14.4.1 Git Concepts

Click one of the buttons to take you to that part of the video.

Git Concepts 0:00-0:35

Today, professionals collaborate on larger projects than ever before, projects that involve hundreds of team members and thousands of files. This situation can be a recipe for disaster. We experience this challenge here at TestOut as we create our courses.

One way to collaborate more efficiently is to use a distributed version control system such as the Linux program Git. Version control systems track changes to files and help keep all users' files up to date. When many people work on the same files, this helps avoid problems like saving over one anothers' important work and collisions from two people editing a file at once.

Git Components 0:36-1:56

Git is comprised of four separate storage locations: your workspace,

the local index, the local repository, and the remote repository.

The remote Git repository is a copy of your project's files stored on a remote server. This repository is important for projects that have multiple collaborators. It gives you a central place to store your team members' work. It's also the only part of Git that's outside your computer.

You can create a remote repository on your own Git server, or you can use one of the many Git repositories online. Widely used services include GitHub, GitLab, and Bitbucket.

The local Git repository is a copy of the remote repository that's stored on your computer. This is where you make all your additions, changes, and deletions. When you're ready, you can push all your local changes to the remote repository.

This local Git repository is a huge advantage. You can make changes to your local repository when you don't have access to the internet, and then you can apply those changes to the remote repository later, when you're connected.

The index is, technically, part of the local repository. The index stores the list of files and tracks changes to the files.

And finally, we have the development workspace. This is the folder on your computer where you do your work. This is where you add new files to your project, modify files, and remove files. After you make changes to your workspace, you can tell Git to update all the repositories to reflect your changes.

Clone and Pull 1:57-3:16

To begin working on a project, you will need to create a repository. One way to do this is to use the git init command. The git init command is only used once for the initial setup. You'll typically run it in the folder where your existing project file are located. It will create a new .git sub-directory in your current working directory and a new master branch for your project (we'll discuss branches more in a moment). Git init can also be used to create a new remote repository.

While you can use git init to create a new repository, most projects will already have a remote Git repository that contains the project files. In this case, the command you'll use is git clone. This command will make a copy of the project repository on your computer. Git clone will create the local working directory, add a .git sub-directory to create the local repository, and then put a copy of all the project's files on your computer.

Once you've cloned the repository, it's a good idea to frequently update your local repository from the remote repository. This keeps you working from the latest version of the files in your project and prevents conflicts with your team members.

You can use the git pull command to update the local repository and your development workspace.

You can update just the local repository using the git fetch command.

The git config command is also helpful for setting Git configuration values in configuration files.

Stage Files 3:17-4:11

As you do your work on the files in your workspace, you need to keep Git informed. There are many commands that let you communicate with Git.

git add - tells the index to add a file to the next commit for you. This is sometimes called staging the file.

git mv - moves files just like the mv command, but the git mv command also notifies the index of the change.

git cp - copies files or directories just like the cp command. Like git mv, it also notifies the index of the change.

git rm - creates a file or directory delete request in the index. The requested file or directory will be removed from your filesystem and the local Git repository at the next commit.

The git status command allows you to see which files have changed in your working directory that haven't been added to the local index.

If needed, your Git repository can be configured to ignore certain files or directories. This will prevent Git from tracking or saving changes to them. These ignored files are specified in a special file named .gitignore that's located at the root of your repository.

Commit and Push 4:12-5:07

After you edit your project files, you'll want to save your changes to the repository. This is called committing your changes.

To commit, you'll use the git commit command. When you do, changes that you've indexed will be saved to the local Git repository. This includes any files that you've requested be copied, removed, moved, or added.

The next step is to send your commits to the remote git repository using git push. This allows the rest of you team to have access to your work.

However, before you do, you should use the git pull command to pull, or download, any changes from the remote repository. This will try to get the latest changes and put them in your local Git repository.

If there are no conflicts with the changes you've been making on your local repository, then you can push your changes out to the remote server.

If there are conflicts, then you have the option to merge the changes from the remote repository.

The git log command displays a list of all of your commits so you can review the project history and search for specific changes.

Revert to a Previous Commit 5:08-6:10

One of the great benefits of using a version control system is that you can go back to previous versions of your work. This is possible because each commit is tracked in your local Git repository and the remote Git repository. This gives you access to any version of your work along the way. If a mistake is made or the project is messed up so badly it's just easier to revert to a previous version, you can reset or roll back to any of your previous commits.

Another great benefit of Git is branching. When you want to add a new feature or experiment with something new in your project, you can use git branch to create a new branch for that change. Working in the separate branch makes it harder for you to break the main project with an unstable change. You can later discard the branch, or you can use git merge to merge the changes you made in that branch back into the main project.

You can work from multiple remote Git repositories for different projects or for different pieces of a project.

So, you can see how Git allows true multi-user, multi-project collaboration by allowing each user's workstation its own Git repository that works in conjunction with the remote repository.

Summary 6:11-6:28

That's it for this lesson.

In this lesson, we introduced the distributed version control system called Git.

We described how to work within Git by cloning, pulling, staging, committing, and pushing files.

We also discussed the great advantages that Git provides by allowing you to roll back to a previous version, create branches for experimental work, and collaborate with others on multiple projects.
