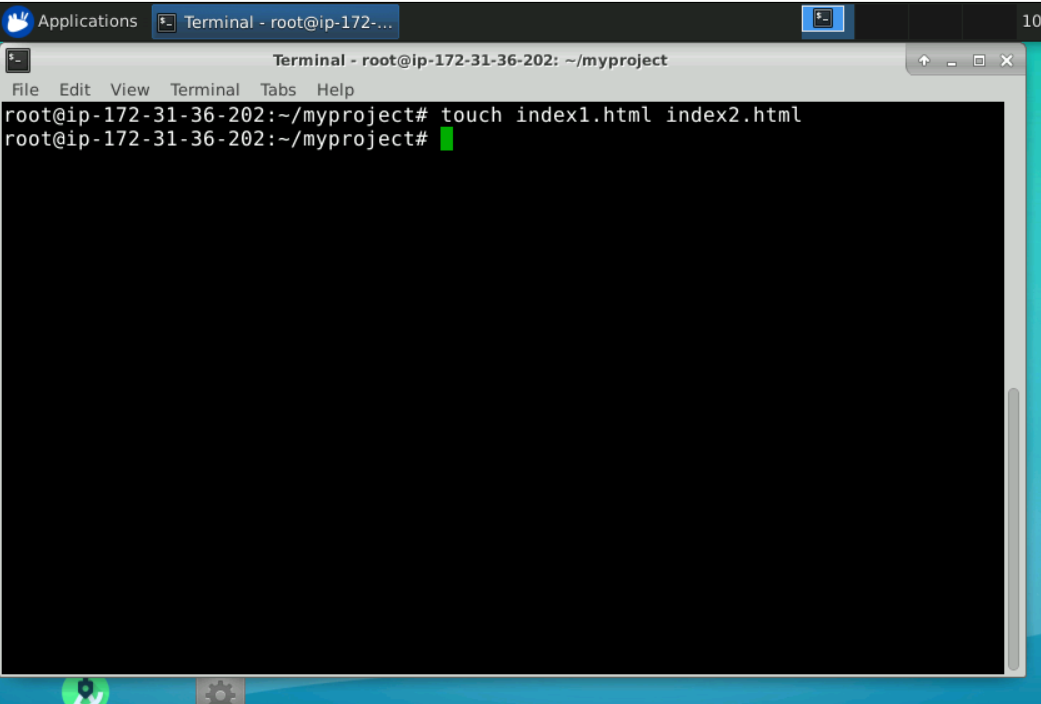
**2. CRUD Operations in git**

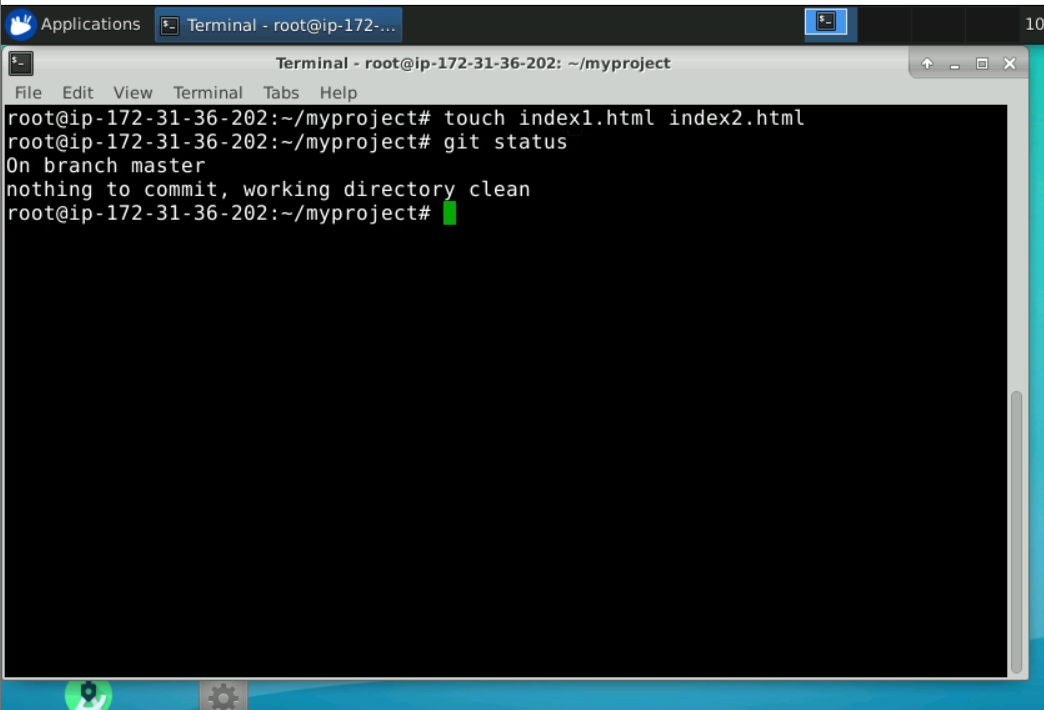
Create a file inside the directory

