**git**

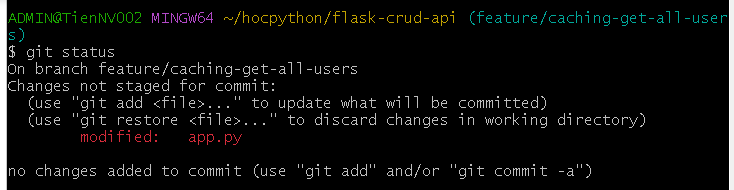
24/10/2024

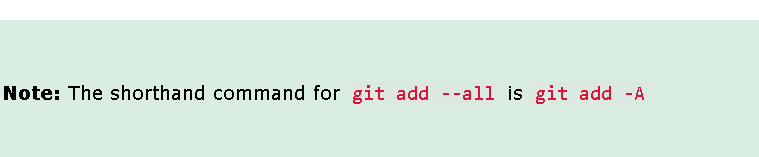
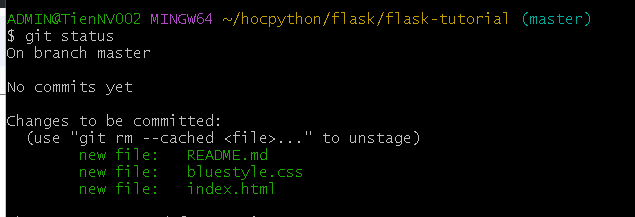
* To show how it changes : **git show *commithash***

23/10/2024

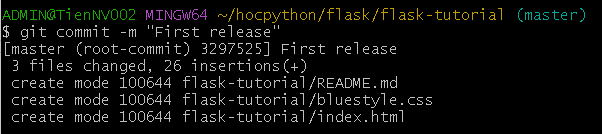
* **Git : .gitignore/**
  + **.venv/** bỏ qua các môi trường ảo (Python)
  + **# Byte-compiled / optimized / DLL files**
    - **\_\_pycache\_\_/**
    - **\*.py[cod]**
    - **\*$py.class**
  + **# C extensions**
    - **\*.so**
  + **# Distribution / packaging**
    - **.Python**
    - **env/**
    - **venv/**
    - **ENV/**
    - **env.bak/**
    - **.venv/**
    - **build/**
    - **dist/**
    - **\*.egg-info/**
    - **\*.egg**
  + **# Installer logs**
    - **pip-log.txt**
    - **pip-delete-this-directory.txt**
  + **# Jupyter Notebook**
    - **.ipynb\_checkpoints**
  + **# Python test output**
    - **.pytest\_cache/**
    - **.coverage**
    - **.coverage.\***
    - **nosetests.xml**
    - **coverage.xml**
    - **\*.cover**
    - **.hypothesis/**
    - **htmlcov/**
  + **# Sphinx documentation**
    - **docs/\_build/**
    - **build/**
    - **\*.log**
  + **# Pyre type checker**
    - **.pyre/**
  + **# mypy**
    - **.mypy\_cache/**
  + **# PyCharm**
    - **.idea/**
    - **\*.iml**
  + **# VSCode**
  + **.vscode/**
  + **# Sublime Text**
    - **\*.sublime-project**
    - **\*.sublime-workspace**
  + **# Rope**
    - **.ropeproject/**
  + **# MacOS system files**
    - **.DS\_Store**
  + **# Windows system files**
    - **Thumbs.db**
    - **ehthumbs.db**
  + **# Linux trash folder**
    - **.trash/**
  + **# Environment variable files**
    - **.env**
    - **.env.local**
    - **.env.\*.local**
  + **# Docker**
    - **\*.tar**
    - **\*.log**
    - **docker-compose.override.yml**
  + **# Others**
    - **\*.pkl # Pickle files**
    - **\*.dat # Data files**
    - **\*.csv # CSV files (if you don't want to track them)**

22/10/2024-24/10/2024

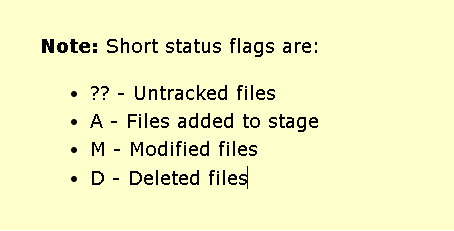
* Git : <https://www.w3schools.com/git/default.asp?remote=github>
  + Chức năng:
    - Quản lý dự án với **Repositories.**
    - **Clone** 1 dự án để làm việc trên bản sao cục bộ.
    - Kiểm soát, theo dõi các thay đổi với **Staging** và **Committing.**
    - **Branch + Merge** để cho phép làm việc trên các phần, phiên bản khác nhau của dự án.
  + Làm việc:
    - Initialize(khởi tạo) Git on a folder, making it a **Repository**
    - Git now creates a hidden folder to keep track(theo dõi) of changes in that folder
    - When a file is changed, added or deleted, it is considered **modified(đã thay đổi)**
    - You select the modified files you want to **Stage**
    - The **Staged** files are **Committed**, which prompts(nhắc nhở) Git to store a permanent snapshot of the files
    - Git allows you to see the full history of every commit.
    - You can revert back to any previous commit.
    - Git does not store(lưu trữ) a separate copy of every file in every commit, but keeps track of changes made in each commit!
  + Using:
    - Version: git -- version
    - Config: git config --global user.name “...” // git config --global user.email “ …”
    - Create folder: mkdir … => cd …
    - Initialize: git init
    - Showing list of folder: ls 
    - Check status: git status 
      * có 2 trạng thái:
        + Tracked: files that git knows about and are added to repository
        + Untracked: files that are in ur working directory but not adđe to repository
      * When you first add files to an empty repository, they are all untracked. To get Git to track them, you need to stage them, or add them to the staging environment.
    - Staging Environment:
      * Add a file to **Staging Environment** : **git add …**



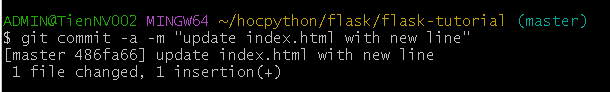
* + - Commit: git commit -m “...”



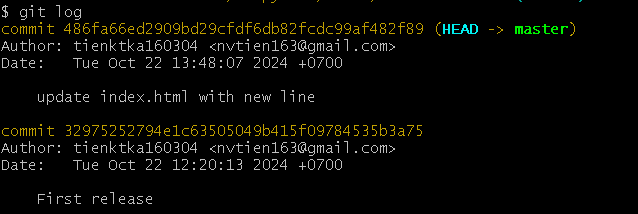
* + - * **commit** performs a commit
      * **-m “*message*”** adds a message
      * **-a option** auto stage every changed, already tracked file
      * **git status --short** : to see the changes in a more compact(nho gon) way.



