Git and GitHub usage

Instructor: Prof. Jiaji Wang & Prof. Xuguang Wang

Brief introduction:

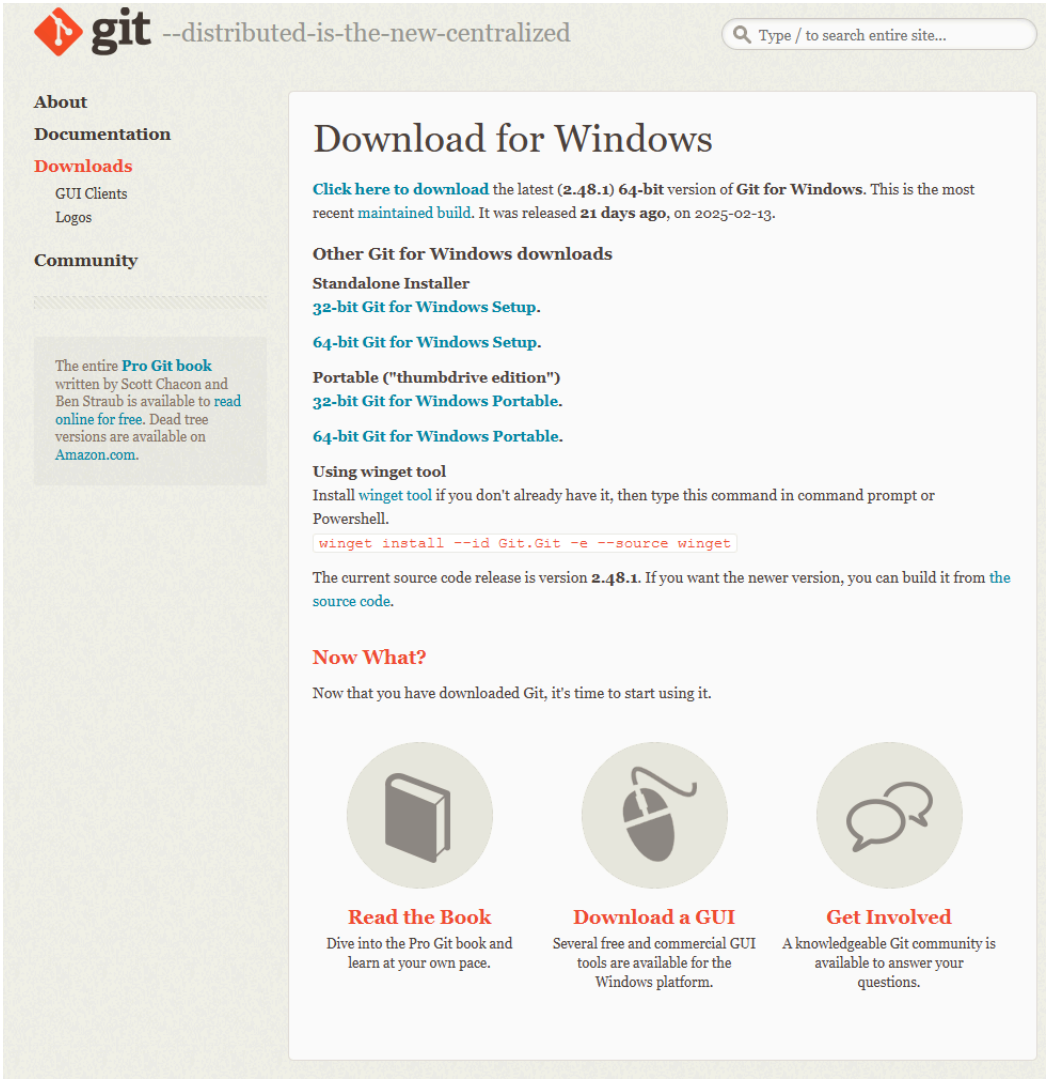
GitHub is a web-based platform that utilizes Git, a powerful version control system. It serves as a central hub for software development projects, offering a collaborative environment for developers worldwide.

