AZURE DevOps Interview Questions & Answers

1. Azure DevOps Basics and Overview

1. **What is Azure DevOps?**
    - **Answer**: Azure DevOps is a suite of development tools provided by Microsoft that enables developers to plan work, collaborate on code, and build, test, and deploy applications. Azure DevOps encompasses various services such as Azure Boards, Azure Repos, Azure Pipelines, Azure Test Plans, and Azure Artifacts.
2. **What are the components of Azure DevOps?**
    - **Answer**:
        - **Azure Boards**: Work tracking and management (tasks, stories, bugs).
        - **Azure Repos**: Source code management with Git and TFVC.
        - **Azure Pipelines**: CI/CD pipelines for building, testing, and deploying code.
        - **Azure Artifacts**: Package management for artifacts like Maven, npm, NuGet.
        - **Azure Test Plans**: Automated and manual testing tools.
3. **What are the main differences between Azure DevOps and GitHub?**
    - **Answer**: Azure DevOps focuses on providing enterprise-grade CI/CD pipelines, project tracking, and more integration with Microsoft services like Azure. GitHub, while also offering CI/CD (via GitHub Actions), is more focused on open-source collaboration. Azure DevOps is often used for more complex, corporate development environments, while GitHub is popular for individual projects and open-source development.
4. **What is the purpose of Azure DevOps Agent Pools?**
    - **Answer**: Agent pools in Azure DevOps are collections of agents that are used to run builds and deployments. These agents can be self-hosted (on your own machines) or provided by Microsoft (Microsoft-hosted agents). Agent pools allow you to distribute the load of running CI/CD pipelines across multiple agents.
5. **What is Azure DevOps Organization and Project?**
    - **Answer**: An **Azure DevOps Organization** is the top-level container that can host one or more **projects**. Projects are the specific containers within an organization where teams store code, track work, and manage CI/CD pipelines. A project can have its own repositories, boards, and pipelines.
6. **What is the difference between Microsoft-hosted and Self-hosted Agents in Azure DevOps?**
    - **Answer**: **Microsoft-hosted agents** are virtual machines provided by Azure that automatically scale to handle build and release tasks. **Self-hosted agents** are

machines managed by the user, providing more control over environment customization, resources, and software versions.

7. **How do you create a project in Azure DevOps?**
    - **Answer**: To create a project in Azure DevOps:
        - Navigate to your **Azure DevOps Organization**.
        - Click **New Project**.
        - Provide the project name, description, and visibility settings (private/public).
        - Set the version control system (Git or TFVC) and work item process (Agile, Scrum, CMMI).
        - Click **Create**.
8. **What is YAML, and how is it used in Azure Pipelines?**
    - **Answer**: YAML (YAML Ain't Markup Language) is a human-readable data serialization language. In Azure Pipelines, YAML is used to define pipelines as code. With YAML-based pipelines, developers can version control their pipeline configurations alongside their code and track changes over time.
9. **What is a Service Connection in Azure DevOps?**
    - **Answer**: A Service Connection in Azure DevOps is a link between your DevOps project and external services (e.g., Azure, AWS, Docker Hub). It allows your pipelines to securely access these services for tasks like deployments or container management.
10. **What are the benefits of using Azure DevOps?**
    - **Answer**: Benefits of using Azure DevOps include:
        - **Integrated tools** for the entire software development lifecycle (SDLC).
        - **Scalability** with cloud-hosted CI/CD pipelines.
        - **Version control** with Git and TFVC repositories.
        - **Work item tracking** for managing tasks, bugs, and features.
        - **Artifacts management** for storing dependencies.
        - **Test management** for automating and managing test cases.

---

2. Azure DevOps Repositories

11. **What is Azure Repos?**
    - **Answer**: Azure Repos is a service in Azure DevOps that provides Git repositories or Team Foundation Version Control (TFVC) repositories for managing source code. It supports both distributed (Git) and centralized (TFVC) version control systems.

12. **How do you clone a repository from Azure Repos?**
    - **Answer**:
        1. Navigate to your repository in Azure Repos.
        2. Click **Clone**.
        3. Copy the **HTTPS URL** or use the **SSH URL**.
        4. In your terminal, run:

```
git clone https://dev.azure.com/{organization}/{project}/_git/{repository}
```

13. **What is a pull request in Azure Repos?**
    - **Answer**: A pull request (PR) is a request to merge code from one branch into another (typically from a feature branch to the main branch). It allows developers to review, comment on, and approve code changes before merging them into the mainline.
14. **How do you create a pull request in Azure Repos?**
    - **Answer**:
        1. In Azure Repos, navigate to the **Pull Requests** tab.
        2. Click **New Pull Request**.
        3. Select the source branch (feature branch) and target branch (e.g., main).
        4. Add a title and description.
        5. Optionally, assign reviewers.
        6. Click **Create**.
15. **How do you enforce branch policies in Azure Repos?**
    - **Answer**: Branch policies enforce quality control for code changes in Azure Repos. To add branch policies:
        1. Navigate to **Repos** > **Branches**.
        2. Select the branch you want to add policies to.
        3. Click on **Branch Policies**.
        4. Enable policies such as:
            - **Require a minimum number of reviewers**.
            - **Check for linked work items**.
            - **Enforce merge strategies**.
            - **Build validation** before merging.
16. **What is a Git branch, and how is it used in Azure Repos?**

- **Answer**: A Git branch is a lightweight pointer to a commit in your repository, allowing you to work on different parts of your codebase independently. In Azure Repos, branches are used to isolate changes, create pull requests, and enable CI/CD workflows. Common branching strategies include **Git Flow**, **Feature Branching**, and **Trunk-Based Development**.

17. **How do you resolve a merge conflict in Azure Repos?**
    - **Answer**:
        1. Pull the latest changes from the remote repository.
        2. Open the conflicting files in your local branch.
        3. Manually resolve the conflicts by choosing which lines to keep.
        4. After resolving, mark the file as resolved using:

git add <file>

        5. Commit and push the changes.

18. **What are Git tags, and how do you create a tag in Azure Repos?**
    - **Answer**: Git tags are references to specific commits, often used to mark release points (e.g., v1.0). To create a tag in Azure Repos:
        1. In your terminal, run:

git tag v1.0

git push origin v1.0

19. **What is GitFlow branching strategy?**
    - **Answer**: GitFlow is a branching strategy that uses multiple long-running branches such as **main** (production), **develop** (integration), and temporary branches like **feature**, **release**, and **hotfix**. It ensures that new features, bug fixes, and releases are organized and isolated from one another until they're merged into the main branch.

20. **How do you implement branch permissions in Azure Repos?**
    - **Answer**:
        1. Go to **Repos** > **Branches**.
        2. Click on the **More actions** button next to the branch.
        3. Select **Branch Security**.
        4. Add users or groups and assign permissions (e.g., read, write, contribute).
        5. Customize permissions to enforce read-only access or restricted write access.

---

3. CI/CD Pipelines in Azure DevOps

21. **What are Azure Pipelines?**
    - **Answer**: Azure Pipelines is a cloud service in Azure DevOps that allows you to build, test, and deploy applications. It supports CI/CD and can handle projects in various languages and platforms, including .NET, Java, Node.js, Python, and more.
22. **How do you create a build pipeline in Azure DevOps?**
    - **Answer**:
        1. Go to **Pipelines** > **Builds**.
        2. Click **New Pipeline**.
        3. Select your repository (e.g., GitHub, Azure Repos).
        4. Choose **YAML** or **Classic Editor**.
        5. Define the stages in the YAML file or add tasks in the Classic Editor.
        6. Save and run the pipeline.
23. **What are YAML pipelines in Azure DevOps?**
    - **Answer**: YAML pipelines allow developers to define their CI/CD pipeline as code using the YAML language. The pipeline is stored in the repository along with the source code, making it easier to track, version, and share pipeline configurations. Example:

```
trigger:
  branches:
    include:
      - main

pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: Maven@3
    inputs:
      mavenPomFile: 'pom.xml'
      goals: 'package'
```

24. **How do you set up continuous deployment with Azure Pipelines?**
    - **Answer**: Continuous deployment can be set up using a release pipeline:
        1. Go to **Pipelines** > **Releases**.

2. Click **New Release Pipeline**.
3. Add a stage and configure the tasks (e.g., deploy to Azure App Service).
4. Link the pipeline to a **build artifact**.
5. Set the **continuous deployment trigger** to automatically deploy when the build is successful.

25. **What is a build agent in Azure Pipelines?**
    - **Answer**: A build agent is a machine that executes the tasks defined in a pipeline (e.g., build, test, deploy). Azure Pipelines provides **Microsoft-hosted agents** (on-demand VMs) and **self-hosted agents** (your own machines) to run pipeline jobs.
26. **How do you configure parallel jobs in Azure Pipelines?**
    - **Answer**: Parallel jobs allow you to run multiple tasks simultaneously to speed up builds. To configure parallel jobs:
        1. Go to **Organization Settings** > **Parallel Jobs**.
        2. Allocate the number of parallel jobs based on your agent type (Microsoft-hosted or self-hosted).
        3. In the pipeline YAML, you can specify multiple jobs to run in parallel using jobs:

```
jobs:
 - job: Build
   steps:
    - script: echo "Building project"

 - job: Test
   steps:
    - script: echo "Running tests"
```

27. **How do you set up a multi-stage pipeline in Azure DevOps?**
    - **Answer**:
        1. Define the pipeline stages in YAML.
        2. Add jobs and tasks under each stage.
        3. Use conditions to control the flow between stages. Example:

```
stages:
 - stage: Build
   jobs:
```

```
  - job: build
    steps:
      - script: echo "Building the app"

- stage: Deploy
  dependsOn: Build
  jobs:
    - job: deploy
      steps:
        - script: echo "Deploying the app"
```

28. **What is the difference between a build pipeline and a release pipeline?**
    - **Answer**:
        - A **build pipeline** compiles, tests, and packages code. It prepares the application for deployment by producing an artifact.
        - A **release pipeline** deploys the artifact to various environments (e.g., staging, production) based on predefined approval gates and conditions.
29. **How do you use variables in Azure Pipelines?**
    - **Answer**: Variables are used to store values that can be referenced throughout the pipeline. You can define variables at the pipeline, stage, or job level. Example:

```
variables:
  buildConfiguration: 'Release'

steps:
  - script: echo $(buildConfiguration)
```

30. **How do you use pipeline artifacts in Azure DevOps?**
    - **Answer**: Pipeline artifacts are files produced by build or deployment jobs that can be shared across stages or between pipelines. Artifacts are stored in Azure DevOps and can be published using the PublishPipelineArtifact task:

```
steps:
  - publish: $(System.DefaultWorkingDirectory)/build
    artifact: drop
```

---

4. Azure Artifacts

31. **What is Azure Artifacts?**
    - **Answer**: Azure Artifacts is a package management service within Azure DevOps that allows you to store and share Maven, npm, NuGet, Python, and Universal packages. It provides a central location for managing dependencies and libraries across projects.
32. **How do you create a feed in Azure Artifacts?**
    - **Answer**:
        1. Go to **Artifacts** in your Azure DevOps project.
        2. Click **+ New Feed**.
        3. Provide a name for the feed and set its visibility (public or private).
        4. Click **Create** to initialize the feed.
33. **What types of packages can be stored in Azure Artifacts?**
    - **Answer**: Azure Artifacts supports the following package types:
        - **Maven** (for Java projects).
        - **npm** (for Node.js projects).
        - **NuGet** (for .NET projects).
        - **Python** (for Python projects).
        - **Universal Packages** (for arbitrary file types and binaries).
34. **How do you publish a package to Azure Artifacts?**
    - **Answer**: To publish a package to Azure Artifacts, you can use the respective package manager for the language you're using (e.g., Maven, npm, NuGet). Here's an example for npm:
        1. Authenticate with the Azure Artifacts feed:

```
npm login --registry=https://pkgs.dev.azure.com/{organization}/_packaging/{feed}/npm/registry/
```

        2. Publish the package:

```
npm publish
```

35. **How do you consume a package from Azure Artifacts in your project?**
    - **Answer**: To consume a package, you need to configure the package manager to point to the Azure Artifacts feed. Here's an example for npm:
        1. Configure .npmrc to point to your feed:

```
registry=https://pkgs.dev.azure.com/{organization}/_packaging/{feed}/npm/registry/
```

always-auth=true

2. Install the package:

npm install <package-name>

---

5. Azure Boards

36. **What is Azure Boards?**
    - **Answer**: Azure Boards is a tool within Azure DevOps that provides work tracking capabilities for managing software development projects. It supports tracking tasks, bugs, features, and epics using Agile, Scrum, or Kanban methodologies.
37. **What are work items in Azure Boards?**
    - **Answer**: Work items in Azure Boards represent individual units of work, such as tasks, bugs, stories, features, or epics. Each work item tracks the details of the work to be completed and is linked to commits, pull requests, and pipelines.
38. **How do you create a new work item in Azure Boards?**
    - **Answer**:
        1. Go to **Boards** > **Work Items**.
        2. Click **+ New Work Item** and choose the type (e.g., task, bug, user story).
        3. Fill in the details such as title, description, priority, and assignee.
        4. Click **Save** to create the work item.
39. **What is a Kanban board in Azure Boards?**
    - **Answer**: A Kanban board is a visual tool for tracking work items. It consists of columns representing different stages of the development process (e.g., To Do, In Progress, Done). Work items move across these columns as they progress. Azure Boards provides built-in Kanban boards for managing tasks in real-time.
40. **How do you customize work item types in Azure Boards?**
    - **Answer**:
        1. Go to **Organization Settings** > **Process**.
        2. Select the process (e.g., Agile, Scrum, CMMI) that your project is using.
        3. Click on **Work Item Types** and select the type you want to modify (e.g., User Story, Bug).
        4. Add or remove fields, change form layout, or add rules to customize the work item.
41. **How do you manage sprints in Azure Boards?**

- **Answer**: To manage sprints in Azure Boards, you can use the **Sprint Planning** and **Sprint Backlog** features:
    1. Go to **Boards** > **Sprints**.
    2. Click on a specific sprint to view the sprint backlog.
    3. Add work items (tasks, bugs, user stories) to the sprint.
    4. Track progress by moving work items through the various stages (To Do, In Progress, Done).
    5. Use **capacity planning** to ensure that the workload assigned to team members aligns with their available capacity for the sprint.

