Git & GitHub Training Module

HKUST COMP4900 Series

WONG, Lap Ming

Hong Kong University of Science and Technology

15th April, 2026

Why this workshop exist

- Learn basics of git
- Learn fundamental skill of version control on code/project
- Having a fundamental skill of version control before taking advance course
- Learn the beauty of GitHub
- Fulfill COMP4900 requirements

What is Version Control

A system that records changes to file(s) over time so that you can recall a specific version later.

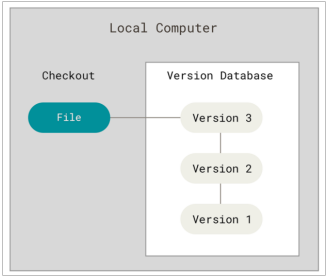
A Version Control System (VCS) allows you to

- Revert selected file back to previous state
- Revert entire project back to previous state
- Compare change over time
- See who last modified the files

Different kinds of version control system

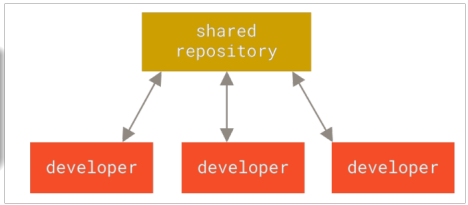
Local Version Control Systems

Having a database to kept all changes to file



Centralized Version Control Systems

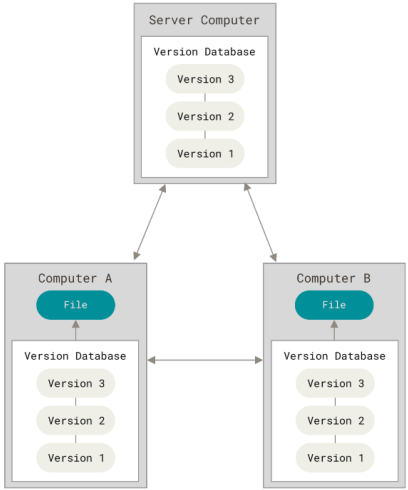
- Have a single server contains all version of file
- Can easily know how other people project are doing



Different kinds of version control system

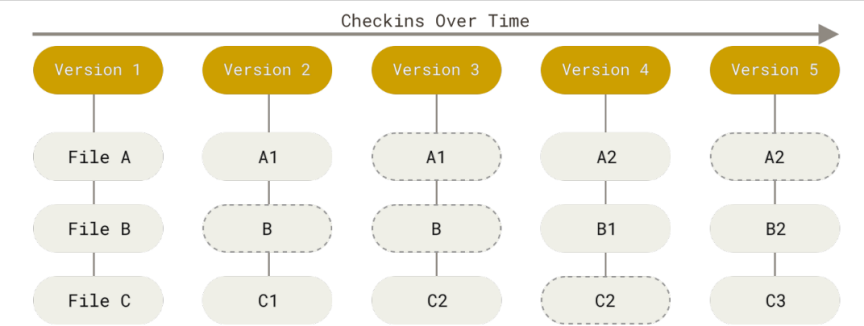
Distributed Version Control Systems

- Clients fully mirror the repository history
- Even a server dies, the system collaborate via that server can be copied back up server to restore it.
- Every clone is a fully backup



What is git

- Git is a **distributed version control system** that helps us track changes in your code over time.
- Think the **change of data** are store as stream of snapshots
- A git repository are having a `.git` folder inside the project



What is GitHub

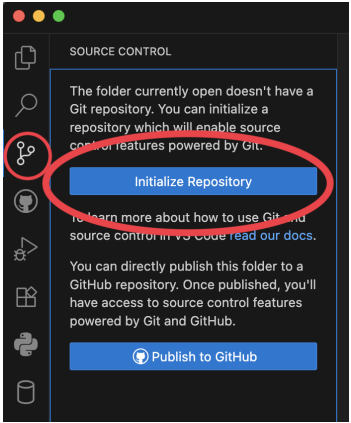
- A website and cloud base services to help us save our code/project
- The most common used platform for saving our repository
- You can share your code online
- Also can be use as developer resume

Installation

In this workshop, we will use Git, GitHub, and VScode to finish our practice.
We will use `https://github.com/HKUST-CRS/crs` as the demonstration repository.
Please follow the website instruction to setup the enviroment before attend the workshop

Getting a Local Git Repository

As we mentioned earlier, a git repository will contain a `.git` folder inside a project.
If we want to make our project a git project, we go to our folder root directory click `Initialize Repository`



Using terminal:

```
1 git init
```

Basic Workflow of a Git Project

A git project contains of three different stage (locally)

Modified

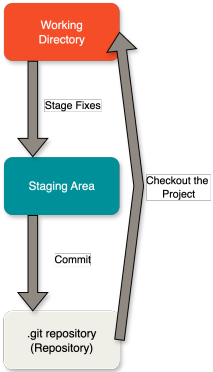
The file have been modified but not committed to your database yet

Staged

The file has been marked as modified in its current version to be included in your next commit snapshot.

