**Branching Development Model**

**Step 1:** Create a local directory “mthetarepo” in the local Ubuntu machine and create two files “firstfile” & “secondfile” and commit the files.

$mkdir mthetarepo  
$cd mthetarepo  
$git init  
$touch firstfile  
$vim firstfile   
Add text “This is the first file in git repository”  
$touch secondfile  
$vim secondfile  
Add text “ This is the second file in git repository”

$git add .

$git commit –m “commit for firstfile and secondfile”  
  
**Step 2 :** Push the local repository “mthetarepo” to the git hub repository.  
In the git hub, create a new repository “mthetarepo” and copy the https link.

Using the https link – push the local repository to the git hub repository  
  
$git remote add origin <https://github.com/karmasonam/mthetarepo.git>  
$git push –u origin master

**Step 3:** Once the mthetarepo is pushed to the git hub repository, check the current branches present in the mthetarepo

$ git branch

The current branchis displayed as master

**Step 4 :** Rename the default master branch to production and the new branch “production” will be the master branch.

$ git branch –m master production

Push the new master branch production to the git hub repository

$ git push –u origin production



**Step 5:** In git hub repository, change the default branch to production and delete the master branch. The production branch will now be considered the master branch for the mthetarepo repository.

**Step 6:** Create two new branches hotfix and integration

$ git checkout –b hotfix

$ git checkout –b integration

**Step 7:** Create a new branch feature1 and merge it into the integration branch

$ git checkout –b feature1

$ git branch

$ ls

$ vim firstfile

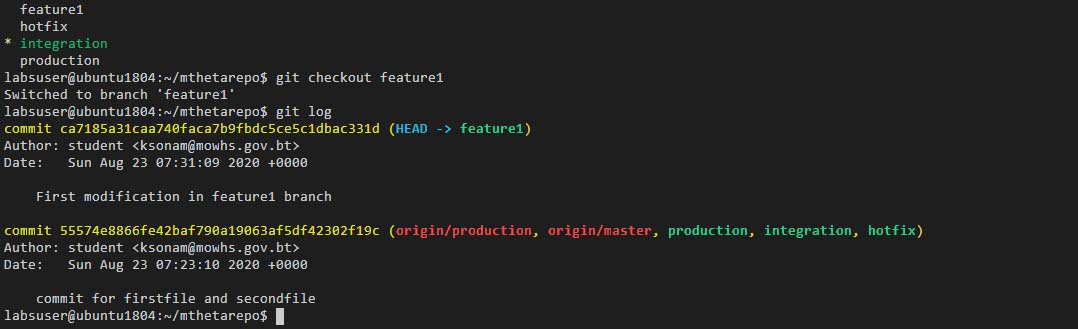
Add text “This is a change made in the feature1 branch”

$ git add .

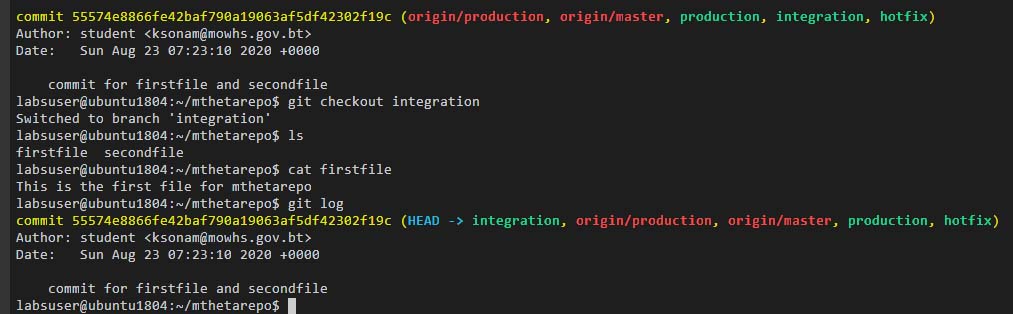
$ git commit –m “Update on firstfile using feature1 branch”

$ git log

Displays the following changes was done in feature1 branch



**Step 8:** Switch to the integration branch and check the git log if the change made in the feature1 branch is reflected in integration branch.



The integration branch does not show the changes made to firstfile in the feature1 branch and therefore, the developer assigned on the feature1 branch can do his modification and then submit for approval to the integration branch.

**Step 9:** Using fast forward merging to merge feature1 branch to the integration branch to reflect the changes made to firstfile.

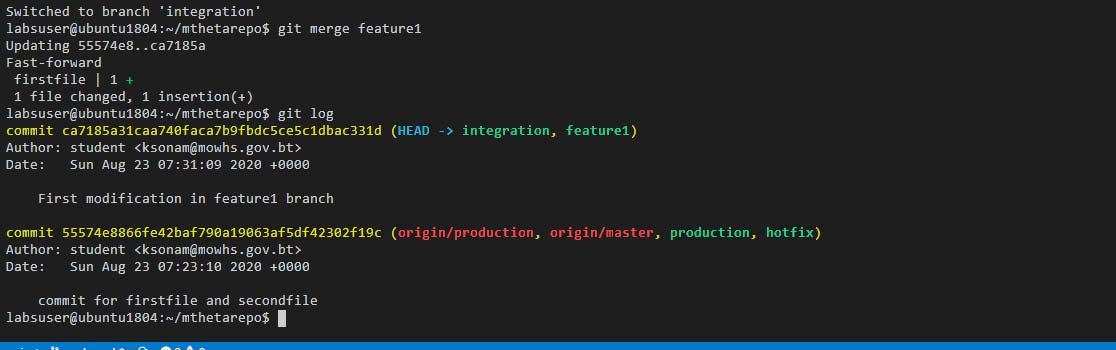
$ git branch

Shows that we are in the integration branch

$ git merger feature1

$ git log

The changes made in the feature1 branch is now shown in the integration branch as below with the use of fast forward merge.