* + - * **git commit -a -m “...”** => to commit the modified file



* + - * To view the history of commits: **git log**



* + - Git Help: *git command -help* list help viết tắt

**

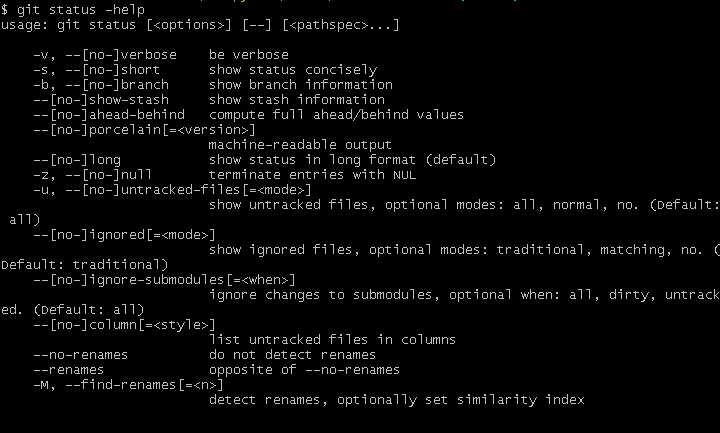
* + - **git –help**: to open the relevant Git manual page



* + - **git help --all:** a very long list of commands.



* + - Show possible options for the status command:**git status -help**



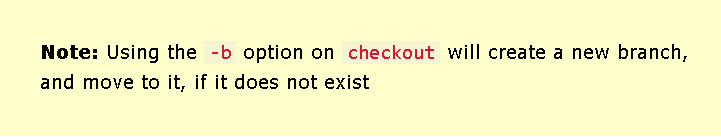
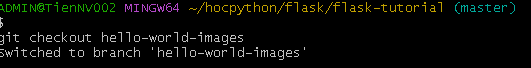
* + - Git Branch : *branch* is a new/separate(riêng biệt) version of the main repo
      * Allow to work on different parts of a project without impacting the main branch.
      * Create new git branch: **git branch hello-world-**



* + - * Confirm that we have created a new ***branch*** : **git branch**



* + - * + \* beside master specifies that we are currently on that ***branch***
      * **Checkout**: To move us from current ***branch*** to the one specified at the end of the command: git checkout hello-world-images



* + - * **Switching between branches**:
        + **ls** : to list the files in the current directory



“img\_hello-world.jpg” is only in “hello-world-images” **branch**

* + To both create and move on the new branch: git checkout -b ….
    - Create a new branch to deal with the emergency: **git checkout -b emergency-fix**



* + - * To fix an emergency: example : We will add 1 line code on index.html and git add index.html -> **git commit -m “...”**
* Git Branch Merge:
  + Change to the master branch first: **git checkout master**
  + Merge the current branch with emergency-fix**: git merge emergency-fix**



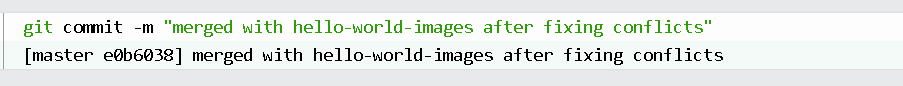
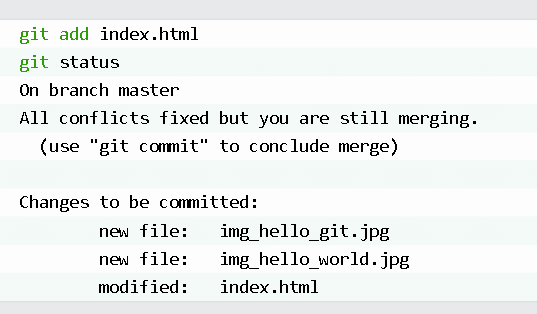
* + \* To delete a branch that is no longer needed: **git branch -d ….**



* + **Merge Conflict** (xung đột):



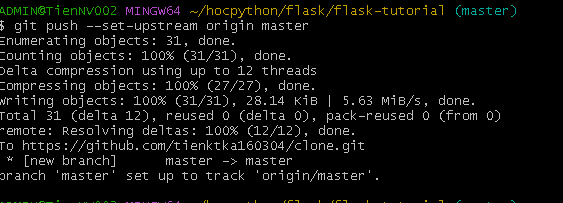
* + After fix in index.html:



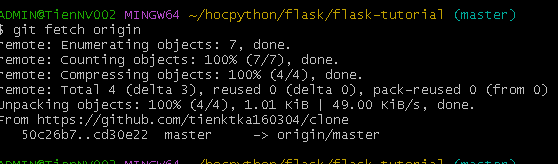
* Git GitHub Getting Started:
  + To add a remote repository, with the specified URL, as an origin :



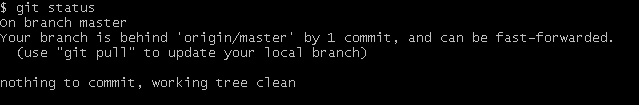
* + Push out  ***Master*** branch to the origin url, set it as the default remote branch:**git push --set-upstream origin master**

****

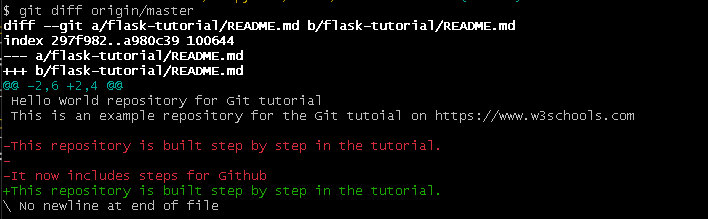
* Git Github Edit code:
  + **Pull:** got 2 different commands:
    - **fetch** : gets all the change history of a track branch/repo
      * **git fetch origin** :



=> after this , check status: **git status**



=> Double check by viewing the log : **git log origin/master**

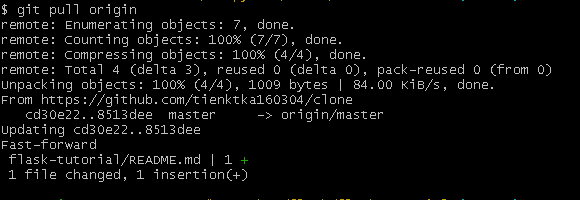
=> Showing the differences between local **master** and **origin/master**: 

* Git Merge :
  + **merge : combines the current branch, with a specified branch**
    - **Confirmed that the updates are as expected, -> merge the current branch(master) with origin/master:**

****

* Git Pull :
  + **pull** is a combination of **fetch** and **merge**. => pull (kéo) all changes from a remote spo into the branch ur working on.

