Task-1 git By Prasad.

SCM= Source code Management tool

CI= Continuous Integration

CD= Continuous Delivery

Jenkins= Plugin based tool

GIT commands:

1. How to initialize git repository or how can we create git repository?

> $git init

By default, git manager is MASTER

1. How can we see the status of the files in git repository?

>$git status if o/p file color is green =staging, red= workspace.

1. How can we configure username and mail IDs?

>$git config --global user.name “DevOps”

>$git config --global user.email “DevOps@gmail.com”

1. How can we see the all configuration files?

>$git config --list

1. How can we see local repository files?

$git log

1. How can we add files from workspace to staging phase?

$git add filename

1. How can we add multiple files at a time from workspace to staging phase?

$git add file1 file2 file3 …………

$git add .

$git add \*

$git add –A

$git add –u (to add all changes along with modified changes of files)

1. How can we revert back file from staging to workspace?

$git reset head filename

1. How can we revert back file from local repository to staging?

$git reset --soft cid (previous cid)

Here we must use before versions cid. To reset back file from local repository to staging we need to use previous cid

1. How can we revert back file from local repository to staging with last cid?

$git reset HEAD~1 (~=tilde)

1. How can we revert back file from local repository to workspace directly?

We can’t add a file directly from work space to local repository. But we can revert back files from local repository to workspace directly. But while doing this process it collects files from staging also. But the good way is (local->staging->workspace)

$git reset –mixed cid

1. How can we edit file by without using reset command as well as without terminating file in local repository.

$vi filename or $vim filename

1. How can we add modified files directly to local repo?

We can add a modified file directly from workspace to local repository by using this command.

$git commit –am “label” filename (only modified files can add directly)

1. How can we copy the central repo from github?

$git clone pathofrepo (similar copy of file download from central repo to local machine)

$git push (To push the data from local repository to centralized location or git hub)

$git show (to see last committed file’s cid)

$git pull (to copy latest changed files from git hub to repository).

$git log --oneline (to get a log result in one line)

$git log -number (to get a log result in last commits by number)

$git log --oneline –number (to get a log result in one line along with required number)

$git log --author=authorname (to see particular author commits)

$git log --since=YY-MM-DD (to see commits which are done in particular time from midway to end)

$git log --until=YY-MM-DD (to see commits which are done in particular time from start to midway)

$git log --since=YY-MM-DD --until=YY-MM-DD (to see commits which are done in particular time from midway to midway or certain range)

$git branch (to see which branches we are having).

$git branch branchname (to create a new branch)

$git checkout branchname (to change cursor from one branch to another branch)

$git merge branchname (to take data from another branch)

$git branch -d branchname (to delete a branch)

$git branch -D branchname (to delete a branch forcefully)

We can’t delete a branch when we are inside of it. So we have to go another branch and delete or remove which are need to delete.

$git push path \_of \_central \_repository branchname (to create new branch in central repository for branch along with pushing data)

(OR)

$git push origin branchname (to create new branch in central repo and to push data)

$git push origin -d branchname (to delete a branch in central repository)

$git push origin -D branchname (to delete a branch forcefully in central repository)

How can we delete file of central repository through git commands

Step 1: $rm filename>ls>$git status>$git add .>$git commit –m “label”

Step 2:$git push (go to github browser and refresh it and check)

How can we delete directory (file inside it) of central repository through git commands

Step 1: $rm –rf dirname>ls>$git status>$git add .>$git commit –m “label”

Step 2:$git push (go to github Brower and refresh it and check)

Step 3:$git log (it tells about recently deleted files)

Step 4: $git show cid (by this we can know full information of file who deleted a particular file by giving its cid)

Conflict problems:

Whenever multiple developers are working on same file or same directories. Incase if they give same filenames in different branches. Then at the time to merge files from same master to another then we will get conflict problem. To come out from those conflict problem we use command

$git merge --abort

Whenever if we add something data in a file intermediately which was already exist. In that case if we want to see difference of previous data our addition data. The command is

$git diff filename (for files)

$git diff file1 file2 (b/w 2files)

$git diff master master1 (b/w 2 branches)

$git cherry-pick cid (whenever we committed, then it commits lot of files at a time. In that case if we want to get only particular data of file, then we get that by cid)

Stash Memory: it is a temporary storage area in our local machine. But it won’t in central repository area.

Which type of data store in stash memory?

We can add only staging or index area files only

$git stash list (to know the stash memory list)

$git stash save “label” (name created automatically itself)

$git stash show –p stash\_memory\_name (to see the files and data inside it)

ALIAS

Aliasing in nothing to give a temporary code names to the existing actual codes.

$git config --global alias.aliasname “actual command”

$git config --global alias.s “status” (giving aliasing name to status command as “s”. from to see status $git s is enough)

$git config --global alias.l “oneline”( giving aliasing name to oneline command as “l”. from to see oneline: $git log --l is enough).

To remove aliasing names

$git config --global --unset alias. “aliasname”

$git config --global --unset alias.l (removing aliasing name to oneline command as “l”).

$git config --global --unset alias.s (removing aliasing name to status command as “s”).

