configurations (MOF document), and placed on the Automation DSC Pull Server.

Step 4: Onboard the virtual machines to Azure State Configuration

Onboarding an Azure VM for management with Azure Automation State Configuration

Step 5: Check the compliance status of the node.

Viewing reports for managed nodes. Each time Azure Automation State Configuration performs a consistency check on a managed node, the node sends a status report back to the pull server. You can view these reports on the page for that node.

On the blade for an individual report, you can see the following status information for the corresponding consistency check:

The report status "" whether the node is "Compliant", the configuration "Failed", or the node is "Not Compliant" (when the node is in ApplyandMonitor mode and the machine is not in the desired state).

References:

https://docs.microsoft.com/en-us/azure/automation/automation-dsc-getting-started

Question: 600 CertyIQ

HOTSPOT -
You have a project in Azure DevOps.
You plan to create a build pipeline that will deploy resources by using Azure Resource Manager templates. The templates will reference secrets stored in Azure
Key Vault.
You need to ensure that you can dynamically generate the resource ID of the key vault during template deployment.
What should you include in the template? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

```
"resources":[
{
  "apiversion": "2018-05-01",
  "name" : "secrets",
  "type":
```

- "Microsoft.KeyVault/vaults",
- "Microsoft.Resources/deployment".
- "Microsoft.Subscription/subscriptions".

```
  "properties":{
     "mode" : "Incremental",
```

:{

- "deployment"
- "template"
- "templateLink"

```
contentVersion" : "1.0.0.0",
          "uri" : "[uri(parameters('_artifactsLocation'),
          concat('./nested/sqlserver.json',
          parameters('_artifactsLocationSasToken')))]"
      },
      "parameters":{
          "secret":{
             "reference":{
                "keyVault":{
                   "id": "[resourceId(parameters(
                   'vaultSubscription'),parameters(
                   'vaultResourceGroupName'),
                   'Microsoft.KeyVault/vaults',
                   parameters('vaultName'))]"
                },
                "secretName": "[parameters('secretName')]"
                 }
                }
               }
              }
             }
          ],
```

Answer:

Answer Area



Explanation:

Box 1: "Microsoft.Resources/deployments"

Reference a secret with dynamic ID. You need to reference a key vault secret that varies based on the current deployment.

Example:

"resources": [

"apiVersion": "2018-05-01",

"name": "dynamicSecret",

"type": "Microsoft.Resources/deployments",

"properties":

"mode": "Incremental",

"templateLink":

Box 2: "templateLink"

In your parent template, you add the linked template and pass in a parameter that contains the dynamically generated resource ID.

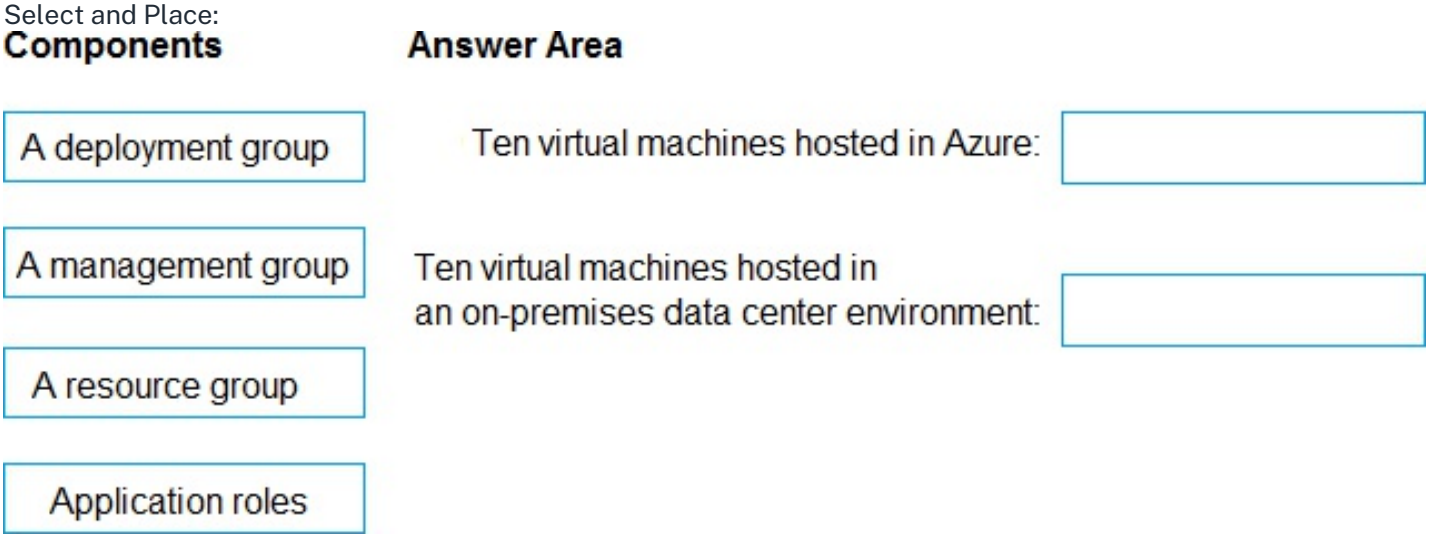
References:

https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-keyvault-parameter

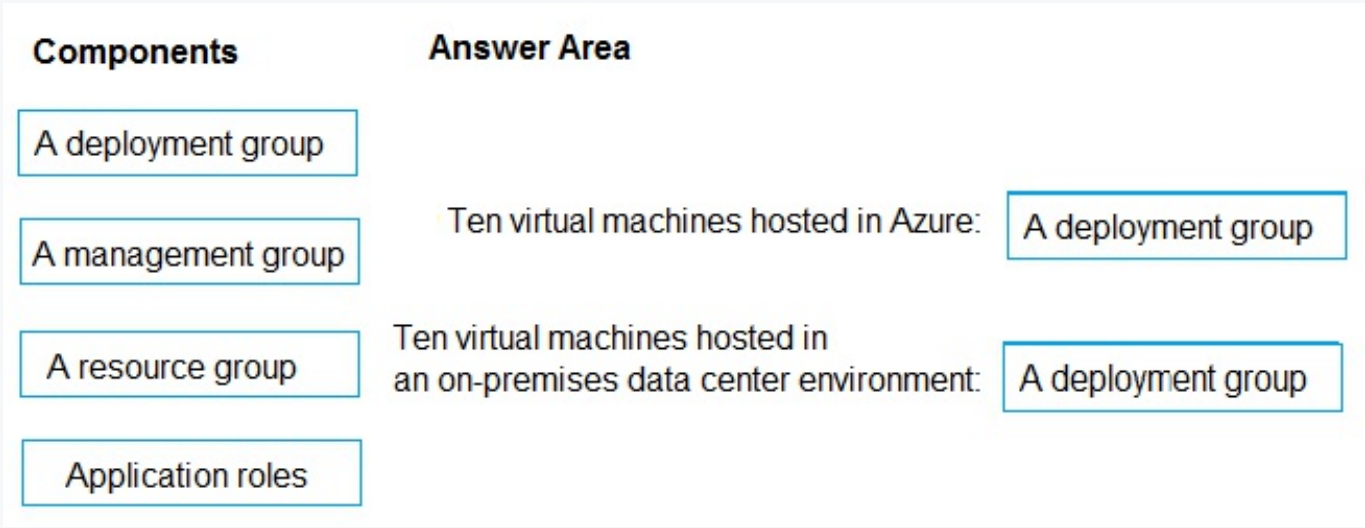
Question: 601 CertyIQ

DRAG DROP -
Your company plans to deploy an application to the following endpoints:
- Ten virtual machines hosted in Azure
- Ten virtual machines hosted in an on-premises data center environment

All the virtual machines have the Azure Pipelines agent.
You need to implement a release strategy for deploying the application to the endpoints.
What should you recommend using to deploy the application to the endpoints? To answer, drag the appropriate components to the correct endpoints. Each component may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: A deployment group -

When authoring an Azure Pipelines or TFS Release pipeline, you can specify the deployment targets for a job using a deployment group.

If the target machines are Azure VMs, you can quickly and easily prepare them by installing the Azure

Pipelines Agent Azure VM extension on each of the VMs, or by using the Azure Resource Group Deployment task in your release pipeline to create a deployment group dynamically.

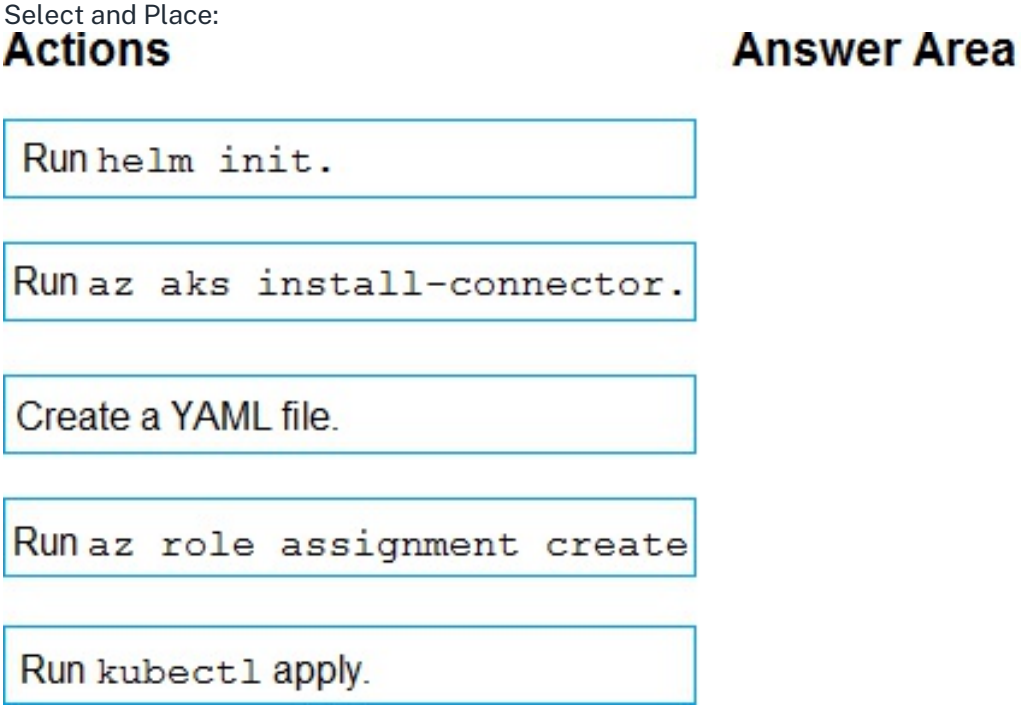
Box 2: A deployment group -

References:

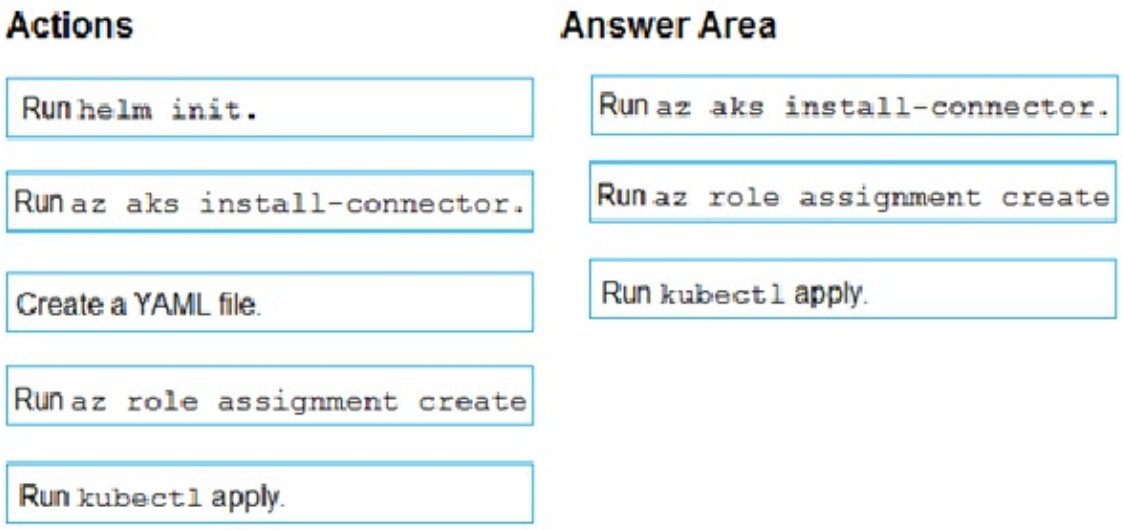
https://docs.microsoft.com/en-us/azure/devops/pipelines/release/deployment-groups

Question: 602 CertyIQ

DRAG DROP -
You have an Azure Kubernetes Service (AKS) implementation that is RBAC-enabled.
You plan to use Azure Container Instances as a hosted development environment to run containers in the AKS implementation.
You need to configure Azure Container Instances as a hosted environment for running the containers in AKS.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

The goal is to integrate **Azure Container Instances (ACI)** into an **RBAC-enabled Azure Kubernetes Service (AKS)** cluster. This requires installing the connector, granting permissions, and then deploying the workload.

