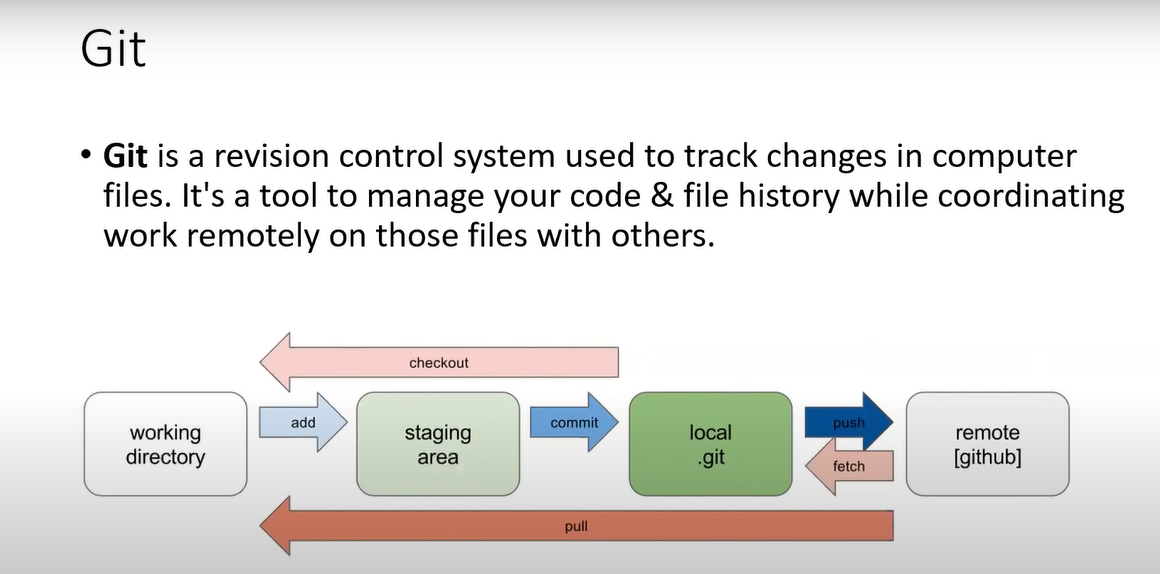
**GIT is a Distributed Version Control System(DVCS).** The previous line will not make any sense to you if you have not gone through my previous post on Version Control System.

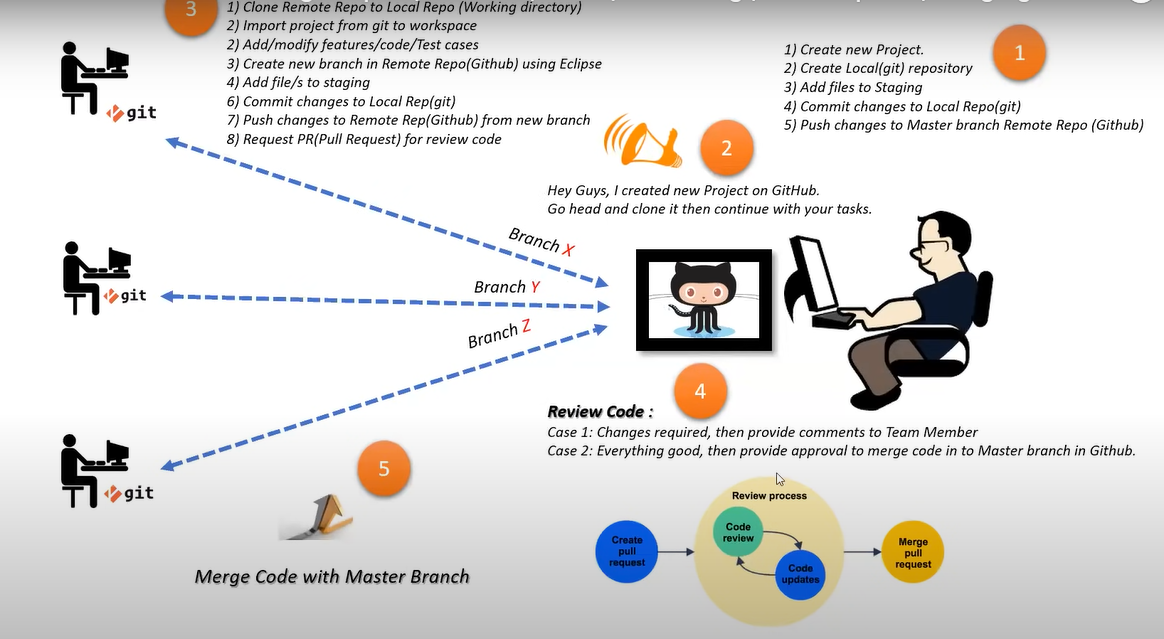
A Version Control System is a utility or software which helps us to track and manage changes to a file system or code repo for better collaboration among developers to develop an application. It can keep the history of changes into a file including who has changed and what was changed. In Distributed Version Control System, files are mirrored with full version history to other systems. Git is a distributed version control system in which repositories are cloned to the developer’s system.

GIT is free and open-source software. It is designed to handle everything from small to very large projects with speed and efficiency.



A picture containing diagram

Description automatically generated



**Distributed**

GIT is a distributed system i.e. you can always clone the entire repository with the entire history in the user’s system. It allows multiple backups of the repository and supports different workflows.

**Staging Area**

You must have heard about a staging environment where application codes are reviewed and tested before releasing or deploying to production**. Git has a “staging area” or “index” which is an intermediate area where commits can be formatted and reviewed before completing the commit.** GIT allows you to quickly stage some of your files and commit them without committing all of the other modified files in your working directory.

