

DS605 Fundamentals of Machine Learning

Academic Year 2025-26 (Autumn)

LAB-1

Getting familiar with Git

"Imagine you're writing an important essay. You save it as `essay_v1.doc`. Then you make changes and save it as `essay_v2.doc`, and then `essay_final.doc`, and finally `essay_REALLY_final_v2.doc`. It's messy and confusing! What if you want to go back to a change you made in v1?

Git is like a time machine for your files. It lets you save "snapshots" (called commits) of your project. You can visit any snapshot from the past, see what changed, and you'll never lose work again.

GitHub is like a cloud drive (e.g., Google Drive) specifically for your Git projects. It saves your project history online, so you have a backup and can easily share and work with others.

Key Idea: Git is the tool on your computer; GitHub is the website that stores your projects.

Part 1: Setup

Before we start, we need two things.

Create a GitHub Account: Go to <https://github.com> and sign up. It's free!

Install Git: Go to <https://git-scm.com/downloads> and download the installer for your operating system (Windows, Mac, or Linux). Accept all the default settings during installation.

Configure Git: Open your terminal (on Mac/Linux) or Git Bash (on Windows) and run these two commands, replacing the text with your own information. This is like signing your work.

```
git config --global user.name "Your Name"
```

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

Part 2: The Core Workflow

This is the main cycle you'll use every day. We'll create a project on GitHub, copy it to our computer, make a change, and send it back.

Step 1: Create a Repository on GitHub

A repository (or "repo") is just a project folder.

On the GitHub website, click the + icon in the top-right corner and select New repository.

Give your repository a name, like my-first-project.

Keep it Public.

Important: Check the box that says Add a README file. This creates an empty file so we have something to work with.

Click Create repository.

You now have your project's home on the internet!

Step 2: Clone the Repository to Your Computer

Cloning means making a copy of the GitHub repo on your local machine.

On your new repository's page on GitHub, click the green <> Code button.

Make sure HTTPS is selected, and click the copy icon next to the URL.

In your terminal or Git Bash, navigate to where you want to store your projects (like your Desktop or Documents folder).

Bash

Example: navigate to the Desktop

cd Desktop

Run the git clone command with the URL you copied.

Bash

git clone https://github.com/YOUR_USERNAME/my-first-project.git

This downloads the project folder to your computer. You can now cd my-first-project to enter it.

Step 3: Make a Local Change

Let's edit the project.

Open the my-first-project folder on your computer.

You'll see the README.md file. Open it with any text editor (like Notepad, VS Code, or TextEdit).

Add a line of text, like "Hello World! This is my first Git project." Save the file.

Step 4: The Holy Trinity - Add, Commit, Push

This three-step process is how you send your changes back to GitHub.

ADD - Stage Your Changes:

First, you tell Git which files you want to save in your snapshot. Think of it as putting files into a box before shipping. In your terminal (make sure you're inside the project folder), run:

Bash

git add .

The . means "add all the files in this folder that have changed."

COMMIT - Create the Snapshot:

Now, you save the snapshot with a descriptive message. This is like labeling the box.

Bash

git commit -m "Added a welcome message to the README"

The -m stands for "message." Good messages explain what you changed.

PUSH - Send it to GitHub:

Finally, you send your committed changes up to your GitHub repository. This is like shipping the box.

Bash

git push

Go back to your repository page on GitHub and refresh. You'll see your changes live!

Part 3: Getting Changes from Others

What if a teammate makes a change? You need to pull their work down to your computer.

Step 1: Simulate a Teammate's Change

Let's edit the file directly on GitHub to pretend we are a collaborator.

On your GitHub repo page, click the README.md file.

Click the pencil icon to edit the file.

Add a new line, like "This change was made on GitHub."

Scroll down and click the Commit changes button.

Your GitHub repo is now ahead of your local copy.

Step 2: Pull the Changes

On your computer, in the terminal, run the git pull command to download the latest changes and merge them into your local files.

Bash

git pull

Now, open the README.md file on your computer. You'll see the new line you added on GitHub. You're perfectly in sync!

Summary: The Cheat Sheet

Here are the essential commands you learned today:

git clone <url>: Copy a repository from GitHub to your computer.

git add .: Stage your changed files for a snapshot.

git commit -m "message": Create the snapshot with a descriptive message.

git push: Send your committed snapshots to GitHub.


git pull: Get the latest changes from GitHub.

git status: Check the status of your changes (a very useful command!).

That's the basic cycle of Git and GitHub! You now know enough to start version-controlling your own projects and collaborating with others. Good luck!

