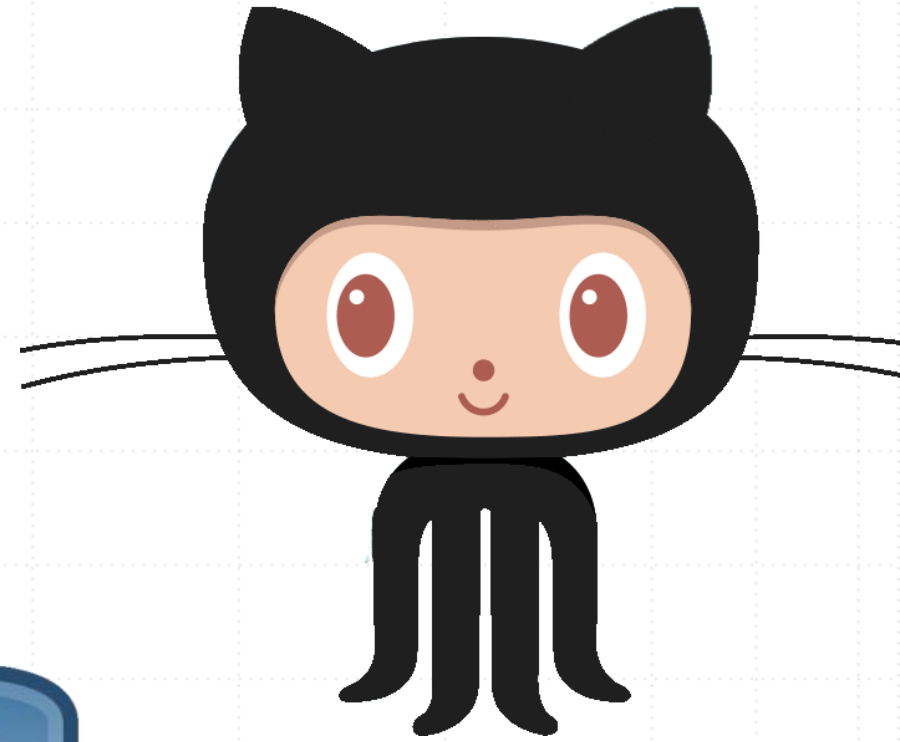


# Introduction To Packages, Modules & Version Control(Git, GitHub)

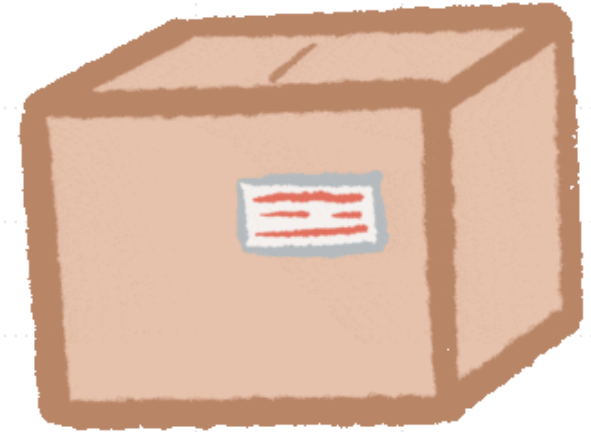




# Outline

- Introduction To Packages and Modules.
- Introduction To Version Control.
- Introduction To Git.
- Introduction To GitHub.

# Introduction To Packages & Modules





# Outline:

- The Importance of Packages and Modules
- What is a Module, Package, Library, & Framework.
- How To use Modules & Packages in Python.

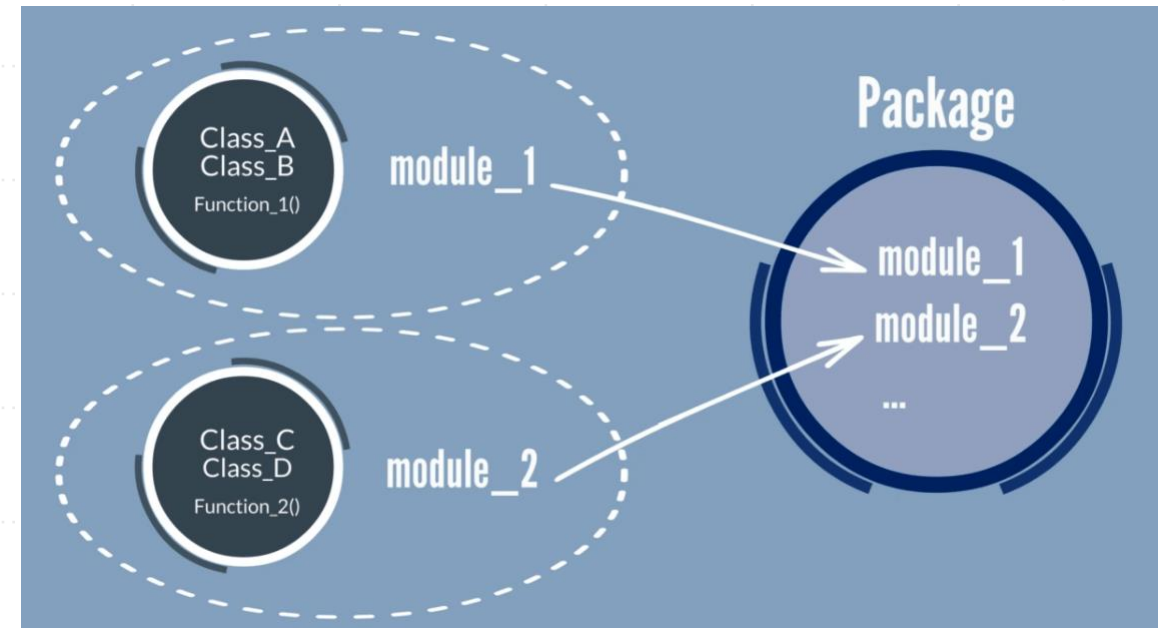
# Why we need packages and modules:

- Modular programming:
  - is the process of breaking a large programming task into separate, smaller, more manageable subtasks or modules.
- Individual modules can then be combined like Legos to create a larger app.
- There are several advantages to modularizing code:
  - ✓ Simplicity
  - ✓ Maintainability
  - ✓ Reusability
  - ✓ Scoping
- Functions, modules and packages are all constructs in Python that promote code modularization.



# What is a Module, Package, Library, & Framework

- Module is a file which contains various Python functions and global variables.
- It is simply just .py extension file which has python executable code.
- Package is a collection of modules.
- Library is a collection of packages.
- Framework is a collection of libraries.



# Naming convention for each:

01

## **Module/Library/Framework**

- have short, all-lowercase names.
- Underscores can be used in the module name if it improves readability.

02

**Python packages should also have short, all-lowercase names.**

03

**Class names should normally use the CapWords convention.**

# To install them in python:

- Use PIP to manage your packages:
- PIP is a package manager for Python packages, or modules.
- If you have Python version 3.4 or later, PIP is included by default.
- Open cmd and type: “pip --version” to check pip version.
- If you don't have pip, go to <https://pypi.org/project/pip/> and install it.
- To install a package, go to cmd and type “pip install <<package name>>”

```
C:\Users\marya>pip --version
pip 20.2.4 from C:\Users\marya\anaconda3\lib\site-packages\pip (python 3.8)

C:\Users\marya>pip install numpy
Requirement already satisfied: numpy in c:\users\marya\anaconda3\lib\site-packages (1.19.2)
```





## Which of the following is not an advantage of using modules?

A) Provides a means of reuse of program code.

B) Provides a means of dividing up tasks.

C) Provides a means of reducing the size of the program.

D) Makes it easier to maintain code.

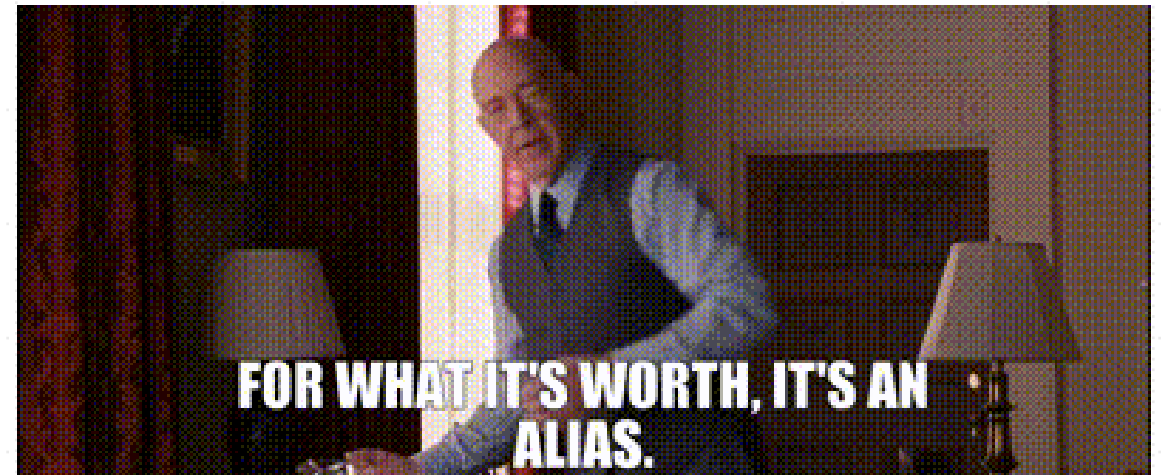
 Multiple Choice

# How To use Them:

- Use Import statement:

- Import <module\_name> : imports all content of a module.
- Import <module\_name> as <alt\_name> : imports all content of a module and give it an alias.
- From <module\_name> import <name(s)> : imports individual object from a module.
- From <module\_name> import <name> as <alt\_name> : imports individual object from a module and gives them an alias to use in code.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import defaultdict
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score, r2_score
from sklearn.ensemble import RandomForestRegressor, AdaBoostClassifier
from sklearn.linear_model import LinearRegression
```



# Introduction To Version Control Systems VCS



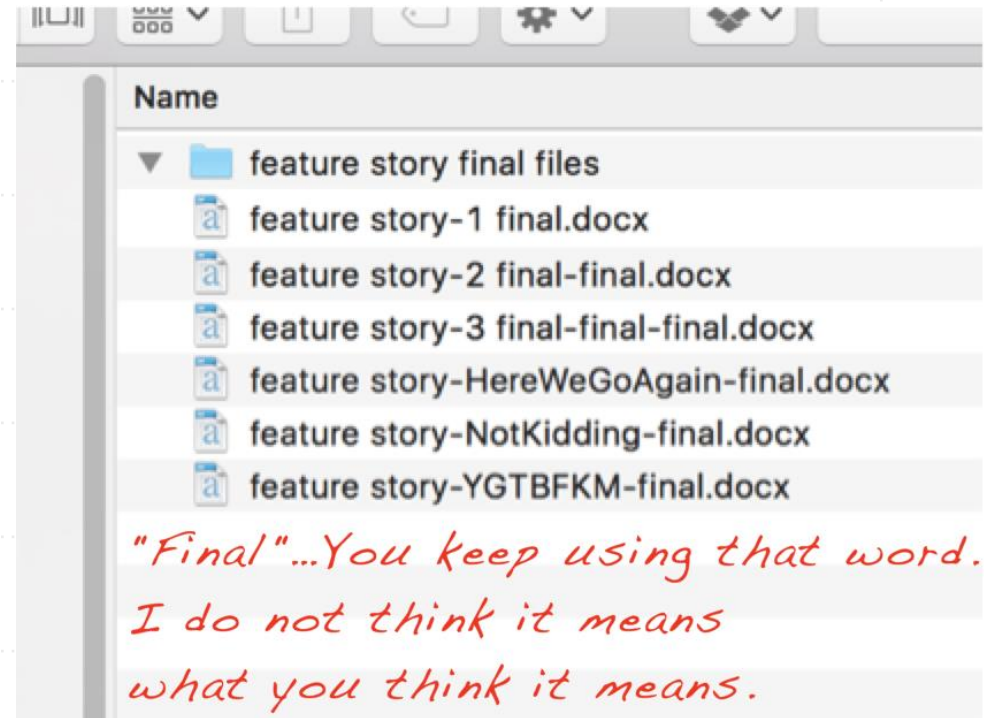


# Outline:

- Why we need Version Control.
- What is Version Control & Who Can Benefit From using It.
- Types of Version Control:
- Introduction To Git & GitHub.

# When Working on a Project:

- Many people copy files into another directory (perhaps a time-stamped directory, if they're clever).
- This approach is very common because it is so simple, but it is error prone.
- It is easy to forget which directory you're in and accidentally write to the wrong file or copy over files you don't mean to.
- It's not efficient when working with others on the same project.





# The Importance of (VCS)

It allows you to revert selected files back to a previous state.

Revert the entire project back to a previous state.

Compare changes over time.

See who last modified something that might be causing a problem.

See who introduced an issue and when, and more.

Using a VCS means that if you screw things up or lose files, you can easily recover.

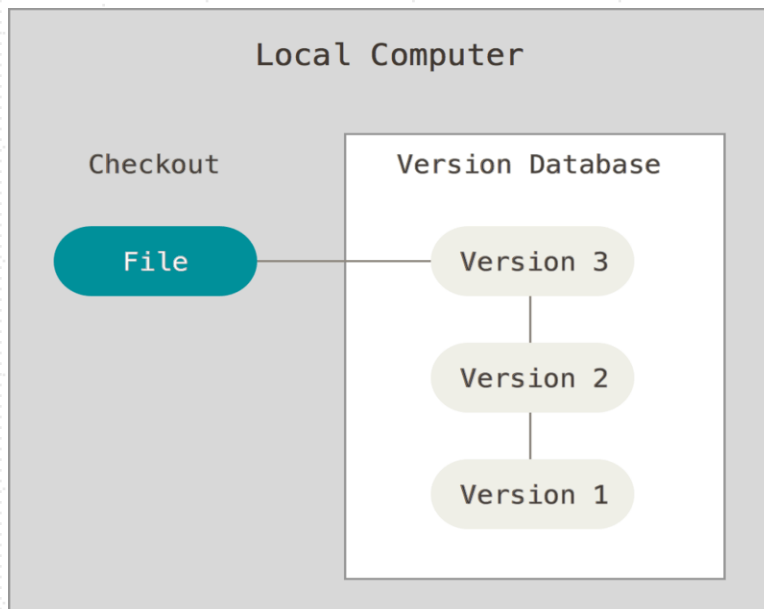
# VCS is:

- System that records changes to a file or set of files over time so that you can recall specific versions later.
- Graphic, web designer, programmers, data scientist....etc. who want to keep every version of an image, layout, or a project or anyone that works with computer files can benefit from using Version Control.

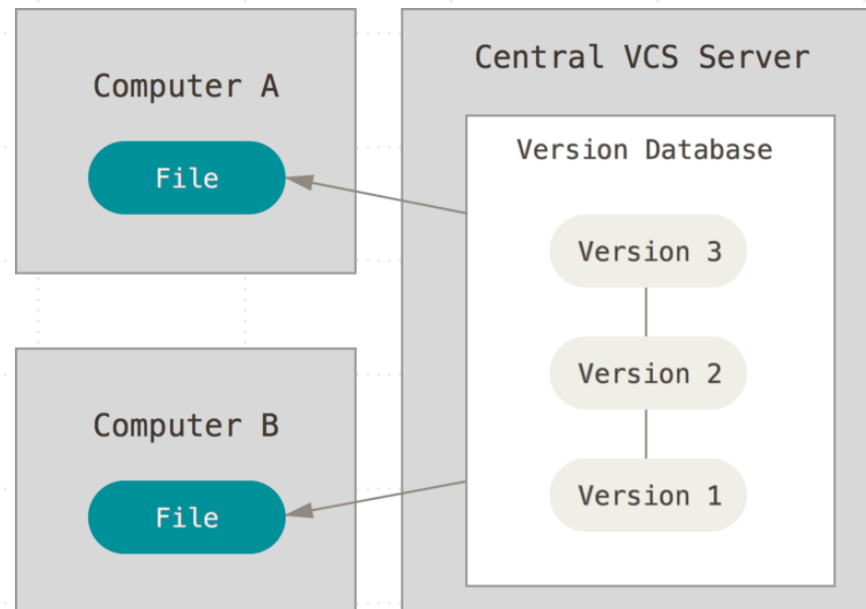


# There are 3 types of VCS:

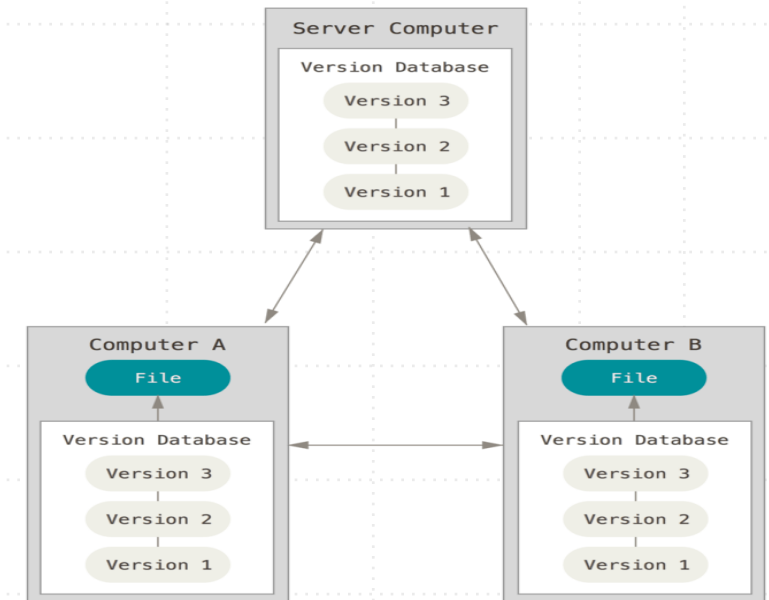
## Local VCS



## Centralized VCS



## Distributed VCS







# What is full form of VCS?

- A. Version Configuration System.
- B. Version Consolidated Solutions.
- C. Version Configuration Solutions.
- D. Version Control System.



Multiple Choice

# Introduction To Git





# Outline:

- What is Git.
- Why use Git.
- Installation and basic commands.

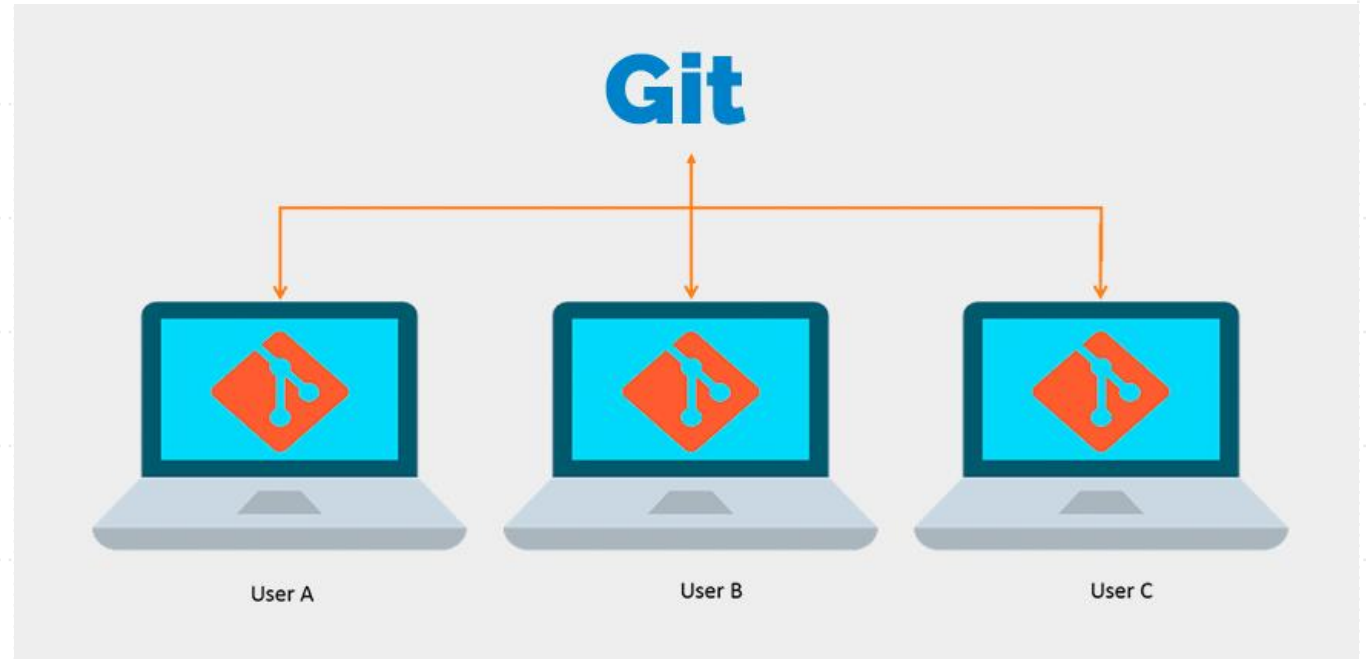
# What is Git?

- Git is a popular version control system.
- It is used for:
  - Tracking code changes
  - Tracking who made changes
  - Coding collaboration



# Why Git?

- Over 70% of developers use Git!
- Developers can work together from anywhere in the world.
- Developers can see the full history of the project.
- Developers can revert to earlier versions of a project.



# Installation:

- First, check if Git installed:
  - Open a Terminal/Command Prompt and try running
  - If it outputs a version number, skip ahead.
- If it didn't output a version number download it from:
  - <https://www.git-scm.com/>

```
git --version  
git version 2.30.2.windows.1
```



# Configure Git:

- Now, tell Git who you are.
- This is crucial for version control systems, because each Git commit relies on it.
- You can specify Git configuration settings with the “git config” command

```
$ git config --global user.name "John Doe"  
$ git config --global user.email johndoe@example.com
```

# Creating Git Folder:

- You can Create new folder with “mkdir” command followed by the folder name.
- “cd” command changes the current working directory (folder).

