

# Lab 2

## Git and GitHub in VS code

---

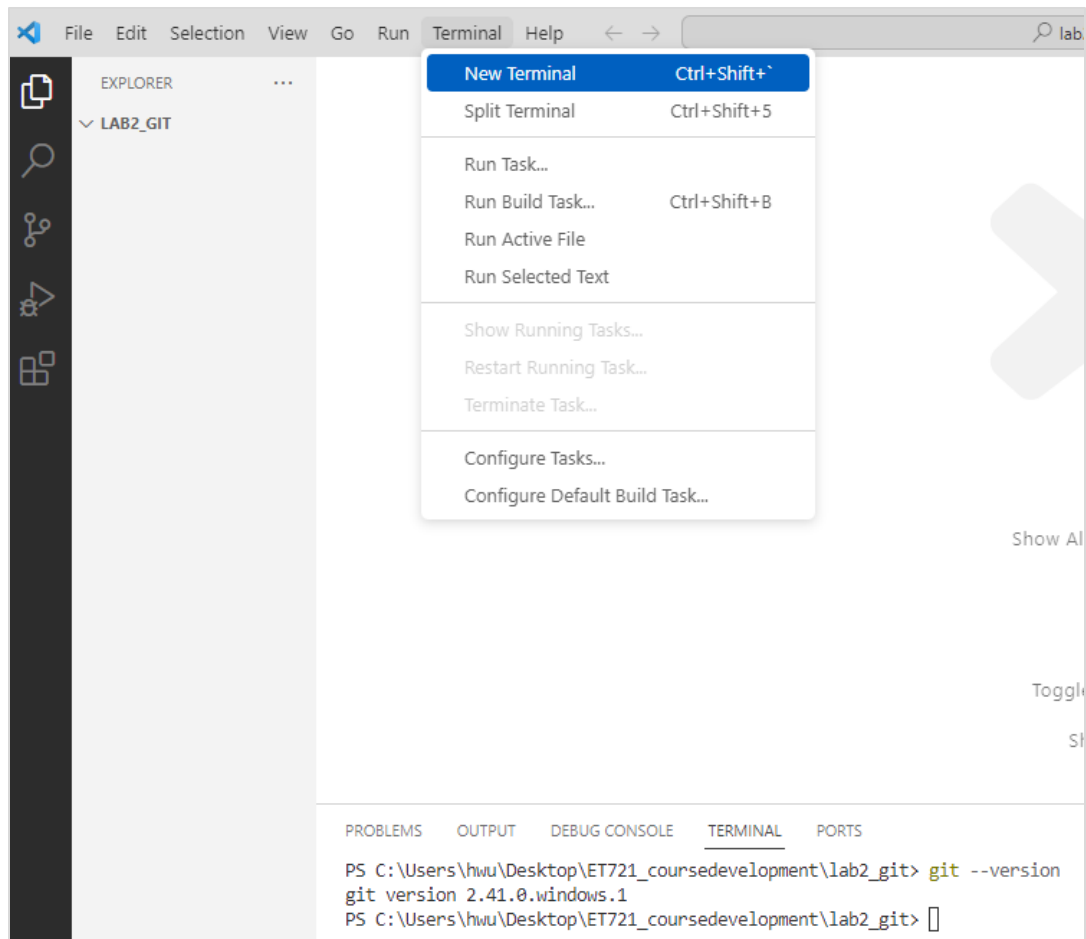
### Learning Outcome:

By the end of this lab, students will be able to effectively use Git to manage and collaborate on software projects. Specifically, they will be able to:

1. **Commit Changes:** Understand and execute the process of staging and committing changes to a local Git repository, including writing meaningful commit messages that accurately reflect the nature of the changes.
2. **Create and Manage Branches:** Demonstrate the ability to create, switch between, and delete branches to isolate and manage different lines of development within a project.
3. **Push Changes:** Effectively push local commits to a remote repository, ensuring that changes are synchronized and available to collaborators.
4. **Merge Branches:** Execute branch merging operations to integrate changes from one branch into another, including resolving conflicts that may arise during the merge process.
5. **Create and Manage Pull Requests:** Create, review, and manage pull requests in a remote repository hosting service (e.g., GitHub) to propose changes, conduct code reviews, and facilitate collaborative development.

