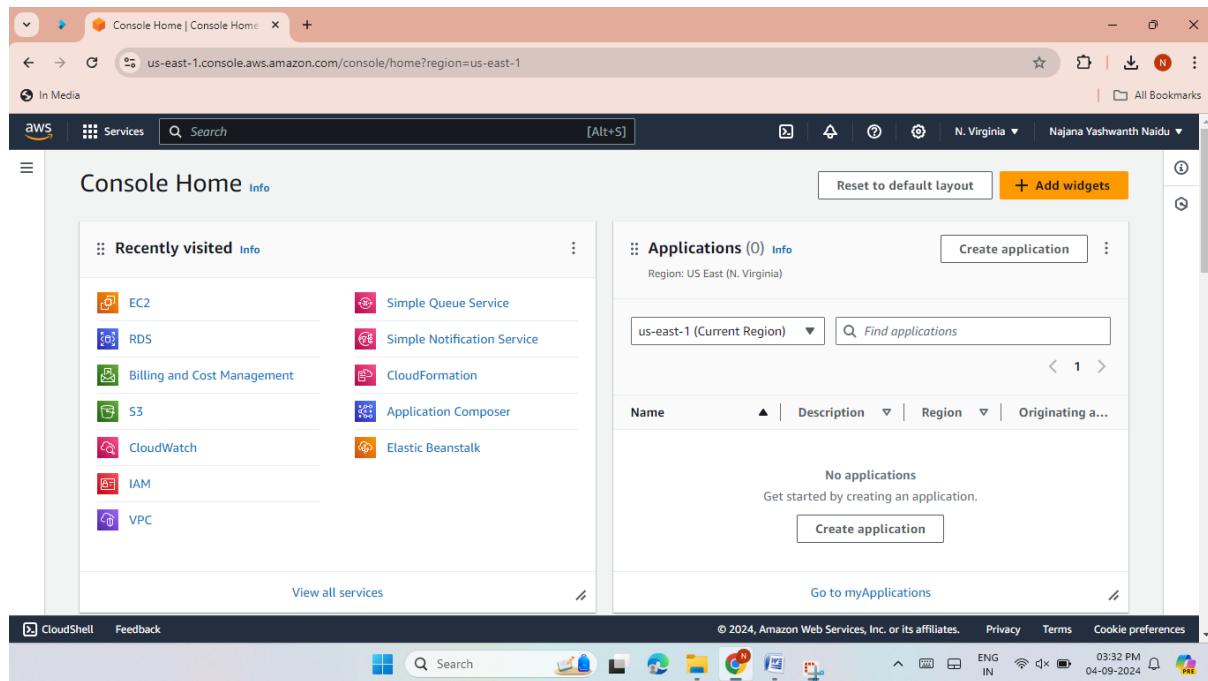


# GIT MINI PROJECT

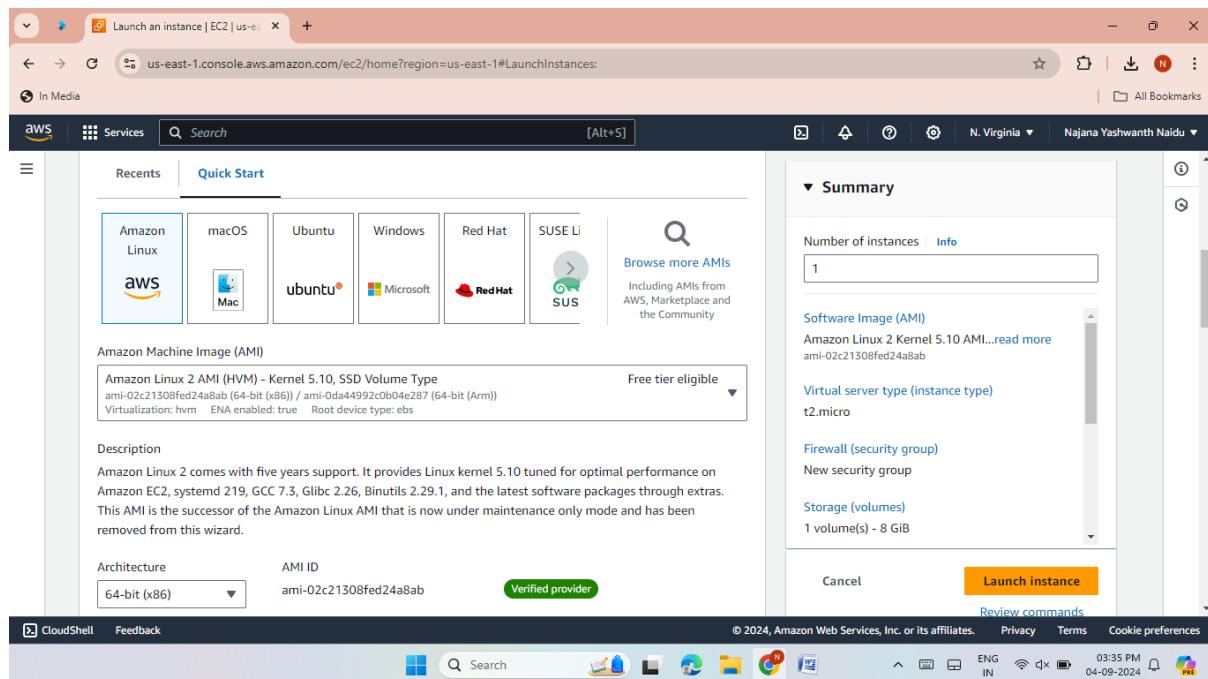
N Yashwanth Naidu

## LAB 1: CREATE EC2 INSTANCE

- login to AWS console.



- Create a server with Amazon Linux 2 AMI/RHEL/UBUNTU.



- Here the instance was launched.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, and Instances. Under Instances, there are sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and Images. The main pane displays a table of instances. One instance, 'lab1-linux2' with ID i-0fbe051e6803a4d85, is listed as 'Running'. Another instance, 'lab-10-instance' with ID i-001055d6abeb98a62, is listed as 'Terminated'. The table includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability. Below the table, a detailed view for 'lab1-linux2' is shown, including its instance ID, public IPv4 address (54.91.135.99), private IPv4 address (172.31.22.218), public IPv4 DNS (ec2-54-91-135-99.compute-1.amazonaws.com), and an 'open address' link. The status summary shows it's running.

- Connect the server only with putty/Gitbash.

The screenshot shows a Windows terminal window titled 'CloudShell'. The command entered is 'ssh -i "yash.pem" ec2-user@ec2-54-91-135-99.compute-1.amazonaws.com'. The terminal output shows the host key fingerprint and asks if the user wants to continue connecting. The user responds with 'yes'. The terminal then displays a welcome message for Amazon Linux 2, noting that the end of life is June 30, 2025, and that a newer version of Amazon Linux is available. It also mentions that Amazon Linux 2023 is supported until March 2028 and provides a link to the Amazon Linux 2023 documentation.

```

ec2-user@ip-172-31-22-218:~ 
yaswa@LAPTOP-GM32QJHO MINGW64 ~
$ cd keys

yaswa@LAPTOP-GM32QJHO MINGW64 ~/keys
$ ssh -i "yash.pem" ec2-user@ec2-54-91-135-99.compute-1.amazonaws.com
The authenticity of host 'ec2-54-91-135-99.compute-1.amazonaws.com (54.91.135.99)' can't be established.
ED25519 key fingerprint is SHA256:crl/Tsym4yIw7CdXAj3UrVUmwgIeevfv/ZvNPw7UmHU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-91-135-99.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

      #_
      _\###_      Amazon Linux 2
      _\###\_
      \###|      AL2 End of Life is 2025-06-30.
      \#/ ,-->
      V~, .-> A newer version of Amazon Linux is available!
      .-. / /    Amazon Linux 2023, GA and supported until 2028-03-15.
      _/ /' /     https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-172-31-22-218 ~]$ |

```

## LAB 2 (CREATE REPO IN LOCAL MACHINE)

- Create a folder on my local machine using the command **mkdir**.
- Install Git in local machine using the command **sudo yum -y install**.

```
ec2-user@ip-172-31-22-218:~$ ssh -i "yash.pem" ec2-user@ec2-54-91-135-99.compute-1.amazonaws.com
The authenticity of host 'ec2-54-91-135-99.compute-1.amazonaws.com (54.91.135.99)' can't be established.
ED25519 key fingerprint is SHA256:cRiTsyn4yIw7CdxAj3UrUmwgIeevfv/ZvNPW7UmHU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-91-135-99.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

,
  _#
  ~\_ ####_      Amazon Linux 2
  ~~ \####\_
  ~~  \|##|      AL2 End of Life is 2025-06-30.
  ~~  \|#/      ~` , __>
  ~~   V~ , __> A newer version of Amazon Linux is available!
  ~~  /` /`      Amazon Linux 2023, GA and supported until 2028-03-15.
  ~~ /` /`      https://aws.amazon.com/linux/amazon-linux-2023/
  _/m/ , /`      ~` , __>

[ec2-user@ip-172-31-22-218 ~]$ mkdir yash
[ec2-user@ip-172-31-22-218 ~]$ ls
yash
[ec2-user@ip-172-31-22-218 ~]$ git init yash
-bash: git: command not found
[ec2-user@ip-172-31-22-218 ~]$ sudo yum install git -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Resolving Dependencies
--> Running transaction check
--> Package git.x86_64 0:2.40.1-1.amzn2.0.3 will be installed
--> Processing Dependency: git-core = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: git-core-doc = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl-Git = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
[ec2-user@ip-172-31-22-218 ~]$ | 3.6 kB 00:00:00
03:48 PM 04-09-2024 ENG IN
```

- Initialize this folder using **git init** command.
- Go inside this folder and run **git status** command to check the status.

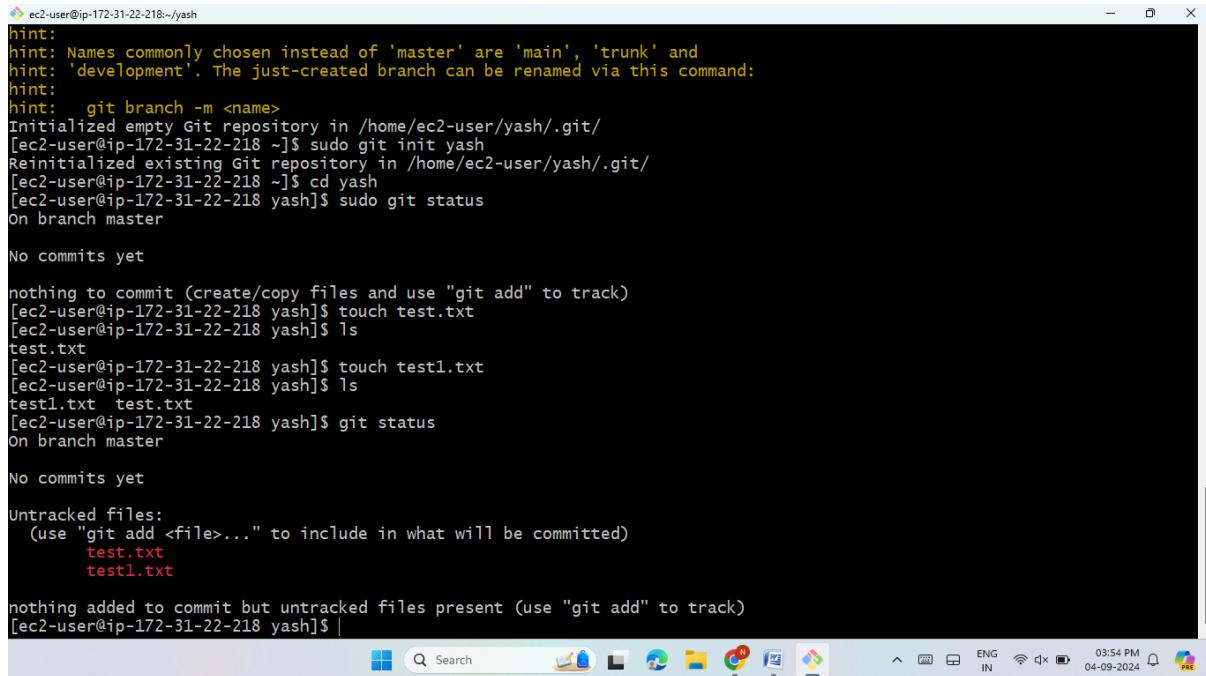
```
git-core.x86_64 0:2.40.1-1.amzn2.0.3 git-core-doc.noarch 0:2.40.1-1.amzn2.0.3 perl-Error.
perl-Git.noarch 0:2.40.1-1.amzn2.0.3 perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2

Complete!
[ec2-user@ip-172-31-22-218 ~]$ git init yash
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/ec2-user/yash/.git/
[ec2-user@ip-172-31-22-218 ~]$ sudo git init yash
Reinitialized existing Git repository in /home/ec2-user/yash/.git/
[ec2-user@ip-172-31-22-218 ~]$ cd yash
[ec2-user@ip-172-31-22-218 yash]$ sudo git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
[ec2-user@ip-172-31-22-218 yash]$ | 03:51 PM 04-09-2024 ENG IN
```