Correct Sequence:

1. **Run az aks install-connector.**
2. **Run az role assignment create.**
3. **Run kubectl apply.**

Explanation of the Sequence

1. Run az aks install-connector: This command installs the **Virtual Kubelet** into the AKS cluster, which acts as the crucial link between AKS and ACI.

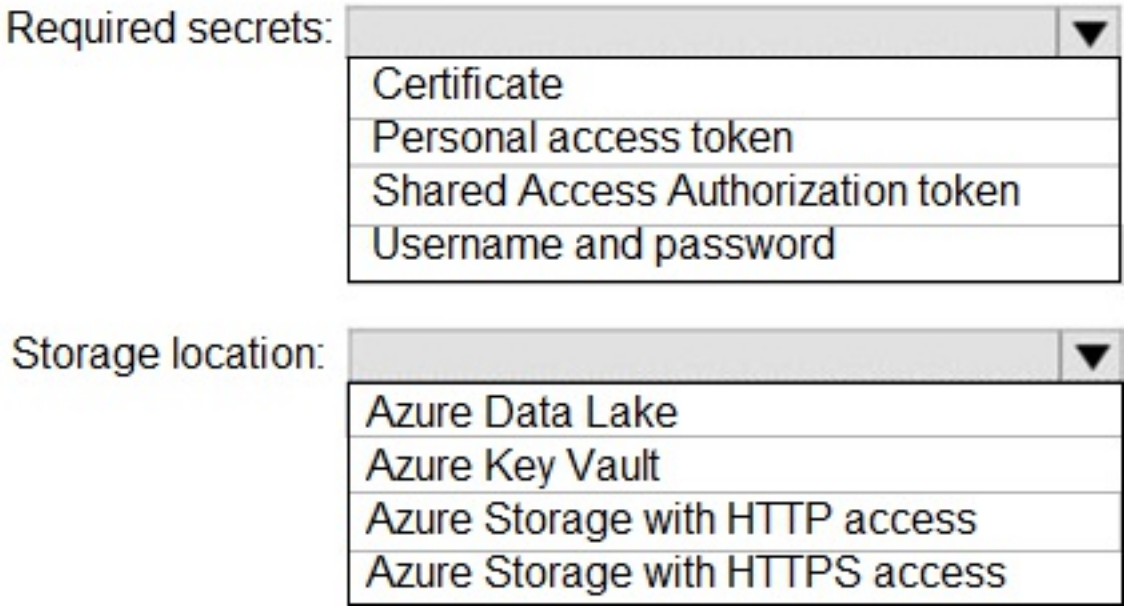
2. Run az role assignment create: Because the AKS cluster is **RBAC-enabled**, the Virtual Kubelet needs explicit permissions (an Azure Role Assignment) to create and manage the ACI resources in your resource group.

3. Run kubectl apply: Once the connector is installed and has the correct permissions, you use this command to deploy your container's YAML file. The YAML file will contain a nodeSelector that targets the ACI-backed "Virtual Node," instructing AKS to run the container as an ACI instance.

Question: 603 CertyIQ

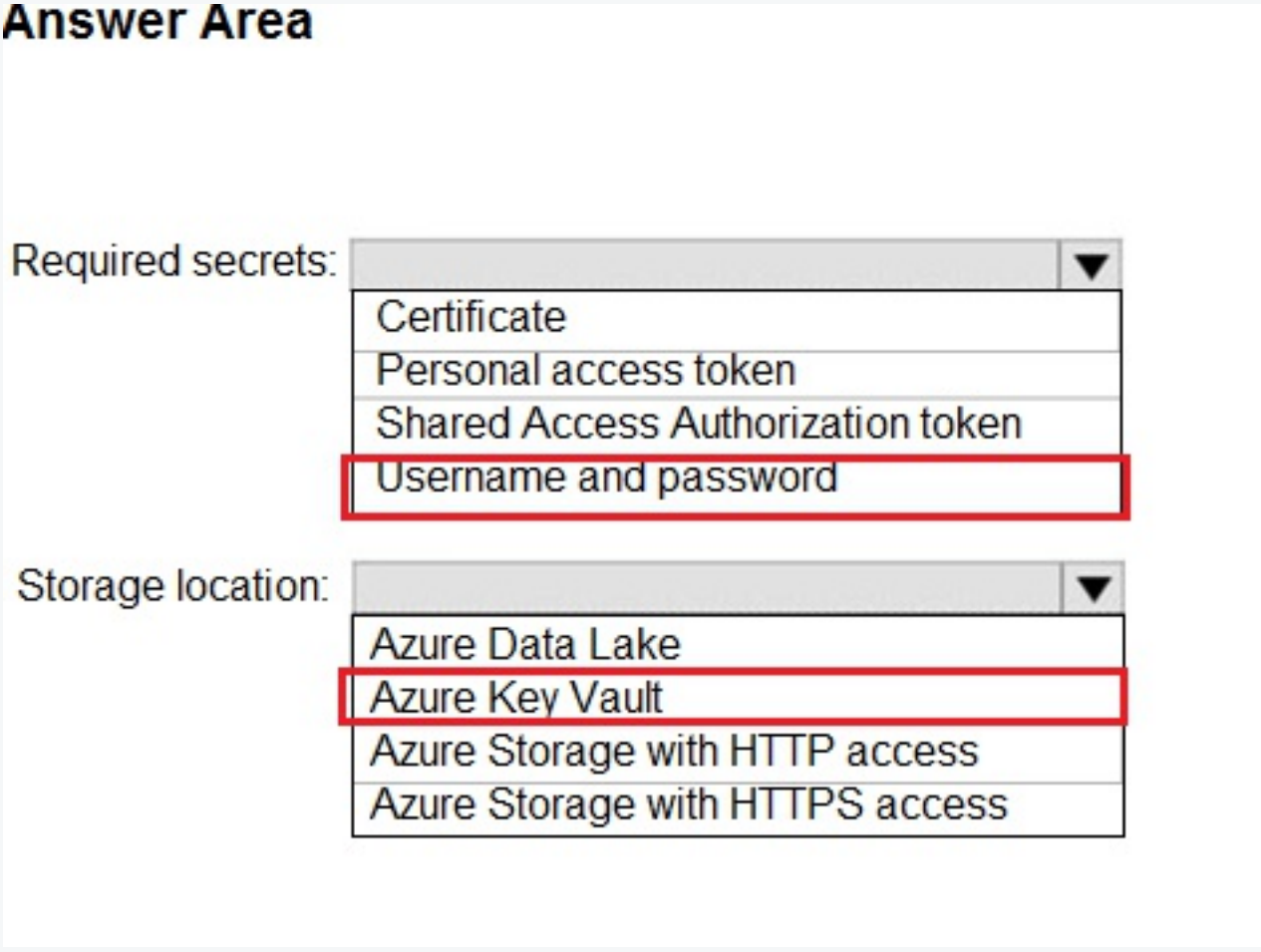
HOTSPOT -
You need to configure a cloud service to store the secrets required by the mobile applications to call the share pricing service.
What should you include in the solution? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:

Required Secrets → **Username and Password.**

The source system (e.g., SQL database, FTP server, on-prem system) requires authentication using a

username and password.

Storage Location → **Azure Key Vault.**

Azure Key Vault is the enterprise-grade, centralized, secure secret-storage solution for Azure.

Question: 604 CertyIQ

HOTSPOT -
How should you configure the release retention policy for the investment planning applications suite? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

Global release:

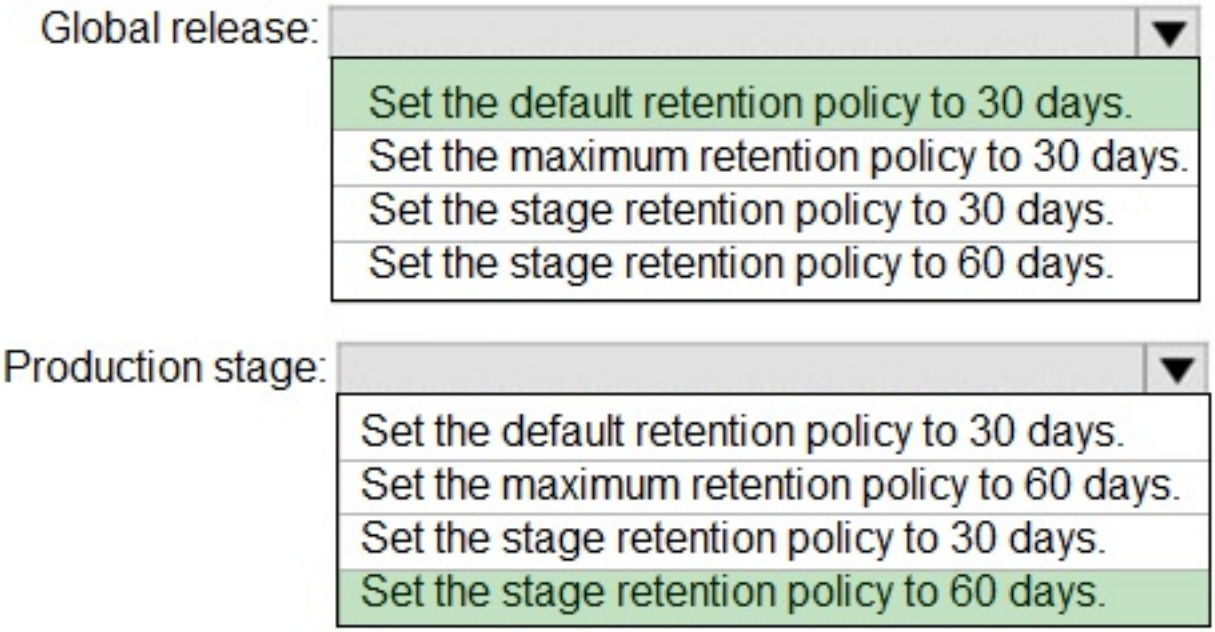
- Set the default retention policy to 30 days.
- Set the maximum retention policy to 30 days.
- Set the stage retention policy to 30 days.
- Set the stage retention policy to 60 days.

Production stage:

- Set the default retention policy to 30 days.
- Set the maximum retention policy to 60 days.
- Set the stage retention policy to 30 days.
- Set the stage retention policy to 60 days.

Answer:

Answer Area



Explanation:

Scenario: By default, all releases must remain available for 30 days, except for production releases, which must be kept for 60 days.

Box 1: Set the default retention policy to 30 days

The Global default retention policy sets the default retention values for all the build pipelines. Authors of build pipelines can override these values.

Box 2: Set the stage retention policy to 60 days

You may want to retain more releases that have been deployed to specific stages.

