

# ▶ Git Hands on-01

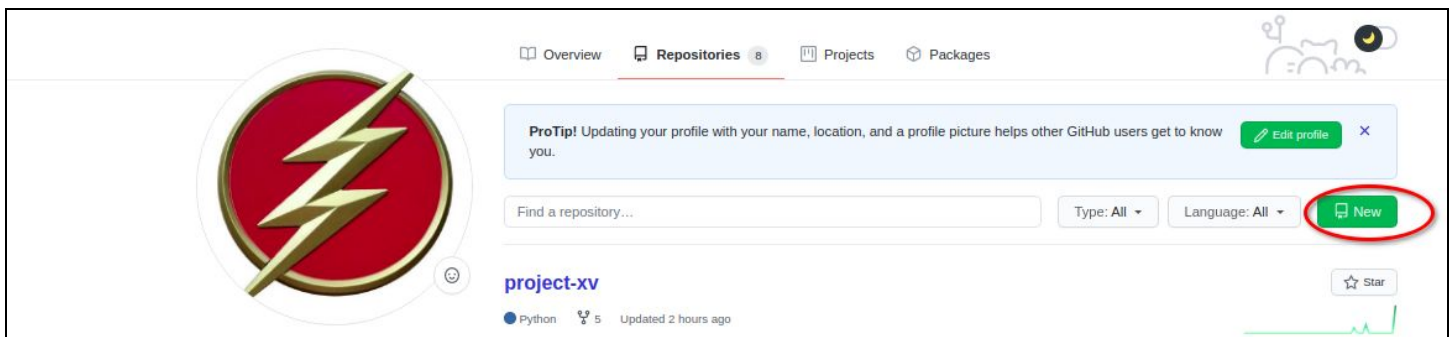
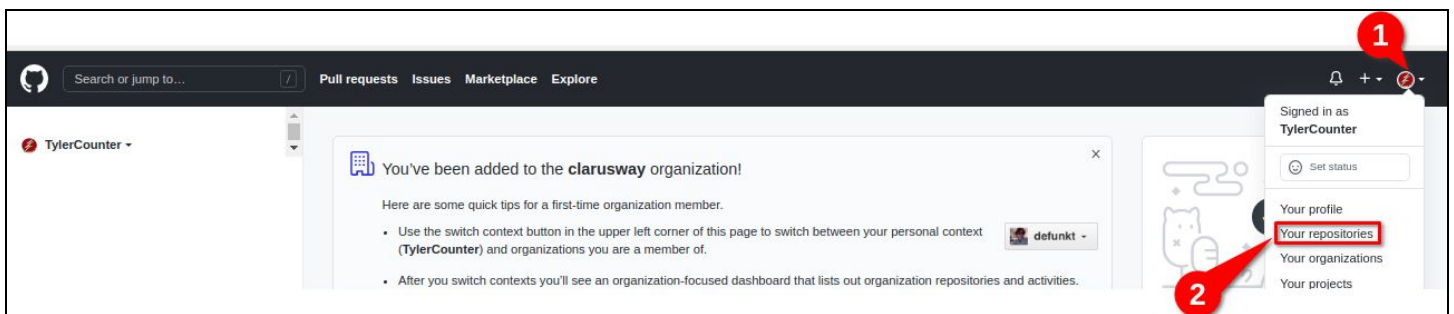


CLARUSWAY©  
WAY TO REINVENT YOURSELF

## Part 1:

### 1. Create a public repository in GitHub:

- named `python-project`
- write a description of your repository
- add README.md file



## Create a new repository

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

---


### Repository template

Start your repository with a template repository's contents.

No template ▾

---

**Owner \*** **Repository name \***

1  TylerCounter ▾ / python-project ✓

Great repository names are short and memorable. Need inspiration? How about [potential-invention?](#)

**Description (optional)**

2 this repository for my python project

3 ☒ **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:**  
Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

4 [Create repository](#)



- After clicking the Create repository. GitHub creates a public repository and led you to the following page.

[TylerCounter / python-project](#)

---


<> Code ⓘ Issues 🔗 Pull requests ⚙️ Actions 📁 Projects 📖 Wiki ⓘ Security 📊 Insights ⚙️ Settings

---


 main ▾  1 branch 🔍 0 tags

[Go to file](#) [Add file ▾](#) [Code ▾](#)

---

 **TylerCounter** Initial commit c5842f7 now ⌚ 1 commit

---

 README.md Initial commit now

---

README.md

**python-project**

this repository for my python preject

## Part 2:

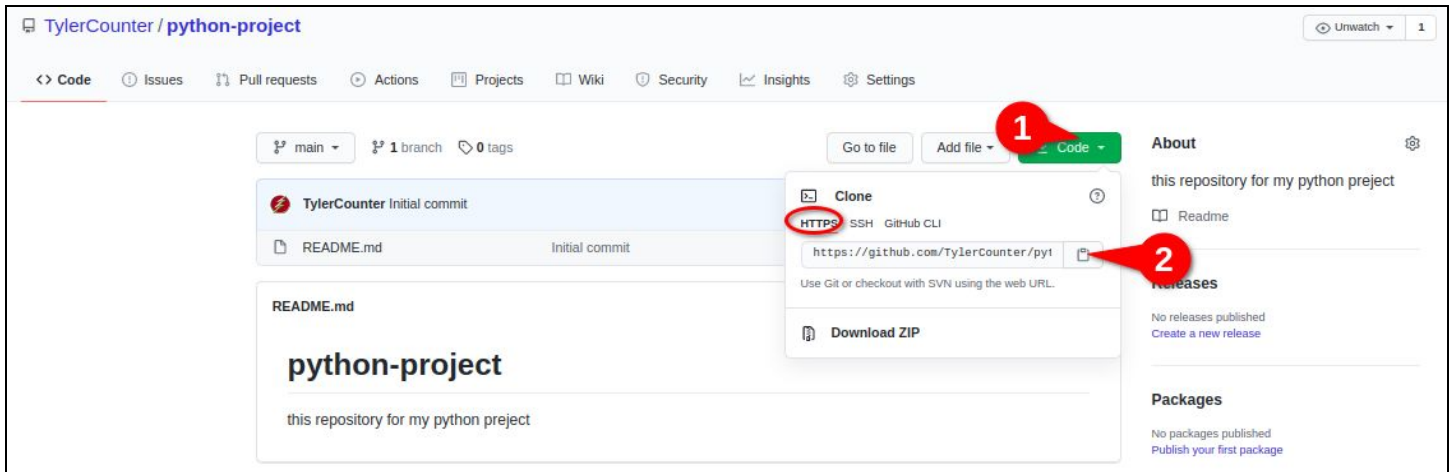
### 2. Clone your remote repository to your computer:

- Open your terminal (for Windows, run "Git Bash")
  - Make a directory named git-lesson under the desktop directory and cd into it.

```
mkdir git-lesson
```

```
cd git-lesson
```

- Clone your remote repository (Syntax: `git clone <remote-url>` )
  - Copy your remote repository URL



- run the following command

**`git clone https://github.com/TylerCounter/python-project.git`** (→ use your remote repo URL)

Output:

