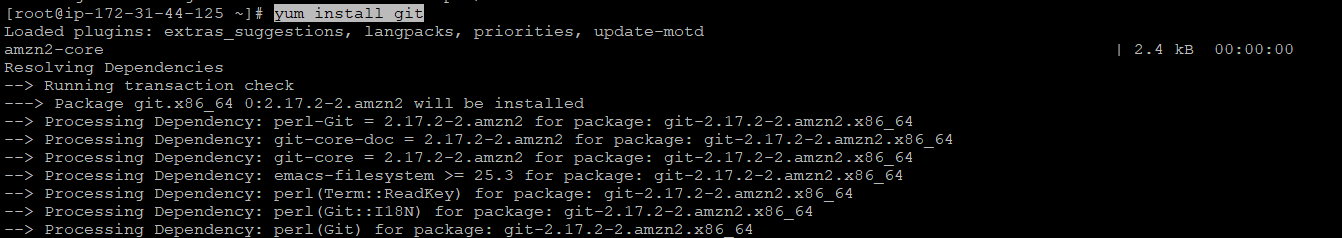
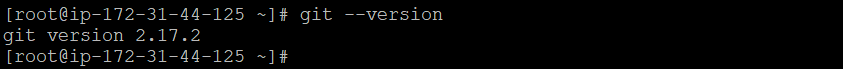
**GIT Commands**

**GIT Installation :**

Command : **yum install git**



To check the version installed : **git --version**



**Initial Configuration :**

**git config --global user.name "MuthuSusithra"**

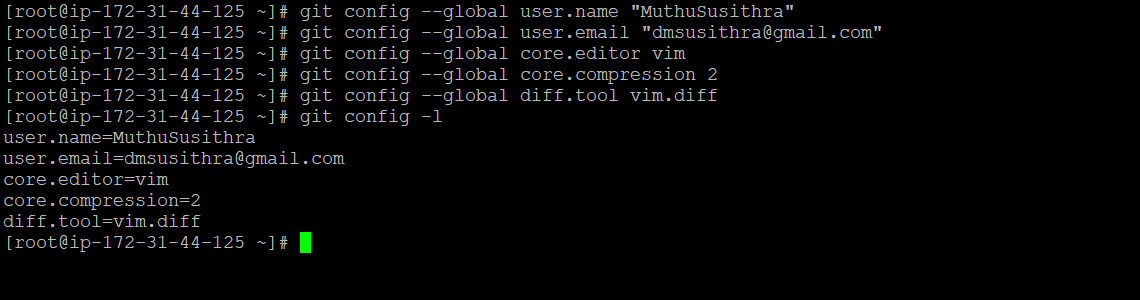
**git config --global user.email "dmsusithra@gmail.com"**

**git config --global core.editor vim**

**git config --global core.compression 2**

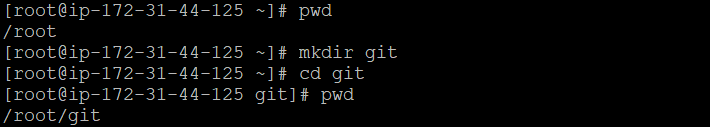
**git config --global diff.tool vim.diff**

To list : **git config -l**

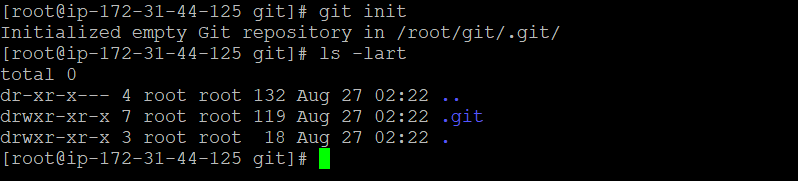


**Creating a new local repository :**

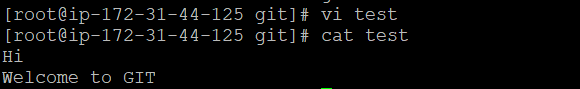
* Create a directory to contain the project.
* Go into the new directory.



* Type **git init -** to create a local repo



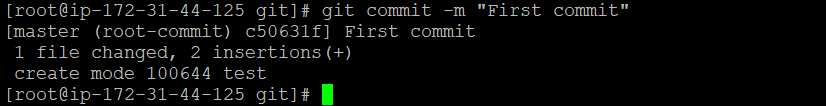
* Write some code.

