Module 2: Case Study - Resolving Merge Conflicts

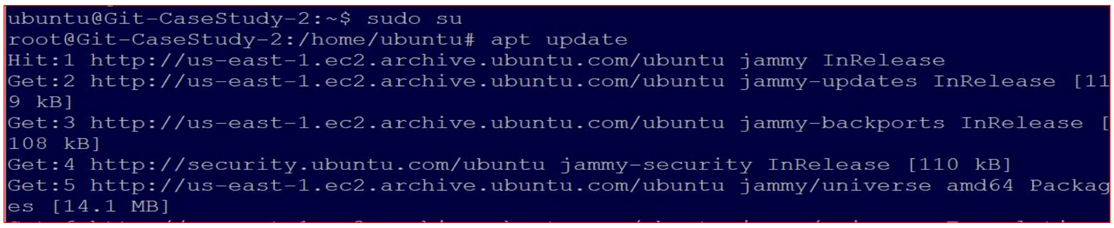
Problem Statement:
You work for Zendrix Software & Co. You have been assigned the task of updating the master branch of their Git repository with all the features from the feature branches.
Here I used GitHub:-
create merge-conflict folder into github and upload main.c file into github. Paste github URL with main.c file in it.

Consider:

- Feature1 branch to be a public branch.
- Feature2 branch to be a private branch.

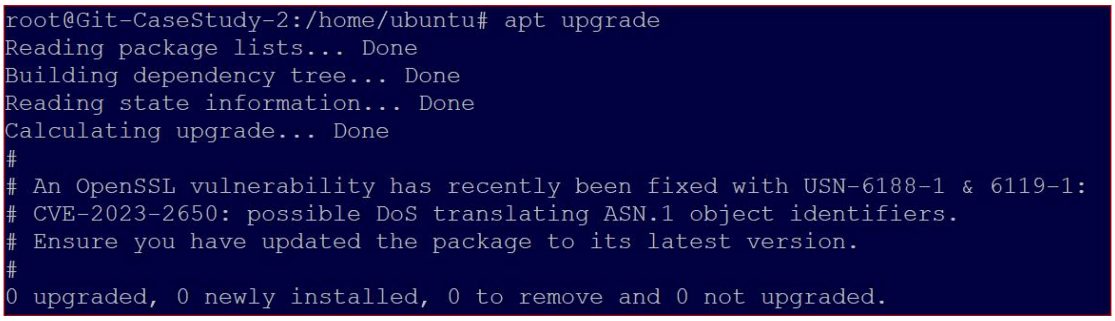
The company relies on a monolithic architecture and for now all the code resides in one file "main.c".
The respective features have been added in the feature branches for main.c.
Meanwhile, a security patch was made to the master branch, and now feature1 and feature2 branches are behind from master by 1 commit.
Tasks To Be Performed:
1. Update Feature1 and Feature2 branch with the Security Patch
2. Apply changes of Feature1 and Feature2 branches on master
3. Finally push all the branches to github
For solving this, please fork the repository to your github account and then work.
As a solution, please submit your GitHub's repository link.
Solution:-
Method 1:- Here we use simple method to resolve merge conflict.

```
$ sudo su
# apt update
```



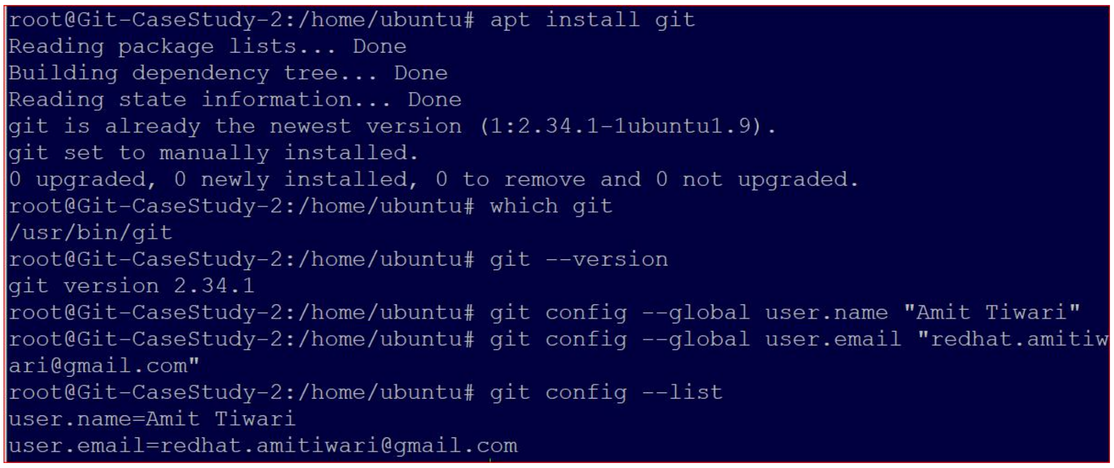
```
ubuntu@Git-CaseStudy-2:~$ sudo su
root@Git-CaseStudy-2:/home/ubuntu# apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [11
9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [
108 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packag
es [14.1 MB]
```

# apt upgrade



```
root@Git-CaseStudy-2:/home/ubuntu# apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
#
# An OpenSSL vulnerability has recently been fixed with USN-6188-1 & 6119-1:
# CVE-2023-2650: possible DoS translating ASN.1 object identifiers.
# Ensure you have updated the package to its latest version.
#
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

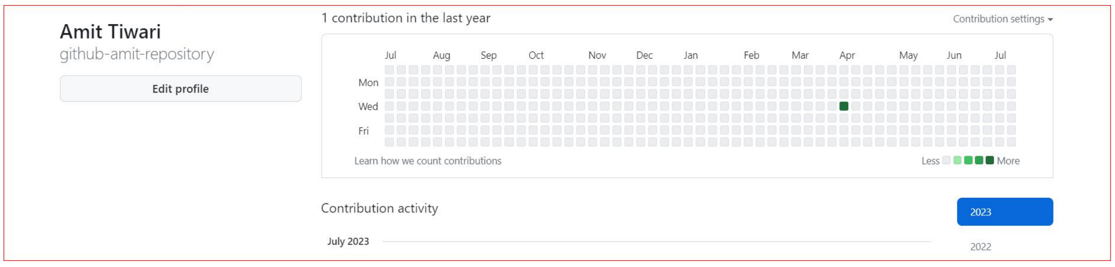
# apt install git
# which git
# git --version
# git config --global user.name "Amit Tiwari"
# git config --global user.email "redhat.amitiwari@gmail.com"
git config --list



