Git:

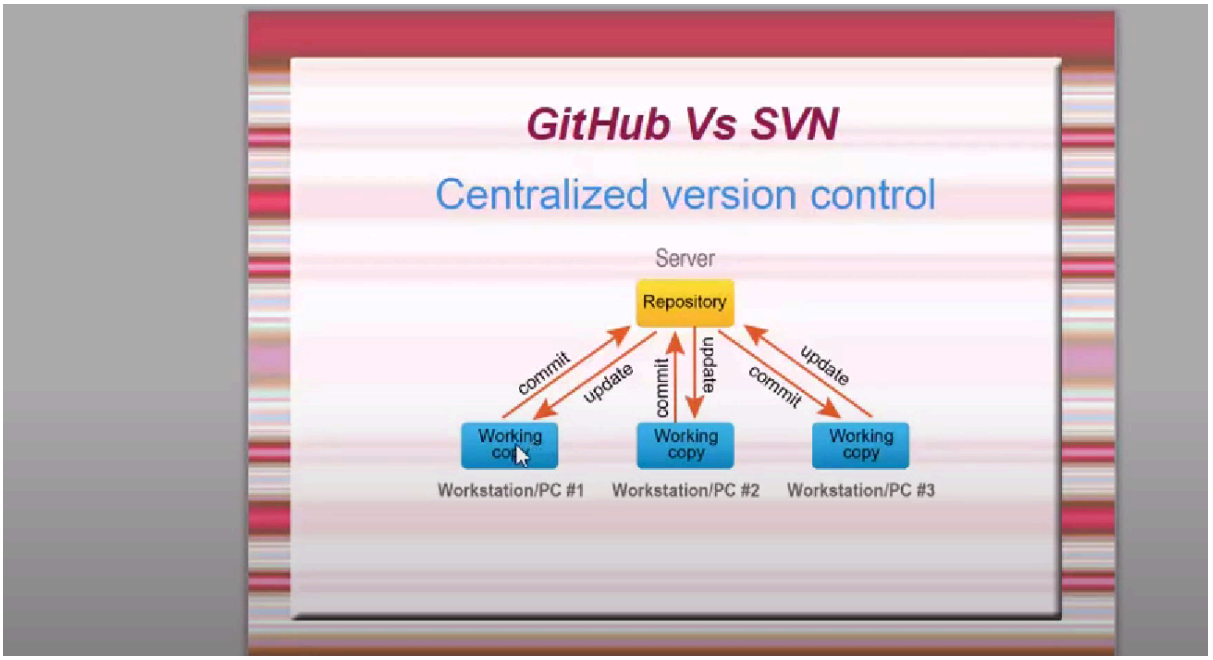
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Difference between git and github:

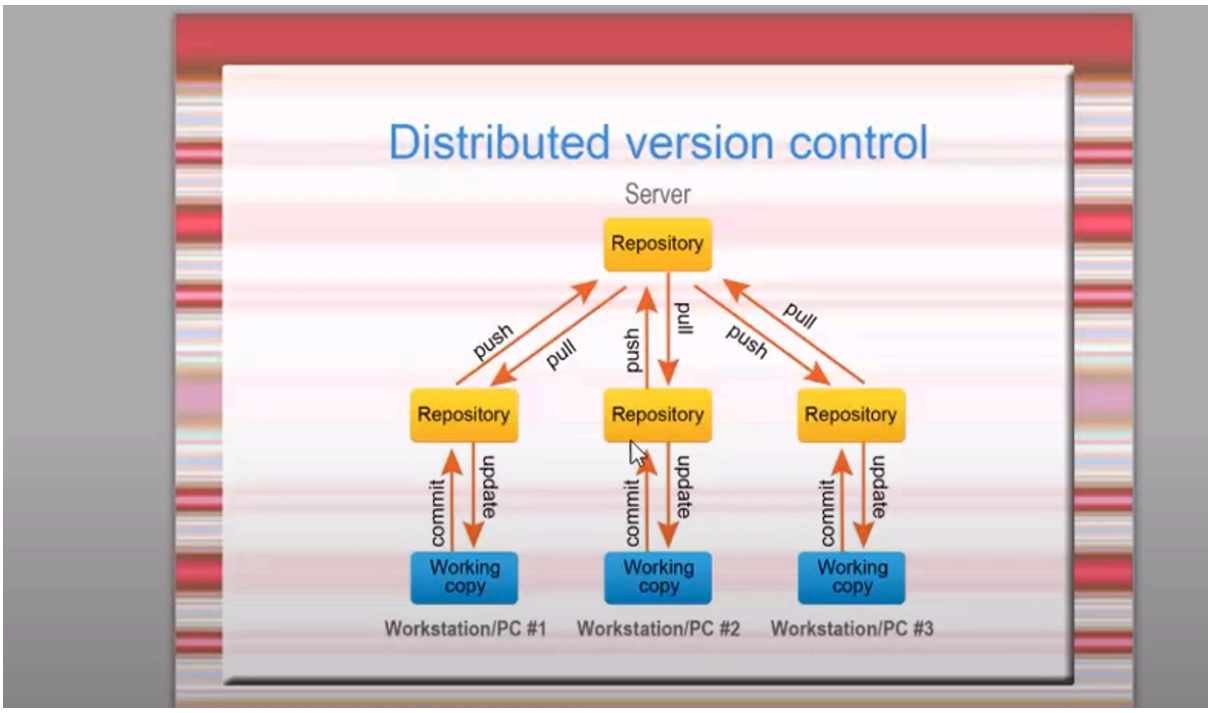
Git is a version control system that allows you to track changes to your code, and it is a great way to keep track of your work. It is also a great way to share your code with others, and it can be very useful when you are working on a project that involves a lot of collaboration.

Github is a web-based Git repository, that is. Cloud-based storage is offered by this hosting company. Git's distributed version control and source code management capabilities are all provided by GitHub, along with some additional features. Utilising Git facilitates collaboration. The public can access GitHub repositories as well. GitHub serves as a networking platform for online professionals since developers from all over the world may engage, contribute to one another's code, and edit or improve it. Social coding is another name for the process of participation and engagement.

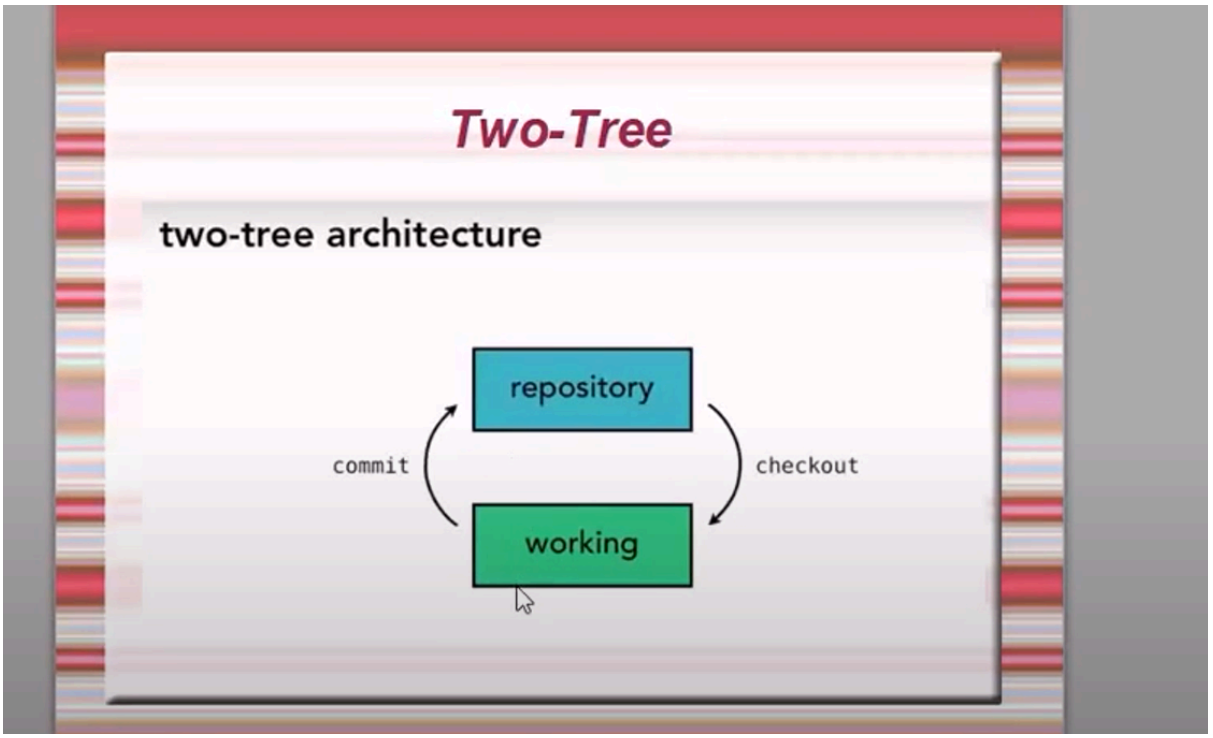
Differencebetween git and svn:



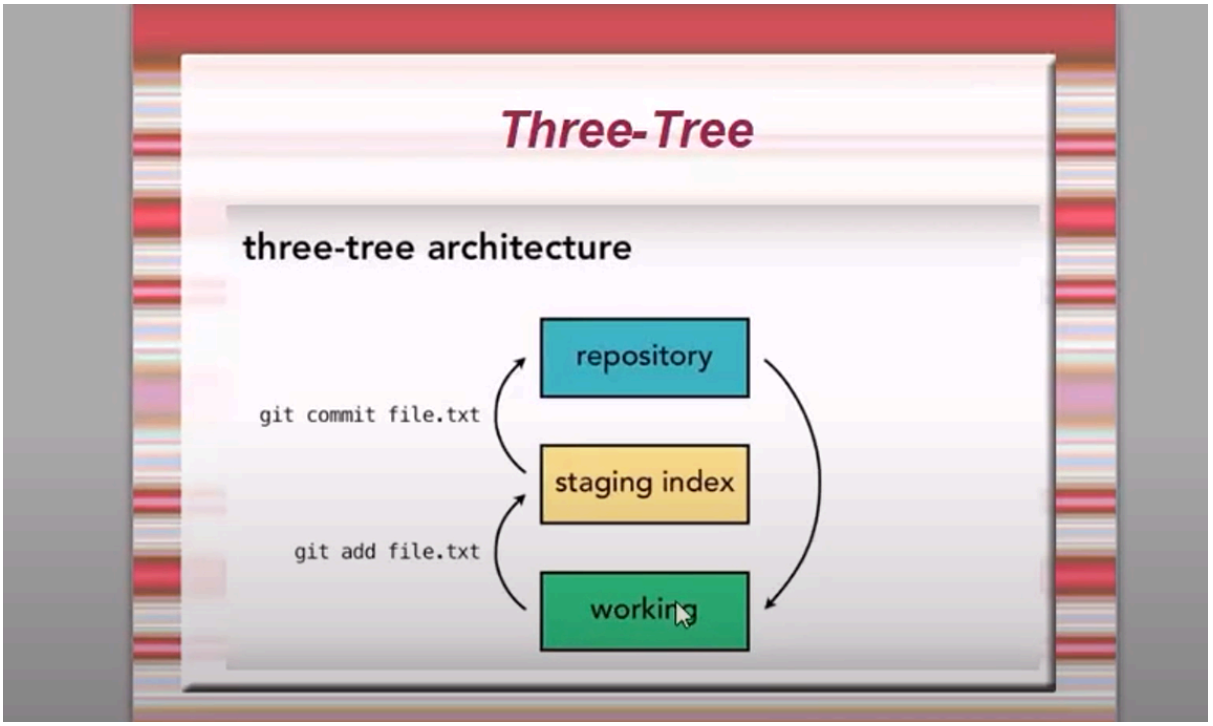
u must have internet connection.



