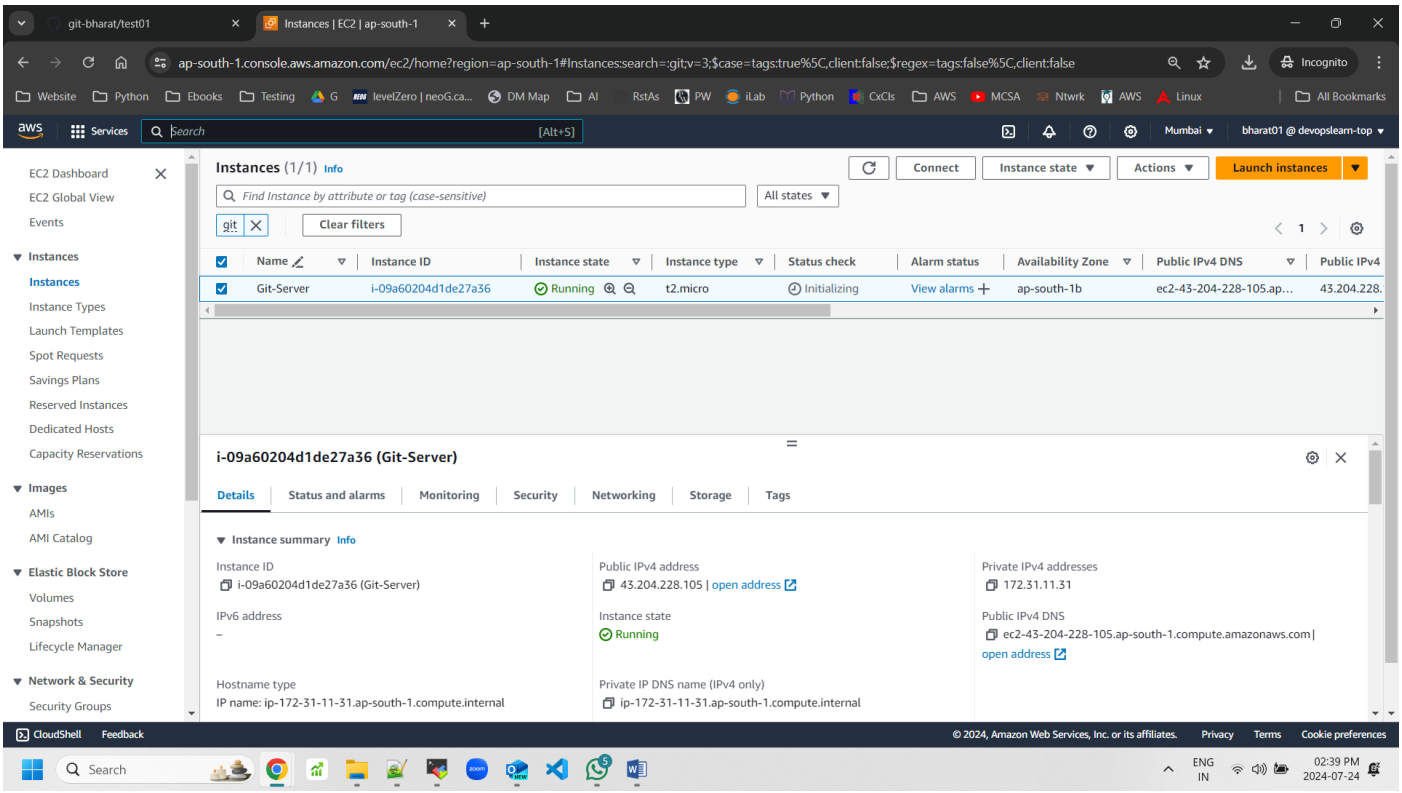
Setup on AWS

1>Launched AWS EC2 instance



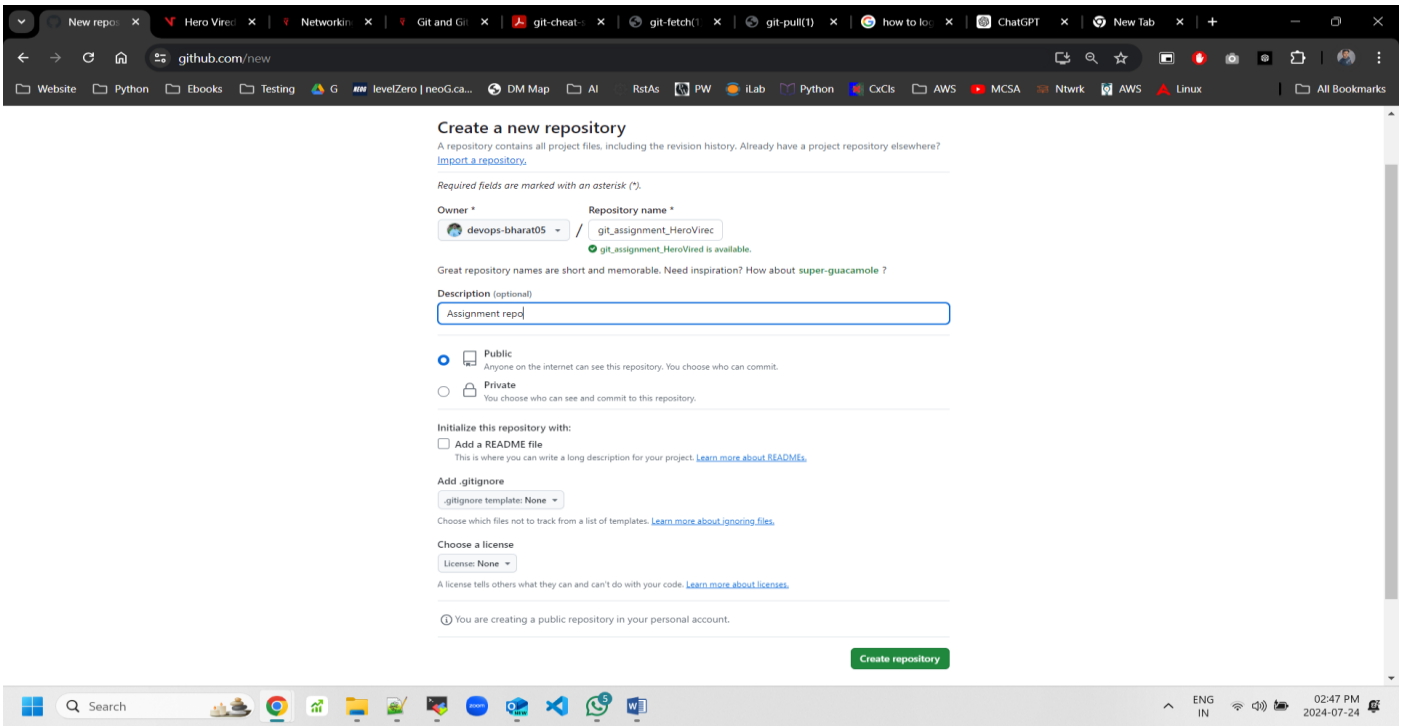
2>Install git on the aws-liunx server

 # sudo yum install git –y

Step-by-Step Instructions:

a. Create a repository name: git_assignment_HeroVired

1. Go to GitHub and create a new repository named `git_assignment_HeroVired`.



b. Create a `dev` branch and add the given code

1. Clone the repository to your local machine

#git clone https://github.com/devops-bharat05/git_assignment_HeroVired.git

#cd git_assignment_HeroVired

```
[root@ip-172-31-11-31 git_assignment_HeroVired]# git status
On branch main
nothing to commit, working tree clean
[root@ip-172-31-11-31 git_assignment_HeroVired]# git checkout dev
Switched to branch 'dev'
[root@ip-172-31-11-31 git_assignment_HeroVired]# git status
On branch dev
nothing to commit, working tree clean
[root@ip-172-31-11-31 git_assignment_HeroVired]# git remote add git-cal git@github.com:devops-bharat05/git_assignment_HeroVired.git
[root@ip-172-31-11-31 git_assignment_HeroVired]# git push -u git-cal main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 348 bytes | 348.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To github.com:devops-bharat05/git_assignment_HeroVired.git
 * [new branch]      main -> main
branch 'main' set up to track 'git-cal/main'.
[root@ip-172-31-11-31 git_assignment_HeroVired]# git checkout dev
Already on 'dev'
[root@ip-172-31-11-31 git_assignment_HeroVired]# git push -u git-cal dev
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:      https://github.com/devops-bharat05/git_assignment_HeroVired/pull/new/dev
remote:
To github.com:devops-bharat05/git_assignment_HeroVired.git
 * [new branch]      dev -> dev
branch 'dev' set up to track 'git-cal/dev'.
[root@ip-172-31-11-31 git_assignment_HeroVired]#
```

2. Create and switch to the `dev` branch:

git checkout -b dev

3. Add the provided code to a file named `calculator.py`:

4. Commit the changes:

#git add calculator.py

git commit -m "Add initial code for CalculatorPlus"

c. Merge this branch with the main branch and make a release of version 1 of the 'calculator plus app'

1. Push the `dev` branch to the remote repository:

#git push origin dev

2. Switch to the `main` branch:

#git checkout main

3. Merge the `dev` branch into the `main` branch:

git merge dev

4. Push the changes to the remote repository:

git push origin main

5. Create a release for version 1:

- Go to the repository on GitHub.

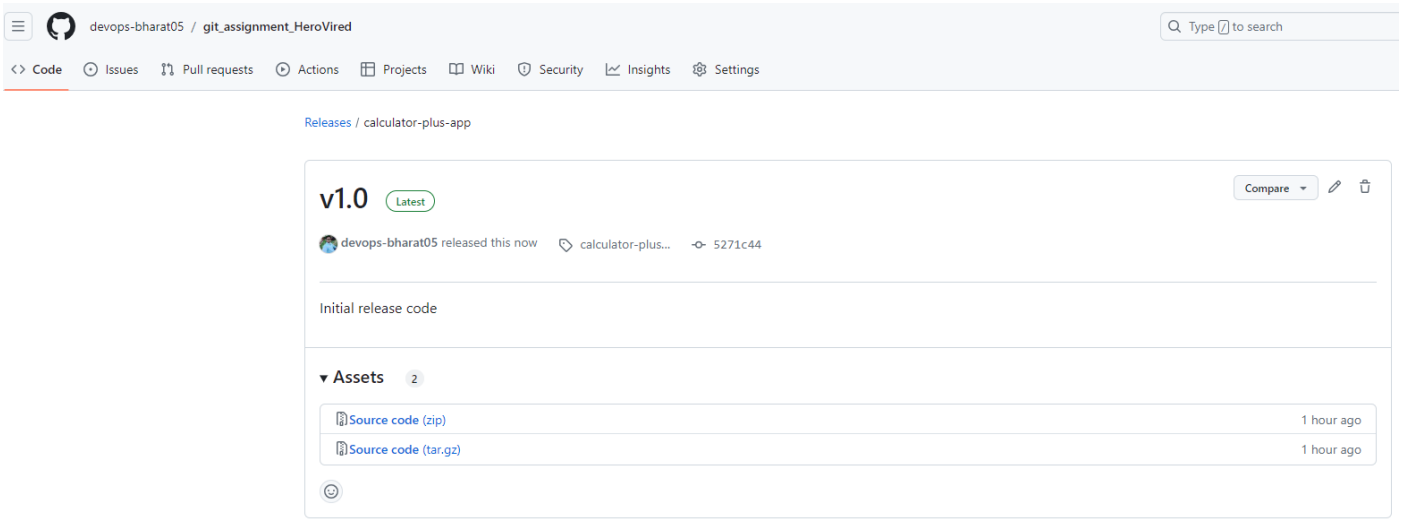
- Navigate to the "Releases" tab.

