Course Outline

Comprehensive Git Using GitHub

Course Summary

Description

This comprehensive course on Git and GitHub introduces students to the fundamentals of Version Control, Git, and GitHub. Over the course of three days, students will learn how to use Git and GitHub for version control, collaboration, and continuous integration. The course is designed to provide practical, hands-on training that will enable students to use Git and GitHub in real-world scenarios.

Objectives

At the end of this course, students will be able to:

- Explain the benefits of using Version Control in software development and the importance of Git and GitHub.
- Set up Git and GitHub accounts and repositories, and create and manage branches.
- Use Git commands to add, commit, push, pull, and merge changes in a repository.
- Work with remotes, and fetch, update, and push changes to remote repositories.
- Collaborate with others using Git and GitHub, including managing issues, pull requests, and code reviews.
- Understand the anatomy of a commit message and write effective commit messages.
- Use Git tags, log, diff, and patch to manage changes across different environments.
- Apply best practices for using Git, including repository structure, hooks, and aliases.

Topics

- Introduction to Version Control
- Working with Remotes
- Advanced Git Techniques

Audience

This course is designed for developers who need to learn how to work with Git.

Prerequisites

- Familiar with the software development process is assumed.
- Comfortable working on windows is required
- Launching applications and working with the file system
- Experienced working from the command line is an asset

Duration

Three days

Course Outline

Comprehensive Git Using GitHub

Course Outline

I. Introduction to Version Control

A. What is Version Control?
   1. Definition of Version Control
   2. Importance of Version Control in Software Development
   3. Types of Version Control Systems

B. Introduction to Git
   1. Definition of Git
   2. Benefits of using Git for Version Control
   3. Git vs Other Version Control Systems
   4. Setting up Git

C. The Basic Git Workflow
   1. Creating a Git Repository
   2. Adding and Committing Changes
   3. Branching and Merging
   4. Understanding the Anatomy of a Commit Message

D. Using Git Locally
   1. Skipping the Staging Area
   2. Getting More Information About Our Changes
   3. Deleting and Renaming Files
   4. Undoing Changes Before Committing
   5. Amending Commits
   6. Rollbacks
   7. Identifying a Commit

II. Working with Remotes

A. Introduction to GitHub
   1. Definition of GitHub
   2. Benefits of using GitHub
   3. Setting up a GitHub account

B. Working with Remotes
   1. What is a Remote?
   2. Setting up a Remote Repository
   3. Fetching New Changes
   4. Updating the Local Repository
   5. Pushing Remote Branches
   6. Rebasing Your Changes
   7. Best Practices for Collaboration

C. Collaboration
   1. What are Code Reviews?
   2. The Code Review Workflow
   3. Using Code Reviews in GitHub
   4. Managing Collaboration
   5. Tracking Issues
   6. Continuous Integration

III. Advanced Git Techniques

A. Keeping Historical Copies
   1. Understanding the Importance of Historical Copies
   2. Using Git Tags
   3. Git Log

B. Diffing Files
   1. Comparing Changes with Git Diff
   2. Visualizing Changes with Git Diff-Tree

C. Applying Changes
   1. Introduction to Git Patch
   2. Applying Git Patches

D. Practical Application of diff and patch

E. Using Diff and Patch to Manage Changes Across Different Environments

F. Advanced Patch Techniques

G. Best Practices for Using Git
   1. Creating a Git Repository Structure
   2. Writing Good Commit Messages
   3. Using Git Hooks
   4. Using Git Aliases