\*Make another change to Readme.md

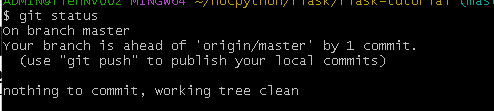
=> Use **pull** to update our local Git:

**=>**

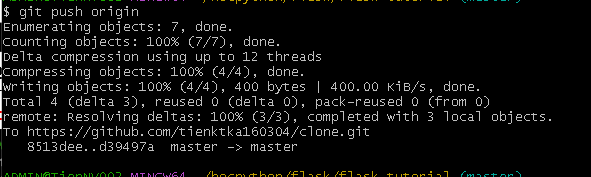
* Git Push to GitHub:
  + Make some changes to our local git and push them to github. => Commit the changes: **git commit -a -m “...”**

****

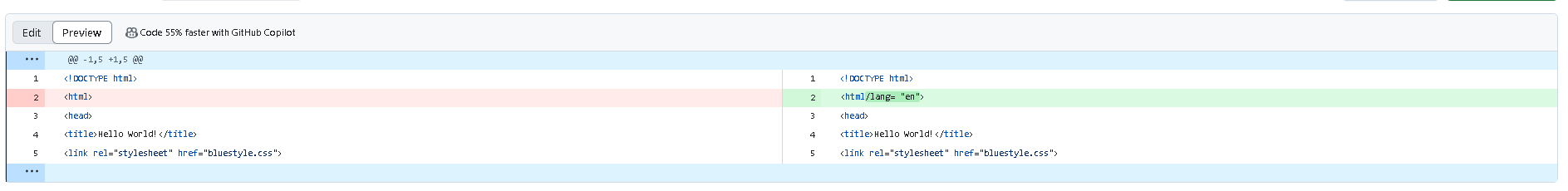
* + => check status :



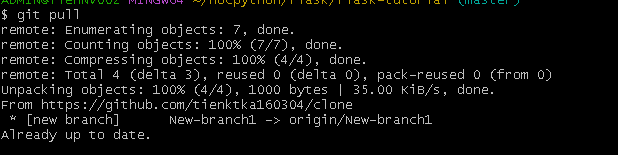
* + Now, Push our changes to our remote origin: **git push origin**

****

* Git GitHub Branch:
  + Create a new Branch on GitHub. After changing a file, we can preview changes:

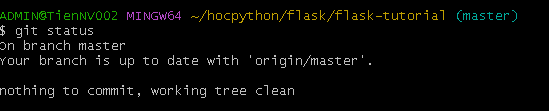


* Git Pull Branch from GitHub:
  + **pull** from out GitHub repo again so that our code is up-to-date:**git pull**

****

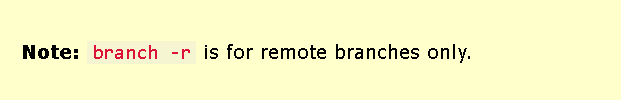
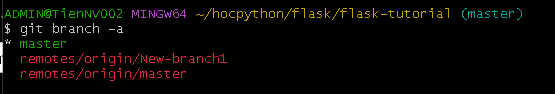
=> we can see that there is a new branch available on GitHub. (New-branch1)

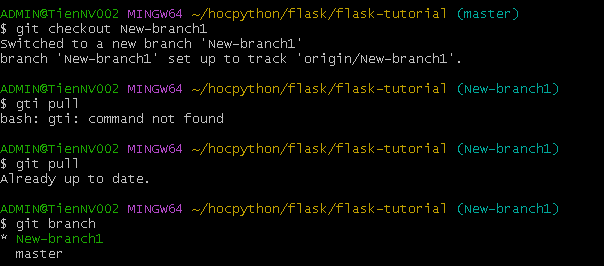
=> Do a quick status check:



=> confirm the branch we have, work at that moment: 

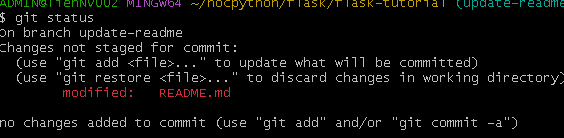
=> Use -a option to see all local and remote branches:



=> we can checkout, pull the New-branch1 branch and still work when we git **branch :** 

* Git push Branch to GitHub:
  + Start by creating a branch: **git checkout -b update-readme**

=> Add some new changes to the README.md

=> Check the status after changing:

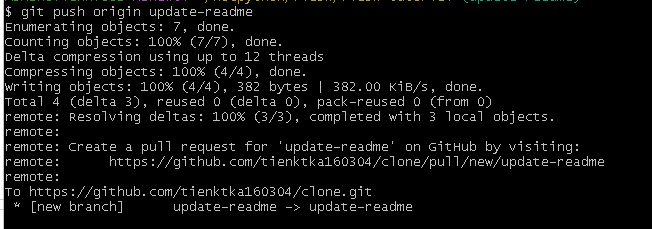
=> **git add README.md**

=> check status again

=> commit them to the branch: **git commit -m “...”**

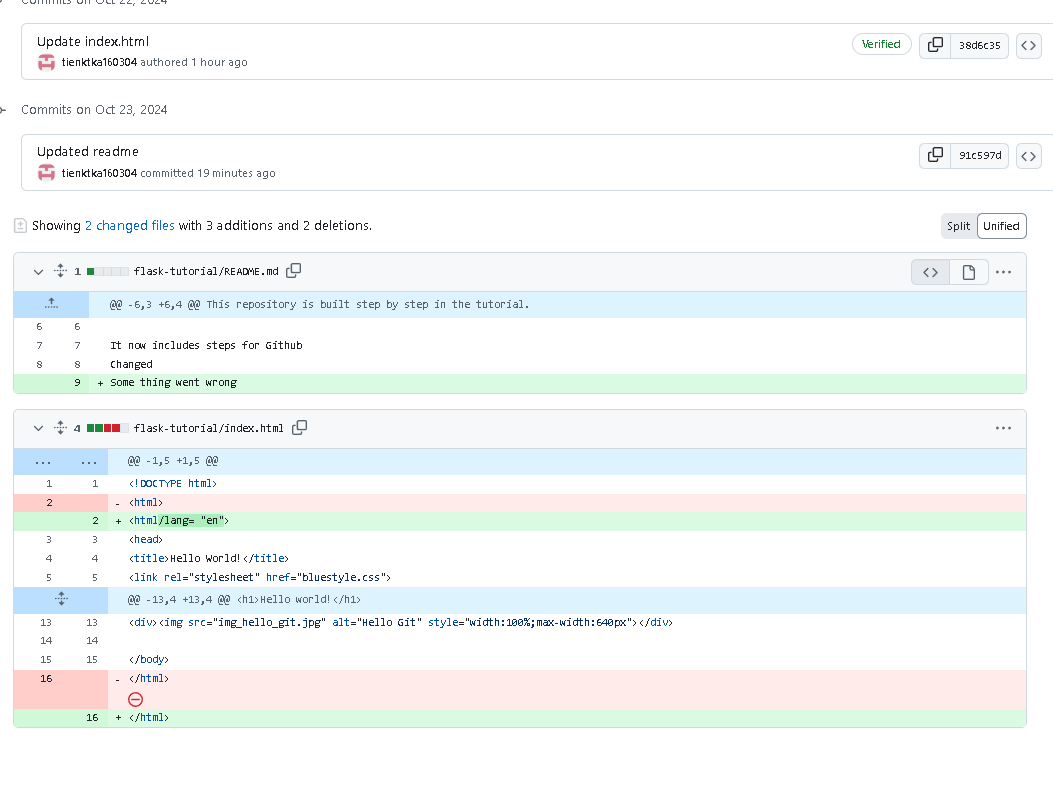


=> Push the branch from local Git repo to GitHub: **git push origin updated-readme**

