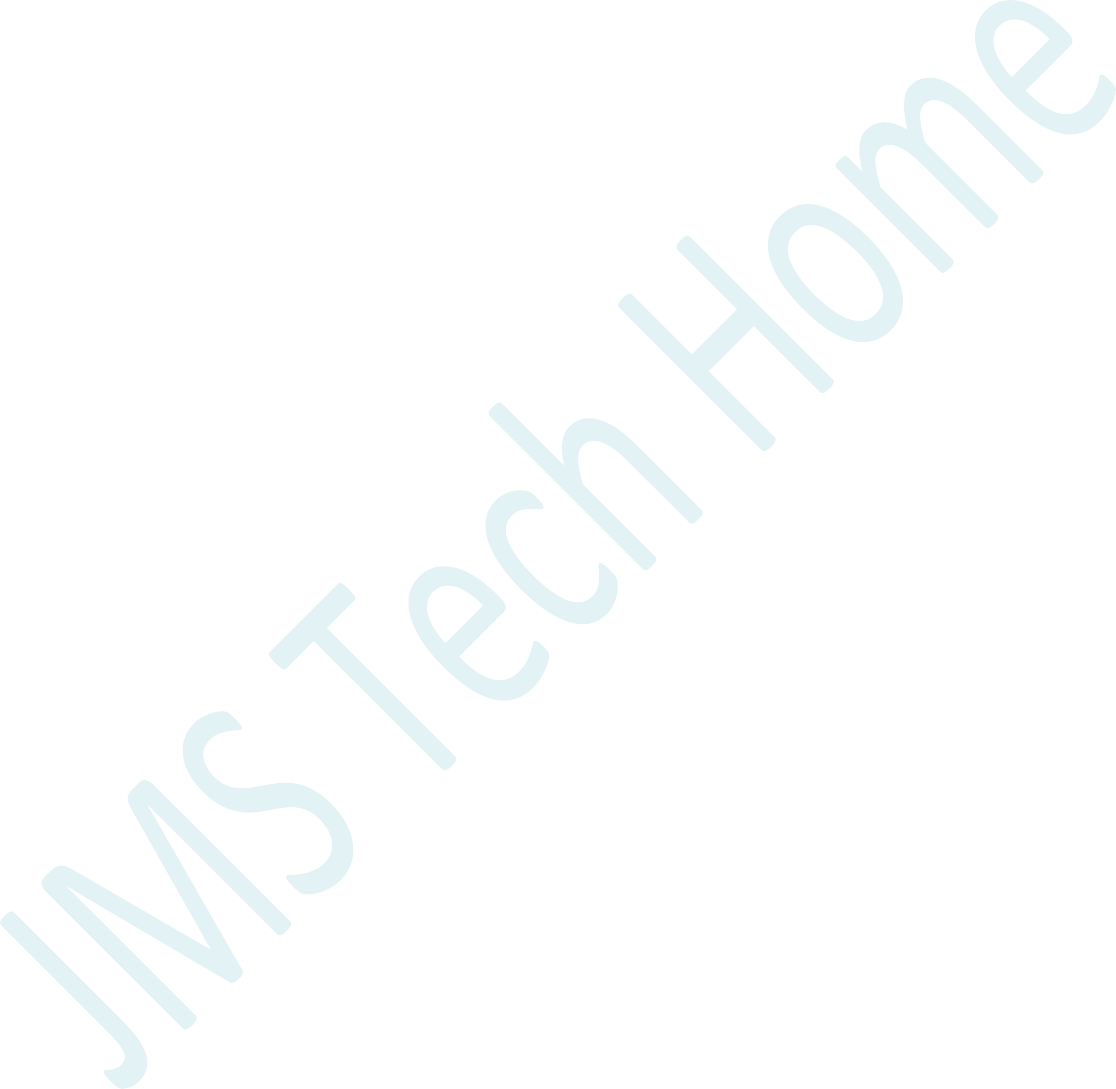
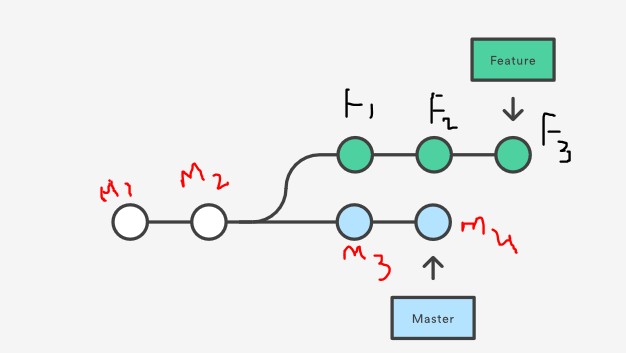
**Git Merge and Rebase**

**Senario:**

You are working on a new feature in your project? Chances are that you created a separate branch for that purpose. But what if you want to add these changes to the master branch? Then MERGE or REBASE can be helpful, so let’s have a look at these commands!

**git-merge** - Join two or more development histories together. for ex:



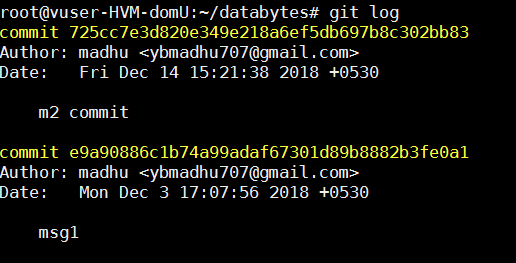
The master branch having 4 commits m1,m2,m3,m4 and feature branch having m1,m2,f1,f2,f3