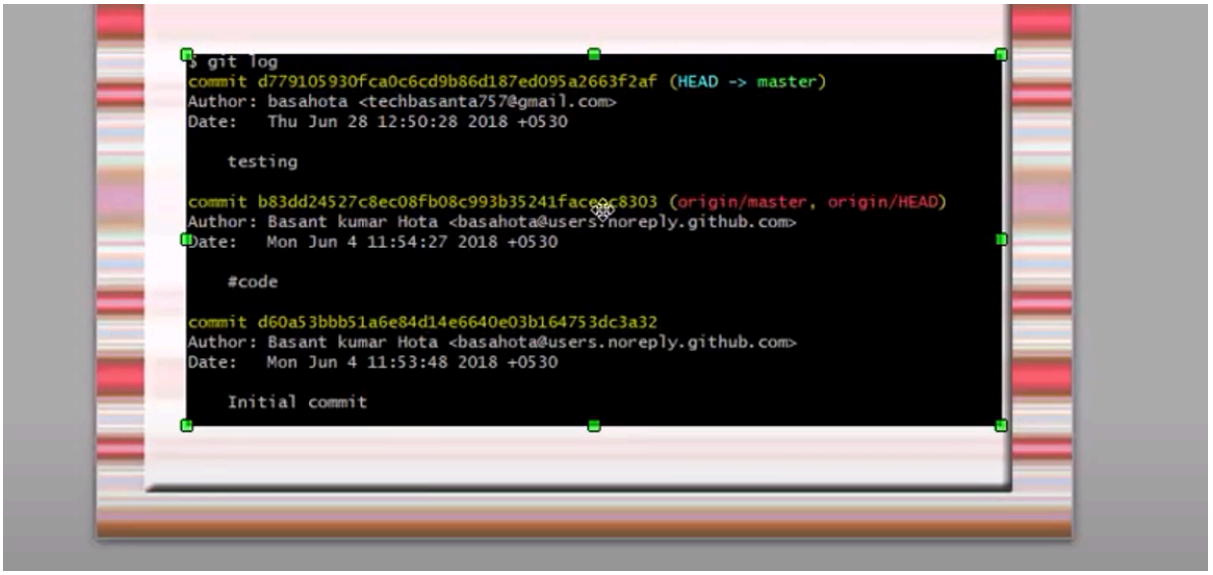
SVN:



GIT:

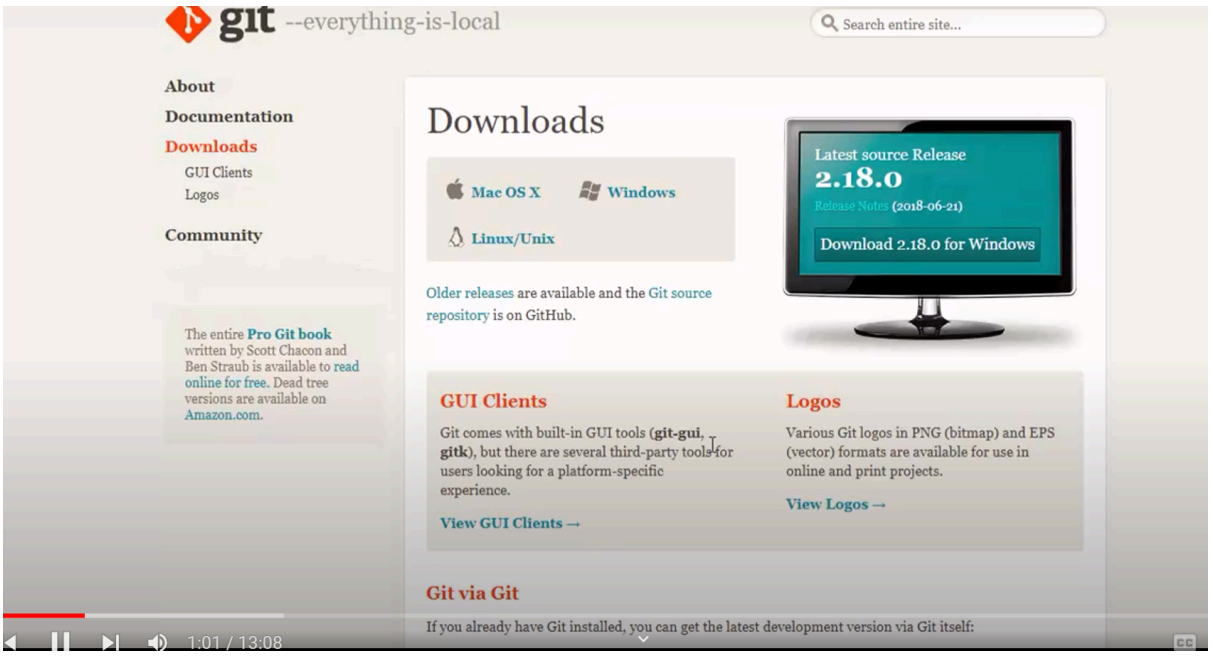


Every changes there will be a version.



