Missing Semester - Collaborating With Git (Week 4)

Introduction to Collaborating with Git

Welcome to today's lesson on collaborating with Git! Git is a powerful version control system that allows multiple developers to work together on a project efficiently. Whether you are working on a small team or contributing to a large open-source project, understanding how to collaborate effectively with Git is essential.

In this lesson, we will explore various Git features and workflows that enable seamless collaboration. We will cover topics such as branching, merging, resolving conflicts, and using remote repositories like GitHub or GitLab. By the end of this lesson, you will have a solid understanding of how to work collaboratively with Git, ensuring smooth coordination and efficient development.

So, let's dive in and discover the world of collaborating with Git!# Collaborating with Git

When working on a project with multiple developers, it is essential to have a collaborative workflow to ensure smooth coordination and avoid conflicts. Git provides several features and techniques to facilitate collaboration among team members. In this lesson, we will explore some of the key concepts and practices for collaborating with Git.

Branching

Branching is a fundamental concept in Git that allows developers to work on different features or bug fixes simultaneously without interfering with each other's work. Each branch represents an independent line of development, enabling developers to isolate their changes and merge them back into the main codebase when ready.

To create a new branch, use the `git branch` command followed by the desired branch name. For example, `git branch feature-xyz` creates a new branch named "feature-xyz." To switch to the newly created branch, use `git checkout feature-xyz`.

Once you have made changes on a branch, you can commit them using the usual `git commit` command. It is good practice to regularly push your branch to the remote repository using `git push origin branch-name`. This ensures that your work is backed up and accessible to other team members.

Pull Requests

Pull requests are a powerful collaboration feature in Git, commonly used in open-source projects and team-based development. A pull request allows developers to propose changes to the main codebase and request a review from other team members before merging the changes.

To create a pull request, first, ensure that your branch is up to date with the latest changes from the main codebase. Use `git pull origin main` to fetch and merge the latest changes from the "main" branch. Then, push your branch to the remote repository using `git push origin branch-name`.

Next, navigate to the repository's web interface (e.g., GitHub, GitLab) and locate your branch. Click on the "New Pull Request" button and provide a title and description for your changes. Select the appropriate branches for comparison (e.g., your branch and the main branch) and submit the pull request.

Other team members can review your changes, leave comments, and suggest modifications directly on the pull request. Once the changes are approved, they can be merged into the main codebase.

Resolving Conflicts

Conflicts may arise when merging branches that have diverged and contain conflicting changes. Git provides tools to help resolve these conflicts and ensure a smooth merge process.

When a conflict occurs, Git marks the conflicting sections in the affected files. Open the conflicting file(s) in a text editor and locate the conflict markers (`<<<<<<<`, `=======`, `>>>>>>>`). Edit the file to resolve the conflicts manually, keeping the desired changes and removing the conflict markers.

After resolving the conflicts, save the file(s) and stage them using `git add`. Then, commit the changes using `git commit`. Git will automatically create a merge commit, incorporating the resolved conflicts.

Conclusion

Collaborating with Git is crucial for efficient teamwork and codebase management. By leveraging branching, pull requests, and conflict resolution techniques, developers can work together seamlessly, review each other's changes, and maintain a clean and stable codebase.

Resources and Exercises for Further Studies.

1. [BLOG] A no-jargon blog on Git - https://uoftcoders.github.io/studyGroup/lessons/git/collaboration/lesson/
2. [BLOG] Read this article about Collaboration using Github - https://pages.nist.gov/git-novice-MSE/08-collab/