CAE 暑期實習 Workshop

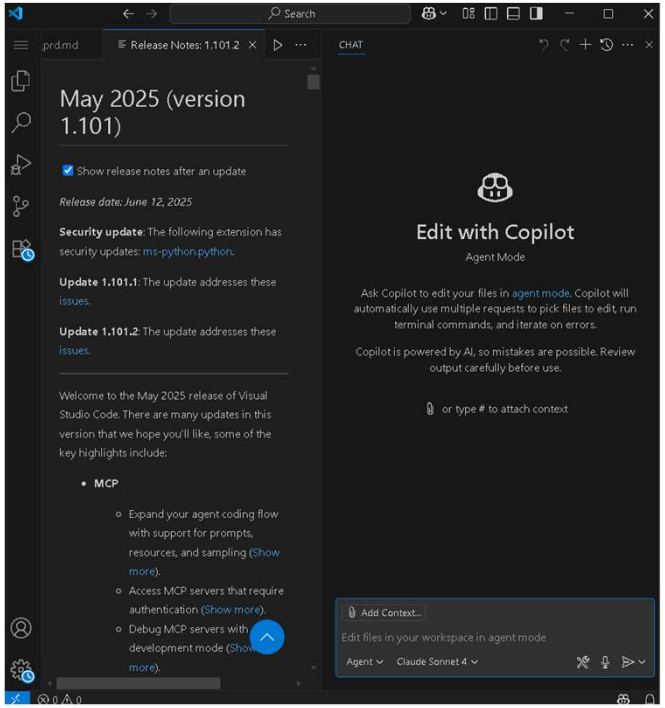
Git & GitHub Tutorial

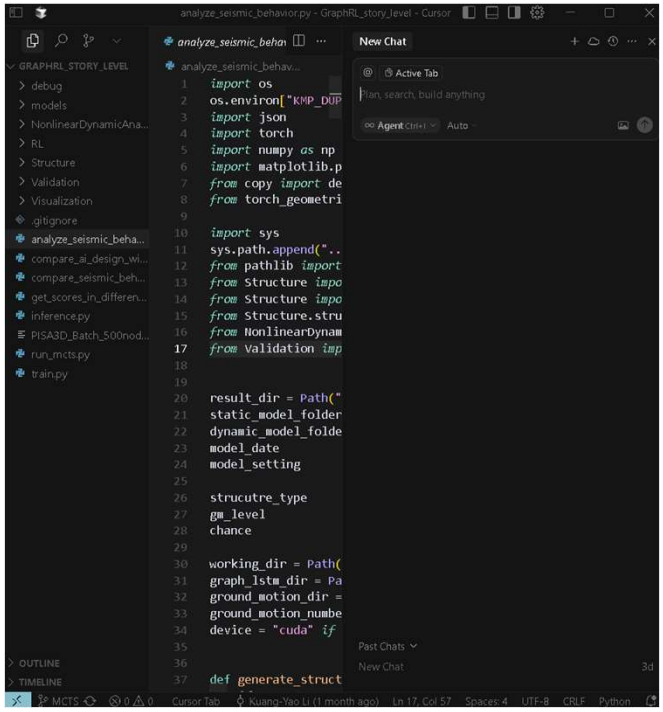
CAE 碩二 張壹翔
Kyle_chang@caece.net

Vibe Coding?

CURSOR

Coding agent in IDEs





GitHub Copilot in VS Code

https://docs.github.com/en/copilot/how-tos/manage-your-account/get-free-access-to-copilot-pro

CURSOR

https://cursor.com/cn/students

Version control systems → Git!

What is Git?

- Git is the most popular **version control system** that helps developers track and manage changes to their code over time.
- It's like a **time machine** for your project — you can go back to earlier versions, see what changed, and fix things if something breaks.

Git can do more...

- Save snapshots of your work
- Track changes of files
- Travel back to old versions
- Revert and recover
- Merge changes
- Collaborate with others

What is GitHub?

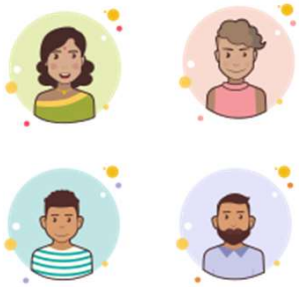
- **GitHub** is an **online platform** that lets you **store**, **manage**, and **share Git repositories** online.

- Think of it as a **social network for developers** and **a home for your code**.

Why Git and Github?

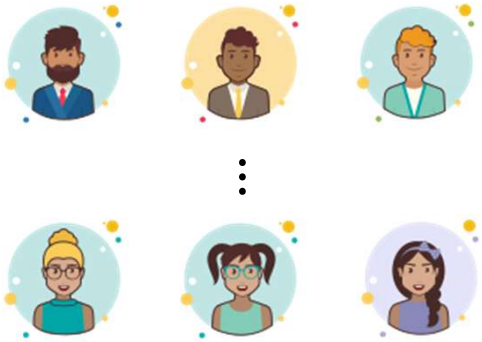
Single Person

- Make a series of huge mistakes, want to go back to the old, correct version?
- Thinking of trying vibe coding, but not sure if the AI-generated code will work?



Team

- How to share new codes with other members? Line? Drive?
- What if your code have conflict with others?

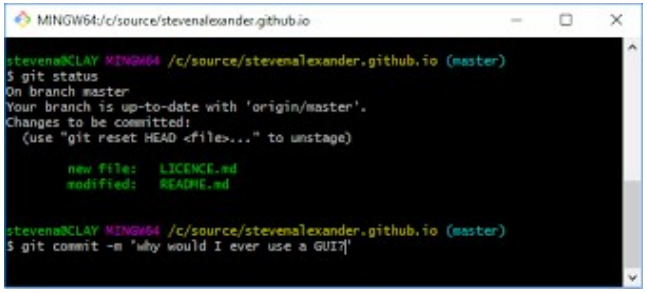


Open Source Project

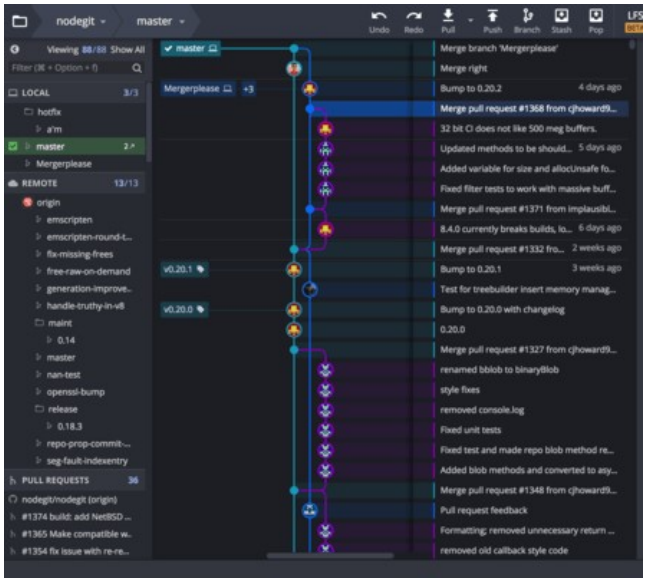
- Have a great feature idea on others Github project, use email to tell them this good idea?

