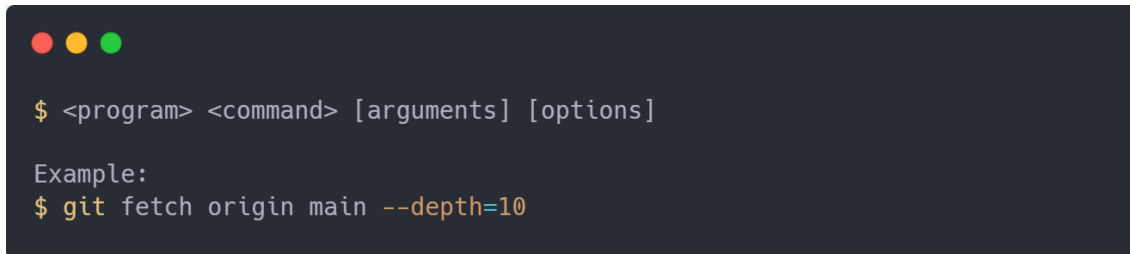


GIT

Quick introduction about a command line interface

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It displays the general command line syntax and an example command.

```
$ <program> <command> [arguments] [options]  
Example:  
$ git fetch origin main --depth=10
```

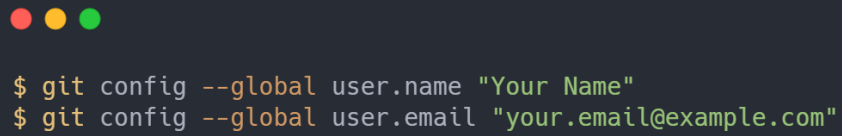
A command line interface is an interface where we can type a command for the computer to run some tasks. What you see in the image above is a command line syntax. Here's the explanation

1. The program is "git"
2. The command is "fetch"
3. The arguments are "origin" and "main"
4. The option name is "depth" and the value of the option is 10

You run a command inside a shell. Shell is a user interface that will be in charge of running all of the commands that you type in the CLI. Examples of shells are Windows Powershell, Command Prompt, Fish, Git Bash, etc.

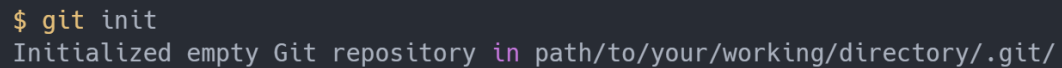
Git Setup

1. First, configure the global user name and email for your machine so Git can identify who you are.



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

2. Create a new folder which will be the root folder of the project.
3. You can initialize a local repository using the command below. This will initialize a git repository inside the folder you are working on.



```
$ git init
Initialized empty Git repository in path/to/your/working/directory/.git/
```

4. Once you initialize a repository, you can add files inside the folder. Git will track those files and mark them as an untracked file. Untracked file means a new file has been added to the directory.

Git Workflow

1. Git can track multiple files inside a repository. Let's add 2 files here, `readme.md` and `code_of_conduct.md`

```
$ ls <-- a command to list files inside a folder
```

Mode		LastWriteTime		Length	Name
-a---		11/10/2022	08:03	0	code_of_conduct.md
-a---		11/10/2022	08:02	0	readme.md

2. Check the status of the files by using the command below. You will see those 2 files are marked as Untracked

```
$ git status
```

On branch main

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

code_of_conduct.md
readme.md

nothing added to commit but untracked files present (use "git add" to track)

3. Use the command "git add" to stage the changes. This will add all unstaged files to the staging area. Staged files are ready to be committed.

```
$ git add .

$ git status

On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   code_of_conduct.md
        new file:   readme.md
```

4. Once you stage those files, you can commit them using the command below. You must specify a message on every commit using the option -message or -m

```
$ git commit -m "Initial commit"

[main (root-commit) 8a10833] Initial commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 code_of_conduct.md
 create mode 100644 readme.md
```

5. To check the past commits, you can use the command "git log"

```
$ git log
commit 8a10833a69bab33439dbc388ef0ce162906bd2f6 (HEAD -> main)
Author: Novan R. Kuncoro <novan.kuncoro@zettacamp.pro>
Date:   Tue Oct 11 08:21:47 2022 +0700

    Initial commit
```

6. You can modify the committed files and those files will be marked as Modified.
7. To demonstrate, open readme.md and put the text "Hello from Git" inside the file.

