# **Git and Github**

## What is Git and Github?

Git is a version control system designed to help users keep track of changes to files within their projects. Git was designed to fix the problem that it's creator, Linus Torvalds was having with managing the huge challenge of keeping track of all changes to the kernel, the operating system for Linux.

The benefits that Git offers over similar systems include, better speed and performance, reliability, free and open-source access, and an accessible syntax.

It's also important to note that Git is used predominantly via the command line. Developers tend to find Git syntax and commands easy to learn.

The other service commonly used by web developers is GitHub. GitHub is a Cloud-based hosting service that lets you manage Git repositories from a user interface.

A Git repository is used to track all changes to files in a specific folder, and keep a history of all those changes.

It incorporates Git version control features and extends these by providing its own features on top. Some of the most common of these features include access control, pull requests, and automation.

Diagram

Description automatically generated with medium confidence

*Can you keep track of changes to files within projects using Git?*

* *Yes*

## Creating and cloning a repository

-n/a

*GitHub provides you with many options for cloning a repository. With which of the following options do you need to set up SSH keys and assign them to the user accounts?*

* *SSH*

## How Git works

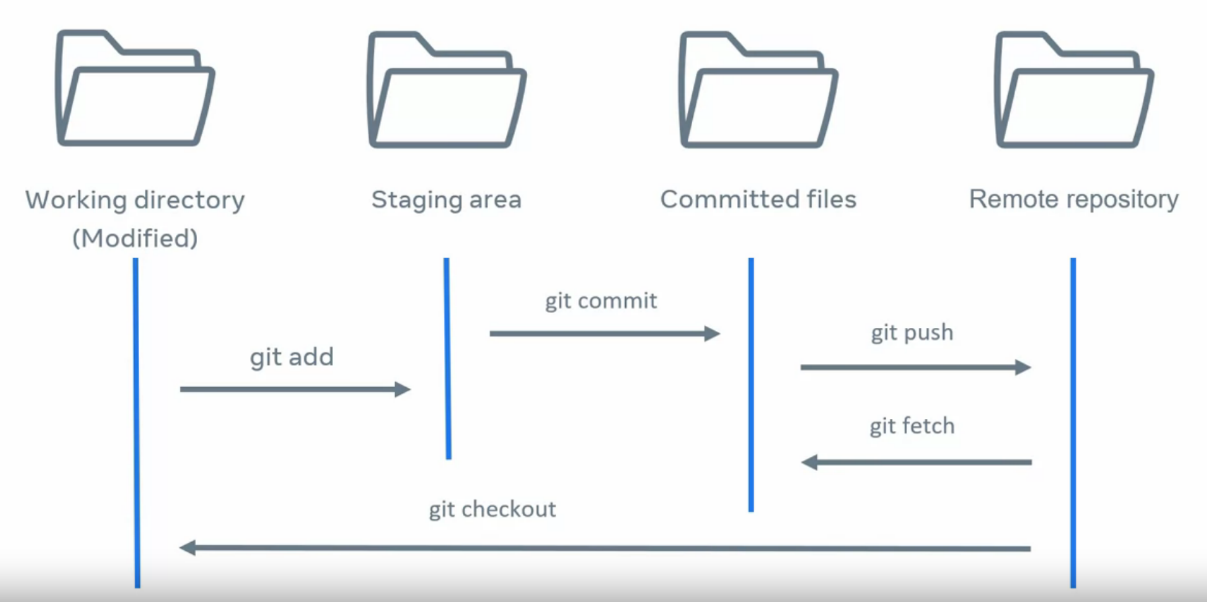
Git uses workflows which can be broken into three states namely, modified, staged, and committed.

Let's start with the first state, adding removing and updating any file inside the repository is considered a modified state. Git knows the file has changed, but does not track it. This is where the staging state comes in. Let's turn to it now. In order for Git to track a file, it needs to be put in the staged area. Once added, any modifications are tracked.

Then, the last state is the committed state. Committing a file in Git is like a save point in many ways. Git will save the file, and have a snapshot of the current changes.

A picture containing icon

Description automatically generated



*You are working on a project and have to share some code with a colleague. What is the correct order the code will flow in Git workflow?*

* *Working directory, staging area, committed files, remote repository*

## Add and commit

The purpose of the git add command is that I'm essentially prompting git and letting it know that I want to track this file, and that it will be included as part of my commit.

then at the end I will use the git commit command. The staged area is really important because you're essentially preparing to get all of the files and changes that you want as part of whatever feature you're working on. Basically, you are getting all of that content ready for commit.

*You've learned that Git works on the principles of the edit, stage, and commit pattern. But when you edit a file in your Git working directory it will not be tracked until it's added. You can do this by using the Git commit command.*

* *False*

## Branches

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.

*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.*

* *True*

## Remote vs. local

-n/a

*You have created a new local repository called “test-repo”. Which command would you use to set the remote repository in git?*

* *Git remote*

## Push and pull

-n/a

*When you perform a Git Push command, it copies content from your local repository and then uses it to merge all of the content in the remote repository.*

* *False*

## HEAD

-n/a

*Whenever a change occurs for a commit, the single hashed ID will update to be the latest commit for that working directory.*

* *True*

## Diff commands

-n/a

*You have returned to development of your hotel booking app after some time away, and want to see the last changes you made. At which levels can you compare the differences between versions using Git diff?*

* *Individual files*
* *Commits*
* *Branches*

## Blame

One day you might be overseeing a big team of developers. Can you imagine how complex it gets to keep track of everyone to changes and updates to files? Fortunately, git has a very helpful command for keeping track of who did what and when. It's called git blame.

One of the core functions of git is its ability to track and record the full history of changes for every file in the repository. In order to view and verify those changes, git provides a set of tools to allow users to step through the history and view the edits made to each file. The git blame command is used to look at changes of a specific file and show the dates, times, and users who made the changes.

*You are a team leader at a software development company. You run the git blame command to see the changes on all the files for the week. What is the order in which the change information will display in each line?*

* *<ID><Author><Date><Time><Line number><Content>*