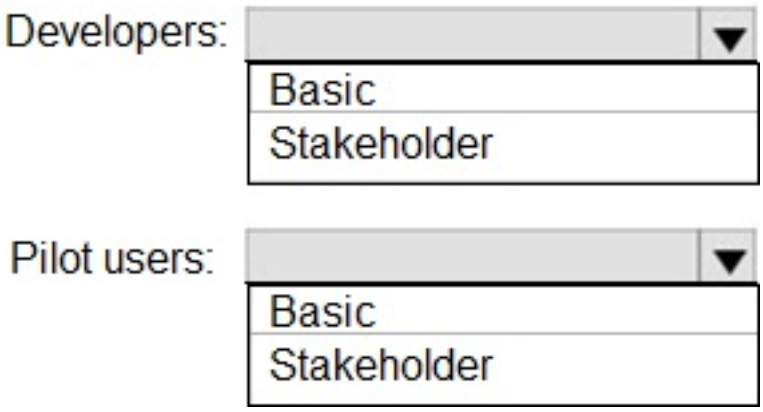
References:

https://docs.microsoft.com/en-us/azure/devops/pipelines/policies/retention

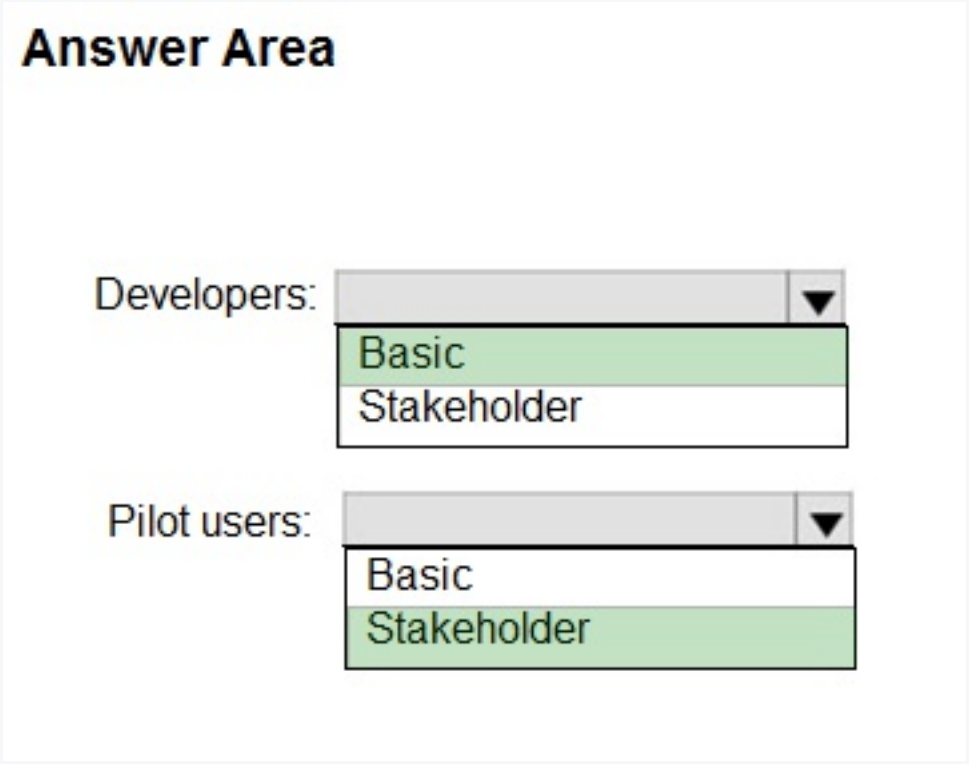
Question: 605 CertyIQ

HOTSPOT -
Your company is building a new web application.
You plan to collect feedback from pilot users on the features being delivered.
All the pilot users have a corporate computer that has Google Chrome and the Microsoft Test & Feedback extension installed. The pilot users will test the application by using Chrome.
You need to identify which access levels are required to ensure that developers can request and gather feedback from the pilot users. The solution must use the principle of least privilege.
Which access levels in Azure DevOps should you identify? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:



Explanation:

Box 1: Basic -

Assign Basic to users with a TFS CAL, with a Visual Studio Professional subscription, and to users for whom you are paying for Azure Boards & Repos in an organization.

Box 2: Stakeholder -

Assign Stakeholders to users with no license or subscriptions who need access to a limited set of features.

Note:

You assign users or groups of users to one of the following access levels:

Basic: provides access to most features

VS Enterprise: provides access to premium features

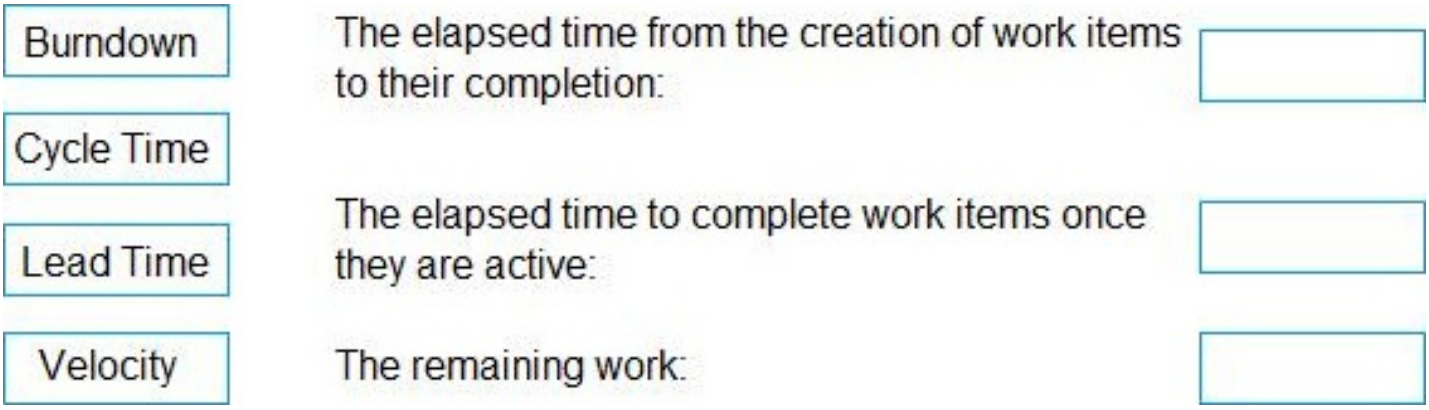
Stakeholders: provides partial access, can be assigned to unlimited users for free

References:

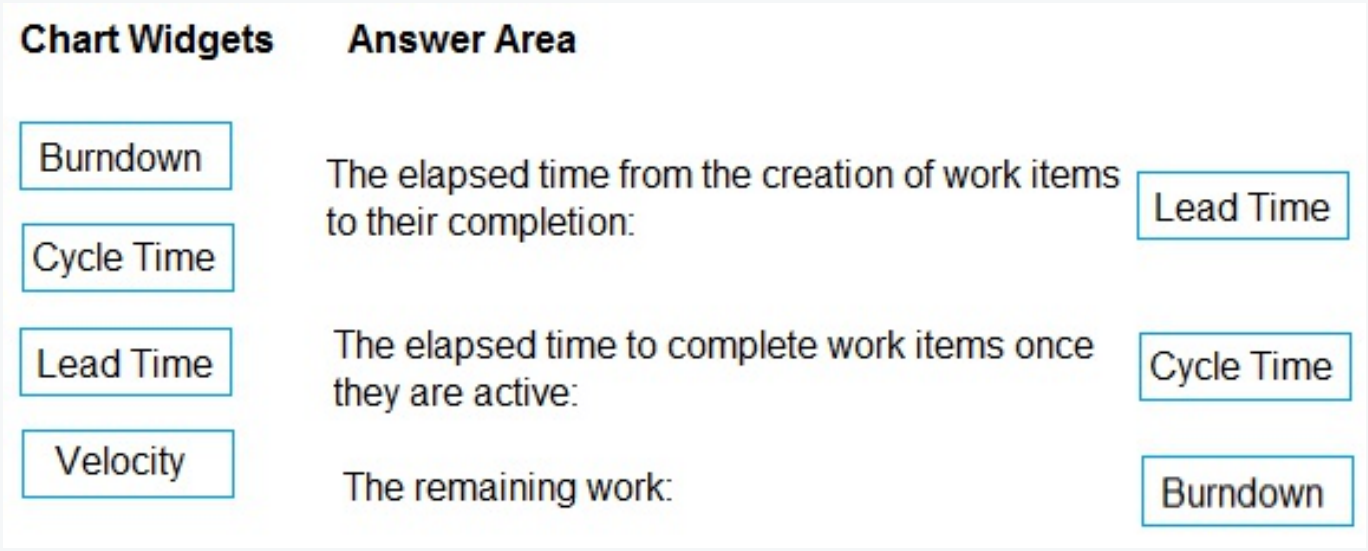
https://docs.microsoft.com/en-us/azure/devops/organizations/security/access-levels?view=vsts

Question: 606 CertyIQ

DRAG DROP -
You need to recommend project metrics for dashboards in Azure DevOps.
Which chart widgets should you recommend for each metric? To answer, drag the appropriate chart widgets to the correct metrics. Each chart widget may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Lead time -

Lead time measures the total time elapsed from the creation of work items to their completion.

Box 2: Cycle time -

Cycle time measures the time it takes for your team to complete work items once they begin actively working on them.

Box 3: Burndown -

Burndown charts focus on remaining work within a specific time period.

Incorrect Answers:

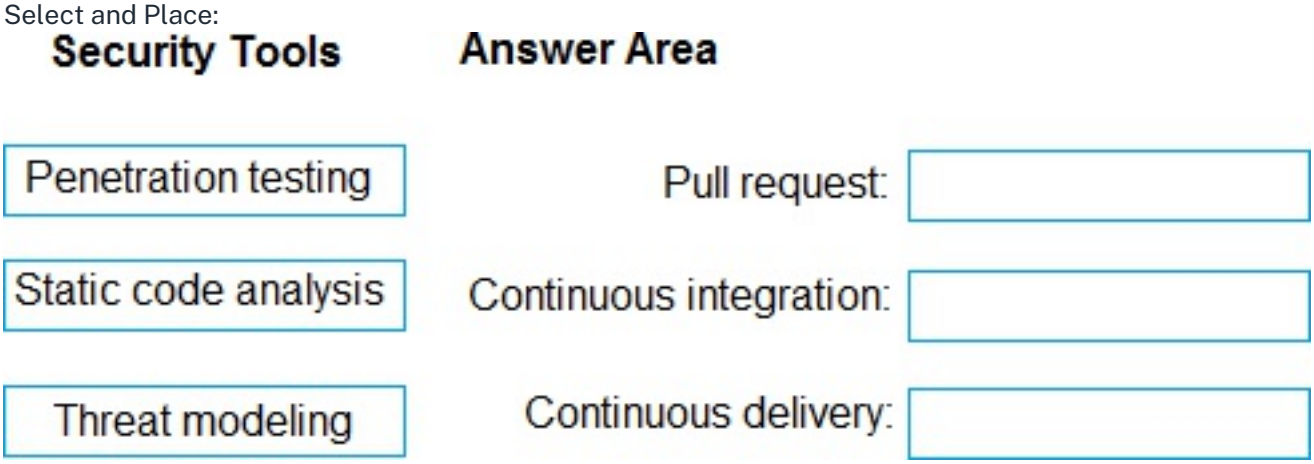
- Velocity provides a useful metric for these activities:
- Support sprint planning
- Forecast future sprints and the backlog items that can be completed
- A guide for determining how well the team estimates and meets their planned commitments

References:

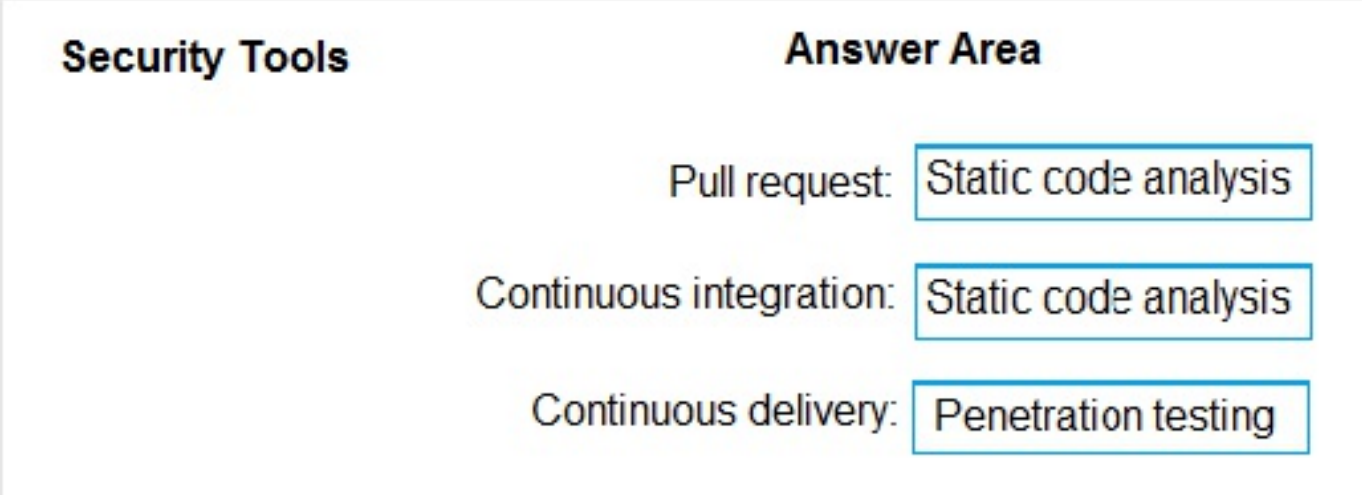
https://docs.microsoft.com/en-us/azure/devops/report/dashboards/velocity-guidance?view=vsts
https://docs.microsoft.com/en-us/azure/devops/report/dashboards/cycle-time-and-lead-time?view=vsts
https://docs.microsoft.com/en-us/azure/devops/report/dashboards/configure-burndown-burnup-widgets?view=vsts

Question: 607 CertyIQ

DRAG DROP -
You need to increase the security of your team's development process.
Which type of security tool should you recommend for each stage of the development process? To answer, drag the appropriate security tools to the correct targets. Each security tool may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: static code analysis

Box 2: Static code analysis -

Validation in the CI/CD begins before the developer commits his or her code. Static code analysis tools in the IDE provide the first line of defense to help ensure that security vulnerabilities are not introduced into the CI/CD process.