- Created some empty files using **touch** command.
- Again, run **git status** to see the changes, you can notice that the file is available but not tracked by Git.



```

ec2-user@ip-172-31-22-218:~/yash
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/ec2-user/yash/.git/
[ec2-user@ip-172-31-22-218 ~]$ sudo git init yash
Reinitialized existing Git repository in /home/ec2-user/yash/.git/
[ec2-user@ip-172-31-22-218 ~]$ cd yash
[ec2-user@ip-172-31-22-218 yash]$ sudo git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
[ec2-user@ip-172-31-22-218 yash]$ touch test.txt
[ec2-user@ip-172-31-22-218 yash]$ ls
test.txt
[ec2-user@ip-172-31-22-218 yash]$ touch test1.txt
[ec2-user@ip-172-31-22-218 yash]$ ls
test1.txt test.txt
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch master

No commits yet

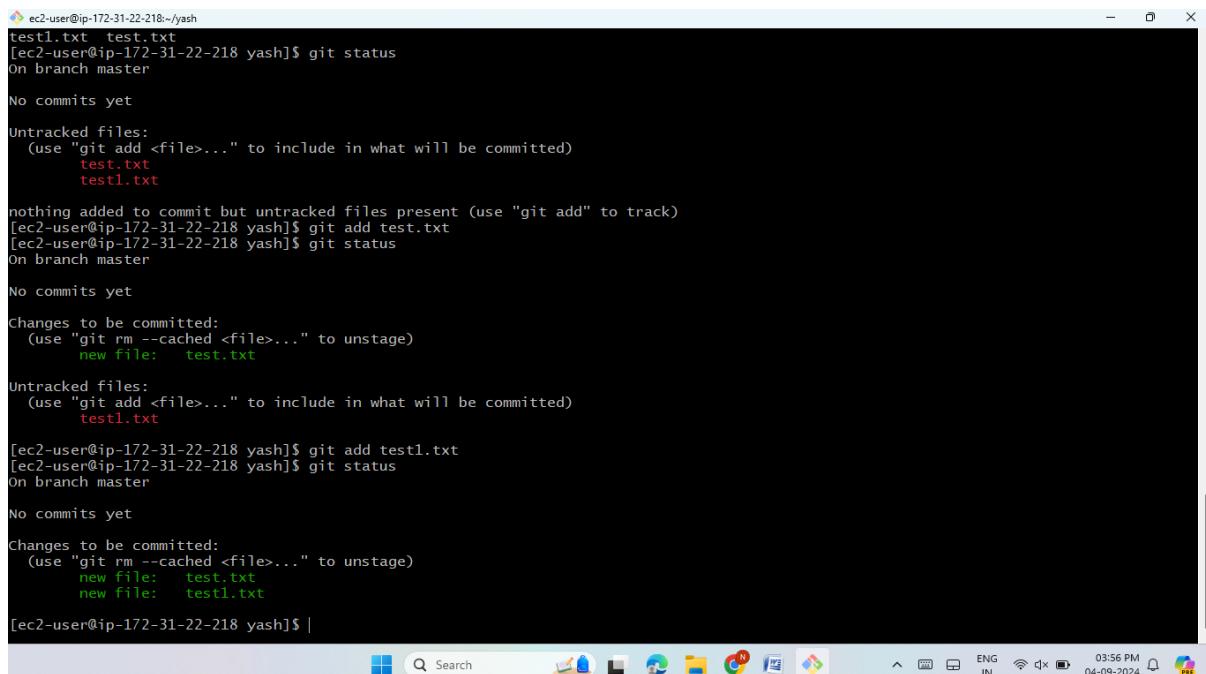
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.txt
    test1.txt

nothing added to commit but untracked files present (use "git add" to track)
[ec2-user@ip-172-31-22-218 yash]$ |

```

The screenshot shows a Windows terminal window with a black background and white text. It displays the output of several commands: `git init`, `ls` (listing `test.txt`), `touch test1.txt`, `ls` (listing `test1.txt` and `test.txt`), and `git status` (showing the repository is on branch `master` with no commits). The terminal window has a title bar, a scroll bar on the right, and a taskbar at the bottom with various icons and system status indicators like battery level and signal strength.

- Run **git add <filename>** to stage this change (git will start tracking this file), you may check is using **git status** once again.



```

ec2-user@ip-172-31-22-218:~/yash
test1.txt test.txt
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.txt
    test1.txt

nothing added to commit but untracked files present (use "git add" to track)
[ec2-user@ip-172-31-22-218 yash]$ git add test.txt
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   test.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test1.txt

[ec2-user@ip-172-31-22-218 yash]$ git add test1.txt
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch master

No commits yet

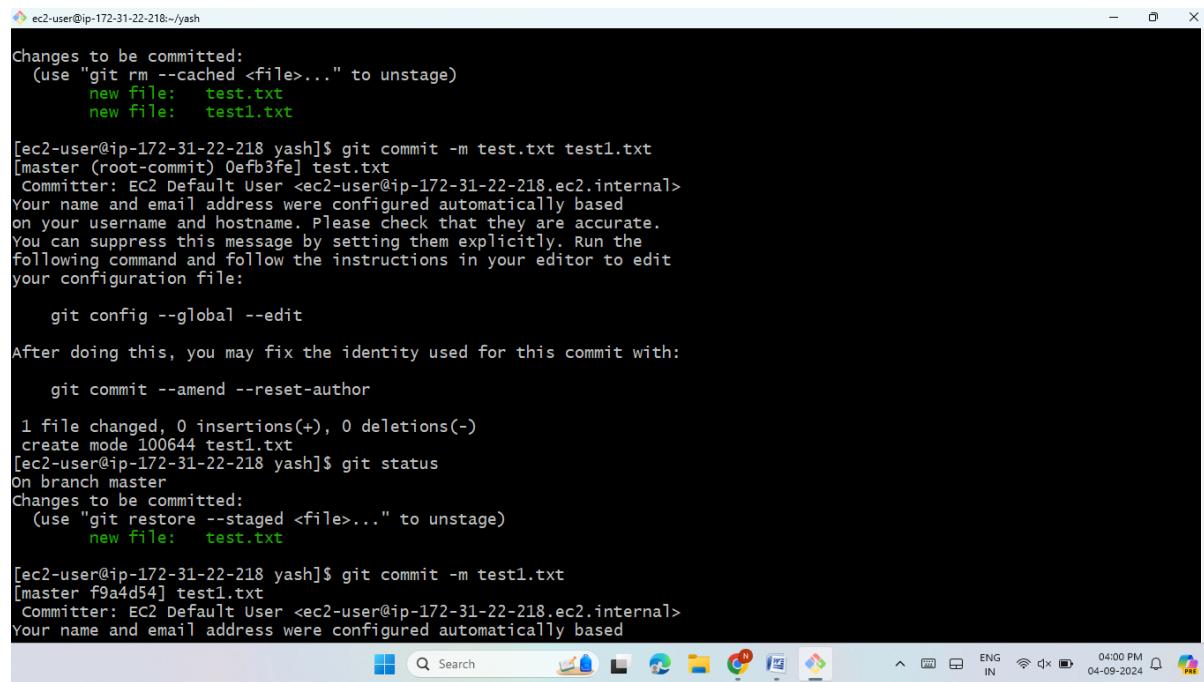
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   test.txt
    new file:   test1.txt

[ec2-user@ip-172-31-22-218 yash]$ |

```

This screenshot shows the terminal after running `git add` on both `test.txt` and `test1.txt`. The output from `git status` now shows staged changes for both files. The terminal window interface is identical to the one in the previous screenshot, with a black background and white text, and a taskbar at the bottom.

- Now commit our changes by running **git commit -m “added test files”**.



```

ec2-user@ip-172-31-22-218:~/yash
changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:  test.txt
  new file:  test1.txt

[ec2-user@ip-172-31-22-218 yash]$ git commit -m test.txt test1.txt
[master (root-commit) 0efb3fe] test.txt
Committer: EC2 Default User <ec2-user@ip-172-31-22-218.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

  git config --global --edit

After doing this, you may fix the identity used for this commit with:

  git commit --amend --reset-author

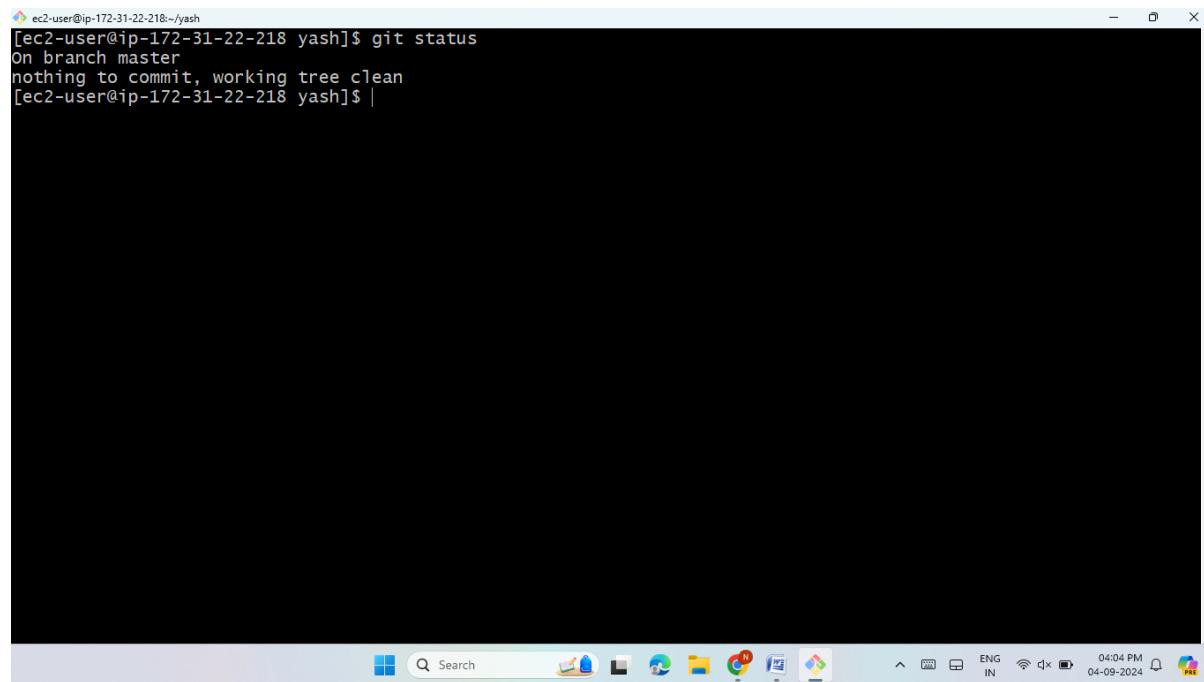
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test1.txt
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch master
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
  new file:  test.txt

[ec2-user@ip-172-31-22-218 yash]$ git commit -m test1.txt
[master f9a4d54] test1.txt
Committer: EC2 Default User <ec2-user@ip-172-31-22-218.ec2.internal>
Your name and email address were configured automatically based

```

The screenshot shows a Windows terminal window with a black background and white text. It displays the command-line session of a user named 'ec2-user'. The user runs 'git commit' with a message 'test.txt test1.txt', which creates a new commit on the 'master' branch. The commit hash is '0efb3fe'. The terminal then shows the result of 'git status', indicating that the working tree is clean ('nothing to commit, working tree clean'). The taskbar at the bottom of the screen shows various icons for system functions like search, file explorer, and network.

- Run **git status** once again and it will show you that the working tree is clean.



```

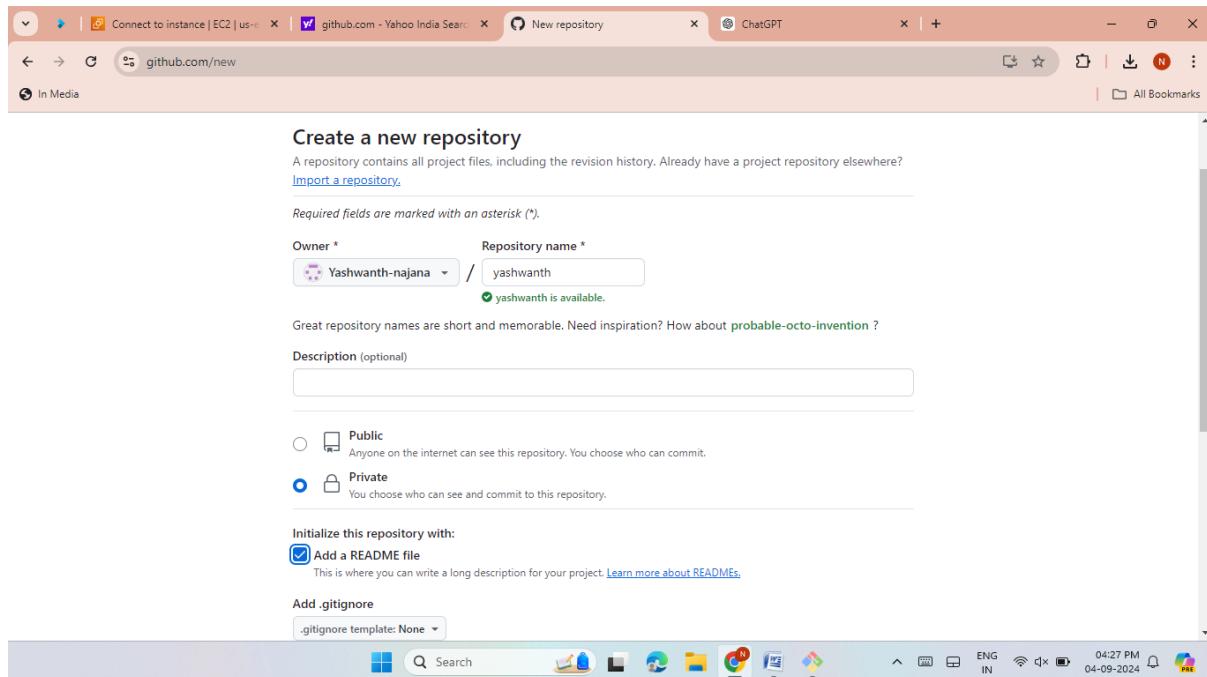
ec2-user@ip-172-31-22-218:~/yash
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch master
nothing to commit, working tree clean
[ec2-user@ip-172-31-22-218 yash]$ |

