Contents

Creating your first repository

Now that you have set up your environment, you are ready to start learning Git. As you progress along the guide, information and references about the concepts discussed will be provided, these will be critical in properly understanding and using Git in your workflow.

In this chapter, you will be familiarised with the basic ideas surrounding Git as well as starting to use Git.

Pre-requisite concepts

Git

Simply put, Git is a version control system. According to the article, a version control system is

> ... a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special database

Git is not the first version control system but it is the most commonly used (?) one in present time. Other version control systems include SVN and Mercurial.

Essentially, a version control system helps to provide a "history" of changes to a piece of code over time. Git, in particular, is well-known as it allows developers to collaborate and work on a codebase without an active connection to a central code repository.

However, in order for collaboration to first happen, Git requires a remote server to store this code history. There are plenty of free options available - one of which being Github which we will explore later - but there are also ways to create your own Git remote server (more details). This guide will not be investigating the ways to do so, but you can learn more about it in the link above. Although, fun fact, some larger organisations will use in-house Git servers to manage their projects as they may require full control over it - something services like GitHub may not offer.

More details about how Git works, but this knowledge should be sufficient for this chapter.

GitHub

GitHub is a company that provides hosting for Git. This means that any code that is uploaded to the Git server is managed and maintained by the GitHub team.

GitHub is one of the more popular version control sites out there, but other options include GitLab and BitBucket.

Now that we have gotten the basic terminology out of the way, let's get into creating your first repository.

First repository

There are two ways to create a repository. This guide will go through both in detail. There are specific use cases for either and it's best that you are familiar with them.

GitHub first

The first approach is "GitHub first". This is where the repository is created on GitHub (creating a remote repository first) before it is made available on your local machine. (Do not worry about the remote vs local idea for now, this will be explained in chapter 3).

First, login to GitHub and in the top-right corner of the menu bar, you should see a drop-down. Expand it and you will see an option for a "New Repository". Select this option.

You will be brought to a page where you can "customise your repository". You will have options to customise the following:

1. Project/repository name (mandatory)
2. Project/repository description (optional)
3. Private vs public repository (mandatory, public by default)
4. README file (optional)
5. License (optional)
6. .gitignore (optional)

This chapter will explain the first three concepts. The remaining concepts will be elaborated upon in chapter 6.

Every time you wish to start a new project, you will create a new repository on GitHub. The term repository has dual meanings, one of which will be discussed in later chapters. However, when referring to a project, the term "repository" can be used interchangeably in this guide. The repository's name is the name of the project that you will be working on. This can be any name that you wish to

give it. However, common convention often has repository names using kebab naming convetion or pascal naming convention. For this guide, the project name chosen will be `learning-git`, but you can choose anything.

Similarly, the project's description is used to provide a brief description of what the project is about.

Lastly, you can specify if the project is publicly-accessible or privately-accessible. For school projects or personal projects that you wish to keep under wraps, you can set the repository to private. This means that no one can view your repository unless you explicitly give them permission to do so. For open-source projects, you can set the repository to public so that other developers can view your code and use it.

Now that you have created a project on GitHub, you can then create a local copy of this project on your local machine. To do so, first open up your terminal or Git Bash and navigate to a folder where you wish to store your project. For now, we will store all projects in a `Projects/` folder found on your user directory (`/home/<user>` for unix-based systems and `C:\Users\<user>\` for Windows). If you are unfamiliar with using the terminal, a quick guide on the basic commands used in this guide can be found in chapter 7 so if you need to, quickly jump over to familiarise yourself with them.

```
cd ~/Projects/
```

The process of creating a local copy of the project is known as "cloning". Where you are essentially cloning the current version of the repository onto your local machine where you can perform changes and then upload it back to GitHub or any other Git server. In your empty repository page, you will notice that a URL is provided. Copy this URL to your clipboard and go to the terminal. The URL should have the following format: `https://github.com/<username>/<project name>.git`. For example, `https://github.com/woojiahao/learning-git.git`.

```
git clone https://github.com/woojiahao/learning-git.git
```

You will observe some logging messages show up. Once completed, check that the project has been cloned. `ls` should display that a new folder, `learning-git`, has been created in the folder you have navigated to.

```
ls
```

Finally, you can navigate into your local copy.

```
cd learning-git/
```

Local machine first