42. **What are iterations in Azure Boards, and how are they used?**

- **Answer**: **Iterations** represent time-based cycles in which teams deliver work. In Azure Boards, iterations are used to group work items by time (e.g., Sprints). Teams can plan, assign, and track work over specific iterations to ensure progress toward larger goals.

43. **What are backlogs in Azure Boards?**

- **Answer**: A backlog in Azure Boards is a prioritized list of work items that need to be completed. The backlog can include features, epics, user stories, tasks, and bugs. Teams use backlogs to plan and prioritize work for future sprints or releases.

44. **How do you create a custom query in Azure Boards?**

- **Answer**:
    1. Go to **Boards** > **Queries**.
    2. Click **New Query**.
    3. Define your query criteria by selecting fields, conditions, and values (e.g., Assigned To, State, Work Item Type).
    4. You can use **And/Or operators** to combine conditions.
    5. Click **Run Query** to see the results and save the query for future use.

45. **How do you track progress in Azure Boards using Dashboards?**

- **Answer**: Dashboards in Azure Boards provide visual insights into project progress. You can create widgets to track:
    - Work item status.
    - Burndown charts.
    - Team capacity and velocity.
    - Blockers and risk items.
    - To create a dashboard:

1. Go to **Dashboards** > **New Dashboard**.
2. Add relevant widgets (burndown charts, cumulative flow diagrams, query results).
3. Customize the layout to suit your project needs.

46. **How do you track dependencies between work items in Azure Boards?**

- **Answer**: Azure Boards allows you to link work items to indicate dependencies (e.g., "Parent-Child," "Related," "Blocked By"). To track dependencies:
    1. Open a work item.
    2. Scroll down to the **Links** section and click **Add Link**.
    3. Choose the relationship type (e.g., **Parent, Child, Related, Dependent**).
    4. Select the dependent work item and click **OK**.

47. **What is the velocity chart in Azure Boards, and how is it used?**

- **Answer**: The **Velocity Chart** helps track the amount of work a team completes during a sprint. It displays the number of story points or tasks completed in previous sprints, enabling teams to estimate future work based on past performance. You can find the velocity chart under **Boards** > **Sprints** > **Analytics**.

48. **How do you use the cumulative flow diagram in Azure Boards?**

- **Answer**: The **Cumulative Flow Diagram (CFD)** visualizes the flow of work items through various stages of a process (e.g., New, Active, Resolved, Closed). It helps identify bottlenecks by showing how work is accumulating at different stages. To use the CFD:
    1. Go to **Boards** > **Sprints**.
    2. Select the **Analytics** tab.
    3. Add a **Cumulative Flow Diagram** widget to the dashboard.

49. **What is Azure DevOps Process Template, and how do you customize it?**

- **Answer**: Azure DevOps Process Templates define the default work item types, states, and rules for managing work in a project. To customize a process template:
    1. Go to **Organization Settings** > **Process**.
    2. Select a process template (Agile, Scrum, or CMMI).
    3. Clone the process to create a custom version.
    4. Modify work item types, states, and rules according to your needs.

50. **How do you use work item tags in Azure Boards?**

- **Answer**: Tags are used to categorize work items, making it easier to filter, query, and group them. To add tags to a work item:
    1. Open a work item.

2. In the **Tags** section, enter a tag name and press **Enter**.
3. Tags can be used in queries to filter work items by specific categories.

---

6. Infrastructure as Code (IaC) with Azure DevOps

51. **What is Infrastructure as Code (IaC)?**

- **Answer**: Infrastructure as Code (IaC) is the practice of defining and managing infrastructure using code. With IaC, infrastructure components (e.g., servers, databases, networking) are described in machine-readable files (e.g., JSON, YAML), enabling automated provisioning, consistency, and repeatability.

52. **How does Azure DevOps support Infrastructure as Code (IaC)?**

- **Answer**: Azure DevOps supports IaC by integrating with tools like **Terraform**, **ARM templates**, and **Ansible** within pipelines. Using these tools, you can automate the creation and configuration of Azure resources in a consistent and repeatable manner.

53. **What is the role of ARM templates in Azure DevOps?**

- **Answer**: **ARM (Azure Resource Manager) templates** are JSON files that define Azure resources. You can deploy infrastructure as code by defining the desired resources in an ARM template and using an Azure DevOps pipeline to deploy those resources to your Azure environment.

54. **How do you integrate Terraform with Azure DevOps for IaC?**

- **Answer**:
    1. Install the **Terraform** task in your Azure DevOps pipeline.
    2. Use a pipeline to run terraform init, terraform plan, and terraform apply to create and manage resources in Azure.
    3. Example YAML for a Terraform pipeline:

```
trigger:
  branches:
    include:
      - main

pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: TerraformInstaller@0
```

```
  inputs:
    terraformVersion: '0.14.x'

- script: |
    terraform init
    terraform plan
    terraform apply -auto-approve
```

55. **What is the difference between ARM templates and Terraform in Azure DevOps?**

- **Answer**: ARM templates are specific to Azure and use JSON to define Azure resources. **Terraform**, on the other hand, is a cloud-agnostic IaC tool that can manage infrastructure across multiple cloud providers (Azure, AWS, GCP) using a common configuration language (HCL).

56. **How do you use Ansible with Azure DevOps?**

- **Answer**: **Ansible** is an open-source automation tool that can manage infrastructure configuration and application deployment. You can use an Azure DevOps pipeline to trigger Ansible playbooks that provision resources or configure servers in Azure. Example:

```
pool:
  vmImage: 'ubuntu-latest'

steps:
- task: UsePythonVersion@0
  inputs:
    versionSpec: '3.x'
    addToPath: true

- script: |
    ansible-playbook playbook.yml -i inventory
```

57. **What are Azure Blueprints, and how do they relate to IaC?**

- **Answer**: **Azure Blueprints** allow you to define a repeatable set of Azure resources, policies, and RBAC assignments. Blueprints help ensure compliance and governance across multiple environments. They complement IaC by providing an easy way to deploy and enforce configuration standards alongside ARM templates or Terraform.

58. **How do you use the Azure Resource Manager (ARM) deployment task in Azure DevOps?**

- **Answer**: To deploy Azure resources using ARM templates in an Azure DevOps pipeline:

1. Create an ARM template that defines the Azure resources.
2. Use the **Azure Resource Group Deployment** task in your pipeline to deploy the template:

```
steps:
 - task: AzureResourceManagerTemplateDeployment@3
   inputs:
    deploymentScope: 'Resource Group'
    azureResourceManagerConnection: 'MyAzureServiceConnection'
    subscriptionId: 'xxxxx-xxxx-xxxx-xxxx-xxxxxxx'
    action: 'Create Or Update Resource Group'
    resourceGroupName: 'myResourceGroup'
    location: 'East US'
    templateLocation: 'Linked artifact'
    csmFile: '$(System.DefaultWorkingDirectory)/armTemplate.json'
```

59. **How do you automate the deployment of Azure infrastructure using Terraform in Azure DevOps?**

- **Answer**: Automating the deployment of Azure infrastructure with Terraform involves:
    1. Writing the Terraform configuration files (e.g., main.tf).
    2. Creating an Azure DevOps pipeline to run Terraform commands (init, plan, apply).
    3. Using a service principal for authentication to Azure within the pipeline.

60. **What are the benefits of using Infrastructure as Code (IaC) in Azure DevOps?**

- **Answer**: Benefits of using IaC include:
    - **Consistency**: Infrastructure configurations are consistent across environments.
    - **Automation**: Reduces manual intervention and errors by automating resource provisioning.
    - **Version Control**: Infrastructure changes can be tracked, rolled back, and audited.
    - **Scalability**: Infrastructure can be quickly scaled up or down based on code changes.
    - **Collaboration**: Infrastructure as code allows teams to collaborate on infrastructure in the same way they collaborate on application code.

---

7. Azure DevOps Security and Permissions

61. **What are Azure DevOps Security Groups?**

- **Answer**: **Security Groups** in Azure DevOps allow you to manage permissions for a group of users. These groups control access to resources like repositories, pipelines, boards, and artifacts. Common groups include **Project Administrators**, **Contributors**, and **Readers**.

62. **How do you manage user permissions in Azure DevOps?**

- **Answer**:
    1. Go to **Project Settings** > **Permissions**.
    2. Select a group or user.
    3. Assign specific permissions (e.g., read, write, manage) for each resource (repos, pipelines, boards).
    4. Click **Save Changes**.

63. **What is a Service Principal, and how is it used in Azure DevOps?**

- **Answer**: A **Service Principal** is an Azure AD identity that represents an application or service. In Azure DevOps, service principals are used to authenticate and authorize pipelines to deploy or manage resources in Azure. Service connections to Azure often rely on service principals for secure access.

64. **What is the difference between role-based access control (RBAC) and permissions in Azure DevOps?**

- **Answer**: **RBAC** in Azure applies to Azure resources, controlling who can access resources like VMs, storage accounts, and networks. **Permissions** in Azure DevOps apply to DevOps resources like repositories, pipelines, and boards. RBAC controls access within Azure, while Azure DevOps permissions control access within the DevOps environment.

65. **How do you configure branch protection policies in Azure Repos?**

- **Answer**: Branch protection policies ensure code quality and prevent unauthorized changes. To configure branch policies:
    1. Go to **Repos** > **Branches**.
    2. Select a branch and click **Branch Policies**.
    3. Configure policies such as:
        - **Require minimum number of reviewers**.
        - **Build validation**.
        - **Comment resolution**.
        - **Work item linking**.
    4. Save the policies to enforce them.

66. **How do you secure access to pipelines in Azure DevOps?**

- **Answer**: Pipeline access can be secured using:
    - **Pipeline permissions**: Control who can view, edit, or run pipelines.

- **Environment permissions**: Secure environments by restricting access to specific users or groups.
- **Service connections**: Use secure service connections (e.g., service principals) for deploying resources.
- **Pipeline secrets**: Store sensitive data (e.g., passwords, API tokens) in Azure Key Vault or pipeline variables.

67. **How do you secure access to Azure Artifacts feeds?**

- **Answer**: To secure an Azure Artifacts feed:
    1. Go to **Artifacts** > **Feeds**.
    2. Select the feed and click on **Feed Settings**.
    3. Set the visibility to **Private**.
    4. Assign permissions to specific users or groups to control access (e.g., readers, contributors, owners).

68. **What is Azure Key Vault, and how do you use it in Azure Pipelines?**

- **Answer**: **Azure Key Vault** is a service that securely stores and manages secrets, keys, and certificates. In Azure Pipelines, you can integrate with Azure Key Vault to retrieve secrets during the pipeline execution. Example:

```
steps:
 - task: AzureKeyVault@2
   inputs:
    azureSubscription: 'MyAzureServiceConnection'
    KeyVaultName: 'myKeyVault'
    SecretsFilter: '*'
    RunAsPreJob: false
```

69. **How do you implement secure deployment pipelines in Azure DevOps?**

- **Answer**: Secure deployment pipelines can be implemented by:
    - Using **service connections** with least privilege.
    - Storing sensitive data in **Azure Key Vault** and using **pipeline secrets**.
    - Restricting **environment access** so that only authorized users can deploy to production.
    - Enforcing **build validations** and **approval gates** before releasing code.

70. **What are Azure DevOps Audit Logs, and how do you use them?**

- **Answer**: **Audit logs** in Azure DevOps track changes made to an organization or project, such as user access, permissions changes, and pipeline executions. They provide insights for security and compliance audits. To access audit logs:
    1. Go to **Organization Settings** > **Auditing**.
    2. Review logs of actions taken by users and services.

---

8. Testing in Azure DevOps

71. **What are Azure Test Plans?**

- **Answer**: **Azure Test Plans** provide tools for manual and automated testing in Azure DevOps. Teams can create test cases, execute test runs, track results, and analyze test coverage, making it easier to ensure software quality.

72. **How do you create a test case in Azure Test Plans?**

- **Answer**:
    1. Go to **Test Plans** in Azure DevOps.
    2. Click **New Test Case**.
    3. Fill in the test case details, including title, steps, expected results, and priority.
    4. Save the test case and add it to a test plan.

73. **How do you integrate automated tests in an Azure Pipeline?**

- **Answer**: To integrate automated tests in an Azure Pipeline:
    1. Define the test tasks (e.g., NUnit, JUnit) in the pipeline.
    2. Add a step to run the tests after the build stage.
    3. Publish test results to Azure Pipelines for reporting. Example YAML:

```
steps:
 - script: dotnet test
   displayName: 'Run Unit Tests'
 - task: PublishTestResults@2
   inputs:
    testResultsFiles: '**/*.trx'
```

74. **How do you use test plans in CI/CD pipelines?**

- **Answer**: Test plans can be used in CI/CD pipelines to validate deployments before moving to the next stage. For example, you can run a **Test Plan** in the pipeline using the **Run Tests** task:

```
steps:
```

```
- task: VSTest@2
  inputs:
    testSelector: 'testPlan'
    testPlan: '12345'
    testSuite: '1'
```

75. **What is the purpose of Test Suites in Azure DevOps?**

- **Answer**: **Test Suites** group test cases into logical categories (e.g., functional tests, regression tests). You can create different test suites for different scenarios and track the overall status of the suite within test plans.

---

9. Monitoring in Azure DevOps

76. **How do you monitor pipeline performance in Azure DevOps?**

- **Answer**: Azure Pipelines provides built-in analytics for monitoring pipeline performance, including:
    - **Build success rate**.
    - **Test pass/fail rate**.
    - **Pipeline duration**.
    - **Code coverage**. These analytics can be accessed via the **Pipelines** > **Analytics** tab.

77. **What is Azure Monitor, and how does it integrate with Azure DevOps?**

- **Answer**: **Azure Monitor** provides full-stack monitoring for applications and infrastructure in Azure. It integrates with Azure DevOps to provide real-time telemetry and insights into applications deployed using pipelines. Alerts can be triggered based on performance metrics or failures.

78. **What is Application Insights, and how do you integrate it with Azure Pipelines?**

- **Answer**: **Application Insights** is an application performance monitoring service that helps developers detect performance issues, exceptions, and telemetry data in real-time. It can be integrated with Azure Pipelines by:
    1. Deploying Application Insights to the target environment.
    2. Adding the Application Insights SDK to the application.
    3. Using telemetry from Application Insights to track issues in production.

79. **How do you monitor infrastructure deployed with Azure DevOps?**

- **Answer**: Infrastructure deployed using Azure DevOps can be monitored using:
    - **Azure Monitor**: Tracks the health and performance of Azure resources.
    - **Log Analytics**: Provides logs for troubleshooting and querying metrics.