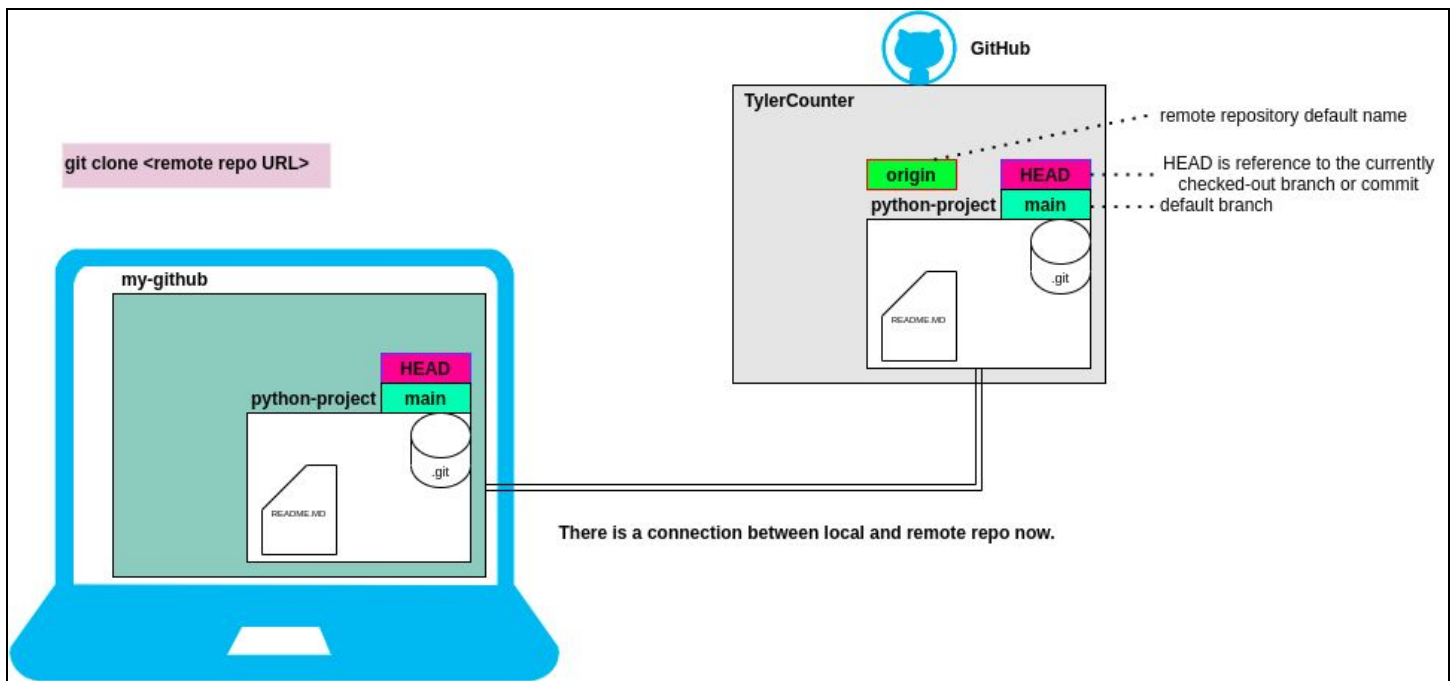
```
Cloning into 'python-project'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

- See the current state of the project:

```
git status
```

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```



### Part 3:

- Create a file named hello-world.py

```
touch hello-world.py
```

- stage it

```
git add hello-world.py
```

- store it in the local repository

```
git commit -m "created hello-world.py"
```

- open hello-world.py and add a line, then save and close.

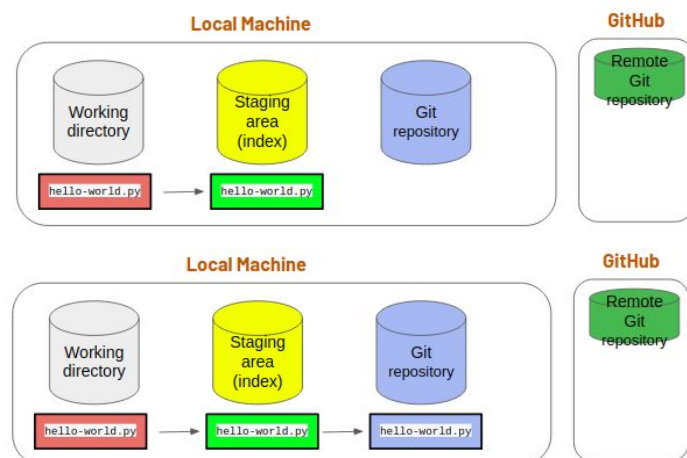
```
vim hello-world.py
```

- check the status of the folder

```
git status
```

- store it to local repository, and check the state of the folder