­­­­­­­­­­­

* Type **git add <file\_name> -** to add from working to staging area



* Type **git commit -m "Commit message" –** to add from staging to local repo

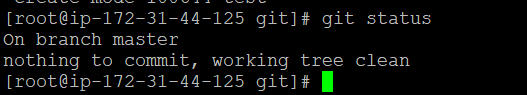


**git commit –a -m "Commit message" –** to add directly from working to local repo

**To check status :**

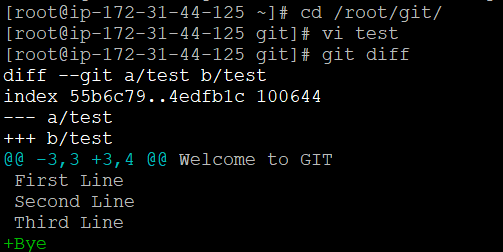
**git status**

**git status -s**

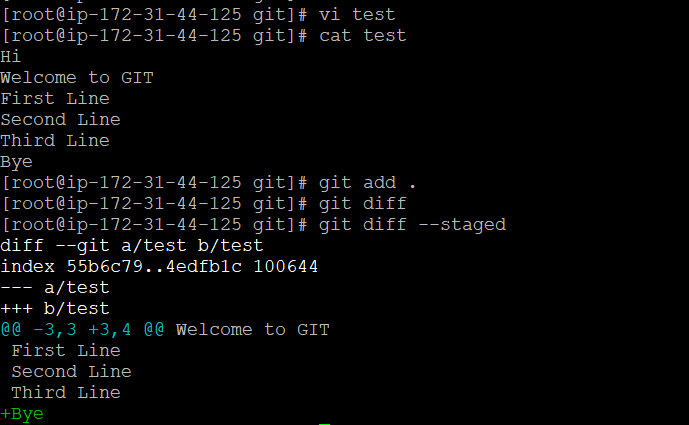


**DIFF :**

Command : **git diff** – to view the difference before moving to staging area



**git diff --staged** – to view the difference once moved to staging area

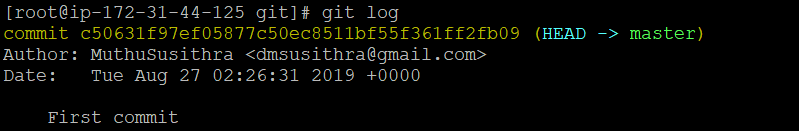