$ touch index1.html index2.html

****

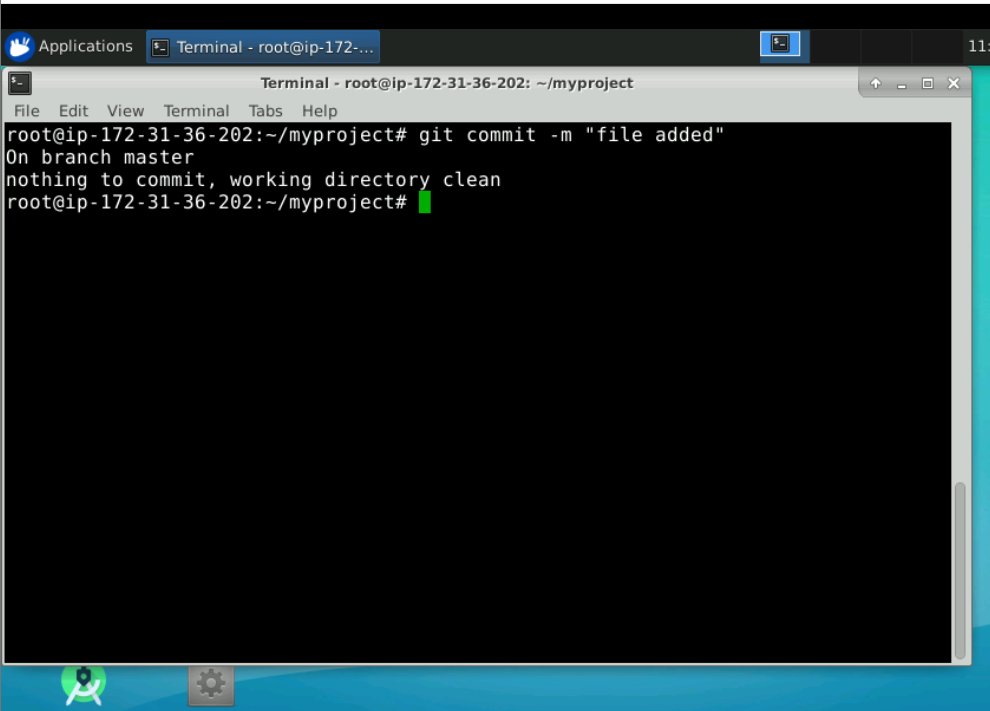
See the status of current working directory

