**Devops(Development + operation)-Methodology**

**Version Controlling System/SCM – GIT, Bit Backet, Mercurial**

**But in Market Mostly Used GIT.**

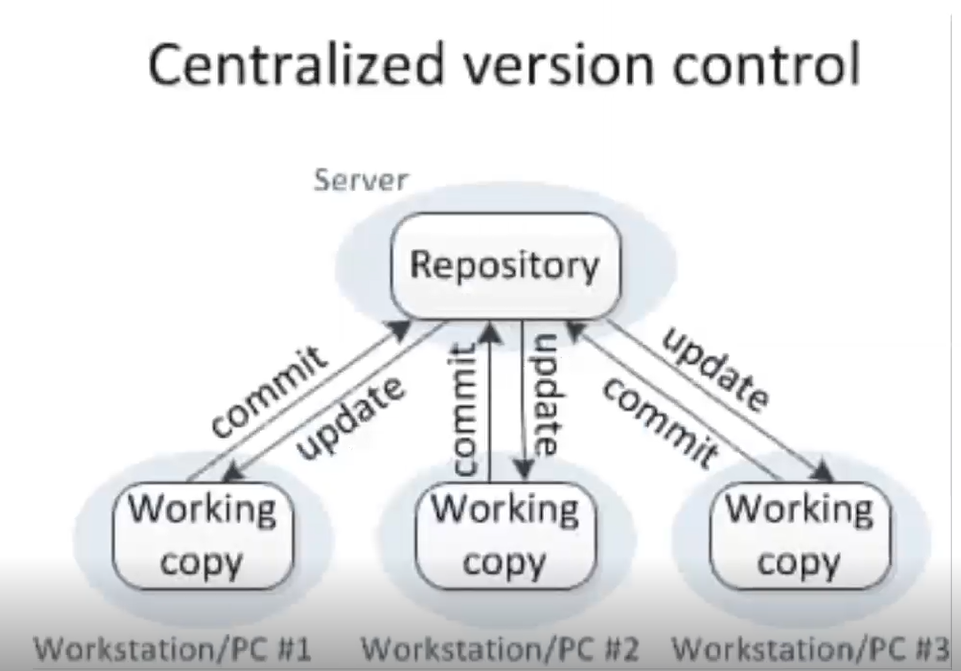
**GIT- VCS/source code management (SCM)**

**GIT is Free. It was Created by Linus Torvalds in 2005 to develop Linux Kernel**

**GIT Mean Stupid Person. Because during that time It haven’t Working Properly.**

* **Local Version Control System**
* **CVCS- Centralized Version Control System**
* **Distributed Version Control System**

