

CPE 393 Machine Learning Operations

Lab-3 Git Tutorial with GitHub

Installation

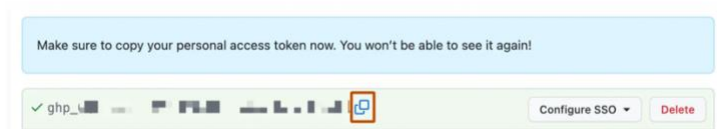
- For Windows: Download and install from git-scm.com.
- For macOS/Linux: Use the package manager (e.g., brew install git or sudo apt install git)
- **Check Git Version** ~ git --version

Account Creation

- Sign up/in your GitHub account
- Confirm your email if you haven't done yet for verification

Create Personal Access Token (Keep Safe)

- 1 [Verify your email address](#), if it hasn't been verified yet.
- 2 In the upper-right corner of any page on GitHub, click your profile photo, then click ⚙️ **Settings**.
- 3 In the left sidebar, click <> **Developer settings**.
- 4 In the left sidebar, under 🔑 **Personal access tokens**, click **Tokens (classic)**.
- 5 Select **Generate new token**, then click **Generate new token (classic)**.
- 6 In the "Note" field, give your token a descriptive name.
- 7 To give your token an expiration, select **Expiration**, then choose a default option or click **Custom** to enter a date.
- 8 Select the scopes you'd like to grant this token. To use your token to access repositories from the command line, select **repo**. A token with no assigned scopes can only access public information. For more information, see [Scopes for OAuth apps](#).
- 9 Click **Generate token**.
- 10 Optionally, to copy the new token to your clipboard, click 📋.



- 11 To use your token to access resources owned by an organization that uses SAML single sign-on, authorize the token. For more information, see [Authorizing a personal access token for use with SAML single sign-on](#) in the GitHub Enterprise Cloud documentation.

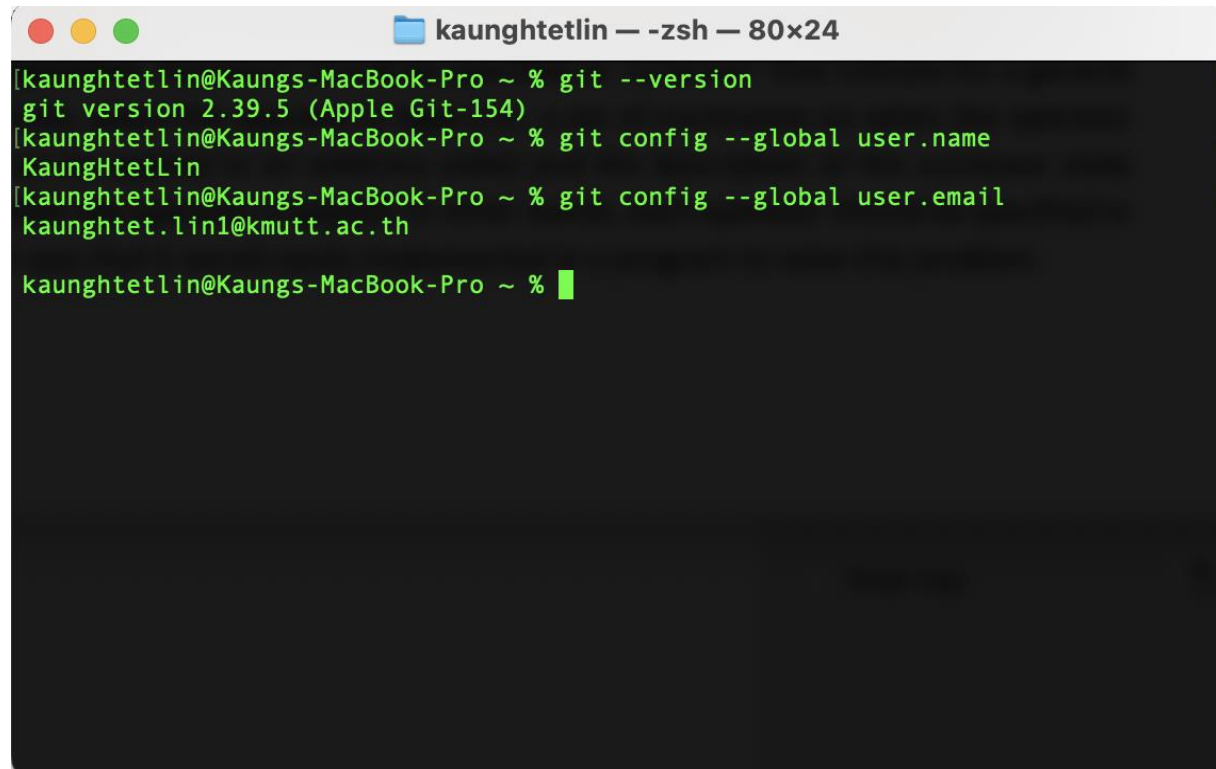
Assignment 1: Practicing Basic Git commands with command line interface