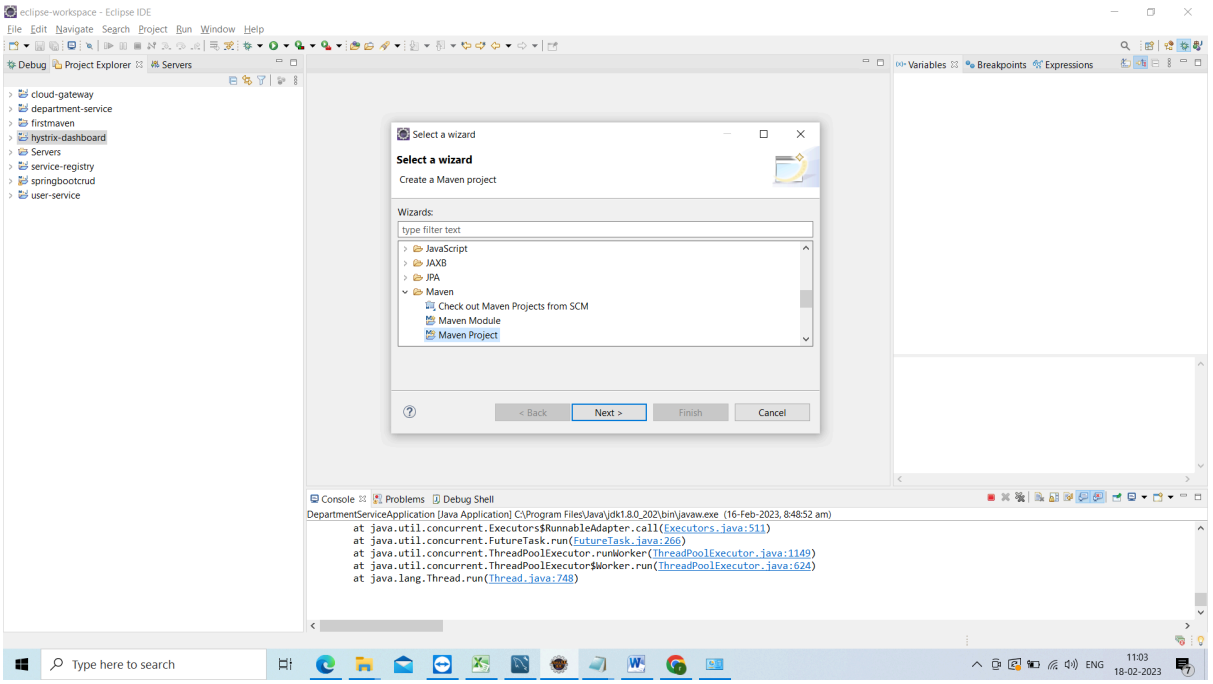
Need to install git using below command

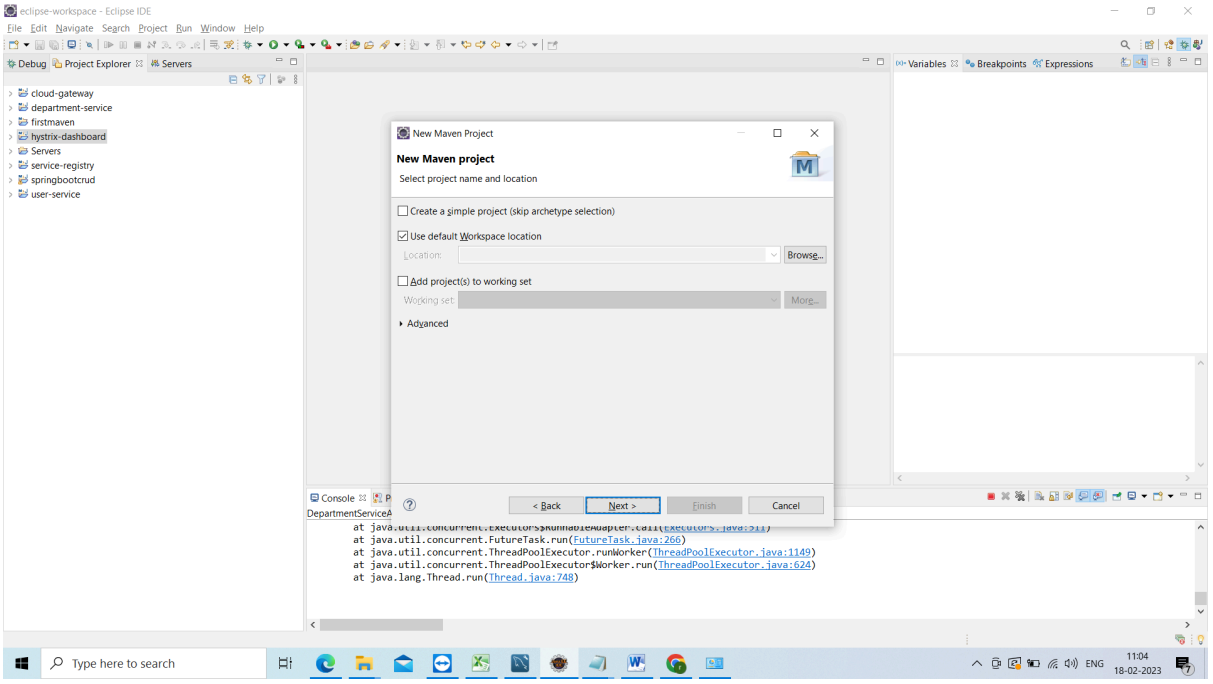
https://git-scm.com/downloads

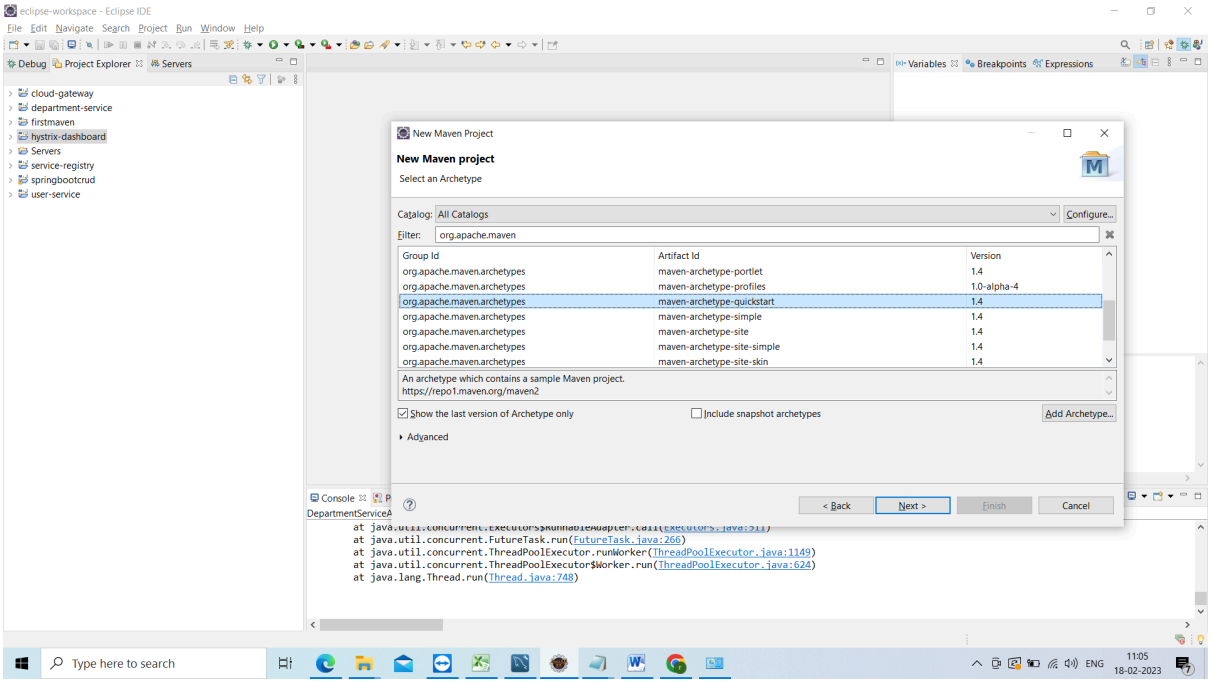


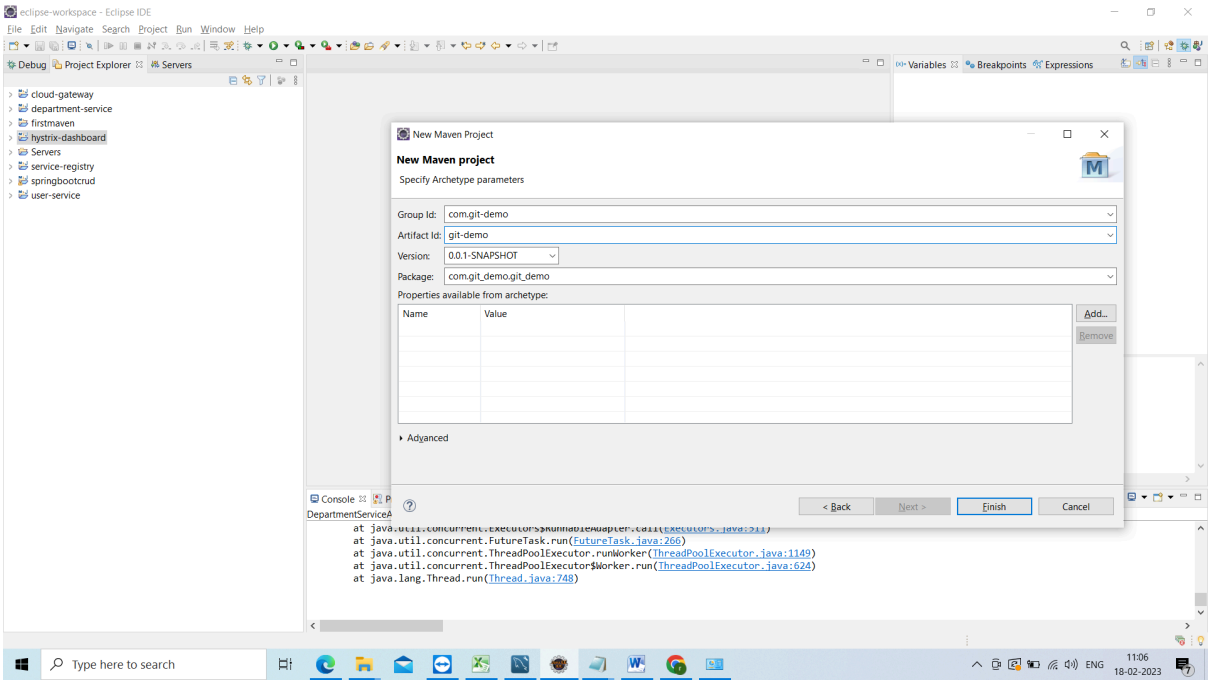
Click on download2.18.0 for windows

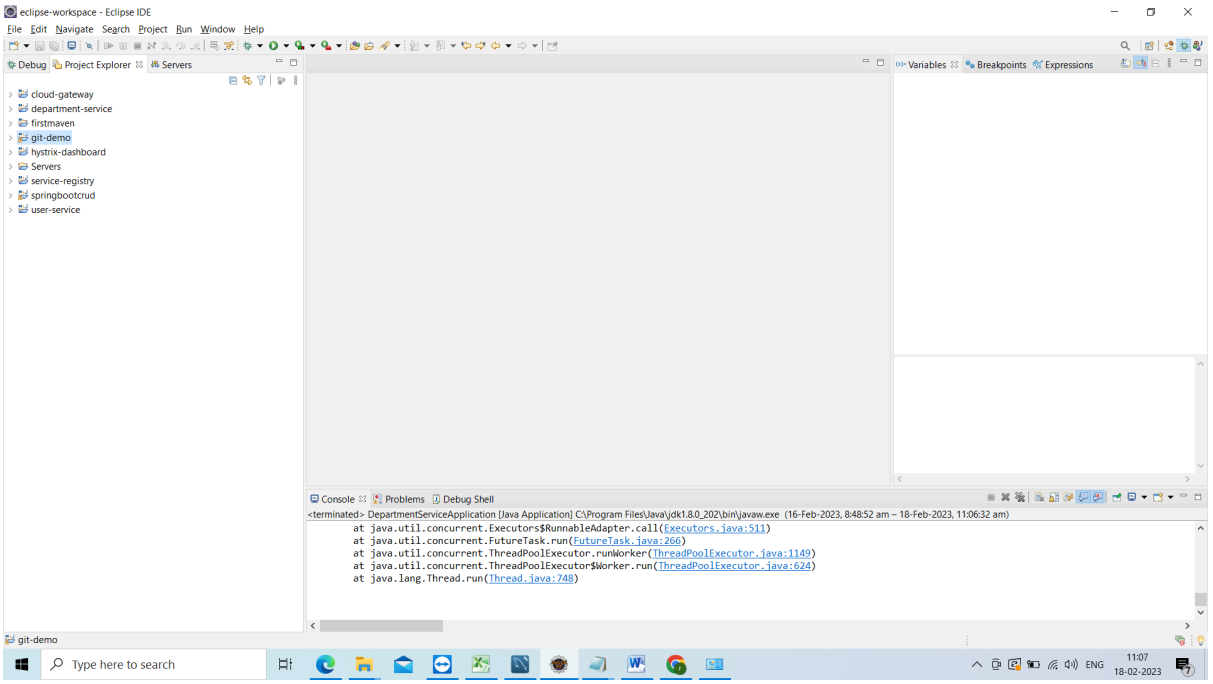
Lets create the maven project in eclipse






eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Debug Project Explorer Servers
cloud-gateway
department-service
firstmaven
hystrix-dashboard
Servers
service-registry
springbootcrud
user-service
Variables Breakpoints Expressions
New Maven Project
New Maven project
Select an Archetype
Catalog: All Catalogs
Configure...
Filter: org.apache.maven
Group Id
Artifact Id
Version
org.apache.maven.archetypes maven-archetype-portlet 1.4
org.apache.maven.archetypes maven-archetype-profiles 1.0-alpha-4
org.apache.maven.archetypes maven-archetype-quickstart 1.4
org.apache.maven.archetypes maven-archetype-simple 1.4
org.apache.maven.archetypes maven-archetype-site 1.4
org.apache.maven.archetypes maven-archetype-site-simple 1.4
org.apache.maven.archetypes maven-archetype-site-skin 1.4
An archetype which contains a sample Maven project.
https://repo1.maven.org/maven2
Show the last version of Archetype only
Include snapshot archetypes
Add Archetype...
Advanced
< Back
Next >
Finish
Cancel
Console
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at java.lang.Thread.run(Thread.java:748)
Type here to search
ENG
11:05
18-02-2023

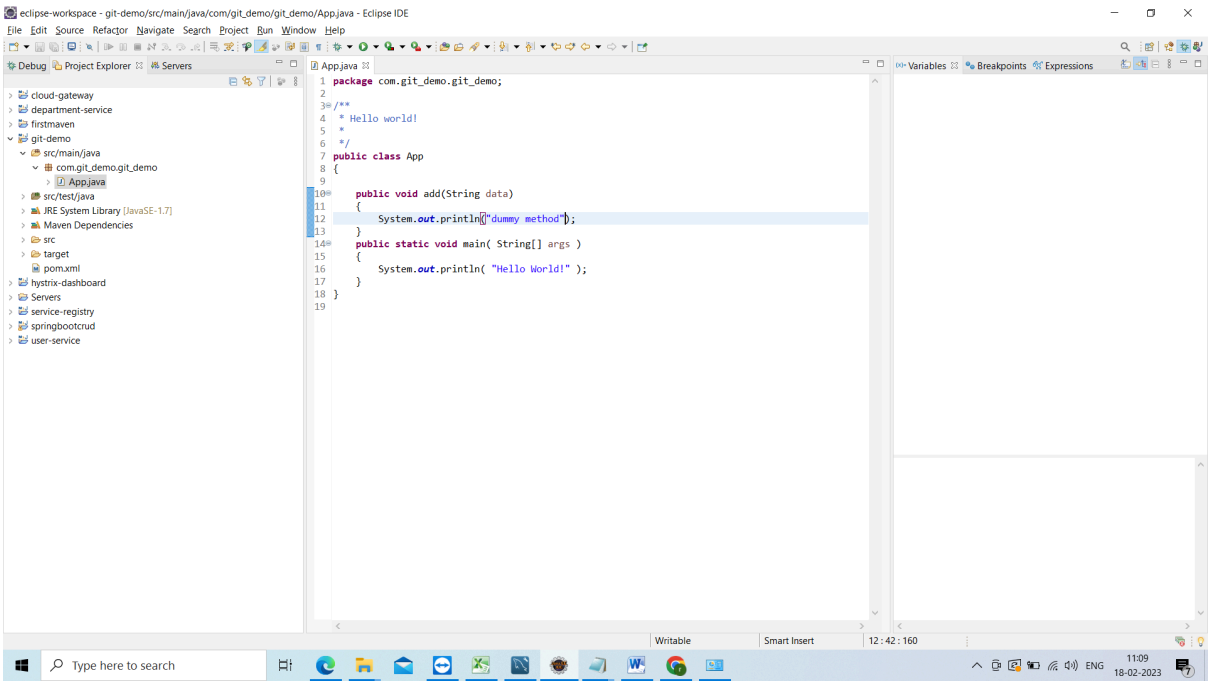

eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Debug Project Explorer Servers
cloud-gateway
department-service
firstmaven
hystrix-dashboard
Servers
service-registry
springbootcrud
user-service
Variables Breakpoints Expressions
New Maven Project
New Maven project
Specify Archetype parameters
Group Id: com.git-demo
Artifact Id: git-demo
Version: 0.0.1-SNAPSHOT
Package: com.git_demo.git_demo
Properties available from archetype:
Name
Value
Add...
Remove
Advanced
< Back
Next >
Finish
Cancel
Console
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at java.lang.Thread.run(Thread.java:748)
Type here to search
ENG
11:06
18-02-2023



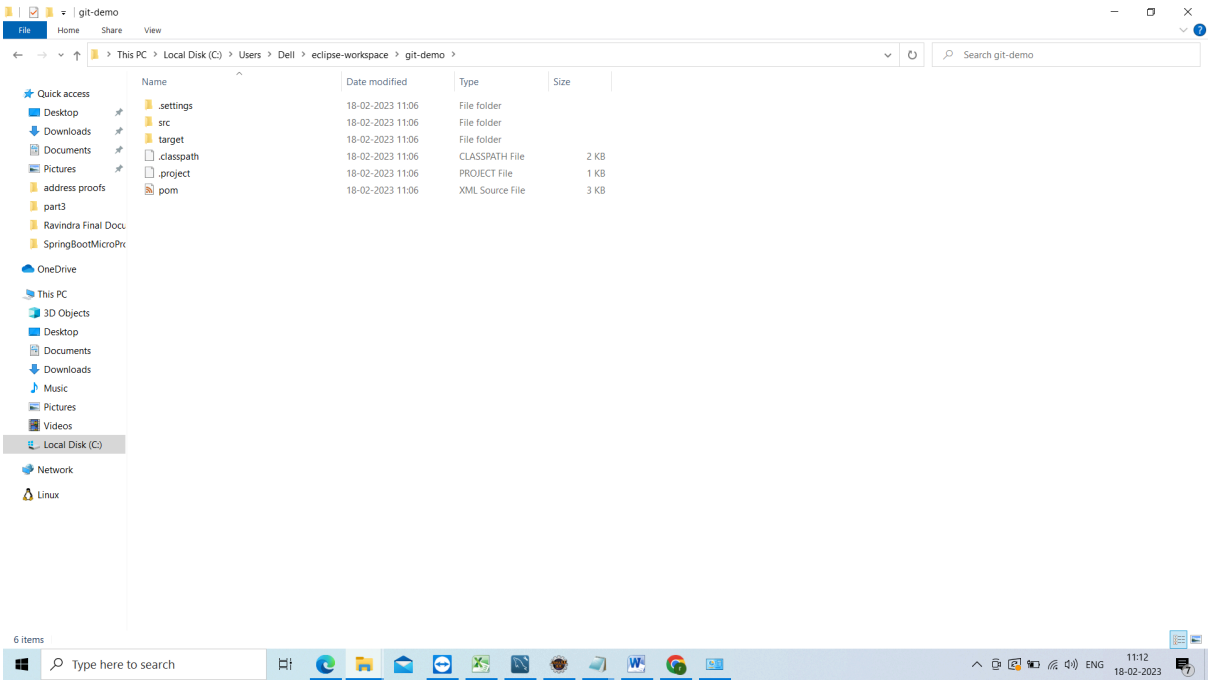
Project has been created.

