Devops Shack

Azure DevOps

INTERVIEW GUIDE

By DevOps Shack

DevOps Shack

200 Real Time Azure Devops Questions and Answers

1. What is Azure DevOps, and what services does it offer?

Answer:
Azure DevOps is a cloud service provided by Microsoft that supports end-to-end software development and delivery. It offers the following services:

- Azure Repos: Version control using Git or Team Foundation Version Control (TFVC).
- Azure Pipelines: Build and release management through CI/CD.
- Azure Boards: Agile project management with Kanban boards, Scrum, and tracking tools.
- Azure Test Plans: Manual and automated testing services.
- Azure Artifacts: Package management for NuGet, npm, Maven, etc.

2. How do you create a YAML-based pipeline in Azure Pipelines?

Answer:
To create a YAML-based pipeline:

1. Go to the Pipelines section in Azure DevOps.
2. Click Create Pipeline and select the repository.
3. Choose YAML as the pipeline type.

Use a YAML file (`azure-pipelines.yml`) to define the pipeline:

```
  trigger:
    branches:
      include:
        - main
pool:
vmImage: 'ubuntu-latest'

steps:
    - task:
      UseNode@2
      inputs:
   version: '14.x'
    - script: |
      npm install npm
        test
  displayName: 'Install dependencies and run tests'
```

4. Commit the YAML file to your repository.

3. How do you implement branching strategies in Azure Repos?

Answer:

Azure Repos supports the following branching strategies:

- Feature Branching: Create a branch for each feature; merge into `main` or `develop` after completion.
- GitFlow: Uses `main`, `develop`, and supporting branches (`feature`, `release`, `hotfix`).
- Release Flow: Create branches for releases, keep `main` stable. Use policies like branch policies, pull requests, and required reviewers to enforce best practices.

4. How do you configure a Multi-Stage Pipeline in Azure Pipelines?

Answer:

A Multi-Stage Pipeline allows you to define multiple stages (e.g., Build, Test, Deploy) in a YAML file:

```
stages:
    - stage:
      Build jobs:
        - job:
          BuildJob
          steps:
            - script: echo "Building project"
    - stage: Deploy
      dependsOn:
      Build jobs:
        - job:
          DeployJob
          steps:
            - script: echo "Deploying application"
```

5. How do you secure sensitive information like passwords in Azure Pipelines?

Answer:

Use Azure Key Vault or Pipeline Variables:

- Pipeline Variables: Store secrets in the pipeline settings as variable groups or individual variables.
- Key Vault Integration: Add a Key Vault task in the pipeline to fetch secrets securely.

6. What are pull requests, and how are they managed in Azure Repos?

Answer:
A pull request is a mechanism to merge code changes from one branch into another.

- Steps to manage pull requests:
    1. Open Azure Repos and navigate to Pull Requests.
    2. Create a new pull request by specifying the source and target branches.
    3. Add reviewers and ensure policies are met (e.g., required approvals, successful builds).
    4. Merge the pull request once it is approved and passes validations.

7. How do you enforce branch policies in Azure Repos?

Answer:
Branch policies enforce standards to protect branches:

- Enable required reviewers.
- Set up build validation to trigger pipelines on pull requests.
- Restrict force pushes.
- Use status checks to ensure compliance before merging.

8. What are self-hosted agents, and why use them?

Answer:
Self-hosted agents run on your infrastructure rather than Azure-hosted pools:

- Use them when accessing private networks or specialized hardware.
- Install the agent software and register it in the Azure DevOps organization.

9. Explain artifacts in Azure Pipelines.

Answer:
Artifacts are the output of a build pipeline that can be used in release pipelines:

- Examples: Compiled binaries, Docker images, or configuration files.

Use the `PublishPipelineArtifact` task to publish:

```
- task: PublishPipelineArtifact@1
  inputs:
    targetPath: '$(Build.ArtifactStagingDirectory)'
    artifact: 'drop'
```

10. How do you integrate Azure Pipelines with Kubernetes?

Answer:

- Use a Kubernetes Service Connection.
- Add a deployment YAML file to the repository.

Configure a pipeline to use `kubectl` tasks:

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

11. What are swimlanes in Azure Boards, and how are they used?

Answer:
Swimlanes organize work items on Kanban boards:

- Create swimlanes based on priority, teams, or categories.
- Use them to visualize progress and identify bottlenecks.

12. How do you customize work item templates in Azure Boards?

Answer:

- Go to the Project Settings → Process.
- Select a process (e.g., Scrum or Agile) and customize fields, layouts, and rules for work items.

13. What is Universal Packages in Azure Artifacts?

Answer:
Universal Packages are a mechanism to store and share large files:

- Install the Azure Artifacts CLI.
- Use the `az artifacts universal` commands to publish and download packages.

14. How do you configure retention policies for artifacts?

Answer:

- Go to Project Settings → Artifacts → Retention.
- Specify retention duration and conditions to automatically delete old packages.

15. How do you perform load testing in Azure DevOps?

Answer:

- Azure Load Testing is integrated into pipelines.
- Create a load test plan and configure test parameters (e.g., user count, duration).

Add tasks in the pipeline for automated execution:

```
- task: LoadTest@1
  inputs:
    testPlan: 'loadtest.jmx'
```

16. How can you link test cases to requirements in Azure Boards?

Answer:

- Use the Link Work Items feature.
- Link test cases to user stories or features to ensure traceability.

17. What are Service Principals, and how do you use them in Azure DevOps?

Answer:
Service Principals are Azure AD applications used for authentication:

- Create a Service Principal in Azure AD.
- Add it to Azure DevOps Service Connections for accessing Azure resources.