8. Now check the status of the repository using the command "git status"

```
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.md

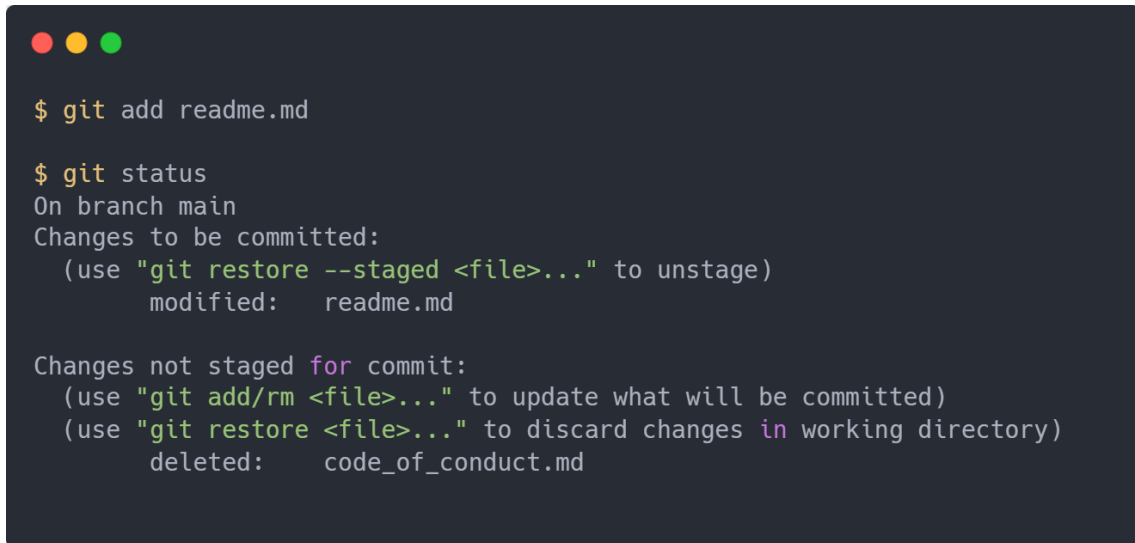
no changes added to commit (use "git add" and/or "git commit -a")
```

9. You can see there is a single file in the unstaged area. The file is readme.md and it is marked as Modified.
10. Now, delete the file code_of_conduct.md and check the status of the repository again.
11. Notice the file code_of_conduct.md is marked as Deleted in the command output.

```
$ git status
On branch main
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    code_of_conduct.md
        modified:   readme.md

no changes added to commit (use "git add" and/or "git commit -a")
```

12. This time, add only readme.md to the staging area. To achieve it, we can use the command "git add <file>"



```
$ git add readme.md

$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   readme.md

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    code_of_conduct.md
```

13. So the command "git add" can add single or multiple files to the staging area.

14. By running the command "git commit", you can see only the file readme.md is committed.

```
$ git commit -m "Edited readme.md"

[main 944bf06] Edited readme.md
 1 file changed, 1 insertion(+)

$ git log

commit 944bf0660b78e714e1b275a40ec9e44464dfe1e0 (HEAD -> main)
Author: Your Name <your.mail@example.com>
Date:   Tue Oct 11 09:00:19 2022 +0700

    Edited readme.md

commit 8a10833a69bab33439dbc388ef0ce162906bd2f6
Author: Your Name <your.mail@example.com>
Date:   Tue Oct 11 08:21:47 2022 +0700

    Initial commit

$ git status

On branch main
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    code_of_conduct.md

no changes added to commit (use "git add" and/or "git commit -a")
```