```

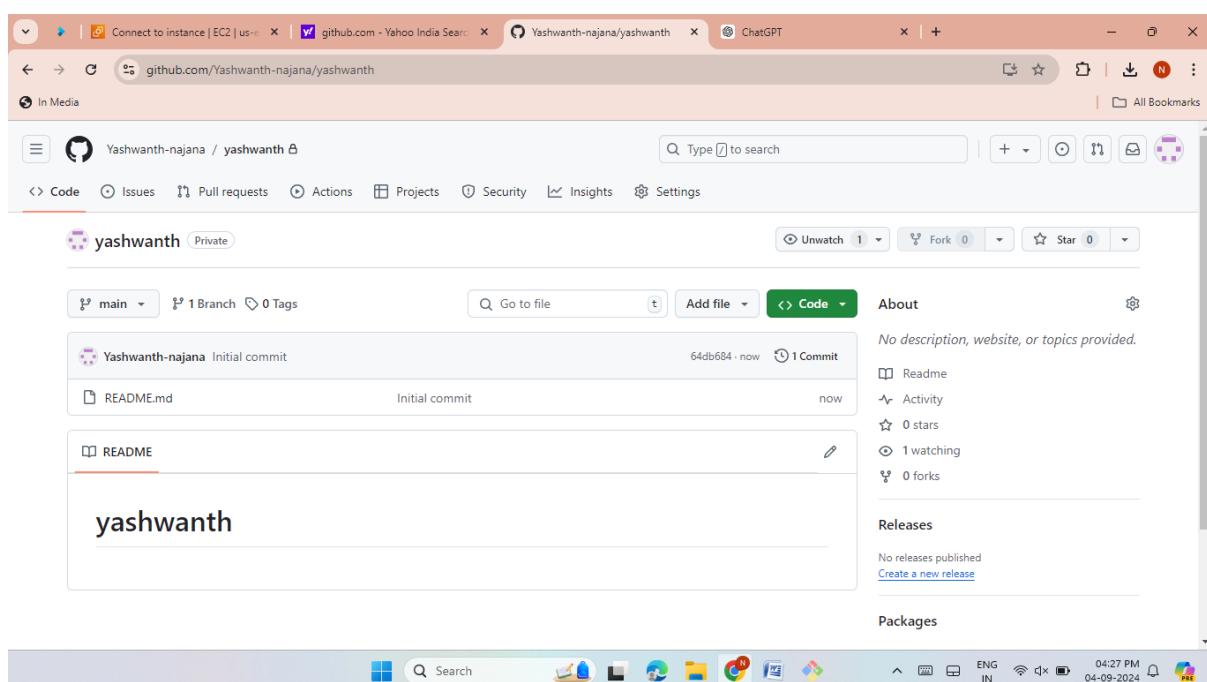
This screenshot shows the same Windows terminal window as the previous one, but the output of 'git status' has been truncated with three vertical ellipses ('...'). The terminal shows the user is still on the 'master' branch and there are no changes to commit. The working tree is described as 'clean'. The taskbar at the bottom remains the same.

## LAB 3 (CREATING REPO IN REMOTE LOCATION – GITHUB)

- In remote location – GitHub.
- Create a new repository by clicking on new button.
- Provide repo name.
- Select whether it is a private or public repo. (I selected private)
- Initialize the repo by adding a README.md file.
- Click on create repo and done.

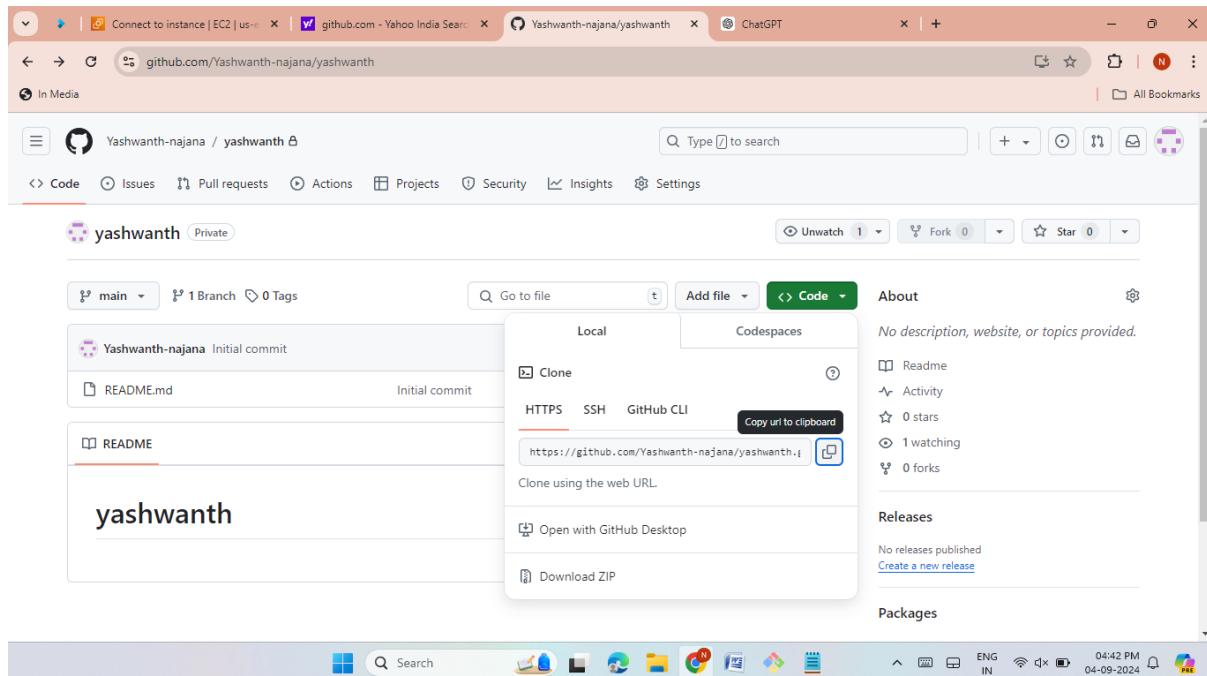


- Here the new repository is created.



## LAB-4 (WORKING WITH REMOTE REPO)

- Take the remote repo to your local.
- Machine make the changes.
- Commit the changes.
- Push the changes to remote.
- Pick the clone URL of the repository from the Github repo.



- Go to your local machine and clone this repo using git clone command.
- you will be asked to provide username and password.
- Here you will get one error for password based authentication depreciation.

```
[ec2-user@ip-172-31-22-218 ~]$ sudo git clone https://github.com/Yashwanth-najana/yashwanth.git
Cloning into 'yashwanth'...
Username for 'https://github.com': Yashwanth-najana
Password for 'https://Yashwanth-najana@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/Yashwanth-najana/yashwanth.git'
[ec2-user@ip-172-31-22-218 ~]$
```

- Now , we need to create the personal access token to work with this repo.
- Go to settings → go to developer settings → go to personal access token → select tokens (classic) → select generate token.

The screenshot shows a browser window with multiple tabs open. The active tab is 'Personal Access Tokens (Classic)' on GitHub. The page content includes a sidebar with options like GitHub Apps, OAuth Apps, Personal access tokens (selected), Fine-grained tokens (Beta), and Tokens (classic). The main area shows a table for managing tokens, with one entry for 'git-mini — repo'. A note at the bottom explains that personal access tokens function like ordinary OAuth access tokens and can be used instead of a password for Git over HTTPS or for authenticating to the API over Basic Authentication.

- Clone the repo again in your local machine, and this time provide the token in place of password while cloning.
- It successfully cloned the repository.

```

[ec2-user@ip-172-31-22-218 ~]$ sudo git clone https://github.com/Yashwanth-najana/yashwanth.git
Cloning into 'yashwanth'...
Username for 'https://github.com': Yashwanth-najana
Password for 'https://Yashwanth-najana@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/Yashwanth-najana/yashwanth.git'
[ec2-user@ip-172-31-22-218 ~]$ sudo git clone https://github.com/Yashwanth-najana/yashwanth.git
Cloning into 'yashwanth'...
Username for 'https://github.com': Yashwanth-najana
Password for 'https://Yashwanth-najana@github.com':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
[ec2-user@ip-172-31-22-218 ~]$ ls
yash yashwanth
[ec2-user@ip-172-31-22-218 ~]$
```

- Once cloned, go to the repo folder and add some sample files .we can use **touch** command to create empty files.
- First I gave **touch filename** but it shows permission denied.
- So we must give **sudo** before touch.

```
ec2-user@ip-172-31-22-218:~/yashwanth
[ec2-user@ip-172-31-22-218 ~]$ cd
[ec2-user@ip-172-31-22-218 ~]$ ls
yash yashwanth
[ec2-user@ip-172-31-22-218 ~]$ cd yashwanth/
[ec2-user@ip-172-31-22-218 yashwanth]$ ls
README.md
[ec2-user@ip-172-31-22-218 yashwanth]$ touch file1.txt
touch: cannot touch 'file1.txt': Permission denied
[ec2-user@ip-172-31-22-218 yashwanth]$ sudo touch file1.txt
[ec2-user@ip-172-31-22-218 yashwanth]$ sudo touch file2.txt
[ec2-user@ip-172-31-22-218 yashwanth]$ ls
file1.txt file2.txt README.md
[ec2-user@ip-172-31-22-218 yashwanth]$
```

- Stage these changes by running **git add file1 file2**.
- Commit these changes by running **git commit -m "<any message>"**.
- Check the status using **git status** command. It shows working tree is clean.

```
ec2-user@ip-172-31-22-218:~/yashwanth
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/ec2-user/.git/
[ec2-user@ip-172-31-22-218 ~]$ ls
> ^C
[ec2-user@ip-172-31-22-218 ~]$ ls
yash yashwanth
[ec2-user@ip-172-31-22-218 ~]$ cd yashwanth/
[ec2-user@ip-172-31-22-218 yashwanth]$ ls
file1.txt file2.txt README.md
[ec2-user@ip-172-31-22-218 yashwanth]$ sudo git add file1.txt
[ec2-user@ip-172-31-22-218 yashwanth]$ sudo git add file2.txt
[ec2-user@ip-172-31-22-218 yashwanth]$ sudo git commit -m file1.txt
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[ec2-user@ip-172-31-22-218 yashwanth]$ sudo git commit -m file2.txt
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[ec2-user@ip-172-31-22-218 yashwanth]$
```