- GitHub uses Git to track changes in code over time. Developers can create different versions of their projects, revert to previous states if needed, and understand the evolution of the codebase.
- Multiple developers can work on the same project simultaneously. They can contribute code, review each other's changes, and resolve conflicts through pull requests and code reviews.
- It is a vibrant community for open-source projects. Many popular open-source software, such as Linux kernel, TensorFlow, and React, are hosted on GitHub. Users can discover, contribute to, and learn from these projects.

Git

To use GitHub on your personal computer, it is necessary to install git at first.

1. Download the Git installer from official Git website: https://git-scm.com/download/win
   Pick the right version to download first. Then you need to install the downloaded git software.

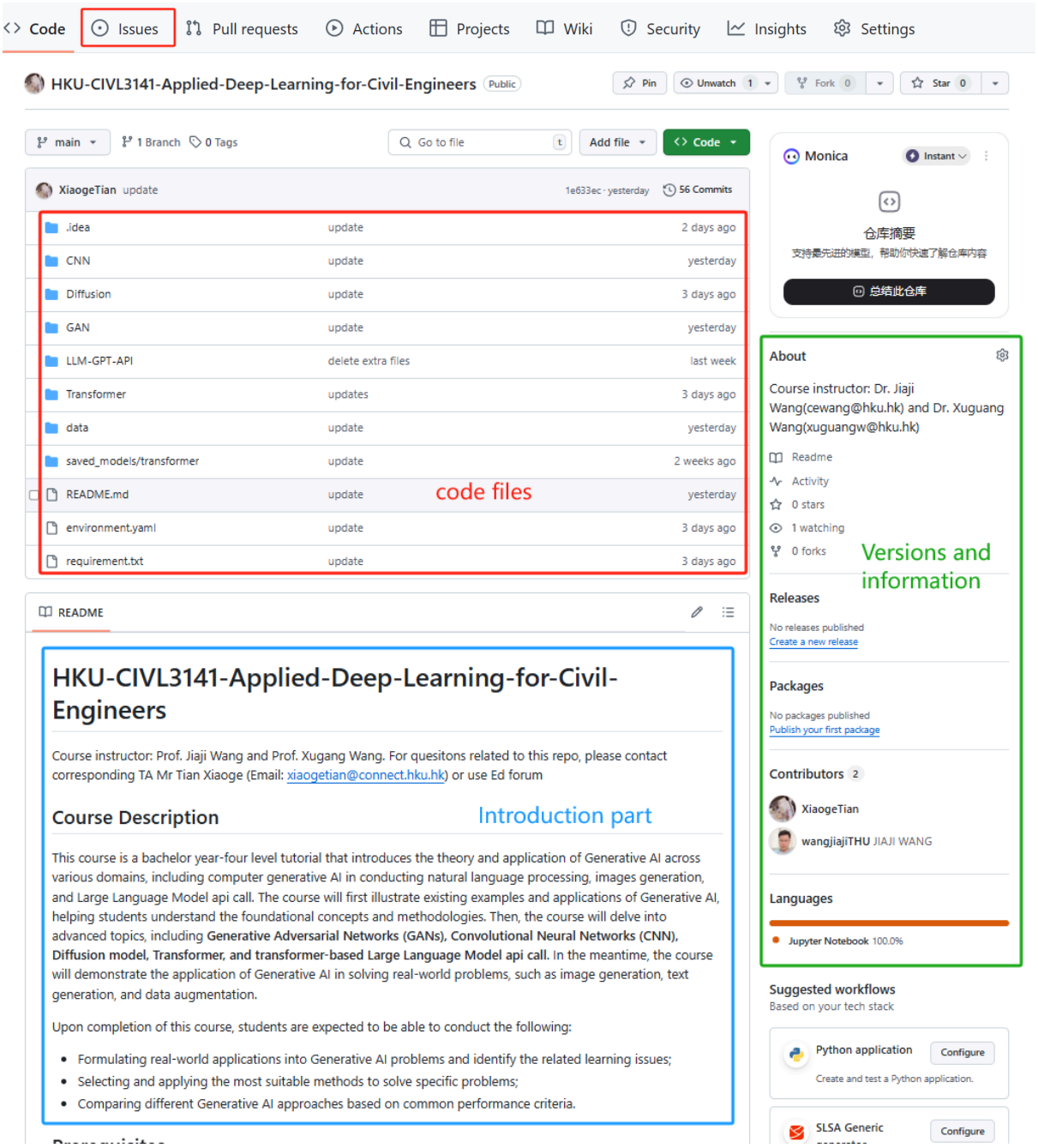


Install git by following the guideline of the installer:

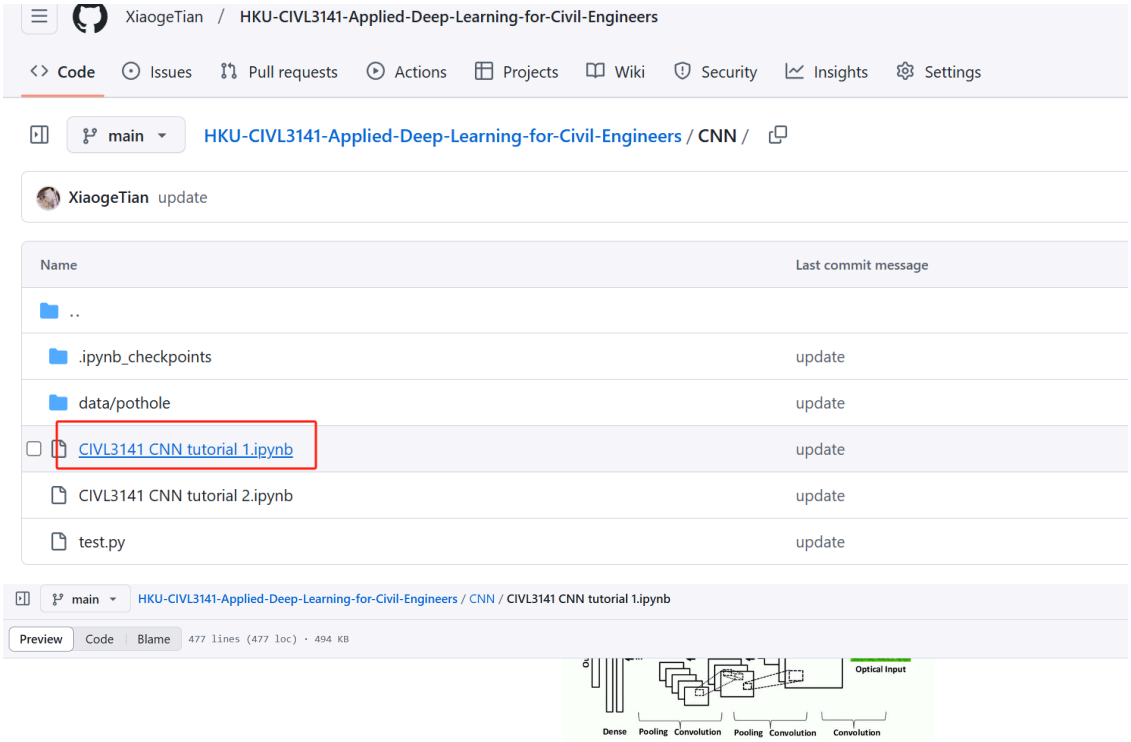
For macOS: You can use Homebrew (if installed) to install Git by running **brew install git** in the Terminal. Alternatively, download the installer from the official website.
For Linux: On Ubuntu or Debian, use sudo apt-get install git. On Fedora, use **sudo dnf install git**.

GitHub

To use GitHub on your personal computer, you need to first know the GitHub website concept. Students can visit the official github website and explore the various open-sourced code for multiple usage:
https://github.com/
You will need to first register an account by using your email address or google account. Later, we can explain more about how to use GitHub on your PC.
Let's take the repo in this course as an example, the website is as follows:
https://github.com/XiaogeTian/HKU-CIVL3141-Applied-Deep-Learning-for-Civil-Engineers
Once you click the repo, you can enter the GitHub page of this course. Same as other GitHub repo, this website is composed of several key components including the code files, readme introduction, brief project information, and issues, etc.
You can check issues posted by other users to know if there are other users who have difficulties in dealing with parts of the repo, so that you can avoid those unnecessary steps.

