Microsoft Azure DevOps Solutions Fast Track (AZ-400)

Duration: 5 days

Course Overview

This seven-MOC packaged set aligned to Azure Exam: 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.

Courses in this packaged set:

1. **AZ-400T01: Implementing DevOps Development Processes**
2. **AZ-400T02: Implementing Continuous Integration**
3. **AZ-400T03: Implementing Continuous Delivery**
4. **AZ-400T04: Implementing Dependency Management**
5. **AZ-400T05: Implementing Application Infrastructure**
6. **AZ-400T06: Implementing Continuous Feedback**
7. **AZ-400T07: Designing a DevOps Strategy**

Who should attend

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.

To become a Microsoft Certified: Azure DevOps Engineer Expert, you must either earn the Azure Administrator Associate or Azure Developer Associate certification.

Great pre-requisite courses for those certifications are Microsoft Azure Administrator (AZ-103) or Developing Solutions for Microsoft Azure (AZ-203)

Course Objectives

1. **After completing this course, students will be able to:**

2. **Describe the benefits of using source control**
3. **Migrate from TFVC to Git**
4. **Scale Git for Enterprise Dev Ops**
5. **Implement and manage build infrastructure Manage application config & secrets**
6. **Implement a mobile DevOps strategy**

7. Explain why continuous integration matters
8. Implement continuous integration using Azure DevOps
9. Configure builds and the options available
10. Create an automated build workflow
11. Integrate other build tooling with Azure DevOps
12. Create hybrid build processes
13. Differentiate between a release and a deployment
14. Define the components of a release pipeline
15. Explain things to consider when designing your release strategy
16. Classify a release versus a release process, and outline how to control the quality of both
17. Describe the principle of release gates and how to deal with release notes and documentation
18. Explain deployment patterns, both in the traditional sense and in the modern sense
19. Choose a release management tool
20. Explain the terminology used in Azure DevOps and other Release Management Tooling
21. Describe what a Build and Release task is, what it can do, and some available deployment tasks
22. Classify an Agent, Agent Queue and Agent Pool
23. Explain why you sometimes need multiple release jobs in one release pipeline
24. Differentiate between multi-agent and multi-configuration release job
25. Use release variables and stage variables in your release pipeline
26. Deploy to an environment securely, using a service connection
27. Embed testing in the pipeline
28. List the different ways to inspect the health of your pipeline and release by using, alerts, service hooks and reports
29. Create a release gate
30. Describe deployment patterns
31. Implement Blue Green Deployment
32. Implement Canary Release
33. Implement Progressive Exposure Deployment
34. Recommend artifact management tools and practices
35. Abstract common packages to enable sharing and reuse
36. Inspect codebase to identify code dependencies that can be converted to packages
37. Identify and recommend standardized package types and versions across the solution
38. Refactor existing build pipelines to implement version strategy that publishes packages
39. Manage security and compliance
40. Inspect open source software packages for security and license compliance to align with corporate standards
41. Configure build pipeline to access package security and license rating
42. Configure secure access to package feeds

43. Apply infrastructure and configuration as code principles
44. Deploy and manage infrastructure using Microsoft automation technologies such as ARM templates, PowerShell, and Azure CLI
45. Describe deployment models and services that are available with Azure
46. Deploy and configure a Managed Kubernetes cluster
47. Deploy and configure infrastructure using 3rd party tools and services with Azure, such as Chef, Puppet, Ansible, SaltStack, and Terraform
48. Define an infrastructure and configuration strategy and appropriate toolset for a release pipeline and application infrastructure
49. Implement compliance and security in your application infrastructure
50. Describe what is meant by code quality and how it is measured
51. Detect code smells
52. Integrate automated tests for code quality
53. Report on code coverage during testing
54. Add tooling to measure technical debt
55. Detect open source and other licensing issues
56. Implement a container build strategy
57. Design practices to measure end-user satisfaction
58. Design processes to capture and analyze user feedback from external sources
59. Design routing for client application crash report data
60. Recommend monitoring tools and technologies
61. Recommend system and feature usage tracking tools
62. Configure crash report integration for client applications
63. Develop monitoring and status dashboards
64. Implement routing for client application crash report data
65. Implement tools to track system usage, feature usage, and flow
66. Integrate and configure ticketing systems with development team's work management system
67. Analyze alerts to establish a baseline
68. Analyze telemetry to establish a baseline
69. Perform live site reviews and capture feedback for system outages
70. Perform ongoing tuning to reduce meaningless or non-actionable alerts
71. Plan for the transformation with shared goals and timelines.
72. Select a project and identify project metrics and KPIs.
73. Create a team and agile organizational structure.
74. Develop a project quality strategy.
75. Plan for secure development practices and compliance rules.
76. Migrate and consolidate artifacts.
77. Migrate and integrate source control measures.

Course Content

1. Getting started with Source Control
2. Scaling git for enterprise Dev Ops
3. Implement & Manage Build Infrastructure
4. Managing application config & secrets

5. **Implement a mobile DevOps strategy**
6. **Implementing Continuous Integration in an Azure DevOps Pipeline**
7. **Managing Code Quality and Security Policies**
8. **Implementing a Container Build Strategy**
9. **Design a Release Strategy**
10. **Set up a Release Management Workflow**
11. **Implement an appropriate deployment pattern**
12. **Hands-On Lab: Microsoft 365 Tenant and Service Management**
13. **Designing a Dependency Management Strategy**
14. **Manage security and compliance**
15. **Infrastructure and Configuration Azure Tools**
16. **Azure Deployment Models and Services**
17. **Create and Manage Kubernetes Service Infrastructure**
18. **Third Party and Open Source Tools available with Azure**
19. **Implement Compliance and Security in your Infrastructure**
20. **Planning for DevOps**
21. **Planning for Quality and Security**
22. **Migrating and Consolidating Artifacts and Tools**