AZ-400 Renewal Questions_2025

1. **You are developing a process for blameless retrospectives. What are three pieces of information that you should gather as part of a blameless retrospective process?**

- A. Assumptions made by the team
- B. Effects that were observed
- C. Justification for user actions
- D. Timeline of events that occurred

Explanation: The three most important pieces of information to gather during a blameless retrospective are:

- A. Assumptions made by the team:
  - Explanation: Understanding the underlying assumptions that guided the team's decisions is crucial. These assumptions, whether explicit or implicit, can reveal blind spots, flawed logic, or incorrect interpretations of information. By identifying and examining these assumptions, the team can uncover the root causes of issues and prevent similar problems in the future.
- B. Effects that were observed:
  - Explanation: It's essential to document the observable consequences of the event or issue being reviewed. This includes both positive and negative effects, both intended and unintended. Analyzing the effects helps the team understand the full impact of their actions and identify areas for improvement.
- D. Timeline of events that occurred:
  - Explanation: A clear and accurate timeline provides context and helps the team understand the sequence of events leading up to the issue. This chronology can reveal dependencies, delays, and unexpected occurrences that contributed to the problem.

Why C. Justification for user actions is not as critical:

- Focus on systemic issues: Blameless retrospectives should focus on systemic issues and process improvements, not individual actions. While understanding the rationale behind individual decisions can be helpful, it's more important to understand the broader context and identify underlying factors that contributed to the issue.

2. **You have a globally distributed Azure web app named WebApp1 that is configured for Azure Application Insights. You receive a Smart Detection-generated alert regarding an issue with WebApp1 that results in an exception every time users in Singapore open a specific page. You need to verify whether the issue is limited to specific regions. What should you do?**

- A. Configure availability tests

- B. Configure user flows

- C. Modify the Smart Detection rules

- D. Update the Application map components

Explanation: Availability tests in Azure monitor allow you to proactively check the availability and performance of your web applications from various locations around the world.

Why other options are less suitable:

- **B. Configure user flows:** User flows are primarily used to analyze user behavior within a web application. While they can provide insights into user interactions, they don't directly help in verifying regional performance issues.
- **C. Modify the Smart Detection rules:** Modifying the rules might alter the alert behavior but won't provide the necessary data to verify regional limitations.
- **D. Update the Application map components:** Application Map provides a visual representation of dependencies within your application. While helpful for understanding application flow, it doesn't directly address the need to verify regional performance.

3. You receive a smart detection notification from Application Insights for an app connected to an Azure SQL database. In the notification, you notice that exceptions are being thrown. What can you use to investigate the exceptions?

- A. .NET Reflector

- B. Fiddler

- C. Snapshot Debugger

- D. SQL Server Profiler

Explanation:

- **SQL Server Profiler** is a tool designed to monitor and analyze activity on an SQL Server instance.

Why other options are less suitable:

- **A. .NET Reflector:** Primarily used for .NET assembly analysis and reverse engineering. While helpful for understanding application code, it doesn't directly address SQL-related exceptions.
- **B. Fiddler:** A web debugging proxy that captures HTTP traffic. Useful for analyzing network-related issues, but less effective for investigating SQL-specific exceptions.
- **C. Snapshot Debugger:** A feature of Visual Studio that captures a snapshot of your application's state at the time of an exception. Helpful for debugging application code, but less direct for analyzing SQL-related issues.

4. You are integrating Azure Policy with your Azure DevOps CI/CD pipelines. A pipeline that deploys resources must ensure that the resources comply with applied policies. When a resource fails to comply with a policy, the deployment must stop. What should you add to each pipeline?

- A. A check gate
- B. A webhook
- C. A CodeQL test
- D. A Visual Studio test

Explanation:

- **Check gates** in Azure DevOps pipelines allow you to enforce specific conditions before a stage or deployment can proceed.