- **Azure Alerts**: Can be set up to notify teams when resource thresholds are exceeded.

80. **What is the purpose of Azure DevOps Pipeline Retention Policies?**

- **Answer**: **Pipeline Retention Policies** determine how long build and release artifacts, logs, and results are retained before being deleted. Proper retention policies help manage storage costs while preserving critical data for compliance and troubleshooting.

10. Advanced Scenarios and Troubleshooting in Azure DevOps

81. **How do you troubleshoot pipeline failures in Azure DevOps?**

- **Answer**: To troubleshoot pipeline failures:
    1. Review the pipeline logs to identify the step that failed.
    2. Check for common issues like missing dependencies, incorrect environment variables, or invalid credentials.
    3. Use the **Debug Logging** feature to get detailed logs for the failing task.
    4. Use **Azure DevOps Auditing** to track who made changes that might have impacted the pipeline.

82. **How do you roll back a deployment in Azure Pipelines?**

- **Answer**: To roll back a deployment:
    1. Go to the **Releases** section.
    2. Identify a previous successful release.
    3. Click **Redeploy** to deploy the previous version to the target environment.

83. **How do you troubleshoot slow pipeline performance in Azure DevOps?**

- **Answer**: Troubleshooting slow pipeline performance involves:
    - **Identifying bottlenecks** in build or test stages.
    - **Optimizing tasks** (e.g., caching dependencies or parallelizing tests).
    - **Using pipeline analytics** to monitor build and release durations over time.

84. **What are pipeline timeouts, and how do you manage them in Azure Pipelines?**

- **Answer**: Pipeline timeouts prevent long-running jobs from consuming unnecessary resources. In Azure Pipelines, you can define timeouts at the pipeline or job level. Example:

```
jobs:
  - job: Build
    timeoutInMinutes: 30
```

85. **How do you handle secrets in Azure Pipelines securely?**

- **Answer**: To handle secrets securely in Azure Pipelines:
    - Use **Azure Key Vault** to store sensitive data.
    - Use **pipeline variables** to inject secrets securely into the pipeline.
    - Set variables as **secret** to mask them in logs.

86. **What is a self-hosted agent in Azure DevOps, and how do you configure one?**

- **Answer**: A **self-hosted agent** is a machine that you manage to run Azure Pipelines jobs. It allows for more control over the environment compared to Microsoft-hosted agents. To configure a self-hosted agent:
    1. Go to **Organization Settings** > **Agent Pools** > **New Agent**.
    2. Download and extract the agent package on your machine.
    3. Configure the agent with your Azure DevOps URL, Personal Access Token (PAT), and other settings.
    4. Start the agent using:

./svc.sh install

./svc.sh start

87. **What is a release gate in Azure Pipelines, and how is it used?**

- **Answer**: **Release gates** are conditions that must be met before a release can be promoted to the next stage. They allow you to validate the health of an environment before proceeding. Examples include:
    - Checking the status of monitoring tools like Application Insights.
    - Running Azure Functions or REST API checks.

88. **How do you handle pipeline failures caused by environmental issues?**

- **Answer**: To troubleshoot environmental issues:
    1. Ensure that the agent environment is consistent (e.g., required software installed).
    2. Use **Microsoft-hosted agents** to avoid environment-specific issues.
    3. Check for network connectivity, firewall, or authentication issues between the agent and Azure services.
    4. Use tools like **Azure Monitor** to track resource usage and performance.

89. **What are common causes of slow pipeline execution in Azure DevOps?**

- **Answer**:
    - Insufficient agent resources.
    - Inefficient tasks, such as downloading large dependencies each time.

- Long-running unit tests or integration tests.
- Misconfigured cache or artifacts.

90. **What are deployment slots in Azure App Services, and how do you use them in Azure Pipelines?**

- **Answer**: **Deployment slots** are live app instances in Azure App Services that can host different versions of an application. They allow for zero-downtime deployments by swapping slots. To use them:
    1. Add the **Azure App Service Deploy** task to the release pipeline.
    2. Select the deployment slot (e.g., staging) in the task configuration.
    3. Use the **Swap Slots** task to switch the staging slot to production after validation.

---

12. Azure DevOps Integration with Cloud Platforms

91. **How do you set up a pipeline in Azure DevOps to deploy to Azure Kubernetes Service (AKS)?**

- **Answer**:
    1. Create a Kubernetes cluster in **Azure Kubernetes Service (AKS)**.
    2. In your pipeline, use the **kubectl** task to apply Kubernetes manifests (e.g., deployment.yaml, service.yaml) to AKS.
    3. Example YAML:

```
trigger:
  branches:
    include:
      - main

pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: KubectlInstaller@0
    inputs:
      kubectlVersion: 'latest'

  - task: Kubernetes@1
```

```
  inputs:
    connectionType: 'Azure Resource Manager'
    azureSubscriptionEndpoint: 'MyAzureSubscription'
    kubernetesCluster: 'MyAKSCluster'
    namespace: 'default'
    command: 'apply'
    arguments: '-f deployment.yaml'
```

92. **How do you configure a pipeline to deploy a function app to Azure Functions?**

- **Answer**:
    1. Add the **Azure Functions App Deploy** task to the pipeline.
    2. Configure the **Azure subscription** and select the **Function App**.
    3. Set the **package location** (e.g., the artifact produced from the build).
    4. Deploy the package using:

```
steps:
 - task: AzureFunctionApp@1
   inputs:
    azureSubscription: 'MyAzureSubscription'
    appType: 'functionApp'
    appName: 'MyFunctionApp'
    package: '$(System.DefaultWorkingDirectory)/drop/package.zip'
```

93. **How do you use Azure DevOps to deploy resources using Azure CLI commands?**

- **Answer**:
    1. Use the **Azure CLI task** in your pipeline.
    2. Specify the CLI commands to deploy resources or manage infrastructure (e.g., creating VMs, resource groups, etc.).
    3. Example:

```
steps:
 - task: AzureCLI@2
   inputs:
    azureSubscription: 'MyAzureSubscription'
```

```
  scriptType: 'bash'
  scriptLocation: 'inlineScript'
  inlineScript: |
    az group create --name MyResourceGroup --location eastus
    az vm create --resource-group MyResourceGroup --name MyVM --image UbuntuLTS
```

94. **How do you manage Azure Storage accounts using Azure DevOps pipelines?**

- **Answer**:
    1. Add the **Azure CLI task** or **Azure PowerShell task** to the pipeline.
    2. Use the respective commands to create, update, or delete storage accounts.
    3. Example to create a storage account using CLI:

```
steps:
- task: AzureCLI@2
  inputs:
    azureSubscription: 'MyAzureSubscription'
    scriptType: 'bash'
    scriptLocation: 'inlineScript'
    inlineScript: |
      az storage account create --name mystorageaccount --resource-group myResourceGroup --location eastus --sku Standard_LRS
```

95. **What is the benefit of using Azure Resource Manager (ARM) templates in Azure DevOps pipelines?**

- **Answer**: ARM templates allow you to define infrastructure in a declarative way, enabling consistent and repeatable deployments. They are integrated with Azure DevOps pipelines to automate the provisioning of Azure resources without manual intervention, ensuring that deployments are idempotent.

---

13. GitOps and Azure DevOps

96. **What is GitOps, and how does it relate to Azure DevOps?**

- **Answer**: **GitOps** is a DevOps practice where Git is the single source of truth for both application and infrastructure code. With GitOps, every change is triggered by a Git commit, and automated workflows (via CI/CD pipelines) handle deployments. Azure DevOps can implement GitOps by managing infrastructure as code (IaC) in Azure Repos and using pipelines to deploy changes.

97. **How do you implement GitOps for Kubernetes in Azure DevOps?**

- **Answer**:
    1. Store Kubernetes manifests (e.g., deployment.yaml) in an Azure Repo.
    2. Create an Azure Pipeline to apply these manifests to a Kubernetes cluster using kubectl.
    3. Trigger the pipeline whenever changes are committed to the Git repository.

98. **How do you automate infrastructure changes using GitOps with Terraform in Azure DevOps?**

- **Answer**:
    1. Store Terraform configuration files (e.g., main.tf) in an Azure Repo.
    2. Create an Azure Pipeline to run Terraform commands (init, plan, apply) whenever changes are pushed to the repository.
    3. Use service connections for authentication with Azure resources.

99. **What is Flux, and how does it integrate with Azure DevOps?**

- **Answer**: **Flux** is a GitOps tool that automates the deployment of Kubernetes manifests by syncing them from a Git repository. It monitors the repo for changes and automatically updates the Kubernetes cluster. In Azure DevOps, you can use Flux to deploy Kubernetes configurations stored in Azure Repos.

100. **How do you set up Flux in a Kubernetes cluster using Azure DevOps?**

- **Answer**:
    1. Install Flux in the Kubernetes cluster by running the **fluxctl** command in an Azure Pipeline.
    2. Point Flux to the Azure Repo containing Kubernetes manifests.
    3. Configure Flux to automatically sync changes from the repo to the cluster.

---

14. Azure Repos Advanced

101. **What is the purpose of branch policies in Azure Repos, and how do you configure them?**

- **Answer**: Branch policies in Azure Repos ensure code quality by enforcing rules before changes are merged into a protected branch. You can configure policies such as requiring code reviews, build validation, and linked work items. To configure:
    1. Go to **Repos** > **Branches**.
    2. Select a branch and click **Branch Policies**.
    3. Enable policies like **Minimum number of reviewers**, **Build validation**, etc.

102. **How do you use code reviews in Azure Repos?**

- **Answer**: Code reviews are performed via pull requests (PRs). Once a PR is created, reviewers can inspect the code, leave comments, and suggest changes before approving or rejecting the PR. You can configure branch policies to enforce a code review before a PR is merged.

103. **What are Git submodules, and how are they used in Azure Repos?**

- **Answer**: **Git submodules** allow you to include one Git repository as a subdirectory in another repository. This is useful for reusing code across multiple projects. In Azure Repos, you can manage submodules by including them in the pipeline build process using the submodule keyword.

104. **How do you manage pull request (PR) approvals in Azure Repos?**

- **Answer**: PR approvals are managed by setting up branch policies that require a minimum number of reviewers to approve the PR before it can be merged. You can also set up policies that ensure reviewers from specific teams or with specific roles are required for approval.

105. **What is a Git fork, and how is it used in Azure DevOps?**

- **Answer**: A **fork** is a copy of a repository that you manage independently from the original. In Azure DevOps, forks are commonly used when developers want to contribute to a project but do not have permission to push directly to the original repository. After making changes in the forked repo, developers create a pull request to merge their changes into the original repo.

---

15. DevSecOps in Azure DevOps

106. **What is DevSecOps, and how does it integrate with Azure DevOps?**

- **Answer**: **DevSecOps** is the practice of integrating security into every stage of the DevOps lifecycle. In Azure DevOps, security tools (e.g., static code analysis, vulnerability scanning) can be integrated into CI/CD pipelines to ensure security checks are automated and enforced as part of the deployment process.

107. **How do you integrate SonarQube with Azure DevOps for static code analysis?**

- **Answer**:

    1. Install the **SonarQube extension** from the Azure DevOps marketplace.
    2. Add the **Prepare Analysis Configuration** and **Run Code Analysis** tasks to your build pipeline.
    3. Specify your SonarQube server details and project key in the pipeline.

108. **What is OWASP ZAP, and how can it be integrated into Azure Pipelines?**

- **Answer**: **OWASP ZAP** is a security testing tool used for identifying vulnerabilities in web applications. You can integrate it into Azure Pipelines by adding a task that runs ZAP against the application after it is deployed to a test environment. Example:

steps:

```
- task: OWASPZAP@2
  inputs:
    url: 'http://myapplication'
```

109. **How do you use Trivy for vulnerability scanning in Azure DevOps?**

- **Answer**: **Trivy** is a container and filesystem vulnerability scanner. To use it in Azure DevOps:
    1. Add a step in your pipeline to run Trivy on the container image or file system.
    2. Example:

yaml

Copy code

```
steps:
- script: |
    trivy image myimage:latest
    trivy filesystem /path/to/files
```

110. **How do you implement security scanning in CI/CD pipelines in Azure DevOps?**

- **Answer**: To implement security scanning:
    1. Integrate tools like **SonarQube**, **OWASP ZAP**, **Trivy**, or **WhiteSource Bolt** into your build or release pipeline.
    2. Run security scans after the build and before deploying to production to ensure that vulnerabilities are detected and addressed early.

---

16. Azure DevOps Testing and Quality

111. **What are the different types of tests you can run in Azure DevOps pipelines?**

- **Answer**: In Azure DevOps, you can run several types of tests:
    - **Unit Tests**: Test individual components of code.
    - **Integration Tests**: Test how different components interact.
    - **Functional Tests**: Test the functionality of the application.
    - **Load Tests**: Test the performance of the application under load.
    - **Security Tests**: Test the security of the application using tools like OWASP ZAP.

112. **How do you run unit tests in Azure Pipelines?**

- **Answer**: To run unit tests in a pipeline:
    1. Add a testing task (e.g., **Visual Studio Test**, **VSTest**, or **NUnit**) to your build pipeline.
    2. Specify the location of the test files (e.g., test.dll) and run the tests.

3. Example for .NET tests:

```
steps:
 - task: DotNetCoreCLI@2
   inputs:
    command: 'test'
    projects: '**/*.csproj'
```

113. **How do you collect and publish test results in Azure Pipelines?**

- **Answer**: Use the **Publish Test Results** task to collect and publish test results to Azure DevOps for analysis:

```
steps:
 - task: PublishTestResults@2
   inputs:
    testResultsFiles: '**/*.trx'
```

114. **How do you use the Test Plans feature in Azure DevOps for manual testing?**

- **Answer**: **Test Plans** in Azure DevOps enable teams to manage manual testing. To use:
  1. Create test cases under **Test Plans**.
  2. Organize test cases into suites (e.g., functional tests, regression tests).
  3. Execute test cases and track results, bugs, and issues.
  4. Generate test reports to analyze test execution progress and quality.

115. **How do you automate functional tests in Azure Pipelines?**

- **Answer**: Functional tests can be automated using testing frameworks (e.g., Selenium for web apps). To run functional tests in a pipeline:
  1. Add the necessary test runner (e.g., **Selenium**, **TestComplete**).
  2. Run the functional tests after the application is deployed to a test environment.
  3. Publish the test results.