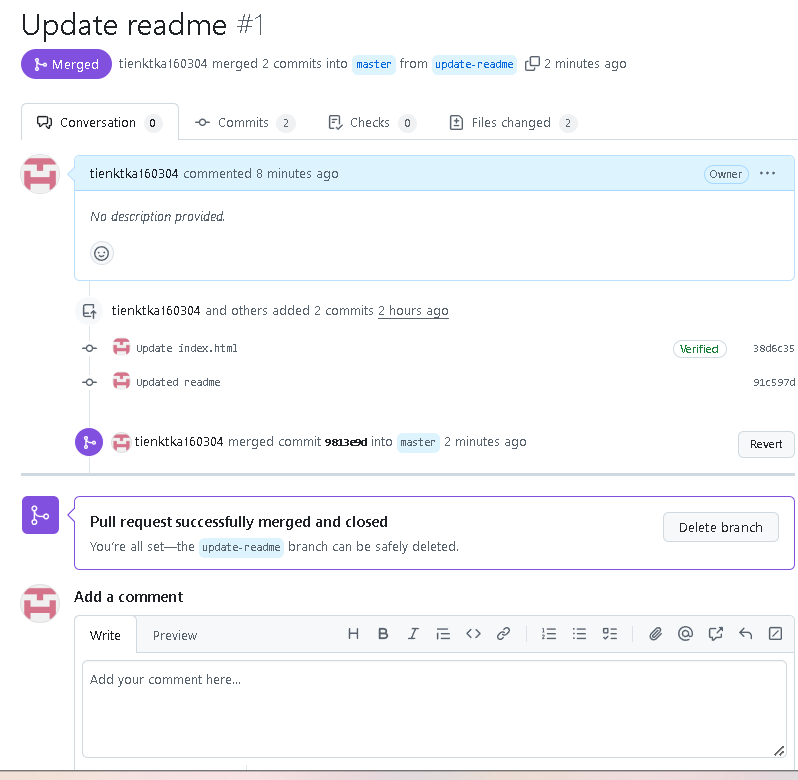


=> 3 branches in our repo now.

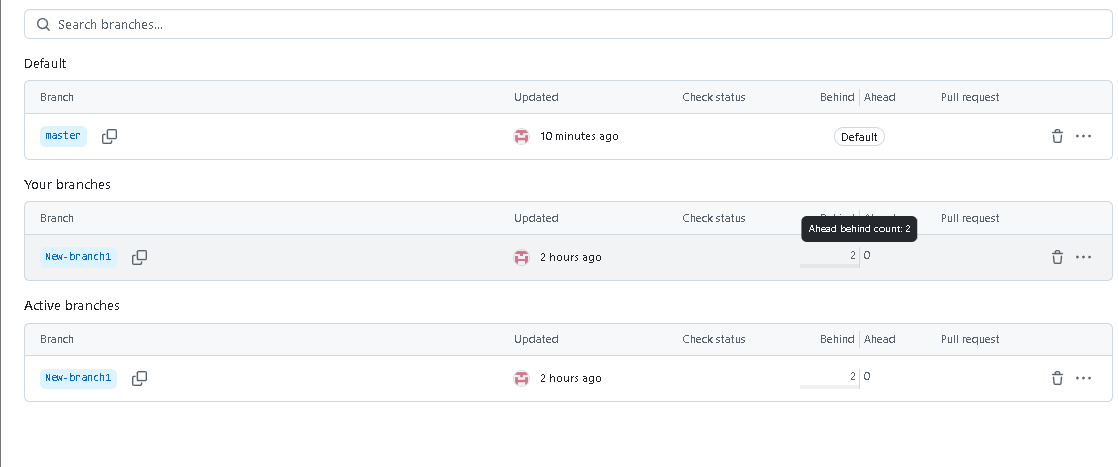
=> We can see changes and merge them into the master branch if we approve it

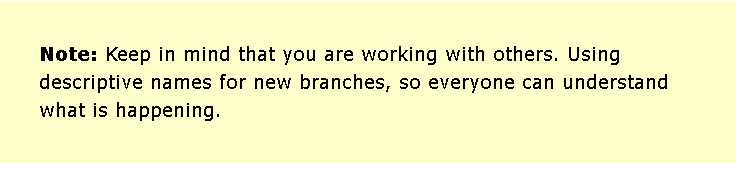
=> Press “Compare & pull request” : we can see both changes from 2 branches.

=> After ”create pull request” -> ” Merge pull request”: Pull request till record the changes -> we can go through them later to figure out the changes made.

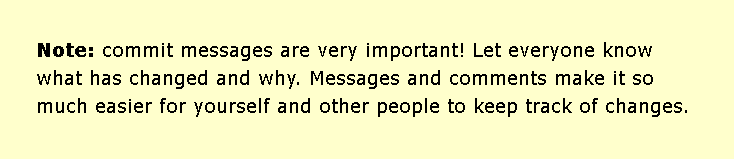
=> the result: 

=> to delete the unused branch “Delete branch”.

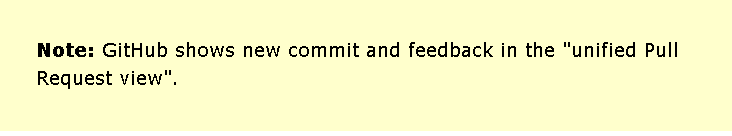
* After confirming the changes from the previous , delete that as well 
* Git GitHub Flow: How to get the best out of working
  + How it works:
    - Create a new Branch
    - Make changes and add Commits
    - Open a Pull Request
    - Review
    - Deploy(trien khai)
    - Merge
  + Create a new Branch :
    - Branch is the key concept in Git
    - The rule: **the master branch is ALWAYS deployable.**
    - Create a new branch, try something new or experiment that does not affect the main branch.
    - When the new branch is ready , it can be reviewed , discussed and merged with the main branch.



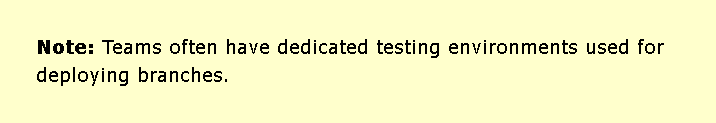
* + Make Changes and Add Commits :
    - Making by adding, editing and deleting files.
    - Adding the changes to your branch by commit when reach a small milestone(một cột mốc nhỏ).
    - Adding a commit keeps track of your work.
    - Should have a message explaining what has changed and why.
    - Each commit becomes a part of the history of the branch and a point -> can revert( quay lại ) back to if you need to.



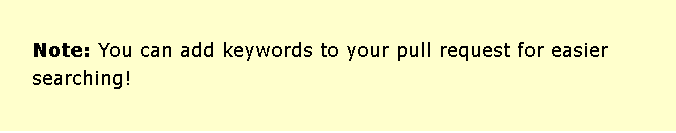
* + Open a Pull Request :
    - Pull requests are a key part of GitHub.
    - Notify people you have changes ready for them to consider and review.
    - Ask others to review your changes or pull your contribution and merge it into their branch.
  + **Review**:
    - After a Pull Request , it can be reviewed by whoever has the proper access(quyen truy cap thich hop) to the branch.
    - Allow people to work together easily and produce better results together.
    - *If you receive feedback and continue to improve your changes, you can push your changes with new commits, making further reviews possible*.



