**GIT COMMANDS**

* **Git Config:** This command allows you to specify the username and email address that will be used with your commits.

$ git config --global user.name <user name>

$ git config --global user. email <user email>

* **Git init**: A git repository must first be created before you can make commits or do anything else with it.

$ git init <name>

* **Git Status**: It will display the status of the file, where are the files located.

$ git status

* **Git Log**: To check what is done.

$ git log

* **Git Clone**: We can add the original location as a remote so you can easily fetch from it again and push it if you have permissions.

$ git clone <URL>

* **Git Add**: This command adds a single file or more than one file to the staging area.

$ git add <file name> - Specific file

$ git add \* - to all files

* **Git Commit**: This command records or snapshots the file permanently in the version history.

$ git commit -m “commit message”

* **Git Diff**: This command shows the file differences which are not yet staged.

$ git diff

* **Git Diff**: This command shows the file differences which are not yet staged.

$ git diff

* **Git Branch**: Below are some important git branch commands.

**$ git branch :** To list branches present in local repository.

**$ git branch --list :** To list all the branches

**$ git branch <branch name> :** To create a new branch.

**$ git checkout <branch name> :** To switch particular branch.

**$git checkout –b <branch name> :**  Used create and switch branches.

**$ git branch –d <branch name> :**  To delete the specific branch

**$ git branch –m <old name> <new name> :** To rename the branches.

**$ git merge <branch name> :** To merge the branches.

* **Git Remote:** This command is used to connect your local repository to the remote server.

$ git remote <variable name> <URL>

* **Git Push:** This command is used to push committed changes your local repository to the remote server and vice versa.

$ git push <variable name> <master/branch>

* **Git Pull:** This command is used to update your current HEAD branch with the latest changes from the remote server.

$ git pull <variable name>

* **Git Fetch:** This command is used to download new data from remote repository.

$ git fetch <variable name>

* **Git Delete/remove:** To remove or delete the files.

$ git rm <file name>

* **Git Rename/move:** To rename or moving the files.

$ git mv <file1> <file2>

* **Git Reset:** The term reset stands for undoing changes. The git reset command is used to reset the changes. The git reset command has three core forms of invocation.

$ git reset

$ git reset --soft head ~1

$ git reset --hard head ~1

$ git reset --mixed head ~1

* **Git Revert:** The git revert command is used for undoing changes to a repository's commit history.

$ git revert

* **Git Tag:** Tags make a point as a specific point in Git history. Tags are used to mark a commit stage as relevant. We can tag a commit for future reference. Primarily, it is used to mark a project's initial point like v1.1.

$ git tag

**git tag:**  list all the available tags from the repository.

**git show:** It will display a description of the tags.

**git tag -l ".\* ” :** It displays the available tags using a wild card pattern

* **Git Stash:** Sometimes you want to switch branches, but you are working on an incomplete part of your current project. You don't want to make a commitment of half-done work. Git stashing allows you to do so. The **git stash command** enables you to switch branches without committing to the current branch.
* **Git stash:** The directory gets cleaned and we can switch to other branches and work on them.
* **Git stash save:** This command temporarily stores all the modified tracked files.
* **Git stash list:** This command lists all stashed changesets.
* **Git stash apply:** We can re-apply the changes that you just stashed by using the git stash command.
* **Git stash changes:** It will show the file that is stashed and changes made on them.
* **Git stash pop:** This command restores the most recently stashed files.
* **Git stash drop:** This command discards the most recently stashed changeset.
* **Git stash clear:** It will delete all the stashes that exist in the repository.
* **Git stash branch**: It will create a new branch and transfer the stashed work to that.
* **Git Rebase:** Rebasing is a process to reapply commits on top of another base trip. It is used to apply a sequence of commits from distinct branches into a final commit. It is an alternative of git merge command.

$ git rebase

* **Git Squash:** In Git, the term squash is used to squash the previous commits into one. It is not a command; instead, it is a keyword. The squash is an excellent technique for group-specific changes before forwarding them to others. You can merge several commits into a single commit with the compelling interactive rebase command.

$ git squash