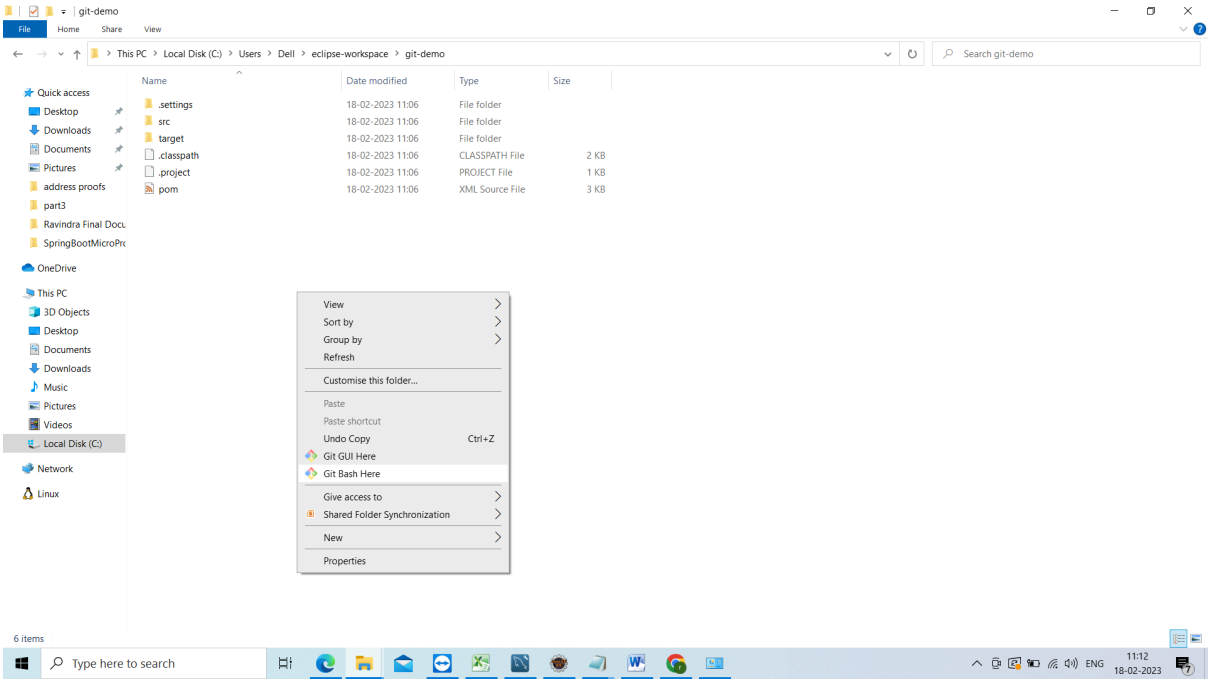
Create some dummy method

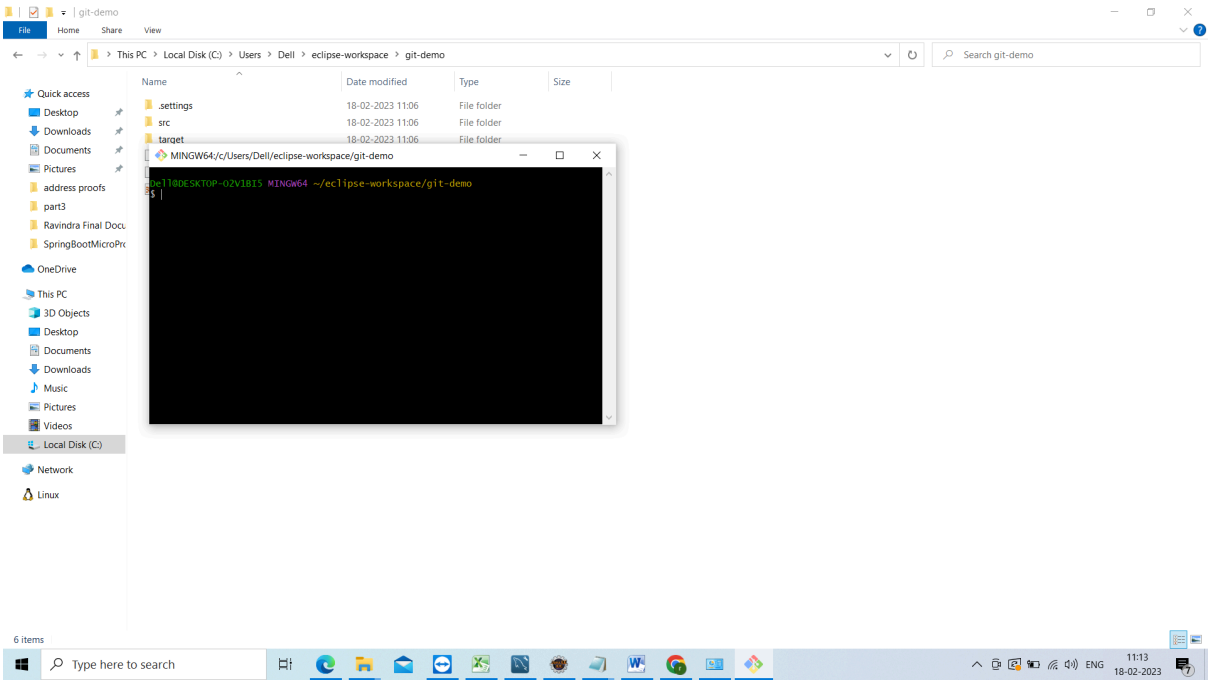
In APP.java



Now go to the project directory and do the git bash here


git-demo
File
Home
Share
View
This PC > Local Disk (C:) > Users > Dell > eclipse-workspace > git-demo >
Search git-demo
Quick access
Desktop
Downloads
Documents
Pictures
address proofs
part3
Ravindra Final Docu
SpringBootMicroPr
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Network
Linux
Name
Date modified
Type
Size
.settings 18-02-2023 11:06 File folder
src 18-02-2023 11:06 File folder
target 18-02-2023 11:06 File folder
.classpath 18-02-2023 11:06 CLASSPATH File 2 KB
.project 18-02-2023 11:06 PROJECT File 1 KB
pom 18-02-2023 11:06 XML Source File 3 KB
6 items
Type here to search
ENG
11:12
18-02-2023

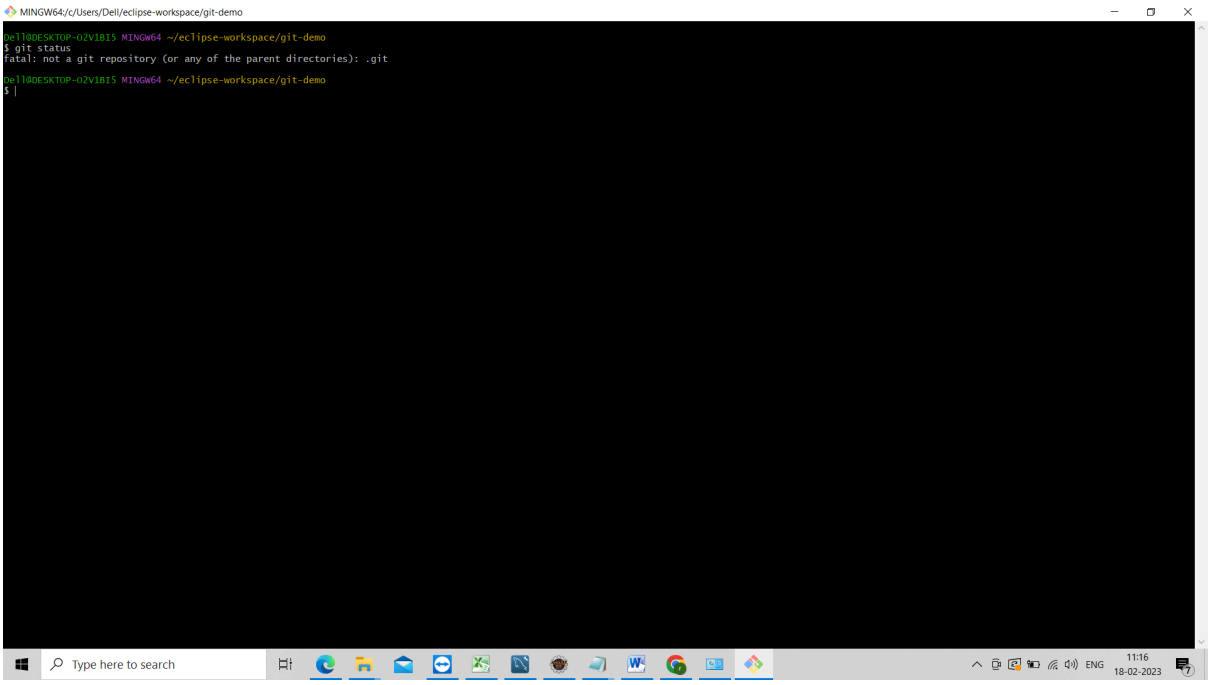

git-demo
File
Home
Share
View
This PC > Local Disk (C:) > Users > Dell > eclipse-workspace > git-demo
Search git-demo
Quick access
Desktop
Downloads
Documents
Pictures
address proofs
part3
Ravindra Final Docu
SpringBootMicroPr
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Network
Linux
Name
Date modified
Type
Size
.settings 18-02-2023 11:06 File folder
src 18-02-2023 11:06 File folder
target 18-02-2023 11:06 File folder
.classpath 18-02-2023 11:06 CLASSPATH File 2 KB
.project 18-02-2023 11:06 PROJECT File 1 KB
pom 18-02-2023 11:06 XML Source File 3 KB
View
Sort by
Group by
Refresh
Customise this folder...
Paste
Paste shortcut
Undo Copy Ctrl+Z
Git GUI Here
Git Bash Here
Give access to
Shared Folder Synchronization
New
Properties
6 items
Type here to search
ENG
11:12
18-02-2023



