QUICKSTART

Over 35 Years Of Technology Training

Document Generated: 12/16/2025

Learning Style: Virtual Classroom

Technology:

Difficulty: Beginner

Course Duration: 2 Days

Introduction to GitLab (TTDV7553)

About This Course:

Version control is a foundational skill in modern software development, helping teams collaborate effectively, track changes, and deliver better software faster. GitLab, a powerful web-based DevOps platform built around Git, offers both developers and technical teams the tools to manage code, streamline daily

workflows, and automate delivery. This expert-led, two-day class is designed to teach you how to work confidently with Git and GitLab, whether you are just getting started or want to strengthen your foundational skills.

This course is ideal for developers, engineers, QA professionals, technical project leads, or anyone involved in software development who needs to understand and contribute to version-controlled projects. You will learn how to manage branches, handle merge conflicts, rebase cleanly, and configure Git to suit your environment. You will explore techniques to review and clean up commit history, simplify your workflow using aliases, stash changes, and work with remote repositories effectively. About half of the course is hands-on, giving you time to apply what you are learning to real-world scenarios with support from your instructor.

You will also gain experience using GitLab built-in tools for collaboration and automation. From using GitLab Flow to managing merge requests and working with CI/CD pipelines, you will learn practical techniques to improve quality, reduce errors, and support consistent delivery. By the end of the class, you will be ready to contribute to team projects with confidence, troubleshoot common issues, and apply GitLab to organize, automate, and enhance your development process.

Course Objectives:

- Create and manage Git repositories. Set up projects and track changes to your code using Git and GitLab.

- Work with branches and resolve merge conflicts. Use branching strategies to organize work and handle changes from multiple team members.

- Clean up and review commit history. Understand how to review, edit, and simplify commit logs for better collaboration.

- Configure Git for your daily workflow. Set up your Git environment, create shortcuts, and manage ignored files.

- Collaborate using GitLab Flow. Apply a real-world GitLab workflow, including merge requests, approvals, and issue tracking.

- Build and troubleshoot GitLab CI/CD pipelines. Create simple automation to test and deliver your code more reliably.

Audience:

- The introductory-level course is designed for anyone who works with software projects and wants to better understand version control and team collaboration using Git and GitLab. It is a great choice for developers, QA engineers, DevOps professionals, and technical team leads looking to strengthen their workflow and project organization. Whether you are brand new to Git or have some experience and want to build better habits, this course will help you work more smoothly with your team.

Prerequisites:

- Basic Command Line Use: Know how to open a terminal and run simple commands.

  Familiarity with Software Projects: Understand how code is organized and versioned in a team setting.

  Using Web-Based Tools: Be comfortable navigating online platforms and user interfaces.

Course Outline:

1. Git and GitLab Introduction and Basics

- Understanding version control and Git essentials
- Exploring GitLab as a collaboration platform
- Installing and configuring Git on your system
- Working with repositories and tracking changes
- Making your first commit and navigating GitLab UI

2. GitLab Flow

- Overview of GitFlow vs. GitLab Flow
- Selecting the right workflow for your team
- Managing environments: staging, production, and feature branches
- Using issues and merge requests in GitLab
- Implementing protected branches and approvals

3. Branching

- Creating and switching branches
- Naming conventions and organizational tips
- Visualizing branch structures
- Tagging versions and releases
- Cleaning up and deleting merged branches

4. Configuring Git

- Setting up global and local configuration
- Understanding .gitconfig settings
- Creating aliases to simplify commands
- Managing ignored files with .gitignore
- Using Git credentials and SSH keys securely

5. Rebasing

- Introduction to rebasing vs. merging
- Rebasing local branches
- Handling conflicts during rebases
- Interactive rebasing and squashing commits
- Best practices and troubleshooting tips

6. Merging

- Merging strategies and fast-forward vs. no-fast-forward
- Performing merges in GitLab
- Understanding merge commits
- Handling merge requests and code reviews
- Tools for visualizing merge conflicts

7. Resolving Merge Conflicts

- Identifying the cause of merge conflicts
- Using Git commands to resolve conflicts
- Tools and editors to assist with conflict resolution
- Best practices for preventing common conflicts
- Committing and testing after resolving conflicts

8. Remote Repositories

- Connecting to remote repositories
- Cloning and pushing projects
- Pulling and fetching updates
- Setting up upstream and tracking branches
- Collaborating with remotes and forks

9. Reviewing the Commit History

- Exploring git log, git show, and git diff
- Using shortcuts to navigate revisions
- Blame and annotate features for tracking changes
- Reverting and amending commits
- Cleaning up history with rebase and squash

10. Improving Your Daily Workflow

- Saving work-in-progress with git stash
- Interactive staging and reviewing diffs
- Automating common tasks with aliases
- Managing build artifacts in the workflow
- Creating readable and meaningful commit messages

11. Continuous Integration / Continuous Delivery (CI/CD)

- Introduction to GitLab CI/CD and pipelines
- Installing and configuring GitLab Runner
- Understanding .gitlab-ci.yml syntax and structure
- Creating and customizing your first CI/CD pipeline
- Visualizing, debugging, and optimizing pipelines