---

17. Azure DevOps for Mobile Development

116. **How do you build and deploy a mobile application using Azure DevOps?**

- **Answer**:
  1. Set up a pipeline that builds the mobile application (e.g., using **Xamarin**, **React Native**, or **Android/iOS native** build tasks).
  2. Use the **App Center Distribute** task to deploy the mobile app to test users.

3. Example YAML for Xamarin app:

```
trigger:
  branches:
    include:
      - main

pool:
  vmImage: 'macOS-latest'

steps:
  - task: XamarinAndroid@1
    inputs:
      projectFile: '**/*.csproj'
      outputDir: '$(build.artifactStagingDirectory)'
```

117. **How do you automate mobile app testing using Azure DevOps and App Center?**

- **Answer**: To automate mobile app testing:
    1. Set up a pipeline to build the mobile app (Android/iOS).
    2. Integrate with **App Center Test** to run automated UI tests.
    3. Example for App Center test task:

```
steps:
  - task: AppCenterTest@1
    inputs:
      appCenterAppSlug: 'myorg/myapp'
      devices: 'mydevicegroup'
      appFile: '$(build.artifactStagingDirectory)/myapp.apk'
      testSeries: 'master'
      locale: 'en_US'
      testFramework: 'appium'
```

118. **How do you deploy a mobile app to App Store/Google Play using Azure DevOps?**

- **Answer**:
    1. Set up a pipeline that builds the mobile app using tasks like **Xcode** for iOS or **Android Gradle** for Android.
    2. Use the **App Store Release** task or **Google Play Release** task to publish the app directly to the respective app store.
    3. Example for deploying to Google Play:

```
steps:
  - task: GooglePlayRelease@3
    inputs:
      serviceEndpoint: 'Google Play Connection'
      apkFile: '$(build.artifactStagingDirectory)/*.apk'
      track: 'alpha'
```

119. **How do you handle code signing for iOS applications in Azure DevOps?**

- **Answer**: iOS applications require code signing before they can be deployed to the App Store or distributed. In Azure DevOps, you can use the **Apple App Store Connect** task to manage certificates and provisioning profiles automatically. Additionally:
    1. Use a **Secure Files Library** to securely store the .p12 certificate and provisioning profile.
    2. Use the **InstallAppleCertificate** and **InstallAppleProvisioningProfile** tasks to install the required files during the build pipeline.

120. **How do you build an Android APK file using Azure DevOps?**

- **Answer**:
    1. Use the **Android Gradle** task to compile the Android project and create an APK.
    2. Example YAML:

```
steps:
  - task: Gradle@2
    inputs:
      gradleWrapperFile: 'android/gradlew'
      tasks: 'assembleRelease'
      options: ''
```

```
  publishJUnitResults: true
  testResultsFiles: '**/TEST-*.xml'
```

121. **How do you perform unit testing for a mobile app using Azure DevOps?**

- **Answer**: To perform unit testing for mobile apps, you can use frameworks like **JUnit** for Android and **XCTest** for iOS. To integrate unit tests into an Azure Pipeline:
    1. Add a task for running the appropriate test framework.
    2. Publish test results to Azure DevOps for analysis.

---

19. Advanced Azure DevOps Scenarios

122. **How do you handle multi-stage pipelines in Azure DevOps?**

- **Answer**:
    1. Define multiple stages in your YAML pipeline (e.g., build, test, deploy).
    2. Each stage can have multiple jobs, and conditions can be used to control the flow between stages.
    3. Example:

```
stages:
 - stage: Build
   jobs:
    - job: BuildApp
      steps:
       - script: echo "Building app"

 - stage: Test
   dependsOn: Build
   jobs:
    - job: RunTests
      steps:
       - script: echo "Running tests"

 - stage: Deploy
   dependsOn: Test
   jobs:
```

```
  - job: DeployApp
    steps:
      - script: echo "Deploying app"
```

123. **How do you implement approvals and gates in a multi-stage pipeline?**

- **Answer**: Approvals and gates can be used in Azure Pipelines to pause execution until a user manually approves or a gate condition is met:
    1. In **Release Pipelines**, configure **pre-deployment approvals** and **post-deployment approvals**.
    2. Define **gates** for conditions like checking Azure Monitor alerts, querying APIs, or checking for active incidents in the pipeline.

124. **What are conditional steps in Azure Pipelines, and how do you use them?**

- **Answer**: Conditional steps allow you to execute a task only if a condition is met. Example:

```
steps:
  - script: echo "Running tests"
    condition: eq(variables['Build.SourceBranch'], 'refs/heads/main')
```

125. **What is the difference between a job and a stage in Azure Pipelines?**

- **Answer**:
    - A **stage** is a collection of jobs, typically representing a phase in the pipeline (e.g., build, test, deploy).
    - A **job** is a set of steps that run on the same agent. Jobs can run in parallel within a stage, whereas stages typically run sequentially (unless explicitly configured for parallel execution).

126. **How do you run jobs in parallel in Azure Pipelines?**

- **Answer**: You can configure jobs to run in parallel by defining multiple jobs within a stage. Each job will be run on a separate agent:

```
stages:
  - stage: BuildAndTest
    jobs:
      - job: Build
        steps:
          - script: echo "Building"


      - job: Test
```

```
    steps:
      - script: echo "Testing"
```

127. **How do you set up Azure DevOps pipelines to handle different environments (Dev, QA, Prod)?**

- **Answer**:
    1. Use **multi-stage pipelines** to define separate stages for Dev, QA, and Prod.
    2. Configure environment-specific tasks, variables, and approvals.
    3. Example:

```
stages:
  - stage: Dev
    jobs:
      - job: DeployToDev
        steps:
          - script: echo "Deploying to Dev"

  - stage: QA
    dependsOn: Dev
    jobs:
      - job: DeployToQA
        steps:
          - script: echo "Deploying to QA"

  - stage: Prod
    dependsOn: QA
    jobs:
      - job: DeployToProd
        steps:
          - script: echo "Deploying to Prod"
```

128. **How do you implement a Blue/Green deployment in Azure DevOps?**

- **Answer**: Blue/Green deployment involves deploying a new version of an application (Green) alongside the current version (Blue) in a separate environment, then swapping traffic between them. In Azure DevOps:

1. Deploy the new version to a **staging slot** (Green).
2. Use the **Azure App Service Slot Swap** task to swap the Green slot with the Blue slot (production).

129. **How do you use caching in Azure Pipelines to speed up builds?**

- **Answer**: The **Cache task** in Azure Pipelines allows you to store dependencies or intermediate build artifacts. Example for caching npm packages:

```
steps:
 - task: Cache@2
   inputs:
    key: 'npm | "$(Agent.OS)" | package-lock.json'
    path: '$(npm_cache)'
    restoreKeys: 'npm | "$(Agent.OS)"'
```

130. **How do you run a pipeline on a schedule in Azure DevOps?**

- **Answer**: To schedule a pipeline:
    1. In YAML pipelines, use the schedules trigger to define a cron-like schedule:

```
schedules:
 - cron: "0 0 * * *"
   displayName: Daily midnight build
   branches:
    include:
     - main
```

---

20. Monitoring and Observability in Azure DevOps

131. **What is Application Insights, and how does it integrate with Azure Pipelines?**

- **Answer**: **Application Insights** is an application performance monitoring service that tracks requests, exceptions, dependencies, and other telemetry data. It integrates with Azure Pipelines to monitor applications after deployment. You can use the **Azure Monitor** task in pipelines to trigger alerts based on Application Insights telemetry.

132. **How do you set up monitoring for an application deployed via Azure DevOps pipelines?**

- **Answer**:
    1. Deploy the application with Application Insights enabled.
    2. Use **Azure Monitor** to set up alerts and dashboards.

3. Configure alerts to trigger based on thresholds for errors, latency, or throughput.

133. **How do you monitor the health of Azure DevOps agents?**

- **Answer**: You can monitor Azure DevOps agents using:
    - The **Agent Pools** dashboard in Azure DevOps, which provides insights into agent status, capacity, and jobs.
    - Azure Monitor and Application Insights for more detailed telemetry about self-hosted agents.

134. **How do you create alerts in Azure Monitor for pipeline failures?**

- **Answer**:
    1. Go to **Azure Monitor** and create an **Alert Rule**.
    2. Select the **Signal Type** (e.g., Pipeline Run Failed) and configure the conditions.
    3. Set the action (e.g., email, SMS, webhook) to notify the team when a pipeline failure occurs.

135. **What is Azure Log Analytics, and how does it work with Azure DevOps?**

- **Answer**: **Azure Log Analytics** is a service in Azure Monitor that collects and analyzes telemetry data from Azure resources. You can query logs, create dashboards, and set up alerts based on data collected from Azure DevOps builds, deployments, and infrastructure.

136. **How do you use Azure Monitor for infrastructure monitoring with Azure Pipelines?**

- **Answer**:
    1. Add monitoring tasks (e.g., checking resource utilization) in your pipeline.
    2. Use **Azure Monitor** to track performance, availability, and security metrics of your infrastructure.
    3. Set up **alerts** and **dashboards** to visualize the health of resources.

137. **How do you monitor pipeline performance in Azure DevOps?**

- **Answer**: Pipeline performance can be monitored through built-in **pipeline analytics** in Azure DevOps, which provides insights into:
    - Build success rates.
    - Deployment durations.
    - Test results.
    - Time spent in different pipeline stages.

138. **How do you use Grafana to visualize Azure DevOps metrics?**

- **Answer**:
    1. Integrate **Azure Monitor** as a data source in Grafana.

2. Create dashboards to visualize key metrics (e.g., build success rates, deployment times).
3. Use queries to pull specific metrics from Azure Monitor (e.g., logs from Application Insights).

---

21. Azure DevOps and GitHub Integration

139. **How do you integrate Azure Pipelines with GitHub?**

- **Answer**:
  1. Create a pipeline in Azure DevOps.
  2. Select **GitHub** as the repository source.
  3. Authenticate to GitHub and choose the repository.
  4. Define the pipeline tasks in the YAML file, and set triggers to build the project on every GitHub commit.

140. **What are GitHub Actions, and how do they differ from Azure Pipelines?**

- **Answer**: **GitHub Actions** is GitHub's built-in CI/CD service. While both GitHub Actions and Azure Pipelines provide similar CI/CD capabilities, GitHub Actions is integrated directly into GitHub, making it more seamless for GitHub projects. Azure Pipelines, on the other hand, offers a more enterprise-grade CI/CD tool with more advanced features like pipelines as code and integration with Azure services.

141. **How do you trigger an Azure DevOps pipeline from a GitHub webhook?**

- **Answer**:
  1. In your GitHub repository, go to **Settings** > **Webhooks**.
  2. Create a new webhook pointing to the Azure DevOps URL.
  3. Configure the webhook to trigger on specific events (e.g., push, pull request).
  4. In Azure DevOps, set up a **Service Connection** to GitHub to authenticate the webhook.

142. **What is GitHub Packages, and how do you integrate it with Azure DevOps?**

- **Answer**: **GitHub Packages** is a package hosting service integrated with GitHub repositories. You can publish and consume packages (e.g., npm, NuGet, Docker) directly from GitHub. To integrate it with Azure DevOps:
  1. Configure the pipeline to use the GitHub Packages registry for storing or retrieving packages.
  2. Example for npm:

```
steps:
 - script: |
```

npm install --registry https://npm.pkg.github.com/

143. **How do you handle GitHub pull requests in Azure Pipelines?**

- **Answer**: Azure Pipelines can be configured to automatically build and test changes in a GitHub repository when a pull request (PR) is created. To enable PR validation:
    1. In the pipeline YAML, set the pr trigger to listen for PR events.

```
trigger:
  branches:
    include:
      - main

pr:
  branches:
    include:
      - '*'
```

144. **How do you secure GitHub Actions or Azure Pipelines when deploying sensitive code?**

- **Answer**: To secure pipelines:
    - Use **secrets** to store sensitive data (e.g., tokens, keys).
    - Limit access to pipeline configurations and secrets.
    - Implement **branch protection rules** to ensure only approved changes can be merged into sensitive branches.
    - Require code reviews for pull requests before merging.

---

22. Azure DevOps Security, Compliance, and Governance

145. **How do you manage access to Azure DevOps pipelines?**

- **Answer**:
    1. Go to **Project Settings** > **Pipelines** > **Security**.
    2. Manage access permissions by assigning roles like **Contributor**, **Reader**, and **Pipeline Administrator**.
    3. Limit who can create or approve builds and deployments.

146. **How do you enforce security policies for pipelines in Azure DevOps?**

- **Answer**: Enforce security policies by:

- Requiring approval gates before deploying to sensitive environments (e.g., production).
- Using **Azure Policy** to enforce security controls on deployed infrastructure.
- Configuring branch policies to require code reviews and successful builds before merging code.

147. **What is Azure Policy, and how is it used in Azure DevOps?**

- **Answer**: **Azure Policy** helps enforce rules and ensure compliance for Azure resources. You can integrate Azure Policy with Azure DevOps by adding it as a step in pipelines to check if the infrastructure being deployed complies with your organization's policies.

148. **How do you handle role-based access control (RBAC) in Azure DevOps for different environments?**

- **Answer**:
  1. Use **Azure AD groups** to manage access for different environments (e.g., dev, staging, prod).
  2. Apply **RBAC** policies to limit which users can deploy to sensitive environments like production.
  3. Use environment-level controls in pipelines to restrict who can approve or execute deployments to production.

149. **How do you use Azure Key Vault in Azure Pipelines for secret management?**

- **Answer**:
  1. Set up an **Azure Key Vault** and store your secrets (e.g., API keys, passwords).
  2. In the pipeline, use the **Azure Key Vault task** to retrieve secrets and use them during the build or release process.
  3. Example:

```
steps:
 - task: AzureKeyVault@2
   inputs:
    azureSubscription: 'MyAzureSubscription'
    KeyVaultName: 'myKeyVault'
    SecretsFilter: '*'
```

150. **How do you ensure compliance and governance using Azure Blueprints in Azure DevOps?**

- **Answer**: **Azure Blueprints** allow you to define and automate the deployment of environments with predefined compliance rules and governance structures. In Azure DevOps, Blueprints can be used alongside ARM templates or Terraform to ensure all environments meet compliance requirements (e.g., security baselines, RBAC settings).

151. How do you implement custom approval gates in a pipeline?

- **Answer**: To implement custom approval gates in a pipeline:
    1. Add a **pre-deployment approval** step in your release pipeline.
    2. Define approval policies, including specific users or groups who must approve the deployment.
    3. You can also add automated approval gates like querying REST APIs or monitoring alerts.