```
mkdir myproject  
cd myproject
```



# Initialize Git:

- After navigating to the correct folder.
- Initialize Git on that folder with “git init” command to create your first repository.



```
git init
```

```
Initialized empty Git repository in /Users/user/myproject/.git/
```

Note: A software repository, or “repo” for short, is a storage location for software projects

# New Files:

- Create any kind of file then added it to the repo folder (here it is an html file called index).
- Use “ls” Command to list the files in the directory(folder).
- Use “git status” command to see if the new file is a part of your repo.

```
ls  
index.html
```


```
git status  
On branch master
```

```
No commits yet
```

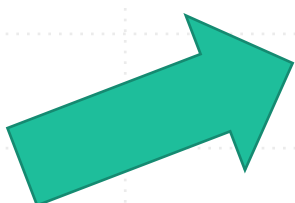
```
Untracked files:  
(use "git add ..." to include in what will be committed)  
index.html
```

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


Note: Git is aware of the file but has not added it the repo we need to “Stage it” as it is “Untracked file”.



There are 2 possible states for a file in a Git repository:



Tracked: files that Git knows about and are added to the repository.



Untracked: files that are in your working directory, but not added to the repository

Note: Files first added to an empty repository are all untracked

# Git Staging:

- One of the core functions of Git is the concept of staging files.
- Staging files mean:
  - files that are ready to be **committed** to a repository when you reach a milestone or finish a task.
- To stage a file:
  - Use “git add ‘filename’ “ command to add one file.
  - Use “git add - -all” command to add more than one file at once.
  - Check the status.

```
git add index.html
```

```
git add --all
```

```
git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached ..." to unstage)
```

```
new file: index.html
```

# Git Commit:

- After staging we need to commit our changes
- A commit is a point in the project you can go back to "save point".
- You can add a message to each commit
- By adding clear messages to each commit, it is easy for yourself (and others) to see what has changed and when.
  - Use "git commit -m "message"" command to commit after staging..

```
git commit -m "First release of Hello World!"
```

```
[master (root-commit) 221ec6e] First release of Hello World!
```

```
3 files changed, 26 insertions(+)
```

```
create mode 100644 README.md
```

```
create mode 100644 bluestyle.css
```

```
create mode 100644 index.html
```

# Other useful command:

- “git log” to view the history of commits for a repository.
- “git command name – help” to see all the available options for the specific command
- “git help –all” to see all possible commands.

```
git log
```

```
commit 09f4acd3f8836b7f6fc44ad9e012f82faf861803 (HEAD -> master)
```

```
Author: w3schools-test
```

```
Date:   Fri Mar 26 09:35:54 2021 +0100
```

```
    Updated index.html with a new line
```

```
git commit -help
```

```
usage: git commit [] [--] ...
```

```
-q, --quiet          suppress summary after successful commit
```

```
-v, --verbose        show diff in commit message template
```



# Git Branches

- In Git, a branch is a new/separate version of the main repository.
- Branches allow you to work on different parts of a project without impacting the (main/master) default branch.
- With Git, you can switch between branches and edit different projects without them interfering with each other.
- Use “git checkout -b <<branch\_name>>” to switch to it from your current branch.
- You can now repeat previous steps and commit with the new branch



```
git checkout -b emergency-fix
```

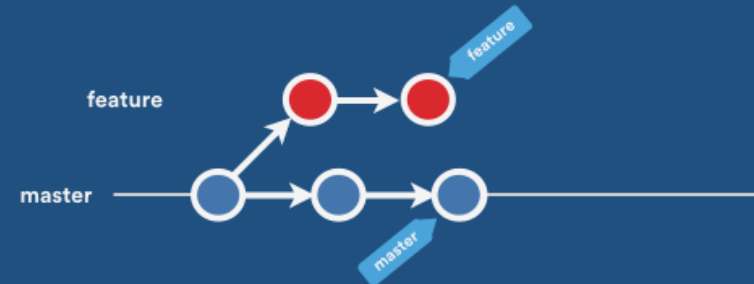
```
Switched to a new branch 'emergency-fix'
```

# Git Branch Merge:

- We need to merge branches so that all changes done aren't done to original branch(can't be viewed by all other branches)
- Merging is when we combine project branches.

## What is a merge?

A process that unifies the work done in two branches





# Git Branch Merge:

- Before merging two branches:
  - Make sure that you did at least a commit on both.
  - Use command “git branch -a” to see all branches viewed by git.
- Now you can merge both by:
  - Change to the master branch (the branch you want to merge to)
  - Use command “git merge <<branch-name>>” to merge them.
  - If there is a conflict between 2 branches which you are merging to the master branch merge will fail.

Note : when working with git it is better to work on a separate branch then merge to the master branch.



```
git merge emergency-fix
Updating 09f4acd..dfa79db
Fast-forward
 index.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

# Which of the following commands used to return to the master branch?

- A. git checkout origin
- B. git checkout -b master
- C. git checkout master
- D. git checkout branche.

 Multiple Choice

# Introduction to GitHub





# Outline:

- Why People use GitHub.
- What Is GitHub.
- How To Use it.

# What Is GitHub & Why People use GitHub:

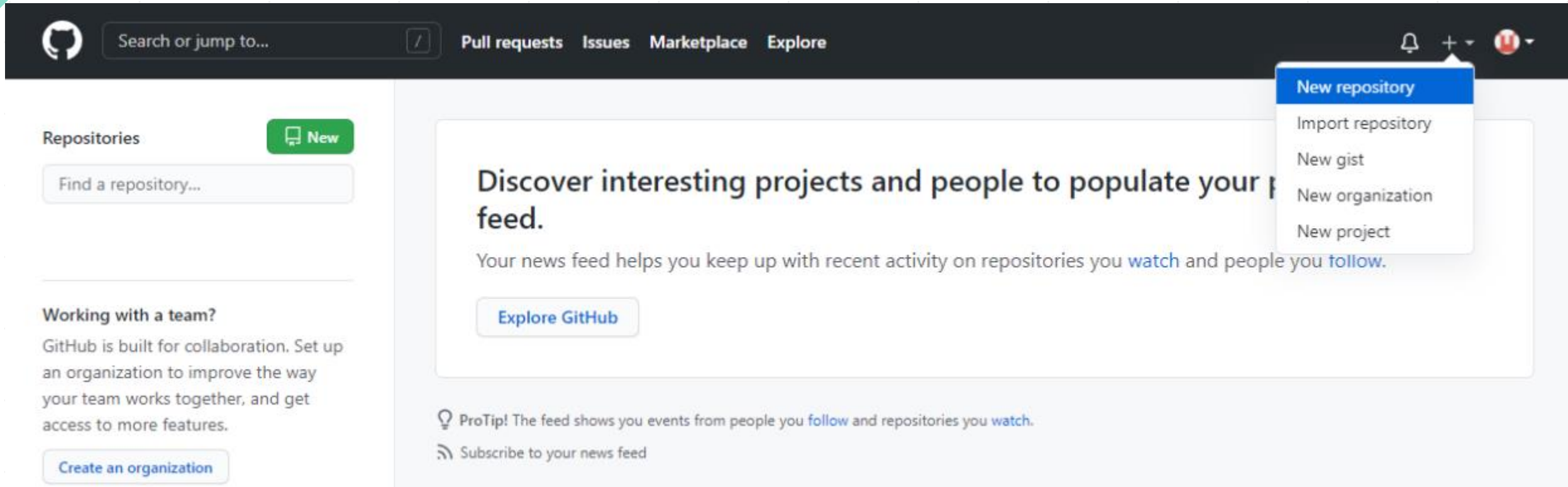
- GitHub is a web-based interface that uses Git.
- Allows for real-time collaboration.
- GitHub encourages teams to work together to build and edit their shared projects.
- GitHub allows multiple developers to work on a single project at the same time.
- Reduces the risk of duplicative or conflicting work and can help decrease production time.
- With GitHub, developers can build code, track changes, and innovate solutions to problems that might arise during the development process simultaneously.

Go to : <https://github.com/> &  
sign up to GitHub world.



# How to use it:


- After you have made a GitHub account, sign in, and create a new Repo.




# How to use it:

- And fill in the relevant details, & choose public:

PUBLIC

 hubot

 / 


hello-world 

Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.


**Description** (optional)

Just another repository

---

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.


---

☒ **Initialize this repository with a README**

This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** ▾

 | 

Add a license: **None** ▾ 

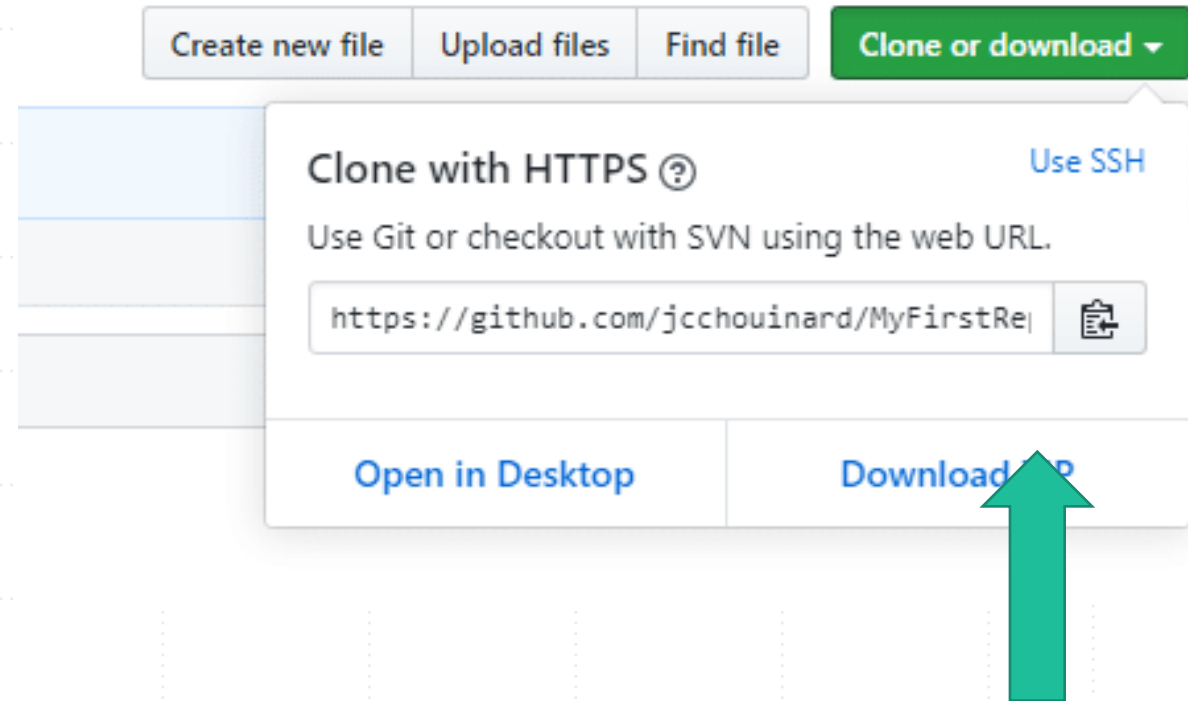
---

Create repository



# How to use it:

- After setting up a **local Git repo**, we are going to **push** that to GitHub.
- Copy the URL, or click the clipboard pointed to in the image.
- Paste link the following command:
  - “git remote add origin <<URL>>”
  - This command specifies that you are adding a remote repository, with the specified URL, as an origin to your local Git repo.



```
git remote add origin https://github.com/MaryamGKK/hello.git
```