Committed

The data is safely stored in your local repository

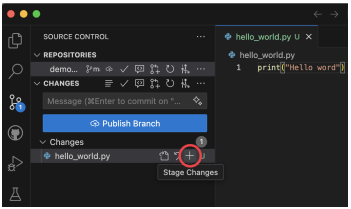


Add a file to staging area

So, how we add the file to the staging area? For example, we create a `hello_world.py` file

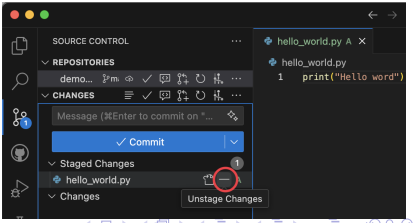
You can add the file to the stage area by clicking + sign
Using terminal:

```
# change hello_world.py to your file you want to
    stage
git add hello_world.py
```



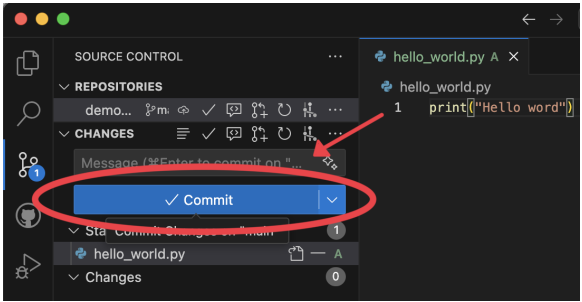
If you want to revert the change in your staging area, just click the – sign
Using terminal:

```
git reset hello_world.py
```



Commit a file from staging area to repository

After you finish staging all the file you want. You can write the commit message and save the change to your local git repository

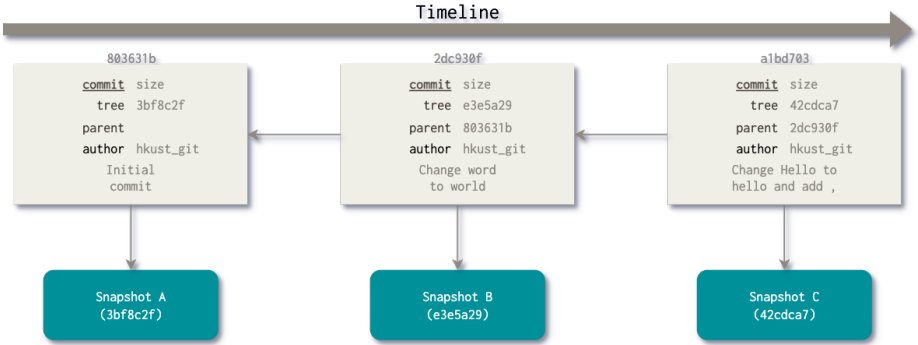


Using terminal:

```
git commit -m "commit message"
```

Commit object

When you do a commit, what you actually do is create a commit object, and the commit object will store a pointer to the snapshot, a pointer to it previous commit, author information and commit message.

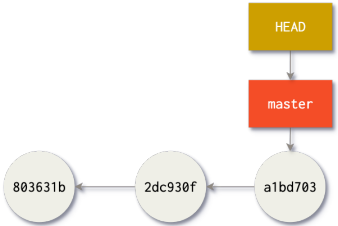


The snapshot save the change of your data in that commit.

Git Branching

Sometimes we might want to do changes of original project (e.g. fixing a bug), but you don't want to do on project directly. Instead of copy and paste the whole repository, you can make use of branch.

A branch is a pointer that pointing to one of these commit, usually the default branch is call `master` branch.

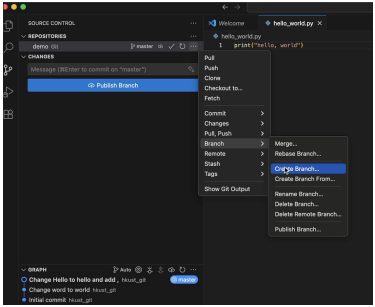


And `HEAD` is a pointer that telling you what pointer you are at currently.