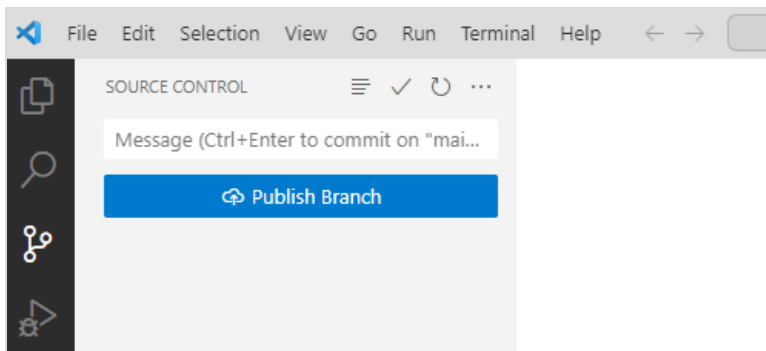
### Lab instruction

1. Create a project folder and name it 'lab2\_git'.
2. Open Visual Studio code and open the project folder 'lab2\_git'
3. Before start the program, check if 'git' is already installed. To do so, we can go to the top menu and click on 'Terminal' tab. From the 'Terminal' list, click on the 'New Terminal' link to open VS code Terminal. In the Terminal, type 'git --version', if it shows a version number, it means that git is already installed in the computer:

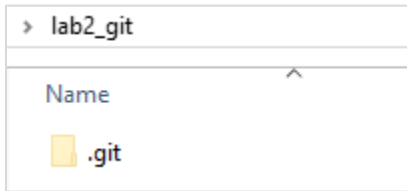


If it doesn't show a version number, it means that git was not installed and you will need to install git.

4. In the project folder, click on the 'source control' icon, which is the third icon from the left column.
5. In the 'source control' has two options 'initialize repository' and 'publish to GitHub'. For this lab, we are going to initialize the repository.
6. After it, it should the source control window the text field to type a message and the button to publish branch as shown below:



- Once the repository is initialized, you should see a hidden holder named 'git' in the project folder:

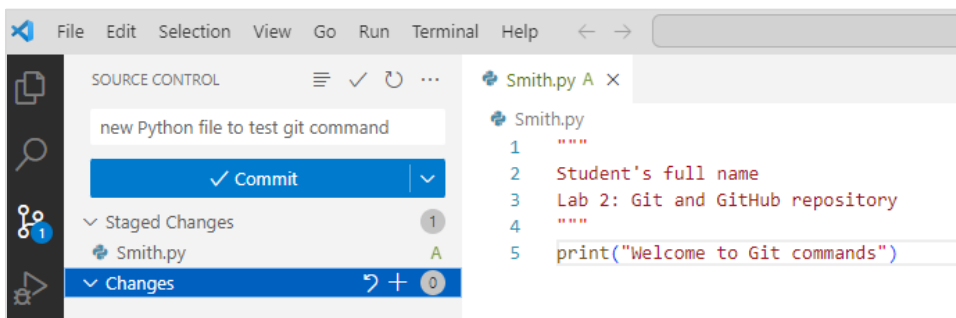


- Create a new Python file named with your last name, for example, if the student's last name is Smith, then we can save the file as 'Smith.py'. In the Python file, we are going to add the following lines:

```
"""
Student's full name
Lab 2: Git and GitHub repository
"""

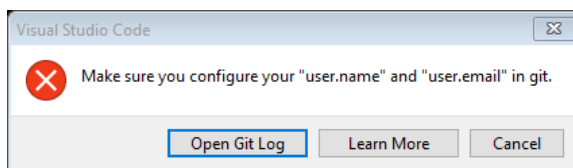
print("Welcome to Git commands")
```

- Save the Python file and click in the 'version control' icon.
- In the 'version control', we are going to commit the changes. For this, we type as a message 'new Python file to test git commands' in message text field.
- After it, we are going to add the changes by clicking on the '+' sign next to the file name. The file should be under 'Changes'.
- After clicking the '+' sign, the Python file should be moved to 'Staged Changes' as shown below:



Click the 'Commit' button to save all changes.