**View Commit Log :**

**git log**

**git log --oneline**

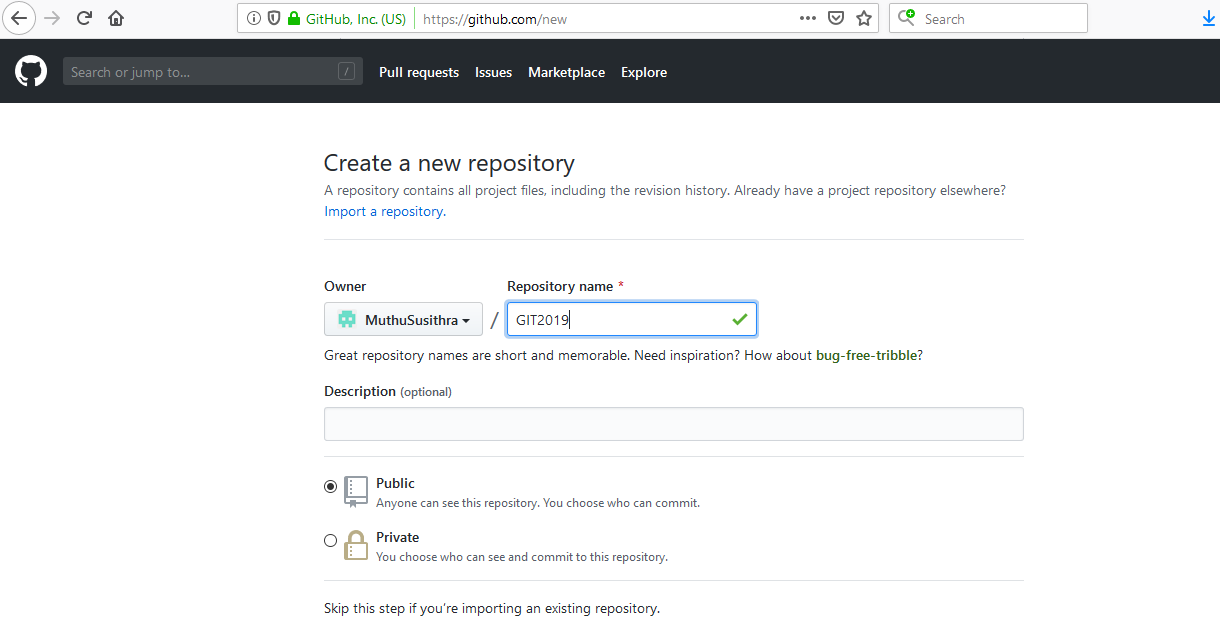
**git show <commit\_id>**





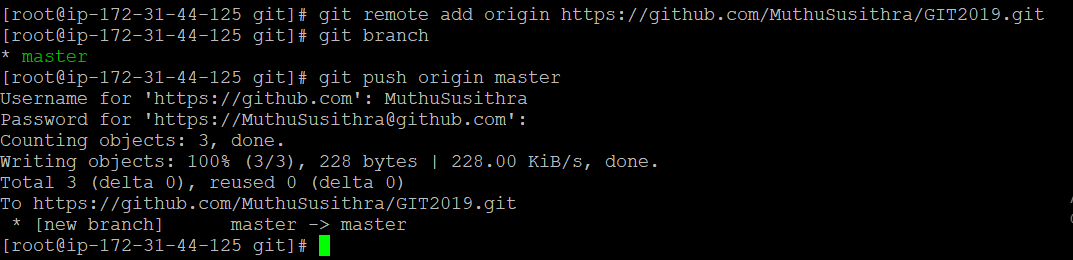


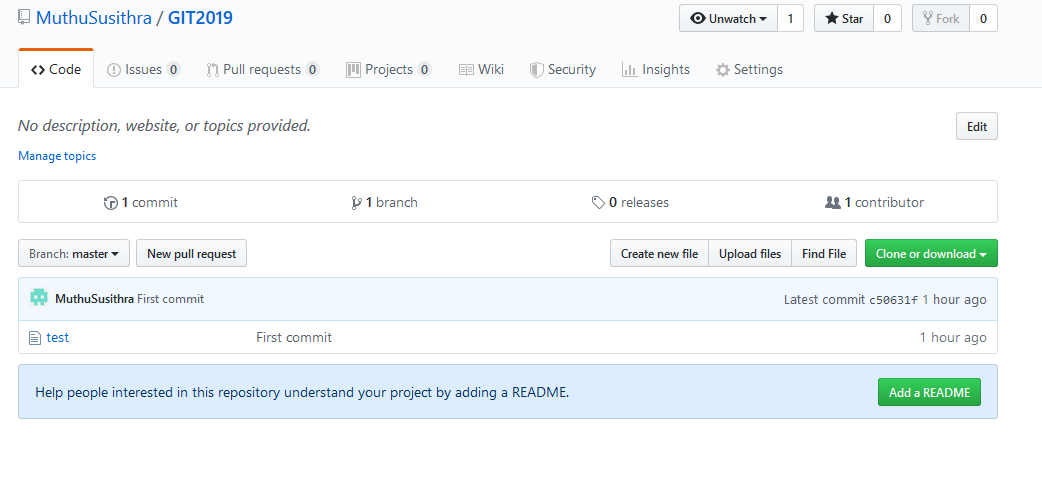
**Connect to GITHUB :**

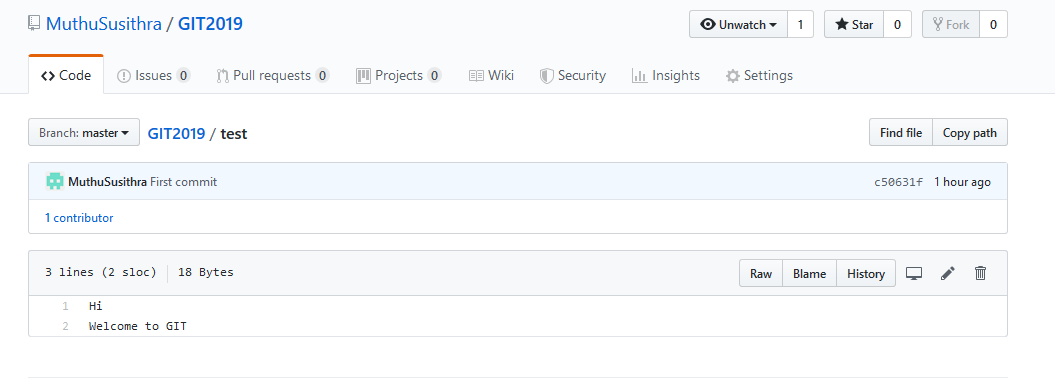


**git remote add origin https://github.com/MuthuSusithra/GIT2019.git**

**git push origin master**







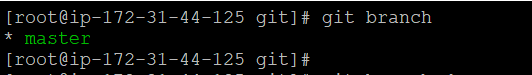
**BRANCH :**

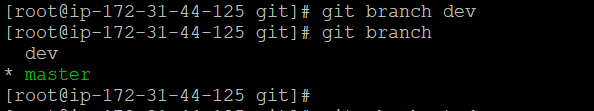
**git branch** – to view what is the current branch