```
root@Git-CaseStudy-2:/home/ubuntu# apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-1ubuntu1.9).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@Git-CaseStudy-2:/home/ubuntu# which git
/usr/bin/git
root@Git-CaseStudy-2:/home/ubuntu# git --version
git version 2.34.1
root@Git-CaseStudy-2:/home/ubuntu# git config --global user.name "Amit Tiwari"
root@Git-CaseStudy-2:/home/ubuntu# git config --global user.email "redhat.amitiw
ari@gmail.com"
root@Git-CaseStudy-2:/home/ubuntu# git config --list
user.name=Amit Tiwari
user.email=redhat.amitiwari@gmail.com
```

* Create fork of intellipaat repository:-

* Create github account



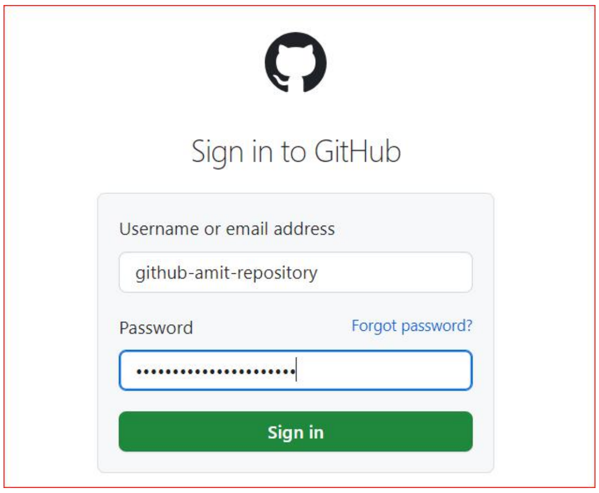
Amit Tiwari
github-amit-repository

1 contribution in the last year

Contribution activity

July 2023

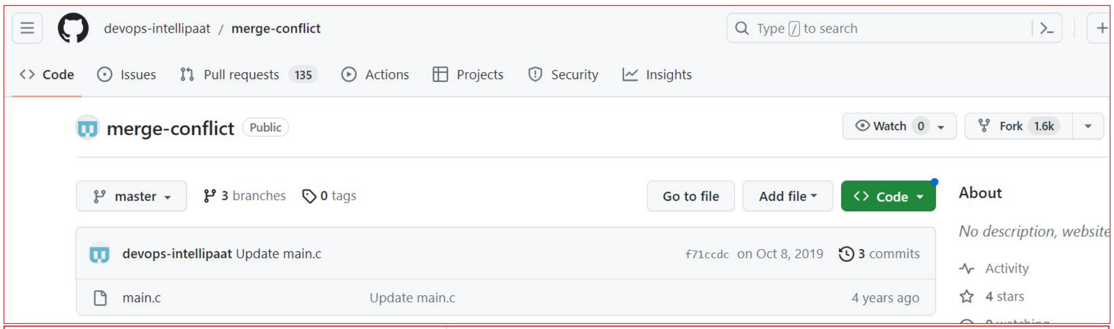
* github account login


Sign in to GitHub

Username or email address: github-amit-repository

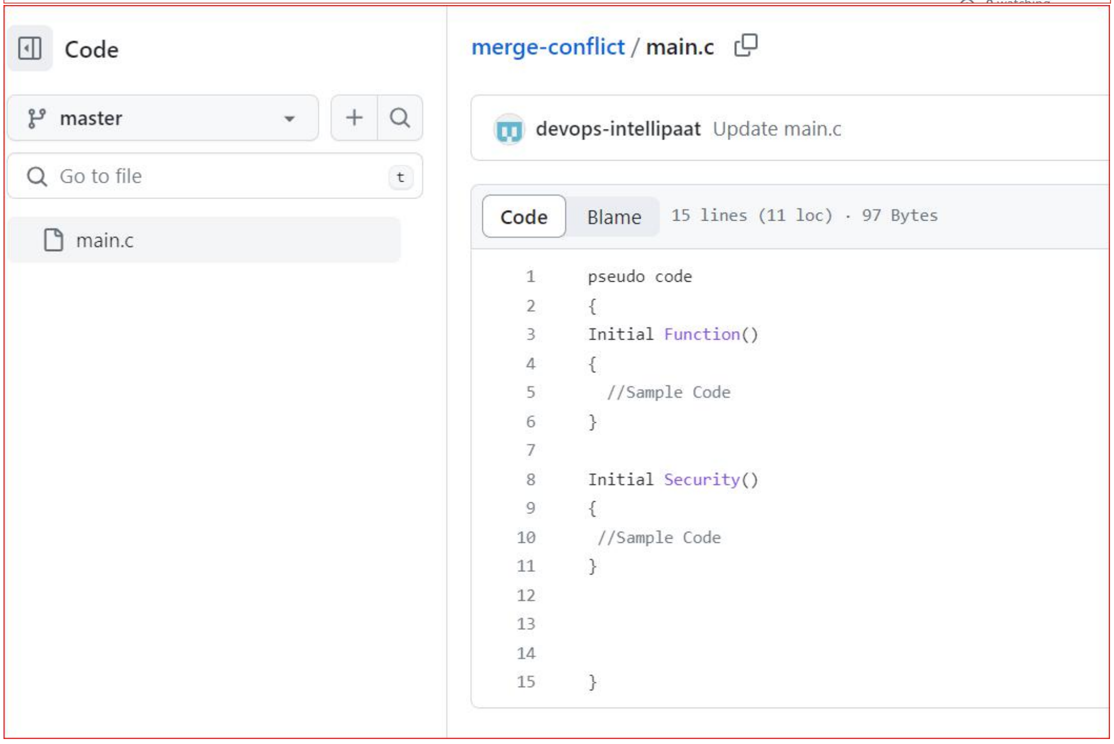
Password: ••••••••••••••••••••••

* Go to merge conflict link provided by intellipaat:-
https://github.com/devops-intellipaat/merge-conflict



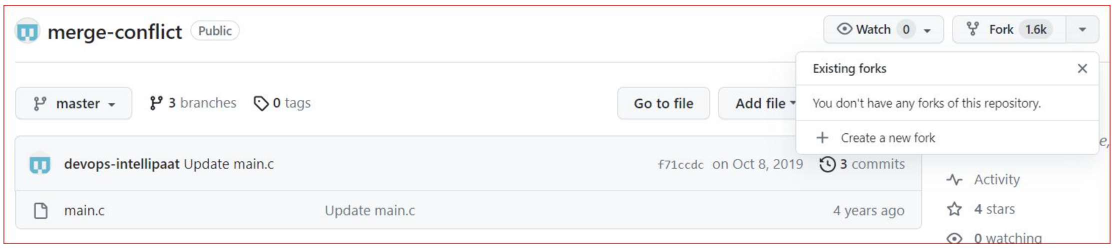
devops-intellipaat Update main.c — f71ccdc on Oct 8, 2019 — 3 commits

main.c — Update main.c — 4 years ago


Code
master
Go to file
t
main.c
merge-conflict / main.c
devops-intellipaat Update main.c
Code
Blame
15 lines (11 loc) · 97 Bytes

