DUMPSARENA

Microsoft Azure DevOps Solutions

Microsoft AZ-400

Version Demo

Total Demo Questions: 59

Total Premium Questions: 591

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Configure processes and communications	66
Topic 2, Design and implement source control	65
Topic 3, Design and implement build and release pipelines	258
Topic 4, Develop a security and compliance plan	80
Topic 5, Implement an instrumentation strategy	97
Topic 6, Mix Questions	4
Topic 7, Case Study 1	5
Topic 8, Case Study 2	3
Topic 9, Case Study 3	3
Topic 10, Case Study 4	5
Topic 11, Case Study 5	5
Total	591

QUESTION NO: 1

Your company is currently making use of Team Foundation Server 2013 (TFS 2013), but intend to migrate to Azure DevOps.

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:

Using the TFS Database Import Service to perform the upgrade is correct because the supported migration path from an on-premises Team Foundation Server/Azure DevOps Server environment to Azure DevOps Services is a database import-based migration. For an older deployment such as TFS 2013, Microsoft's migration guidance requires first upgrading the on-premises server to a supported release, then validating and importing the collection database into Azure DevOps Services by using the Azure DevOps data migration tooling and import service. This approach preserves source code, work items, test artifacts, project history, identities, and other collection data far more completely than a manual or partial migration approach. In practical terms, the upgrade to a recent release prepares the collection schema and platform version so the import service can accept it, while the database import process performs the actual move into Azure DevOps Services. Microsoft documents this as the recommended path for high-fidelity migrations from Azure DevOps Server/TFS to Azure DevOps Services. See the Azure DevOps migration overview and the import guidance.

QUESTION NO: 2

You have a GitHub Actions workflow named workflow1 that is used to deploy an app name Appl. You need to ensure that App1 is deployed only when the following conditions are met:

• A pull request is merged to the main branch.

• The change includes a JavaScript file.

Solution. You configure the YAML file of workflow1 to include the following.

```
...
on:
  pull_request:
    paths:
    - '**.js'
jobs:
  app1-deploy:
  if: ${{ github.event.pull_request.base.ref == 'main' }}
...
```

Does this meet the goal?

A. Yes

B. No

ANSWER: A

Explanation:

Yes is correct because the workflow configuration shown uses GitHub Actions event filtering to limit execution to the intended pull request scenario. For a `pull_request` event, the `branches` filter evaluates the base branch of the pull request, so specifying `main` ensures the workflow applies to pull requests targeting the main branch. The `paths` filter restricts the workflow to changes that include matching files, such as JavaScript files when a pattern like `**.js` or `**/*.js` is used. To ensure the deployment happens after the pull request is completed, the workflow must use the `closed` pull request activity type and check that the pull request was actually merged, typically with a condition such as `github.event.pull_request.merged == true`. This combination ensures the deployment job runs only when the pull request has been merged into the main branch and the change set includes JavaScript content. GitHub documents these filters and event payload conditions in its workflow syntax and event reference: workflow syntax for pull request branch filters and events that trigger workflows.

QUESTION NO: 3

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. Create a branch in the build.

B. Implement a branch policy.

C. Create a fork in the build.

D. Implement a feature flag.

ANSWER: D

Explanation:

Implement a feature flag is correct because it lets the team deploy the updated App1 code to production while keeping the new service disabled until App2 is ready. A feature flag, also called a feature toggle, separates deployment from release: the code can be shipped as part of the normal deployment pipeline, but execution of the new functionality is controlled through configuration. When App2 becomes available, the service in App1 can be enabled by changing the flag state rather than rebuilding, rebranching, or redeploying the application. This approach is a common DevOps practice for progressive exposure, reducing deployment risk, and supporting scenarios where dependent services are not yet available. Microsoft specifically describes feature flags as a way to turn functionality on or off without deploying new code, and Azure App Configuration provides built-in feature management support for .NET and other applications. See Microsoft guidance on phasing features with feature flags and Azure App Configuration feature management concepts.

QUESTION NO: 4

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: D E

Explanation:

To finish implementing a self-hosted Azure Pipelines agent on an on-premises build server, you must authenticate the agent registration and then configure the local agent software. “From DevOps, create a personal access token (PAT)” is correct because the agent configuration process requires credentials that can register the agent with Azure DevOps, typically a PAT with the appropriate agent pool permissions. Microsoft’s setup guidance for self-hosted Windows agents specifically directs you to create a PAT before running the configuration script. “From the build server, run config.cmd” is also correct because this script performs the actual agent configuration: it connects the downloaded agent to your Azure DevOps organization, assigns it to an agent pool, sets the agent name, and configures whether it runs interactively or as a service. After configuration, Azure Pipelines can send jobs to that on-premises server while the source code can remain in the on-premises repository. See Microsoft’s agent setup documentation for the Windows self-hosted agent process: Microsoft Learn: Self-hosted Windows agents and PAT guidance: Microsoft Learn: Use personal access tokens.

QUESTION NO: 5

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 object ID

B. the tenant ID

C. the app ID

D. the client secret

E. the subscription ID

ANSWER: B C D

Explanation:

To authenticate to Azure by using an Azure AD service principal from an Azure Pipelines task, the task needs the identity of the application registration, the directory where that identity exists, and a credential proving that the pipeline is allowed to use it. The tenant ID identifies the Microsoft Entra ID tenant that contains the service principal. The app ID, also called the client ID or service principal ID in many Azure DevOps contexts, identifies the application identity that will sign in. The client secret is the password-like credential used with that app ID to complete the service principal authentication flow. This matches the standard Azure CLI service principal sign-in pattern, which uses an app/client ID, client secret, and tenant ID, and it is also consistent with Azure DevOps service connection configuration guidance for service principal authentication. Microsoft documents this pattern for Azure CLI sign-in with `az login --service-principal` and for Azure Pipelines connections to Azure. See Sign into Azure with a service principal using the Azure CLI and Connect to Microsoft Azure from Azure Pipelines.

QUESTION NO: 6

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: A C

Explanation:

Run npm install and specify the --production flag is correct because npm can install only the packages required for runtime use, excluding packages listed as development-only dependencies. This ensures the dependency tree available to the scanner represents what will be deployed to production rather than the full development environment. Modify the devDependencies section of the project's Package.json file is also correct because Node.js projects distinguish production dependencies from development tooling in package.json. Libraries used only for development should be declared under devDependencies, while runtime libraries belong under dependencies. When development-only packages are correctly categorized and npm is run in production mode, WhiteSource Bolt evaluates the production dependency set instead of reporting licenses for tools that are not shipped with the application. This aligns with npm's documented behavior for omitting dev dependencies during production installs and with the standard package.json dependency model. See the npm documentation for npm install and the package.json documentation for devDependencies.

QUESTION NO: 7

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:

No is correct because the requirement is to start a build automatically when code is checked in, which is handled by a continuous integration trigger on the build pipeline, not by pre-deployment conditions in a release pipeline. In Azure DevOps, CI triggers monitor source code changes in the configured repository and queue a build whenever matching changes are pushed. The “Batch changes while a build is in progress” setting is related to batching CI-triggered changes so that Azure Pipelines does not start multiple overlapping builds while one is already running; it does not, by itself from release pre-deployment conditions, create the required automatic build behavior. Pre-deployment conditions are part of the release process and control when a release stage can deploy, such as after a release is created, after approvals, or when gates pass. To meet the goal, the build pipeline must have continuous integration enabled for the relevant branch or repository. Microsoft documents CI triggers for Azure Pipelines here: Azure Pipelines build triggers, and release triggers separately here: Release triggers.

QUESTION NO: 8 - (SIMULATION)

SIMULATION

Task 8

In Project1, you need to create a service hook that will integrate with Azure Storage.

Use the queue1 queue from the storage48901628 storage account.

ANSWER: See the explanation for the answer

Explanation:

Task 8: Create a Service Hook to Integrate with Azure Storage (queue1)

Step 1: Verify the Storage Account and Queue

In the Azure portal (https://portal.azure.com):

Go to Storage accounts.

Find the storage48901628 storage account.

In the left menu, click on Queues.

Ensure that a queue named queue1 exists.If not, create it by:

Clicking + Queue.

Enter Name: queue1.

Click OK.

Step 2: Prepare the Storage Account’s Shared Access Signature (SAS)

Azure DevOps uses SAS tokens to send messages to a storage queue.

In the Azure portal, go to the storage48901628 storage account.

In the left menu, click Shared access signature.

Set the following:

Allowed services: Queue.

Allowed resource types: Object.

Allowed permissions: Add, Write.

Start and expiry date/time: Adjust as needed (e.g., 1 year).

Click Generate SAS and connection string.

Copy the Queue SAS URL (e.g., https://storage48901628.queue.core.windows.net/queue1?sv=...).

Step 3: Create the Service Hook in Azure DevOps

In your Azure DevOps Project (Project1):

In the bottom-left, click Project settings.

In the left menu, click Service hooks.

Click + Create subscription.

Step 4: Configure the Azure Storage Service Hook

In the Create a subscription wizard, choose Azure Storage Queues and click Next.

Choose the Trigger:

For example: Code pushed (or other events you want to send to the queue).

Click Next.

Configure the Action:

Queue endpoint URL: The Queue SAS URL you copied in Step 2 (including the queue1 endpoint and SAS token).

Example:

bash

Copy

https://storage48901628.queue.core.windows.net/queue1?sv=...

Step 5: Finalize and Test the Integration

Click Test to ensure Azure DevOps can send a message to the queue.

If the test succeeds, click Finish to create the service hook.

Step 6: Validate in the Azure Portal

In the Azure portal, go to storage48901628 > Queues > queue1.

You should see new messages appear in the queue whenever the configured Azure DevOps event (like push or PR) occurs.

QUESTION NO: 9

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:

At the root of the wiki, create a file named .order that defines the page hierarchy is correct for a published wiki in Azure DevOps. A published wiki is backed by a Git repository, and Azure DevOps uses special metadata files named .order to determine how pages and folders are displayed in the wiki navigation pane. The .order file lists the page or folder names in the desired sequence, with one item per line, allowing you to control the order shown to readers in the Azure DevOps portal. For ordering content inside subfolders, the same pattern can be used by adding an .order file within the relevant folder. This is the documented mechanism for controlling navigation order in a wiki that is published from repository content. Microsoft documents how published wikis are sourced from Git repositories and how wiki file structure, including ordering behavior, affects the rendered navigation. See Publish a Git repository to a wiki and Wiki file and folder structure.

QUESTION NO: 10

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: B D

Explanation:

Azure DevOps classic release pipelines support gates that pause a deployment flow until an external condition is satisfied. For ServiceNow change management, this is commonly implemented by adding a ServiceNow change management gate so that a change request is created or validated and the release waits until the change reaches the required state before production deployment continues. Defining a pre-deployment gate before the deployment to the Prod stage is correct because it directly blocks the production stage until the ServiceNow change process is complete. Defining a post-deployment gate after the deployment to the QA stage is also correct when QA is the stage immediately before production, because the pipeline can be held after QA and before the next stage begins, ensuring the change request is processed before any production components are deployed. Microsoft documents release gates as a way to collect external health or approval signals before or after a stage, and the Azure DevOps ServiceNow integration is designed for exactly this type of change-control validation. See Azure Pipelines gates and ServiceNow change management integration.