```
git commit -am "updated hello world.py"
```



## git status

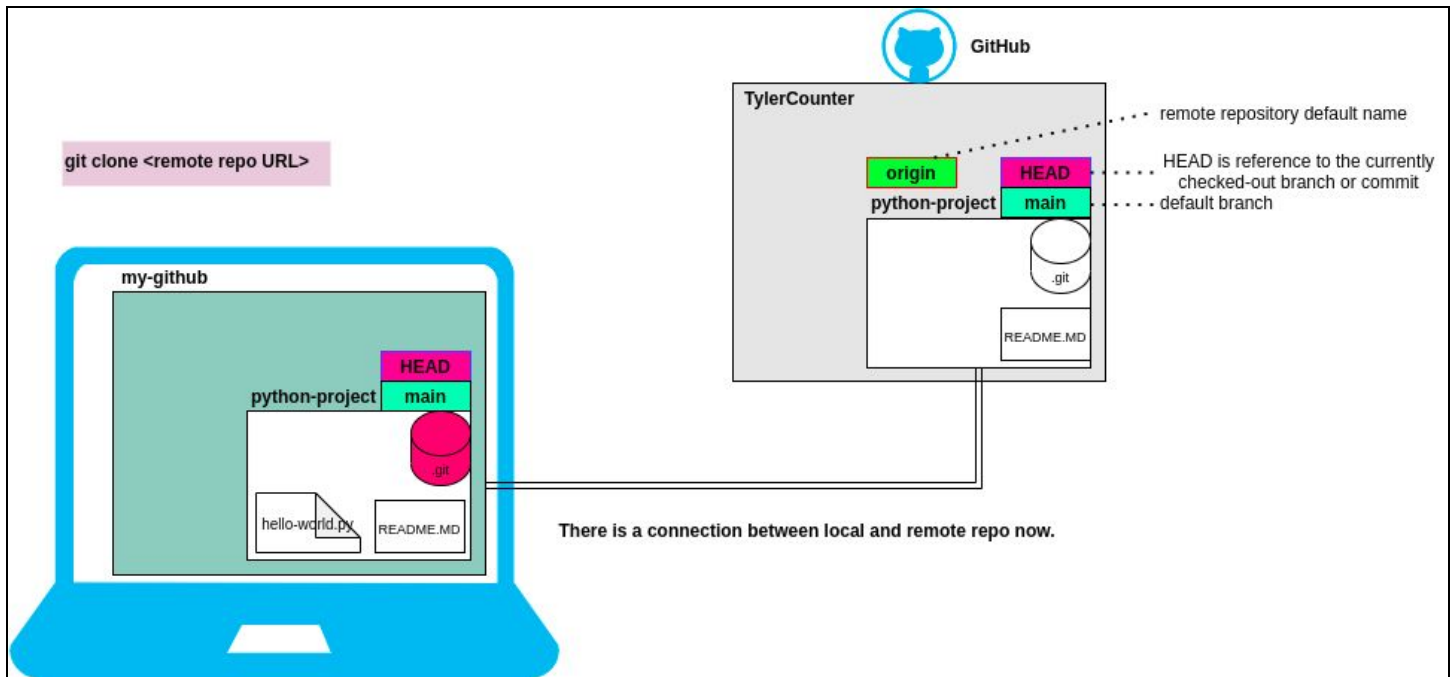
- See the commit history

-

## git log

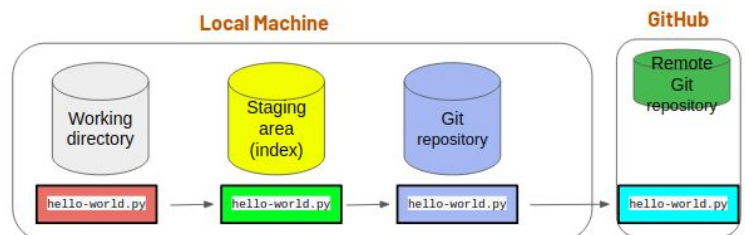
`git log --pretty=oneline`

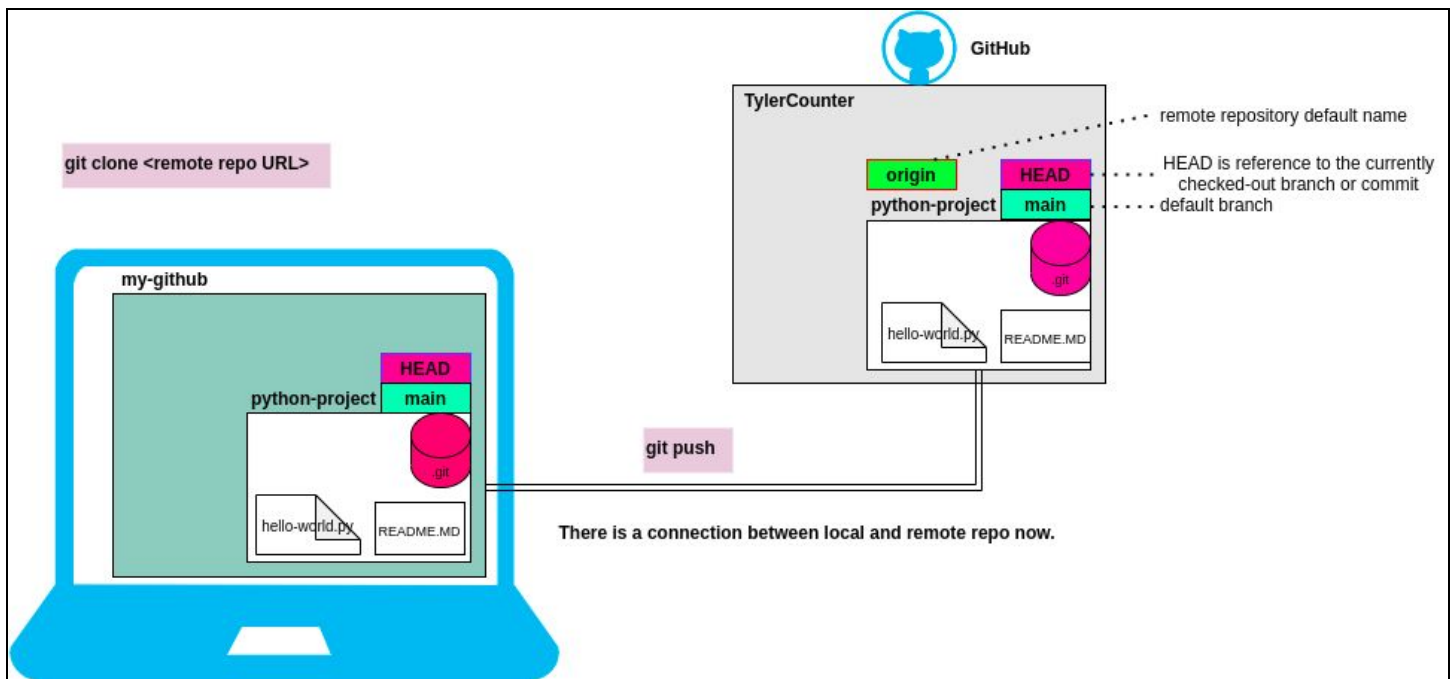
`git log --oneline`



- Then send the changes to your remote repo.

## git push





- Go to your GitHub account and see the changes:

TylerCounter / python-project

<> Code ⓘ Issues 🔗 Pull requests ⚙️ Actions 📁 Projects 📖 Wiki 🛡️ Security 📊 Insights ⚙️ Settings

main 1 branch 0 tags Go to file Add file Code

TylerCounter created hello-world.py 7ff3ba0 20 seconds ago 2 commits

README.md	Initial commit	now
hello-world.py	created hello-world.py	20 seconds ago

README.md

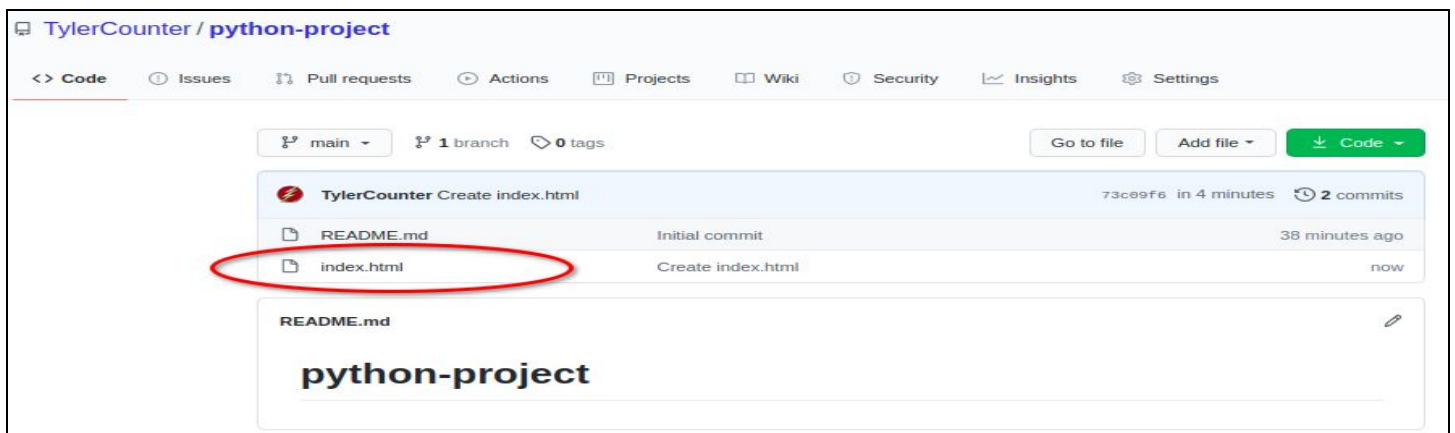
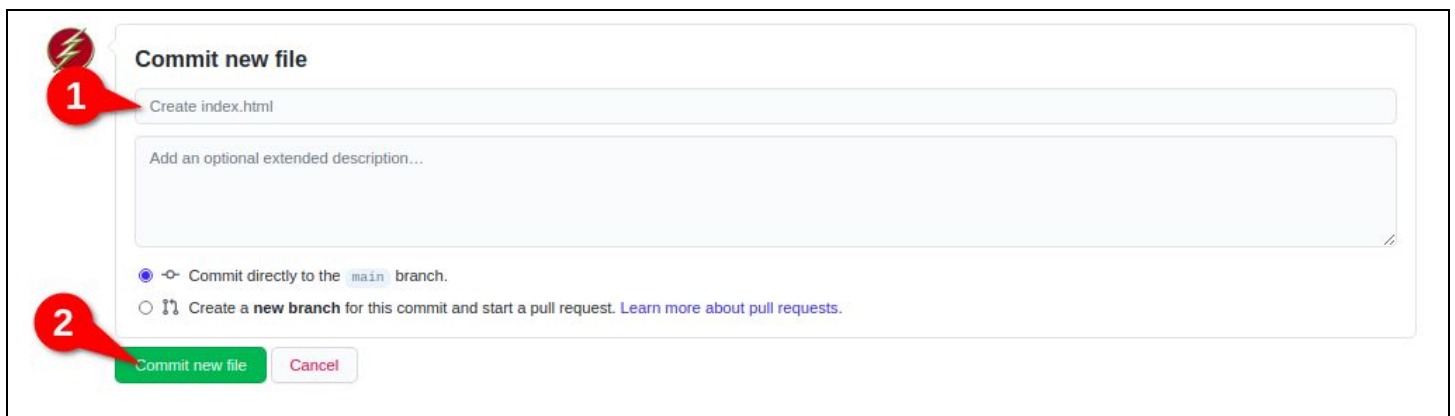
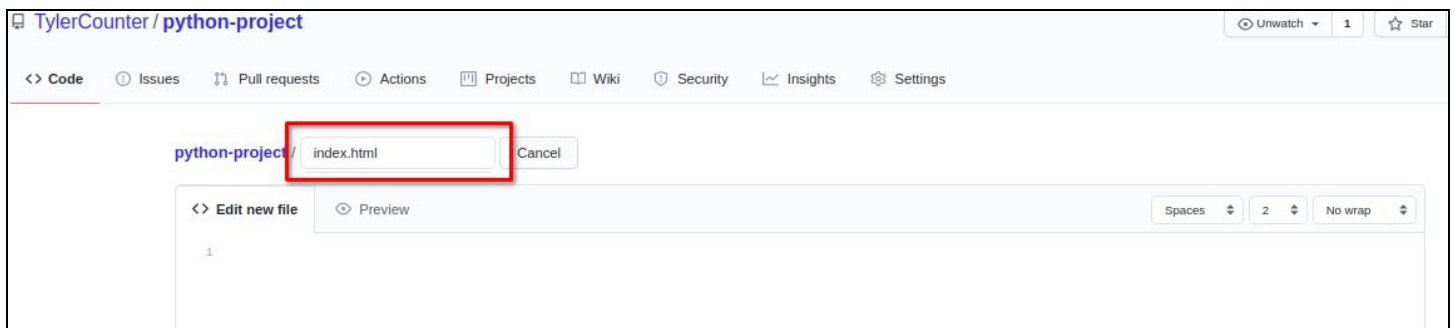
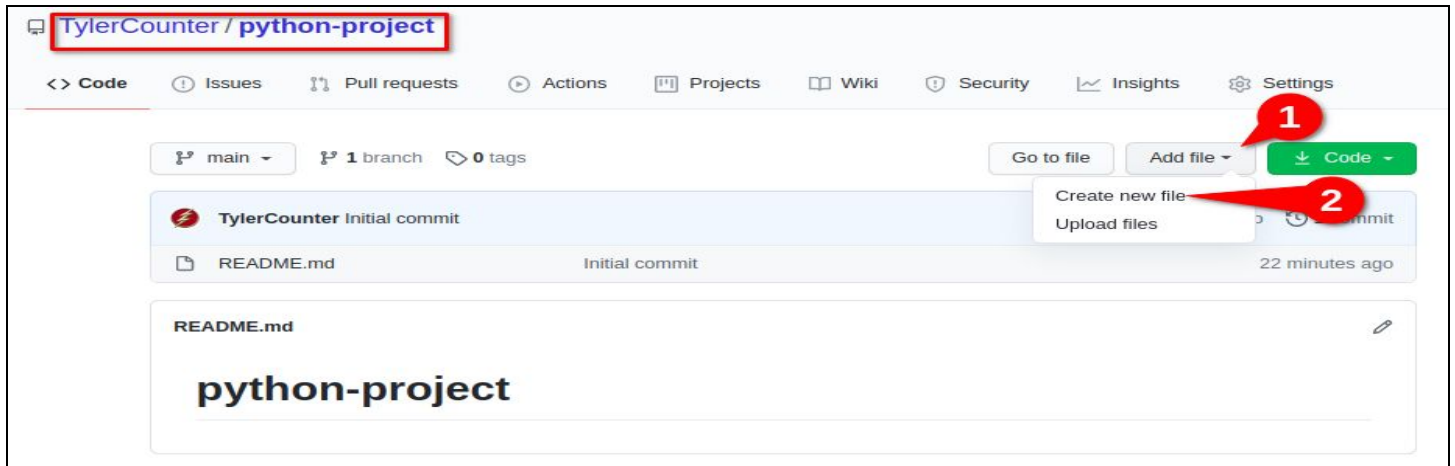
## python-project

This repository for my python project.

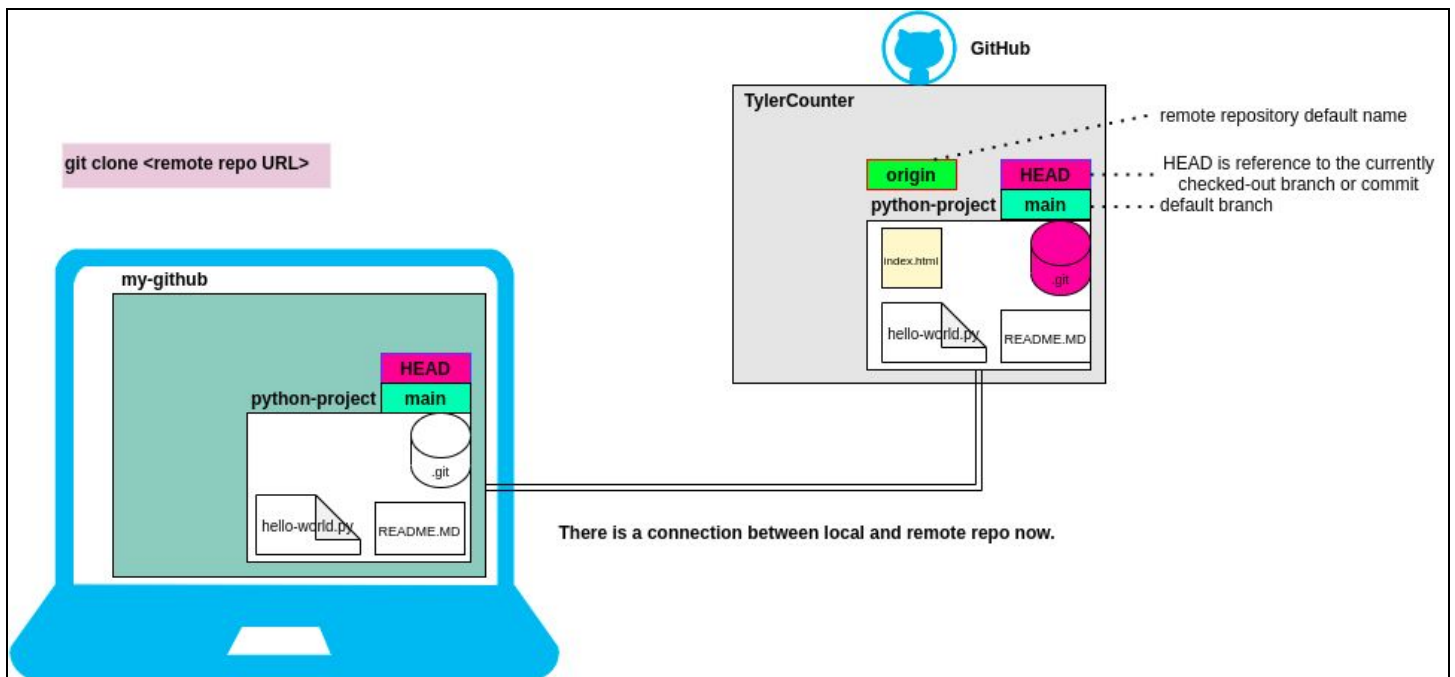


## Part 4:

- Create a new file named index.html in the python-project repository by using GitHub.



- The current state of our project:



- At GitHub check the commits: (click commits)

You will see the three commits

The screenshot shows the GitHub repository page for "TylerCounter / python-project". The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, the repository name "python-project" is displayed. The main content area shows the commit history. The first commit, "TylerCounter Create index.html", is highlighted with a red circle around the commit count "3 commits". The commit list includes:

Commit Message	Author	Time
Initial commit	TylerCounter	4 minutes ago
created hello-world.py	TylerCounter	5 minutes ago
Create index.html	TylerCounter	now

The screenshot shows the commit history page for the repository. The top navigation bar includes links for main, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area shows the commit history for the "main" branch. The commit list includes:

Commit Message	Author	Time	Commit Hash
Create index.html	TylerCounter	committed in 1 minute	f226e7f
created hello-world.py	TylerCounter	committed 6 minutes ago	7ff3ba8
Initial commit	TylerCounter	committed 5 minutes ago	a2011b7



## Part 5:

- Go to the terminal and see the commit history

`git log`