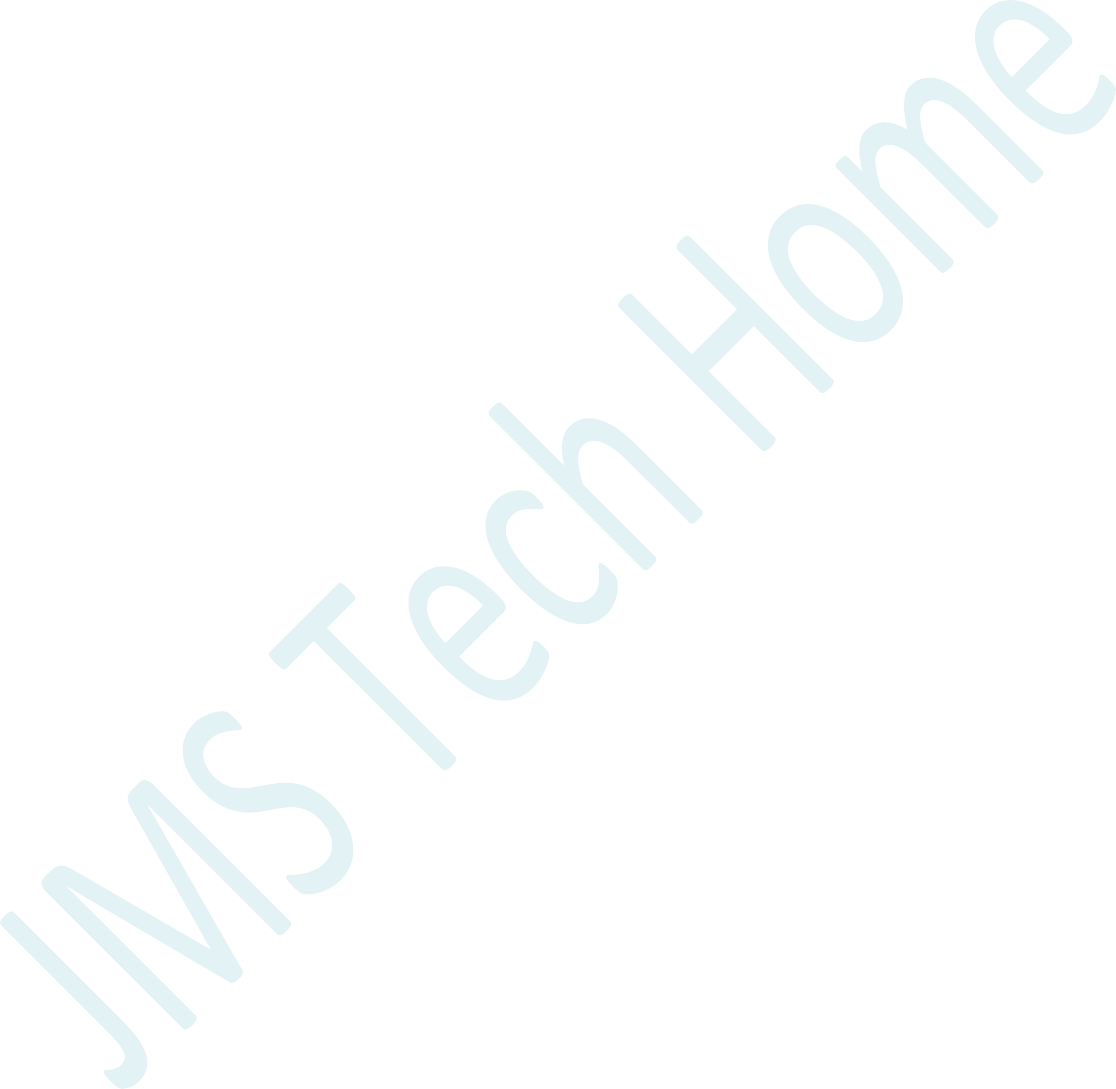
now i want to combined the changes of both into one place . In this scenario we will go for merge or rebase commands. let's see the what the difference between these two commands and usages.

**Merge scenario:**

In this scenario

1. first we have master branch and master branch having two commits i.e.m1,m2.

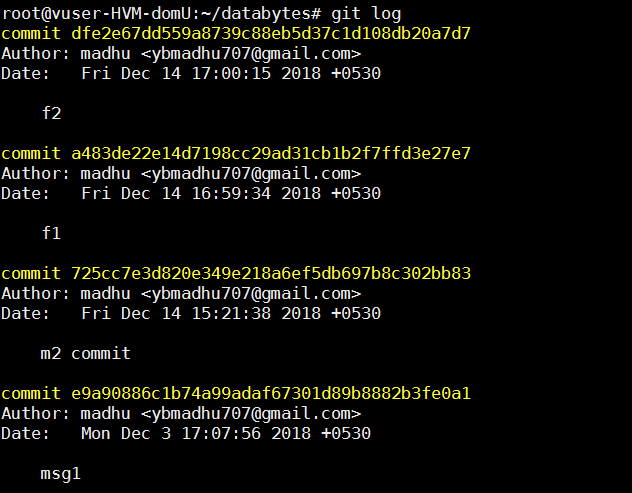


1. we got a new work and we created for this new branch called **feature**. this feature branch having data up to m1,m2 commits.

$ git fetch

$git checkout feature

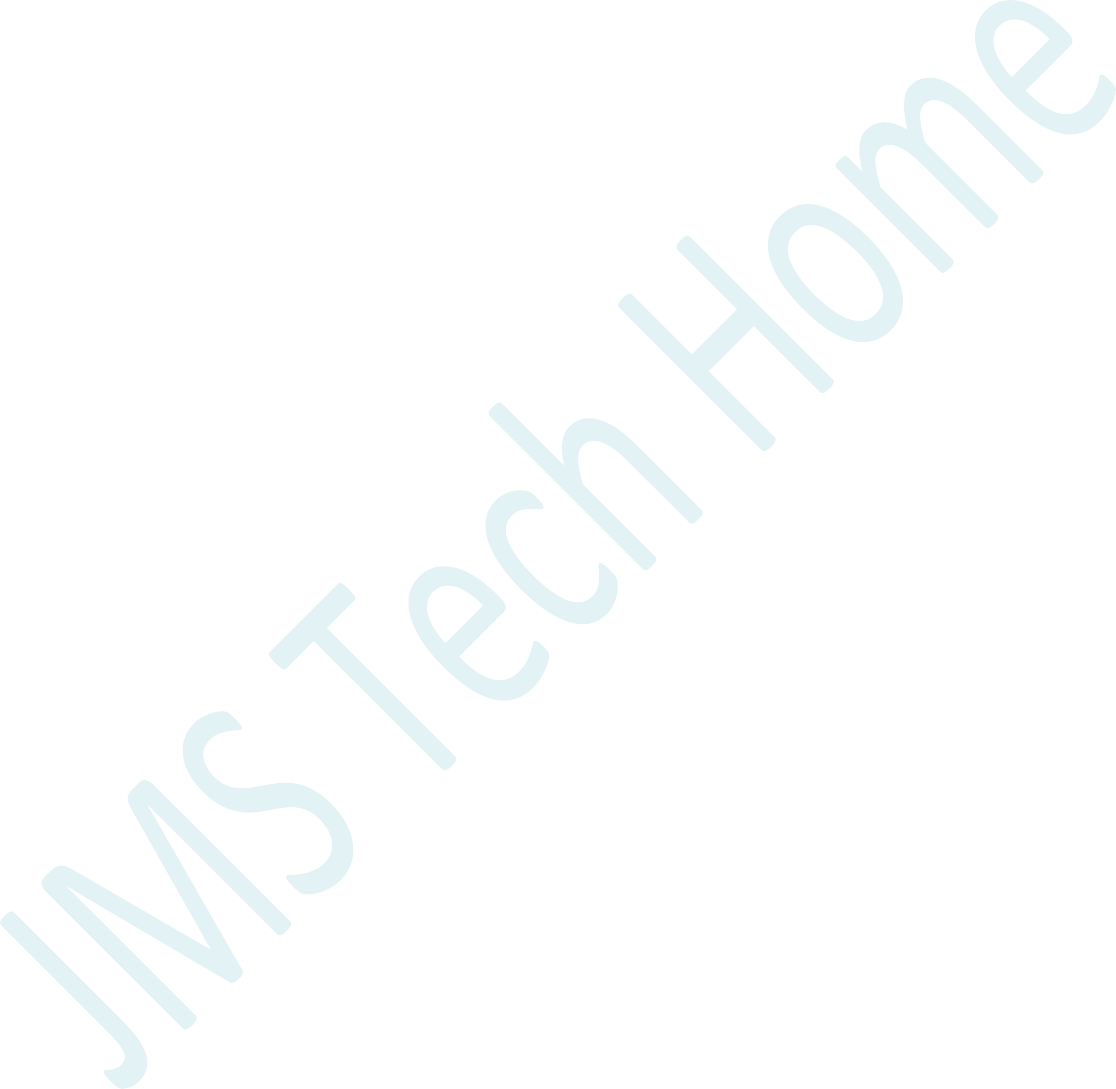
1. Now i implemented some code in feature branch and did two commits in feature branch called f1 and f2. Now feature branch having four commit logs i.e. m1,m2,f1,f2.