- If the repository is not configured yet, it will pop a window asking to configure the repository:



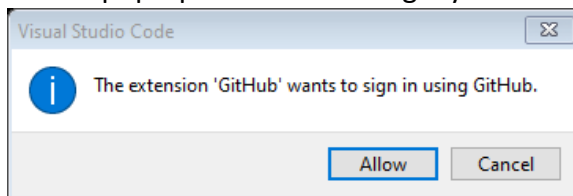
14. To configure the git repository, we go to the Terminal and type:  
`git config --global user.name "professorWu"`

press 'Enter'

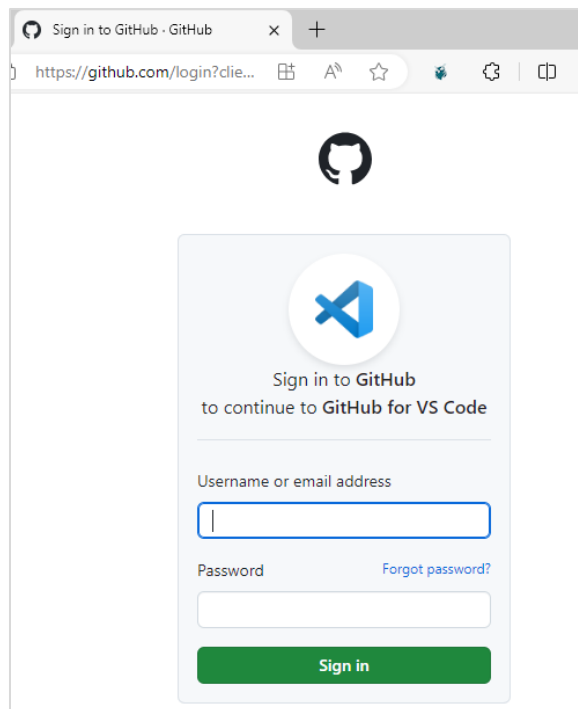
15. Type:  
`git config --global user.email "students@email.com"`

press 'Enter'

16. After if, we publish it to the GitHub repository by clicking on the 'Publish Branch' link. It should pop-up a window asking if you want to sign in using GitHub:



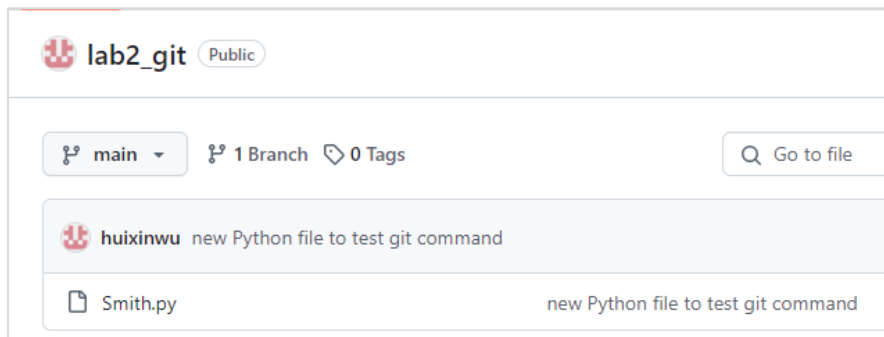
We click 'Allow' and it will take the user to the GitHub account:



17. If you have a GitHub account, log in to your GitHub account, If you don't have a GitHub account, click on 'Create an account'

18. After it, in Visual Studio Code will ask if you want to publish the repository as private or public. For this lab, we are going to set the repository as 'public'

19. Open your GitHub account and you should see the new repository created in your GitHub account as shown below:



20. Back to Visual Studio Code, the following line before the print() command in your Python file:

```
username = input("Enter a username: ")
```

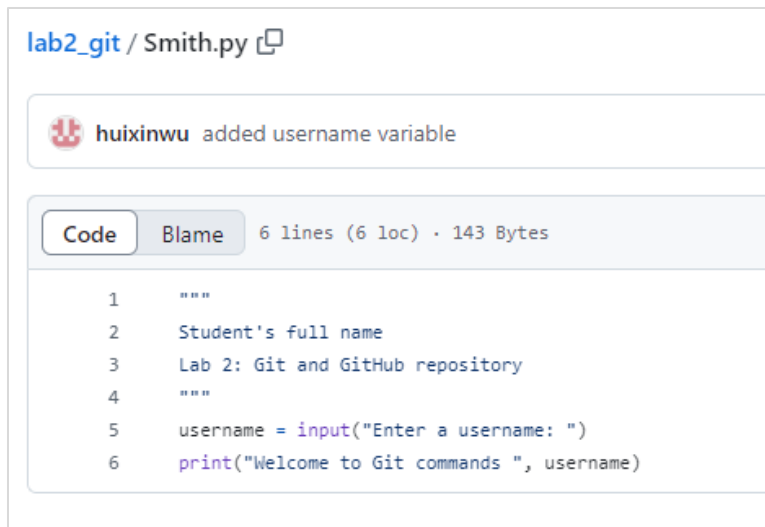
The complete Python file should have the following lines:

```
"""
Student's full name
Lab 2: Git and GitHub repository
"""
username = input("Enter a username: ")
print("Welcome to Git commands ", username)
```