Creating and Switching Branches

In VSCode, you can create a new branch (and change to that branch) by selecting

$\longrightarrow$ Branch $\longrightarrow$ Create Branch
$\longrightarrow$ [type branch name]



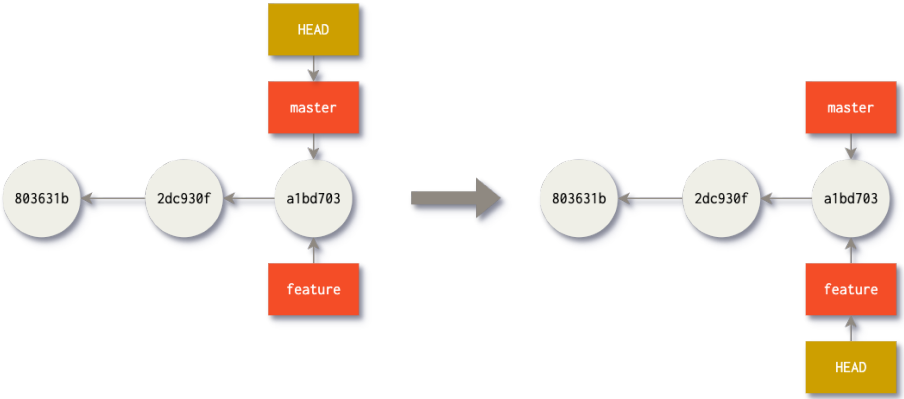
or use either one of this command in terminal

```
1 git branch feature # create new branch
2 git checkout feature # move your head pointer to feature branch
```

```
1 git checkout -b feature
```

Creating and Switching Branches

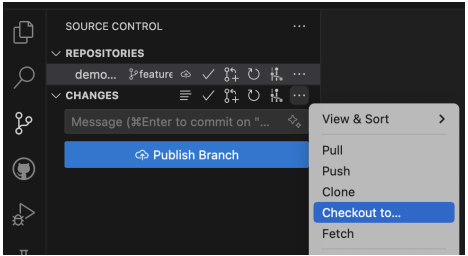
Then it will create a new branch that point to same commit as `HEAD`.



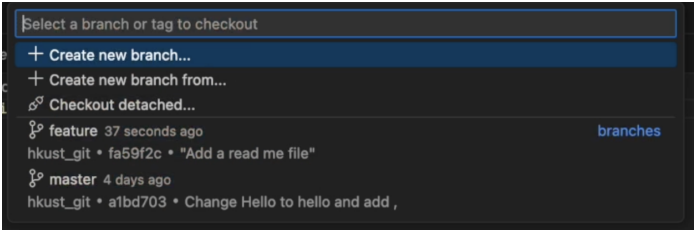
Creating and Switching Branches

In VSCode, you can checkout to other branch by

$\longrightarrow$ Checkout to

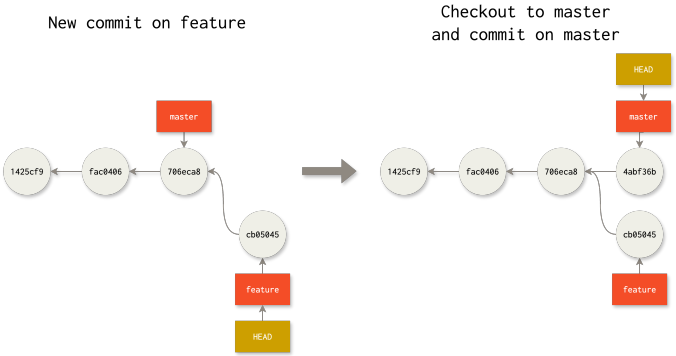


$\longrightarrow$ Select the branch you want to go



Creating and Switching Branches

When you make a commit to the new branch you create, the commit will diverge from the work from your master branch.

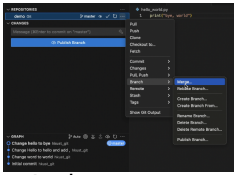


Even when you make a new commit to master branch, it won't affect your new branch.You can make use of this for collaborate development.

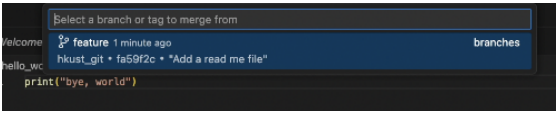
Merge Branch

When you finish your work and want to merge your change together, you can first checkout to your branch you gonna merge into (`HEAD`), and do following:
In VSCode:

$\longrightarrow$ Branch $\longrightarrow$ Merge



$\longrightarrow$ Select the branch you want to merge from

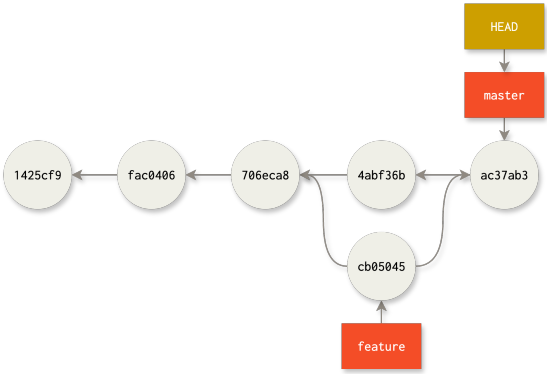


Using terminal:

```
git checkout master # go to branch you want to merge to
git merge feature # merge branch you want to merge from
```