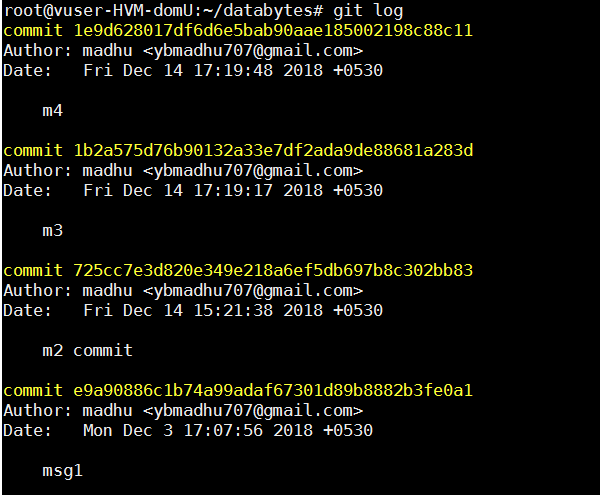
1. now i want to merge my feature branch to master branch. i am using merge command for this scenario.

Note : mean while some other worked on the master branch and they committed code with 2 commits called m3,m4.

$git checkout master

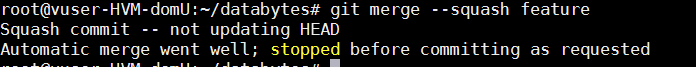
$git log

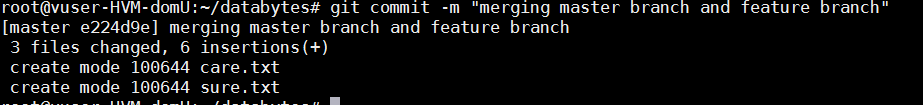
this master branch logs

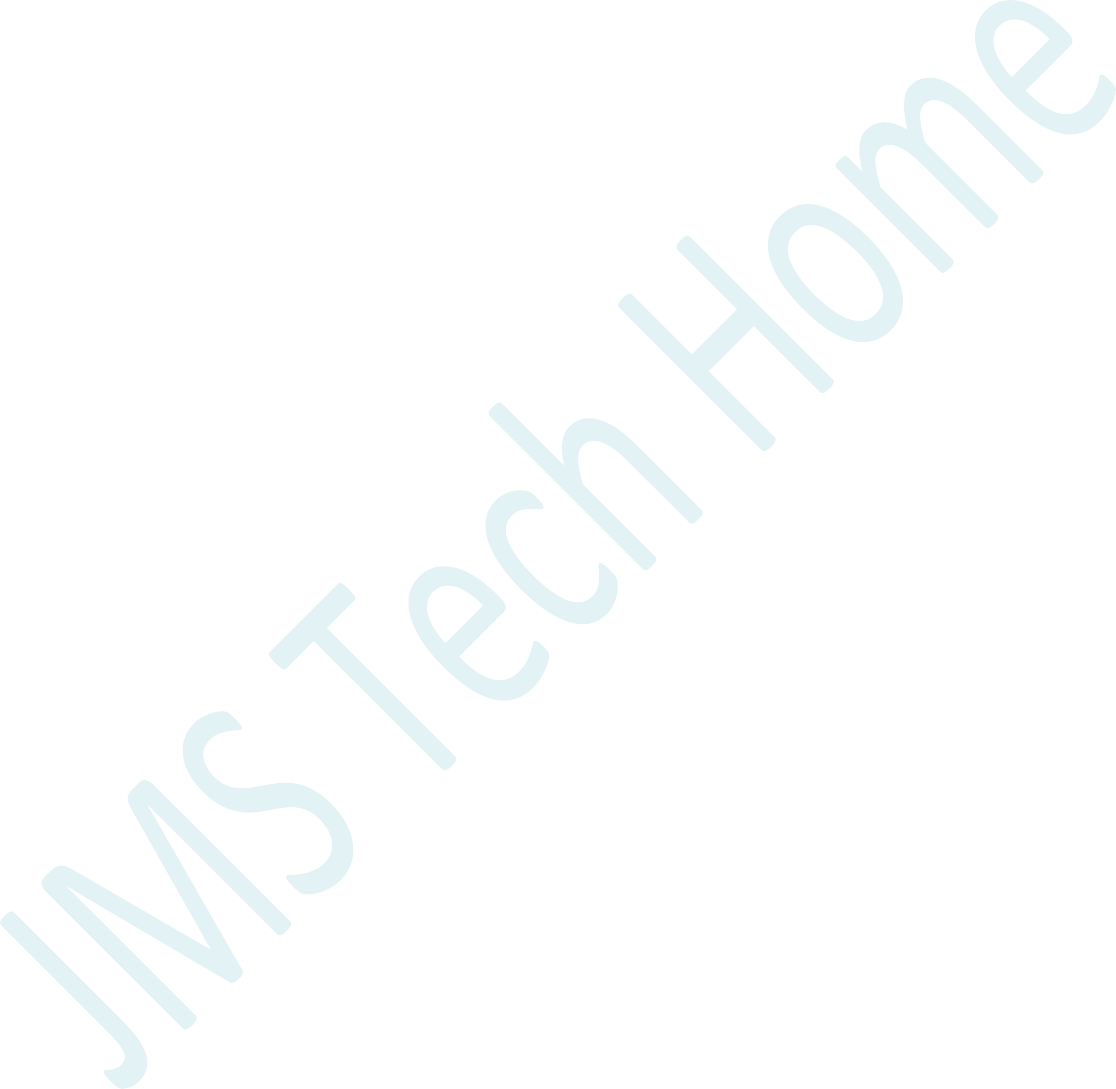


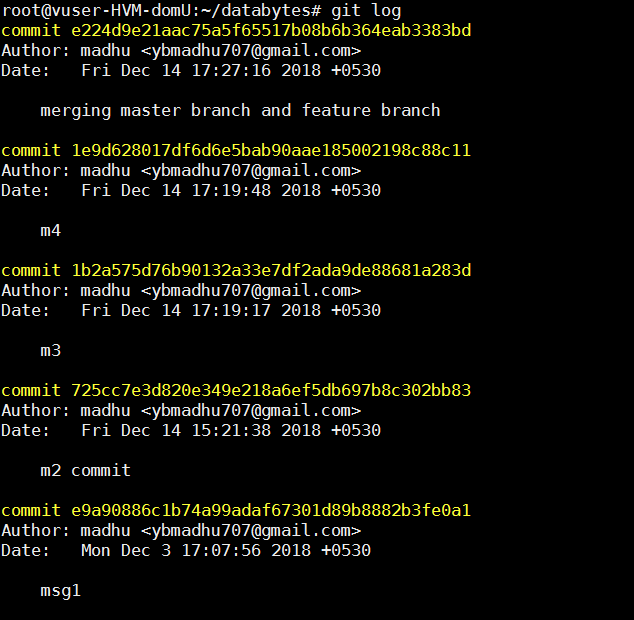