Box 3: Penetration testing -

Once your code quality is verified, and the application is deployed to a lower environment like development or QA, the process should verify that there are not any security vulnerabilities in the running application. This can be accomplished by executing automated penetration test against the running application to scan it for vulnerabilities.

References:

https://docs.microsoft.com/en-us/azure/devops/articles/security-validation-cicd-pipeline?view=vsts

Question: 608 CertyIQ

HOTSPOT -
Your company uses Team Foundation Server 2013 (TFS 2013).
You plan to migrate to Azure DevOps.
You need to recommend a migration strategy that meets the following requirements:
- Preserves the dates of Team Foundation Version Control changesets
- Preserves the changes dates of work items revisions
- Minimizes migration effort
- Migrates all TFS artifacts

What should you recommend? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

On the TFS server:

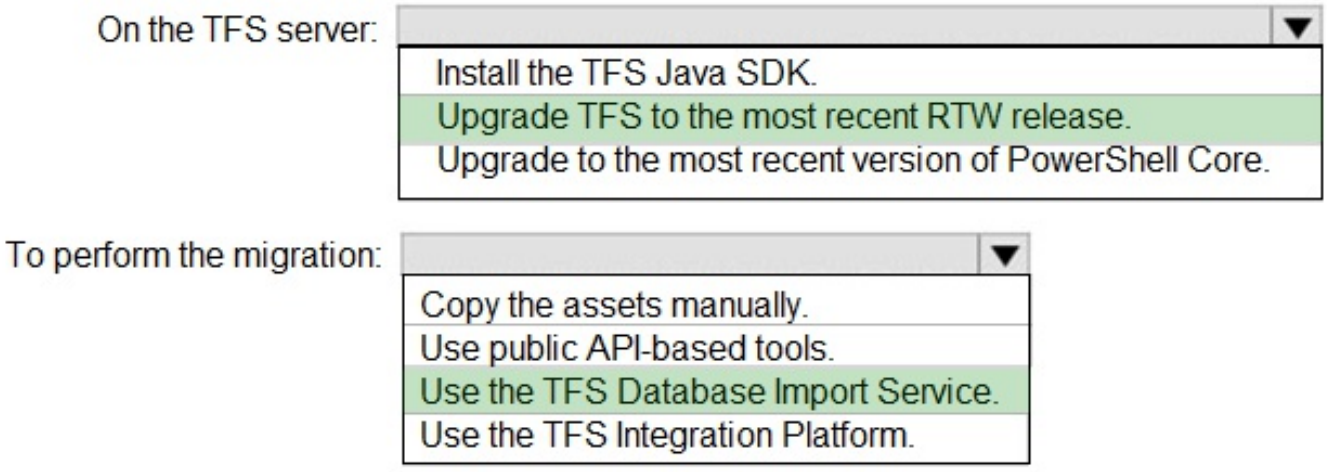
- Install the TFS Java SDK.
- Upgrade TFS to the most recent RTW release.
- Upgrade to the most recent version of PowerShell Core.

To perform the migration:

- Copy the assets manually.
- Use public API-based tools.
- Use the TFS Database Import Service.
- Use the TFS Integration Platform.

Answer:

Answer Area



Explanation:

Box 1: Upgrade TFS to the most recent RTM release.

One of the major prerequisites for migrating your Team Foundation Server database is to get your database schema version as close as possible to what is currently deployed in Azure Devops Services.

Box 2: Use the TFS Database Import Service

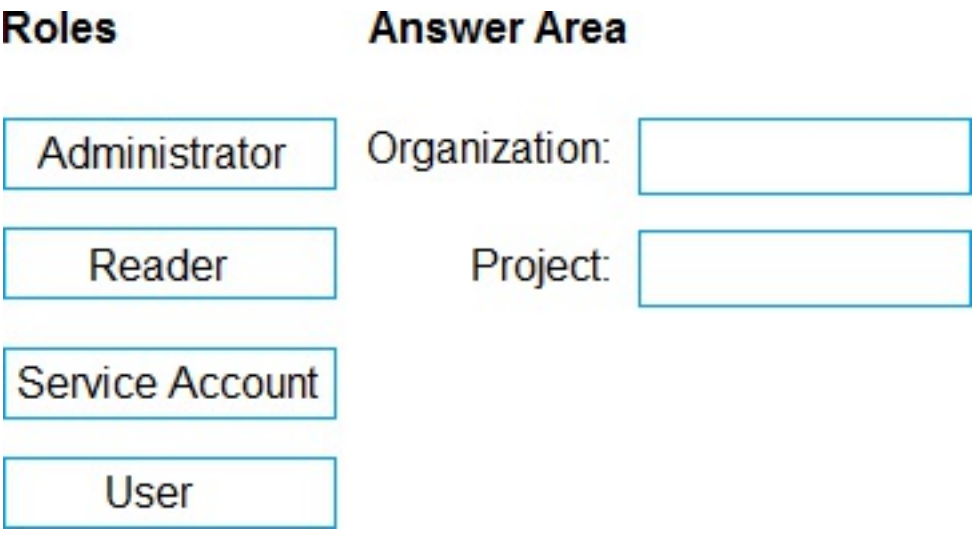
In Phase 3 of your migration project, you will work on upgrading your Team Foundation Server to one of the supported versions for the Database Import Service in

Azure Devops Services.

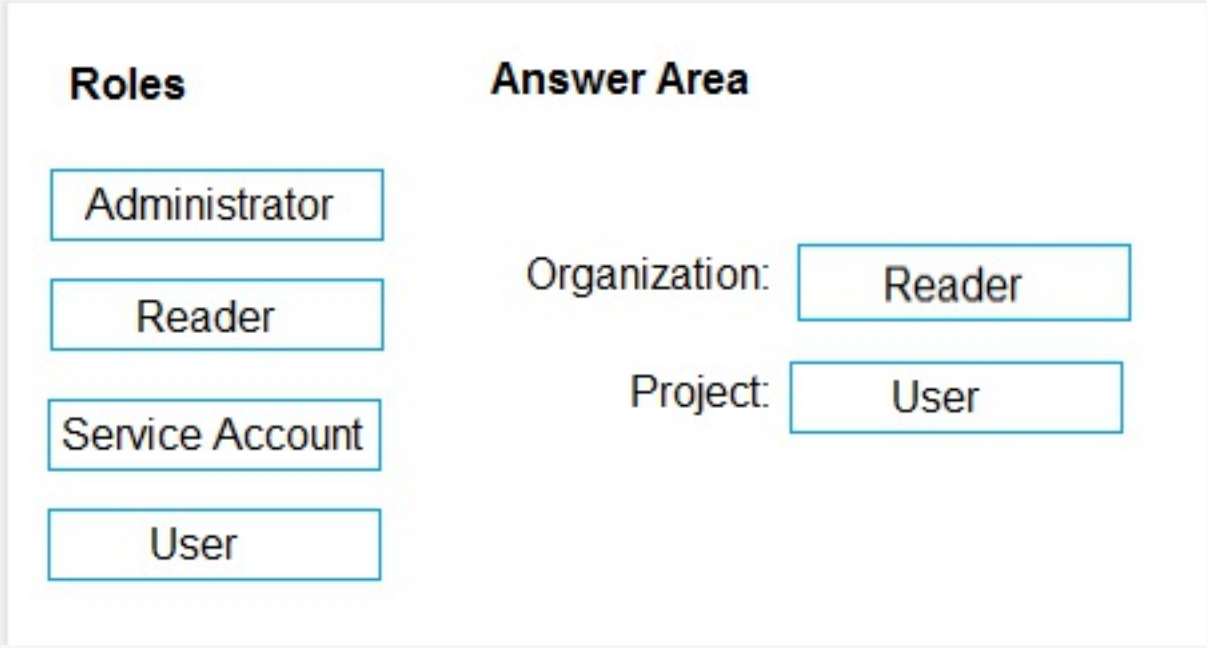
Question: 609 CertyIQ

DRAG DROP -
You need to configure access to Azure DevOps agent pools to meet the following requirements:
- Use a project agent pool when authoring build or release pipelines.
- View the agent pool and agents of the organization.
- Use the principle of least privilege.

Which role memberships are required for the Azure DevOps organization and the project? To answer, drag the appropriate role memberships to the correct targets. Each role membership may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Reader -

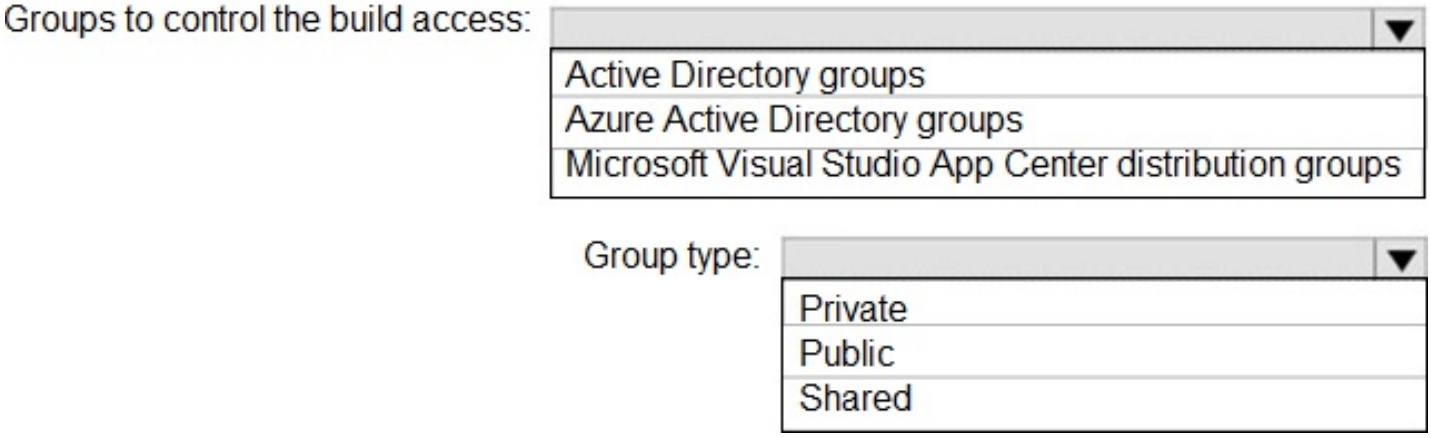
Members of the Reader role can view the organization agent pool as well as agents. You typically use this to add operators that are responsible for monitoring the agents and their health.

