DevOps Shack

Comprehensive Guide to AZURE DevOps

[Click Here To Enrol To Batch-6 | DevOps & Cloud DevOps]

1. Introduction to Azure DevOps

Setting up Azure DevOps Account:

Step-by-Step Setup:

1. **Sign Up:**
   - Go to the Azure DevOps homepage.
   - Click on "Start free" or "Start free with GitHub" to sign up.
   - Sign in with your Microsoft or GitHub account.
2. **Create an Organization:**
   - After signing in, you will be prompted to create a new organization.
   - Enter an organization name (must be unique) and choose your region.
   - Click "Continue".
3. **Create a Project:**
   - In the Azure DevOps dashboard, click on "New Project".
   - Enter a project name and description.
   - Choose the visibility (public or private).
   - Click "Create".

2. Azure Repos

Creating a Repository:

Step-by-Step Repository Setup:

1. **Initialize Repository:**
   - Navigate to your project in Azure DevOps.
   - Go to Repos > Files.

    - Click on "Initialize" if it's a new repository.
2. **Clone Repository:**
    - Copy the repository URL.
    - Clone the repository to your local machine:

```
git clone
https://dev.azure.com/yourorganization/yourproject/_git/yourrep
ository
```

Branching Strategies:

Branching Best Practices:

- **Master/Main Branch:** Stable version of the code, ready for production.
- **Develop Branch:** Integration branch for features, usually the next release.
- **Feature Branches:** Separate branches for new features.
- **Release Branches:** Branches created for each release.
- **Hotfix Branches:** Branches for critical fixes in production.

Pull Requests and Code Reviews:

Pull Request Workflow:

1. **Create a Pull Request:**
    - Go to Repos > Pull Requests.
    - Click "New Pull Request".
    - Select the source and target branches.
    - Add a title, description, and reviewers.
    - Click "Create".
2. **Review and Approve:**
    - Reviewers check the code changes.
    - Add comments and request changes if needed.
    - Approve the PR when satisfied.
3. **Complete the Merge:**
    - After approval, click "Complete".
    - Choose the merge strategy (merge commit, squash, or rebase).
    - Complete the merge and delete the source branch if desired.

3. Azure Pipelines

Creating CI/CD Pipelines:

Detailed Pipeline Setup:

1. **Create a YAML Pipeline:**
    - Go to Pipelines > Pipelines.
    - Click "New pipeline" and select your repository.
    - Configure the pipeline using YAML.

Example `azure-pipelines.yml` for a Python Project:

```
trigger:
- master

pool:
  vmImage: 'ubuntu-latest'

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

- script: |
    python -m pip install --upgrade pip
    pip install -r requirements.txt
  displayName: 'Install dependencies'

- script: |
    pytest
  displayName: 'Run tests'

- task: PublishTestResults@2
  inputs:
    testResultsFiles: '**/test-*.xml'
    testRunTitle: 'Python Test Results'

- task: ArchiveFiles@2
  inputs:
    rootFolderOrFile: '$(System.DefaultWorkingDirectory)'
    includeRootFolder: false
    archiveType: 'zip'
    archiveFile: '$(Build.ArtifactStagingDirectory)/output.zip'
    replaceExistingArchive: true

- publish: $(Build.ArtifactStagingDirectory)/output.zip
  artifact: drop
```

2. **Create a Release Pipeline:**
    - Go to Pipelines > Releases.
    - Click "New pipeline" and define stages, artifacts, and deployment jobs.

Example Deployment Job:

```
stages:
- stage: Deploy
  jobs:
  - job: DeployJob
    pool:
      vmImage: 'ubuntu-latest'
    steps:
    - script: echo "Deploying application..."
      displayName: 'Deploy application'
```

4. Azure Boards

Work Items: Issues, Tasks, Bugs:

Creating and Managing Work Items:

1. **Create Work Items:**
    - Navigate to Boards > Work Items.
    - Click "New Work Item" and select the type (Issue, Task, Bug).
    - Fill in the details and save.
2. **Manage Work Items:**
    - Use queries to filter and find work items.
    - Create custom queries and pin them to dashboards.
    - Use tags and custom fields for better organization.

Sprint Planning and Tracking:

Effective Sprint Management:

1. **Plan a Sprint:**
    - Go to Boards > Sprints.
    - Click "New Sprint" and define the sprint dates.
    - Add tasks to the sprint by dragging them from the backlog.
2. **Track Progress:**
    - Use the sprint board to update task statuses.
    - Monitor sprint progress with the burndown chart.

Customizing Azure Boards:

Customization Options:

1. **Custom Work Item Types:**
    - Navigate to Project Settings > Boards > Process.
    - Customize work item types, states, and fields.
2. **Custom States and Transitions:**
    - Define custom states for work items.
    - Set up state transitions and rules.