1. here i am using --squash option for auto merge to avoid merge conflicts

$ git merge --squash feature

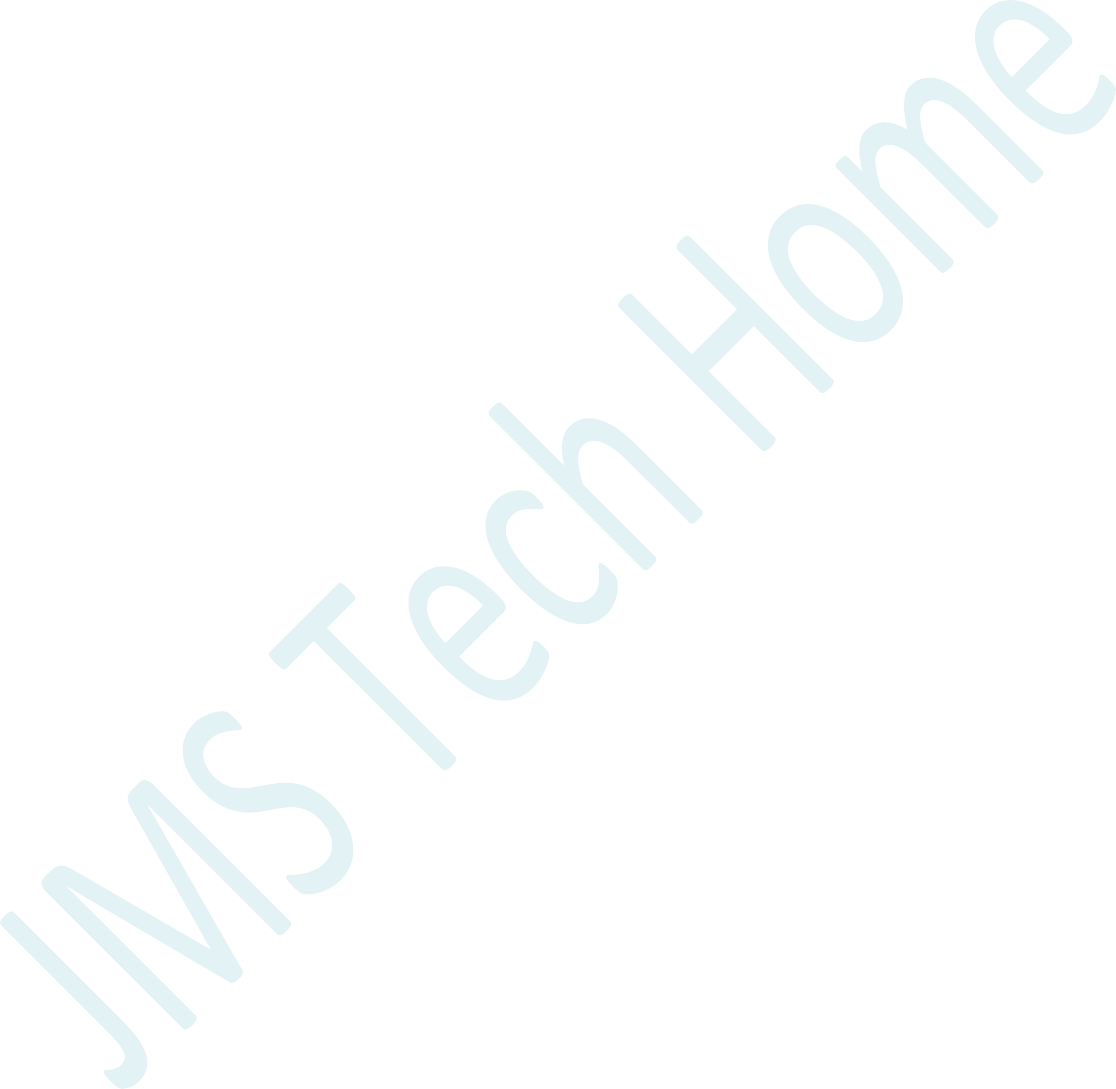


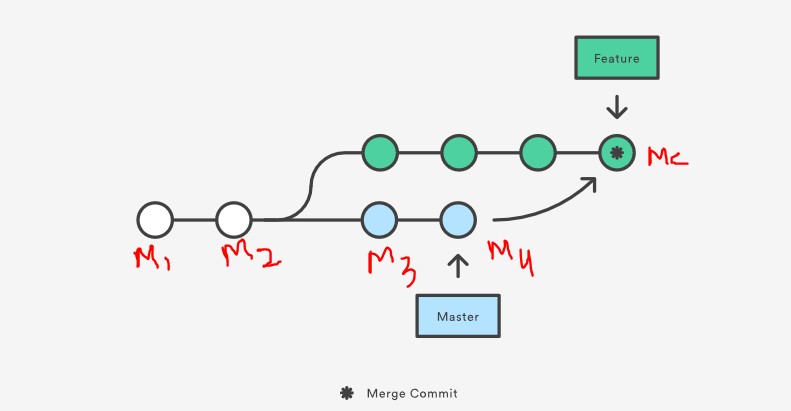


after see the project structure you will get the all files from feature branch. let's see the logs from master branch.



# Note : we are successfully merge the feature branch to master. here we miss the history commit messages of feature branch. if you want to history of feature branch committed messages we should use rebase.