Merge Branch

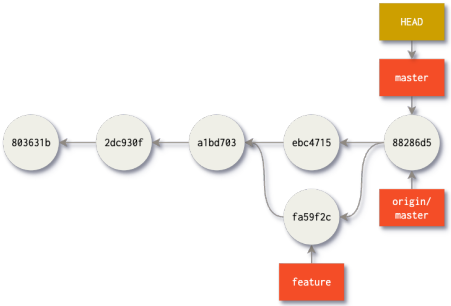
It will merge the change in feature to main branch, and git will automatically do a commit on this merge



Remote repository

Remote repositories are version of your project that hosted internet or otherwhere. The branch in remote repositories are remote branch, which denote as

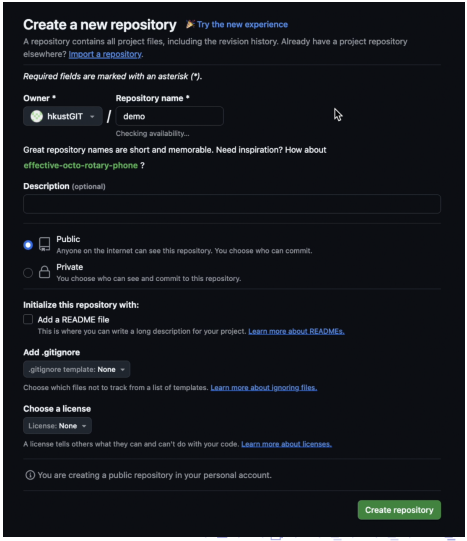
```
<remote-name>/<branch-name>

```



The remote-name is call origin as default

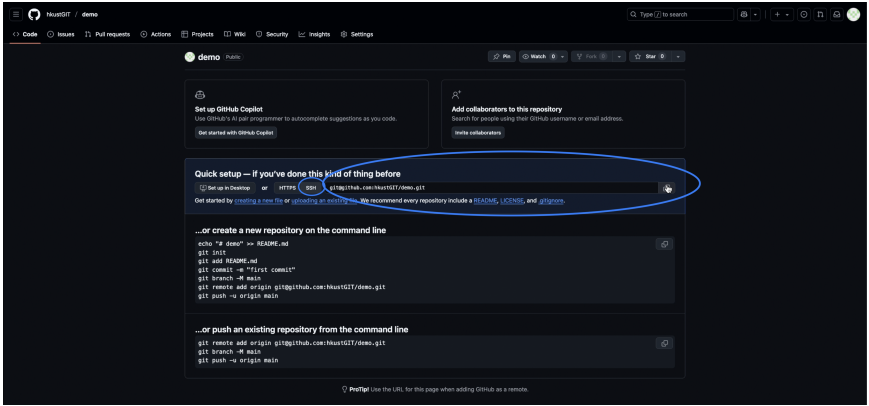
Setting Up GitHub Repository

We will use GitHub to host our remote repository.
First, create a repository in GitHub first
Remember Do **NOT** initialize with README, .gitignore, or license



Setting Up GitHub Repository

Then you copy this link Screenshot focus: copy the repository URL from the green **Code** button.

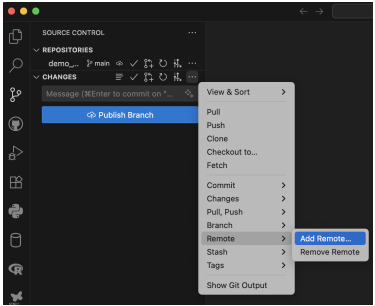


And we will use this link to help us setup the remote

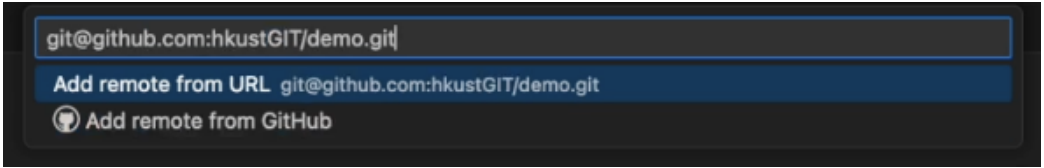
(Remember to setup the ssh in git which mention when installation step)

Adding remote in VSCode

In VSCode
$\rightarrow$ Remote $\rightarrow$ Add Remote

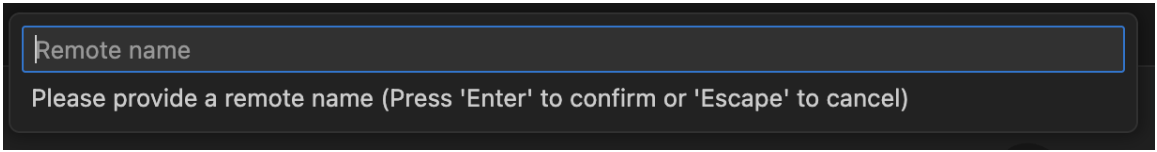


Then you add the link and press enter



Adding remote in VSCode

The type your remote name and press enter Screenshot focus: use a short remote name like `origin`.

