Document Generated: 02/25/2026

Learning Style: Virtual Classroom

Technology:

Difficulty: Intermediate

Course Duration: 2 Days

Intermediate GitHub for Developers (TTDV7552)

About This Course:

This is a fast-paced, hands-on course designed for developers who already have foundational Git and GitHub experience and want to deepen their skills with advanced collaboration and workflow features. This class goes beyond the basics, focusing on practical techniques for branching, merging, conflict resolution,

workflow optimization, and leveraging GitHub Actions to automate development pipelines. Students will gain confidence in handling complex scenarios and learn best practices for maintaining clean, efficient project histories.

Course Objectives:

- Branching with Git
- Merging Pull Requests
- Viewing Local Project History
- Streaming Your Workflow with Aliases
- Workflow Review Project: GitHub Games
- Resolving Merge Conflicts
- Working with Multiple Conflicts
- Merge Strategies: Rebase
- Understanding GitHub Actions
- Working with Workflows
- Managing Jobs and Steps within a Workflow
- Dealing with complex Workflows

Audience:

- This class does assume prior experience with Git, The students should also have basic coding or programming knowledge.

Prerequisites:

- This class does assume prior experience with Git, The students should also have basic coding or programming knowledge.
- If you are new to Git and GitHub, we recommend starting with our companion course TTDV7551 – Introduction to GitHub for Developers, which provides the essential foundations before progressing to this intermediate-level training.

Course Outline:

1 - Branching with Git

- Branching Defined
- Activity: Creating a Branch with GitHub
- Introduction
- Creating and Switching Branches
- Tracking and Merging Branches
- Deleting and Restoring Branches
- Class Diagram
- Interaction Diagrams
- Sequence Diagrams
- Communication Diagrams
- State Machine Diagrams
- Activity Diagram
- Implementation Diagrams

2-Merging Pull Requests-

- Merge Explained
- Merging Your Pull Request
- Updating Your Local Repository
- Cleaning Up the Unneeded Branches
- Reviewing and Commenting

3-Viewing Local Project History

- Using Git Log
- Using git show, and git diff

- Visualizing History with Git Graphs
- Filtering Commits by Author, Date, or Message

4-Streaming Your Workflow with Aliases

- Creating Custom Aliases
- Using Shell Scripts with Git Commands

5-Workflow Review Project: GitHub Games

- User Accounts vs. Organization Accounts
- Introduction to GitHub Pages
- What is a Fork?
- Creating a Fork
- Workflow Review: Updating the README.md

6 - Resolving Merge Conflicts

- Local Merge Conflicts
- Understanding Conflict Scenarios
- Manual Conflict Resolution
- Using Visual Merge Tools
- Best Practices to Avoid Conflicts

7-Working with Multiple Conflicts

- Remote Merge Conflicts
- Exploring
- Handling Complex Merge Situations
- Conflict Markers and Resolution Strategy
- Using git mergetool and git rerere

8-Getting it Back

- You just want that one commit
- Oops, I didn't mean to reset

9- Merge Strategies: Rebase

- About Git rebase
- Understanding Git Merge Strategies
- Creating a Linear History
- Understanding Rebase vs Merge
- Interactive Rebase for Clean History
- Avoiding Rebase Pitfalls in Collaboration

10- GitHub Actions

- What Are GitHub Actions?
- Workflows
- Events
- Jobs
- Steps
- Runners
- Marketplace and Prebuilt Actions

11- Complex Workflows

- Adding scripts to your workflow
- Using variables
- Sharing data between jobs
- Working with dependencies
- Working with Services

- Working with Workflows
- Managing Jobs and Steps within a Workflow
- Dealing with Complex Workflows
- Conditional Execution
- Reusable Workflows and Composite Actions