21. After you made the changes to the Python file, in order to save your changes to your repository in GitHub, we must commit the changes. To do so, add the message 'added username variable', click on the '+' sign to stage the changes, and press the 'commit' button.

The link 'Sync Changes' appears and if we want synchronous the changes to our repository in GitHub, we click on 'Sync Changes'.

22. Go to your GitHub repository and click on the Python to open the file. You should see the Python file with the latest update as shown below:



The screenshot shows a GitHub interface for a file named 'Smith.py' in a repository called 'lab2\_git'. At the top, it indicates that user 'huixinwu' added the 'username variable'. Below this, there are tabs for 'Code' and 'Blame', with 'Code' being the active tab. The file statistics show '6 lines (6 loc) · 143 Bytes'. The code content is as follows:

```
1  """
2  Student's full name
3  Lab 2: Git and GitHub repository
4  """
5  username = input("Enter a username: ")
6  print("Welcome to Git commands ", username)
```

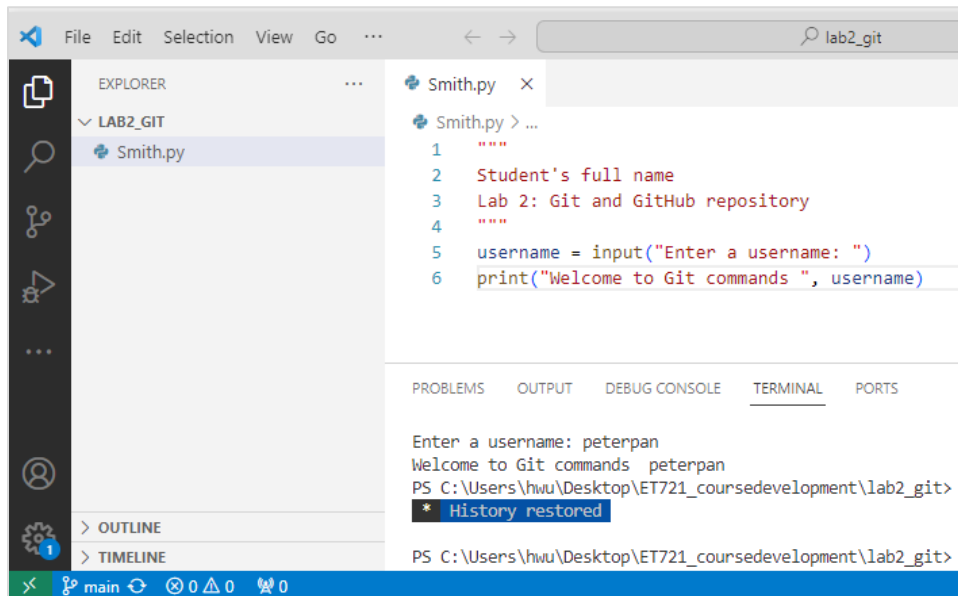
## Branches

In Git, a branch is essentially a separate line of development within a repository. It allows you to work on different features, bug fixes, or experiments without affecting the main codebase. Here's a breakdown of how branches work and why they're useful:

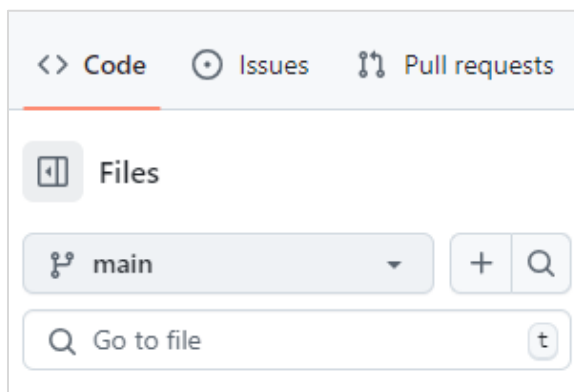
- **Isolation:** Each branch operates independently from the others. This isolation helps you to work on different tasks in parallel. For instance, you might have a main branch (often called master in older repositories) that represents the stable code, and separate branches for new features or bug fixes.
- **Creating a Branch:** When you create a branch, you're essentially creating a new pointer to a specific commit in your project's history. This new branch will start from the point where you created it, but any changes you make will only affect that branch.
- **Switching Between Branches:** You can switch between branches using commands like `git checkout` or `git switch`. This allows you to move your working directory to the state of the branch you want to work on.
- **Merging:** Once you've completed work on a branch, you can merge it back into another branch (e.g., main). Merging integrates the changes from one branch into another, combining the work done in both branches.
- **Deleting Branches:** After a branch has been merged and is no longer needed, you can delete it to keep the repository clean and manageable.
- **Branching Strategies:** There are different strategies for managing branches in a project, such as Git Flow or GitHub Flow. These strategies help organize work and ensure a smooth development workflow.

Branches are a powerful feature in Git because they let you experiment and collaborate effectively, all while keeping your main codebase stable.

23. Go back to VS code, on the bottom-left corner, you can check the branch that the lab 2 is currently working on. By default, when we create the repository, it was created in the 'main' branch:

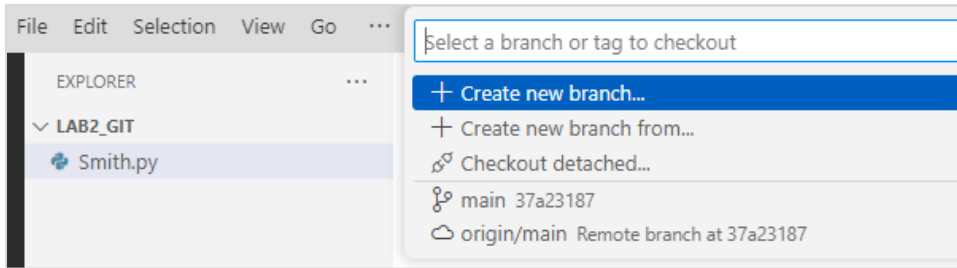


24. Go to your GitHub repository and you will see that repository is linked to the 'main' branch:

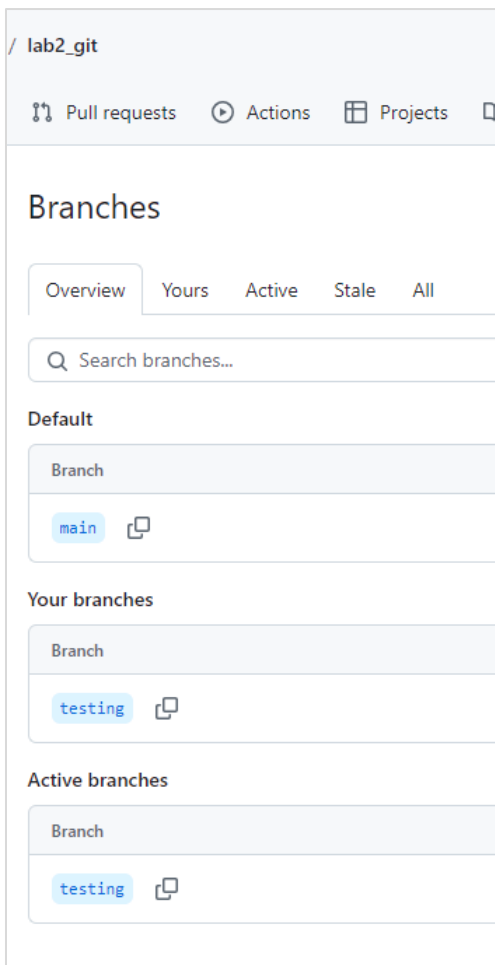


This means that the work done in the local workstation and the remote repository will synch to the 'main' branch.

