Git tutorial

Set username

**git config --global user.name "Muhammad Haseeb"**

**git config --global user.email "uhaseeb731@gmail.com"**

check the files using git config --list

Create new repositrory

inititate a new repository:

**git init**

add new files to repository : git add "new.txt"

to add all files :

**git add .**

to add java files:

**git add \*java**

for checking the status use :

**git status**

also use :

**git status -s**

after changing the files present in the repository you can check the status by:

**$ git status -s**

it shows you that the files have been commited you can also commit them into repository by:

**$ git commit -a -m "readme.txt"**

we can get help by using git

we can get help on the webpage by command

**$ git status --help**

and we can get help on git by

**$git status --h**

we can remove the file from directory by using command

**$git rm --cached main.cpp**

and the file main.cpp will be removed from directory

in case you have added all the files to the directory and now you have to remove them all from directory use

**$git rm -r --cached bin/**

now i can create a new file in directory by

**$ touch .gitignore**

now we can easily write in this file and make changes in the file

**$ nano .gitignore**

we can tell the file to ignore the specific extension file,folder and name of the file. After saving the file we can now add all the files freely in directory and commit the files by typing a message added .gitignore and removed bin/ folder.

We can view all git history by command

**$git log**

we can also view specific commits by

**$git log -2**

we can view only one line commit by

**$git log prettyline=oneline**

To ammend the changes in the last git comment use

**$ git commit --amend**

write changes in the file and save and check again

**$git log -1**

we can ammend message in vi core editor by changing config of git

**$git config core.editor "vi"**

then write

**git commit --amend**

vi text editor will open and we can perform functions on it we have to enter into insert mode by pressing I and then edit the commit after inserting press esc to close insert mode and typ :wq to save and move forward we can see the updated commmit by vi editor

If you changes the file and want to commit changes without last commit statemnt you can use

**$git commit --amend --no-edit**

after checking the logs nd matching it with last commit meesage there is only channge in the checksum.

If we edit a file and create a new file and we added both of them to directory, now we want to remove one file from staging area to have different snapshots of both commits and want to commit only one file now we can use

**$git restore --staged readme.txt**

and now it will be removed from staging area. If wewant to undo changes in the file we can follow the command

**$ git restore readme.txt**

To create a remote repository first of all check if there is any remote repository of your files stored in local

**$ git remote -v**

then you can create a remote repository by copying the https: link of your repository created in github and paste it as following

**$ git remote add origin https://github.com/uhaseeb/learngit.git**

now the files are being pushed on your remote repository you can see them in the github repoistory.

to ensure your repository with ssh connection first of all check its connection

**$git remote -v**

to change the connection from ssh to https you have to copy the ssh link from the github account and write command

**$ git remote set-url origin git@github.com:uhaseeb/learngit.git**

now check the connection agqin with

**$git remote -v**

now it will be changed to ssh

now you do some changes in the file and run the command

**$ git commit -a -m "Test SSH connection"**

and then push the changes the github repository by usng the command

**$ git push**

now we donot need to write git push origin master because it was already familiar with the origin master

we can create a new branch from the master branch and folow it by the following code

**$ git checkout -b CodeDebugger001**

now new branch codedebugger has been created and pointed

now we can move to different folder and create new files we are going to create new file bin/ folder

**$touch debug.cpp**

now we can find thelist of files in current directory

**$ls -l**

now we can navigate back to the previous directory by

**$ cd ..**

we can check the log history of specific branch by command

**$ git log CodeDebuggers001**

we can switch to already existing branch by writing

**$ git checkout master**

you can also navigate to the specific commit that has been performed you simply have to write 4 digits of that specific commit

**$ git checkout 4b89**

and you can switch back to your master branch by

**$ git switch -**

You can check all of your branches in repoistory by the command

**$ git branch -a**

we can delete any branch by the following code

**$ git branch -d Test**

Test branch will be deleted from the repository

To delete branch from the remote repository also we can use this command

**$ git push origin --delete Test**

now it will be removed from github also

You can push a branch on github/remote repository by the following command

**$git push origin CodeDebugger001**

You can also use switch to navigate to diiferent brnches

**$git switch CodeDebuuger001**

Using switch to navigate betwwen branches is a good option than checkout.

Now after successully pulling your request on git repository and merging your working branch directory with the master directory all the content and commits of branch directory has been stored on master branch. now you have to pull all your changes into your local repository on master origin. you have to use the comand :

**$git fetch**

But after running this command we have noticed that changes/commits are not being implemented on local repository but why? because fetch command only tells about the information to the local repsotiry that the change has been implemneted in the master origin repository.

to implement the change you have to run the command :

**$ git merge**

and now all the changes are implemented on your local repository

But we can you another powerful command to implement fetch and merge at a same time with the help of pull command

**$git pull**

Now our best practice is to keep our git history clean so for this approah we are going to squash the commits the same commits or the commits that are being performed on the same file for this purpose we have to use the commnad

**$git rebase -i HEAD~3 (we are squashing 3 commits)**

now notepad will open and you have to edit the other 2 commits and squueze then to the first commit by writing s in the start instead of pick

We have basically 3 types of reset command in git. In Soft reset the files are present in the directory but not been commited we have to commit them

**$ git reset --soft 3410**

(3410 is the commit number on which soft reset is performed)

In mixed reset the files are removed from staging area and we have to add then in the staging area and then perform commit on them

**$git reset --mixed 3410**

In the hard reset all the files are reoved from the .git local directory

**git fetch --all**

with the help of this command we can fetch all the files back of the remote repository

**git reset --hard origin/master**

this command will reset back all the changes in the master branch

we can also reset with the help of head pointer

**$git reset --hard HEAD~2**

The main difference between git reset command and git checkout is that in checkout master and head doesnot point to the same commit in .git directory but in git reset both master and head point to the same commit.

If we were asked to discontinue your important task and start a new more important task but we donot want to loose our changes in the branch we can use stash command

**$git stash apply**

now after this changes were being stored in stash and we cn do our new task easily If we want these changes in our new task then we have to use

**$git stash apply**

to view the list of stash use comand

**$git stash list**

we can add untracked files in our stash by

**$git stash -u**

and when you have completed work on super master branch you can create new branch from stash

**git stash branch new-task**

we can drop any stash from out list by

**$git stash drop stash@{0**}

Now if we have lost our commits and we hve to restore them into our memmory first we have to reset our commits to perform reflog

**$git reset --hard HEAD~2**

we are losing our last to commits

now we can find our commits in git reflog history

**$git reflog**

we can find all the commits here . now we have to copy commit no and press q to quit

create a new branch and recover these commits here

**$git branch lost-data 4b51**

and now check the log hostory of branch and we can find the lost commits are in th history

we can also specify check history of commits

**$git reflog --since="1.hour"**

Now we will move to the concept of cherry-picking it is basically a process in which commit from one branch is copied to the other branch

first of all select the commit from git log from the branch you want to duplicate check for the commit and copy first four digits now create a new branch and switch to that branch

**$git switch -c lost-data**

in this branch open the file on which the commit is being performed in other branch in this case we are going edit index.html open the file and add one new line in the file now perform

**$git add .**

and then

**$git commit -m "Some commit"**

now execute

**$git cherry-pick 0c92** (the digits copied from commit of the other branch)

there will be an error in cherry picking process and shows the branch is in the state of cherry-picking now check the log andyou will see that index.html is in the unmerged path you have to edit index.html remove the conflicts in the file and then add it into the directory now perform

**$git cherry-pick --continue**

after completing the cherry-pick process you will see that in the git log history of branch same commit is present in history but the commit number or digit is diiferemt from the commit in the other branch.