Git

What is Git?

Git is the most popular version control system.
A version control system records the changes made to our code over time in a special database called repository.
Without a version control system we'll have to constantly store copies of the entire project in various folders.

With a version control system, we can track our project history & work together.

$$\begin{tikzcd}
 & \text{Version Control System} \arrow[dl] \arrow[dr] & \\
\text{Centralized} & & \text{Distributed}
\end{tikzcd}$$

Centralized System: In a centralized system, all team members connect to a central server to get the latest copy of the code & to share their changes with others.
e.g Subversion & Team Foundation Server.

Problem with centralized system: The problem with the centralized architecture is the single point of failure. If the server goes offline we cannot collaborate or save snapshots of our project.

Date
Page

Distributed Systems: In distributed systems, we do not have these problems. Every team member has a copy of the project with its history on their machine.
If the central servers is offline, we can synchronize our work directly with others.
e.g Git & Mercurial

Why Git?

- Free
- Open Source
- Super Fast
- Scalable

Why command line over GUI?

- GUI tools have limitations
- GUI tools are not available always.

Setting:

- Name
- Email
- Default Editor
- Line Ending

We can specify this configuration settings at 3 different levels:

Page

System → All users
Global → All repositories of the current user
Local → The current repository.

· gitconfig →

```
git config --global user.name "___"
git config --global user.email ___
```

To set default editor :

```
git config --global core.editor "code --wait"
git config --global -e
```

End of lines

Windows	macOS / Linux
abc \r \n	abc \n

\n → Line feed
\r → Carriage return

If we don't handle end of lines properly we're going to run into some weird issues down the road.

To prevent this we have to configure a

property called core.autocrlf which is a short for carriage return line ~~fe~~ feed.

$$\begin{tikzcd} & \text{true (windows)} \\ \texttt{git config --global core.autocrlf} \arrow[ur] \arrow[dr] & \\ & \text{input (macOS/Linux)} \end{tikzcd}$$

Initializing a repository:

```
mkdir Moon
cd Moon
git init
    ↳ initialized empty git repository
```

Commit:

ID
Message
Date/time
Author
Complete Snapshot.

Each commit contains a unique identifier that gets generated by git.

Staging Files:

```
echo hello > file1.txt
```
used to create a file with content in it.

Date
Page

git status
To see the status of the working directory and the staging area.

git add file1.txt [more files]...
To add file1.txt to the staging area.
We can also use patterns like *.txt, that means all the files with the txt extension.
We also have . [period] which adds the entire directory recursively.

git commit -m " "
To commit this snapshot to permanently store it in our git repository.
-m stands for message.
-a means all modified files

git ls-files
list files in staging area.

git rm file1.txt *.txt
Git removes this file from both working directory as well as staging area.

git mv file1.txt main.js
Renames files in both working directory as well as staging area.

git log
To look at history.