```
pseudo code
{
Initial Function()
{
  //Sample Code
}

Initial Security()
{
 //Sample Code
}



}
```

* Now creating the fork of this repo:-



merge-conflict Public

Watch 0 Fork 1.6k

Existing forks

You don't have any forks of this repository.

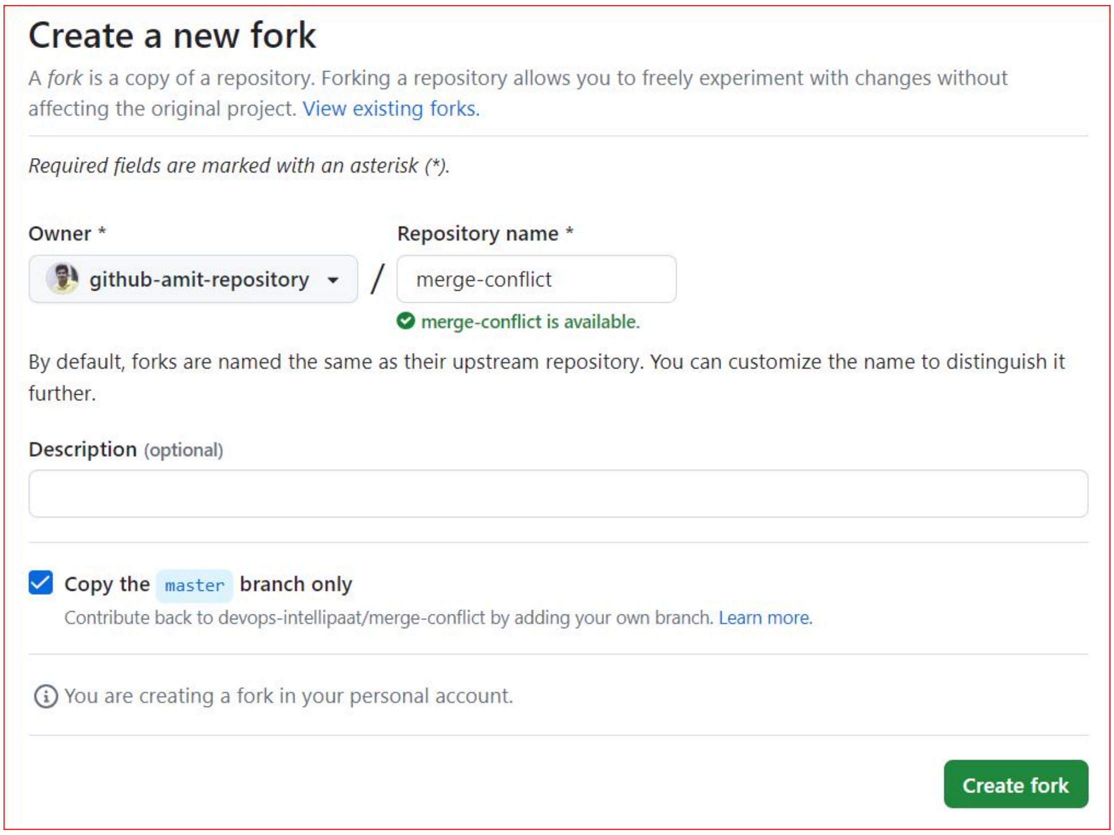
Create a new fork

master 3 branches 0 tags Go to file Add file

devops-intellipaat Update main.c f71ccdc on Oct 8, 2019 3 commits

main.c Update main.c 4 years ago

Activity 4 stars 0 watching



Create a new fork

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

Required fields are marked with an asterisk (*).

Owner * github-amit-repository / Repository name * merge-conflict

merge-conflict is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

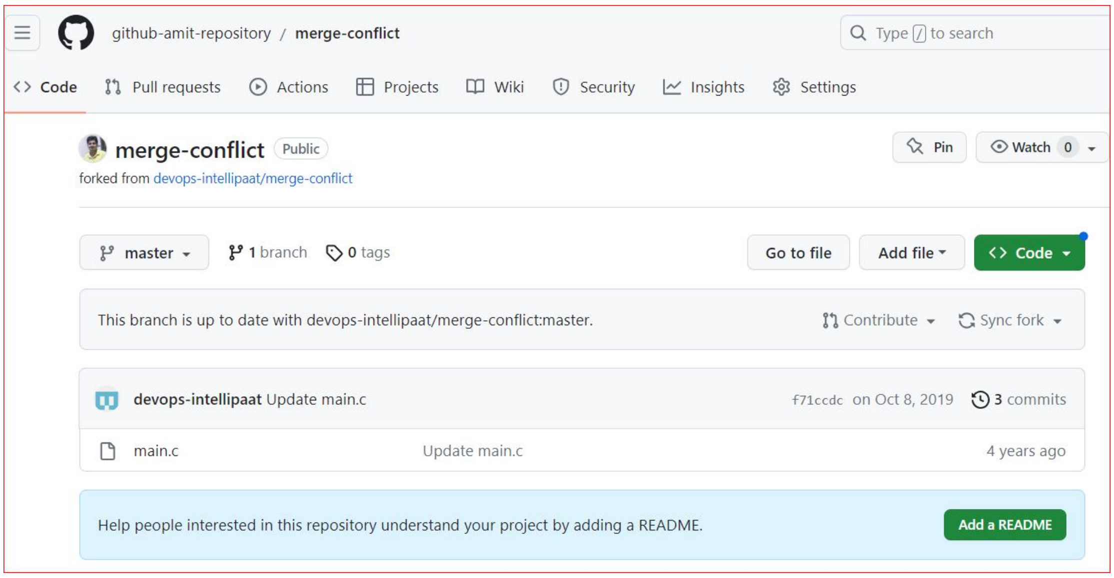
Description (optional)

[✓] Copy the `master` branch only

Contribute back to devops-intellipaat/merge-conflict by adding your own branch. Learn more.

You are creating a fork in your personal account.

Create fork



github-amit-repository / merge-conflict

Code Pull requests Actions Projects Wiki Security Insights Settings

merge-conflict Public Pin Watch 0

forked from devops-intellipaat/merge-conflict

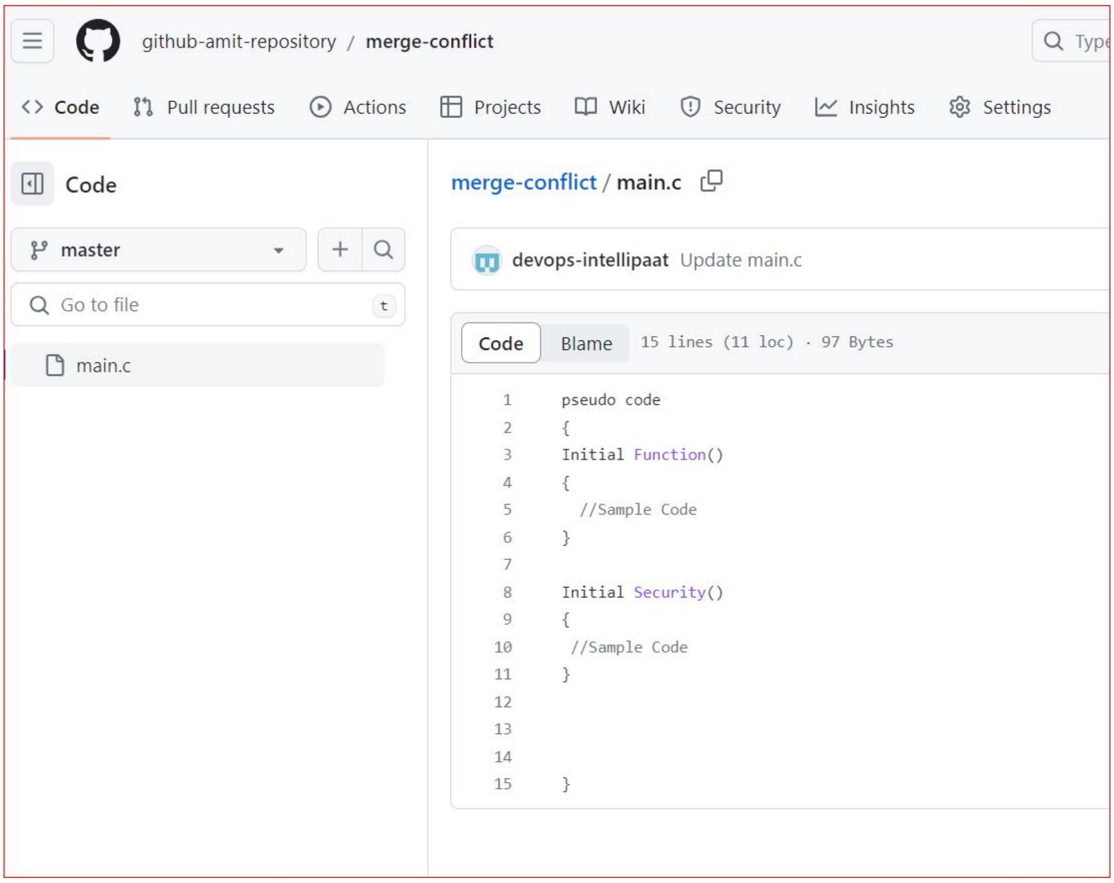
master 1 branch 0 tags Go to file Add file Code

This branch is up to date with devops-intellipaat/merge-conflict:master. Contribute Sync fork

devops-intellipaat Update main.c f71ccdc on Oct 8, 2019 3 commits

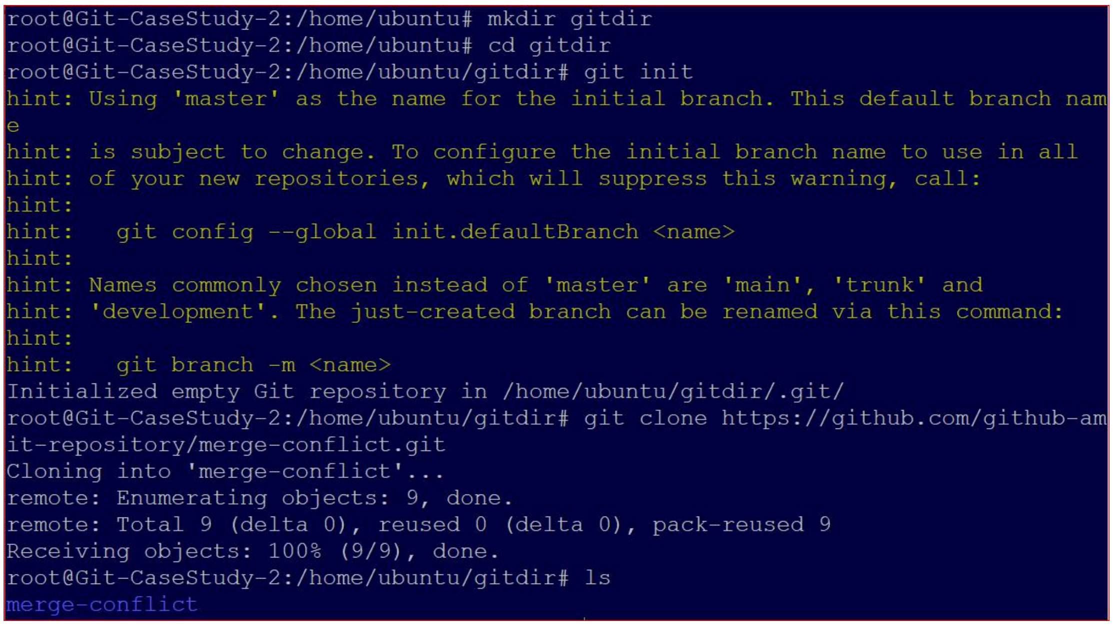
main.c Update main.c 4 years ago

Help people interested in this repository understand your project by adding a README. Add a README



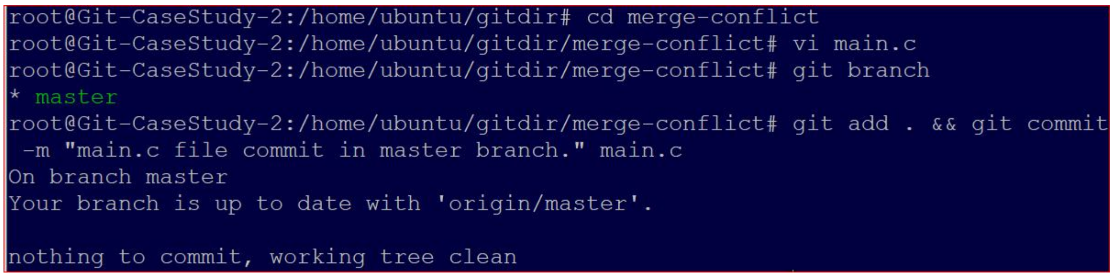
```
pseudo code
{
Initial Function()
{
  //Sample Code
}

Initial Security()
{
 //Sample Code
}



}
```

```
# mkdir gitdir
# cd gitdir
# git init
# git clone https://github.com/github-amit-repository/merge-conflict.git
# ls
```



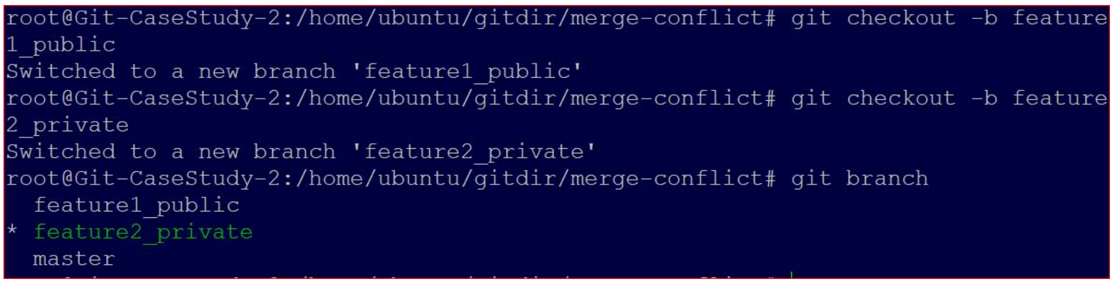
```
root@Git-CaseStudy-2:/home/ubuntu# mkdir gitdir
root@Git-CaseStudy-2:/home/ubuntu# cd gitdir
root@Git-CaseStudy-2:/home/ubuntu/gitdir# git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ubuntu/gitdir/.git/
root@Git-CaseStudy-2:/home/ubuntu/gitdir# git clone https://github.com/github-am
it-repository/merge-conflict.git
Cloning into 'merge-conflict'...
remote: Enumerating objects: 9, done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 9
Receiving objects: 100% (9/9), done.
root@Git-CaseStudy-2:/home/ubuntu/gitdir# ls
merge-conflict
```

# cd merge-conflict
# vi main.c
# git branch
# git add . && git commit -m “main.c file commit in master branch.” main.c



