Introduction to GitLab

Duration: *2 Day(s)*

Course Overview

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.

Review this course online at https://www.alta3.com/courses/TTDV7553

Objectives

- Create and manage Git repositories efficiently.
- Work with branches and resolve merge conflicts seamlessly.
- Configure Git for personalized daily workflows.
- Collaborate effectively using GitLab's tools and workflows.

Who Should Attend

- Developers
- QA Engineers
- DevOps Professionals
- Technical Team Leads

Prerequisites

To get the most out of this GitLab Quick Start course, it helps to have a few basic skills. No prior Git or GitLab experience is needed, but some general tech comfort will make things easier:

```
 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

Git and GitLab Introduction and Basics

1. Understanding version control and Git essentials
2. Exploring GitLab as a collaboration platform
3. Installing and configuring Git on your system
4. Working with repositories and tracking changes
5. Making your first commit and navigating GitLab UI

GitLab Flow

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

Branching

11. Creating and switching branches
12. Naming conventions and organizational tips
13. Visualizing branch structures
14. Tagging versions and releases
15. Cleaning up and deleting merged branches

Configuring Git

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

Rebasing

21. Introduction to rebasing vs. merging
22. Rebasing local branches
23. Handling conflicts during rebases
24. Interactive rebasing and squashing commits
25. Best practices and troubleshooting tips

Merging

26. Merging strategies and fast-forward vs. no-fast-forward

27. Performing merges in GitLab
28. Understanding merge commits
29. Handling merge requests and code reviews
30. Tools for visualizing merge conflicts

Resolving Merge Conflicts

31. Identifying the cause of merge conflicts
32. Using Git commands to resolve conflicts
33. Tools and editors to assist with conflict resolution
34. Best practices for preventing common conflicts
35. Committing and testing after resolving conflicts

Remote Repositories

36. Connecting to remote repositories
37. Cloning and pushing projects
38. Pulling and fetching updates
39. Setting up upstream and tracking branches
40. Collaborating with remotes and forks

Reviewing the Commit History

41. Exploring git log, git show, and git diff
42. Using shortcuts to navigate revisions
43. Blame and annotate features for tracking changes
44. Reverting and amending commits
45. Cleaning up history with rebase and squash

Improving Your Daily Workflow

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

Continuous Integration / Continuous Delivery (CI/CD)

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