18. How do you audit access and actions in Azure DevOps?

Answer:

- Enable Azure DevOps Auditing to track user activities.
- Access audit logs via Organization Settings → Auditing.

19. Explain Blue-Green Deployment in Azure Pipelines.

Answer:
Blue-Green Deployment minimizes downtime by having two environments:

- Deploy new changes to the "green" environment.
- Switch traffic to "green" after validation.
- Configure routing using Azure Traffic Manager or a Load Balancer.

20. What are Gates in Azure Pipelines?

Answer:
Gates are pre-deployment or post-deployment checks in release pipelines:

- Examples: Invoke REST APIs, check monitoring metrics.
- Configure Gates under pipeline settings.

21. How do you implement CI/CD for microservices in Azure DevOps?

Answer:

- Create separate pipelines for each microservice.
- Use YAML templates to standardize pipeline configurations.
- Manage dependencies with Azure Artifacts.

22. How do you troubleshoot failed pipeline jobs?

Answer:

- Review logs for errors.
- Enable Verbose Logging in pipeline settings.
- Re-run the job with specific debug flags.

23. What is a pipeline trigger, and what are its types?

Answer:
Pipeline triggers automate pipeline execution. Types include:

- **Continuous Integration (CI)**: Triggers on repository changes (e.g., push or PR merge).
- **Scheduled**: Runs pipelines at specific times.
- **Manual**: Initiated by a user.
- **Pipeline Triggers**: Runs one pipeline after another. Example of CI trigger in YAML:

```
trigger:

  branches:

    include:

      - main
```

24. What is deployment strategy in Azure Pipelines?

Answer:
Deployment strategies control how changes are rolled out:

- **Canary Deployment**: Incrementally shift traffic to a new version.
- **Rolling Deployment**: Deploy in phases, with each batch replacing part of the environment.
- **Blue-Green Deployment**: Maintain two environments and switch traffic after validation.

25. What is a pipeline variable, and how is it used?

Answer:
Pipeline variables store values used in pipelines:

- **Types**: Pipeline-scoped, stage-scoped, and job-scoped. Example:

```
variables:
  environment: 'QA'
steps:
  - script: echo $(environment)
```

26. How do you implement conditional tasks in pipelines?

Answer:
Use the `condition` property:

```
steps:
  - script: echo "Running"
    condition: eq(variables['Build.SourceBranchName'], 'main')
```

27. What is the difference between an Epic and a Feature in Azure Boards? Answer:

- **Epic**: Represents large business objectives or deliverables.
- **Feature**: Breaks down an Epic into smaller, manageable units.
- **Hierarchy**: Epics > Features > User Stories.

28. How do you track progress in Azure Boards?

Answer:

- Use **Dashboards** to display progress metrics.
- Create **Queries** for custom reports.
- Leverage **Cumulative Flow Diagrams** for bottleneck analysis.

29. What are Work Item Queries, and how are they used?

Answer:
Queries filter and group work items:

- **Types**: Flat list, Tree, or Direct links.
- Example: Retrieve all active user stories assigned to a specific team.

30. What is the difference between feeds and packages in Azure Artifacts?

Answer:

- **Feed**: Repository for storing packages.
- **Package**: Actual artifact stored in a feed (e.g., npm, NuGet, Maven).

31. How do you set up retention policies for packages in Azure Artifacts?

Answer:

- Define the number of versions to retain.
- Automatically delete unused packages after a specified duration.

32. How do you integrate Azure DevOps with Slack or Microsoft

Teams? Answer:

- Add a service hook for Slack or Teams in **Project Settings** → **Service Hooks**.
- Configure notifications for work item updates, pull requests, or pipeline statuses.

33. How do you integrate Azure DevOps with Terraform?

Answer:

- Use a pipeline task to install Terraform.

Automate Terraform commands (e.g., `init`, `plan`, `apply`) in a pipeline:

```
- script: terraform init

- script: terraform plan -out=tfplan

- script: terraform apply tfplan
```

34. What are test plans, and how do you create them in Azure Test Plans?

Answer:

Test plans define the scope of manual and automated tests:

- Create a plan under **Test Plans** in Azure DevOps.
- Add test cases, assign configurations, and track results.

35. What are the benefits of automated testing in Azure Pipelines?

Answer:

- Ensures consistent test execution.
- Provides quick feedback during CI/CD.
- Supports integration with popular frameworks (e.g., Selenium, JUnit).

36. What is the role of PAT (Personal Access Token) in Azure

DevOps? Answer:

- PAT provides scoped access to Azure DevOps services.
- Use it for automations or integrations where OAuth is not supported.

37. How do you manage permissions for teams in Azure DevOps?

Answer:

- Go to **Project Settings** → **Permissions**.
- Assign roles like Contributor, Reader, or Administrator to teams or users.

38. How do you use templates in Azure Pipelines?

Answer:

Templates enable reusability in pipelines:

```
# Template file: build-template.yml

  steps:

    - script: echo "Build

  step" # Main pipeline

stages:

    - stage:

      Build jobs:

   - template: build-template.yml
```

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

Answer:

- Use the **Pipeline Analytics** extension for metrics.
- Analyze logs and duration trends to optimize pipeline efficiency.

40. How do you migrate Jenkins pipelines to Azure Pipelines?

Answer:

- Export Jenkins jobs as XML.
- Translate pipeline scripts to YAML.
- Use Azure DevOps REST APIs or the UI to recreate jobs.

41. How do you handle multiple environments (Dev, QA, Prod) in pipelines?

Answer:

- Use stages for each environment.

Pass environment-specific variables:

```
stages:

  - stage:

    Dev

    variables:

      environment: 'Dev'

  - stage: QA

    dependsOn:

    Dev

    variables:
```

```
        environment: 'QA'
```

42. What is the best way to manage secrets in Azure

Pipelines? Answer:

- Use Azure Key Vault integration for secure storage.
- Store secrets in pipeline variable groups with encryption.

43. What are agent pools, and how are they managed?

Answer:

- Agent pools group agents to execute pipelines.
- Managed under **Organization Settings** → **Agent Pools**.

44. How do you scale self-hosted agents in a production environment?

Answer:

- Use a containerized agent setup.
- Implement auto-scaling using Kubernetes or Virtual Machine Scale Sets.

45. What is a Multi-Stage YAML Pipeline, and how does it differ from a classic release pipeline?

Answer:
A Multi-Stage YAML Pipeline defines the build and release process in a single YAML file, offering version control and infrastructure-as-code benefits:

- **Advantages**: Single source of truth, CI/CD in one file.
- **Difference**: Classic pipelines use UI-based editors, while YAML pipelines are defined programmatically.

46. How do you implement parallel jobs in Azure Pipelines?

Answer:
Use the `parallel` property:

```
jobs:
  - job:
    Build1
    steps:
      - script: echo "Build 1"
  - job:
    Build2
    steps:
      - script: echo "Build 2"
```

47. What are deployment slots, and how are they used with Azure Web Apps?

Answer:
Deployment slots allow multiple live environments for Azure Web Apps:

- Deploy to a staging slot, validate, and then swap with production.
- Benefits: Zero-downtime deployments and quick rollbacks.

48. How do you create custom dashboards in Azure Boards?

Answer:

- Navigate to **Dashboards** in Azure Boards.
- Add widgets like **Burndown Chart**, **Work Item Queries**, and **Pipeline Status**.
- Customize layout for team-specific needs.

49. What is a cumulative flow diagram, and why is it useful?

Answer:

- A cumulative flow diagram shows the status of work items over time.
- Useful for identifying bottlenecks and tracking progress.

50. How do you configure iterations and sprints in Azure Boards?

Answer:

- Go to **Project Settings** → **Iterations**.
- Define sprint timelines and assign work items.

51. How do you set up upstream sources in Azure Artifacts?

Answer:

- Enable upstream sources to consume packages from public or private feeds.
- Use the **Feed Settings** → **Upstream Sources** to configure.

52. What is the difference between a hosted feed and an external feed?

Answer:

- **Hosted Feed**: Managed within Azure Artifacts.
- **External Feed**: Connected to third-party repositories like npm or PyPI.

53. How do you restrict access to sensitive branches in Azure Repos?

Answer:

- Configure branch security under **Repos** → **Branch Policies**.
- Restrict permissions (e.g., push, delete) for specific users or groups.

54. What is the purpose of service connections in Azure DevOps?

Answer:
Service connections provide secure access to external services like Azure, AWS, or Kubernetes clusters:

- Create under **Project Settings** → **Service Connections**.
- Use in pipelines for deployments or integrations.

55. How do you run automated tests in Azure Pipelines?

Answer:
Use tasks specific to testing frameworks:

Example for NUnit:

```
- task: DotNetCoreCLI@2

  inputs:

    command: 'test'

    projects: '**/*.Tests.csproj'
```

56. What are test configurations in Azure Test Plans?

Answer:
Test configurations define environment parameters (e.g., OS, browser) to validate across multiple scenarios.

57. How do you implement Infrastructure as Code (IaC) in Azure Pipelines?

Answer:

- Use tools like Terraform or ARM templates.

Example with Terraform:

```
- script: terraform init

- script: terraform apply -auto-approve
```

58. How do you manage dependencies in monorepo setups in Azure Pipelines?

Answer:

Use path filters to trigger pipelines only for affected projects:

```
trigger:
  paths:
include:
  - src/project1/*
```

59. What is the purpose of a pipeline decorator in Azure DevOps?

Answer:

Pipeline decorators inject steps into existing pipelines:

- Useful for organization-wide policies like logging or security scans.

60. How do you deploy a containerized application using Azure DevOps?

Answer:

Build a Docker image:

```
- task:
  Docker@2
  inputs:
  command: 'buildAndPush' containerRegistry:
    'MyContainerRegistry' repository: 'my-
    app'
tags: '$(Build.BuildId)'
```

- Deploy using Kubernetes tasks or Azure App Services.

61. How do you handle rollback in Azure Pipelines?

Answer:

- Maintain versioned artifacts.
- Automate rollback by redeploying the previous version.

62. What is the best way to handle environment-specific variables?

Answer:

- Use variable groups scoped to environments.
- Alternatively, store in Azure Key Vault.

63. How do you optimize pipeline performance?

Answer:

Use parallel jobs and caching:

```
- task:

  CacheBeta@1

  inputs:

  key: 'npm | "$(Agent.OS)" | package-lock.json' path:

    'node_modules'
```

- Limit unnecessary triggers and optimize script execution.

64. How do you enable auditing in Azure DevOps?

Answer:

- Enable auditing from **Organization Settings** → **Auditing**.
- Review activity logs for compliance and troubleshooting.

65. How do you handle multi-region deployments in Azure

DevOps? Answer:

- Use pipeline stages for each region.
- Leverage templates to standardize deployment processes.

66. How do you use Azure Monitor with Azure DevOps?

Answer:

- Integrate Azure Monitor logs or alerts in pipelines.
- Trigger actions based on metrics.

