Using Git Locally

In this lesson, we'll show Git in action!

We'll cover the following

Git jargon

Git uses very particular vocabulary, and familiarity with it will make it easier for you to communicate with your team members and the online community. Here are some common terms,

Repository

To put it simply, a project is a repository. Git repositories or 'repos' contain all of the code and version history of a certain project.

Working directory

The working directory is the folder on your local computer where your project exists. Git would track any changes made within that folder.

Commit

Git does not save or store any changes made to the files within your working directory until you ‘commit’ it. Commits save the changes you made to Git itself.

Staging

However, suppose you made changes to 8 files within your working directory, but you only want to commit 4 of them because the other 4 are buggy or not complete yet. How do you commit only 4? Well, you put them in the ‘staging area’ after which you commit. Staging a file means that you have marked it for a commit.

Checking if Git is installed

If you’re following along on a local set up (you don’t have to, but just in case you are), start by checking if Git exists on your system with the following command.

```
git --version
```

If it is installed; great, if not, install it for Mac and Windows if you’d like. Otherwise, follow along here!

Setting up and starting a new Git repo

To mark a directory as a Git working directory, call the following command in that directory

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
git init
```

Git should be tracking any changes we make within this folder now. So let’s add a file and see if Git notices anything with

```
git status
```

```
git init
touch index.html
git status
```

So Git noticed all of the files in our directory!

Also, note that each of our coding playgrounds is like an individual virtual machine, that gets created and destroyed upon execution, which is why we have to initialize a repo each time! This is also why so many ‘random’ files already exist. Also, to make the console output less cluttered, pass the quiet flag like `-q`.

Adding new files to the staging area

Add new files to the staging area with:

```
git add folder/that/contains/files
```

A Git status call on line **4** would show us all the changes to be committed.

```
git init -q
touch index.html
git add .
git status
```

Committing the files

Commit the files with,

```
git commit -m "a message to commit with"
```

```
git init -q
```

```
touch index.html
git add .
git commit -m "Our first commit!"
```

Checking commit history

to print out Git's commit history, type

```
git log
```

The output is something like

```
__ed_create_user.sql
__ed_destroy_user.sql
__ed_javaRunner.sh
__ed_script.sh
__ed_sql_runner.sh
index.html main.sh output

commit 3094b0bcf3483de4955be7c4c15a0b65c3e765b3
Author: Your Name <you@example.com>
Date: Wed Jan 2 07:38:11 2019 +0000 Our first commit!
```

Where the first few lines represent files that were modified or added, the numbers after the `commit` field represent the hash value of the commit (a unique string that identifies the commit). The `Author` and `Date` fields contain information about the author, time of commit, and the message the author sent with the commit.

```
git init -q
touch index.html
git add .
git commit -m "Our first commit!" -q
git log
```

Reverting to a previous commit

To revert your working directory to any previous commit, type the command

```
git checkout hashvalue
```

We can't try reverting with hash values on our platform because a different one is generated for each time the code is run. So we'll demonstrate reverting to the previous state in the following. Notice how even though `index.html` is removed on line **5**, it is restored with Git checkout.

```
git init -q
touch index.html
git add .
git commit -m "Our first commit!" -q
rm index.html
git checkout -- .
ls
```

Test your Git knowledge!

1 What happens if the following command is executed after a fresh install of Git?

```
git init
```

- A) A new repo would be initialized
- B) A repository would get created on GitHub
- C) It would cause an error

COMPLETED 0%

1 of 3

So with Git, you can restore files back to any previously committed state even if they are deleted or modified. However, if your machine gets stolen, crashed, or lost, you still lose your project even with Git. Also, you still can’t truly collaborate. For that, we’ll look at how you can host your repo on another server in the next chapter!