week 0_2:

git & GitHub

rajin teaches programming

In this session, I'll introduce you to the fundamental concepts of **version control** using **git** and the collaborative platform **GitHub.** By the end of this session, you'll understand the basics of **version control**, setting up a **GitHub account**, and performing essential **git operations.**

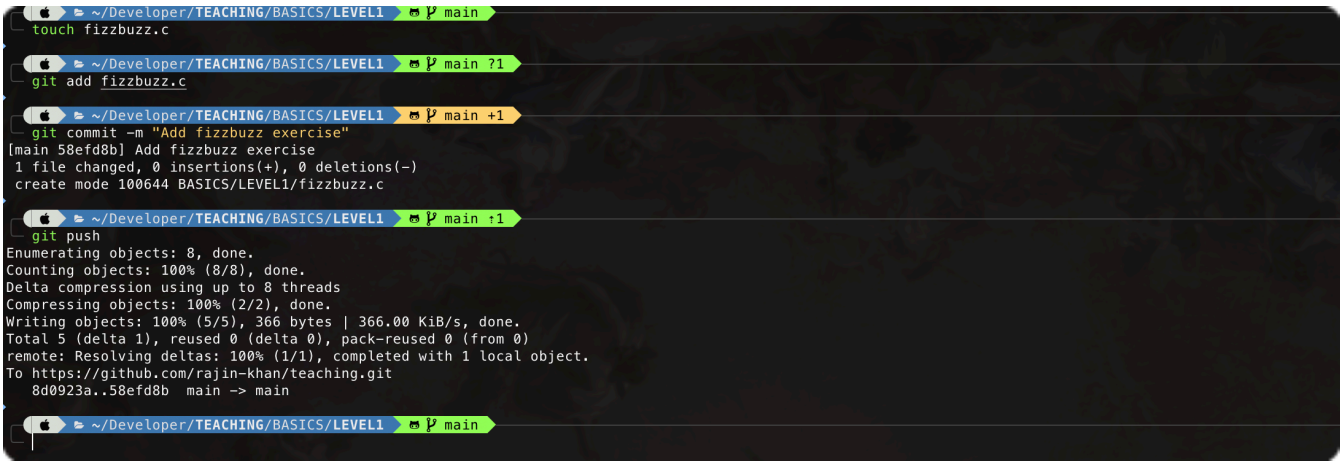
UNDERSTANDING VERSION CONTROL:

– What is git?

Git is a **distributed version control system** used to **track changes** in source code during software development.
It allows **multiple** developers to work on the **same** project **concurrently** while keeping track of **changes**, facilitating **collaboration** and code **management.**

– Why use git?

git enables developers to:
a. **Track changes** and **revert** to **previous** versions if needed.
b. **Work collaboratively** on projects without **conflicts.**
c. Maintain a **centralized repository** for **code storage** and **sharing.**



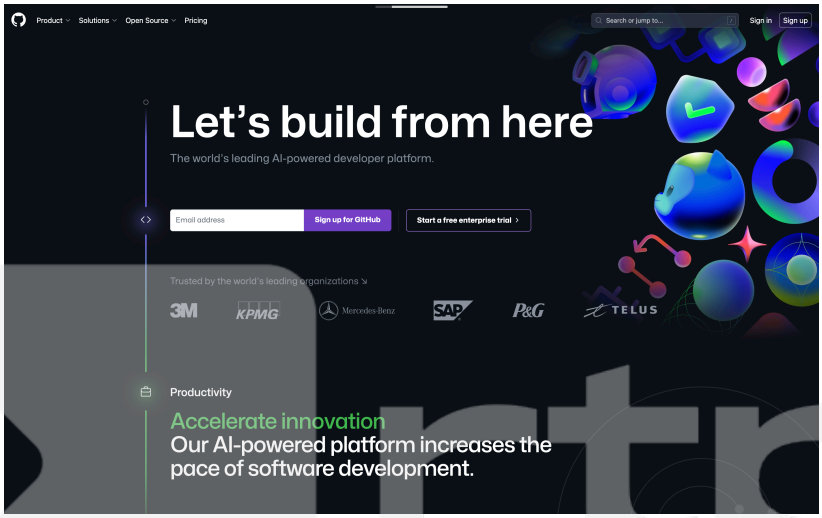
A typical git workflow.