**What is git init?**

The “git init” command creates a new empty git repository or initializes an existing directory as a git directory. Once the directory is a git directory you can run any git commands.

The “git init” command creates a new hidden folder named “.git” in the same directory where it was run. The “.git” folder will save metadata of the new git repository. Once it becomes a git directory the default branch is master. Remember during the installation of git we had an option to change this default branch name.

<https://stackoverflow.com/questions/4308610/how-to-ignore-certain-files-in-git>

**What is the “git add” command?**

A working directory is a local copy of your repo where you can add a new file or make changes to any existing files to the repo.

Before we commit changes, **we must need to add them from the working directory to the Staging area.**

git add is the first command in a chain of operations that directs Git to “save” a snapshot of the current project state, into the commit history.

The Staging area is a rough draft space where we can use the “git add” command to add a newer version of file/s. It included addition/deletion/modification of files. Changes added in the staging area will be committed and can be pushed to the repository.

We can always remove or modify already added files in the staging area.

**When we ran the “git add” command then changes from the working directory were copied (not moved) to the staging area.**

Remember if we have linked local repo to remote repo then this new file will not be present in remote repo till, we commit and push.

**So, in the short staging area is in your local only.**

It is something like a staging area as a level up of working directory where we have some confirmed changes to be committed.

If we would like to see the difference between these two i.e. versions of “myFile1.txt” in staging area and working directory then we can use another git command **“git diff“.**

**git diff To see the difference**

**Adding multiple files to staging area**

As of now we were adding single file to staging area at a time. We can add multiple files together as well. We can pass multiple file names seperate by white space as shown below.

If we have multiple files then typing name of all files with paths is not a feasible idea.

We can use “git add .” and “git add -A” commands.

We can use “git add .” and “git add -A” commands.

Difference between these two commands is that “git add .” will not add files from higher directories i.e. it will add files from currently selected directory only.

**Unstage a file**

We can always unstage file if required. We need to use “git rm –cached <file>” to unstage a file/s.

The “**cached**” option specifies that the **removal should happen only on the staging index**. Working directory files will be left alone.

git rm –cached <file>

**What is the “git commit” command?**

We have added changes from the working directory to the staging area using “git add” command. Once we reach that step or we have staged changes (after using git add command) then we need to use git commit command.

The git commit command captures a snapshot of the project’s currently staged changes and creates a new commit.

Please note this snapshot is committed to the local repository. I mean if we have a central repository like BitBucket etc, assosiated with the project then the commit will not push changes to the central repository automatically.

**git commit -m <commit msg>**

As we already know that git commit command creates a snapshot of the project’s currently staged changes and we can have multiple commits so we should provide a commit message which can help us to identify them. It is a kind of label that we give to each commit.

**Get list of files in the commit stage**

Git show

To get list of all files which are in commit stage

**Git Push – Upload Committed Local Repository Changes To Remote Repository Branch**

The Git Push command is used to push committed local repository changes to the remote repository’s branch.

git remote add origin https://github.com/SudheerKasarla/RestAssured.git

git push -u origin master

**Add remote URL to local GIT repo**

We have received a remote URL for a remote repository. We need to link to add this to our local repository so that we can push changes to remote.

git remote add origin <https://github.com/SudheerKasarla/RestAssured.git>

git push -u origin master

**The GIT Remote Command**

If you want to know if any remote URL is linked to your local repository then you can run the below commands. The command “git remote” lists the remote

git remote

git remote -v

git remote -v

git remote -v

connections while “git remote -v” lists all the remote connections with URLs.

**The GIT Remote Add Command**

To create a new connection with a remote repository we need to use the below command-

git remote add <a name given to remote url> <remote repo url>

“name” is a name given to a remote URL that you can use as a shortcut for other git commands. You can give any name to it. **Here I will use “origin” which is generally used by GIT users.**

**Push the local changes to a remote repo**

git push <remote url or name> <branch name>

git push origin master

**Git Branch**

You can run “git branch” command to see all branches.

**What is GIT CLONE command?**

git clone <url of remote repo>

git log

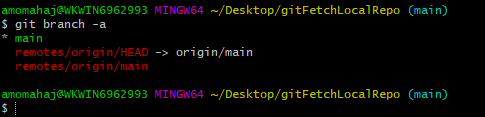
Note: “git log” is the command to display to see all existing commands.

# **Git Fetch – Download (Not Merging) Changes From Remote Repository**

**The “git fetch” command downloads commits**, files, and refs from a remote repository into your local repo but not in your currently checked-out branch. It helps to isolate changes fetched from the remote repository and local changes. In simple words, no merging of code will happen.

**Run “git log” command to view existing commits.**

**Run “git branch -a” command to see all branches after cloning.**



In short **“main”** is a currently checked-out local branch.

In GitHub default branch name is **“main”** which is like “master” in other Git hosting service providers.

The branch **“remotes/origin/main”** is a local copy of the main branch of the remote repository.

The branch **“remotes/origin/HEAD”** represents the default branch on the remote which is “main”.

git fetch origin main

“git fetch origin main” command.

This command says that fetch the content from the “main” branch in the “origin” repo.

**“remotes/origin/main”** branch keeps a copy of the “main branch” of the remote repository. When we do “git fetch” command then this branch is updated in local repository.

**“git checkout“**

let’s run another git command “git checkout“. We will navigate to the branch where changes were fetched to verify if changes were fetched.

Now if we want to merge or bring remote changes to our local branch then we need to use **“git merge”** command. The command **“git merge origin/main”** says that merge changes from **“origin/main”** branch to my currently checked out branch which is “main“.

**To Delete Branch**

git branch -d Sprint1

git branch -D Sprint1