Why other options are less suitable:

- **B. A webhook:** Webhooks are used to trigger actions in external systems based on events in Azure DevOps. While they can be used to notify about policy violations, they don't directly prevent non-compliant deployments.
- **C. A CodeQL test:** CodeQL is a code analysis tool used to find security vulnerabilities in code. It's not directly applicable to enforcing Azure Policy compliance for deployed resources.
- **D. A Visual Studio test:** Visual Studio tests are used to validate the functionality of application code. They don't assess the compliance of deployed resources with Azure Policies.

5. You use pipelines to deploy Azure infrastructure. You plan to configure an Azure policy to ensure that automatic remediation occurs if an Azure SQL database is created without Transparent Data Encryption (TDE). Which policy definition effect should you use?

- A. auditIfNotExists

- B. deployIfNotExists

- C. deny

- D. modify

Explanation:

- **modify:** This policy effect allows you to automatically change existing resources to meet the policy requirements. In this case, the policy would automatically enable TDE on any SQL database created without it.

Why other options are less suitable:

- **A. auditIfNotExists:** This effect only audits the resource if it doesn't exist, which is not applicable in this scenario where the database already exists.
- **B. deployIfNotExists:** This effect deploys the resource if it doesn't exist. It's not relevant for ensuring TDE on existing databases.
- **C. deny:** This effect blocks the creation of the resource if it doesn't comply with the policy. This would prevent the initial creation of the database, but it doesn't address the need to remediate existing databases without TDE.

6. You are creating pipelines to build and deploy an app. You need to include dynamic penetration tests that run the application by using known attack patterns. What should you use?

- A. OWASP ZAP tests

- B. Regression tests

- C. Static code analysis

- D. Unit tests

Explanation:

- **OWASP ZAP (Zed Attack Proxy)** is an open-source web application security scanner.
- **Dynamic penetration tests:** These tests involve actively interacting with the running application to identify vulnerabilities.

Why other options are less suitable:

- **B. Regression tests:** These tests verify that existing functionality still works correctly after code changes. While important, they don't specifically focus on security vulnerabilities.
- **C. Static code analysis:** This technique analyzes source code without executing it. It can identify potential security issues, but it might miss runtime vulnerabilities that only appear during dynamic testing.
- **D. Unit tests:** These tests focus on individual units of code (e.g., functions, classes). They are crucial for code quality but have limited effectiveness in detecting security vulnerabilities at the application level.

7. You use pipelines to deploy Azure infrastructure. You plan to implement an Azure Policy solution to ensure compliance with regulatory requirements. The solution must meet the following requirements: Audit unencrypted Azure SQL databases. Monitor network security groups (NSGs) for specific ports. Monitor whether vulnerability assessment solutions are installed on virtual machines. Group all requirements for ease of administration. What should you include in the solution?

- A. Initiative definition

- B. Policy definition

- C. Policy remediation

- D. Policy scope

Explanation:

- **Azure Policy initiatives** are designed to group multiple related policy definitions together.

Why other options are less suitable:

- **B. Policy definition:** A single policy definition focuses on a specific rule or condition. It's not suitable for grouping multiple, diverse compliance requirements.
- **C. Policy remediation:** Remediation is a specific action taken to bring a non-compliant resource into compliance. It's not a mechanism for grouping policy definitions.
- **D. Policy scope:** The scope defines where a policy or initiative is applied (e.g., subscription, resource group). While important, it doesn't address the need to group multiple policy definitions.

8. You are reviewing GitHub actions that have been created in your organization. You need to resolve the following issues that you identified: The definitions of many actions are long and complex. Ongoing management of actions is difficult because the same code is repeated in many actions. What should you do to resolve the issues?

- A. Remove all secrets embedded in the action files

- B. Create chainable actions

- C. Separate variable definitions into separate files

- D. Separate variable group definitions into separate files

Explanation:

- **Chainable actions** promote reusability and modularity. By breaking down complex actions into smaller, more manageable units, you can:

Why other options are less suitable:

- **A. Remove all secrets embedded in the action files:** While a good security practice, this doesn't directly address the issues of long, complex definitions and code duplication.
- **C. Separate variable definitions into separate files:** This can improve readability but doesn't solve the core issue of code duplication across multiple actions.
- **D. Separate variable group definitions into separate files:** Similar to option C, this improves organization but doesn't address the fundamental problem of code redundancy.