**git branch <branch\_name>** – to create a new branch

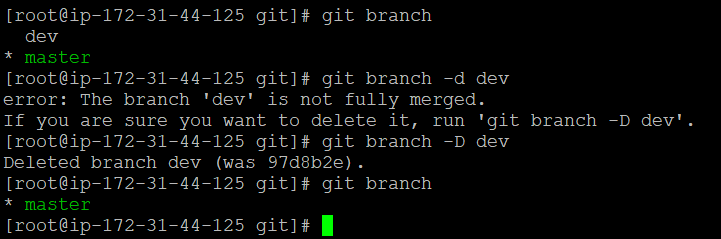
**git checkout <branch\_name> –** to navigate to another branch

**git branch -d <branch\_name>** – to delete the local branch, only if you have already pushed and merged it with your remote branches.  
**git branch -D <branch\_name>** – to force delete the branch regardless of its push and merge status, so be careful using this one!







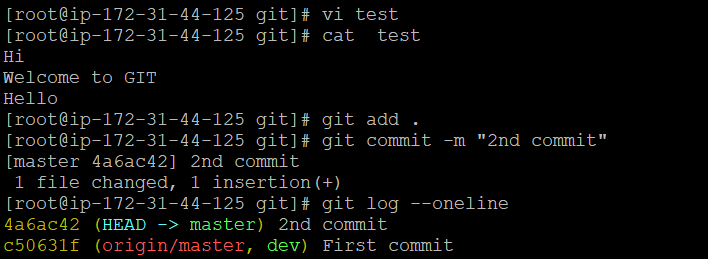


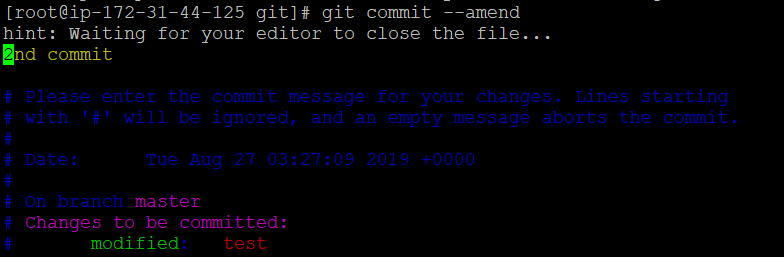
**Remove/Rename Operations :**

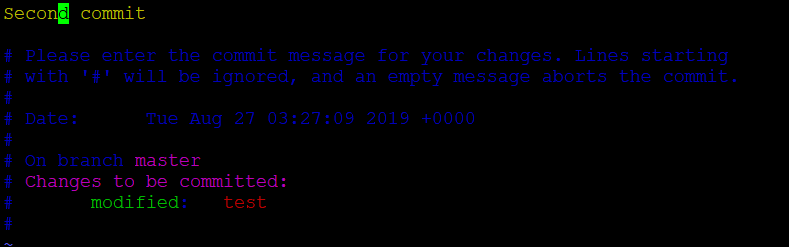
To remove file from staging area **: git reset <file\_name>**

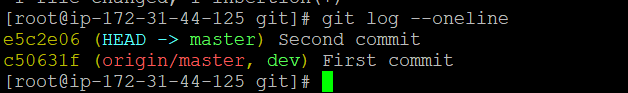
To remove all untracked files **: git clean –f**

To rename the recent commit : **git commit --amend**









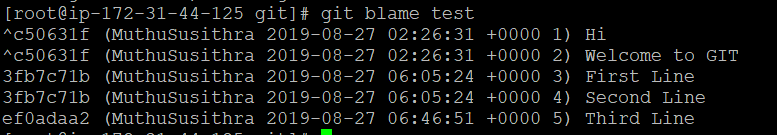
To delete the commits :

**git reset –-hard HEAD^**

**git reset –-hard HEAD~n –** to delete the last “n” number of commits [Eg : **git reset –hard HEAD~2** will delete the last 2 commits]

