Module 2: Git Assignment - 1

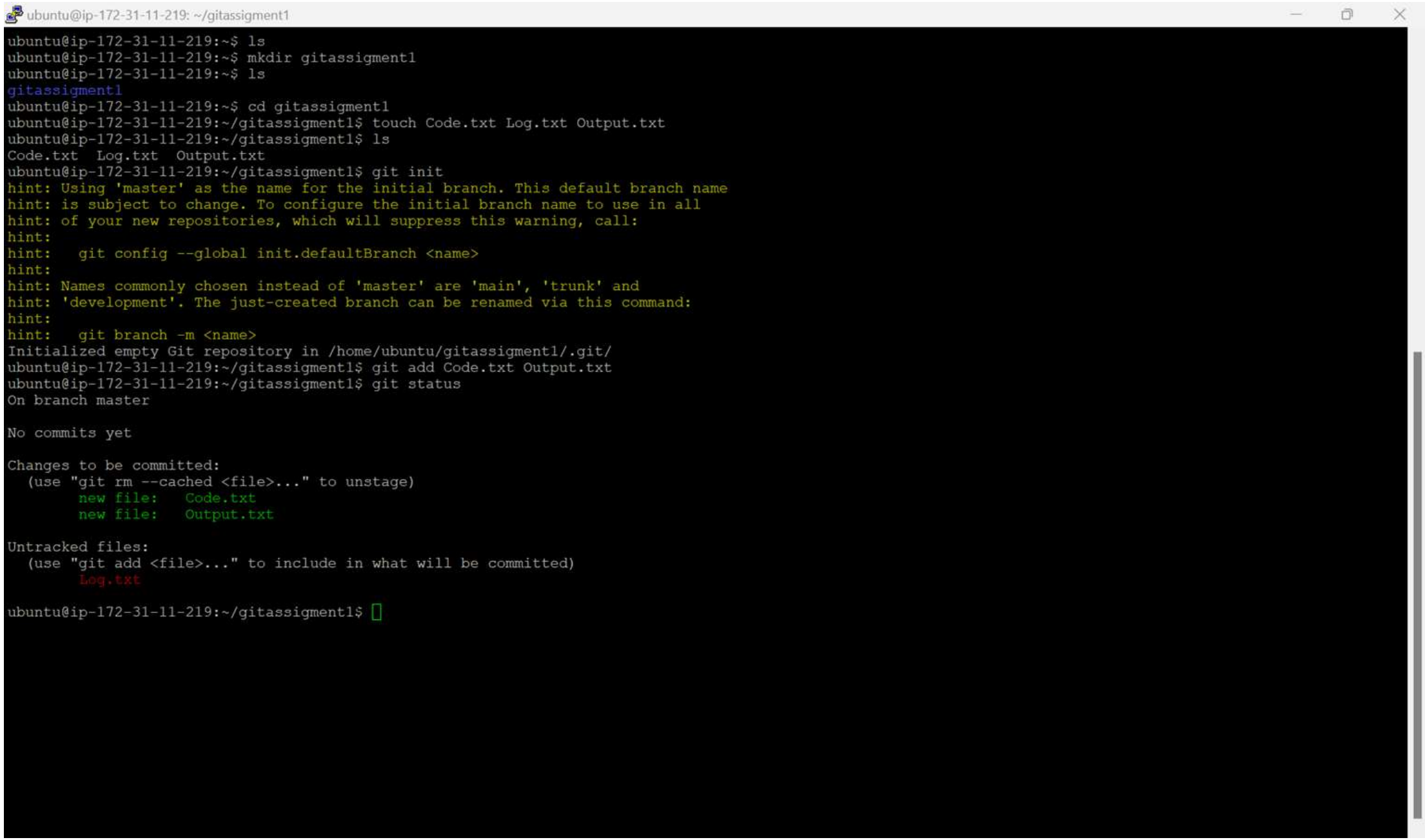
Tasks To Be Performed:

1. Based on what you have learnt in the class, do the following steps:
    a. Create a new folder
    b. Put the following files in the folder
        - Code.txt
        - Log.txt
        - Output.txt

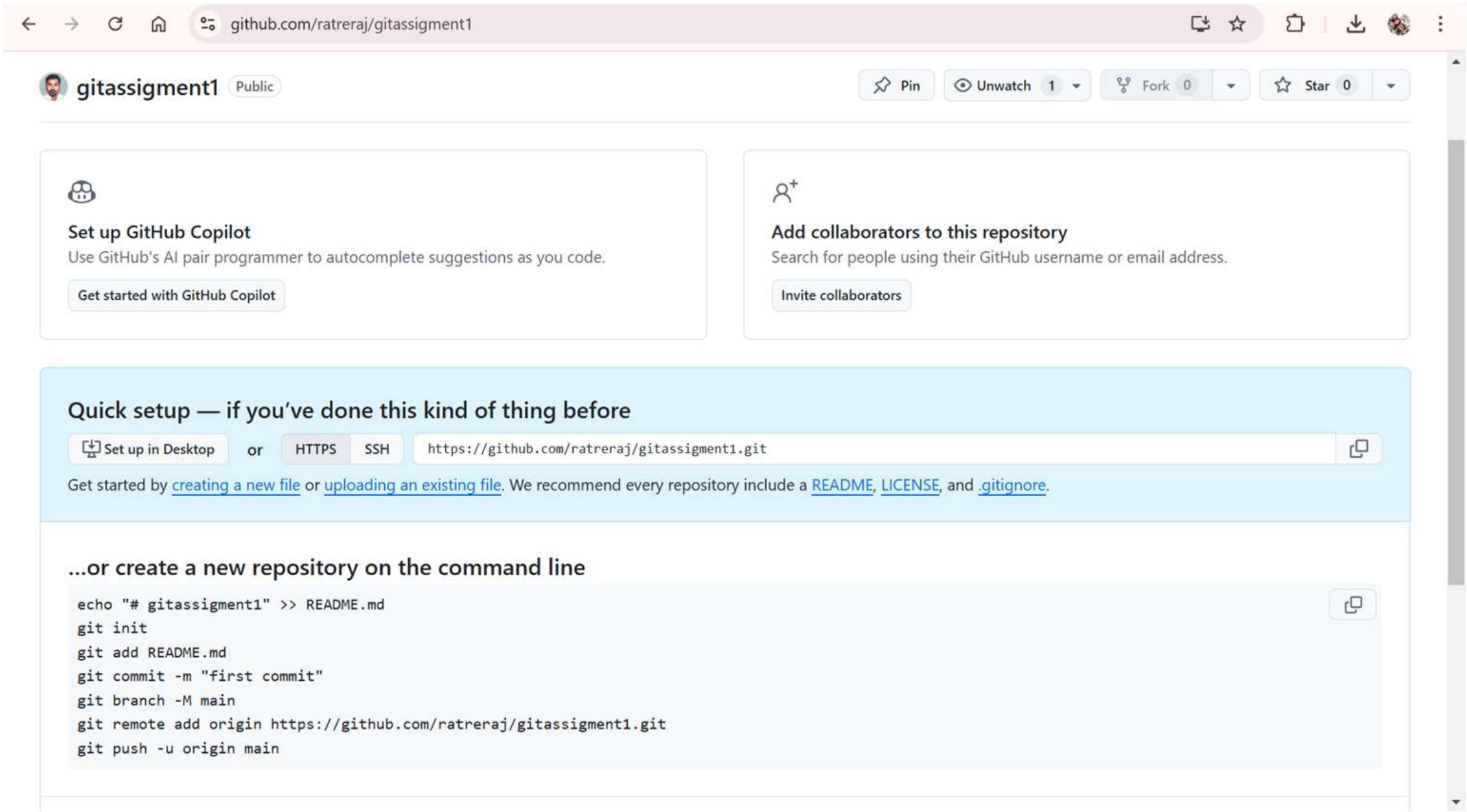
    c. Stage the Code.txt and Output.txt files
    d. Commit them
    e. And finally push them to GitHub

2. Please share the commands for the above points

Output


ubuntu@ip-172-31-11-219: ~/gitassigment1
ubuntu@ip-172-31-11-219:~$ ls
ubuntu@ip-172-31-11-219:~$ mkdir gitassigment1
ubuntu@ip-172-31-11-219:~$ ls
gitassigment1
ubuntu@ip-172-31-11-219:~$ cd gitassigment1
ubuntu@ip-172-31-11-219:~/gitassigment1$ touch Code.txt Log.txt Output.txt
ubuntu@ip-172-31-11-219:~/gitassigment1$ ls
Code.txt Log.txt Output.txt
ubuntu@ip-172-31-11-219:~/gitassigment1$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
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/gitassigment1/.git/
ubuntu@ip-172-31-11-219:~/gitassigment1$ git add Code.txt Output.txt
ubuntu@ip-172-31-11-219:~/gitassigment1$ git status
On branch master
No commits yet
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
new file: Code.txt
new file: Output.txt
Untracked files:
(use "git add <file>..." to include in what will be committed)
Log.txt
ubuntu@ip-172-31-11-219:~/gitassigment1$