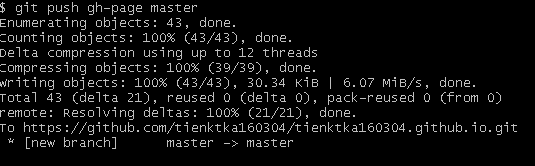
* + **Deploy** :
    - After reviewing the pull request, GitHub allows to deploy from a branch for final testing in production before merging with the master branch.
    - If any issues arise, you can undo the changes by deploying the master branch into production again.

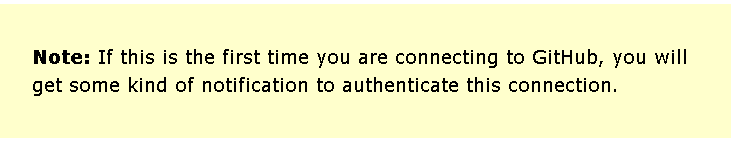


* + **Merge** :
    - After exhaustive testing( ktra toan` dien) , you can merge the code into the master branch.
    - Pull Requests keep records of changes to your code, and if you commented and named changes well, you can go back and understand why changes and decisions were made.



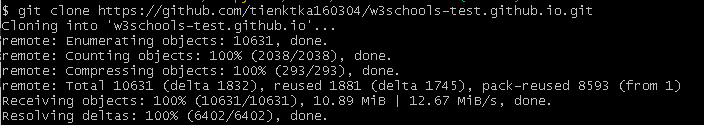
* + **Git GitHub Pages** :
    - Host Your Page on GitHub:
      * GitHub allow you to host a webpage from your repo
    - Create a New Repo:
      * Create a new repo that needs a special name to function as a GitHub page
      * It needs to be your GitHub username, followed by .github.io .
  + **Push Local Repository to Github Pages** :
    - Add the URL as a new remote: **git remote add gh-page** [**https://github.com/tienktka160304/tienktka160304.github.io.git**](https://github.com/tienktka160304/tienktka160304.github.io.git)

**=> Make sure you are on the master branch, then push the master branch to the new remote : **

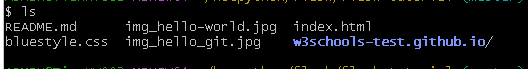
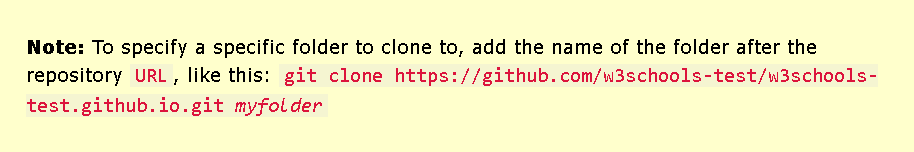
****

**\*Check if you have received all the files.**

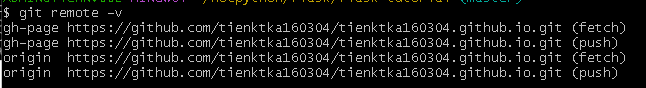
* + **Check Out Your Own GitHub Page:**
  + Git GitHub Fork :
    - **Add someone else’s repo**: At the heart of Git is collaboration.
      * Git does not allow you to add code to someone else’s repo without access rights.
    - **Fork a Repo** : a fork is a copy of a repo. It that means you will “fork” a repo from someone else’s
  + **Git clone from GitHub** :
    - **Clone a Fork from GitHub** : a clone is a full copy of a repo, including all logging and versions of files.
      * Copy the URL : **git clone** [**https://github.com/tienktka160304/w3schools-test.github.io.git**](https://github.com/tienktka160304/w3schools-test.github.io.git)

****

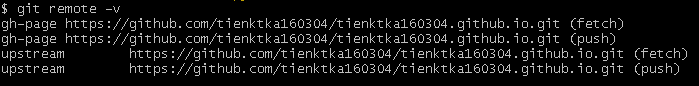
* + - * Now list ur file system: **ls**

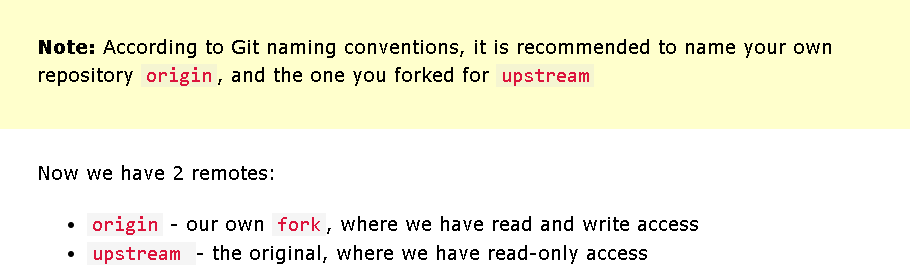
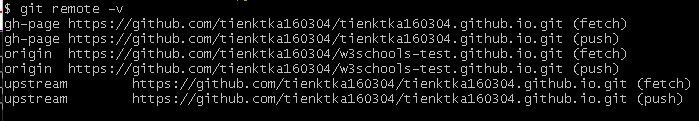
* + **Configuring Remotes** :
    - To see how the remotes of the git is set up: **git remote -v**

****

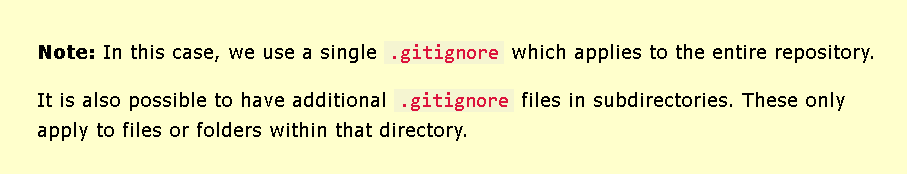
* + - To rename the original origin remote :**git remote rename origin upstream **

=> we will **git remote -v** to see after we rename: 