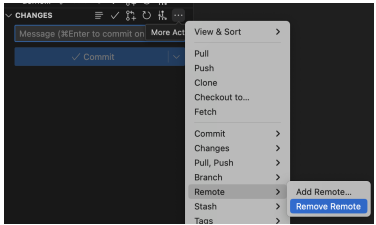


Using terminal:

```
# replace <remote-name> with your remote name
# replace <github-link> with your url
git remote add <remote-name> <github-link>
```

Removing remote in VSCode

In VSCode
→ Remote → Remove Remote



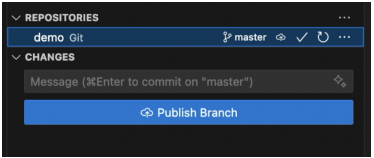
Then you pick the remote you want to remove
Using terminal:

```
# replace <remote-name> with your remote name
git remote remove <remote-name>

```

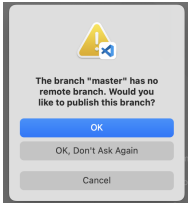
Push your work to repository

After you setup, you can push your branch to repository by

In VSCode → Pull, Push → Push



→ Click ok

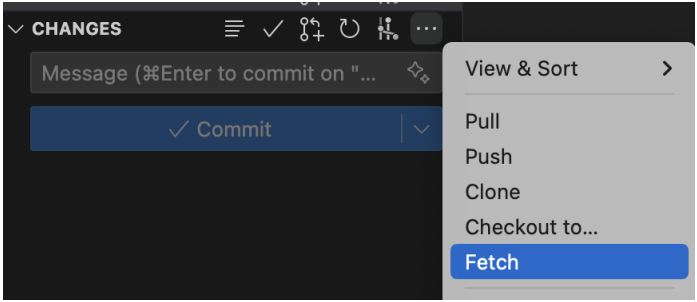


Using terminal:

```
# replace <remote-name> with your remote name
# replace <branch-name> with your branch name or --all if want push all branch
# you don't need --set-upstream at the second time you push
git push --set-upstream <remote-name> <branch-name>

```

Sync from repository

You can use fetch to sync your branch from remote branch in VSCode Screenshot focus: use **Fetch** to download remote updates first without changing your current local files.

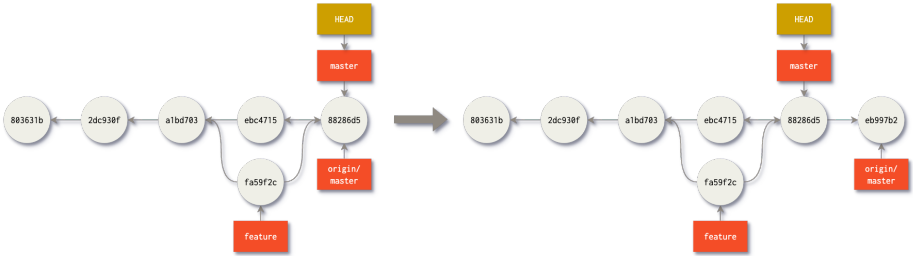


Using terminal:

```
# replace <remote-name> with your remote name
git fetch <remote-name>

```

Sync from repository

This will download the branch at the remote repository Screenshot focus: `origin/master` moves forward, while your local branch stays the same until you merge.



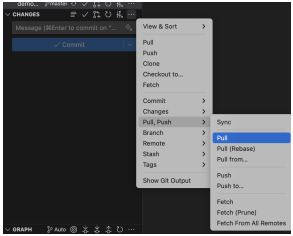
Then you can merge the origin/master to master by

```
1 git merge <remote-name>/<branch-name> <branch-name>
```

Please refer how to do merge in previous page

Sync from repository

Or you can simply just Screenshot focus: **Pull** combines fetch + merge in one step when you are ready to update your local branch immediately.