Now check any local repository created or not

For that need to use

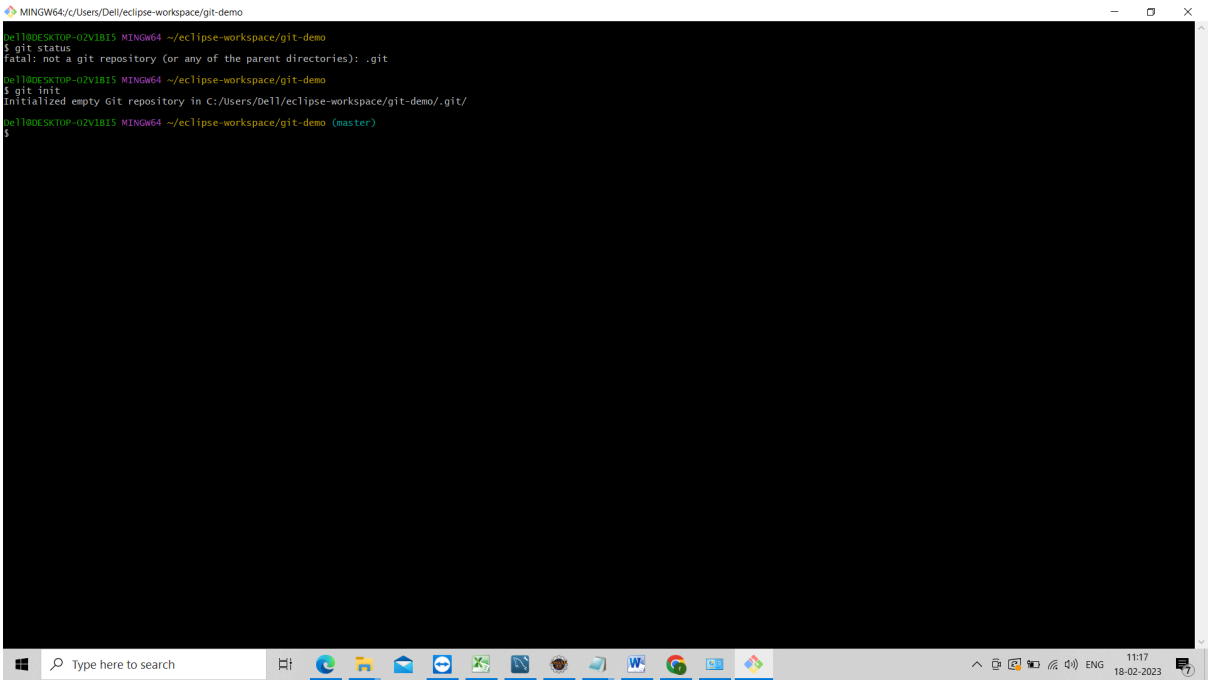
Git status



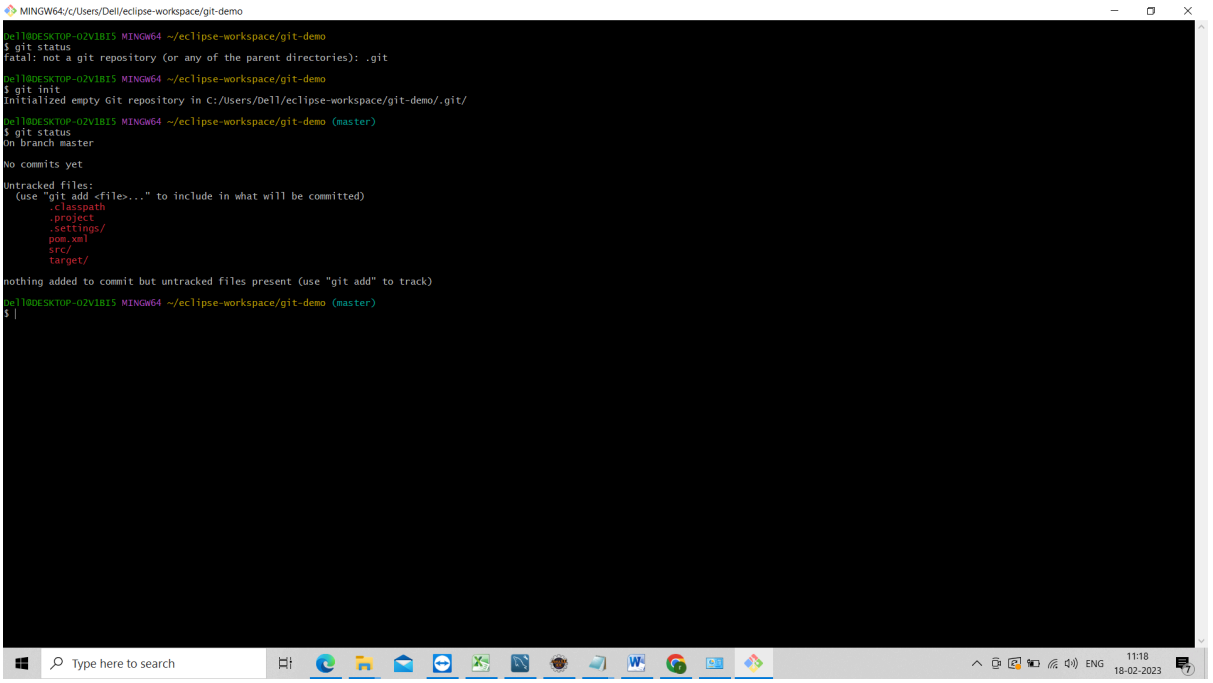
Not created

To initialize the local repository we need to use command called

Git init



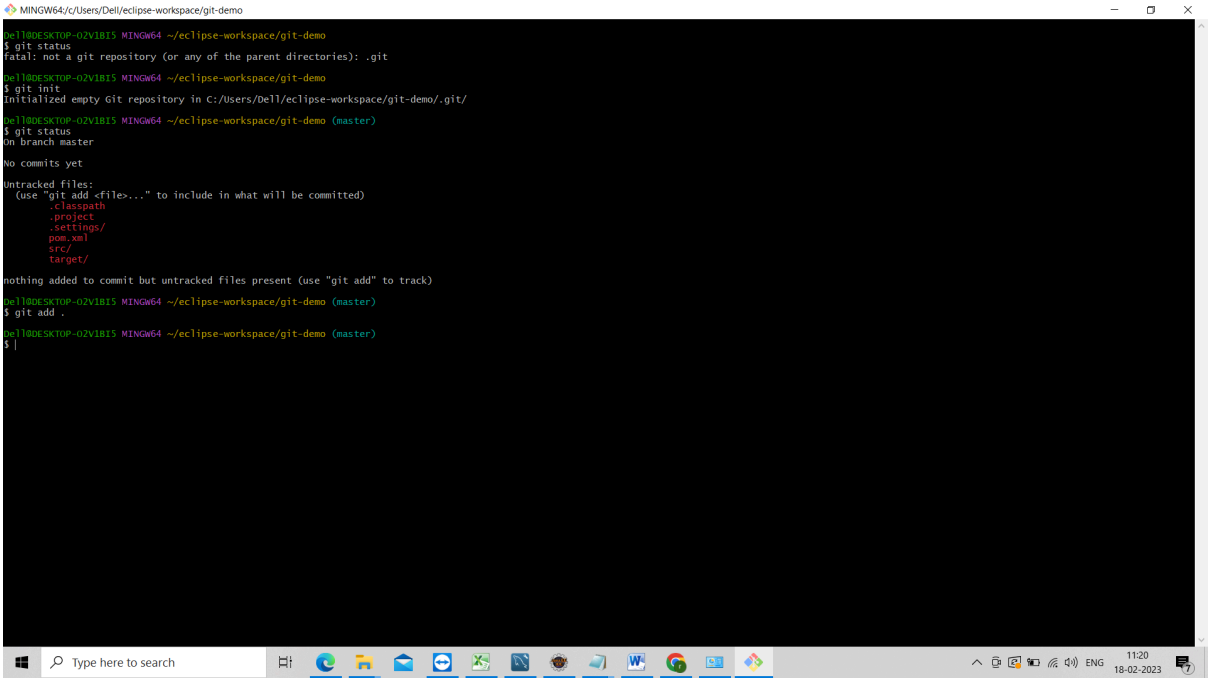
Now check changes in the code



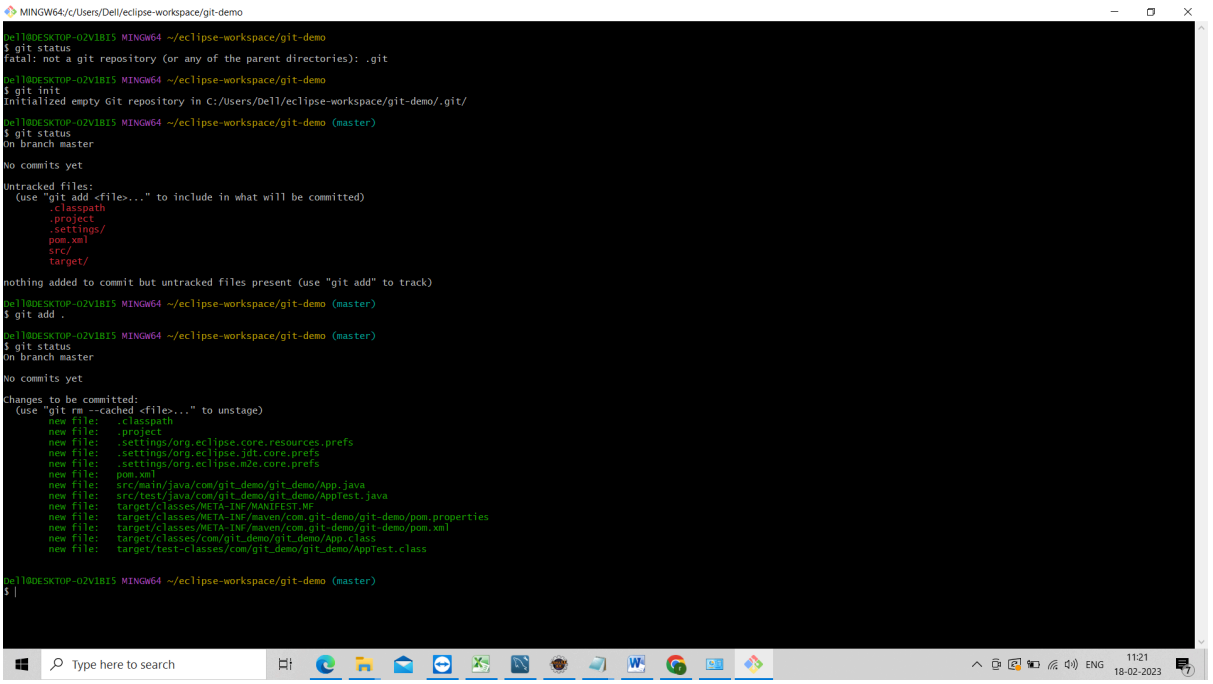
Now we need to add those changes to staging area

Git add .

It will add all the files to the staging area

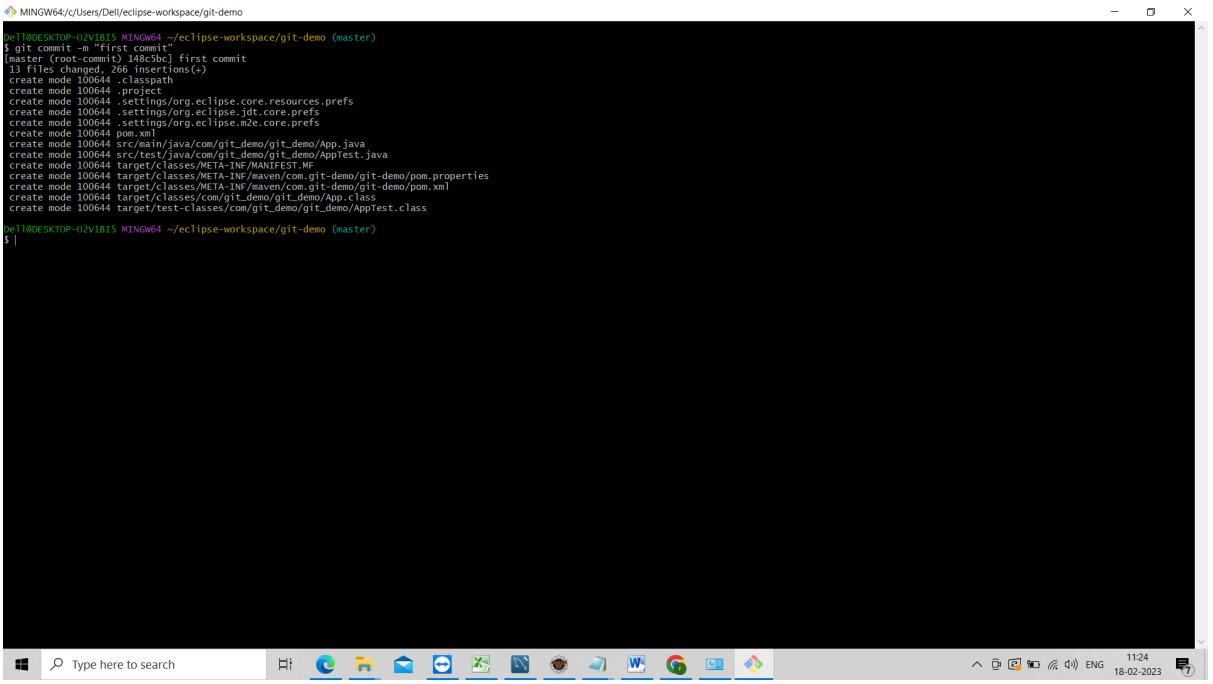


Now check the git status

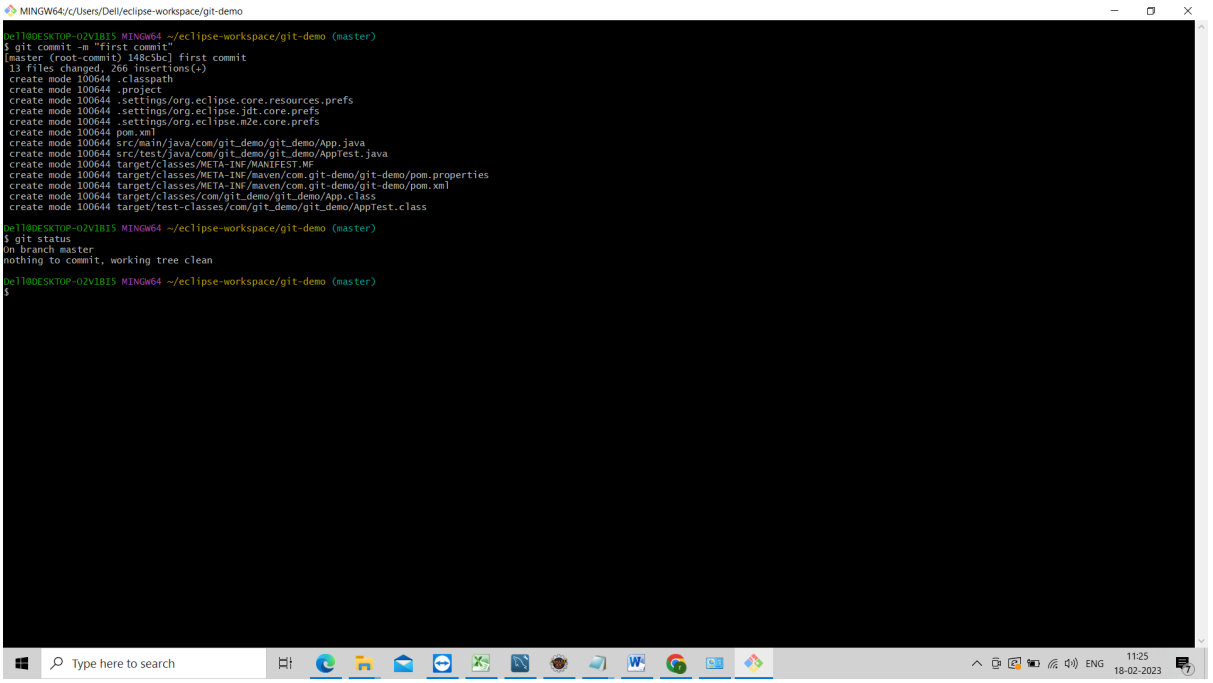


Now add the all the files to the local repository for that need to use command called

