CLI, NPM & GITHUB

Download file and follow along!

Lecture 8, Week 8

TODAY

- ❏ CLI
- ❏ NPM
- ❏ GITHUB

CLI

- ❏ CLI stands for:
  - ❏ Command Line Interface
  - ❏ Command Line Interpreter
  - ❏ Command Line Input
- ❏ CLI is a command line program that accepts text input to execute operating system functions.
- ❏ In the 1960s, CLIs were used extensively.
- ❏ Back then, people had only a keyboard as an input device and the computer screen could only display text information. The operating systems like **MS-DOS** used the CLI as the standard user interface.

- Then, after years of only using a keyboard and risking to misuse the command line, **the mouse was invented**.
- The invention of a mouse marked beginning of the point-and-click method as a new way to interact with the computer.
- This method is much safer for average users, thus drove them away from CLI.
- Apart from that, operating systems started to develop an attractive way of computing, using GUI (Graphical User Interaction). GUI itself was phenomenal because of the use of buttons and menus to represent specific commands. This approach has been proven to be very intuitive.
- However, CLI is still used by software developers and system administrators to configure computers, install software, and access features that are not available in the graphical interface.

Shell – The Foundation Behind CLI

If we dive from CLI into the deeper part of an operating system, we will find the shell.

Shell is a user interface responsible for processing all commands typed on CLI. It reads and interprets the commands and instructs the operating system to perform tasks as requested.

In other words, a shell is a user interface that manages CLI and acts as the man-in-the-middle, connecting users with the operating system.

In practice, there are many things that shell can process, such as:

- ❏ Working with files and directories
- ❏ Opening and closing a program
- ❏ Managing computer processes
- ❏ Performing repetitive tasks

Among many types of shell, the most popular ones are Windows shell (for Windows) and bash (for Linux and MacOS).

Windows Shell

The default shell in Windows is **CMD.exe** or the **Command Prompt**. In fact, **Microsoft** has used the Command Prompt since the old days, where **MS-DOS** was the main operating system.

To open the Command Prompt, you can click **Start** -> **All Programs** -> **Accessories** -> **Command Prompt**. Or, you can simply press **Windows+R**, then type **CMD**, and press **enter**.

Depending on what you need, type in either a single command or a combination. You can also type commands that run within a sequence (one command is executed after another).

The Command Prompt is so robust that it can manage many things within the **Windows** operating system:

- ❏ Changing directories, listing directories, content, etc
- ❏ Handling networking like displaying IP networks settings
- ❏ Managing files like renaming, moving, etc
- ❏ Managing media like formatting and renaming volumes

Now, let’s learn how to use some syntax in the command prompt:

- ❑ **Changing directory**
  To navigate to a specific directory or folder in the command prompt, use **CD [path]**. Make sure you add space before the intended path. For example:
  ```
  CD C:\Program Files
  ```

- ❑ **Renaming a file**
  To rename a file within a specific folder, use **REN [drive:][path] [source] [target]**.If you mention the location, that means the renamed file will be saved in the same folder. For example:
  ```
  REN d:untitled.txt untitled1.txt
  ```

- ❑ **Deleting a file**
  To delete a file in the command prompt use **DEL [filename]**. If you want to add the options like **force deletion**, you can add it before the file name. For example:
  ```
  DEL /F untitled.txt
  ```

- ❑ **Renaming a Volume Disk**
  To edit the name of a specific volume disk, use **LABEL [drive:][new volume name]**. Keep in mind that you can use up to 32 characters on NTFS volume and 11 on FAT volume. For example:
  ```
  D:\ > LABEL d:MyData
  ```

Bash

Bash stands for **Bourne Again SHell** and was developed by the **Free Software Foundation**.

Bash is a type of shell used in **MacOS** and many **Linux** distributions. However, you can also install bash Linux on Windows 10.

In Linux, Bash shell is one of many shells the Linux users can use. The other types are Tchs **shell**, **Ksh shell**, and **Zsh shell**.

In most Linux distributions, the shell is located under the **Utilities** menu. If you use **Gnome** desktop, the name is **Terminal**, but if you use **KDE**, the name is **Konsole**.

Meanwhile, in MacOS, the program is **Terminal.app**. To run this program, go to **Application** -> **Utilities** -> **Terminal**. Or, you can simply type **terminal** using Spotlight search.

Once the terminal's opened, you can start typing a command. Basically, most commands consist of: **the command itself, the argument, and the option**.

While the command contains the instruction we want to perform, the argument tells where the command should operate and the option requests the modification of the output.

Now it's time to learn how to use the shell.

To begin with, you need to know the syntax to deal with the shell. This is also known as shell scripting – ways of using the script in CLI to run certain tasks.

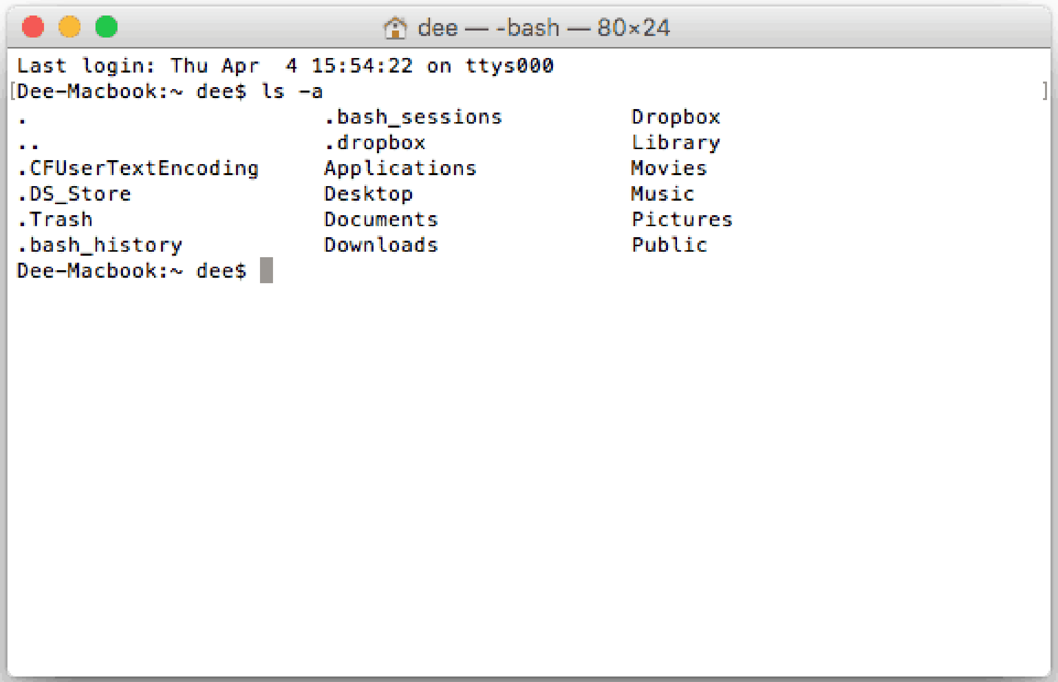
While there are many commands you can use with CLI, they all fall into **two categories**:

- ❏ The commands that **handle the processes**
- ❏ The commands that **handle the files**

To understand the command syntax in MacOS, let's learn from these examples:

- ❏ **List all files in a folder**
  To know what files under a specific folder, use **ls**.
  The default command will exclude the hidden files. To show all files, you can add **-a**. For example:

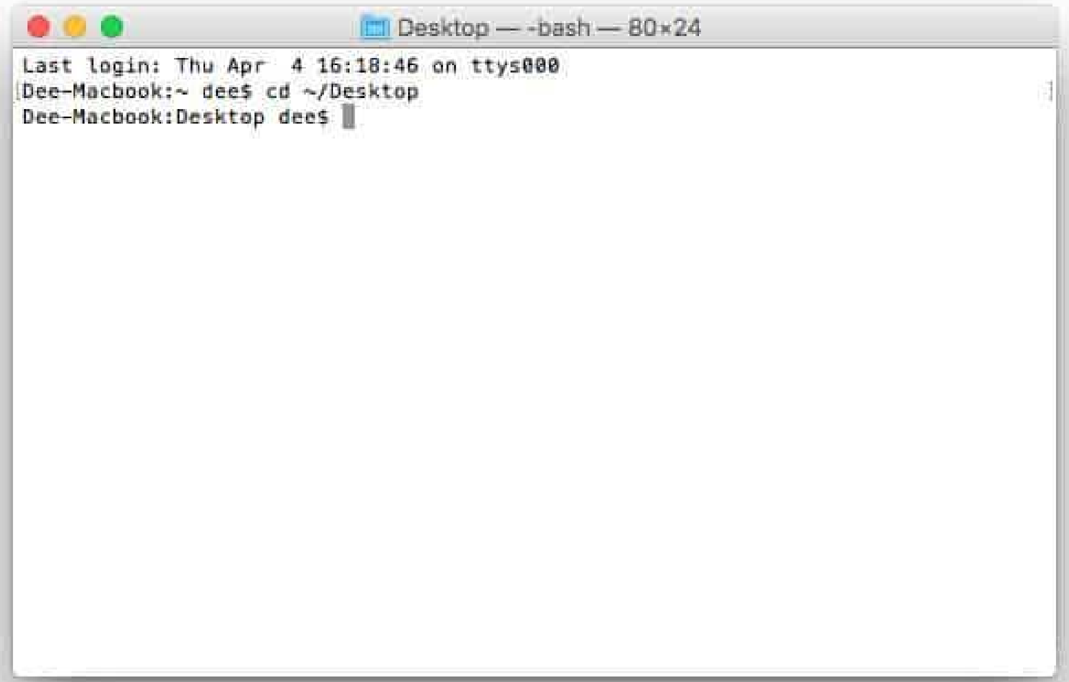
ls -a



```
Last login: Thu Apr  4 15:54:22 on ttys000
Dee-Macbook:~ dee$ ls -a
.                       .bash_sessions          Dropbox
..                      .dropbox                Library
.CFUserTextEncoding     Applications            Movies
.DS_Store               Desktop                 Music
.Trash                  Documents               Pictures
.bash_history           Downloads               Public
Dee-Macbook:~ dee$ 
```

- ❑ **Change directory**
  To move to a specific directory, use **cd destination**. For example:

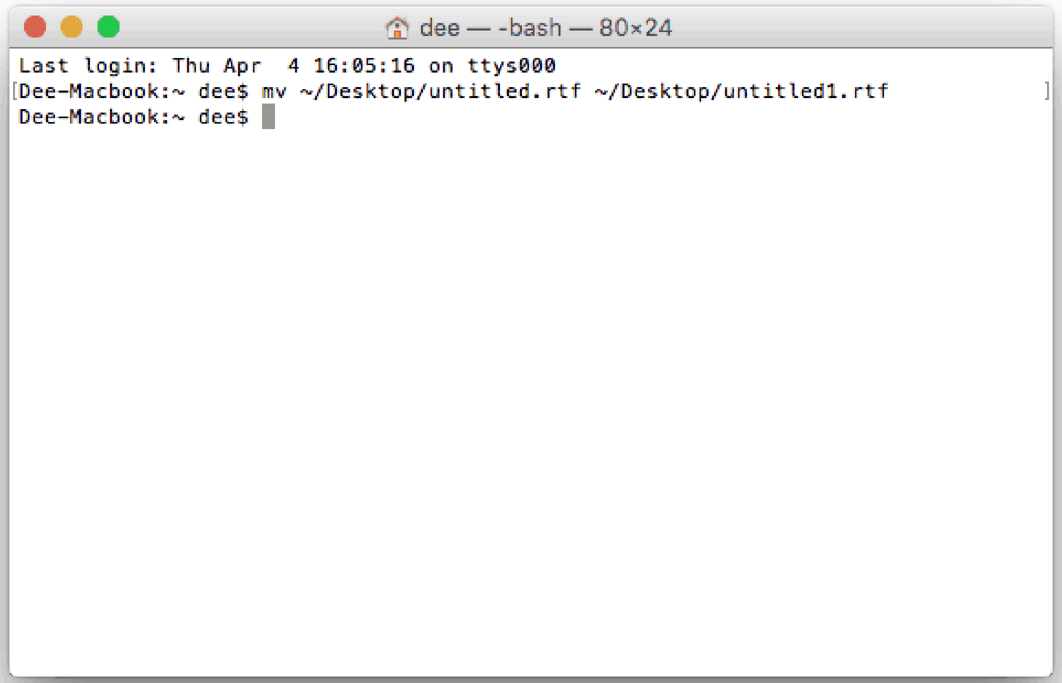
cd ~/Desktop



```
Last login: Thu Apr  4 16:18:46 on ttys000
Dee-Macbook:~ dee$ cd ~/Desktop
Dee-Macbook:Desktop dee$ 
```

❑ **Rename file**
To rename a file within a specific folder, use **mv source destination**. Keep in mind, that you need to make sure the name of the file and the extension. For example:

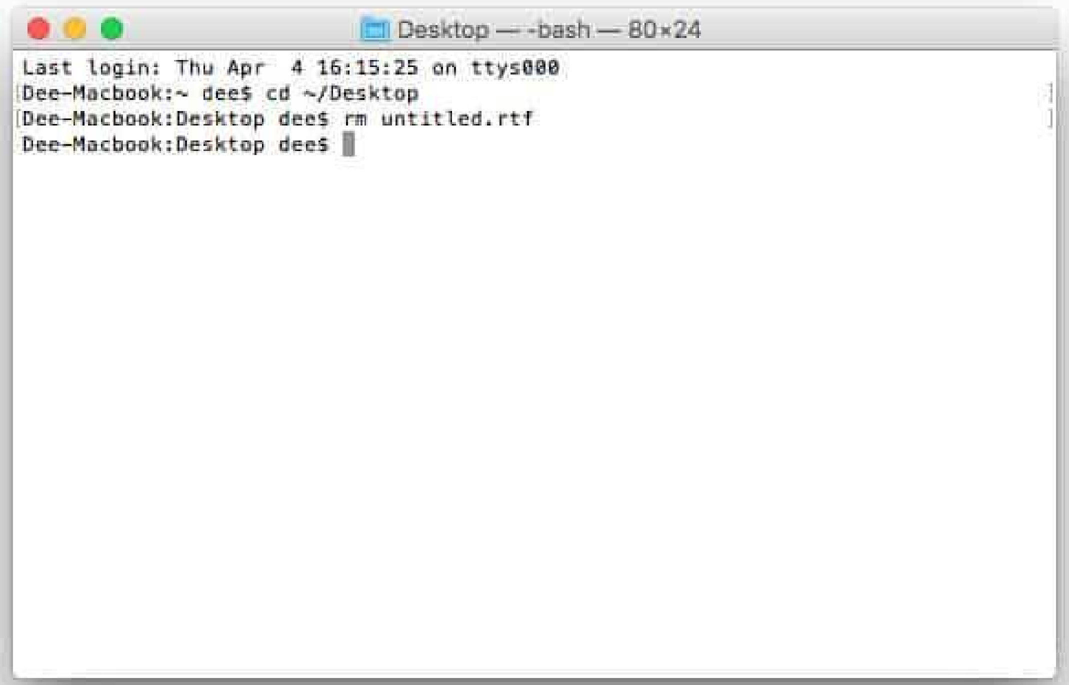
mv ~/Desktop/untitled.rtf ~/Desktop/untitled1.rtf



```
Last login: Thu Apr  4 16:05:16 on ttys000
[Dee-Macbook:~ dee$ mv ~/Desktop/untitled.rtf ~/Desktop/untitled1.rtf
Dee-Macbook:~ dee$ 
```

- **Delete a file**
  To delete a file in the specific folder use **rm** filename. To avoid deleting the wrong file, make sure you move to the right folder destination first. For example:

rm untitled.rtf



```
Last login: Thu Apr  4 16:15:25 on ttys000
Dee-Macbook:~ dee$ cd ~/Desktop
Dee-Macbook:Desktop dee$ rm untitled.rtf
Dee-Macbook:Desktop dee$ 
```

Again, typing the right command is important. That means you should pay attention to each character you use, including space. Not only that, make sure you type the correct case.
If for certain reasons you want to stop the ongoing process on the Command Prompt or Bash, simply hit **Control+C**.

NPM

The name npm (Node Package Manager) stems from when npm first was created as a package manager for Node.js.

It is a widely used repository for the publishing of open-source Node.js projects. Meaning that is an online platform where anyone can publish and share tools written in JavaScript.

npm is a command line tool that helps to interact with online platforms, such as browsers and servers. This utility aids in installing and uninstalling packages, version management, and dependency management needed to run a project.

To use it, you have to install **node.js** – as they are bundled together.

The npm command line utility enables node.js to work properly. In order to use packages, your project must contain a file called **package.json**. Inside that package, you'll find metadata specific to the projects.
The metadata shows a few aspects of the project in the following order:

- ❑ The project's name
- ❑ The initial version
- ❑ Description
- ❑ The entry point
- ❑ Test commands
- ❑ The **git** repository
- ❑ Keywords
- ❑ License
- ❑ Dependencies
- ❑ The devDependencies

Metadata helps to identify the project and acts as a baseline for users to get information about it.
Here is an example of how you can identify a project through its metadata:

```
{
 "name": "gbodo-cares",
 "version": "1.0.0",
 "description": "npm guide for beginner",
 "main": "beginner-npm.js",
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1"
 },
 "keywords": [
   "npm",
   "example",
   "basic"
 ],
 "author": "Gbodo ICT Centre",
 "license": "MIT",
 "dependencies": {
   "express": "^4.16.4"
 }
}
```

How to Install npm Modules and Start Projects?

First things first, you have to make sure that **node.js** and **npm** have been installed. Do so by running a few simple commands.

To see if node.js is installed, open the **Terminal** or a command line tool, and type `node -v`. This should show a version number if you already have it.

To see if npm is installed, type in `npm -v`. This should present the version number.

Initializing A Project with npm

If you already have Node and npm, and you want to get on to building your project, run the `npm init` command. This will trigger the initialization of your project.

For example, let’s create a directory called `test-npm` and `cd` into it.

Installing npm Modules

A package in node.js contains all the files you need for a module. Modules are JavaScript libraries you can include in your project.

Installing modules is one of the most basic things you should learn to do when getting started with the Node package manager. Here’s the command to install a module into the current directory.

Installing npm Modules

A package in node.js contains all the files you need for a module. Modules are JavaScript libraries you can include in your project.

Installing modules is one of the most basic things you should learn to do when getting started with the Node package manager. Here’s the command to install a module into the current directory.

GITHUB

GitHub is a code hosting platform for collaboration and version control.

GitHub lets you (and others) work together on projects.

GitHub is a cloud-based service that hosts a version control system (VCS) called Git. It allows developers to collaborate and make changes to shared projects while keeping detailed track of their progress.

GitHub essentials are:

- ❑ Repositories
- ❑ Branches
- ❑ Commits
- ❑ Pull Requests
- ❑ Git (the version control software GitHub is built on)

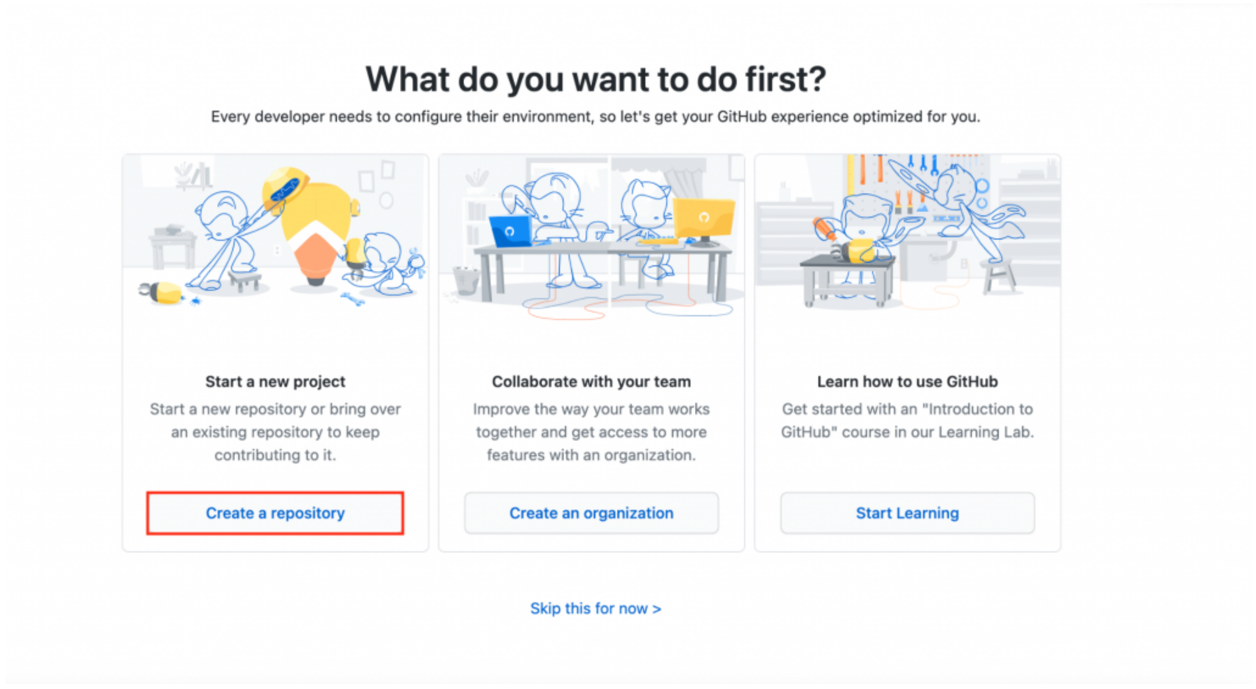
How to Get Started With GitHub?

You can try out GitHub for free. A basic plan is available that includes unlimited repositories and collaborators but only 500 MB of storage space.

1. Create a GitHub Repository

A repository, or repo, will be the central hub of your project. It could be one file or a collection of files containing code, images, text, or anything else.
To begin the process, follow these steps:

- ❑ Click **Create a repository** to start a new project.



What do you want to do first?

Every developer needs to configure their environment, so let's get your GitHub experience optimized for you.

Start a new project

Start a new repository or bring over an existing repository to keep contributing to it.

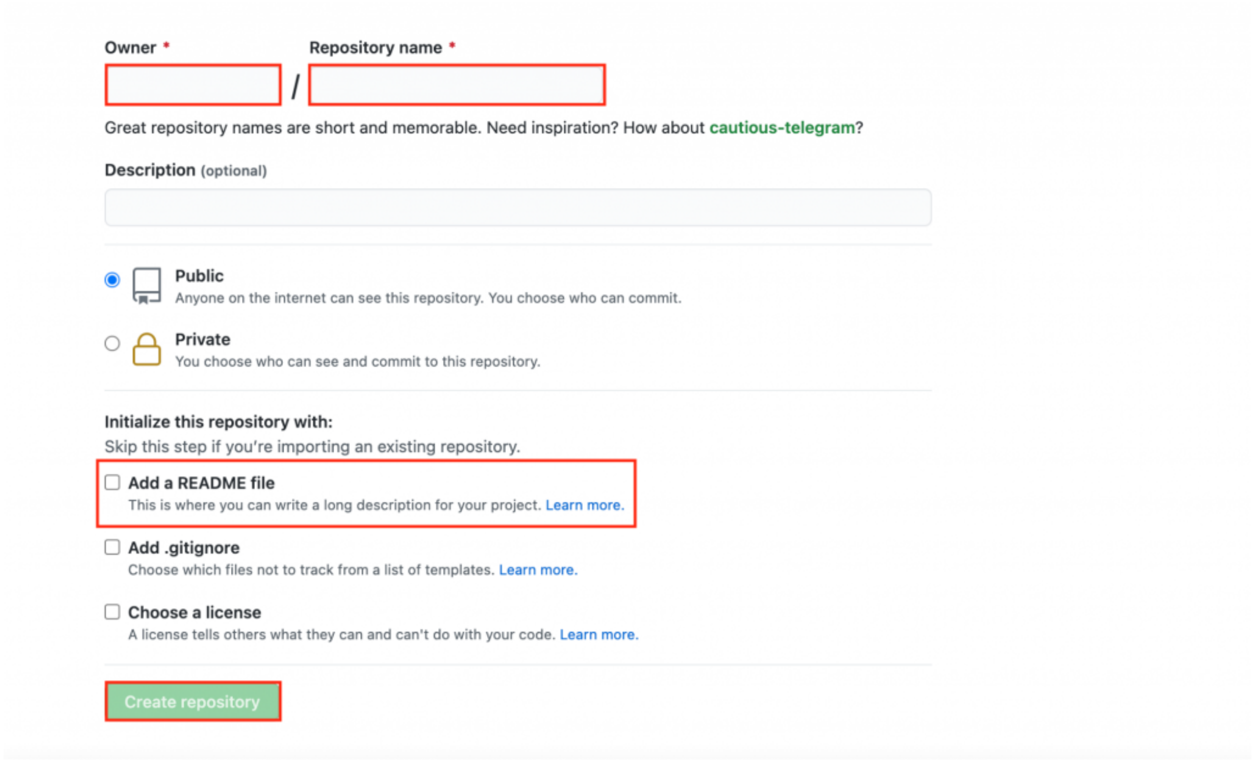
Collaborate with your team

Improve the way your team works together and get access to more features with an organization.

Learn how to use GitHub

Get started with an "Introduction to GitHub" course in our Learning Lab.

- ❑ The **Owner** section will already have your account name. Create a **Repository Name**. Check if it’s set to **Public** to make it open-source, and then check the **Add a README file** box. Finally, click **Create repository**.



Owner * / Repository name *

Great repository names are short and memorable. Need inspiration? How about cautious-telegram?

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.

☐ Choose a license
A license tells others what they can and can't do with your code. Learn more.

NOTE: Note that you don't have to set your repository to be open-source (Public). You can set it to **Private** to manage who gets to see it and make changes.
Congratulations, you have now created a new repository that will contain the original file of your project. The next step is to learn what you can do with it.

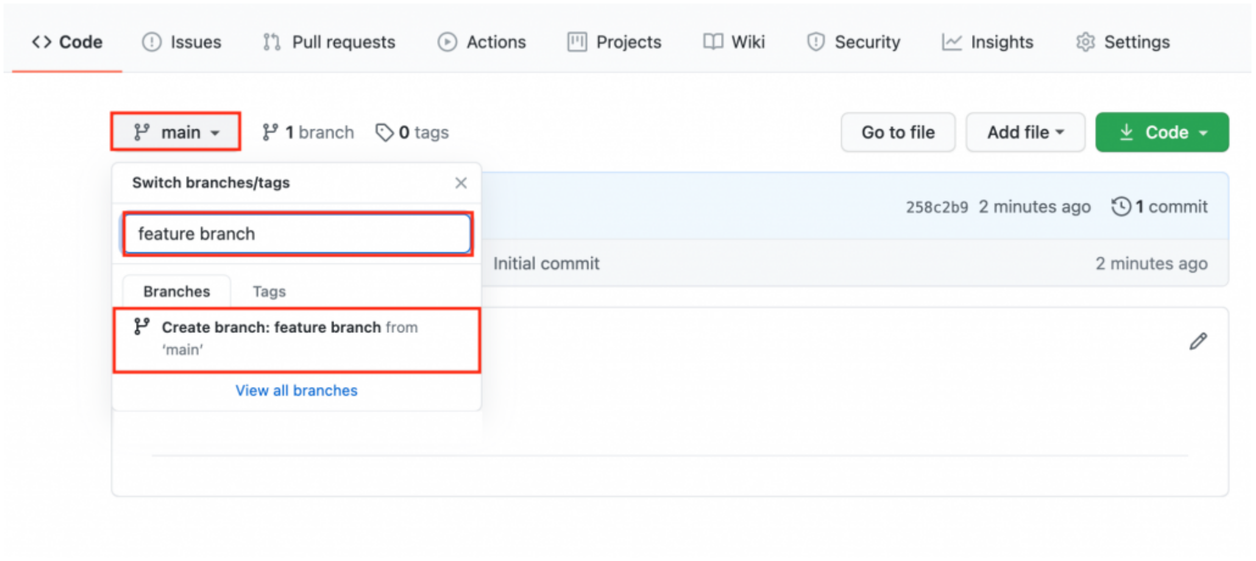
2. Creating GitHub Branches

By creating branches, you generate different versions of a repository. By making project changes to the feature branch, a developer can see how it will affect the master project when it's integrated.

Here's how you can generate a feature branch:

- Go to your new repository. Press the **main** button and enter the name of your new feature branch. Click on **Create**

branch.



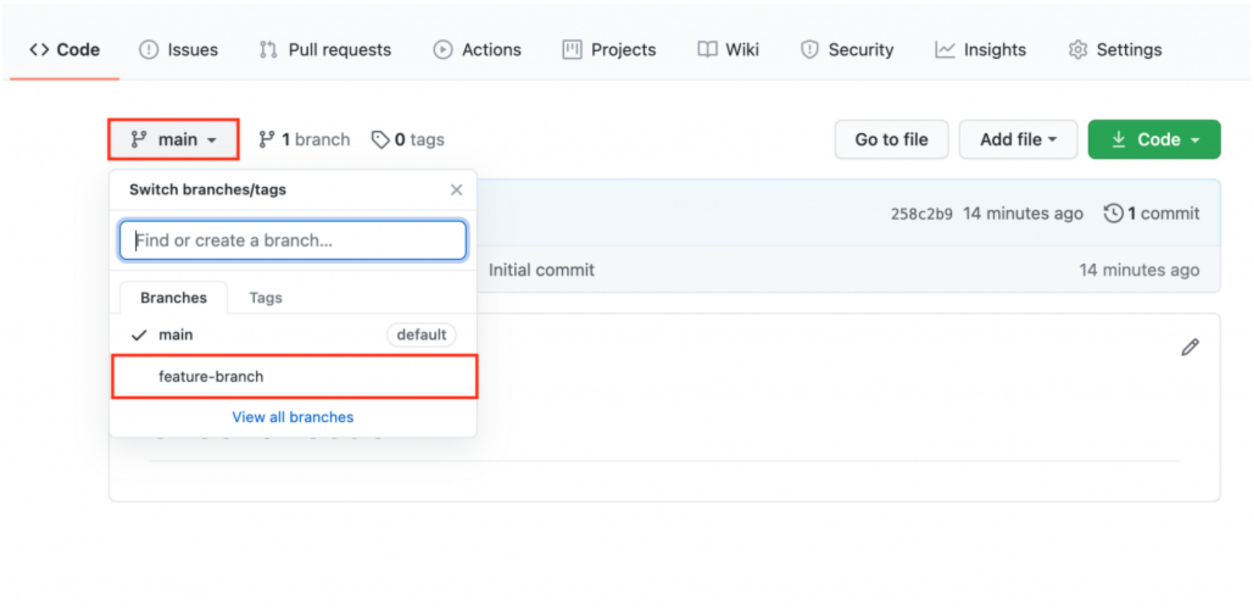
You have now created a feature branch that looks identical to the master branch. You can get started on making changes to it freely without affecting the project.

3. Understanding GitHub Commits

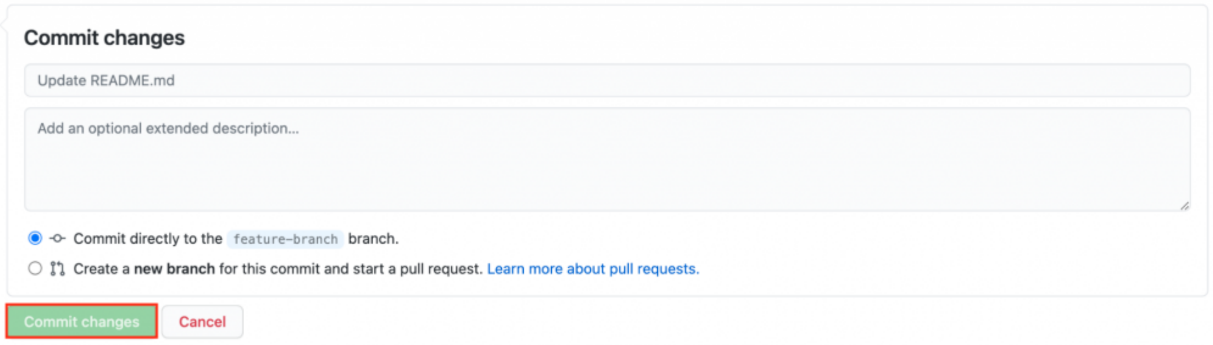
Commits are what saved changes on GitHub are called. Every time you'll change the feature branch's file, you'll have to **Commit** it to keep it.

Here's how to make and commit a change:

❑ Access the feature branch by clicking **main** and selecting your newly created branch from the dropdown menu.



❑ Click on the "pencil icon" to get started on editing the file. Once you're finished, write a short description of what changes were made. Click **Commit changes**.


Commit changes

Update README.md

Add an optional extended description...

[✓ Commit directly to the `feature-branch` branch.]

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

4. Creating GitHub Pull Requests

To propose changes you just made to other developers working on the same project, you should create a **pull request**.

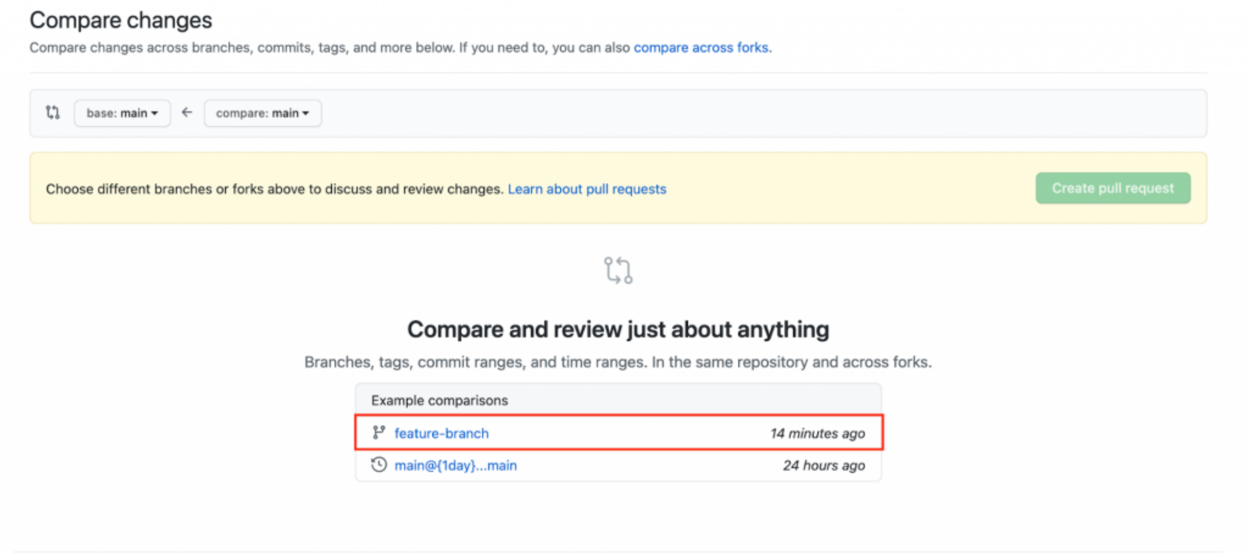
They are what makes working together on projects so easy, as they're the main tool for collaboration on GitHub.

Pull requests let you see the differences between the original project and your feature branch.

It’s how you ask your peers to review them. If the other developers approve, they can **merge pull request**, which will apply those changes to the main project.

To make a pull request follow the steps below:

❑ Click on **Pull requests** -> **New pull request**. In the **Example comparisons** section, select the **feature branch** you were just working on.



Compare changes

Compare changes across branches, commits, tags, and more below. If you need to, you can also compare across forks.

Choose different branches or forks above to discuss and review changes. Learn about pull requests

Compare and review just about anything

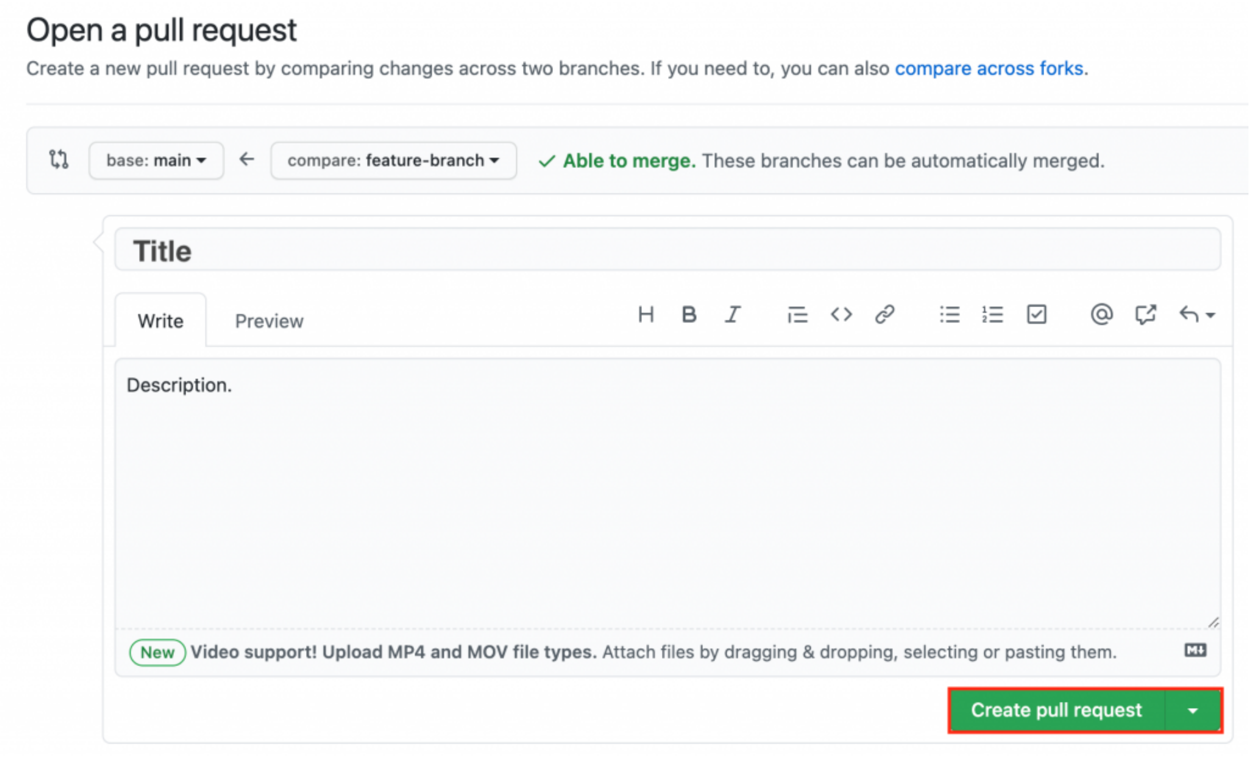
Branches, tags, commit ranges, and time ranges. In the same repository and across forks.

Example comparisons

feature-branch	14 minutes ago
main@{1day}...main	24 hours ago

❑ Look over the changes once more and click **Create pull request**. On the new page, write the title and provide a short description of what you worked on to encourage the merge.

Click **Create pull request.**



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: main ← compare: feature-branch ✓ Able to merge. These branches can be automatically merged.

Title

Write Preview

Description.

New Video support! Upload MP4 and MOV file types. Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Now other developers will be able to merge the changes you made with the original project files.

For everything else you may need to know about getting started on GitHub, check out this **guide**.