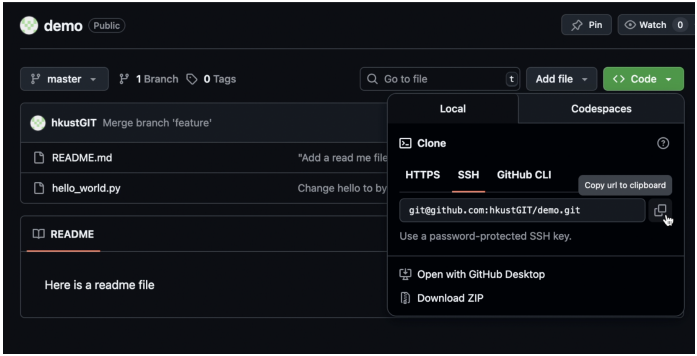
Using terminal:

```
1 git pull <remote-name>
```

This will do the fetch and merge together

Git Clone

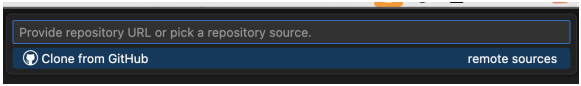
First, get the link from GitHub. Screenshot focus: copy this clone URL, then run `Git: Clone` in VSCode.



In VSCode: `Command+Shift+P` and run `Git Clone`.

Git Clone

Then paste your link here Screenshot focus: paste the repository URL and confirm.



And select your repository destination Screenshot focus: choose the destination folder for the local clone.

Or in terminal, go to your directory, and

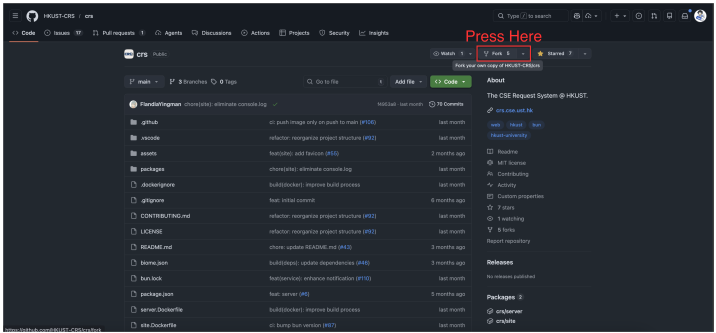
```
# change <remote-url> to your repository want to clone
git clone <remote-url>
```

When you use `git clone`, Git automatically creates a remote called `origin`.

Fork on GitHub

Fork creates your own copy of another repository under your GitHub account.

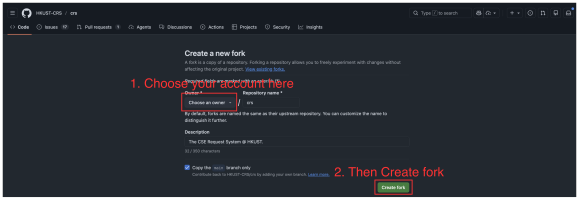
- Use **clone** when you already have write access to the original repository.
- Use **fork** when you do not have write access (class tasks / open-source).
- You can submit changes back by opening a Pull Request from your fork.

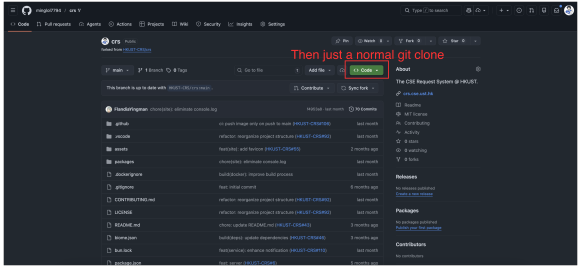


Screenshot focus: click **Fork** on the original repository to create your personal copy.

Fork on GitHub

For this workshop, use `https://github.com/HKUST-CRS/crs` as the repository to fork. Screenshot focus: left image shows fork creation options; right image confirms the new repository under your account.

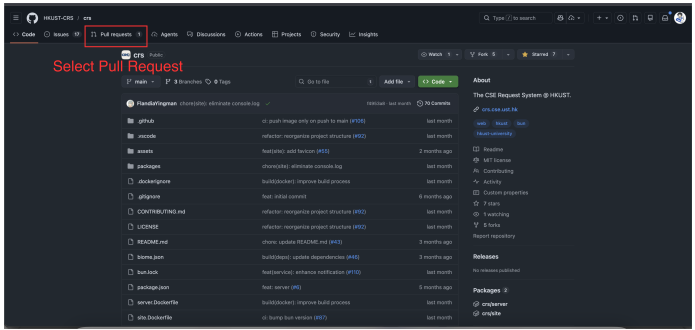




Pull Request Workflow

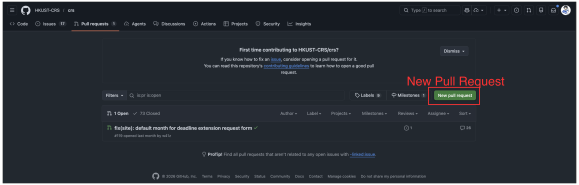
A Pull Request (PR) is used to propose, discuss, and merge changes safely.