Git Interface

- Command line

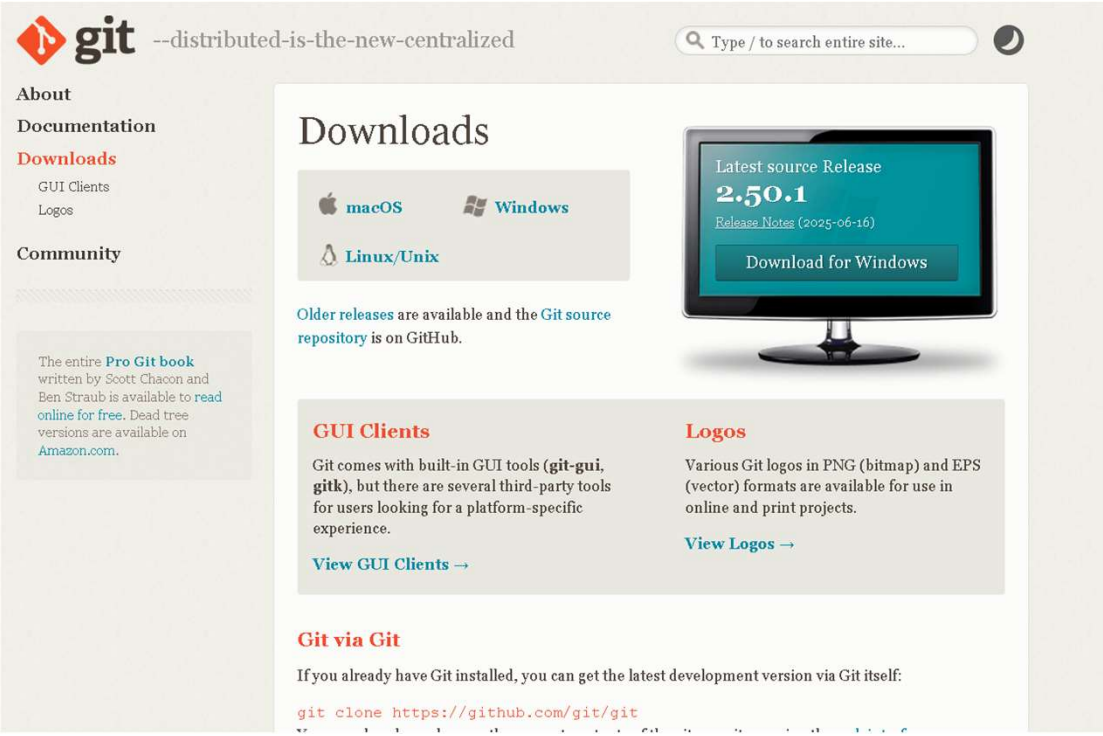


- Graphical user interface



GitKraken

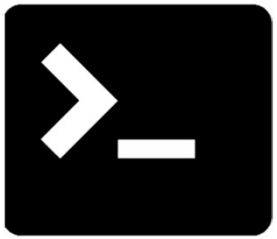
Git Installation



https://git-scm.com/downloads

Check Installation

- Open up your terminal



Command prompt

or

Git Bash on Windows

- Verify installation

```
> git --version
```

Configuring Git

```
> git config --list
```

- Setup user name & email

```
> git config –-global user.name ”Kyle Chang"
> git config –-global user.email “kylechang@gmail.com"
```

- Setup text editors

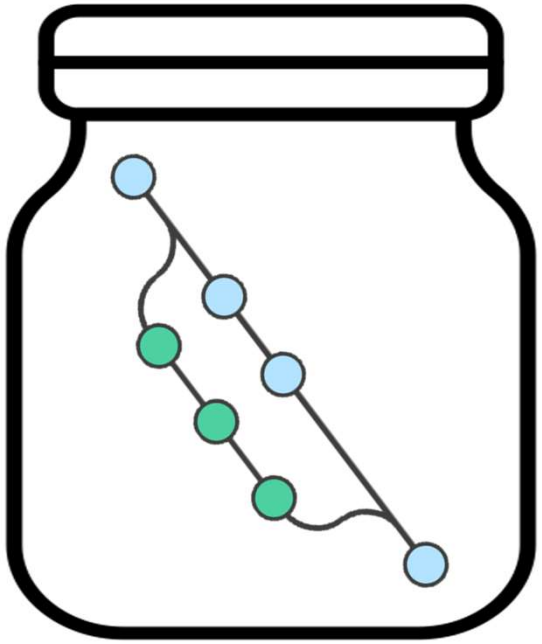
```
> git config --global core.editor "code --wait"
```

1. Git Basic Commands

Repository

A Git "Repo" is a workspace which tracks and manages files in a folder.

When we want to use Git with a project, we need to create a new git repository. We can create as many git repo as we want in the computer. Each of them will have separate histories and contents.



Status, Initialization

- get the status of the repo (use this constantly to check repo status!)

