CS 240: Commits Transcript

[00:00:00] When you're using GIT, um there's a word that's used a lot, which is the word commit.

Start visual description. The professor demonstrates how to use GIT by explaining the concept of a commit. He shows the process of creating a project and copying files into it. End visual description.

[00:00:06] So if we're going to understand how to use GIT, we're going to need to understand what a commit is and why that's important.

[00:00:13] So as we saw earlier, we created a project, we, we copied in some files into the project.

[00:00:22] We ran the get status command and GIT says, I don't know anything about all these files that are there.

[00:00:28] And the reason it doesn't know about them is because we haven't added them to the repository yet.

[00:00:33] So once you've created a file or a, um a bunch of files, the first thing you want to do is run the command, get a and so the process of adding files to the repository um really just says, hey, get, this is a file that I want to actually put in the repository and I want you to track its history from here on.

[00:00:54] You may have some files in your project folder that you don't want to put in get permanently.

[00:00:59] And so what you have to do is you have to tell which of the, all the files that I have do, I actually want to put in the repository.

[00:01:06] And so the get a command can be used to do that.

[00:01:10] And let’s assume that we, you know, have created a bunch of files and then we call, get a, to add all of them to the um repository.

[00:01:19] And then once we’ve added the files to the repository, the next thing we need to do to make that permanent is we need to run the command git commit.

Start visual description. The professor demonstrates adding files to the repository using the git add command and explains that this command stages the files for committing. End visual description.

[00:01:27] And so when we were run the get commit command that takes those files and makes them a permanent part of the repository.

[00:01:36] And so nothing’s permanent and get until you’ve committed it.

[00:01:40] And so in the get world, the word commit can be used as a verb or as a noun.

[00:01:45] So as a verb, the word is used it, it refers to the act of, of saving changes to the git repository.

[00:01:55] So I’m going to commit my changes. I’m going to commit my files to get that just means I’m saving those, those files or those changes to, to the git repository permanently.

[00:02:05] Now, the G or the commit word can also be used as a noun.

[00:02:10] So if I take a bunch of changes that I’ve made to my files, I can commit all of those changes as a unit.

[00:02:18] And so when we take a bunch of changes that we've made to our project, and we commit those changes to the repository, um that group of changes is called a commit.

[00:02:30] So when somebody says, refers to a commit, what they're really referring to is all the changes that were, were committed as a group to the repository.

[00:02:40] So just think of it this way, when you're working on your project, you're going to make changes to existing files, you're going to add new files, you're going to rename files, you're going to delete files, you're going to, you're going to do all kinds of stuff.

[00:02:52] And then at some point, you're going to want to take all those changes that you've done since the last time you committed and you're going to want to, you're going to want to commit those changes uh to the repository and all those changes that you made as a unit would be called a commit.

[00:03:07] This slide shows the process of making a commit in your get repository.

[00:03:14] So at the top here, we have the directory or the folder that contains all your project files and that's, that's the set of files that you're going to be modifying as we described.

[00:03:27] Now, let's suppose you've made a bunch of changes to your, your, your code or your project again, you've added files, renamed files, changed files, whatever.

[00:03:37] And at some point, you want to take a group of changes that you've made and you want to commit those to, to get.

[00:03:43] Now, it could be that you've made some changes to your project files that you do not want to commit to get. Maybe they're temporary uh for some reason.

[00:03:52] And so the idea in get is that before you commit your changes, the first thing you have to do is you have to tell, get which changes or which files that have been changed, do you want to commit to? Get? And so, and there's this thing and get called the index or the stage.

[00:04:11] And so what you do is before you commit your changes and get is, is you run, get a, now the G A command, what it does is it stages a file to be committed in the near future.

[00:04:25] And so all the files that you've changed that you want to commit, you would just do get a, on all those files and each one would then be staged and now it's ready to commit.

[00:04:35] Really all you're doing there is you're, you're telling it explicitly which files do you want to include in the commit that you're about to do again, there could be changes you've made to files that you don't want to include.

[00:04:46] So that's the process of staging is you're just telling it which files do I want to include in the commit and so on. And once you've added all the files to the stage or the index, then you're ready to commit.

[00:04:58] And so that's when you would run the git commit command and that would make all those changes that you staged permanent in the git repository.

[00:05:09] Now that you've committed those changes to the repository, they're always accessible.

Start visual description. The professor demonstrates how to restore deleted files using the git checkout command, which retrieves the files from the repository's history. End visual description.

[00:05:14] So even if you went back to your, your director here and deleted all your source code, which isn’t recommended very often.

[00:05:20] But if I went back here and deleted all my files, I could then run a get checkout command and it would restore all the files that I had deleted from the repository.

[00:05:29] Because, because the repository has a, a complete history of all the files.

[00:05:33] So if I just ran a get checkout, that would restore all my files that I had deleted.

[00:05:40] Just make sure you don’t delete the doc git folder.