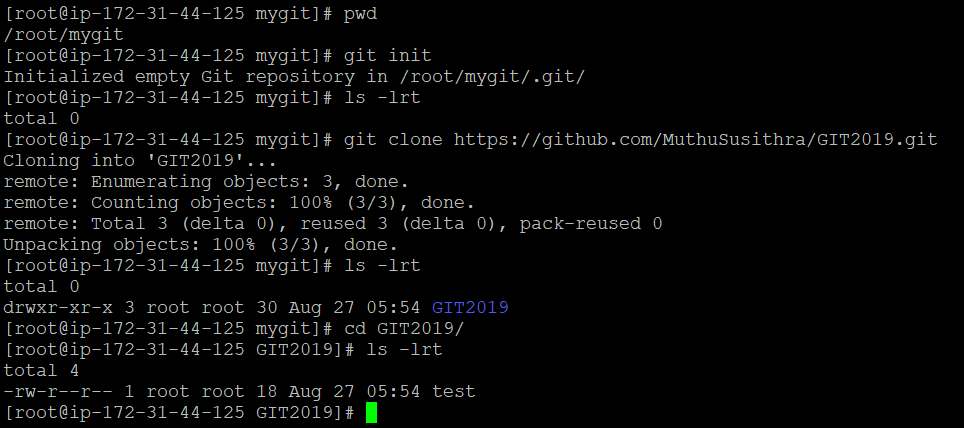


To view line by line history : **git blame <file\_name>**



**CLONE and FORK:**

**git clone <https://github.com/MuthuSusithra/GIT2019.git> -** to copy an existing repository from github



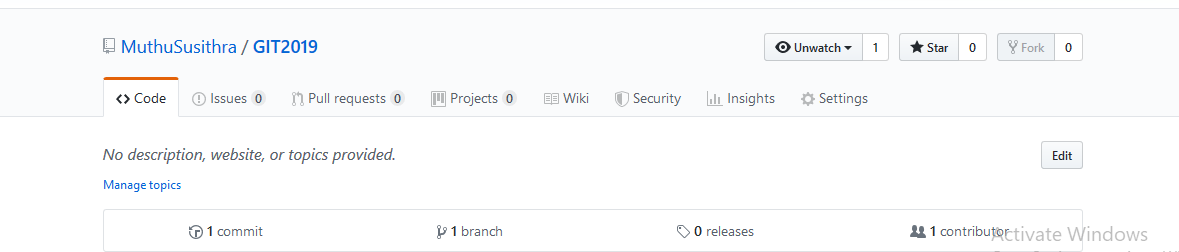
A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

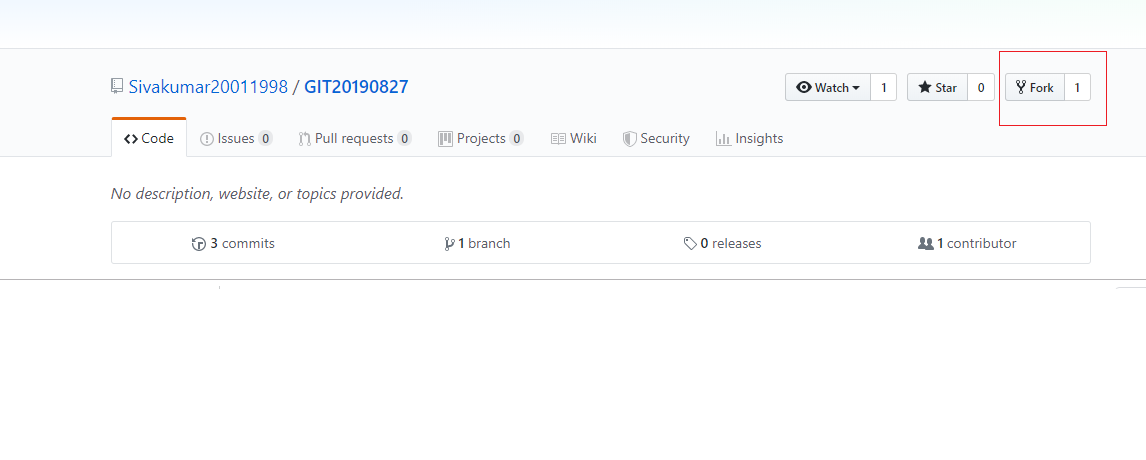
Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea.