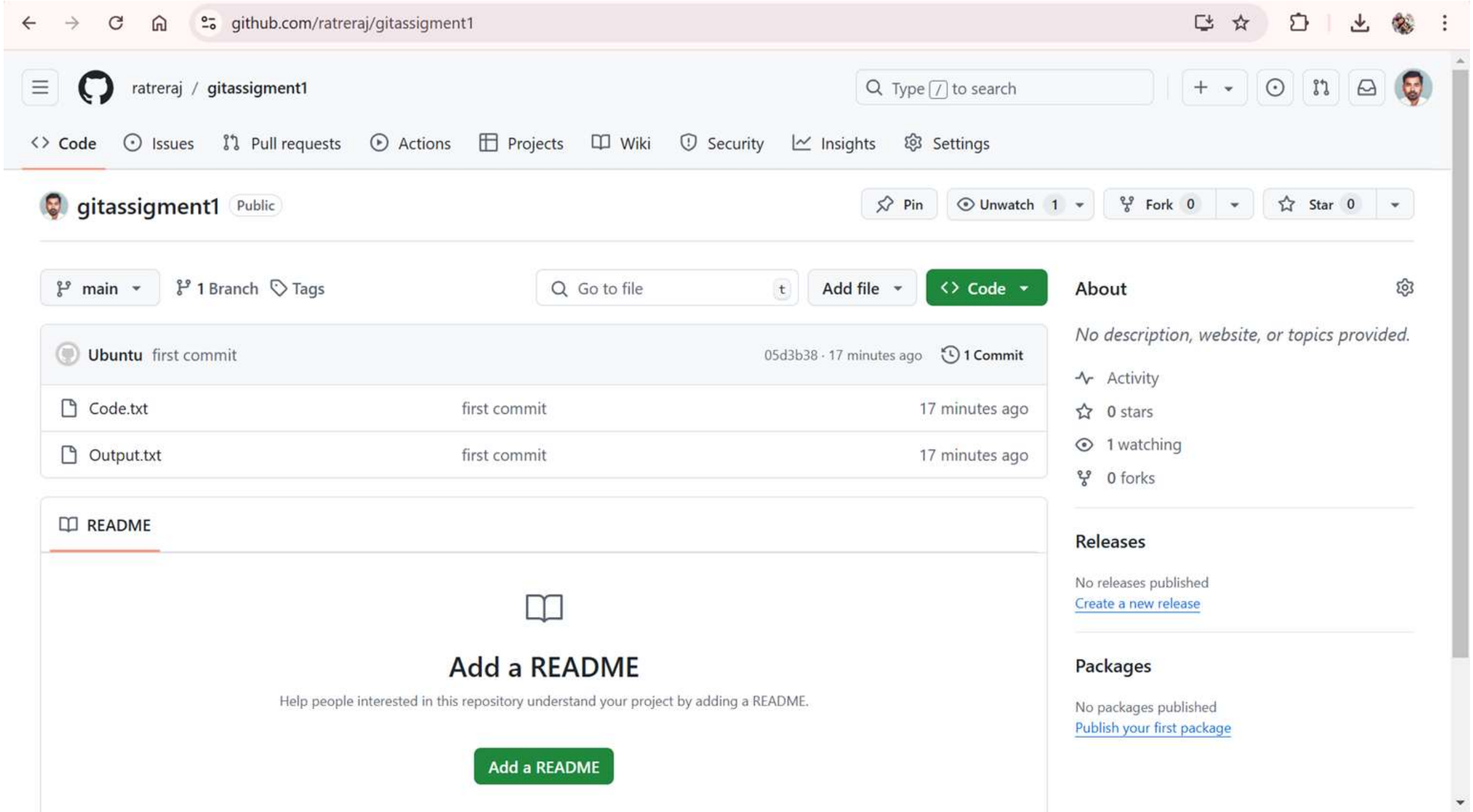
GitHub Output



List of commands

1. git –version
2. git config --global user.name "ratreraj"
3. git config --global user.email "ratre.raj@gmail.com"
4. git config --global core.editor "nano"
5. mkdir gitassigment1
6. ls
7. cd gitassigment1
8. touch Code.txt Log.txt Output.txt
9. ls
10. git init
11. git add Code.txt Output.txt
12. git status
13. git commit -m "first commit“
14. git branch -M main
15. git remote add origin https://github.com/ratreraj/gitassigment1.git
16. git branch
17. git push -u origin main

GitHub Output



Module 2: Git Assignment - 2

Tasks To Be Performed:

1. Create a Git working directory with feature1.txt and feature2.txt in the master branch
2. Create 3 branches develop, feature1 and feature2
3. In develop branch create develop.txt, do not stage or commit it
4. Stash this file and check out to feature1 branch
5. Create new.txt file in feature1 branch, stage and commit this file
6. Checkout to develop, unstash this file and commit
7. Please submit all the Git commands used to do the above steps

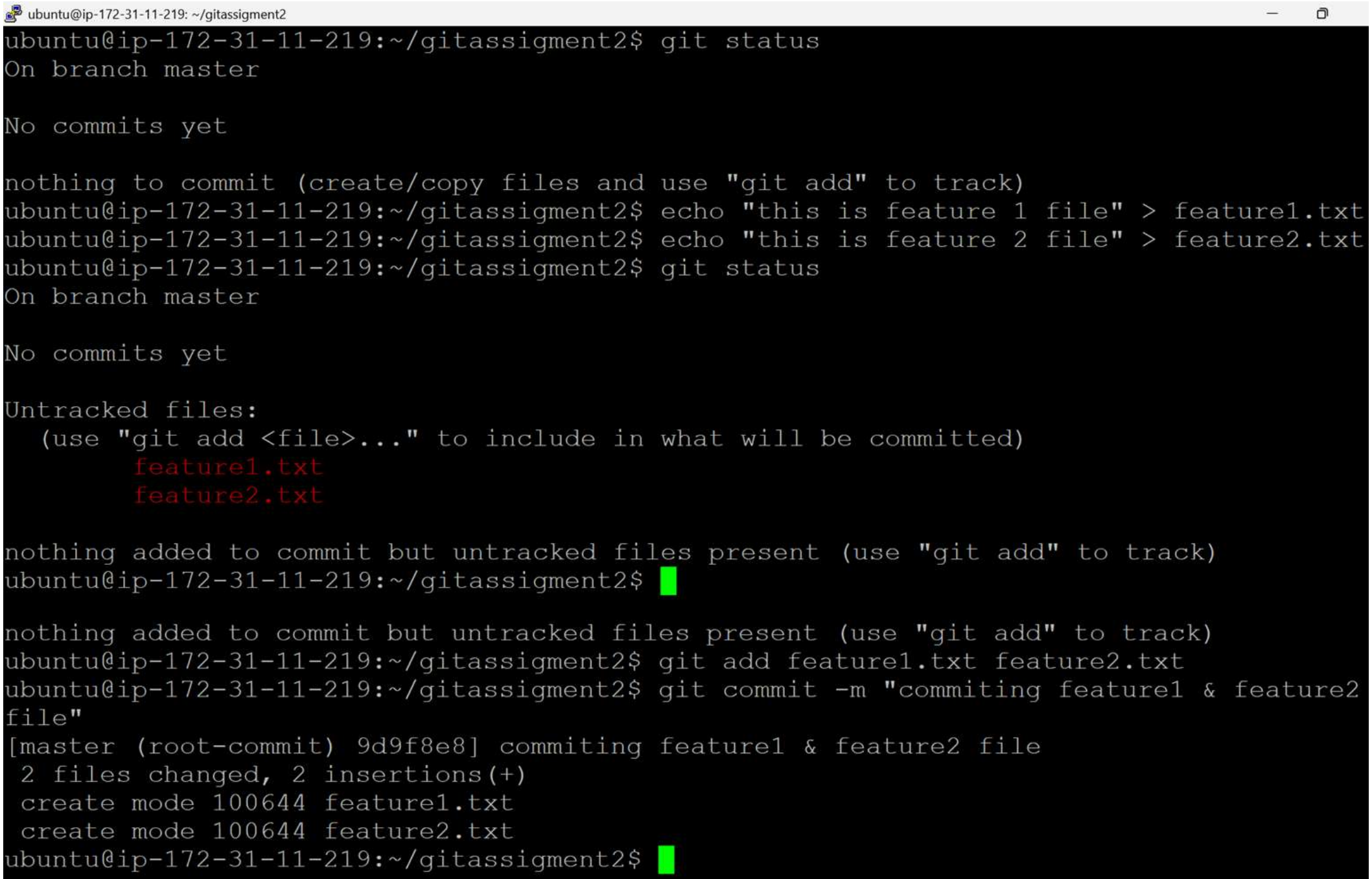
Step 1: Set Up the Working Directory and Create Files in the master Branch

```
# Initialize a new Git repository
git init

# Create feature1.txt and feature2.txt in the master branch
echo "Content for feature1" > feature1.txt
echo "Content for feature2" > feature2.txt

# Stage and commit these files
git add feature1.txt feature2.txt
git commit -m "Add feature1.txt and feature2.txt"
```

Output


ubuntu@ip-172-31-11-219: ~/gitassigment2
ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch master
No commits yet
nothing to commit (create/copy files and use "git add" to track)
ubuntu@ip-172-31-11-219:~/gitassigment2$ echo "this is feature 1 file" > feature1.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ echo "this is feature 2 file" > feature2.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch master
No commits yet
Untracked files:
(use "git add <file>..." to include in what will be committed)
feature1.txt
feature2.txt
nothing added to commit but untracked files present (use "git add" to track)
ubuntu@ip-172-31-11-219:~/gitassigment2$
nothing added to commit but untracked files present (use "git add" to track)
ubuntu@ip-172-31-11-219:~/gitassigment2$ git add feature1.txt feature2.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ git commit -m "commiting feature1 & feature2 file"
[master (root-commit) 9d9f8e8] commiting feature1 & feature2 file
2 files changed, 2 insertions(+)
create mode 100644 feature1.txt
create mode 100644 feature2.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$