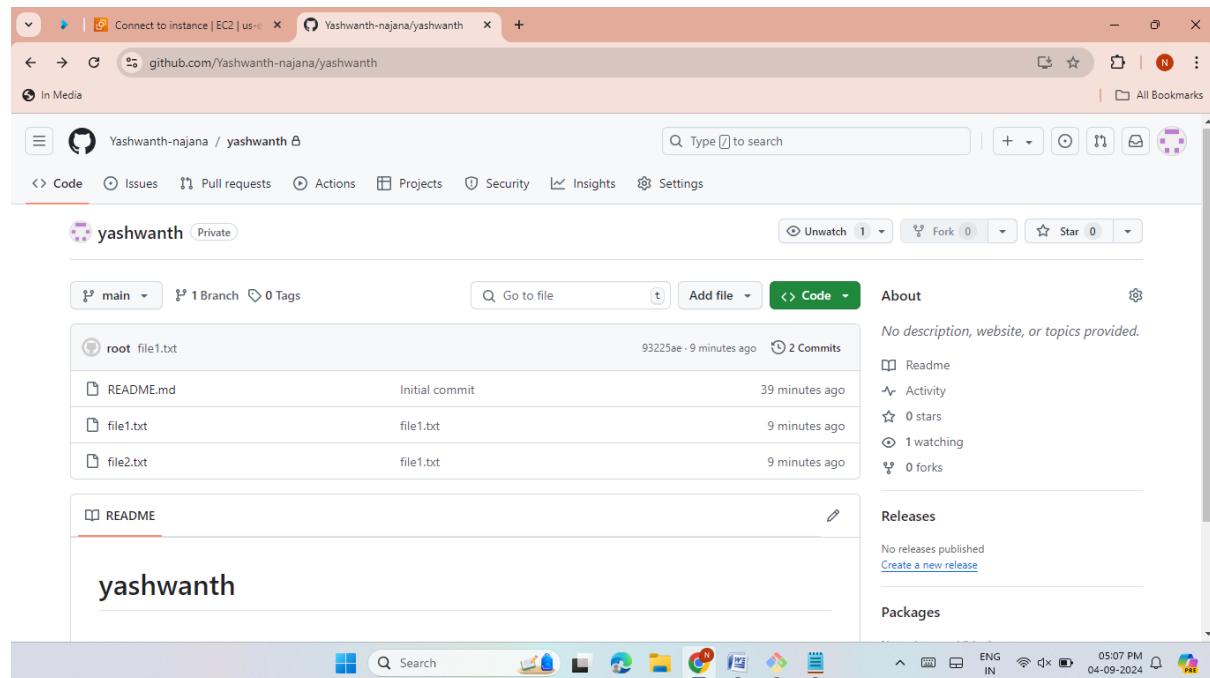
- **git push** → it will ask you for username and password , provide the username and personal access token in place of password which we created before.

```

ec2-user@ip-172-31-22-218:~/yashwanth$ sudo git push
Username for 'https://github.com': Yashwanth-najana
Password for 'https://Yashwanth-najana@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 285 bytes | 285.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Yashwanth-najana/yashwanth.git
  64db684..93225ae main -> main
[ec2-user@ip-172-31-22-218 yashwanth]$

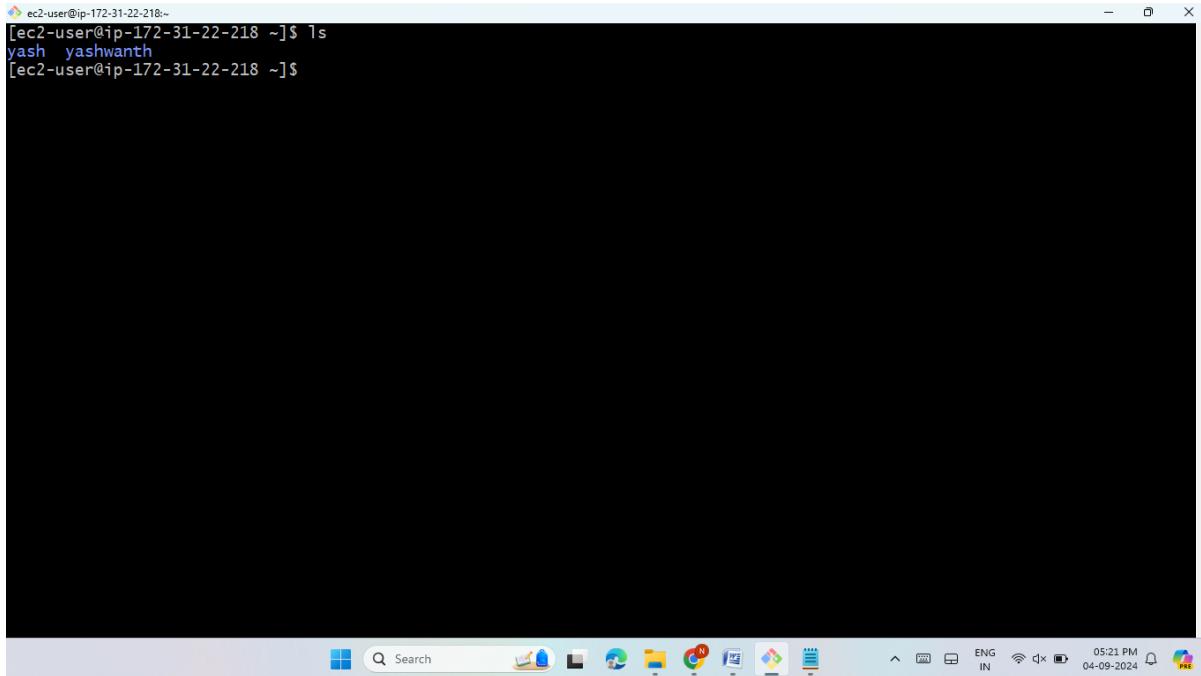
```

- Go to the remote repo and see , you will be able to find your new changes.
- The file1 and file2 is added in remote repository.



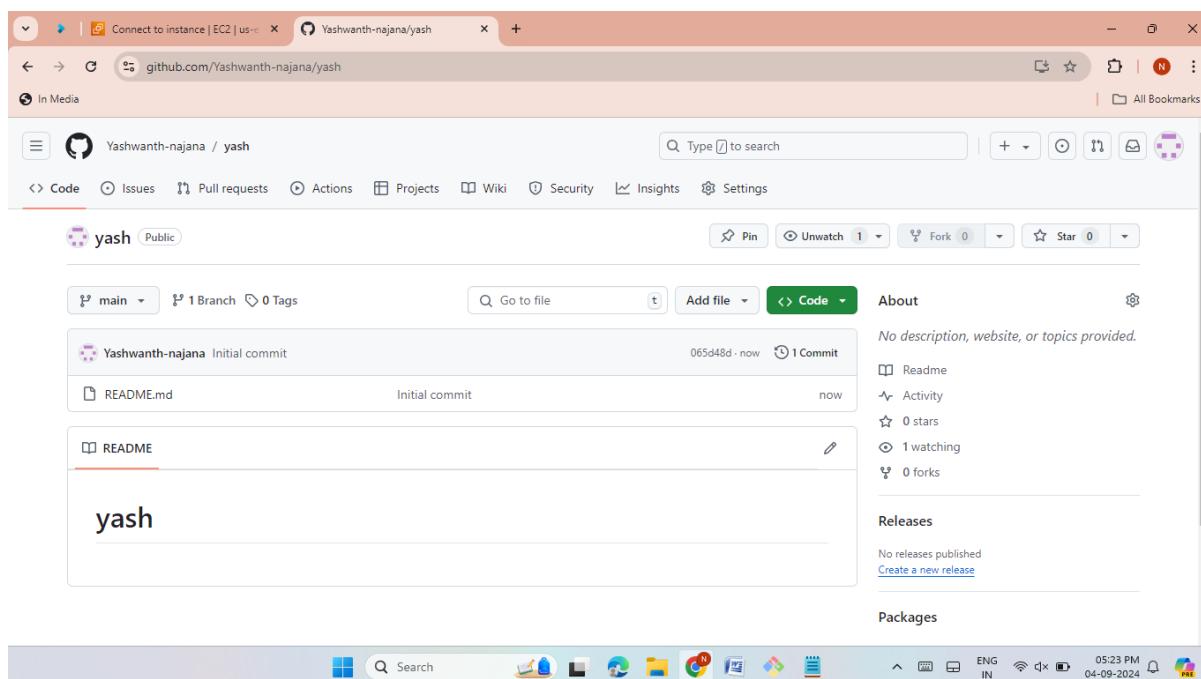
## LAB 5: (PUSHING A LOCALLY CREATED REPO TO GITHUB)

- Created one repo in our local machine and initialize is locally. (we have done it in lab -2)
- The new repository is yash.

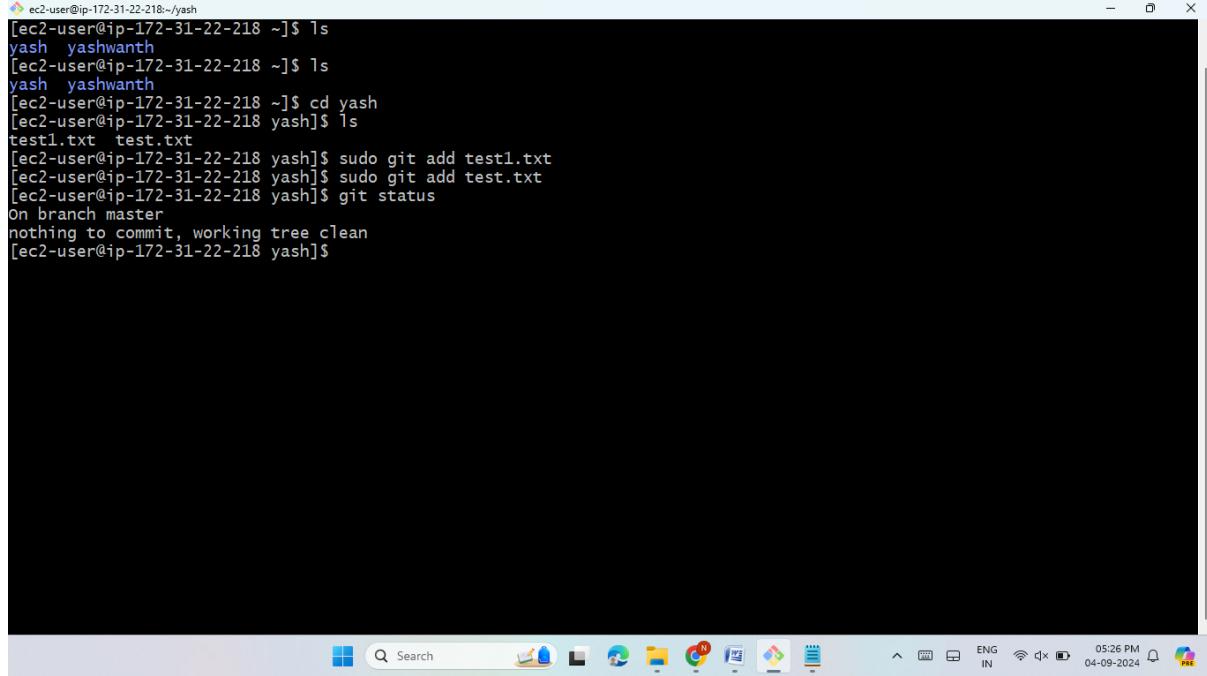


```
ec2-user@ip-172-31-22-218:~ [ec2-user@ip-172-31-22-218 ~]$ ls
yash yashwanth
[ec2-user@ip-172-31-22-218 ~]$
```

- Create one remote repo with the same name as local repo in github and do not initialize it.
- Click on new and then create new repository.
- The new repository name yash.



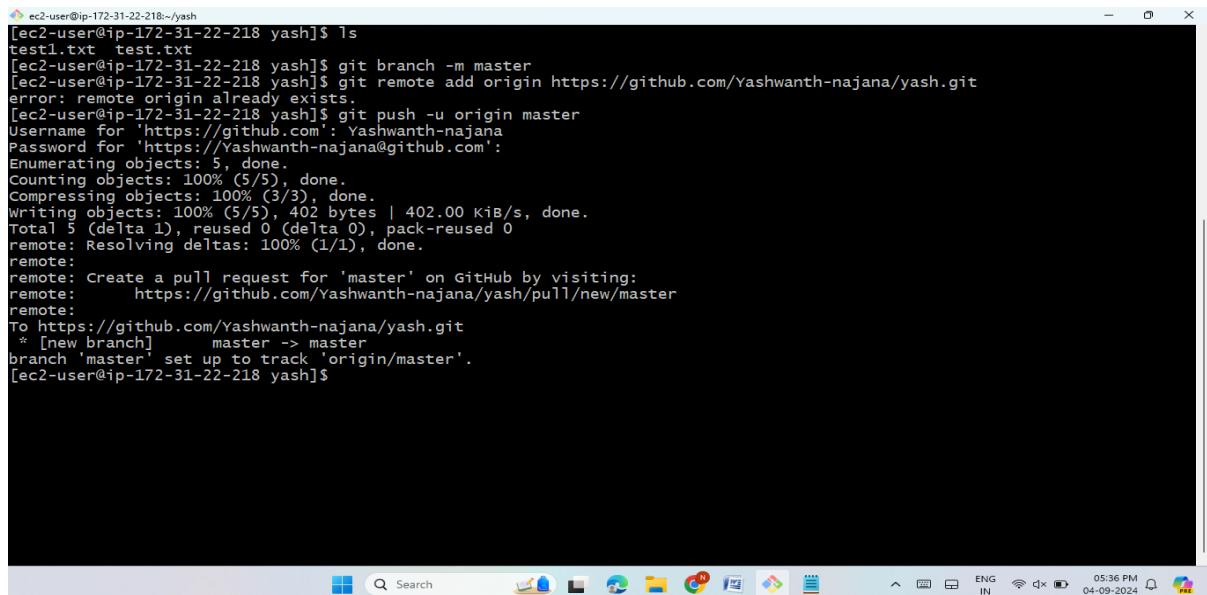
- Come to your local machine and the following commands from inside your local repo.
  - **git add <file name>**
  - **git commit -m <file name>**
  - **git status**
- It shows nothing to commit, working tree is clean.



```
ec2-user@ip-172-31-22-218:~/yash$ ls
yash yashwanth
[ec2-user@ip-172-31-22-218 ~]$ ls
yash yashwanth
[ec2-user@ip-172-31-22-218 ~]$ cd yash
[ec2-user@ip-172-31-22-218 yash]$ ls
test1.txt test.txt
[ec2-user@ip-172-31-22-218 yash]$ sudo git add test1.txt
[ec2-user@ip-172-31-22-218 yash]$ sudo git add test.txt
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch master
nothing to commit, working tree clean
[ec2-user@ip-172-31-22-218 yash]$
```

The screenshot shows a Windows command prompt window titled 'yash'. The terminal output shows the user has a directory 'yash' containing files 'test1.txt' and 'test.txt'. They run 'git add' on both files and then 'git status', which returns 'nothing to commit, working tree clean'. The taskbar at the bottom shows various icons for Microsoft Office applications like Word, Excel, and PowerPoint, along with system status indicators like battery level and signal strength.

