CodeCruise

SAIL SMOOTH IN TECH OCEAN

A74, TechnoPark, Andheri, Mumbai. Phone: +91 80809 75897 | +91 70345 62050 Email: ask@codecruise.in

AZ-400 Designing and Implementing Microsoft Azure DevOps Solutions

Exam: Yes
Course Vendor: Microsoft
Lab/Exercise: Yes

Audience profile

As a DevOps engineer, you're a developer or infrastructure administrator who also has subject matter expertise in working with people, processes, and products to enable continuous delivery of value in organizations.

Your responsibilities for this role include designing and implementing strategies for collaboration, code, infrastructure, source control, security, compliance, continuous integration, testing, delivery, monitoring, and feedback.

Who Should Enroll?

This course is perfect for:

- DevOps Engineers
- Developers
- Infrastructure Administrators
- IT Professionals looking to enhance their DevOps skills

Tech Stack To Be Covered

Azure

Devops

Module 01: Configure processes and communications

1.1 Configure activity traceability and flow of work

•Plan and implement a structure for the flow of work and feedback cycles

•Identify appropriate metrics related to flow of work, such as cycle times, time to recovery, and lead time

•Integrate Azure Pipelines and GitHub Actions with work item tracking tools

•Implement traceability policies decided by development

•Integrate a repository with Azure Boards

1.2 Configure collaboration and communication

•Communicate actionable information by using custom dashboards in Azure Boards

•Document a project by using tools, such as wikis and process diagrams

•Configure release documentation, including release notes and API documentation

•Automate creation of documentation from Git history

•Configure notifications by using webhooks

Module 02: Design and implement source control

2.1 Design and implement a source control strategy

•Design and implement an authentication strategy

•Design a strategy for managing large files, including Git Large File Storage (LFS) and git-fat

•Design a strategy for scaling and optimizing a Git repository, including Scalar and cross-repository sharing

•Implement workflow hooks

2.2 Plan and implement branching strategies for the source code

•Design a branch strategy, including trunk-based, feature branch, and release branch

•Design and implement a pull request workflow by using branch policies and branch protections

•Implement branch merging restrictions by using branch policies and branch protections

2.3 Configure and manage repositories

•Integrate GitHub repositories with Azure Pipelines

•Configure permissions in the source control repository

•Configure tags to organize the source control repository

•Recover data by using Git commands

•Purge data from source control

Module 03: Design and implement build and release pipelines

3.1 Design and implement pipeline automation

•Integrate pipelines with external tools, including dependency scanning, security scanning, and code coverage

•Design and implement quality and release gates, including security and governance

•Design integration of automated tests into pipelines

•Design and implement a comprehensive testing strategy (including local tests, unit tests, integration tests, and load tests)

•Design and implement UI testing

•Implement orchestration of tools, such as GitHub Actions and Azure Pipelines

3.2 Design and implement a package management strategy

•Design a package management implementation that uses Azure Artifacts, GitHub Packages, NuGet, and npm

•Design and implement package feeds, including upstream sources

•Design and implement a dependency versioning strategy for code assets and packages, including semantic versioning and date-based

•Design and implement a versioning strategy for pipeline artifacts

3.3 Design and implement pipelines

•Select a deployment automation solution, including GitHub Actions and Azure Pipelines

•Design and implement an agent infrastructure, including cost, tool selection, licenses, connectivity, and maintainability

•Develop and implement pipeline trigger rules

•Develop pipelines, including classic and YAML

•Design and implement a strategy for job execution order, including parallelism and multi-stage

•Develop complex pipeline scenarios, such as containerized agents and hybrid

•Configure and manage self-hosted agents, including virtual machine (VM) templates and containerization

•Create reusable pipeline elements, including YAML templates, task groups, variables, and variable groups

•Design and implement checks and approvals by using YAML environments

3.4 Design and implement deployments

•Design a deployment strategy, including blue/green, canary, ring, progressive exposure, feature flags, and A/B testing

•Design a pipeline to ensure reliable order of dependency deployments

•Plan for minimizing downtime during deployments by using virtual IP address (VIP) swap, load balancer, and rolling deployments

•Design a hotfix path plan for responding to high-priority code fixes

•Implement load balancing for deployment, including Azure Traffic Manager and the Web Apps feature of Azure App Service

•Implement feature flags by using Azure App Configuration Feature Manager

•Implement application deployment by using containers, binary, and scripts

3.5 Design and implement infrastructure as code (IaC)

•Recommend a configuration management technology for application infrastructure

•Implement a configuration management strategy for application infrastructure, including IaC

•Define an IaC strategy, including source control and automation of testing and deployment

•Design and implement desired state configuration for environments, including Azure Automation State Configuration, Azure Resource Manager, Bicep, and Azure Automanage Machine Configuration

•Design and implement Azure Deployment Environments for on-demand self-deployment

3. 6 Maintain pipelines

•Monitor pipeline health, including failure rate, duration, and flaky tests

•Optimize pipelines for cost, time, performance, and reliability

•Analyze pipeline load to determine agent configuration and capacity

•Design and implement a retention strategy for pipeline artifacts and dependencies

Module 04: Develop a security and compliance plan

4. 1 Design & implement strategy for managing sensitive information in automation

•Implement and manage service connections

•Implement and manage personal access tokens

•Implement and manage secrets, keys, and certificates by using Azure Key Vault, GitHub secrets, and Azure Pipelines secrets

•Design and implement a strategy for managing sensitive files during deployment

•Design pipelines to prevent leakage of sensitive information

4.2 Automate security and compliance scanning

•Automate analysis of source code by using GitHub Advanced Security, including code scanning, secret scanning, and dependency scanning for both GitHub and Azure DevOps.

•Automate the use of pipeline-based scans, and SonarQube

•Automate security scanning, including container scanning and OWASP Zed Attack Proxy (ZAP)

•Automate analysis of licensing, vulnerabilities, and versioning of open-source components by using Mend Bolt and GitHub Dependency Scanning

•Integrate GitHub Advanced Security with Microsoft Defender for Cloud

Module 05: Implement an instrumentation strategy

5.1 Configure monitoring for a DevOps environment

•Configure and integrate monitoring by using Azure Monitor

•Configure and integrate with monitoring tools, such as Azure Monitor, Application Insights, and the Prometheus managed service

•Manage access control to the monitoring platform

•Configure alerts for pipeline events

5.2 Analyze metrics

•Inspect distributed tracing by using Application Insights

•Inspect application performance indicators

•Inspect infrastructure performance indicators, including CPU, memory, disk, and network

•Identify and monitor metrics for business value

•Analyze usage metrics by using Application Insights

•Interrogate logs using basic Kusto Query Language (KQL) queries