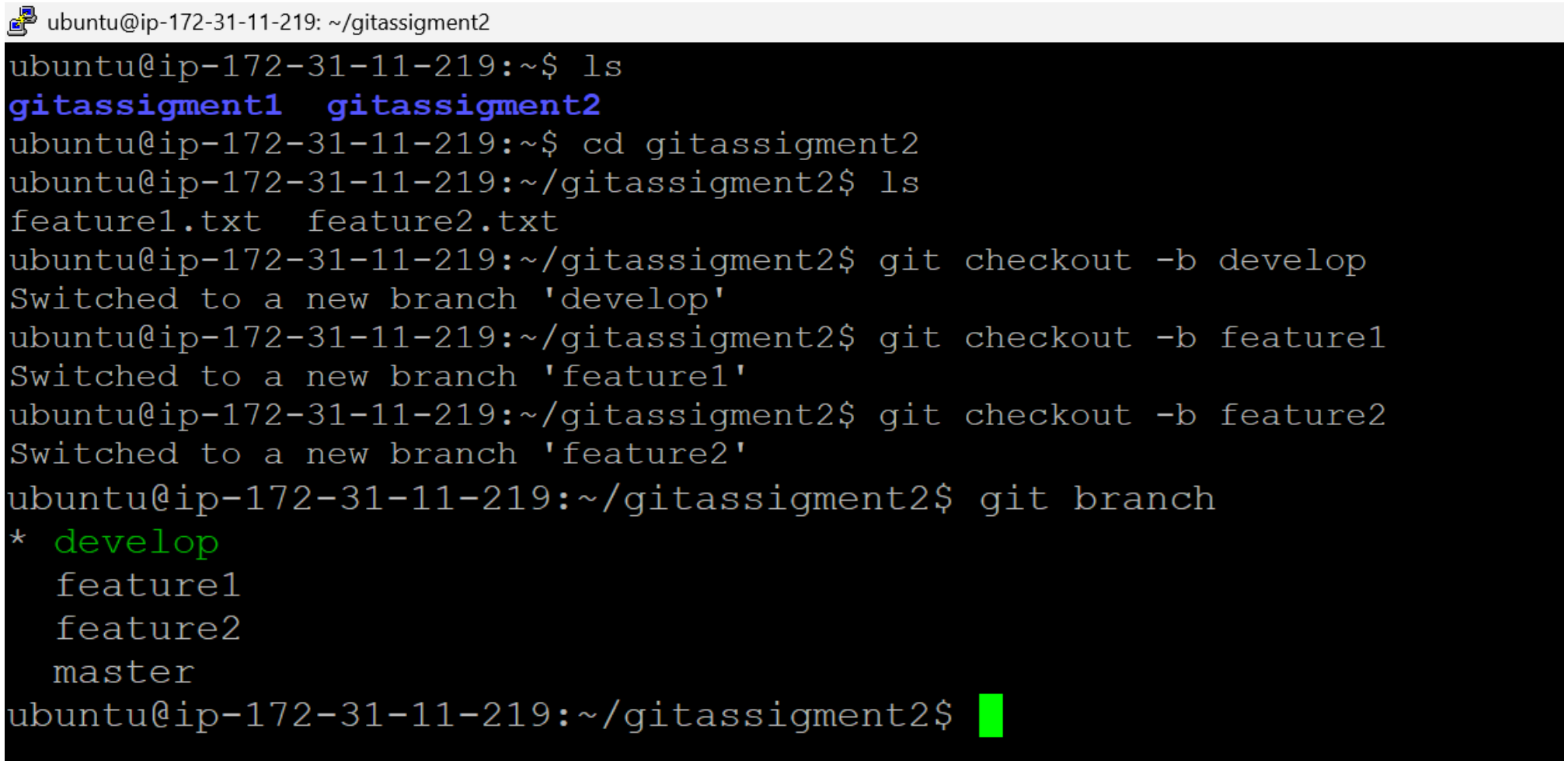
Step 2: Create the develop, feature1, and feature2 Branches

```
# Create and push develop branch
git checkout -b develop

# Create feature1 and feature2 branches from develop
git checkout -b feature1
git checkout -b feature2

# Go back to the develop branch to continue
git checkout develop
```

Output


ubuntu@ip-172-31-11-219: ~/gitassigment2
ubuntu@ip-172-31-11-219:~$ ls
gitassigment1 gitassigment2
ubuntu@ip-172-31-11-219:~$ cd gitassigment2
ubuntu@ip-172-31-11-219:~/gitassigment2$ ls
feature1.txt feature2.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ git checkout -b develop
Switched to a new branch 'develop'
ubuntu@ip-172-31-11-219:~/gitassigment2$ git checkout -b feature1
Switched to a new branch 'feature1'
ubuntu@ip-172-31-11-219:~/gitassigment2$ git checkout -b feature2
Switched to a new branch 'feature2'
ubuntu@ip-172-31-11-219:~/gitassigment2$ git branch
* develop
  feature1
  feature2
  master
ubuntu@ip-172-31-11-219:~/gitassigment2$

Step 3: Create develop.txt in the develop Branch Without Staging or Committing It

```
# Create develop.txt in the develop branch
echo "Content for develop" > develop.txt
```

Output

```
ubuntu@ip-172-31-11-219:~/gitassigment2$ git branch
* develop
  feature1
  feature2
  master
ubuntu@ip-172-31-11-219:~/gitassigment2$ echo "this is develop branch" > develop.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ ls
develop.txt  feature1.txt  feature2.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$
```

```
ubuntu@ip-172-31-11-219:~/gitassigment2$ git branch
* develop
  feature1
  feature2
  master
ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        develop.txt

nothing added to commit but untracked files present (use "git add" to track)
ubuntu@ip-172-31-11-219:~/gitassigment2$
```

Step 4: Stash develop.txt and Check Out to the feature1 Branch

```
# Stash the uncommitted develop.txt
git stash

# Switch to feature1 branch
git checkout feature1
```

Output

```
ubuntu@ip-172-31-11-219:~/gitassigment2$ echo "new file in dev branch" > dev2.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ git add dev2.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   dev2.txt

ubuntu@ip-172-31-11-219:~/gitassigment2$ git stash
Saved working directory and index state WIP on develop: 9d9f8e8 commiting feature1 & feature2 file
ubuntu@ip-172-31-11-219:~/gitassigment2$
```

Output

ubuntu@ip-172-31-11-219: ~/gitassigment2

```
ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        develop.txt

nothing added to commit but untracked files present (use "git add" to track)
ubuntu@ip-172-31-11-219:~/gitassigment2$ git add develop.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   develop.txt

ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   develop.txt

ubuntu@ip-172-31-11-219:~/gitassigment2$ git checkout feature1
A       develop.txt
Switched to branch 'feature1'
ubuntu@ip-172-31-11-219:~/gitassigment2$ git branch
  develop
* feature1
  feature2
  master
ubuntu@ip-172-31-11-219:~/gitassigment2$
```

Step 5: Create and Commit new.txt in the feature1 Branch

```
# Create new.txt in feature1 branch
echo "Content for new feature" > new.txt

# Stage and commit new.txt
git add new.txt
git commit -m "Add new.txt in feature1 branch"
```

Output

```
ubuntu@ip-172-31-11-219: ~/gitassigment2
ubuntu@ip-172-31-11-219:~/gitassigment2$ git branch
  develop
* feature1
  feature2
  master
ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch feature1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   develop.txt

ubuntu@ip-172-31-11-219:~/gitassigment2$ echo "the new content" > new.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch feature1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   develop.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new.txt

ubuntu@ip-172-31-11-219:~/gitassigment2$ git add new.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ git commit -m "commiting new.txt file to feature branch"
[feature1 8c5ecaf] commiting new.txt file to feature branch
 2 files changed, 2 insertions(+)
 create mode 100644 develop.txt
 create mode 100644 new.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$ git status
On branch feature1
nothing to commit, working tree clean
ubuntu@ip-172-31-11-219:~/gitassigment2$
```

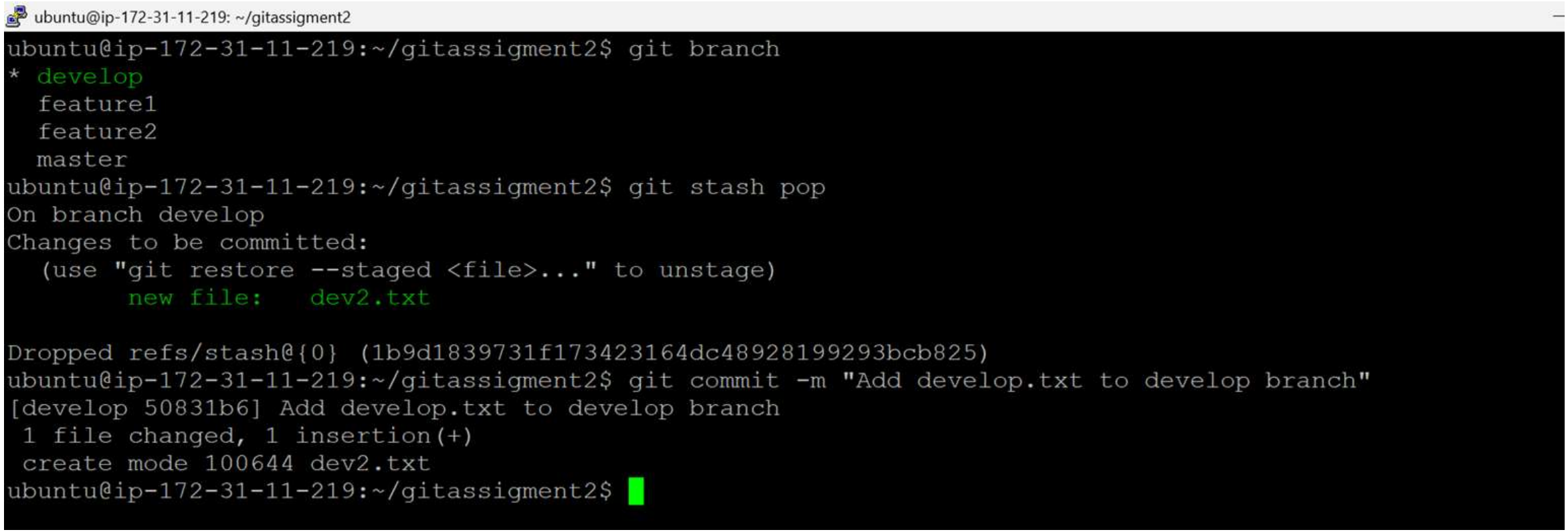
Step 6: Switch Back to develop, Unstash develop.txt, and Commit It

```
# Switch back to the develop branch
git checkout develop

# Unstash to recover develop.txt
git stash pop

# Stage and commit develop.txt
git commit -m "Add develop.txt to develop branch"
```

Output


ubuntu@ip-172-31-11-219: ~/gitassigment2
ubuntu@ip-172-31-11-219:~/gitassigment2$ git branch
* develop
feature1
feature2
master
ubuntu@ip-172-31-11-219:~/gitassigment2$ git stash pop
On branch develop
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file: dev2.txt
Dropped refs/stash@{0} (1b9d1839731f173423164dc48928199293bcb825)
ubuntu@ip-172-31-11-219:~/gitassigment2$ git commit -m "Add develop.txt to develop branch"
[develop 50831b6] Add develop.txt to develop branch
1 file changed, 1 insertion(+)
create mode 100644 dev2.txt
ubuntu@ip-172-31-11-219:~/gitassigment2$

Module 2: Git Assignment - 3

Tasks To Be Performed:

1. Create a Git working directory, with the following branches:
    - Develop
    - F1
    - f2

2. In the master branch, commit main.txt file
3. Put develop.txt in develop branch, f1.txt and f2.txt in f1 and f2 respectively
4. Push all these branches to GitHub
5. On local delete f2 branch
6. Delete the same branch on GitHub as well

