Git Introduction

Checkpoints are practical for working on complex projects that span multiple files. When you use a Versioning Control System, when something breaks (because something will break), you can always backtrack to a previous stage.

Working in software development will have you create files, scripts, and assets that can get lost (or ruined) in the process. On top of this, you will want to add features, while avoiding compromising the entire project. Doing this without a VCS would mean creating a copy of the project (more files = more problems), and if you include teamwork, the issues get multiplied. If two people are working on different features, how do you combine them?

If you have ever worked on a long text, you probably had a problem like this and realized how time-consuming it is. Programmers are ~~lazy~~ efficient, so they came up with Version Control Systems (VCS).

Version Control System (VCS)

Have you ever had different versions of the same file? The meme tells that story:

> Final thesis.docx
> Final Final thesis.docx
> Final Final thesis revised.docx
> Final Final thesis revised print.docx

A VCS is a supercharged interpretation of this rudimentary way of keeping different versions of files. Version control will protect the code in case of wrongdoing, as humans are prone to mistakes. Also, any VCS will help to keep track of changes and merge different versions of the same project. There are many different versions of this, but for now, you will be learning about **Git**.

Read more about it: What is Version Control? | W3 Schools↗

Git

Git is the most popular VCS. According to Open Hub↗, in August 2022, 73% of repositories registered with them used Git instead of any other VCS. Git helps with change tracking and collaboration.

- **Git is not GitHub**, and GitHub is not GitHub Desktop. You do not need to understand the difference right now, as you will get to that later. For now, have clarity: you are learning **Git**.
- **Git uses mainly commands.** However, several Git User Interfaces (GUI) show information in different ways. They help but make sure you understand the underlying command. Knowing commands will provide a solid foundation for your understanding of Git.

Git User Interfaces

At its core, Git is just commands, but, you can find many Git User Interface Clients↗, they give you visual feedback and are easier to get started. Any GUI should have the same underlying workings, so it does not matter too much which one you choose. However, not everything is the same:

- GitHub Desktop↗ is the most known client and is free to use. **GitHub** (without the **Desktop**) is a different thing. Git is usually equated to GitHub because it is the most widely known **Git repository hosting service**.
- Sourcetree↗ is a free Git client for Windows and macOS.
- Fork↗ is a paid client ($49.99, comes with a free indefinite trial *à la* WinRAR). It is very intuitive, and shows information in an easy-to-understand manner.