QUESTION NO: 11

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: C D E

Explanation:

To implement limited user testing on a virtual machine scale set, you can use a canary-style update by changing the scale set model and then applying that model only to selected instances. Since WebApp1 is installed by the Custom Script extension and the script is stored in sa1, Upload a custom script file to sa1 is correct because the updated installation or deployment logic must first be made available to the VMSS extension. Modify the Custom Script extension settings of VMSS1 is also correct because the scale set model must be updated to reference the new script or force the extension to rerun with the updated deployment instructions. Update the configuration of a virtual machine in VMSS1 is correct because applying the latest scale set model to only one or a small number of VMSS instances enables limited exposure of the new UI change before broader rollout. This aligns with the VMSS model-based update process, where changes to the scale set model can be applied manually to specific instances for controlled testing. See Microsoft guidance on upgrading scale set instances and the Custom Script Extension.

QUESTION NO: 12

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 lakes 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:

Yes is correct because Azure Pipelines pipeline caching is specifically designed to reduce build time by reusing files from previous pipeline runs, such as package manager dependency caches. For a React.js application, npm dependency installation can be a significant portion of the pipeline duration. By using the Cache task, commonly Cache@2, the pipeline can cache the npm package cache directory and restore it on later runs when the cache key still matches, typically using files such as package-lock.json as part of the key. This means unchanged packages do not need to be downloaded repeatedly from the npm registry, which can substantially reduce the time spent during npm install or npm ci.

Microsoft documents pipeline caching as a way to improve build performance when dependencies are restored repeatedly, and provides examples for npm scenarios using a cache key based on npm and the lock file. Therefore, recommending pipeline caching directly meets the stated goal of reducing pipeline execution time for repeated npm package installation in Azure Pipelines. See Pipeline caching and Build JavaScript apps.

QUESTION NO: 13

You have an Azure DevOps organization named Contoso that contains a project named Project 1.

You provision an Azure key vault name Keyvault1.

You need to reference Keyvault1 secrets in a build pipeline of Project1.

What should you do first?

A. Create an XAML build service.

B. Create a variable group in Project1.

C. Add a secure file to Project1.

D. Configure the security policy of Contoso.

ANSWER: B

Explanation:

Create a variable group in Project1 is correct because Azure Pipelines references Azure Key Vault secrets through a variable group in the project's Pipeline Library. In Azure DevOps, you create a variable group, enable the option to link secrets from an Azure Key Vault, select or create the Azure service connection, choose Keyvault1, and then select the secrets that should be exposed to the pipeline as variables. After the variable group is linked and authorized for use, the build pipeline can reference the selected Key Vault secrets just like pipeline variables, without storing the secret values directly in the YAML file or pipeline definition. Microsoft documents this as the standard approach for using Key Vault secrets in Azure Pipelines variable groups. The linked variable group stores only secret names and metadata, while the secret values are retrieved securely from Azure Key Vault at runtime. See Link a variable group to secrets in Azure Key Vault and Use variable groups.

QUESTION NO: 14

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.

D18912E1457D5D1DDCBD40AB3BF70D5D

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: B C E

Explanation:

To let a server application securely retrieve API secrets from Azure Key Vault without embedding credentials in configuration files, the application should be modified to access the key vault by using the Azure SDK or Key Vault REST APIs. Deploying a virtual machine that uses a system-assigned managed identity provides the application host with an automatically

managed Microsoft Entra identity, so the app can request tokens for Key Vault without storing a client secret, certificate, or password on the Server Core machine. A Key Vault access policy then grants that managed identity the required secret permissions, such as Get or List, allowing the application to retrieve only the secrets it needs. This pattern is a Microsoft-recommended approach for workloads running on Azure virtual machines because it reduces secret sprawl and avoids manual credential rotation for the app's Azure authentication. For implementation details, see Microsoft's guidance on using Key Vault with managed identities for Azure VMs at Use Key Vault from an Azure virtual machine and the Key Vault configuration provider guidance at Azure Key Vault configuration provider in ASP.NET Core.

QUESTION NO: 15

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. Reviewthe 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: A C

Explanation:

Reviewthe historical graph for the agent pools is correct because Azure DevOps provides an agent pool consumption report that shows historical utilization for an agent pool. This report is designed for understanding pool activity over time, including how many jobs are running or queued, which helps identify capacity pressure and whether additional parallel jobs or agents are needed. Query the TaskAgentPoolSizeSnapshot/TaskAgentPoolSizeSnapshots endpoint is also correct because Azure DevOps Analytics exposes pipeline and agent-pool-related data through OData entity sets. TaskAgentPoolSizeSnapshots provides time-based agent pool capacity data that can be queried directly or used in custom dashboards and Power BI reports to analyze agent pool trends programmatically. Together, these two approaches cover both the built-in portal experience and a query-based reporting approach. Microsoft documents the built-in pool consumption report in Pool consumption report and describes Analytics OData entities for Azure Pipelines in the Azure Pipelines Analytics entity reference.

QUESTION NO: 16

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. parameters

B. modules

C. resources

D. outputs

ANSWER: B

Explanation:

modules is correct because Bicep modules are the built-in mechanism for reusing deployment logic from other Bicep files. A module lets one Bicep template, such as Template3, reference another Bicep template, such as Template1 or Template2, and deploy the resources defined there as part of the overall deployment. This is the recommended approach when you want to break infrastructure definitions into smaller, reusable units, for example separating virtual machine deployment logic from website deployment logic and then composing them from a parent template. In Template3, you would define module declarations that point to Template1 and Template2, optionally pass parameter values into them, and consume their outputs if needed. Microsoft's Bicep documentation describes modules as a way to improve readability, enable reuse, and simplify complex deployments. See Bicep modules and Bicep file structure and syntax for the official guidance.

QUESTION NO: 17

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 trusted root certificate

B. a publisher certificate

C. Azure Active Directory (Azure AD)

D. GitHub user credentials

E. a personal access token (PAT)

ANSWER: D E

Explanation:

The correct authentication methods are GitHub user credentials and a personal access token (PAT). When you create a GitHub connection from Azure Boards, Azure DevOps can authenticate to GitHub by having you sign in with your GitHub account and authorize access to the repositories that Azure Boards needs to link to work items, commits, pull requests, and branches. Microsoft documentation for connecting Azure Boards to GitHub describes this sign-in and authorization flow for GitHub repositories. A personal access token (PAT) is also a supported authentication method, especially in scenarios where token-based access is required, such as connecting to GitHub Enterprise Server or using an account/token with the appropriate repository permissions. The PAT must grant sufficient permissions for Azure Boards to read repository metadata and create the integration. These authentication approaches are part of the supported Azure Boards-GitHub connection experience documented by Microsoft. See Connect Azure Boards to GitHub and Connect Azure Boards to GitHub Enterprise Server.

QUESTION NO: 18

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:

A single long-running branch with multiple short-lived feature branches is the best fit for this agile Git workflow. The long-running branch, typically main, represents the stable integration line and should be kept in a releasable state through pull requests, automated builds, tests, and branch policies. Short-lived feature branches allow developers to work independently on separate tasks without destabilizing the main branch. Because each feature is isolated in its own branch, the team can experiment freely, validate changes before merging, and abandon incomplete or unsuccessful work simply by deleting the branch. This aligns with Microsoft guidance to keep a simple branching strategy, use feature branches for new work, and protect important branches with quality checks before changes are integrated. Azure Repos branch policies can enforce reviewers, build validation, and other checks so that the shared branch remains healthy and releasable. See Microsoft's guidance on adopting a Git branching strategy and Azure Repos branch policies.

QUESTION NO: 19

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 togs from multiple sources

• Identify the root cause of issues.

What advanced feature of a monitoring tool should you include in the solution?

A. synthetic monitoring

B. Alert Management

C. analytics

D. directed monitoring

ANSWER: C

Explanation:

Analytics is the correct feature to include because it provides the ability to collect, query, correlate, and interpret log and telemetry data from multiple sources. In an Azure DevOps and Azure monitoring context, analytics capabilities are typically delivered through tools such as Azure Monitor Logs and Log Analytics, where data from pipelines, applications, infrastructure, and other services can be queried together. This makes it possible to search across different log streams, filter events by time or severity, detect patterns, and correlate failures with related platform or deployment activity.

For root cause analysis, analytics is especially important because it goes beyond simply showing that something failed. It helps teams investigate why the failure occurred by using structured queries, aggregations, visualizations, and historical comparisons. Azure Monitor Logs supports powerful querying with Kusto Query Language, which is designed for exploring large volumes of operational data and identifying relationships between events. Microsoft describes Log Analytics as a tool for editing and running log queries against Azure Monitor Logs data, which aligns directly with parsing logs and diagnosing issues. See Log Analytics overview and Get started with log queries in Azure Monitor.

QUESTION NO: 20

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: A B E

Explanation:

Create a new feed in Azure Artifacts, publish the package to a feed, and add the package URL to the NuGet Package Manager settings in Visual Studio are the correct actions. Azure Artifacts provides centralized package management for teams and supports NuGet feeds that can be shared across an organization or project. Creating the feed gives the development team a managed package source where custom packages can be stored, versioned, secured, and discovered. Publishing the package to that feed makes the custom package available from the central source rather than requiring developers to copy files manually or retrieve packages from source control. Finally, adding the feed URL as a package source in Visual Studio's NuGet Package Manager settings allows developers to browse, install, restore, and update the package directly from Visual Studio. Once the feed is configured, Visual Studio and NuGet can detect available package versions from the feed, enabling developers to consume the latest published version through normal NuGet package management workflows. Microsoft documents this flow as creating an Azure Artifacts feed, connecting to the feed, and using NuGet-compatible tools such as Visual Studio to consume packages. See Get started with NuGet packages in Azure Artifacts and Consume NuGet packages from Azure Artifacts.

QUESTION NO: 21

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 ApiConnection trigger

B. a Request trigger

C. an HTTPWebhook trigger

D. an HTTP trigger

ANSWER: B

Explanation:

a Request trigger is correct because Azure Monitor/Application Insights alert actions invoke a Logic Apps workflow by calling an HTTP endpoint exposed by the workflow. In Azure Logic Apps, that endpoint is created by the built-in Request trigger, commonly shown in the designer as "When an HTTP request is received." After the workflow is saved, the trigger generates an HTTP POST URL that can be used by an Azure Monitor action group or webhook-style alert notification to start the

workflow. The Logic App can then parse the alert payload and run follow-up actions such as sending email, creating tickets, or notifying other systems.

This is the standard integration pattern documented by Microsoft for responding to Azure Monitor alerts with Logic Apps: create a workflow that starts with the Request trigger, then configure the alert action or action group to call that workflow. See Microsoft's guidance on handling Azure Monitor alerts with Logic Apps and the Logic Apps documentation for the Request trigger.