```
root@Git-CaseStudy-2:/home/ubuntu/gitdir# cd merge-conflict
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# vi main.c
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git branch
* master
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git add . && git commit
 -m "main.c file commit in master branch." main.c
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

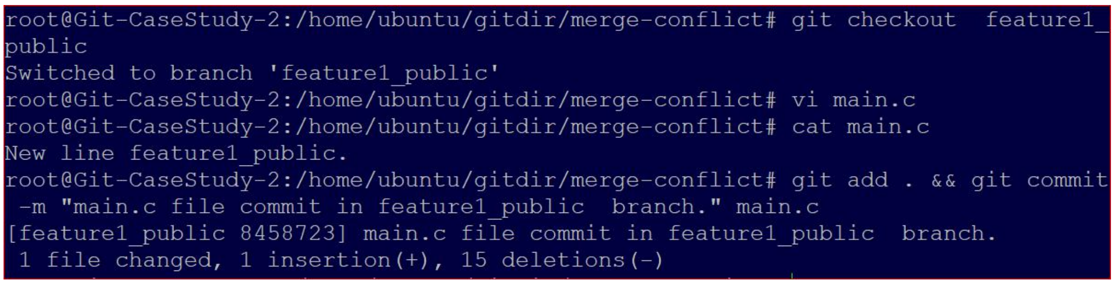
- Feature1 branch to be a public branch.
  # git checkout -b feature1_public
- Feature2 branch to be a private branch.
  # git checkout -b feature2_private
  # git branch



```
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git checkout -b feature
1_public
Switched to a new branch 'feature1_public'
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git checkout -b feature
2_private
Switched to a new branch 'feature2_private'
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git branch
  feature1_public
* feature2_private
  master
```

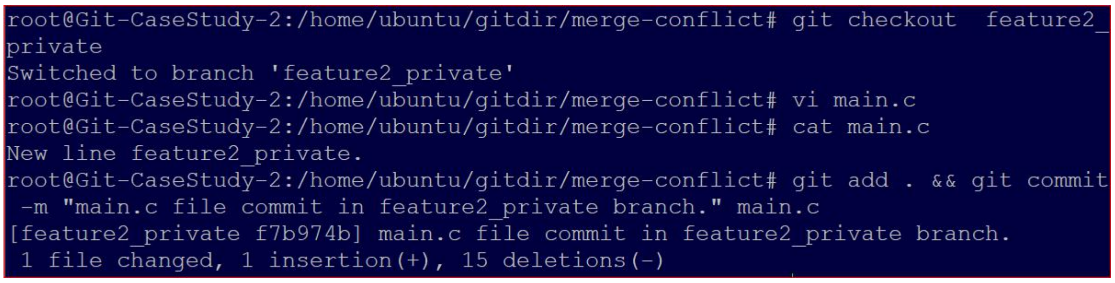
1. Update Feature1 and Feature2 branch with the Security Patch

# git checkout feature1_public
# vi main.c
New line feature1_public.
# cat main.c
O/p:- New line feature1_public.
# git add . && git commit -m “main.c file commit in feature1_public branch.” main.c



```
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git checkout  feature1_
public
Switched to branch 'feature1_public'
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# vi main.c
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# cat main.c
New line feature1_public.
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git add . && git commit
 -m "main.c file commit in feature1_public  branch." main.c
[feature1_public 8458723] main.c file commit in feature1_public  branch.
 1 file changed, 1 insertion(+), 15 deletions(-)
```

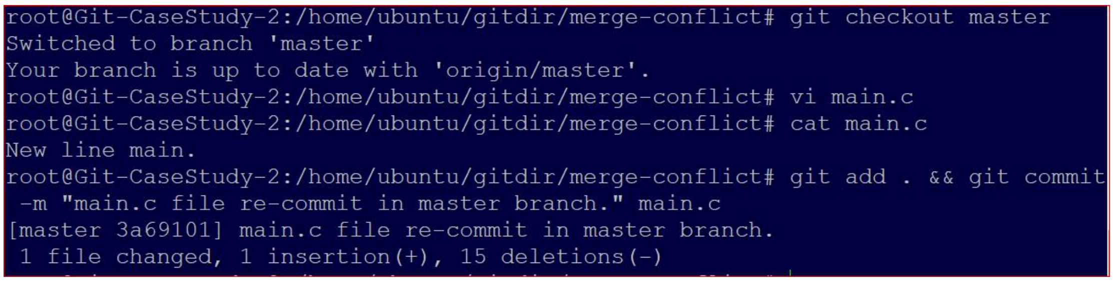
```
# git checkout  feature2_private
# vi main.c
New line feature2_private.
# cat main.c
O/p:- New line feature2_private.
# git add . && git commit -m “main.c file commit in
feature2_private branch.” main.c
```



```
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git checkout  feature2_
private
Switched to branch 'feature2_private'
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# vi main.c
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# cat main.c
New line feature2_private.
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git add . && git commit
 -m "main.c file commit in feature2_private branch." main.c
[feature2_private f7b974b] main.c file commit in feature2_private branch.
 1 file changed, 1 insertion(+), 15 deletions(-)
```

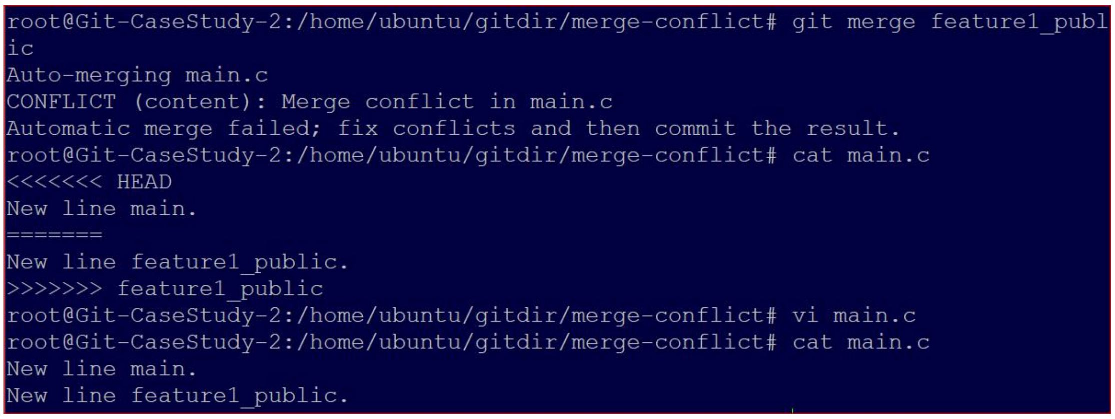
2. Apply changes of Feature1 and Feature2 branches on master

```
# git checkout master
# vi main.c
New line main.
# cat main.c
O/p:-New line main.
# git add . && git commit -m “main.c file re-commit in master
branch.” main.c
```



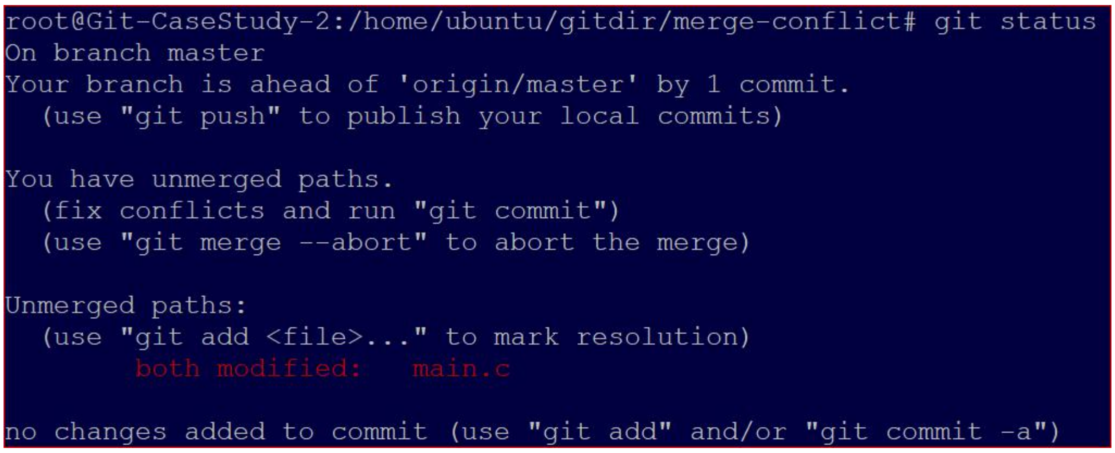
```
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# vi main.c
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# cat main.c
New line main.
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git add . && git commit
 -m "main.c file re-commit in master branch." main.c
[master 3a69101] main.c file re-commit in master branch.
 1 file changed, 1 insertion(+), 15 deletions(-)
```