- Click on "Draft a new release".

- Set the tag version as `v1.0`.

- Provide a title and description.

- Click "Publish release".



d. Add any of your classmates as collaborators

1. Go to the repository on GitHub.
2. Navigate to the "Settings" tab.
3. Click on "Collaborators".
4. Add the username of your classmate and invite them.

e. Implement a feature by creating a new branch called 'feature-sqrt'

1. Create and switch to the `**feature-sqrt**` branch:

   # git checkout -b feature-sqrt

2. Add the `square_root` function to `calculator.py` (already added above):

   def square_root(self, x):

   return math.sqrt(x)

3. Commit the changes:

   #git add calculator.py

   # git commit -m "Add square root feature"

4. Push the `**feature-sqrt**` branch to the remote repository:

   # git push origin **feature-sqrt**

f. Add the `sqrt` code to it (already done in the above step)

g. Bug fix in the divide function

1. Switch back to the `dev` branch:

   #git checkout dev

2. Update the `divide` function to handle division by zero:

   def divide(self, a, b):

   if b == 0:

   raise ValueError("Cannot divide by zero.")

   return a / b

3. Commit the bug fix:

   #git add calculator.py

   #git commit -m "Fix divide function to handle division by zero"

```
[root@ip-172-31-11-31 git_assignment_HeroVired]# git checkout dev
Switched to branch 'dev'
Your branch is up to date with 'git-cal/dev'.
[root@ip-172-31-11-31 git_assignment_HeroVired]# ls
calculator.py
[root@ip-172-31-11-31 git_assignment_HeroVired]# vi calculator.py
[root@ip-172-31-11-31 git_assignment_HeroVired]# git status
On branch dev
Your branch is up to date with 'git-cal/dev'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   calculator.py

no changes added to commit (use "git add" and/or "git commit -a")
[root@ip-172-31-11-31 git_assignment_HeroVired]# git add calculator.py
[root@ip-172-31-11-31 git_assignment_HeroVired]# git commit -m "Fix divide function to handle division by zero"
```

4. Push the changes to the remote repository:

   #git push origin dev

5. Switch back to the `**feature-sqrt**` branch and rebase with `dev` to keep it up-to-date:

   #git checkout **feature-sqrt**

   # git rebase dev

6. Resolve any merge conflicts if they arise and continue the rebase:

   #git add calculator.py

   #git rebase --continue

7. Push the updated `**feature-sqrt**` branch:

   #git push origin **feature-sqrt** --force

h. Create a pull request targeting the main branch

1. Go to the repository on GitHub.
2. Navigate to the "Pull requests" tab.
3. Click on "New pull request".
4. Select the `**feature-sqrt**` branch as the compare branch and `main` as the base branch.

5. Create the pull request.

i. Request a code review from a team member

1. Add a reviewer to the pull request by selecting your classmate or team member.

j. Merge the `feature-sqrt` branch into the `dev` branch after approval

1. Once the pull request is approved, merge it into the `dev` branch.

k. Testing and merging into the `main` branch, then create `version 2` release

1. Switch to the `dev` branch:

   #git checkout dev

2. Merge the `**feature-sqrt**` branch into the `dev` branch:

   #git merge **feature-sqrt**

3. Test the application in the `dev` branch to ensure all features work correctly.

4. Switch to the `main` branch:

   # git checkout main

5. Merge the `dev` branch into the `main` branch:

   # git merge dev

6. Push the changes to the remote repository:

   #git push origin main

7. Create a release for version 2:

   - Go to the repository on GitHub.

   - Navigate to the "Releases" tab.

   - Click on "Draft a new release".

   - Set the tag version as **`v2.0`.**

   - Provide a title and description.

   - Click "Publish release".