Written by Thierry MEMEL Muhammad, Melrry Coding University Founder

Please follow me on Github: MELJOESTEIN (click on it)

GITHUB , GIT ... WHERE TO START?

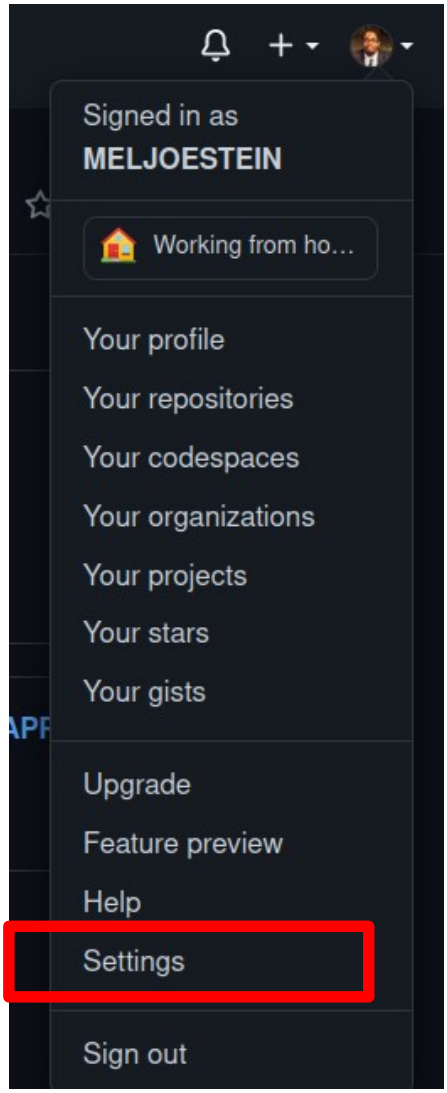
Roadmap

Step 0 - Create an account on Github [if you do not have one already]

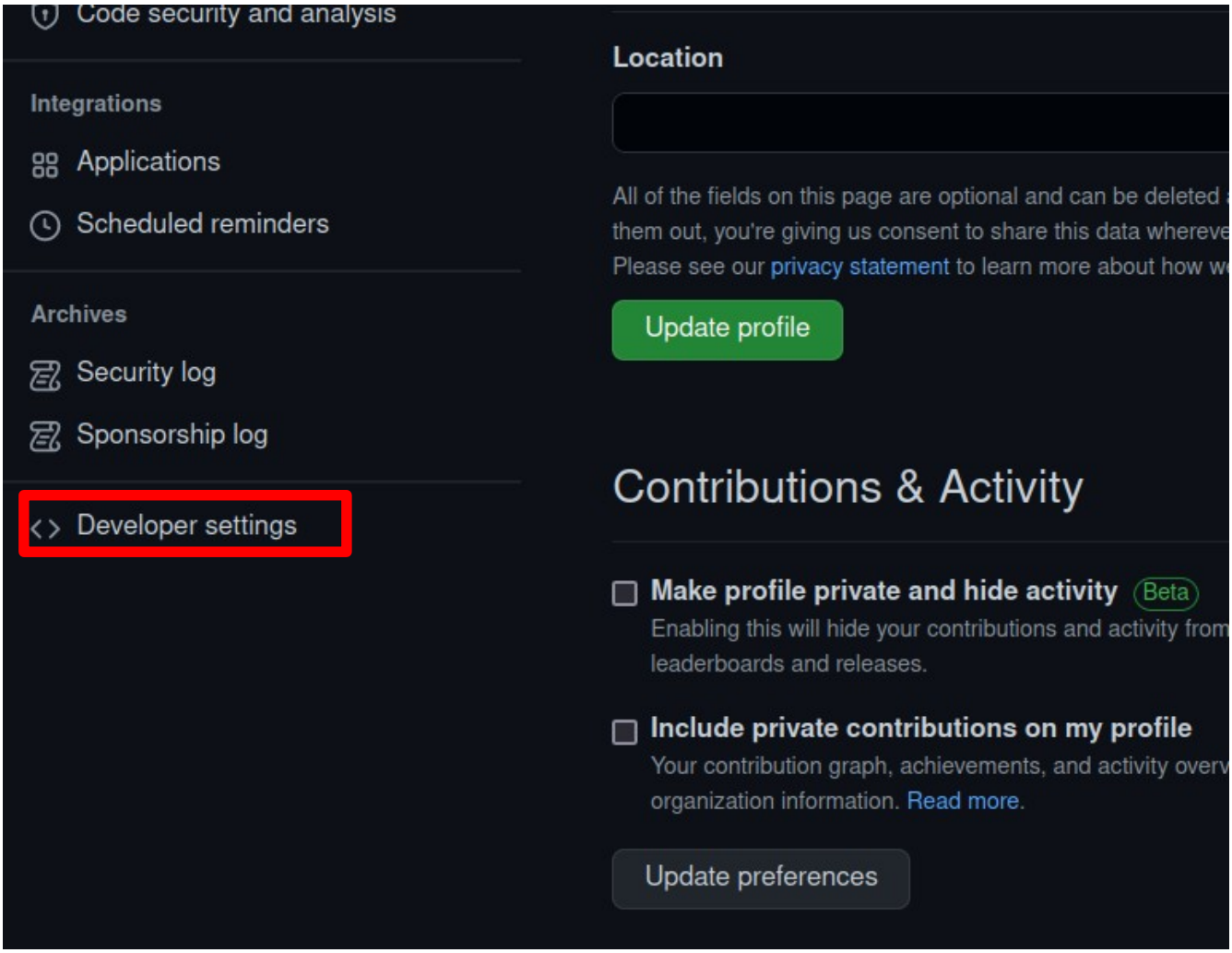
Step 1 - Create a Personal Access Token on Github

How to generate the token? Please follow the **screenshots**

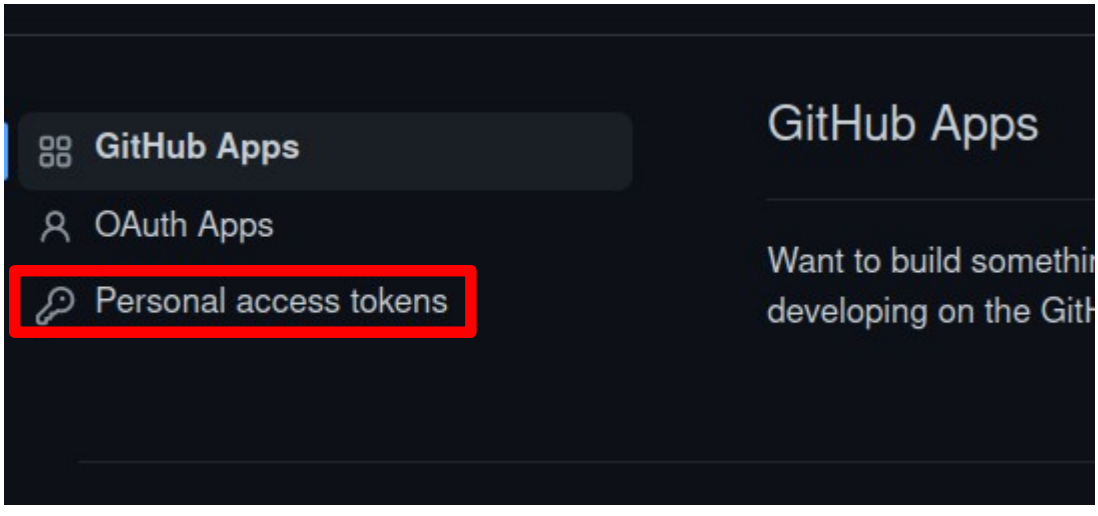
1.



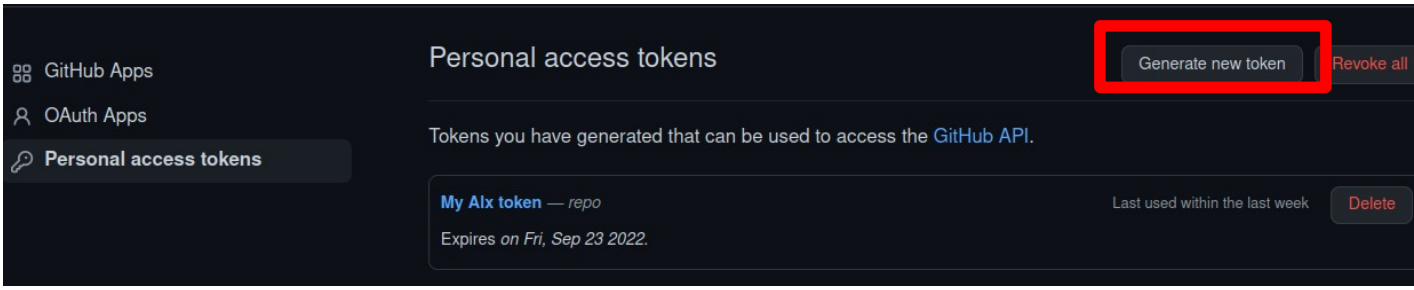
2.

