The history of GIT

Was created by Linus Torvalds to track changes to a project from multiple authors

Who likes the idea of travelling through time?

GIT is a way to travel through time for your project!

What is it?

GIT is a command line tool

We use GIT to keep track of changes to files

We can travel through those changes and see how a project appeared earlier in time

How does it work?

GIT **does not automatically** add your changes, **until you tell it to**

GIT will keep track of any file changes after you've used `git add` to track them

Each change or set of changes are recorded within the GIT repository as a **commit**

You can write a message with each commit

Commit Messages

Optional BUT

Who are you writing these messages for?

NOT for the computer!

- You are writing for your future self
- Co-workers, future developers, anyone else who might be working on the project

What should you write?

- Your message should make sense
- The message should explain in a short summary - what that commit did - what was the purpose

Example of a bad commit message

"Minor changes"

Example of a good commit message

"Fixed bug where video element was being generated as div"

Deleting files

Deleting a file is like any other change, you still have to add the deleted file before committing

> ! Even if you delete a file, it still exists in your history

Cheat Sheet

Bash

`la` / `ll -a` - shows hidden files

Bash commands can be broken down into smaller parts

git = command commit = subcommand -m = flag `git commit -m`

GIT

`git init` - create new git repository

`git add` - adds a file or files to a repository

`git add filename` add a single file `git add .` add EVERYTHING

`git status` - check current status

`git log` - shows you the history of a repository

`git checkout` - allows you to change to a different branch / commit hash

`git rebase` - very dangerous, but very powerful

Configuring GIT

Remember GitHub and GIT are separate entities

When configuring GIT, this has nothing to do with your GitHub account

There is no such thing as a "GIT account"

When you configure GIT;

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

There is no "verification" of that data - you can write anything you like

If you want change any of the configuration settings, you can just run those commands again

Cheat Sheet

Bash

`la` / `ll -a` - shows hidden files

Bash commands can be broken down into smaller parts

git = command commit = subcommand -m = flag `git commit -m`

Local and remote repositories

Local = anything that is available locally on your machine Remote = a repository which lives on a shared network resource

GitHub really shines here - the most post popular and easy to use remote GIT service

`git remote -v` - view the link to the remote repository

You can link to your remote repositories using either the **HTTPS** protocol or the **SSH** protocol

HTTPS

Using **HTTPS** means that every time you want to interact with your remote repository, you have to enter your username and password

SSH

Secure Shell

Before you can use **SSH**, you have to generate an **SSH** key - which will automatically be used when you interact with your remote repository

More work to setup to setup at the start, but saves you time compared to HTTPS

GIT

`git init` - create new git repository

`git add` - adds a file or files to a repository

`git add filename` add a single file `git add .` add EVERYTHING

`git status` - check current status

`git log` - shows you the history of a repository

`git checkout` - allows you to change to a different branch / commit hash

`git rebase` - very dangerous, but very powerful

`git remote -v` - show the remote repository

`git clone` - clone a repository from a remote location (you must include the URL)

> Use the SSH version of the URL, not the HTTPS version 3:52 GitHub tutorial

1. Created a `.ssh` folder in our home directory (`~`)#
2. `ssh-keygen -t ed25519 -C "email@email.com"`
3. Accept the defaults by pressing enter
4. `cat id_ed25519.pub`
5. Paste the public key in GitHub > Settings > SSH and PGP keys