**Git**

Git is a version control system that enables us to record changes to files over time.

**Staging Area**

It is the area which contains all the files that are going to be a part of the next commit.

**git init**

The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository. Executing git init creates a .git subdirectory in the current working directory, which contains all of the necessary Git metadata for the new repository. This metadata includes subdirectories for objects, refs, and template files. A HEAD file is also created which points to the currently checked out commit.

**git clone**

git clone is dependent on git init. git clone is used to create a copy of an existing repository. Internally, git clone first calls git init to create a new repository. It then copies the data from the existing repository, and checks out a new set of working files.

**git add fileName**

To add the specified file name into staging area

**git add .**

To add all files to the staging area

**git reset HEAD fileName**

It is used to remove a file from the staging area

**git log**

It is used to show the commit logs

**git status**

It is used to show the working tree status

**git diff**

It is used to track the difference between the changes made on a file by comparing that file in working directory and repository

**git diff --staged**

It is used to compare files in the staging area against the repository.

**git rm**

Used to delete a file by typing git rm fileName.

**git checkout -- fileName**

It is used to download (or) clone a specific file from the repository into the our working folder.

**git checkout hashNumber -- fileName**

It is used to get a specific version of the fileName from the repository to the working directory.

hashNumber can be obtained by typing the git log. It is around 40 digits you need to just type 7 digits at the place of the hashNumber.

**git push**

It is used if we want to add files (or) modified files into a repository by typing

git push -u repository\_url.

Instead of always using the repository url we can give some nick name to it so that we can use this name instead of url. This is done by typing

git remote add nickName repository\_url

Now we can push into repository even by typing

git push -u nickName

**Branch**

Branch is a copy of our main repository (or) master repository

**Pull**

After performing the required activities in the branch and if we want to add (or) merge these changes with master branch then we will use pull request

**Workflow**

Working Directory / staging area / repository

Initially all the files are stored in our current working directory. To send these files into our github repository we need to first add them to the staging area by typing git add filename then we need to commit by typing git commit -m “commit message” (message is optional). So now the files which are added and committed will be available in our repository.

If you want to send all the modified files that are in the current working directory to the repository then you can directly send (without adding first them into a staggered area and then commit) to the repository by typing git commit -am “commit message” (message is optional). Here -a allows us to skip the staging area and the commiting directly from our working copy to the repository.