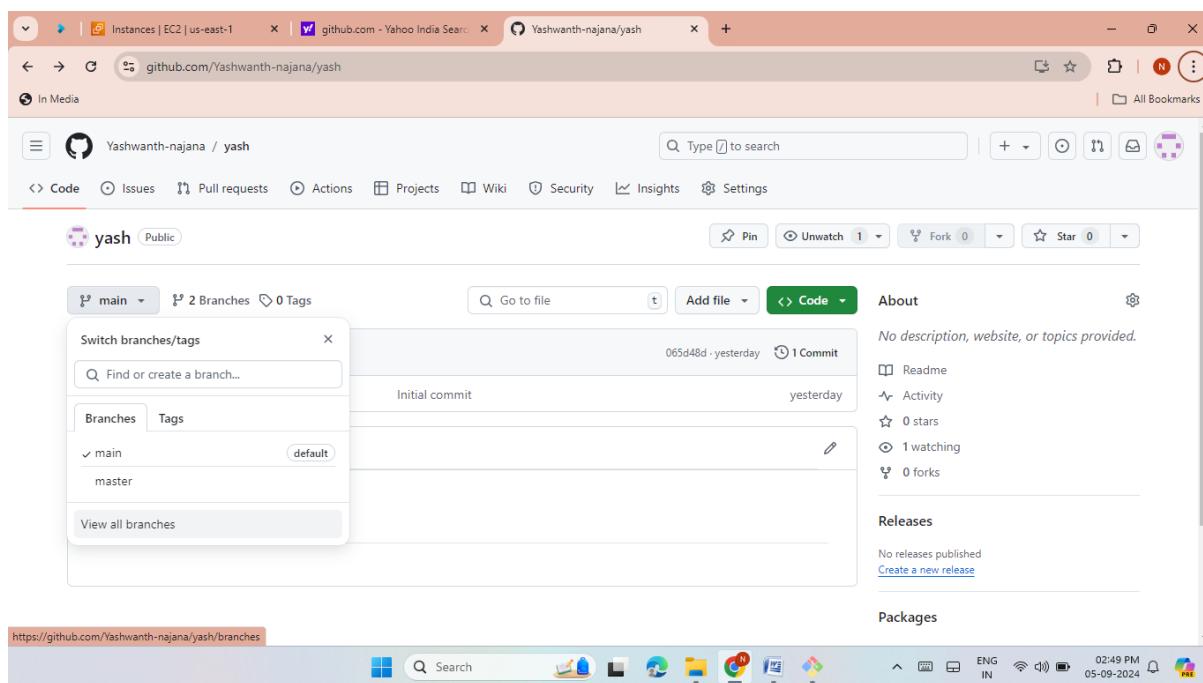
- **git branch -m master** (to change the name of branch as master branch)
- **git remote add origin<URL of your remote repo>**
- **Git push -u origin master.**(to push your local branch to remote repo)
- It will ask you for username and password, provide the username and personal access token in place of password which we created before.



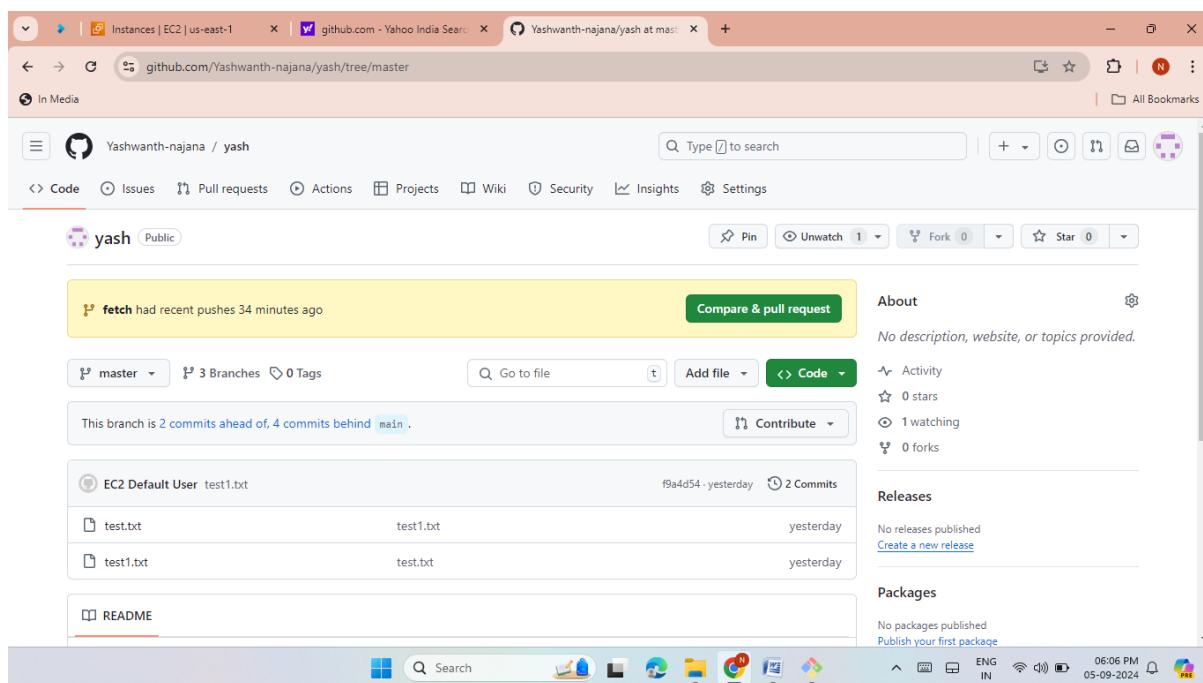
```
ec2-user@ip-172-31-22-218:~/yash$ ls
test1.txt test.txt
[ec2-user@ip-172-31-22-218 yash]$ git branch -m master
[ec2-user@ip-172-31-22-218 yash]$ git remote add origin https://github.com/Yashwanth-najana/yash.git
error: remote origin already exists.
[ec2-user@ip-172-31-22-218 yash]$ git push -u origin master
Username for 'https://github.com': Yashwanth-najana
Password for 'https://Yashwanth-najana@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 402 bytes | 402.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/Yashwanth-najana/yash/pull/new/master
remote:
To https://github.com/Yashwanth-najana/yash.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
[ec2-user@ip-172-31-22-218 yash]$
```

The screenshot shows a Windows command prompt window titled 'yash'. The terminal output shows the user changing the local branch name to 'master', adding a remote repository 'origin' pointing to a GitHub URL, and then pushing the local 'master' branch to the remote 'origin'. A GitHub pull request link is generated. The taskbar at the bottom shows various icons for Microsoft Office applications like Word, Excel, and PowerPoint, along with system status indicators like battery level and signal strength.

- Here the master branch is created.

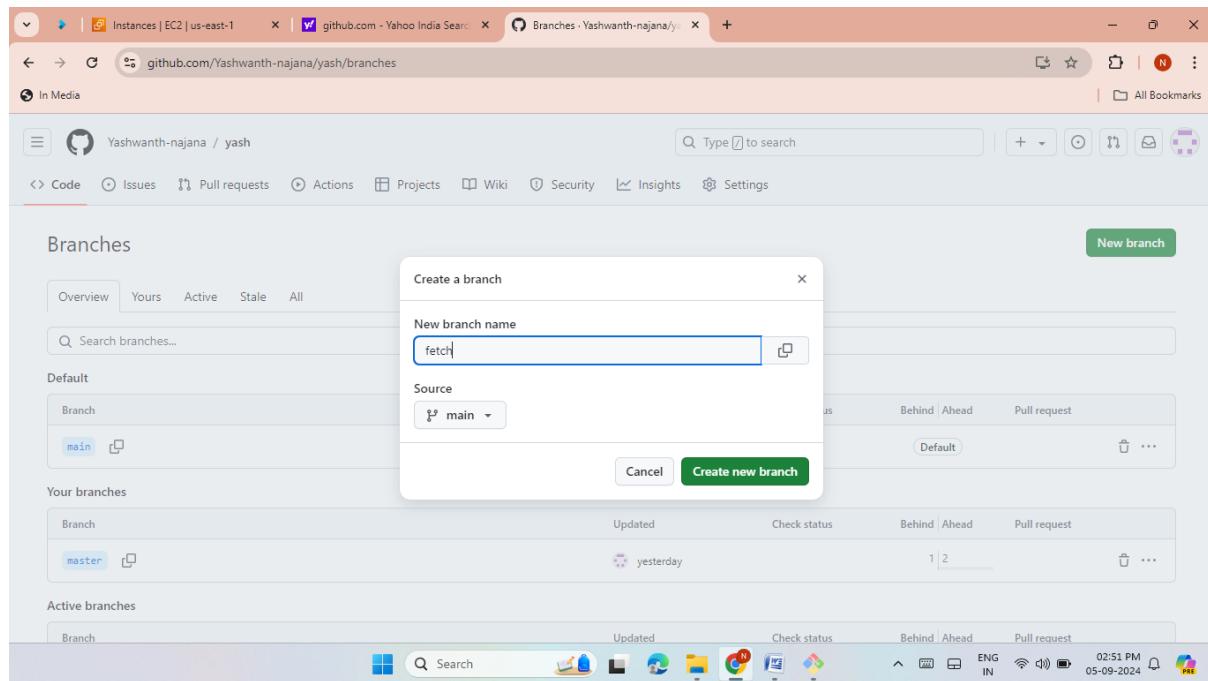


- Here the files also pushed.

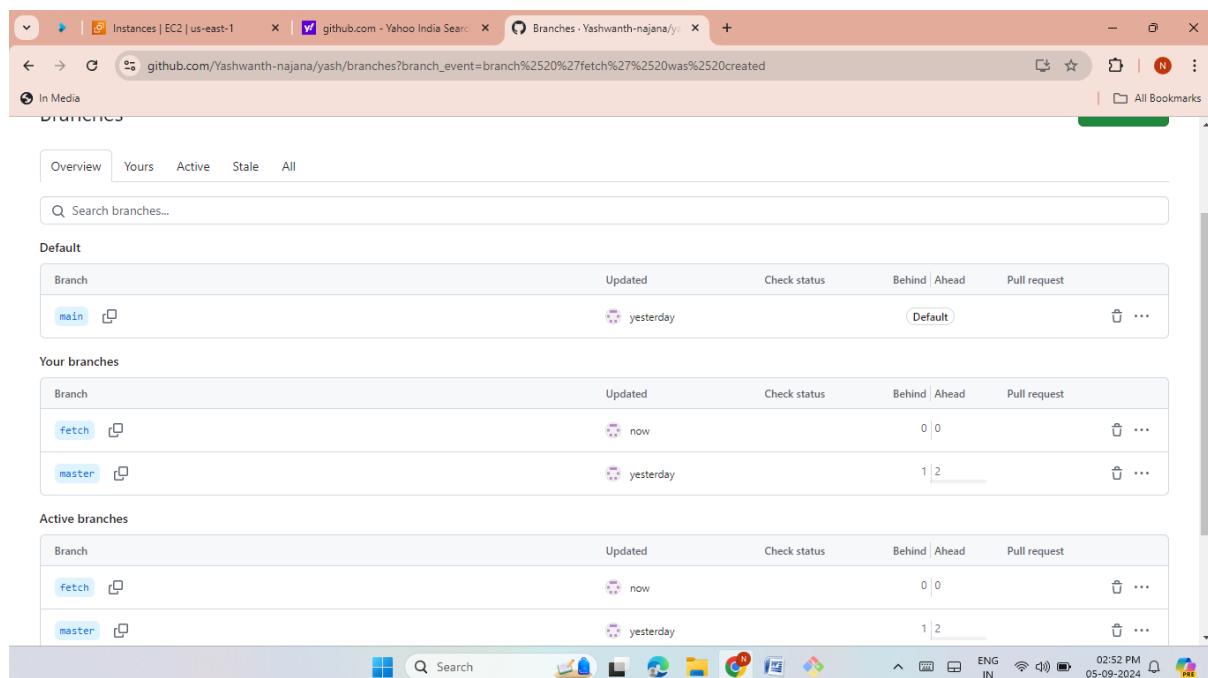


## LAB 6 (CREATING A NEW BRANCH FROM YOUR MAIN BRANCH)

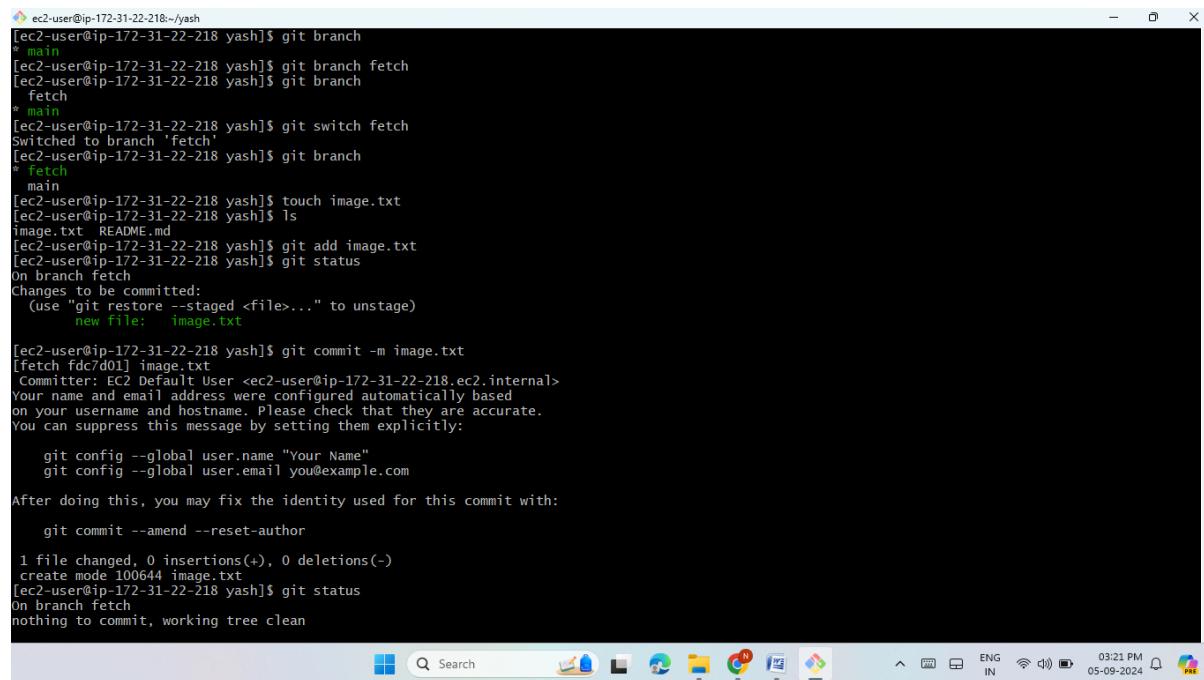
- Go to the repo at the place of main and click on the branch dropdown.
- Type the name of your new branch that you want to create and click on create button.
- Here the fetch is the new branch.
- A new branch will be created.