TAGS

Tag is nothing but a release. To mark the release code

$git tag (to show all tags in machine).

$git tag tagname (to create a new tags in machine based on last cid in machine).

$git tag –d tagname (to delete a particular tag).

Ex: $git tag –d v-1.1

$git tag tagname cid (to create tag up to particular cid. Need to pass cid otherwise it took latest cid)

$git show tagname (to show full information about that tag)

How can we store tags in central repository.

$git push origin tagname (to push a particular tag into central repository)

Ex: git push v-1.1

$git push origin --tags (to push all tags into central repository at a time).

$git push origin --branches (to push all branches at a time)

$git push origin –d tagname (to delete tag from central repository)

$yum install git (to install git in Linux)

AMEND: To change commit message or label message we use amend but it was works for only latest labels only. –amend

$git commit –amend –m “label” (to change the label message or commit message)

$git commit –amend –m “label” (to commit new files into already existed commit id)

$git show cid (to see how many file in our cid)

IGNORE

Whenever we made any changes in any file, those files automatically comes into workspace area or local repository. To hide those files which are keep on changing and show as per requirement we use ignore command.

Add files into .ignore file hide file. And remove from .ignore when we want to see that file. To unhide them, remove in hide file MERGE & REBASE

Whenever we merge one master into another master we will get an extra and unwanted cid. So to overcome that we can use rebase command. Here instead of merge we use rebase.

Ex: git rebase \_master branch name

$git pull {push =fetch+merge} (it took the all data from central repository to local repository)

$git push (it push the data from local repository to central repository)

$git clone pathofcentralrepo

$git branch –r (it lists origin master)

$git branch –a (it lists normal master and remote master)

$git branch (it lists normal branches)

$git show origin/master (to list the originmasters data)

$git merge origin/master –r (it lists fetch and updates of origin master)

CHERRY-PICK

To copy or get a particular commit from one branch to another branch we use cherry-pick command

$git cherry-pick cid

Task-2 git By Khaza.

* Git will not detect(gurtinchadam) empty folders as changes because git works with files and paths of files
* **Untracked** files are the newly added files which are not part of git local repository and modified are the changes done the existing files which are part of git local repository
* **Branch creation commands**
* git branch <branch-name>
* git checkout -b <branch-name> creating a branch along going to inside of that branch
* **To move from one branch to other**
* git checkout branch\_name
* head is always look for latest working branch or commit
* To check mode of file **$git cat-file -p (23d6214)cid**
* To check type file **$git cat-file –t cid**
* **Git objects:** 
  + Two major object types in git are
    - **Blob**: represents a file (giving information about files)
    - **Tree**: represents a directory/folder
  + We have the following object types
    - **Commit**: Commit is set of changes submitted along with date time, message, author name and email.

To check mode of file **$git cat-file -p (23d6214)cid**

To check type file **$git cat-file –t cid (**it tells file type like blob or tree**)**

* HEAD file in .git folder represents the branch. It should ideally point towards the branch. If HEAD is pointing to some commit rather than branch it is called as DETACHED HEAD state
* git cat-file -t cid (to see the type of file)
* git cat-file –p cid (to see the parent cid or contents of commit)
* git cat-file -t firstcommit cid (gives documents information)
* The only option to de-code the content is “cat-file –p” command

Merging Branches:

* Fast-Forward: When branch moves to the latest commit of branch which we are merging from i.e. considered as fast forward
* Merge: This creates an extra commit on the destination branch which has merge commit (which has two parents)
* Steps for merging:
  + Checkout to the branch where we need to merge  git checkout <dest-branch>
  + Now execute the command git merge <source-branch>

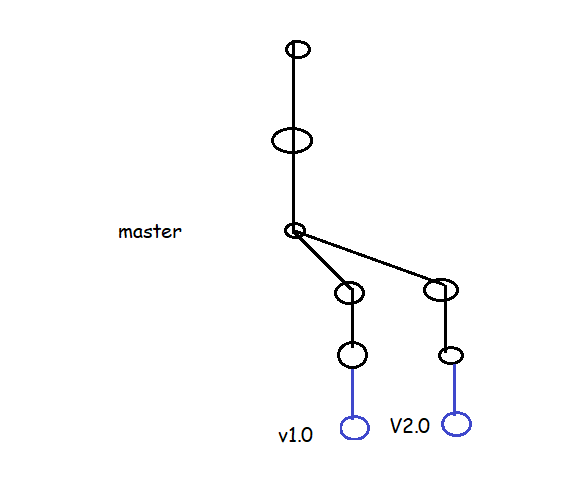
To move one individual commit from one branch to another branch we use cherry-pick command. Here we need to add last commit of v2.0 to v1.0. Then first we need to checkout to that v1.0 branch, later we need to cherry-pick that last commit of v2.0. Here we might be to get a chance of conflict problem, if we get conflict here then remove those arrow marked lines by vim editor.

