Git and GitHub workflows

In this train, weâ€™ll bring together the knowledge gained from using Git and GitHub separately, illustrating how they complement each other in a collaborative development environment.

Learning objectives

By the end of this train, you should be able to:

1. Set up a Git repository for a collaborative project.
2. Clone a repository, make changes, and push them back to GitHub.
3. Use pull requests for code review and collaboration in GitHub.

Outline

Introduction to Git and GitHub

Git and GitHub, though distinct, work in tandem to streamline software development, particularly in team environments. Understanding their interplay is crucial for managing and collaborating on projects.

Git: Your local version control system

- **Functionality:** Git is a distributed version control system. It tracks and manages changes to files in a project, allowing multiple versions (branches) of the project to be worked on simultaneously.
- **Local repository:** Each contributor has a local copy (or clone) of the remote repository on their computer. This enables working offline and making incremental changes (commits).
- **Branching and merging:** Git's branching feature allows developers to work on different features or parts of the project without affecting the main codebase. Once the work on a branch is complete, it can be merged back into the main branch (e.g., `main` or `master`).

GitHub: Collaborative hub in the cloud

- **Remote repository:** GitHub acts as a remote repository where the Git-tracked project is stored online. This makes it accessible to all team members, irrespective of their location.
- **Pull requests and code review:** One of GitHub's key features is the ability to create pull requests. These are requests to merge changes from one branch into another, often used for code review and discussion before integration.
- **Issue tracking and project management:** GitHub provides tools for tracking issues (bugs, enhancements, tasks) and managing projects through features like issues, project boards, and milestones.

Workflow Integration

1. **Initial setup:** A user creates a repository on GitHub. This repository is then cloned to the userâ€™s local machine using Git.
2. **Development cycle:**
   - **Local changes:** Developers make changes locally in their branch, commit these changes using Git, and periodically sync their work with the GitHub repository.
   - **Pull requests:** When a feature or fix is ready, the developer pushes their branch to GitHub and opens a pull request.
   - **Code review:** Team members review the pull request, discuss changes, and suggest improvements.
   - **Merge:** Once approved, the changes are merged into the main branch on GitHub.
3. **Ongoing collaboration:** The cycle of pull, modify, commit, push, and new pull request continues as the project evolves.

Creating a repository for sustainable forest management

Our first task is to set up a Git repository. This repository will host code and documents related to sustainable forest management, such as scripts for analysing environmental data or tracking deforestation.

Steps:

1. Log in to GitHub.
2. Click on the "New repository" button.
3. Name your repository (e.g., "sustainable-forest-management").
4. Choose "Public" and initialise with a README file.