15. Use the command "git restore" to discard the changes that you made.

```
$ git status

On branch main
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    code_of_conduct.md

no changes added to commit (use "git add" and/or "git commit -a")

$ ls

Mode                LastWriteTime         Length Name
----                -
-a---             11/10/2022   08:54           16  readme.md

$ git restore .

$ git status

On branch main
nothing to commit, working tree clean

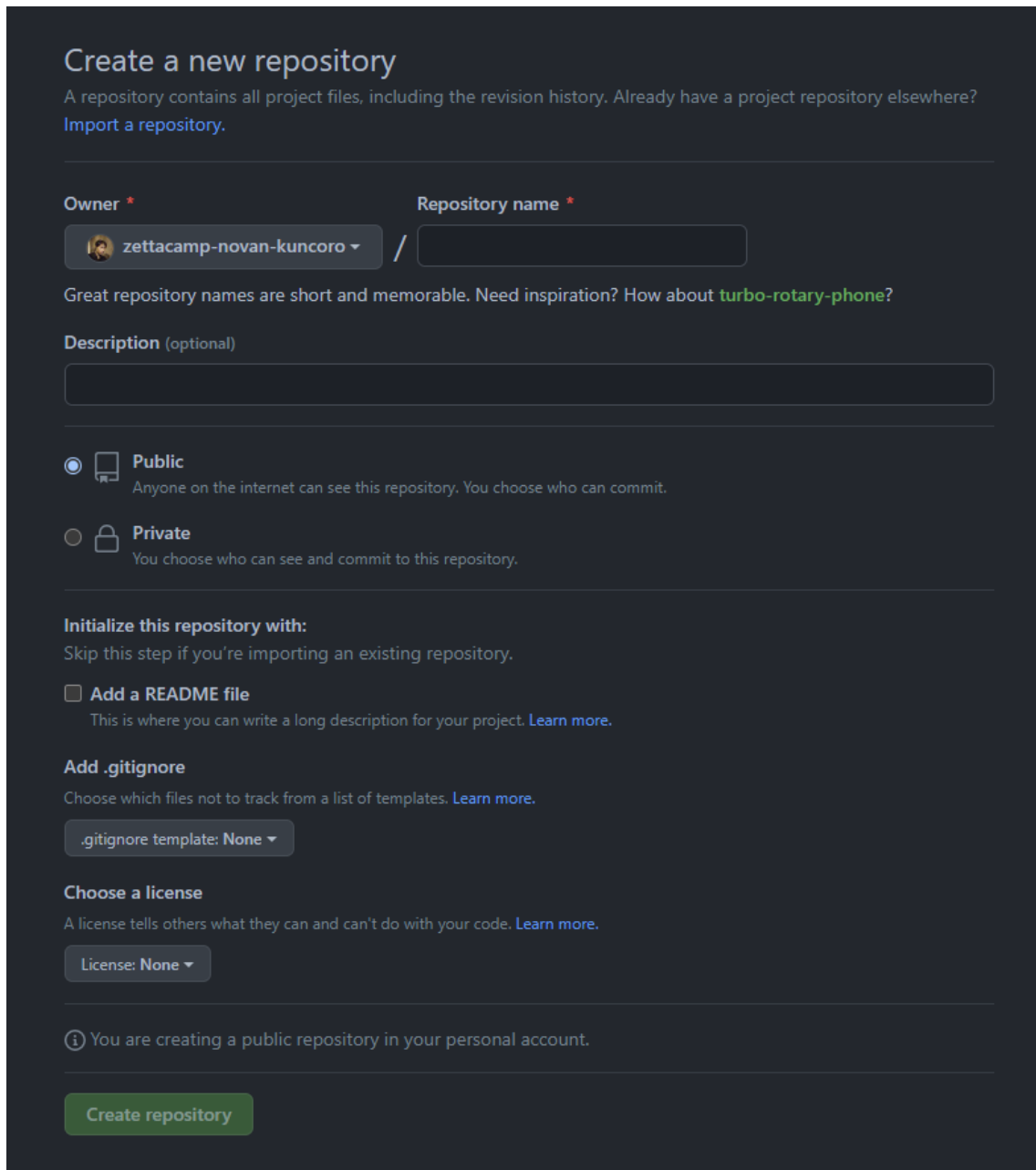
$ ls

Mode                LastWriteTime         Length Name
----                -
-a---             11/10/2022   09:03           0  code_of_conduct.md
-a---             11/10/2022   08:54           16  readme.md
```


Git Remote

1. Remote repository is a repository that is hosted on a network like the internet or a local office network. It can be accessed by multiple users hence allowing them to collaborate on the repository.
2. An example of a repository hosting platforms are GitHub, GitLab and BitBucket. We will use GitHub.
3. First, you need a GitHub account which you can create on <https://github.com>

4. Then you need to create a repository. Open the link <https://github.com/new>. This is how the form looks like




The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two main sections: 'Owner' and 'Repository name'. The 'Owner' dropdown is set to 'zettacamp-novan-kuncoro'. The 'Repository name' field is empty. A suggestion for a repository name, 'turbo-rotary-phone?', is shown below the name field. The 'Description (optional)' field is also empty. Under the 'Public/Private' section, the 'Public' option is selected. The 'Initialize this repository with:' section includes checkboxes for 'Add a README file' and 'Add .gitignore'. The 'Choose a license' section has a 'License: None' dropdown. At the bottom, there is a green 'Create repository' button. A note at the bottom states: 'You are creating a public repository in your personal account.'

