Have you ever applied for a job?

You know the process, prepare your resume, look for jobs,

submit an application, prepare for interviews.

That is an example of a workflow,

in computer programming workflows are really important.

By the end of this video, you will be able to describe what a workflow is, and

you will also be able to identify different workflows available.

Now, let's start with an example to illustrate why workflows are important.

As a developer working on a project, you first need to pull the project down from

a remote repository to your local machine.

This is commonly called checking out a project or pulling a project.

Once on your local machine, you can build and run the project and make changes.

When you're done, you have to push the changes

you made back to the remote repository so other developers can see them.

From this example, you can understand that the purpose of a workflow is to guide you

and other members of your team.

It should not disrupt or cause blockers for deployment or testing or for

any other developer who contributes to the project itself.

Choosing a workflow needs careful consideration.

It can depend on the size of the team, the culture of your workplace and

also the type of product you intend to build or update.

With that in mind, let me explain feature branching,

a common workflow used by many developers.

Feature branching means you create a new branch from the main line and

work on this dedicated branch until the task is completed.

Rules and conditions need to be made in order for

this branch of code to be kept in a good state.

Every code base has a main repository which is essentially the source of truth

for the application.

All changes such as add, edit or delete are submitted

directly to the feature branch, the main branch stays as is.

When you are ready and happy with the code you have added,

you have to commit the changes and then push to the server repository.

To commit, you push the changes and as it's a feature branch,

a pull request follows.

The pull request is compared to the main branch, so

developers who peer reviewed the code can see exactly what was changed.

Once it's reviewed and approved, it can then be merged into the mainline.

Now let me guide you through how this works using Git and Git hub.

Before creating a new branch, always ensure you have the latest code.

You can do this by running the git pull command to pull

the latest code from the remote repository.

Next you need to create your new branch.

You can do this by passing the -b flag with the check out action.

Next, let's add new content to this branch.

Let's create a README.md file, to do this type git add.

or git add README.md and press Enter.

Next, we need to commit the new file and provide a meaningful message so

other developers can see what you added.

To do that Run the git commit command with a -m option to include

a message with a short description of the changes being committed.

The file has now been added to the local branch,

this means that the file is only visible locally to you.

To allow other developers to see the changes,

you need to push the file to the remote repository.

You can do that by running the git push command and referencing the new file.

The changes are now pushed to the remote repository on git hub.

Your next action is to get a review as part of a pull request, but

more about that later.

And that brings us to the end of this video.

Now, you know what a workflow is and how a feature workflow works, well done.

As part of a development team tasked to change an existing app, you have created a feature branch. Is it possible for your team members to see the changes you are currently working on?



No



Yes

Incorrect

Not quite. The branch you created is only visible to you locally. You have to commit your changes and push the accepted changes to the main server before your team members can see them.

Correct

That's right. The branch you created, is only visible to you. You have to commit your changes and push the accepted changes to the main server before your team members can see them.