**Git Commands -  Basic Git Commands**

git hub:

https://github.com/

## git config

**Usage**

$ git config --global user.name "Your name"

$ git config --global user.email "Your email"

$ git config --global –list

## git version

**Usage**

$ git version

## git init

## **Usage**

## $ git init

## Or you can use the repository name with your git init command.

## $ git init <your repository name>

## git clone

## **Usage**

## git clone <your project URL>

## git add

## **Usage**

## $ git add your\_file\_name

## (it will add a single file to your staging area)

## $ git add \* (this option will add all the modified and new files to the staging area)

## git commit

## **Usage**

## $ git commit -m “your useful commit message”

## git status

## **Usage**

## $ git status

## git branch

## **Usage**

## **To list all branches:**

## $ git branch

## **To create a new branch:**

## $ git branch <branch\_name>

## **To delete a branch:**

## $ git branch -d <branch\_name>

## git checkout

## **Usage**

## $ git checkout <branch\_name>

## Also, you can create and checkout to a branch in a single like, here is the usage for that

## $ git checkout -b <your\_new\_branch\_name>

## git remote

## **Usage**

## $ git remote add <shortname> <url>

## **Example**

## $ git remote add origin https://dev.azure.com/aCompiler/\_git/DemoProject

## git push

## **Usage**

## $ git push -u <short\_name> <your\_branch\_name>

## **Example**

## $ git push -u origin feature\_branch You should have origin and upstream set up before you use Git push. And here is the command to set up upstream.

## **Usage**

## $ git push --set-upstream <short\_name> <branch\_name>

## **Example**

## $ git push --set-upstream origin feature\_branch

## git fetch

## **Usage**

## $ git fetch

## git pull

## **Usage**

## $ git pull <remote\_url>

## git stash

## **Usage**

## $ git stash

## And you can view all of your stashes with the following command

## $ git stash list

## And if you need a apply a stash to a branch, simply use apply

## $ git stash apply

## git log

## **Usage**

## $ git log

## By default, it will show you all the commits of the currently checked out branch, but you can force it to see all the commits of all the branches with all options.

## $ git log --all

## git shortlog

## **Usage**

## $ git shortlog

## git show

## **Usage**

## $ git show <your\_commit\_hash>

## git rm

## **Usage**

## $ git rm <your\_file\_name>

## git merge

## Git merge helps you to integrate changes from two branches into a single branch.

## **Usage**

## $ git merge <branch\_name>

## This command will merge the <branch\_name> into your current selected branch.

## git rebase

## **Usage**

## $ git rebase <base>

## git bisect

## **Usage**

## i) To start the git bisect

$ git bisect start

ii) let git bisect know about a good commit

$ git bisect good a123

iii) And let git bisect know about a bad commit

$ git bisect bad z123

With Git bisect you can narrow down the broken code within a few minutes.

## git cherry-pick

**Usage**

$ git cherry-pick <commit-hash>

Git cherry-pick doesn’t modify the history of a repository; instead, it adds to the history.

## git archive

**Usage**

$ git archive --format zip HEAD > archive-HEAD.zip

It will create a zip archive of the current revision.

### 25. git pull --rebase

**Usage**

$ git pull --rebase

It will help you to keep the history clean. Also, you can avoid multiple merges.

### 26. git blame

**Usage**

$ git blame <your\_file\_name>

### 27. git tag

**Usage**

$ git tag -a v1.0.0

### 28. git verify-commit

**Usage**

$ git verify-commit <commit>

### 29. git verify-tag

In the same way, you can confirm a tag.

**Usage**

$ git verify-tag <tag>  
30. git diff

**Usage**

i) to compare the working directory with the local repo:

$ git diff HEAD <filename>

ii) to compare two branches:

$ git diff <source branch> <target branch>

### 31. git citool

Git citool is a graphics alternative of the Git commit.

**Usage**

$ git citool

### 32. git mv

To rename a git file. It will accept two arguments, source and target file name.

**Usage**

$ git mv <old-file-name> <new-file-name>

### 33. git clean

**Usage**

$ git clean

### 34. git help

**Usage**

$ git help <git\_command>

### 35. git whatchanged

**Usage**

$ git whatchanged