```
ubuntu:~/Desktop/my-github/python-project(main)
$ git log
commit 7ff3ba066178a6556f002816ee1770172e38e227 (HEAD -> main, origin/main, origin/HEAD)
Author: TylerCounter <"tyler@clarusway.com">
Date: Thu Dec 17 04:59:31 2020 -0500

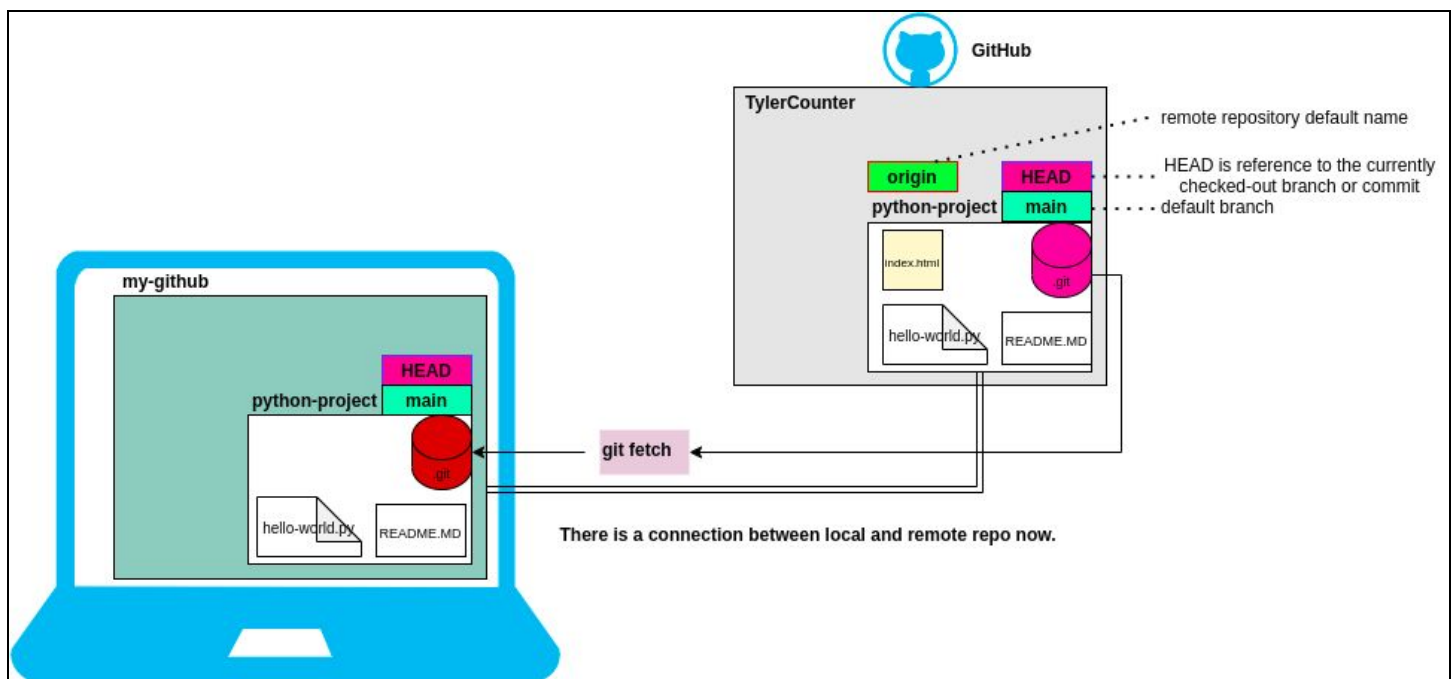
    created hello-world.py

commit a2011b756e457212ab810e7635fb6e0c71d80766
Author: TylerCounter <63177502+TylerCounter@users.noreply.github.com>
Date: Thu Dec 17 05:00:50 2020 -0500