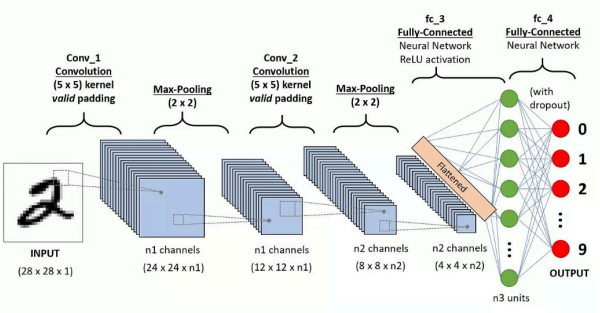


Besides, you can click the code files directly to see the detailed code content such as in our course GitHub repo, you can click and open the CNN/GAN/Diffusion/Transformer/LLM-GPT-API modules to see the detailed notebook files.



The key parts of CNNs are listed as these four parts as shown below:

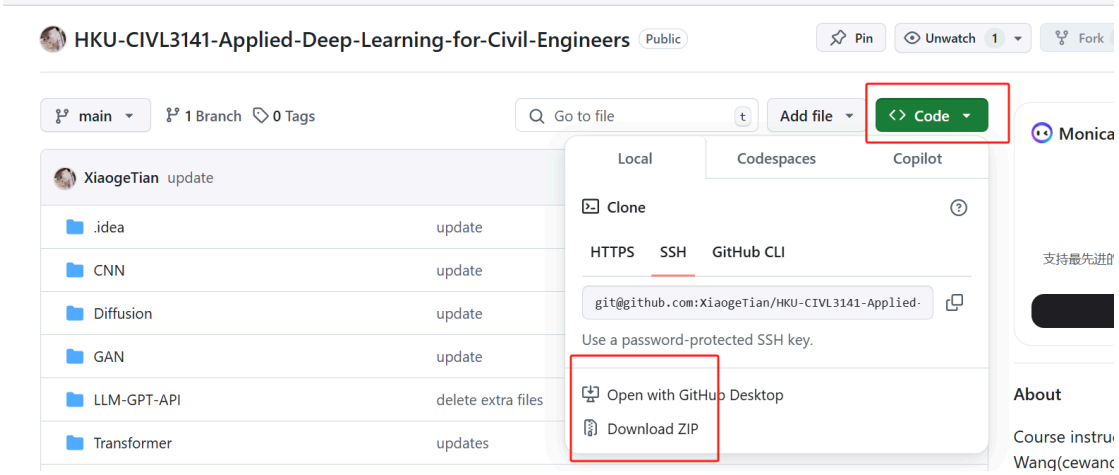
- convolutional layers
- Nonlinear activation functions
- Pooling layers
- Fully connected neural networks



Download the repo

You can download the repo in two methods.

1. You can click the green code button and download directly by downloading zip file.



Or you can try to use Git software that you just installed to connect your local PC with the remote git repo you created.

2. Using Git Locally

Before using your git to connect your remote repo, it is necessary to connect the ssh-key to your github account.

2.1 Why Use SSH Keys with GitHub?

- Enhanced Security: SSH keys provide a more secure way to authenticate with GitHub compared to using passwords. Passwords can be vulnerable to brute - force attacks, while SSH keys use strong cryptographic algorithms.
- Convenience: Once set up, you can access your GitHub repositories without having to enter your username and password every time you perform a git push or git pull operation.

2.2 Generating an SSH Key Pair

2.2.1 On Linux or macOS

Open the Terminal and run the following command:

ssh-keygen -t rsa -b 4096 -C "your_email@example.com"

- -t rsa: Specifies the type of key to generate, in this case, RSA.
- -b 4096: Sets the number of bits in the key to 4096, which provides a high level of security.
- -C "your_email@example.com": Adds a comment to the key, usually your email address, which helps you identify the key later.