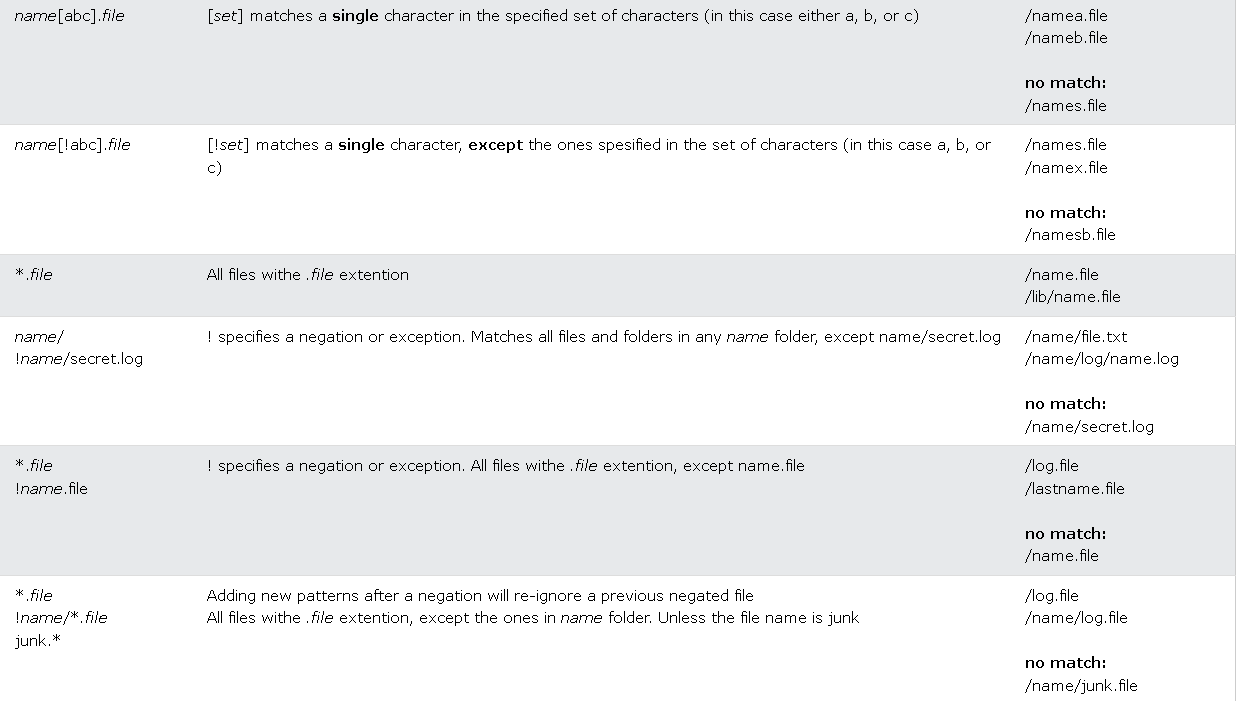
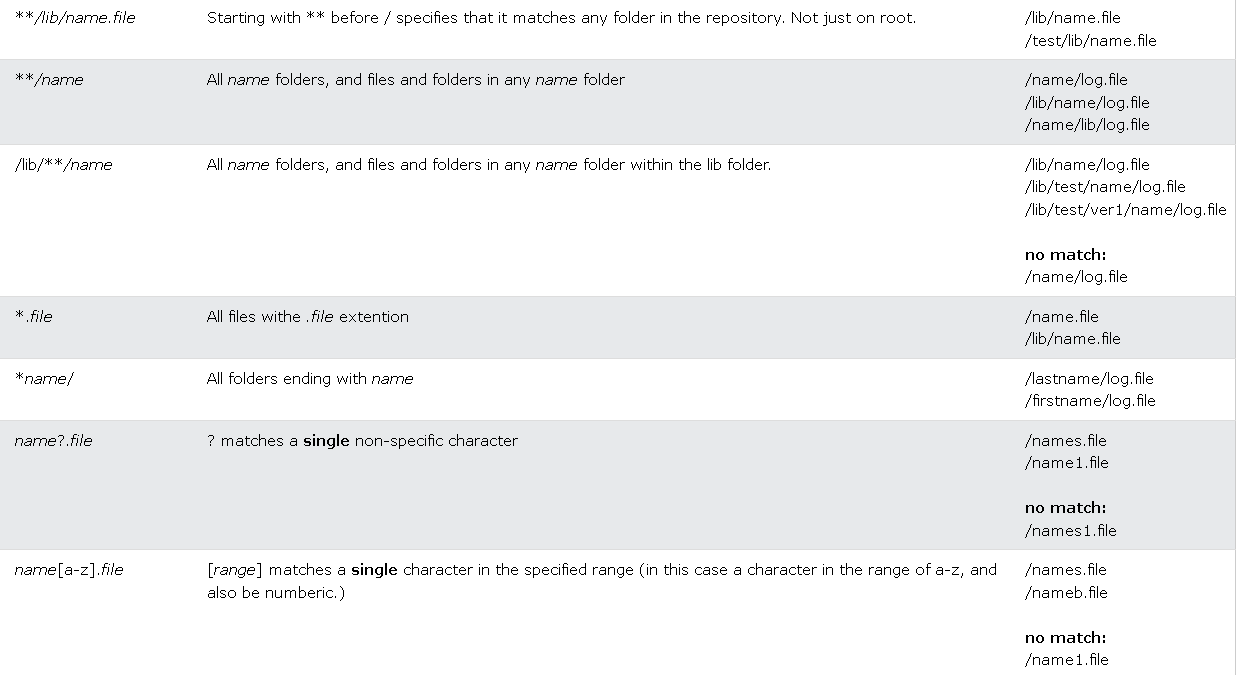
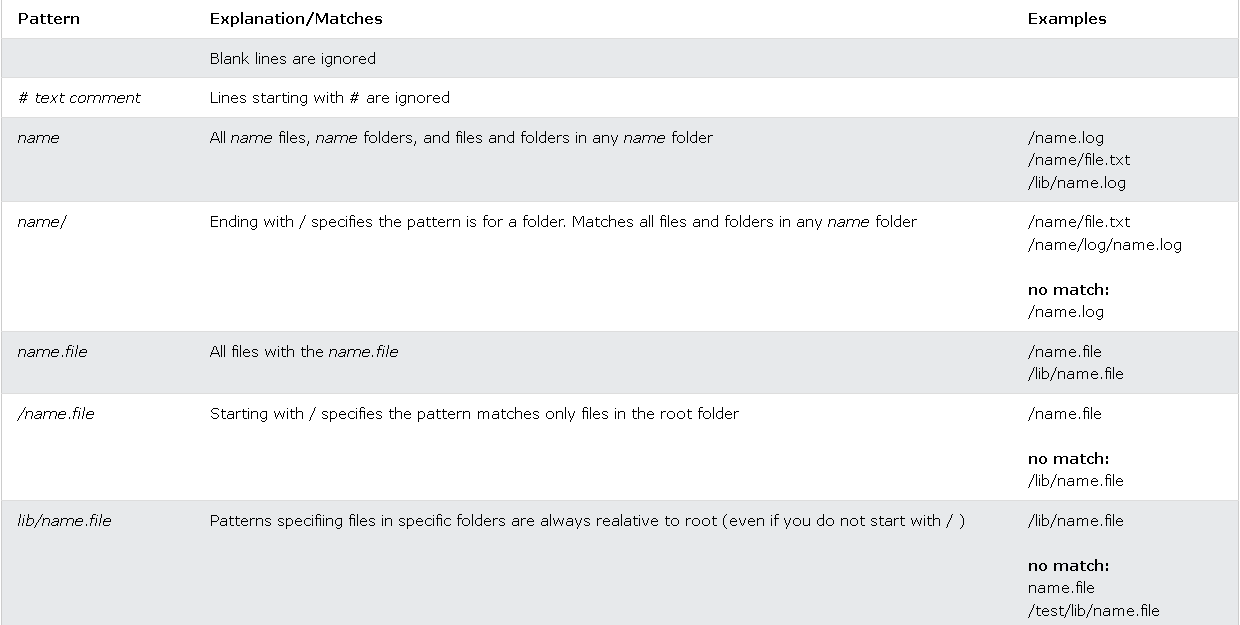
=> Fetch the URL of our own fork : (after fork someone else’s repo, we will copy <https://github.com/tienktka160304/w3schools-test.github.io.git>, like this)