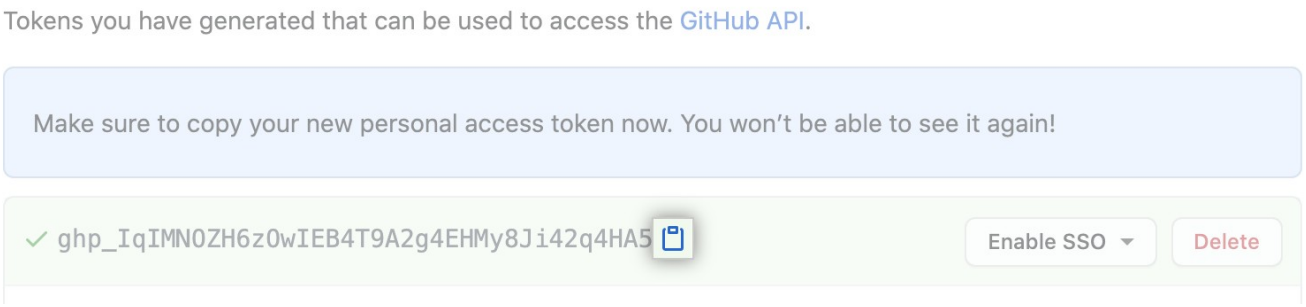

Code security and analysis
Integrations
Applications
Scheduled reminders
Archives
Security log
Sponsorship log
Developer settings
Location
All of the fields on this page are optional and can be deleted
them out, you're giving us consent to share this data whereve
Please see our privacy statement to learn more about how w
Update profile
Contributions & Activity
Make profile private and hide activity
Beta
Enabling this will hide your contributions and activity from
leaderboards and releases.
Include private contributions on my profile
Your contribution graph, achievements, and activity overv
organization information. Read more.
Update preferences

3.


GitHub Apps
OAuth Apps
Personal access tokens
GitHub Apps
Want to build somethi
developing on the Gith

4.





Step 3 - Create your first repository

Using the graphic interface on the Github website, create your first repository.

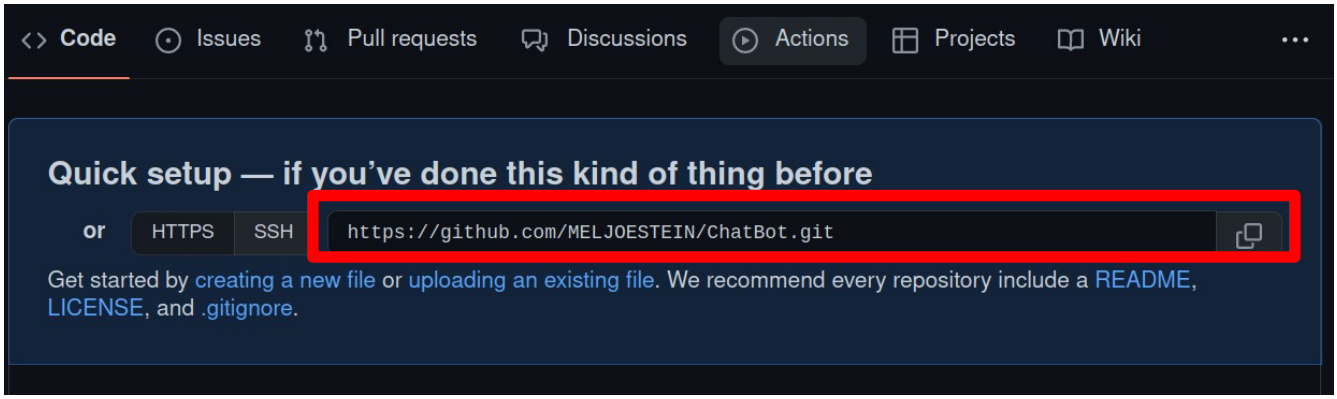
- Name: give a name of your repository
- Description: Give a description
- Public repo : Make it public or private depending on your need.
- No README, .gitignore, or license