Submission: GitHub URL of your repository

Configuration

`git config --global user.name "Your Name"`

`git config --global user.email "your.email@example.com"`



```
kaunghtetlin — -zsh — 80x24
[kaunghtetlin@Kaungs-MacBook-Pro ~ % git --version
git version 2.39.5 (Apple Git-154)
[kaunghtetlin@Kaungs-MacBook-Pro ~ % git config --global user.name
KaungHtetLin
[kaunghtetlin@Kaungs-MacBook-Pro ~ % git config --global user.email
kaunghtet.lin1@kmutt.ac.th

kaunghtetlin@Kaungs-MacBook-Pro ~ % █
```

Part 1: Basic Git Workflow

1. Create a new repository for your project on GitHub website.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Required fields are marked with an asterisk (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *

 ayehninnkhine ▾

Repository name *

git-practice

✔ git-practice is available.

Great repository names are short and memorable. Need inspiration? How about **solid-potato** ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

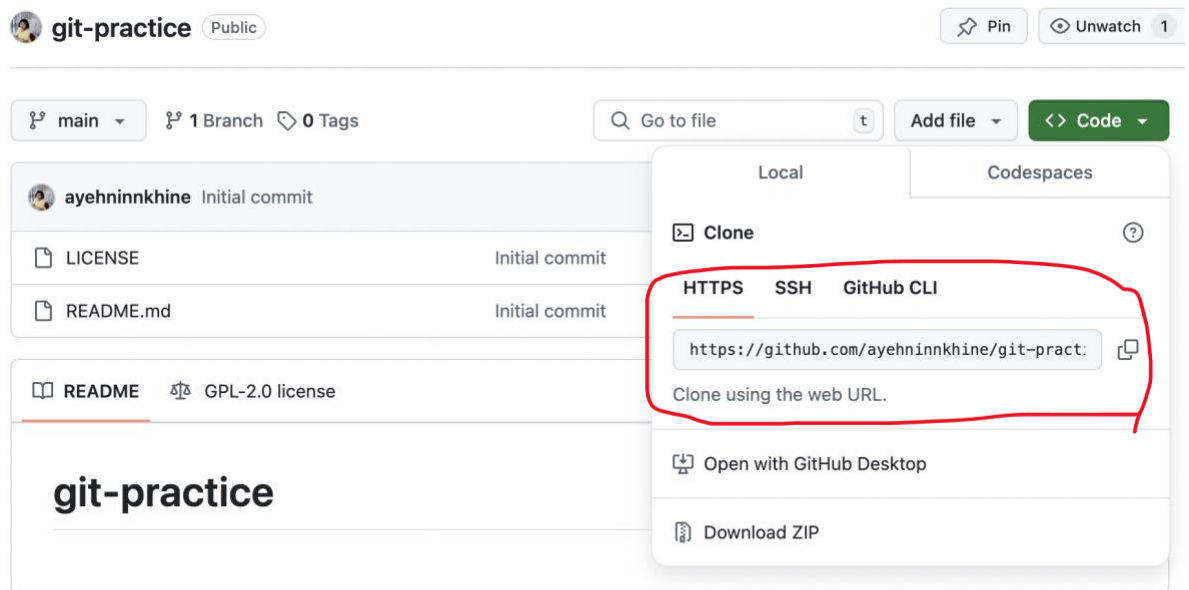
Choose a license

License: GNU General Public License v2.0 ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

2. Clone a Git repository to local machine



Command ~ git clone <https://github.com/ayehnnnkhine/git-practice.git>

After cloning the repository, you should see a folder on your local machine. Go to your cloned repository location in terminal.

