Basic Workflow:

Git is a version control tool to save versions of your code. It allows you to *keep track of changes* made to a project, **storing** those changes, then allowing you to reference them as needed.

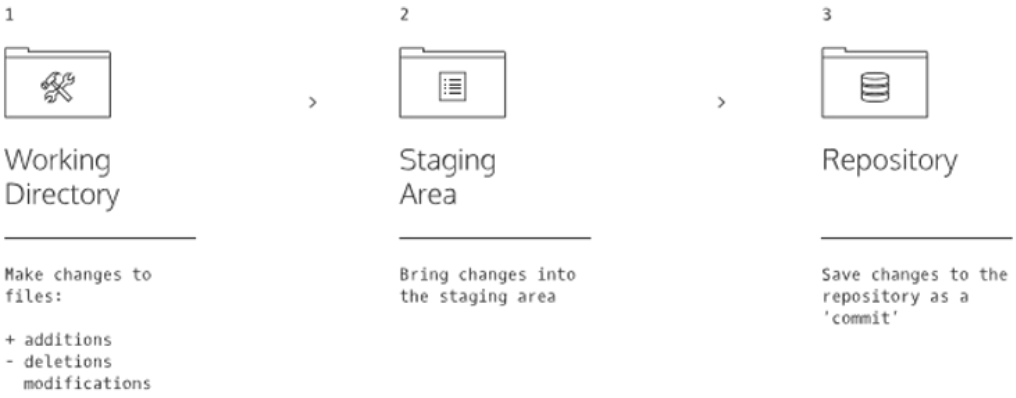
To *initialize* a ***Git Project***, type in the Terminal:

```
$ git init
```

A Git Project can be thought of as having **three** parts:

- A ***Working Directory*** where you'll be doing all the work: *creating*, *editing*, *deleting* and *organizing* files;
- A ***Staging Area*** where you'll *list* changes you make to the *Working Directory*;
- A ***Repository*** where Git permanently *stores those changes* as different version of the project.

The Git **Workflow** consists of *editing* files in the *Working Directory*, *adding* files to the *Staging Area*, and *saving* changes to a *Git Repository*.



In Git, we can save changes with a ***commit***. Whether you have *changed the content* of the *Working Directory*, you can check the ***status*** of those changes with:

```
$ git status
```

In the *output*, notice the file in red under **untracked files**. *Untracked* means that the Git sees the file but has not started *tracking changes* yet.

In order to Git start *tracking* a file, the file needs to be *added* to the *Staging Area*. To do it so, we must type in the Terminal:

```
$ git add [filename]
```

Now, if you check the *status* output of the project in Git, you'll notice that Git indicates the **changes to be committed** with "***new file: [filename]***" in **green** text.Here Git tells us that the file was **added** to the *Staging Area*.

Whether a file is *tracked*, you can check the **differences** between the ***Working Directory*** and the ***Staging Area*** with:

```
$ git diff [filename]
```

Notice its *output*, the text in **white** is the current file content in the ***Staging Area***. *Changes* to the file are marked with a **"+"** and are indicated in **green**.

A ***commit*** is the last step in our *Git Workflow*. A ***commit*** permanently **stores changes** from the *Staging Area* inside the *Repository*.

```
$ git commit -m "[commit message]"
```

The option **"*-m*"** means **"*message*"** and have to follow the *Standard Conventions for Commit Messages*, being they:

- Must be in **quotation marks**;
- Written in the **present tense**;
- Should be **brief** (50 characters or less) when using **"*-m*"**.

Often with Git, you'll have to **refer back** an earlier version of a project. *Commits* are stored **chronologically** in the *Repository* and can be viewed with:

```
$ git log
```

In the *output*, notice:

- A *40-character code*, called a ***SHA***, that uniquely identifies the *commit*. This appears in **orange** text;
- The commit author;
- The date and time of the *commit*;
- The *commit* message.

+ References...

> Basic Git Workflow
> An introduction to Git and a few of its core features