Box 2: user

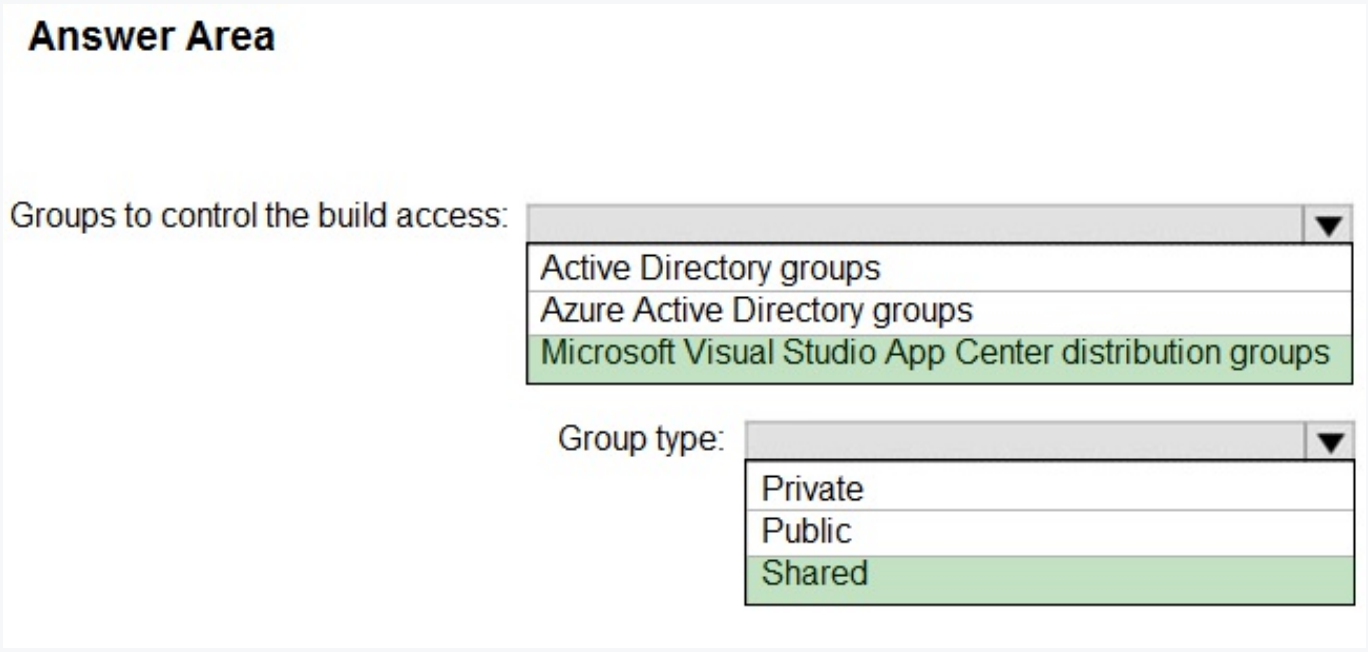
Question: 610 CertyIQ

HOTSPOT -
Your company is creating a suite of three mobile applications.
You need to control access to the application builds. The solution must be managed at the organization level.
What should you use? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:



Explanation:

Box 1: Microsoft Visual Studio App Center distribution Groups

Distribution Groups are used to control access to releases. A Distribution Group represents a set of users that can be managed jointly and can have common access to releases. Example of Distribution Groups can be teams of users, like the QA Team or External Beta Testers or can represent stages or rings of releases, such as Staging.

Box 2: Shared -

Shared distribution groups are private or public distribution groups that are shared across multiple apps in a single organization. Shared distribution groups eliminate the need to replicate distribution groups across multiple apps.

Note: With the Deploy with App Center Task in Visual Studio Team Services, you can deploy your apps from Azure DevOps (formerly known as VSTS) to App

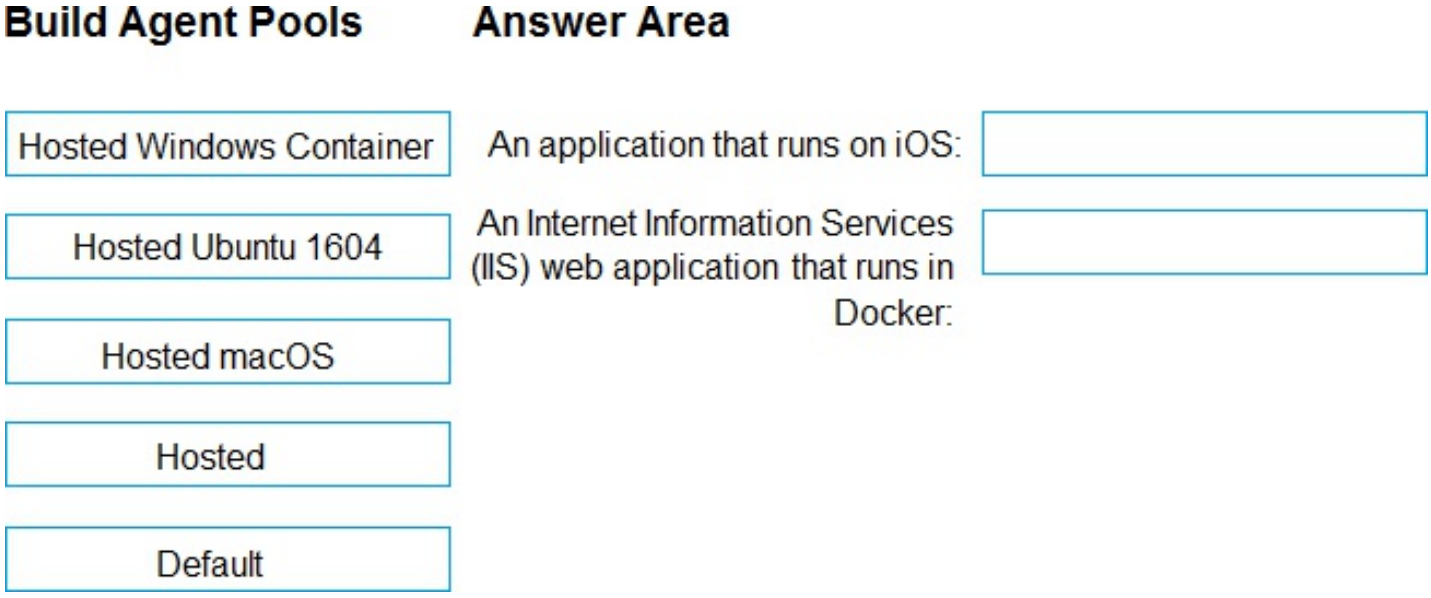
Center. By deploying to App Center, you will be able to distribute your builds to your users.

References:

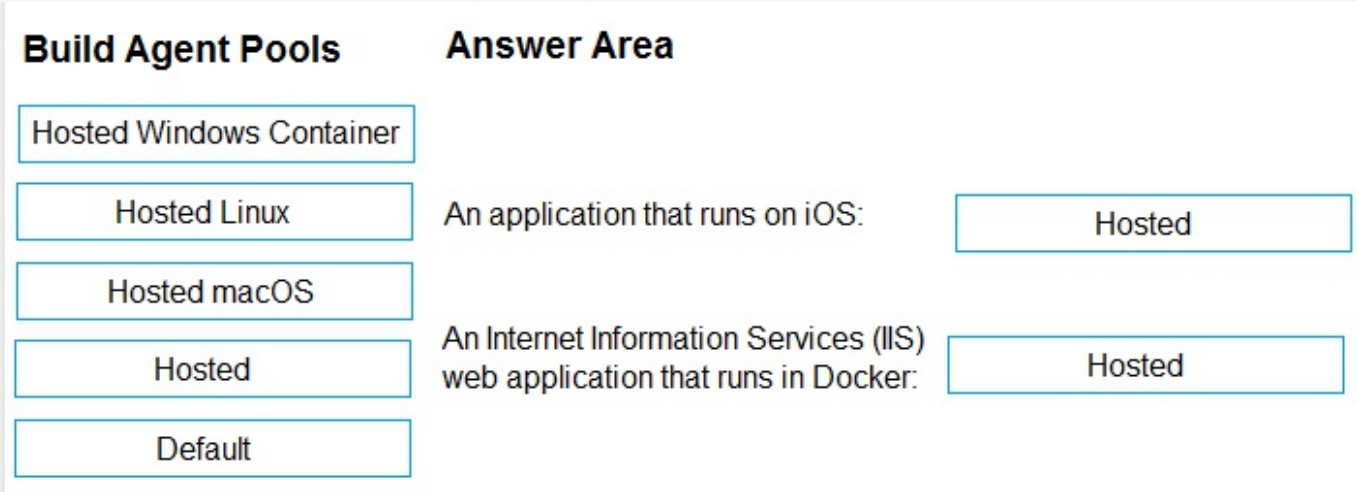
https://docs.microsoft.com/en-us/appcenter/distribution/groups

Question: 611 CertyIQ

DRAG DROP -
You are configuring Azure DevOps build pipelines.
You plan to use hosted build agents.
Which build agent pool should you use to compile each application type? To answer, drag the appropriate build agent pools to the correct application types. Each build agent pool may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Hosted

Box 2: Hosted -

Hosted pool (Azure Pipelines only): The Hosted pool is the built-in pool that is a collection of Microsoft-hosted agents.

Incorrect Answers:

Default pool: Use it to register self-hosted agents that you've set up.

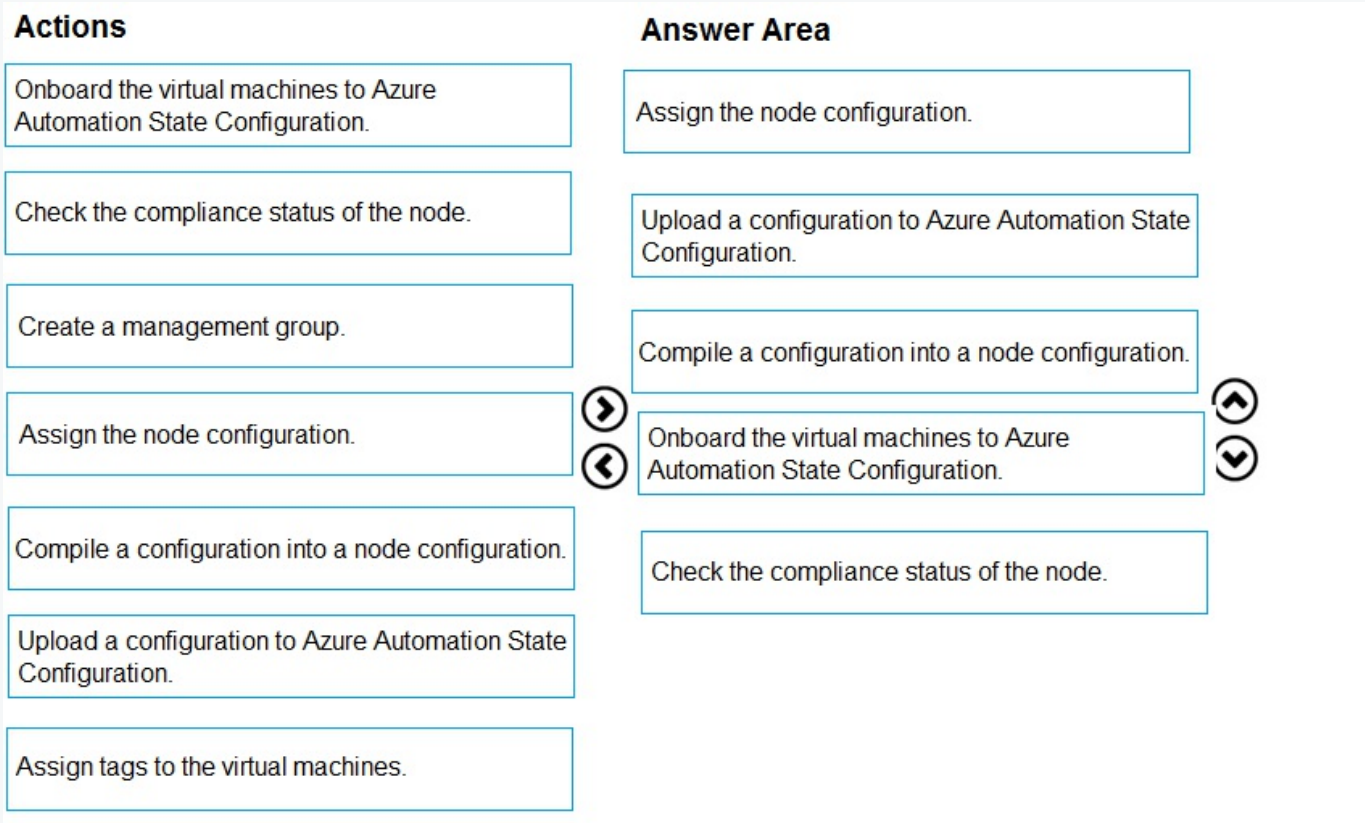
Hosted Windows Container pool (Azure Pipelines only): Enabled you to build and release inside Windows containers. Unless you're building using containers,

Windows builds should run in the Hosted VS2017 or Hosted pools.