$ git status

****

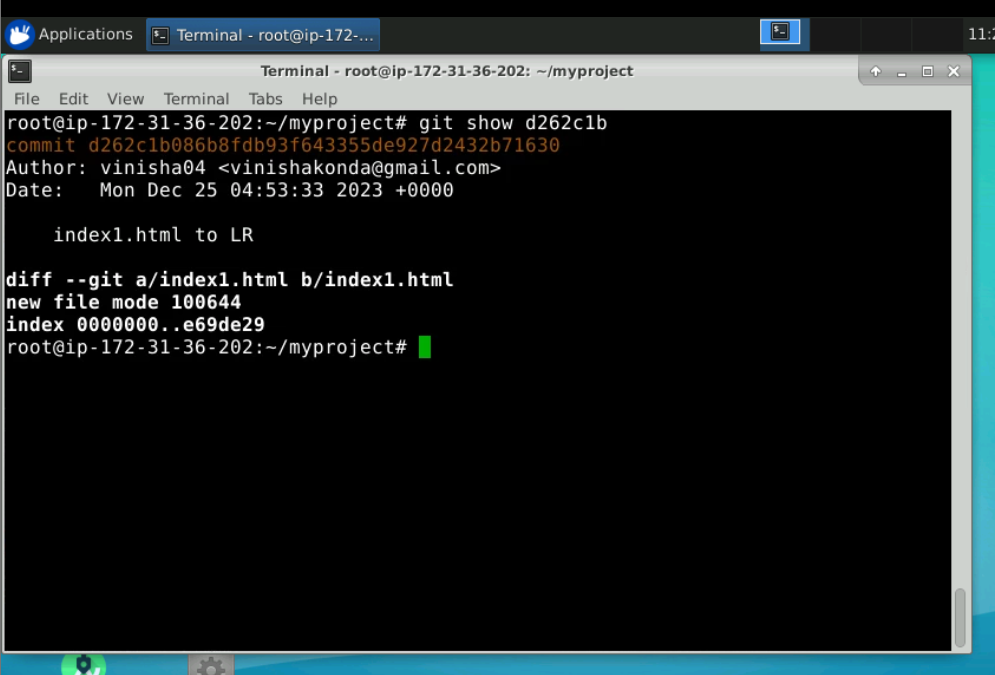
Commit the files or changes to LR

$ git commit -m “file added”

****

See what happened in that commit

$ git show <commitid>

****

Take the files that are present in Local repo:

Choose any one the file and make modifications on it

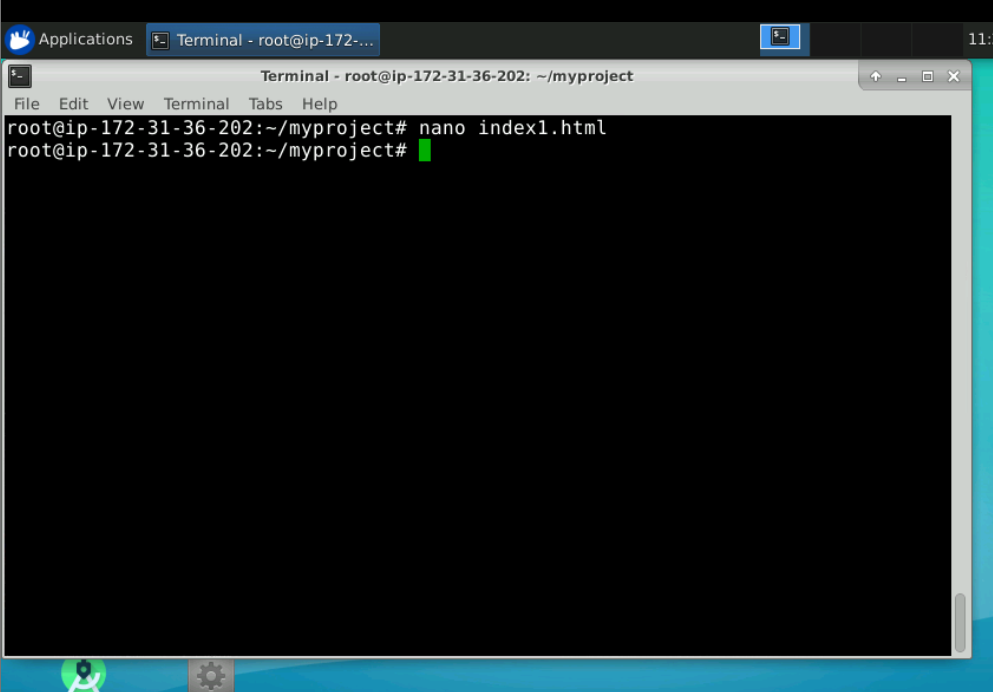
$ nano index1.html

Write some lines -> add content

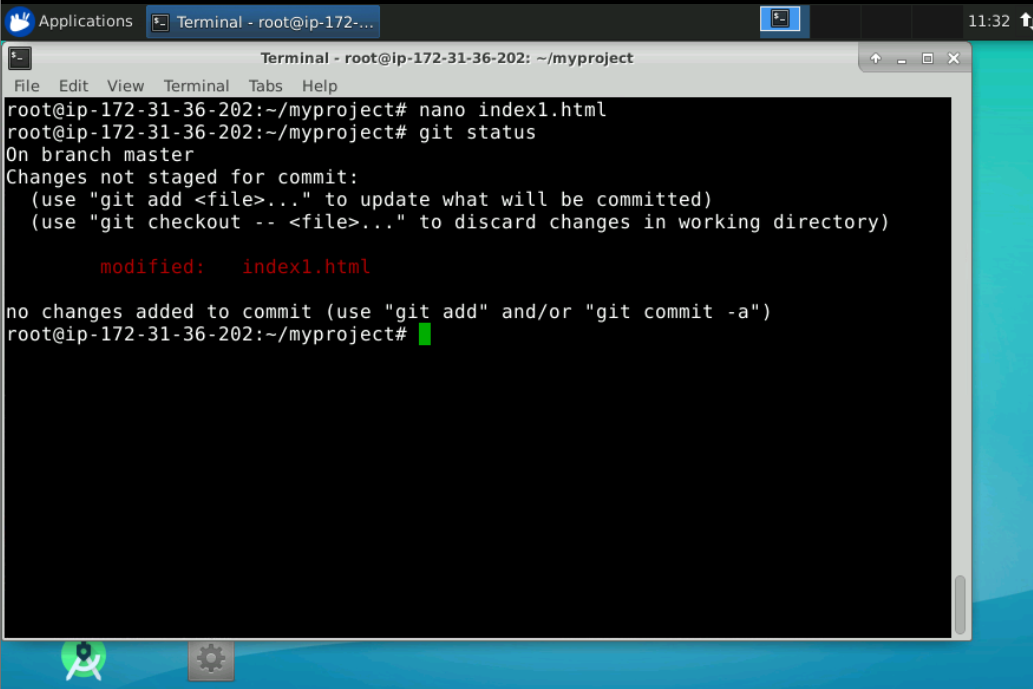
Save the file =>

press CTL x and

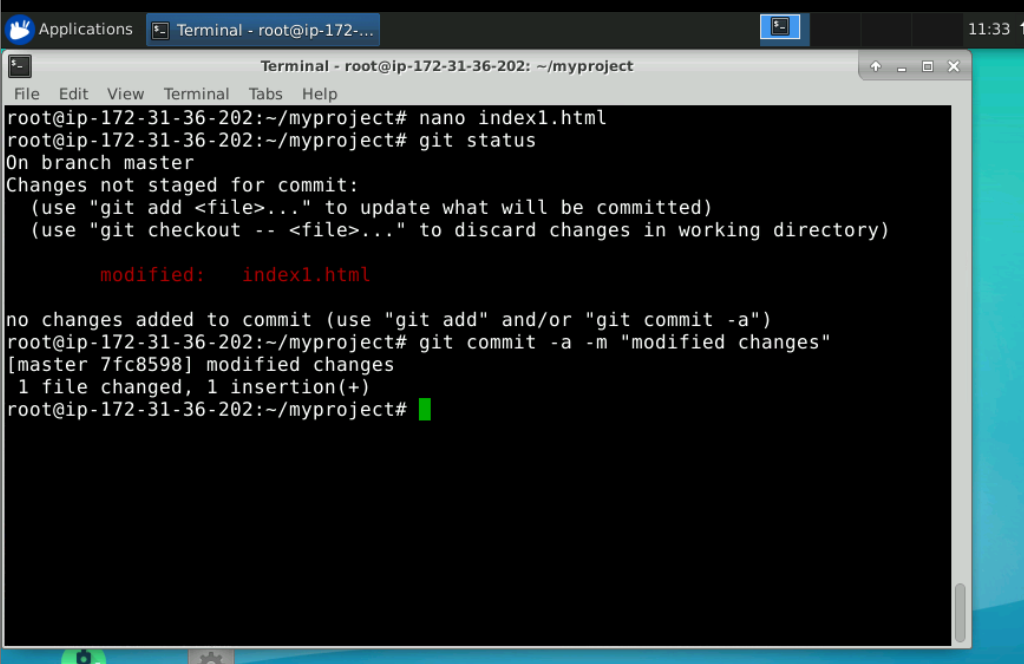
press y and then press enter key on your keyboard



$ git status



$  git commit -a -m "modified changes"