---

152. What is the purpose of deployment conditions in Azure Pipelines?

- **Answer**: Deployment conditions control when and where deployments are triggered. Examples of conditions include:
    - Deploying only when previous stages succeed.
    - Deploying based on manual approval.
    - Using automated checks like health checks, alerts, or success criteria.

---

153. How do you configure pipeline triggers to only build when certain files are modified?

- **Answer**: You can configure triggers in a YAML pipeline to only fire when specific files are modified:

```
trigger:
  branches:
    include:
      - main
  paths:
    include:
      - src/**
    exclude:
      - docs/**
```

---

154. How do you manage pipeline dependencies between multiple repositories?

- **Answer**: In Azure DevOps, you can manage dependencies between repositories by:
    1. Using **multi-repo checkout** in the pipeline YAML to clone multiple repositories.

2. Defining dependencies between pipelines by triggering builds from one repo's pipeline to another.

---

155. What is a manual intervention task, and how do you configure it in a release pipeline?

- **Answer**: A **manual intervention task** is a pause in the release pipeline that requires human action before proceeding. To configure:
    1. Add a **manual intervention task** in the release pipeline.
    2. Define the approvers who can resume the pipeline.

---

156. How do you use caching in Azure Pipelines to speed up build times?

- **Answer**: You can use caching to store build dependencies or intermediate results and avoid downloading or rebuilding them during every pipeline run. Use the Cache task in YAML:

```
steps:
 - task: Cache@2
   inputs:
    key: 'npm | "$(Agent.OS)" | package-lock.json'
    path: '$(npm_cache)'
```

---

157. How do you set up a deployment pipeline for microservices in Azure DevOps?

- **Answer**:
    1. Create a pipeline with multiple stages for each microservice.
    2. Define deployment steps for each microservice individually (e.g., deploying to AKS or App Service).
    3. Use kubectl tasks or helm tasks to apply Kubernetes manifests for each service.

---

158. How do you manage rollbacks in Azure Pipelines?

- **Answer**: To manage rollbacks:
    1. Keep a record of artifacts or container images from successful builds.
    2. Use a **release pipeline** to deploy the previous known good version by selecting an earlier build artifact and deploying it.
    3. You can also use version tags on container images for quick rollbacks.

---

159. How do you trigger a pipeline based on a tag in Git?

- **Answer**: In a YAML pipeline, you can use the trigger keyword to specify that the pipeline should trigger on tags:

```
trigger:
  tags:
    include:
      - v*
```

---

160. How do you deploy a multi-container Docker application to AKS using Azure Pipelines?

- **Answer**:
    1. Build and push your Docker images to a container registry (e.g., Azure Container Registry).
    2. Use the kubectl or helm task in your pipeline to apply the Kubernetes deployment and service manifests for each container in the application.
    3. Example:

```
steps:
  - task: Docker@2
    inputs:
      command: 'buildAndPush'
      repository: 'myrepo/app'
      tags: '$(Build.BuildId)'

  - task: KubectlInstaller@0

  - task: Kubernetes@1
    inputs:
      command: 'apply'
      arguments: '-f deployment.yaml'
```

161. How do you configure the retry policy for failed tasks in Azure Pipelines?

- **Answer**: You can configure retry policies for pipeline tasks using the retry property in YAML:

```
steps:
  - script: echo "Running tests"
    retry:
```

```
  count: 3
  interval: 2
```

---

162. How do you implement parallel testing in Azure Pipelines?

- **Answer**:
    1. Split tests into separate jobs that can run in parallel.
    2. Use jobs in YAML to run tests simultaneously on multiple agents:

```
jobs:
 - job: Test1
   steps:
    - script: echo "Running Test 1"

 - job: Test2
   steps:
    - script: echo "Running Test 2"
```

---

163. How do you integrate AppDynamics with Azure Pipelines for monitoring?

- **Answer**:
    1. Install the **AppDynamics** agent on your application environment.
    2. Use the AppDynamics **REST API** or **Azure Monitor** integration to trigger alerts or gather metrics during the pipeline run.
    3. Example of a pipeline step using AppDynamics API:

```
steps:
 - script: |
   curl -X GET 'https://api.appdynamics.com/controller/rest/applications/YourApp'
```

---

164. What are task groups in Azure Pipelines, and how do you use them?

- **Answer**: **Task groups** allow you to group a set of pipeline tasks and reuse them across multiple pipelines. To use:
    1. Create a task group from an existing pipeline.
    2. Define input variables for the task group.
    3. Add the task group to other pipelines as needed.

---

165. How do you configure custom agents in Azure Pipelines to work with private networks?

- **Answer**: To configure custom (self-hosted) agents in a private network:
    1. Set up the agent VM inside your private network.
    2. Use a **VPN** or **private endpoint** to connect the agent to Azure DevOps.
    3. Install the agent on the VM and register it to the appropriate agent pool.

---

166. What is a service connection in Azure DevOps, and how is it used?

- **Answer**: A **service connection** in Azure DevOps allows pipelines to connect to external services (e.g., Azure, AWS, Docker Hub) securely. To use:
    1. Go to **Project Settings** > **Service Connections**.
    2. Create a new service connection (e.g., Azure Resource Manager, Docker).
    3. Use the connection in pipeline tasks to authenticate to the service.

167. How do you perform A/B testing in Azure App Service using Azure Pipelines?

- **Answer**:
    1. Deploy different versions of your application to separate deployment slots (e.g., version A to the main slot, version B to the staging slot).
    2. Use **Azure App Service Slot Swap** or routing rules to direct a portion of traffic to each version.
    3. Monitor the performance and behavior of both versions before finalizing the swap.

---

168. What is a Universal Package in Azure Artifacts, and how do you use it?

- **Answer**: A **Universal Package** is a package type in Azure Artifacts that can store arbitrary files. To use:
    1. Install the **Azure Artifacts CLI**.
    2. Publish a package:

az artifacts universal publish --organization "https://dev.azure.com/MyOrganization" --feed "MyFeed" --name "MyPackage" --version "1.0.0" --path "./myfiles"

---

169. How do you configure environment variables in Azure Pipelines?

- **Answer**: You can configure environment variables in the YAML pipeline or at the agent level. Example:

```
variables:
  MY_ENV_VAR: "production"

steps:
  - script: echo $(MY_ENV_VAR)
```

---

170. How do you run pipeline tasks in Docker containers using Azure Pipelines?

- **Answer**: To run tasks in a Docker container:
    1. Use the container keyword to specify the container image.
    2. Run the tasks inside the container. Example:

```
pool:
  vmImage: 'ubuntu-latest'

container: 'node:14'

steps:
  - script: npm install
  - script: npm test
```

---

171. How do you secure access to Docker registries in Azure Pipelines?

- **Answer**: Use a **service connection** to authenticate with Docker registries. For private Docker registries:
    1. Create a Docker registry service connection in Azure DevOps.
    2. Use the connection in Docker tasks to authenticate:

```
steps:
  - task: Docker@2
    inputs:
```

```
    containerRegistry: 'MyDockerRegistry'
    repository: 'myrepo/app'
    command: 'buildAndPush'
```

---

172. How do you enable pull request validation for an Azure Repo?

- **Answer**:
    1. Create a pipeline that triggers on pull requests.
    2. In YAML, use the pr trigger:

```
pr:
  branches:
    include:
      - '*'
```

    3. The pipeline will run automatically when a PR is created or updated.

---

173. How do you integrate Slack with Azure Pipelines for notifications?

- **Answer**: You can use the **Slack Notifications** task in Azure Pipelines to send messages to Slack channels. Example:

```
steps:
  - task: SlackNotification@1
    inputs:
      slackConnection: 'MySlackConnection'
      channel: '#build-status'
      message: 'Build $(Build.BuildNumber) completed successfully'
```

174. How do you store secrets securely in Azure Pipelines?

- **Answer**: Use the following methods to store secrets securely:
    1. **Pipeline variables** marked as secret: These values are masked in logs.
    2. **Azure Key Vault** integration: Fetch secrets securely during pipeline execution.
    3. **Secure files**: Store certificates and sensitive files securely and access them in pipelines.

175. What is the difference between static and dynamic pipelines in Azure DevOps?

- **Answer**:
    - **Static pipelines** have a fixed set of tasks and steps.
    - **Dynamic pipelines** can adapt based on variables, conditions, or runtime inputs, allowing for more flexible workflows depending on the context (e.g., branch, environment).

176. How do you use Helm charts in Azure Pipelines to deploy to Kubernetes?

- **Answer**:
    1. Use the **Helm Installer** task to install Helm on the agent.
    2. Use the **Helm Upgrade** task to deploy a Helm chart to Kubernetes:

```
steps:
 - task: HelmInstaller@1
   inputs:
    helmVersion: 'latest'

 - task: HelmDeploy@0
   inputs:
    connectionType: 'Kubernetes Service Connection'
    kubernetesServiceEndpoint: 'MyKubeEndpoint'
    chartType: 'FilePath'
    chartPath: 'charts/mychart'
    releaseName: 'myrelease'
```

---

177. How do you implement continuous deployment with GitHub Actions and Azure Pipelines?

- **Answer**: To integrate GitHub Actions with Azure Pipelines for continuous deployment:
    1. Create a GitHub Action to trigger an Azure Pipeline via a webhook or API.
    2. Set the GitHub Action to run whenever code is pushed or merged into the main branch.
    3. In the Action YAML, use:

```
name: Trigger Azure Pipeline

on:
  push:
    branches:
      - main

jobs:
  runAzurePipeline:
    runs-on: ubuntu-latest
    steps:
      - name: Trigger Azure DevOps Pipeline
        run: |
          curl -X POST https://dev.azure.com/{organization}/{project}/_apis/build/builds?api-version=6.0
\
          -H "Authorization: Bearer ${{ secrets.AZURE_TOKEN }}" \
          -d '{"definition": {"id": "1"}}'
```

---

178. How do you use conditionals to skip certain pipeline tasks?

- **Answer**: In YAML pipelines, use the condition keyword to control whether a task should be executed:

```
steps:
  - script: echo "Running tests"
    condition: and(succeeded(), eq(variables['Build.SourceBranch'], 'refs/heads/main'))
```

---

179. How do you configure cross-project pipelines in Azure DevOps?

- **Answer**: To trigger a pipeline in one project based on the outcome of a pipeline in another project:
    1. Use the **Pipeline Resource** trigger to link pipelines across projects.
    2. Example:

```
resources:
  pipelines:
```

```
- pipeline: mypipeline
  source: myproject/mybuild
```

180. How do you manage deployment approvals for different teams in Azure DevOps?

- **Answer**:
    1. Use **environment approvals** to assign specific teams or users as approvers for different stages.
    2. In a release pipeline, configure the **Pre-deployment approvals** section under each environment.
    3. Add specific users or Azure AD groups as approvers.

181. How do you debug a pipeline failure in Azure DevOps?

- **Answer**: To debug a pipeline failure:
    1. Review the **pipeline logs** for errors or warnings.
    2. Enable **diagnostic logs** for more verbose output.
    3. Use **step-level outputs** and environment variables to troubleshoot issues.
    4. Check **Azure DevOps Auditing** to track recent changes that might have impacted the pipeline.

182. How do you configure build retention policies in Azure DevOps?

- **Answer**:
    1. Go to **Pipelines** > **Retention**.
    2. Define retention policies for build artifacts, logs, and test results based on age, number of builds, or outcome.
    3. Apply the policy to a specific pipeline or all pipelines.

183. How do you set up and use Azure DevTest Labs with Azure Pipelines?

- **Answer**: **Azure DevTest Labs** provides development and test environments. To use it in a pipeline:
    1. Create a DevTest Lab in the Azure portal.
    2. Use the **Azure DevTest Labs VM** task in Azure Pipelines to create VMs in the lab:

```
steps:
  - task: AzureDevTestLabsVM@1
```

```
  inputs:
    labName: 'MyDevTestLab'
    vmName: 'myVM'
```

184. How do you implement automatic rollback in Azure Pipelines if a deployment fails?

- **Answer**: To implement automatic rollback:
    1. Use a **post-deployment trigger** in your release pipeline that monitors the deployment for issues.
    2. If a failure is detected, redeploy the previous stable build artifact.

185. What is Canary Deployment, and how do you implement it in Azure Pipelines?

- **Answer**: **Canary deployment** involves gradually rolling out a new version of the application to a subset of users while keeping the previous version live. In Azure Pipelines:
    1. Use deployment slots (App Service) or label-based routing (AKS) to direct a small percentage of traffic to the new version.
    2. Monitor the performance and gradually increase traffic if successful.

186. How do you customize pipeline notifications in Azure DevOps?

- **Answer**:
    1. Go to **Project Settings** > **Notifications**.
    2. Customize notifications by selecting specific pipeline events (e.g., build failures, test results).
    3. Configure email recipients, Slack channels, or webhook URLs for notifications.

187. How do you run security scanning as part of a CI pipeline in Azure Pipelines?

- **Answer**: You can integrate tools like **OWASP ZAP**, **Trivy**, or **WhiteSource Bolt** into your pipeline for security scanning:

```
steps:
  - script: |
      trivy filesystem /path/to/project
  - task: OWASPZAP@2
    inputs:
      targetUrl: 'http://myappurl'
```

---

188. How do you set up a multi-cloud deployment pipeline using Azure DevOps?

- **Answer**: To set up a multi-cloud deployment pipeline:
    1. Create separate stages for each cloud provider (Azure, AWS, GCP).
    2. Use service connections for authentication with each provider.
    3. Example with Azure and AWS:

```
stages:
  - stage: DeployToAzure
    jobs:
      - job: AzureDeploy
        steps:
          - task: AzureCLI@2
            inputs:
              script: "az deployment create"

  - stage: DeployToAWS
    jobs:
      - job: AWSDeploy
        steps:
          - task: AWSCLI@1
            inputs:
              awsServiceEndpoint: 'AWSConnection'
              script: "aws s3 sync . s3://mybucket"
```

---

189. How do you manage pipeline concurrency in Azure DevOps?

- **Answer**: Pipeline concurrency can be managed using the **exclusive lock** mechanism:
    1. In the pipeline YAML, use jobs with the dependsOn keyword to control the execution order.
    2. Example:

```
jobs:
  - job: Build
    steps:
```

```
    - script: echo "Building..."

  - job: Test
    dependsOn: Build
    steps:
      - script: echo "Testing..."
```

---

190. How do you configure access control to Azure Pipelines?

- **Answer**:
    1. Use **role-based access control (RBAC)** to assign roles such as **Reader**, **Contributor**, or **Administrator**.
    2. Configure pipeline-specific security by managing permissions under **Pipeline Settings** > **Security**.
    3. Limit access to environments, pipelines, and repositories by assigning users or groups to roles.

