Course Outline

AZ-400T00 A: Designing and Implementing Microsoft DevOps Solutions

Course Summary

Description

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.

Objectives

At the end of this course, students will be able to:

- 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

Course Outline

AZ-400T00 A: Designing and Implementing Microsoft DevOps Solutions

Course Summary

- 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 hoo ks, and reports
- Create a release gate
- Describe deployment patterns
- Implement Blue Green Deployment
- Implement Canary Release
- Implement Progressive Exposure Deployment
- 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
- Implement a mobile DevOps strategy
- 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, Pu ppet, Ansible, SaltStack, and Terraform
- Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline an d application infrastructure
- Implement compliance and security in your application infrastructure
- 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
- 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

Course Outline

AZ-400T00 A: Designing and Implementing Microsoft DevOps Solutions

Course Summary

Topics

- Planning for DevOps
- Getting started with Source Control
- Scaling Git for enterprise DevOps
- Consolidating Artifacts & Designing a Dependency Management Strategy
- Implementing Continuous Integration with Azure Pipelines
- Managing Application Config and Secrets
- Managing Code Quality and Security Policies
- Implementing a Container Build Strategy
- Manage Artifact versioning, security & compliance
- Design a Release Strategy
- Set up a Release Management Workflow
- Implement an appropriate deployment pattern
- Implement process for routing system feedback to development teams
- Implement a mobile DevOps strategy
- Infrastructure and Configuration Azure Tools
- Azure Deployment Models and Services
- Create and Manage Kubernetes Service Infrastructure
- Third Party Infrastructure as Code Tools available with Azure
- Implement Compliance and Security in your Infrastructure
- Recommend and design system feedback mechanisms
- Optimize feedback mechanisms

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.

Duration

Five Days

Course Outline

AZ-400T00 A: Designing and Implementing Microsoft DevOps Solutions

Course Outline

I. Planning for DevOps
- A. Transformation Planning
- B. Project Selection
- C. Team Structures
- D. Migrating to Azure DevOps

Lab : Agile Planning and Portfolio Management with Azure Boards

II. Getting started with Source Control
- A. What is Source Control
- B. Benefits of Source Control
- C. Types of Source Control Systems
- D. Introduction to Azure Repos
- E. Introduction to GitHub
- F. Migrating from Team Foundation Version Control (TFVC) to Git in Azure Repos
- G. Authenticating to Git in Azure ReposLab : Version Controlling with Git

III. Scaling Git for enterprise DevOps
- A. How to Structure your Git Repo
- B. Git Branching Workflows
- C. Collaborating with Pull Requests in Azure Repos
- D. Why care about GitHooks
- E. Fostering Inner Source

Lab : Code Review with Pull Requests

IV. Consolidating Artifacts & Designing a Dependency Management Strategy
- A. Packaging Dependencies
- B. Package Management
- C. Migrating and Consolidating Artifacts

Lab : Updating Packages

V. Implementing Continuous Integration with Azure Pipelines
- A. The concept of pipelines in DevOps
- B. Azure Pipelines
- C. Evaluate use of Hosted vs Private Agents
- D. Agent Pools
- E. Pipelines and Concurrency
- F. Azure DevOps and Open Source Projects (Public Projects)
- G. Azure Pipelines YAML vs Visual Designer
- H. Continuous Integration Overview
- I. Implementing a Build Strategy
- J. Integration with Azure Pipelines
- K. Integrate External Source Control with Azure Pipelines
- L. Set Up Private Agents
- M. Analyze and Integrate Docker Multi-Stage Builds

Lab : Enabling Continuous Integration with Azure Pipelines
Lab : Integrating External Source Control with Azure Pipelines
Lab : Integrate Jenkins with Azure Pipelines
Lab : Deploying a Multi-Container Application

VI. Managing Application Config and Secrets
- A. Introduction to Security
- B. Implement secure and compliant development process
- C. Rethinking application config data
- D. Manage secrets, tokens, and certificates
- E. Implement tools for managing security and compliance in a pipeline

