AZ 400 Microsoft Azure DevOps Solutions Training Curriculum

STRUCTURE

www.cromacampus.com | helpdesk@cromacampus.com | +91-120-4155255

AZ 400 Microsoft Azure DevOps Solutions Training Curriculum

"Become a DevOps Engineer Expert and learn how to deliver continuous business values with AZ 400 Microsoft Azure DevOps Solutions Training Program"

Course Objectives:

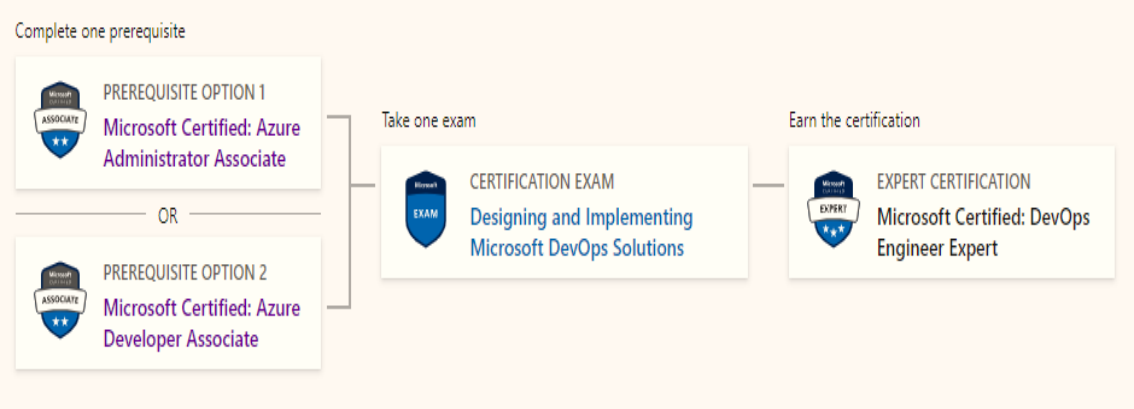
- Prepare yourself for the certification exam and clear your certification exam in the first attempt
- Add an attractive credential in your resume that is really appreciated by Companies.
- Improve your overall Cloud management skills, azure development skills, solution designing, implementation skills, and explore more job prospects with better salary packages.
- Boost your social media profiles especially LinkedIn by adding this certification and become one of the top persons to be chosen by industries.

AZ 400 Certification Training Description:

With our AZ 400 Certification Training Program, you will gain the subject matter expertise working with people, processes, and technologies to continuously deliver business value. The job role as a DevOps Engineer include designing and implementing strategies for collaboration, code, infrastructure, source control, security, compliance, continuous integration, testing, delivery, monitoring, and feedback.

Prerequisites for the Certification Exam:

A candidate for this certification must be familiar with both AZ 103 Azure administration and AZ 203 development and must be expert in at least one of these areas.



Here are some strong reasons why should you consider this certification course.

- Validate your technical skills like storage, networking, compute, security, and other Cloud operations on Microsoft Azure.
- Validate your solution designing and DevOps architect skills by successful implementation of cloud solutions at the workplace.
- Top-paying info-tech certification in the world.
- It provides you with global recognition for your knowledge, skills, and experience.
- The organization looks for those who know Oracle Cloud, AWS, Azure, etc.

Necessary Details about Certification You must Know

- Certification Name – *AZ 400 Microsoft Azure DevOps Solutions*
- Exam Duration: 150 minutes
- Number of Questions: 40-60
- Passing score: 700 (Out of 1000)
- Exam Cost: USD 165.00
- Validity: 2 years

Certification Exam Structure:

- Develop an instrumentation strategy (5-10%)
- Develop a Site Reliability Engineering (SRE) strategy (5-10%)
- Develop a security and compliance plan (10-15%)
- Manage source control (10-15%)
- Facilitate communication and collaboration (10-15%)
- Define and implement continuous integration (20-25%)
- Define & implement continuous delivery, release management strategy (10-15%)

Course Content:

Module 1: Introduction

- Course Overview
- Introduction to Cloud computing
- Introduction to Azure
- Self-hosting to cloud hosting
- Azure Services
- Azure Regions
- Introduction to DevOps?
- Where did DevOps come from?
- What problems led to the creation of DevOps?
- How is DevOps different from Traditional IT?
- Why you need DevOps?
- DevOps Implementation Features
- DevOps Lifecycle Phases and Measures
- Workflow in DevOps
- How DevOps Improved the Development and Operation Process?
- High Demand roles in DevOps
- Future scope of DevOps