# git merge feature1_public
# cat main.c
# vi main.c
# cat main.c



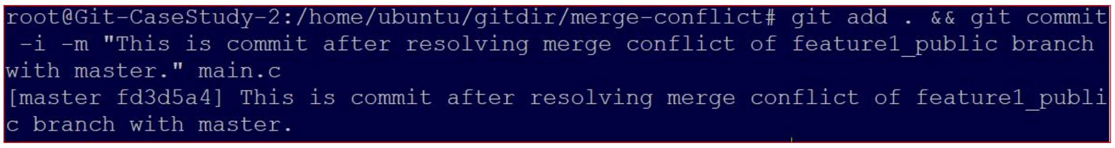
```
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git merge feature1_publ
ic
Auto-merging main.c
CONFLICT (content): Merge conflict in main.c
Automatic merge failed; fix conflicts and then commit the result.
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# cat main.c
<<<<<<< HEAD
New line main.
=======
New line feature1_public.
>>>>>>> feature1_public
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# vi main.c
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# cat main.c
New line main.
New line feature1_public.
```

# git status



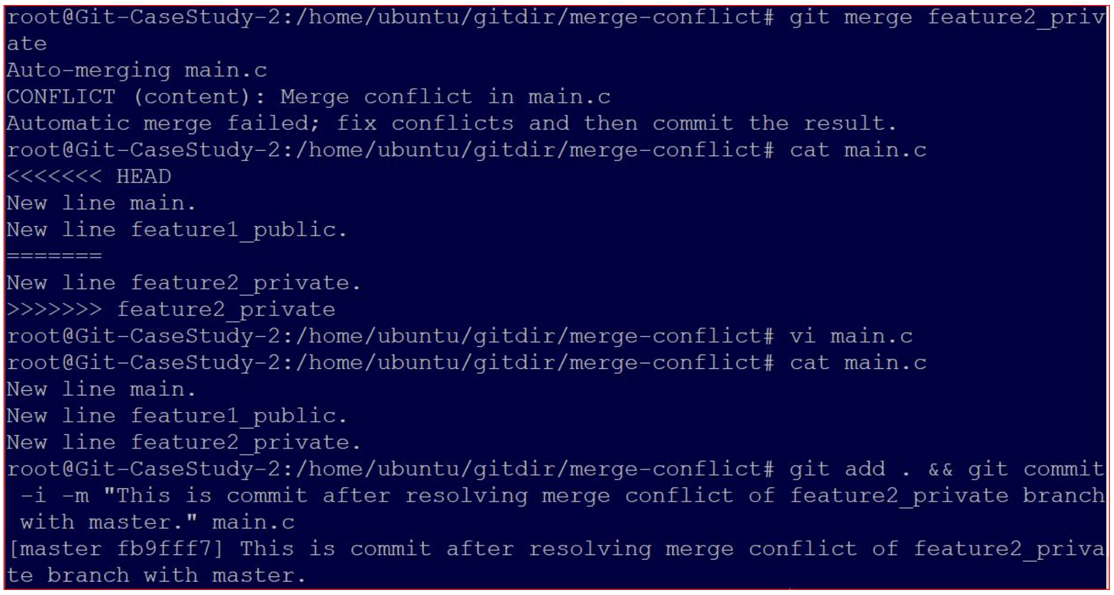
```
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   main.c

no changes added to commit (use "git add" and/or "git commit -a")
```

# git add . && git commit -i -m "This is commit after resolving merge conflict of feature1_public branch with master." main.c



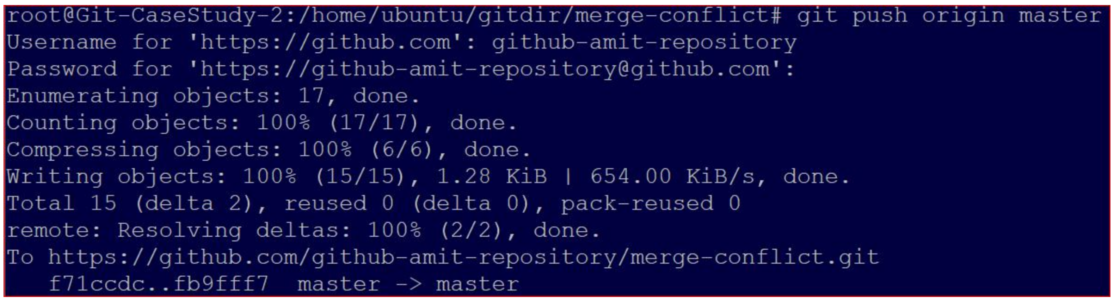
```
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git add . && git commit
 -i -m "This is commit after resolving merge conflict of feature1_public branch
with master." main.c
[master fd3d5a4] This is commit after resolving merge conflict of feature1_publi
c branch with master.
```

# git merge feature2_private
# cat main.c
# vi main.c
# cat main.c
# git add . && git commit -i -m "This is commit after resolving merge conflict of feature2_private branch with master." main.c

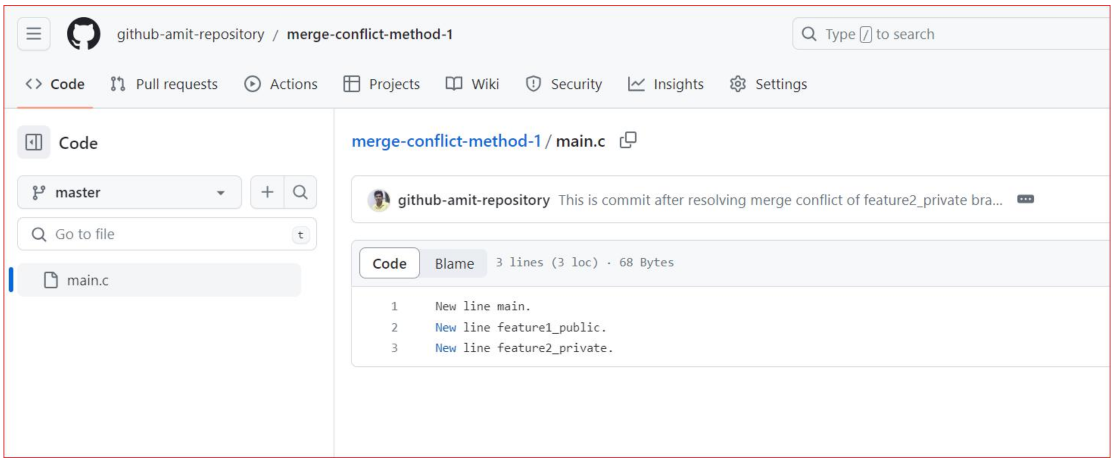


```
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git merge feature2_priv
ate
Auto-merging main.c
CONFLICT (content): Merge conflict in main.c
Automatic merge failed; fix conflicts and then commit the result.
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# cat main.c
<<<<<<< HEAD
New line main.
New line feature1_public.
=======
New line feature2_private.
>>>>>>> feature2_private
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# vi main.c
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# cat main.c
New line main.
New line feature1_public.
New line feature2_private.
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git add . && git commit
 -i -m "This is commit after resolving merge conflict of feature2_private branch
 with master." main.c
[master fb9fff7] This is commit after resolving merge conflict of feature2_priva
te branch with master.
```

# git push origin master



```
root@Git-CaseStudy-2:/home/ubuntu/gitdir/merge-conflict# git push origin master
Username for 'https://github.com': github-amit-repository
Password for 'https://github-amit-repository@github.com':
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (15/15), 1.28 KiB | 654.00 KiB/s, done.
Total 15 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/github-amit-repository/merge-conflict.git
   f71ccdc..fb9fff7  master -> master
```



github-amit-repository This is commit after resolving merge conflict of feature2_private bra...