Create a new repository


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


Owner * Repository name *

 zettacamp-novan-kuncoro /

Great repository names are short and memorable. Need inspiration? How about **turbo-rotary-phone?**

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:
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.](#)

.gitignore template: None ▾

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

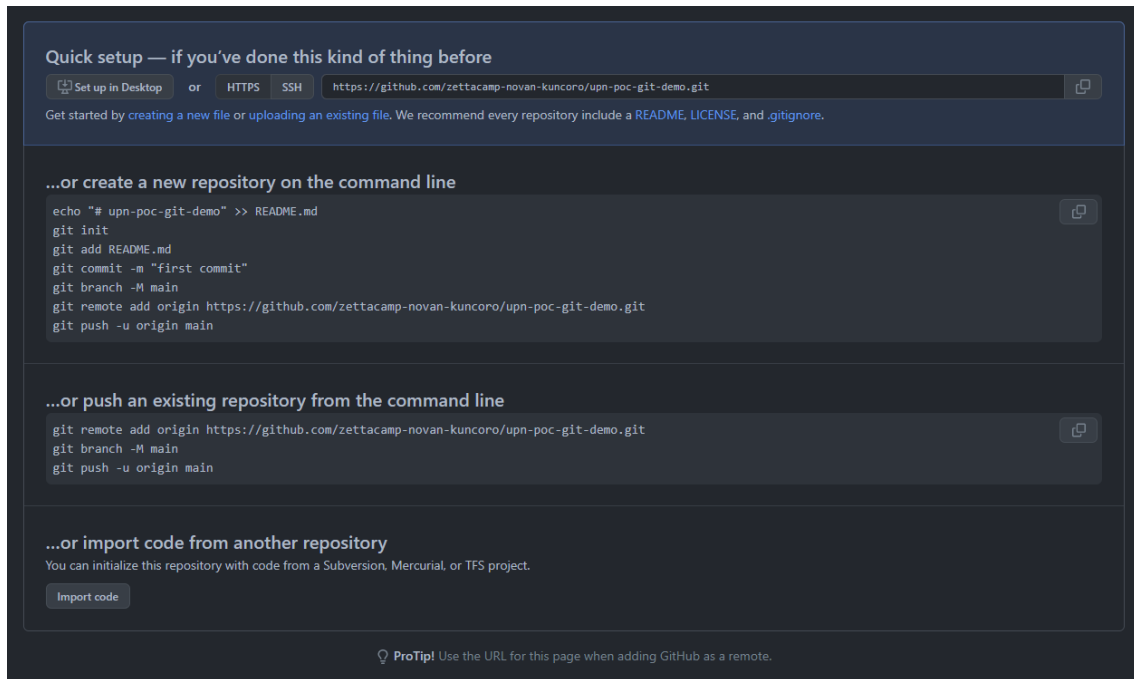
License: None ▾

 You are creating a public repository in your personal account.

