Advanced git commands

Merging vs Rebasing

Restting, checking out, and Reverting

Advanced Git log

Git Hooks

Refs and the Reflog

Git submodules

Git subtree

Large repositories in Git

Git LFS

Git gc

Git prune

Git Bash

How to store dotfiles

Git Cherry Pick

GitK

Git-show.

**GUI Tools for Git**

GitKraken

SourceTree

TortoiseGIT

Git UI

**Git Cherry pick and git rebase git squash**

$ cd gitr

$ cd dir1

dir1$ touch a

$ git add . && git commit -m “a first ”

$ touch b

$ git add . && git commit -m “b”

$ touch c

$ git add . && git commit -m “c”

$ git log –oneline

$ git checkout -b win

$ touch d

$ git add . && git commit -m “d”

$ touch e

$ git add . && git commit -m “e”

$ touch f

$ git add . && git commit -m “f”

$ touch g

$ git add . && git commit -m “g”

$ git log –oneline

// merge will go in timestamp manner,

// rebase will go on top of master branch

// If I don’t want to merge the entire branch and wants //to merge only particular commit. Then we can go for //cherry-pick

$ git checkout master

$ git cherry-pick id

// this can merge commit id into master.

// you can combine 2 commits also.

$ git cherry-pick id1 id2

$ git log –oneline

$ ls

// now I want one file / commitid to be in top position

// rebase is the solution

// always leave the first commit , consider one less than total commits

$ git rebase -i HEAD~6

// now you can see the one file which is having all the

// commits .

yy – cut

dd-- delete

pp – paste

// now copy and paste commits according to your wish

// and save the file.

$ git log –oneline

//Now HEAD is pointing to our commit.

SQUASH

// Consider commit ids are more.

// for docker file you made lot of commits

// squash reduces the number of commits

$ open the commitid file

$ git rebase -i HEAD~6

$

**Squashing** reduces the number of commits.

**Amend**

Touch m

Git add . && commit -m “m”

Touch a

Git add. && commit -m “m” –amend

Touch b

Git add . && commit -m “m” --amand

Touch c

Git add . && commit -m “m” --amend

**Under single commit id all changes saved using amend.**

**Git log –oneline => It shows only one commit id**

**Git show m =>**

**Git show commitid => it shows all the commits under single commit id.**

**stash**

$ Touch file2

$ Git stash

$ touch firle3

$ git add file3 => now I don’t want to commit.

$ git status

$ git stash

$ touch file4

$ git add file4

$ git status

$ ls

File1 file4 => what happen to file2 and file3

$ git stash list => stash files stores in the form of array.

$ touch file5

$ git add file5

$ git stash

$ git stash show stash@ {0}

$ git stash show stash@{1}

$ git stash show stash@{2}

Pull the stashed file and commit any time.

$ git stash apply stash@ {1}

$ git status => now file in commit area

$ git commit -m “file3”

$ git stash list

$ git stash show stash@{1}

$ git stash drop stash@{1}

$ git stash list

$ git stash pop => pull the latest and delete.

**Stashing normally do to maximum 5 commits.**

**Switch**

**.gitignore**

**Git stash -a**

**Restore**

**Reset vs Revert**

Soft => git reset –soft

Undo from repository (HEAD)

Git reset –soft

mixed => it is the default reset

undo changes from repo and undo changes from staging area also.

hard=> It will repo, staging and file

git reset –hard

git reset –mixed

**Undoing changes before commit(reset)**

**Undoing chages after commit.**