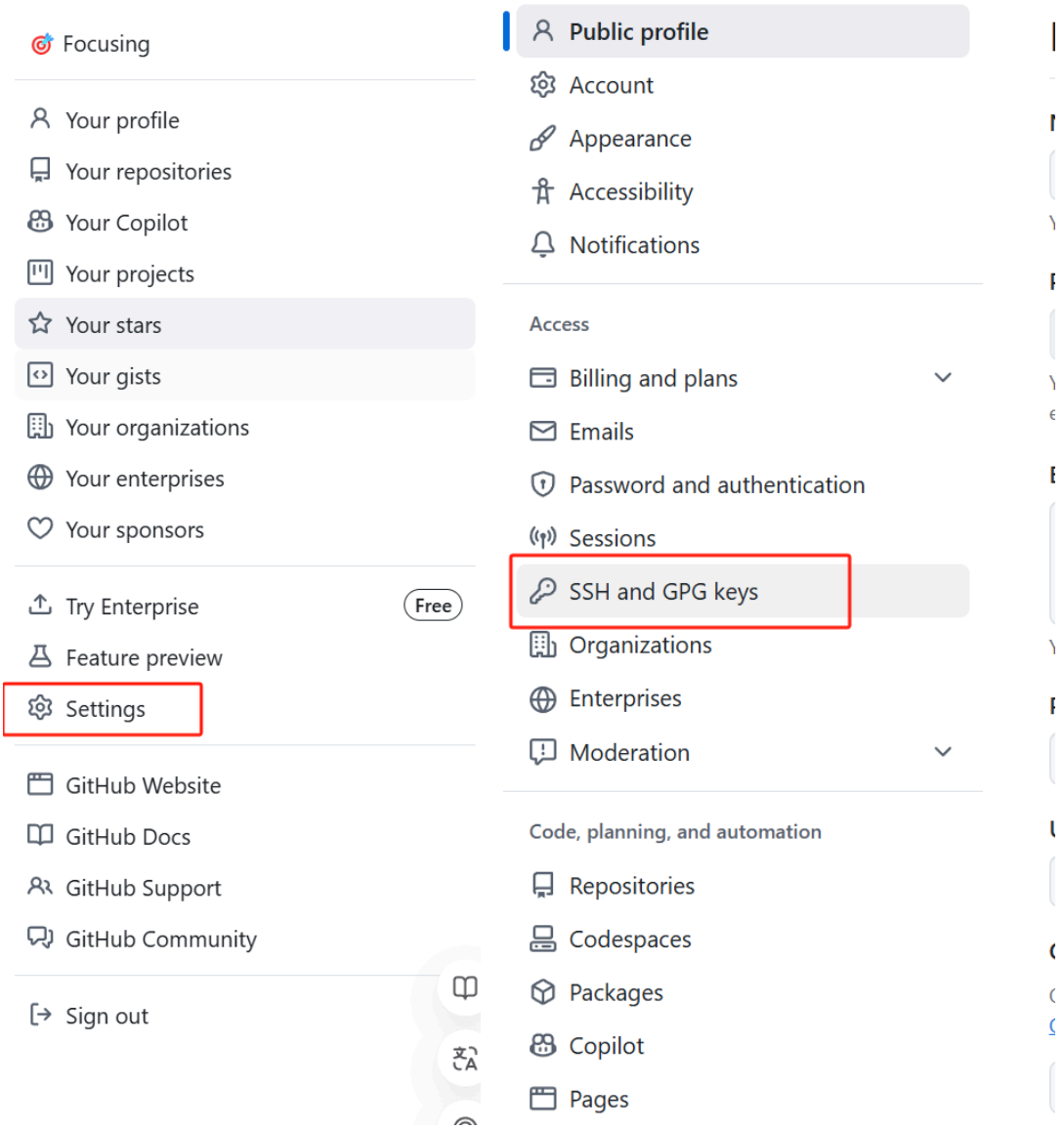
When prompted, you can choose a location to save the key (the default is usually ~/.ssh/id_rsa). You can also set an optional passphrase for an extra layer of security.