1. Create 2 files

$ touch file1.txt file2.xml

We want these files to be ignored

Create .gitignore file

$ nano .gitignore

Add filename to be ignored

\*.xml

\*.txt

Save the file

Now give

$ git status

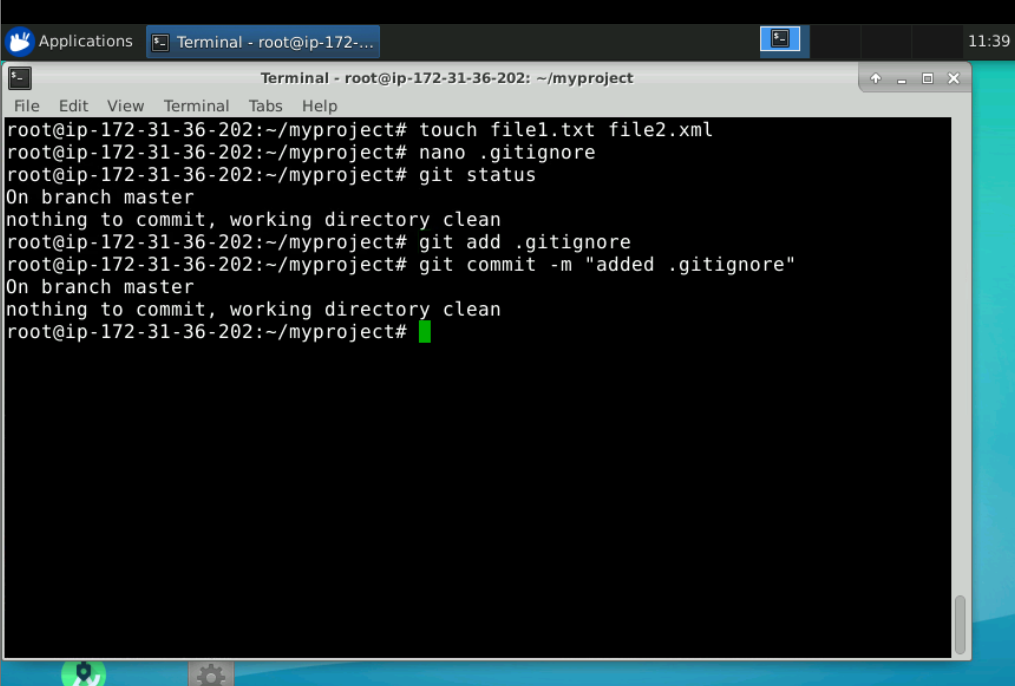
You will see both the files txt and xml are not being tracked by git

