# Azure AZ-400

# Microsoft Azure DevOps Solutions Fast Track

## Overview

## Duration-5Days

Azure Developer Associate contains courseware that helps prepare students for Exam AZ-400. Passing this exam is required to earn the Azure Developer Associate certification.

## Audience

Students in this course are interested in implementing DevOps processes or in passing the Microsoft Azure DevOps Solutions certification exam.

## Prerequisites

Fundamental knowledge about Azure, version control, Agile software development, and core software development principles. It would be helpful to have experience in an organization that delivers software.

## Course Objectives

After completing this course, students will be able to:

- Describe the benefits of using source control
- Migrate from TFVC to Git
- Scale Git for Enterprise DevOps
- Implement and manage build infrastructure Manage application config & secrets
- Implement a mobile DevOps strategy
- Explain why continuous integration matters
- Implement continuous integration using Azure DevOps
- Configure builds and the options available
- Create an automated build workflow
- Integrate other build tooling with Azure DevOps
- Create hybrid build processes
- Differentiate between a release and a deployment
- Define the components of a release pipeline

- Explain things to consider when designing your release strategy

- Classify a release versus a release process, and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely, using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using, alerts, service hooks and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating · Configure secure access to package feeds

- Apply infrastructure and configuration as code principles
- Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI

- Describe deployment models and services that are available with Azure
- Deploy and configure a Managed Kubernetes cluster
- Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, SaltStack, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
- Implement compliance and security in your application infrastructure
- Describe what is meant by code quality and how it is measured
- Detect code smells
- Integrate automated tests for code quality
- Report on code coverage during testing
- Add tooling to measure technical debt
- Detect open source and other licensing issues
- Implement a container build strategy
- Design practices to measure end-user satisfaction
- Design processes to capture and analyze user feedback from external sources
- Design routing for client application crash report data
- Recommend monitoring tools and technologies
- Recommend system and feature usage tracking tools
- Configure crash report integration for client applications
- Develop monitoring and status dashboards
- Implement routing for client application crash report data
- Implement tools to track system usage, feature usage, and flow
- Integrate and configure ticketing systems with development team's work management system
- Analyze alerts to establish a baseline
- Analyze telemetry to establish a baseline
- Perform live site reviews and capture feedback for system outages
- Perform ongoing tuning to reduce meaningless or non-actionable alerts
- Plan for the transformation with shared goals and timelines.
- Select a project and identify project metrics and KPIs.
- Create a team and agile organizational structure.
- Develop a project quality strategy.
- Plan for secure development practices and compliance rules.
- Migrate and consolidate artifacts.

- Migrate and integrate source control measures.

## Outline: Microsoft Azure DevOps Solutions Fast Track (AZ400)

### Getting started with Source Control

- What is Source Control?
- Benefits of Source Control
- Types of source control systems
- Introduction to Azure Repos
- Migrating from TFVC to Git
- Authenticating to your Git Repos

### Scaling git for enterprise DevOps

- How to structure your git repo? Mono Repo or Multi Repo?
- Git Branching workflows
- Collaborating with Pull Requests
- Why care about GitHooks?
- Fostering Internal Open Source
- Git Version
- public projects
- Storing Large files in Git

### Implement & Manage Build Infrastructure

- The concept of pipelines in DevOps
- Azure Pipelines
- Evaluate use of Hosted vs. Private Agents
- Agent pools
- Pipelines & Concurrency
- Azure DevOps loves Open Source projects
- Azure Pipelines YAML vs Visual Designer
- Setup private agents
- Integrate Jenkins with Azure Pipelines
- Integration external source control with Azure Pipelines
- Analyse & Integrate Docker multi stage builds

**Managing application config & secrets**

- Demo: SQL Injection attack
- Implement secure & compliant development process
- Rethinking application config data
- Manage secrets, tokens & certificates
- Implement tools for managing security and compliance in pipeline

**Implement a mobile DevOps strategy**

- Introduction to Visual Studio App Center
- Manage mobile target device sets and distribution groups
- Manage target UI test device sets
- Provision tester devices for deployment