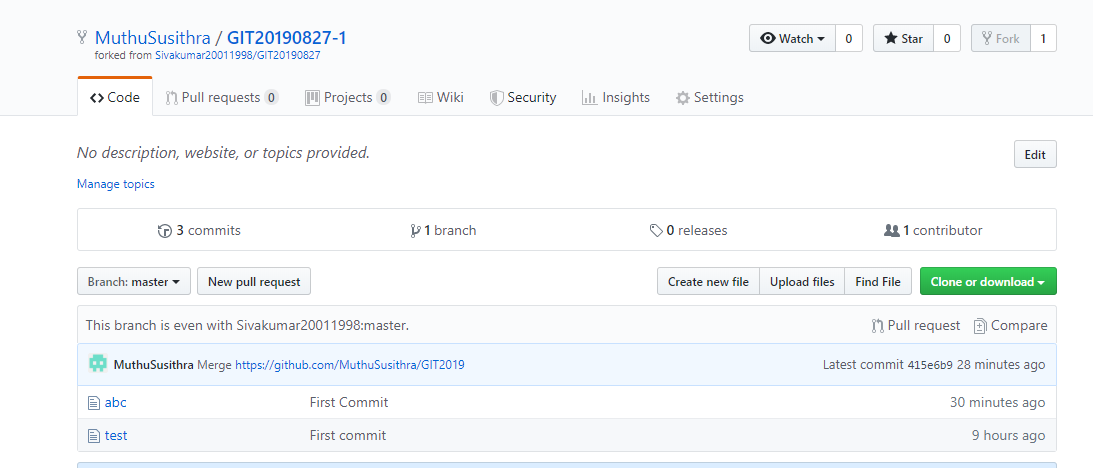
A great example of using forks to propose changes is for bug fixes. Rather than logging an issue for a bug you've found, you can:

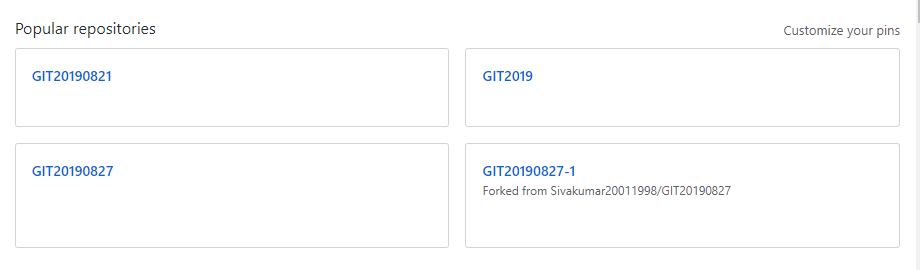
* Fork the repository.
* Make the fix.
* Submit a pull request to the project owner.

If the project owner likes your work, they might pull your fix into the original repository!



****



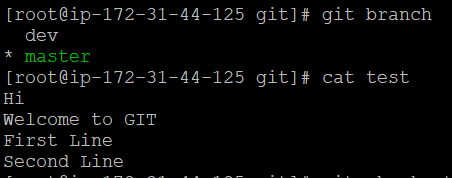


**MERGE :**

**git merge <branch\_name>** - To merge the branch with master branch

Note : merge should be executed only from master branch .

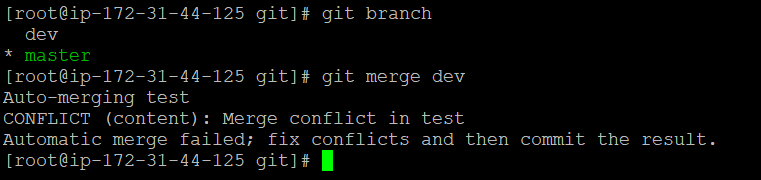
Example : Consider two branches “master” and “dev” with files as below .





To merge dev with master : **git merge dev** (from master branch)

In this case, it creates conflict as the last line in both the files are different .



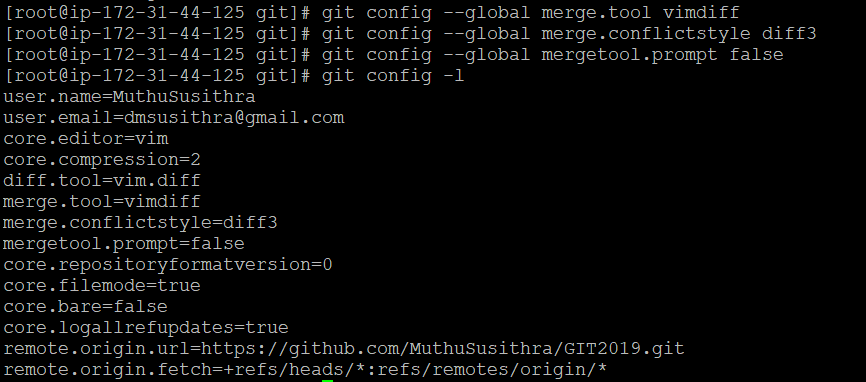
To resolve this conflict , we need to use mergetool .