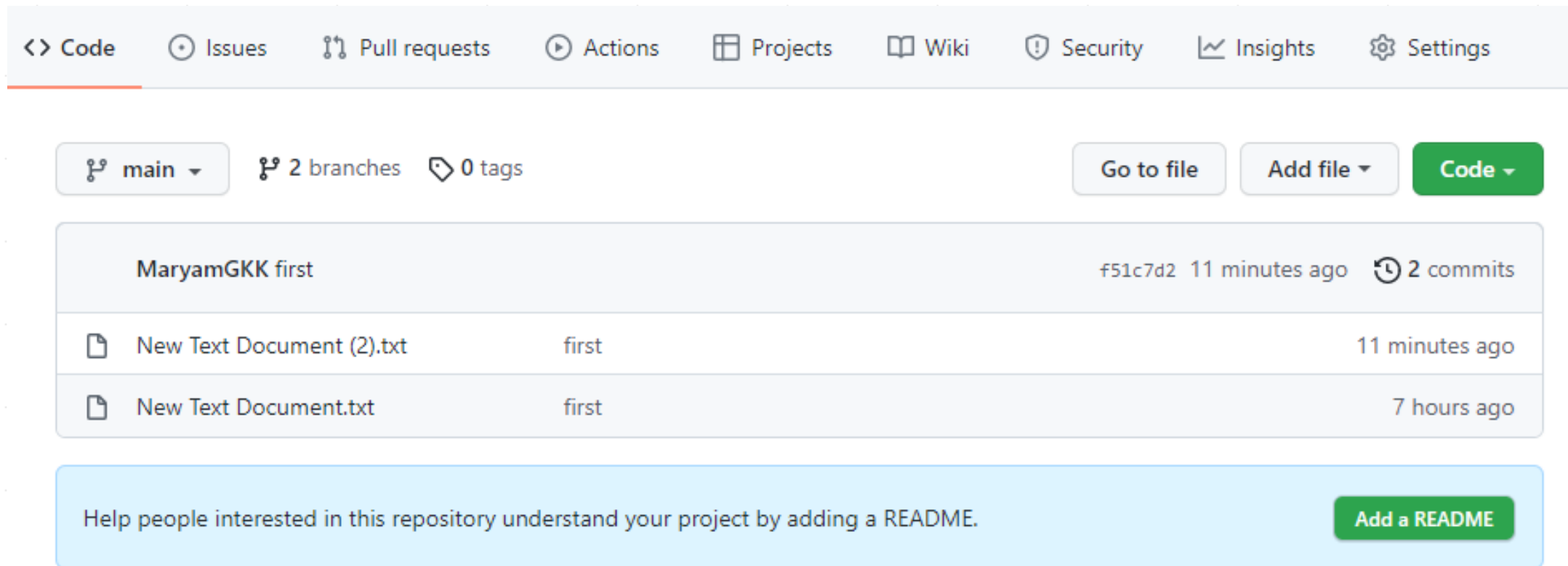
# How to use it:

- Now we are going to push our master branch to the origin URL.
- Then set it as the default remote branch:
  - Use command “git push --set-upstream origin main”

```
marya@DESKTOP-5Q0TAAH MINGW64 ~/OneDrive/ /New folder (main)
$ git push -f origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 260 bytes | 260.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MaryamGKK/hello.git
+ 0660fd6...f51c7d2 main -> main (forced update)
```

# How to use it:

- Now, go back into GitHub and see that the repository has been updated:



The screenshot shows the GitHub interface for a repository named 'first' by user 'MaryamGKK'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, the repository's branch status is shown as 'main' with 2 branches and 0 tags. Action buttons for 'Go to file', 'Add file', and 'Code' are present. The commit history table shows two commits: 'New Text Document (2).txt' and 'New Text Document.txt', both on the 'first' branch. A light blue banner at the bottom encourages adding a README file.

MaryamGKK first		f51c7d2 11 minutes ago	🕒 2 commits
📄 New Text Document (2).txt	first	11 minutes ago	
📄 New Text Document.txt	first	7 hours ago	

Help people interested in this repository understand your project by adding a README. [Add a README](#)



# GitHub is:

# Project 3 - Thanos.py >\_



# Introduction to GitKraken



**RELEASE THE  
KRAKEN**



# Outline:

- What Is GitKraken?
- How To Use it?

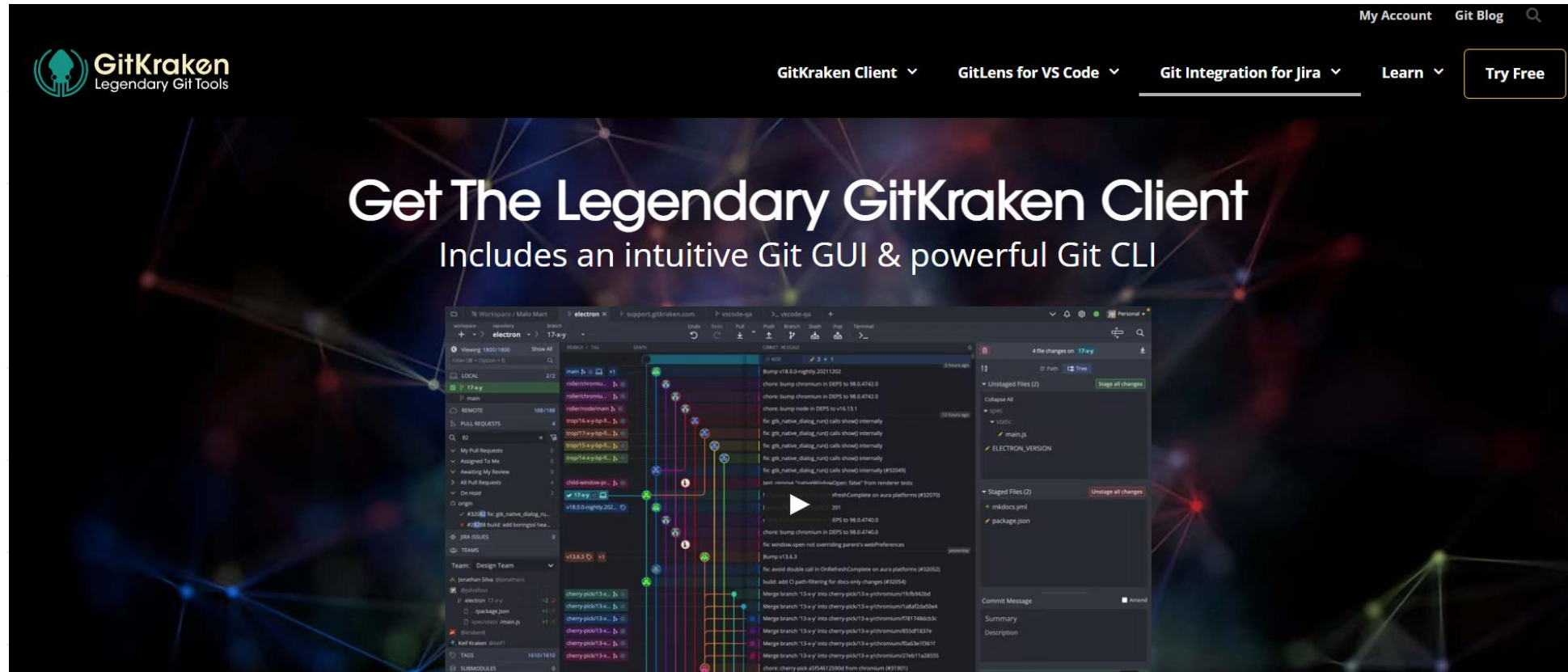
# What is GitKraken?

- It is a GUI Git client that facilitates efficient and reliable usage of Git on a desktop and offer most of the command line operations.
- It can be easily integrated with your GitHub, Bitbucket, and GitLab accounts.
- Most of the actions done by Git can be done by GitKraken with just one click like undo, redo, commit, etc..



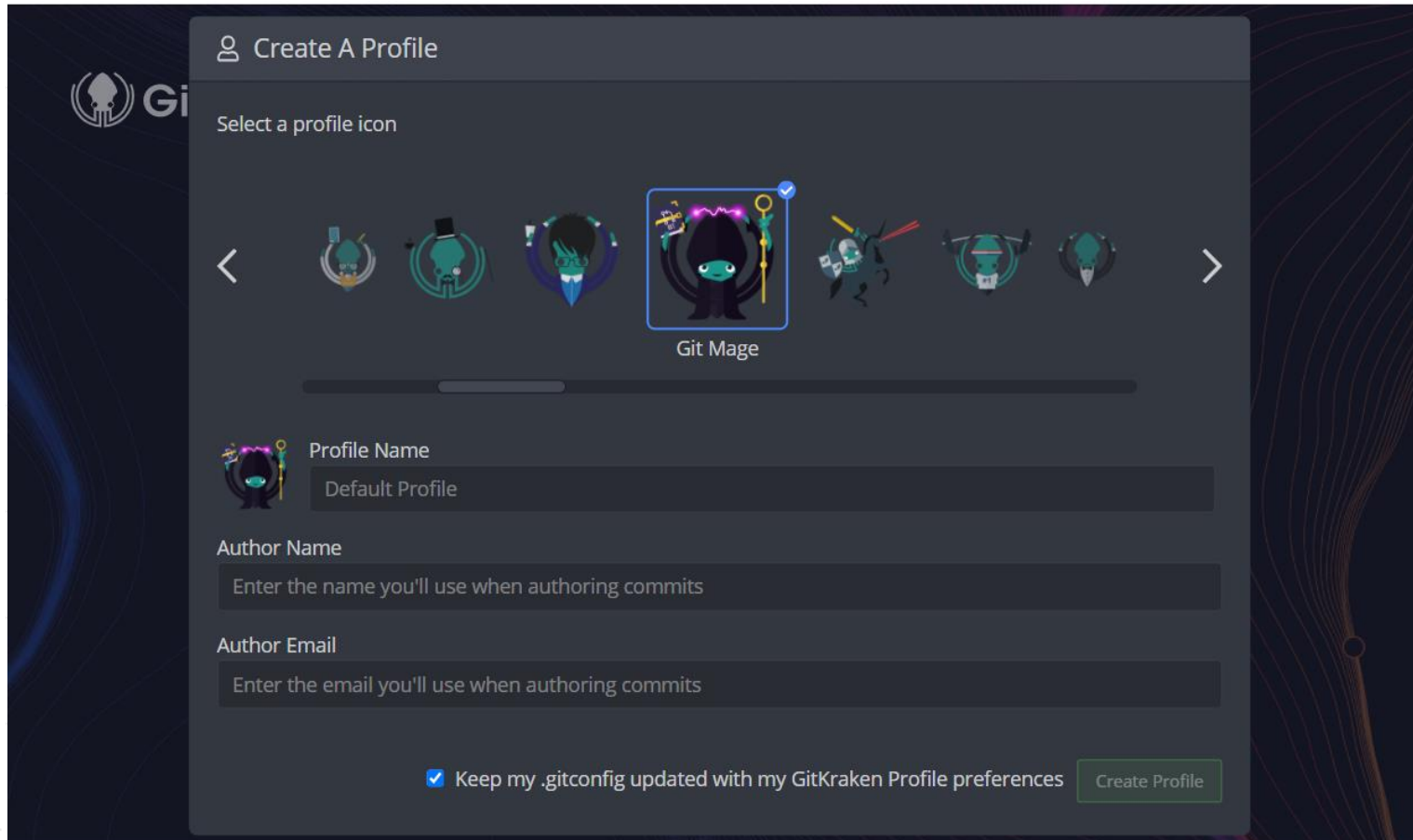
# How to use it?

- Download Gitkraken from its website ( <https://www.gitkraken.com/git-client/try-free> )
- Press (GitKraken Client) then (Download for free)



# How to use it?

- After the setup, you could create your profile



Create A Profile

Select a profile icon

< [Icons] >

Git Mage

Profile Name  
Default Profile

Author Name  
Enter the name you'll use when authoring commits

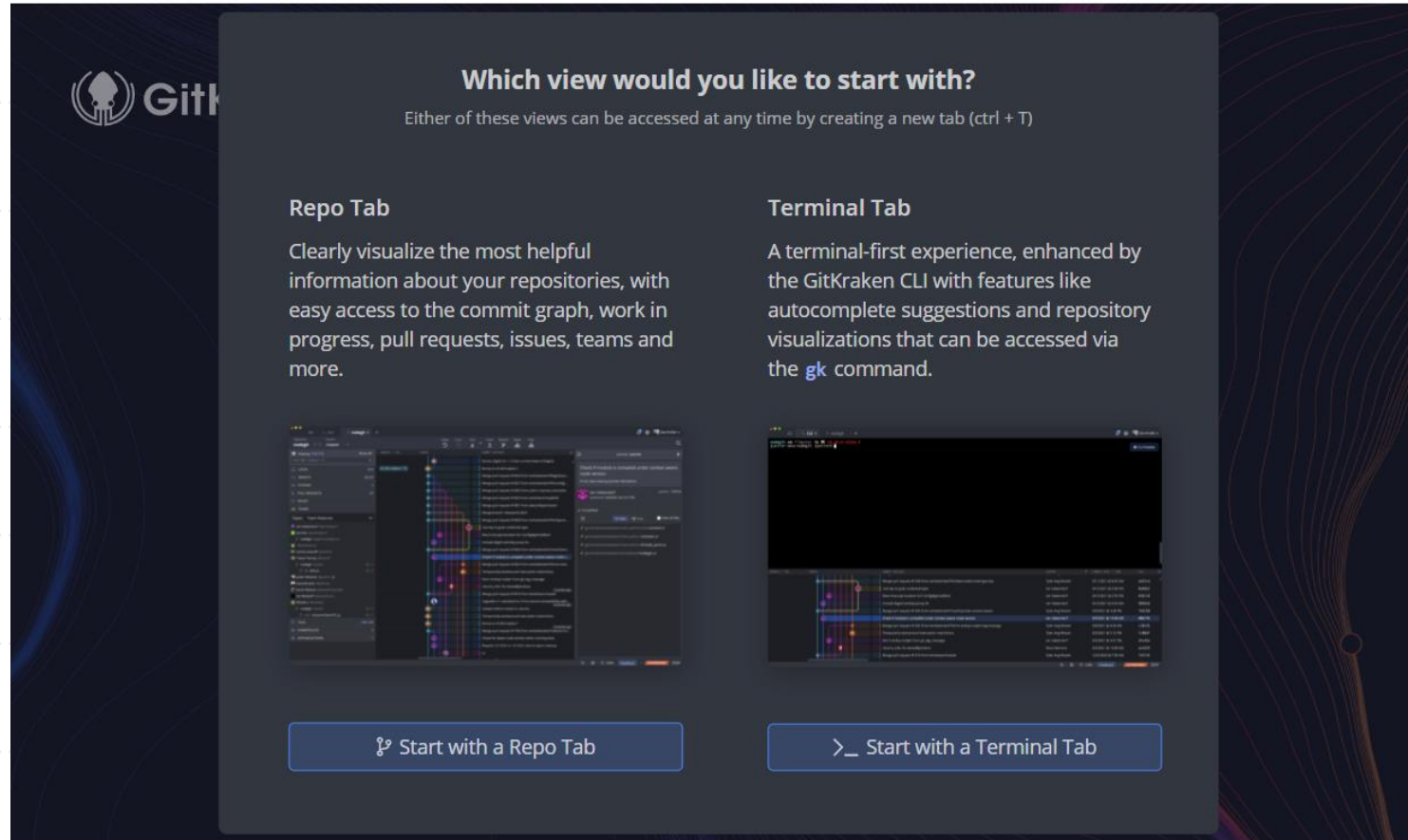
Author Email  
Enter the email you'll use when authoring commits

☒ Keep my .gitconfig updated with my GitKraken Profile preferences

Create Profile

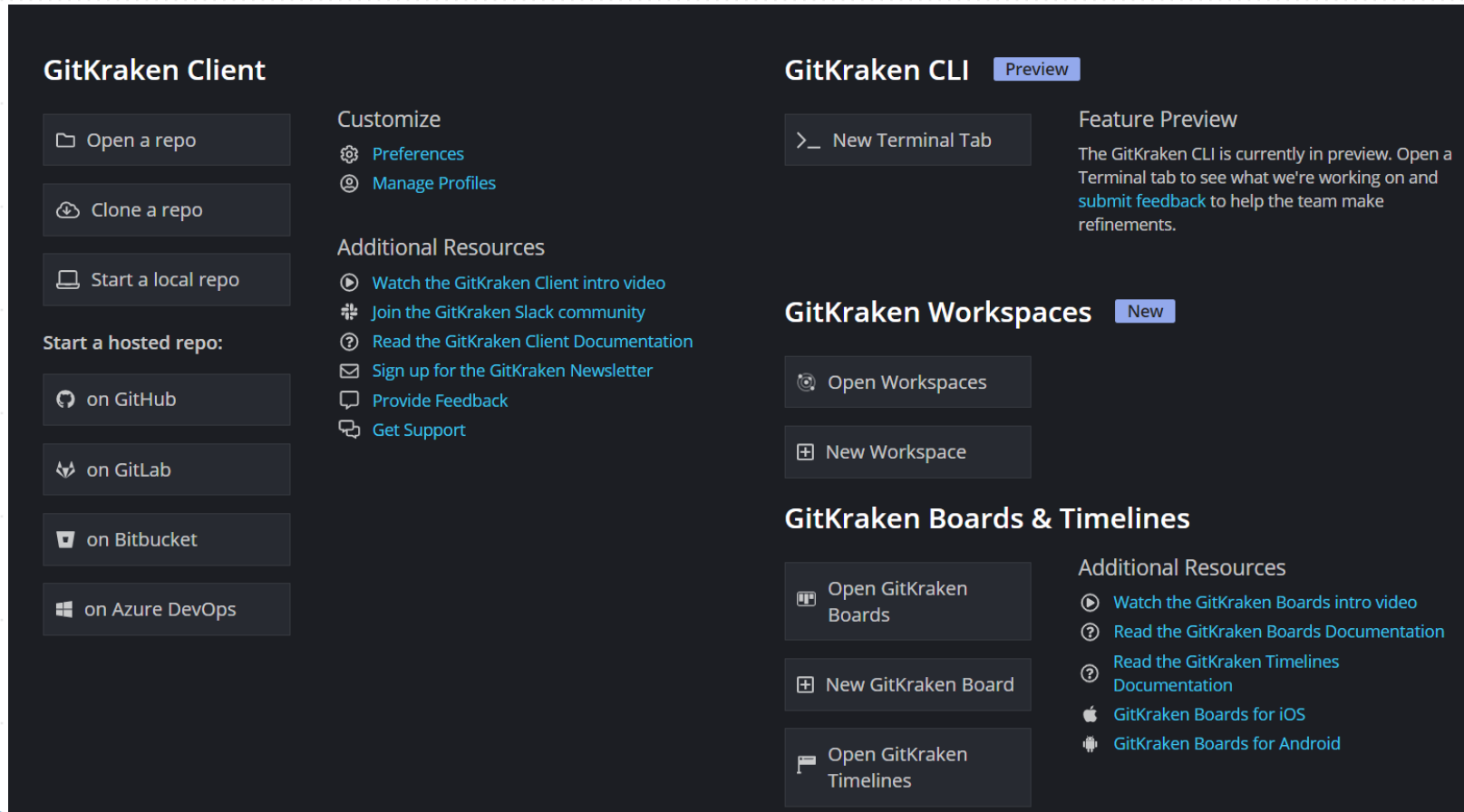
# How to use it?

- Let's start with the (Repo tab)
- We will not choose terminal tab for now, because it provides CLI and we want GUI to be easy for usage



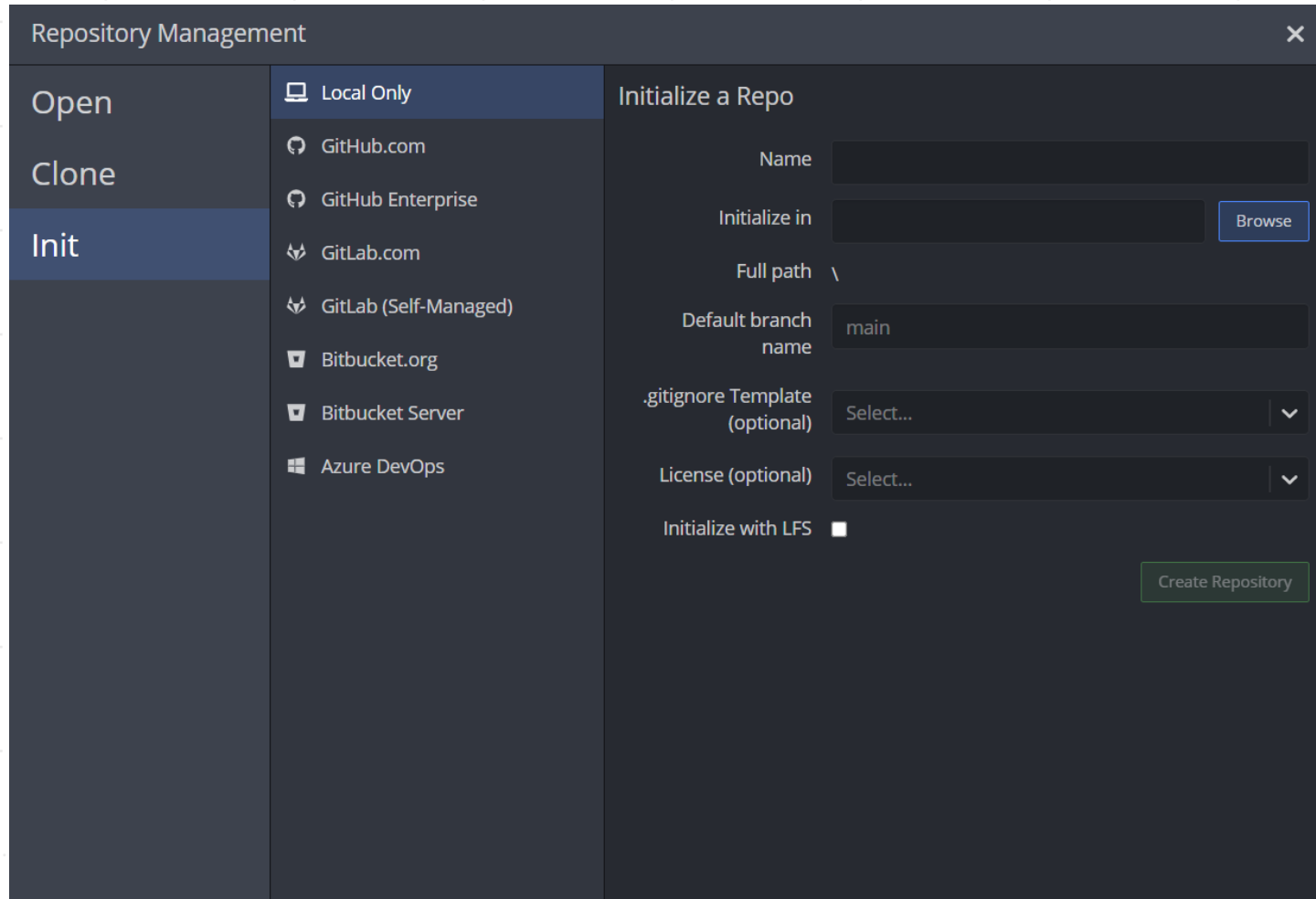
# How to use it?