    Initial commit
```

- Download the changes from the remote repository to your local repository

`git fetch`



- See the changes in local repository

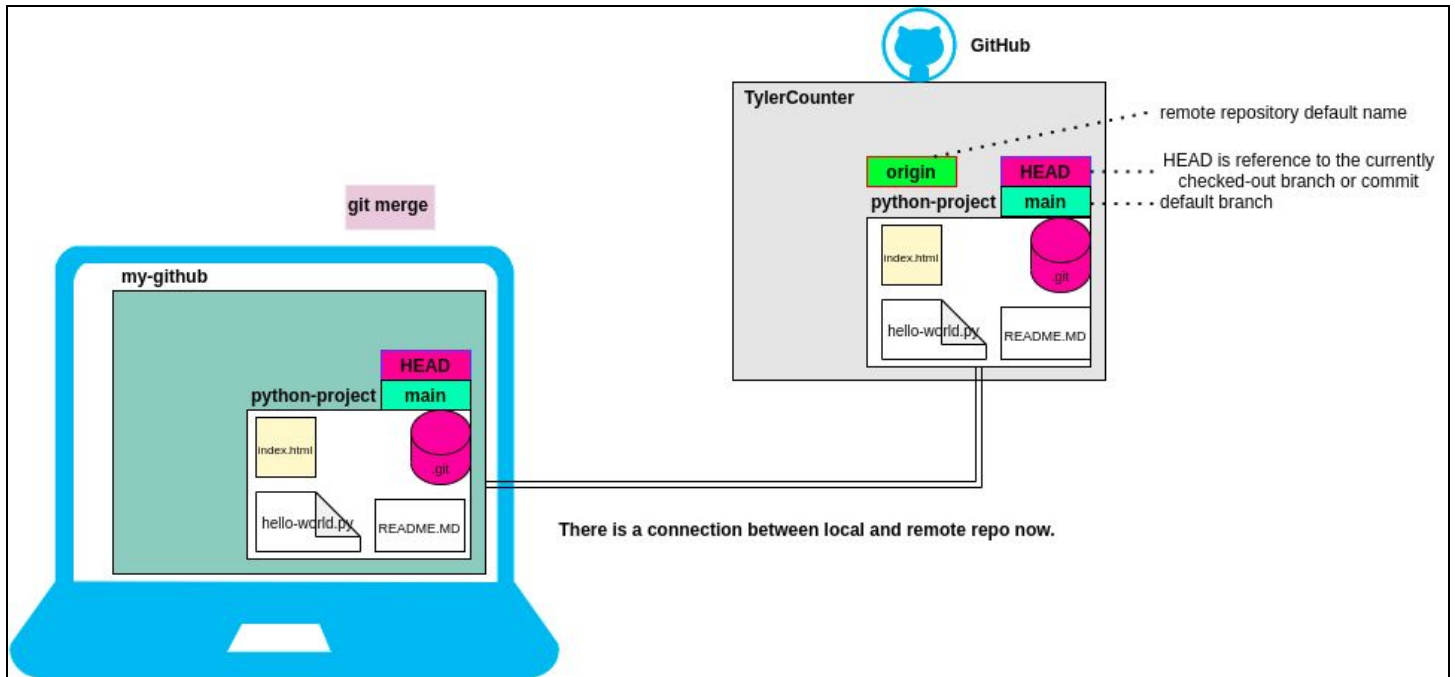
`git diff main origin/main`

- Combine main and origin/main

`git merge`

- Then list the files in your working directory

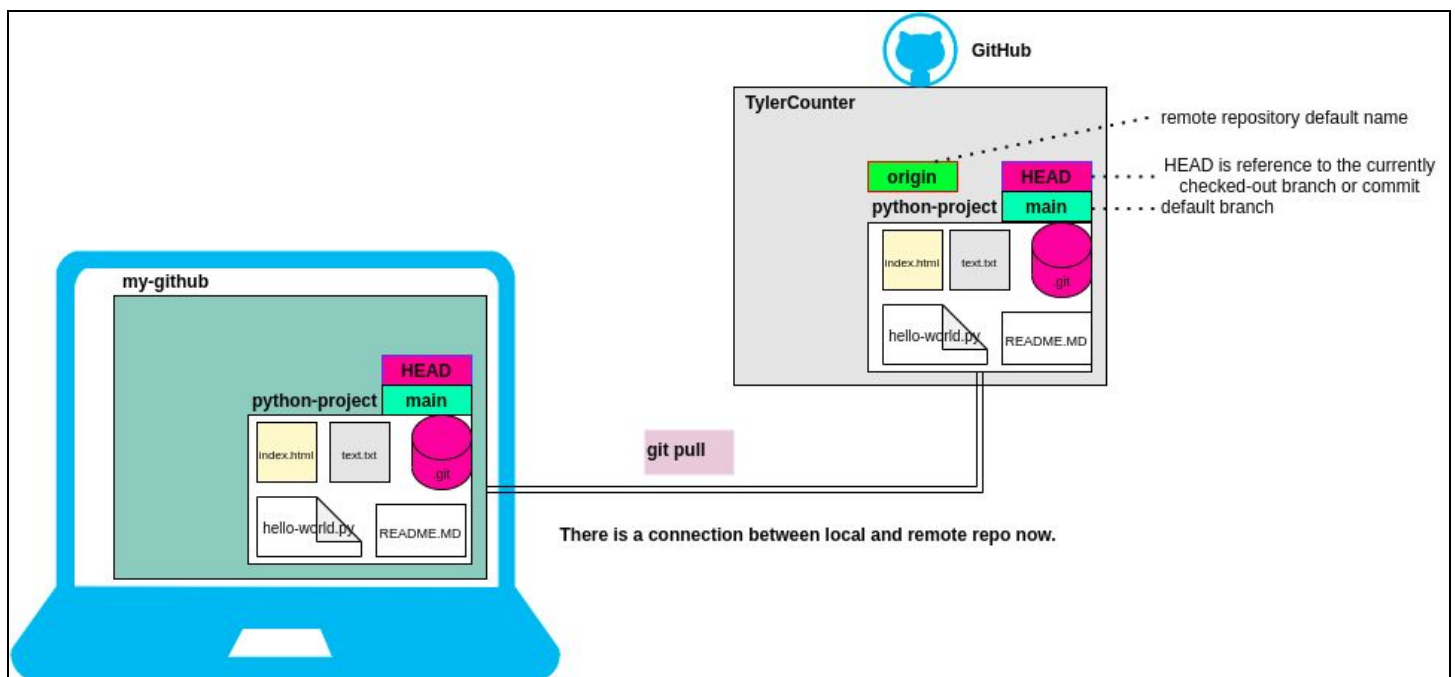
`ls`



## Part 5:

- At GitHub, create a new file named **test.txt**
- Download all changes to your computer (terminal)

(that perform **git fetch + git merge** automatically)



- See the commit history

```
git log --oneline
```

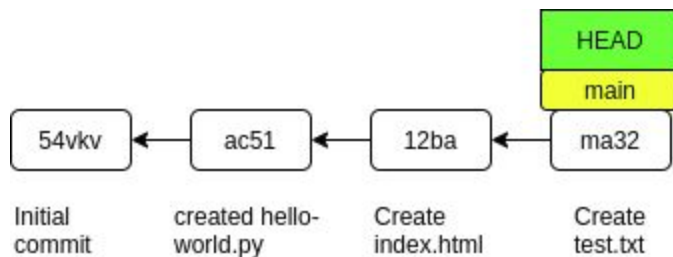
```
2fa656d (HEAD -> main, origin/main, origin/HEAD) Create test.txt
f226e7f Create index.html
7ff3ba0 created hello-world.py
a2011b7 Initial commit
```

- Lets go the first commit, and see the changes in the working directory

```
git checkout <commitID>
```

- Switch the last commit again. (main)

```
git checkout main
```



## Part 6:

- Create a new branch named **front-end**

```
git branch front-end
```

- See branches

```
git branch (show local branches)
```

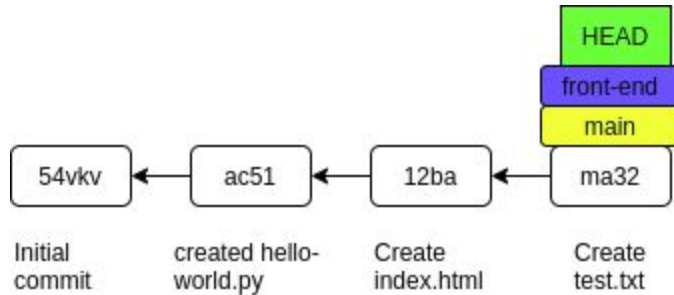
```
git branch -r (show remote branches)
```

```
git branch -a (show all local and remote branches)
```

```
front-end
* main
remotes/origin/HEAD -> origin/main
remotes/origin/main
```

- Switch to **front-end** branch

**git checkout front-end**



- List the files and check the status of the working directory

**ls**  
**git status**

- Make some changes in the **test.txt** file, and check the status

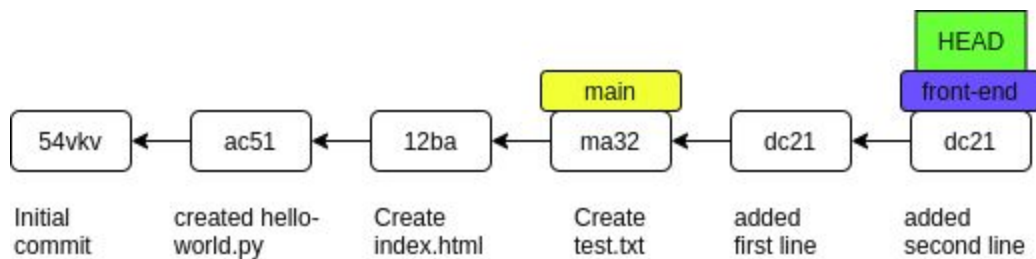
**vim test.txt**  
**git status**

- Store the changes to the repo and check the status

**git commit -am "added first line"**  
**git status**

- Add another line to **test.txt** and store it to the local repo.

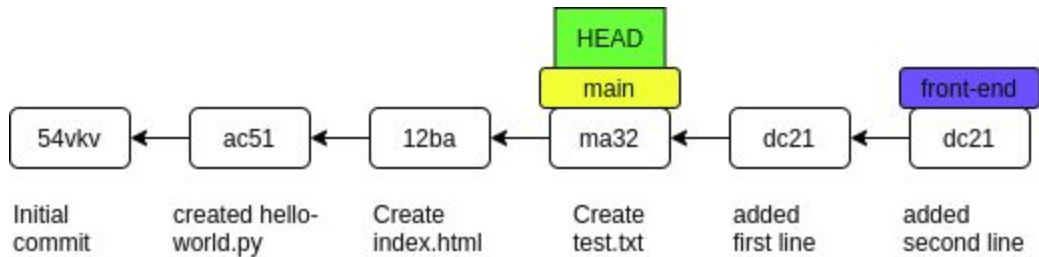
**vim test.txt**  
**git commit -am "added second line"**



- Switch the main branch and see the content of the **test.txt**