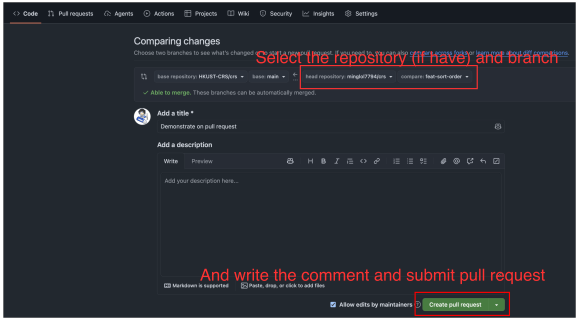
1. Open the original repository and go to **Pull requests**
2. Click **New pull request**
3. Select base branch and your fork branch
4. Write title/description and create the PR



Pull Request Workflow

Screenshot focus: compare branches carefully, then complete the PR form before submission.

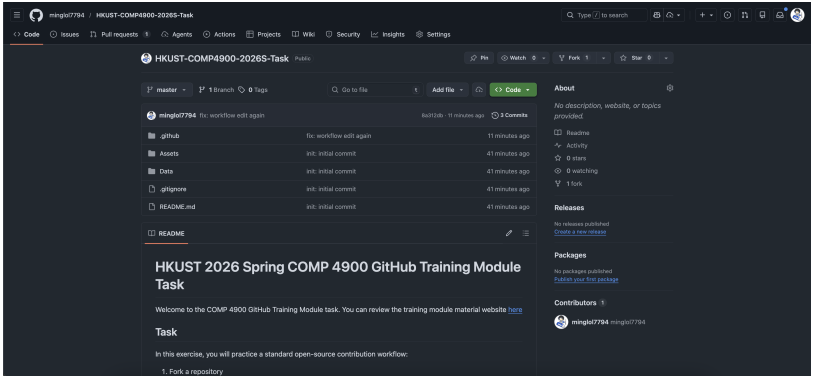




That all!
Any Question?

Workshop Task Overview

We will use the following repository for our workshop task Screenshot focus: make sure you are opening the correct COMP4900 task repository before starting.



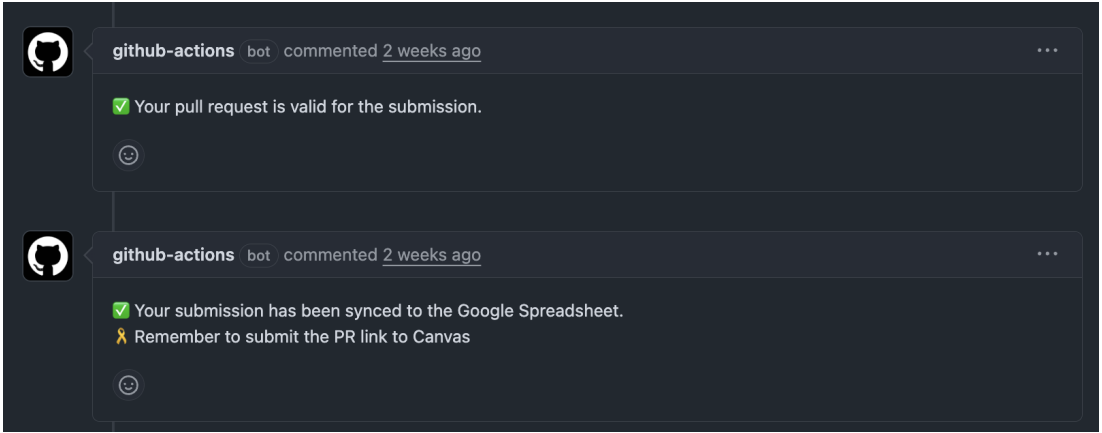
Original repository: `https://github.com/minglol7794/HKUST-COMP4900-2026S-Task`

Workshop Task Instruction

1. Fork the original task repository
2. Clone your fork to local machine
3. Create a new branch (example: `add-my-student-info`)
4. Edit `Data/student.json` with your own record
5. Commit and push your branch
6. Open a Pull Request to the original repository
7. Wait for auto-validation result in PR comments
8. Submit your PR link on Canvas