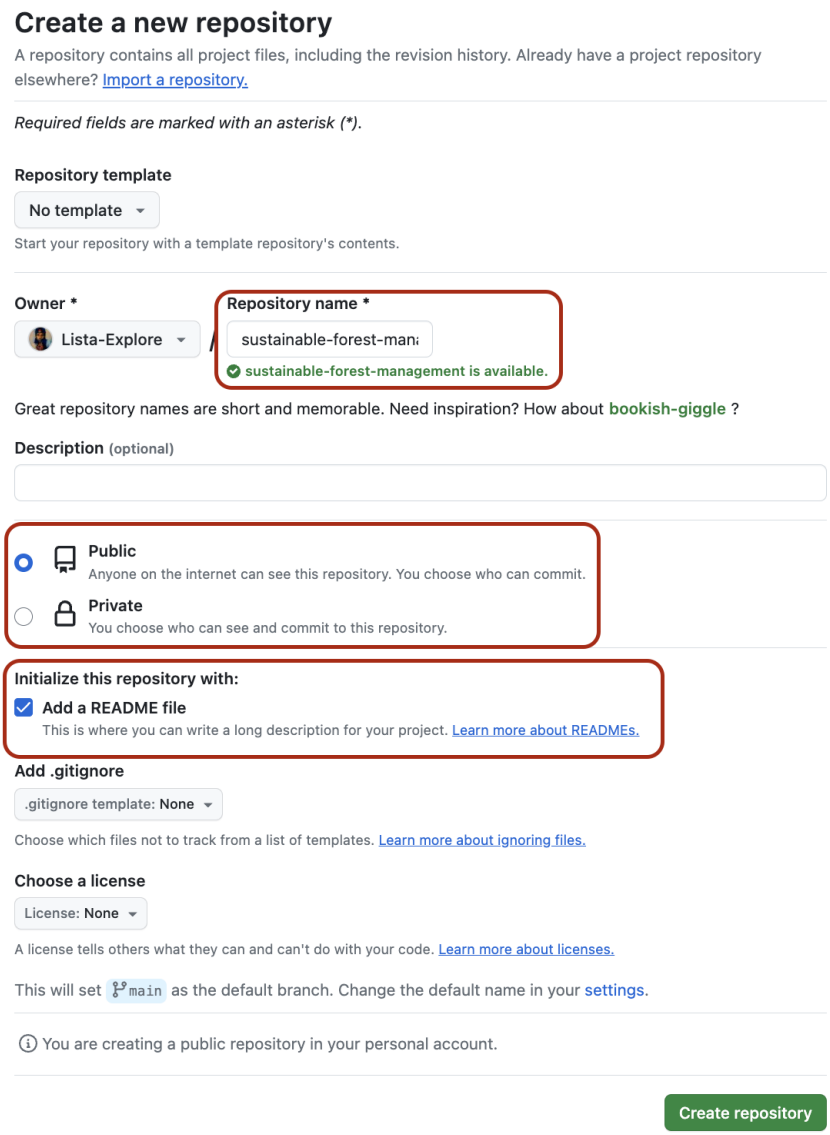

Figure 1: Creating a repository for sustainable forest management

Cloning and modifying a repository

After setting up your repository, the next step is to clone it to your local machine and make some changes.

Steps:

1. Copy the repository's URL from GitHub.

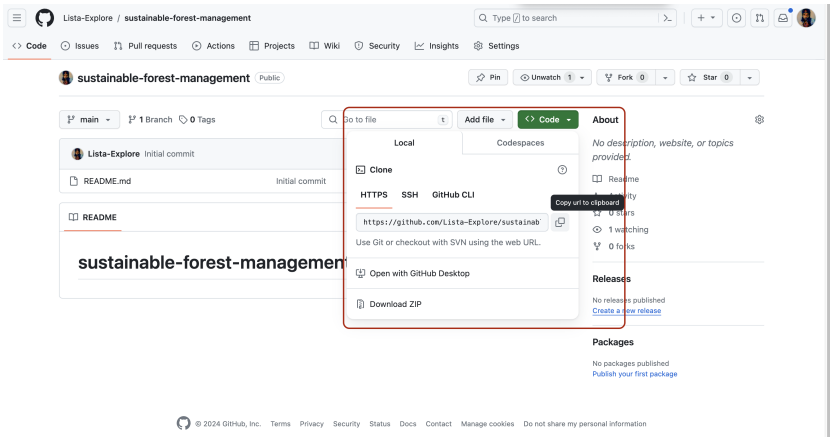

Figure 2: Copy the repository's URL from GitHub

2. Open your terminal or Git Bash and type:`git clone [repository URL]`. Replace `[repository URL]` with the actual URL.

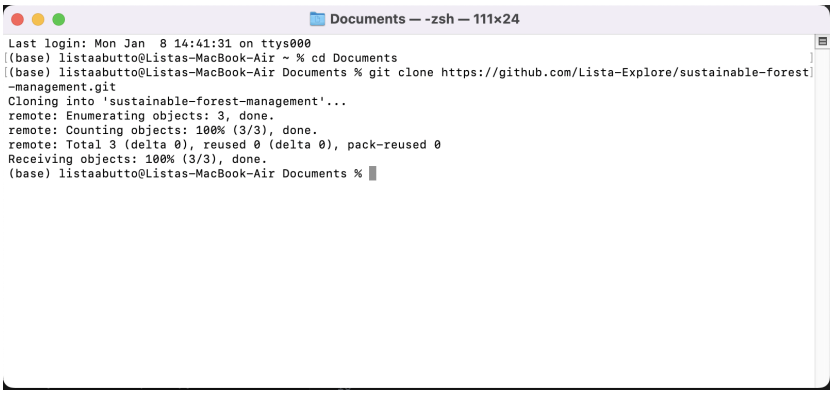

Figure 3: Clone the repository

3. Add a new file or make changes to an existing one. For example, create a simple text file with ideas on how technology can aid forest management.