2.2.2 On Windows

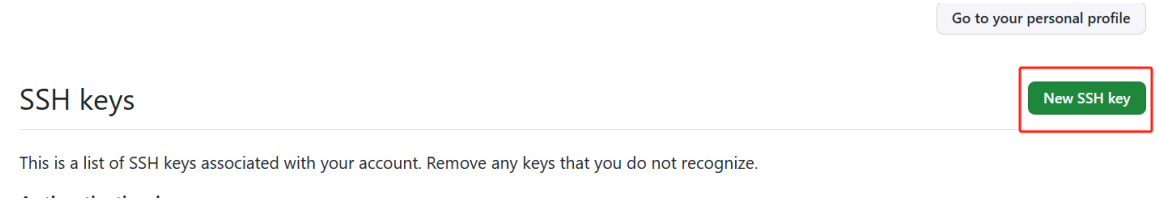
You can use Git Bash (which comes with Git for Windows) to generate SSH keys. The process is the same as on Linux or macOS. Open Git Bash and run the ssh-keygen command as described above.

2.2.3 Connect to your GitHub account

First, open your personal page and click setting. Click **SSH and GPG keys**.



Open the ssh-key you generated previously and click the button **New SSH key.** Then paste your public ssh-key information. You should be able to connect to your GitHub right now. If you have any questions about this part, you can also watch tutorial videos: https://www.youtube.com/watch?v=Wx7WPDnwcDg



You can use Git Bash (which comes with Git for Windows) to generate SSH keys. The process is the same as on Linux or macOS. Open Git Bash and run the ssh-keygen command as described above.

2.3 Configure Git

Open your terminal or command prompt and set your username and email associated with your Git commits:

```
git config --global user.name "Your Name"
git config --global user.email "your_email@example.com"
```

2.4 Initialize a Local Repository

If you have a project on your local machine that you want to manage with Git, navigate to the project's directory in the terminal and run:

```
git init
```

This will create a new Git repository in that directory.

2.5 Track and Commit Changes

- Add files to the staging area: When you make changes to your project files, you first need to add them to the staging area. You can add specific files:

  ```
  git add file1.txt file2.py
  ```

  Or add all changed files in the current directory:

  ```
  git add.
  ```

- Commit changes: After adding files to the staging area, you can commit them with a descriptive message:

  ```
  git commit -m "Add new feature"
  ```

2.6 Branch Management

- Create a new branch: To create a new branch for a new feature or bug - fix, use:

  ```
  git branch new_feature
  ```

- Switch to a branch: To switch to the newly created branch:

  ```
  git checkout new_feature
  ```

  You can also create and switch to a new branch in one command:

git checkout -b new_feature

- Merge branches: When you're done with your work on a branch and want to integrate it into the main branch (usually named main or master), first switch back to the main branch:

  git checkout main

  Then merge the new_feature branch:

  git merge new_feature

3. Connecting Local Git Repository to GitHub

3.1 Create a New Repository on GitHub

- Log in to your GitHub account.
- Click the + icon in the top - right corner and select New repository.
- Fill in the repository name, description (optional), and choose other settings like public or private. Then click Create repository.

3.2 Link Local and Remote Repositories

- On the newly created GitHub repository page, you'll see a URL (either HTTPS or SSH). Copy the URL.
- In your local terminal, in the project directory, run the following command to link your local repository to the remote one:

  git remote add origin <repository_url>

  For example, if using HTTPS:

  git remote add origin https://github.com/your_username/your_repository.git

3.3 Push Local Commits to GitHub

After linking the repositories, you can push your local commits to the remote repository:

git push -u origin main

The -u option sets up the upstream so that in the future, you can simply run git push to push changes to the same branch on the remote.

4. Collaborating on GitHub

4.1 Forking a Repository

- If you want to contribute to someone else's project on GitHub, you can fork their repository. On the repository page, click the Fork button in the top - right corner. This will create a copy of the repository in your own GitHub account.
- Clone the forked repository to your local machine:

  git clone https://github.com/your_username/forked_repository.git

4.2 Pull Requests

- After making changes to the forked repository on your local machine, push the changes to your forked repository on GitHub:

  git push origin your_branch

- Then, on the GitHub page of your forked repository, click the New pull request button. Compare the changes between your branch and the original repository's target branch (usually main). Write a descriptive message about your changes and submit the pull request. The owner of the original repository can then review and merge your changes.