Git and GitHub

Week 1

> Adib Sakhawat
> Batch 21
> BSc in Software Engineering
> Islamic University of Technology

1. What is Git?

Imagine you're writing a book with friends. Every time someone writes a new chapter or edits an old one, you need to keep track of who wrote what, so nothing gets lost. Git is like a notebook that keeps a detailed log of every change in your project, making it easy to go back and see past versions.

Key Terms:

- **Repository (Repo)**: This is like your project folder. It contains all your files, and Git keeps a history of every change made to those files.
- **Commit**: A snapshot of your project at a specific point in time. Think of it as saving your work or taking a picture of your progress.

2. Setting Up Git

Before we start, we need to set up Git. It's like getting your notebook ready to keep track of all the changes.

1. **Install Git** if you haven't yet. You can download it from git-scm.com.
2. Open your terminal (a command line where you type in instructions) and configure Git with your name and email so it knows who's making changes.

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

   This is like writing your name in the notebook, so it knows who's contributing.

3. Creating a Repository (Starting Your Notebook)

To start tracking a project with Git, you need to create a repository.

Example:

You're starting a new book, so you grab a notebook. The notebook in this analogy is your **repository**.

1. Go to your project folder in the terminal.
2. Type:

   ```
   git init
   ```

   Now, Git is tracking your project. You'll see a hidden folder called `.git` inside your project folder. This is where Git keeps its record of changes.

---

4. Making Your First Commit (Saving a Snapshot)

Once you have some files (like the first chapter of your book), it's time to save your work in Git.

Example:

You write Chapter 1 of your book. Now, you want to make sure you don't lose it, so you **commit** it.

1. First, tell Git which files you want to save. You do this by **staging** the files.

   ```
   git add .
   ```

   This command tells Git to get all the files ready for saving. It's like putting them on a table, ready to take a photo.

2. Now, commit (save) them with a message:

   ```
   git commit -m "Added Chapter 1"
   ```

   The `-m` flag lets you write a message describing the changes. This is helpful so you and others can understand what's changed later on.

---

5. What is GitHub?

GitHub is an online platform where you can store and share your Git repositories. Imagine it as a library where everyone's notebooks (projects) are stored, and you can see what others have worked on.

You can upload your project to GitHub so that:

1. You have a backup.
2. Other people can see your project or even work with you on it.

---

6. Pushing Your Project to GitHub

To get your project onto GitHub, you need to first create a GitHub account (if you don't already have one).

1. **Create a Repository on GitHub**
   - Go to GitHub and create a new repository by clicking the **New** button.
   - Give your repository a name (like `MyBookProject`).
2. **Connect Your Local Repository to GitHub**
   - In your terminal, link your project folder to the GitHub repository:

     ```
     git remote add origin <https://github.com/yourusernam
     e/MyBookProject.git>
     ```

   - Replace `yourusername` and `MyBookProject` with your GitHub username and repository name.
3. **Push Your Work to GitHub**
   - Now, upload your project to GitHub:

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

   This command sends your work from your computer to GitHub, like uploading your notebook to the library.

---

7. Updating Your Project (Pulling, Branching, and Merging)

As you keep working, you'll make more changes, and maybe others will help you. Here are a few more key actions:

- **Pulling**: If you're working with friends, you'll want to get any updates they make. This is called **pulling**.

  ```
  git pull origin main
  ```

  This downloads any changes from the GitHub library to your local notebook.

- **Branching**: Think of branches like parallel timelines. You can create a new branch to try something new without changing the main story.

  ```
  git branch new-chapter
  git checkout new-chapter
  ```

- **Merging**: When you're ready to add your new changes to the main project, you **merge** your branch back.

  ```
  git checkout main
  git merge new-chapter
  ```

---

8. Summary of Key Commands

- **Initialize a repository**: `git init`
- **Add files**: `git add .`
- **Commit changes**: `git commit -m "Message"`
- **Connect to GitHub**: `git remote add origin URL`
- **Push to GitHub**: `git push -u origin main`
- **Pull updates from GitHub**: `git pull origin main`
- **Create a branch**: `git branch branch-name`
- **Switch branches**: `git checkout branch-name`
- **Merge a branch**: `git merge branch-name`