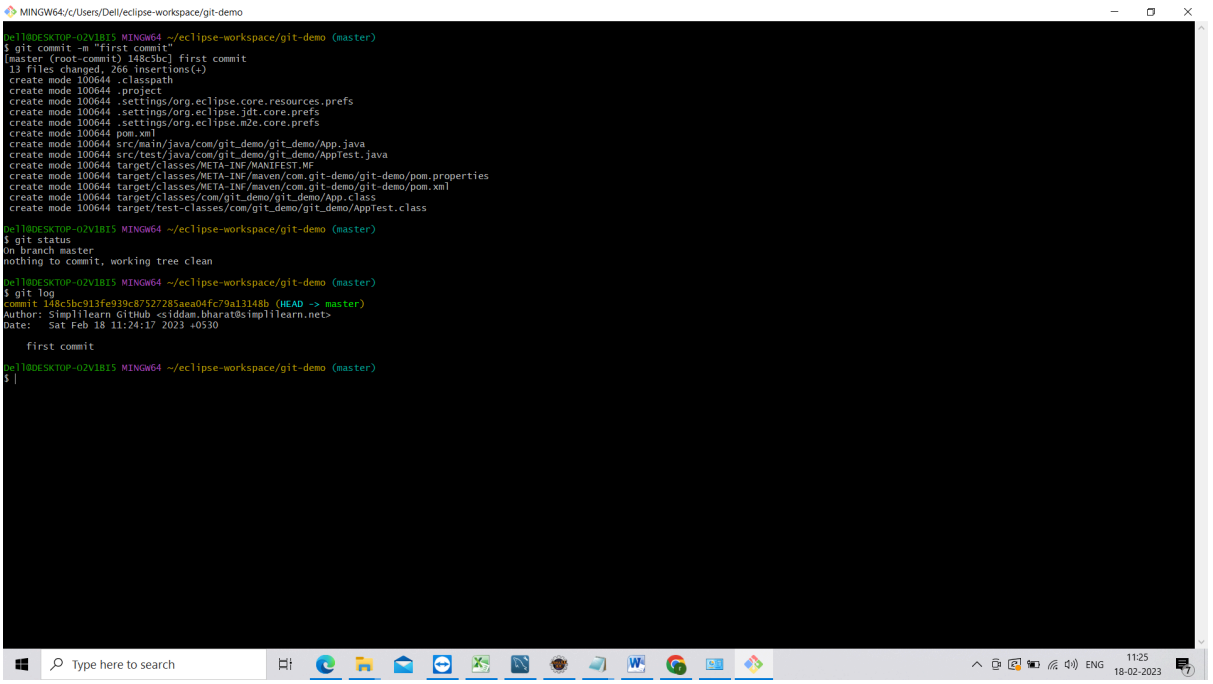
Git commit –m “commit message any thing”



Now check the git staus



Now check git log



Checks has been created.

Now go to code do some changes

Now go to code do some changes

And again check git status

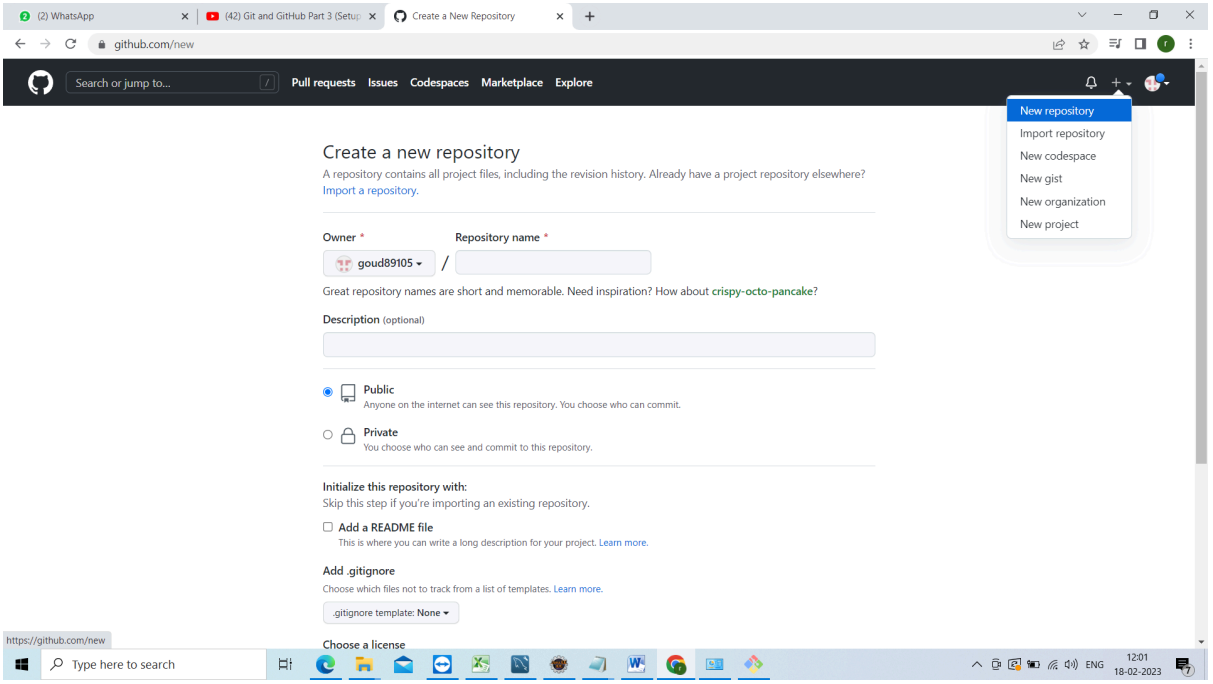
Use git add .

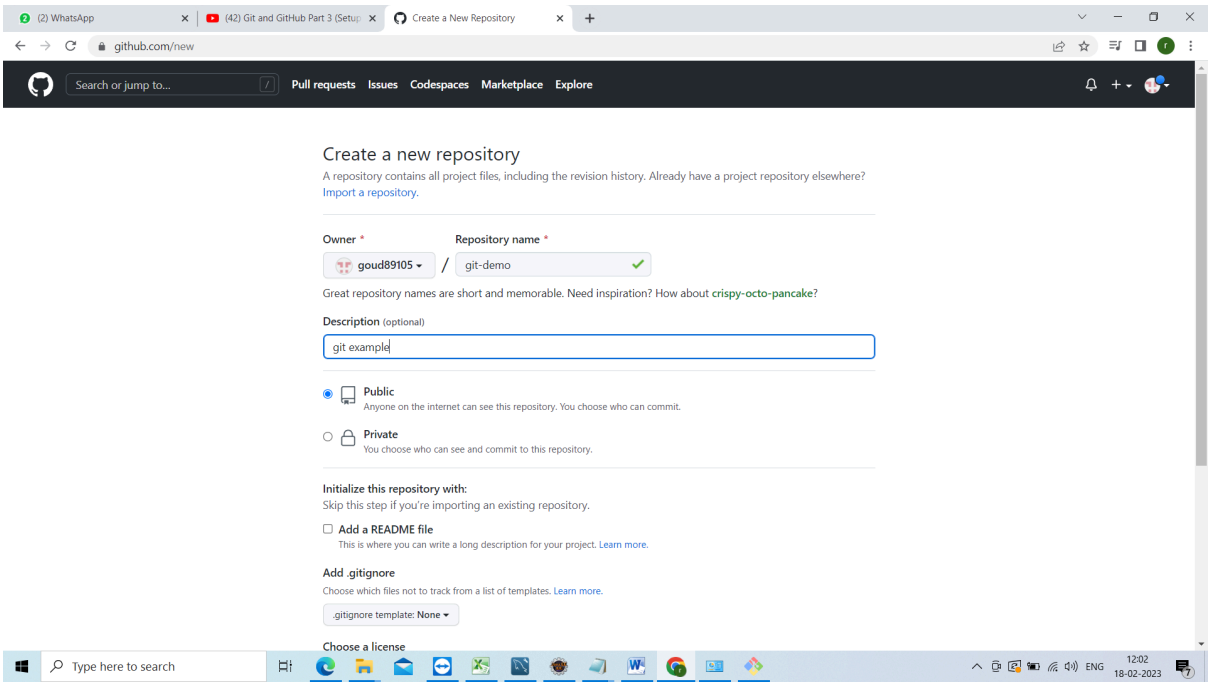
Git commit –m “message”

Next am going show u how to push local repository changes to remote repository.

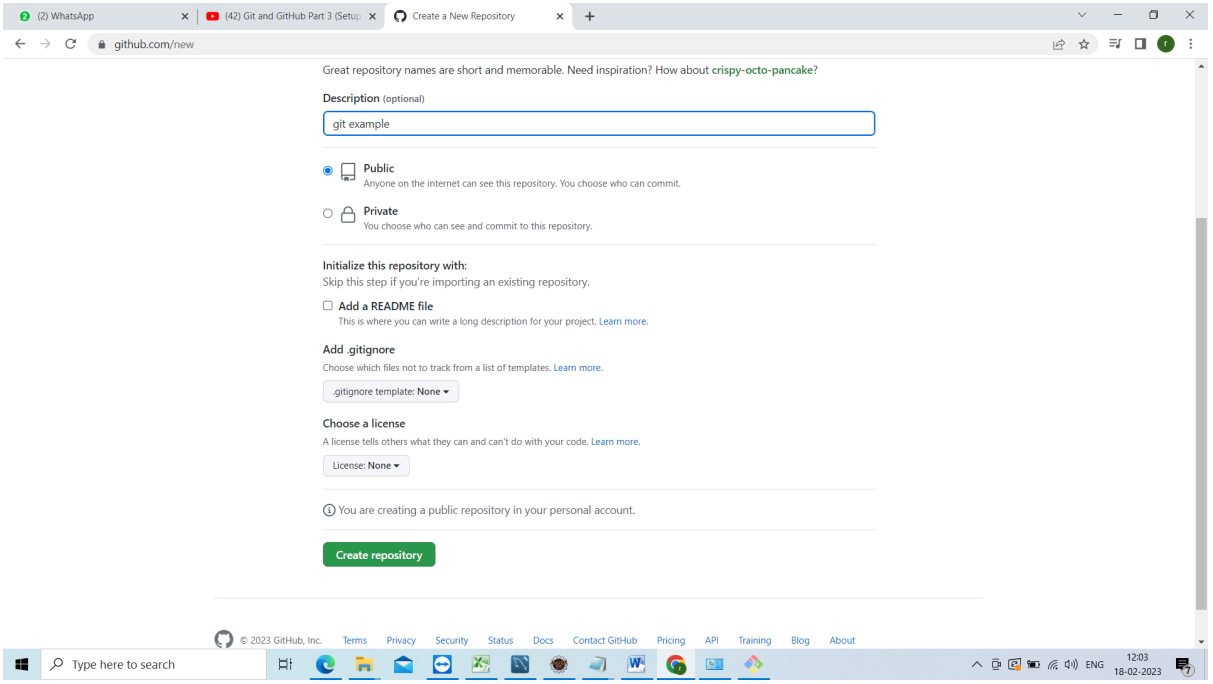
For that u have to create the remote repository in github.