- You can start a local repo by pressing (start a local repo) button



# How to use it?

- Choose the name and path of your repository





The screenshot shows a 'Repository Management' dialog box with a dark theme. On the left, there is a sidebar with three tabs: 'Open', 'Clone', and 'Init'. The 'Init' tab is currently selected. To the right of the sidebar, there is a list of repository providers: 'Local Only', 'GitHub.com', 'GitHub Enterprise', 'GitLab.com', 'GitLab (Self-Managed)', 'Bitbucket.org', 'Bitbucket Server', and 'Azure DevOps'. The 'Local Only' option is selected. The main area of the dialog is titled 'Initialize a Repo' and contains several input fields and a 'Create Repository' button. The fields include: 'Name' (a text input), 'Initialize in' (a text input with a 'Browse' button next to it), 'Full path' (a text input with a backslash character), 'Default branch name' (a text input with 'main' entered), '.gitignore Template (optional)' (a dropdown menu with 'Select...' and a downward arrow), 'License (optional)' (a dropdown menu with 'Select...' and a downward arrow), and 'Initialize with LFS' (a checkbox that is currently unchecked). The 'Create Repository' button is located at the bottom right of the dialog.

Open	Local Only	Initialize a Repo
Clone	GitHub.com	Name <input type="text"/>
Init	GitHub Enterprise	Initialize in <input type="text"/> <button>Browse</button>
	GitLab.com	Full path <input type="text" value="\"/>
	GitLab (Self-Managed)	Default branch name <input type="text" value="main"/>
	Bitbucket.org	.gitignore Template (optional) <input type="text" value="Select..."/>
	Bitbucket Server	License (optional) <input type="text" value="Select..."/>
	Azure DevOps	Initialize with LFS <input type="checkbox"/>
		<button>Create Repository</button>

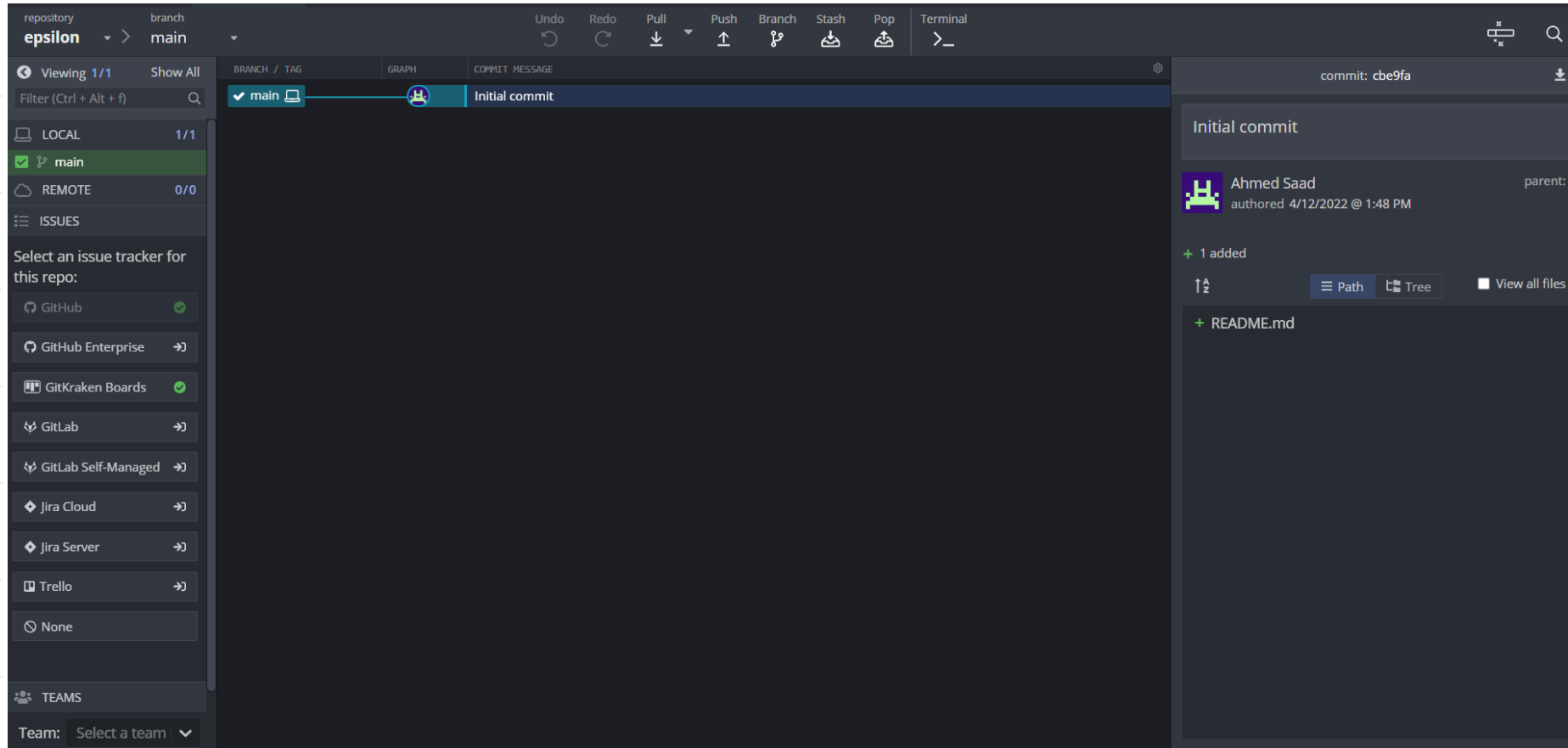
# How to use it?

- Gitkraken has created a folder for you with default readme file and (.git) folder on the specified directory

 .git	4/12/2022 1:48 PM	File folder	
 README	4/12/2022 1:48 PM	MD File	1 KB

# How to use it?

- Main page of the repo
- It contains a graph for your commits and branches.



# How to use it?

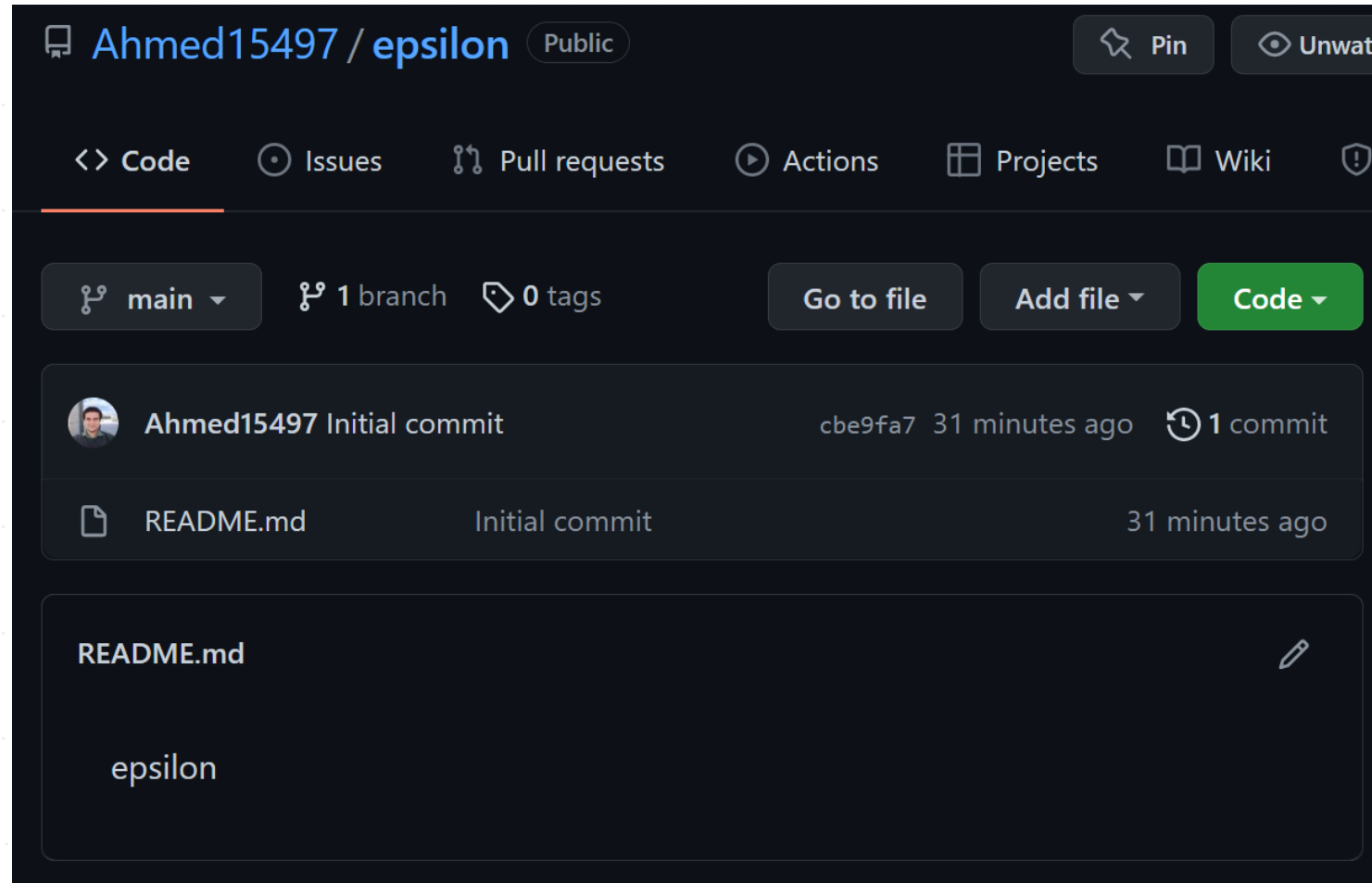
- Let's create a remote repo on github and connect it to our local repo
- Press (Remote) button on the left of the page

The screenshot shows the GitHub web interface. On the left sidebar, the 'REMOTE' button is selected. The main content area displays the 'Add Remote' dialog box. The dialog box has tabs for 'URL', 'GitHub', 'GitLab', and 'Bitbucket'. The 'GitHub' tab is active. The 'Account' dropdown is set to 'Ahmed15497'. The 'Repository Name' field contains 'epsilon'. The 'Remote Name' field contains 'origin'. The 'Description' field contains 'my remote repo'. The 'Access' dropdown is set to 'Public'. A green button at the bottom of the dialog box says 'Create remote and push local refs'.