Question: 612 CertyIQ

DRAG DROP -You need to use Azure Automation State Configuration to manage the ongoing consistency of virtual machine configurations.Which five actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.Select and Place:

Answer:



Explanation:

Step 1: Assign the node configuration.

You create a simple DSC configuration that ensures either the presence or absence of the Web-Server Windows Feature (IIS), depending on how you assign nodes.

Step 2: Upload a configuration to Azure Automation State Configuration.

You import the configuration into the Automation account.

Step 3: Compiling a configuration into a node configuration

Compiling a configuration in Azure Automation

Before you can apply a desired state to a node, a DSC configuration defining that state must be compiled into

one or more node configurations (MOF document), and placed on the Automation DSC Pull Server.

Step 4: Onboard the virtual machines to Azure State Configuration

Onboarding an Azure VM for management with Azure Automation State Configuration

Step 5: Check the compliance status of the node.

Viewing reports for managed nodes. Each time Azure Automation State Configuration performs a consistency check on a managed node, the node sends a status report back to the pull server. You can view these reports on the page for that node.

On the blade for an individual report, you can see the following status information for the corresponding consistency check:

The report status "" whether the node is "Compliant", the configuration "Failed", or the node is "Not Compliant" (when the node is in ApplyandMonitor mode and the machine is not in the desired state).

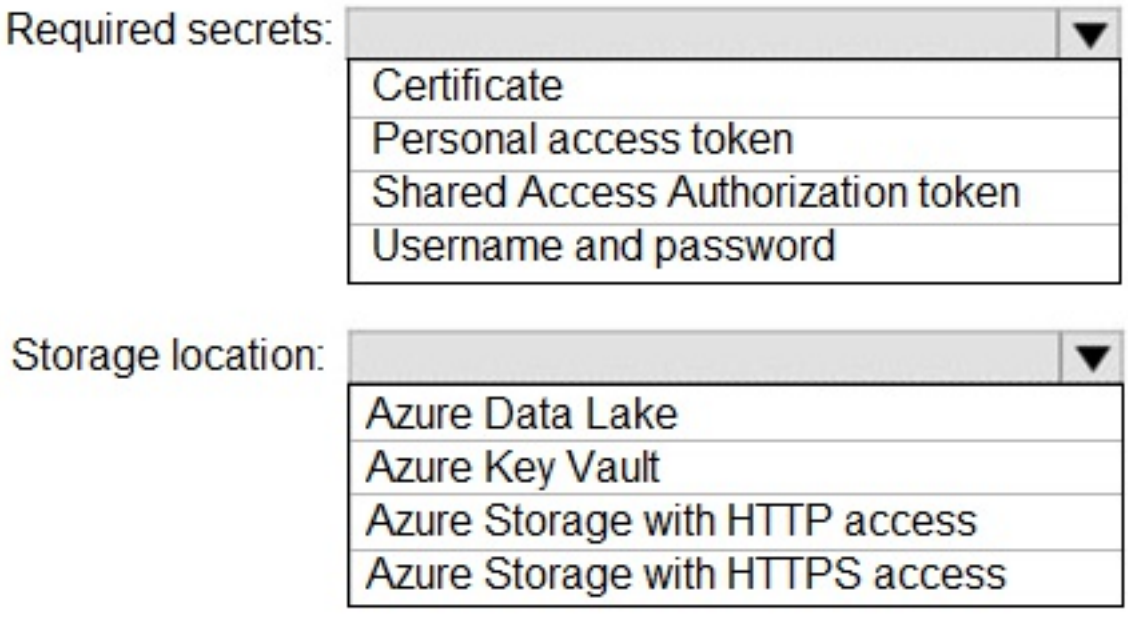
References:

https://docs.microsoft.com/en-us/azure/automation/automation-dsc-getting-started

Question: 613 CertyIQ

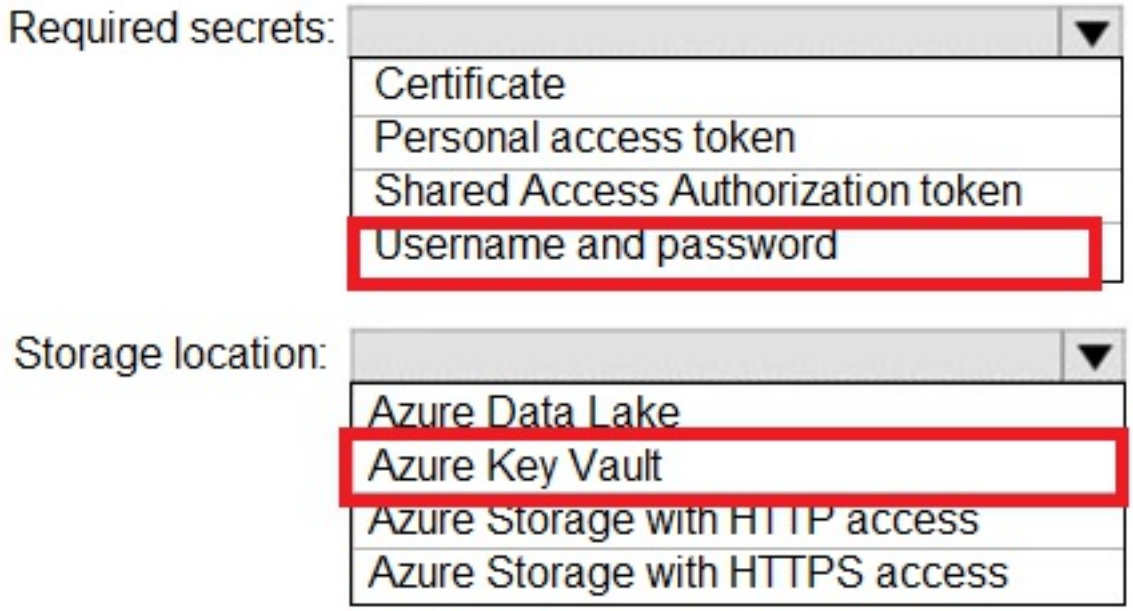
HOTSPOT -
You need to configure a cloud service to store the secrets required by the mobile applications to call the share pricing service.
What should you include in the solution? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area



Answer:

Answer Area



Explanation:

1. Required Secrets: Username and password

The most generic and fundamental secret type used for application-to-service authentication is a **username and password** (or API Key, which is equivalent to a secret key).

Certificate is overkill and complex for simple service access by a mobile app.

Personal Access Token (PAT) is typically used for Git repositories (like Azure DevOps/GitHub) or user-specific credentials, not for a generic mobile app calling a shared service.

Shared Access Authorization Token (or SAS Token) is primarily used for granting time-limited, specific permissions to resources in **Azure Storage**, not for authenticating to a general "share pricing service."

2. Storage Location: Azure Key Vault

Azure Key Vault is the **dedicated, industry-standard cloud service** designed specifically for the secure storage and management of **secrets**, keys, and certificates.

It offers strong encryption (with keys protected by Hardware Security Modules - HSMs), fine-grained access policies (RBAC), and full audit logging.

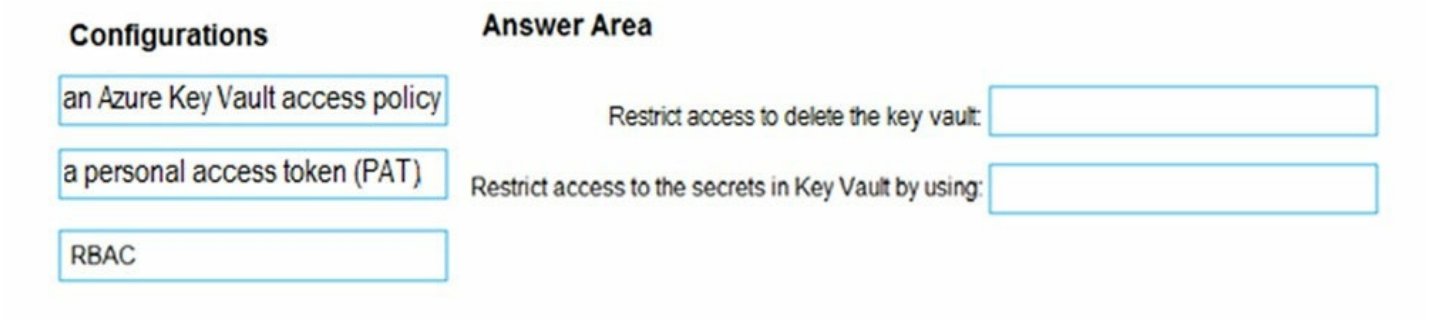
Azure Storage (with or without HTTP/HTTPS access) and **Azure Data Lake** are designed for storing large amounts of data (files, blobs, queues, unstructured data), **not** for the secure, centralized management of application credentials. Storing passwords in a storage account is a security anti-pattern.

References:

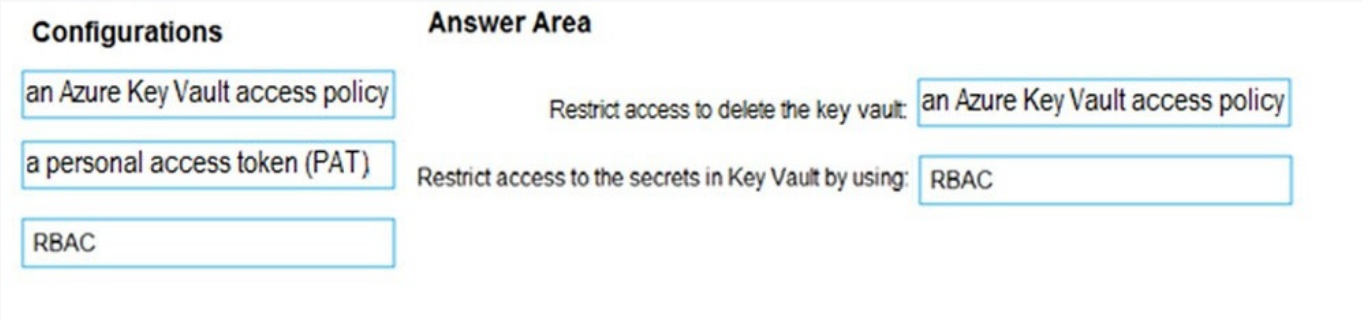
https://docs.microsoft.com/en-us/rest/api/storageservices/authorize-with-shared-key

Question: 614 CertyIQ

DRAG DROP -
Your company has a project in Azure DevOps.
You plan to create a release pipeline that will deploy resources by using Azure Resource Manager templates. The templates will reference secrets stored in Azure
Key Vault.
You need to recommend a solution for accessing the secrets stored in the key vault during deployments. The solution must use the principle of least privilege.
What should you include in the recommendation? To answer, drag the appropriate configurations to the correct targets. Each configuration may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: An azure key vault access policy.

A Key Vault advanced access policy is the specific setting within the Key Vault configuration used to enable interaction with core Azure services.

The required setting is **"Enable access to Azure Resource Manager for template deployment."** When an ARM template is used to deploy resources and needs to retrieve a secret from Key Vault (using dynamic references), this advanced policy must be enabled.

Box 2: RBAC -

Management plane access control uses RBAC.

The management plane consists of operations that affect the key vault itself, such as:

- Creating or deleting a key vault.
- Getting a list of vaults in a subscription.
- Retrieving Key Vault properties (such as SKU and tags).
- Setting Key Vault access policies that control user and application access to keys and secrets.

RBAC (Role-Based Access Control) is the recommended modern method for managing fine-grained permissions to secrets, keys, and certificates within Azure Key Vault, fully aligning with the **principle of least privilege**.

You would assign a specific Azure role (like **Key Vault Secret User** or a custom role with minimal permissions) to the service principal or managed identity running the Azure DevOps pipeline.

While an **Access Policy** (the older model) can also restrict access, **RBAC** is the Azure standard for least privilege and is preferred for new configurations due to its consistency with permissions across all Azure resources.