9. You have a GitHub workflow that deploys an Azure web app. You need to configure the workflow to use a pull request that includes a label to trigger a deployment. The workflow includes the following section: 01 on: 02 pull_request: 03 What should you add at line 03 of the workflow?

- A. action: [labeled]

- B. if: [labeled]

- C. name: [labeled]

- D. types: [labeled]

Explanation:

- **on: pull_request** indicates that the workflow should be triggered by pull request events.
- **types** is the property within the pull_request event that specifies the types of pull request events that should trigger the workflow.

Other options are incorrect:

- **action: [labeled]:** This is not the correct syntax for specifying event types.

- **if: [labeled]:** The if condition is used to conditionally execute steps within the workflow, not to filter the types of events that trigger the workflow.
- **name: [labeled]:** The name property is used to give a name to the workflow, not to specify event types.

10. You plan to add a job to a GitHub workflow that will deploy a container-based web app to Azure Web Apps. You need to ensure that a previous job in the workflow completes successfully before the new job can run. Which GitHub workflow element should you add to the workflow?

- A. contains

- B. **needs**

- C. runs-on

- D. uses

Explanation:

- In a GitHub workflow, the needs keyword is used to define dependencies between jobs.
- By specifying the needs property for a job, you indicate that the job must wait for the successful completion of the listed jobs before it can start.

Why other options are less suitable:

- **A. contains:** This keyword is not used to define dependencies between jobs.
- **C. runs-on:** This property specifies the runner or machine type that will execute the job. It doesn't define dependencies.
- **D. uses:** This keyword is used to reference reusable actions within a workflow. It's not related to job dependencies.

11. You have a GitHub workflow that deploys an Azure web app. The workflow is configured to trigger a deployment following a pull request that includes a label. You plan to configure the pull request of the workflow to trigger the deployment only if the label is set to a string of "stage". The workflow includes the following section. if: contains(<missing element>.event.pull_request.labels.*.name, 'stage') What should you add for the missing element of the workflow?

- A. action

- B. github

- C. name

- D. workflow

Explanation:

- The if condition in GitHub Workflows allows you to conditionally execute steps or jobs based on certain criteria.
- To access information about the GitHub event that triggered the workflow, you use the github context.

Other options are incorrect:

- **action:** This refers to individual steps within a job, not the context of the GitHub event.
- **name:** This is used to name the workflow or a job within the workflow.
- **workflow:** This refers to the entire workflow definition.

12. Recent acquisitions have led to your organization to use multiple version control systems. You plan to standardize on Azure DevOps for version control. You need to migrate systems that are supported by Azure DevOps and replace the other systems. Which two types of version control systems can you migrate?

- A. CVS

- B. Git

- C. SVN

- D. TFVC

Azure DevOps supports the following two types of version control systems:

- **Git:** This is a distributed version control system that has become the industry standard. Git allows developers to work independently and then merge their changes back into the main codebase. Azure DevOps provides built-in support for Git repositories.
- **Team Foundation Version Control (TFVC):** This is a centralized version control system that was originally developed by Microsoft. TFVC is well-suited for teams that prefer a more centralized approach to version control.

13. You are developing a public open-source project by using GitHub. You need to ensure that only the project maintainer can push to the official repository. Which type of branching workflow should you implement?

- A. trunk-based

- B. GitHub flow

- C. feature branch

- D. forking

Explanation:

- **Forking Workflow:**
  - In this model, contributors create their own forks of the main repository.
  - They make changes in their forks and then submit pull requests to the main repository.
  - The project maintainer reviews the pull requests and merges them (or rejects them) into the main repository.
  - This workflow effectively restricts direct pushes to the main repository, ensuring that all changes are reviewed and approved by the maintainer before they are integrated.