Command ~ cd git-practice

3. Create a file called project_plan.txt
4. Add the initial content to the file
5. Stage and commit the file with a descriptive message
6. Make additional changes to the file
7. Stage and commit the new changes
8. View the commit history using git log

Create project plan file

```
echo "Project Outline: Machine Learning" > project_plan.txt
```

Stage and commit initial file

```
git add project_plan.txt
```

```
git commit -m "Initialize project plan document"
```

Make additional changes

```
echo "Added initial project goals and scope" >> project_plan.txt
```

Stage and commit changes

```
git add project_plan.txt
```

```
git commit -m "Update project plan with initial goals"
```

View commit history

```
git log
```

Part 2: Branching and Merging

1. Create a new branch called feature-branch
2. Switch to the new branch
3. Create a new file called feature_details.txt
4. Add content to the file
5. Stage and commit the changes
6. Switch back to the main branch
7. Merge the feature branch into main
8. Resolve any merge conflicts if they occur

Create and switch to feature branch

```
git checkout -b feature-branch
```

Create feature details file

```
echo "Feature 1: User Authentication System" > feature_details.txt
```

```
git add feature_details.txt
```

```
git commit -m "Add initial feature details for authentication"
```

Switch back to main branch and merge

```
git checkout main
```

```
git merge feature-branch
```

Part 3: Commit Message Practice

1. Make a series of small, incremental changes
2. Write descriptive, meaningful commit messages for each change
3. Use git commit --amend to modify the most recent commit message

Make incremental changes

```
echo "Updated authentication requirements" >> feature_details.txt
```

```
git add feature_details.txt
```

```
git commit -m "Refine authentication feature requirements"
```

```
# Amend last commit message
```

```
git commit --amend -m "Comprehensive authentication feature requirements"
```

Part 4: Remote Repository Simulation

1. Create a new repository on GitHub
2. Clone the repository to your local machine
3. Create a new branch and make changes
4. Push the branch to the remote repository
5. Create a pull request
6. Practice merging the pull request

```
# Create GitHub repository (through GitHub web interface)
```

```
git remote add origin https://github.com/yourusername/git-lab-project.git
```

```
# Create and push new branch
```

```
git checkout -b remote-feature
```

```
echo "Remote feature implementation" > remote_feature.txt
```

```
git add remote_feature.txt
```

```
git commit -m "Add remote feature implementation"
```

```
git push -u origin remote-feature (paste your personal access token when password is asked)
```

```
# Create pull request via GitHub web interface or GitHub CLI
```

```
gh pr create
```

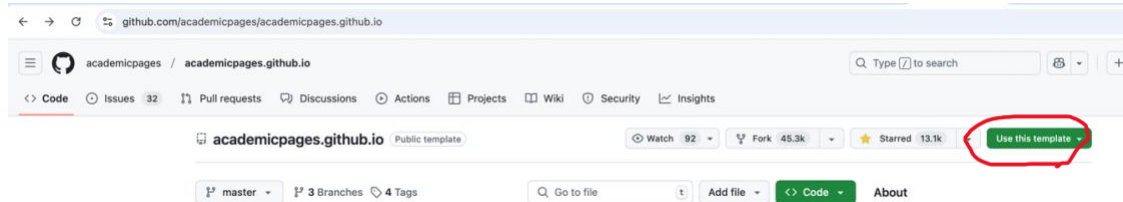
Part 5: Merging branches

You should see a PR on your repository in GitHub. Merge your pull request on your repository in web interface.

Assignment 2: Building Academic Portfolio

Go to academic pages repository

(<https://github.com/academicpages/academicpages.github.io>)



1. Click the "Use this template" button in the top right.
2. On the "New repository" page, enter your repository name as "[your GitHub username].github.io", which will also be your website's URL.
3. Set site-wide configuration and add your content.
4. Upload any files (like PDFs, .zip files, etc.) to the files/ directory. They will appear at [https://\[your GitHub username\].github.io/files/example.pdf](https://[your GitHub username].github.io/files/example.pdf).
5. Check status by going to the repository settings, in the "GitHub pages" section
6. Edit the template with your information.
7. Push the updated files to the repository.
8. Visit <https://username.github.io> to view your customized page.

Submission – Take a screenshot of your profile, include your GitHub.io URL and Git repository URL in a PDF.