Previously you learned about the hidden folder called dot git.

That is located in each project.

You know that this folder is responsible for

keeping track of all changes across the project.

How does it know what branch you're currently on?

Let me explain.

It keeps a special pointer called head which is one of the files inside the dot

git folder.

This file refers to the current committee you are viewing.

You will now learn how to identify the current committee you are working on.

First, open the dot git folder on your terminal type cd.git and press enter.

Next type cat HEAD and press enter.

In kit we only work on a single branch at a time.

This file also exists inside the dot Git folder under the refs

forward slash heads path.

Let me show you, type cd.git and press enter,

next type cat forward slash refs forward slash

heads forward slash main press enter.

After you entered this command, a single hashed ID appears,

the single hashed ID is a reference for the current commit.

Let's switch branches to see how the head is moved to point to a new branch.

Typed git checkout testing and

press enter next type get branch and press enter again.

This moves the head to point to the testing branch.

Let me explain how this happens by using a diagram.

We have two branches, the main and testing branch.

When you run the check out command, it moves the head to now point to the testing

branch, to check the contents of the head file inside the dot git folder.

You have to enter the last command, type less dot git slash head and press enter.

You can now verify that the head has changed from main to testing.

Now I will demonstrate how git head works with a simple example.

So I am here in the terminal, to see what branch I am in.

I am running get branch when pressing the enter key,

I can see I'm on the main branch to confirm that I run the cat dot git

forward slash head command and press enter.

That brings me back to the reference to where it actually points to,

namely; ref ,refs forward slash heads forward slash main.

In this case you can see the references pointing to the heads main.

If I change my branch to feature testing branches above I use

the git checkout command.

Get checkout feature forward slash testing branches.

I then go to my head file inside the git folder by typing cat dot

get forward slash head.

The ref is now pointing to the feature testing branches namely: wraps

forward slash head forward slash feature forward slash testing branches.

Notice that my branch is up to date with origin forward slash feature

forward slash testing branches.

I'll go back into my main branch by typing git checkout main and then check

the reference file inside the main directory using the cat command again.

Cat dot git forward slash refs forward slash heads forward slash main.

When I press enter I get a hash ID this is a reference

to the latest commits of that working directory.

I can show you that if you make a change to any files within this directory that

this ID.

Will then change after a commit has happened.

So let's do a simple update to the read me file.

You can do this with any editor such as VS code or

I can also do this by executing the Vim command.

Type Vim read me dot MD and press enter, went inside the read me file and

some text in this case add minor update to your my first repo line and save it.

Then check the ID again.

Just to verify type cat dot git forward slash refs

forward slash heads forward slash main and enter.

If you would like to learn more about them.

There is a link to an additional reading at the end of this lesson.

The ID should be the same because we haven't committed anything.

We've just made a change to run the cat command.

Therefore the ID is exactly the same as before.

If I do a get status, it is telling me that I have modified the read me file,

on the screen the words modified read me dot MD displays in red.

I'll now add that file.

The shorthand to add a single file is get add space dot type

the command and press the enter key.

I am then going to do a commit type git commit dash M for

message by adding minor update to the command line and press enter.

The data confirms that one file changed.

There was one insertion and one deletion.

I am using the cat command to verify what is in the reference file by typing

cat dot git forward slash refs forward slash heads forward slash main.

Press the enter key to confirm that the idea has changed.

Originally the ID started with an eight B55 and

now it starts with 9 C90.

Whenever a change occurs for a commit this ID will then update to

be the latest commit for that working directory.

And that's get head.

You now know what the purpose of head is.

You can also change the head to point to a different branch.

Whenever a change occurs for a commit, the single hashed ID will update to be the latest commit for that working directory.



True



False

Correct

That's correct.  Whenever a change occurs for a commit, the single hashed ID will update to be the latest commit for that working directory.