GIT:

Git is a distributed version control system (DVCS) commonly used for open source and commercial software development. A version control system tracks the changes done by an individual or a team. In version control system such as GIT teams can collaborate on projects from different timelines.

Distributed version control system (DVCS):

Instead of working on a single repository every team member has a flexibility of creating their own Repo distributed and maintained in a branch and later merging these changes into the master. Distributed version control system allow full access to every file, branch, and iteration of a project, and allows every user access to a full and self-contained history of all changes.

REPOSITORY:

A repository is simply a [project](https://git-scm.com/) which consists of the entire collection of files and folders associated with a project, along with each file’s revision or changes history. The file history appears as snapshots in time called commits, and the commits exist as a linked-list relationship, and can be organized into multiple lines of development called branches. Working in repositories keeps development projects organized and protected. Developers are encouraged to fix bugs, or create fresh features, without fear of derailing mainline development efforts.

GIT COMMANDS:

**1. Initialization of GIT:**

$ git init

This is the first command used to initialize a new directory or repo and start tracking the changes

**2. Checking the status:**

$ git status

It shows the status of changes as untracked, modified, or staged.

**3. Adding and removing the changes:**

$ git add <file>

It stages the changes that is includes these changes into history. Staging and committing separately gives developers complete control over the history of their project without changing how they code and work

$ git rm <file>

It removes the staged changes or the files from the history.

**4. Commiting the changes:**

$ git commit –m “message”

Git commit saves the changes to the repository. –m indicates the message that is to be shown about the commit. If –m is not mentioned then text editor is opened where the message is to be written and saved.

**5. Branching:**

$ git branch <name>

Branch is an independent line of development. By default branch is master. A new branch can be created using command

$ git branch

It shows the branches being worked on locally.

$ git merge <branch name>

It merges the branch to the master branch. But before merging you need to checkout the branch that is to be merged.

$ git checkout <branch name>

It moves the head from the master to the mentioned branch or vice-versa as mentioned in the branch name for changes.

$ git branch –m <oldName> <newName>

It renames a branch with a new name.

$ git branch –d <branchName>

It deletes a branch. We can’t delete a branch which we are currently on.

$ git rebase <branch>

Rebase commits on top of another base tip that is the mentioned branch.

$ git cherry-pick <commits>..

It picks the mentioned commits in the command and forms a new set of them in a branch.

**6. GIT push and pull:**

$ git push

It updates the remote repository with any commits made locally to a branch. There are many more options to push command.

$ git fetch

It fetches the changes from the remote repository. To update the repository we need to use “$ git merge” after “$ git fetch”.

$ git pull

It updates the local line of development with updates from its remote counterpart. Developers use this command if a teammate has made commits to a branch on a remote, and they would like to reflect those changes in their local environment. It is simply a combination of “$ git fetch” and “$ git merge”.

**7. Cloning:**

$ git clone <hyperlink>

It clones a repository from link in to a new directory.

**8. Reset:**

$ git reset

It resets the current Head to the specified state.

$ git revert <commit>

It reverts an existing commit that is mentioned.