However .gitignore is a file that has to be tracked

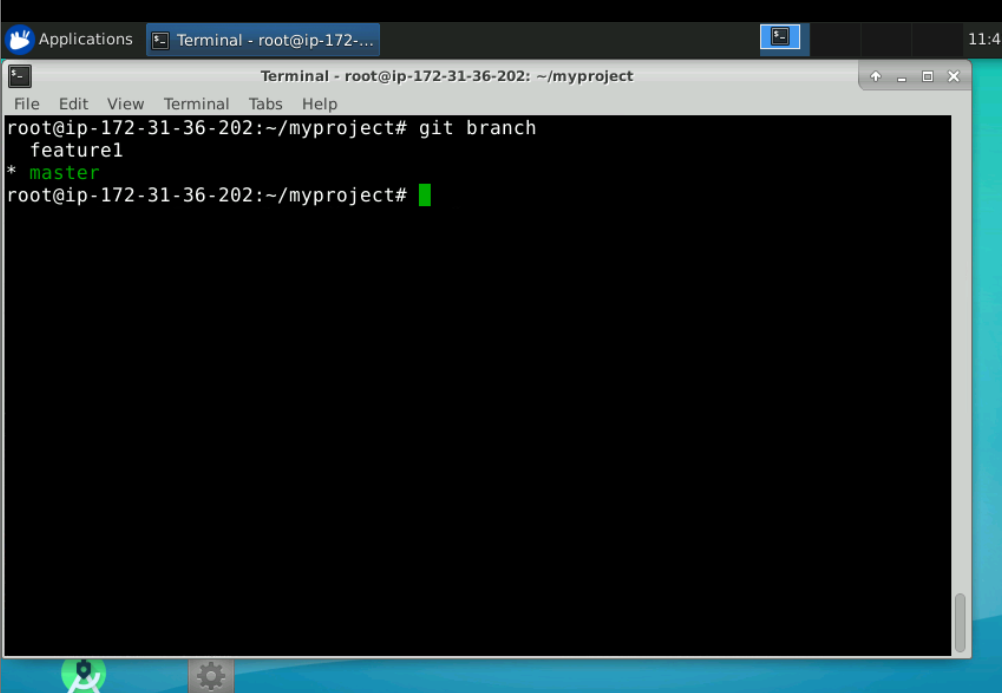
So

$ git add .gitignore

$ git commit -m “added .gitignore”



$ git branch



$ git status ⇒ working tree should be clean

Command to create a branch with name as feature1

$ git branch feature1

Command to switch to branch with name feature1

$ git checkout feature1

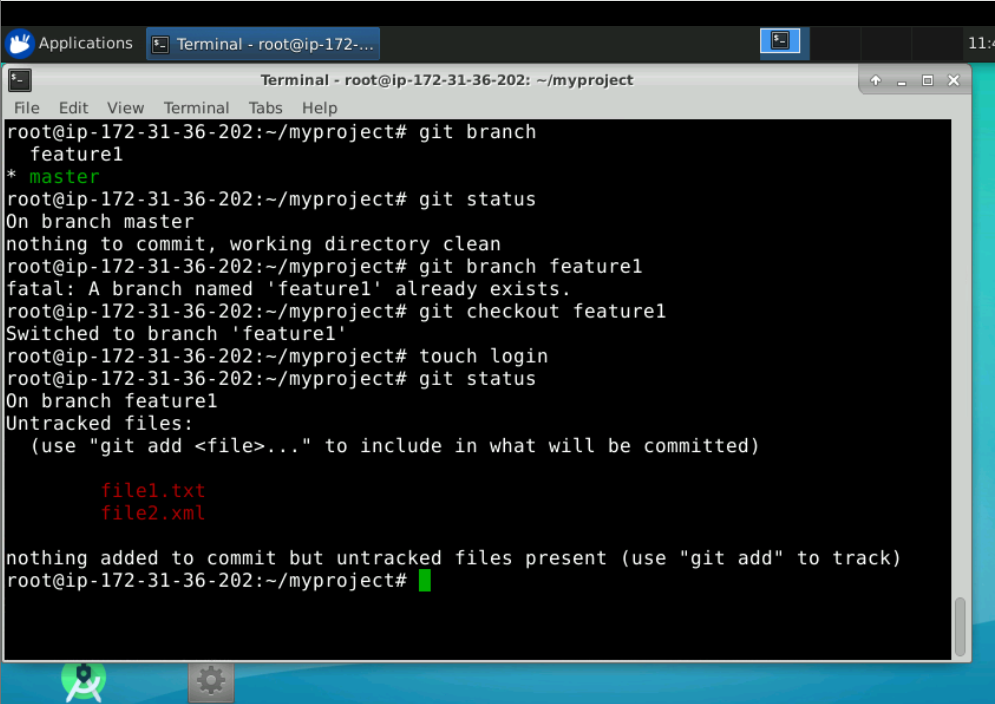
Switched to branch feature1

Create a new file on the branch feature1

$ touch login

$ git status

Untracked file



$ git add login

$ git commit -m “on branch feature1”