Add the below configuration for using mergetool :

**git config --global merge.tool vimdiff**

**git config --global merge.conflictstyle diff3**

**git config --global mergetool.prompt false**



**git mergetool** – to merge when conflicts are present

Operations in mergetool :

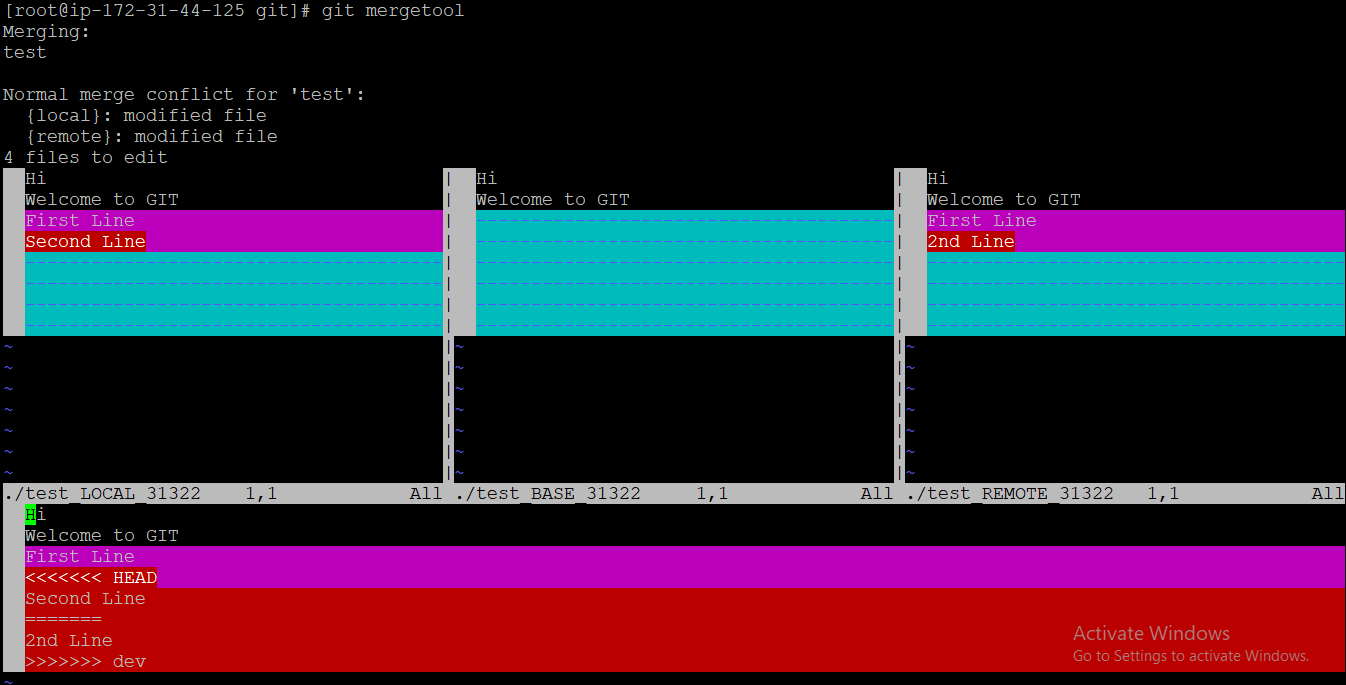
**Ctrl+W** – to navigate to each box

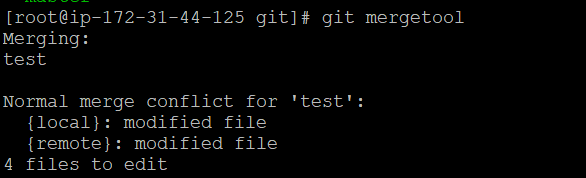
**:diffg LO** – to select LOCAL box

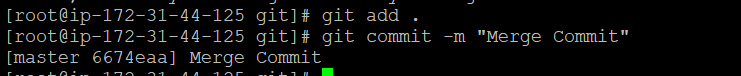
**:diffg RE** – to select REMOTE box

**:diffg BA** – to select BASE box

**:wqa** – To save and quit all









**REBASE :**

**git rebase master – to be executed from feature branch**

**git checkout master**

**git merge <feature\_branch>**



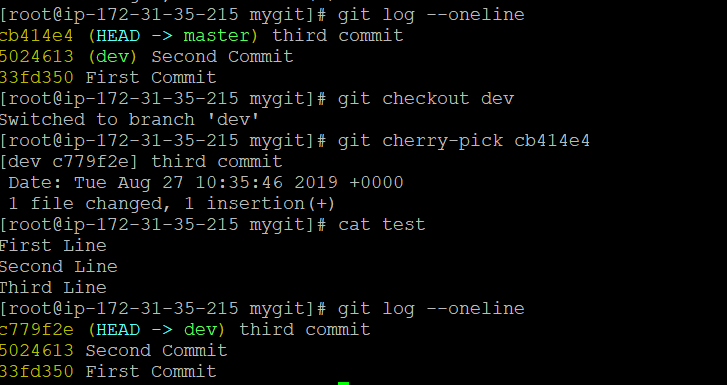
Execute “git rebase master” from feature branch to change the head of feature branch to that in master branch .