Step 1: Set Up a Git Working Directory and Create Branches

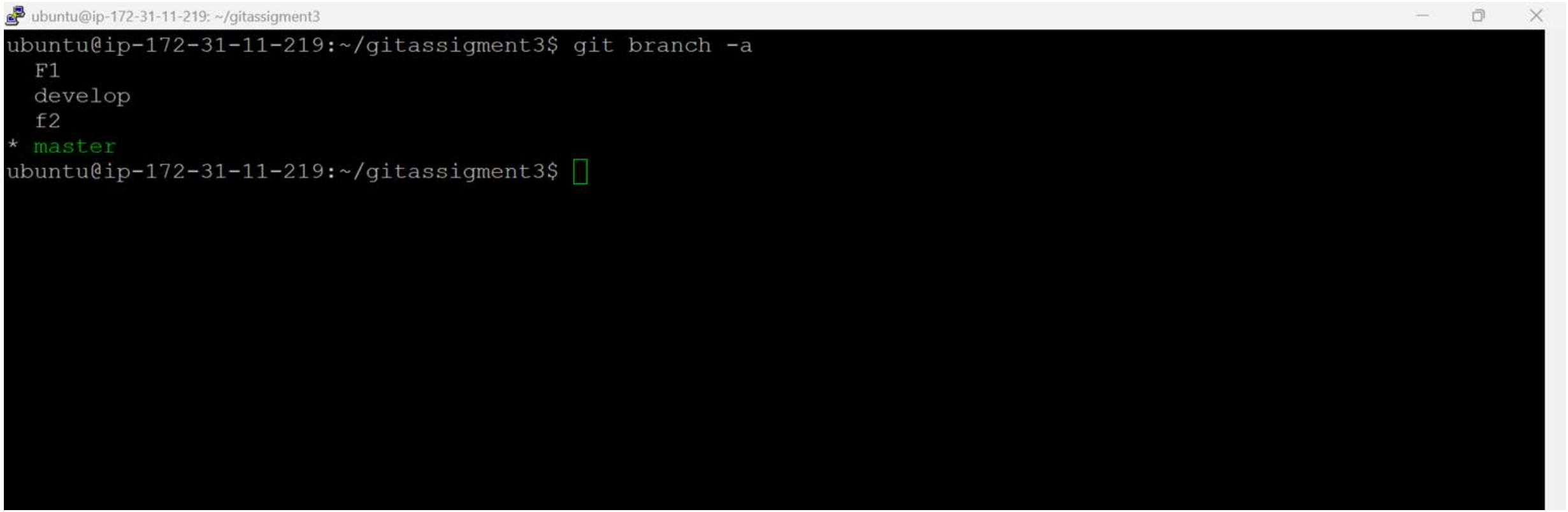
```
# Create a new directory and navigate into it
mkdir gitproject
cd gitproject

# Initialize Git repository
touch sample.txt
git init
git add .
git commit –m “this first commit”


# Create and switch to develop, F1, and f2 branches
git branch develop
git branch F1
git branch f2
Git branch -a

# Switch back to the master branch to start
git checkout master
```

Output


ubuntu@ip-172-31-11-219: ~/gitassigment3
ubuntu@ip-172-31-11-219:~/gitassigment3$ git branch -a
F1
develop
f2
* master
ubuntu@ip-172-31-11-219:~/gitassigment3$

Step 2: Commit main.txt in the Master Branch

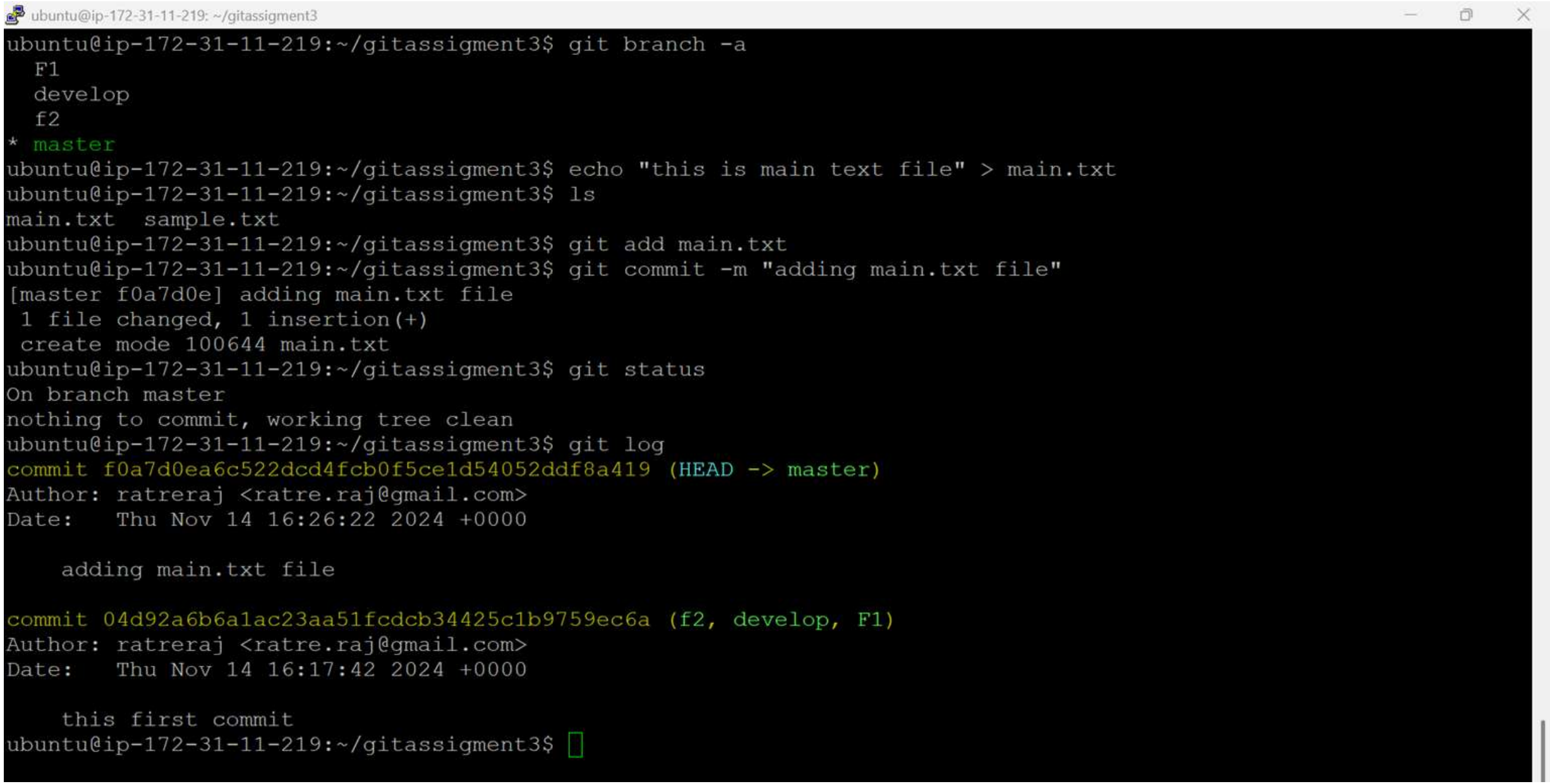
```
# Create main.txt file and commit it

echo "This is the main file" > main.txt

git add main.txt
git commit -m "adding main.txt file"

git status
git log
```

Output


ubuntu@ip-172-31-11-219: ~/gitassigment3
ubuntu@ip-172-31-11-219:~/gitassigment3$ git branch -a
F1
develop
f2
* master
ubuntu@ip-172-31-11-219:~/gitassigment3$ echo "this is main text file" > main.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ ls
main.txt sample.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git add main.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git commit -m "adding main.txt file"
[master f0a7d0e] adding main.txt file
1 file changed, 1 insertion(+)
create mode 100644 main.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git status
On branch master
nothing to commit, working tree clean
ubuntu@ip-172-31-11-219:~/gitassigment3$ git log
commit f0a7d0ea6c522dcd4fcb0f5ce1d54052ddf8a419 (HEAD -> master)
Author: ratreraj <ratre.raj@gmail.com>
Date: Thu Nov 14 16:26:22 2024 +0000
adding main.txt file
commit 04d92a6b6a1ac23aa51fcdcb34425c1b9759ec6a (f2, develop, F1)
Author: ratreraj <ratre.raj@gmail.com>
Date: Thu Nov 14 16:17:42 2024 +0000
this first commit
ubuntu@ip-172-31-11-219:~/gitassigment3$

Step 3: Add Specific Files to Each Branch and Commit Them

```
# Add develop.txt in develop branch
git checkout develop
echo "this develop branch" > develop.txt
git add develop.txt
git commit -m "added develop.txt file"

# Add f1.txt in F1 branch
git checkout F1
echo "this F1  branch" > F1.txt
git add f1.txt
git commit -m "adding F1.txt file“

# Add f2.txt in f2 branch
git checkout f2
 echo "this is f2 branch" > f2.txt
git add f2.txt
git commit -m "adding f2.txt file"
```

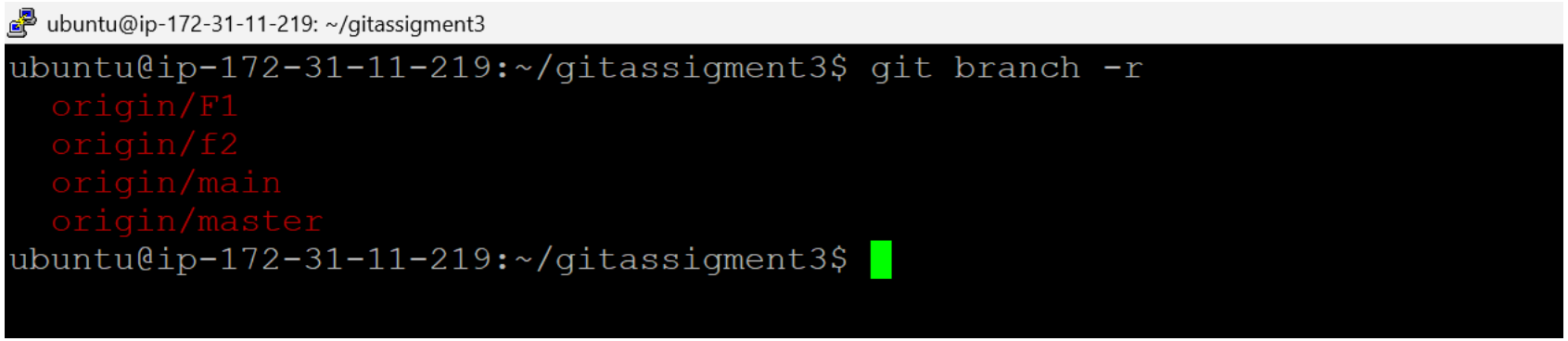
Output

```
ubuntu@ip-172-31-11-219: ~/gitassigment3
ubuntu@ip-172-31-11-219:~/gitassigment3$ git checkout develop
Switched to branch 'develop'
ubuntu@ip-172-31-11-219:~/gitassigment3$ git branch -a
  F1
* develop
  f2
  master
ubuntu@ip-172-31-11-219:~/gitassigment3$ echo "this develop branch" > develop.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git add develop.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git commit -m "added develop.txt file"
[develop c00453d] added develop.txt file
 1 file changed, 1 insertion(+)
 create mode 100644 develop.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git checkout F1
Switched to branch 'F1'
ubuntu@ip-172-31-11-219:~/gitassigment3$ echo "this F1  branch" > F1.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git add F1.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git commit -m "adding F1.txt file"
[F1 401c67c] adding F1.txt file
 1 file changed, 1 insertion(+)
 create mode 100644 F1.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git checkout f2
Switched to branch 'f2'
ubuntu@ip-172-31-11-219:~/gitassigment3$ echo "this is f2 branch" > f2.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git add f2.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$ git commit -m "adding f2.txt file"
[f2 4e09e76] adding f2.txt file
 1 file changed, 1 insertion(+)
 create mode 100644 f2.txt
ubuntu@ip-172-31-11-219:~/gitassigment3$
```