**Implementing Continuous Integration in an Azure DevOps Pipeline**

- Continuous Integration Overview
- Implementing a Build Strategy

**Managing Code Quality and Security Policies**

- Managing Code Quality
- Managing Security Policies

**Implementing a Container Build Strategy**

- Implementing a Container Build Strategy

**Design a Release Strategy**

- Introduction to Continuous Delivery
- Release strategy recommendations
- Building a High Quality Release pipeline
- Choosing a deployment pattern
- Choosing the right release management tool
- Building a release strategy
- Differentiate between a release and a deployment
- Define the components of a release pipeline

- Explain things to consider when designing your release strategy
- Classify a release versus a release process, and outline how to control the quality of both
- Describe the principle of release gates and how to deal with release notes and documentation
- Explain deployment patterns, both in the traditional sense and in the modern sense
- Choose a release management tool

**Set up a Release Management Workflow**

- Introduction
- Create a Release Pipeline
- Provision and Configure Environments
- Manage And Modularize Tasks and Templates
- Integrate Secrets with the release pipeline
- Configure Automated Integration and Functional Test Automation
- Automate Inspection of Health
- Building a release management workflow
- Explain the terminology used in Azure DevOps and other Release Management Tooling
- Describe what a Build and Release task is, what it can do, and some available deployment tasks
- Classify an Agent, Agent Queue and Agent Pool
- Explain why you sometimes need multiple release jobs in one release pipeline
- Differentiate between multi-agent and multi-configuration release job
- Use release variables and stage variables in your release pipeline
- Deploy to an environment securely, using a service connection
- Embed testing in the pipeline
- List the different ways to inspect the health of your pipeline and release by using, alerts, service hooks and reports
- Create a release gate

**Implement an appropriate deployment pattern**

- Introduction into Deployment Patterns
- Implement Blue Green Deployment
- Implement Canary Release

- Implement Progressive Exposure Deployment
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment

**Hands-On Lab: Microsoft 365 Tenant and Service Management**

- Exercise 1: Set up a Microsoft 365 trial tenant
- Exercise 2: Managing Microsoft 365 users, groups, and administration
- Exercise 3: Configuring Rights Management and compliance
- Exercise 4: Monitor and troubleshoot Microsoft 365

**Designing a Dependency Management Strategy**

- Introduction
- Packaging dependencies
- Package management
- Implement versioning strategy
- Recommend artifact management tools and practices
- Abstract common packages to enable sharing and reuse
- Inspect codebase to identify code dependencies that can be converted to packages
- Identify and recommend standardized package types and versions across the solution
- Refactor existing build pipelines to implement version strategy that publishes packages
- Manage security and compliance

**Manage security and compliance**

- Introduction
- Package security
- Open source software
- Integrating license and vulnerability scans
- Inspect open source software packages for security and license compliance to align with corporate standards
- Configure build pipeline to access package security and license rating · Configure secure access to package feeds

**Infrastructure and Configuration Azure Tools**

- Learning Objectives
- Infrastructure as Code and Configuration Management
- Create Azure Resources using ARM Templates
- Create Azure Resources using Azure CLI
- Create Azure Resources by using Azure PowerShell
- Additional Automation Tools
- Version Control
- Lab Deploy to Azure using ARM templates
- Module Review Questions

**Azure Deployment Models and Services**

- Learning Objectives
- Deployment Models and Options
- Azure Infrastructure-as-a-Service (IaaS) Services
- Azure Automation with DevOps
- Desired State Configuration (DSC)
- Azure Platform-as-a-Service (PaaS) services
- Azure Service Fabric
- Lab Azure Automation - IaaS or PaaS deployment
- Module Review Questions

**Create and Manage Kubernetes Service Infrastructure**

- Learning Objectives
- Azure Kubernetes Service
- Lab Deploy and Scale AKS Cluster
- Module Review Questions

**Third Party and Open Source Tools available with Azure**

- Learning Objectives
- Chef
- Puppet
- Ansible
- Cloud-Init
- Terraform
- Lab Provision and configure an App in Azure Using X

- Module Review Questions

**Implement Compliance and Security in your Infrastructure**

- Security and Compliance Principles with DevOps
- Azure Security Center
- Lab Integrate a scanning extension or tool in an AZ DevOps pipeline/security center
- Module Review Questions

**Planning for DevOps**

- Transformation Planning
- Project Selection
- Team Structures

**Planning for Quality and Security**

- Planning a Quality Strategy
- Planning Secure Development

**Migrating and Consolidating Artifacts and Tools**

- Migrating and Consolidating Artifacts
- Migrating and Integrating Source Control