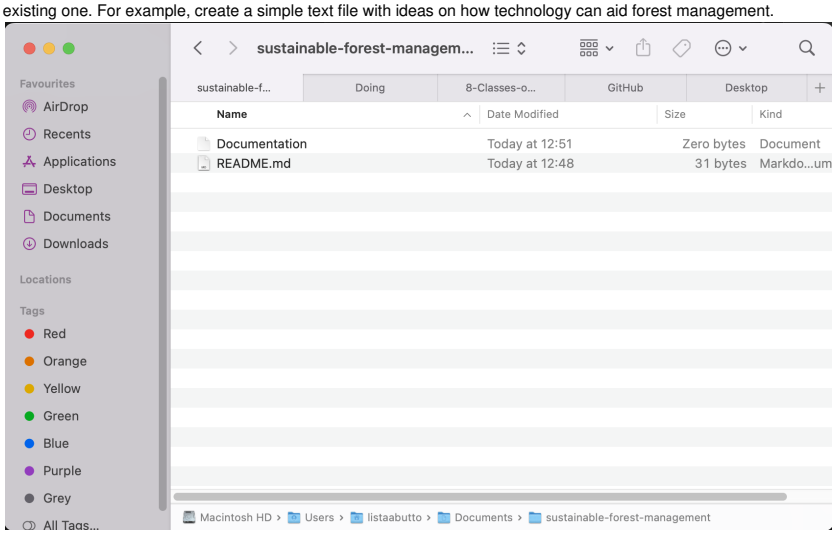

Figure 4: Add a file

Using push and pull requests

Once you've made changes, you'll need to push them back to GitHub and learn how to use pull requests for collaboration.

Steps:

1. In your terminal, navigate to your repository and type:
   - git add .
   - git commit -m "Your commit message"
   - git push origin main

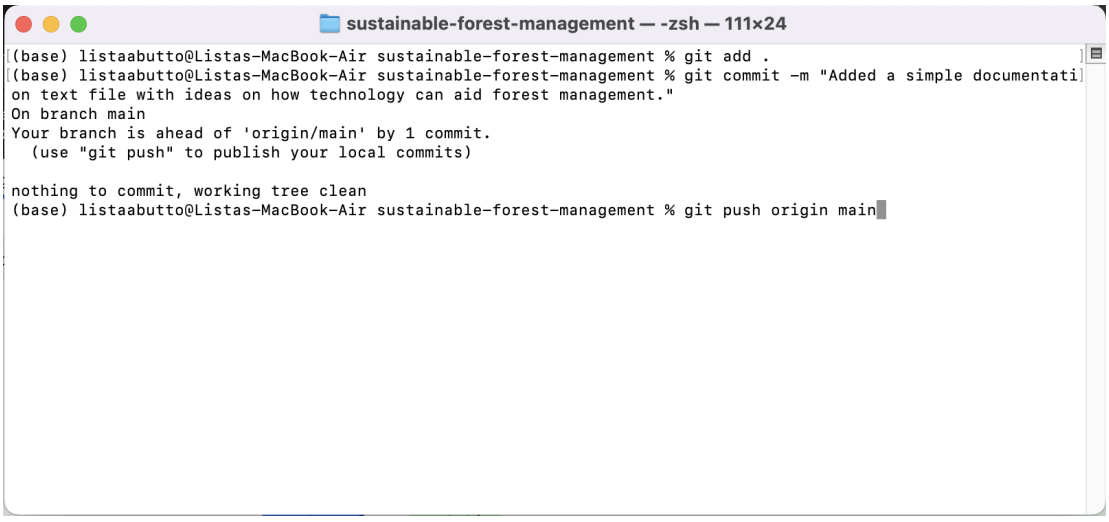

Figure 5: Push changes

2. Go to your GitHub repository, click on "Pull requests" > "New pull request" to review your changes and create a pull request.

Best practices in collaborative workflows

Effective collaboration in software development, especially in projects like sustainable forest management, requires adherence to certain best practices. These practices not only streamline the development process, but also ensure clarity and efficiency in teamwork.

Regular commits

Regular commits help in keeping track of changes made to the project. This practice is crucial in projects with multiple contributors, as it allows for a clear history of who did what and when.

> **Tip:** Commit small, incremental changes rather than large, infrequent updates. This makes it easier to identify and fix issues.

Clear commit messages

Clear and descriptive commit messages are essential. They provide context to the changes made, making it easier for teammates to understand each commit's purpose.

Example: Instead of "Fixed bug", use "Fixed data parsing error in deforestation tracking module".

Branching strategies

Using branches effectively is key in collaborative projects. Create separate branches for different features or sections of your project. This allows multiple team members to work on different aspects of the project simultaneously without interfering with one another's work.

Exercise: Create a branch named 'carbon-calculation' and make changes related to calculating carbon sequestration in forests.

Handling merge conflicts

Merge conflicts occur when different team members make changes to the same part of the code. Learning how to resolve these conflicts is crucial.

Tip: Regularly pull changes from the main branch to your working branch to minimise conflicts.

Code reviews and pull requests

Use pull requests for code reviews. This practice ensures that changes are reviewed by team members before being merged into the main project. It helps maintain code quality and consistency.

Task: Submit a pull request for your changes and have a team member review it and provide feedback.

Documentation

Good documentation is invaluable, especially for projects like sustainable forest management, where the project's purpose and impact are crucial. Ensure that your code is well-documented and easy to understand.

Exercise: Update the README file with detailed information about your project's contribution to sustainable forest management.

Understanding and implementing these best practices in Git and GitHub workflows will significantly enhance a team's efficiency and effectiveness, especially in complex and impactful projects.