Step 4: Push All Branches to GitHub

```
git branch -M main
git remote add origin https://github.com/ratreraj/gitassigment3.git
git push -u origin main
git branch
git push -u origin master
git push
git checkout F1
git push -u origin F1
git checkout f2
git push -u origin f2
```

Output



Using single command Push All Branches Using –all

```
git push --all origin
```

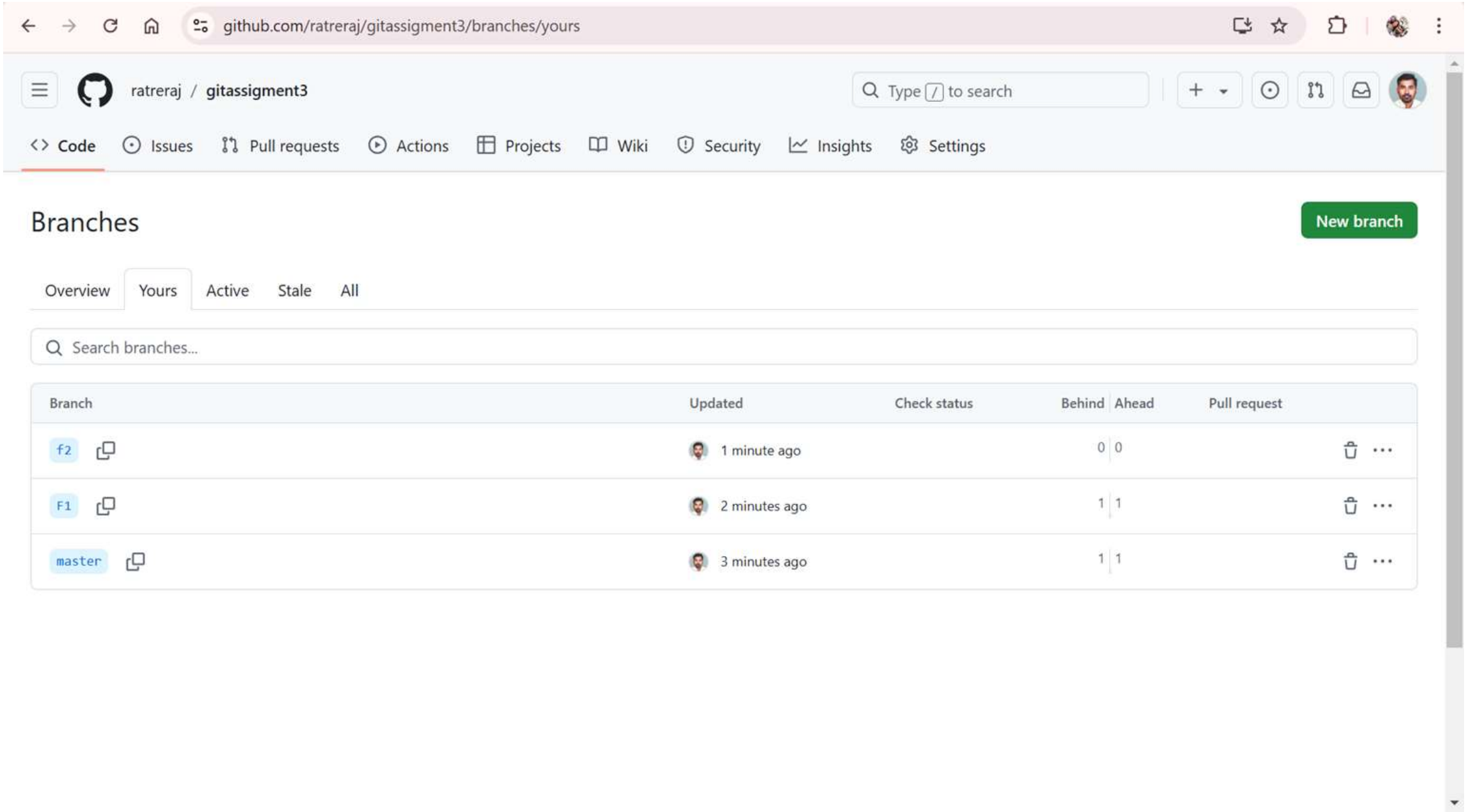
Output

```
ubuntu@ip-172-31-11-219:~/gitassigment3$ git branch -M main
ubuntu@ip-172-31-11-219:~/gitassigment3$ git remote add origin https://github.com/ratreraj/gitassigment3.git
ubuntu@ip-172-31-11-219:~/gitassigment3$ git push -u origin main
Username for 'https://github.com': ratreraj
Password for 'https://ratreraj@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for
information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/ratreraj/gitassigment3.git/'
ubuntu@ip-172-31-11-219:~/gitassigment3$ git push -u origin main
Username for 'https://github.com': ratreraj
Password for 'https://ratreraj@github.com':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 466 bytes | 466.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ratreraj/gitassigment3.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
ubuntu@ip-172-31-11-219:~/gitassigment3$
```

Output

```
ubuntu@ip-172-31-11-219:~/gitassigment3$ git checkout F1
Switched to branch 'F1'
ubuntu@ip-172-31-11-219:~/gitassigment3$ git push -u origin F1
Username for 'https://github.com': ratreraj
Password for 'https://ratreraj@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 289 bytes | 289.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'F1' on GitHub by visiting:
remote:      https://github.com/ratreraj/gitassigment3/pull/new/F1
remote:
To https://github.com/ratreraj/gitassigment3.git
 * [new branch]      F1 -> F1
branch 'F1' set up to track 'origin/F1'.
ubuntu@ip-172-31-11-219:~/gitassigment3$ git checkout f2
Switched to branch 'f2'
ubuntu@ip-172-31-11-219:~/gitassigment3$ git push -u origin f2
Username for 'https://github.com': ratreraj
Password for 'https://ratreraj@github.com':
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'f2' on GitHub by visiting:
remote:      https://github.com/ratreraj/gitassigment3/pull/new/f2
remote:
To https://github.com/ratreraj/gitassigment3.git
 * [new branch]      f2 -> f2
branch 'f2' set up to track 'origin/f2'.
ubuntu@ip-172-31-11-219:~/gitassigment3$
```

GitHub Branches



Step 5: Delete the f2 Branch Locally

git branch -d f2

Output

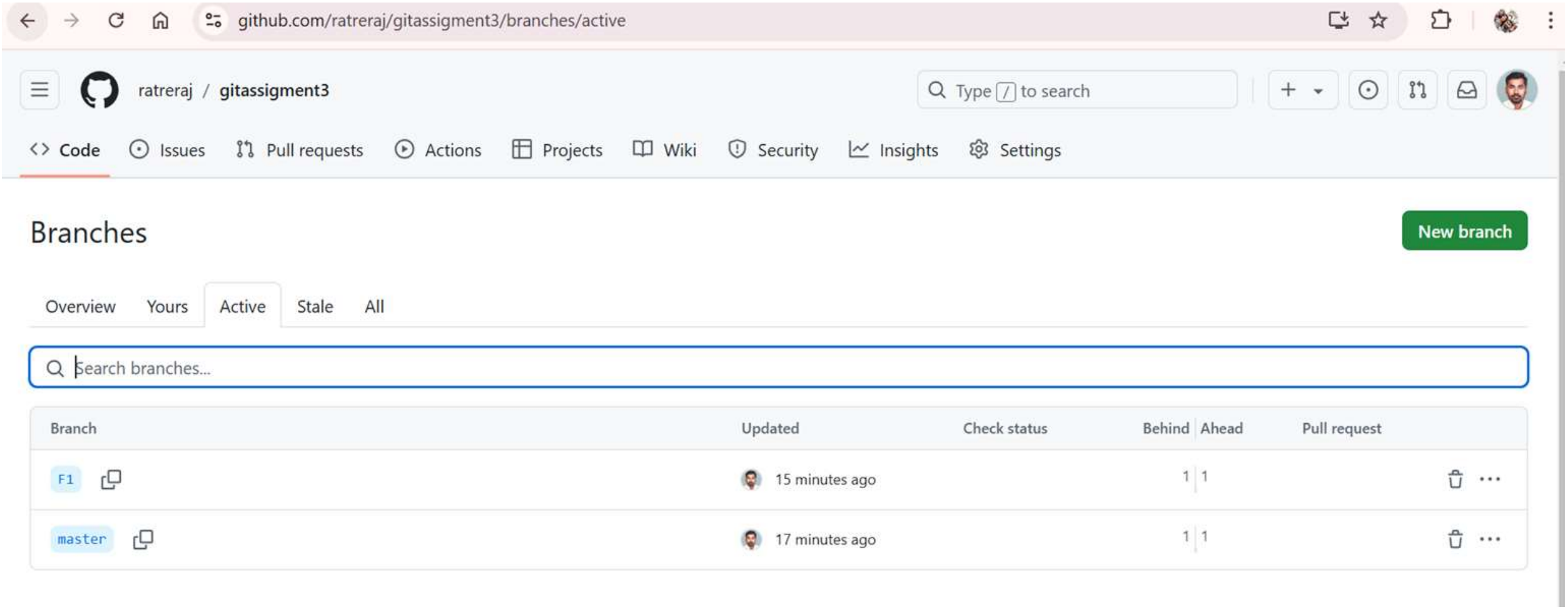
Step 6: Delete the f2 Branch on GitHub

git push origin --delete f2

Output

```
ubuntu@ip-172-31-11-219:~/gitassigment3$ git branch
  F1
  develop
  main
* master
ubuntu@ip-172-31-11-219:~/gitassigment3$ git branch -r
  origin/F1
  origin/f2
  origin/main
  origin/master
ubuntu@ip-172-31-11-219:~/gitassigment3$ git push origin --delete f2
Username for 'https://github.com': ratreraj
Password for 'https://ratreraj@github.com':
To https://github.com/ratreraj/gitassigment3.git
 - [deleted]         f2
ubuntu@ip-172-31-11-219:~/gitassigment3$ git branch -r
  origin/F1
  origin/main
  origin/master
ubuntu@ip-172-31-11-219:~/gitassigment3$
```