Why other options are less suitable:

- **A. Trunk-based:** This workflow involves frequent integration of small changes directly into the main branch. It's generally not suitable for open-source projects with multiple contributors, as it increases the risk of breaking the main branch.
- **B. GitHub Flow:** This workflow also involves frequent integration into the main branch. While it's a popular choice for many projects, it's less suitable in this case where you need to strictly control direct pushes to the main repository.
- **C. Feature Branch:** This workflow involves creating branches for individual features. While it's a good practice for managing development, it doesn't inherently restrict direct pushes to the main repository.

14. You are building an app in an Azure DevOps repository and plan to deploy the app by using Azure Pipelines. Your company uses a project management system named PM1. PM1 is not directly integrated with Azure Pipelines but provides a webhook to receive notifications. When a successful build of the app occurs, you need to send a notification to PM1. What should you create first?

- A. an event

- B. a notification

- C. a service hook

- D. a subscription

Explanation:

- **Service hooks** in Azure DevOps allow you to integrate with external services by defining custom events and actions.

Why other options are less suitable:

- **A. an event:** Events are part of the service hook definition. You don't create events independently.
- **B. a notification:** The notification itself is the action performed by the service hook. You don't create notifications independently.
- **D. a subscription:** While the concept of subscribing to events is involved, "subscription" is not the specific term used in Azure DevOps for this integration mechanism.

15. You use a GitHub repository named Repo1 that is used as the source control when building an app. Notifications have been configured for several events. You need to customize, triage, and manage your updates. What should you use?

- A. Notifications inbox
- B. Organization settings
- C. Repo1 settings
- D. User account settings

Explanation:

- The **Notifications inbox** in GitHub is the central location for managing all the notifications you receive.

Why other options are less suitable:

- **B. Organization settings:** While organization settings allow you to configure general notification preferences for the entire organization, they don't provide the level of customization and management needed for individual user notifications.
- **C. Repo1 settings:** Repository settings allow you to configure notifications for specific events within that repository, but they don't provide the tools for managing received notifications directly.
- **D. User account settings:** User account settings allow you to configure general notification preferences for your GitHub account, but they don't provide the specific tools for managing and triaging notifications from the inbox.

16. You are configuring GitHub Packages to access a NuGet registry. Which authentication method should you use?

- A. Basic
- B. Certificate
- C. OAuth
- D. PAT

Explanation:

- **Personal Access Tokens (PATs)** are the most secure and recommended method for authenticating with GitHub APIs, including GitHub Packages.

Why other options are less suitable:

- **A. Basic Authentication:** Using usernames and passwords directly is generally not recommended due to security concerns. If the credentials are compromised, attackers can potentially access your repositories and packages.
- **B. Certificate:** While certificates can be used for authentication, they are typically more complex to manage and may not be necessary for most use cases involving GitHub Packages.
- **C. OAuth:** OAuth is primarily used for authorizing third-party applications to access user data on behalf of the user. While it can be used for some API interactions, it's not the standard or recommended method for accessing GitHub Packages.

17. You are configuring access permissions for a package in GitHub Packages. You need to ensure that a user named User1 can upload, download, and manage permissions for the package. The solution must follow the principle of least privilege. Which permission level should you assign to User1?

- A. admin permission
- B. both Read and Write permissions
- C. only Read permission
- D. only Write permission

Explanation:

- To fulfill all these requirements while adhering to the principle of least privilege, the most appropriate permission level is **admin**.

Why other options are less suitable:

- **B. both Read and Write permissions:** While this allows uploading and downloading, it does not provide the necessary authority to manage permissions for the package.
- **C. only Read permission:** This level restricts the user to only downloading the package, not uploading or managing permissions.
- **D. only Write permission:** This allows uploading but restricts downloading and permission management.

18. You need to configure GitHub Packages to access a NuGet registry. Which two actions should you perform?

- A. Add a PAT to the Username key in the file.

- B. Add GitHub Packages to packageSources.

- C. Create a .npmrc file.

- D. Create a nuget.config file.