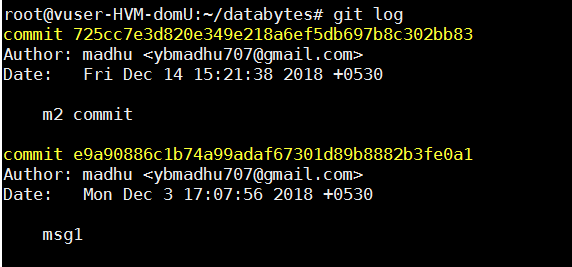
After merge out structure is look likes below diagram. \*mc is merge commit

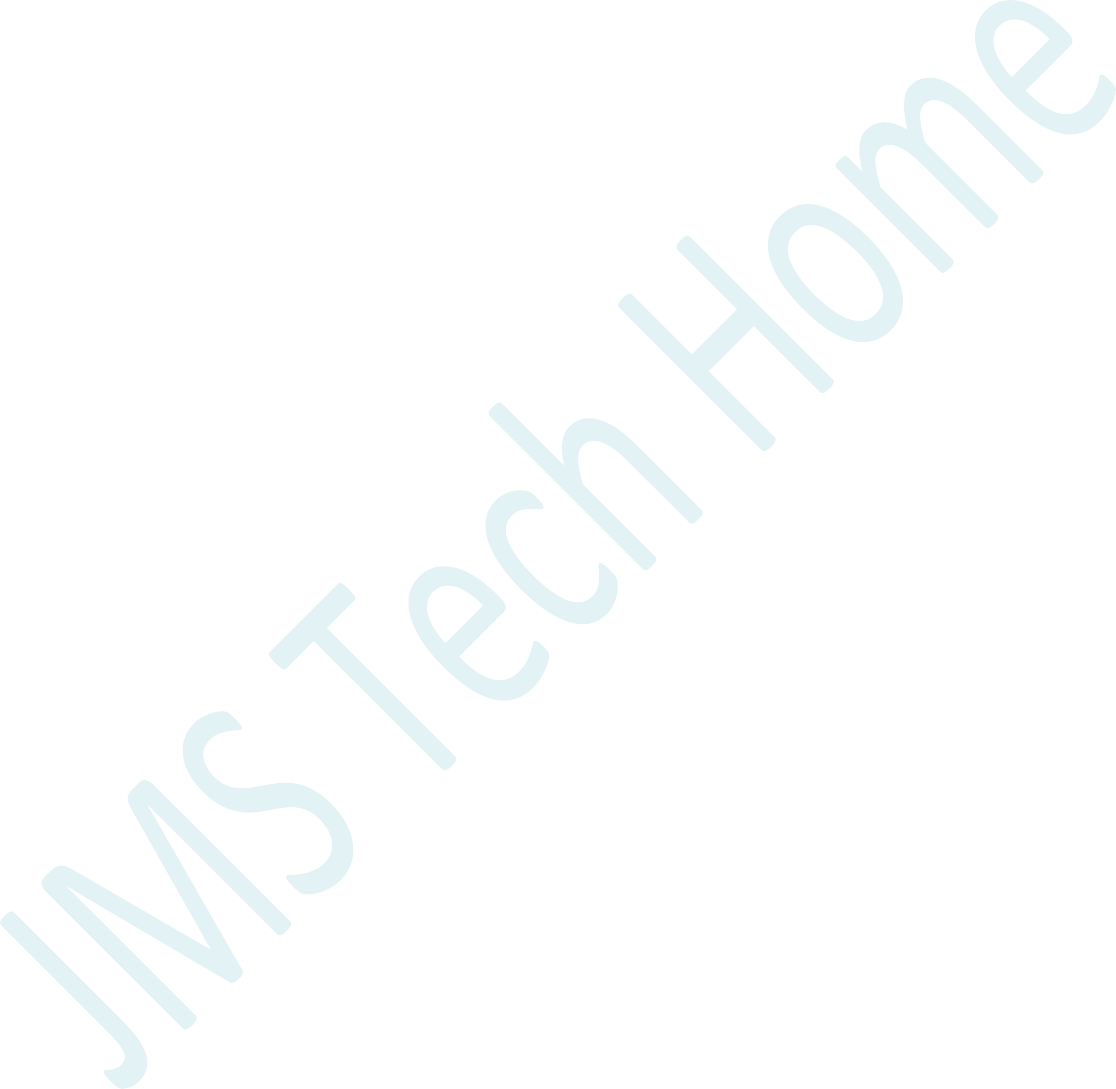


Note : this is new scenario using rebase how we can merge. for this scenario i am going to back how the previous project look like same.

$ git reset --hard HEAD~1 ---> i am deleting latest commits (last 3 commits i deleted)

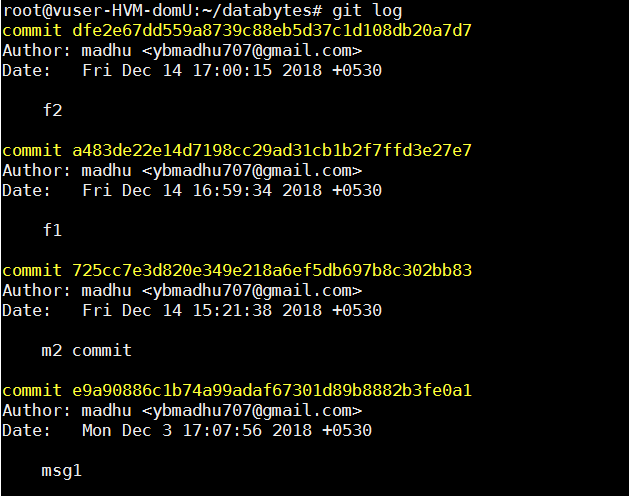
Now see the logs at master branch. Now we can see the only m1,m2 committed messages only.