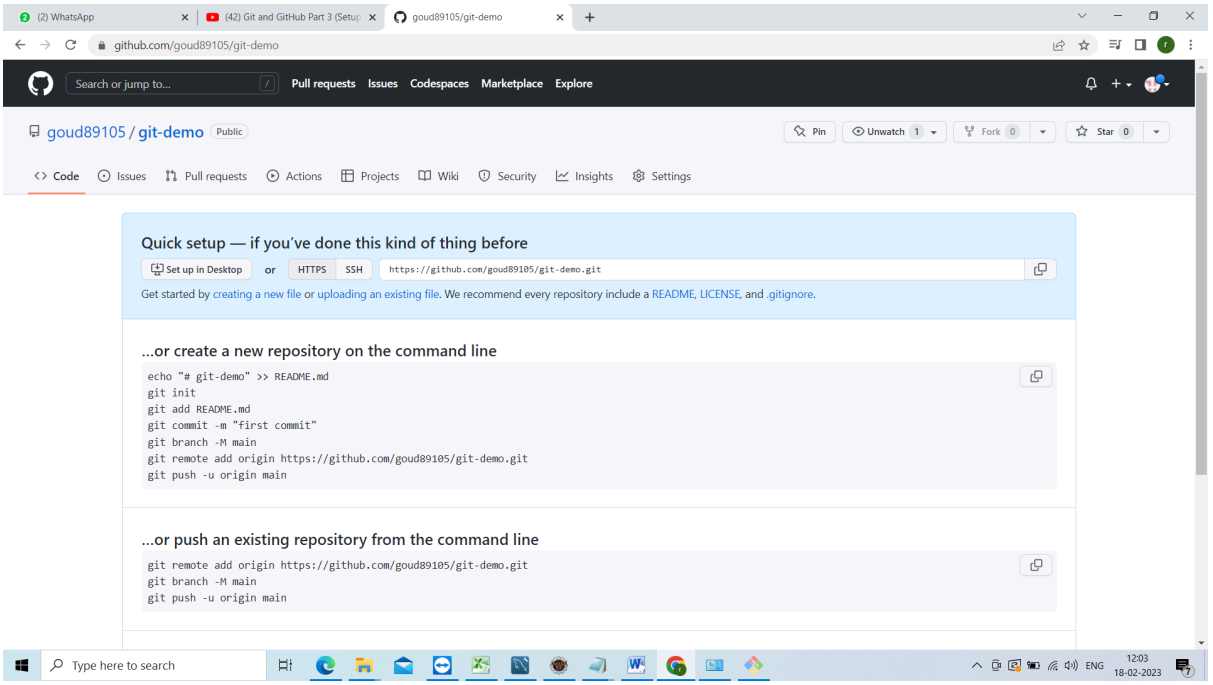
Click on + and select new repository



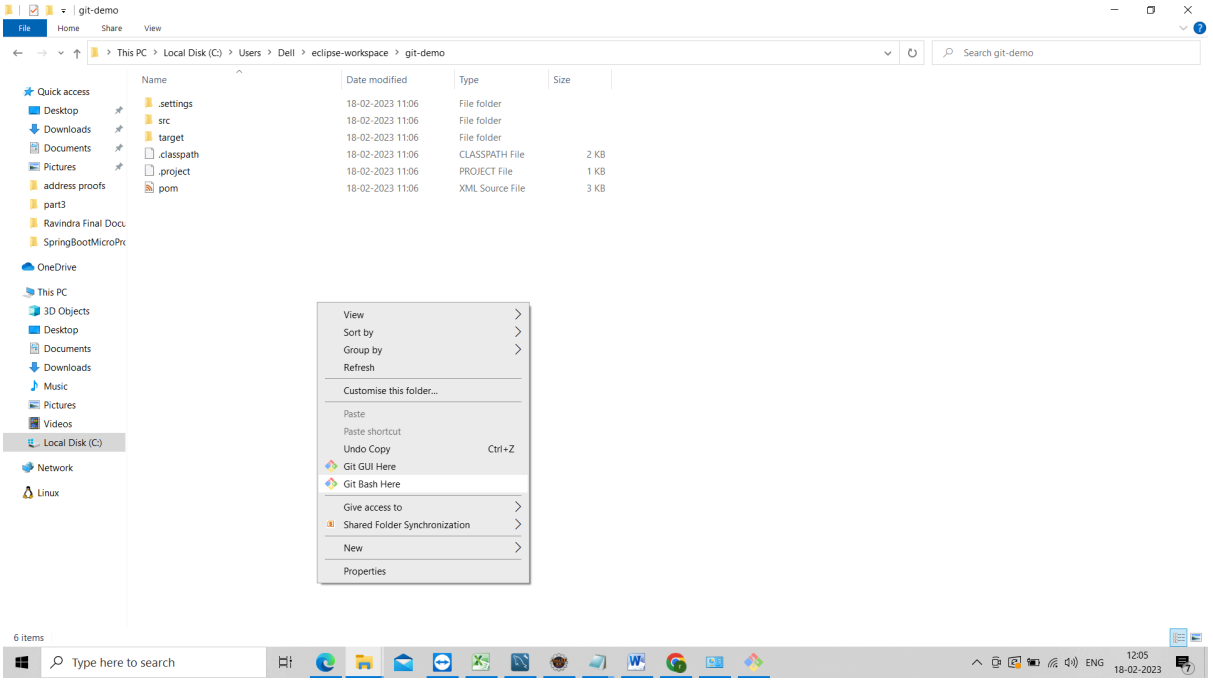


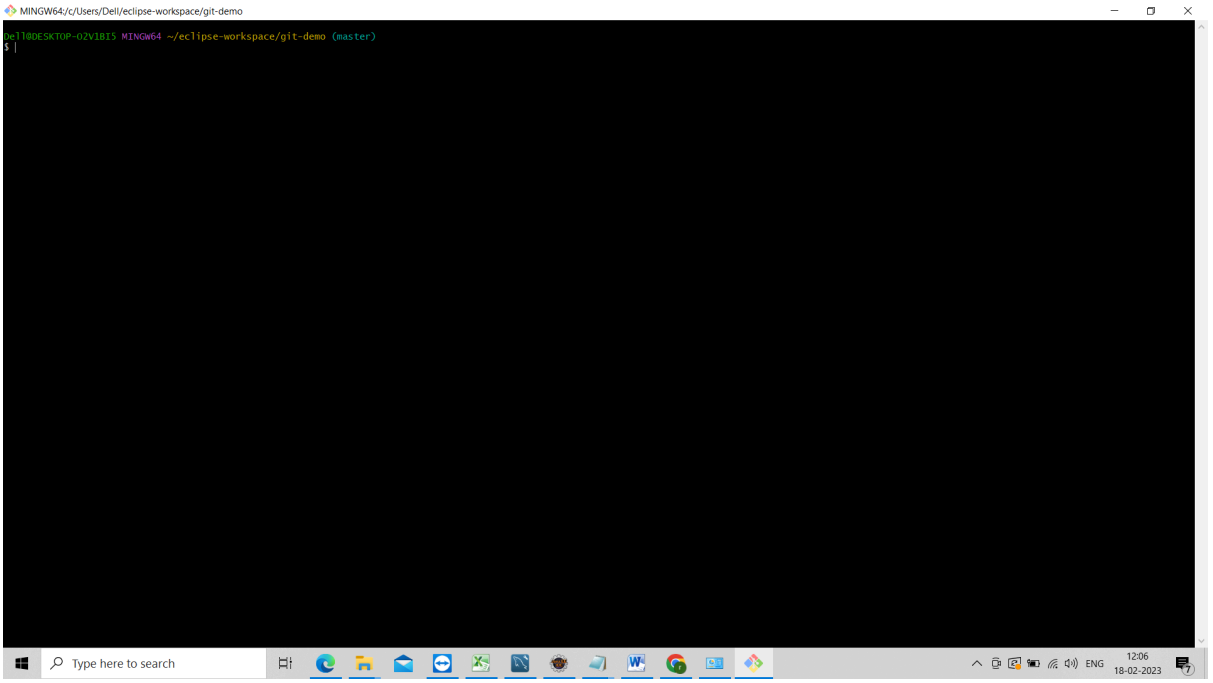
Click on create repository.



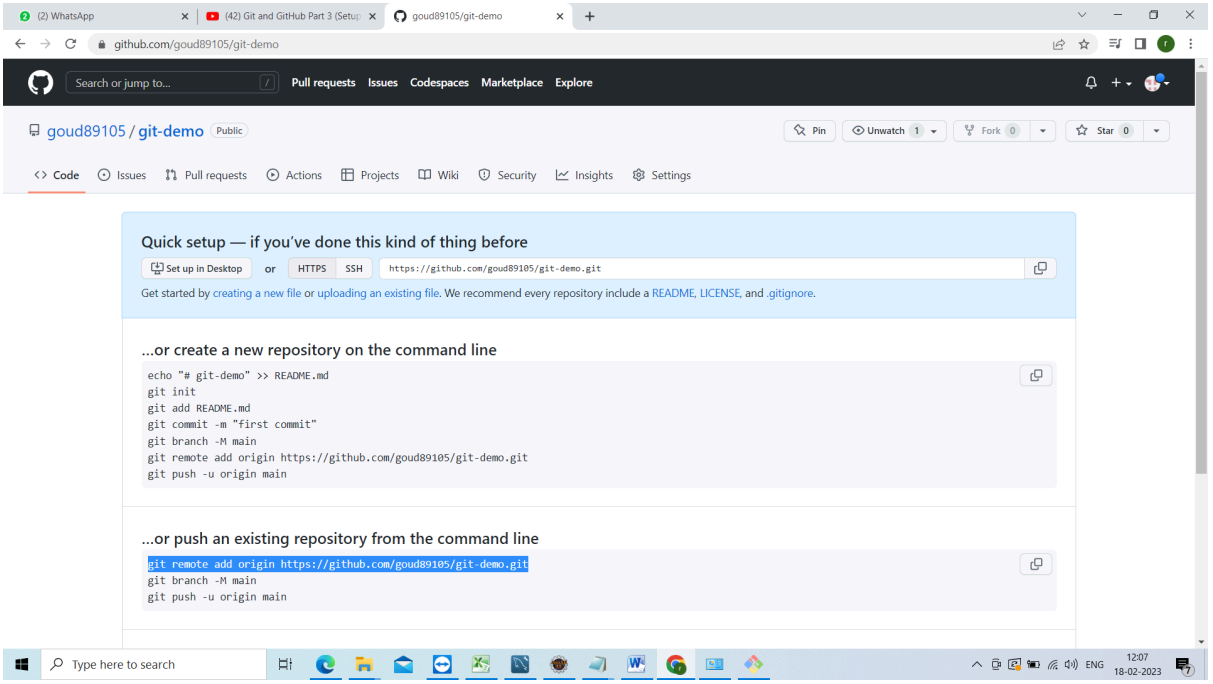


Now go to project directory which u have created previously do git bash from there.

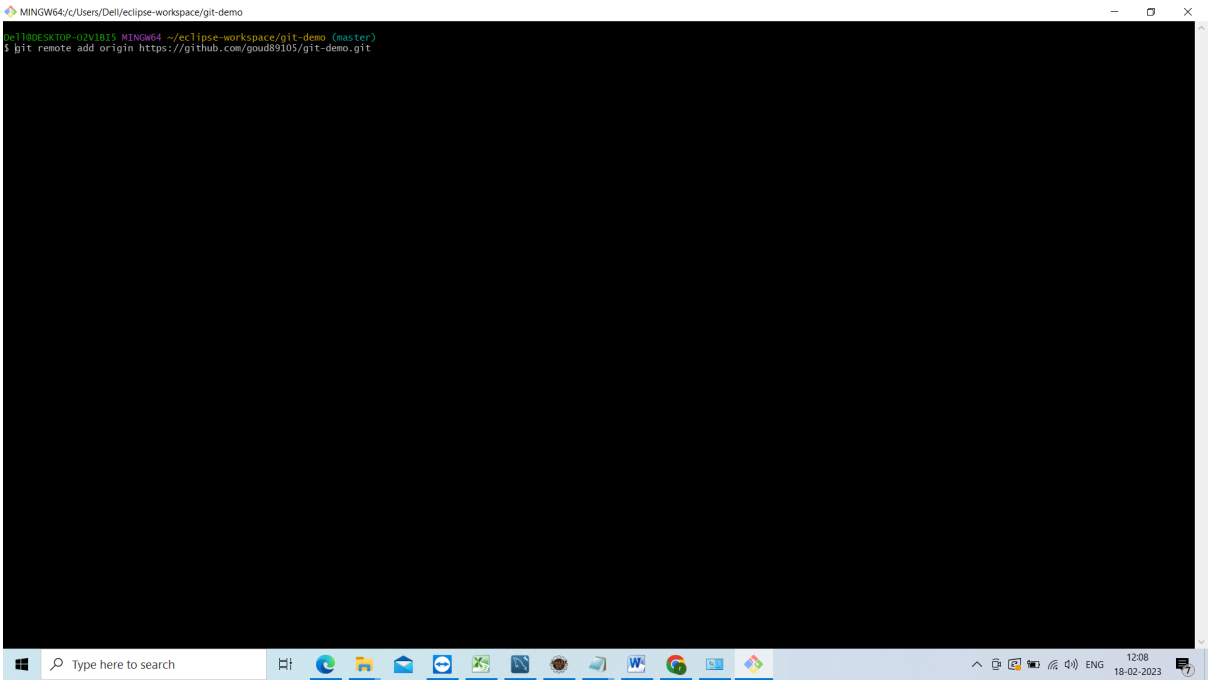




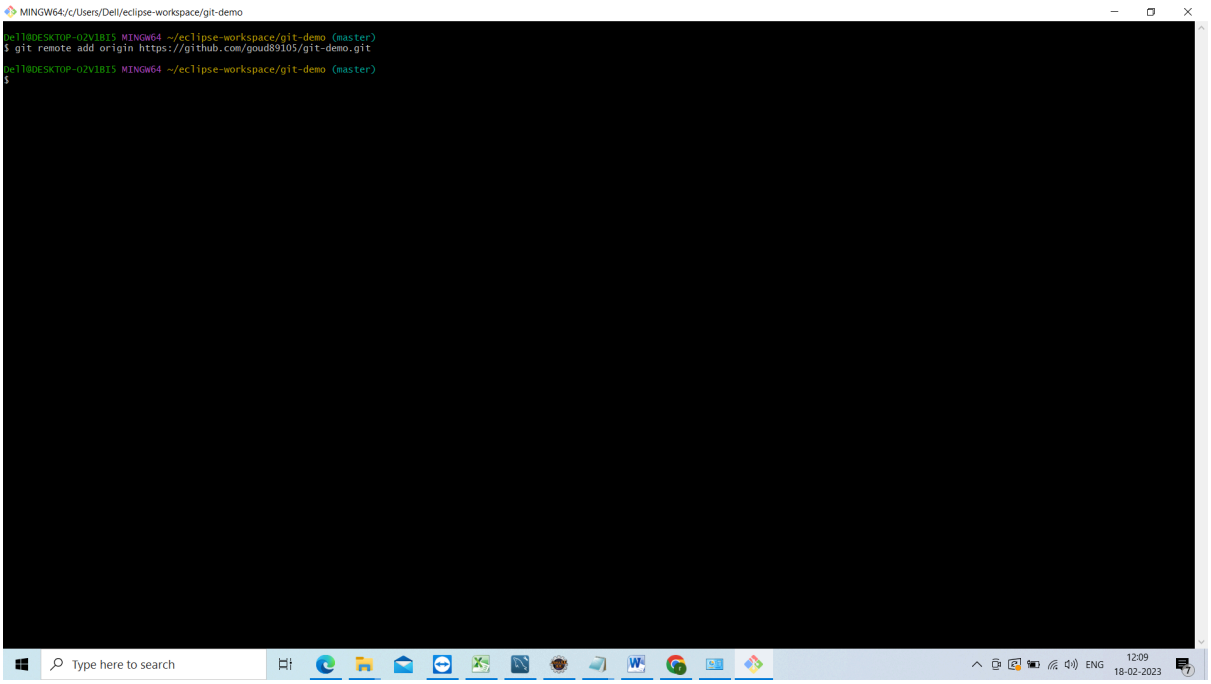
We need to set from local repository to remote repository to set that use below command