Explanation:

1. **Add GitHub Packages to packageSources:**
   - You need to add the URL of your GitHub Packages feed to the packageSources section of your NuGet configuration. This tells your application where to find and download packages.
2. **Create a nuget.config file:**
   - The nuget.config file is where you store your NuGet configuration settings, including the list of package sources.
   - You can create this file in the root of your project or in a global location (e.g., %AppData%\Roaming\NuGet) to apply the settings to all your projects.

Why other options are less suitable:

- **A. Add a PAT to the Username key in the file:** You should not directly add your Personal Access Token (PAT) to the nuget.config file. This is considered a security risk. Instead, use environment variables or other secure methods to store and access your PAT.
- **C. Create a .npmrc file:** The .npmrc file is used for configuring npm (Node Package Manager), not NuGet.

19. You accidentally delete a package from GitHub Packages. What is the maximum number of days that you have to restore the package?

- A. 30

- B. 60

- C. 90

- D. 120

- **Restoration window:** After 30 days, the deleted package will be permanently removed and cannot be recovered.
- **Conditions:**
  - The same package namespace must be available (not used for a new package).
  - You must have the necessary permissions (admin for the package or repository).

20. You have a project in Azure DevOps named Project1. Project1 contains five self-hosted agents. Each self-hosted agent appears with a different list of capabilities in the Azure DevOps portal. You plan to create a YAML build pipeline named Pipeline1 in Project1. You need to ensure that Pipeline1 will run only on the self-hosted agents that include the npm capability. What should you do?

- A. In Pipeline1, include the depends on YAML schema element.

- B. In Pipeline1, include the demands YAML schema element.

- C. In Project1, configure a new service connection.

- D. In Project1, configure an execution plan of the agent job.

Explanation:

- In Azure DevOps YAML pipelines, the demands element is used to specify the capabilities that a job or step requires to run.
- By including the demands element in your Pipeline1 definition, you can ensure that the pipeline will only run on self-hosted agents that have the npm capability.

Why other options are less suitable:

- **A. depends on:** This element is used to define dependencies between jobs within a pipeline, not to specify agent requirements.
- **C. Configure a new service connection:** Service connections are primarily used to connect to external services (e.g., Azure subscriptions, Git repositories) and are not directly related to agent capabilities.
- **D. Configure an execution plan of the agent job:** Execution plans are used to control the scheduling and prioritization of agent jobs, not to filter agents based on capabilities.

21. You have a project in Azure DevOps named Project1. You deploy an Azure virtual machine named VM1 that runs Linux Ubuntu 18.04. You plan to add VM1 to Project1 as a self-hosted agent by running the build-agent.sh script from GitHub. Which two environment variables must you set for the script to complete successfully?

- A. AZP_AGENT_NAME
- B. AZP_AGENT_VERSION
- C. AZP_POOL
- D. AZP_TOKEN
- E. AZP_URL

Explanation:

- **AZP_TOKEN:** This variable stores an access token that grants the script the necessary permissions to register the agent with your Azure DevOps project. You can obtain this token using the Azure CLI command:

Bash

```
export AZP_TOKEN=$(az account get-access-token --query accessToken -o output)
```

This command retrieves an access token for your current Azure account. Make sure the account has the appropriate permissions (e.g., Agent Pools Administrator) to manage agent pools in Project1.

- **AZP_URL:** This variable specifies the URL of the Azure DevOps pool where you want to register the agent. You can construct this URL using the following steps:

1. **List Agent Pools:** Use the Azure CLI command:

Bash

```
az pipelines agent pool list --query "[?name=='Default'].pool.id" -o output
```

This command retrieves the ID of the default agent pool (Default) in Project1 (replace "Default" with the desired pool name if different).

2. **Construct AZP_URL:** Combine the pool ID with the base URL for the distributed task API:

Bash

```
export AZP_URL="https://dev.azure.com/<your-organization-name>/<your-project-name>/_apis/distributedtask/pools/$(az pipelines agent pool list --query "[?name=='Default'].pool.id" -o output)"
```

Replace <your-organization-name> and <your-project-name> with your actual Azure DevOps organization and project names.