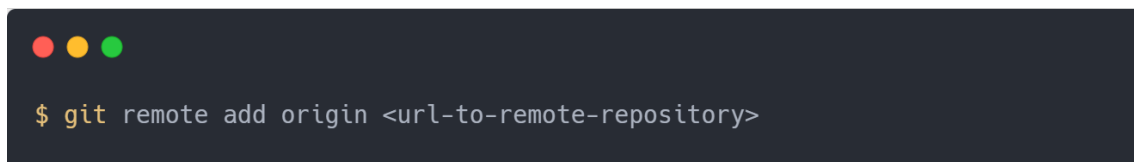
Create repository

5. Fill the repository name and description. Leaves other fields as empty and then click on the Create Repository button

6. This is what you will see after creating a repository on GitHub

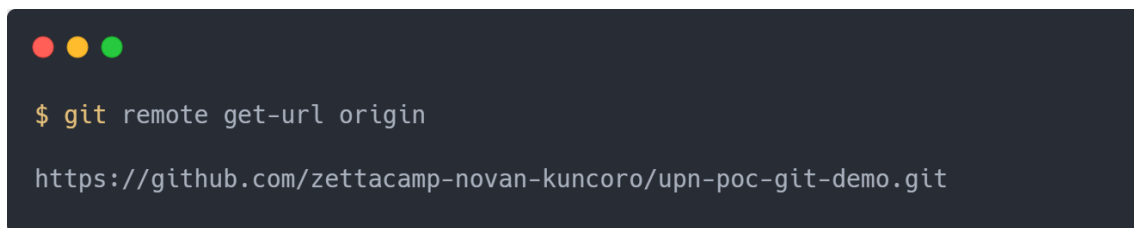


7. Let's link the local repository to the remote repository. You can achieve this using the command "git remote".



origin is the name of the remote. We can have multiple remotes in our local repository

8. We can check the added remote url using the command below. The url will be outputted in the line below.



9. Now let's upload our changes to the remote repository using "git push".

```
$ git push -u origin main

Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 527 bytes | 263.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/zettacamp-novan-kuncoro/upn-poc-git-demo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

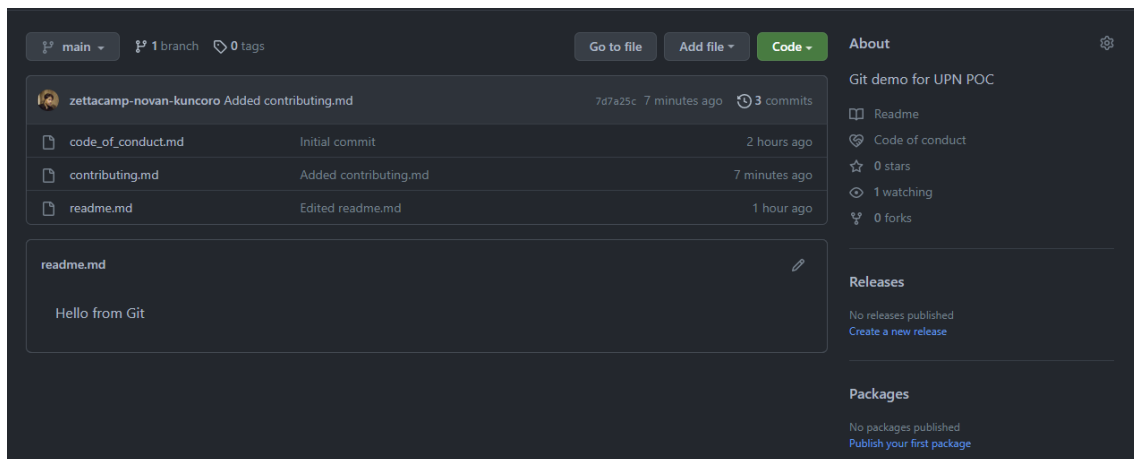
This command accepts an option `--set-upstream` or `-u`. `origin` is the remote repository upstream and `main` is the remote branch upstream. Once we set the upstream, we don't need to set it again on the next push.

10. Now let's create a new file with the name "contributing.md" with content "Hello from contributing.md"
11. Stage the change and commit with message "Added contributing.md"
12. Now push the changes from the local repository to the remote repository using "git push". The push succeeded without needing to use the `--set-upstream` option since we already provided it before.

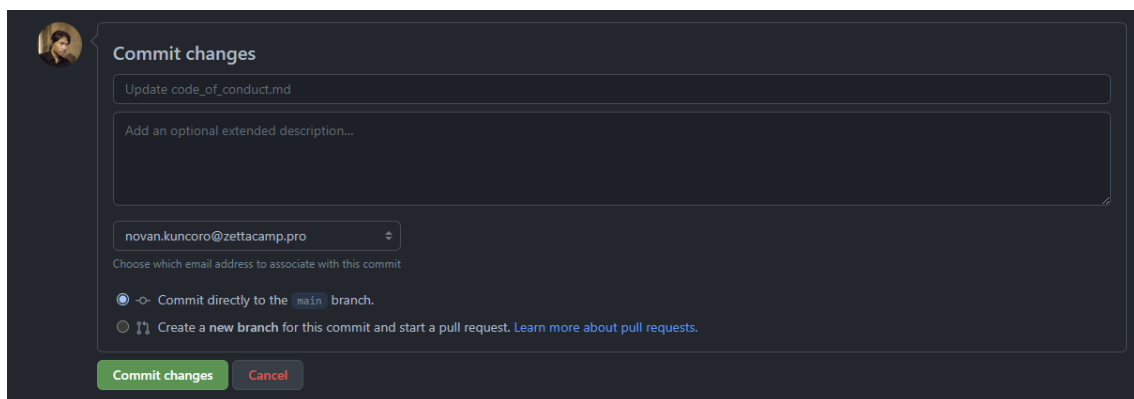
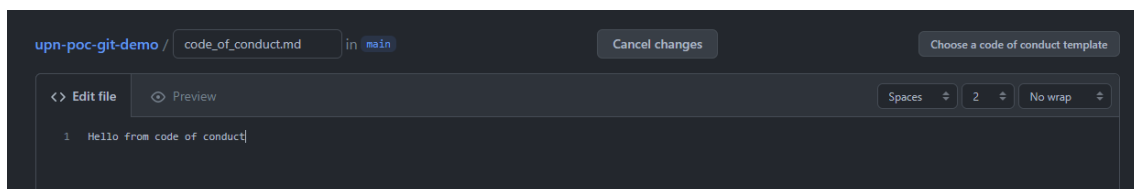
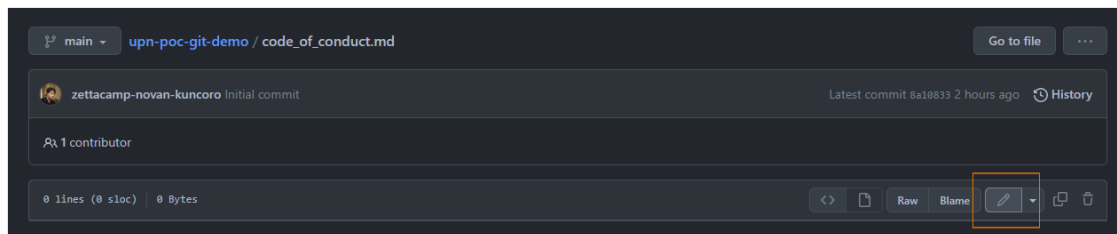
```
$ git push

Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 363 bytes | 363.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/zettacamp-novan-kuncoro/upn-poc-git-demo.git
 944bf06..7d7a25c  main -> main
```

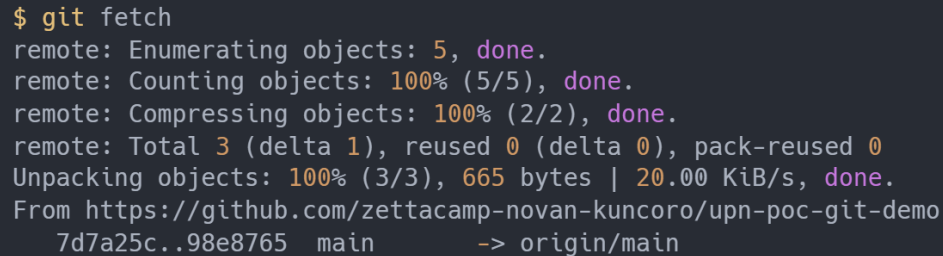
13. Check the remote repository on GitHub and you will find the contributing.md is present



14. Try edit code_of_conduct.md in GitHub. Just click on the file and look for the pencil icon button. Put the text “Hello from code of conduct” in the file and commit the changes

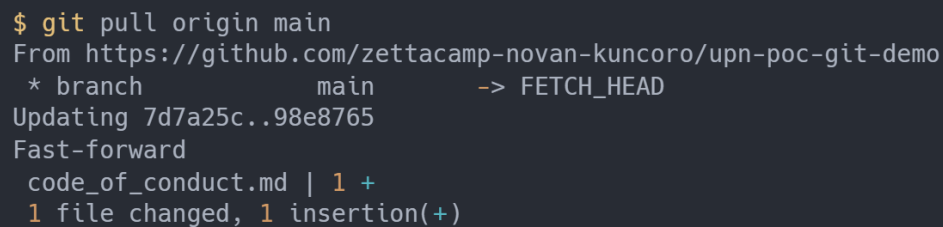


15. Back to the shell/terminal and run “git fetch” to fetch the information about what files are changed