**CVCS- Centralized Version Control System**

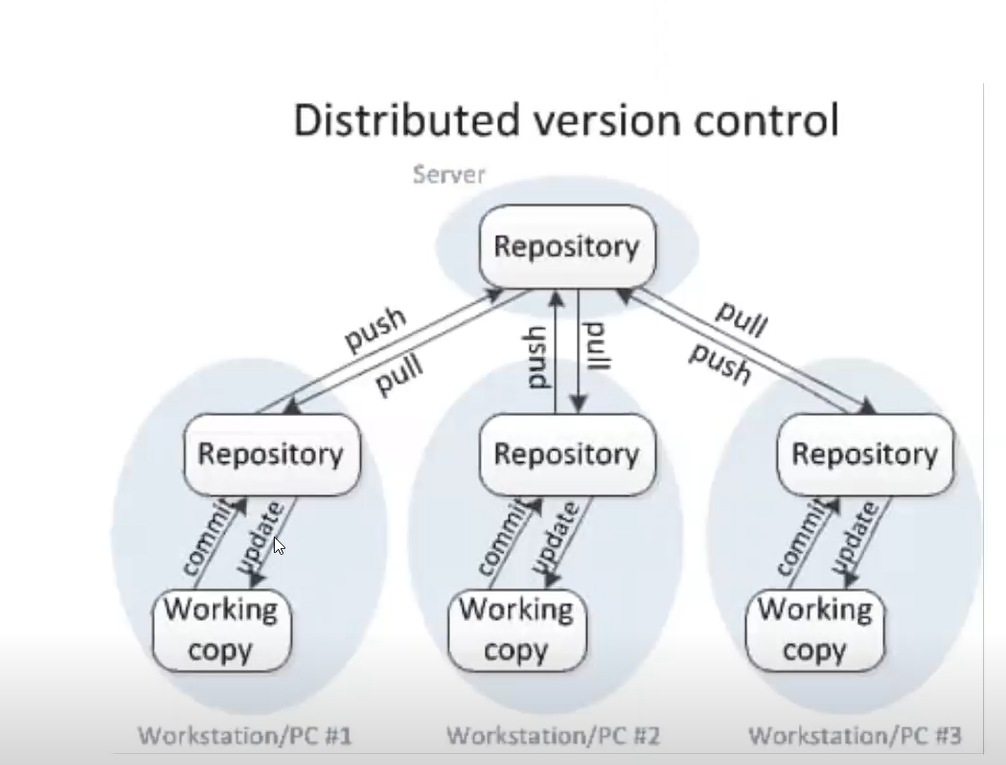
****

**After reaching Repository then application will develop.**

**Disadvantage of CVCS:**

* **Slow Latency**
* **Dependency of Continues Internet**

**Distributed Version Control System:**

****

**We can commit and update in Local Repository in our local laptop. Here there is no internet dependency.**

**Internet only using at one time after confirming the file in our local repository.**

**GIT- LR (Local Repository)**

**GIThub- RR (Remote Repository)**

**Install GIT**

**# yum install git**

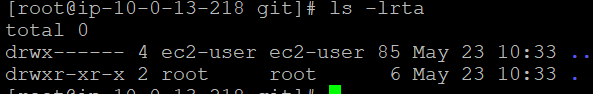
**# git –version**

**# mkdir git**

**# cd git**

**# ls -lrt**

**# ls -lrta ( hidden files)**

****

**# git init (initialize the local repository)**

**# ls -lrta**

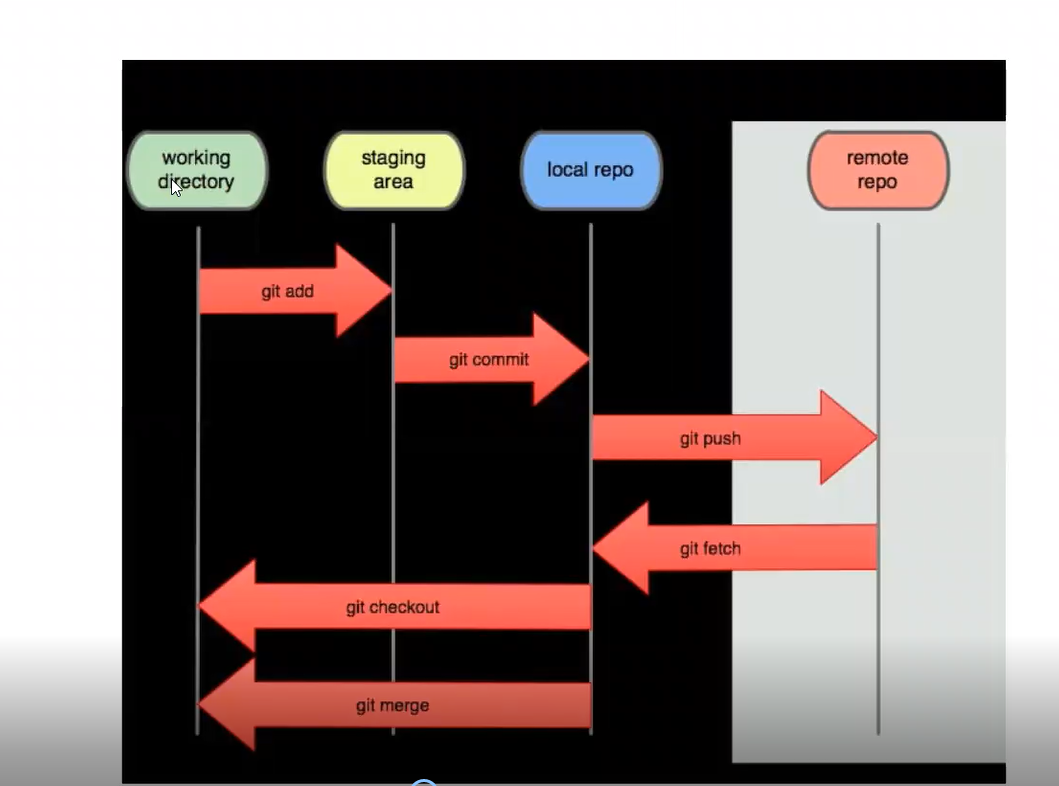
**Then it will show .git**

**#cd .git**

**# ls -lrt**

**/home/ec2-user/git/.git – this git local Repository. (LR)**

**/home/ec2-user/git – this is working directory (WD)**

****

**Working Directory to staging area (we can rollback)**

**Vi text.txt ----> first -----> wq**

**ls -lrt**

**cat vi test**

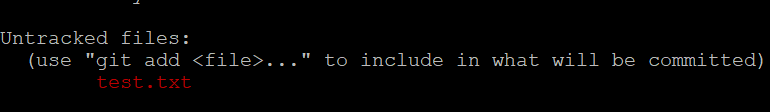
**now this file is there in working directory**

**ghp\_7jWB7GKlCjl3BCAqK3gpmHzcsMleFh1TdXKr**

**git status -s**

****

**git status**

****

**Now it is in untraced file.**

**How to Move Staging area:**

**# git add . or git add <filename>**

**# git status -s**

**# git status**

**# git config --global user.name “arunkeerthi”**

**# git config –global user.email** [**arunsivan31@gmail.com**](mailto:arunsivan31@gmail.com)

**# git log**

**It is showing some error which is not commit to master.**

**# git commit -m “description”**

**# git log  
   
 Now it showing which has committed with proper details.**