22. You have a project in Azure DevOps named Project1. Project1 contains a self-hosted agent pool named Pool1. You deploy an Azure virtual machine named VM1 that runs Linux Ubuntu 18.04. You need to add VM1 to Pool1. Which authentication method should you use?

- A. a managed identity in Microsoft Azure Active Directory (Azure AD), part of Microsoft Entra
- B. a personal access token
- C. a service principal in Microsoft Azure Active Directory (Azure AD), part of Microsoft Entra
- D. SSH keys

Explanation:

- **Service Principal:** A service principal is an identity within Azure AD that represents an application or service.
- **Benefits:**
  - **Enhanced Security:** Service principals offer strong security by separating application identities from individual user accounts.
  - **Fine-grained Control:** You can grant the service principal specific permissions within Azure DevOps, limiting its access to only the necessary resources (e.g., register agents, access pipelines).
  - **Ease of Management:** Service principals are easily managed and rotated within Azure AD, improving security and reducing the risk of compromised credentials.
  - **Integration with Azure:** Seamlessly integrates with other Azure services and security features.

Why other options are less suitable:

- **A. Managed Identity:** While managed identities are useful for applications running within Azure, they are not the ideal choice for authenticating a self-hosted agent running on a separate virtual machine.
- **B. Personal Access Token (PAT):**
  - While PATs can be used, they are generally less secure than service principals.
  - Managing and rotating PATs can be more cumbersome.
- **D. SSH Keys:** SSH keys are primarily used for secure remote access to the VM itself. They do not directly authenticate the agent with Azure DevOps.

23. You plan to create a YAML Azure Pipelines build pipeline that will use Azure virtual machines from your Azure subscription as build agents. Which YAML schema elements must you include in the YAML build pipeline?

- A. environment
- B. pool name
- C. pool visage
- D. server

Explanation:

- pool: This element specifies which pool of agents to use for the job. You can define the pool name to use Azure virtual machines from your subscription.

The other options are not correct for this scenario:

- environment: This is used for deployment jobs to specify the target environment.
- pool visage: This is not a valid YAML schema element.
- server: This is not a valid YAML schema element for specifying build agents.

24. You need to add a job to a GitHub workflow that will deploy a container-based web app to Azure Web Apps. You need to ensure that a previous job in the workflow completes successfully before the new job can run. Which GitHub workflow element should you add to the workflow?

- A. contains
- B. needs
- C. runs-on
- D. uses

Explanation:

- needs: This element specifies that the current job depends on the successful completion of one or more previous jobs. By using needs, you can define dependencies between jobs, ensuring that the dependent job runs only after the specified jobs have completed successfully.

The other options are not correct for this scenario:

- contains: This is not a valid GitHub workflow element.
- runs-on: This specifies the type of runner to use for the job, but does not handle job dependencies.
- uses: This is used to reference an action or a reusable workflow, not to define job dependencies.

25. You have a project in Azure DevOps named Project1. Project1 contains five self-hosted agents. Each self-hosted agent appears with a different list of capabilities in the Azure DevOps portal. You plan to create a YAML build pipeline named Pipeline1 in Project1. You need to ensure that Pipeline1 will run only on the self-hosted agents that include the npm capability. What should you do?

- A. In Pipeline1, include the depends on YAML schema element.
- B. In Pipeline1, include the demands YAML schema element.
- C. In Project1, configure a new service connection.
- D. In Project1, configure an execution plan of the agent job.

Explanation:

- demands: This element allows you to specify that a job should run only on agents that have certain capabilities. By using demands, you can ensure that your pipeline runs only on agents that meet the specified criteria, such as having the npm capability.

The other options are not correct for this scenario:

- depends on: This element is used to specify dependencies between jobs, not to filter agents based on capabilities.
- service connection: This is used to define connections to external services, not to filter agents.
- execution plan of the agent job: This is not a valid configuration for filtering agents based on capabilities.