Then $git status

$git add filename

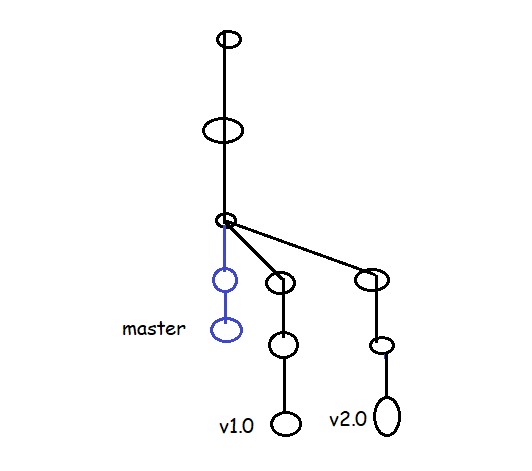
$git commit (no need to give -m and label here)

* **Post Cherry-pick**



## Rebase

* As shown in the below image we have made changes in the master branch and we want those changes in the v1.0 and v2.0 branches



What are the commits we are done in master branch, if we want those commits in v1.o, v2.0 or other branches, we can do in it by two ways

1. cherry-pick

2. rebase

1. By following cherry-pick we can bring only one commit at a time. It takes lot of time if we have to bring number of commits.

2. **rebase**: as shown in above image we can do by following rebase command.

Made changes in master branch, later

$git checkout v1.0 || v2.0

$git log --oneline || ‘

$git rebase master ||

$git log --oneline ||

Meanwhile if we got conflict errors, edit files by vim

$git add .

$git rebase --continue

* View history of last n commits

git log -n

git log -3

* View the graph

git log --decorate --graph --oneline --all

* Search commit by message

git log --grep "expression"

* Viewing the differences b/w any two commits

git diff <commitA>..<commitB>

git diff <branchA>..<branchB>

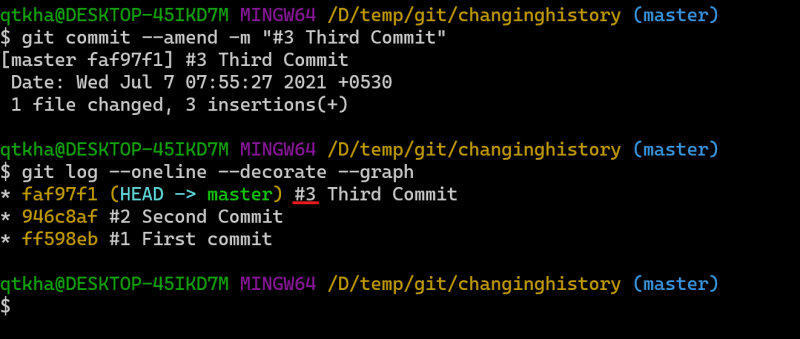
git diff --name-only <commitA>..<commitB>

git diff --name-only <branchA>..<branchB>

* Tree /f 🡪to see list of folders in tree structure

AMEND: To change commit message or label message we use amend but it was works for only latest labels only. –amend

* $git commit --amend -m “label” (to change the label message or commit message)
* $git commit --amend -m “label” (to commit new files into already existed commit id)
* $git show cid (to see how many file in our cid)



\*The Git commit is used to commit the changes from staging area to local git repo

\*git commit --amend can be used to change the message of the immediate commit

\* We can also perform operations such as:

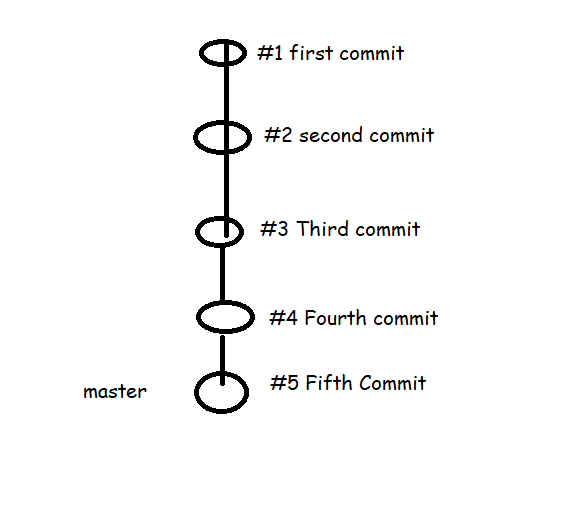
\*deleting commits

\*changing commit messages

\* combining commit messages

\*For the above mentioned changes we need to do interacting rebasing

* Change the commit messages way back in history



* Understanding the commits and changes;
  + Now let’s use the pattern #0001 from #1 i.e single digit to four digits
* git rebase -i --root
* git rebase -i HEAD~2
* Delete the commit in the history. Let’s delete the second commit

git rebase -i HEAD~4

drop commit 2

* Combining two commits into one commit
  + Here we use squash option

Git rebase -i HEAD~2

pick third commit

squash fourth commit

pick fifth commit

## To get clone the remote repo into local repo

$ git remote add origin https: //github.com/na resh808 /ltecommerce.git

$ git push -u origin master

## INIT: init means we don’t have repository and we need to create a repository.

## Clone: means we have repository and need to add that repository into local repository.

## Done upto khaza 8 classes