```
git checkout main
```

```
cat text.txt
```

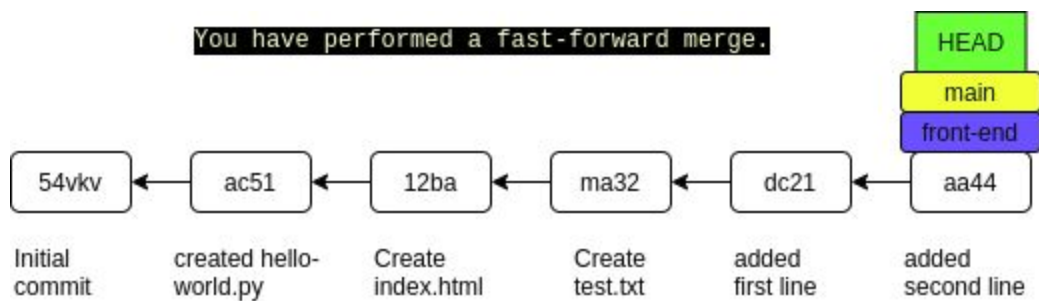


- Merge front-end branch to **main** branch.

```
git merge
```

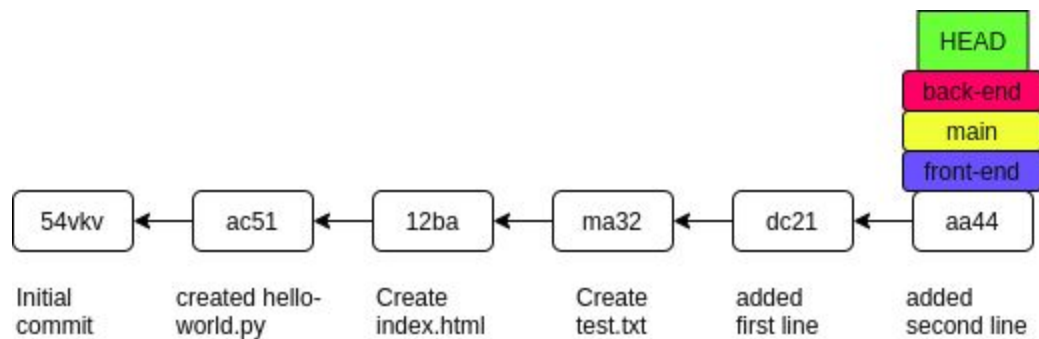
```
cat test.txt
```

**You have performed a fast-forward merge.**



- Create a new branch named **back-end** and switch to it

```
git checkout -b back-end
```

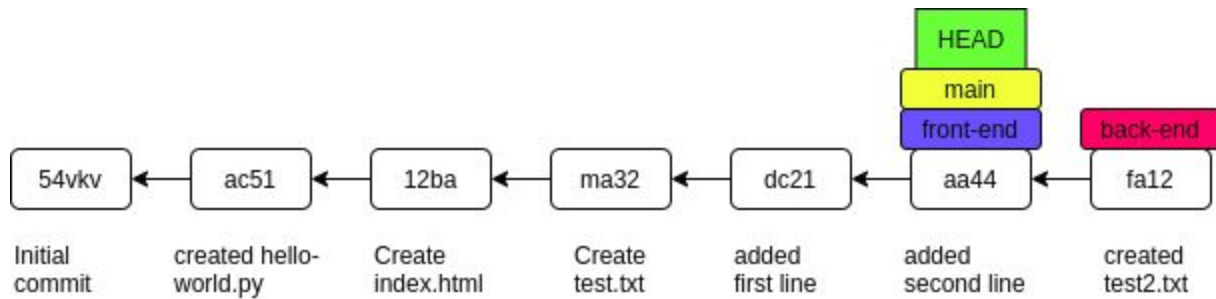


- Create a new file named **test2.txt** and store the changes to repo.

```
touch test2.txt
git add .
git commit -m "created text2.txt"
```

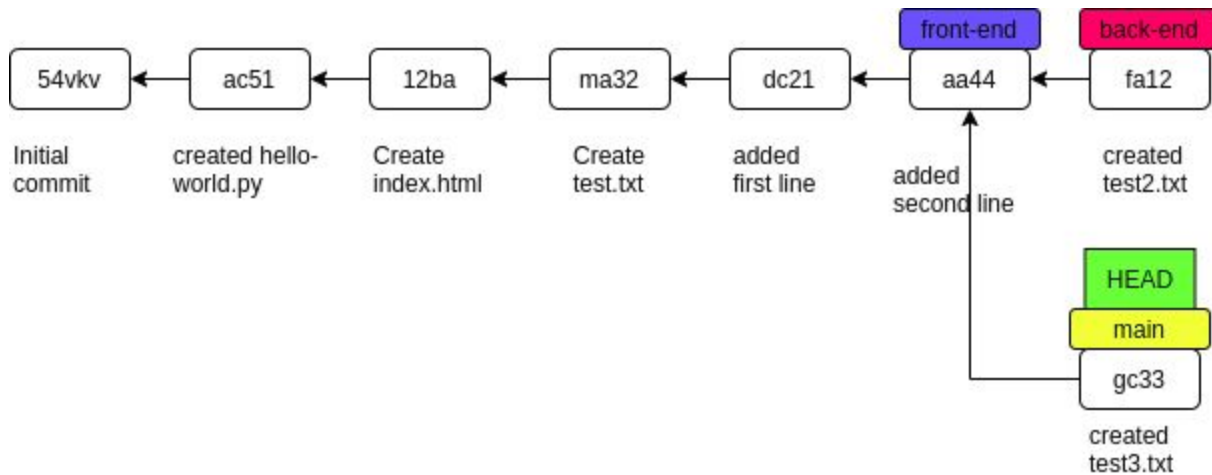
- Switch the main branch again