```
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 665 bytes | 20.00 KiB/s, done.
From https://github.com/zettacamp-novan-kuncoro/upn-poc-git-demo
  7d7a25c..98e8765  main      -> origin/main
```

16. After git knows which files are changed, now we can run “git pull” to pull those changes from remote repository to our local repository



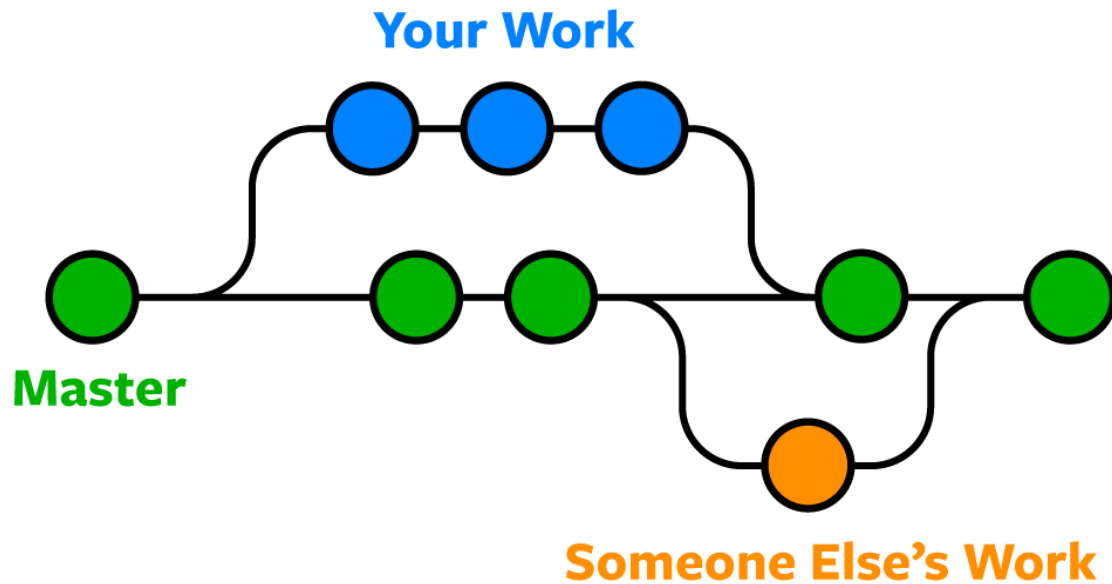
```
$ git pull origin main
From https://github.com/zettacamp-novan-kuncoro/upn-poc-git-demo
 * branch          main      -> FETCH_HEAD
Updating 7d7a25c..98e8765
Fast-forward
 code_of_conduct.md | 1 +
 1 file changed, 1 insertion(+)
```

The command above means to pull from a remote repository named “origin” and from a remote repository branch named “main”. So it will pull from <https://github.com/zettacamp-novan-kuncoro/upn-poc-git-demo.git/origin/main> to the local main branch

17. Now the remote repository and the local repository are in sync.

Git Branching

1. Git has a concept called branch, an independent line of development.



<https://www.nobledesktop.com/image/gitresources/git-branches-merge.png>

2. When you initialize a repository, you will start at a 'main' or 'master' branch depending on your machine's Git configuration. You can see the branch list using the `git branch` command.

```
$ git branch  
  
* main <-- this can be master depending on your machine
```

3. There are 2 ways to create a branch from your current branch. First using `git branch <branch-name>` and the second using `git checkout -b <branch-name>`

```
$ git branch feature/landing-page  
  
$ git branch  
  
feature/landing-page <-- the new branch that we just created  
* main <-- this can be master depending on your machine
```

4. The active branch will be automatically switched if you use `git checkout -b <branch-name>`

```
$ git checkout -b feature/login-page

Switched to a new branch 'feature/login-page'

$ git branch

feature/landing-page
* feature/login-page <-- now feature/login-page is active
main <-- this can be master depending on your machine
```

5. Use `git checkout <branch-name>` to switch between branches

```
$ git checkout feature/landing-page

Switched to a new branch 'feature/landing-page'

$ git branch

* feature/landing-page <-- now feature/landing-page is active
feature/login-page
main
```

6. Try adding an `index.html` file with the content below and commit them with message "Add index.html"

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Landing Page</title>
</head>
<body>
  <h1 style="text-align: center;">Hello from Landing Page</h1>
</body>
</html>
```


7. Now switch the active branch from 'feature/landing-page' to 'feature/login-page' and you can see the index.html file is not present in the branch 'feature/login-page'.

```
$ git checkout feature/login-page
Switched to branch 'feature/login-page'

$ ls
```