**# git status**

**Once we commit everything the status will show “nothing to commit, working tree clear”**

**Then we do second Commit for the same text file.**

**# vi text.txt ------> second ------> wq**

**#git status -s**

**(It shows red color M) . (Why it will be shown like we modify same file so that)**

**#git add .**

**#git status -s**

**(It shows Green Color M)**

**# git commit -m “second commit”**

**#git status -s**

**# git log**

**( now it will be shown two logs)**

**# git log --oneline (for short)**

**Git Hub-------**

**Create repository in GitHub.**

**Name should be created unique in your GitHub.**

**#git push  
  
 ( now it showing some error commit URL)**

**#git remote add origin URL (name origin (customized name) we give variable for that URL (git hub repro URL))**

**# git remote add origin**

**# git branch**

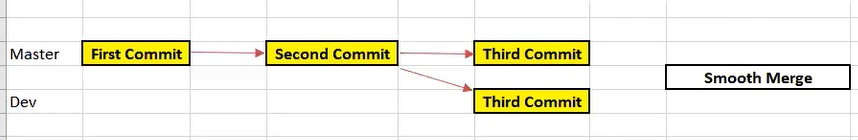
**# git push origin master (master - branch name)**

1. **Now it requires username and password**
2. **We need to give the credential**
3. **If there asking token, we need to go git hub**
4. **GitHub🡪setting🡪developer setting🡪personal access tokens🡪note (customized name) and give it full access🡪generate token**
5. **And we take the token number and past it in the password which has come in GitHub.**
6. **Then go and check the git hub that file will show in git hub repository.**

**#git log   
 it will be shown push details too.**

**BRANCHES**

**SCENARIO-1 SMOOTH MERGE**

****

* **Master it is nothing but original copy**
* **But real scenario we will not commit in master branch**
* **Real time mostly we use the feature branch only**
* **If it everything done by “Maker and Checker”, then only we will merge to the master branch.**
* **Feature branch nothing but we may mean “backup”**
* **Master Branch only access senior person or admin person**

