Git basic commands

#Git

--> **Download git**

* In Linux-> yum install git

* window->

-- download git bash tool from Internet

-- https://notepad-plus-plus.org/downloads/v7.8.6/

#To initialize the git workspace

--> **git init:-**

This command turns a directory into an

empty Git repository

Directory----------Git repository

#To check git version

--> git --version

Create directory

* mkdir krishan-repo

* cd krishna-repo/

* git init==>initilize an empty git repository.

#Add files to the staging area

--> git add:-

Add files to the staging area

for Git

Files----------Staging Area

(Before a file is available to commit to a repository.the file needs to be added to the git index (staging area))

* touch krishna.py
* gedit krishna.py
* git add krishna.py
* git add .(to add all the file in staging area)

#Record the changes made to the file in a local repository

--> git commit

Record the changes made to the file in a local repository

Staging Area----------local repo

(For easy reference each commit has a unique ID)

.good to give commit a message

.It help us to tell what changes was done is particular commit.

* git commit -m "first commit"

#This command returns the current state of the repository

--> git status:-

Returns current working branch | if a file in the staging area,but not committed,it shown with git status

|

|

(If there are no changes it'll return nothing to commit,working direc clean)

* git status
* touch hello.py

* gedit hello.py

* git add hello.py

* git status (it show us status)

#Configuration of github in git

--> git config:-

Name E-Mail

| |

| |

----------|----------

Name and email address

assigned to commit from a local

repository.

(With Git, there are many configuration and setting possible.git config is how to assign these settings.Two important settings are user user.name and user.email

* git config --global user.name "krishna"

* git config --global user.email ""

#Branch and Merge

--> Branching:-

=> Git Branch

| **git branch** :- Checkout your current branch |

| **git merge** :- Integrate branches together |

| **git checkout** :- used for switching branches |

* git branch (To see all the branches)

* git branch devloper

* git branch:-

    Here we see two branches that is *master and *devloper

* git checkout devloper (This point towards our feature branch)

* ls

If we want to change something in file

We have two file that is krishna.py and hello.py (I want to change in krishna.py)

=> Again use whatever editor we want

* gedit krishna.py(In dev branch i have made changes to krishna.py file)

=> Add this changes at the modify file in the staging area

* git add . (I want all the changes should be there in my staging area so git add and period)

=> Now i wanna commit this changes

* git commit -m "developer commit"

If i want to merge developer branch to master branch

1. check master branch

   * git checkout master (It switch to branch 'master')

2. We merge

   * git merge developer (It will merge developer branch with master branch)

If we want to delete our developer branch

   * git branch -d developer(It will deleted feature branch)

One more way to create a branch that is:-

* git checkout -b Name of your branch (new developer)

    - It will not only create a branch(new developer) but also check in that branch.

To make changes in file

--> open editor gedit hello.py

--> Do changes save it

now again

* git add .

* git commit -m "new developer"

==> How to connect remote repository

--> I have remote repo in my github account, i created repository there

1. I want to connect with that repository.

- I need to add that origin

* git remote add origin and the ssh link

(https://github.com/KrishnaSharma25/ML-Feature-Extraction-Method.git)

- We successfully added the origin

=> We created local repository with github account

#Working With Remote Repositories:-

--> git remote-

Local ==========> Remote

Repository Connects a local Repository

repository with a

remote repository

- krishna-repo is my local repository as shown above

- A remote repository i have shown in my github account

--> git clone-(to copy and download the repository to local computer)

Remote ==========>Local Working Copy

Repository Creates a local working copy

of an existing remote repository

-- cloning is equivalent to git init when working with a remote repository, git will create a directory locally with all the files in repository history.

1. I wanna create one more directory
    * cd ..
    * mkdir git-repo
    * cd git-repo/(moving in this repository)
    * git clone and ssh link
        (https://github.com/KrishnaSharma25/ML-Feature-Extraction-Method.git)
        - It clonning git commands
    * ls (To check what files are there)
    * cd git-commands/
      Here we have krishna README.md

#Pull and Push Concept

--> git pull-

Remote ==========>Local computer
Repository This pulls the changes from the remote repository to local computer

---> Now i can go ahead and pull whatever changes i made in the file that is there in the github account

1. git hub
    krishna (make some chnages)

---> Now i want all the changes i have made in my remote repository in local machine

2. git

* git pull origin master

* ls (To check whatever files we have)

* gedit krishna (Here changes have been reflected)

--> git push-

local ========== Remote Repository Sends local commits to the remote repository Repository

---> Now let us see how to push changes in the remote repository

* touch git25.py

* gedit git25.py (make changes to this particular file)

-- Before we push to the remote repository, we need to add this changes in the staging area, we make a commit and then only we push it.

* git add git25.py

* git commit -m "Remote"

* git push origin master (on github we see git25.py file added)

Some Advanced Git Commands:-

1. git stash :- To save changes made when they're not in a state to commit them to a repository

2. git log :- This helps give context and history for a repository

3. git rebase :- Takes a set of commits, copies them and store them outside the

repository

4. git revert :- It helps you to roll back to the previous version of file

==> Use of git stash:-

* touch stash.py (Creating a new file)
* gedit stash.py (make some changes)
* git add . (Adding to the staging area)
* git status (It will show new file in the staging area)

  -- It is not looking good, so i can put all the uncommited changes to stash.
* git stash -u
* git status

  -- It converted my dirty directory to clean one with the help if git stash.
* git stash list
* git stash show (If we want to inspect)

==> Use of git log:-

- create a new repository
  * mkdir git-log (name of my directory)
  * cd git-log/ (go into the directory)
  * git init (to initilize it)
  * gedit krishna1.py
  * git add . (add in staging area)
  * git commit -m "log" (finally commit it)
  * git log (It shows the commit history for the repository)
  * git log -before="give Date here" (It provide parameter here as well)
  * git log --author="name of author" (show commit based on the author)
  * git log --before="date" (It give according to date as well)

==> Use of git revert:-

- How to revert to the previous commit

1. make some changes in file again

   * gedit krishna1.py

2. Add to the staging area

   * git add .

3. commit it

   * git commit -m "last commit"

   * git log --oneline(It show in one line)

4. GO back to the previous commit

   * git revert 7af537f(last commit)

   * cat krishna1.py

-- Now i go ahead and revert to the last commit as well

   * git revert HEAD

   * ls

   * cat krishna1.py

   -- whatever file changes that have been done after git revert will be reflected commit itself.

==> Use of rebase:-

rebase is the way of combine the work between the branches

-- What rebase does:

1. Take set of commits
2. copy them
3. store them outside our repository

Advantage of rebase is that- It can be used to make linear sequence of commit.

   * git rebase master(It show current master up to date)

-- move our work from current branch to master branch

-- They look like they developed sequentially, but they developed parallely.

-- Create branch

* git branch krishna2507
* git checkout krishna2507 (It switched to branch that we have created)
* ls
* gedit krishna25.py(make some changes)
* git add .
* git commit -m "rebase"
* git rebase master(It show current branch krishna2507 is up to date)