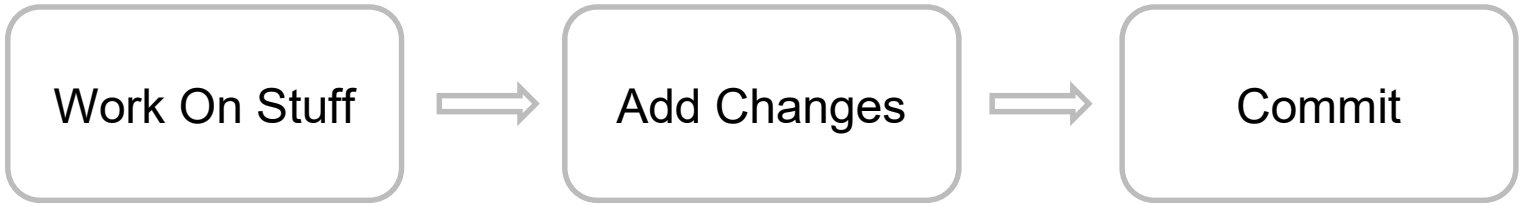
```
> git status
```

- initialize a project folder as a git repo

```
> git init
```

Add, Commit

The Basic Git Workflow

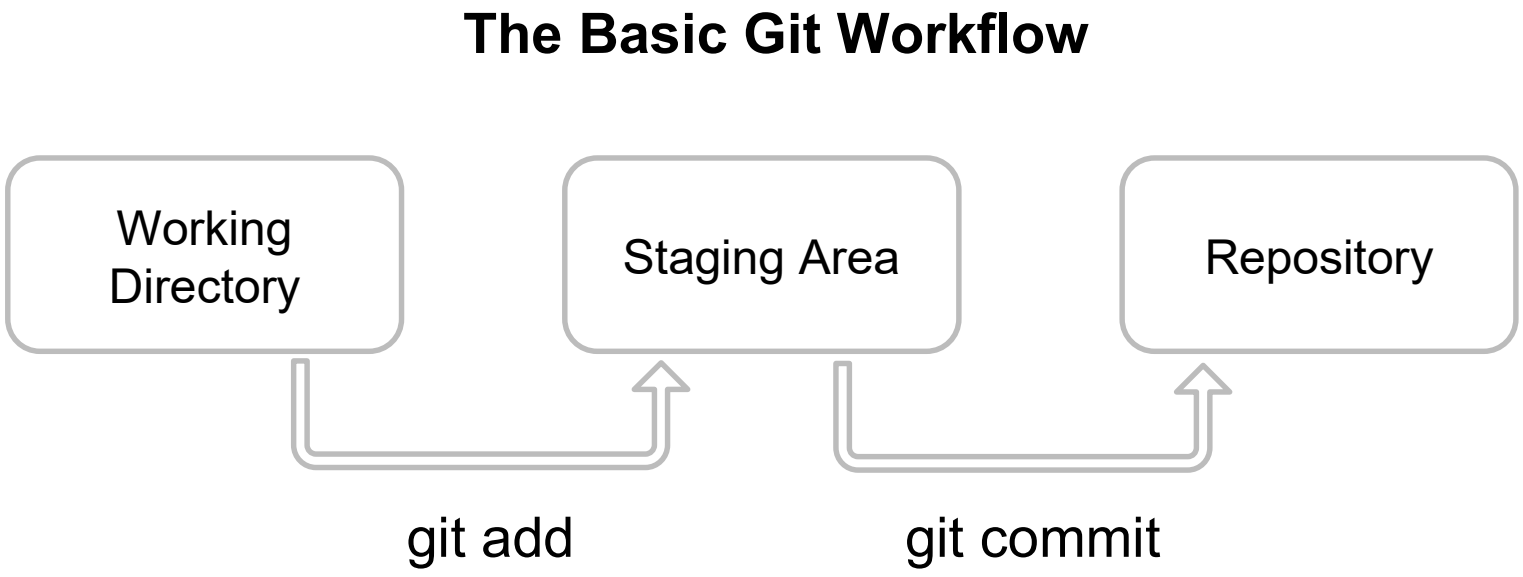


Make new files, edit files, delete files…

Group specific changes together, prepare for commit

Commit everything that are added previously

Add, Commit


The Basic Git Workflow
Working Directory
Staging Area
Repository
git add
git commit

Add, Commit

- stage the changes → "Git, please include these files in our next commit"

```
> git add file1 file2
> git add .
```

- commit the staged changes and provide a message that summarized the changes

```
> git commit –m "my message"
```

Log

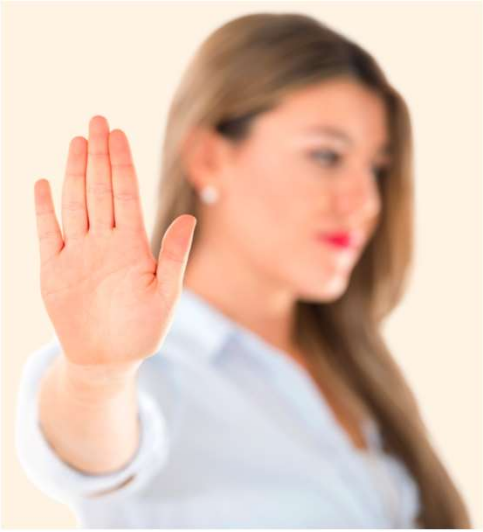
- Use git log to review and read the history of everything that happens in a repository

```
> git log
> git log --oneline
```

.gitignore

We can tell Git which files and directory to ignore in the repo, using a **.gitignore** file. This is useful for files you know you never want to commit, including:

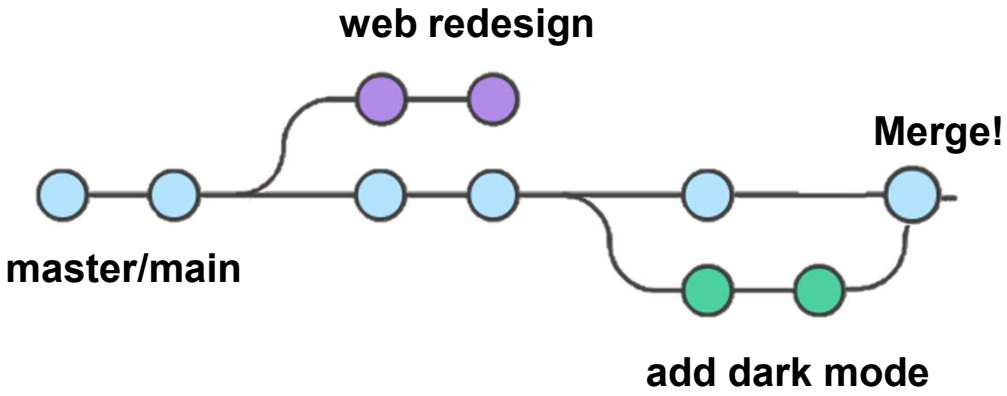
- secrets, API keys
- .DS_Store
- __pycache__/file.pyc



Branch

- Branches enable us to create separate contexts where we can try new things, or even work on multiple ideas in parallel.

- If we make **changes** on one branch, they **do not impact the other branches** (unless we merge the changes).