**#git branch (to view the branch details)**

**#git branch dev (To create the new feature Branch, dev-customized name)**

**#git branch**

**#git checkout dev (switch to the other branch)**

**#git branch**

**#git log**

**Now we are dev branch**

**#vi text.txt🡪third🡪wq**

**#git commit -am “third Commit”**

**#git log or git log --oneline**

**Now it is showing third commit in dev**

**Now we go to master branch and check the log what will there?**

**#git branch**

**#git checkout master**

**#git branch**

**(Then here It will show two commits only)**

**Now we are going to do merge feature to master branch**

**Before merging we need to check in which branch are we there. Because we need to do merge at “master branch”**

**#git branch (check which branch are we there?)**

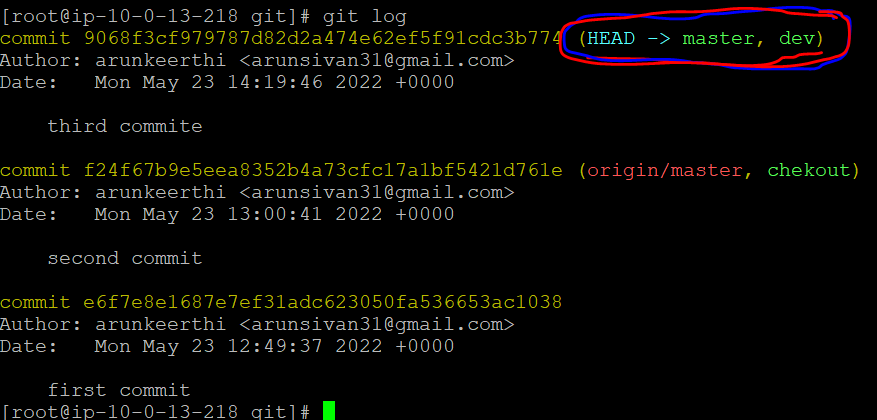
**#git merge dev**

**And view the file we able to see which done in Feature branch**

**#cat test.txt**

**#git log**

**The above-mentioned steps are smooth merging.**

****

**New push the central repository**

**#git push origin master**

**--------------------------------------------------------------------------------------------------------------------------------------**

**SCENARIO-2 CONFLICT MERGE:**

**# mkdir merge**

**# cd merge**

**# ls -lrta**

**# git init**

**#vi merge.txt🡪 first🡪wq**

**Note: untracked file we can’t directly commit first time (for this try it (git commit -am “first commit”))-error will occur**

