CS 240: Creating a Local Git Repository Transcript

[00:00:00] So the next thing that we need to learn about once we've got get installed and configured is how to create a get repository.

[00:00:08] Now, we use this, this word repository.

[00:00:10] When we say repository, we're really just referring to your, the folder of files that make up your project.

[00:00:19] So that's what we mean when, when we say repository.

[00:00:22] So the first thing we want to do with GIT is typically to create a new project that is configured for GIT.

Start visual description. The professor demonstrates how to create a new project folder named "CS 240" and a subfolder called "chess" within it. The professor shows the directory structure on the screen, highlighting the newly created folders. End visual description.

[00:00:30] And so in this case, I'm just going to create a folder here, I've already created the folder named CS 240.

[00:00:36] So let's just assume this is the folder that I'm going to keep all my CS 240 projects in and I've already created that. And then inside of that, I'm just going to create a sub folder called, let's just call it uh chess.

[00:00:52] Now, of course, that's relevant to this class.

[00:00:55] So now that I've got my chess folder created, I'm going to change directory down into the chess directory and I'm going to now initialize it to use GIT. So right now, it's not set up to use GIT.

[00:01:06] So to, to initialize your project to use GIT, you need to run the command, get space in it.

Start visual description. The professor demonstrates the command git init to initialize the project for Git. The terminal screen shows the command being executed and the creation of the. git folder within the "chess" directory. End visual description.

[00:01:14] And if you run, get space and net and just hit enter it will set up all the things you need to, to use get with your project.

[00:01:23] Now, really all it does when you do that, if you look at the contents of the chess folder, now, all it really did is it created this subfolder called dot git.

[00:01:36] So that's how, you know if a, if a project is set up to use Git, if you find that dot git folder in there.

[00:01:43] Now, the tricky thing is that operating systems will typically hide folders that whose names start with A dot Like in this case, if I just list the contents of this folder, it doesn't show dot G.

[00:01:56] And so really to see the dot Get folder, you need to ask your operating system to show you everything that's in the folder.

[00:02:04] So that's, that's how I can see that I have a dot Get folder.

[00:02:07] So what is the dot Get folder? Well, that's the place where G stores all the information about the history of your, of your project and its files.

[00:02:16] And so if we actually go into the dot Get folder, we can see here that, you know, it's got all kinds of, of subdirectories, it's got files that are mysterious and things we don't know about and it doesn't really matter to us what's in this folder, just

know that this is the, the place where get stores all the information about your project that it needs to remember about it.

[00:02:40] And so when I, when I said that G lets you go back in history to any previous version of your, your files, then this is where it would be storing that information.

[00:02:51] There's also a file in here called Config and that's where you can store project specific settings if you don't want to store your settings globally.

[00:03:02] OK? So now that I've got my, my project initialized to use yet, what I'm going to do next is I'm going to run a command called get status.

[00:03:18] And so get status as a command, you're going to run frequently to just um see which files have been modified since the last time you saved your changes.

Start visual description. The professor demonstrates the git status command to check the status of the project. The terminal screen displays the output, indicating that there are no changes yet and the project is empty. End visual description.

[00:03:28] And so in this case, it basically says nothing has happened in inside this project yet.

[00:03:33] It's all empty.

[00:03:35] And so what I'm going to do here is I'm actually going to create a file.

[00:03:41] So let's call it.

[00:03:42] Um This is my first file and I'm going to store that string in a file name notes dot MD.

[00:03:56] So now I have my first file in my project.

[00:04:00] And so now if I run get status, you can see that it says, oh, you've got this file here that we don't know about this file is not actually part of the official G repository at this point.

Start visual description. The professor demonstrates creating a new file named notes.md and running the git status command again. The terminal screen shows the new file as untracked, meaning Git does not yet recognize it. End visual description.

[00:04:13] I created it and it's there, but I haven't officially added it to the get repository yet.

[00:04:18] And so that's what the status command is showing me here.

[00:04:20] It's saying that this file is currently untracked, which means git doesn't know anything about it.

[00:04:27] All right. The next thing I want to do, um because a single file doesn't, isn't very realistic as far as projects go is I want to copy in some existing files. Um So that we can have a more realistic project.

[00:04:40] Now, the files that I'm about to copy into to this project are the chess starter files that we give you at the very beginning of the chess project.

[00:04:50] And so you'll, you'll become familiar with the files that I'm about to copy in.

[00:04:53] But I've already downloaded the chess dot zip file from um the course GitHub.

[00:05:01] And so I'm going to go ahead and unzip that file right here.

[00:05:07] And so all the contents of that file will be added. So now my project has lots of stuff in it.

Start visual description. The professor demonstrates unzipping the chess.zip file into the project directory. The screen shows the extraction process and the addition of multiple files and folders to the project. End visual description.

[00:05:13] I've got files, I've got folders and, and, and lots of stuff inside those folders.

[00:05:18] So this is a more realistic project. So now if I run the get status command again, you can see that get now says, oh, there's, there's a bunch of stuff here that I'm not familiar with now.

[00:05:30] So it's basically saying I, I'm not aware of anything in this project.

[00:05:34] And so, um that's where we, we begin.

[00:05:38] Now, this is an Intel J project. And so Intelli J of course, is the ID E that we're using in the class to do our job of programming.

[00:05:45] And so now that I've copied these files into my project, I can actually open Intel J and ask it to open this, this project.

[00:06:17] So there's the chess project that I just created, and I'll open it and Intel J, there we go.

[00:06:30] So you can see here that I've got lots of files and they're all in my project. I could actually uh compile and run um these programs.

[00:06:40] And so now I've got a project. It's, it's initialized use GIT, and I've got a bunch of files in there, but GIT doesn't know about those yet.

[00:06:49] So the next thing we'll learn about is, is how to use GIT commands to actually put them in, in the repository.