Branch

- View current branch

```
> git branch
> git branch -v
```

- create new branch and switch to the new branch

```
> git branch <branch-name>
> git switch <branch-name>
```

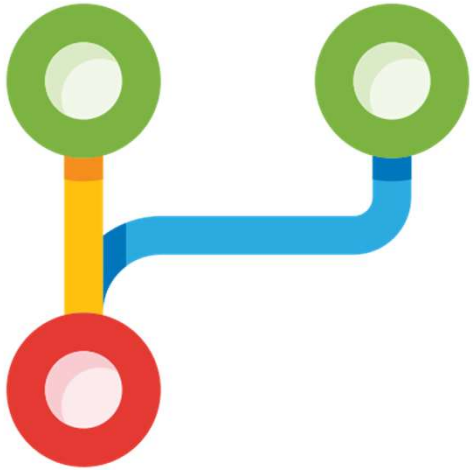
```
> git switch -c <branch-name>
```

```
> git branch -d <branch-name>
```

Merge

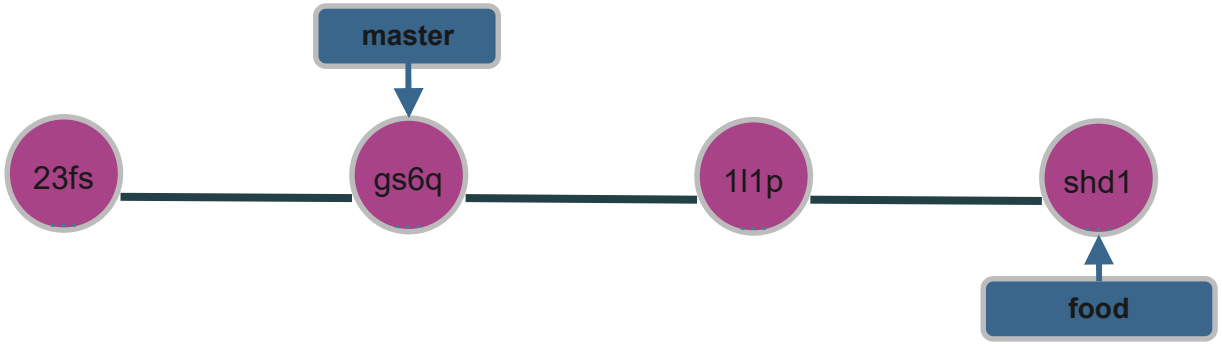
- Branching make it is easy and clear to work on a separate context. However, sometimes we will want to incorporate changes from one branch to another!
- Two step in merging:

1. switch to the branch you want to merge changes into
2. merge changes from a specific branch into the current branch



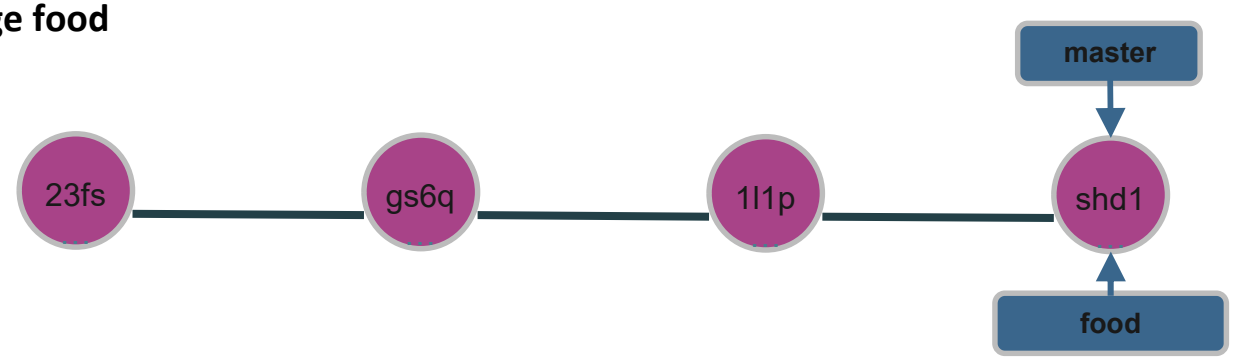
```
> git switch master
> git merge food
```