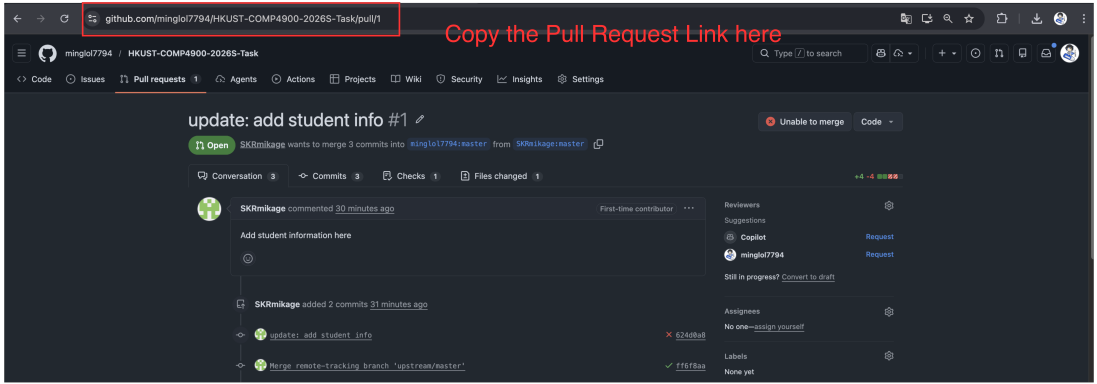
Workshop Task Instruction

Success message example after validation: Screenshot focus: this confirmation indicates your PR checks passed and submission is accepted.



Workshop Task Instruction

Copy your PR link from browser address bar and submit to Canvas. Screenshot focus: copy the full PR URL from this page, not the repository homepage URL.

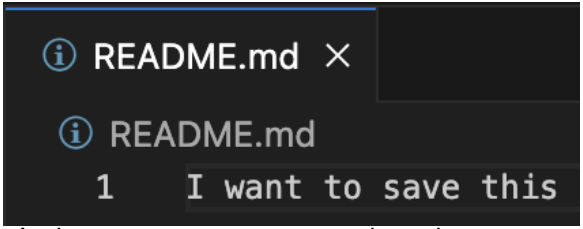


You finish the workshop task when the PR is valid and the link is submitted.

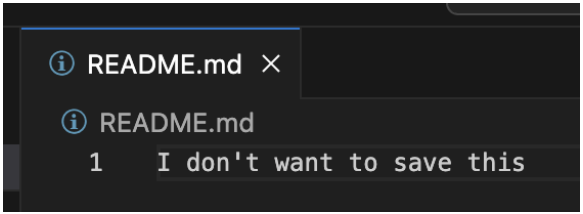
Extra - Merge Conflicts

Sometimes you might edit the same file in two different commit. And you will end up with merge conflicts when they merge together. For example, I got two different content in my `master` branch and `wrong` branch

→ Master Branch



→ Wrong Branch

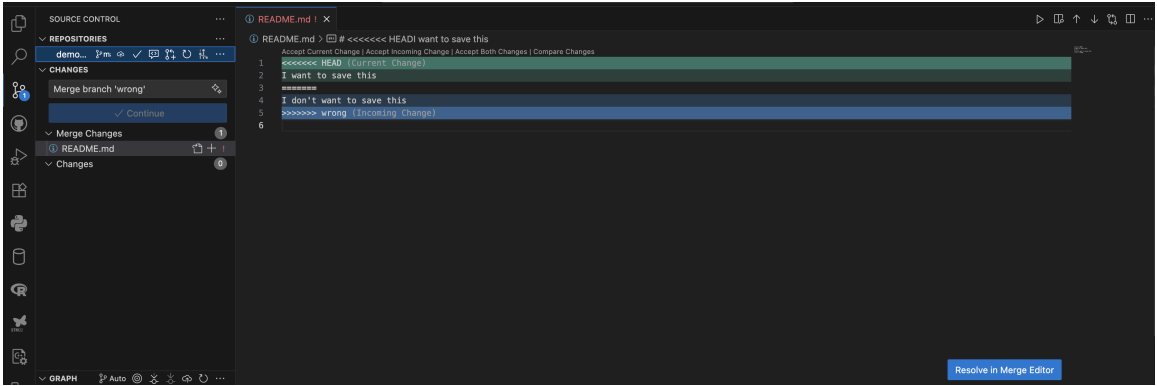


And you try to merge wrong branch to master branch

```
1 git checkout master
2 git merge wrong
```

Extra - Merge Conflicts

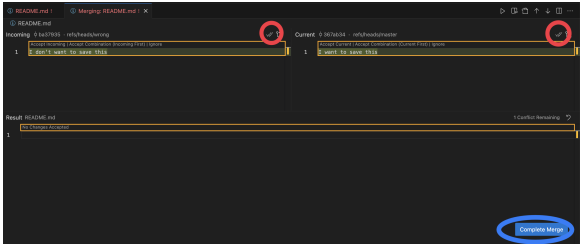
Then it will have merge conflict Screenshot focus: this panel highlights files with conflicts and gives access to `Resolve in Merge Editor`.



You can click `Resolve in Merge Editor` and resolve the conflict

Extra - Merge Conflicts

You can select what you want to keep, and click `Complete Merge`



Then click `Continue` on the CHANGES tab