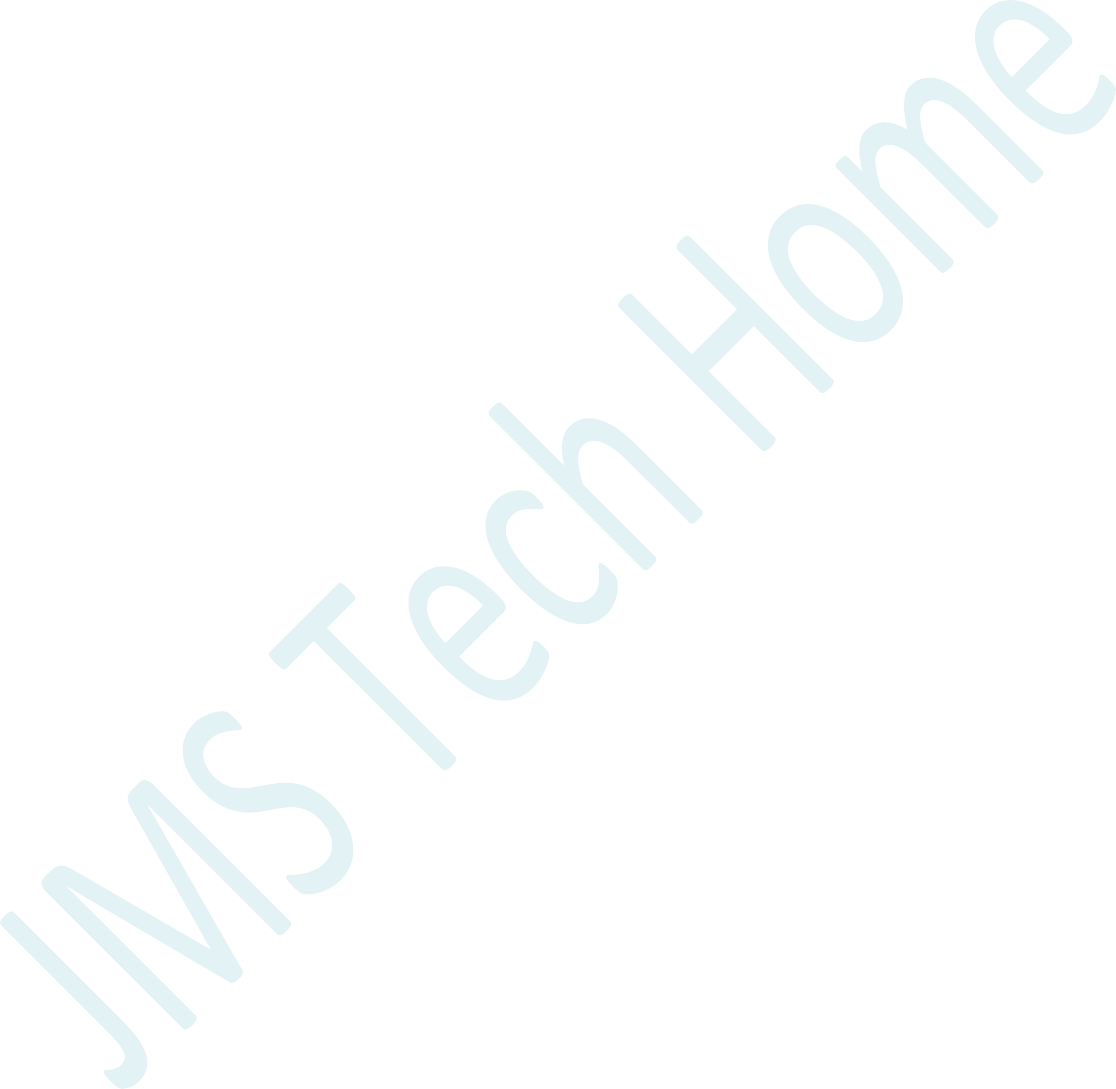
1. I am switching to feature branch.

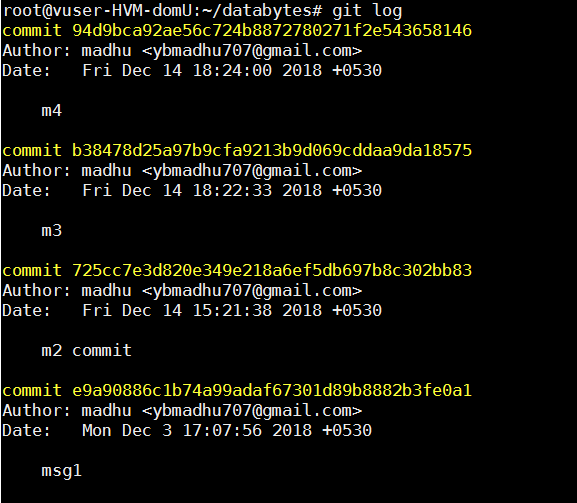
$git checkout feature

$git log



1. mean while some other persons are push to master code with m3,m4 commits..