- A new branch will be created.



- Make some changes in this branch directly from the console.
- Here you will see that your changes are only applied to your new branch but not to the main branch.
- First clone the repository then check the files.
- Run the following commands.
  - **git branch**
  - **Git branch fetch** (to get the branch fetch)
  - **Git switch fetch** (to change the branch main to fetch)
- Create some files using the **touch** command.
  - Here I created image.txt file.
- **git add and commit** the files
- Then check the **status**, it shows nothing to commit, working tree is clean.
- **git push <file name>** (it shows error)
- **git push** (it shows error)
- **git push -u origin <branch>**. (we must use this command to push the files from local to remote)



```

ec2-user@ip-172-31-22-218:~/yash
[ec2-user@ip-172-31-22-218 yash]$ git branch
* main
[ec2-user@ip-172-31-22-218 yash]$ git branch fetch
[ec2-user@ip-172-31-22-218 yash]$ git branch
* fetch
* main
[ec2-user@ip-172-31-22-218 yash]$ git switch fetch
Switched to branch 'fetch'
[ec2-user@ip-172-31-22-218 yash]$ git branch
* fetch
* main
[ec2-user@ip-172-31-22-218 yash]$ touch image.txt
[ec2-user@ip-172-31-22-218 yash]$ ls
image.txt README.md
[ec2-user@ip-172-31-22-218 yash]$ git add image.txt
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch fetch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   image.txt

[ec2-user@ip-172-31-22-218 yash]$ git commit -m image.txt
[fetch fdc7d01] image.txt
  Committer: EC2 Default User <ec2-user@ip-172-31-22-218.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
  git config --global user.name "Your Name"
  git config --global user.email you@example.com
After doing this, you may fix the identity used for this commit with:
  git commit --amend --reset-author
1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 image.txt
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch fetch
nothing to commit, working tree clean

```

```
ec2-user@ip-172-31-22-218:~/yash
nothing to commit, working tree clean
[ec2-user@ip-172-31-22-218 yash]$ git push image.txt
fatal: invalid gitfile format: image.txt
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
[ec2-user@ip-172-31-22-218 yash]$ git push
fatal: The current branch fetch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin fetch

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

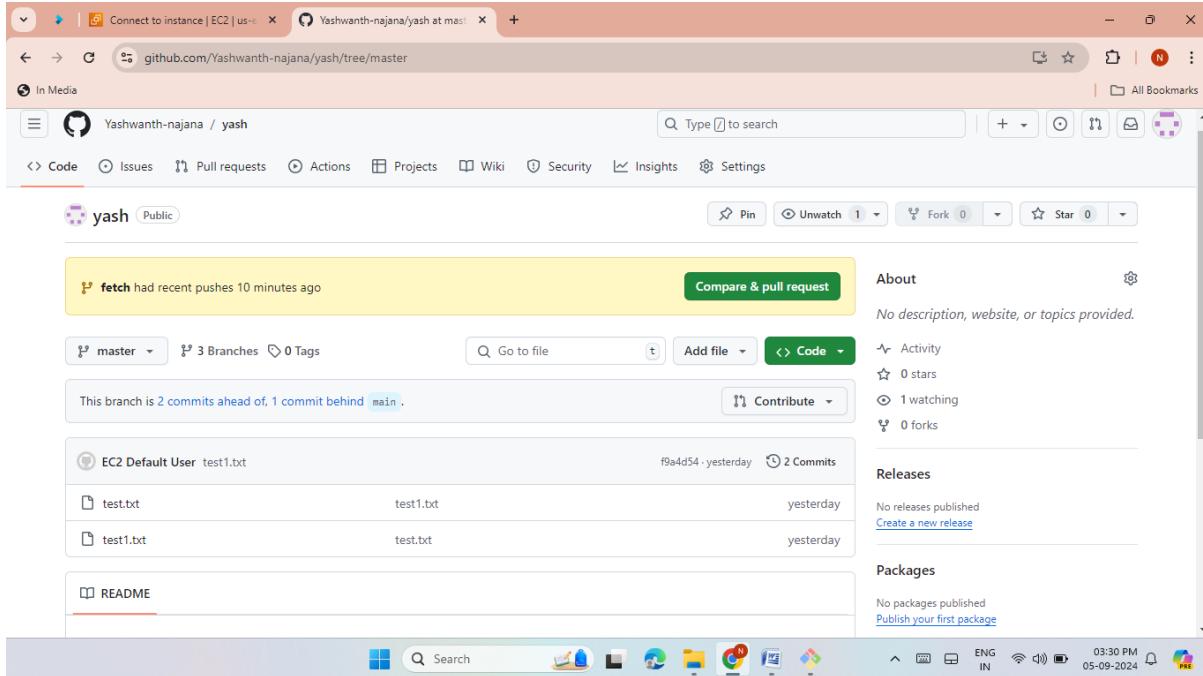
[ec2-user@ip-172-31-22-218 yash]$ git push -u origin image.txt
error: src refspec image.txt does not match any
error: failed to push some refs to 'https://github.com/Yashwanth-najana/yash.git'
[ec2-user@ip-172-31-22-218 yash]$ git push -u origin fetch
Username for 'https://github.com': Yashwanth-najana
Password for 'https://Yashwanth-najana@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
Total 3 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/Yashwanth-najana/yash.git
  065d48d..fdc7d01  fetch -> fetch
branch 'fetch' set up to track 'origin/fetch'.
[ec2-user@ip-172-31-22-218 yash]$ |
```

03:22 PM 05-09-2024

- Here the fetch branch got the file (image.txt).

The screenshot shows a GitHub repository page for the user 'Yashwanth-najana' named 'yash'. The 'Code' tab is selected. A yellow banner at the top left indicates that the 'fetch' branch had recent pushes 10 minutes ago. Below the banner, it says 'This branch is 1 commit ahead of main.' The repository contains three branches: 'fetch', 'main', and 'develop'. The 'fetch' branch is the active one, showing two commits: 'EC2 Default User image.txt' (fd7d01) and 'Initial commit' (yesterday). There is also a file 'README.md'. On the right side of the page, there are sections for 'About', 'Releases', and 'Packages', all of which are currently empty.

- We can check remaining branch also. Only fetch branch got the file. Remaining branch have no changes.



## LAB 7 (PULL ALL THE BRANCHES IN YOUR LOCAL MACHINE)

- Go to your local machine where you have the copy of your remote branch (lab -4).
- Run the command “**git pull**” to pull all the new changes such as branches form the remote location.
- Run “**git branch -a**” to list down all the branches.
- The branch which starts with remotes/→ are remote branches.
- The branch without remote /→ they are available on your local copy as well.
- Checkout to the feature branch or the branch that you crated in (lab-6)
- **git checkout<branch name>** (main)
- Make sure that are on the new branch by running git status or git branch command. (notice the \* mark)

```

[ec2-user@ip-172-31-22-218:~/yash]
* main
[ec2-user@ip-172-31-22-218 yash]$ git branch
branch 'fetch' set up to track 'origin/fetch'.
Switched to a new branch 'fetch'
[ec2-user@ip-172-31-22-218 yash]$ git branch
* fetch
  main
[ec2-user@ip-172-31-22-218 yash]$ clear
[ec2-user@ip-172-31-22-218 yash]$ git pull
Already up to date.
[ec2-user@ip-172-31-22-218 yash]$ git branch -a
* fetch
  main
    remotes/origin/HEAD -> origin/main
    remotes/origin/fetch
    remotes/origin/main
    remotes/origin/master
[ec2-user@ip-172-31-22-218 yash]$ git checkout fetch
Already on 'fetch'
Your branch is up to date with 'origin/fetch'.
[ec2-user@ip-172-31-22-218 yash]$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
[ec2-user@ip-172-31-22-218 yash]$ git branch
  fetch
* main
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
[ec2-user@ip-172-31-22-218 yash]$ touch doc.txt

```

- Make some changes in this branch such as adding the files “ **touch file3 file4**”.(doc.txt,doc1.txt)
- **git status**
- **git add<filename>** (doc.txt,doc1.txt)
- **git commit -m “<msg>”** (doc.txt,doc1.txt)
- Run git status to show working tree is clean.
- **Git push**
- It will ask you for username and password , provide the username and personal access token in place of password which we created before.

```

nothing to commit, working tree clean
[ec2-user@ip-172-31-22-218 yash]$ touch doc.txt
[ec2-user@ip-172-31-22-218 yash]$ touch doc1.txt
[ec2-user@ip-172-31-22-218 yash]$ ls
doc1.txt  doc.txt  README.md
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    doc.txt
    doc1.txt

nothing added to commit but untracked files present (use "git add" to track)
[ec2-user@ip-172-31-22-218 yash]$ git add doc.txt
[ec2-user@ip-172-31-22-218 yash]$ git add doc1.txt
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   doc.txt
    new file:   doc1.txt

[ec2-user@ip-172-31-22-218 yash]$ git commit -m doc.txt
[main 80843b3] doc.txt
  Committer: EC2 Default User <ec2-user@ip-172-31-22-218.ec2.internal>
  Your name and email address were configured automatically based
  on your username and hostname. Please check that they are accurate.
  You can suppress this message by setting them explicitly:

```

```

ec2-user@ip-172-31-22-218:~/yash
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"
git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

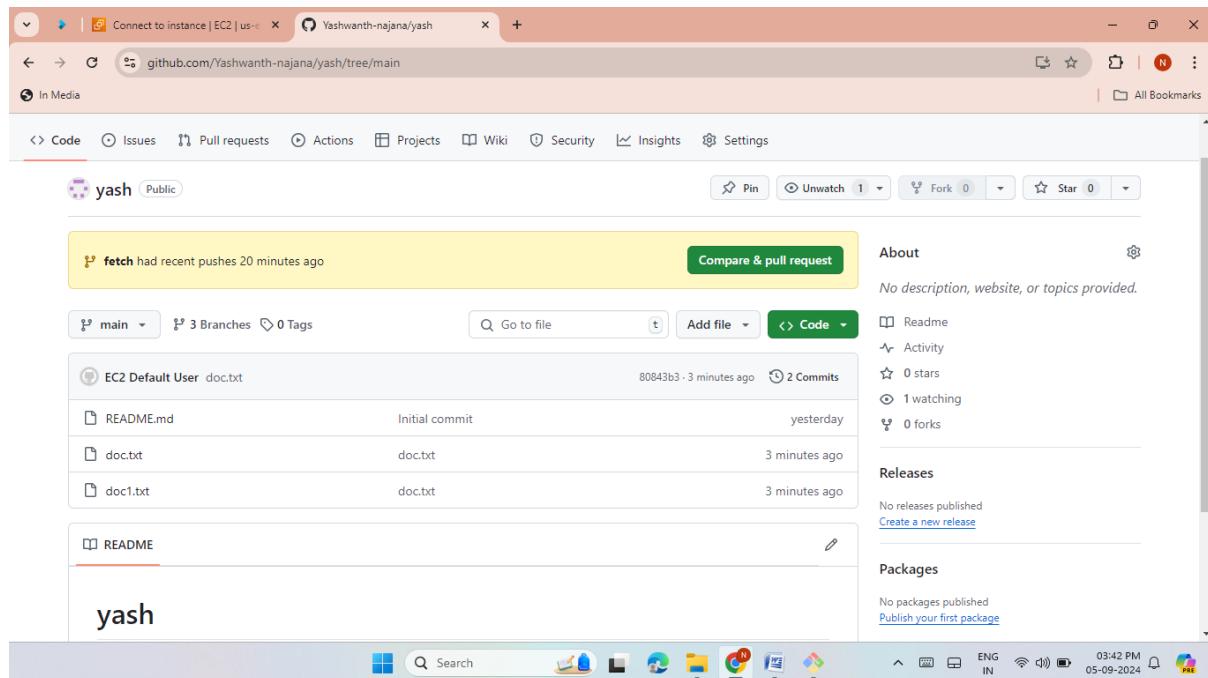
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 doc.txt
create mode 100644 doc1.txt
[ec2-user@ip-172-31-22-218 yash]$ git commit -m doc1.txt
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

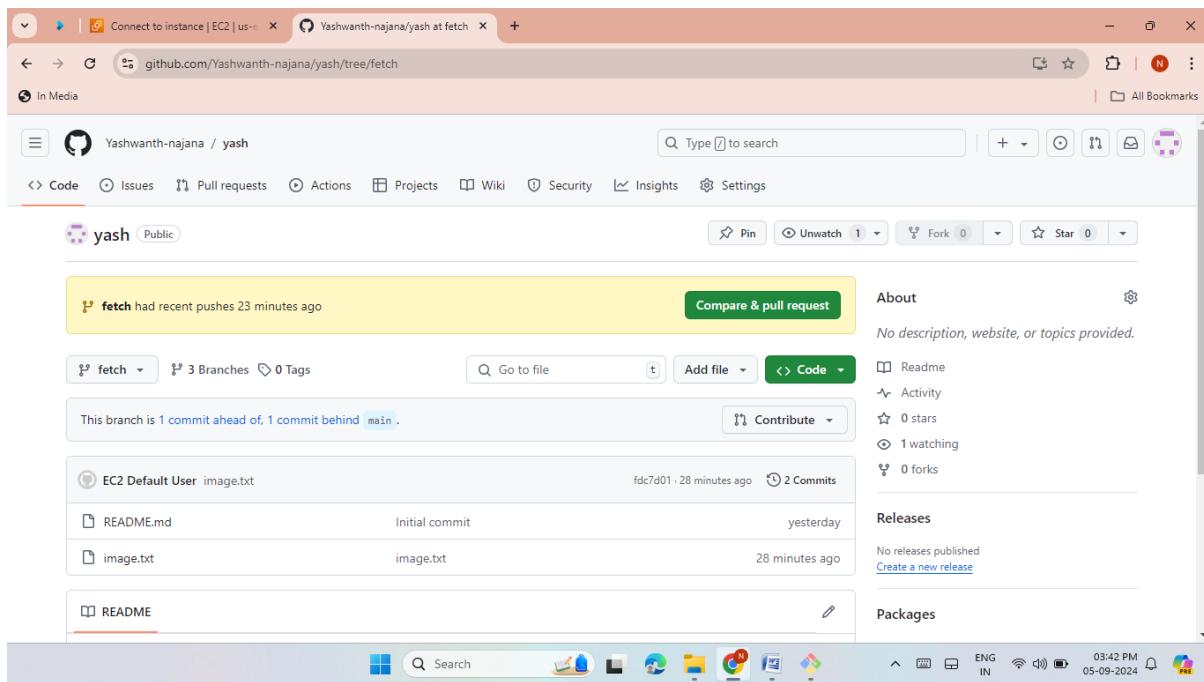
nothing to commit, working tree clean
[ec2-user@ip-172-31-22-218 yash]$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[ec2-user@ip-172-31-22-218 yash]$ git push
Username for 'https://github.com': Yashwanth-najana
Password for 'https://Yashwanth-najana@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
Total 3 (delta 0), reused 1 (delta 0), pack-reused 0
To https://github.com/Yashwanth-najana/yash.git
  065d48d..80843b3  main -> main
[ec2-user@ip-172-31-22-218 yash]$

