[**GIT**](https://git-scm.com/docs)

* A free **version control system**
* Manage source code changes
* Save code snapshots (‘**commit**’)
* Work with alternative code versions (‘**branches**’)
* Move between branches and commit (‘**checkout**’)
* With Git, you can easily roll back to older code snapshots or develop new features without breaking production code.
* **What is** [**Version Control System**](https://www.geeksforgeeks.org/version-control-systems/#:~:text=Course-,Version%20Control%20Systems,-Last%20Updated%20%3A)**?**
  + A version control system (VCS) is a software tool that helps developers manage changes to their codebase.
  + It allows developers to keep track of different versions of code and work on the same files simultaneously. This can help promote cooperation and collaboration between developers.
  + Basically, VCS keeps track on changes made on a particular software and takes a snapshot of every modification. Let’s suppose if a team of developer add some new functionalities in an application and the updated version is not working properly so as the version control system keeps track of our work so with the help of version control system, we can omit the new changes and continue with the previous version.
* **Github:**

**A screenshot of a computer

Description automatically generated**

* **Git Repositories:**
* Git features can be used in projects with **Git repositories.**
* A **repository** is as folder used by git to track all changes a given project
* Git commands require a repository in a project
* **Github Repositories**
  + Cloud storage for your Git repositories for Backup, cross-device usage, collaboration.
  + We can make repositories **private or public.**

**A screenshot of a computer

Description automatically generated**

* **Git commands:**

1. **git init:**
   * Create an empty repository or reinitialize an existing one
2. **git clone:** 
   * Clone a repository into a new directory.
3. **git add <filename>:**
   * Moves changes from the working directory to the staging area.
   * This gives you the opportunity to prepare a snapshot before committing it to the official history.
   * **‘.’** Is used to add all changed file in staging area
4. **git status:**
   * git status is used for checking status of **stagging files**
5. **git commit:**
   * Record changes to the repository.
   * Takes the staged snapshot and commits it to the project history. Combined with git add, this defines the basic workflow for all Git users.
6. **git log;**
   * show previous commit logs
7. **git checkout <id>:**
   * this command is used to switch between different commits
8. **git revert <id>:**
   * this command is used to undo given commit based on id.
   * Revert changes of commit by creating a new commit
9. **git reset --hard <id>:**
   * Undo changes by deleting all commits since <id>
   * Use carefully this command because once any commits is deleted you couldn’t get back that commit
10. **git branch <branchName>:**
    * This command is used to create a new branch.
    * **git branch** -> is used to list all branches which we created.
    * To switch branch use **git checkout <branchName>**
    * **git checkout -b <branchName>** this is creating new branch and checkout that branch
    * **git branch -D <branchName>:** used to delete branch

**A screenshot of a computer program

Description automatically generated**

1. **git merge:**
   * A powerful way to integrate from divergent branches. After forking the project history with git branch, git merge lets you put it back together again.

**A screenshot of a computer code

Description automatically generated**

1. **git remote:**
   * Manage set of tracked repositories.
   * A convenient tool for administering remote connections. Instead of passing full URL to the fetch, pull, and push commands, it lets you use a more meaningful shortcut.
2. **git push:**
   * Pushing is the opposite of fetching (with a few caveats). It lets you move a local branch to another repository, which serves as a convenient way to publish contributions. This is like svn commit, but it sends a series of commits instead of a single changeset

**git push <originName> <branchName>**

1. **git clone <repoLink>:**
   * Clone a repository into a new directory