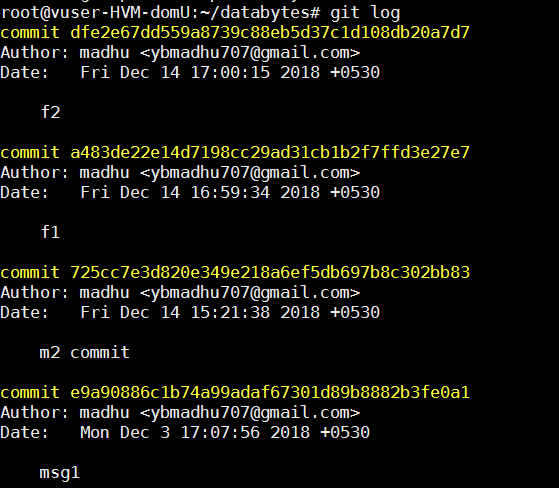
see the logs

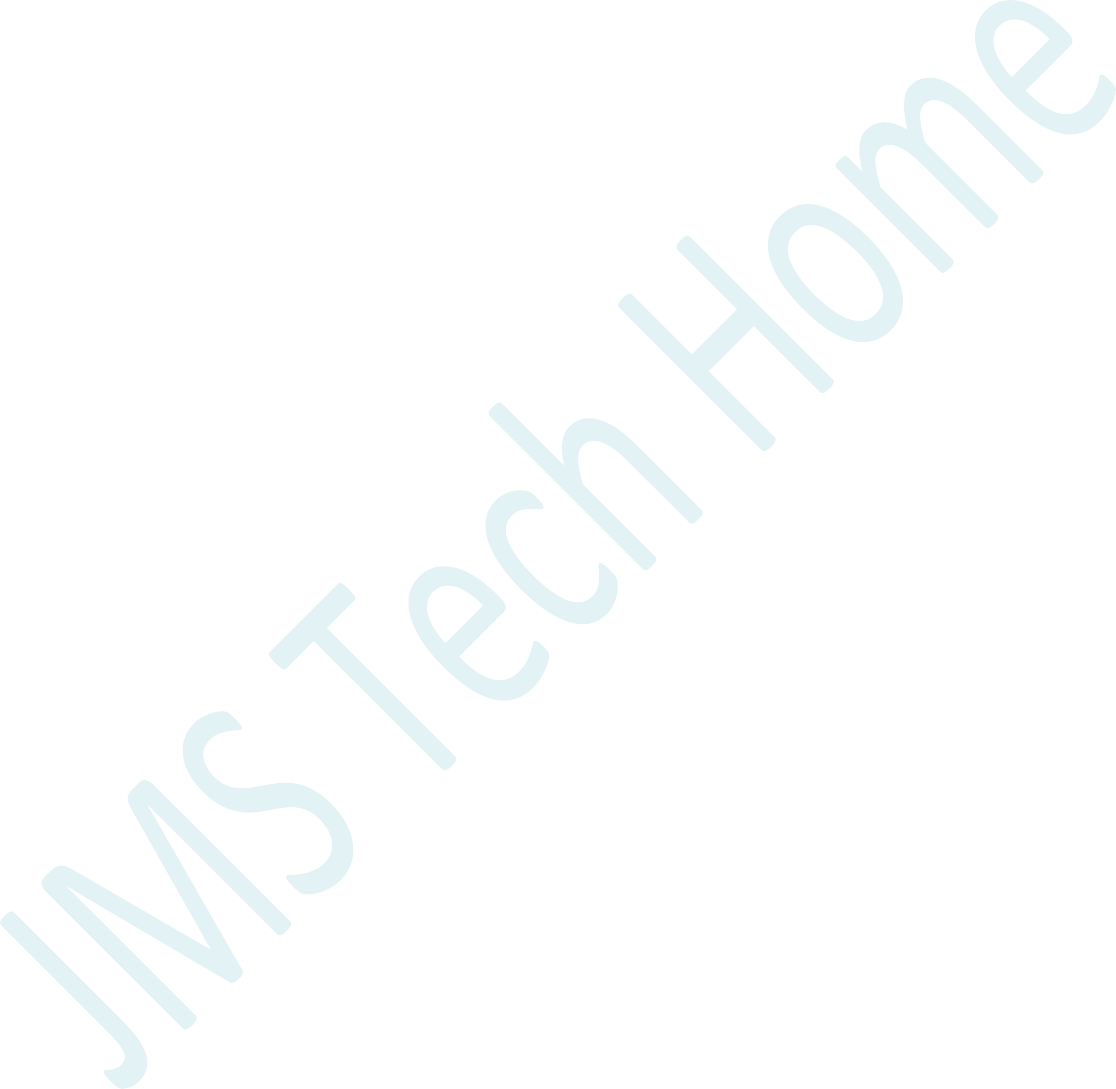


7. now think you're in feature branch if not switch the feature branch.