Module 2: Develop an instrumentation strategy

- Design and implement logging
    - assess and Configure a log framework
    - design a log aggregation and storage strategy (e.g. Azure storage)
    - design a log aggregation using Azure Monitor
    - manage access control to logs (workspace-centric/resource-centric)
    - integrate crash analytics (App Center Crashes, Crashlytics)
- Design and implement telemetry

- design and implement distributed tracing
- inspect application performance indicators
- inspect infrastructure performance indicators
- define and measure key metrics (CPU, memory, disk, network)
- implement alerts on key metrics (email, SMS, webhooks, Teams/Slack)
- integrate user analytics (e.g. Application Insights funnels, Visual Studio App Center, TestFlight, Google Analytics)

- Integrate logging and monitoring solutions
  - configure and integrate container monitoring (Azure Monitor, Prometheus,)
  - configure and integrate with monitoring tools (Azure Monitor Application Insights, Dynatrace, New Relic, Nagios, Zabbix)
  - create feedback loop from platform monitoring tools (e.g. Azure Diagnostics VM extensions, Azure Platform Logs, Event Grid)
  - manage Access control to the monitoring platform

Module 3: Develop a Site Reliability Engineering (SRE) strategy

- Develop an actionable alerting strategy
  - identify and recommend metrics on which to base alerts
  - implement alerts using appropriate metrics
  - implement alerts based on appropriate log messages
  - implement alerts based on application health checks
  - analyze combinations of metrics
  - develop communication mechanism to notify users of degraded systems
  - implement alerts for self-healing activities (e.g. scaling, failovers)
- Design a failure prediction strategy
  - analyze behavior of system with regards to load and failure conditions
  - calculate when a system will fail under various conditions
  - measure baseline metrics for system
  - recommend the appropriate tools for a failure prediction strategy
- Design and implement a health check
  - analyze system dependencies to determine which dependency should be included in
  - health check
  - calculate healthy response timeouts based on SLO for the service
  - design approach for partial health situations
  - integrate health check with compute environment
  - implement different types of health checks (liveness, startup, shutdown)

Module 4: Develop a security and compliance plan (10-15%)

- Design an authentication and authorization strategy
  - design an access solution (Azure AD Privileged Identity Management (PIM), Azure AD
  - Conditional Access, MFA)
  - organize the team using Azure AD groups
  - implement Service Principals and Managed Identity
  - configure service connections
- Design a sensitive information management strategy

- evaluate and configure vault solution (Azure Key Vault, Hashicorp Vault)
- generate security certificates
- design a secrets storage and retrieval strategy
- formulate a plan for deploying secret files as part of a release

- Develop security and compliance
  - automate dependencies scanning for security (container scanning, OWASP)
  - automate dependencies scanning for compliance (licenses: MIT, GPL)
  - assess and report risks
  - design a source code compliance solution (e.g. GitHub security, pipeline-based scans, Git hooks, SonarQube)
- Design governance enforcement mechanisms
  - implement Azure policies to enforce organizational requirements
  - implement container scanning (e.g. static scanning, malware, crypto mining)
  - design and implement Azure Container Registry Tasks (Azure Policy)
  - design break-the-glass strategy for responding to security incidents

Module 5: Manage source control

- Develop a modern source control strategy
  - integrate/migrate disparate source control systems (e.g. GitHub, Azure Repos)
  - design authentication strategies
  - design approach for managing large binary files (e.g. Git LFS)
  - design approach for cross repository sharing (e.g. Git sub-modules, packages)
  - implement workflow hooks
- Plan and implement branching strategies for the source code
  - define Pull Requests (PR) guidelines to enforce work item correlation
  - implement branch merging restrictions (e.g. branch policies, branch protections, manual, etc.)
  - define branch strategy (e.g. trunk based, feature branch, release branch, GitHub flow)
  - design and implement a PR workflow (code reviews, approvals)
  - enforce static code analysis for code-quality consistency on PR
- Configure repositories
  - configure permissions in the source control repository
  - organize the repository with git-tags
  - plan for handling oversized repositories
  - plan for content recovery in all repository states
  - purge data from source control
- Integrate source control with tools
  - integrate GitHub with DevOps pipelines
  - integrate GitHub with identity management solutions (Azure AD)
  - design for GitOps
  - design for ChatOps
  - integrate source control artifacts for human consumption (e.g. Git changelog)

