Continuous Delivery Using Azure DevOps Services

Course Summary

Description

This two-day 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.

Objectives

After taking this course, students will be able to understand:

- DevOps principles, benefits, and related practices
- The Three Ways: flow, feedback, and learning
- Technical excellence at scale
- Working as a team in an Azure DevOps environment
- Increasing flow through a technical value stream
- Practices and mindset of a Professional Scrum Team
- Scaled Professional Scrum (Nexus) framework
- Identifying and eliminating dependencies
- Minimizing dependencies through feature teams
- Configuring Azure DevOps for product development
- Planning work at scale: epics, features, user stories
- Product Backlog Refinement benefits and practices
- Creating a definition of “Ready”
- Types of dependencies and the related risks
- Cross-team refinement to mitigate dependencies
- Working in small batches and limiting WIP
- Using Azure Boards to plan and execute a Sprint
- Creating and abiding-by a definition of “Done”
- Reporting data via queries, charts, and widgets
- Using dashboards as information radiators
- Collaborating as a cross-functional team
- The collective ownership mindset
- Git version control primer and workflows
- Branching strategies and potential side effects
- Visualizing and managing code dependencies
- Adopting an internal open source model
- Creating fast feedback loops
- Automated testing in Visual Studio and Azure Pipelines
- Writing and running unit tests in Visual Studio
- Creating and managing Azure Pipeline builds
- Installing and configuring Azure Pipeline agents
- Queuing and monitoring automated builds
- Running automated tests as part of a build
- Collecting code coverage information
- Using Test Impact Analysis to improve performance
- Continuous Integration (CI) and CI+
- Using Azure Pipelines for deployment
- Using Microsoft Azure for DevOps
- Automated deployment to an Azure Virtual Machine
- Creating and deploying a release
- Infrastructure as Code (IaC)
- Creating and importing YAML builds
- Automatic creation of deployment environments

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

Continuous Delivery Using Azure DevOps Services

Course Summary (cont'd)

- Azure Resource Manager and ARM templates
- Continuous Delivery (CD) and using RM triggers
- Hypothesis-Driven Development (HDD)
- Customizing Azure DevOps to implement HDD
- Using feature flags to manage feature availability
- Telemetry and application performance management
- Using feature flags to support A/B testing
- Using LaunchDarkly and Application Insights for HDD
- Exploratory testing, testing "tours" practice
- Using the Microsoft Test and Feedback extension
- Understanding, identifying, and avoiding technical debt
- Using SonarQube to measure your technical debt
- Building a culture of learning and improvement
- Agile metrics and reporting
- Communities of Practice (COPs)
- Recording learnings in the wiki for transparency

Topics

- Increasing Flow at Scale
- Planning and Executing at Scale
- Sharing Code
- Integrating Continuously
- Delivering Continuously
- Empowering the Product Owner
- Learning and Improving Continuously

Audience

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.

Prerequisites

Attendees should be familiar with C#, Visual Studio, Scrum, and have basic experience with Azure DevOps Services or Azure DevOps Server.

Duration

Two days

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

Continuous Delivery Using Azure DevOps Services

Course Outline

I. *Increasing Flow at Scale*
   - A. The complexity of software development
   - B. The need for empirical process control
   - C. Increasing flow through a technical value stream
   - D. Scrum and Professional Scrum
   - E. The Nexus scaled Scrum framework
   - F. Practices for organizing teams
   - G. Establishing feature teams to minimize dependencies

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

III. *Sharing Code*
   - A. Working collaboratively as a team
   - B. Collective ownership mindset
   - C. Git version control workflow (optional)
   - D. Branching strategies and related side effects
   - E. Using Code Maps to visualize code dependencies
   - F. Using Package Management to share binaries
   - G. Adopting an internal open source model

IV. *Integrating Continuously*
   - A. Why and how to create fast feedback loops
   - B. The importance of automated testing
   - C. Unit testing in Visual Studio
   - D. Automated builds in Azure Pipelines
   - E. Build definitions and build tasks
   - F. Cloning and managing build definitions
   - G. Hosted build agents and agent pools
   - H. Running tests during an automated build
   - I. Code coverage and regression testing
   - J. Configuring and using Test Impact Analysis
   - K. Continuous Integration (CI) and CI+

V. *Delivering Continuously*
   - A. Azure Pipelines deployment
   - B. Release definitions, environments, and releases
   - C. Deployment targets, IaaS, PaaS, containers
   - D. Using Microsoft Azure for DevOps
   - E. Configuring endpoints and deployment groups
   - F. Automated deployment to an Azure Virtual Machine
   - G. Installing and configuring Azure Pipelines agents
   - H. Release tasks and phases
   - I. Creating and deploying a release
   - J. Infrastructure as Code
   - K. Creating and importing YAML builds
   - L. Automatic creation of environments
   - M. Azure Resource Manager and ARM templates
   - N. Release and environment triggers
   - O. Continuous Delivery (CD)

VI. *Empowering the Product Owner*
   - A. Build-Measure-Learn explained
   - B. Hypothesis-Driven Development (HDD)
   - C. Customizing Azure DevOps to implement HDD
   - D. Feature flags overview
   - E. Using LaunchDarkly to manage feature flags
   - F. Telemetry and application performance management
   - G. Application Insights for gathering telemetry
   - H. A/B testing
   - I. Using feature flags to support A/B testing
   - J. Exploratory testing, testing “tours” practice
   - K. Using the Microsoft Test and Feedback extension
   - L. Understanding and identifying technical debt
   - M. Using SonarQube to gauge your technical debt
   - N. Practices for paying off technical debt

VII. *Learning and Improving Continuously*
   - A. Building a culture of learning and improvement
   - B. Agile metrics and reporting
   - C. Communities of Practice (COPs)
   - D. Lean thinking and practices to eliminate waste
   - E. Using the wiki to build tribal knowledge

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically