Git/GitHub Documentation: Local System

Gagan Pradhan

December 23, 2025

1 Introduction & CLI Basics

Before setting up Git, we must understand the Command Line Interface (CLI), which is where we give instructions to the computer.

- **Default Directory:** By default, the CLI directory is set to the `$HOME` folder.
- `ls`: Returns files and folders in the current directory.
- `cd`: Changes the current directory.

```
cd ~/Desktop/Repositories/ANN-Model
```

Listing 1: Navigating to a project folder

2 Project Setup (Installing Git)

2.1 Creating a Version

To start tracking a project, we must set up Git inside the folder.

```
git init
```

This setup command initializes Git inside the folder, meaning Git begins tracking changes within that directory.

2.2 Checking Status

To see what changes have been made since the previous version:

```
git status
```

Example Output:

```
On branch master
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .venv/
    1.py
    2.py
    Coding_a_layer.py

nothing added to commit but untracked files present
```

3 The Git Workflow

We create versions in a two-step process: **Staging** and **Committing**.

3.1 Staging Changes (git add)

`git add` allows us to pick what changes we want to have in the next version.

- `git add 1.py`: Adds specific file changes to the staging area.
- `git add .`: Adds **all** content in the current folder (represented by `.`) to the staging area.

Note: `git add` doesn't create a new version; it only selects what needs to be added.

3.2 Committing Changes (git commit)

`git commit` creates a newer version. We refer to a commit as a "version".

```
git commit -m "A layer of neurons"
```

- `-m`: Attaches a message to the commit.
- This creates the first version of your project with the given message.

3.3 Configuration

If this is your first time, you must configure your identity:

```
git config --global user.name "skysys"
git config --global user.email "gagan07pradhan@gmail.com"
```

4 Managing History

4.1 Viewing History

```
git log
```

This gives us the version history of the project.

4.2 Amending Commits

If we missed files in a version or want to change the commit message of the *latest* commit without creating a new version:

```
git add README.md
git commit -m "A layer of neurons and a readme file added" --amend
```

This adds the new files to the previous commit and updates the message.

5 Visualizing Changes

5.1 In the Code Editor (VS Code)

We can visualize changes and versions in the Source Control section of the editor.

- **Changes:** Shows differences since the previous version.
- **Staged Changes:** Changes ready to be added to the next version.
- **Graph:** Shows the commit history visually.
- **UI Tip:** You can use the + button in VS Code to stage changes instead of typing `git add`.

5.2 Undo/Reset

- **Unstage:** VS Code allows resetting changes using the **-** button.
- **CLI Reset:** `git reset .` removes changes from the staging area.
- **Revert Modifications:** `git checkout -- <file>` (or just `git checkout <file>`) reverts changes to the file, discarding modifications in the working directory.

6 Switching Between Versions & Branching

Each commit has a **Commit Hash** (ID).

Example Git Log Output

commit 8a12... (HEAD -¿ master)
Author: skysys
Date: Tue Dec 23 17:06:06 2025
Version 3
commit b26e...
Author: skysys
Date: Tue Dec 23 17:05:33 2025
Version 2
commit 5091...
Author: skysys007
Date: Tue Dec 23 14:08:08 2025
A layer of neurons and a readme file added

6.1 Navigating History (HEAD)

HEAD represents the current version we are looking at. To revert or switch to Version 2:

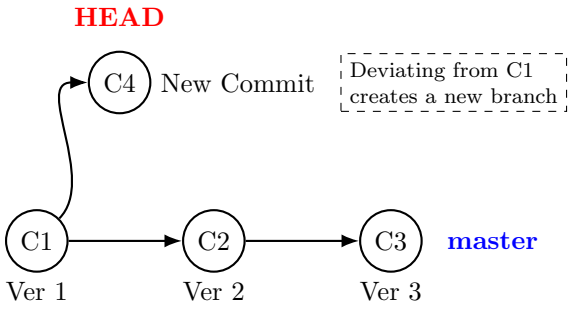
```
git checkout b26e85a69468a7f634574bcf2b247abc9bc5cc88
```

6.2 Branching Concepts

If we go back to Version 1 and commit a new version, it creates a new timeline (Branch).

```
git log --all --graph
```

This command visualizes all commits including branches.



6.3 Restoring Files from Past Versions

You can restore a specific file from the past without reverting the whole project.

- **Syntax:** `git checkout <commit-hash> <filename>`
- **Example:** If we are at Version 3 but want the `readme.md` from Version 1:

```
git checkout 509150ee readme.md
```

This restores the file from Version 1, keeps you at Version 3, and automatically stages the change.
To switch back to the main path:

```
git checkout master
```

7 Advanced Utilities

7.1 Git Aliases

Shorten common commands using aliases.

```
git config --global alias.s "status"
git config --global alias.cm "commit -m"
```

Now you can type `git s` instead of `git status`.

7.2 Ignoring Files (.gitignore)

To prevent Git from tracking specific files (e.g., study materials like `source.txt`):

1. Create a file named `.gitignore`.
2. Write `source.txt` inside it.

Git will now ignore `source.txt` even if you try to add it.

7.3 Removing Git

To stop tracking a project entirely (delete Git from the folder):

```
rm -rf .git
```