Lab : Integrating Azure Key Vault with Azure DevOps

VII. Managing Code Quality and Security Policies
- A. Managing Code Quality
- B. Managing Security Policies

Lab : Managing Technical Debt with Azure DevOps and SonarCloud

VIII. Implementing a Container Build Strategy
- A. Implementing a Container Build Strategy

Lab : Modernizing Existing ASP.NET Apps with Azure

IX. Manage Artifact versioning, security & compliance
- A. Package security
- B. Open source software
- C. Integrating license and vulnerability scans
- D. Implement a versioning strategy (git version)

Lab : Manage Open Source Security and License with WhiteSource

X. Design a Release Strategy
- A. Introduction to Continuous Delivery
- B. Release strategy recommendations
- C. Building a High-Quality Release pipeline
- D. Choosing a deployment pattern
- E. Choosing the right release management tool

XI. Set up a Release Management Workflow
- A. Create a Release Pipeline
- B. Provision and Configure Environments
- C. Manage and Modularize Tasks and Templates
- D. Integrate Secrets with the release pipeline
- E. Configure Automated Integration and Functional Test Automation
- F. Automate Inspection of Health

Lab : Configuring Pipelines as Code with YAML
Lab : Setting up secrets in the pipeline with Azure Key vault
Lab : Setting up and Running Functional Tests
Lab : Using Azure Monitor as release gate
Lab : Creating a release Dashboard

Course Outline

AZ-400T00 A: Designing and Implementing Microsoft DevOps Solutions

Course Outline (cont'd)

XII. Implement an appropriate deployment pattern
- A. Introduction to Deployment Patterns
- B. Implement Blue Green Deployment
- C. Feature Toggles
- D. Canary Releases
- E. Dark Launching
- F. AB Testing
- G. Progressive Exposure Deployment

Lab : Feature Flag Management with LaunchDarkly and Azure DevOps

XIII. Implement process for routing system feedback to development teams
- A. Implement Tools to Track System Usage, Feature Usage, and Flow
- B. Implement Routing for Mobile Application Crash Report Data
- C. Develop Monitoring and Status Dashboards
- D. Integrate and Configure Ticketing Systems

Lab : Monitoring Application Performance

XIV. Implement a mobile DevOps strategy
- A. Introduction to Mobile DevOps
- B. Introduction to Visual Studio App Center
- C. Manage mobile target device sets and distribution groups
- D. Manage target UI test device sets
- E. Provision tester devices for deployment
- F. Create public and private distribution groups

XV. Infrastructure and Configuration Azure Tools
- A. Infrastructure as Code and Configuration Management
- B. Create Azure Resources using ARM Templates
- C. Create Azure Resources using Azure CLI
- D. Create Azure Resources by using Azure PowerShell
- E. Desired State Configuration (DSC)
- F. Azure Automation with DevOps
- G. Additional Automation Tools

Lab : Azure Deployments using Resource Manager Templates

XVI. Azure Deployment Models and Services
- A. Deployment Modules and Options
- B. Azure Infrastructure-as-a-Service (IaaS) Services
- C. Azure Platform-as-a-Service (PaaS) services
- D. Serverless and HPC Computer Services
- E. Azure Service Fabric

Lab : Azure Automation - IaaS or PaaS deployment

XVII. Create and Manage Kubernetes Service Infrastructure
- A. Azure Kubernetes Service

Lab : Deploying a multi-container application to Azure Kubernetes Service

XVIII. Third Party Infrastructure as Code Tools available with Azure
- A. Chef
- B. Puppet
- C. Ansible
- D. Terraform

Lab : Infrastructure as Code

Lab : Automating Your Infrastructure Deployments in the Cloud with Terraform and Azure Pipelines

XIX. Implement Compliance and Security in your Infrastructure
- A. Security and Compliance Principles with DevOps
- B. Azure security Center

Lab : Implement Security and Compliance in an Azure DevOps Pipeline

XX. Recommend and design system feedback mechanisms
- A. The inner loop
- B. Continuous Experimentation mindset
- C. Design practices to measure end-user satisfaction
- D. Design processes to capture and analyze user feedback
- E. Design process to automate application analytics

Lab : Integration between Azure DevOps and Teams

XXI. Optimize feedback mechanisms
- A. Site Reliability Engineering
- B. Analyze telemetry to establish a baseline
- C. Perform ongoing tuning to reduce meaningless or non-actionable alerts
- D. Analyze alerts to establish a baseline
- E. Blameless Retrospectives and a Just Culture