

GIT Tutorial

This is enough git command line for a software developer to use 😊

To configure the user name and email:

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

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

Show all Git config properties throughout all of the variously scoped Git file:

```
$ git config --list
```

Show the working tree status:

```
$ git status
```

Create an empty Git repository or reinitialize an existing one:

```
$ git init
```

The new (or untracked), deleted and modified files will be added to your Git staging area:

```
$ git add --a or .
```

Captures a snapshot of the project's currently staged changes:

```
$ git commit
```

Record changes to the repository:

```
$ git commit -m "<initial commit>"
```

To show the working tree status:

```
$ git status
```

Adds a change in the working directory to the staging area

```
$ git add <file name>
```

▪ Cloning a Remote Git Repository from GitHub:-

To removes specific files or a group of files from a Git repository :

```
$ rm -rf .git
```

To clone a repo from the Github into a new directory:

```
$ git clone
```

If you want to change the name of the cloned repo:

```
$ git clone url <new name>
```

To change the directories:

```
$ cd <directory name>
```

List the content:

```
$ ls
```

Present working directory:

```
$ pwd
```

you should change the commit name after any change in the file

shift+insert to paste the copied text on gitbash

▪ .gitignore: Ignoring Files in Git

To generate the new blank file:

```
$ touch <file name>
```

Git will not track files and folders specified in .gitignore. However, the .gitignore file itself is tracked by Git.

- **Git Diff: Showing Changes Between Commits/Staging Area & Working Directory:-**

To compare the working directory with staging area:

```
$ git diff
```

- **Skipping The Staging Area:-**

To skip the staging area of tracked file:

```
$ git commit -a -m "<commit>"
```

- **Moving and Renaming Files In Git:-**

To remove files from the working tree and from the index:

```
$ git rm <file name>
```

To rename a file, a directory, or a symlink:

```
git mv <old name> <new name>
```

To untrack the file from staging area:

```
$ git rm--cached
```

- **Viewing & Changing Commits In Git:-**

To show commit logs:

```
$ git log
```

The git log patch command displays the files that have been modified. It also shows the location of the added, removed, and updated lines:

```
$ git log -p
```

To show the history of n number of commit change:

```
$ git log -p <-n number>
```

The log command displays the files that have been modified. It also shows the number of lines and a summary line of the total records that have been updated:

```
$ git log--stat
```

To show all the history of commit in one line:

```
$ git log --pretty=oneline
```

To show all the history of commit in short:

```
$ git log --pretty=short
```

To show all the history of commit in detail:

```
$ git log --pretty=full
```

Filtering the Commit History:

By Date:

```
$ git log --since=<number of days>
```

```
$ git log --since=<number of months>
```

```
$ git log --since=<number of years>
```

By author name, mail etc:

```
$ git log --pretty=format: "%h -- %an" [h- commit hash an- author name ae- author email]
```

To modify the most recent commit:

```
$ git commit --amend
```

Vim editor fill open up if you are using Git Bash

Use

Esc : to type

i: to edit

wq : to exit

▪ **Unstaging & Unmodifying Files In Git:-**

To unstage any file:

```
$ git restore --staged filename
```

To unstage or even discard uncommitted local changes:

```
$ git restore <file name>
```

To restore/unmodify, match with the last commit:

```
$ git checkout --<file name>
```

To match the working directory from last commit:

```
$ git checkout -f
```

▪ **Working with Remote Repositories:-**

Let you create, view, and delete connections to other repositories:

```
$ git remote
```

To push file to remote:

```
$ git remote add <connection name> <website url>
```

To show connection name with URL:

```
$ git remote -v
```

▪ **Setting Alias In Git:-**

If you don't want to type the entire text of each of the Git commands, you can easily set up an alias for each command:

```
$ git config --global alias.co checkout
```

```
$ git config --global alias.br branch
```

```
$ git config --global alias.ci commit
```

```
$ git config --global alias.st status
```

```
$ git config --global alias.ustaged 'restore --staged --'
```

- **Branching (To avoid changes in master branch , people create branches and work in that branch)**

To create new branch

```
$ git checkout -b <new-branch-name>
```

To switch into different branch

```
$ git checkout <branch-name>
```

To check all the branches in git

```
$ git branch
```

To merge branch :

```
$ git merge <branch name to be merge>
```

- **Resolving Merge Conflicts**

To show branch and the last commit message and commit hash:

```
$git branch -v
```

To show the already merged branch:

```
$ git branch --merged
```

To show the branch that is not merged:

```
$ git branch --no-merged
```

To delete the branch(gives warning if the branch is not merged):

```
$git branch -d <branch name>
```

To forcefully delete branch with no error:

```
$ git branch -D <branch name>
```

Conflict resolution master

<<<<<<head:index.html

▪ **Pushing Git Branches To Remote Repositories:-**

To push the branch:

```
$ git push <connection name> <branch name >
```

To change the branch name on remote:

```
$ git push <connection name> <old branch name> :<new branch name>
```

To delete the branch on remote:

```
$ git push -d <connection name> <branch name>
```

To rename the connection on remote:

```
$ git remote rename <old name > <new name>
```