College of Engineering
Department of Software Engineering
Addis Ababa Science and Technology University
Course: Software Component Design
Title: **GitHub Basics Documentation**

Group 5

Name	Student ID	Section
Daniel Mekonnen	ETS 0351/13	B
Doi Amdisa	ETS 0385/13	B
Fasika G/Hana	ETS 0493/13	B
Haweten Girma	ETS 0595/13	B
Hawi Abdi	ETS 0596/13	B

Submitted date : December 19, 2024

GitHub Basics Documentation

Understanding GitHub Basics

1. What is GitHub, and how is it used in software development?

GitHub is a web-based platform for version control and collaboration. It allows developers to store, track, and manage code changes efficiently. GitHub is used in software development to:

- Host and manage code repositories.
- Collaborate with team members through features like pull requests and issues.
- Track changes using version control.
- Integrate with other tools for testing, deployment, and project management.

2. GitHub's Purpose and Key Features

GitHub aims to simplify collaboration and code management. Key features include:

- **Repositories**: Central storage for code and related files.
- **Branches**: Parallel versions of a codebase for experimentation.
- **Pull Requests**: Mechanism for proposing and discussing code changes.
- **Issues**: Tool for tracking tasks, bugs, and feature requests.
- **Actions**: Workflow automation tools.
- **Wikis**: Documentation hosting for projects.

3. Git vs. GitHub: Why Use Both?

- **Git**: A distributed version control system to track changes in code locally and remotely.
- **GitHub**: A platform to host Git repositories and enable collaboration. Software engineers use Git to manage local changes and GitHub to share and collaborate on those changes with a team.

4. What is a Repository in GitHub, and How to Create One?

A repository is a storage location for your project's files and version history.

Steps to Create a Repository:

1. Log in to GitHub.
2. Click on the "New" button in the Repositories section.
3. Provide a repository name and description.
4. Choose public or private visibility.
5. Initialize with a README (optional) and click "Create Repository."

5. Purpose of Branches in GitHub

Branches allow developers to work on features or fixes independently of the main codebase. They support collaborative development by enabling:

- Experimentation without affecting the main branch.
- Parallel development by multiple contributors.

6. What Are Commits and Why Are They Important?

Commits are snapshots of changes made to the code. They:

- Record the history of changes.
- Allow you to revert to previous versions if needed.
- Enable collaboration by showing contributors' updates.

Practical GitHub Usage

1. Cloning a Repository to Your Local Machine

Command:
`git clone <repository_url>`

This command copies the repository's content to your local machine.

2. Staging Changes and Committing Locally

Commands:

1. Stage changes:
   `git add <file_name>`
   Stages specific file(s) for the next commit.
2. Commit changes:
   `git commit -m "Your commit message"`
   Saves the staged changes locally with a message describing the update.

3. Creating a Pull Request

Steps:

1. Push your changes to a remote branch.
   `git push origin <branch_name>`
2. Navigate to the repository on GitHub.
3. Click "Pull Requests" and then "New Pull Request."

4. Compare your branch with the main branch and submit the pull request. A pull request is necessary when merging changes from one branch to another.

4. Resolving Merge Conflicts

Merge conflicts occur when two branches have conflicting changes. To resolve:

1. GitHub highlights conflicting files.
2. Edit the file to resolve the conflicts.
3. Commit the resolved file.
4. Continue the merge process.

5. Tracking Change History

Command:
`git log`

Displays the commit history, including messages, authors, and timestamps.

Collaboration and Teamwork

1. Adding Collaborators to a Repository

Steps:

1. Go to the repository settings.
2. Click “Collaborators.”
3. Enter the collaborator’s username or email and set permissions (e.g., Read, Write).

2. Purpose of Issues in GitHub

Issues help in tracking tasks, bugs, and feature requests, improving project management.

Exercise:

1. Create an issue describing a task.
2. Assign it to a teammate.
3. Track its resolution using comments and status updates.

3. Importance of Pull Request Reviews

Pull request reviews ensure code quality by allowing team members to:

- Catch bugs or issues early.

- Maintain consistent coding standards.
- Provide constructive feedback.