GitHub Branches



Module 2: Git Assignment - 4

Tasks To Be Performed:

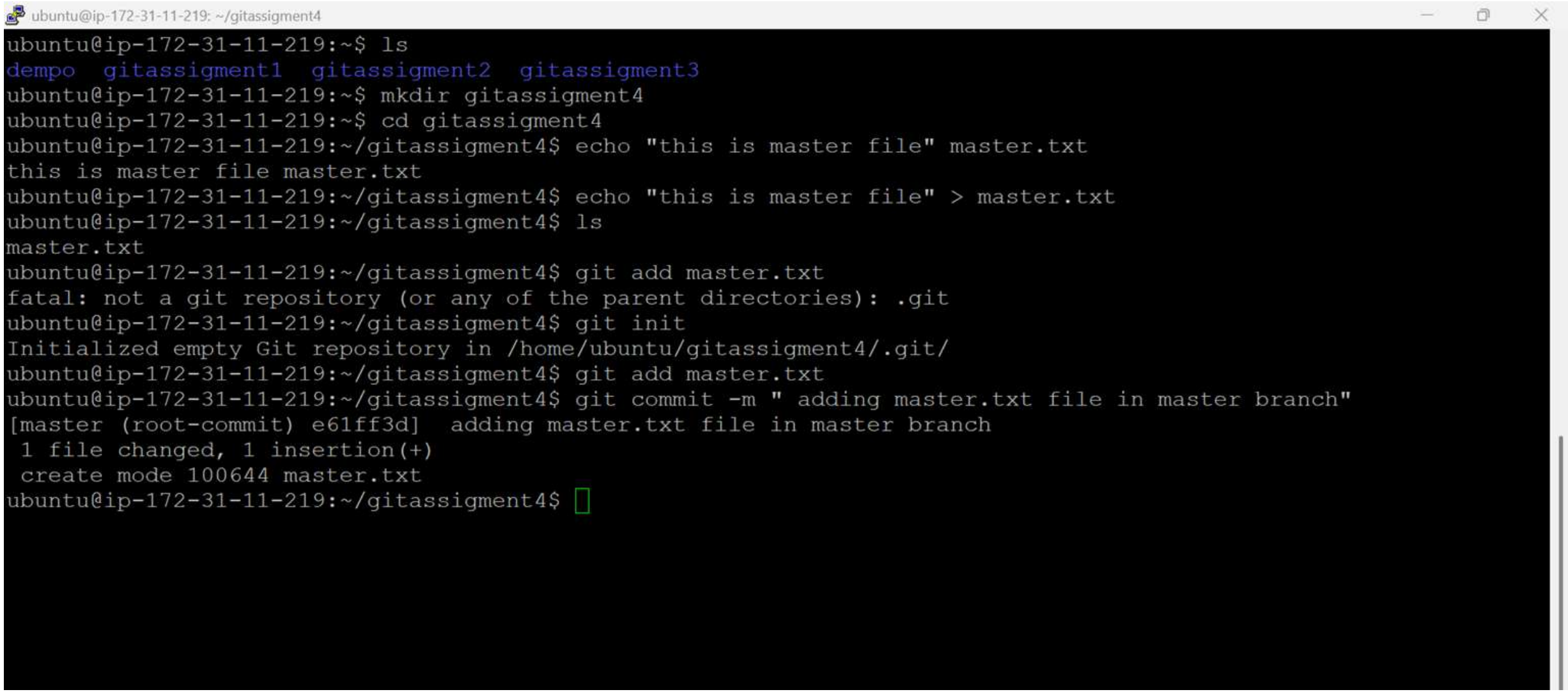
1. Put master.txt on master branch, stage and commit
2. Create 3 branches: public 1, public 2 and private
3. Put public1.txt on public 1 branch, stage and commit
4. Merge public 1 on master branch
5. Merge public 2 on master branch
6. Edit master.txt on private branch, stage and commit
7. Now update branch public 1 and public 2 with new master code in private
8. Also update new master code on master
9. Finally update all the code on the private branch

Step 1: Create and Commit master.txt on the master Branch

```
# Setting Directory
mkdir gitassigment4
cd gitassigment4

# Create master.txt on master branch, stage, and commit it
echo "this is master file" master.txt
git init
git add master.txt
git commit -m " adding master.txt file in master branch"
```

Output

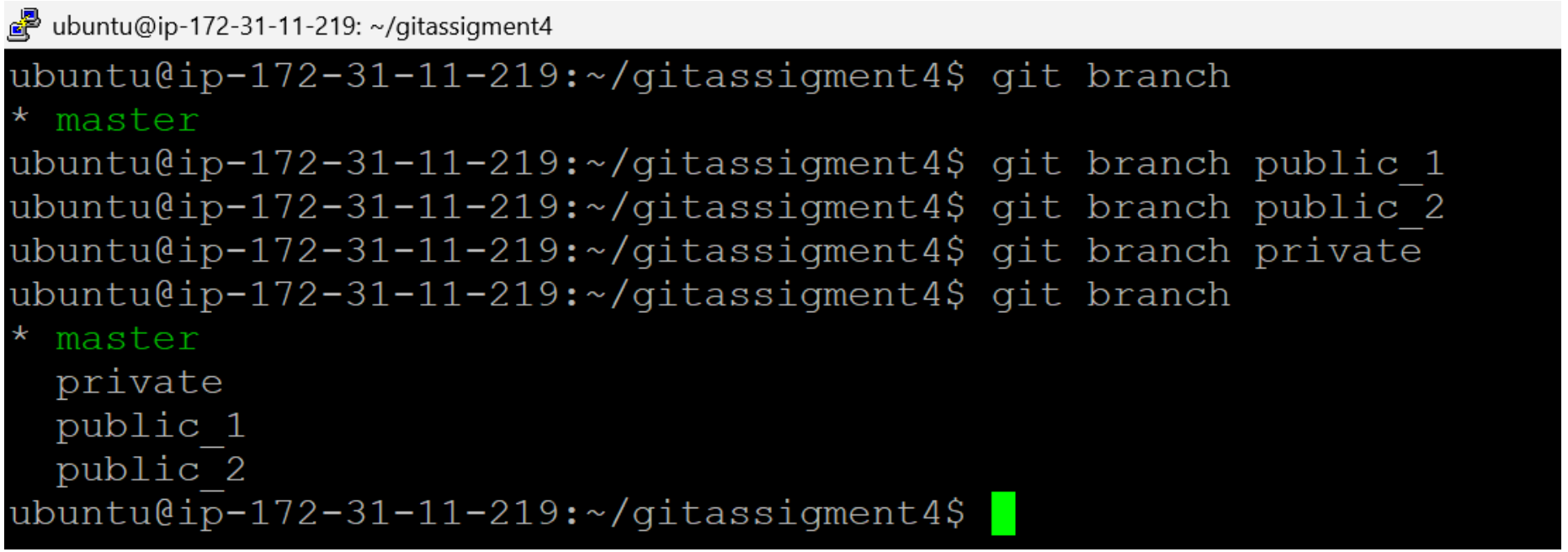

ubuntu@ip-172-31-11-219: ~/gitassigment4
ubuntu@ip-172-31-11-219:~$ ls
dempo gitassigment1 gitassigment2 gitassigment3
ubuntu@ip-172-31-11-219:~$ mkdir gitassigment4
ubuntu@ip-172-31-11-219:~$ cd gitassigment4
ubuntu@ip-172-31-11-219:~/gitassigment4$ echo "this is master file" master.txt
this is master file master.txt
ubuntu@ip-172-31-11-219:~/gitassigment4$ echo "this is master file" > master.txt
ubuntu@ip-172-31-11-219:~/gitassigment4$ ls
master.txt
ubuntu@ip-172-31-11-219:~/gitassigment4$ git add master.txt
fatal: not a git repository (or any of the parent directories): .git
ubuntu@ip-172-31-11-219:~/gitassigment4$ git init
Initialized empty Git repository in /home/ubuntu/gitassigment4/.git/
ubuntu@ip-172-31-11-219:~/gitassigment4$ git add master.txt
ubuntu@ip-172-31-11-219:~/gitassigment4$ git commit -m " adding master.txt file in master branch"
[master (root-commit) e61ff3d] adding master.txt file in master branch
 1 file changed, 1 insertion(+)
 create mode 100644 master.txt
ubuntu@ip-172-31-11-219:~/gitassigment4$

Step 2: Create the public_1, public_2, and private Branches

```
# Create branches public_1, public_2, and private
git branch public_1
git branch public_2
git branch private

# Return to master branch to continue
git checkout master
```

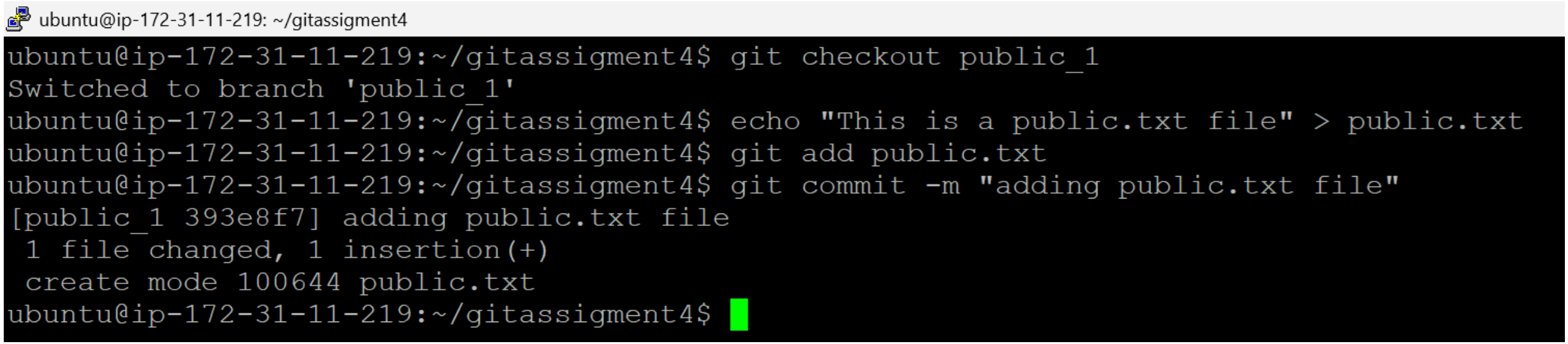
Output



Step 3: Add public1.txt to public_1 Branch and Commit

```
# Switch to public 1 branch, create public1.txt, stage, and commit
git checkout public_1
echo "This is a public.txt file" > public.txt
git add public1.txt
git commit -m "adding public.txt file"
```