---

191. How do you set up multi-stage deployments across multiple regions using Azure Pipelines?

- **Answer**:
    1. Create a pipeline with separate stages for each region (e.g., East US, West Europe).
    2. Deploy to each region in parallel using **Azure CLI** or **ARM templates**.
    3. Example:

```
stages:
  - stage: DeployToEastUS
    jobs:
      - job: Deploy
        steps:
          - task: AzureCLI@2
            inputs:
```

```
        script: "az deployment group create --resource-group MyResourceGroup --template-file azuredeploy.json --parameters location=eastus"

  - stage: DeployToWestEurope
    jobs:
      - job: Deploy
        steps:
          - task: AzureCLI@2
            inputs:
        script: "az deployment group create --resource-group MyResourceGroup --template-file azuredeploy.json --parameters location=westeurope"
```

192. How do you manage cross-team collaboration in Azure DevOps?

- **Answer**:
    1. Use **Azure Boards** to track work items across teams.
    2. Create **shared repositories** and pipelines for cross-team collaboration.
    3. Implement **branch policies** to ensure proper review and approval workflows before merging code.
    4. Use **Wiki pages** and **dashboards** to share documentation and metrics across teams.

193. How do you automate package versioning in Azure Pipelines?

- **Answer**:
    1. Use pipeline variables and tasks to automatically increment version numbers.
    2. Example using GitVersion:

```
steps:
  - task: GitVersion@5
    inputs:
      updateAssemblyInfo: true
  - script: echo "##vso[build.updatebuildnumber]$(GitVersion.FullSemVer)"
```

---

194. How do you implement static code analysis in Azure Pipelines?

- **Answer**: You can use **SonarQube** for static code analysis:
    1. Add the **SonarQube Prepare Analysis Configuration** task to the pipeline.
    2. Run the code analysis as part of the build pipeline:

```
steps:
 - task: SonarQubePrepare@4
   inputs:
    SonarQube: 'MySonarQubeConnection'
    projectKey: 'myproject'

 - script: ./build.sh

 - task: SonarQubeAnalyze@4
```

195. How do you implement feature flags in Azure DevOps pipelines?

- **Answer**:
    1. Use feature flag management tools (e.g., **LaunchDarkly** or **Azure App Configuration**) integrated with your pipelines.
    2. Manage the feature flag rollout in the pipeline based on the environment, branch, or user group.

196. How do you manage dependencies between multiple pipelines in Azure DevOps?

- **Answer**: Use **Pipeline Resources** to trigger pipelines based on the outcome of other pipelines. Example:

```
resources:
 pipelines:
  - pipeline: 'upstream-pipeline'
    source: 'upstream-repo'
    trigger:
     branches:
      include:
       - main
```

197. How do you automate the release of artifacts in Azure Pipelines?

- **Answer**: To automate artifact release:
    1. Use the **Publish Artifact** task in your build pipeline to store artifacts.
    2. Use **Azure Artifacts** to store and version the artifacts.
    3. In the release pipeline, trigger deployment of the latest artifact.

198. How do you manage pipeline secrets in Azure Pipelines?

- **Answer**: You can manage secrets in pipelines by:
    - Storing secrets in **Azure Key Vault** and accessing them using the **Key Vault task**.
    - Using **pipeline variables** and marking them as secrets.
    - Limiting access to secrets by restricting pipeline permissions.

199. How do you implement continuous testing in Azure Pipelines?

- **Answer**: Continuous testing involves running automated tests (unit, integration, functional) as part of the pipeline:
    1. Add tasks to execute tests for each stage of the pipeline (e.g., unit tests during build, integration tests during deployment).
    2. Use **test reports** and **test plans** to track test outcomes.

200. How do you configure custom notifications for pipeline events?

- **Answer**:
    1. Go to **Project Settings** > **Notifications**.
    2. Create a custom notification rule for pipeline events (e.g., build success, release failure).
    3. Choose the recipients for notifications (e.g., email, Slack, Teams, webhook).

201. How do you implement infrastructure as code (IaC) with Azure DevOps using Terraform?

- **Answer**: To implement IaC using **Terraform**:
    1. Store Terraform configuration files (e.g., main.tf) in Azure Repos.

2. Use a pipeline to run Terraform commands (init, plan, apply) for resource creation.
3. Example pipeline YAML for Terraform:

```
trigger:
  branches:
    include:
      - main

pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: TerraformInstaller@0
    inputs:
      terraformVersion: '0.12.x'
  - script: |
      terraform init
      terraform plan
      terraform apply -auto-approve
```

---

202. How do you use Azure DevOps for disaster recovery automation?

- **Answer**: To automate disaster recovery in Azure DevOps:
    1. Use **Azure Site Recovery** for setting up DR between Azure regions or on-prem to Azure.
    2. Create pipelines that back up critical data and infrastructure (e.g., VMs, databases).
    3. Automate recovery using **ARM templates**, **Terraform**, or **Bicep** for infrastructure re-deployment.

---

203. How do you enable auditing in Azure DevOps?

- **Answer**: **Azure DevOps Auditing** helps track changes and user activities within the organization. To enable auditing:
    1. Navigate to **Organization Settings** > **Auditing**.

2. You can filter events such as changes in permissions, pipeline runs, repository changes, etc.
3. Export logs or connect them to **Azure Monitor** for better reporting.

---

204. What is Azure Blueprints, and how can you use it with Azure Pipelines?

- **Answer**: **Azure Blueprints** allows organizations to define and manage infrastructure and governance policies. You can integrate Azure Blueprints with Azure Pipelines to automate compliance:
    1. Use an ARM template or Terraform for provisioning resources.
    2. Apply the **Blueprints** policy in the pipeline for each environment to ensure compliance.
    3. Example using ARM:

```
steps:
  - task: AzureCLI@2
    inputs:
      azureSubscription: 'MyAzureSubscription'
      scriptType: 'bash'
      scriptLocation: 'inlineScript'
      inlineScript: |
        az blueprint assignment create --name 'MyBlueprint' --location 'EastUS'
```

---

205. How do you configure Azure Pipelines for rolling updates in AKS (Azure Kubernetes Service)?

- **Answer**: For **rolling updates** in AKS:
    1. Use a **Kubernetes deployment** YAML file that specifies a **RollingUpdate** strategy.
    2. Configure your Azure Pipeline to apply this YAML file using the kubectl task.
    3. Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 3
```

```
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxUnavailable: 1
    maxSurge: 1
```

---

206. How do you handle database migrations in Azure DevOps pipelines?

- **Answer**: To handle database migrations:
    1. Use migration tools like **Entity Framework** (for .NET), **Flyway**, or **Liquibase** to create migrations.
    2. Add a step in your build or release pipeline to run the migration scripts before deploying the application.
    3. Example:

```
steps:
- script: |
    dotnet ef database update
```

---

207. How do you implement a deployment strategy using deployment slots in Azure App Service?

- **Answer**: **Deployment slots** in Azure App Service allow zero-downtime deployments. To implement this:
    1. Use separate slots for production and staging.
    2. Deploy the new version to the **staging slot**.
    3. After testing, swap the **staging slot** with the **production slot** using the **Slot Swap** task in Azure Pipelines.

---

208. How do you set up a CI/CD pipeline for Azure Static Web Apps?

- **Answer**: To set up a CI/CD pipeline for **Azure Static Web Apps**:
    1. Use Azure Pipelines or GitHub Actions.
    2. Use the azure/static-web-apps-deploy action for building and deploying a static web app.
    3. Example GitHub Action:

```
name: Azure Static Web Apps CI/CD
```

```
on:
  push:
    branches:
      - main

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Build
        run: npm run build
      - name: Deploy
        uses: Azure/static-web-apps-deploy@v0.0.1-preview
        with:
          azure_static_web_apps_api_token: ${{ secrets.AZURE_STATIC_WEB_APPS_API_TOKEN }}
```

---

209. What are release annotations in Azure Pipelines, and how do you use them?

- **Answer**: **Release annotations** in Azure Pipelines allow you to mark important events (e.g., deployments) in monitoring tools like **Application Insights**. To use them:
    1. Add the **Application Insights release annotation** task in your pipeline.
    2. Provide details such as the release ID, build number, and environment.

---

210. How do you automate infrastructure monitoring with Prometheus and Grafana in Azure Pipelines?

- **Answer**:
    1. Set up **Prometheus** and **Grafana** in your Kubernetes environment.
    2. Create a pipeline that installs or updates the Prometheus and Grafana Helm charts.
    3. Use the **Helm task** to deploy or update monitoring tools.

---

211. How do you implement Azure AD authentication in your Azure DevOps pipelines?

- **Answer**: To use **Azure AD authentication**:

1. Use **Azure AD App Registrations** to create a service principal.
2. Use the **Azure CLI task** or **ARM template deployment** task in the pipeline to authenticate using the service principal.
3. Example:

```
steps:
 - task: AzureCLI@2
   inputs:
    azureSubscription: 'MyAzureSubscription'
    scriptType: 'bash'
    scriptLocation: 'inlineScript'
    inlineScript: |
     az login --service-principal -u $(ClientId) -p $(ClientSecret) --tenant $(TenantId)
```

---

212. How do you use variable groups in Azure DevOps pipelines?

- **Answer**: **Variable groups** are collections of variables that can be shared across multiple pipelines. To use:
  1. Go to **Pipelines** > **Library** and create a new variable group.
  2. In your pipeline YAML, link the variable group:

```
variables:
 - group: MyVariableGroup
```

---

213. How do you configure build and release approvals in Azure Pipelines?

- **Answer**:
  1. Navigate to the **Environments** section of your release pipeline.
  2. Under **Pre-deployment conditions**, enable **Approvals**.
  3. Select users or groups who must approve the release before deployment can continue.

---

214. How do you manage multiple Kubernetes clusters in Azure DevOps?

- **Answer**:
  1. Use different **Kubernetes service connections** for each cluster.

2. In your pipeline, specify which connection to use when deploying to different clusters:

```
steps:
 - task: Kubernetes@1
   inputs:
     connectionType: 'Azure Resource Manager'
     kubernetesCluster: 'ClusterA'
```

---

215. How do you build and deploy a React application using Azure Pipelines?

- **Answer**: To build and deploy a React app:
    1. Use **npm** to build the React app.
    2. Deploy the build files to **Azure App Service** or **Azure Static Web Apps**.
    3. Example pipeline:

```
pool:
 vmImage: 'ubuntu-latest'

steps:
 - task: NodeTool@0
   inputs:
     versionSpec: '12.x'
 - script: npm install
 - script: npm run build
 - task: AzureWebApp@1
   inputs:
     azureSubscription: 'MyAzureSubscription'
     appName: 'my-react-app'
     package: '$(Build.ArtifactStagingDirectory)/**/build.zip'
```

---

216. How do you integrate Azure DevOps with Jira for issue tracking?

- **Answer**: To integrate Azure DevOps with Jira:
    1. Use the **Azure Pipelines for Jira** integration available in the Jira marketplace.

2. Configure the integration to sync work items between Azure Boards and Jira issues.

---

217. What is a retention lease in Azure Pipelines, and how do you use it?

- **Answer**: **Retention leases** allow you to keep specific builds, releases, or artifacts longer than the default retention policy. To use:
    1. Navigate to the **Pipelines** or **Releases** tab.
    2. Select a pipeline or release, then click on **Retain** to prevent it from being deleted automatically.

---

218. How do you monitor Azure DevOps pipelines using Application Insights?

- **Answer**:
    1. Set up **Application Insights** for your application environment.
    2. Add **release annotations** from Azure Pipelines to track deployments.
    3. Monitor custom metrics such as success rate and duration for pipeline runs.

---

219. How do you use Azure Repos to manage large monorepos in Azure DevOps?

- **Answer**:
    1. Use **Git submodules** to divide the monorepo into smaller projects.
    2. Configure pipelines to only build the affected parts of the monorepo, using path filters in triggers.

---

220. How do you implement automated performance testing in Azure Pipelines?

- **Answer**:
    1. Use performance testing tools like **JMeter** or **K6**.
    2. Add a task in the pipeline to run performance tests after the build step.
    3. Example using JMeter:

```
steps:
  - task: CmdLine@2
    inputs:
      script: |
        jmeter -n -t test-plan.jmx -l results.jtl
```

---

221. How do you build and deploy a .NET Core application using Azure Pipelines?

- **Answer**:
    1. Use the **DotNetCoreCLI** task to build and publish the application.
    2. Deploy to **Azure App Service** or **Azure Kubernetes Service**.
    3. Example YAML:

```
pool:
  vmImage: 'windows-latest'

steps:
  - task: UseDotNet@2
    inputs:
      packageType: 'sdk'
      version: '3.x'

  - task: DotNetCoreCLI@2
    inputs:
      command: 'build'
      projects: '**/*.csproj'

  - task: AzureWebApp@1
    inputs:
      azureSubscription: 'MyAzureSubscription'
      appName: 'my-dotnet-app'
      package: '$(Build.ArtifactStagingDirectory)/drop.zip'
```

---

222. How do you handle conditional deployment to multiple environments in Azure Pipelines?

- **Answer**:
    1. Use conditions in your pipeline to control deployment based on variables or other pipeline outcomes.
    2. Example:

```
stages:
```

```
- stage: DeployToDev
  jobs:
    - job: Deploy
      condition: eq(variables['Build.SourceBranch'], 'refs/heads/develop')