```
git checkout main
```



- Create a new file **test3.txt** and send the changes to local repo.

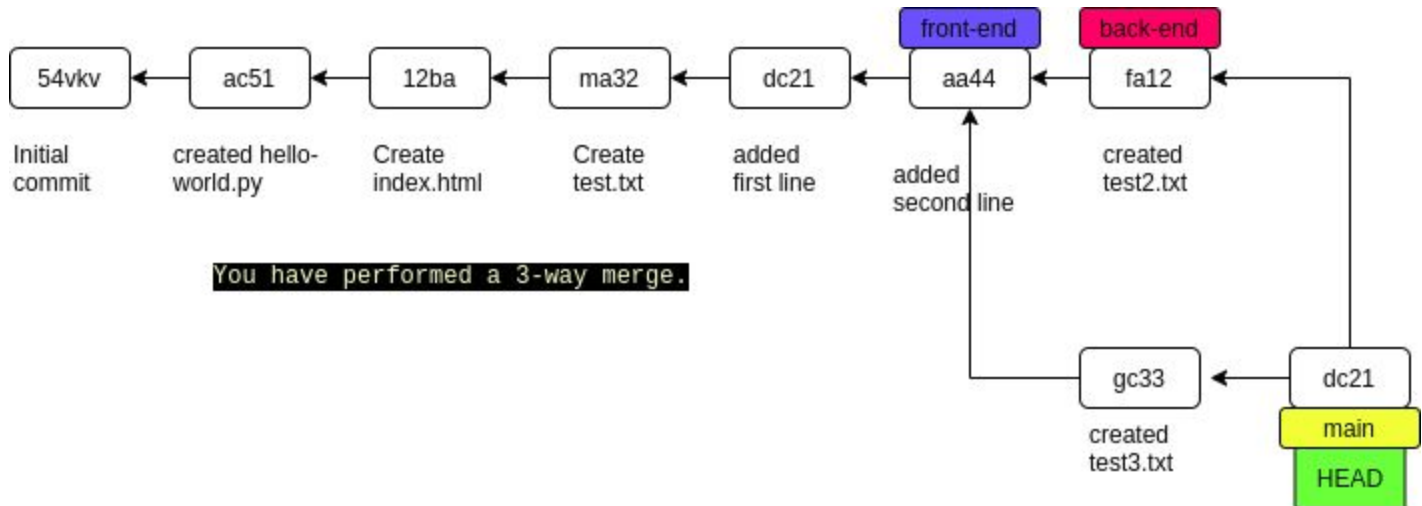
```
touch test3.txt
git add .
git commit -m "created text3.txt"
```





- Merge **main** branch with **back-end** branch

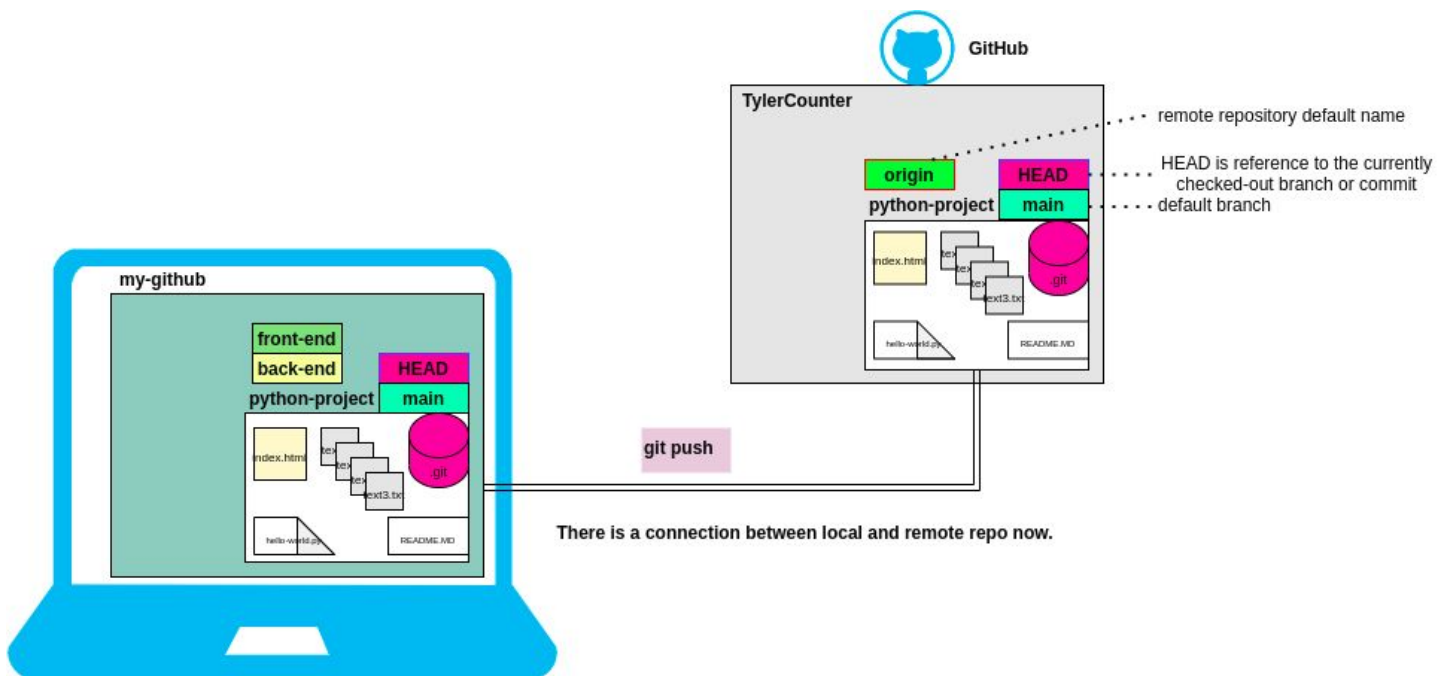
`git merge back-end`



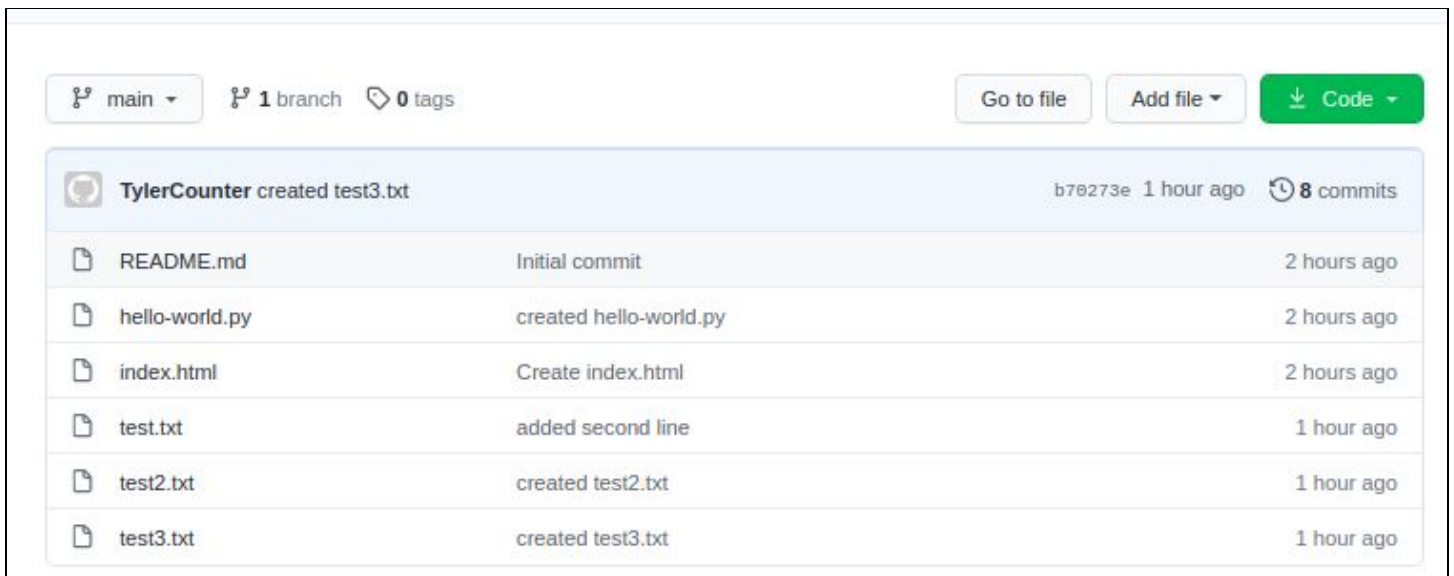
## Part 7:

- Send the changes to the remote repository

`git push`



- Go and check the remote repository



### **Part 7:**

- Go to the terminal and delete the branches named **front-end** and **back-end**

**git branch -d front-end**  
**git branch -D back-end**

- List the all branches

**git branch -a**