67. What are gated check-ins in Azure Pipelines, and how do you configure them?

Answer:

Gated check-ins ensure that code changes build successfully before being committed:

- Enable under **Branch Policies**.
- Configure required builds to validate pull requests.

68. How do you implement artifact versioning in pipelines?

Answer:

Use variables to generate unique artifact versions:

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

69. What is a deployment queue, and why is it useful?

Answer:
Deployment queues manage parallel deployments to avoid resource contention:

- Useful in environments with limited capacity (e.g., Dev, QA).

70. How do you debug failed pipelines?

Answer:

- Enable **Debug Logging** in pipeline settings.
- Analyze detailed logs or download them for offline review.

71. What is capacity planning in Azure Boards?

Answer:

- Capacity planning allocates team members' availability for a sprint.
- Access via **Boards** → **Sprints** → **Capacity**.

72. How do you manage dependencies between work items in Azure Boards?

Answer:

- Use **Link Work Items** to establish relationships like "Blocks" or "Depends On."
- Visualize dependencies using queries or boards.

73. What is the difference between a Kanban board and a Scrum board?

Answer:

- **Kanban Board**: Focuses on workflow and limits work in progress.
- **Scrum Board**: Emphasizes sprint goals with planned tasks.

74. How do you promote packages in Azure Artifacts?

Answer:

- Use feeds to segregate environments (e.g., Dev, QA, Prod).
- Publish to the next feed after validation.

75. How do you handle package dependencies across projects?

Answer:

- Use Azure Artifacts feeds for dependency management.
- Leverage upstream sources to share dependencies.

76. What are the differences between OAuth tokens and PATs?

Answer:

- **OAuth Tokens**: Short-lived tokens used for service connections.
- **PATs**: Long-lived tokens for user or script authentication.

77. How do you implement two-factor authentication in Azure DevOps?

Answer:

- Enable MFA in Azure AD for users in the organization.
- Configure Conditional Access policies for additional security.

78. What is exploratory testing, and how is it performed in Azure DevOps?

Answer:

- Exploratory testing involves unscripted testing based on user scenarios.
- Use the **Test & Feedback** extension for browsers to perform and log findings.

79. How do you configure automated testing in a Multi-Stage Pipeline?

Answer:

Define test stages and include tasks for frameworks:

```
stages:

    - stage: Build

    - stage: Test

      dependsOn:

      Build jobs:

        - job: RunTests

          steps:

            - task: VSTest@2

              inputs:

              testSelector: 'testAssemblies' testAssemblyVer2:

                '**/*.dll'
```

80. What is the purpose of conditionally skipped stages in pipelines?

Answer:

- Skipping stages optimizes pipelines by avoiding unnecessary executions.

Example:

```
Stages:
```

```
  - stage: Deploy

    condition: eq(variables['Build.SourceBranchName'],
'main')
```

81. How do you set up a CI/CD pipeline for Azure Functions?

Answer:

Use an Azure Function App and configure deployment tasks:

```
- task: AzureFunctionApp@1

  inputs:

    azureSubscription: 'AzureServiceConnection'

    appType: 'functionApp'

    appName: 'MyFunctionApp'
```

82. What is the use of environment approvals in pipelines?

Answer:

- Environment approvals restrict deployments until manual or automated checks are complete.
- Configure under **Environment Settings** in Azure Pipelines.

83. How do you implement Blue-Green Deployment in Azure Pipelines?

Answer:

- Use two environments (e.g., Blue and Green).
- Deploy to the non-production environment and switch traffic using Azure Traffic Manager.

84. How do you manage cross-project dependencies in Azure DevOps?

Answer:

- Use Azure Artifacts for shared packages.
- Link work items across projects using queries or boards.

85. How do you monitor live deployments in Azure DevOps?

Answer:

- Enable monitoring tasks in pipelines (e.g., Azure Monitor metrics).
- Use dashboards to track health and performance.

86. How do you manage multiple organizations in Azure DevOps?

Answer:

- Use **Azure AD Tenants** to manage users across organizations.
- Establish resource sharing using Azure DevOps Project Collection features.

87. What are agent capabilities, and how do they affect pipelines?

Answer:

- Capabilities define what an agent can execute (e.g., specific tools or software).
- Match job demands with agent capabilities to ensure successful pipeline execution.

88. How do you optimize the cost of Azure DevOps usage?

Answer:

- Use Azure-hosted agents efficiently by reducing idle time.
- Implement resource quotas and clean up unused artifacts.

89. How do you handle dynamic environments in Azure Pipelines?

Answer:

Use dynamic variables and templates:

```
variables:

- name: environment

      value: ${{ parameters.env }}
```

90. How do you use templates for standardizing CI/CD pipelines?

Answer:

Create reusable YAML templates for build and release stages:

```
# Template file: standard-template.yml

  steps:

    - script: echo "Standard Step"
```

91. How do you migrate from GitHub Actions to Azure Pipelines?

Answer:

- Translate GitHub Actions workflows into Azure YAML pipelines.
- Use Azure Pipeline tasks for equivalent functionality.

92. How do you implement rolling deployments in Azure Pipelines?

Answer:

Rolling deployments update a subset of instances at a time to ensure availability:

- Use deployment strategies like **Deployment Groups** or **Kubernetes Rolling Updates**.