INTRODUCTION TO GITHUB:

- What is GitHub?

GitHub is a web-based **platform** that provides **hosting** for Git **repositories.** It offers features like issue **tracking, pull requests,** and **project management** tools. GitHub serves as a central **hub** for **collaboration** and **open-source** development.

GitHub ->



<- git

- git vs GitHub:

git is the version control system used for tracking changes in code locally. GitHub is a hosting platform for git repositories, allowing collaboration and sharing of code with others.

Getting started with git:

– Installing git on your system:

(Windows)

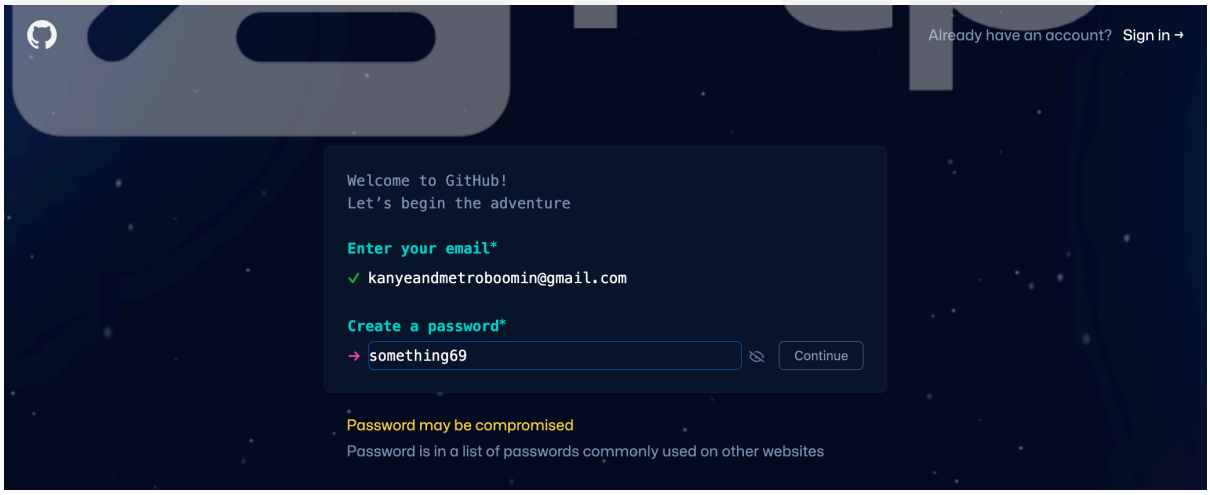
Download git from: **https://git-scm.com/download/win)**
Follow the installation instructions. After installation, open **git Bash** or git CMD to access git commands.

(Mac)

git comes **pre-installed** on MacOS. Just open up your **terminal** and check your installed version using `git --version`. If you don't have the **latest version** installed, just visit the same link and download the latest version.

Getting started with GitHub:

– Setting up a GitHub account:



Visit the **sign-up** page *(https://github.com/join)* and provide the required information.
Choose a **username**, enter your **email** address, and create a **strong password**. Then click "Sign up for GitHub" to create your **account**.

