Git: A Time Machine

Stop naming files ‘final_v2_REALLY_FINAL.docx’

Jayrup Nakawala

Why are we here?

Because you probably manage files like this:

```
project/
├── index.html
├── index_backup.html
├── index_new.html
├── index_final.html
├── index_final_FINAL.html
└── index_final_v2_dont_touch.html
```

Thats stupid.

What is Git?

Git is a save button for your entire project.

It allows you to:

- Save “snapshots” of your folder.
- Travel back in time when you inevitably break something.
- Collaborate without emailing zip files like it’s 2003.

Git ≠ GitHub

What is Git?

Git

- The tool installed on your machine (CLI).
- Local.
- Works without internet.
- The Engine.

GitHub

- A website owned by Microsoft.
- Remote.
- Needs internet.
- The Garage where you park your code.

Installation

Open this website:

`https://git-scm.com/install/`

and follow the installation steps.

To verify use

```
git --version
```

Basics of terminal

List - `ls`

List all - `ls -a`

Change Directory - `cd [path]`

Make Directory - `mkdir [path]`

Copy File - `cp [path] [path]`

Move File - `mv [path] [path]`

Create file if it doesn't exist - `touch [path]`

for more information, ask LLMs

Initialization

```
mkdir my_project # create folder
cd my_project # move into that folder
git init . # initialize git
```

> **Note**
>
> The '.' represents the current folder.

Congratulations, you have created a hidden .git folder that tracks everything.

To check do,

```
ls -a
```

Configure

Configure git to use your name and email.

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

Rename the default branch

```
git branch -m main
```

Check Yourself

Before you do anything, always ask Git what it sees.

```
touch test.txt # create a file
git status # check status
```

Tip

Run git status constantly. If you are confused, run `git status`.

Add (Stage)

You created test.txt. Git sees it as “Untracked”.

To add a specific file:

```
git add test.txt
```

To add everything:

```
git add .
```

again, the ‘.’ represents the current folder.

Commit (Save)

Seal the box with a label.

```
git commit -m "Created the test file"
```

The `-m` flag is mandatory. If you omit it, Git opens Vim.

If you don’t know how to exit Vim, you will live inside that terminal forever.

Try it yourself

- Make any changes, be it create, edit or delete.
- see the status `git status`
- Stage (add) the changes `git add .`
- Commit (save) the changes `git commit -m "msg"`

History

You committed 5 times. How do you see them?

```
git log
```

You will see a list of “hashes” (random looking numbers) and your messages.

Along with your name and email.

The “Time Machine” Part

You deleted the wrong file. You broke the code. Go back to the last safe point.

```
git restore .
```

this will revert your whole folder to the last commit.

if you want a previous version:

```
git checkout [hash]
```

GitHub & Remotes

You want to upload your code to the internet.

Create a Repo on GitHub (empty).

To easily link your local git to github, use VSCode.

After that:

```
git remote add origin {url}
```

Ship it:

```
git push -u origin master
```

Keep in mind

- If there are commits on your local machine that you want to `upload`, use `git push`.
- If there are commits on your cloud that you want to `download`, use `git pull`.

Summary

git init (Start)

git add . (Stage)

git commit -m “msg” (Save)

git push (Upload)

Questions?

Start using Git today, or enjoy losing your thesis/codebase to a corrupted hard drive.

Branching

Imagine you want to try something risky, but you don’t want to ruin your working code.

Create a parallel universe.

```
git checkout -b new-feature
```

Do your work. If it sucks? Delete the branch. If it works? Merge it back to master.