References:

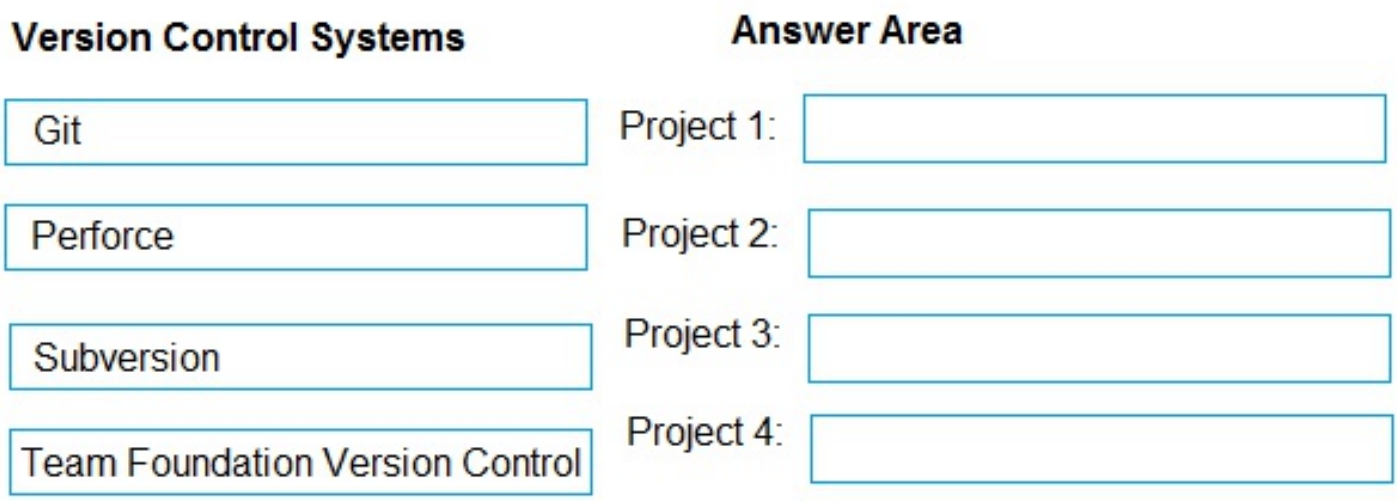
https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-tutorial-use-key-vault

Question: 615 CertyIQ

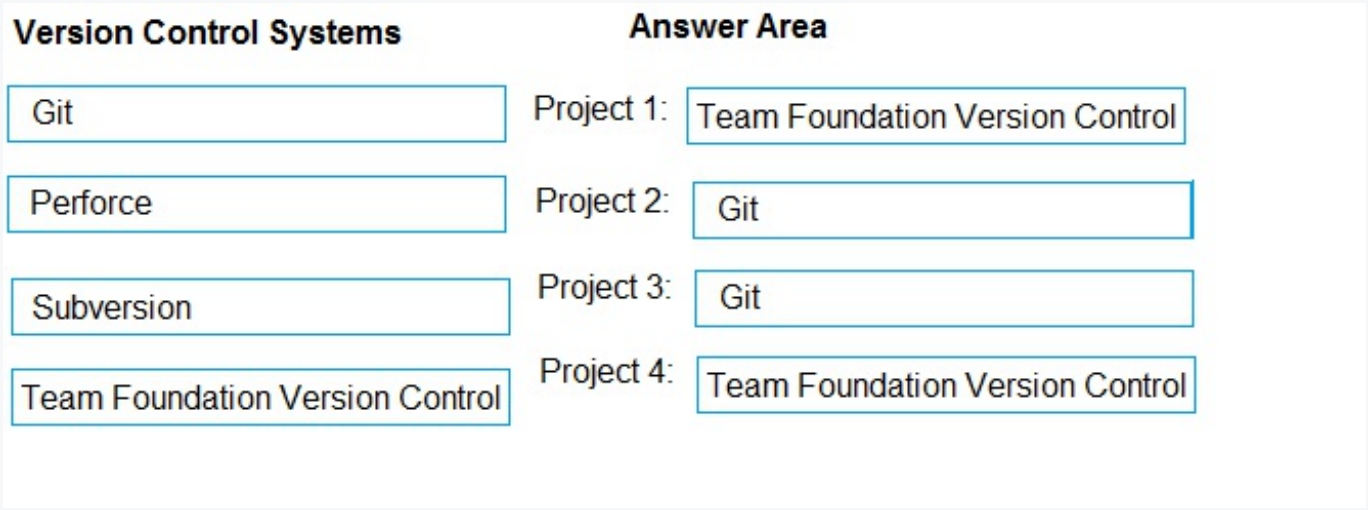
DRAG DROP -
Your company has four projects. The version control requirements for each project are shown in the following table.

Project	Requirement
Project 1	Project leads must be able to restrict access to individual files and folders in the repository.
Project 2	The version control system must enforce the following rules on the server before merging any changes to the main branch: • Changes must be reviewed by at least two project members. • Changes must be associated by at least one work item
Project 3	The project members must be able to work in Azure Repos directly from Xcode.
Project 4	The release branch must only be viewable or editable by the project leads.

You plan to use Azure Repos for all the projects.
Which version control system should you use for each project? To answer, drag the appropriate version control systems to the correct projects. Each version control system may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.
Select and Place:



Answer:



Explanation:

Box 1: Team Foundation Version Control

TFVC lets you apply granular permissions and restrict access down to a file level.

Box 2: Git -

Git is the default version control provider for new projects. You should use Git for version control in your projects unless you have a specific need for centralized version control features in TFVC.

Box 3: Git

Box 4: Team Foundation Version Control

Question: 616 CertyIQ

Use the drop-down menus to select the answer choice that answers each question based on the information presented in the graphic.
NOTE: Each correct selection is worth one point.
Hot Area:

Answer Area

How many stages have triggers set?

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

Which component should you modify to enable continuous delivery?

- The Development stage
- The Internal Review stage
- The Production stage
- The Web Application artifact

Answer:

Answer Area

How many stages have triggers set?

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

Which component should you modify to enable continuous delivery?

- The Development stage
- The Internal Review stage
- The Production stage
- The Web Application artifact

Explanation:

Box 1: 7

1. How many stages have triggers set?

The image shows the following stages:

1. **Development**
2. **Internal Review**
3. **QA**
4. **Stakeholder Review**
5. **Pre-Prod**

6. **Load Test**
7. **Production**

In Azure DevOps, the lines connecting stages often represent the flow, and an icon like a person ($\hat A $) or a lock ($\boldsymbol i $) indicates a **Pre-deployment Approval** or a **Gate**.

In this diagram, the small person icon ($\hat A $), representing an **Approval**, is present on the incoming line for **every single stage**.

Crucially, every stage must have some form of trigger (either automated or manual/approval-based) to be executed. If a stage is part of the pipeline, it has a trigger mechanism configured.

Given the common exam context where "triggers set" means "how many stages are configured to potentially run," and the fact that all **seven** stages are present and connected in the flow, they all have a trigger defined (even if it's a manual approval trigger).

New Conclusion for Box 1: Since all seven stages are connected and configurable, they all have a trigger defined, even if it's a manual approval.

2. Which component should you modify to enable continuous delivery?

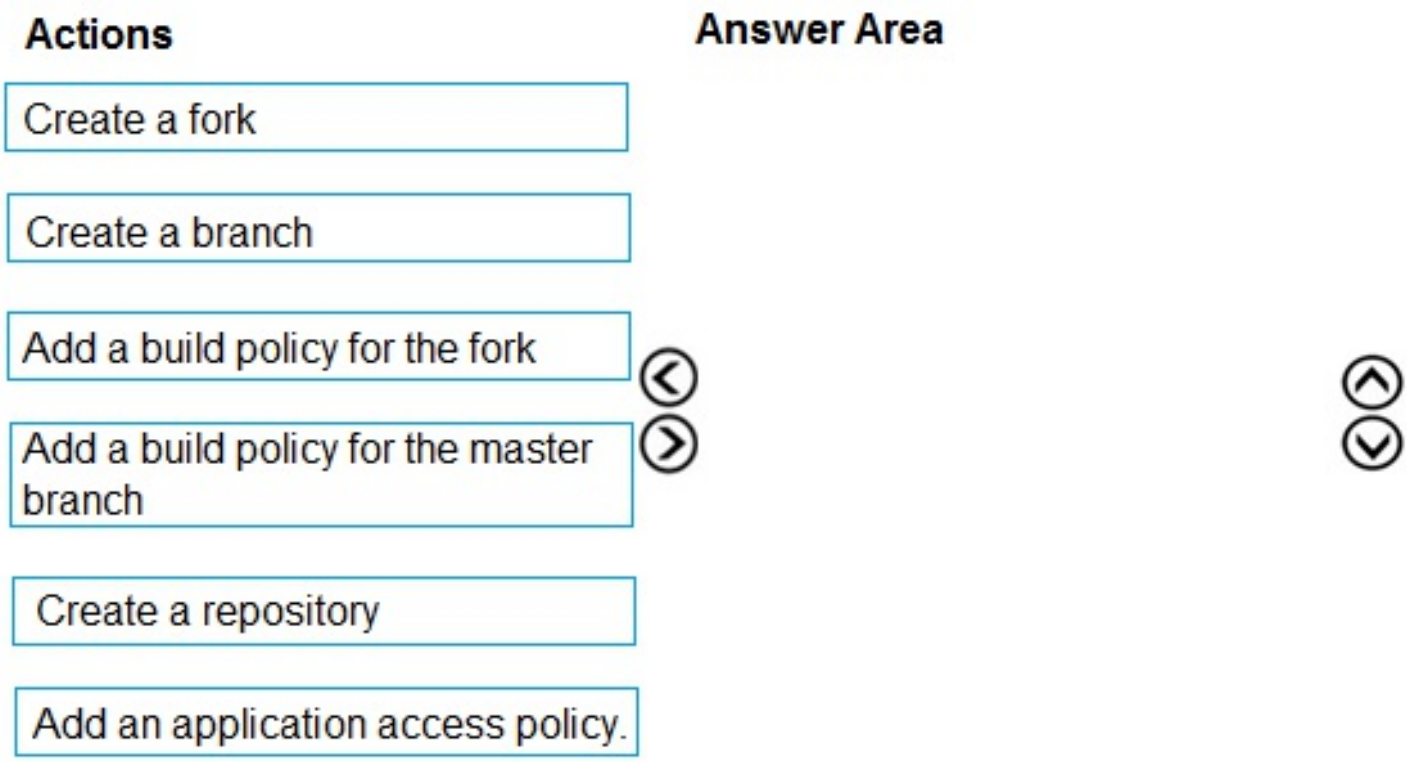
Box 2:**The Web Application artifact**,

is the correct component for enabling **Continuous Delivery (CD)**.