Hands-on Example: Using Azure Boards for Agile Project Management

Step-by-Step Agile Management:

1. **Create User Stories and Tasks:**
    - Go to Boards > Work Items.
    - Create new user stories and break them down into tasks.
2. **Plan and Execute Sprints:**
    - Organize work items into sprints.
    - Use the sprint board to track progress.
3. **Monitor and Report:**
    - Use dashboards and queries to monitor project status.
    - Generate reports to track team performance and velocity.

5. Azure Test Plans (Expanded)

Creating and Managing Test Cases:

Step-by-Step Test Case Management:

1. **Create Test Cases:**
    - Go to Test Plans > Test Cases.
    - Click "New Test Case" and define the test steps.
2. **Organize Test Suites:**
    - Group test cases into test suites for better organization.
    - Use requirement-based, static, and query-based suites.

Manual and Automated Testing:

Detailed Testing Workflow:

1. **Create Test Plans:**
    - Go to Test Plans > Test Plans.
    - Click "New Test Plan" and define the plan.
2. **Execute Manual Tests:**
    - Select a test case from the test plan.
    - Click "Run" to execute the test manually.
    - Record the results and save.
3. **Automate Tests:**
    - Integrate automated tests into CI/CD pipelines.
    - Use tools like Selenium, Appium, and Postman.

Test Reporting:

Generating Test Reports:

1. **View Test Results:**
    - Go to Test Plans > Runs.
    - View detailed test run results.
2. **Generate Custom Reports:**
    - Use Analytics views to create custom test reports.
    - Track test coverage and quality trends.

Hands-on Example: Setting up and executing test plans

Step-by-Step Test Execution:

1. **Create Test Cases:**
    - Go to Test Plans > Test Cases.
    - Click "New Test Case" and enter the test details.
2. **Plan and Execute Tests:**
    - Create a test plan and add test cases.

    - Execute test cases and record results.
3. **Automate and Report:**
    - Integrate automated tests into pipelines.
    - Generate and analyze test reports.

6. Azure Artifacts (Expanded)

Creating and Publishing Packages:

Step-by-Step Package Management:

1. **Create a Feed:**
    - Navigate to Artifacts > Feeds.
    - Click "New Feed" to create a feed.
2. **Publish Packages:**
    - Configure package source in your project.
    - Publish packages using tools like NuGet, npm, and Maven.

Example: Publishing an npm package:

```
# Add the feed to your npm configuration
npm config set registry
https://pkgs.dev.azure.com/yourorganization/_packaging/yourfeed/npm/registr
y/

# Publish the package
npm publish
```

Consuming Packages in Pipelines:

Step-by-Step Package Consumption:

1. **Configure Package Source:**
    - Add the feed to your pipeline's YAML file.
2. **Install Packages in Pipelines:**
    - Use tasks to install packages during builds.

Example: Consuming an npm package in a pipeline:

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

steps:
- script: |
    npm config set registry
https://pkgs.dev.azure.com/yourorganization/_packaging/yourfeed/npm/registr
y/
    npm install yourpackage
  displayName: 'Install npm packages'
```

7. Integrating with Other Services

GitHub Integration:

Step-by-Step GitHub Integration:

1. **Create GitHub Service Connection:**
    - Navigate to Project Settings > Service connections.
    - Add a new GitHub

connection and authorize access.

2. **Configure Pipeline to Use GitHub:**
    - Create a new pipeline and select GitHub as the source.

Service Hooks and Webhooks:

Setting Up Service Hooks:

1. **Create Service Hook Subscriptions:**
    - Navigate to Project Settings > Service hooks.
    - Create new subscriptions for various events.
2. **Configure Webhooks:**
    - Set up webhooks to integrate with third-party services.

Integrating with Third-party Tools:

Using Azure DevOps Extensions:

1. **Install Extensions:**
    - Browse the Azure DevOps Marketplace.
    - Install extensions for tools like Slack, JIRA, and Jenkins.
2. **Configure Integrations:**
    - Follow extension-specific instructions for configuration.

Hands-on Example: Integrating Azure DevOps with GitHub

Step-by-Step GitHub Integration:

1. **Create a GitHub Service Connection:**
    - Go to Project Settings > Service connections.
    - Add a new GitHub connection and authorize access.
2. **Configure Pipeline:**
    - Create a new pipeline and select GitHub as the source.
    - Configure the pipeline steps using the YAML file.

8. Security and Compliance (Expanded)

Role-Based Access Control (RBAC):

Managing Permissions:

1. **Assign Roles and Permissions:**
    - Navigate to Project Settings > Permissions.
    - Assign roles and permissions to users and groups.
2. **Define Policies:**
    - Set up policies for repositories, pipelines, and artifacts.

Policies and Permissions:

Enforcing Policies:

1. **Branch Policies:**
    - Define branch policies to enforce code quality.
    - Require a minimum number of reviewers for pull requests.
2. **Pipeline Permissions:**
    - Set up permissions for pipelines to control access.

Secure DevOps Kit:

Implementing Security Practices:

1. **Use Security Tools:**
    - Integrate tools like Microsoft Security Code Analysis.
    - Implement security scanning in pipelines.
2. **Implement Best Practices:**
    - Follow DevSecOps principles for secure development.

Hands-on Example: Implementing Security in Azure DevOps

Step-by-Step Security Implementation:

1. **Define Branch Policies:**
    - Go to Repos > Branches.
    - Select a branch and configure policies.
2. **Set Up Pipeline Permissions:**
    - Go to Pipelines > Pipelines.
    - Select a pipeline and configure access control.
3. **Integrate Security Scanning:**
    - Add security scanning tools to your pipeline YAML file.

Example: Security Scan Integration:

```
steps:
- script: |
    # Run security scan
    your-security-scan-tool
  displayName: 'Run security scan'
```

9. Monitoring and Reporting (Expanded)

Setting Up Dashboards:

Creating Custom Dashboards:

1. **Create Dashboard:**
    - Go to Dashboards and click "New Dashboard".
    - Add widgets for various metrics and data.
2. **Customize Widgets:**
    - Add widgets for pipelines, work items, and test plans.
    - Customize widgets to display relevant information.

Custom Reports and Analytics:

Generating Custom Reports:

1. **Create Analytics Views:**
    - Go to Analytics views and create a new view.
    - Define the data source and metrics for the report.
2. **Generate Insights:**
    - Use analytics views to generate insights into build and release pipelines.

Alerts and Notifications:

Setting Up Alerts:

1. **Create Alert Rules:**
    - Go to Project Settings > Notifications.
    - Create new alert rules and configure triggers.
2. **Configure Notifications:**
    - Set up notifications for email, SMS, and webhooks.

Hands-on Example: Monitoring and Reporting in Azure DevOps

Step-by-Step Monitoring and Reporting:

1. **Create a Dashboard:**
    - Go to Dashboards and click "New Dashboard".
    - Add widgets for various metrics and data.
2. **Set Up Alerts:**
    - Go to Project Settings > Notifications.
    - Create new alert rules and configure triggers.
3. **Generate Custom Reports:**
    - Go to Analytics views and create a new view.
    - Define the data source and metrics for the report.

10. Advanced Topics

Scaling Azure DevOps for Large Teams:

Organizing Projects and Teams:

1. **Use Azure DevOps Organizations:**
    - Create multiple projects within an organization.
    - Manage teams and permissions at the organization level.
2. **Share Resources:**
    - Share repositories, pipelines, and artifacts across projects.

Managing Multiple Projects:

Effective Multi-Project Management:

1. **Navigate Between Projects:**
    - Use the project switcher to navigate between projects.
2. **Share Pipelines and Resources:**
    - Share pipelines, artifacts, and service connections across projects.

DevOps Best Practices:

Implementing Best Practices:

1. **Continuous Feedback and Improvement:**
    - Implement continuous feedback loops.
    - Use monitoring and analytics to improve processes.
2. **Infrastructure as Code:**
    - Use tools like Azure Resource Manager (ARM) templates and Terraform.
    - Automate infrastructure provisioning and management.

Hands-on Example: Advanced Pipeline Techniques

Step-by-Step Advanced Pipeline Techniques:

1. **Use Templates in YAML Pipelines:**
    - Create reusable templates for common pipeline tasks.

Example Template File `template.yml`:

```
parameters:
- name: buildSteps
  type: stepList
  default: []

steps:
- ${{ each step in parameters.buildSteps }}:
  - ${{ step }}
```

2. **Implement Infrastructure as Code:**
    - Use ARM templates or Terraform to define infrastructure.

Example ARM Template Deployment:

```
steps:
- task: AzureResourceManagerTemplateDeployment@3
  inputs:
    deploymentScope: 'Resource Group'
    azureResourceManagerConnection: 'YourAzureRMConnection'
    subscriptionId: 'YourSubscriptionId'
    action: 'Create Or Update Resource Group'
    resourceGroupName: 'YourResourceGroup'
    location: 'YourLocation'
    templateLocation: 'Linked artifact'
    csmFile: '$(Build.ArtifactStagingDirectory)/azuredeploy.json'
    csmParametersFile:
'$(Build.ArtifactStagingDirectory)/azuredeploy.parameters.json'
```