Once the head is changed , you can merge the changes from master branch .

**CHERRY-PICK :**

To select the particular commit using its commit id

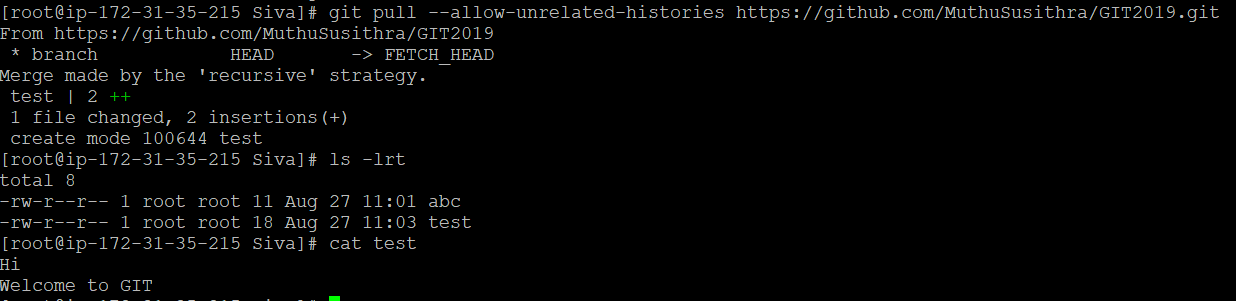
Command : **git cherry-pick <commit\_id>**

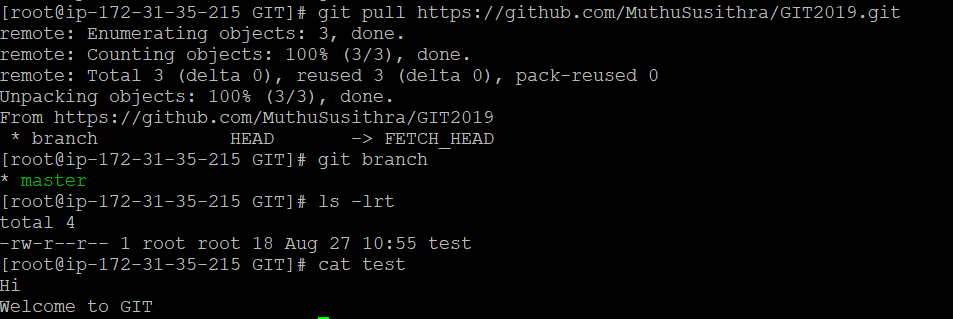


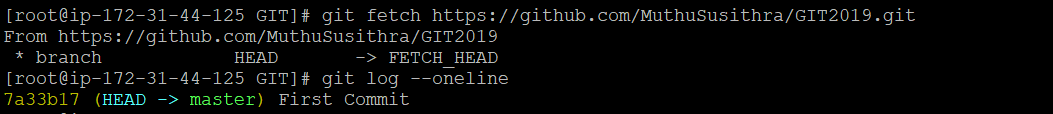
**FETCH & PULL :**

The **git pull** command first runs **git fetch** which downloads content from the specified remote repository. Then a **git merge** is executed to merge the remote content refs and heads into a new local merge commit.









**STASHING :**

Used when you want to record the current state of the working directory and the index, but want to go back to a clean working directory .

Commands :

**git stash** – to move your changes from current working directory to stash

**git stash pop** - removes the changes from your stash and reapplies them to your working copy.

**git stash apply** - Reapplies the changes to your working copy and keep them in your stash . [This is useful if you want to apply the same stashed changes to multiple branches.]

**git stash list** – to list the stashes

**git stash save “description”** – to give description for the stash

**git stash clear** – to clear all of your stashes

**git stash drop stash@{1}** – to delete the particular stash

**WEBHOOKS :**

Webhooks provide a way for notifications to be delivered to an external web server whenever certain actions occur on a repository or organization.

You can configure a webhook to execute whenever:

* A repository is pushed to
* A pull request is opened
* A GitHub Pages site is built
* A new member is added to a team