**Step 10:** Create another branch feature2 and merge to the integration branch.

$ git checkout –b feature2

$ vim secondfile

Add the text “This is first modification in second file using feature2 branch”

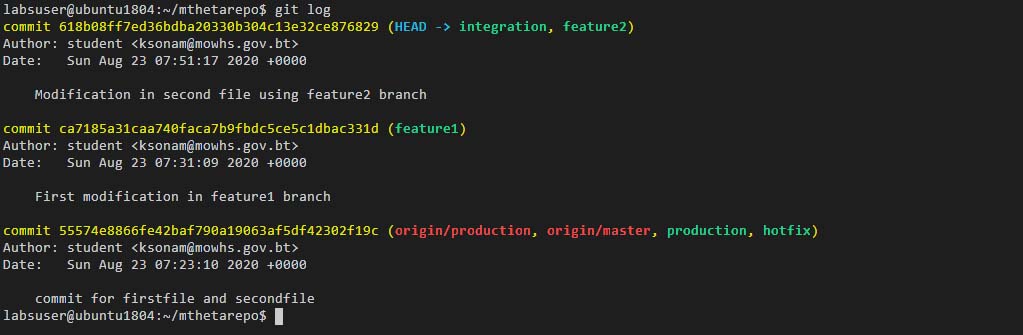
$ git add .

$ git commit –m “Modification in second file using feature2 branch”

$ git merge feature2

$ git log

Displays the following change in the integration branch after the merge.



**Step 11:** Merging the integration branch to the hotfix branch

$ git checkout integration

$touch integrationfile

$vim integrationfile

Add text “This is a new edit in integration branch”

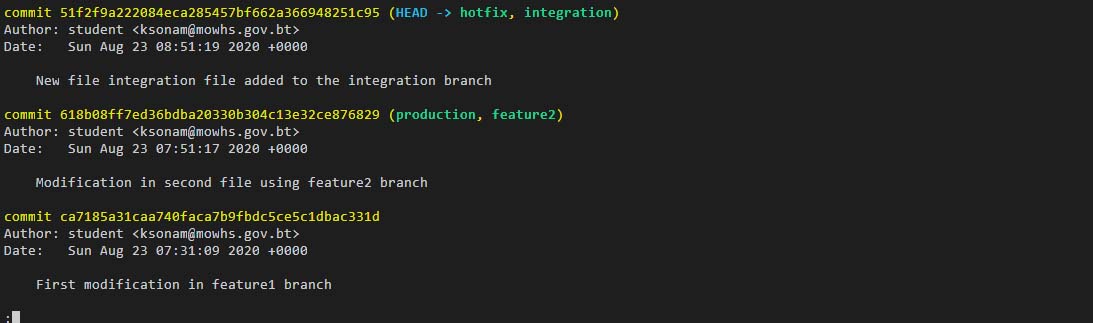
$ git add .

$ git commit –m “New file integration file added to the integration branch”

$ git checkout hotfix

$ git merge integration

$git log



**Step 12:** Merging the hotfix branch & integration branch to the production branch

$ git checkout hotfix

$touch hotfixfile

$vim hotfixfile

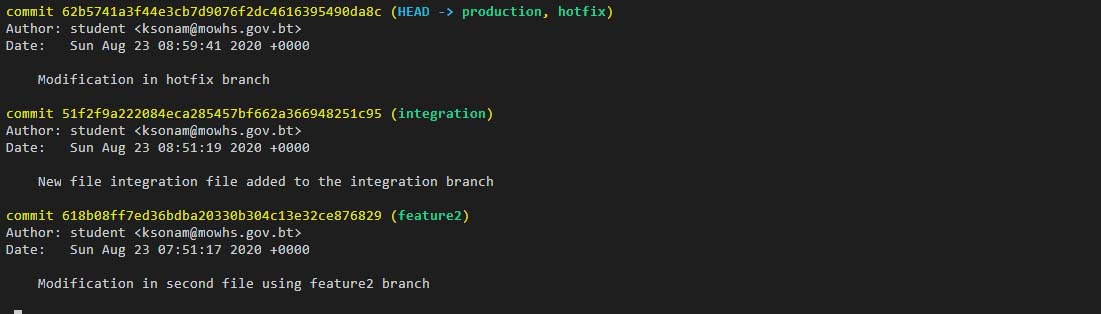
Type text “This is a modification in the hotfix branch”

$git add .

$git commit –m “Modification in hotfix branch”

$git checkout production

$ git merge hotfix



The figure shows that both the modifications at the integration branch and hotfix branch is captured in the production branch.