– Connecting GitHub to your computer:

After creating your GitHub account, you'll need to **set up** git on your **local machine.**
git **configuration files** (.gitconfig) store **settings** and **preferences** for **git operations.** Since git is already installed now, open your **terminal, command prompt,** or git **Bash,** and enter the following commands:

terminal

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

Basic git workflow:

– GitHub terminology:

Repository (repo)	A folder or directory where your project's files and version history are stored.
add	to prepare a file for committing, almost like telling git “this file is going to be saved”.
commit	A snapshot of changes made to the repository at a specific point in time (like hitting save progress in a game).
branch	A separate line of development within a repository (like creating a copy of your code so you can edit and add stuff to it, without worrying about editing your main code).
clone	Copying an existing repository from GitHub to your local machine.
pull	Copying an existing repository from GitHub to your local machine.

git operations:

– Initializing a repository:

Initializing a repository is basically telling git that you want it to track the changes and create **“snapshots”** of a specific directory. To do that, open up **git Bash** in the folder you want (right click –> open git Bash here), and enter the following command:

I will explain what this line does in class. Ideally, this alone would be enough to keep track of the files you want, but we want to **connect** this folder to a **repository** on **GitHub**, so we have to enter some **additional** commands. Again, I will explain these lines in class:

After my explanation, you will realize that you **cannot** push anything onto this repo because you have **not added nor committed** any files to prepare them for pushing on to the repo. After doing that, we’re good to go.

Then, we can move on to the usual git workflow.

– Add and Commit changes:

As mentioned previously, you can create **“snapshots”** of your code. To prepare the file for doing so, you have to **add** it to the **staging area**. I will explain this in detail in class. You can do so by entering this command:

terminal

```
git add .
```

Finally, to **commit**, or create a **“snapshot”** of your file, enter the following:

terminal

```
git commit –m “Commit message”
```

Whenever you commit a file, you must give provide a **commit message** too. This allows for you, and other people viewing your code, to keep track of the changes you made, and why you decided to commit, or what the change was from the previous commit.

– Push changes:

Finally, to **push** changes from your local repo on to your remote GitHub repo, enter your last command:

terminal

```
git push
```

If you follow all these steps, along with my explanation, you should face no difficulties so far, and everything should go smoothly.

– Cloning a repo (and pulling changes):

Pretty self explanatory, this will help you create a **copy** of any public repo on to your **local machine.** To do so, enter the following:

terminal

```
git clone *link-to-repo*
```

Cloning will also create a **copy** of the **.git** directory on to your local machine. If you want to **disconnect** this **local** repo from the **remote** repo, simply **remove** this directory. If you are **added as a collaborator**, you may **push** changes on to this repo. However **before** doing so (in order to avoid **conflicts**, which will be **explained in class**), you must make sure your **local repo** is **up to date** with the **remote repo.** You can do so by using the **git pull** command:

terminal

```
git pull
```

IMPORTANT POINTS

To summarize, **after** setting up and connecting your local repo to the remote one on GitHub, your usual workflow would be to to first **add**, then **commit**, and later **push** changes on to your remote repository.

Think of **pushing** as **uploading** your files on to GitHub, and **pulling / cloning** as **downloading** files (or changes).

Additionally, each repo optionally requires you to add a **README.md** (**markdown**) file, which is where you store information about your project and let others know what it is about. Try to code **daily**, and usually you should get into the flow of **3–5 commits** per day. Don't get too caught up on your contribution graph, although showing it off is kind of fun.

These git commands should be enough for you to have a basic understanding of git and manage your own files. However, it is important to know that there is a LOT more to learn about git and GitHub out there. After this introduction, you can go out and learn more on your own as needed. Finally, here's a small cheat sheet to help you remember stuff along with new commands to explore:

command	definition
git init	Initialize a new Git repository in the current directory.
git clone *link*	Clone a repository from a remote URL to your local machine.
git add *file*	Add changes in your file to the staging area.
git commit -m *message*	Commit staged changes with a descriptive message.
git push	Push committed changes from local repository to remote repository.
git pull	Pull changes from remote repository to local.
git status	Show the current status of the repository and changes.
git log	Display commit history.
git diff	Show changes between commits, commit and working tree.
git help	Get help on any Git command.

next class 1_1:

programming basics

rajin teaches programming

reach out for classes at: rajin.khan2001@gmail.com