Configure in YAML:

```
strategy:
  rolling:
maxSurge: 1
maxUnavailable: 0
```

93. How do you use conditional insertion for tasks in Azure Pipelines?

Answer:

Add the `condition` property to dynamically include tasks:

```
steps:
- script: echo "Only on main branch"
    condition: eq(variables['Build.SourceBranchName'], 'main')
```

94. What are pipeline stages, and how do you organize them?

Answer:

Stages divide the pipeline into logical groups (e.g., Build, Test, Deploy):

Example YAML:

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

```
- stage: Deploy
  dependsOn:
  Build jobs:
    - job: DeployJob
```

95. How do you configure shared queries in Azure Boards?

Answer:

- Create a query in **Boards** → **Queries**.
- Save it under **Shared Queries** for team-wide access.

96. What is velocity tracking in Azure Boards?

Answer:

- Velocity tracking shows the amount of work completed in sprints.
- Use **Velocity Charts** to analyze team performance.

97. How do you use tags in work items for better organization?

Answer:

- Add tags to work items for categorization (e.g., "Critical," "Backend").
- Use queries to filter work items by tags.

98. How do you automate publishing packages in Azure Artifacts?

Answer:

Use pipeline tasks like `NuGetCommand@2` or `npm`:

```
- task: NuGetCommand@2
  inputs:
    command: 'push'
    packagesToPush: '**/*.nupkg'
    publishFeed: 'MyFeed'
```

99. How do you clean up unused artifacts?

Answer:

- Configure retention policies in **Artifacts Settings**.
- Automate cleanup with scheduled pipelines using Azure CLI.

100. What are environment permissions in Azure DevOps?

Answer:
Environment permissions control who can deploy to specific environments:

- Configure under **Pipelines** → **Environments** → **Security**.

101. How do you manage secret rotation in Azure Pipelines?

Answer:

- Rotate secrets in Azure Key Vault.
- Pipelines automatically retrieve updated secrets during execution.

102. What is an Azure DevOps Service Principal?

Answer:

- A Service Principal is a secure identity used to access Azure resources.
- Useful for automating deployments and provisioning infrastructure.

103. How do you integrate Selenium tests into Azure Pipelines?

Answer:

Add Selenium test tasks in the pipeline:

```
steps:
    - task: UsePythonVersion@0
      inputs:
    versionSpec: '3.x'
    - script: |
      pip install selenium
        python run_tests.py
```

104. What is code coverage, and how do you measure it in Azure Pipelines?

Answer:

Code coverage determines the percentage of code tested:

- Enable coverage tools like `JaCoCo` or

`Cobertura`. Publish coverage results using:

```
- task: PublishCodeCoverageResults@1
  inputs:
  codeCoverageTool: 'Cobertura'
    summaryFileLocation: 'coverage.xml'
```

105. How do you implement approval workflows in Azure Pipelines?

Answer:
Use environments with manual or automated approvals:

Example:

```
environment:
  name: 'QA'
  resourceName: 'MyEnvironment'
  approvals:
    - user: 'approver@example.com'
```

106. How do you enable pipeline caching to optimize performance?

Answer:

Cache dependencies like `npm` or `NuGet`:

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

107. What are Azure Resource Manager (ARM) templates, and how are they used in Azure DevOps?

Answer:
ARM templates define Azure infrastructure as

code: Deploy using pipeline tasks:

```
- task: AzureResourceManagerTemplateDeployment@3
  inputs:
    deploymentScope: 'Resource Group'
    templateLocation: 'Linked artifact'
    csmFile: 'template.json'
```

108. How do you handle feature toggles in Azure Pipelines?

Answer:

Use variables to enable/disable features:

```
variables:
  featureX: true
steps:
  - script: echo "Feature X is enabled"
    condition: eq(variables['featureX'],
    true)
```

109. How do you implement disaster recovery for Azure DevOps pipelines?

Answer:

- Backup pipeline YAML files in version control.
- Use multiple agents and regions for fault tolerance.

110. How do you deploy a serverless architecture using Azure DevOps?

Answer:

Deploy Azure Functions or AWS Lambda using tasks:

```
- task: AzureFunctionApp@1

  inputs:

    azureSubscription: 'MySubscription'

    appName: 'MyFunctionApp'
```

111. How do you manage billing and usage in Azure DevOps?

Answer:

- Use the **Usage** section in **Organization Settings**.
- Optimize agent pools and retention policies to reduce costs.

112. What are build retention policies, and how are they configured?

Answer:
Retention policies clean up old builds:

- Go to **Project Settings** → **Retention**.
- Define criteria for deletion (e.g., older than 30 days).

113. How do you enable pipeline triggers across multiple repositories?

Answer:
Use `resources` in YAML:

```
resources:

  repositories:

    - repository: Repo2

      type: git
```

```
      name: OrgName/Repo2
```

114. How do you configure Azure DevOps for hybrid cloud deployments?

Answer:

- Use service connections to link on-premises and cloud resources.
- Implement deployment agents with hybrid connectivity.

115. How do you monitor pipeline performance across teams?

Answer:

- Use **Pipeline Analytics** for team-level insights.
- Create custom dashboards to visualize metrics.