Output



Step 4: Merge public_1 & public_2 into master

git merge public_1 -m "merging public_1 branch to master“

git merge public_2 -m "merging public_2 branch to master"

Output

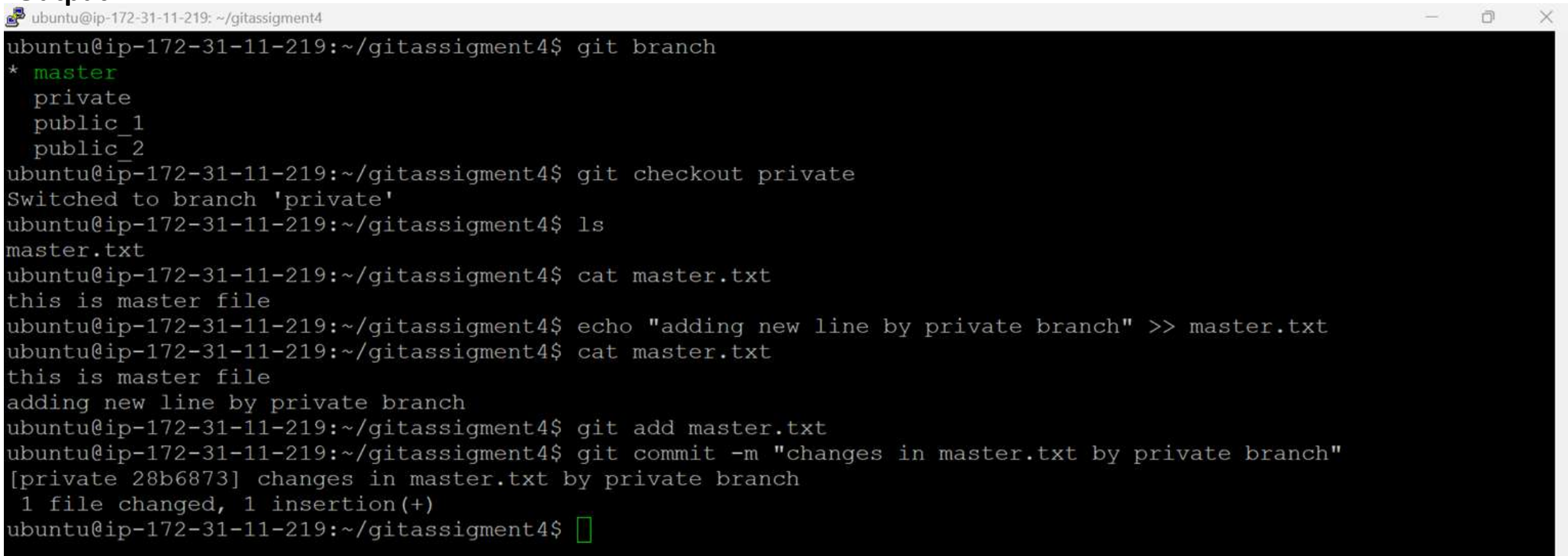
```
ubuntu@ip-172-31-11-219:~/gitassigment4$ git checkout master
Switched to branch 'master'
ubuntu@ip-172-31-11-219:~/gitassigment4$ git merge public_1 "merging public_1 branch to master"
merge: merging public_1 branch to master - not something we can merge
ubuntu@ip-172-31-11-219:~/gitassigment4$ git merge public_1 -m "merging public_1 branch to master"
Updating e61ff3d..393e8f7
Fast-forward (no commit created; -m option ignored)
 public.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 public.txt
ubuntu@ip-172-31-11-219:~/gitassigment4$ git merge public_2 -m "merging public_2 branch to master"
Already up to date.
```

Step 6: Edit master.txt on the private Branch and Commit

```
# Switch to private branch
git checkout private

# Edit master.txt, then stage and commit
echo "adding new line by private branch" >> master.txt
git add master.txt
git commit -m "changes in master.txt by private branch"
```

Output



Step 7: Update public_1 & public_2 with Latest Code from master in private

```
# Switch to public_1 branch
git checkout public_1

# Merge changes from master (via private)
git merge private -m "merging private branch changes in public_1”


# Switch to public 2 branch
git checkout public_2
# Merge changes from master (via private)
git merge private -m "merging private branch changes in public_2"
```

Output


ubuntu@ip-172-31-11-219: ~/gitassigment4
ubuntu@ip-172-31-11-219:~/gitassigment4$ git checkout public_1
Switched to branch 'public_1'
ubuntu@ip-172-31-11-219:~/gitassigment4$ git merge private -m "merging private branch changes in public_1"
Merge made by the 'ort' strategy.
master.txt | 1 +
1 file changed, 1 insertion(+)
ubuntu@ip-172-31-11-219:~/gitassigment4$ git checkout public_2
Switched to branch 'public_2'
ubuntu@ip-172-31-11-219:~/gitassigment4$ git merge private -m "merging private branch changes in public_2"
Updating e61ff3d..28b6873
Fast-forward (no commit created; -m option ignored)
master.txt | 1 +
1 file changed, 1 insertion(+)
ubuntu@ip-172-31-11-219:~/gitassigment4$

Step 8: Update master with the Latest Changes (From private)

Step 9: Update the private Branch with All Latest Changes

```
# Step 8
git checkout master
git merge private -m "merging private branch to master branch"
# Step 9
git checkout private
git merge master -m "merging master to private branch"
```

Output

```
ubuntu@ip-172-31-11-219:~/gitassigment4$ git checkout master
Switched to branch 'master'
ubuntu@ip-172-31-11-219:~/gitassigment4$ git merge private -m "merging private branch to master branch"
Merge made by the 'ort' strategy.
 master.txt | 1 +
 1 file changed, 1 insertion(+)
ubuntu@ip-172-31-11-219:~/gitassigment4$ git checkout private
Switched to branch 'private'
ubuntu@ip-172-31-11-219:~/gitassigment4$ git merge master -m "merging master to private branch"
Updating 28b6873..07a60d7
Fast-forward (no commit created; -m option ignored)
 public.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 public.txt
ubuntu@ip-172-31-11-219:~/gitassigment4$
```

Module 2: Git Assignment - 5

Tasks To Be Performed:

1. Create a Git Flow workflow architecture on Git
2. Create all the required branches
3. Starting from the feature branch, push the branch to the master, following the architecture
4. Push a urgent.txt on master using hotfix

Step 1: Set Up the Git Flow Workflow Architecture
In Git Flow, the primary branches are:

master: For production-ready code.
develop: For ongoing development.
feature/*: For individual features.
hotfix/*: For urgent fixes in production

Create the Initial Git Repository

```
# Initialize a Git repository
git init
git add main.txt
git commit -m "first commit“

# Create develop branch off master
git branch develop
git checkout -b feature/feature1 develop
```

Output

ubuntu@ip-172-31-11-219: ~/gitassigment5

```
ubuntu@ip-172-31-11-219:~$ mkdir gitassigment5
ubuntu@ip-172-31-11-219:~$ cd gitassigment5
ubuntu@ip-172-31-11-219:~/gitassigment5$ touch main.txt
ubuntu@ip-172-31-11-219:~/gitassigment5$ git add main.txt
fatal: not a git repository (or any of the parent directories): .git
ubuntu@ip-172-31-11-219:~/gitassigment5$ git init
Initialized empty Git repository in /home/ubuntu/gitassigment5/.git/
ubuntu@ip-172-31-11-219:~/gitassigment5$ git add main.txt
ubuntu@ip-172-31-11-219:~/gitassigment5$ git commit -m "first commit"
[master (root-commit) 7d35d91] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 main.txt
ubuntu@ip-172-31-11-219:~/gitassigment5$ git branch
* master
ubuntu@ip-172-31-11-219:~/gitassigment5$
ubuntu@ip-172-31-11-219:~/gitassigment5$ git branch develop
ubuntu@ip-172-31-11-219:~/gitassigment5$ git branch
  develop
* master
ubuntu@ip-172-31-11-219:~/gitassigment5$ git checkout -b feature/feature1 develop
Switched to a new branch 'feature/feature1'
ubuntu@ip-172-31-11-219:~/gitassigment5$ git branch
  develop
* feature/feature1
  master
```

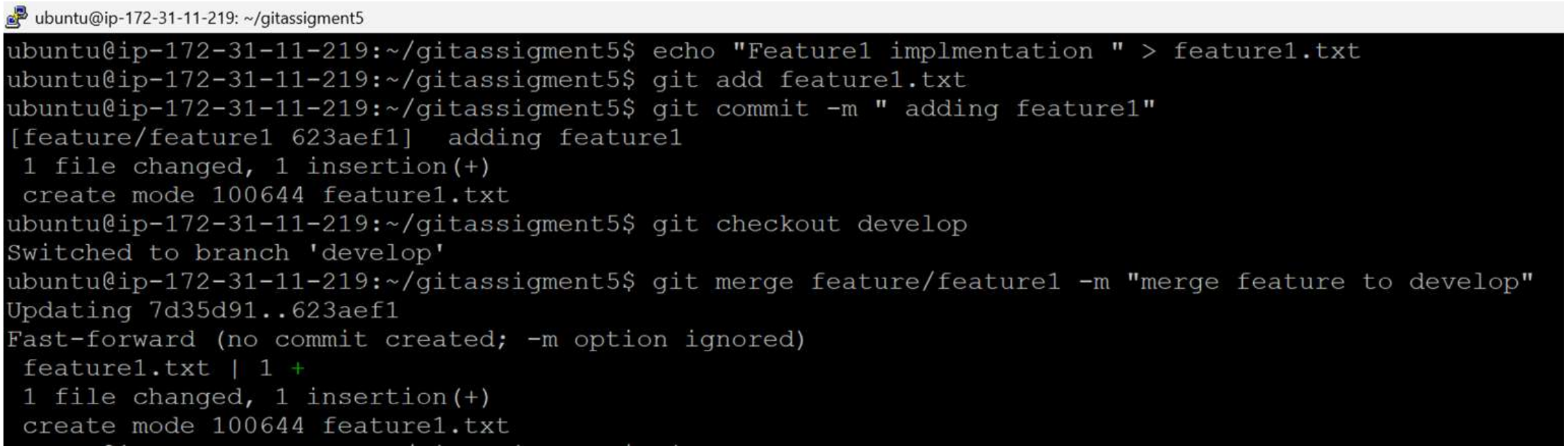
Step 2: Push the Feature Branch to master