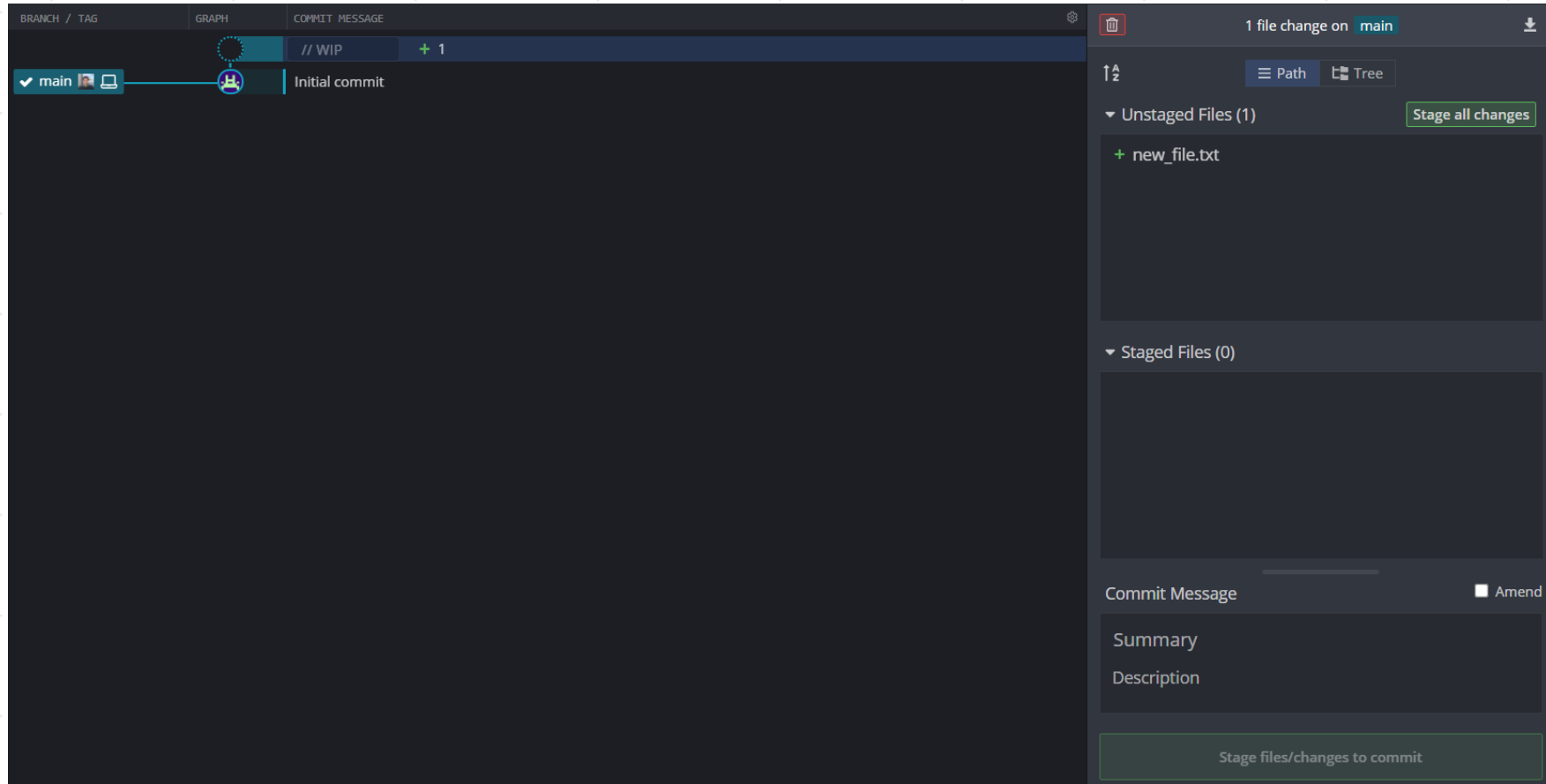
# How to use it?

- The remote repo has been created on github



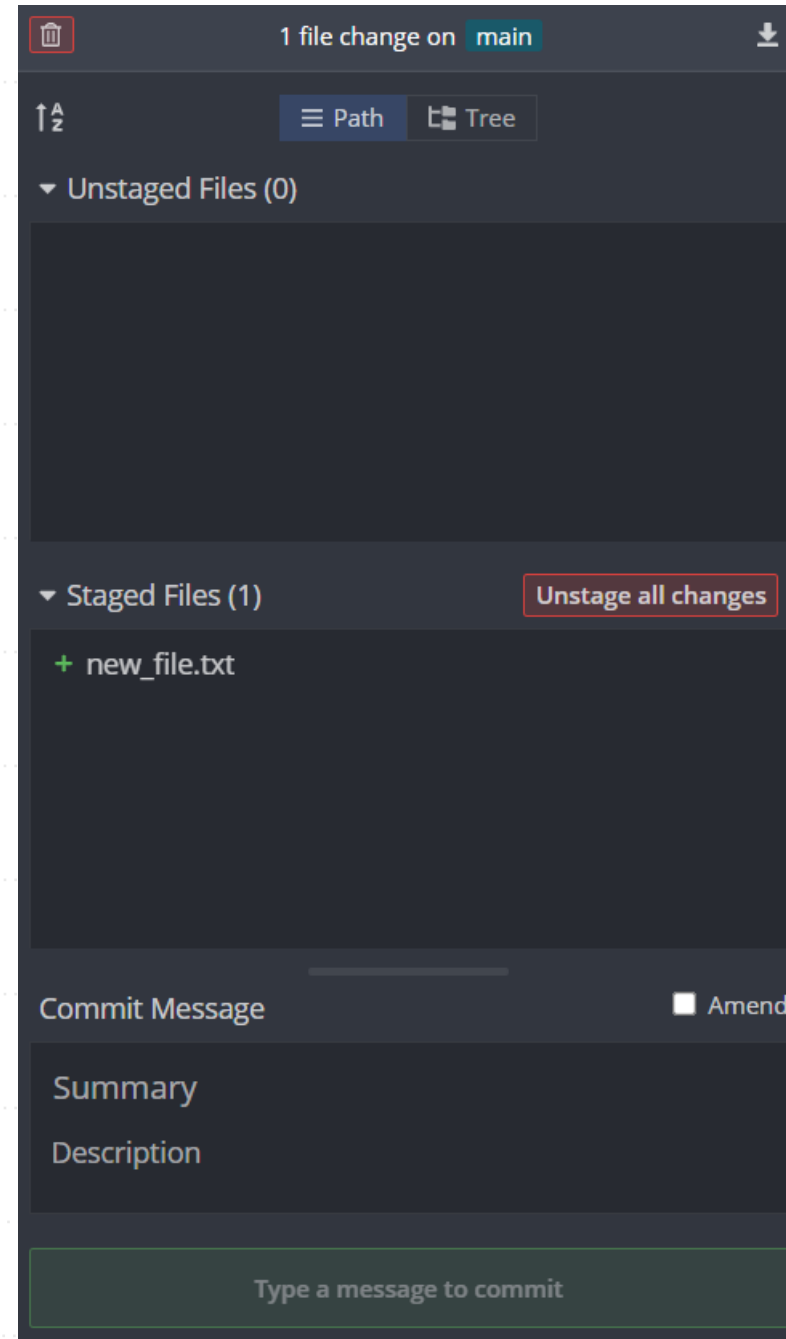
# How to use it?

- If you create new file, it will be on the unstaged area on the right section of your main page



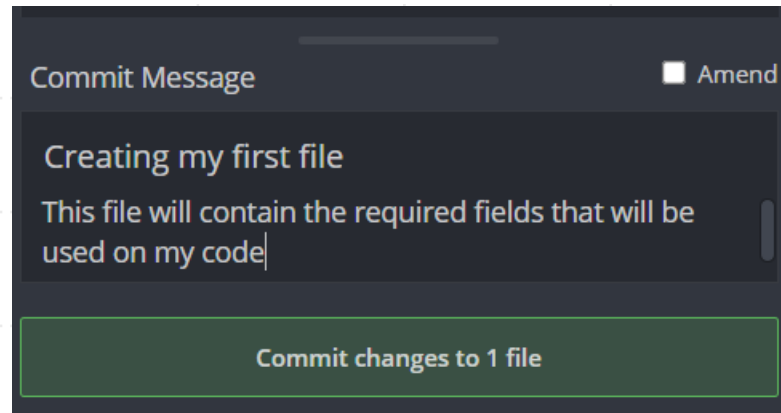
# How to use it?

- To stage your changes, press stage all changes.



# How to use it?

- Commit your changes by writing your commit message, then press commit changes button

A screenshot of a dark-themed commit message dialog box. At the top left, it says "Commit Message". At the top right, there is a checkbox labeled "Amend". The main text area contains the message "Creating my first file" followed by "This file will contain the required fields that will be used on my code|". A green button at the bottom is labeled "Commit changes to 1 file".