25. To create a new branch, go the bottom-left corner of VS code and click on version control icon. Once it is clicked, a dialog box appears on the top of the VS code screen as shown in the follow image:

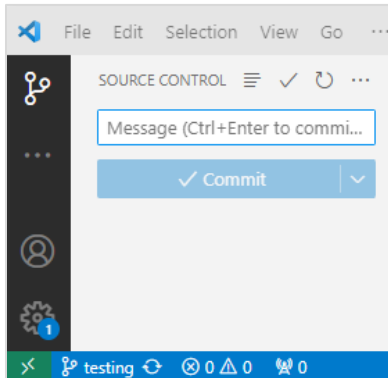


26. Click on 'Create new branch' from the drop-down menu and name the branch as 'testing'
27. After creating the branch, you need to publish it to sync the new branch with your cloud repository on GitHub. To publish the branch, click the version control icon in the left column, and then click on 'Publish Branch'."
28. To check if the new branch is already published, go your repository in GitHub and click on 'Branches', on Branches, it shows which is the Default, Your branches, and the Active branches:





29. In GitHub, if you switch between branches, you will see that they have the same files. However, the changes that we make the file in your repository will only synch to the 'testing' branch. It is because the version control in VS code is linked to the 'testing' branch. To confirm it, you can go back to VS code and check the bottom-left corner icon branch connection, which is linked to 'testing':



30. On VS code, go the Python file and add the following lines:

```
num = 8
print(f"Double of number {num} is {num*2}")
```

31. Go to version control and commit the changes with message 'new lines of code to double a number'. After it, click the 'Sync Changes'.
32. Go back to GitHub and you can check the Python file in both branches, main and testing, and only the Python file in the 'testing' branch saves the changes. The 'main' branches should have the original version of the Python file before the 'testing' branch was created.

Note: You can always switch between branches in VS code by going to the bottom-left corner and click on the version control icon. This should open a dialog box at the top of VS code. From the list in the dialog box, you can see all the branches link to your repository, local and remote.

## Merge and pull request

In Git, "merge" and "pull request" are related but distinct concepts used in the process of integrating changes from one branch to another.

- **Merge** is a Git operation used to integrate changes from one branch into another.
- **Pull Request** is a collaborative workflow feature used to propose, review, and discuss changes before they are merged into a target branch.

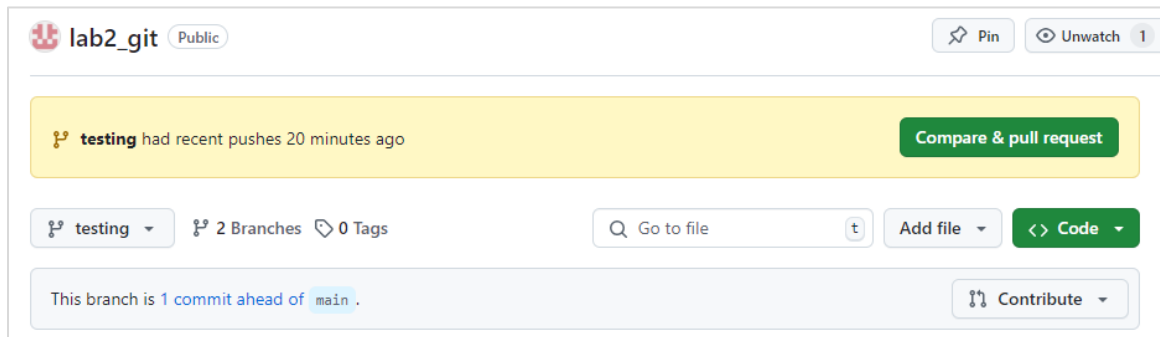
**Merge** is a Git operation that combines changes from one branch into another. Here's how it works:

- **Purpose:** The goal of merging is to integrate changes from different branches. For example, you might merge a feature branch into the main branch to incorporate new features or fixes.
- **Process:** When you perform a merge, Git looks at the changes made in both branches and attempts to combine them. The merge operation typically involves:
  1. **Fast-Forward Merge:** If the target branch has not diverged from the source branch, Git simply moves the target branch pointer forward.
  2. **Three-Way Merge:** If both branches have diverged, Git creates a new commit that reconciles the changes from both branches, using a common ancestor as a reference point.
- **Conflict Resolution:** Sometimes, merging may result in conflicts if changes in the branches are incompatible. Git will prompt you to resolve these conflicts manually before completing the merge.