QUESTION NO: 22

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:

Azure PowerShell is correct because a Visual Studio Azure Resource Group deployment project includes a deployment script, commonly named Deploy-AzureResourceGroup.ps1, that is intended to deploy the project's Azure Resource Manager template and associated parameters. In an Azure DevOps release pipeline, the Azure PowerShell task can run this checked-in script directly by using an Azure service connection, so the pipeline reuses the same deployment logic that developers already use from Visual Studio. This minimizes administrative effort because you do not need to recreate the deployment process manually as several separate pipeline tasks or maintain a second deployment workflow. The Azure PowerShell task is specifically designed to run Azure PowerShell scripts in Azure Pipelines and authenticate them against a target Azure subscription. Microsoft documents the Azure PowerShell pipeline task here: Azure PowerShell task. Microsoft also describes deploying ARM templates through Azure Pipelines here: Integrate ARM templates with Azure Pipelines.

QUESTION NO: 23

You add the virtual machines as managed nodes in Azure Automation State Configuration.

You need to configure the computer in Group7.

What should you do?

A. Run the Register-AzureRmAutomationDscNode Azure Powershell cmdlet.

B. Modify the ConfigurationMode property of the Local Configuration Manager (LCM).

C. Install PowerShell Core.

D. Modify the RefreshMode property of the Local Configuration Manager (LCM).

ANSWER: D

Explanation:

Modify the RefreshMode property of the Local Configuration Manager (LCM) is correct because a computer that is not directly registered as an Azure virtual machine node must be configured to pull its Desired State Configuration from Azure Automation State Configuration. In PowerShell DSC, the LCM controls how a node receives and applies configurations. Setting RefreshMode to Pull tells the node to contact a DSC pull service, such as Azure Automation State Configuration, to retrieve its assigned configuration and required resources. This is the key LCM setting that changes the node from expecting locally pushed configurations to using a centralized pull-based configuration service. Azure Automation onboarding for non-Azure or manually configured machines relies on generating and applying a DSC metaconfiguration to the node, which includes the pull server endpoint, registration key, and LCM settings such as RefreshMode. Microsoft's documentation describes onboarding machines to Azure Automation State Configuration by configuring them as DSC nodes and using LCM pull settings. See Azure Automation State Configuration onboarding and Configuring the Local Configuration Manager.

QUESTION NO: 24

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: B C D

Explanation:

To allow cloud-hosted Jenkins to retrieve source code from Azure Repos, Jenkins needs an Azure Repos-compatible source control integration and valid credentials. Add the Team Foundation Server (TFS) plug-in to Jenkins is correct because the Jenkins TFS plug-in supports integration with Azure DevOps/TFS repositories, including polling and retrieving source from Team Foundation Version Control or Azure DevOps-hosted projects. Create a personal access token (PAT) in your Azure DevOps account is correct because Azure DevOps supports PATs as an authentication mechanism for tools and services that need access to Azure Repos. The PAT should be scoped appropriately, typically with Code read permissions for repository checkout. Add a personal access token to your Jenkins account is also correct because Jenkins must store and use that token as a credential when connecting to Azure Repos during the build checkout step. Microsoft documents PAT-based authentication for Azure DevOps at Use personal access tokens, and the Jenkins TFS integration is documented in the Jenkins TFS plug-in documentation.

QUESTION NO: 25

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 onepoint.

A. authorization credentials

B. the project name

C. the deployment group name

D. the organization URL

E. the agent pool name

ANSWER: A D

Explanation:

For an unattended self-hosted Azure Pipelines agent configuration, the script must include the organization URL and authorization credentials. The organization URL tells the agent which Azure DevOps organization or collection to register with, typically passed by using the `--url` parameter, such as `https://dev.azure.com/{organization}`. Authorization credentials are also required so the configuration process can authenticate as a user or service identity that has permission to register the agent. In common Azure DevOps Services scenarios, this is done with PAT-based authentication by passing values such as `--auth pat` and `--token`. Microsoft's unattended configuration guidance states that the agent setup must know the organization or collection URL and the credentials of someone authorized to set up agents; other setup responses can be supplied or defaulted depending on the scenario. See Microsoft's documentation for unattended agent configuration and PAT-based agent registration.

QUESTION NO: 26

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. Enable Test Impact Analysis (TIA).

B. Enable test slicing.

C. Clear Flaky tests included in test pass percentage

D. Set Flaky test detection to Off

E. Manually mark the test as flaky.

ANSWER: C E

Explanation:

To prevent an intermittently failing response-time test from causing the build pipeline to fail, the test should be treated as a flaky test in Azure Pipelines. Manually mark the test as flaky is correct because Azure DevOps lets you classify a known unreliable test as flaky from test results or test analytics when the failure is not a reliable indicator of product quality. After a test is marked flaky, Azure Pipelines can distinguish its failures from regular test failures.

Clear Flaky tests included in test pass percentage is also correct because this setting controls whether flaky tests affect the reported pass percentage and pipeline outcome. When flaky tests are excluded from the test pass percentage, their intermittent failures do not cause the pipeline to fail purely because of that known unstable test. This is the intended use of flaky test management: keep visibility into unreliable tests while preventing them from blocking builds unnecessarily. Microsoft documents this behavior in Azure Pipelines flaky test management and test result analysis guidance: Manage flaky tests and Review test results.

QUESTION NO: 27

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:

No is correct because the proposed solution does not establish a valid deployment mechanism for on-premises servers. The Download Build Artifacts task is used to retrieve artifacts produced by a build pipeline so they can be consumed later in a release or deployment job. By itself, it does not provide a way to execute deployment steps on on-premises machines or securely target those servers. For Azure Pipelines to deploy software into an on-premises environment, the pipeline must have an agent or deployment target that can reach those servers, such as a self-hosted Azure Pipelines agent installed on a machine in the on-premises network, a deployment group agent for classic release pipelines, or an environment virtual machine resource for YAML pipelines. Microsoft documents that agents run pipeline jobs, and self-hosted agents are commonly used when access to private networks or on-premises resources is required. Therefore, simply deploying a Docker build and adding an artifact download task does not satisfy the requirement to deploy the build artifacts to on-premises servers. See Azure Pipelines agents and Azure Pipelines virtual machine resources.

QUESTION NO: 28

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: D E

Explanation:

A self-hosted agent and an External Git service connection are the required components for this scenario. Because the Bitbucket Server is on-premises and protected by a firewall that blocks inbound Internet traffic, Azure DevOps Services and Microsoft-hosted build agents cannot directly reach the repository to clone source code. A self-hosted agent installed inside the corporate network can access the internal Bitbucket Server while still communicating outbound to Azure DevOps to receive jobs and publish results. This is the standard pattern when build infrastructure needs line-of-sight to private, on-premises resources.

An External Git service connection is also required so Azure DevOps can define and authenticate the connection to the non-Azure Repos Git repository. The service connection stores the repository endpoint and credentials used by the pipeline when checking out code from Bitbucket Server. Microsoft documents that Azure Pipelines supports external Git repositories and that self-hosted agents are appropriate when access to private network resources is required. See Pipeline options for Git repositories and Azure Pipelines agents.

QUESTION NO: 29

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:

Kanban boards is correct because Azure Boards uses Kanban boards specifically to help teams visualize the flow of work from backlog through completion. In Azure DevOps, a Kanban board provides an interactive, visual representation of work items as cards arranged in columns that reflect the team's workflow states. This aligns directly with the agile Kanban practice of visualizing work, limiting work in progress, and continuously improving flow. Teams can customize board columns, swimlanes, card fields, and rules so that the board matches how the team actually delivers work. This makes it well suited for tracking sprint-related work while still focusing on the movement of items across the delivery process.

Microsoft documentation describes Azure Boards Kanban boards as tools for visualizing the flow of work and monitoring progress as work items move across the board. For an Azure DevOps team that needs an agile visualization of work, Kanban boards are the intended component. See Microsoft's guidance on Kanban board fundamentals and customizing Kanban boards.

QUESTION NO: 30

You use Git for source control.

You need to commit a 3-G3 ZIP file that contains virtual machines used for testing. The solution must meet the following requirements:

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. Store files in Azure Storage and enable blob versions.

B. Install the Git IFS extension and associate the extension to ZIP files.

C. Use GZip to compress the file before committing the file.

D. Install the git-stash extension and associate the extension to ZIP files.

E. Install the git-fat extension and associate the extension to ZIP files.

ANSWER: B E

Explanation:

Large binary files such as multi-gigabyte ZIP archives should not be stored as normal Git objects because they rapidly bloat the repository and make cloning, fetching, and history operations inefficient. The appropriate approach is to use a large-file extension that stores a lightweight pointer in Git while keeping the large binary payload in external large-file storage. The intended action in “Install the Git IFS extension and associate the extension to ZIP files” is Git LFS, which is the Microsoft-supported approach in Azure Repos for managing large files in Git; ZIP files can be tracked by configuring the file pattern, such as through Git attributes. “Install the git-fat extension and associate the extension to ZIP files” is also a valid large-file-management approach because git-fat uses Git filters and attributes to keep large file content outside the regular Git object database while preserving the association with the repository. Microsoft documents Git LFS for Azure Repos as the recommended way to manage large files in Git: Manage and store large files in Git. The git-fat project documents the same external-storage pattern here: git-fat.

QUESTION NO: 31 - (HOTSPOT)

HOTSPOT

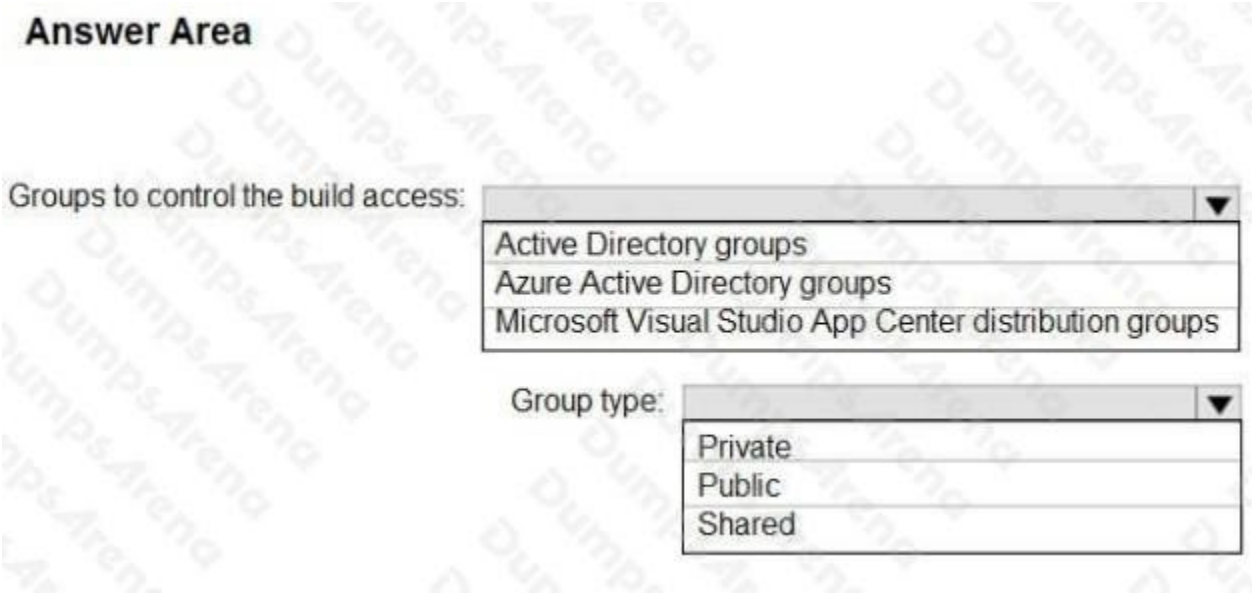
Your company is creating a suite of three mobile applications.

