Configuring The Editor -

git config --system core.editor <editor>

Git commit Workflow-

Working Directory
git add . or file name
git commit -m "" message"
git commit -am " message "
git commit -am " message"
git commit - when git editor is configured
git commit --amend

Status -

git status

Log -

git log
git log --oneline
git log --abbrev-commit
git log --graph --decorative --oneline

Working With Branches -

git branch
git branch <branch-name>
git checkout <branch-name>
git switch <branch-name>
git switch -c <branch-name>
git switch -
git checkout -b <branch-name>
git branch -d <branch - name>
git branch --delete <branch-name>
git branch -D <branch-name>
git branch -m <old-name> <new-name>
git branch -m <new-name>
git branch -v
git branch -vv
git branch -r

Branch merging with no conflict and with merge conflicts -

git merge <branch-name>
git merge <branch to be merged> <branch in which is gonna merge>
git merge --continue
git merge --abort
git merge --quit

Comparing Changes -
git diff
git diff commit hash
git diff <file-name>

git diff HEAD <file name>
git diff --staged or --cached <file-name>
git diff <commit-hash> <commit-hash 2 >
git diff <commit-hash>..<commit-hash 2>
git diff branch1..branch2 <file-name> <file-name>
git diff <commit-hash>..<branch-name> <file> <file>
git show <commit-hash> - it show the changes in the commit hash.....

Git Stashing -
git stash save
git stash <file-name>
git stash pop <stash-id> or all changes
git stash apply <stash id> or all changes
git stash drop <stash-id>
git stash clear
git stash list

Time Travelling and un-doing changes with checkout -

git checkout <commit-hash>
git checkout HEAD OR RELATIVE HEAD <file-name>
git checkout -- <file-name> <file-name>

Un-modifying with Git Restore -

git restore <file-name>
git restore --source HEAD-1 OR relative head <file-name> <file-name>
git restore --staged or --cached

Undoing Commits with git reset -
git reset <commit-hash>
git reset --hard <commit-hash>

Reverting -
git revert <commit-hash>
git revert <commit-hash> --no-commit

Rebasing-
git rebase <branch-name>
git rebase -i HEAD or relative head
-reword
-squash
-fixup
-edit
-drop
-pick
git merge --squash HEAD or relative head

git tags -

git tag
git show <tag-name>
git tag <tag-name>
git tag -a <tag-name>
git tag -d <tag-name>

git tag -l '*beta*'>
git tag <tag-name> <commit-hash>
git tag <tag-name> <commit-hash> -f
git checkout <tag-name>

Custom Aliases -

git config --global.alias 'command name'
or
by git config file

cm = commit -m

Git reflogs -

git reflog show HEAD OR RELATIVE HEAD
-expire
-delete
-exists
-show
git reset --hard master@1

Time based qualifiers-
3.minutes.ago
yesterday
1.day.ago
Fri,12feb 2021 14:06:21 -0800

Configure SSH-
Git remotes -

git remote
git remote add origin(any remote name) link
git remote rename <old name > <new name>
git remote remove <remote name>
git remote -v

Fetching and Pulling -

git fetch
git fetch <origin>
git fetch <remote name> <branch name>
git pull
git pull <remote name >
git pull <remote name > <branch name>
  GIT PULL = GIT FETCH + GIT PULL

git push -

git push <remote name> <branch name>:remote branch
git push -u or --set-upstream <remote name> <branch name>:remote branch
git push
force push - git push -f - it allows you to push when you have file present in the remote repo but you don't have that f
ile in the local repo.

Remote Tracking Branches -

git checkout --track remote/branch-name
git reset --hard remote/branch-name
git switch <branch-name>
FOR THE DEATTACHED HEAD - git checkout remote/branch

Cherry-Picking -

git cherry-pick <commit-hash>
git switch <branch-name> then apply on which branch you want to ..
-edit
-no-commit

Search Changes -

git log -S 'Text Tanish op' - It will return us the commits from where it originated and got modifed

Prune Branches - The Branches which got deleted remotely but exists in your local machine and you want to delete t hem as well but you dont know the names of the branches -

git remote update --prune
git branch -vv
git branch -vv|awk '/: gone] | {print $1}'
git branch -vv|awk '/:gone] | {pring $1}'|
Git Bisect - Binary Search Helps us in debugging

git bisect start
git bisect bad
git bisect good
git bisect good
git bisect bad
git bisect reset
git bisect run

now revert to the commit which you got
git revert <commit-hash>

git clean-

git clean -n then use command 'clean' - It list all the files which is gonna remove
git clean -d then use command 'clean' - It list all the directories which is gonna remove
git clean -n -d then use command 'clean' - It list all the files and directories which is gonna remove
git clean -f - No need to use clean it removes all the files inside the directory it does'nt remove the directory
git clean -f -d - Same as above but it removes all files and directories .
R stands for Recursive and F stands for the force

Git collabrative workflow-

Centalized workflow -
feature branch workflow -
fork and clone workflow -
pull requests with no conflicts and conflicts workflow