Work in the Feature Branch & Merge Feature into develop

```
echo "Feature1 implmentation " > feature1.txt
git add feature1.txt
git commit -m " adding feature1“
```

```
git checkout develop
git merge feature/feature1 -m "merge feature to develop"
```

Output



3- Push All Branches to Remote

git remote add origin https://github.com/ratreraj/assigment5.git
git push origin --all

Output

```
ubuntu@ip-172-31-11-219:~/gitassigment5$ git push origin --all
Username for 'https://github.com': ratreraj
Password for 'https://ratreraj@github.com':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 471 bytes | 471.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ratreraj/assigment5.git
 * [new branch]      develop -> develop
 * [new branch]      feature/feature1 -> feature/feature1
 * [new branch]      master -> master
ubuntu@ip-172-31-11-219:~/gitassigment5$
```

Step 4: Add an Urgent Fix Using a Hotfix Branch

```
#Create a Hotfix Branch
git checkout -b hotfix/urgent-fix master
#git checkout -b hotfix/urgent-fix master
git checkout -b hotfix/urgent-fix master
git add urgent.txt
git commit -m "add urgent.txt for an urgent fix“

#Merge Hotfix into master Switch to master and merge the hotfix:
git checkout master
git merge hotfix/urgent-fix -m "Hotfix: merge hotfxi to master“

# Merge Hotfix into develop To ensure the fix is available in ongoing development:
git checkout develop
git merge hotfix/urgent-fix -m "Hotfix: merge hotfxi to develop"
```

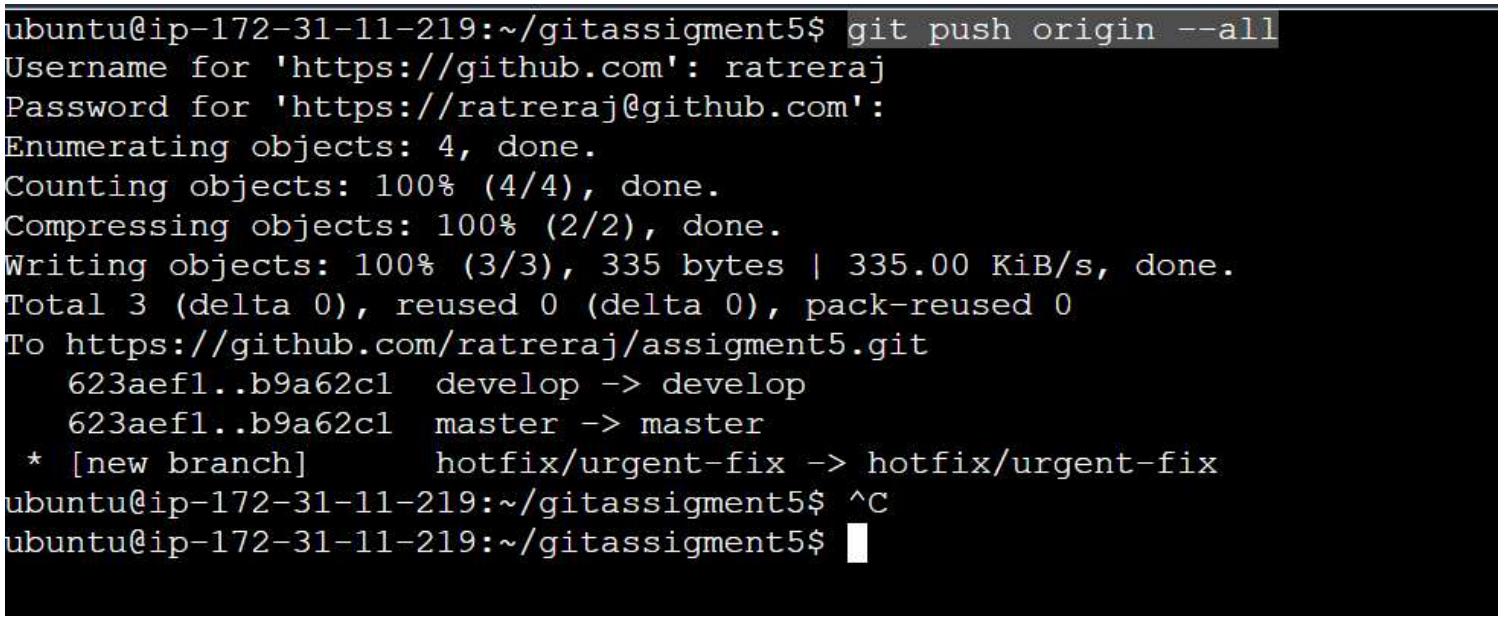
Output

```
ubuntu@ip-172-31-11-219:~/gitassigment5$ git branch
  develop
  feature/feature1
* master
ubuntu@ip-172-31-11-219:~/gitassigment5$ git checkout -b hotfix/urgent-fix master
Switched to a new branch 'hotfix/urgent-fix'
ubuntu@ip-172-31-11-219:~/gitassigment5$ echo "urgent fixed applied" > urgent.txt
ubuntu@ip-172-31-11-219:~/gitassigment5$ git add urgent.txt
ubuntu@ip-172-31-11-219:~/gitassigment5$ git commit -m "add urgent.txt for an urgent fix"
[hotfix/urgent-fix b9a62c1] add urgent.txt for an urgent fix
 1 file changed, 1 insertion(+)
 create mode 100644 urgent.txt
ubuntu@ip-172-31-11-219:~/gitassigment5$ checkout master
checkout: command not found
ubuntu@ip-172-31-11-219:~/gitassigment5$ git checkout master
Switched to branch 'master'
ubuntu@ip-172-31-11-219:~/gitassigment5$ git merge hotfix/urgent-fix -m "Hotfix: merge hotfxi to master"
Updating 623aef1..b9a62c1
Fast-forward (no commit created; -m option ignored)
 urgent.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 urgent.txt
ubuntu@ip-172-31-11-219:~/gitassigment5$ git checkout develop
Switched to branch 'develop'
ubuntu@ip-172-31-11-219:~/gitassigment5$ git merge hotfix/urgent-fix -m "Hotfix: merge hotfxi to develop"
Updating 623aef1..b9a62c1
Fast-forward (no commit created; -m option ignored)
 urgent.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 urgent.txt
ubuntu@ip-172-31-11-219:~/gitassigment5$
```

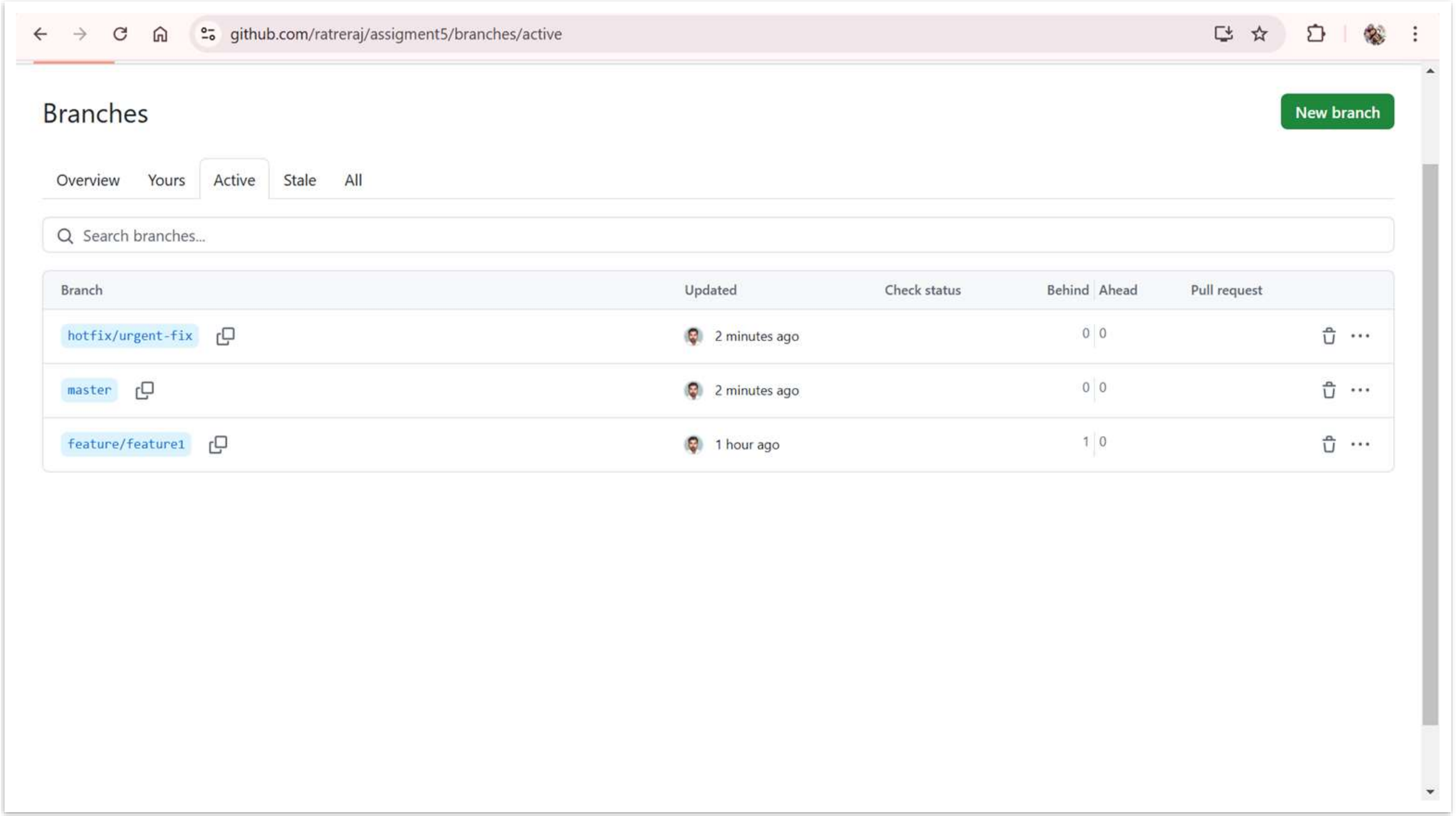
Step 5: Push All Changes Push the updated branches to the remote:

git push origin --all

Output



GitHub Branches



GitHub Commit