You need to control access to the application builds. The solution must be managed at the organization level.

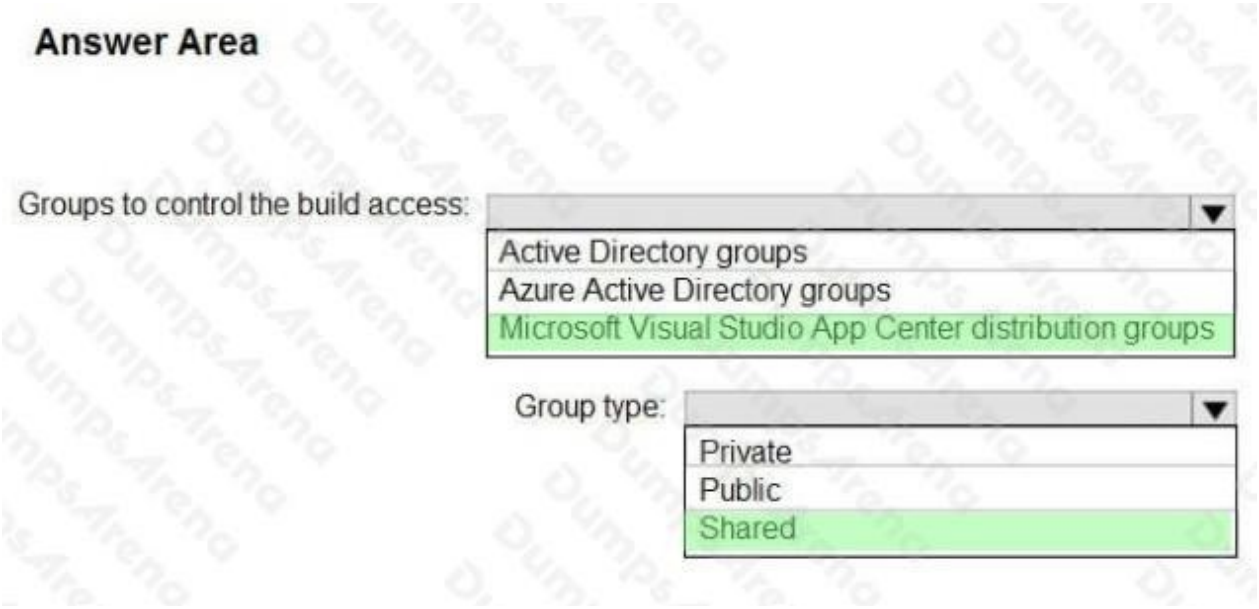
What should you use? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Hot Area:



ANSWER:



Explanation:

The correct selections are “Microsoft Visual Studio App Center distribution groups” and “Shared” because the scenario is specifically about controlling access to mobile application builds across a suite of multiple apps, while managing that access at the organization level. In Visual Studio App Center, distribution groups are used to define the users or testers who are allowed to receive and install distributed builds. These groups are commonly used for internal testing teams, beta testers, QA groups, or staged release rings, and they directly control who can access a release once a build is distributed.

Because the company has three mobile applications, the important detail is that the access model should not be managed separately for each individual app. App Center supports shared distribution groups, which are created at the organization level and can be reused across multiple apps within that organization. This avoids duplicating the same tester list or access policy for every app. Instead, the organization can maintain one shared group and apply it to releases from each of the mobile applications as needed.

This matches Microsoft’s App Center distribution model: distribution groups control release access, and shared distribution groups are intended for multi-app use within a single organization. Microsoft’s documentation describes distribution groups as the way to manage users for app releases and explains that shared groups can be used across apps in an organization. See the Microsoft documentation for App Center distribution groups here: App Center distribution groups. App Center’s distribution features are also described in the broader distribution documentation here: App Center Distribution.

QUESTION NO: 32

You use GitHub for source control of .NET applications.

You need to deploy a documentation solution that meets the following requirements:

Which two tools can you use to compile the website? Each correct answer presents a complete solution.

A. Jekyll

B. Medium

C. caret

D. WordPress

E. DocFX

DocFX is another tool that can be used to compile a static website from Markdown files stored in a GitHub repository. It is an open-source tool that can be used to generate API documentation, reference documentation, and other types of documentation from source code and other files. DocFX supports the use of Markdown for writing documentation and it can automatically recompile the website whenever changes are made to the documentation.

F. Jekyll

ANSWER: E F

Explanation:

Jekyll and DocFX are the appropriate tools for compiling a documentation website from source-controlled content. Jekyll is a static site generator commonly used with GitHub Pages. It can transform Markdown files, layouts, and configuration stored in a GitHub repository into a static website, making it a strong fit for lightweight project or product documentation. GitHub Pages has native support for Jekyll-based publishing, as described in the GitHub Pages and Jekyll documentation.

DocFX is also a complete solution, especially for .NET projects. It can generate documentation from Markdown files and extract API reference content from .NET source code and assemblies, then compile the result into a static documentation website. This makes it particularly suitable for .NET application documentation where conceptual articles and API reference need to be published together. The official DocFX documentation describes its ability to build static sites from source code and Markdown-based content.

QUESTION NO: 33

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) identities.

• 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 Data Explorer

B. Azure dashboards

C. Azure Monitor

D. Microsoft Power BI

ANSWER: B

Explanation:

Azure dashboards is the correct choice because a shared Azure portal dashboard is designed to provide a consolidated operational view across Azure resources, including resources that are spread across multiple resource groups. Dashboards can be published as Azure resources, which means access can be controlled through Azure role-based access control using Microsoft Entra ID identities. They also support Markdown tiles, making it possible to add formatted documentation, instructions, and context directly alongside monitoring visuals. For Azure Monitor data, you can pin charts and query-based visualizations, including Log Analytics results that use Kusto Query Language, to a dashboard so teams can view operational data in one place. Portal dashboard tiles are intended to show current service data rather than a separately cached reporting snapshot, which aligns with the requirement to use the latest available data for each visual. Microsoft documents Azure dashboards as customizable portal views that can be shared with RBAC, and Azure Monitor Log Analytics as using KQL for log queries and visualizations. See Create and share dashboards in the Azure portal and Log Analytics tutorial.

QUESTION NO: 34

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:

No is correct because the proposed solution does not satisfy the key requirement that the build must access an on-premises dependency management system. Microsoft-hosted agents, such as the Hosted VS 2019 agent pool, run on infrastructure managed by Microsoft in Azure and are recreated for each job. They are useful for standard cloud-based builds, but they do not inherently have network line-of-sight to private on-premises systems. For a pipeline that must reach internal dependency repositories or other private resources, Microsoft recommends using a self-hosted agent installed in an environment that has the required network access, such as the on-premises network or a connected virtual network. The Java Tool Installer task can help ensure the correct Java Development Kit is available during the build, but it does not solve private network connectivity or artifact publishing requirements. Azure DevOps can store build outputs as build artifacts, but the agent strategy must first allow the build to restore dependencies successfully from the internal system. See Microsoft guidance on Azure Pipelines agents and self-hosted Windows agents.

QUESTION NO: 35

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: A B

Explanation:

To remove a sensitive file from a GitHub repository, deleting the file in a new commit is not enough because the content remains available in the repository history. A history-rewriting tool is required to remove the file from all affected commits and then force-push the rewritten history. The git filter-branch command can rewrite Git history and remove a file from every commit where it appears, which makes it a valid complete solution for purging sensitive content from repository history. BFG

Repo-Cleaner is also a valid complete solution and is specifically designed to remove large files, passwords, credentials, and other sensitive data from Git repository history more quickly and safely than many manual approaches. GitHub documentation identifies these history-rewriting approaches for removing sensitive data and also recommends follow-up actions such as rotating any exposed secrets because the data may already have been cloned or cached. See GitHub's guidance on removing sensitive data from a repository and the BFG Repo-Cleaner documentation.

QUESTION NO: 36

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:

No is correct because a Subscription Health scan is a point-in-time assessment. It can identify security issues, misconfigurations, obsolete settings, and other risks in an Azure subscription at the moment the scan runs, but it does not continuously enforce or monitor the desired configuration state over time. The stated goal is to prevent configuration drift, which requires an ongoing assurance mechanism that repeatedly evaluates the subscription or project configuration and reports or acts on deviations from the approved baseline. In the Secure DevOps Kit for Azure model, this is the purpose of Continuous Assurance: it operationalizes recurring checks so configuration changes are detected after deployment, not only when an artifact or package is created. This aligns with the broader Azure governance approach of continuously evaluating resources against policies and compliance rules rather than relying only on one-time validation. See the Secure DevOps Kit Continuous Assurance guidance at AzSK Continuous Assurance and Microsoft's Azure Policy overview for continuous governance concepts at Azure Policy overview.

QUESTION NO: 37

You are designing a YAML template for use with Azure Pipelines. The template Will include the Outputfile parameter.

Which two methods can you use to reference the parameter? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

A. Option A

B. Option B

C. Option C

D. Option D

E. Option E

ANSWER: C D

Explanation:

Azure Pipelines YAML templates support template expressions that are evaluated during template expansion, before the pipeline runs. A template parameter such as Outputfile can be referenced through the parameters context by using property dereference syntax, such as `${{ parameters.Outputfile }}`, or by using index syntax, such as `${{ parameters['Outputfile'] }}`. Both forms are valid complete ways to access a parameter value in a template expression. Microsoft documents that template expressions use the `${{ }}` delimiters and that the `parameters` context is available during template parsing. Microsoft also documents that expression contexts can be accessed using either property dereference syntax or index syntax, which makes both methods valid for the same parameter when the parameter name is known. For more details, see Template expressions and Azure Pipelines expressions.

QUESTION NO: 38

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. comment

C. title

D. description

E. label

ANSWER: C D

Explanation:

When Azure Boards is integrated with GitHub, the integration recognizes work item references that use the AB# syntax in supported GitHub pull request fields. For a GitHub pull request, adding the work item reference to the pull request title or to the pull request description automatically creates a link between the pull request and the matching Azure Boards work item. This is useful because the Azure Boards work item then shows the associated GitHub pull request, helping teams trace code changes back to planned work, bugs, or user stories. The title is commonly used when the pull request is focused on one work item, while the description is useful when you want to include the work item reference along with other implementation details, testing notes, or context. Microsoft documents this integration behavior for linking GitHub commits and pull requests to Azure Boards work items by using the AB#ID mention syntax. See Link GitHub commits, pull requests, and issues to work items and Connect Azure Boards to GitHub.