```

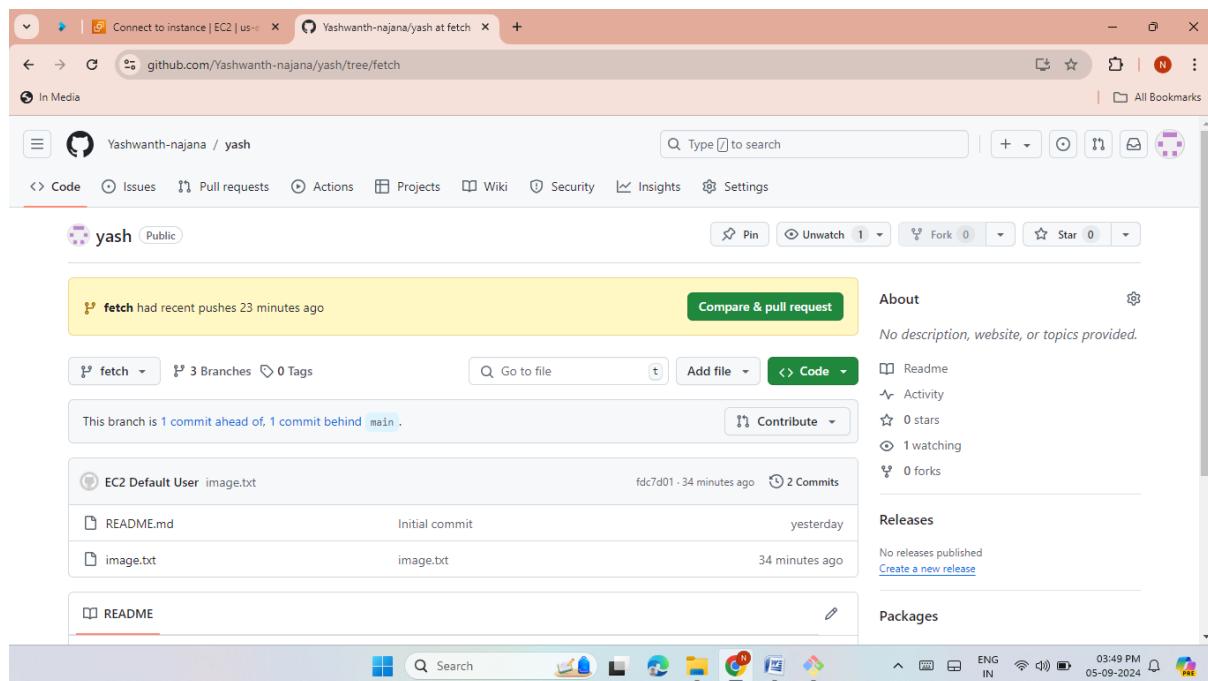
- Go to your github portal once again.
- Here see that the new changes are only available in your feature(main) branch but not in the fetch branch. The files are doc.txt,doc1.txt.



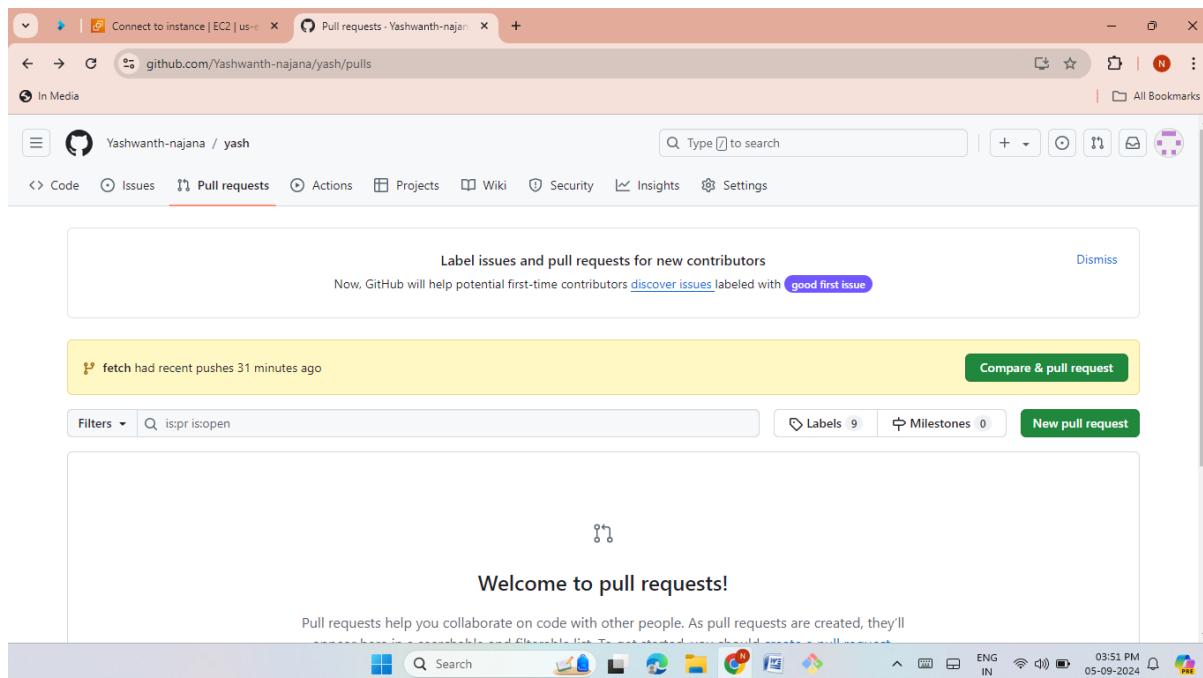


## LAB 8 : (MERGE OUR FEATURE BRANCH WITH MAIN BRANCH)

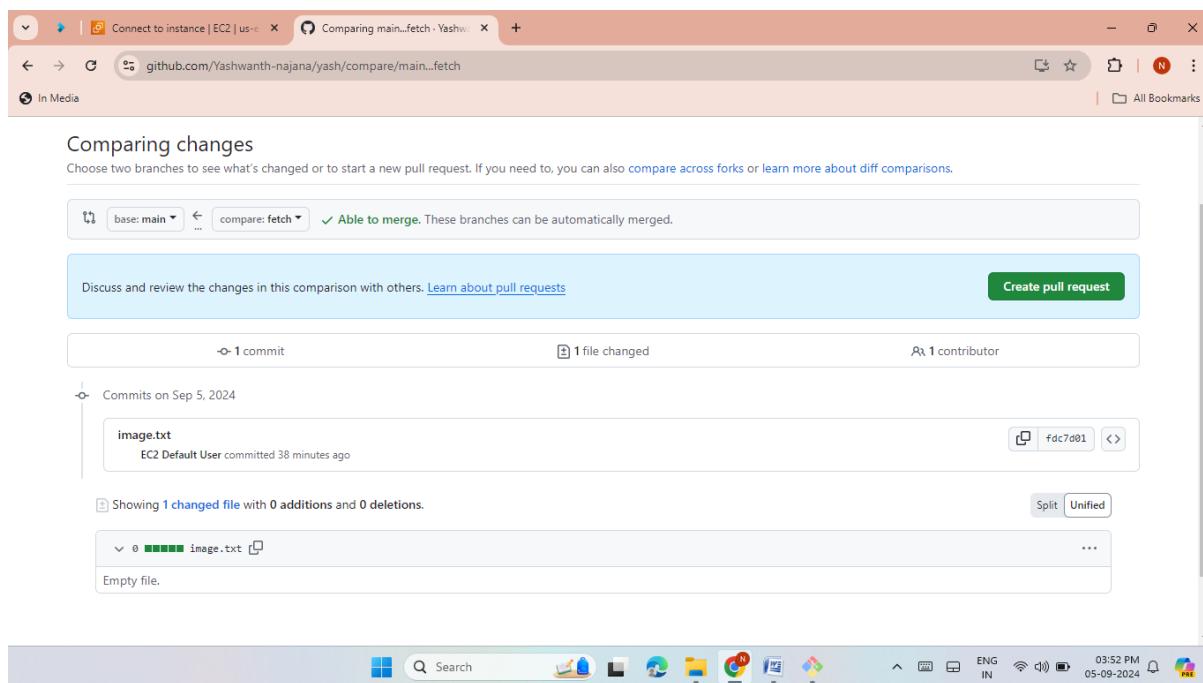
- Go to your git hub repository.
- Check the changes In your feature branch.(fetch branch)
- It have image.txt file only.



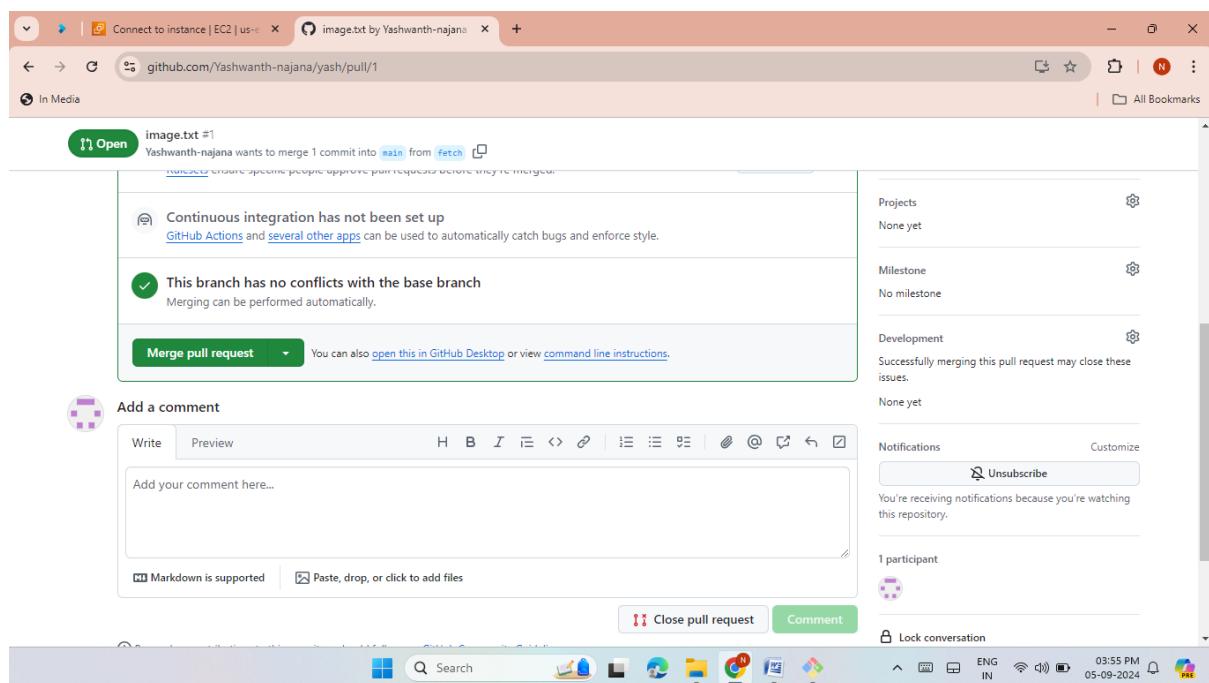
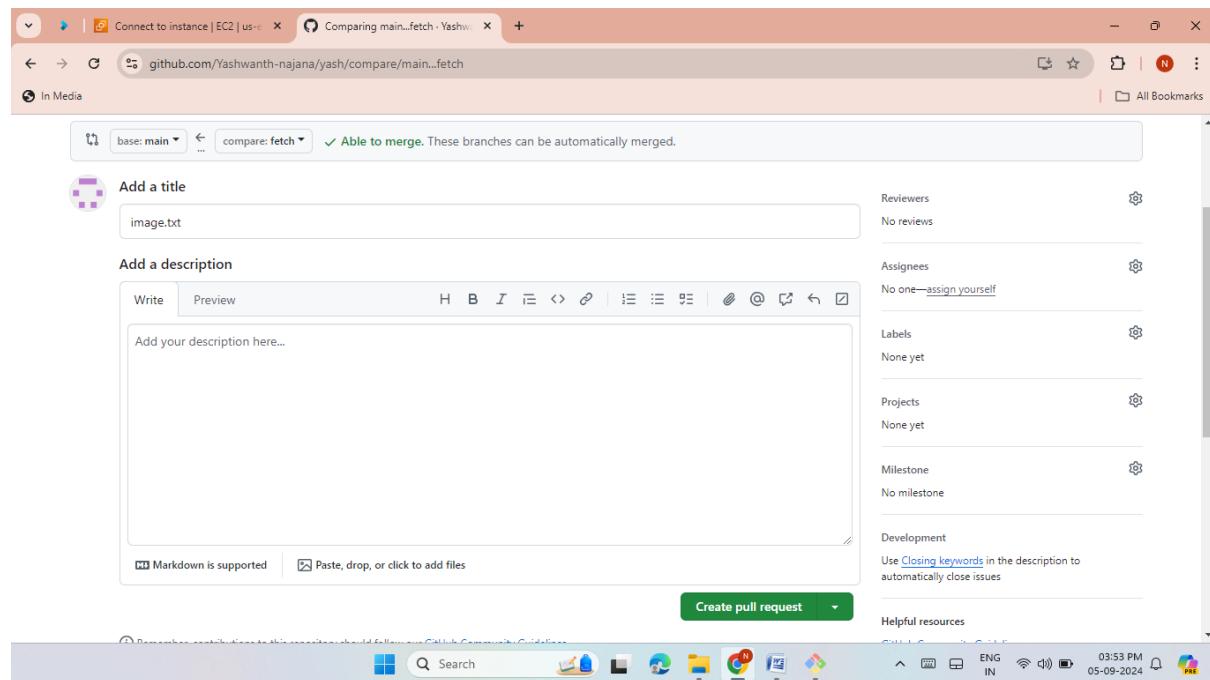
- Go to pull request tab and click on create pull request.



