I've opened my command line.

I should check to ensure that I'm in

the correct directory using the PWD command.

I can see that I'm in the my-first- repo directory.

Now it's good practice to perform

a git status command to make

sure that I have no commits outstanding.

If there are no commits and the shell is clear,

then my branch will show as up-to-date with

origin main outside of the main branch itself.

My next step would be to create a new branch.

To create this new branch,

I use the git checkout command

by typing git checkout dash

b. I then call

this new branch feature forward slash lesson,

which I'll refer to as

feature lesson for the purpose of this video.

But this is just one way to create a branch.

I could also use git branch and pass in the name as well.

These methods are the same and

can both be used to create a branch.

The key difference between them is

that git branch just creates a branch.

But git checkout dash B moves

me from the main branch into the branch that I created.

I can verify that I've been moved between

branches by running the git branch command.

This will then tell me if I have switched from

the main branch to one of the feature lesson branches,

any changes that I make will now

only occur in this new branch.

It's important to remember that

the main branch has no indication or knowledge

of any of these changes even when I

push code to the main repository,

this is because that branch exists in isolation.

The new branch needs to be merged back into

the main branch to recognize

changes in the feature lesson branch.

This is where it'll come in with a pull request.

The purpose of a pull request is to

obtain a peer review of changes made to the branch.

In other words, to validate that

the changes are correct when coding,

many teams will have conditions around

the unit tests and integration tests.

These conditions will usually include validating that

the standards have been met for

merging back into the main line,

a team will also check for

any issues that might have been missed.

The next step is to add a file to the new branch.

I can create a simple text file called

test2.txt using the command touch test2.txt.

Then I add it using the git add command.

I then commit it using the git commit command.

Once I've committed the new file,

I need to push my changes up to

the remote repository with git push.

I type git push- u origin feature lesson.

It's good practice to specify dash U.

This means that I'm only going to

get updates from the upstream,

which in this case will be the main branch.

The result of this is that

the origin won't be my main branch anymore.

Instead, it's feature lesson.

I press the enter key and this pushes

the new branch up to the remote repository.

As I am using https,

I will be prompted for my login information.

Once this action has been completed,

GitHub will recognize that a new branch has been added.

It will then prompt me to create a poll

requests that can be compared against another branch,

in this case, the main branch.

My next step is to open the GitHub UI.

GitHub will show my new branch with a prompt.

Click on the compare and pull request button.

A pull request lets the team

know that I've made new changes that I want them

to review and that I also want to approve or

request changes to the actual poll request itself.

Another thing to note on

the GitHub UI is that I'm

comparing this with the main branch.

I've essentially cut a branch from

the main called feature lesson.

I've then added a new file called test2.txt.

It's this file which is

the main difference between the two branches.

Next, I check the pull request.

I can see that there has been one commit.

In other words, just one file has been changed

and it's been added as test2.txt.

I then click create pull request.

The team will then review the changes

and either approve or decline them.

Once approved, you can then

merge your changes to the main branch.

This is much cleaner than

everyone working off the main branch.

It's particularly useful if you have

features that are closely tied together.

For example, you could be

working with a feature that crosses over with

some code or requires changes that

most likely will be altered by someone else.

The approach of keeping everything at branch level is

much easier than having

everyone working from the main line.

In fact, everyone working off

the same branch is more likely to cause issues.

Having independent branches makes

the project easier to manage.

Also, there's no limit to how many branches you can have.

The team decide on the naming conventions that they use.

In a lot of cases when adding a new feature,

you can add the keyword feature

then followed by the branch name,

like a URL path,

such as feature forward slash lesson in this example.

For bug fixing, fix forward slash can be used.

Because we have no reviewers in my current project,

I'm just going to merge the branch.

Then I'll confirm the merge.

Once confirmed, I'm presented

with the option to delete the branch.

On your own projects,

it's up to you whether you want to

keep the branches or delete them.

For now, I'm going to keep the branch.

Then I can return to my code where the

test2.txt file has merged into the main branch.

I can then confirm that by going back to my command line.

Next, I look at git status

again to check if there is something to commit.

At this point, there's nothing outstanding.

I'm still in the feature lesson branch.

I can check out my main branch by

typing git checkout main.

Then I run the git pull command.

I'll then receive the latest changes that were merged

in from the feature branch that I just created.

Notice that the test2.txt file is now available.

I can also verify that by doing

a simple check within

the directory by using the ls command.

This returns a read-me file test.txt and the test2.txt,

which is from my branch.

You have now learned the branching workflow,

which will use regularly when

collaborating with other developers.

Some Commands

Git checkout \_b feature/lesson

Switch to other branch

Gir branch

See the branches

True or False

The purpose of a pull request is to obtain a peer review of changes made to the branch. In other words, to validate that the changes are correct.



False



True

That’s correct! When you create a pull request you are asking the other developers to review your work and approve it to be merged with the repository.