QUESTION NO: 39

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: C D

Explanation:

Creating an Azure Service Health alert is correct because planned maintenance notifications for Azure platform services are delivered through Azure Service Health. Service Health lets you scope alerts to the relevant subscription, services, regions, and event types, including planned maintenance, so you can be notified before Microsoft performs maintenance that could affect your resources. To deliver the notification as an SMS message, you also need to create and configure an action group. In Azure Monitor, an action group defines who is notified and how they are notified when an alert fires. The Email/SMS message/Push/Voice notification type can include an SMS receiver with a country code and phone number. Together, the Service Health alert detects the planned maintenance event, and the action group sends the SMS notification to the intended recipient. Microsoft documents Service Health alerts as the recommended way to receive notifications for service issues, planned maintenance, and health advisories, and action groups as the mechanism used by Azure Monitor alerts to send SMS, email, voice, push, webhook, or automation actions. See Create Service Health alerts in the Azure portal and Azure Monitor action groups.

QUESTION NO: 40

You have an Azure subscription. The subscription contains virtual machines that run either Windows Server or Linux.

You plan to use Prometheus to monitor performance metrics.

You need to integrate Prometheus and Azure Monitor.

Which two actions should you perform? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Install a Prometheus server on a Windows virtual machine in Azure.

B. On each virtual machine, expose the metrics endpoint.

C. On each virtual machine, enable the Azure Diagnostics extension.

D. On each virtual machine, enable the containerized agent for Azure Monitor.

E. Expose a virtual network service endpoint for Azure Storage.

F. Install a Prometheus server on a Linux virtual machine in Azure.

ANSWER: B F

Explanation:

The correct actions are to install a Prometheus server on a Linux virtual machine in Azure and to expose the metrics endpoint on each virtual machine. Prometheus uses a pull-based collection model: the Prometheus server periodically scrapes HTTP endpoints that publish metrics, commonly at a `/metrics` endpoint. For Azure virtual machines, this typically means running an appropriate exporter or instrumented application on each Windows Server or Linux VM so that Prometheus can collect host or application performance metrics. Hosting the Prometheus server on a Linux VM in Azure provides the scrape engine and configuration point for collecting those metrics from the targets. After Prometheus is collecting the data, it can be integrated with Azure Monitor managed service for Prometheus, including supported ingestion and monitoring workflows in Azure Monitor. Microsoft describes Azure Monitor managed service for Prometheus as the Azure Monitor capability for collecting and analyzing Prometheus metrics; see Azure Monitor managed service for Prometheus. Prometheus also documents that metrics are collected from instrumented targets and exporters; see Prometheus exporters and integrations.

QUESTION NO: 41

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:

.p12 is correct because App Center requires an Apple signing certificate in PKCS#12 format when signing an iOS app build for distribution. For iOS apps, the installable package must be signed with a valid Apple certificate and a provisioning profile that includes the target devices or distribution entitlement. In App Center Build, code signing for iOS uses two uploaded assets: the provisioning profile, typically a .mobileprovision file, and the signing certificate, which is provided as a .p12 file along with its password. The .p12 file contains the private key and certificate needed to sign the app package, allowing App Center to produce a properly signed .ipa that can be distributed to the private distribution group's registered devices. Microsoft's App Center documentation for iOS code signing states that a valid certificate in .p12 format is uploaded together with the provisioning profile for signing builds. See iOS code signing in App Center and Build Xamarin.iOS apps in App Center.

QUESTION NO: 42

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: A

Explanation:

Yes is correct because Octopus Deploy is a supported release-management tool that can integrate with Azure DevOps pipelines and deploy packages created by a build to on-premises targets. In this design, Azure DevOps can publish or pass the build artifacts to Octopus by using Octopus tasks in the pipeline, and Octopus then performs the deployment to the target machines. A Tentacle is the Octopus agent installed on deployment targets, and a polling Tentacle is specifically useful for machines in private networks because the Tentacle initiates outbound communication to the Octopus Server rather than requiring inbound connectivity from the server to the on-premises machine. That makes the proposed solution suitable for deploying Azure DevOps build artifacts to on-premises servers, especially when firewall or network restrictions exist. Microsoft documents that Azure Pipelines can integrate with extensions and tasks from the Marketplace, and Octopus documents the Azure DevOps integration and Tentacle communication modes for deployment targets. References: Install extensions for Azure DevOps and Octopus Tentacle deployment targets.

QUESTION NO: 43

Your company creates a new Azure DevOps team.

D18912E1457D5D1DDCBD40AB3BF70D5D

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:

Kanban boards is the correct Azure DevOps component for visualizing the flow of work. In Azure Boards, a Kanban board provides an interactive, visual representation of work items as they move through workflow states, such as new, active, resolved, and closed, or through customized team-specific columns. This directly supports the agile Kanban practice of visualizing work, limiting work in progress, and improving flow across the team's process.

For sprint planning, teams often use backlogs and sprint boards, but when the requirement is specifically to visualize the flow of work, the Kanban board is the component designed for that purpose. Microsoft describes Kanban boards in Azure Boards as tools that turn a backlog into an interactive signboard and help teams track work as it progresses through defined stages. Teams can customize columns, swimlanes, work-in-progress limits, and card styles to match their process and make workflow bottlenecks visible. See Microsoft documentation on Kanban board basics and customizing Kanban boards.

QUESTION NO: 44 - (DRAG DROP)

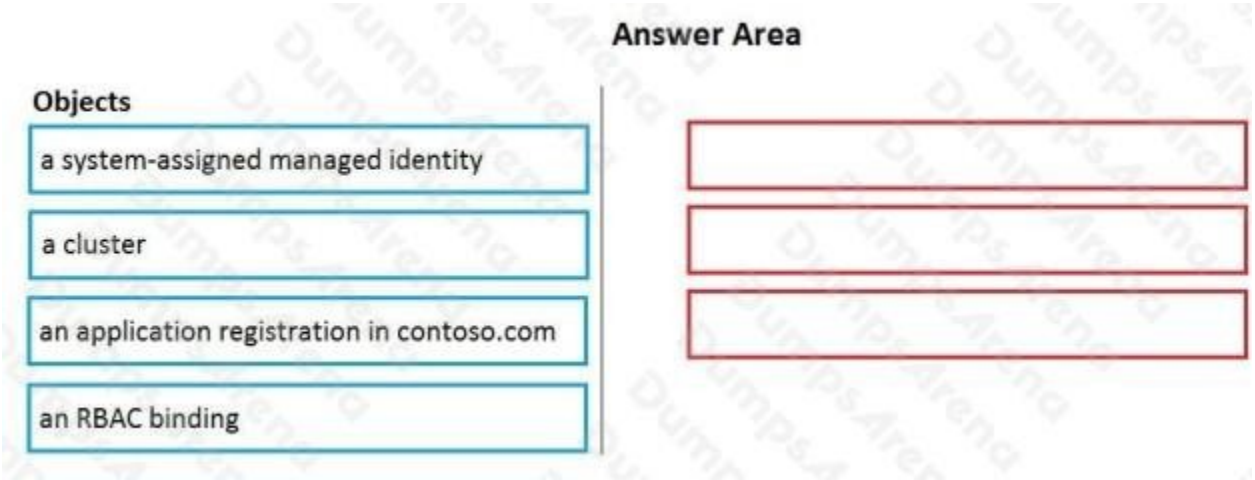
DRAG DROP

Your company has an Azure subscription named Subscription1. Subscription1 is associated to an Azure Active Directory tenant named contoso.com.

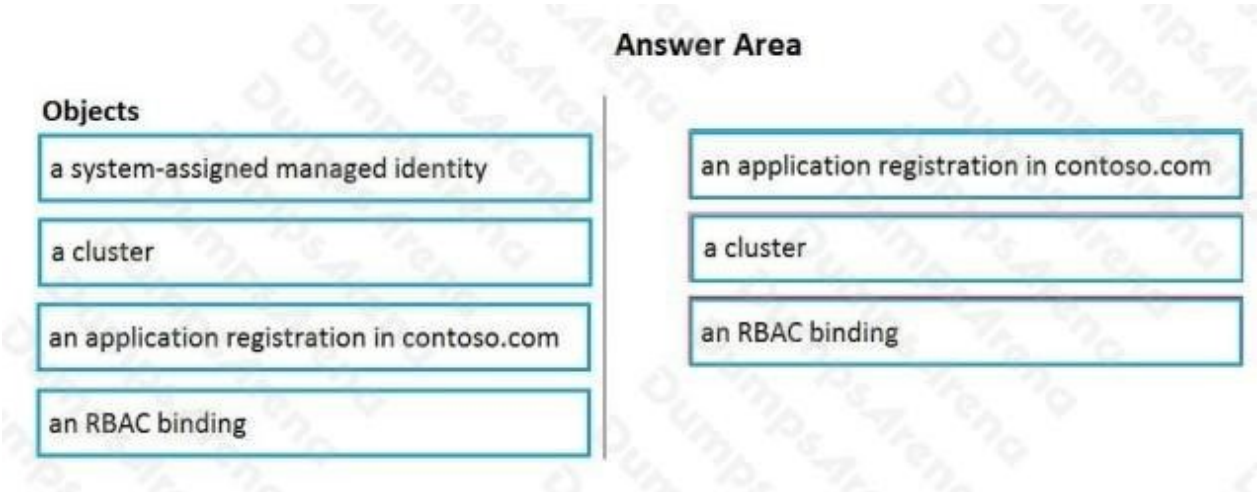
You need to provision an Azure Kubernetes Services (AKS) cluster in Subscription1 and set the permissions for the cluster by using RBAC roles that reference the identities in contoso.com.

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:

The correct sequence is to create an application registration in contoso.com, then create the AKS cluster, and then create the RBAC binding. The key requirement in the question is that the cluster permissions must be assigned by using RBAC roles that reference identities in the Azure Active Directory tenant, contoso.com. For Kubernetes RBAC to work with Azure AD identities, the cluster must be configured for Azure AD authentication, which depends on Azure AD objects being available before the cluster is provisioned. That is why the application registration in the tenant comes first.

After the Azure AD application registration exists, the AKS cluster can be created with Azure AD integration and Kubernetes RBAC enabled. This allows users and groups from the Azure AD tenant to authenticate to the Kubernetes API server. Once the cluster exists and can authenticate those identities, Kubernetes authorization can be configured by creating RoleBinding or ClusterRoleBinding objects. These bindings connect Kubernetes roles, such as permissions to read pods or manage deployments, to Azure AD users or groups from contoso.com.

The RBAC binding must be created last because it is a Kubernetes object that is applied to the running cluster. It references identities that are already known through the Azure AD-integrated authentication flow. Microsoft's AKS documentation explains how Azure AD integration enables tenant identities to authenticate to AKS, and Kubernetes documentation describes how RoleBinding and ClusterRoleBinding objects assign permissions inside the cluster. See AKS identity and access concepts and Kubernetes RBAC authorization for the underlying behavior.