Fast-Forward Merge

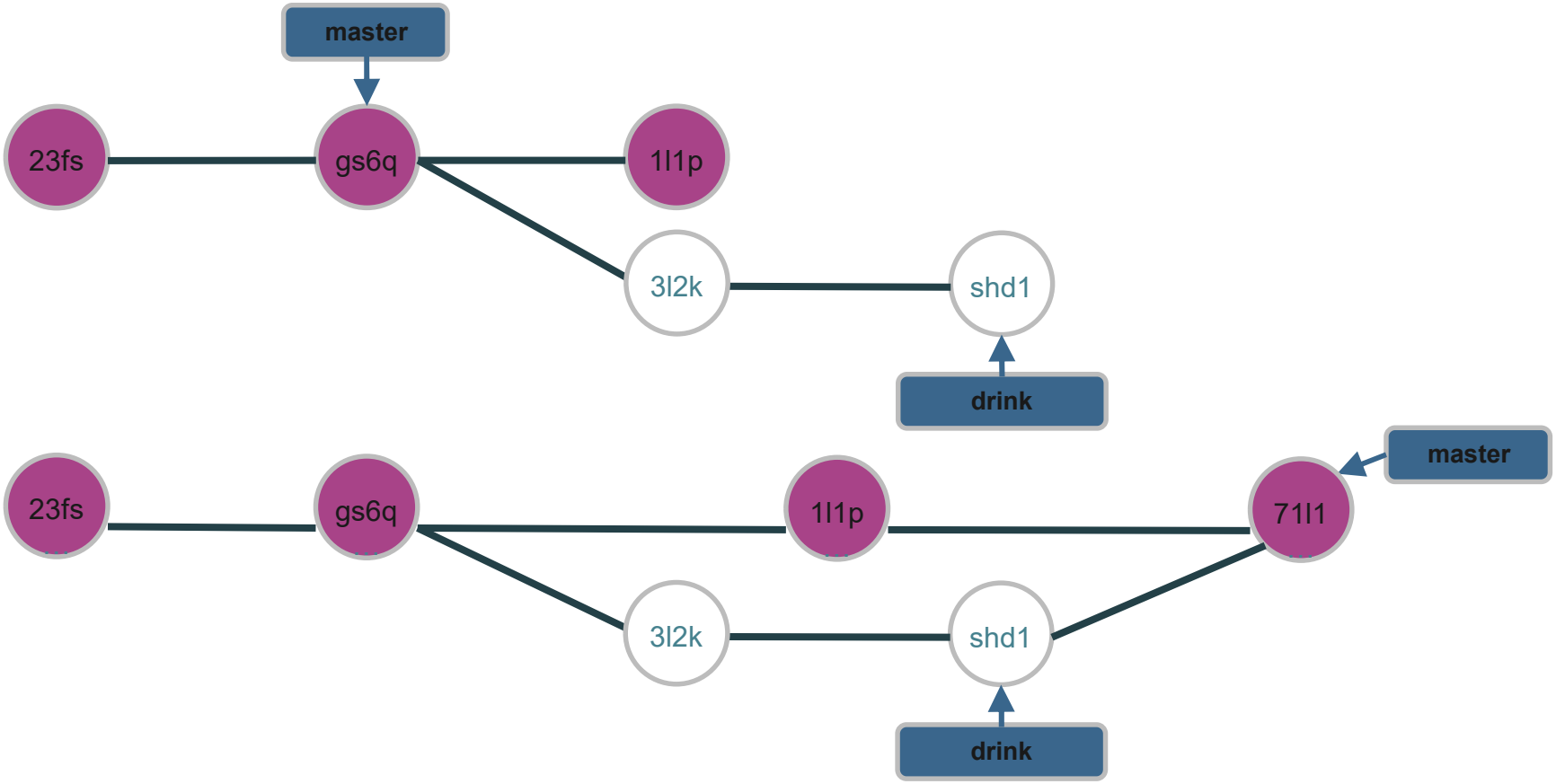


> git switch master

> git merge food

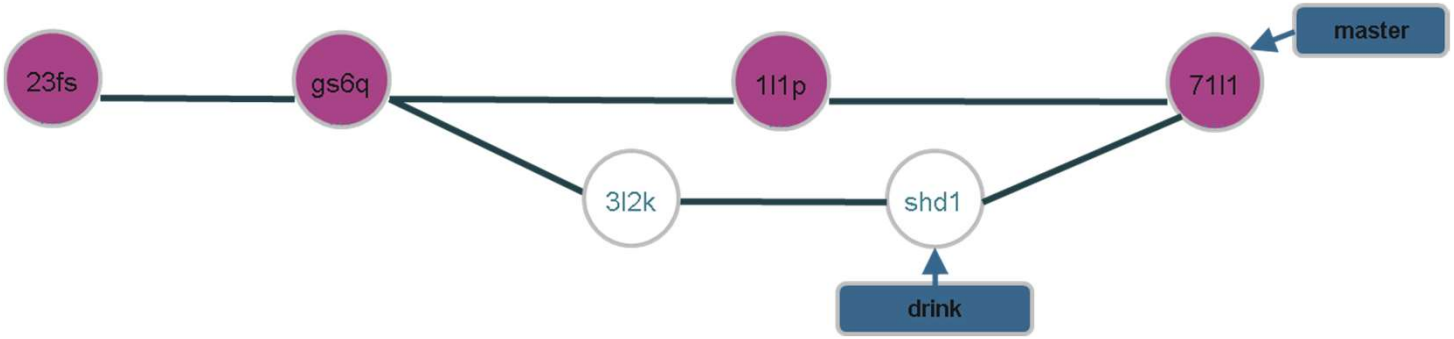


Not All Merges Are Fast Forward Merge!



Conflict Solving

- When you encounter merge conflicts, follow these steps to resolve them:
  1. Open the file(s) with merge conflicts
  2. Edit the file(s) to remove the conflicts. Decide which to keep and remove the conflict markers
  3. Add your changes and make a commit!



Diff

- We can use **git diff** command to view changes between files, commits, branches, working directory. It can help us get a better picture of how the repo changed over time.

```
> git diff
> git diff HEAD
> git diff --staged
> git diff [filename]
> git diff branch1 branch2
> git diff commit1 commit2
```