Mode	LastWriteTime	Length	Name
-a---	11/10/2022 10:29	28	code_of_conduct.md
-a---	11/10/2022 09:51	28	contributing.md
-a---	11/10/2022 08:54	16	readme.md

8. Add login.html in this branch with the content below and commit the file with message "Add login.html"

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>
</head>
<body>
  <h1>Hello from the login page</h1>
</body>
</html>
```

9. Switch the branch to main and you will see there are no login.html and index.html

```
$ git checkout main

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

$ ls
```

Mode		LastWriteTime		Length	Name
-a---	11/10/2022	10:29	28	□	code_of_conduct.md
-a---	11/10/2022	09:51	28	□	contributing.md
-a---	11/10/2022	08:54	16	□	readme.md

10. Let's bring the changes in 'feature/landing-page' and 'feature/login-page' to the 'main' branch. Remember that 'main' could be 'master' on your machine so don't panic if there is no 'main' branch but there is a 'master' branch. Use it.
11. To bring the changes to the 'main' branch, we can use `git merge <branch-name>`
12. Let's merge 'feature/landing-page' first. Make sure you are in the 'main' or 'master' branch and run `git merge feature/landing-page`.

```
$ git merge feature/landing-page

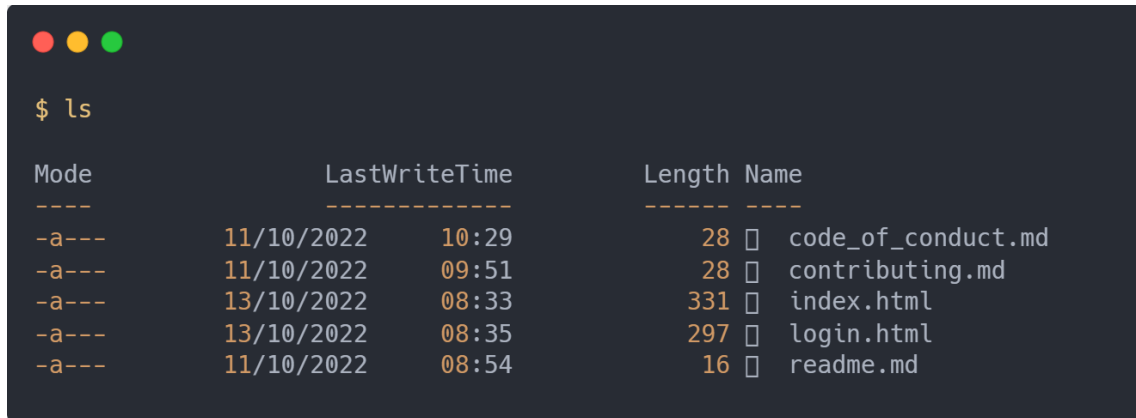
Updating 98e8765..4607821
Fast-forward
 index.html | 12 ++++++++
 1 file changed, 12 insertions(+)
 create mode 100644 index.html

$ ls
```

Mode		LastWriteTime		Length	Name
-a---	11/10/2022	10:29	28	□	code_of_conduct.md
-a---	11/10/2022	09:51	28	□	contributing.md
-a---	13/10/2022	08:33	331	□	index.html
-a---	11/10/2022	08:54	16	□	readme.md

Now you can see index.html in the 'main' or 'master' branch.

13. Next try merging 'feature/login-page' to the 'main' or 'master' branch. The end result should be like the image below. There should be 5 files in the 'main' or 'master' branch.



```
$ ls
```

Mode	LastWriteTime		Length	Name
----	-----		-----	----
-a---	11/10/2022	10:29	28	code_of_conduct.md
-a---	11/10/2022	09:51	28	contributing.md
-a---	13/10/2022	08:33	331	index.html
-a---	13/10/2022	08:35	297	login.html
-a---	11/10/2022	08:54	16	readme.md

What You Have Learnt

1. Basic of CLI
2. Git configuration in your machine
3. How to initialize a local and remote repository
4. How to stage and commit changes in local repository
5. How to push and pull changes from the remote repository to local repository
6. How to create a branch and merge two branches
7. Git commands
 - a. `git init`
 - b. `git add`
 - c. `git commit`
 - d. `git remote`
 - e. `git push`
 - f. `git fetch`
 - g. `git pull`
 - h. `git branch`
 - i. `git checkout`