$ git log --oneline

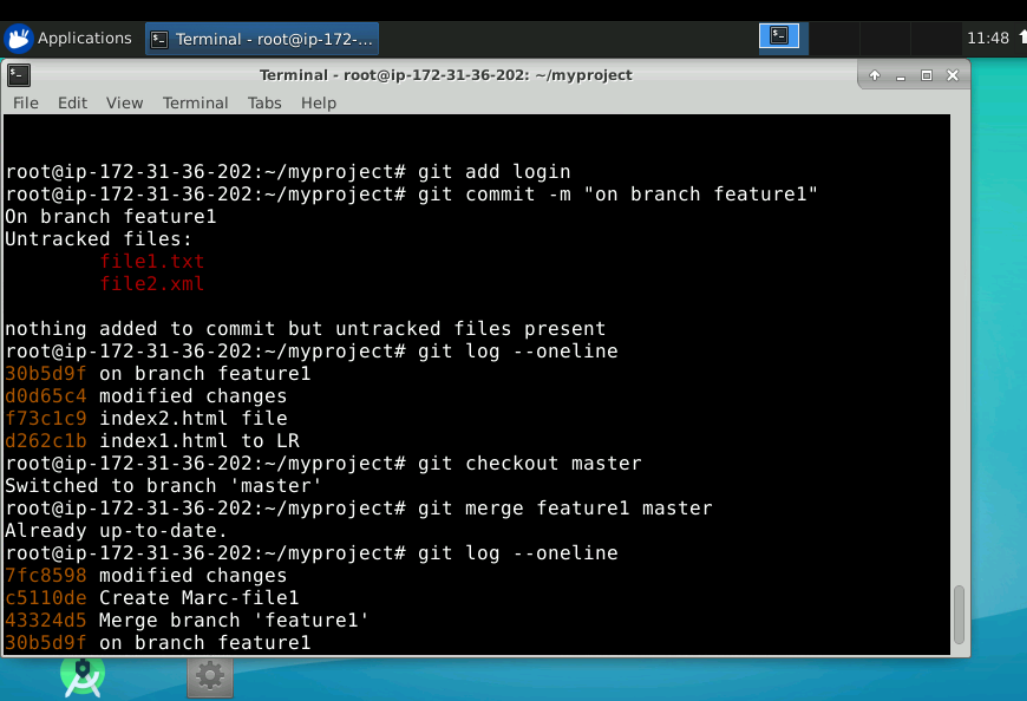
Merging the feature1 branch to master branch

$ git checkout master

$ git merge feature1 master

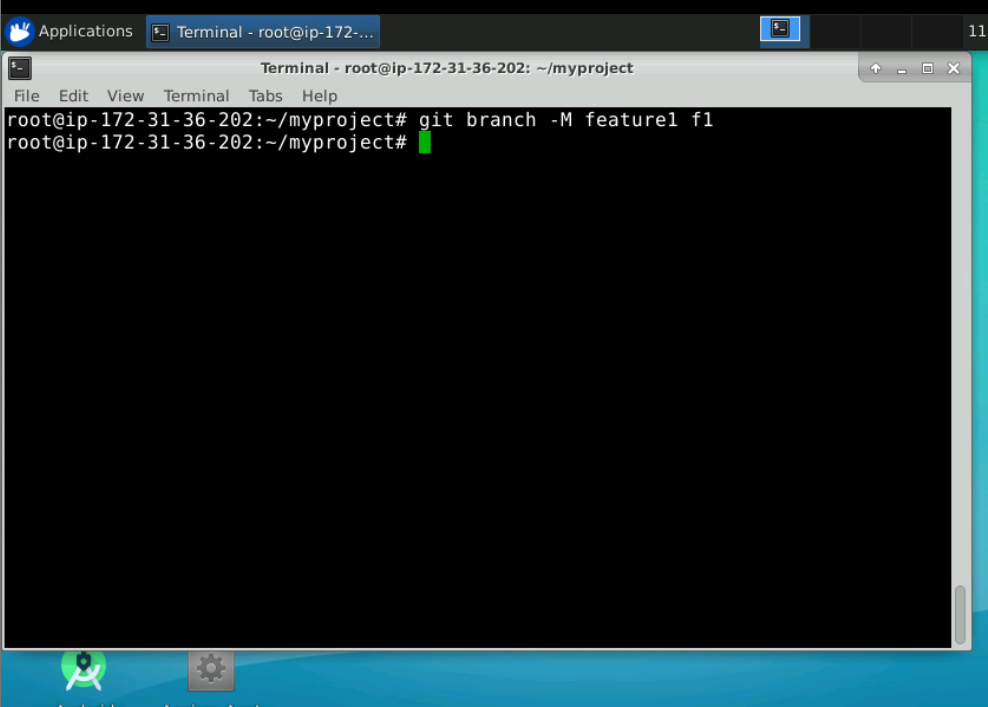
$ git log --oneline

All commits of login are present in master also, login file is also there on master branch



Rename a branch

$ git branch -M feature1 f1



Lets delete the branch.

$ git branch -d branchname

