Contents

First commit

Now that you have a repository setup, you can start to save and upload files. Before that, it is important to preface that this will be the most important chapter in understanding Git. As such, please read through each section carefully and properly understand how each command works.

To understand Git, you have to understand the core concepts in play.

Core concepts

Local vs remote

In the previous chapter, we referred to a repository on your local machine as a **local repository** while one that is created in GitHub as a **remote repository**. We had even added something called a `remote` to the local repository when using the "local machine first" repository creation pattern.

A local repository refers to a project residing on your local machine while a remote repository is one stored on a version control site like GitHub. When working with Git, a lot of the work performed will be done in your local repository. It can be thought of as a local copy of the code, unaffected by external changes to the remote repository unless such changes are pulled or "downloaded" from the remote repository.

A remote repository is like a central hub for any changes made on any number of local repositories. While a local repository can have any number of changes, these changes are not reflected on the remote repository unless explicitly pushed or "uploaded" to it. This is what makes Git an extremely powerful tool for developers - the ability to mess with the local copy of a repository without impacting the remote repository. Git goes a step further and allows users to roll back any changes in the event where there is a breaking change.

This then begs the question of how do we get our code from the local repository into the remote repository.

Note that it is possible to have more than one remote per project, this is a concept that we will return to in future chapters.

Staging area

This is a concept that we will be exploring in practice later in this chapter, for now, let us provide a theoretical understanding of what is going on between the local repository and remote repository.

The staging area in Git is similar to the one you might imagine in schools or convention centres. It is an area where the spotlight is cast on a select few files (or people). The process of casting this spotlight on something is known as tracking. Once a file is tracked (by being added), it will be tracked for **all** changes that occur to it subsequently. This file will be staged for confirmation.

Only files that have been tracked/staged can be committed. Committing tells Git that you are confirming a set of changes and finalising these changes in the project's history.

Only files that have been committed can be pushed to a remote repository.

If a local file changes while being tracked, Git will inform you that you will have to re-add the file. This is not to re-track the file, but rather to add the latest changes.

If this seems confusing, don't worry, once you start using the commands, it will be easier to understand how Git works.

The following diagram illustrates the general process files of new files in the staging area.

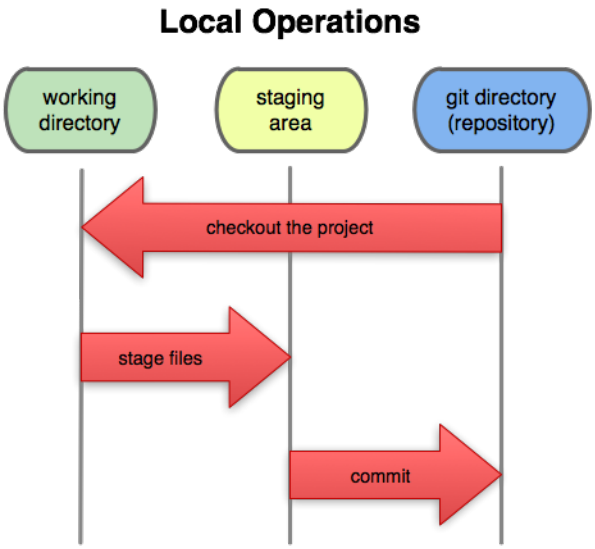

Figure 1: Staging area