

**USE SHIFT+INSERT TO PASTE A LINK IN GIT BASH**

**GIT COMMANDS**

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

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

**git init** = initialises an empty local repo

**ls -lart** = gives us the list of file in .git folder

**ls** = gives the list of files in the directory/folder

**git status** = gives the status

**git add** = adds files to the staging area

**git commit -m “”** = commits the files

**touch <name.(file extension)> (eg. touch hello.html)** = creates blank files in the folder

**git add -A** [or] **git add .** = adds all the files in the folder to the staging area

**git reset <>** = removes the files that were staged/added

**git checkout <filename> (eg. git checkout contact.html)** = if someone edited your file and you want the old file back with latest commit

**git checkout -f** = if you want all those files back say 1000… which had been edited with latest commit

**git log** = gives the list of files that you have committed

**git diff** = compares the working tree to the staging area file(file that is added by git add)

**git diff --staged** = compares the staging area file with the last commit

**git commit -a -m "<>"** = to directly commit your file and skip staging area (its a *bad* practice)

**git rm** <> = if you want to remove the file from the working directly as well as the staging area / we can say delete file from the folder

**git rm -r --cached** <> = the file would be removed only from the staging area … would remain in the folder

**git log -p -2** = will give the last 2 commits in the log , if its -1 then last commit ….used when there are many commits and you wish to see only few of them

**git log --pretty=oneline** (would give a oneline look to all the commits)

**git log –since=2.days** (would give the commits that happened in last 2 days , -> you may even put months/years…)

**git status -s** = shows a better status of your working tree

**git branch <name>** = to create a new branch

**git branch -d <name>** = to delete a branch

**git checkout master**  = to enter the master branch

**git merge <branch name>** = if you want to merge in master you must be in master branch and write the name of the branch you want to merge. YOU MAY GET WARNINGS IF THE MERGE DOES NOT MATCH…

**git checkout -b <branch name>** = to create and enter the branch at same time

NOW TO PUSH YOUR LOCAL REPO TO GITHUB PUBLIC

Firstly , create a remote repo in github and copy the url and type the below

**git remote add origin** https://.....................(url of the remote repo that you have created on github )

{ IT MEANS THAT ADD A REMOTE REPO GLOBALLY/ONLINE NAMED ORIGIN }

( this would connect your local repository and remote repo )

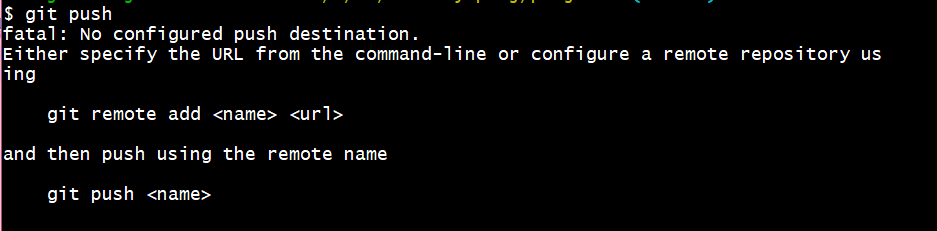
Then : git remote -v = to ensure push/pull links

**git push origin master** = means push my master branch to the origin

{ IT MEANS PUSH MY MASTER BRANCH IN LOCAL TO THE GLOBAL REPO ONLINE NAMED ORIGIN }

**git push origin <branch name>** = to push another branch to repo

NOW IF YOU HAVE A PRIVATE REPO OR YOU WANT TO PUSH WITHOUT LOGGING IN , CREATE A REMOTE REPO:



DO GENERATE THE SSH KEY WHICH WOULD DIRECTLY CONNECT YOUR PC TO GITHUB [ YOU MAY FOLLOW THE DOCUMENTATION and in the end paste the SSH key link on github SSH key space which you get in your git bash (after the tail… step) ]

**git push -u origin master** = to push your local repo

…

SIMILARLY :

**git clone <url>** = would clone the repo to your pc

**git clone <url> <foldername>** = would clone the repo to a folder in your pc (to the folder name…)

ALWAYS INCLUDE A .gitignore FILE IF YOU WANT TO IGNORE ANY FILES

USE: touch .gitignore

AND AFTER OPENING YOU MAY ADD THE FILE NAMES WITH EXTENSIONS IN THE .gitignore FILE.

IF THERE ARE MULTIPLE FILES : \*.<file extension>

TO IGNORE A FOLDER: <folder name>/

{

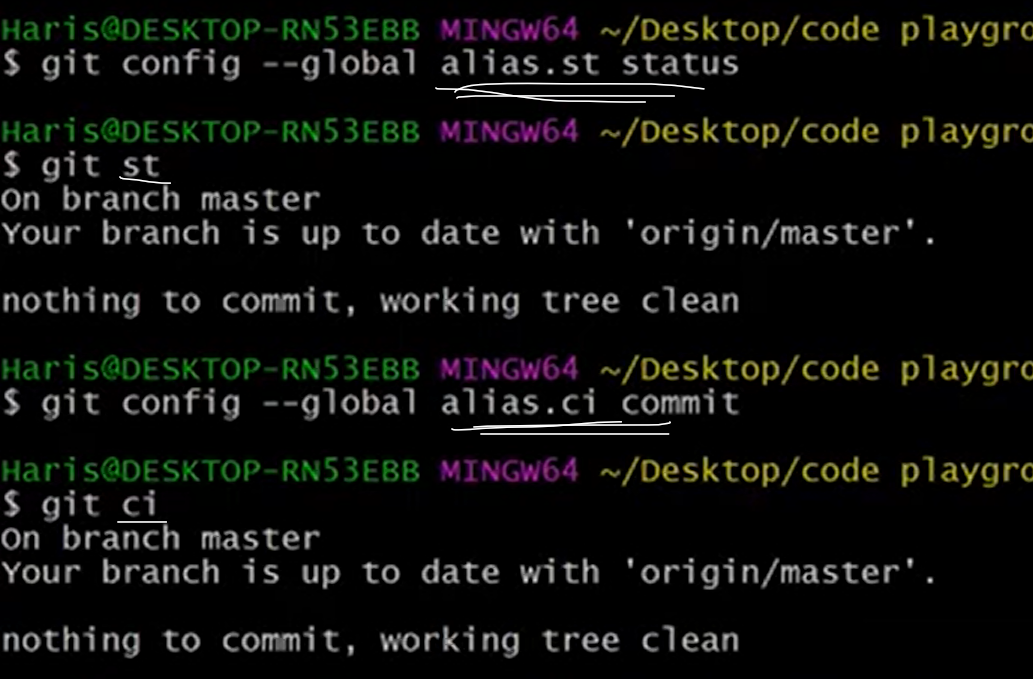
IF YOU HAVE THE FOLDER OF SAME NAME IN ANY OTHER FOLDER , THAT WOULD ALSO BE IGNORED , IF YOU DON’T WANT THAT TO HAPPEN : /<folder name>/

}

TO RENAME YOUR FILE USING GIT : git mv <file.txt…> <new name.txt…>

**git commit --amend = (**TO MERGE YOUR CHANGES TO A COMMIT)

**ALIASES WHICH YOU CAN USE IN GIT :**

****