**01.Git Quick Start**

**Yellow** git command

**Green** linux/editor command

A. Config user information

To set user details

git config --global user.name "Bibhu Prasad Samal"

git config --global user.email [bibhusprasad@gmail.com](mailto:bibhusprasad@gmail.com)

To view all, configure list

git config –list

Create first git project

pwd show current working directory.

git init create git project inside working directory .git

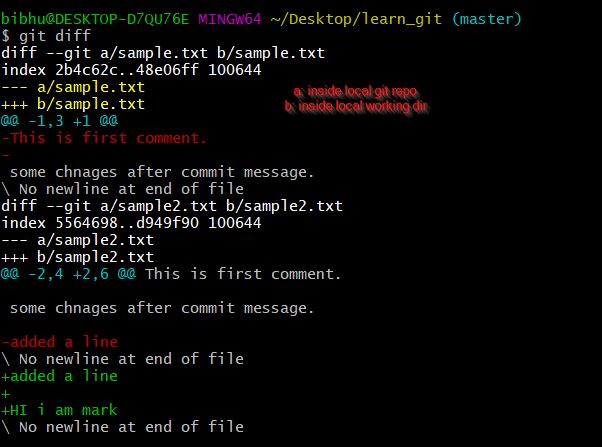
ls -la show hidden file inside that directory

git add . add all untracked file

git commit -m "my first commit" commit message

git log shows all commit log

git diff show difference in modified file (inside staging and working area) (not new file or upstage file).



git diff --staged show difference in staged file (after add . )

git rm sample3.txt delete a file

Add SSH Key in Local from GitHUB

**Checking for existing SSH keys:**

https://help.github.com/articles/checking-for-existing-ssh-keys/

**Generating a new SSH key and adding it to the ssh-agent:**

https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/

**Adding a new SSH key to your GitHub account:**

https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/

B. NotePad ++ As GIT Editor for Windows

First set notepad++ folder in system variable path

git config --global core.editor "\"C:\Program Files\Notepad++\notepad++.exe\\"

start notepad++

alias npp='start notepad++'

**02.Git with GITHUB ADVANCED**

A. Centralized GIT Repository

Create a new repository in GitHub account. Create a local folder inside your machine. Open git bash there.

$ git init Initialized empty Git repository in your local machine

$ git remote add origin <https://github.com/bibhusprasad/010_git_learning_exercise.git> link with your remote origin branch

$ git pull There is no tracking information for the current branch.

Please specify which branch you want to merge with. If no tracking branch is there then we need to specify the branch name.

$ git pull origin master now master branch is in sync

$ git status show the history of changed file

$ git add . add all changed file to staged memory

$ git add filename add particular file to staged memory

$ git commit -m "My first commit" add commit message

$ git push origin master push changed file to centralize git account

$ git log show the log details

B. Create and Merge branch in GIT

$ git branch develop to create a branch

$ git checkout develop to checkout or move to develop branch

$ git branch Show all branch name

$ git merge develop to merge develop branch to master branch in local. checkout master branch at their run above command. After that push changes to master

$ git push origin master push merge changes to master git server

$ git pull origin master pull changes from master branch to another branch

C. Revert a commit in GIT

$ git reset HEAD -- . to reset all files from staging area to working area

$ git reset HEAD Sample.txt to reset a particular file from staging are to working area

$ git reset HEAD~ to revert a last commit i.e. after git commit -m if you want to revert that last changes.

**03.Git Basic Through Explanation**

A. Add and Fork Existing project to GIT

$ git init Initialize .git inside project folder

$ git remote add origin <https://github.com/bibhusprasad/010_git_learning_exercise_02.git> add remote origin branch to your local existing project

$ git pull origin master pull master changes to your branch. If you are getting “fatal: refusing to merge unrelated histories” error message then try below command

$ git pull origin master --allow-unrelated-histories after this push your changes to master

For Fork go to branch you want to fork. Then click on fork button. Then that project will get in your github account.

To clone a project. Click on clone or download button. Copy the http/ssh link

$ git clone <https://github.com/bibhusprasad/010_git_learning_exercise_02.git> to clone an existing project in your local.

> ls -al ls: list the file a: all l: list

> touch newText.txt create a new file

> echo "some random text" > newText.txt add some text into the text file

B. Back-out git changes

> vi test.txt open a file in VI editor

git checkout -- test.txt to revert all changes on a particular file after some modification. (present in red colour)

C. Delete git file

git rm level1/level2/level2.txt to delete a file from local

To get the file again we need to do

git restore --staged level1/level2/level2.txt first need to upstage

git restore level1/level2/level2.txt

git rm -rf level1/level2/ to delete all the folder or file r: recursive and f: forcefully

D. Rename and Move file in git

> mkdir level1

> touch level/level.txt

git mv level1/level1.txt level1/level1rename.txt git mv <old-file-name> <new-file-name>

git mv level2/level2.txt level3 git mv <file-name> <new-folder-name>

E. Git history

git log

git log --oneline --graph –decorate

git log --since="2 days ago"

git log -- level1 show all commit to a particular file

git show 8d96089740dbe43f0e3fe3df7dba6dd11005c715 show details about a particular commit id