**First time we need to add the file working directory 🡪 staging area**

**#git add .**

**#git commit -m “first commit”**

**Then do second commit**

**#vi merge.txt🡪 second 🡪 wq**

**#git commit -am “second commit”**

**#git log**

**Now Create Branch**

**#git branch dev**

**#git log ( check the log status)---->here we stand in master only**

**#git branch**

**#git checkout branch**

**#git log -------> (check log it shows both dev and master are same status)**

**Then do third commit**

**#vi merge.txt🡪 third 🡪 wq**

**#git commit -am “third commit”**

**#git log ------> ( here log status show different)**

**Now some other developer come, and he has directly accessed the master and do some changes.**

****

**#git checkout master**

**#vi merge.txt🡪3rd line 🡪wq**

**#git commit -am “3rd commit”**

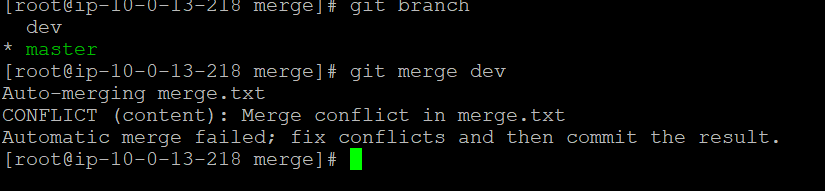
**#git log**

**Now senior developer come and check the details.**

**#git branch**

**#git merge dev**

**It is showing conflict error**

****

**#git log (and check what happen) ---( it showing 3rd commit)that’s why conflict happen**

**# cat dev (file status)**

**So finally, conflict happen between the master and feature branch**

**So that we need to solve the conflict problem**

**Two options is there to resolve the issue----->Example**

**1.we can modify the test file but real time we can’t do because large numbers of coding will be there)**

**2.Install merge conflict to arrest the issue**

**For this case we need to install the merge conflict tool in git.**

**#git config --global merge.tool vimdiff (vimdiff name)**

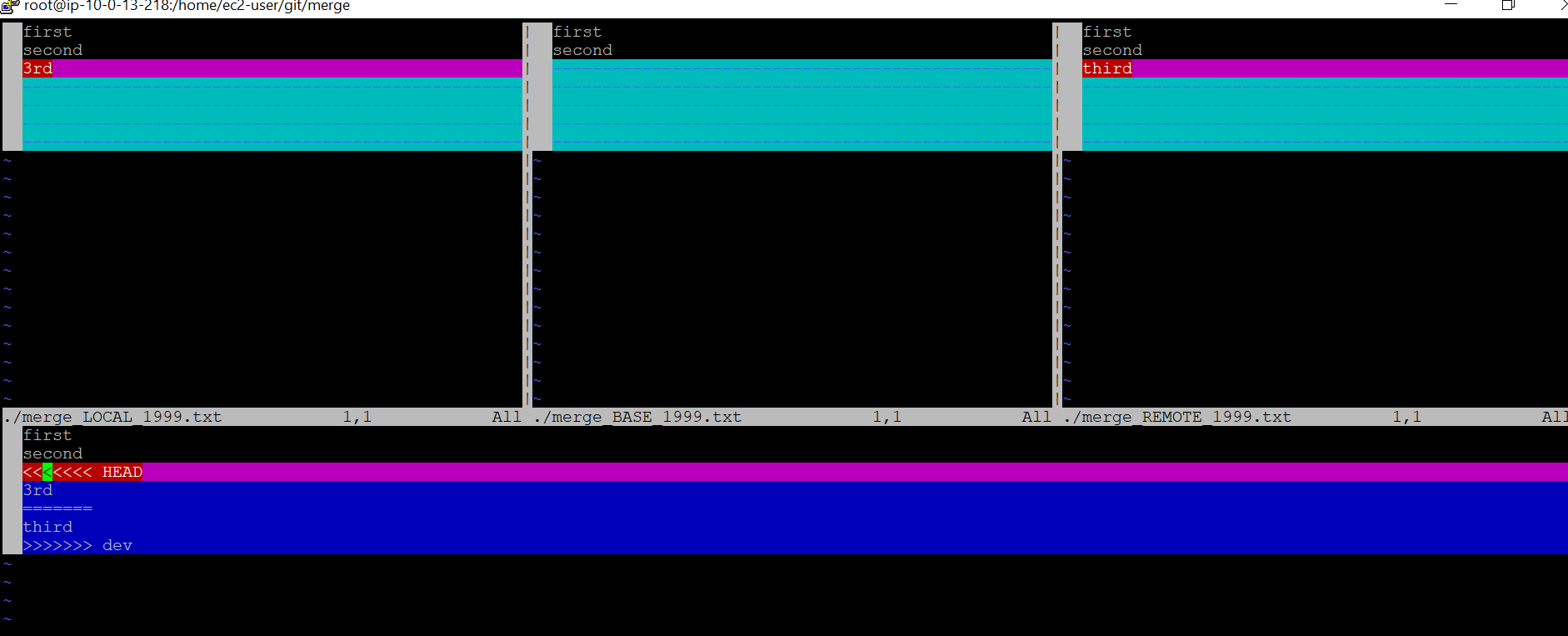
**#git config --global merge.conflictstyle diff3 (diff3 style it show three differentiation)**

**#git config --global mergetool.prompt false (prompting falser to avoid yes or no question)**

**#git mergetool**

**(Here merge tool will do auto analysis for both branches)**

**It is showing like below mentioned tool**

****

**Ctrl+ww the Crouser will move then we may edit which ever we want.**

**Here it is showing the three differentiations**

**./merge\_LOCAL\_1999.txt 🡪master, ./merge\_BASE\_1999.txt🡪base ./merge\_REMOTE\_1999.txt🡪branch**

**By using this below commands we may confirm.**

**:diffg LO enter then :wqa**

**:diffg RE enter then :wqa**

**:diffg BA enter button then :wqa**

**ctrl dd**

**:wqa!**

**runs from master : git merge feature\_branch**

**choose any commands and resolve the merge conflict**

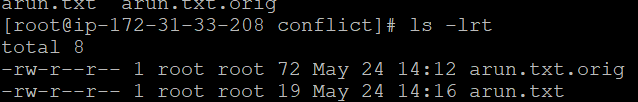
**before given the command we need to press esc button around 10times**