=> add that as origin ; we will see this: 

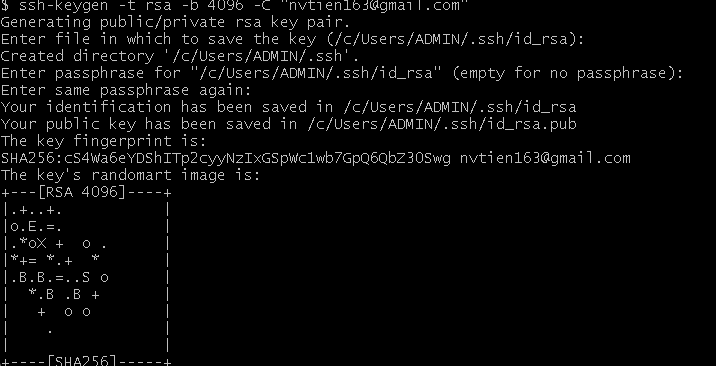
* + **Git GitHub Send Pull Request** :
    - **Push Changes to Our GitHub Fork** :
      * We will push the changes to GitHub fork : **git push origin .** (if something went wrong, run **git pull origin** first)
      * After that, going to github, we see that repo has a new commit -> we can send a ‘ ***Pull Request*** ‘ to the original repository: -> Click that and ‘***create a pull request***’ -> Remember to add an explanation for the administrators .
* Git Ignore and .gitignore :
  + **Git Ignore** :
    - Specify which files or parts of the project should be ignored.
    - NOT Git but .gitignore files itself IS tracked by Git.
  + **Create .gitignore** :
    - To create: touch .gitignore
    - 2 simple rules:
      * Ignore any files with **.log** extension.
        + Keep repo cleaner and more focused on the essential project files.
        + Improve the performance of Git operations:Reduce the number of files that need to be tracked and managed
      * Ignore everything in any directory named **temp/**.(Temporary directories: Thư mục tạm thời)
        + Ensure that any files or directories within that folder are not tracked by Git, even if they have different file extensions or names.
        + Especially useful if the project generates a lot of temp files or directories that you don't want to include in the version control system.



* + - Rules for .gitignore :



* + - Local and personal Git ignore Rules :
      * It can ignore files and folders but not show it in the distributed(phân phối) .gitignore file.
      * These kinds of ignores are specified in the .git/info/exclude file. They work the same way as .gitignore but are not shown to anyone else.
    - Git Security SSH :
      * Should use SSH if you work with unsecured networks.
    - **What is SSH** :
      * SSH is a secure ( vỏ ) shell network protocol(giao thức) that is used for network management, remote file transfer and remote system access.
      * When generating a set of keys, you will generate a “public” and “private” key.
      * The “public” key is the one you share with the remote party.(as the lock)
      * The “private” key is the one you keep for yourself in a secure place. (as a key to the lock)
      * The “public” key can be derived ( lấy ) from the “private” key, not the other way around.
    - Generating an SSH key pair :
      * Creating a new key : using email as a label(nhãn): ssh-keygen -t rsa -b 4096 -C “...” : cS4Wa6eYDShITp2cyyNzIxGSpWc1wb7GpQ6QbZ30Swg



=> Entering a secure passphrase will create an additional layer of security. It will require to supply the passphrase whenever SSH key is used

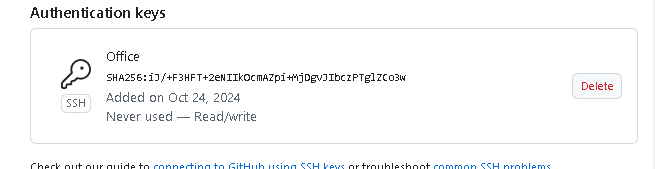
* + - * Adding SSH key pair to the SSH-Agent:(run: eval ssh agent -a first ) ssh-add /Users/…./.ssh/id\_rsa :



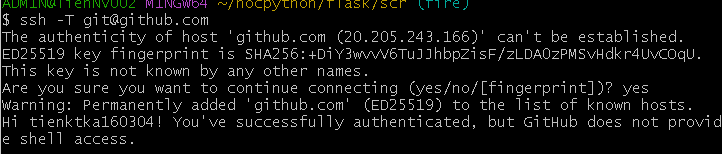
* + - **Copy the SSH Public Key** :
      * Use clip < to copy the public key to our clipboard(the .pub):



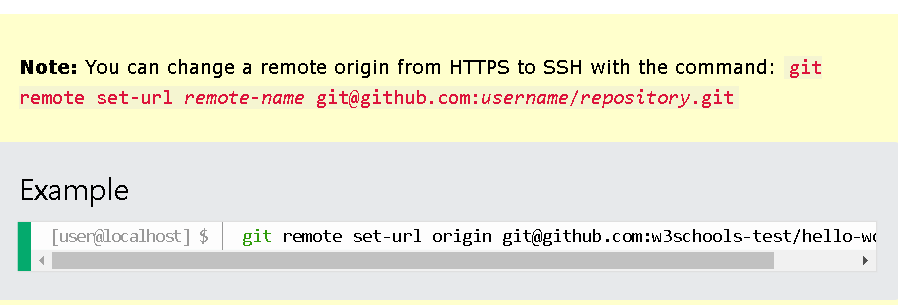
* + - * Go to Github -> Settings -> SSH and GPG keys -> New SSH keys -> title : … -> Key : open id\_rsa.pub to copy and paste.



* + - **Test SSh Connection to GitHub** :
      * Command: ssh -T [git@github.com](mailto:git@github.com)



* + - Add New GitHub SSH Remote :
      * Get the URL and add that address as a new origin : git remote add ssh-origin ….



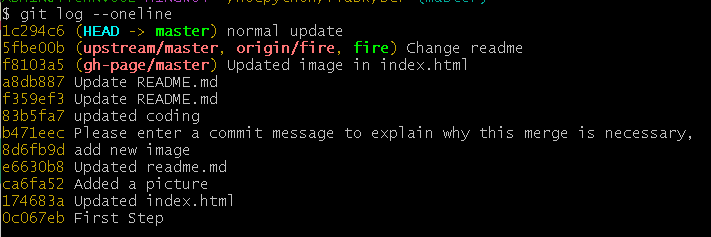
* + - Git Revert :
      * Revert : Take a previous commit and add it as a new commit, keeping the log intact.

=> Step 1: Find the previous commit:

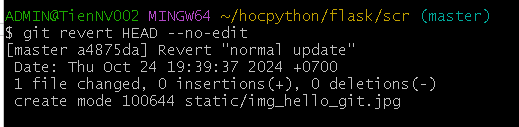
=> Step 2: Use it to make a new commit:

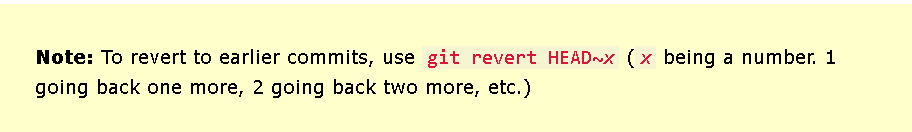
* + - * Make a new commit by “accidentally” deleting a file: git commit -m “ a normal update”
    - Git Revert Find Commit in Log :
      * Find the point we want to return to. -> go through the log.