Live Demo of Git :

1. Create a folder at your desired location (in my case I made demo folder).

Name	Date modified	Type	Size
 demo	16-03-2024 11:53	File folder	

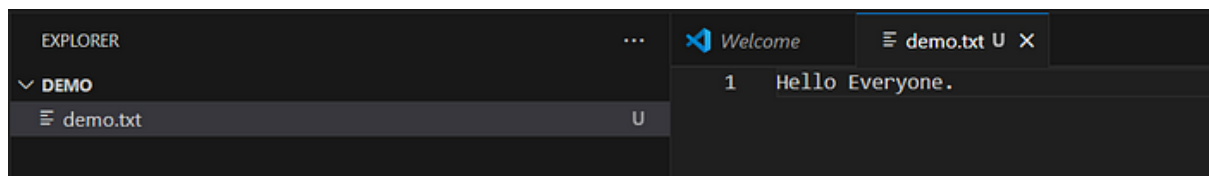
2. Open this terminal inside VS code and in VS code terminal type **git init**.

```
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

D:\data\demo>git init
Initialized empty Git repository in D:/data/demo/.git/

D:\data\demo>
```

3. Now create a file (in my case I create demo.txt) and add in file something.



4. In terminal type **git add .** and check status by **git status**. Congratulations, you add your file in staging area.

```
D:\data\demo>git add .

D:\data\demo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   demo.txt
```

5. Now Commit this file by using :

git commit -m "First commit"

When you type this command and press enter, you will see the following screen :

```
D:\data\demo>git commit -m "first commit"
Author identity unknown

*** Please tell me who you are.

Run

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

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

This tell you please configure your name and email before committing.

```
D:\data\demo>git config --global user.name "sachin"

D:\data\demo>git config --global user.email "sachin123@gmail.com"

D:\data\demo>git commit -m "first commit"
[master (root-commit) 1f4f1de] first commit
1 file changed, 1 insertion(+)
create mode 100644 demo.txt
```

Congratulations, you commit your first file, i.e. it save in you local repo.

6. Now if you want to upload on GitHub(Before this please create your account on GitHub and create there a new repository), and in terminal type below code :

```
git remote add origin https://github.com/sachin0612/demo.git
```

```
git branch -M main
```

```
git push -u origin main
```

Congratulations, your files are reaching at GitHub.

Some important Terminology :

git log: This command shows a history of commits in a repository, displaying information like commit messages, authorship, and timestamps, helping developers understand the project's evolution.

```
D:\data\demo>git log
commit 1f4f1dea428621add29611d6193bc0265922f318 (HEAD -> main, origin/main)
Author: sachin <sachin123@gmail.com>
Date: Sat Mar 16 12:06:50 2024 +0530

    first commit
```

git show : With this command, developers can view detailed information about a specific commit, including changes made to files, providing insights into what was modified and why.

```
D:\data\demo>git log
commit 1f4f1dea428621add29611d6193bc0265922f318 (HEAD -> main, origin/main)
Author: sachin <sachin123@gmail.com>
Date: Sat Mar 16 12:06:50 2024 +0530

    first commit

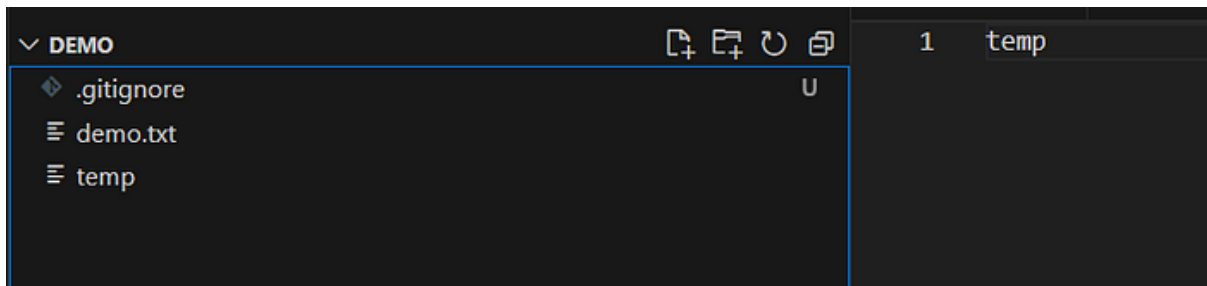
D:\data\demo>

D:\data\demo>git show 1f4f1dea428621add29611d6193bc0265922f318
commit 1f4f1dea428621add29611d6193bc0265922f318 (HEAD -> main, origin/main)
Author: sachin <sachin123@gmail.com>
Date: Sat Mar 16 12:06:50 2024 +0530

    first commit

diff --git a/demo.txt b/demo.txt
new file mode 100644
index 0000000..449d26e
--- /dev/null
+++ b/demo.txt
@@ -0,0 +1 @@
+Hello Everyone.
\ No newline at end of file
```

.gitignore file : This file specifies intentionally untracked files that Git should ignore, such as temporary files or build artifacts, preventing them from being included in commits and cluttering the repository. It helps keep the repository clean and focused on important files.



git branch :

A Git branch is a separate line of development within a Git repository. It allows developers to work on new features, fixes, or experiments without affecting the main codebase. Each branch represents a distinct set of changes, and developers can create, switch between, merge, and delete branches as needed. Branches are important because they enable parallel development, collaboration among team members, experimentation with new ideas, and the isolation of changes for testing and review, all while keeping the main codebase stable and unaffected.

Some important Commands :

To see list of available branches

git branch

To Create new branch

git branch <branch name>

To switch branch

git checkout <branch name>

Files created in workspace will be visible in any of the branch workspace until you commit.

Once you commit, then that files belongs to that particular branch.