→ list changes between staging area and working directory

→ list all changes since last commit

→ list changes between staging area and last commit

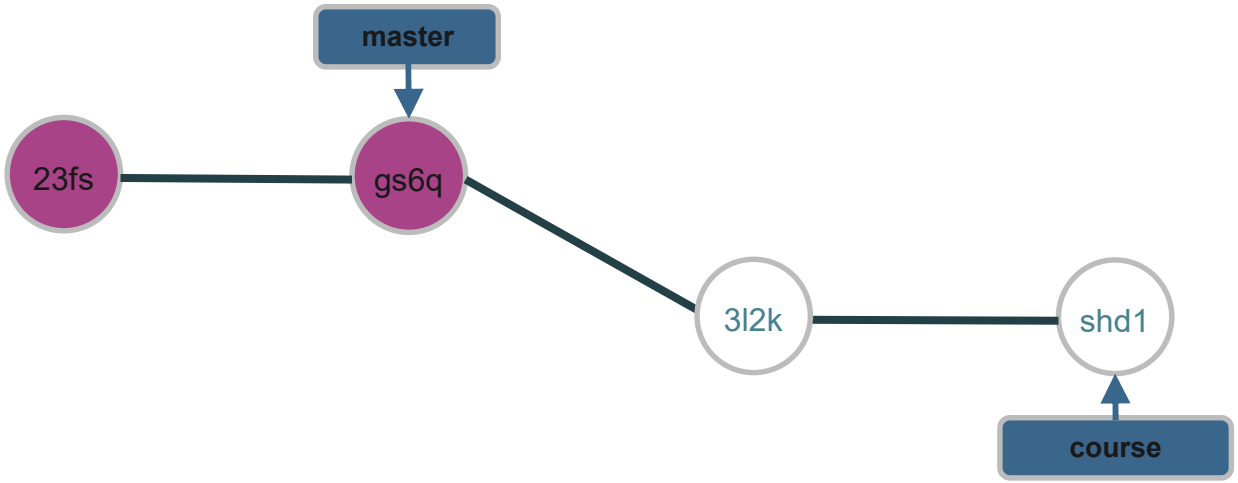
→ list changes in a specific file

→ list changes between two branches

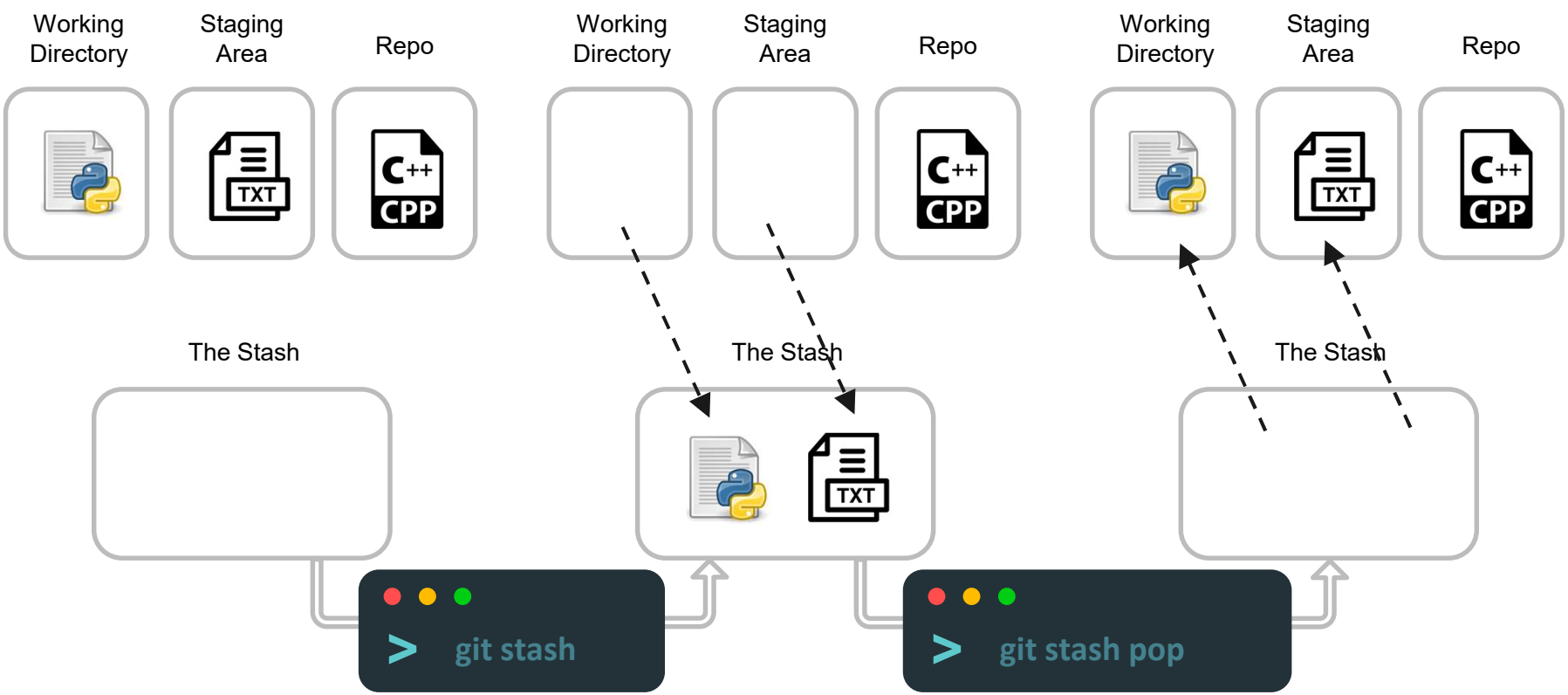
→ list changes between two commits

Stash

- Imagine I am working on branch course, and Boss ask me to check something in the master branch. I've some new work on course, but I don't want to submit yet. What happens if I just switch back to master?



Stash


Working Directory
Staging Area
Repo
TXT
C++
CPP
The Stash
git stash
git stash pop

Undoing Stuff and Time Traveling

- git checkout
- git restore
- git reset
- git revert

Checkout

- We can use git checkout command to view a previous commit

```
> git checkout <commit-hash>
```

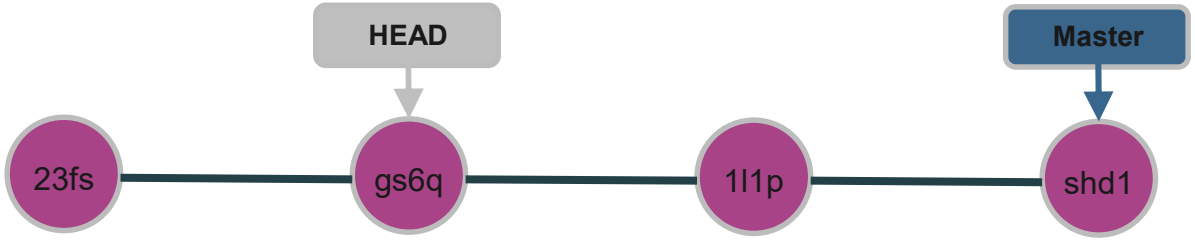
- You may see detached head (斷頭！？), don’t get panic!

```
You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.
```

detached HEAD

You have a couple actions in this state:

1. Stay in detached HEAD to examine the contents of old commit
2. Leave and go back to wherever you were before → git switch master
3. Create a new branch and switch to it, then can now make and save changes → git branch <new-branch>



Restore

Unmodifiying Files

- Suppose you've made some changes since last commit, but you realize you do NOT want these changes anymore!

```
> git restore <file-name>
> git restore –source <commit-hash> <file-name>
```

Unstaging Files

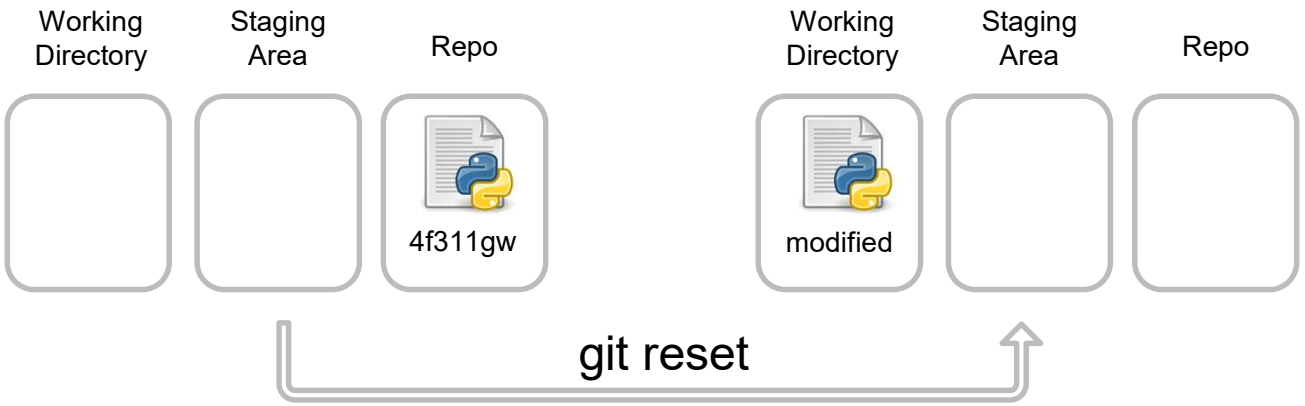
- If you accidentally added a file and you want to remove it

```
> git restore --staged <file-name>
```

Reset

- It will reset the repo back to a specific commit. The commits are gone.

```
> git reset <commit-hash>
```