QUESTION NO: 45

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:

No is correct because the stated solution does not fully address the required deployment scope. The scenario requires the release pipeline to create two resource groups and then deploy different Azure resources into each resource group. Resource groups are subscription-scope resources, so a template that is effectively centered on deploying resources into one resource group is not the appropriate boundary for creating both resource groups and orchestrating deployments into both targets. The reliable ARM pattern is to use a subscription-scope deployment to create the resource groups, and then use child deployments, modules, or separate ARM deployment tasks scoped to each resource group for the virtual machines and SQL databases. Microsoft documents that resource groups are created by subscription-level deployments, and that deployments can then be scoped to target resource groups as needed. See Create resource groups and resources at the subscription level and Resource group deployments with ARM templates.

QUESTION NO: 46

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: C E

Explanation:

Azure Security Center, now Microsoft Defender for Cloud, provides vulnerability assessment capabilities for Azure virtual machines and container images in Azure Container Registry. The Azure virtual machines that run the latest version of Ubuntu are correct because Defender for Cloud can assess machines for known vulnerabilities by using Microsoft Defender Vulnerability Management or an integrated scanner, and Linux virtual machines such as Ubuntu are supported targets. This capability produces security recommendations and vulnerability findings that can be tracked and remediated from Defender for Cloud. See Microsoft's guidance for vulnerability assessment for machines.

The Windows Server 2019 container images hosted in the Azure Container Registry are also correct because Defender for Cloud can scan container images stored in Azure Container Registry and report vulnerabilities found in the image layers and packages. This is part of the container security capabilities that help identify vulnerable images before or after they are deployed. Microsoft documents this under Defender for Containers vulnerability assessment for Azure container registries: Defender for Containers vulnerability assessment.

QUESTION NO: 47 - (HOTSPOT)

HOTSPOT

You manage the Git repository for a large enterprise application.

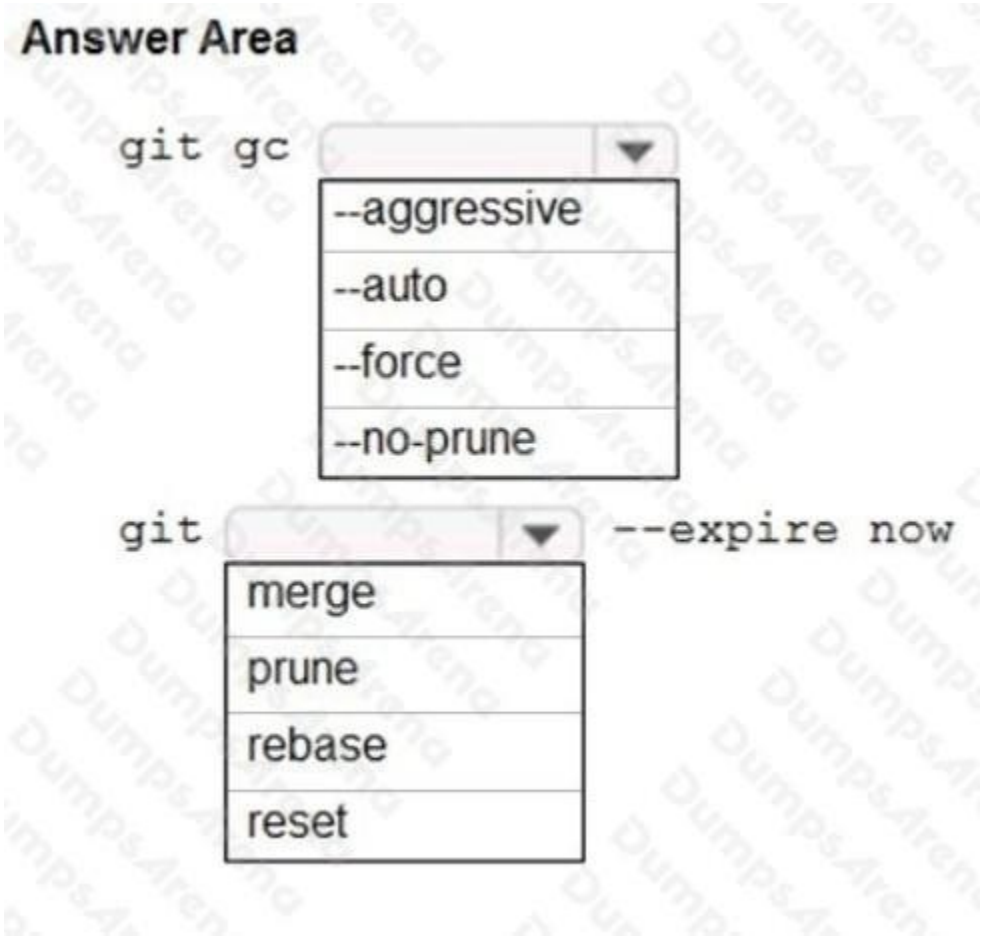
You need to minimize the data size of the repository.

How should you complete the commands? To answer, select the appropriate options in the answer area.

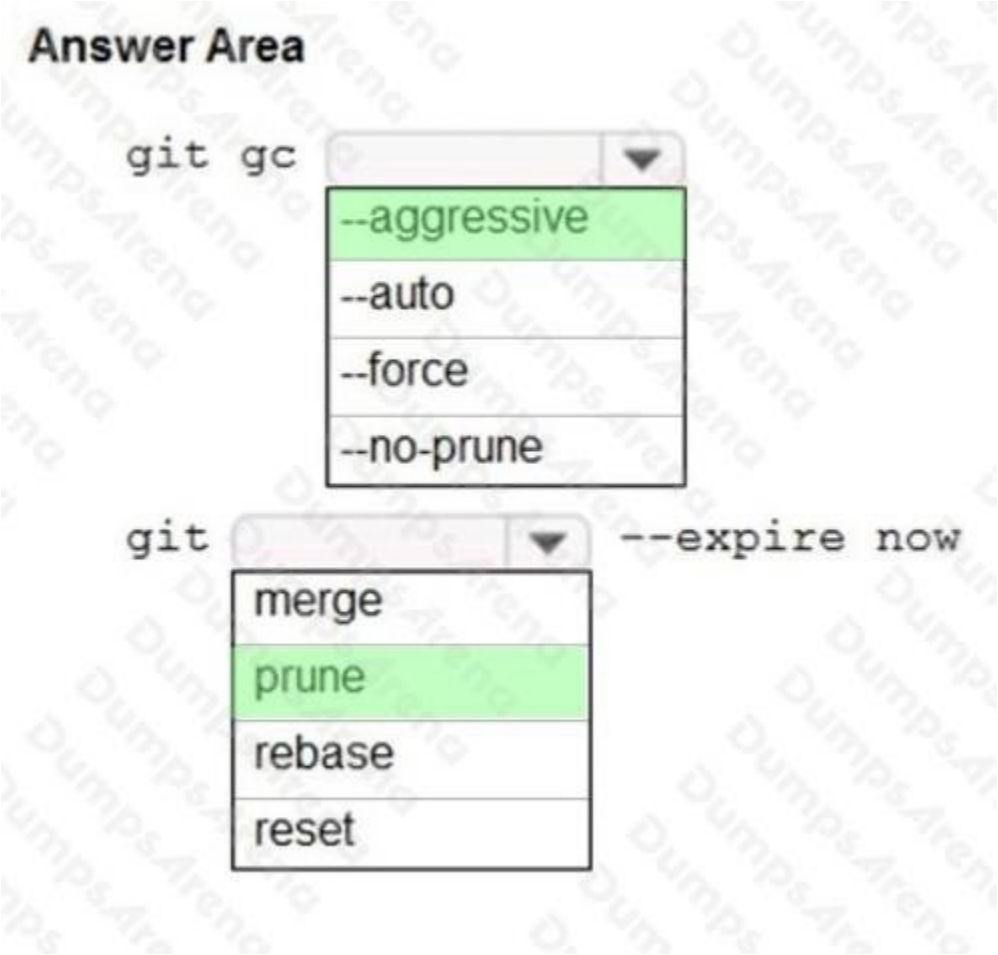
NOTE: Each correct selection is worth one point.

Hot Area:

Answer Area



ANSWER:



Explanation:

The correct completed commands are `git gc --aggressive` and `git prune --expire now`. In Git, `git gc` runs garbage collection for the repository. This cleans up unnecessary files, consolidates loose objects, and repacks objects to make repository storage more efficient. Adding `--aggressive` tells Git to spend extra time optimizing the repository more thoroughly, especially by recomputing object deltas during repacking. That makes it the appropriate choice when the goal is to minimize repository data size rather than simply performing a quick maintenance pass. Microsoft Azure DevOps repositories are Git repositories, so standard Git maintenance behavior applies when managing repository storage locally or in Git-based workflows.

The second completed command, `git prune --expire now`, removes unreachable loose objects from the object database immediately. By default, Git often keeps unreachable objects for a grace period to protect against accidental data loss. Using `--expire now` shortens that retention window to the current time, so objects that are already unreachable can be pruned right away. Together, aggressive garbage collection and immediate pruning are a common way to reduce the size of a Git repository after large files, abandoned commits, or other unreachable objects have been removed from history. The official Git documentation describes these behaviors for git gc and git prune.

QUESTION NO: 48

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. Participating

B. Automatically watch repositories

C. Automatically watch teams

D. Watching

ANSWER: C D

Explanation:

To receive email notifications only for discussions where you have commented or where you are mentioned, you should clear **Automatically watch teams** and **Watching**. In GitHub notification settings, automatically watching teams can subscribe you to team discussion activity, which causes notifications for discussions even when you have not participated. Clearing that setting prevents GitHub from automatically subscribing you to team discussions in that broad way. The **Watching** email notification setting controls email delivery for activity in repositories or teams you are watching. Clearing it stops email for general watched activity, while still allowing email for participating conversations and mentions when those notification types remain enabled. This aligns with GitHub's notification model: "participating" notifications are intended for conversations involving you, such as when you are mentioned or have taken part in the thread, while "watching" is broader and can include all activity for watched repositories or teams. For more details, see GitHub's documentation on configuring notifications and managing subscriptions for activity on GitHub.

QUESTION NO: 49

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:

No is correct because the stated goal is to reduce the repeated time spent installing npm dependencies during pipeline runs. In Azure Pipelines, the feature intended for that scenario is pipeline caching, typically using the Cache task to restore and save the npm cache or dependency-related files based on a key such as the operating system and package-lock.json. When the cache key matches on later runs, npm can reuse previously downloaded package data, which can significantly reduce installation time for a React.js build.

Pipeline artifacts serve a different purpose: they publish and download files produced by a pipeline, such as build outputs, logs, test results, or deployment packages, so those files can be shared across jobs, stages, or pipeline runs. They are not the recommended mechanism for accelerating package restore or dependency installation. Microsoft's guidance for improving dependency restore performance is to use pipeline caching, including examples for npm-based projects. See Cache dependencies and build outputs in Azure Pipelines and Pipeline artifacts in Azure Pipelines.

QUESTION NO: 50

You manage projects by using Azure Boards.

You have a current work item name item A that is dependent on a work item named item3.

You need to define the dependency for item A.

What should you do in the web portal for Azure DevOps?