E. Git Ignore Pattern

Specific File : file name

File Pattern: \*.txt

Folder: level1/

**04. Comparisons in GIT**

git diff test.txt

git diff HEAD test.txt compare working dir with stage dir i.e. after commit

git diff --staged HEAD test.txt compare stage area with REPO

git diff 2dd05bb e4ccfb1 compare two commits

git diff HEAD HEAD^ compare last to last commits

**05. Branching & Merging in GIT**

git branch -a show number of branches in GIT

git branch mybranch to create a branch

git checkout mybranch to switch mybranch

git branch -m mybranch develop to rename a branch from mybranch to develop

git branch -d develop to delete a branch

A. Happy Path Merges

Create a new branch mybranch

git branch mybranch

Checkout to mybranch

git checkout mybranch

Change a file inside mybranch and add that file to by using add and then commit to mybranch. Then we need to commit mybranch changes to master. For that need to go master branch

git merge mybranch git merge target branch name

B. Conflict and Resolve Merge

Go to mybranch. Change in test.txt file and commit your changes. Change the branch go to master and change in the same file test.txt

git merge mybranch -m "merge mybranch changes to master"

Now conflict message will come for test.txt file

<<<<<<< HEAD

add one line for conflict in master branch

add two line for conflict in master branch

=======

added one line for conflict resolution

added two line for conflict resolution

>>>>>>> mybranch

**06. GIT Rebase**

A. GIT Merge vs GIT Rebase

git checkout -b develop create a new branch develop

change test.txt file and commit changes then switch to master and change in newTest.txt and commit the changes. Then checkout to develop branch

git rebase master Successfully rebased and updated refs/heads/develop.Now head is point to develop. Commit all the changes of master to develop. But develop branch changes will not present in master

Now go to master branch add some changes in humans.txt file and commit the changes.

git merge develop now develop branch will merge with master.

git pull --rebase origin master to rebase with origin branch

B. Conflict and Resolve Rebase

git checkout master go to master base. Then do changes in test.txt file and commit the changes. Then go to develop branch and do changes in the same file. And commit the changes. Then go to master and do changes in the same file and commit changes. Again, go to develop

git rebase master now it will through error. So abort the rebase changes

git rebase –abort to abort the rebase

git diff master develop to diff the files. Now again do git rebase master, then resolve conflict. Then commit the changes

git rebase –continue Successfully rebased and updated refs/heads/develop.

**07. GIT Stashing**

A. GIT Stash with Example

touch access.log create access.log file in master and add some text. And modify test.txt file.

git stash It will move modified file to stash area. I,e, only text.tst file

git stash list stash@{0}: WIP on master: 89629b7 local rebase commit1. show all list of stash

git stash apply apply stash changes again to branch

git stash clear to drop stash list/details

B. Stash Untracked Files & Git Stash Pop

Now do some changes in text.txt file and it is in modified state. And access.log is still in untrack mode.

git stash stash modified file but not untracked file i.e. access.log file

git stash -u to stash untracked file. i.e. acess.log file.

git stash -a to stash all file i.e. both modified and untacked file

git stash list

stash@{0}: WIP on master: 8768891 stash changes added

stash@{1}: WIP on master: 8768891 stash changes added

git stash pop to apply/revert most recent comment i.e. stash!{0}. Now access.log file is back to branch.

git stash list

stash@{0}: WIP on master: 8768891 stash changes added

git stash pop to apply/revert most recent comment i.e. stash!{0}. Now test.txt file is back to branch.

git stash list no stash is present to apply

C. Manage Multiple Stash and custom stash name

Now We have one modified file and one untrack file

git stash save "first stash" stash modified file

git stash save "second stash" -u stash untracked file

git stash list

stash@{0}: On master: second stash

stash@{1}: On master: first stash

git stash show stash@{1} Show details about stash@{1}

git stash apply stash@{1} to add particular stash.

git stash list

stash@{0}: On master: second stash

stash@{1}: On master: first stash

git stash drop stash@{1} to drop a applied / particular stash

git stash list

stash@{0}: On master: second stash

D. Stashing into Branch

Now do some changes in text.txt file and it is in modified state. And access.log is still in untrack mode. Do some changes in newtest.txt file and move that file to stage area. i.e. by add file name command

git stash save "first stash" -a to stash all file untracked, stage and unstage file to stash area with name first stash. Then switch to new branch

git stash branch newBranch a newBranch created and all the stash applied to new branch and drop first stash

git stash list no stash list present

**08. GIT Tagging**

git tag rel-v1.0 create a tag with name rel-v1.0

git tag --list show all tag list

git tag --delete list delete a particular tag

git tag -a rel-v2.0 i.e. the source code is specific to this release

git show rel-v2.0 show the particular tag details

git tag -a beta-v1.0 9d8b86f to tag a particular commit

git diff rel-v1.0 beta-v1.0 diff between two tag

git tag -a beta-v1.0 -f d48e9d1 to update a particular commit

**09.** **GIT Common Mistakes and Fixes**

A. Amend comment message

git checkout test.txt to revert the changes in a file. It will revert the changes with the HEAD.

git commit --amend -m "right message" if wrong commit message submitted then amend will to change the wrong commit message to right one

B. Git reset and clean

git reset --soft 86243b3474170084cf4472d34a776e047597cf55 to reset application to a particular commit and move all the file to stage area

git reset 86243b3474170084cf4472d34a776e047597cf55 to reset application to a particular commit and move all the file to unstage area

git reset --hard 86243b3474170084cf4472d34a776e047597cf55 to reset application to particular commit and all the changes will gone. NO data will left

git reflog show all the logs. All deleted present everything

git clean -df clean a complete directory. i.e all the file will be deleted from particular directory.