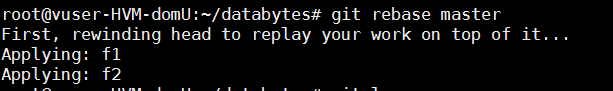
$git checkout feature

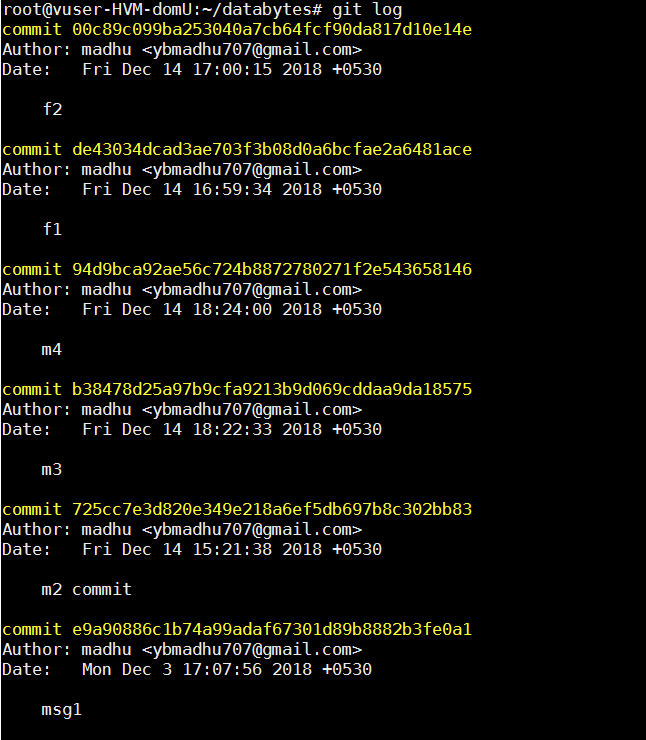
$ git log

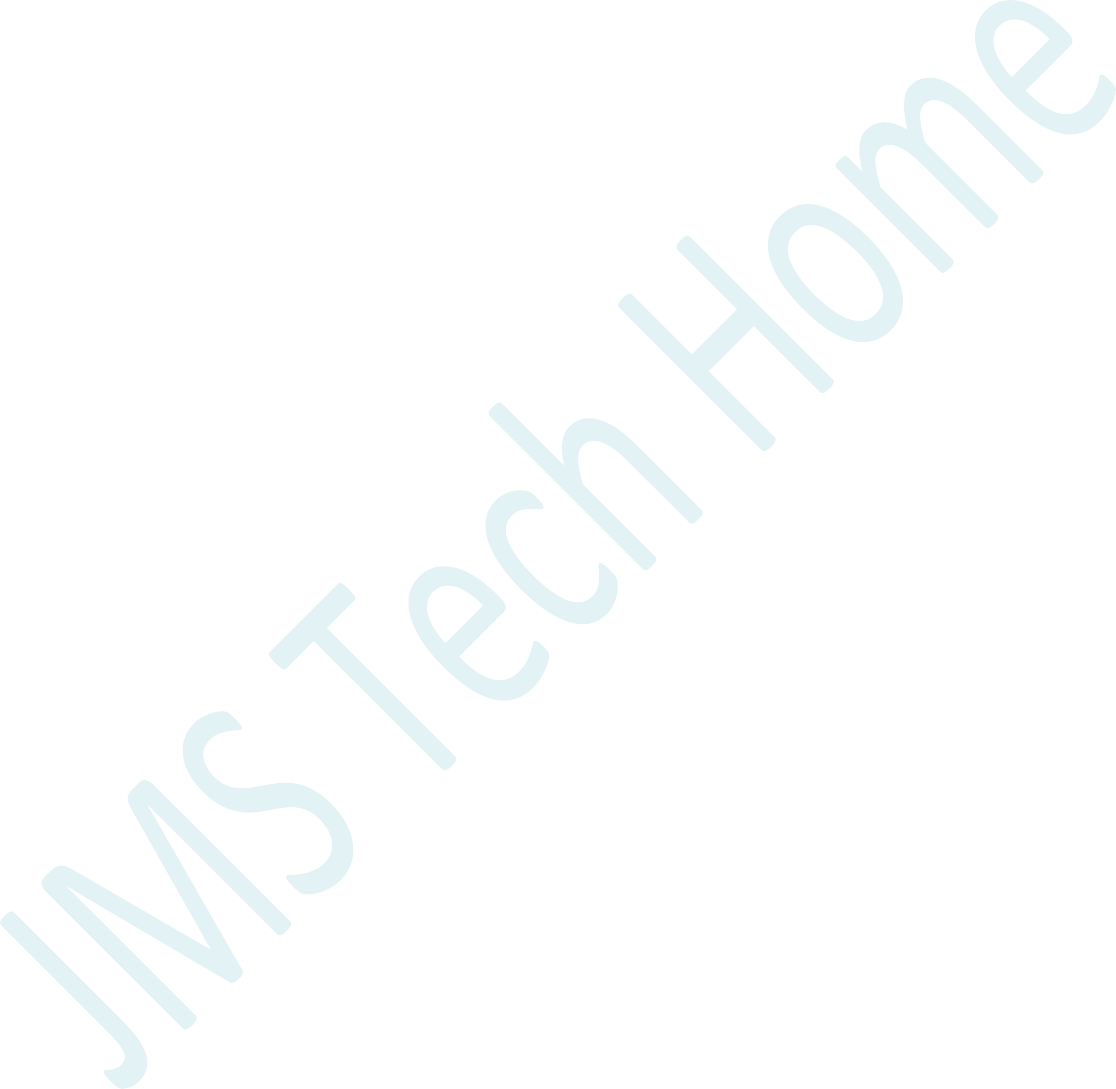


Now do the git rebase on feature branch.

$ git rebase master lets see the logs.



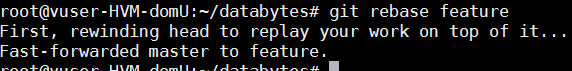


Now checkout to master branch..

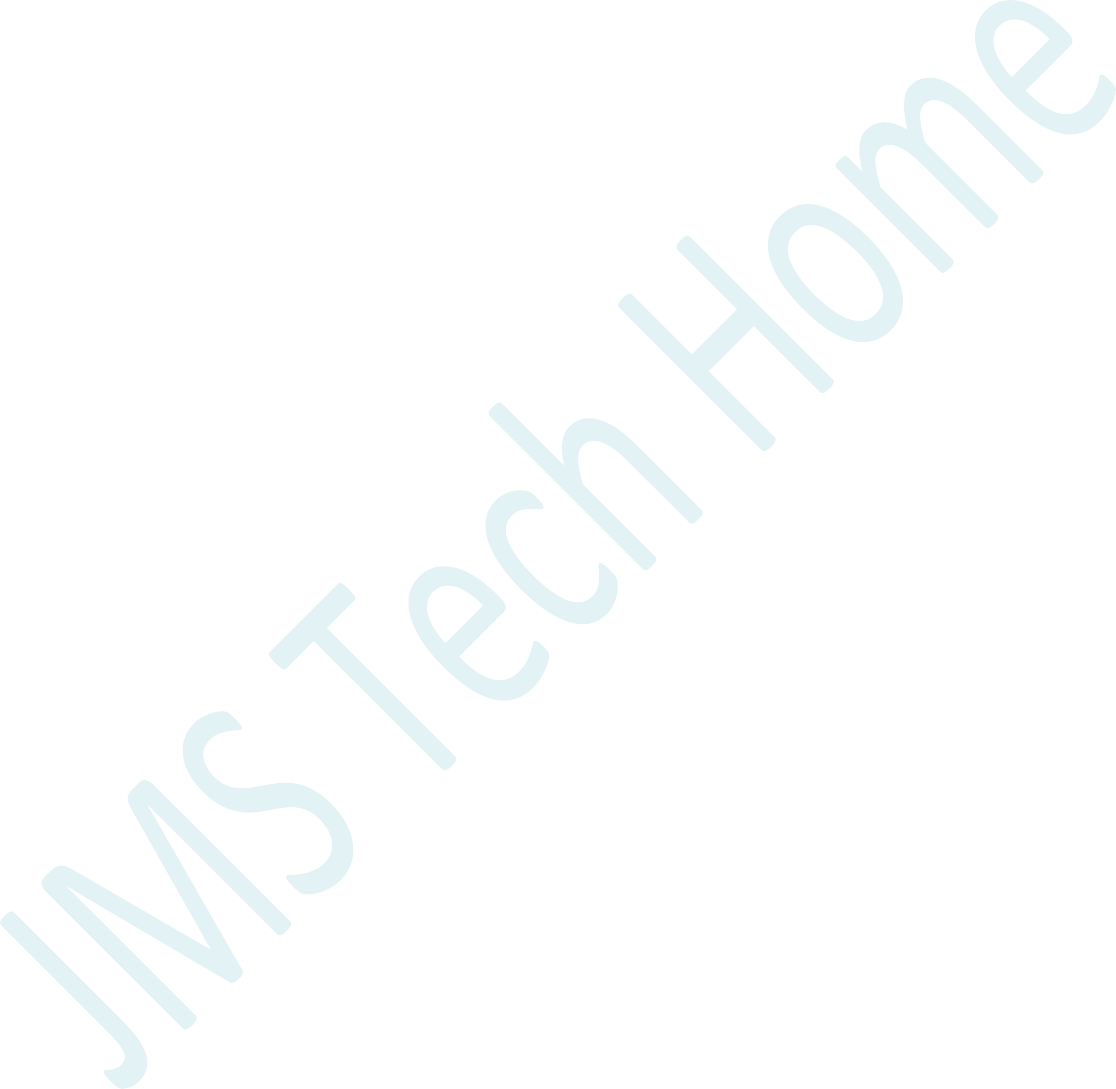
$git checkout master

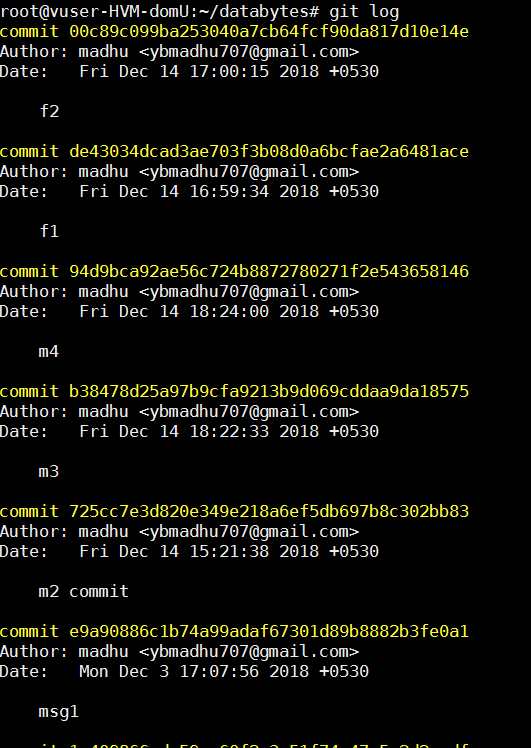
now see the logs at master branch we have only m1,m2,m3,m4 commits. then run the rebase command at master branch

$git rebase feature



see the logs now this time we should see the commits in log m1,m2,m3,m4,f1,f2

$git log



Now we can able to see the history feature branch commits as well...