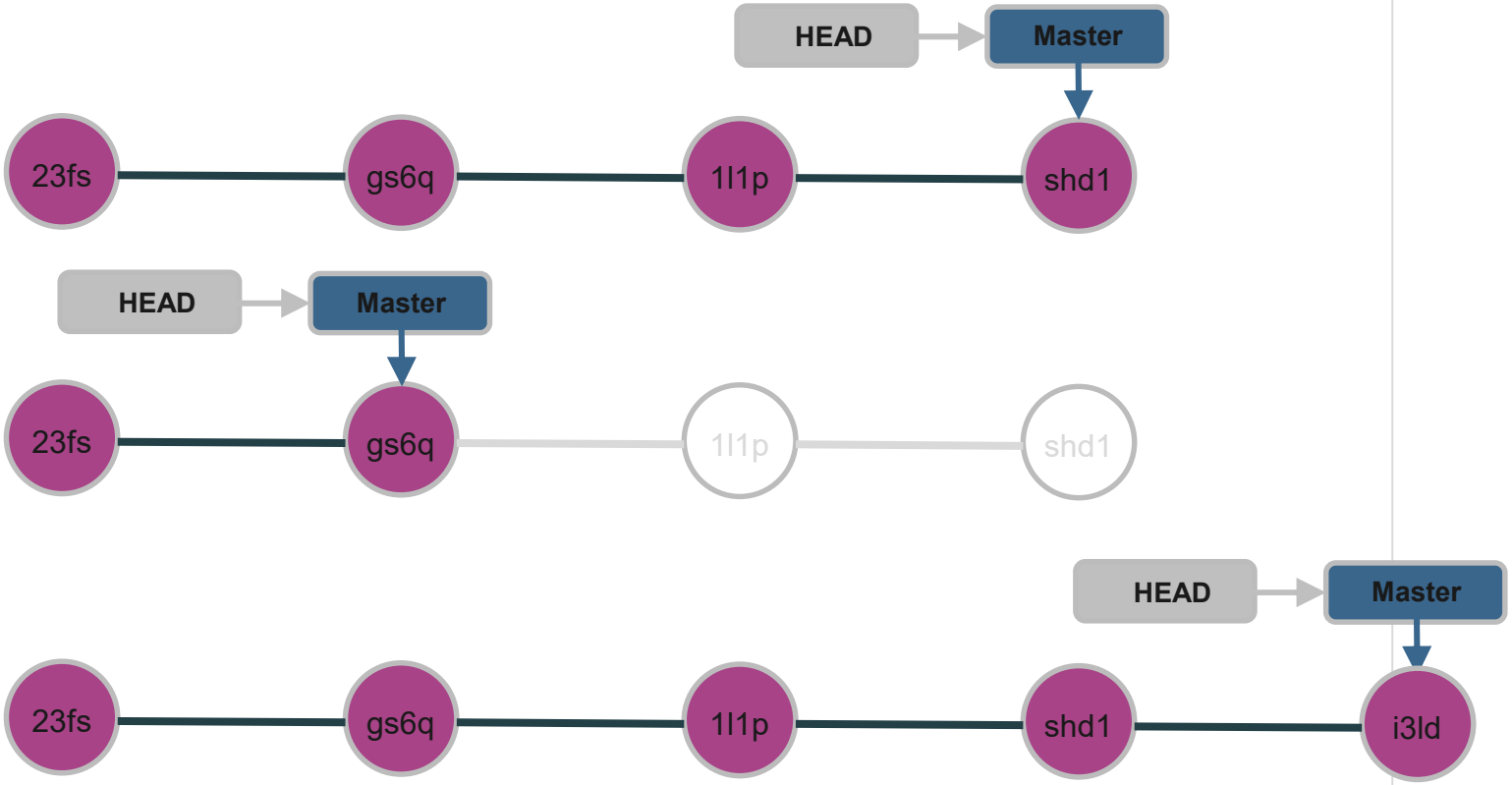


Revert

- **git revert** is similar to **git reset** as they both undo changes, but accomplish in different way.
- **git reset** moves the branch pointer backward, eliminating commits.
- **git revert** instead creates a brand new commit which undo the changes from a commit.

```
> git revert <commit-hash>
```

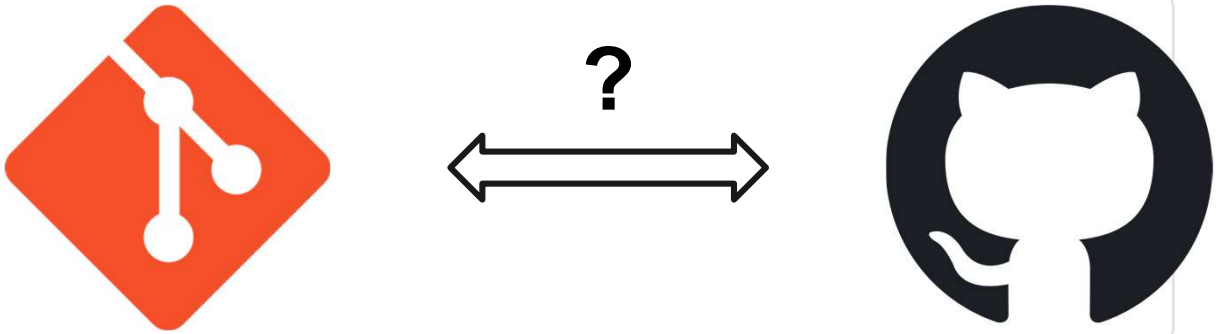
Reset vs. Revert



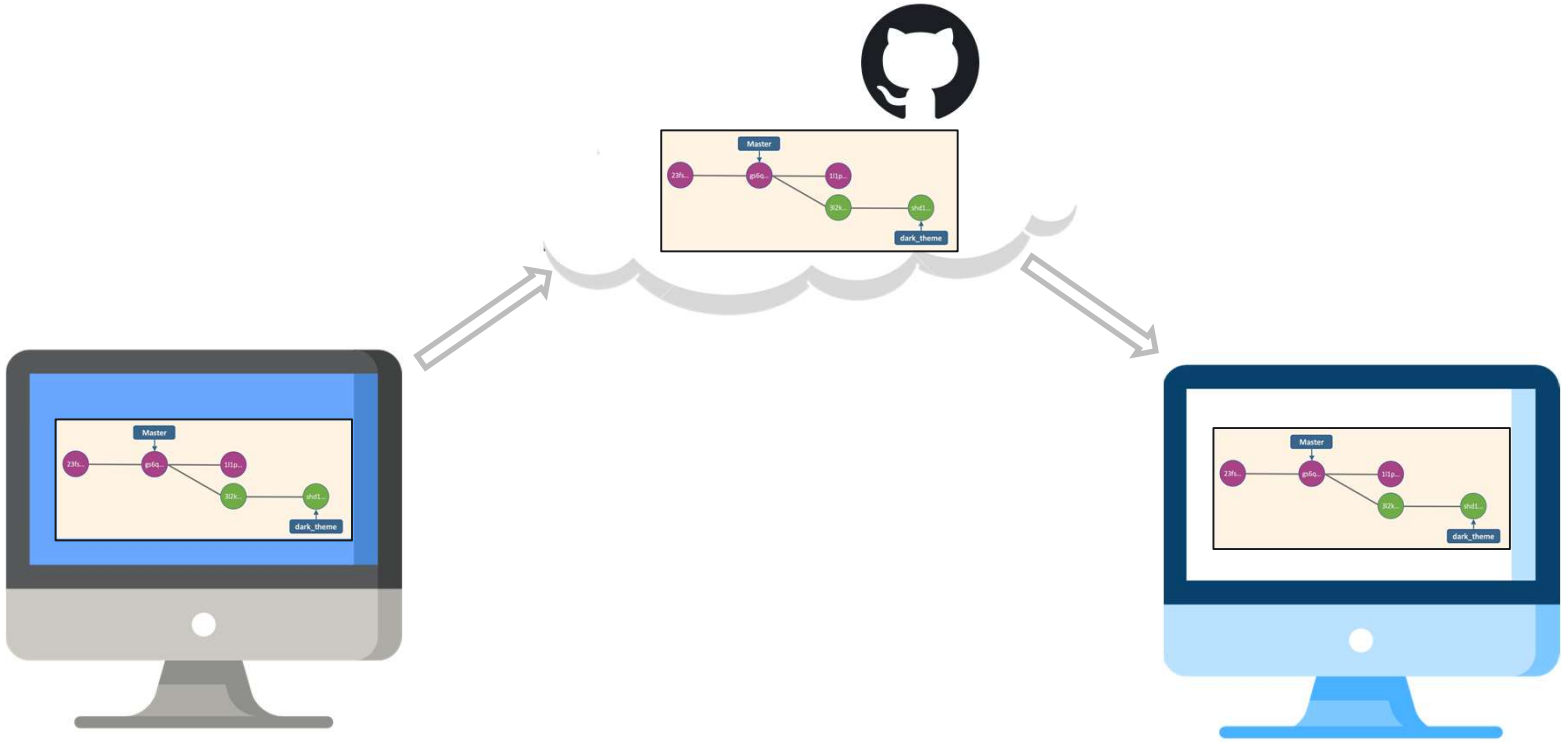
Do I have to remember them all?