Step 4 - Clone your repository

On the terminal, do the following:

- **Clone your repository**

Go to your **GitHub** GUI and copy your repository link that the following:




```
muhammad@muhammad-Latitude-E7240:~$ git clone
https://{YOUR_PERSONAL_TOKEN}@github.com/{YOUR_USERNAME}/ChatBot.git
Cloning into 'ChatBot.git  '...
warning: You appear to have cloned an empty repository.
```

Replace **{YOUR_PERSONAL_TOKEN}** with your token from step 1

Replace **{YOUR_USERNAME}** with your username from step 0 and 1

Step 5 - Create the README.md and push the modifications

- **Navigate to this new directory (*Your new directory here is your clone directory, in this tutorial we use :* ChatBot)**

```
muhammad@muhammad-Latitude-E7240:~$ cd ChatBot
muhammad@muhammad-Latitude-E7240:~ /ChatBot$
```

Create the file **`README.md`** with the content `'My first readme' using` **`touch: to create the file and echo: to write its content`**

- **Update your git identity**

```
muhammad@muhammad-Latitude-E7240:~ /ChatBot$ git config --global user.email
"you@example.com"

muhammad@muhammad-Latitude-E7240:~ /ChatBot$ git config --global user.name
"Your Name"
```

AFTER THE ABOVE STEPS,

Use the following commands all along your coding projects :

$ touch filename : To create the file

$ git add . : To add the file

$ git commit -m "Your commit message here"

$ git push : Push all your changes and modifications

Branches

Branches are an important part of working with Git. Any commits you make will be made on the branch you're currently "checked out" to. Use **git status** to see which branch that is.

When you're making changes to a Git repository, it's a best practice to push to a different branch first. This lets you compare changes before submitting a pull request and finally merging it.

This is especially crucial when working with other developers.

Your repository's main branch, which is regarded as the authoritative branch, is the only branch present by default. Now let's quickly go over how to create branches in Git.

// create a branch and switch to the branch

$ git checkout -b <branch-name>

// create a branch only

$ git branch <branch-name>

// Deletes the specified branch

$ git branch -d <branch-name>

The .gitignore file

Sometimes it may be a good idea to exclude files from being tracked with Git. This is typically done in a special file named **.gitignore** (You can customize it like the following: filename**.gitignore check some** template here

Use : $ touch filename**.gitignore to create it.**

Synchronize changes

Synchronize your local repository with the remote repository on your GitHub

// Downloads all history from the remote tracking branches

$ git fetch

// Combines remote tracking branch into current local branch

$ git merge

// Uploads all local branch commits to GitHub

$ git push

// Updates your current local working branch with all new commits from the corresponding remote branch on GitHub. git pull is a combination of git fetch and git merge

$ git pull

Make changes

//Lists version history for the current branch

```
$ git log
```

//Lists version history for a file, including rename

```
$ git log –follow [file]
```

// Shows content differences between two branches

```
$ git diff [first-branch]...[second-branch]
```

// Outputs metadata and content changes of the specified commit

```
$ git show [commit]
```

Redo commits

Erase mistakes and craft replacement history

// Undoes all commits after [commit], preserving changes locally

```
$ git reset [commit]
```

// Discards all history and changes back to the specified commit

```
$ git reset --hard [commit]
```

PLEASE MEMORIZE ALL THESE COMMANDS BY PRACTICING

 # CAUTION! Changing history can have nasty side effects. If you need to change commits that exist on GitHub (the remote), proceed with caution. If you need help, reach out at

github.community or contact support.

#GLOSSARY

- **git**: an open source, distributed version-control system
- **GitHub**: a platform for hosting and collaborating on Git repositories
- **commit**: a Git object, a snapshot of your entire repository compressed into a SHA
- **branch**: a lightweight movable pointer to a commit
- **clone**: a local version of a repository, including all commits and branches

- **remote**: a common repository on GitHub that all team member use to exchange their changes
- **fork**: a copy of a repository on GitHub owned by a different user
- **pull** request: a place to compare and discuss the differences introduced on a branch with reviews, comments, integrated tests, and more
- **HEAD**: representing your current working directory, the HEAD pointer can be moved to different branches, tags, or commits when using `git checkout`