Continuous Delivery Using Azure DevOps Services

- Get hands-on experience working as a team in an Azure DevOps environment.
- Learn to configure Azure DevOps for product development.
- Practice creating and managing Azure Pipeline builds, as well as installing and configuring Azure Pipeline agents.
- Understand the value of building a culture of learning and improvement

This course provides students with the DevOps principles and related hands-on practices to work better as a team, scale their agility, share and integrate their work, and deliver working software continuously in order to enable faster delivery of value and receive early and valuable feedback. Utilizing more than 50 hands-on activities, students work as a team, a pair, or individually using the tools that reinforce the DevOps principles and practices they are learning. Students will self-organize into cross-functional, collocated teams and work collaboratively on a case study using a shared Azure DevOps environment. All code will be provided.

Who Should Attend

This course is intended for experienced software development professionals who want to learn about DevOps in order to achieve Continuous Integration, Continuous Delivery, Continuous Feedback, and Continuous Learning in a technical value stream as supported by Azure DevOps Services, Visual Studio, and Azure in order to continually deliver working software at scale. Students will also install and evaluate several extensions from the Azure DevOps Marketplace. Those who use Azure DevOps Server (Team Foundation Server) will also benefit from this course. Attendees should be familiar with C#, Visual Studio, Scrum, and have basic experience with Azure DevOps Services or Azure DevOps Server.

Course Outline

Increasing Flow at Scale
The complexity of software development
The need for empirical process control
Increasing flow through a technical value stream
Scrum and Professional Scrum
The Nexus scaled Scrum framework
Practices for organizing teams
Establishing feature teams to minimize dependencies

Planning and Executing at Scale
Organizing and refining the Product Backlog
Creating a definition of “Ready”
Dependencies, types, and related risks
Cross-team refinement to identify dependencies
Planning and executing a Sprint
Limiting work in progress (WIP)
Working in small batches
Creating and obeying a definition of “Done”
Using queries, charts, and dashboards for reporting

Delivering Continuously
Azure Pipelines deployment
Release definitions, environments, and releases
Deployment targets, IaaS, PaaS, containers
Using Microsoft Azure for DevOps
Configuring endpoints and deployment groups
Automated deployment to an Azure Virtual Machine
Installing and configuring Azure Pipelines agents
Release tasks and phases
Creating and deploying a release
Infrastructure as Code
Creating and importing YAML builds
Automatic creation of environments
Azure Resource Manager and ARM templates
Release and environment triggers
Continuous Delivery (CD)

Empowering the Product Owner
Build-Measure-Learn explained
Hypothesis-Driven Development (HDD)

Sharing Code
Working collaboratively as a team
Collective ownership mindset
Git version control workflow (optional)
Branching strategies and related side effects
Using Code Maps to visualize code dependencies
Using Package Management to share binaries
Adopting an internal open source model

Integrating Continuously
Why and how to create fast feedback loops
The importance of automated testing
Unit testing in Visual Studio
Automated builds in Azure Pipelines
Build definitions and build tasks
Cloning and managing build definitions
Hosted build agents and agent pools
Running tests during an automated build
Code coverage and regression testing
Configuring and using Test Impact Analysis
Continuous Integration (CI) and CI+
Customizing Azure DevOps to implement HDD
Feature flags overview
Using LaunchDarkly to manage feature flags
Telemetry and application performance management
Application Insights for gathering telemetry
A/B testing
Using feature flags to support A/B testing
Exploratory testing, testing “tours” practice
Using the Microsoft Test and Feedback extension
Understanding and identifying technical debt
Using SonarQube to gauge your technical debt
Practices for paying off technical debt

Learning and Improving Continuously
Building a culture of learning and improvement
Agile metrics and reporting
Communities of Practice (COPs)
Lean thinking and practices to eliminate waste
Using the wiki to build tribal knowledge