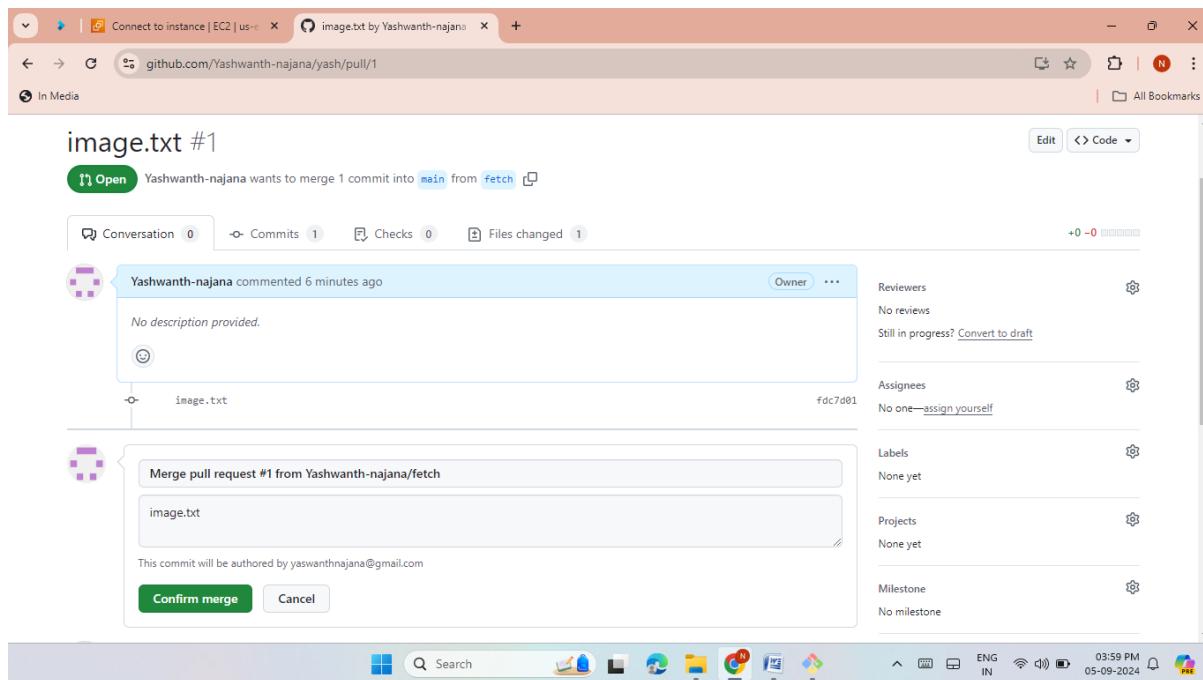
- Put your main branch (where u want to merge/destination) and the feature branch(fetch branch) in the respective blocks.



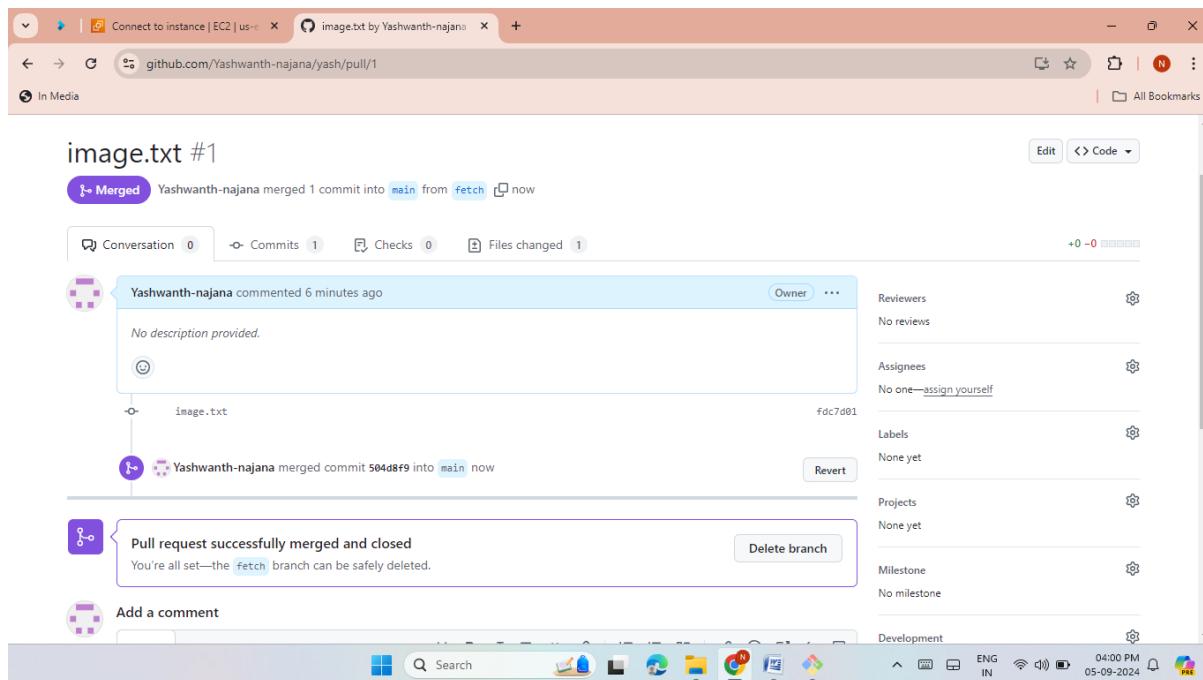
- Click on create pull request and it will ask for a comment, just click again on create pull request.



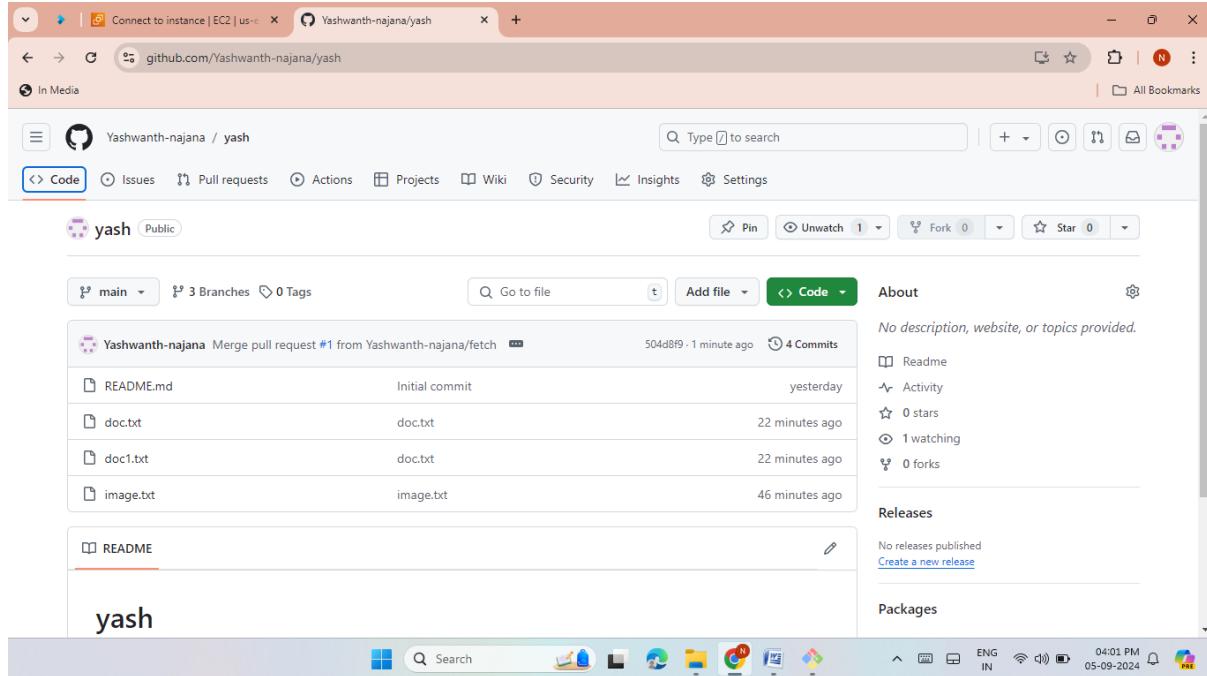
- Click on confirm merge.



- Once done, you can delete this feature branch if required or you can simply ignore it.



- Go to your code in main branch and see the changes are now visible here.
- Here the fetch branch files are added (image.txt).



## LAB 9 : (GO TO LOCAL MACHINE)

- Go to your local machine where you have the copy of your remote repo.
- Checkout to the fetch branch
- **git checkout < branch name>**
- Now run the command “**git pull**” to pull all the new changes such as branches form the remote location.

```
ec2-user@ip-172-31-22-218 ~]$ ls
yash
[ec2-user@ip-172-31-22-218 ~]$ cd yash
[ec2-user@ip-172-31-22-218 yash]$ ls
doc1.txt doc.txt README.md
[ec2-user@ip-172-31-22-218 yash]$ git checkout main
Already on 'main'.
Your branch is up to date with 'origin/main'.
[ec2-user@ip-172-31-22-218 yash]$ git checkout fetch
Switched to branch 'fetch'.
Your branch is up to date with 'origin/fetch'.
[ec2-user@ip-172-31-22-218 yash]$ ls
image.txt README.md
[ec2-user@ip-172-31-22-218 yash]$ git branch
* fetch
  main
[ec2-user@ip-172-31-22-218 yash]$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (2/2), 948 bytes | 948.00 KiB/s, done.
From https://github.com/Yashwanth-najana/yash
 80843b3..504d8f9  main      -> origin/main
Already up to date.
[ec2-user@ip-172-31-22-218 yash]$
```

- Here see that the changes are only available in your fetch branch.

The screenshot shows a GitHub compare page for branches 'fetch' and 'main'. At the top, it says 'Able to merge. These branches can be automatically merged.' Below this, there's a section for pull requests and a summary of 2 commits, 2 files changed, and 2 contributors. The commit list includes a merge pull request from 'Yashwanth-najana/main' to 'fetch' by 'Yashwanth-najana' committed 1 hour ago. The commit details show the file 'doc.txt' was modified, with a diff view showing no additions or deletions. The commit hash is 504d8f9.

The screenshot shows a GitHub repository page for the 'fetch' branch. It displays a yellow banner stating 'fetch had recent pushes 27 seconds ago'. The main area shows 3 branches and 0 tags. A message indicates this branch is 1 commit ahead of 'main'. The commit list shows 5 commits, including a merge pull request from 'Yashwanth-najana/main' to 'fetch' by 'Yashwanth-najana' committed 27 seconds ago. The commit details show the file 'doc.txt' was modified, with a diff view showing no additions or deletions. The commit hash is 72abb3f. The repository page also includes sections for About (no description), Releases (no releases), and Packages (no packages). The status bar at the bottom shows the date as 05-09-2024 and time as 05:32 PM.