Login to git through commands:
git config --global user.name -> git config --global user.name “username”
git config --global user.email -> git config --global user.email “your@gmail.com"
git config --global user.password -> git config --global user.password your_password

Commands to initiate git repository
git init
git add README.md
git add .
git commit -m "first commit"
git remote add origin git@github.com:GitUserName/<reponame>.git
git push -u origin main
git remote remove origin (Optional) -> If you want to remove the origin branch

git status
    it shows git current branch, and uncommitted changes

git branch branch_name
    it creates new branch

git checkout -b branch_name
    it creates a new branch and checkouts into it

git checkout branch_name
    it checkouts into new branch when the current branch has no uncommitted changes

git add . OR git add file_name
    it adds new changes/files to the current branch to make commit

git commit -m "Small note on changes comments"
    it commits changes locally

git push --set-upstream origin branch_name
    This will push the local branch to the remote repository and configure the upstream branch for tracking in the remote repository.

git push
    This will push the locally committed changes to the GitHub remote server only if this branch has upstream remote branch

git pull

First, it fetches changes from the remote repository, and then it merges those changes into your current local branch.

git pull origin

Same as git pull, but it specifies the remote repository to pull from. If you have multiple remote repositories, you can use this command to specify which one you want to pull.

git pull --rebase

It also first fetches changes from the remote repository, and then it rebases your local commits onto the remote branch. This is a good choice if you want to avoid merge commits and keep a linear history.

git merge branch_name

It merges the branch we mention to the current branch locally.

git merge --abort

It undo the merge changes and make it back to the recent commit the one before merge.

git branch --merged

This will gives you the merge information like which two branches got merged

git branch -d branch_name

This will delete a branch, only the branch if already merged.

git branch -D branch_name

This will delete a branch, even if the branch is not merged.

git diff main..local_branch

This will give you the difference w/b any two branches for eg:main and local_branch

git stash

This will save the uncommitted code to the buffer and undo the changes to make it ready to pull new changes. This one is mainly useful when you are working on some branch and you need some updated changes from the remote branch. You can stash your changes to the buffer and then pull remote changes then un-stash your changes back to the current updated branch.

git stash list

This will give you the list of all stashes

git stash list -p

This will lists out all stashed changes

git stash apply

This will apply a most recent stash to the current branch without deleting the stash

git stash pop

This will apply a most recent stash, and remove it from saved stash list

git stash apply (stash reference number)

This will apply a specific stash point to the current branch

git stash save "(description)"

This will create a stash point, be more descriptive for future use

git stash drop stash@{n}

This will delete the nth stash point

git stash show

This will show you the last stashed files affected in the recent stash

git stash show -p

This will displays the detailed changes, showing the exact lines that were added or removed.

git stash show -p stash@{n}

This will show you nth stashed files and content difference

alias graph=”git log --all --decorate --online --graph”

This will give you the branch graph

git reset --hard origin/main

This will reset your local main branch to match the state of origin/main (the remote main branch). Overwrite all local changes, including staged and unstaged files.

git reset --hard HEAD^1

Move the current branch's HEAD to the previous commit (HEAD^1 refers to one commit before the current HEAD). Discard all changes in the working directory and staging area, reverting the state of your project to that previous commit.
this command completely resets your local code to the state of the previous commit, erasing any uncommitted changes.

git merge-base branch1 branch2

This will find and give you a reference id (something like this f77c1384c3a966323ba6d642f8950a466e215acd) to create new branch from the common point where 2 branches have the same code changes. This will avoid unnecessary merge conflicts then use the below command to create new local branch with with 2 branches common commit

git checkout -b branch_name f77c1384c3a966323ba6d642f8950a466e215acd

git tag tag_name

This will create a new tag point mainly used to test different versions of code

git push origin –tags

This will push all the tags to the remote repository

git push origin tag_name

This will push particular tag to the remote repository

git branch -m new_branch

This will change current branch name

git branch -m old_branch new_branch

This will change any branch not related to current branch

git push origin :branch_name

This will delete the remote branch

What if 2 or more persons working on the same file changed the same code. We need to resolve the conflicts, then commit and then push to git.