A. From Backlogs, open the context menu, select Add link and then select item3. Set Link type to Related and add the ID of item A

B. From item A, open the Links tab, and then select Add link. Set Link type to Successor and add the ID of item B.

C. 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 item B.

D. From item A, open the Links tab, and then select Add link. Set Link type to References and add the ID Of item B.

E. From item A, open the Links tab, and then select Add link. Set Link type to Predecessor and add the ID of item3.

ANSWER: E

Explanation:

From item A, open the Links tab, and then select Add link. Set Link type to Predecessor and add the ID of item3 is correct. In Azure Boards, dependency relationships between work items are represented by the Predecessor/Successor link types. When the current work item depends on another work item, the other work item is the predecessor because it should be completed before the current work item can proceed. Since item A is dependent on item3, item3 must be linked as the Predecessor of item A. Creating the link from item A on the Links tab with the Predecessor link type and the ID of item3 correctly records that dependency in the work item tracking system. Microsoft documents the Predecessor/Successor link types as the standard way to track dependency relationships between work items, and work item links can be added directly from the work item form in Azure Boards. See Microsoft's link type reference and Microsoft's guidance for adding links to work items.

QUESTION NO: 51

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:

No is correct because the proposed integration strategy does not satisfy the requirement for the build to access an on-premises dependency management system. A Microsoft-hosted Ubuntu agent runs on infrastructure managed by Microsoft in Azure, and it is created fresh for pipeline jobs. It does not automatically have network connectivity to private on-premises services, such as an internal Maven, Gradle, Nexus, or Artifactory dependency repository. The Java Tool Installer task can help ensure that the required JDK version is available during the build, but it does not provide private network access to on-premises resources.

For this scenario, the appropriate approach is typically to use a self-hosted Azure Pipelines agent located on the corporate network, or in a network that has secure connectivity to the on-premises dependency management system. Microsoft documentation notes that self-hosted agents are useful when you need more control over tools, dependencies, or network access. See Azure Pipelines agents and Microsoft-hosted agents for the relevant guidance.

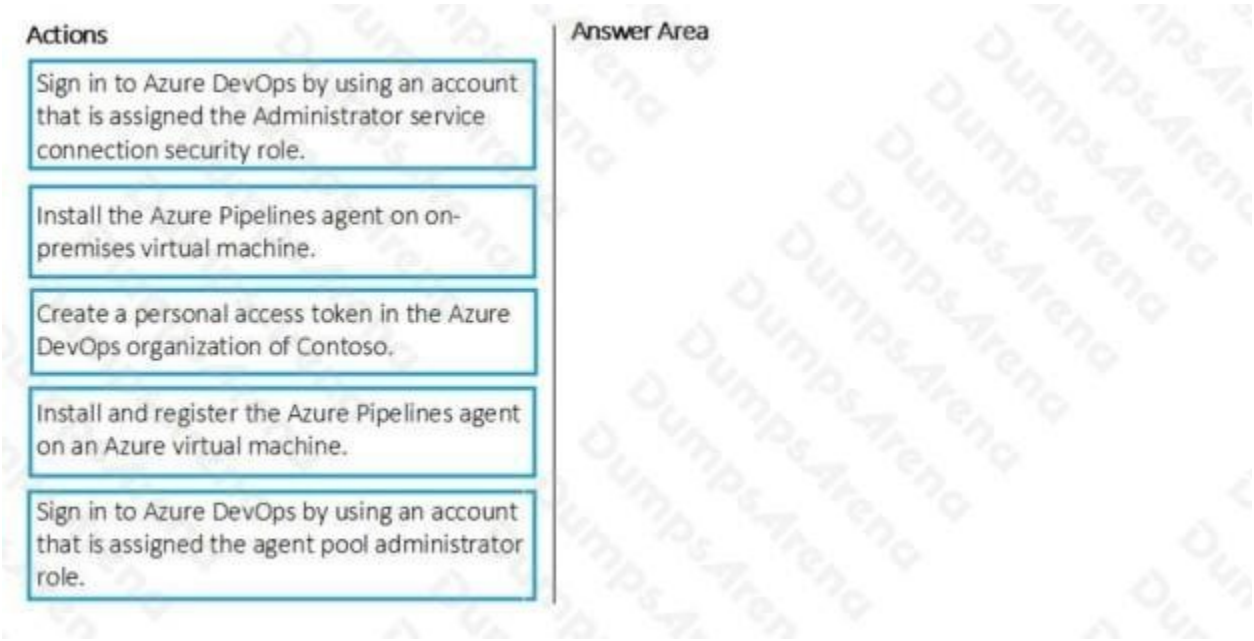
QUESTION NO: 52 - (DRAG DROP)

DRAG DROP

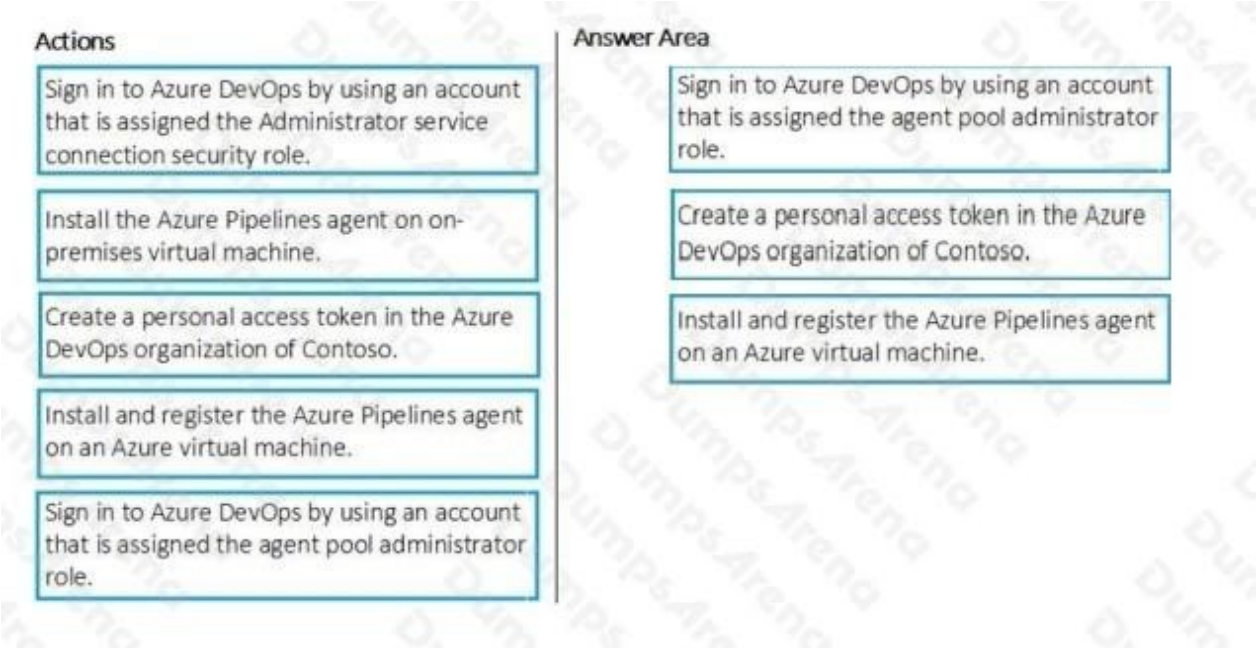
You need to recommend a procedure to implement the build agent for Project1.

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:

To implement a self-hosted Azure Pipelines build agent, the setup must start from the Azure DevOps agent pool permissions model. The account used for the setup needs to be able to administer the agent pool because the agent must be registered into a pool before pipelines can target it. Microsoft documents agent pools as the security boundary for self-hosted agents, and agent registration requires appropriate agent pool permissions such as manage permissions or administrator-level control over the pool. See Microsoft's guidance on Azure Pipelines agents and Azure DevOps permissions.

After signing in with an account assigned the agent pool administrator role, a personal access token should be created in the Azure DevOps organization. The PAT is used during the agent configuration process so the agent machine can authenticate to Azure DevOps and join the intended organization, project, and agent pool. Microsoft's agent installation steps specifically include creating or using credentials during configuration so the agent can register itself securely. The relevant Microsoft setup process is described in Self-hosted Windows agents.

Finally, the Azure Pipelines agent is installed and registered on the Azure virtual machine. This step downloads the agent package, runs the configuration script, supplies the organization URL and authentication token, and registers the machine as an available build agent. Once registration is complete, Project1 can run pipeline jobs on that private agent instead of relying on Microsoft-hosted agents. The key point is that service connection administration is not the permission path for registering build agents; agent pool administration is.

QUESTION NO: 53

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 an agent pool.

C. Create a deployment group.

D. Reduce the size of the repository.

E. Increase the number of parallel jobs.

ANSWER: B E

Explanation:

Create an agent pool and Increase the number of parallel jobs are correct because Azure Pipelines can run separate jobs concurrently only when sufficient agents and parallel job capacity are available. In Azure Pipelines, an agent runs one job at a time. If the pipeline already separates compilation into different jobs for 10 architectures, placing enough agents in an agent pool allows those jobs to be distributed across multiple agents instead of waiting for a single agent to become available. This is especially effective for long-running builds where each architecture can be compiled independently.

Increasing the number of parallel jobs is the other required part of the solution. Parallel jobs define how many jobs can run at the same time in an Azure DevOps organization or project, depending on the type of agents and purchased capacity. Without sufficient parallel job entitlement, additional agents will not reduce total runtime because jobs will still be queued. Microsoft documents that to run more jobs in parallel, you can use self-hosted agents and/or purchase additional parallel jobs. For more details, see Azure Pipelines agents and Configure and pay for parallel jobs.

