As you now know, GitHub is a cloud based hosting service,

that lets you manage Git repositories from a user interface.

It's like a social network.

You can follow users or accept code contributions from anywhere in the world.

In a previous video, you created and cloned a repository to your local device.

I am now going to explain to you how to pull the repository to your local device.

I will be demonstrating commands that you can use in GitBash shell for

Windows users, and terminal for Mac users.

This refers to the application where the commands are typed in.

Let's move to the directory I want by typing cd, and

the name of the directory my-first-repo.

Once inside the directory, run the list all command by typing ls- la.

-la is short for list all.

There are four items in this directory.

I will focus on two of them, the git item, and the README.md item.

Let's start with the README.md file.

This item was added when I created the repository on GitHub.

The other item,

is a folder called .git, which is a hidden folder used to track all the changes.

In Linux, any folder starting with a dot is a hidden folder.

This folder is automatically created when you create a repository.

And you will learn more about it later in this course.

In the command I ran I ended the switch -la.

So we would list all files and folders including the hidden ones.

The .git folder is initialized by running the git INT command.

As the repository was created on GitHub, it was not required for us to run it.

GitHub handled all of this as part of its create new repo flow.

Now, let's focus on Git Workflow.

Git uses workflows which can be broken into three states namely,

modified, staged, and committed.

Now, I will go over each state and then provide an example of adding a new file to

my gift repository to show it in action.

Let's start with the first state, adding removing and

updating any file inside the repository is considered a modified state.

Git knows the file has changed, but does not track it.

This is where the staging state comes in.

Let's turn to it now.

In order for Git to track a file, it needs to be put in the staged area.

Once added, any modifications are tracked.

Which offers a security blanket prior to committing the changes.

Then, the last state is the committed state.

Committing a file in Git is like a save point in many ways.

Git will save the file, and have a snapshot of the current changes.

Let me introduce you to an example that summarizes the workflow clearly.

Suppose you have a workflow that contains the three stages just mentioned,

as well as the remote repository,

a file is added from the working directory to the staging area.

From there the file is committed, and then pushed to the remote repository.

From the remote repository, the file can now be fetched and

checked out, or merged to a working directory.

You will learn more about this later.

Well done, you've covered some of the Git fundamentals and

now know what is inside a Git folder and understand the Git Workflow.

.get any folder start with . it’s a hidden folder

You are working on a project and have to share some code with a colleague.  What is the correct order the code will flow in Git workflow?



Remote repository, working directory, staging area, committed files



Working directory, staging area, committed files, remote repository



Staging area, committed files, remote repository, working directory



working directory, committed files, staging area, remote repository

Incorrect

Not quite! The correct workflow is from the working directory to the staging area, then to committed files, remote repository.

Correct

That's correct! The correct workflow is from the working directory to the staging area, then to committed files, remote repository and lastly to your colleague.