**Pull Request** is a feature of repository hosting services like GitHub, GitLab, or Bitbucket, and is used to propose and discuss changes before they are merged. Here's what you need to know:

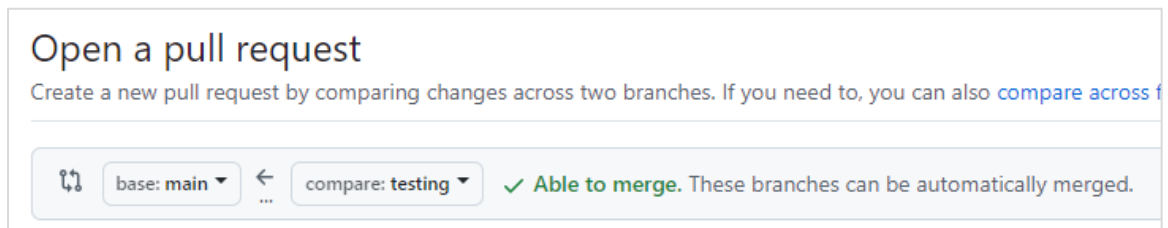
- **Purpose:** Pull requests facilitate code review and collaboration. They provide a way to discuss proposed changes, review code, and ensure that all tests pass before integrating the changes into a target branch (often the main or develop branch).
- **Process:**
  1. **Creating a PR:** You create a pull request from a branch (e.g., a feature branch) into a target branch (e.g., main). This involves selecting the branches and optionally providing a description of the changes.
  2. **Review and Discussion:** Team members review the pull request, leave comments, and suggest changes. This stage helps ensure that the code is of high quality and adheres to project standards.
  3. **Approval and Merge:** Once the pull request is approved, it can be merged into the target branch. The pull request may also include automated checks and tests that must pass before it can be merged.
- **Benefits:** Pull requests provide a structured way to review code, collaborate with team members, and ensure that changes are well-tested and aligned with project goals.

33. To merge and pull changes between repository, on GitHub, select the 'testing' branch and you will see a warning box with a 'compare & pull request':

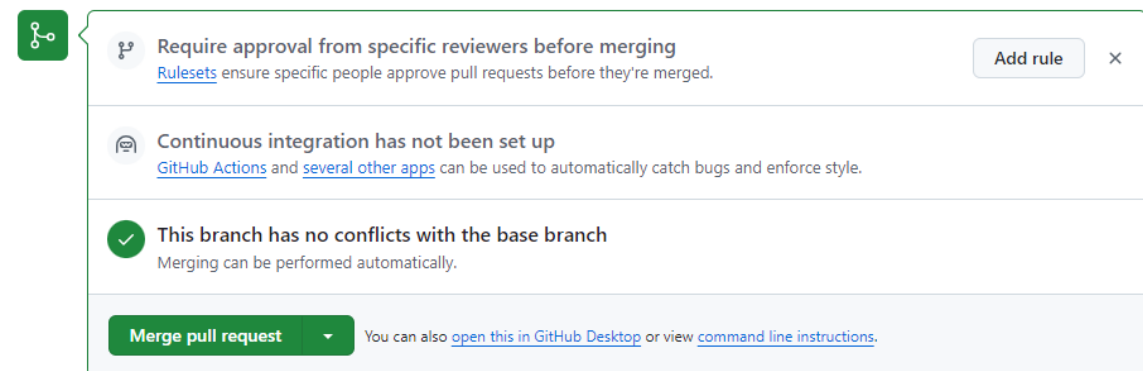


To initialize a pull request, click on the 'Compare & pull request' button.

34. On 'Open and pull request' we can see the direction of the branches that we want to merge. For our lab, we can see that pull will go from the 'testing' branch into the 'main' branch:



35. On the 'Add a description' box, we can write a description to the pull request. For example, for this lab we can write 'new line was tested to working properly. Tested by: student's full name'. After it, click on the 'Create pull request' button.
36. After that, a merge diagnostic will open. If there are no conflicts between the files, it will show a check mark next to 'This branch has no conflicts with the base branch'. This indicates that the branches are ready to be merged. To complete the merge, click on 'Merge pull request', and then 'Confirm merge'.



37. After the merge, you will have an option to keep the branch or to delete it. For this lab, you can give it as reference.

----- **Lab experiment ends here** -----