Class 3 – **Git**

What is Git?

It is a version control system (VCS).

As dev, you develop a mobile app. Once the app becomes popular, so when you want to add a new feature or filter to it. But people did not like it or a you found a bug in it. So how to get back to the original version?

VCS comes into play to track changes in code:

1. It can be used to go back to a pre-version of the code.
2. It also helps when multiple developers are working on the same code.

In git you can save your steps as commit. You can move forward / backward. Using git, we can revert back to the previous step/commit. We can go back to the previous commit and make the required changes in the program.

Git Commands:

git init → initialized empty git repository.

→ any directory, which has git becomes a repository.

Since we have initialized a git repository now, git will start keeping a track of this folder.

ls -a <enter> shows us a hidden file named as '***.git***' in the folder.

In git we have 2 areas:

- Staged Area – files will be green (git status)
- Un-staged or Untracked Area – files will be red

Whenever you make any changes in the git folder, git will add files which are changed to the un-stagged area and for you to commit those changes you would have to move them to stagging area.

git status → which helps us to see which files are stagged and which files are un-stagged / un-tracked.

Un-staging / untracked: git will put any change you make in the git directory / repository in untracked / upstaged folder.

Staged area: it's your duty to put the files in un-stagged area to stagged area.

git add → moves the file from un-stagged area to stagged area.

To add more than one file use the command "***git add .***" This will add all files in the specified folder. Anything in staging area will be saved when we commit. So, to commit the command is,

git commit -m "<message>" → this is a commit.

NOTE: To configure git please follow this command:

git config --global user.name "<name>"
git config --global user.email "<email>"

git log → displays all the commits you have done

commit daf4e88d73b9b0503397a0164302e345707b8905 (HEAD -> master)
Author: first_name last_name <registered.email.com>
Date: Thu Nov 12 22:18:27 2020 +0530

  added remaining files → This is the last commit

commit 5a90f48355b88a371120dfe89cff1cec6913d92e
Author: first-name last_name <registered.email.com>
Date: Thu Nov 12 22:16:08 2020 +0530

  adding ddlj.txt → This is the Second commit

commit e0e55e163d9d6d633f66a0fb80530c90160537ee
Author: first-name last_name <registered.email.com>
Date: Thu Nov 12 21:55:18 2020 +0530

  adding a file → This is the first commit

Overview:

1. Make any directory (mkdir)
2. Go inside it
3. Git init → this command initiates a git repository
4. Echo hello > hello.txt
5. 2 areas in git → (staged area/tracked area & un-staged/untracked area)
6. Whatever changes you do they come in untracked/un-staged area
7. Whatever you want to commit you have to put them in staging/tracked area
8. Once they are in staging/tracked area you run the command ***git commit -m "message"***
9. Then do git log and it will show the commit and the message that we used to perform the commit.

git diff → displays changes you have done to the file.

SHA → a unique number.

- SHA of 1st Commit → e0e55e163d9d6d633f66a0fb80530c90160537ee
- SHA of 2nd Commit → 5a90f48355b88a371120dfe89cff1cec6913d92e
- SHA of 3rd Commit →daf4e88d73b9b0503397a0164302e345707b8905 (HEAD -> master)

If we observe the last commit is the head or the master commit which means the latest version. This unique number helps us to get back to a particular commit. The command used here is

git checkout *<SHA of the commit>*

- It will take you to the position of the given SHA
- When you use ls, it will only show the files that were present at that point in your commit. All the new files that were added after the

commit, or all the changes that were done to the files will not be shown.