**# :diff RE enter then :wqa! (example)**

**#cat dev.txt (Now it showing which we changed in merge conflict)**

**ls -lrt**

**one more file also will come that is org file.**

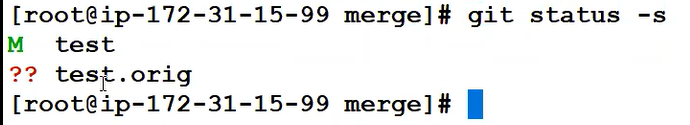
****

**#cat arun.txt.org**

**Here It is showing previous status which we did wrong**

**#git status ( it shows that two file status)**

**Example:--**

****

**#git commit -m “final”**

**#git status -s**

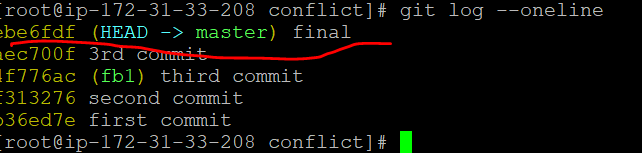
**If we want to delete that additional, we may delete**

**# rm -Rf <file name>**

**#git log –oneline ---> (Note: we stand in master only)**

**Now it is showing five commit details.**

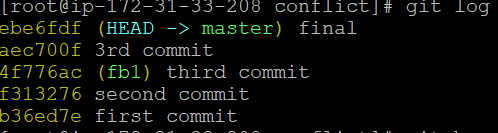
**When we do Merge Conflict, we will be getting additional Commit.**

****

**How to solve the Merge Conflict issue by using Rebase Command:**

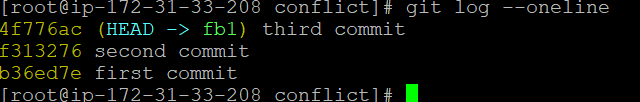
**#git checkout master**

**#git log**

****

**#git checkout fb1**

**#git log**

****

**Above Mentioned two branch logs it shows commits results are different**

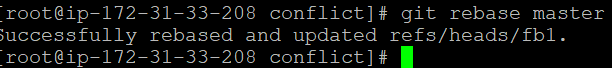
**So that we need do some changes and equalize both branches commit then only**

**We will do further in feature branch**

**By using rebase command we will rectify the issue**

**Rebase command we should use in feature branch.**

**# git rebase master**

****

**# git log**

**Then all commit will same as both branches**

**CHERRY**

**We can pick the particular commit by using Cheery Command from master to feature branch**

**# Mkdir cherry**

**# cd cherry**

**# git init**

**#vi text 🡪 1 🡪 wq**

**# git add .**

**# git commit -m “first commit”**

**#git log**

**# git branch (it show master branch)**

**# git branch dev (now Create the feature Branch)**

**# git log –oneline (Now commit will show in master as same as feature branch)**

**# vi cherry🡪 2 🡪 wq**

**#git commit -m “second commit”**

**#vi cherry🡪3🡪 wq  
#git commit -m “third commit”**

**#git log –oneline ----> ( In Master will show three commit)**

**#git checkout dev**

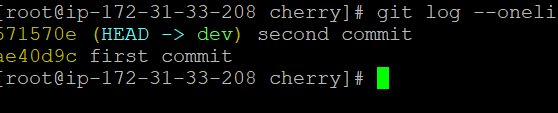
**#git log ---> ( here showing only one commit)**