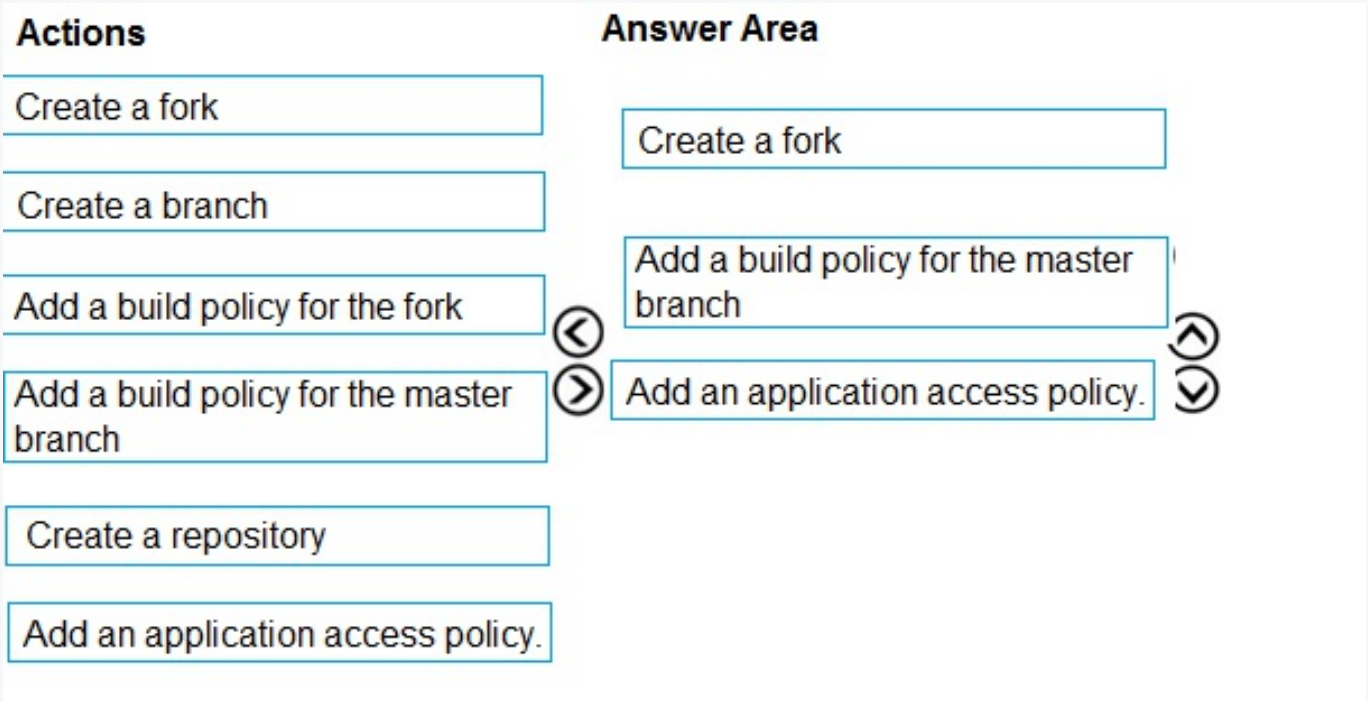
Continuous Delivery (CD) means automatically creating a new release whenever the source artifact (the build) is updated.
You configure the **Continuous deployment trigger** on the **artifact source** itself (the **Web Application artifact**). This is the setting that watches the source (e.g., Azure Artifacts, Jenkins, etc.) and automatically creates a new release instance.
Modifying the **Internal Review stage** (or any stage) only controls what happens after a release has been created and deployment has started; it doesn't initiate the creation of the release itself.

Question: 617 CertyIQ

DRAG DROP -You need to implement the code flow strategy for Project2 in Azure DevOps.Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.
Select and Place:



Answer:



Explanation:

Create a fork

- Add a build policy for the master branch

- Add an application access policy

https://docs.microsoft.com/en-us/azure/devops/repos/git/forks?view=azure-devops&tabs=visual-studio#whats-in-a-fork

Question: 618 CertyIQ

You are creating a new console application using C#. You want to upload a new file into a container (backup) of Azure Storage.

```
BlobContainerClient container = new BlobContainerClient(connectionStri
container.CreateIfNotExists (PublicAccessType.Blob);
var blockBlob = container.GetBlobClient("pic.png");
using (var fileStream = System.IO.File.OpenRead(@"c:\temp\images\pic.p

<line>
```

What you should write instead of ?

A. BlockBlob.upload(blockBlob, fileStream, "pic.png")
B. blockBlob.Upload(fileStream)
C. blockBlob (filestream). Upload
D. fileStream.Upload(blockBlob)

Answer: B

Explanation:

A. BlockBlob.upload(blockBlob, fileStream, "pic.png")

Incorrect. There is no static upload method on a class named BlockBlob in the Azure SDK.

B. blockBlob.Upload(fileStream)

Correct. In the Azure.Storage.Blobs SDK for .NET, the BlobClient (retrieved here as blockBlob) exposes an **Upload** method that takes a stream. This uploads the contents of fileStream to the blob named "pic.png" in the specified container.

C. blockBlob(filestream).Upload

Incorrect. This is invalid syntax in C#.

D. fileStream.Upload(blockBlob)

Incorrect. Streams do not have an Upload method. The Upload method is on the BlobClient object.

Question: 619 CertyIQ

You want to allow traffic between two different virtual networks.
What should you do?

A. Use Azure Application Gateway
B. Create a firewall rule to allow traffic between the two virtual networks
C. Configure a virtual network peering

Answer: C

Explanation:

A.Use Azure Application Gateway

Incorrect. Azure Application Gateway is a layer 7 load balancer used for HTTP/HTTPS traffic routing, not for connecting virtual networks.

B. Create a firewall rule to allow traffic between the two virtual networks

Incorrect. Firewall rules control traffic flow at the network or application level, but they do not natively connect two separate virtual networks.

C. Configure a virtual network peering

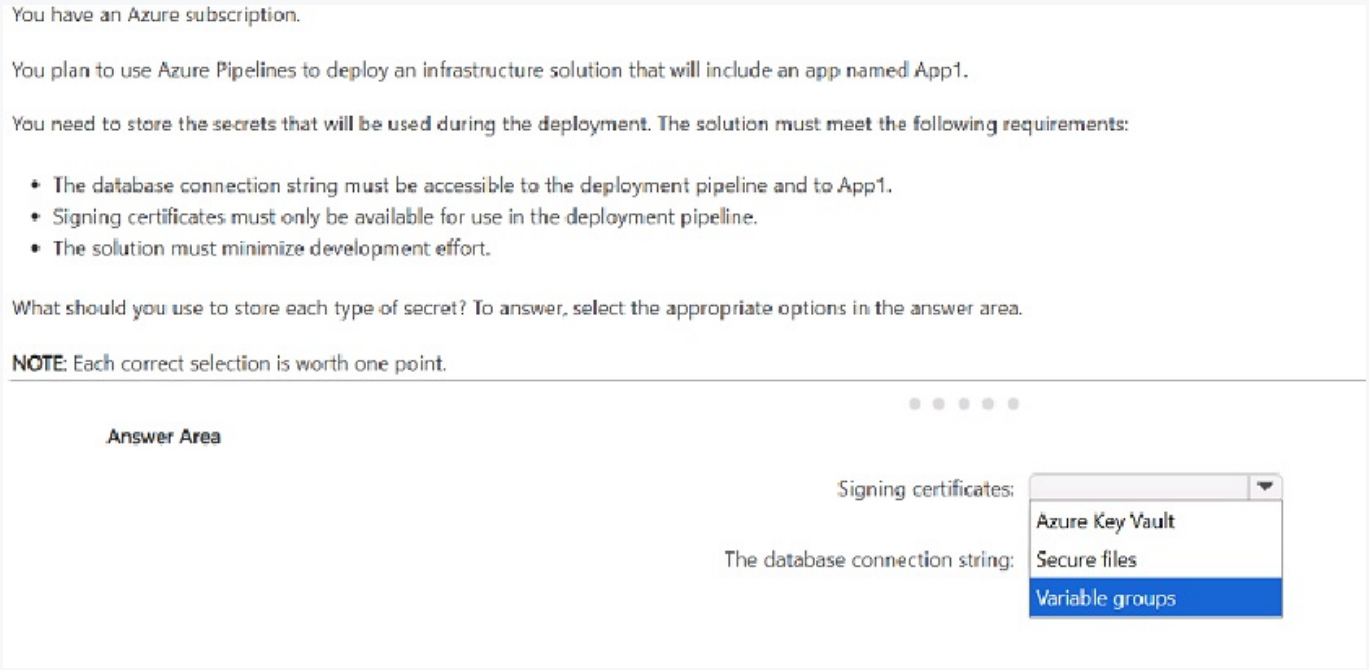
Correct. **Virtual network peering** connects two Azure virtual networks, allowing resources in both networks to communicate with low latency and high bandwidth. This is the rAecommended method to allow traffic between VNets.

Question: 620

CertyIQ

You have an Azure subscription.You plan to use Azure Pipelines to deploy an infrastructure solution that will include an app named App1.You need to store the secrets that will be used during the deployment. The solution must meet the following requirements:The database connection string must be accessible to the deployment pipeline and to App1.Signing certificates must only be available for use in the deployment pipeline.The solution must minimize development effort.What should you use to store each type of secret? To answer, select the appropriate options in the answer area.NOTE: Each correct selection is worth one point.

Answer:



Explanation:

The database connection string: Variable groups.

Question: 621

CertyIQ

You have an Azure Pipelines pipeline.
You need to add a Manual Intervention task that will pause the deployment, require users to alter a configuration manually, and then resume the deployment.
What should you add to the pipeline first?

A. an agentless job
B. a release gate
C. a deployment group job
D. an agent job

Answer: A

Explanation:

A. an agentless job

Correct. Manual Intervention tasks are added to **agentless jobs** because they do not require a build agent to execute. An agentless job allows the pipeline to pause and wait for a human to approve or perform a manual action before continuing.

B. a release gate

Incorrect. Release gates automatically evaluate conditions (like monitoring metrics or approvals) before a deployment starts but do not provide a pause for manual actions mid-deployment.

C. a deployment group job

Incorrect. Deployment group jobs run on a set of target machines using agents; they are not required for Manual Intervention tasks.

D. an agent job

Incorrect. Agent jobs require a build agent to run tasks. Manual Intervention does not need an agent, so an **agentless job** is required.

Question: 622 CertyIQ

You have an Azure subscription. The subscription contains virtual machines that run either Windows Server or Ubuntu.
You need to recommend a configuration management solution for the virtual machines. The solution must meet the following requirements:

- Minimize the number of additional servers required.
- Ensure that configuration management can be performed from the Azure portal.
- Ensure that status reports can be forwarded directly to a Log Analytics workspace.

What should you include in the recommendation?

A. the State configuration (DSC) settings
B. Bicep
C. PowerShell Desired State Configuration (DSC)
D. Azure Resource Manager (ARM) templates

Answer: A

Explanation:

A. the State configuration (DSC) settings

Correct. Azure Automation **State Configuration (DSC)** is a cloud-based configuration management solution. It

does not require additional on-premises servers, can be managed directly from the Azure portal, and integrates with Log Analytics for reporting and compliance tracking.

B. Bicep

Incorrect. Bicep is an Infrastructure as Code (IaC) language used to deploy Azure resources, not for ongoing configuration management of VMs.

C. PowerShell Desired State Configuration (DSC)

Incorrect. Traditional PowerShell DSC requires additional infrastructure (such as a pull server) unless integrated with Azure Automation. On its own, it does not meet the requirement to minimize additional servers or provide direct portal-based management.

D. Azure Resource Manager (ARM) templates

Incorrect. ARM templates are used to deploy resources, not to manage or enforce configuration state on running virtual machines.

Question: 623 CertyIQ

You manage package feeds by using Azure Artifacts.
You plan to create a new package feed that will include the following views:

- @Local
- @Latest
- @Release
- @Prerelease

Which view should you create manually?

A. @Latest
B. @Local
C. @Release
D. @Prerelease

Answer: C

Explanation:

A. @Latest

Incorrect. @Latest is not a default system view and is not automatically created, but it is also not a built-in standard like @Local. The key built-in default view is @Local.

B. @Local

Incorrect. @Local is created automatically when a new Azure Artifacts feed is created. It represents the default view containing all directly published packages.

C. @Release

Correct. @Release is not created by default. If you want a dedicated view for stable release packages, you must create it manually.

D. @Prerelease

Incorrect. Like @Release, this is also not created automatically, but exam-wise the expected single answer is @Release as the manually created promotion view.

Question: 624 CertyIQ

You plan to update the Azure DevOps strategy of your company.
You need to identify the following issues as they occur during the company's development process:

- Licensing violations
- Prohibited libraries

Solution: You implement continuous deployment.
Does this meet the goal?

A. Yes
B. No

Answer: B

Explanation:

A. Yes

Incorrect. Continuous deployment automates the release of code to production, but it does not specifically detect licensing violations or prohibited libraries. Those issues are typically identified using dependency scanning or security/compliance tools.

B. No

Correct. To detect licensing violations and prohibited libraries, you would need tools such as dependency scanning, artifact scanning, or security analysis (for example, Azure Artifacts upstream sources with policies or third-party scanning tools). Continuous deployment alone does not address this requirement.

Thank you

Thank you for being so interested in the premium exam material.
I'm glad to hear that you found it informative and helpful.

If you have any feedback or thoughts on the bumps, I would love to hear them.
Your insights can help me improve our writing and better understand our readers.

Best of Luck

You have worked hard to get to this point, and you are well-prepared for the exam
Keep your head up, stay positive, and go show that exam what you're made of!

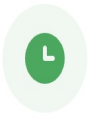
Feedback

More Papers



Future is Secured

100% Pass Guarantee



24/7 Customer Support

Mail us - certyiqofficial@gmail.com



Free Updates

Lifetime Free Updates!

Total: **624 Questions**

Link: https://certyiq.com/papers/microsoft/az-400