QUESTION NO: 54 - (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: See the explanation for the answer

Explanation:

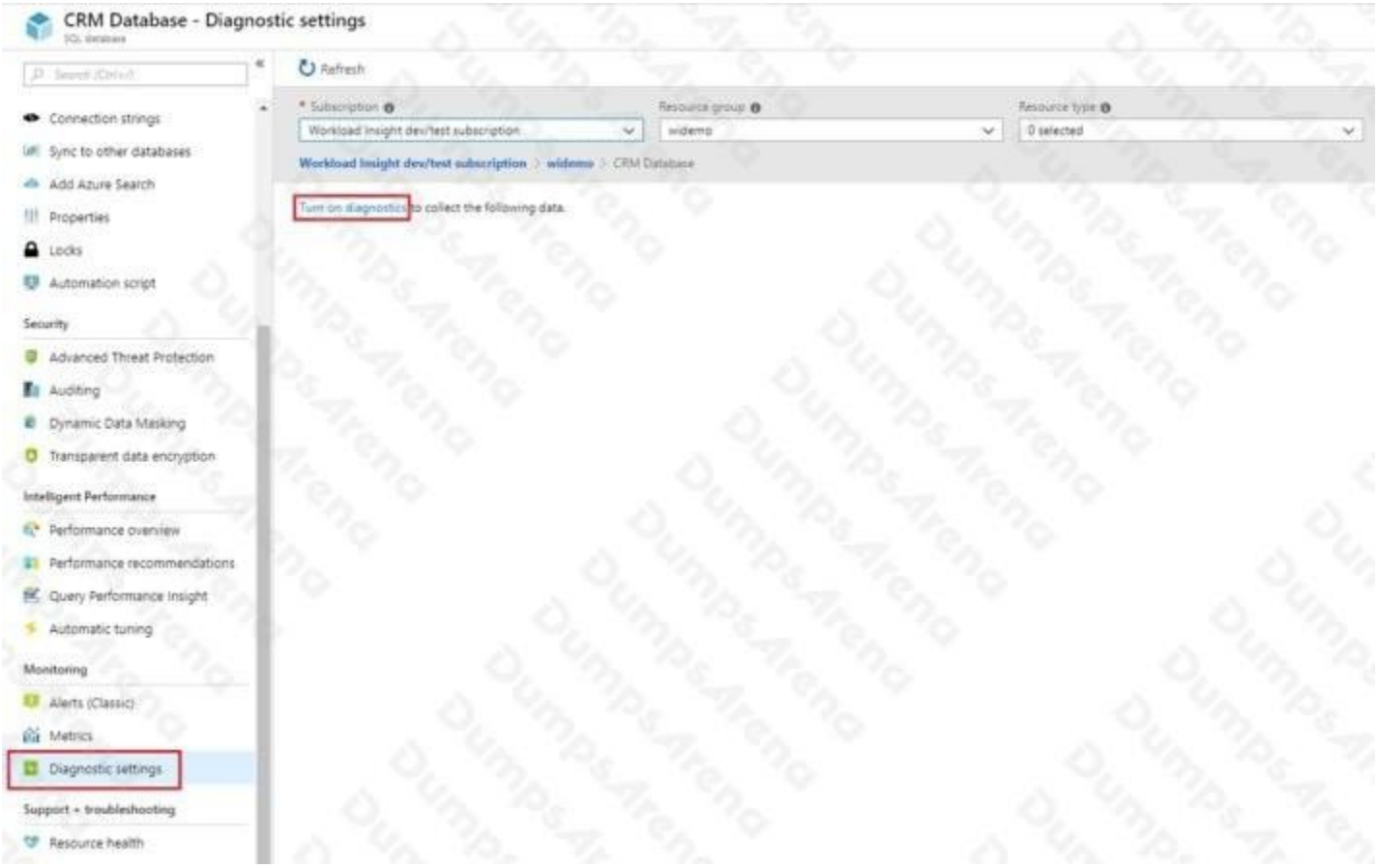
To enable streaming of diagnostic telemetry for a single or a pooled database, follow these steps:

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.

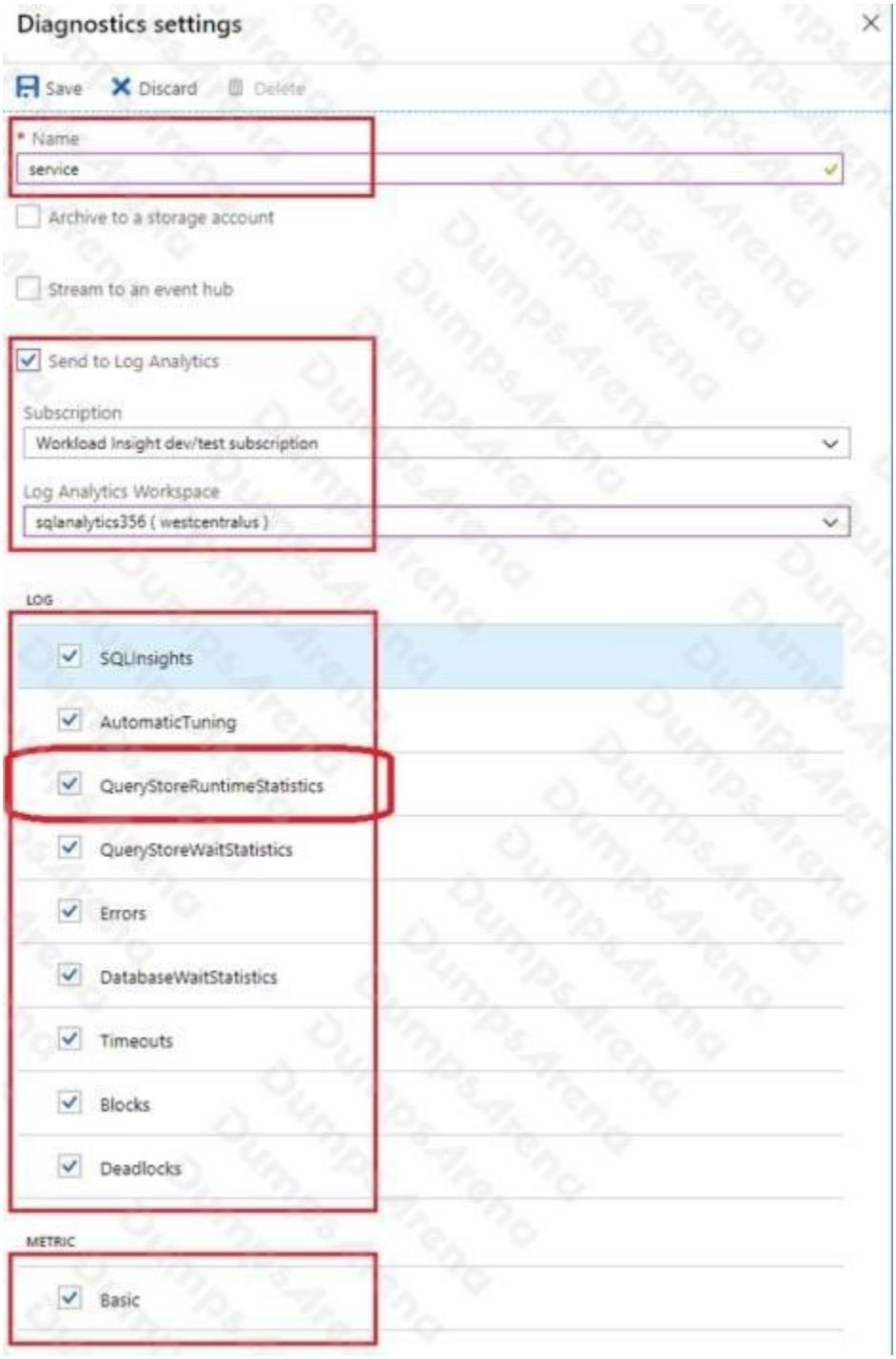
4. Select Add diagnostic setting to configure parallel streaming of diagnostics data to multiple resources.



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



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 NO: 55

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:

conditional access policies in Azure AD is correct because Azure DevOps organizations can be connected to Microsoft Entra ID, and user sign-in to Azure DevOps can then be governed by Entra Conditional Access. Conditional Access policies are designed to make access decisions based on signals such as user, group, device state, application, and network location. If the technical requirement is to restrict access to Azure DevOps to specific IP addresses, you can define named locations that represent allowed or blocked IP ranges and then apply a Conditional Access policy to Azure DevOps access. This provides centralized identity-based access control at the authentication layer, which is the appropriate control point for cloud services such as Azure DevOps Services.

Microsoft documents that Conditional Access can be used with Azure DevOps when the organization is backed by Microsoft Entra ID, and that policies can enforce controls for Azure DevOps sign-ins. Microsoft also documents network location conditions, including IP address ranges and named locations, as supported signals for Conditional Access decisions. For more information, see Conditional Access policies for Azure DevOps and Conditional Access network assignment.

QUESTION NO: 56

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: A C

Explanation:

Full Microsoft .NET Framework applications are supported in Windows container images, so the build agent must be a Windows-based Microsoft-hosted image capable of working with Windows containers. The images **vs2017-win2016** and **win1803** are valid choices in this context because they are Windows Server-based Microsoft-hosted agent images. **win1803** was specifically provided for Windows container scenarios, while **vs2017-win2016** provides a Windows Server 2016 environment with Visual Studio tooling and can be used for building Windows-based workloads such as full .NET Framework applications in Azure Pipelines. In YAML pipelines, the image is selected by setting the Microsoft-hosted pool VM image, for example by using the `vmImage` property. Because .NET Framework container images rely on Windows container support rather than Linux container support, selecting a Windows Server agent image is the key requirement for this scenario. Microsoft documents the available Microsoft-hosted agent images and how to specify them in YAML pipelines in the Azure Pipelines hosted agents documentation: Microsoft-hosted agents. Microsoft also documents Windows container image support for .NET Framework here: Lift and shift to containers.

QUESTION NO: 57

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:

No is correct because pre-deployment gates are evaluated in the release/deployment workflow, typically before a deployment stage is allowed to proceed. They are useful for checking external conditions such as monitoring alerts, work item states, or custom REST API responses before deployment, but they do not inherently inspect source code dependencies for open-source license violations or prohibited libraries while developers are building and integrating code. The requirement is to identify these issues as they occur during the company's development process, which means the checks should be integrated into continuous integration or dependency scanning in the build pipeline. Tools such as Mend/WhiteSource-style software composition analysis can scan project dependencies during builds and report license compliance and policy violations early, before artifacts reach a pre-deployment approval point. Microsoft describes gates as release controls that run before or after deployment stages, not as continuous dependency compliance scanners. See Approvals and gates overview and the Azure DevOps Labs guidance for WhiteSource Bolt integration.

QUESTION NO: 58

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 Dependency agent

B. the Azure Network Watcher Agent for Windows

C. the Telegraf agent

D. the Azure Log Analytics agent

ANSWER: A D

Explanation:

the Azure Log Analytics agent and the Dependency agent are the required pair for collecting detailed guest-level process information for Windows Server virtual machines in the classic Azure Monitor VM insights model. The Azure Log Analytics agent, also known as the Microsoft Monitoring Agent, collects performance counters, event logs, and other monitoring data from the guest operating system and sends that data to a Log Analytics workspace. The Dependency agent works alongside the Log Analytics agent to discover running processes and map process-to-process communication and external dependencies. This combination enables Azure Monitor for VMs/VM insights to provide richer visibility than platform metrics

alone, including process details and dependency maps inside the guest OS. Microsoft documentation describes that VM insights dependency map data is collected by the Dependency agent and that the Dependency agent relies on the Log Analytics agent for delivery to Azure Monitor Logs. See Microsoft's agent overview and VM insights dependency documentation for more detail: Azure Monitor agents overview and Dependency agent for VM insights.

QUESTION NO: 59

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: A B D

Explanation:

GitHub Actions identifies the version of a reusable action in the workflow step by appending a reference after the action name, such as `actions/checkout@v4`. The valid ways to pin or request a version include the SHA-based hashes, the tag, and the branch. A commit SHA is the most precise and stable reference because it points to one exact commit, which makes it the strongest choice for repeatability and supply-chain security. A tag is commonly used for released versions, such as a major version tag or semantic version tag, and is the standard way many marketplace actions publish supported releases. A branch can also be used as a Git ref, for example referencing the default branch or a release branch, although it may move over time as new commits are added. Microsoft and GitHub documentation describe this as specifying a Git ref, SHA, or Docker tag when selecting an action version, and Git refs include branches and tags. See the GitHub workflow syntax documentation for jobs.<job_id>.steps[*].uses and GitHub guidance on using third-party actions securely.