**Then here will show one commit**

**#git cherry-pick <commit Id> this command we should do in feature branch.**

**Note : commite id will be change once we pick the master to feature branch**

**#git log --oneline**

****

**STASH**

**Roll back from staging area (buffer location) to working directory (re-apply is possible).**

**#mkdir stash**

**#vi stash.txt**

**#git add .**

**#git commit -m “first”**

**# git log –oneline**

**#vi stash .txt 🡪 just edit :wq**

**#git stash**

**#git stash list (then we get the statsh id)**

**Then again and go to edit some**

**#vi stash.txt🡪edit🡪wq**

**Then again go stash**

**#git stash**

**#git stash list**

**Here we will two stash details because we have applied two stash.**

**Move 🡪 pop**

**Copy 🡪 Apply**

**#git stash pop stashID**

**(now it will move to work directory)**

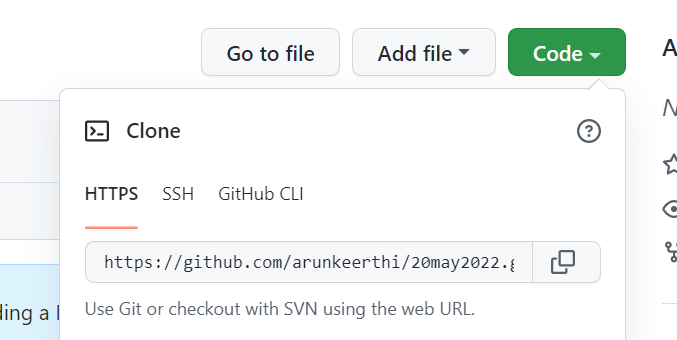
**#git stash list  
 (Now we can check in stash list , id which we use in pop that will be delete in stash list)**

**Clone:**

**Backup and recovery - we take the backup for our EC2 server.**

**Go to git hub and choose any of the project details in repository.**

**Code🡪copy url**

****

**Copy url and go to git wd**

**#git clone url**

**#ls**

**That will fill show in your working directory.**

**Pull:**

**Latest Source Code always available in Github repository**

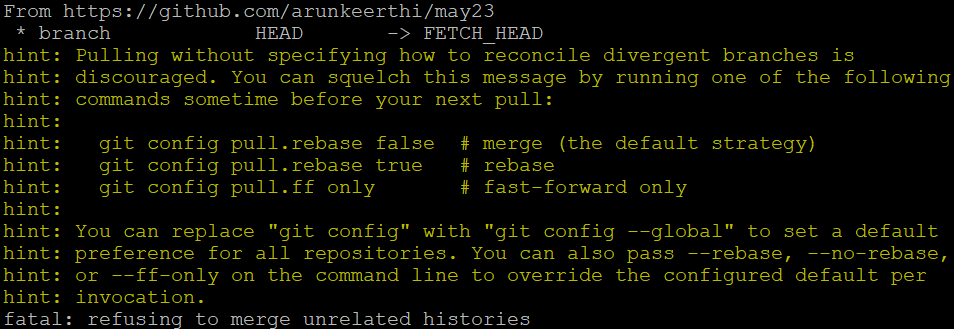
**So that the Purpose of enhance or update we pull the source Code from RR to LR**

**Pull = Fetch - (LR) + Merge - (WD) for the process technically we called us Check Out**

**(Push = LR + RR ----> Check in)**

**1.Go to working directory**

**#git pull url**

****

**If showing like this error above mentioned picture because already we have file**

**So that We need to pull forcefully**

**#git pull --allow-unrelated-histories url**

**#esc 🡪 : 🡪 wq!**

**#ls -lrt**

**FORK**

**Fork is used to transfer the details from git hub one account to another account.**

**By using git hub repository URL link to see the fork details one account to another account**

**git push -f url master**

**#git branch -d dev / force branch delete # git branch -D dev**

**#git diff ( diff between wd to stage area)**

**#git diff - -staged ( stage area to local repo)**

**#git show ( last commit details)**

**#git show <commit id> (particular commit details)**

**#git blame <file name> (we know about update of file command details)**

**#git branch -a ( hidden branch)**

**#git push origin - -delete dev ( remote branch delete)**

**#git rm - -cached file1**

**#** **git restore --staged <file>**

**ghp\_UYmProyHFktYb3412AnKeTdLT3s9p42DsQ1I**

**git stash**

**git stash list**

**git stash pop**

**git stash apply**

**git stah pop < >**

**git stash apply < >**

**git stash drop**

**git stash clear**

