DevOps - Test Driven Development

Duration: 1 Day  
Course Price: $1,999

Course Description

Purpose:
In this course, students will learn and practice the application of Test Driven Development business logic in real world examples, environments, and instructor led demonstrations.

Audience:
* Software Developers  
* Project Managers  
* Quality Assurance Testers

Objectives:
By the end of this course, students will be able to:
* Provide knowledge and understanding of Unit Testing principles and practices  
* Understand the role of Unit Testing in software development and testing  
* Write effective Unit Tests  
* Properties of effective unit tests  
* How to use mock objects to isolate the “system under test”  
* Effective Refactoring of the code base  
* Benefits of test-first and Test-Driven Development  
* Techniques and Practices to aid in the successful adoption of Test-Driven Development  
* How to use Acceptance Testing and Behavior-Driven Development to further advance Test-Driven Development

Course Description:
This 3-day instructor led course will teach the thought process behind TDD, as well as a deep dive that incorporates the steps that are necessary for effective implementation. Unit Tests, User Stories, Design, Refactoring, Frameworks, and how to apply them to existing solutions. We will also explore the implications of code dependencies, fluid requirements, and early detection of issues. This is an interactive class, and individuals will be expected to participate in all exercises to attain maximum benefit of the course.

Course Outline

Module 1. Unit Testing

Unit Testing is a critical component of Test Driven Development (TDD). Small units of code are tested throughout the development process, which isolates functionality to ensure that individual parts work correctly.

Unit Test Fundamentals
* Reason to do Unit Testing  
* What to Test: Right BICEP  
* CORRECT Boundary Conditions  
* Properties of Good Tests

Advanced Unit Testing
* Unit Testing Legacy Applications  
* Techniques for Legacy Code  
* External Dependencies  
* Unit Testing the Database

Example

Description of lab, exercise, or demonstration(s) goes here

Module 2: Test-Driven Development Approach

TDD is directly influenced by design, so it will be a priority to take this into context during implementation. Considering design principles will enable teams to experiment with different approaches, and gear towards more functional programming.

TDD Assertions
* Test Runner  
* Assertions  
* “Fluent” Assertions
Module 3. TDD Essentials

Essential TDD techniques require developers write programs in short development cycles, and there are critical steps that must be taken. Tests are created before the code is written. Once the code passes testing, it is refactored to adhere to the most effective and acceptable standards.

TDD Design

* Design Principle
* Once and Only Once
* Design & Code Simply
* Open Closed Principle
* Principle of Least Knowledge

Test Automation

* Automate, Automate, Automate
* Automate Early & Often
* Additional Topics Identified

Module 4. Refactoring

Refactoring is another essential technique of TDD, and most software development teams are most likely doing some form of refactoring regularly. Refactoring can be used in a number of different workflows which will be explored in this module.

Why Refactor?

* Red, Green, Refactor
* Benefits
* Development without TDD

Refactoring Methods

* Test Last Approach
* Test Last in Parallel
* Test-Driven Design
* Sustainable TDD

Refactoring Cycle

* Reduce Local Variable Scope
* Replace Temp with Query
* Remove Dead Code
* Extract Method
* Remove Unnecessary Code

Exercise

Description of lab, exercise, or demonstration(s) goes here

Module 5. Pair Programming

Pair Programming is an effective way to improve code quality. In this module, we will discuss pairing and how it leads to better software design, and a lower cost of development.

Examples

Description of lab, exercise, or demonstration(s) goes here
Pair Programming

* Pairing Setup
* Results: Time
* Results: Defects
* How it works

Advantages of Pairing

* Both Halves of the Brain
* Focus
* Reduce Interruptions
* Reduce Resource Constraints
* Multiple Monitors
* Challenges

Pairing Techniques

* Pair Rotation
* Ping Pong Pairing
* Promiscuous Pairs
* Pair Stairs
* Cross-Functional Pairing

Module 6. Acceptance Testing & BDD

Acceptance Tests are an important form of functional specification, and Behavior Driven Development dictates what happens as an effect of these tests being run. In this Module, we will cover the difference between them, and how they are closely related.

Acceptance Testing

* Acceptance Tests
* Why Acceptance Tests?
* Acceptance Test Execution
* Who Writes Acceptance Tests
* Pair Test writing

Exercise

Acceptance Test Writing Exercise

Best Practices for Effective Testing

* Keys to Good Acceptance Tests
* Writing Acceptance Criteria
* Acceptance Test Example
* Acceptance Test-Driven Development (ATDD)

BDD vs. ATDD

* Specification by Example
* BDD Frameworks
* BDD Examples

Demonstration/Lab

TDD Queue Exercise
TDD Testing Lab

Module 7. Principles & Benefits

In this Module, we will take a look at the benefits of TDD, the principles that drive them. We will review the collaborative nature of TDD, and focus on the communicative process that makes it work.

Consequences of no Testing

* Ripple Effect of Defects
* Partially Done Work
* Extra Features
* Relearning
* Handoffs
* Task Switching
* Delays
TDD Solutions
* Continuous Unit Testing
* Collective Ownership
* Implementing a Queue
* Enqueue & Dequeue Methods
* Peek Method
* Create a Test List

Module 8. Unit Test Properties
In order to implement Unit Tests efficiently, it is critical that developers understand their properties. This Module will explore this concept.

Unit Test Properties
* More Tests?
  * The Sieve of Eratosthenes
  * Algorithm
  * Understanding the Sieve

Exercise
Description of lab, exercise, or demonstration(s) goes here

Module 9. Frameworks
Developers should use whatever testing framework is appropriate to their software environment. This module will discuss some of the many testing frameworks and tools that are useful in TDD.

Advanced TDD
* Replace Nested Conditional with Guard Clauses
* Inline Method
* Rename Variable
* Collapse Loops
* Remove Dead Code

TDD and FIT
* Fi
  * Finessse
  * FIT & Finessse
* Core FIT Fixtures
* Using a Column Fixture
* Finessse Wiki Syntax
* Create a Column Fixture

Additional TDD Considerations
* Using a RowFixture
* Using an ActionFixture
* SLIM
  * SLIM Tables
* The Gherkin Language
* Step Definitions

Exercise
Description of lab, exercise, or demonstration(s) goes here

Learn iT! San Francisco - 33 New Montgomery St. - Suite 300 - San Francisco, CA 94105 - 855.838.5028 or 415.693.0250
Learn iT! San Jose - 2025 Gateway Place - Suite 300 - San Jose, CA 95110 - 855.838.5028 or 408.200.0953
Learn iT! New York - 330 West 38th St. - Suite 301 - New York, NY 10018 - 855.838.5028
Live Online Training - Remote Training - Home, Office, Anywhere