116. How do you implement matrix builds in Azure Pipelines?

Answer:
Matrix builds run jobs in parallel with different configurations:

```
jobs:
  - job: MatrixBuild
    strategy:
      matrix:
        Linux:
          vmImage: 'ubuntu-latest'
```

```
    Windows:
      vmImage: 'windows-latest'
  steps:
    - script: echo "Running on $(vmImage)"
```

117. How do you handle dependency management in Azure Pipelines?

Answer:
Use pipeline artifacts or caching:

Publish dependencies as artifacts:

```
- task: PublishPipelineArtifact@1
  inputs:
    targetPath: 'bin/output'
    artifact:
    'dependencies'
```

Restore them in subsequent jobs:

```
- task: DownloadPipelineArtifact@2
  inputs:
    artifactName: 'dependencies'
```

118. How do you use template parameters in YAML pipelines?

Answer:
Pass parameters into templates for flexibility:

```
# Template file: build-template.yml
parameters:
  - name: buildConfiguration
    default: 'Release'
steps:
  - script: echo "Building in $(buildConfiguration)
mode" # Main pipeline
jobs:
  - template: build-template.yml
    parameters:
      buildConfiguration: 'Debug'
```

119. How do you prioritize work items in Azure Boards?

Answer:

- Use fields like **Priority** and **Severity** to rank items.
- Drag and drop items on the Kanban board to reorder.

120. How do you generate reports in Azure Boards?

Answer:

- Use built-in widgets like **Burndown** or **Cumulative Flow**.
- Export query results to Excel for custom reporting.

121. What is the purpose of Work Item Rules in Azure Boards?

Answer:
Rules automate updates based on triggers (e.g., status changes).

- Example: Automatically assign a user when a work item is created.

122. How do you secure access to Azure Artifacts feeds?

Answer:

- Configure access control under **Feed Settings**.
- Use Azure AD groups to grant or restrict permissions.

123. What are upstream sources, and why are they useful?

Answer:
Upstream sources allow feeds to consume external or internal packages:

- Enable developers to reuse shared dependencies across teams.

124. How do you enforce MFA for Azure DevOps access?

Answer:

- Enable MFA in Azure AD for organization accounts.
- Use Conditional Access policies for enforcement.

125. How do you audit changes in Azure Repos?

Answer:

- Enable **Auditing** under **Organization Settings**.
- Track changes to branches, policies, and commits.

126. How do you handle secure file storage in Azure Pipelines?

Answer:

- Use **Secure Files** in pipeline libraries.

Access them in pipelines:

```
- task: DownloadSecureFile@1

  inputs:

secureFile: 'mycertificate.pfx'
```

127. What is the difference between functional and non-functional testing in Azure Pipelines?

Answer:

- **Functional Testing**: Verifies features work as expected (e.g., Selenium).
- **Non-Functional Testing**: Measures performance, reliability, or security (e.g., load testing).

128. How do you implement A/B testing with Azure Pipelines?

Answer:

- Use deployment slots for different versions.
- Split traffic using Azure Traffic Manager.

129. What are the best practices for managing infrastructure-as-code in Azure DevOps?

Answer:

- Use separate repositories for IaC.
- Implement pipelines to validate and deploy templates.
- Use linting tools like `terraform fmt` or `arm-ttk`.

130. How do you deploy containerized applications to Azure Kubernetes Service (AKS)?

Answer:

Build and push Docker images:

```
- task: Docker@2
  inputs:
  command: 'buildAndPush' containerRegistry:
    'MyContainerRegistry' repository: 'my-
    app'
tags: '$(Build.BuildId)'
```

Apply manifests using `kubectl` tasks:

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

131. How do you manage database migrations in Azure Pipelines?

Answer:

Use tools like Flyway or Liquibase:

```
- script: |
  flyway -url=jdbc:mysql://dbhost -user=dbuser
-password=dbpass migrate
```

132. How do you automate rollback in case of deployment failure?

Answer:

Retain previous artifacts and use a rollback pipeline stage:

```
- stage: Rollback

  condition: failed()

  steps:

    - script: echo "Deploying previous version"
```

133. How do you integrate monitoring tools like Datadog with Azure DevOps?

Answer:

- Use webhook integrations for alerts.

Embed monitoring tasks in pipelines:

```
- script: |

    curl -X POST -H "Content-Type: application/json" -d
'{"status":"start"}' https://api.datadoghq.com/api
```

134. How do you optimize agent usage in Azure DevOps?

Answer:

- Use `pool` settings to allocate agents efficiently.
- Implement agent caching and parallelism.

135. What are the differences between self-hosted and Microsoft-hosted agents?

Answer:

- **Microsoft-Hosted**: Pre-configured and managed, but may have usage limits.
- **Self-Hosted**: Fully customizable, suitable for private environments.

136. How do you set up billing alerts for Azure DevOps usage?

Answer:

- Monitor usage under **Organization Settings**.
- Use Azure Cost Management for notifications.

137. How do you implement CI/CD for mobile apps using Azure DevOps?

Answer:

- Use tasks like `Xamarin@3` for building

apps. Distribute builds via App Center:

```
- task: AppCenterDistribute@3

  inputs:

    appSlug: 'org/app'

    releaseNotes: 'New release'
```

138. How do you configure pipeline templates for multi-project environments?

Answer:

- Create reusable templates and include them in project-specific pipelines.

139. How do you manage feature branches with CI/CD pipelines?

Answer:

Use branch filters in triggers:

```
trigger:
  branches:
    include:
      - features/*
```

140. How do you implement automated approvals in Azure Pipelines?

Answer:
Use Azure DevOps environments with automated checks:

```
environment:
  name: 'Production'
  approval:
    - user: 'approver@example.com'
```

141. How do you use deployment conditions in Azure Pipelines?

Answer:
Deployment conditions control stage execution:

```
stages:
  - stage: Deploy
    condition: and(succeeded(),
eq(variables['Build.SourceBranch'], 'refs/heads/main'))
```

142. What is the difference between inline scripts and script files in Azure Pipelines?

Answer:

- **Inline Scripts**: Directly written in the YAML file.
- **Script Files**: External files stored in the repository, better for reuse and maintainability.

143. How do you manage secrets in multi-stage pipelines?

Answer:
Use Azure Key Vault or variable groups scoped to stages:

```
variables:

- group: 'MySecrets'
```

144. How do you track blocked work items in Azure Boards?

Answer:

- Use custom fields or tags like "Blocked."
- Filter work items in queries or use Kanban board swimlanes for blocked items.

145. What is a retrospective in Azure Boards, and how is it conducted?

Answer:

- Retrospectives assess sprint performance.
- Use Azure DevOps extensions like "Retrospectives" for structured feedback collection.

146. How do you manage recurring tasks in Azure Boards?

Answer:

- Create templates for tasks that repeat frequently.
- Automate creation using Azure DevOps REST API or Power Automate.

147. How do you migrate packages from an external feed to Azure Artifacts?

Answer:

Use tools like `npm` or `NuGet` to download from the external source and upload to Azure Artifacts:

```
npm publish --registry
  https://pkgs.dev.azure.com/org/_packaging/feed/npm
```

148. What is deduplication in Azure Artifacts, and why is it important?

Answer:

Deduplication reduces storage usage by avoiding duplicate package uploads. It ensures efficient space utilization in feeds.

149. How do you configure pipeline permissions in Azure DevOps?

Answer:

- Set permissions at the pipeline level under **Pipeline Security**.
- Restrict triggers or access to specific user groups.

150. How do you audit pipeline execution in Azure DevOps?

Answer:

- Use **Auditing** under **Organization Settings** to monitor pipeline activities.
- Log access, builds, and deployments for compliance.

151. What are custom roles in Azure DevOps, and how do you create them?

Answer:

- Custom roles define specific permissions for users.
- Configure under **Project Settings** → **Permissions**.

152. How do you integrate JMeter tests with Azure Pipelines?

Answer:

Run JMeter scripts in the pipeline using a container:

```
steps:

- script: |

        docker run -v $(System.DefaultWorkingDirectory):/tests
  jmeter/jmeter -n -t /tests/test-plan.jmx
```

153. How do you handle flaky tests in Azure Pipelines?

Answer:

- Use retry mechanisms or quarantine options for unstable tests.
- Identify patterns using test analytics.

154. What are the benefits of using YAML pipelines over classic pipelines?

Answer:

- **Version Control**: YAML pipelines are stored in repositories.
- **Portability**: Easily transferable across projects.
- **Flexibility**: Advanced features like templates and conditions.

155. How do you implement zero-downtime deployments in Azure DevOps?

Answer:

- Use deployment slots or rolling updates.
- Switch traffic only after verifying the new deployment.

156. How do you manage environment variables across multiple pipelines?

Answer:

- Use variable groups in **Library Settings**.
- Store and share variables across pipelines.

157. How do you implement multi-cloud deployments using Azure Pipelines?

Answer:

- Configure service connections for Azure, AWS, and GCP.

Deploy resources using respective CLI or Terraform:

```
- task: AWSCLI@1

  inputs:

    awsCommand: 's3 cp myfile s3://mybucket'
```

158. How do you monitor application performance post-deployment?

Answer:

- Integrate Azure Monitor or Application Insights.
- Configure alerts for pipeline-based notifications.

159. How do you manage feature flags in Azure Pipelines?

Answer:

- Use feature management tools like LaunchDarkly.
- Update flags during deployment using API calls.

160. How do you manage cross-team dependencies in Azure DevOps?

Answer:

- Use Azure Boards for tracking dependencies with linked work items.
- Establish clear SLA agreements for shared services.

161. What is the best way to manage project templates in Azure DevOps?

Answer:

- Use **Process Templates** to standardize project configurations.
- Include predefined boards, pipelines, and repos.

162. How do you secure self-hosted agents?

Answer:

- Restrict access using agent pool permissions.
- Run agents on dedicated machines with limited internet access.

163. How do you implement Git submodules in Azure Pipelines?

Answer:

Enable submodule fetching in the pipeline:

```
steps:

    - checkout: self

      submodules: true
```

164. How do you schedule pipeline runs in Azure DevOps?

Answer:

Use cron-like syntax for triggers:

```
schedules:

    - cron: "0 2 * * *"

      displayName: Nightly Build

      Branches:
```

```
    include:
      - main
```

165. How do you measure team productivity in Azure DevOps?

Answer:

- Use metrics like **Velocity** and **Cumulative Flow Diagrams**.
- Analyze query-based reports for work item progress.

166. How do you define deployment gates in Azure Pipelines?

Answer:
Deployment gates are checks before or after a deployment in release pipelines.
Example: Validate a REST API response before proceeding.

```
gates:
  preDeployGates:
    - task: InvokeRESTAPI@1
      inputs:
        method: 'GET'
        url: 'https://api.example.com/health'
```

167. How do you use pipeline variables dynamically?

Answer:
Pipeline variables can be dynamically set during runtime:

```
steps:
  - script: echo
"##vso[task.setvariable
variable=dynamicVar]dynamicValue"
  - script: echo $(dynamicVar)
```

168. What are run-time parameters, and how do they differ from variables?

Answer:

- **Run-time Parameters**: Set before pipeline execution and immutable.
- **Variables**: Can be updated during the pipeline execution. Example of a parameter:

```
parameters:
    - name:
      environment
      type: string
      default: 'QA'
```

169. How do you create hierarchical work items in Azure Boards?

Answer:

- Link work items using relationships like **Parent/Child**.
- Use **Backlog View** to organize hierarchy visually.

170. What is the purpose of swimlanes in Azure Boards?

Answer:

Swimlanes organize Kanban boards by categories like priority or work types, helping teams focus on specific objectives.