3 lines (3 loc) · 68 Bytes

```
New line main.
New line feature1_public.
New line feature2_private.
```

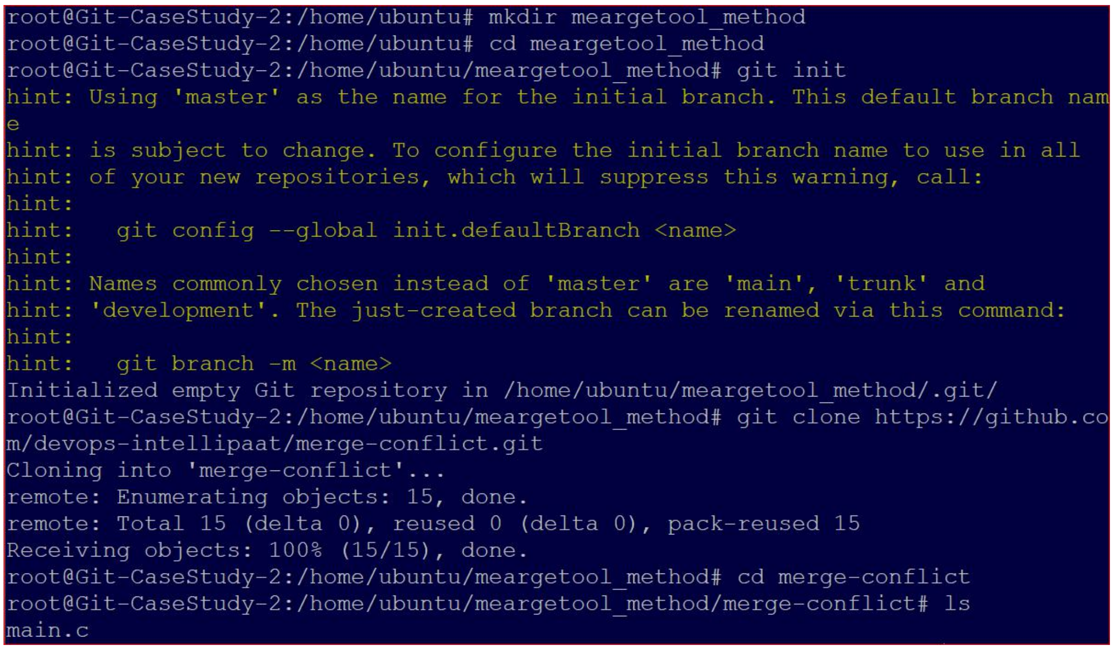
Shared link of GitHub repository:-

https://github.com/github-amit-repository/merge-conflict.git

Method 2 :- Now we use meargetool to resolve conflict.

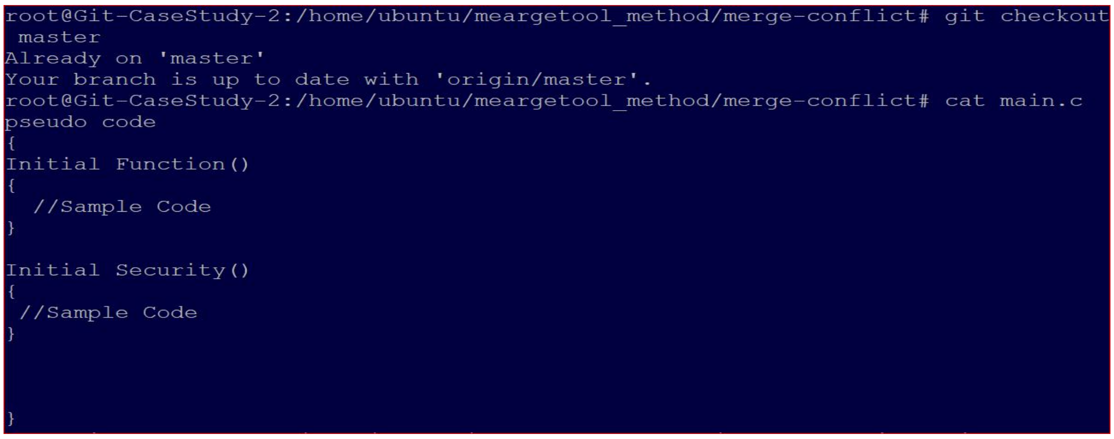
Note:- * We can not create more then one fork of repo within one github account. So I fork the merge-conflict repo provided by intellipaat only for method one. Here in method 2 using mergetool I not use fork of the intellipaat repo.

```
# mkdir meargetool_method
# cd meargetool_method
# git init
# git clone https://github.com/devops-intellipaat/merge-conflict.git
# cd merge-conflict
# ls
```



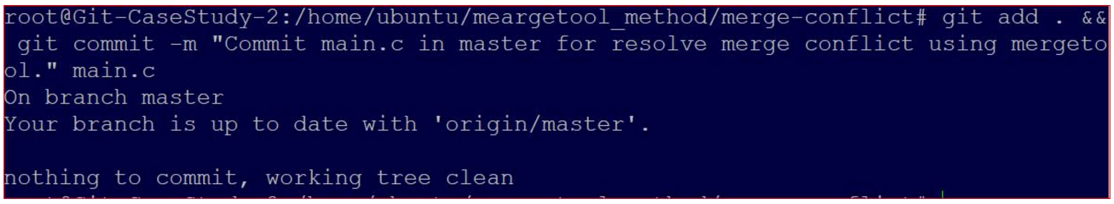
```
root@Git-CaseStudy-2:/home/ubuntu# mkdir meargetool_method
root@Git-CaseStudy-2:/home/ubuntu# cd meargetool_method
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method# git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ubuntu/meargetool_method/.git/
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method# git clone https://github.co
m/devops-intellipaat/merge-conflict.git
Cloning into 'merge-conflict'...
remote: Enumerating objects: 15, done.
remote: Total 15 (delta 0), reused 0 (delta 0), pack-reused 15
Receiving objects: 100% (15/15), done.
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method# cd merge-conflict
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# ls
main.c
```

# git checkout master
# cat main.c



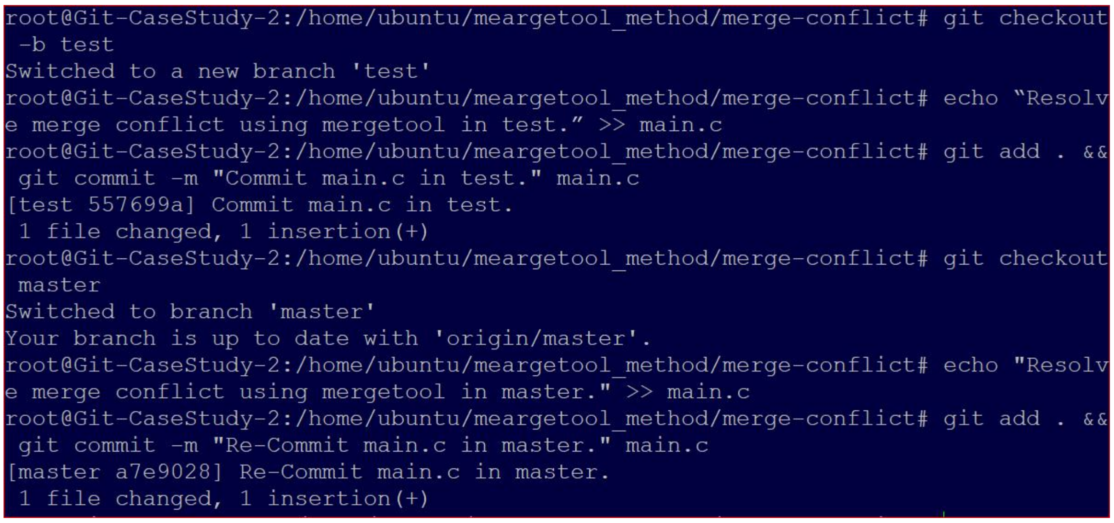
```
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git checkout
 master
Already on 'master'
Your branch is up to date with 'origin/master'.
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# cat main.c
pseudo code
{
Initial Function()
{
  //Sample Code
}

Initial Security()
{
 //Sample Code
}



}
```