=> use the --oneline option to give one line per commit.



* + - Git Revert HEAD :
      * After identifying the commit, using git revert HEAD, adding the option --no-edit to skip the commit message editor(getting default revert message)



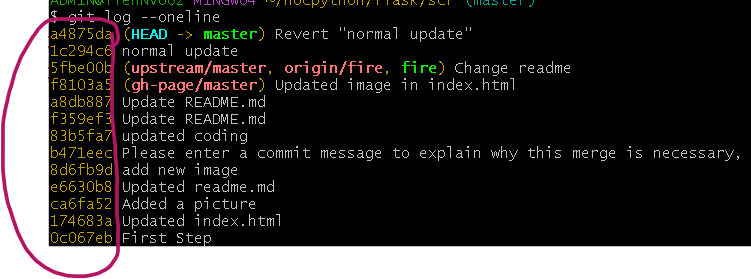
=> git log --oneline again to check.

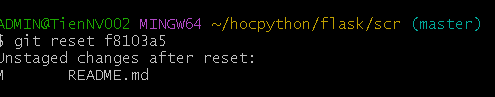
* + - Git Reset :
      * Reset: move the repo back to a previous commit, discarding any changes made after the commit (loại bỏ mọi thay đổi được thực hiện sau khi xác nhận.)

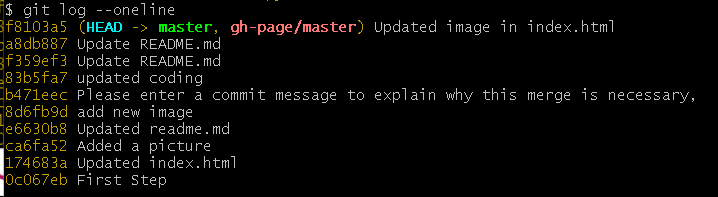
=> Step 1: Find the previous commit :

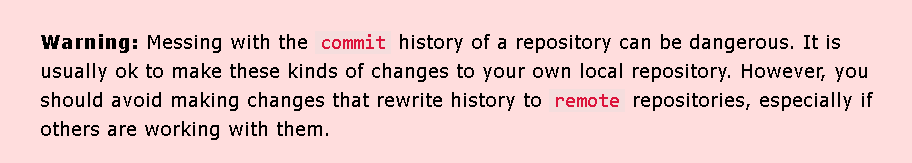
=> Step 2: Move the repo back to that step:

* + - Git Reset Find Commit in Log :
      * Find the point we want to return to. -> git log --oneline again.
    - Git Reset :
      * To reset our repo back to the specific commit : git reset *commithash* (*commithash* being the first 7 characters of the commit hash we found in the log)

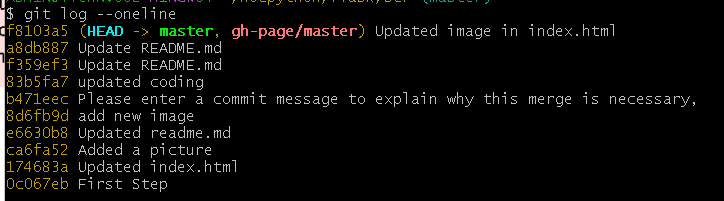


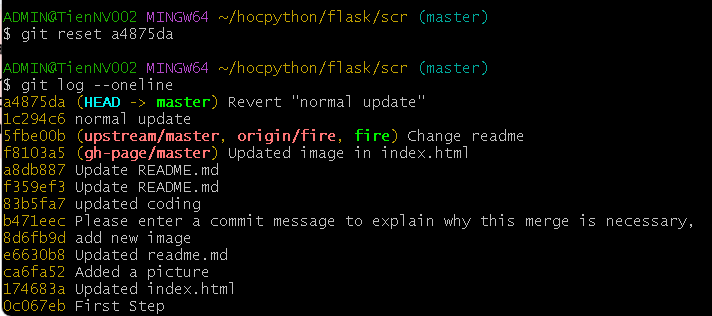






* + - Git Undo Reset :
      * The commits are not removed from the Git after reset.
      * To undo, just reset commithash of the commit we want to return to.





* + - Git Amend :
      * commit --amend : to modify the most recent commit.(sửa đổi cái gần nhất)
      * It combines changes in the staging environment with the latest commit and creates a new commit.
      * The new commit replaces the latest commit entirely.(thay thế hoàn toàn).
    - Git Amend Commit Message :
      * The simplest thing is to change a commit message.

=> Update the README.md and commit => git log to check => Lets amend: git commit --amend -m “....” -> it'll change the message commit without changing anything in our repo or project.

* + - **Git Amend Files** :

**Thẻ 2**

3

24/10/2024

* To show how it changes : **git show *commithash***

23/10/2024

* **Git : .gitignore/**
  + **.venv/** bỏ qua các môi trường ảo (Python)
  + **# Byte-compiled / optimized / DLL files**
    - **\_\_pycache\_\_/**
    - **\*.py[cod]**
    - **\*$py.class**
  + **# C extensions**
    - **\*.so**
  + **# Distribution / packaging**
    - **.Python**
    - **env/**
    - **venv/**
    - **ENV/**
    - **env.bak/**
    - **.venv/**
    - **build/**
    - **dist/**
    - **\*.egg-info/**
    - **\*.egg**
  + **# Installer logs**
    - **pip-log.txt**
    - **pip-delete-this-directory.txt**
  + **# Jupyter Notebook**
    - **.ipynb\_checkpoints**
  + **# Python test output**
    - **.pytest\_cache/**
    - **.coverage**
    - **.coverage.\***
    - **nosetests.xml**
    - **coverage.xml**
    - **\*.cover**
    - **.hypothesis/**
    - **htmlcov/**
  + **# Sphinx documentation**
    - **docs/\_build/**
    - **build/**
    - **\*.log**
  + **# Pyre type checker**
    - **.pyre/**
  + **# mypy**
    - **.mypy\_cache/**
  + **# PyCharm**
    - **.idea/**
    - **\*.iml**
  + **# VSCode**
  + **.vscode/**
  + **# Sublime Text**
    - **\*.sublime-project**
    - **\*.sublime-workspace**
  + **# Rope**
    - **.ropeproject/**
  + **# MacOS system files**
    - **.DS\_Store**
  + **# Windows system files**
    - **Thumbs.db**
    - **ehthumbs.db**
  + **# Linux trash folder**
    - **.trash/**
  + **# Environment variable files**
    - **.env**
    - **.env.local**
    - **.env.\*.local**
  + **# Docker**
    - **\*.tar**
    - **\*.log**
    - **docker-compose.override.yml**
  + **# Others**
    - **\*.pkl # Pickle files**
    - **\*.dat # Data files**
    - **\*.csv # CSV files (if you don't want to track them)**
* **Flask-SQLAIchemy: provides a high-level API for interacting with a SQL databases**