- **Important**
  - git status
  - git init
  - git add
  - git commit
  - git log
- **Nice To Have (ask ChatGPT)**
  - git stash
  - git checkout
  - git restore
  - git reset

2. Connection with Github



Using Github, you can...



SSH Config Setup

- You have to set up SSH config to connect Github from your terminal / git bash(once a computer)

https://docs.github.com/en/authentication/connecting-to-github-with-ssh

Connecting to GitHub with SSH

You can connect to GitHub using the Secure Shell Protocol (SSH), which provides a secure channel over an unsecured network.

About SSH
Checking for existing SSH keys
Generating a new SSH key and adding it to the ssh-agent
Adding a new SSH key to your GitHub account
Testing your SSH connection
Working with SSH key passphrases

Cloning

- use git clone to get a local copy of an existing repository (all of the files and the commit history will be copied to your local machine)

```
> git clone <repo-url>
```

How to put my code on Github?

Option 1: Existing Repo

If you already have an existing repo locally that you want to put on GitHub…

1. Create a new repo on Github
2. Connect your local repo (add a remote)
3. Push your changes to Github

```
> git remote add <name (usually origin)> <url>
> git push <remote-name> <branch>
```

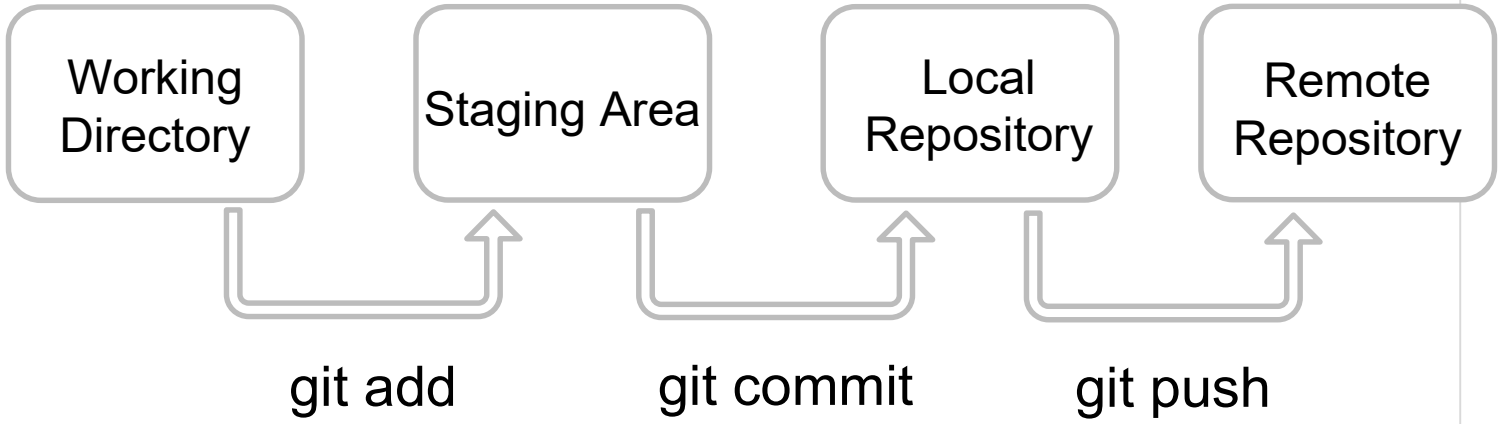
How to put my code on Github?

Option 2: Start From Scratch

If you haven’t begun work on your repo, you can…

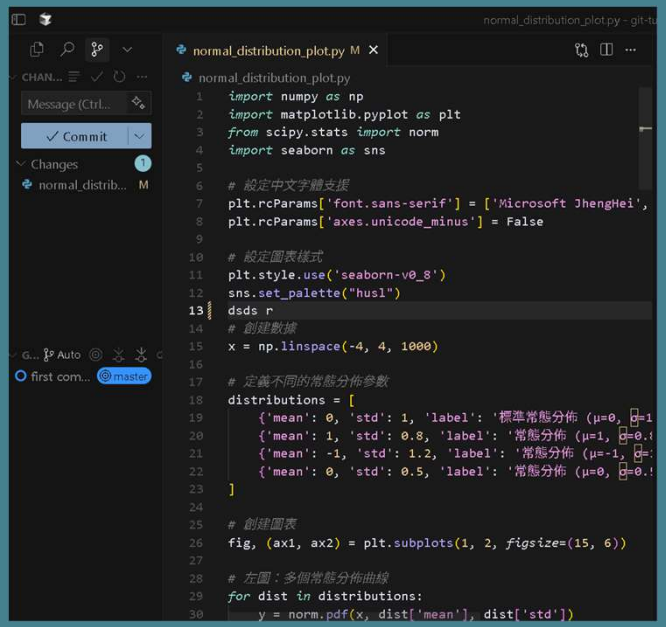
1. Create a branch new repo on Github
2. Clone it down to your machine
3. Do some work locally
4. Push up your changes to Github

The workflow



Back to Vibe Coding

Commits every time AI done something!



https://youtu.be/2Irh_GHlxiE?si=joP06_tkqt0YLLRI

Thanks for listening!