Module 6: Facilitate communication and collaboration

- Communicate deployment and release information with business stakeholders

- create dashboards combining boards, pipelines (custom dashboards on Azure DevOps)
- design a cost management communication strategy
- integrate release pipeline with work item tracking (e.g. AZ DevOps, Jira)
- integrate GitHub as repository with Azure Boards
- communicate user analytics

- Generate DevOps process documentation
  - design onboarding process for new employees
  - assess and document external dependencies (e.g. integrations, packages)
  - assess and document artifacts (version, release notes)
- Automate communication with team members
  - integrate monitoring tools with communication platforms (e.g. Teams, Slack, dashboards)
  - notify stakeholders about key metrics, alerts, severity using communication platform (e.g. Email, SMS, Slack, Teams)
  - integrate build and release with communication platforms (e.g. build fails, release fails)

Module 7: Define and implement continuous integration

- What is Continuous Delivery and Continuous Integration
- Steps for Continuous Integration
- Design a build and learn to automate it
- Design build automation
  - integrate the build pipeline with external tools (e.g., Dependency and security scanning, Code coverage)
  - implement quality gates (e.g. code coverage, internationalization, peer review)
  - design a testing strategy (e.g. integration, load, fuzz, API, chaos)
  - integrate multiple tools (e.g. GitHub Actions, Azure Pipeline, Jenkins)
- Design an application infrastructure management strategy
  - assess a configuration management mechanism for application infrastructure
  - define and enforce desired state configuration for environments
- Design a process for standardizing builds across organization
  - manage self-hosted build agents (VM templates, containerization, etc.)
  - create reusable build subsystems (YAML templates, Task Groups, Variable Groups, etc.)

Module 8: Design a Package Management Strategy

- Introduction to Packet Management
- What is Package management Strategy?
- Steps in defining packet management strategy
- Recommend package management tools (e.g. GitHub Packages, Azure Artifacts, Azure Automation Runbooks Gallery, Nuget, Jfrog, Artifactory)
- Design an Azure Artifacts implementation including linked feeds
- Design versioning strategy for code assets (e.g. SemVer, date based)
- Plan for assessing and updating and reporting package dependencies (GitHub Automated Security Updates, NuKeeper, GreenKeeper)

- Design a versioning strategy for packages (e.g. SemVer, date based)
- Design a versioning strategy for deployment artifacts

Module 9: Implement & Maintain a build strategy

- Implement a build strategy
  - design and implement build agent infrastructure (include cost, tool selection, licenses, maintainability)
  - develop and implement build trigger rules
  - develop build pipelines
  - design build orchestration (products that are composed of multiple builds)
  - integrate configuration into build process
  - develop complex build scenarios (e.g. containerized agents, hybrid, GPU)
- Maintain build strategy
  - monitor pipeline health (failure rate, duration, flaky tests)
  - optimize build (cost, time, performance, reliability)
  - analyze CI load to determine build agent configuration and capacity
  - manage pipeline health
  - identify the number of agents and jobs to run in parallel
  - investigate test failures

Module 10: Develop deployment scripts and templates

- Introduction to Templates
- Deployment Scripts
- Recommend a deployment solution
- Design and implement Infrastructure as code (ARM, Terraform, PowerShell, CLI)
- Develop application deployment process (container, binary, scripts)
- Develop database deployment process (migrations, data movement, ETL)
- Integrate configuration management as part of the release process
- Develop complex deployments

Module 11: Release Management Strategy

- Implement an orchestration automation solution
- Combine release targets depending on release deliverable (e.g., Infrastructure, code, assets, etc.)
- Design the release pipeline to ensure reliable order of dependency deployments
- Organize shared release configurations and process (YAML templates, variable groups)
- Design and implement release gates and approval processes
- Plan the deployment environment strategy
- Design a release strategy (blue/green, canary, ring)
- Implement the release strategy (using deployment slots, load balancer configurations, Azure Traffic Manager, feature toggle, etc.)
- Select the appropriate desired state solution for a deployment environment (PowerShell DSC, Chef, Puppet, etc.)
- Plan for minimizing downtime during deployments (VIP Swap, Load balancer, rolling deployments, etc.)

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

Module 12: Placement Guide

- What is an Interview?
- Tips to clear an Interview
- Common Interview questions and answers
- AZ 400 Interview Questions and Answers
- Resume Building Guide
- Attempt for AZ 400 Global Certification Exam
- Start applying for Jobs