# git add . && git commit -m “Commit main.c in master for resolve merge conflict using mergetool.” main.c



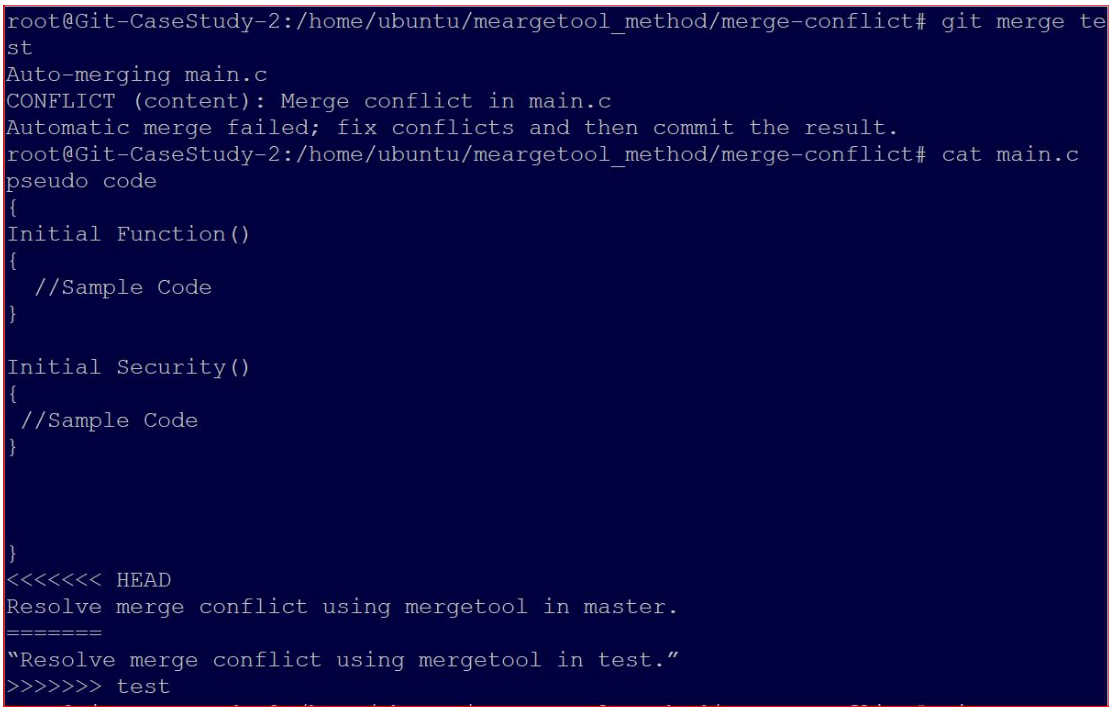
```
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git add . &&
 git commit -m "Commit main.c in master for resolve merge conflict using mergeto
ol." main.c
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

# git checkout -b test
# echo “Resolve merge conflict using mergetool in test.” >> main.c
# git add . && git commit -m “Commit main.c in test.” main.c
# git checkout master
# echo “Resolve merge conflict using mergetool in master.” >> main.c
# git add . && git commit -m “Re-Commit main.c in master.” main.c



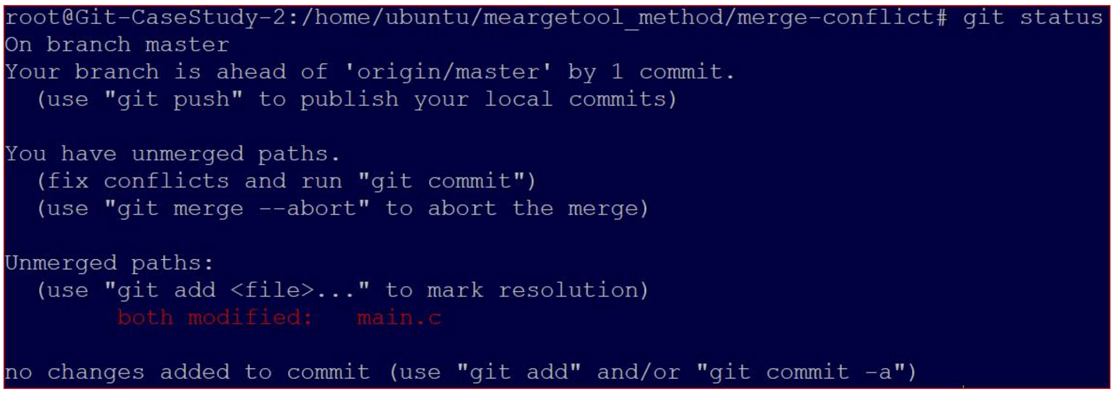
```
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git checkout
 -b test
Switched to a new branch 'test'
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# echo “Resolv
e merge conflict using mergetool in test.” >> main.c
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git add . &&
 git commit -m "Commit main.c in test." main.c
[test 557699a] Commit main.c in test.
 1 file changed, 1 insertion(+)
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git checkout
 master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# echo "Resolv
e merge conflict using mergetool in master." >> main.c
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git add . &&
 git commit -m "Re-Commit main.c in master." main.c
[master a7e9028] Re-Commit main.c in master.
 1 file changed, 1 insertion(+)
```

# git merge test
# cat main.c



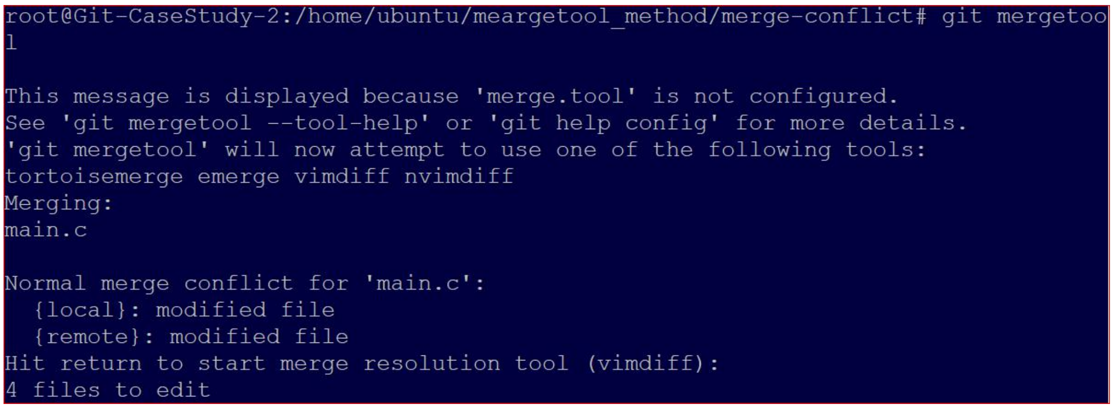
```
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git merge te
st
Auto-merging main.c
CONFLICT (content): Merge conflict in main.c
Automatic merge failed; fix conflicts and then commit the result.
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# cat main.c
pseudo code
{
Initial Function()
{
  //Sample Code
}

Initial Security()
{
 //Sample Code
}



}
<<<<<<< HEAD
Resolve merge conflict using mergetool in master.
=======
“Resolve merge conflict using mergetool in test.”
>>>>>>> test
```

# git status



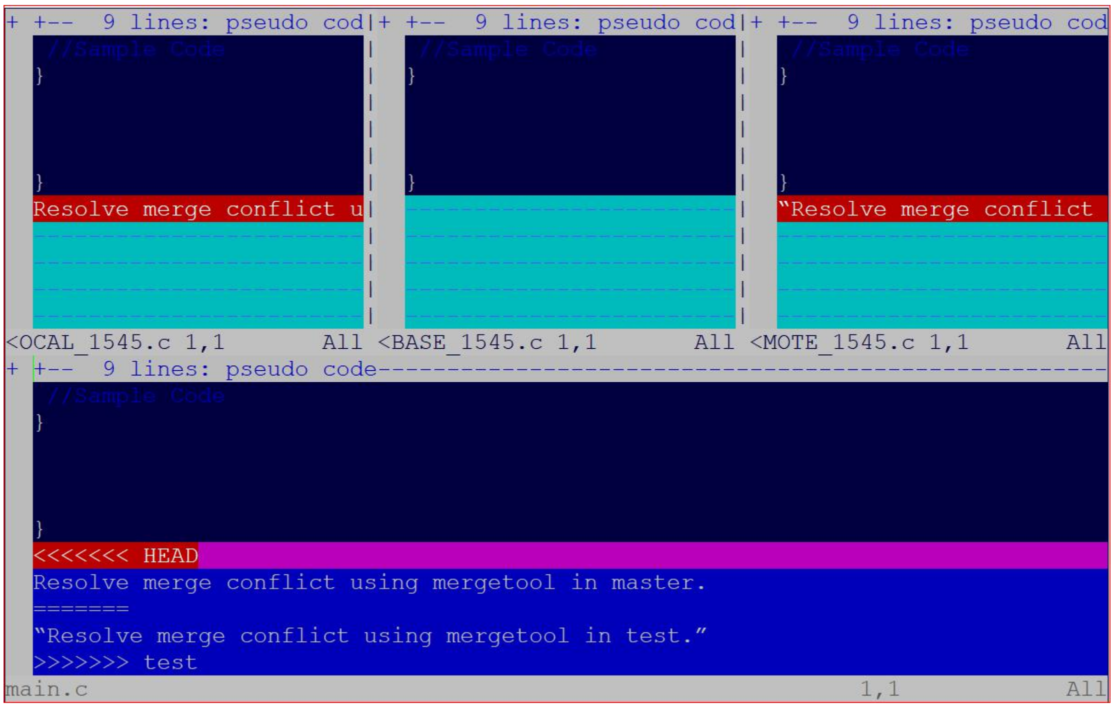
```
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   main.c