- stage: DeployToProd
  jobs:
    - job: Deploy
      condition: eq(variables['Build.SourceBranch'], 'refs/heads/main')
```

---

223. How do you secure Azure Pipelines using RBAC and policies?

- **Answer**: Use **role-based access control (RBAC)** to manage who can access, modify, or trigger pipelines. Policies such as branch protection and pipeline triggers ensure that only authorized code is deployed.

---

224. How do you track test results over time in Azure Pipelines?

- **Answer**: Use the **Test tab** in Azure Pipelines to view trends in test results, failures, and performance over time. You can also generate test result reports for stakeholders.

---

225. How do you manage pipeline concurrency in Azure DevOps?

- **Answer**: Use **runOnce** or **multi-job** strategies to limit the number of concurrent jobs. You can also limit the number of pipeline instances allowed to run simultaneously using **exclusive lock**.

---

226. How do you manage feature toggles in a CI/CD pipeline?

- **Answer**:
    1. Use **feature flag management** tools (e.g., LaunchDarkly, Azure App Configuration).
    2. Integrate feature toggles with your pipeline to control feature availability during different deployment stages.

---

227. How do you configure rollback in case of deployment failure in Azure Pipelines?

- **Answer**:
    1. Store the previous stable build artifact.

2. In case of failure, add a conditional task to roll back to the previous release by deploying the stable artifact.

---

228. How do you implement smoke tests in a release pipeline?

- **Answer**:
    1. Add a post-deployment stage in your pipeline to run smoke tests.
    2. Fail the pipeline if smoke tests do not pass.
    3. Example:

```
steps:
 - task: CmdLine@2
   inputs:
    script: |
     curl -I https://myappurl.com
```

---

229. How do you manage sensitive information like API keys in Azure Pipelines?

- **Answer**: Store sensitive information such as API keys in **Azure Key Vault** or **pipeline variables** marked as secrets. Ensure that the secrets are only accessible by authorized users and stages in the pipeline.

---

230. How do you configure Azure Pipelines to deploy to Azure Kubernetes Service (AKS)?

- **Answer**:
    1. Create a **Kubernetes service connection** in Azure DevOps.
    2. Use kubectl tasks in your pipeline to apply Kubernetes manifests to your AKS cluster.
    3. Example:

```
steps:
 - task: KubectlInstaller@0
 - task: Kubernetes@1
   inputs:
    connectionType: 'Azure Resource Manager'
    kubernetesServiceEndpoint: 'MyAKSCluster'
```

```
    command: 'apply'
    arguments: '-f k8s/deployment.yaml'
```

---

231. How do you deploy a Python application to Azure App Service using Azure Pipelines?

- **Answer**:
    1. Use the **Azure Web App for Python** task to deploy Python code to App Service.
    2. Example:

```
pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: UsePythonVersion@0
    inputs:
      versionSpec: '3.x'

  - script: pip install -r requirements.txt
  - script: python manage.py collectstatic --noinput
  - task: AzureWebApp@1
    inputs:
      azureSubscription: 'MyAzureSubscription'
      appName: 'my-python-app'
      package: '$(Build.ArtifactStagingDirectory)/drop.zip'
```

---

232. How do you integrate SonarCloud for code quality analysis in Azure Pipelines?

- **Answer**:
    1. Install the **SonarCloud** extension from the marketplace.
    2. Add the **SonarCloudPrepare** and **SonarCloudAnalyze** tasks to your pipeline.
    3. Example:

```
steps:
  - task: SonarCloudPrepare@1
    inputs:
      SonarCloud: 'SonarCloudConnection'
      organization: 'MyOrg'
      projectKey: 'myproject'

  - script: ./build.sh

  - task: SonarCloudAnalyze@1
```

---

233. How do you implement infrastructure as code with Bicep in Azure DevOps?

- **Answer**:

1. Store your **Bicep** files in an Azure Repo or GitHub.
2. Use the **Azure CLI task** to deploy Bicep templates:

```
steps:
 - task: AzureCLI@2
   inputs:
    scriptType: 'bash'
    scriptLocation: 'inlineScript'
    inlineScript: |
     az deployment group create --resource-group MyResourceGroup --template-file main.bicep
```

---

234. How do you deploy serverless applications using Azure Pipelines?

- **Answer**:
  1. Use **Azure Functions** or **AWS Lambda** to deploy serverless apps.
  2. In Azure Pipelines, use the **Azure Functions App task** to deploy code to Azure Functions.

---

235. How do you use multi-repo checkout in Azure Pipelines?

- **Answer**: Multi-repo checkout allows you to pull code from multiple repositories in a single pipeline. Example:

```
resources:
 repositories:
   - repository: repo1
     type: git
     name: MyOrg/Repo1

   - repository: repo2
     type: git
     name: MyOrg/Repo2

steps:
 - checkout: repo1
 - checkout: repo2
```

---

236. How do you configure health checks in a pipeline for a web application?

- **Answer**:
    1. Add a post-deployment task that runs health check commands (e.g., curl).
    2. Fail the pipeline if the health check does not pass.

---

237. How do you integrate Azure Pipelines with Jenkins for CI/CD?

- **Answer**: You can trigger Jenkins jobs from Azure Pipelines and vice versa:
    1. Use the **Jenkins Queue Job** task in Azure Pipelines to trigger a Jenkins job.
    2. Use the **Jenkins Integration** service in Azure DevOps to trigger builds from Jenkins.

---

238. How do you configure Azure Artifacts for package management in Azure Pipelines?

- **Answer**:
    1. Create an **Azure Artifacts feed** to store packages (e.g., npm, NuGet, Maven).
    2. Add the **Publish Pipeline Artifact** task to push packages to the feed.

---

239. How do you set up and manage Azure Boards for project tracking?

- **Answer**: Use **Azure Boards** to create and manage work items, sprints, backlogs, and epics. You can integrate Boards with GitHub or Azure Repos for automatic linking of commits and pull requests to work items.

---

240. How do you implement gated check-ins in Azure Pipelines?

- **Answer**: **Gated check-ins** prevent changes from being merged until a build validation passes. To configure:
    1. Enable branch policies for gated check-ins.
    2. Configure the pipeline to trigger on pull request creation, ensuring it passes all required validations.

241. How do you manage feature branches in Azure Repos using branch policies?

- **Answer**:
    1. In **Azure Repos**, set up **branch policies** to enforce rules on feature branches.
    2. Use policies like **required reviewers**, **work item linking**, and **build validation** to ensure quality and compliance before merging into the main branch.

---

242. How do you perform API testing using Azure Pipelines?

- **Answer**: You can integrate API testing frameworks like **Postman** or **RestAssured** into Azure Pipelines:
    1. Add a task to execute the API tests as part of the pipeline.
    2. Example using Postman's Newman CLI:

```
steps:
 - task: NodeTool@0
   inputs:
    versionSpec: '12.x'

 - script: |
    npm install -g newman
    newman run collection.json
```

---

243. How do you configure build validation for pull requests in Azure Repos?

- **Answer**:
    1. In **Branch policies**, set **build validation** to automatically trigger a build when a pull request is created.
    2. This ensures that the code passes all the necessary CI checks before merging.

---

244. How do you configure time-based triggers in Azure Pipelines?

- **Answer**: Use **scheduled triggers** in YAML pipelines to run builds at specific times:

```
schedules:
 - cron: "0 0 * * *"
   displayName: "Nightly Build"
   branches:
    include:
     - main
```

---

245. How do you implement chaos engineering in Azure Pipelines?

- **Answer**:

1. Use tools like **Chaos Mesh** or **Gremlin** to introduce controlled failures in the system.
2. Create tasks in the pipeline to simulate failures in production environments and monitor system behavior.
3. Example with Chaos Mesh:

```
steps:
 - script: |
    kubectl apply -f chaos-experiment.yaml
```

---

246. How do you implement canary releases in Azure Pipelines using Azure Traffic Manager?

- **Answer**:
  1. Set up two deployment slots (e.g., **production** and **canary**).
  2. Use **Azure Traffic Manager** to route a percentage of traffic to the canary slot.
  3. Gradually increase traffic to the canary deployment, and monitor the results before fully deploying.

---

247. How do you use pipeline decorators in Azure DevOps?

- **Answer**: **Pipeline decorators** allow you to inject steps into all pipelines in an organization:
  1. Create a pipeline decorator in YAML or JSON format.
  2. Use it to enforce certain steps, such as security scans, across all pipelines.

---

248. How do you automate infrastructure deployment using ARM templates in Azure Pipelines?

- **Answer**:
  1. Store the **ARM templates** in an Azure Repo.
  2. Use the **AzureResourceManagerTemplateDeployment** task in the pipeline to deploy the templates.
  3. Example:

```
steps:
 - task: AzureResourceManagerTemplateDeployment@3
   inputs:
    deploymentScope: 'Resource Group'
```

```
    azureResourceManagerConnection: 'MyAzureConnection'
    csmFile: '$(System.DefaultWorkingDirectory)/arm-template.json'
    resourceGroupName: 'my-resource-group'
```

---

249. How do you handle versioning of Azure Functions in Azure Pipelines?

- **Answer**:
    1. Use **semver** (semantic versioning) or date-based versioning in your pipeline.
    2. Tag the build with the version number and pass it to the deployment task.
    3. Example:

```
steps:
 - script: echo "##vso[build.updatebuildnumber]1.0.$(Build.BuildId)"
 - task: AzureFunctionApp@1
   inputs:
    azureSubscription: 'MyAzureSubscription'
    appName: 'my-function-app'
    package: '$(Build.ArtifactStagingDirectory)/drop.zip'
    deploymentSlot: 'production'
```

---

250. How do you manage service principal secrets in Azure Pipelines?

- **Answer**:
    1. Store service principal secrets in **Azure Key Vault** or **pipeline variables** marked as secret.
    2. Use the **AzureCLI@2** task to authenticate with the service principal:

```
steps:
 - task: AzureCLI@2
   inputs:
    azureSubscription: 'MyAzureSubscription'
    scriptType: 'bash'
    scriptLocation: 'inlineScript'
    inlineScript: |
```

```
az login --service-principal -u $(ClientId) -p $(ClientSecret) --tenant $(TenantId)
```

---

251. How do you configure multi-stage approvals in Azure Pipelines?

- **Answer**:
    1. Set up **approvals** for each stage in a multi-stage pipeline.
    2. Define which user(s) or group(s) need to approve before moving to the next stage.
    3. Example:

```
stages:
  - stage: Build
    jobs:
      - job: BuildJob

  - stage: DeployToProd
    dependsOn: Build
    jobs:
      - job: Deploy
        steps:
          - script: echo "Deploying to Production"
        approval:
          steps:
            - approve: ${{ parameters.approvers }}
```

---

252. How do you use Bicep templates for infrastructure deployment in Azure Pipelines?

- **Answer**:
    1. Store **Bicep templates** in Azure Repos or GitHub.
    2. Use the **AzureCLI task** to compile and deploy the Bicep templates:

```
steps:
  - task: AzureCLI@2
    inputs:
      scriptType: 'bash'
      scriptLocation: 'inlineScript'
```

```
    inlineScript: |
      az deployment group create --resource-group myResourceGroup --template-file main.bicep
```

---

253. How do you implement database backups in Azure Pipelines?

- **Answer**:
    1. Use **Azure CLI** or **PowerShell** to trigger database backups during the pipeline.
    2. Example using Azure CLI for SQL Database backup:

```
steps:
  - task: AzureCLI@2
    inputs:
      azureSubscription: 'MyAzureSubscription'
      scriptType: 'bash'
      scriptLocation: 'inlineScript'
      inlineScript: |
        az sql db export --resource-group myResourceGroup --server myServer --name myDatabase --admin-user myAdmin --admin-password myPassword --storage-uri https://storageaccount.blob.core.windows.net/backups/myDatabase.bacpac
```

---

254. How do you perform automated rollback in Azure Pipelines?

- **Answer**:
    1. Store the previous successful release artifact.
    2. Add logic to your pipeline to automatically deploy the previous artifact in case of deployment failure:

```
steps:
  - task: DownloadPipelineArtifact@2
    inputs:
      buildType: 'specific'
      project: 'MyProject'
      pipeline: 'MyPipeline'
      artifactName: 'drop'
  - task: DeployToProd@1
    inputs:
```

```
    package: '$(Pipeline.Workspace)/drop/previous_artifact.zip'
```

---

255. How do you configure notification groups in Azure Pipelines?

- **Answer**:
    1. Go to **Project Settings** > **Notifications**.
    2. Create notification groups for specific pipeline events (e.g., build completion, release failure).
    3. Assign users or groups to receive notifications for each event.

---

256. How do you use templates in Azure Pipelines for reusability?

- **Answer**:
    1. Create reusable templates for common pipeline steps.
    2. Reference the templates in different pipelines.
    3. Example template for testing:

```
# test-template.yml
steps:
  - script: |
      echo "Running tests..."
      npm run test
```

Example pipeline using the template:

```
jobs:
  - template: test-template.yml
```

---

257. How do you enable pipeline caching for Maven dependencies in Azure Pipelines?

- **Answer**:
    1. Use the **Cache task** to store Maven dependencies between pipeline runs.
    2. Example:

```
steps:
  - task: Cache@2
    inputs:
      key: 'maven | "$(Agent.OS)" | **/pom.xml'
```

```
    path: '~/.m2/repository'
```

---

258. How do you automate security scanning with OWASP ZAP in Azure Pipelines?

- **Answer**:
    1. Use the **OWASP ZAP** task or run ZAP as a command-line tool in your pipeline.
    2. Example:

```
steps:
 - task: OWASPZAP@2
   inputs:
    targetUrl: 'https://myapp.com'
    scanMode: 'full'
```

---

259. How do you trigger Azure Pipelines from an external system via webhook?

- **Answer**:
    1. Set up a **Service Hook** in Azure DevOps to trigger a pipeline on specific events (e.g., Git push).
    2. Use the external system's webhook to trigger the pipeline via the Azure DevOps REST API.

---

260. How do you manage environment variables for different stages in Azure Pipelines?

- **Answer**:
    1. Define environment-specific variables in each stage of your pipeline.
    2. Use variables or variable groups to share values across stages.
    3. Example:

```
stages:
 - stage: Build
   jobs:
    - job: BuildJob
      variables:
       NODE_ENV: 'development'


 - stage: DeployToProd
```

```
jobs:
 - job: DeployJob
  variables:
   NODE_ENV: 'production'
```

---

261. How do you enforce policies on pull requests in Azure Repos?

- **Answer**:
    1. Use **branch policies** to enforce rules like mandatory reviewers, build validation, and work item linking.
    2. You can also require that certain tests or security scans pass before merging a PR.

---

262. How do you deploy an Angular application to Azure Static Web Apps using Azure Pipelines?

- **Answer**:
    1. Build the Angular app using **npm**.
    2. Deploy the built app to Azure Static Web Apps using the AzureStaticWebApp task.
    3. Example:

```
steps:
 - task: NodeTool@0
  inputs:
   versionSpec: '12.x'
 - script: npm install
 - script: npm run build
 - task: AzureStaticWebApp@0
  inputs:
   azureSubscription: 'MyAzureSubscription'
   appLocation: 'src'
   outputLocation: 'dist/myapp'
```

---

263. How do you manage cross-platform builds in Azure Pipelines?

- **Answer**:
    1. Use different agent pools (e.g., Windows, Linux, macOS) for cross-platform builds.

2. Use conditional tasks to run platform-specific build steps.
3. Example:

```
jobs:
  - job: BuildLinux
    pool:
      vmImage: 'ubuntu-latest'
    steps:
      - script: echo "Building on Linux"
  - job: BuildWindows
    pool:
      vmImage: 'windows-latest'
    steps:
      - script: echo "Building on Windows"
```

---

264. How do you implement policy-driven deployments using Azure Policy in Azure Pipelines?

- **Answer**:
  1. Use **Azure Policy** to define rules and governance for deployed resources.
  2. In your pipeline, validate that deployments comply with Azure Policy before applying them.
  3. Example:

```
steps:
  - task: AzureCLI@2
    inputs:
      azureSubscription: 'MyAzureSubscription'
      scriptType: 'bash'
      scriptLocation: 'inlineScript'
      inlineScript: |
        az policy state list --policy-name "SecurityPolicy"
```

---

265. How do you configure Azure Pipelines to work with private container registries?

- **Answer**:

1. Use a **Docker registry service connection** to authenticate with the private registry.
2. In the pipeline, use docker build and docker push to build and push images to the private registry.
3. Example:

```
steps:
  - task: Docker@2
    inputs:
      containerRegistry: 'MyRegistry'
      repository: 'myapp'
      command: 'buildAndPush'
```

---

266. How do you configure agentless jobs in Azure Pipelines?

- **Answer**:
  1. Use **agentless jobs** (server jobs) when you don't need an agent to run tasks.
  2. Example:

```
jobs:
  - job: ServerJob
    pool: server
    steps:
      - task: ManualValidation@0
        inputs:
          instructions: 'Approve the release.'
```

---

267. How do you handle pipeline variable precedence in Azure Pipelines?

- **Answer**: Pipeline variables follow this precedence order:
  1. Command-line variables passed at runtime.
  2. Pipeline variables defined in YAML.
  3. Variables in **variable groups**.
  4. Environment variables defined at the agent level.

---

268. How do you implement load testing in Azure Pipelines?

- **Answer**:
    1. Use tools like **Apache JMeter** or **k6** for load testing.
    2. Add a task in the pipeline to run load tests and fail the pipeline if performance metrics are not met.

---

269. How do you use Git submodules in Azure Pipelines?

- **Answer**:
    1. Enable submodule checkout in the pipeline YAML.
    2. Example:

```
steps:
 - checkout: self
   submodules: true
```

---

270. How do you deploy Helm charts in Azure Pipelines?

- **Answer**:
    1. Use the **HelmInstaller** task to install Helm.
    2. Deploy the Helm chart using the **HelmDeploy** task.
    3. Example:

```
steps:
 - task: HelmInstaller@1
 - task: HelmDeploy@0
   inputs:
     connectionType: 'Kubernetes Service Connection'
     kubernetesServiceEndpoint: 'MyKubeEndpoint'
     chartType: 'FilePath'
     chartPath: 'charts/mychart'
```

---

271. How do you set up a CI/CD pipeline for a Flask app using Azure Pipelines?

- **Answer**:
    1. Use **Python** to install dependencies and build the Flask app.
    2. Deploy the app to **Azure App Service** or **Azure Functions**.

3. Example:

```
pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: UsePythonVersion@0
    inputs:
      versionSpec: '3.x'

  - script: pip install -r requirements.txt
  - task: AzureWebApp@1
    inputs:
      azureSubscription: 'MyAzureSubscription'
      appName: 'my-flask-app'
      package: '$(Build.ArtifactStagingDirectory)/drop.zip'
```

---

272. How do you configure triggers for pull requests in GitHub with Azure Pipelines?

- **Answer**:
  1. Use the pr keyword in your pipeline YAML to trigger on pull requests.
  2. Example:

```
pr:
  branches:
    include:
      - main
```

---

273. How do you secure pipelines using Azure Active Directory roles?

- **Answer**:
  1. Assign **Azure AD roles** to pipeline users, restricting access to specific resources and pipelines.
  2. Use **Azure AD groups** to manage access at the organization or project level.

---

274. How do you perform database schema migrations in Azure Pipelines?

- **Answer**:
    1. Use tools like **Flyway** or **Liquibase** to manage schema migrations.
    2. Add a pipeline task to run migration scripts.
    3. Example:

```
steps:
 - script: |
    flyway migrate
```

---

275. How do you use Azure Pipelines to build a mobile app?

- **Answer**:
    1. Use platform-specific build tasks like **Xcode** (for iOS) or **Gradle** (for Android).
    2. Example for Android:

```
pool:
 vmImage: 'macOS-latest'


steps:
 - task: Gradle@2
   inputs:
    gradleWrapperFile: 'gradlew'
    tasks: 'assembleRelease'
```

---

276. How do you handle environment-specific configuration in Azure Pipelines?

- **Answer**:
    1. Use **environment variables** or **variable groups** for different environments (e.g., dev, staging, prod).
    2. Apply the variables at the environment level during deployment.

---

277. How do you deploy an ASP.NET Core app to Azure App Service using Azure Pipelines?

- **Answer**:
    1. Use the **DotNetCoreCLI** task to build the app.

2. Deploy it using the **AzureWebApp** task.
3. Example:

```
steps:
 - task: DotNetCoreCLI@2
   inputs:
    command: 'publish'
    projects: '**/*.csproj'

 - task: AzureWebApp@1
   inputs:
    azureSubscription: 'MyAzureSubscription'
    appName: 'my-aspnetcore-app'
    package: '$(Build.ArtifactStagingDirectory)/drop.zip'
```

---

278. How do you handle pipeline retries for failed builds in Azure Pipelines?

- **Answer**:
  1. Use the retry keyword to configure automatic retries for failed builds.
  2. Example:

```
steps:
 - script: echo "Running build"
   retry:
    count: 3
    interval: 10
```

---

279. How do you perform static code analysis using Azure Pipelines?

- **Answer**:
  1. Integrate tools like **SonarQube** or **CodeQL** for static code analysis.
  2. Add a task to run static analysis as part of your build.
  3. Example with SonarQube:

```
steps:
 - task: SonarQubePrepare@4
```

```
  inputs:
    SonarQube: 'MySonarQubeConnection'
    projectKey: 'myproject'

- script: ./build.sh

- task: SonarQubeAnalyze@4
```

---

280. How do you set up pipeline conditions based on runtime variables in Azure Pipelines?

- **Answer**:
    1. Use condition to execute steps based on runtime variables.
    2. Example:

```
steps:
- script: echo "Running deployment"
  condition: and(succeeded(), eq(variables['Release.EnvironmentName'], 'Production'))
```

---

281. How do you configure Azure Pipelines to deploy to multiple Azure regions?

- **Answer**:
    1. Create separate deployment stages for each region.
    2. Use **ARM templates** or **Terraform** to deploy to different regions in parallel.
    3. Example:

```
stages:
- stage: DeployToEastUS
  jobs:
    - job: DeployJob
      steps:
        - task: AzureCLI@2
          inputs:
            azureSubscription: 'MyAzureSubscription'
            scriptType: 'bash'
            scriptLocation: 'inlineScript'
```

```
      inlineScript: |
        az group create --name MyResourceGroup --location EastUS

- stage: DeployToWestEurope
  jobs:
    - job: DeployJob
      steps:
        - task: AzureCLI@2
          inputs:
            azureSubscription: 'MyAzureSubscription'
            scriptType: 'bash'
            scriptLocation: 'inlineScript'
            inlineScript: |
              az group create --name MyResourceGroup --location WestEurope
```

---

282. How do you manage secrets in Azure Pipelines using Azure Key Vault?

- **Answer**:
    1. Set up an **Azure Key Vault** and store your secrets.
    2. Use the **AzureKeyVault@1** task to retrieve the secrets in your pipeline:

```
steps:
  - task: AzureKeyVault@1
    inputs:
      azureSubscription: 'MyAzureSubscription'
      keyVaultName: 'MyKeyVault'
      secretsFilter: '*'
```

---

283. How do you handle blue-green deployments using Azure Pipelines?

- **Answer**:
    1. Set up two deployment slots (e.g., **blue** and **green**).
    2. Deploy the new version to the **green slot**.
    3. After testing, swap the slots using **Azure App Service Slot Swap** task.

---

284. How do you implement database migrations in Azure Pipelines?

- **Answer**:
    1. Use database migration tools like **Entity Framework** or **Flyway**.
    2. Add a task to run the migration as part of your pipeline:

```
steps:
 - script: |
    dotnet ef database update
```

---

285. How do you secure Azure Pipelines using conditional access policies?

- **Answer**:
    1. Use **Azure AD Conditional Access Policies** to restrict access to pipelines based on conditions like user roles, locations, or devices.
    2. Configure **policy-based access** for Azure Pipelines using **Azure AD**.

---

286. How do you implement performance testing in Azure Pipelines?

- **Answer**:
    1. Use tools like **Apache JMeter** or **k6** for performance testing.
    2. Run performance tests as part of your pipeline and fail the pipeline if performance benchmarks are not met.

---

287. How do you build and deploy a Node.js application using Azure Pipelines?

- **Answer**:
    1. Use **npm** to install dependencies and build the Node.js app.
    2. Deploy the app to **Azure App Service** or **AKS**.
    3. Example:

```
steps:
 - task: NodeTool@0
   inputs:
    versionSpec: '14.x'
 - script: npm install
 - script: npm run build
```

```
- task: AzureWebApp@1
  inputs:
    azureSubscription: 'MyAzureSubscription'
    appName: 'my-node-app'
    package: '$(Build.ArtifactStagingDirectory)/drop.zip'
```

---

288. How do you manage multi-environment deployments in Azure Pipelines?

- **Answer**:
    1. Create separate stages for each environment (e.g., dev, staging, prod).
    2. Use **approvals** and **gates** to control when code moves between environments.

---

289. How do you configure automated smoke testing in Azure Pipelines?

- **Answer**:
    1. Add a post-deployment stage for **smoke tests**.
    2. Example using **curl** to run basic tests:

```
steps:
- script: |
    curl https://myapp.com/health
```

---

290. How do you deploy a Spring Boot application to Azure App Service using Azure Pipelines?

- **Answer**:
    1. Use **Maven** to build the Spring Boot app.
    2. Deploy the built JAR file to **Azure App Service**.
    3. Example:

```
steps:
- task: Maven@3
  inputs:
    mavenPomFile: 'pom.xml'
    goals: 'package'

- task: AzureWebApp@1
```

```
  inputs:
    azureSubscription: 'MyAzureSubscription'
    appName: 'my-springboot-app'
    package: '$(Build.ArtifactStagingDirectory)/target/*.jar'
```

---

291. How do you use conditional tasks in Azure Pipelines based on build status?

- **Answer**:
    1. Use the condition keyword to control task execution based on the build status.
    2. Example:

```
steps:
- script: echo "Running only if the build succeeds"
  condition: succeeded()
```

---

292. How do you deploy an API Gateway using Azure Pipelines?

- **Answer**:
    1. Use **ARM templates** or **Terraform** to deploy **Azure API Management**.
    2. Automate the deployment using **Azure CLI** or **ARM deployment** task.

---

293. How do you manage versioning in Azure Pipelines?

- **Answer**:
    1. Use **semver** (semantic versioning) to manage build versions.
    2. Automatically update the build number based on the build ID:

```
steps:
- script: echo "##vso[build.updatebuildnumber]1.0.$(Build.BuildId)"
```

---

294. How do you use conditional deployment strategies in Azure Pipelines?

- **Answer**:
    1. Use conditions to deploy only when certain criteria are met.
    2. Example:

```
steps:
- script: echo "Deploying only to production"
```

```
  condition: eq(variables['Build.SourceBranch'], 'refs/heads/main')
```

---

295. How do you monitor pipeline performance in Azure DevOps?

- **Answer**:
    1. Use the **Pipeline Insights** feature to monitor build and release performance metrics.
    2. Track **build duration**, **test pass rate**, and **failure trends**.

---

296. How do you manage access to sensitive files in Azure Pipelines?

- **Answer**:
    1. Use **Secure Files** to store and manage sensitive files (e.g., certificates).
    2. Add a step in your pipeline to download and use the secure files:

```
steps:
 - task: DownloadSecureFile@1
   inputs:
    secureFile: 'mycert.pfx'
```

---

297. How do you deploy multi-tier applications in Azure Pipelines?

- **Answer**:
    1. Create separate stages for each tier (e.g., frontend, backend, database).
    2. Use conditional tasks to deploy each tier in sequence.

---

298. How do you configure rollback on deployment failure in Azure Pipelines?

- **Answer**:
    1. Store the previous successful build artifact.
    2. On failure, deploy the last known good artifact.
    3. Example:

```
steps:
 - task: DownloadPipelineArtifact@2
   inputs:
    buildType: 'specific'
    project: 'MyProject'
```

```
    pipeline: 'MyPipeline'
    artifactName: 'drop'
- task: DeployToProd@1
  inputs:
    package: '$(Pipeline.Workspace)/drop/previous_artifact.zip'
```

---

299. How do you integrate SonarCloud for code quality analysis in Azure Pipelines?

- **Answer**:
    1. Install the **SonarCloud** extension from the marketplace.
    2. Add the **SonarCloudPrepare** and **SonarCloudAnalyze** tasks to your pipeline.
    3. Example:

```
steps:
- task: SonarCloudPrepare@1
  inputs:
    SonarCloud: 'SonarCloudConnection'
    organization: 'MyOrg'
    projectKey: 'myproject'

- script: ./build.sh

- task: SonarCloudAnalyze@1
```

---

300. How do you configure automated rollbacks in Kubernetes using Azure Pipelines?

- **Answer**:
    1. Use **Helm** or **kubectl** to deploy your Kubernetes resources.
    2. On failure, roll back the deployment using kubectl rollout undo or **Helm rollback**.
    3. Example:

```
steps:
- task: Kubernetes@1
  inputs:
    connectionType: 'Kubernetes Service Connection'
```

```
    kubernetesServiceEndpoint: 'MyAKSCluster'
    command: 'apply'
    arguments: '-f k8s/deployment.yaml'

- task: Kubernetes@1
  inputs:
    command: 'rollout undo'
    arguments: 'deployment/myapp'
  condition: failed()
```