AZ-400T00 Designing and Implementing Microsoft DevOps Solutions

Class Length: 5 Days

Overview

This course provides the knowledge and skills to design and implement DevOps processes and practices. Students will learn how to plan for DevOps, use source control, scale Git for an enterprise, consolidate artifacts, design a dependency management strategy, manage secrets, implement continuous integration, implement a container build strategy, design a release strategy, set up a release management workflow, implement a deployment pattern, and optimize feedback mechanisms.

Prerequisite Comments

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.

Target Audience

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

Course Objectives

Plan for the transformation with shared goals and timelines
Select a project and identify project metrics and KPIs
Create a team and agile organization structure
Describe the benefits of using Source Control
Migrate from TFVC to Git
Scale Git for Enterprise DevOps
Recommend artifact management tools and practices
Abstract common packages to enable sharing and reuse
Migrate and consolidate artifacts
Migrate and integrate source control measures
Manage application config and secrets
Develop a project quality strategy
Plan for secure development practices and compliance rules
Implement and manage build infrastructure
Explain why continuous integration matters
Implement continuous integration using Azure DevOps
Manage code quality including: technical debt, SonarCloud, and other tooling solutions
Manage security policies with open source, OWASP, and WhiteSource Bolt
Implement a container strategy including how containers are different from virtual machines and how microservices use containers
Implement containers using Docker
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
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
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

Course Outline

1 - Planning for DevOps

Transformation Planning
Project Selection
Team Structures
Migrating to Azure DevOps

2 - Getting started with Source Control

What is Source Control
Benefits of Source Control
Types of Source Control Systems
Introduction to Azure Repos
Introduction to GitHub
Migrating from Team Foundation Version Control (TFVC) to Git in Azure Repos
Authenticating to Git in Azure Repos

3 - Scaling git for enterprise DevOps

How to Structure your Git Repo
Git Branching Workflows
Collaborating with Pull Requests in Azure Repos
Why care about GitHooks
Fostering Inner Source

4 - Consolidating Artifacts & Designing a Dependency Management Strategy

Packaging Dependencies
Package Management
Migrating and Consolidating Artifacts

5 - Implementing Continuous Integration with Azure Pipelines

The concept of pipelines in DevOps
Azure Pipelines
Evaluate use of Hosted vs Private Agents
Agent Pools
Pipelines and Concurrency
Azure DevOps and Open Source Projects (Public Projects)
Azure Pipelines YAML vs Visual Designer
Continuous Integration Overview
Implementing a Build Strategy
Integration with Azure Pipelines
Integrate External Source Control with Azure Pipelines
Set Up Private Agents
Analyze and Integrate Docker Multi-Stage Builds

6 - Managing Application Config and Secrets

Introduction to Security
Implement secure and compliant development process
Rethinking application config data
Manage secrets, tokens, and certificates
Implement tools for managing security and compliance in a pipeline

7 - Managing Code Quality and Security Policies

Managing Code Quality
Managing Security Policies

8 - Implementing a Container Build Strategy

Implementing a Container Build Strategy

9 - Manage Artifact versioning, security & compliance

Package security
Open source software
Integrating license and vulnerability scans
Implement a versioning strategy

10 - 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

11 - Set up a Release Management Workflow

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

12 - Implement an appropriate deployment pattern

Introduction to Deployment Patterns
Implement Blue Green Deployment
Feature Toggles
Canary Releases
Dark Launching
AB Testing
Progressive Exposure Deployment

13 - Implement process for routing system feedback to development teams

Implement Tools to Track System Usage, Feature Usage, and Flow
Implement Routing for Mobile Application Crash Report Data
Develop Monitoring and Status Dashboards
Integrate and Configure Ticketing Systems

14 - Infrastructure and Configuration Azure Tools

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
Desired State Configuration (DSC)
Azure Automation with DevOps
Additional Automation Tools

15 - Azure Deployment Models and Services

Deployment Modules and Options
Azure Infrastructure-as-a-Service (IaaS) Services
Azure Platform-as-a-Service (PaaS) services
Serverless and HPC Computer Services
Azure Service Fabric

16 - Create and Manage Kubernetes Service Infrastructure

Azure Kubernetes Service

17 - Third Party Infrastructure as Code Tools available with Azure

Chef
Puppet
Ansible
Terraform

18 - Implement Compliance and Security in your Infrastructure

Security and Compliance Principles with DevOps
Azure security Center

19 - Recommend and design system feedback mechanisms

The inner loop
Continuous Experimentation mindset
Design practices to measure end-user satisfaction
Design processes to capture and analyze user feedback
Design process to automate application analytics

20 - Optimize feedback mechanisms

Site Reliability Engineering
Analyze telemetry to establish a baseline
Perform ongoing tuning to reduce meaningless or non-actionable alerts
Analyze alerts to establish a baseline
Blameless Retrospectives and a Just Culture