171. How do you enable work item templates in Azure Boards?

Answer:

- Navigate to **Boards** → **Work Item Templates**.
- Create templates for repetitive tasks to prefill fields.

172. How do you configure retention policies for Azure Artifacts?

Answer:

- Navigate to **Artifacts → Feed Settings → Retention Policies**.
- Define policies to keep only the latest versions or delete unused packages.

173. How do you handle large packages in Azure Artifacts?

Answer:

- Use **Universal Packages** for files over 1 GB.

Upload using the Azure CLI:

```
az artifacts universal publish --organization
  https://dev.azure.com/org --feed feed-name --name
  package-name --version 1.0.0 --path
  /path/to/files
```

174. How do you enforce branch security in Azure Repos?

Answer:

- Configure **Branch Policies** under the repository settings:
  - Require pull request reviews.
  - Enable **Merge Commit Restrictions**.
  - Require build validation.

175. How do you secure service connections in Azure DevOps?

Answer:

- Use Azure RBAC to restrict permissions.
- Store credentials in Azure Key Vault for enhanced security.

176. How do you manage access for contractors in Azure DevOps?

Answer:

- Create a separate Azure DevOps project with restricted access.
- Use **Stakeholder** access or custom security groups for limited visibility.

177. How do you automate security testing in Azure Pipelines?

Answer:

Use security scanning tools like **OWASP ZAP** or **SonarQube**:

```
steps:
    - script: |
   zap-baseline.py -t https://myapp.example.com
    - task: SonarQubePrepare@5
```

178. How do you implement test prioritization in Azure Pipelines?

Answer:

- Group tests into categories based on criticality.
- Execute high-priority tests first using testing tools like NUnit or MSTest.

179. What are composite actions in Azure Pipelines?

Answer:

Composite actions are reusable YAML templates that can combine multiple steps:

```
steps:
- template: my-template.yml
```

180. How do you handle infrastructure drift with Azure DevOps?

Answer:

- Use Terraform or ARM templates to maintain infrastructure

state. Integrate pipelines to regularly check drift:

```
- script: terraform plan -detailed-exitcode
```

181. How do you manage multi-region deployments with Azure DevOps?

Answer:

Define regions as separate stages:

```
stages:
  - stage: DeployToUS
  - stage: DeployToEU
```

182. How do you implement a hotfix pipeline in Azure DevOps?

Answer:

- Create a branch policy to allow fast-track approvals for hotfix branches.
- Deploy artifacts directly to production using conditional triggers.

183. How do you monitor deployment pipelines?

Answer:

- Integrate with Azure Monitor or Application Insights.
- Use pipeline logs and dashboards for detailed insights.

184. How do you manage concurrent pipeline executions?

Answer:

- Use **Job Limits** in pipeline settings to restrict concurrency.
- Manage resources effectively with parallel jobs.

185. What are organization-level policies in Azure DevOps?

Answer:

- Organization policies control settings across all projects:
  - Enforce secure access.
  - Define repository and pipeline standards.

186. How do you archive inactive projects in Azure DevOps?

Answer:

- Disable project access under **Project Settings**.
- Retain project data for compliance without active use.

187. How do you deploy a microservices architecture using Azure DevOps?

Answer:

- Use separate pipelines for each microservice.
- Automate deployments to Kubernetes or Docker Swarm.

188. How do you integrate Azure DevOps with Jira?

Answer:

- Use the Azure DevOps Marketplace extension for Jira integration.
- Sync work items and issues between platforms.

189. How do you implement blue/green deployment for databases?

Answer:

- Use tools like Flyway or Liquibase for schema versioning.
- Deploy database changes to a staging environment before switching traffic.

190. What are conditional approvals in Azure DevOps?

Answer:

Conditional approvals allow specific approvers based on deployment parameters:

```
approvals:
- user: 'admin@example.com'
  condition: eq(variables['environment'], 'Production')
```

191. How do you manage pipeline failures automatically?

Answer:

Trigger a rollback or notify stakeholders using post-failure conditions:

```
steps:
  - script: echo "Pipeline failed"
    condition: failed()
```

192. How do you set up CI/CD for serverless applications in Azure?

Answer:

- Use Azure Functions deployment tasks.
- Integrate with event-based triggers like Event Grid.

193. What are the differences between Azure DevOps and GitHub Actions?

Answer:

- **Azure DevOps**: Enterprise-grade with boards, repos, pipelines, and artifacts.
- **GitHub Actions**: Lightweight CI/CD focused on repositories.

194. How do you manage ephemeral environments in Azure DevOps?

Answer:

- Use Terraform or Pulumi to provision and destroy environments dynamically in pipelines.

195. What are delivery plans in Azure Boards?

Answer:

- Delivery Plans visualize multiple team timelines and dependencies.

- Configure under **Boards → Delivery Plans**.

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

Answer:

Integrate tools like Gremlin or Chaos Toolkit:

```
- script: chaos run experiment.json
```

197. How do you enable continuous feedback in Azure DevOps?

Answer:

- Use Azure Monitor and Application Insights.
- Automate feedback loops with alerts and dashboards.

198. How do you use release annotations in Azure Pipelines?

Answer:

Add release notes or metadata for tracking:

```
steps:
- script: echo "##vso[release.addartifact]Artifact notes"
```

199. How do you enforce naming conventions for resources in Azure Pipelines?

Answer:

- Use custom scripts or templates to validate resource names during provisioning.

200. What are best practices for scaling Azure DevOps across multiple teams?

Answer:

- Use shared repositories and templates.
- Implement standard policies across organizations.