no changes added to commit (use "git add" and/or "git commit -a")
```

# git mergetool
Note:- [# git config --global merge.tool vimdiff] command also can be used.
* vimdiff is used to display multiple screens of multiple files in one screen.



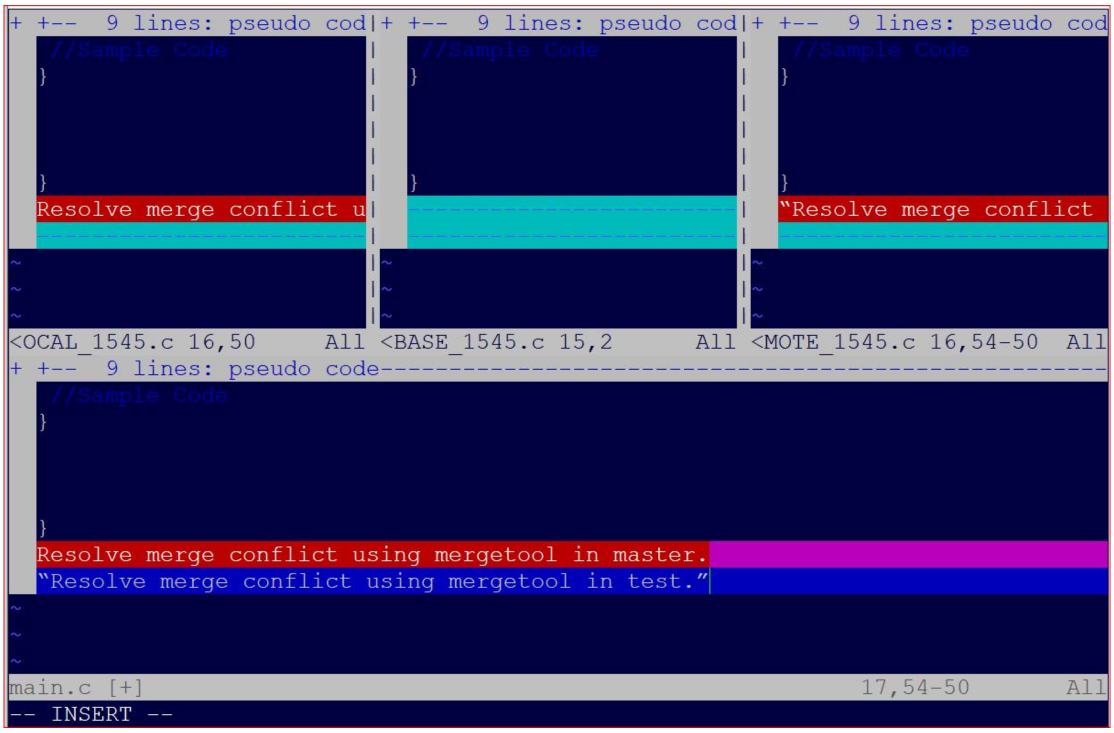
```
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git mergetoo
l

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
tortoisemerge emerge vimdiff nvimdiff
Merging:
main.c

Normal merge conflict for 'main.c':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit
```

* Now press i



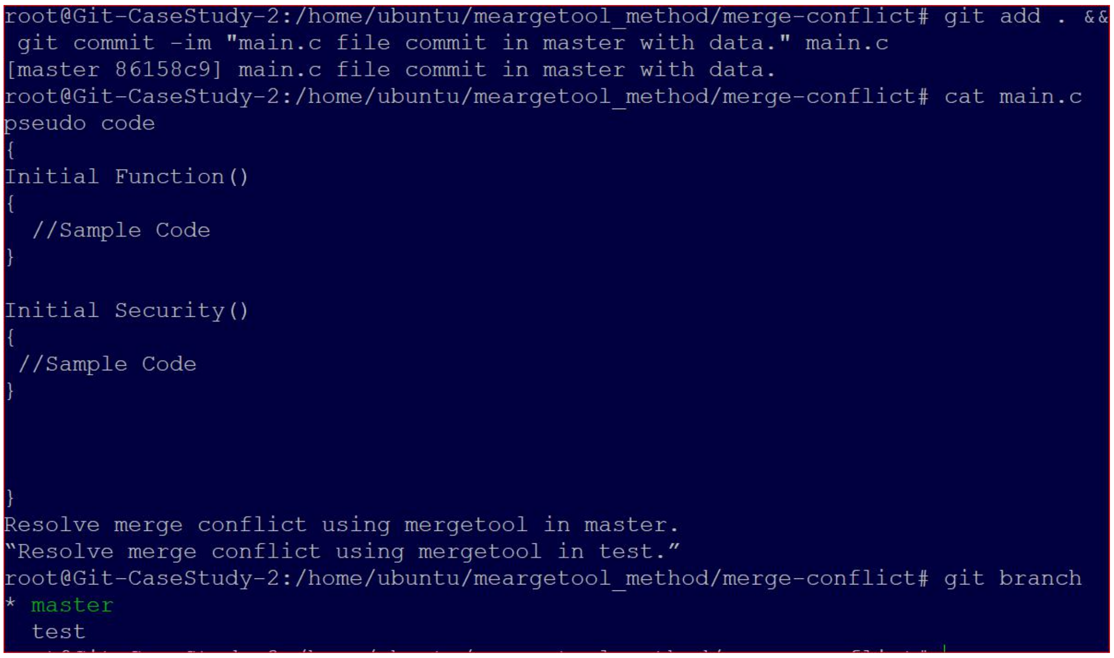
```
+ +--  9 lines: pseudo cod|+ +--  9 lines: pseudo cod|+ +--  9 lines: pseudo cod
   //Sample Code          |   //Sample Code          |   //Sample Code
  }                       |  }                       |  }
                          |                          |
  }                       |  }                       |  }
  Resolve merge conflict u|                          |  "Resolve merge conflict
<OCAL_1545.c 1,1       All <BASE_1545.c 1,1       All <MOTE_1545.c 1,1       All
+ +--  9 lines: pseudo code-------------------------------------------------------
   //Sample Code
  }

  }
  <<<<<<< HEAD
  Resolve merge conflict using mergetool in master.
  =======
  "Resolve merge conflict using mergetool in test."
  >>>>>>> test
main.c                                                        1,1            All
```

* And remove the unwanted lines from main.c file. When you remove the line from main.c file then other files's lines are auto removed.



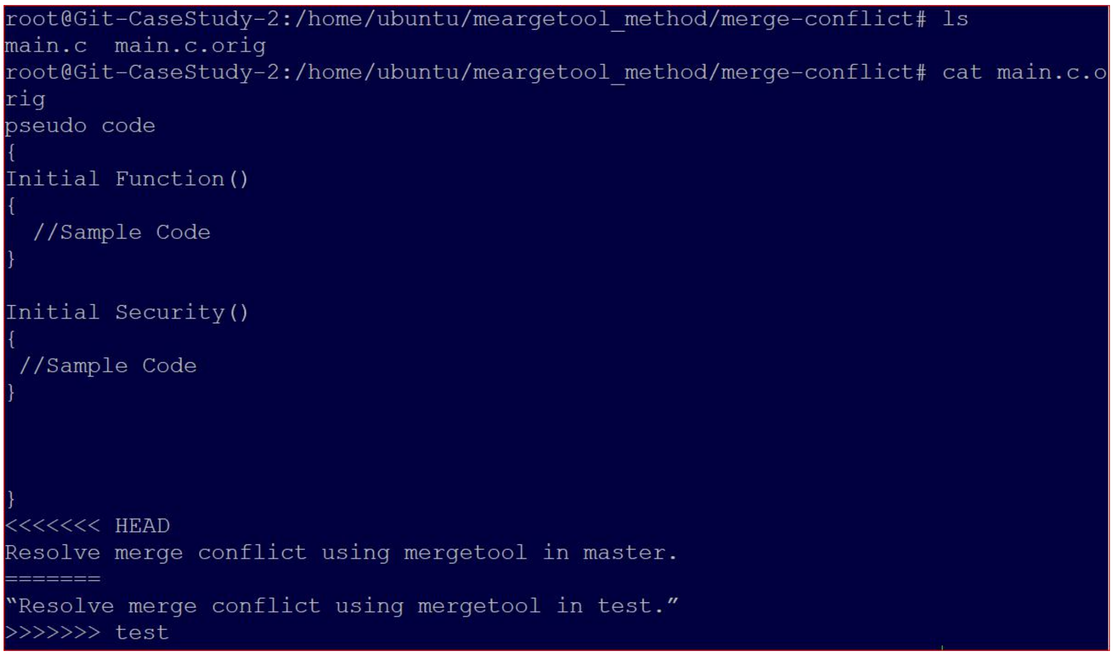
```
+ +--  9 lines: pseudo cod|+ +--  9 lines: pseudo cod|+ +--  9 lines: pseudo cod
   //Sample Code           |   //Sample Code           |   //Sample Code
  }                        |  }                        |  }



  }                        |  }                        |  }
  Resolve merge conflict u |  -------------------------|  "Resolve merge conflict
  -------------------------|  -------------------------|  ------------------------
~                          |~                          |~
~                          |~                          |~
~                          |~                          |~
<OCAL_1545.c 16,50      All <BASE_1545.c 15,2       All <MOTE_1545.c 16,54-50  All
+ +--  9 lines: pseudo code-----------------------------------------------------
   //Sample Code
  }



  }
  Resolve merge conflict using mergetool in master.
  "Resolve merge conflict using mergetool in test."
~
~
~
main.c [+]                                                    17,54-50      All
-- INSERT --
```

# git add . && git commit -m "main.c file commit in master with data" main.c
# cat main.c
# git branch



```
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git add . &&
 git commit -im "main.c file commit in master with data." main.c
[master 86158c9] main.c file commit in master with data.
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# cat main.c
pseudo code
{
Initial Function()
{
  //Sample Code
}

Initial Security()
{
 //Sample Code
}




}
Resolve merge conflict using mergetool in master.
"Resolve merge conflict using mergetool in test."
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# git branch
* master
  test
```

ls
# cat main.c.orig (This file is created automatically and contain backup of original merge file conflict.)



```
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# ls
main.c  main.c.orig
root@Git-CaseStudy-2:/home/ubuntu/meargetool_method/merge-conflict# cat main.c.o
rig
pseudo code
{
Initial Function()
{
  //Sample Code
}

Initial Security()
{
 //Sample Code
}



}
<<<<<<< HEAD
Resolve merge conflict using mergetool in master.
=======
"Resolve merge conflict using mergetool in test."
>>>>>>> test
```