4. Strategies for Smooth Collaboration

- Use clear naming conventions for branches and commits.
- Write descriptive commit messages.
- Regularly update branches to avoid conflicts.

Advanced GitHub Features

1. Purpose of GitHub Actions

GitHub Actions automate workflows like testing, building, and deploying code. Example use case: Automatically run tests when a pull request is created.

2. GitHub Wiki

A GitHub Wiki documents your project and provides helpful resources for collaborators and users.

3. Protecting the Main Branch

Steps:

1. Go to repository settings.
2. Navigate to "Branches" and select "Add Branch Protection Rule."
3. Define rules like requiring pull request reviews.

Benefits:

- Prevents accidental changes to the main branch.
- Enforces code quality standards.

4. Using GitHub Pages

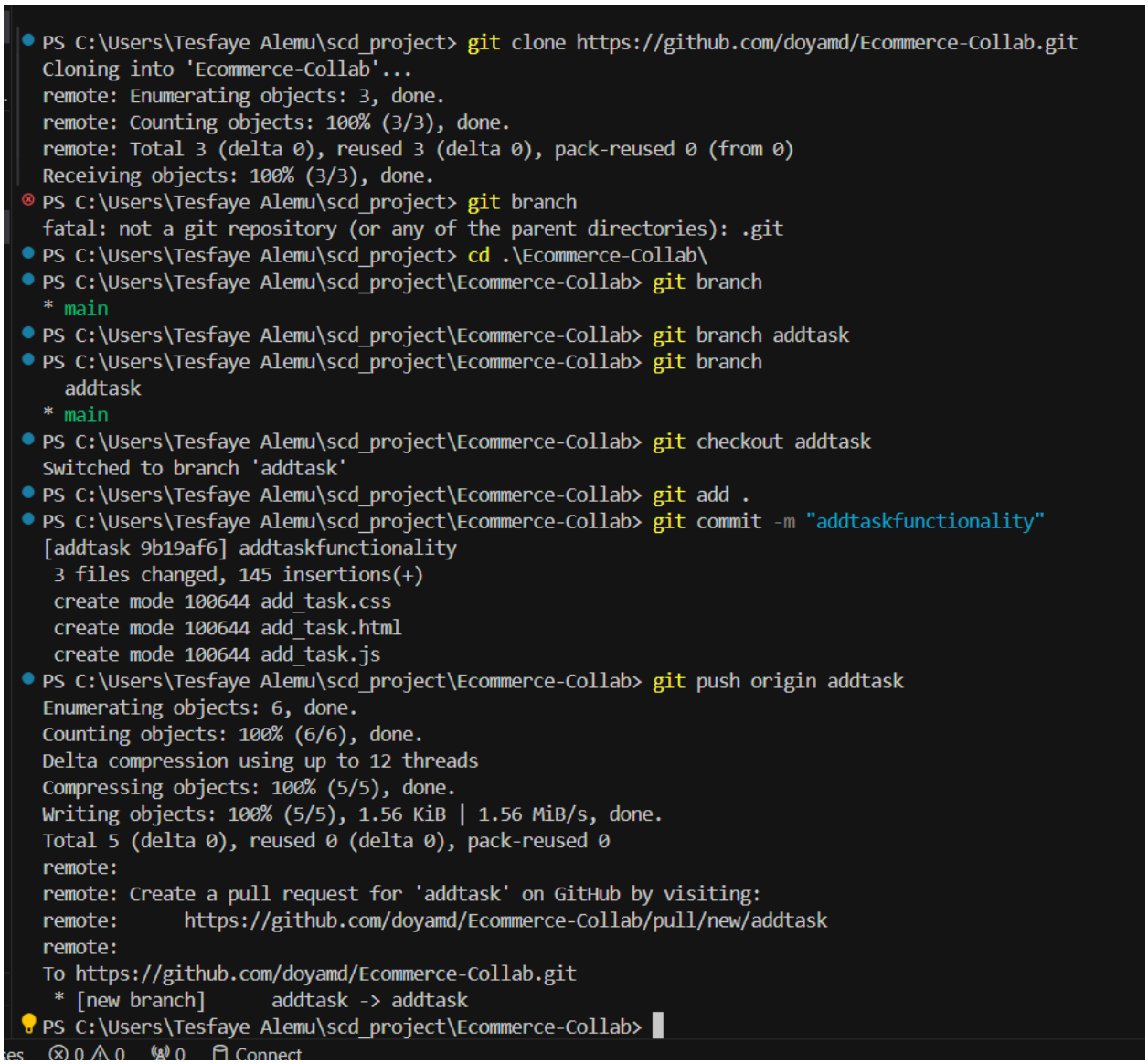
Steps to Deploy a Web Project:

1. Go to repository settings.
2. Enable GitHub Pages under "Pages."
3. Select a source branch and folder.
4. Access your project using the provided URL.

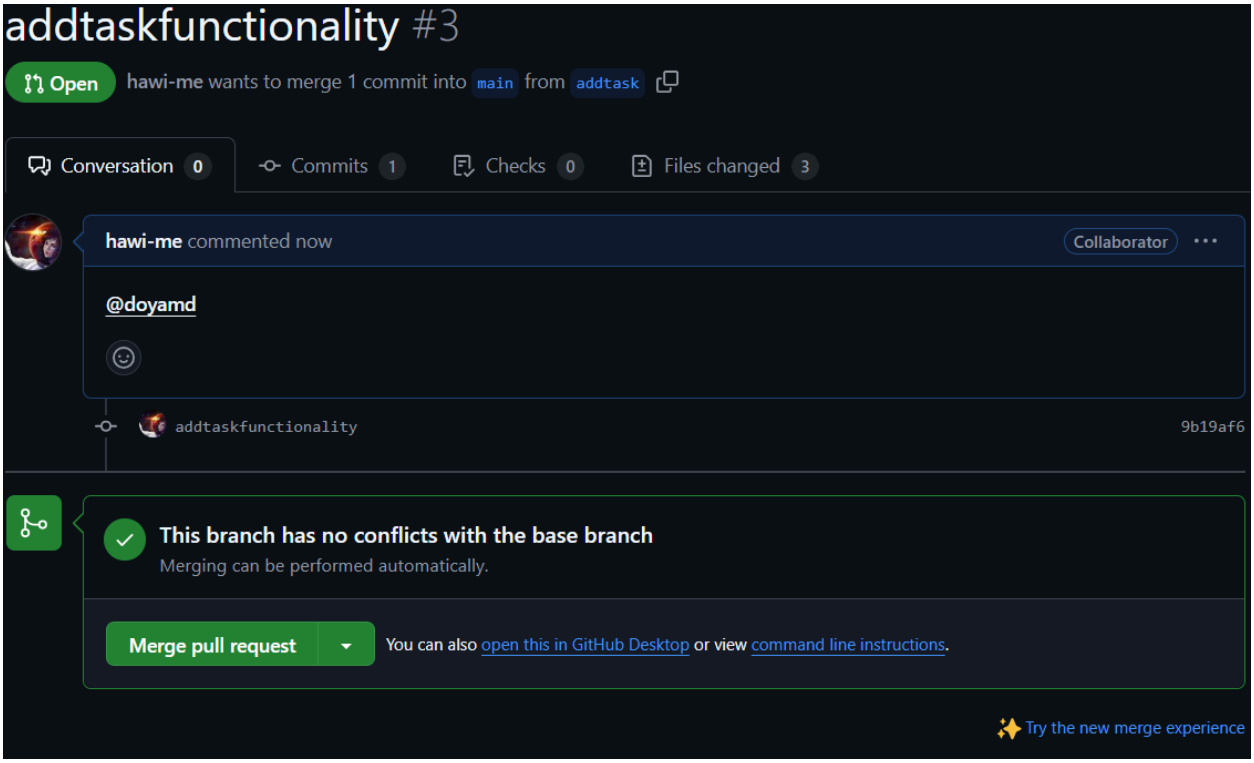
5. Integrating GitHub with External Tools

GitHub integrates with tools like:

- **CI/CD Pipelines**: Automate testing and deployment.
- **Project Management Tools**: Link issues with Jira tasks for streamlined management.
- Example of:
  git clone ,add, commit , push and pull request



Pull Requests



Issues