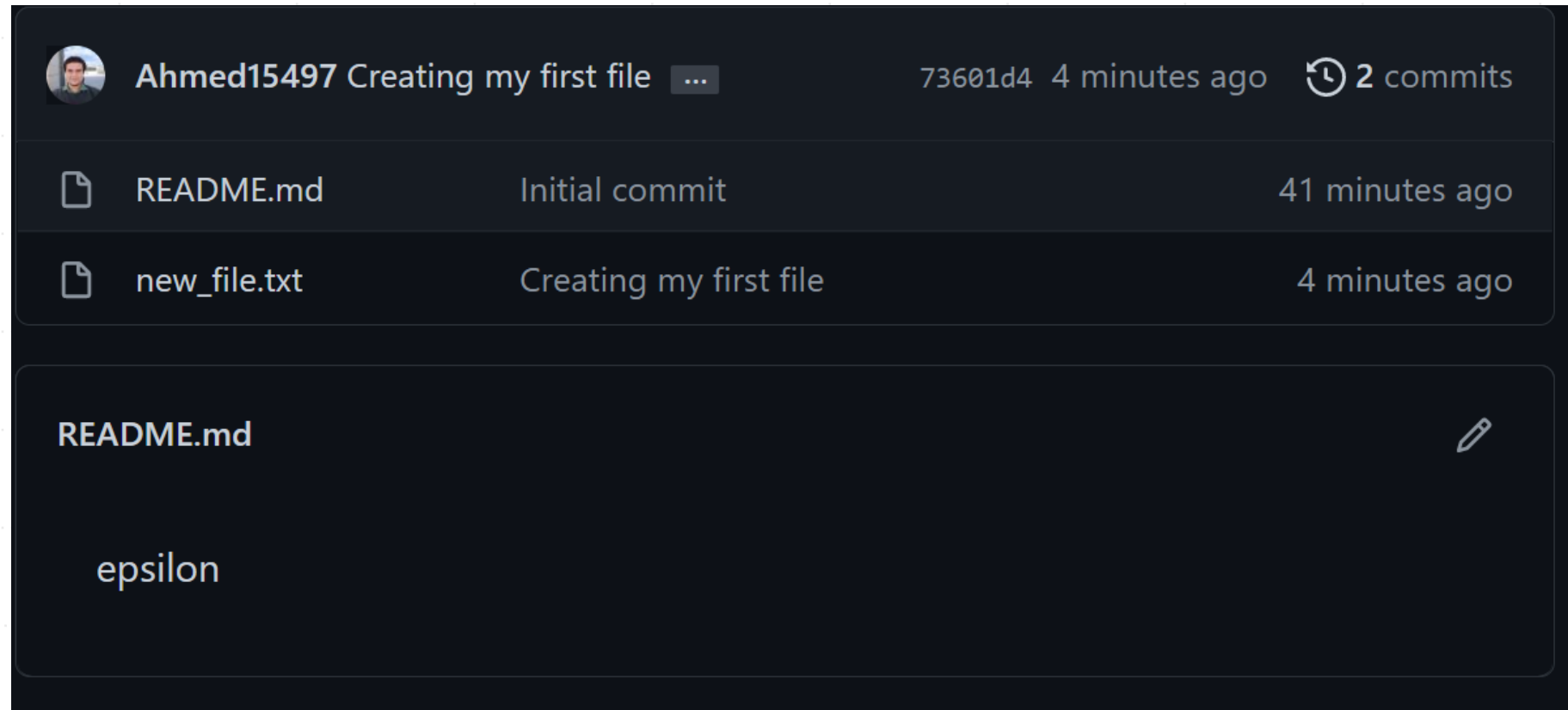
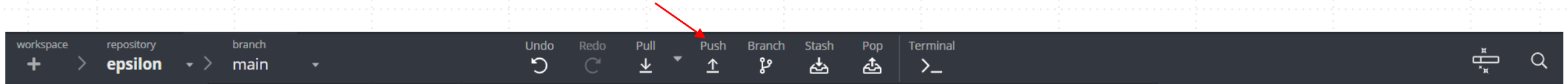
Commit Message ☐ Amend

Creating my first file  
This file will contain the required fields that will be used on my code|

Commit changes to 1 file

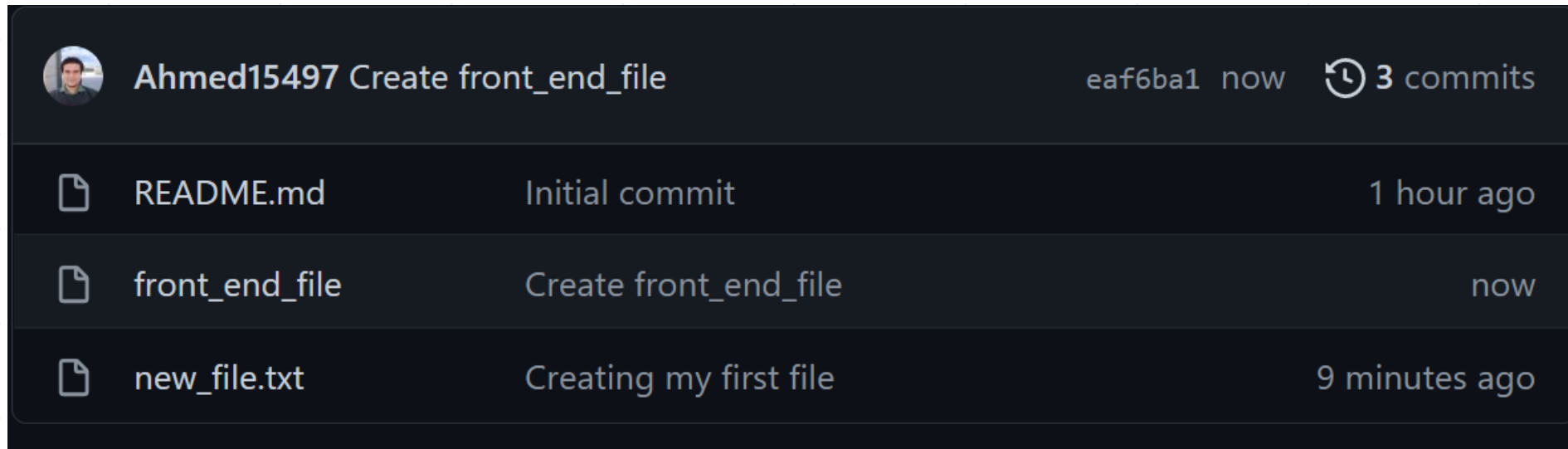
# How to use it?

- Push your changes to your remote repo by pressing the (Push) button on the upper bar of your page






# How to use it?

- The front end team have made some changes on your github repo and you need to pull these changes

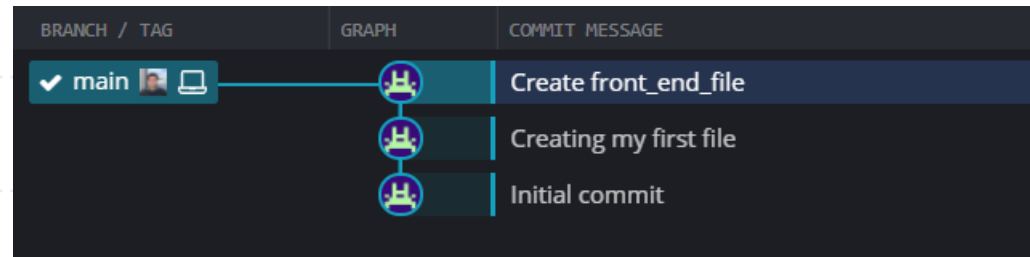
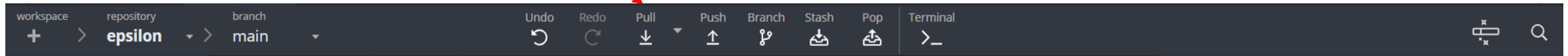


A screenshot of a GitHub commit history interface. At the top, the user profile 'Ahmed15497' is shown with a circular avatar, followed by the commit title 'Create front\_end\_file'. To the right of the title, the commit hash 'eaf6ba1' is displayed, followed by the word 'now' and a clock icon with the text '3 commits'. Below this header, there is a table with three rows, each representing a file change. Each row starts with a document icon, followed by the file name, the commit message, and the time since the commit.

	README.md	Initial commit	1 hour ago
	front_end_file	Create front_end_file	now
	new_file.txt	Creating my first file	9 minutes ago

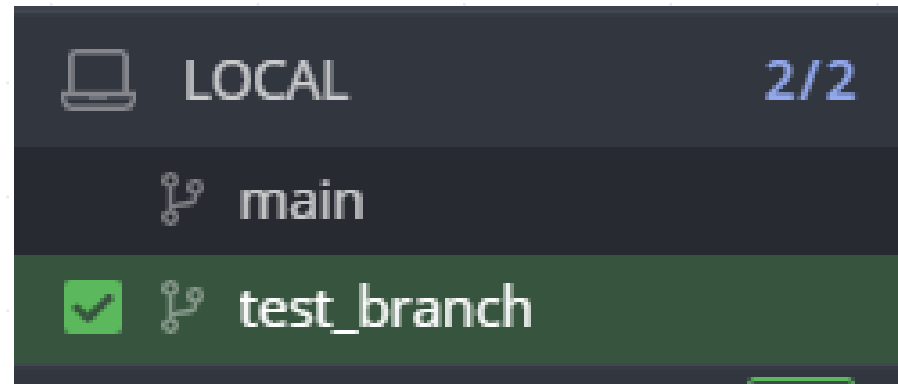
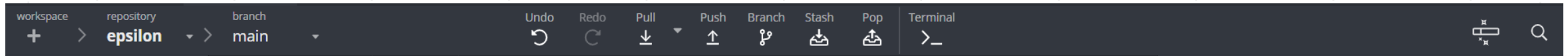
# How to use it?

- Pull their changes from your remote repo by pressing the (Pull) button on the bar at the upper section of the page



# How to use it?

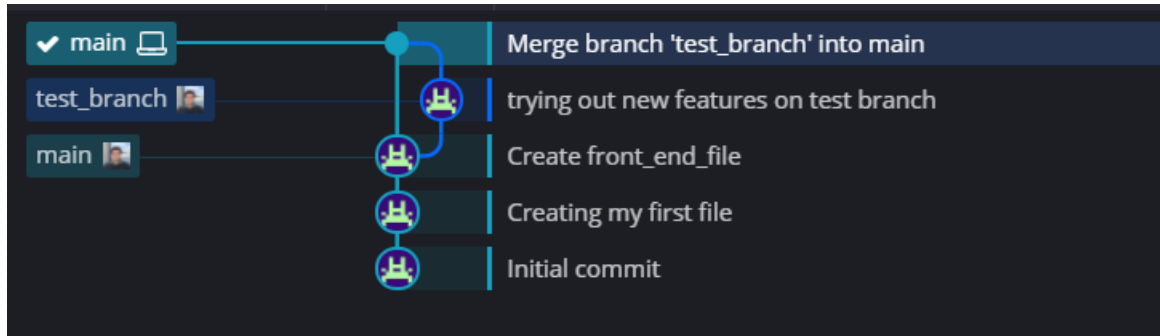
- Creating (test\_branch) by pressing (Branch) button on the upper section of the page
- You can name your new branch now with (test\_branch)





# How to use it?

- Do some changes to some files, then merge with main
- Right click on the (main) branch, then choose (merge test\_branch into main)
- As you can see the graph has been changed to reflect your merge



The screenshot shows the GitHub web interface. On the left, the 'test\_branch' is selected in the 'LOCAL' section. A right-click context menu is open, displaying various actions. A green arrow points from the 'test\_branch' in the interface to the 'Merge test\_branch into main' option in the menu. The menu options include: Pull (fast-forward if possible), Push, Set Upstream, Rebase test\_branch onto main, Fast-forward main to test\_branch, Merge test\_branch into main, Interactive Rebase test\_branch onto main, Checkout main, Create branch here, Cherry pick commit, Reset test\_branch to this commit, Revert commit, Interactive Rebase 1 children of eaf6ba, Edit commit message, Drop commit, Move commit up, Move commit down, Rename main, Delete main, Copy branch name, Copy commit sha, Copy link to this commit on remote: origin, Hide, Solo, Create tag here, and Create annotated tag here.