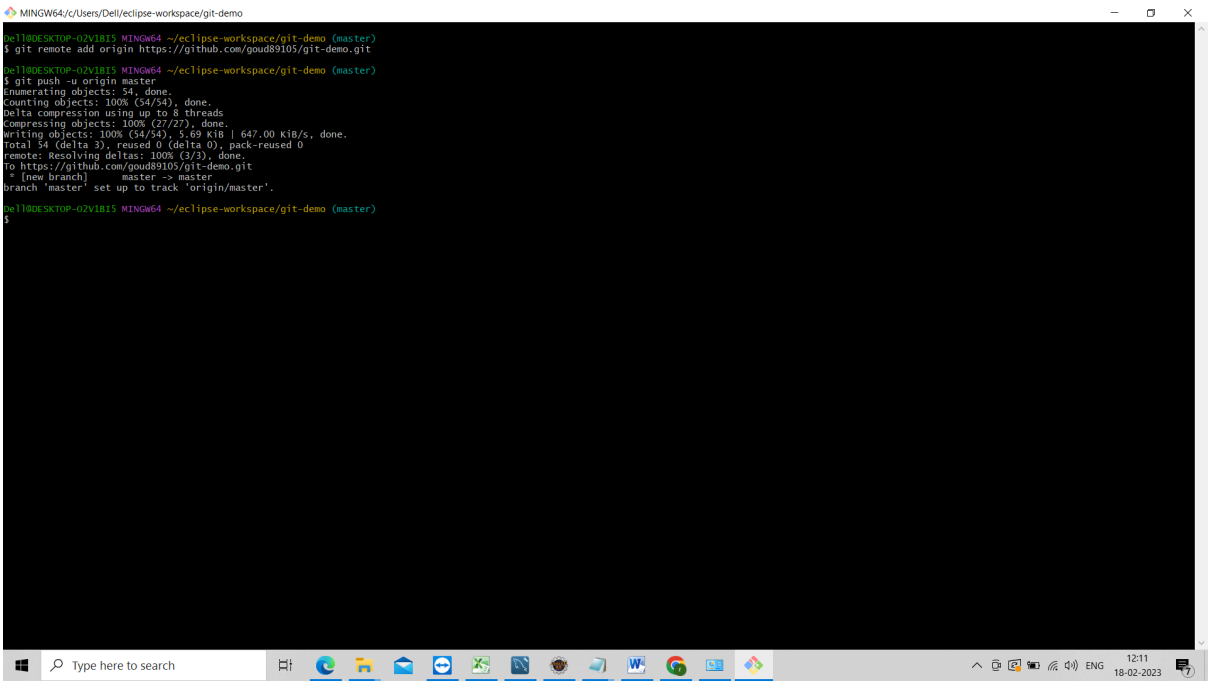
git remote add origin https://github.com/goud89105/git-demo.git



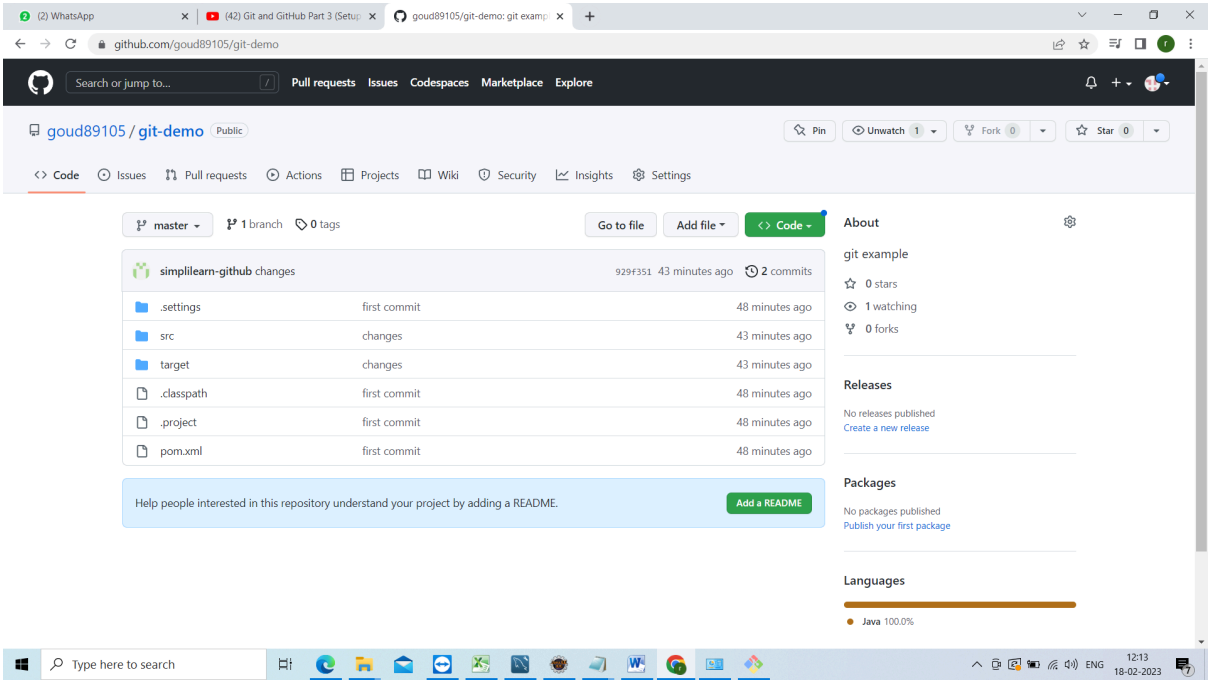
And enter



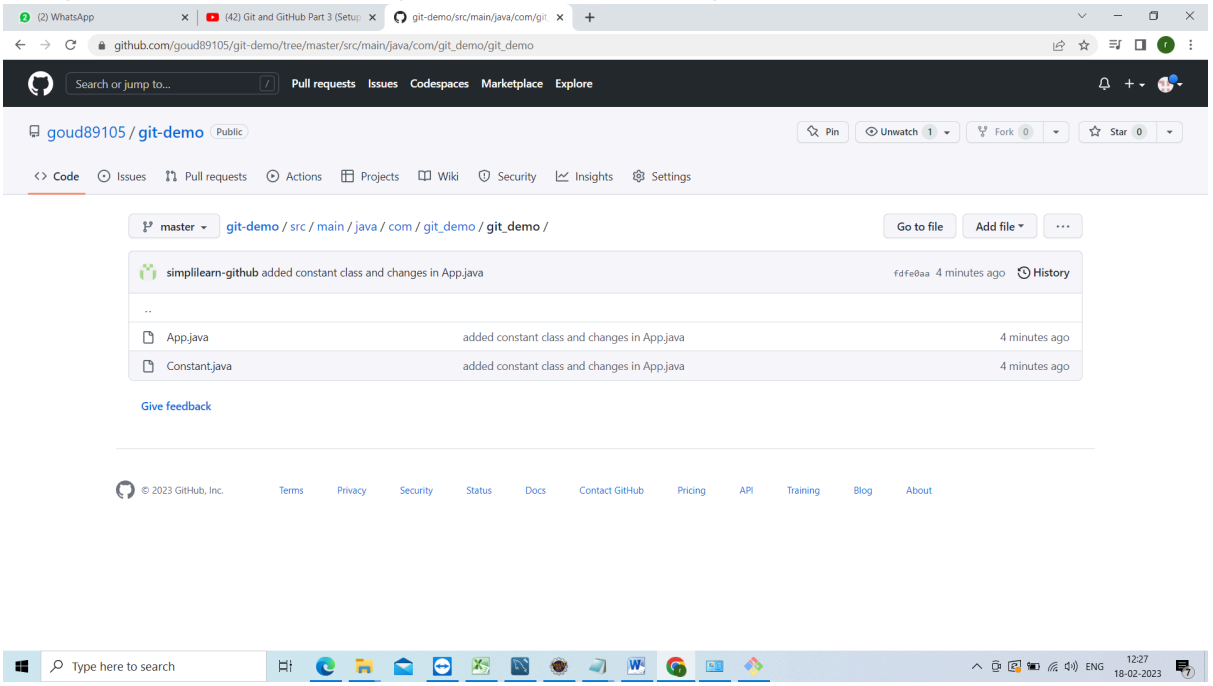
Now push our changes to remote repository
U se below command
Git push –u origin master



Now our local changes push to git hub
Go to github and refresh the page



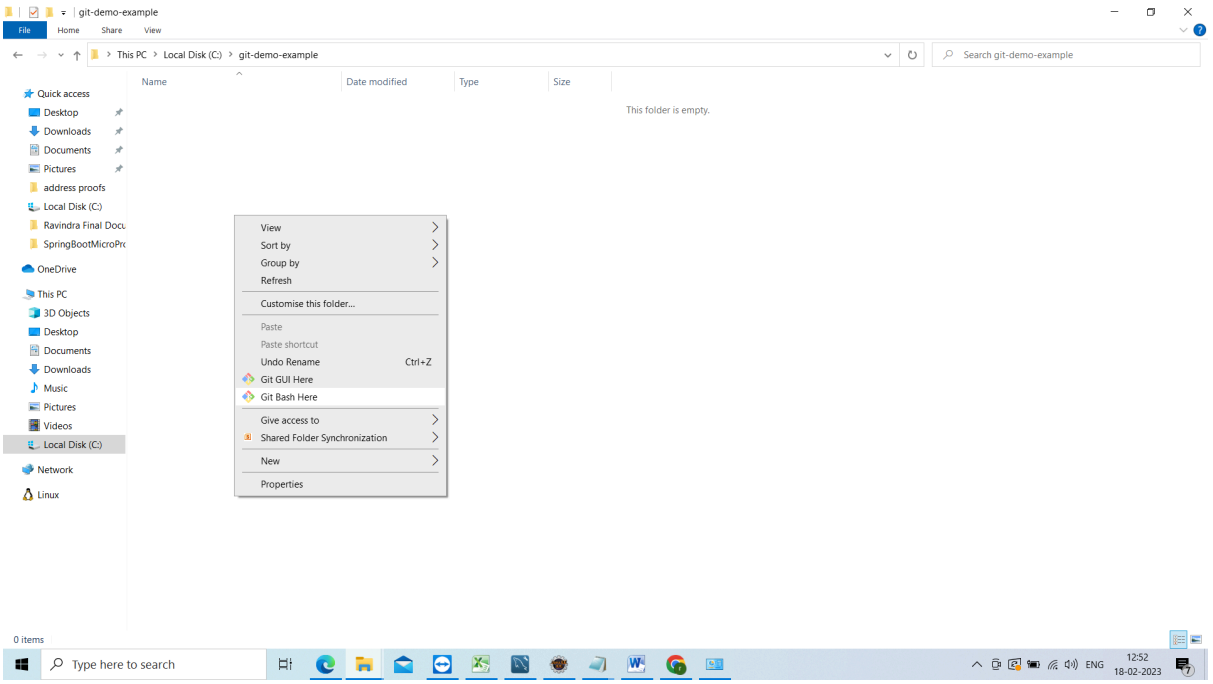
Now go to remote repository and refrest the page

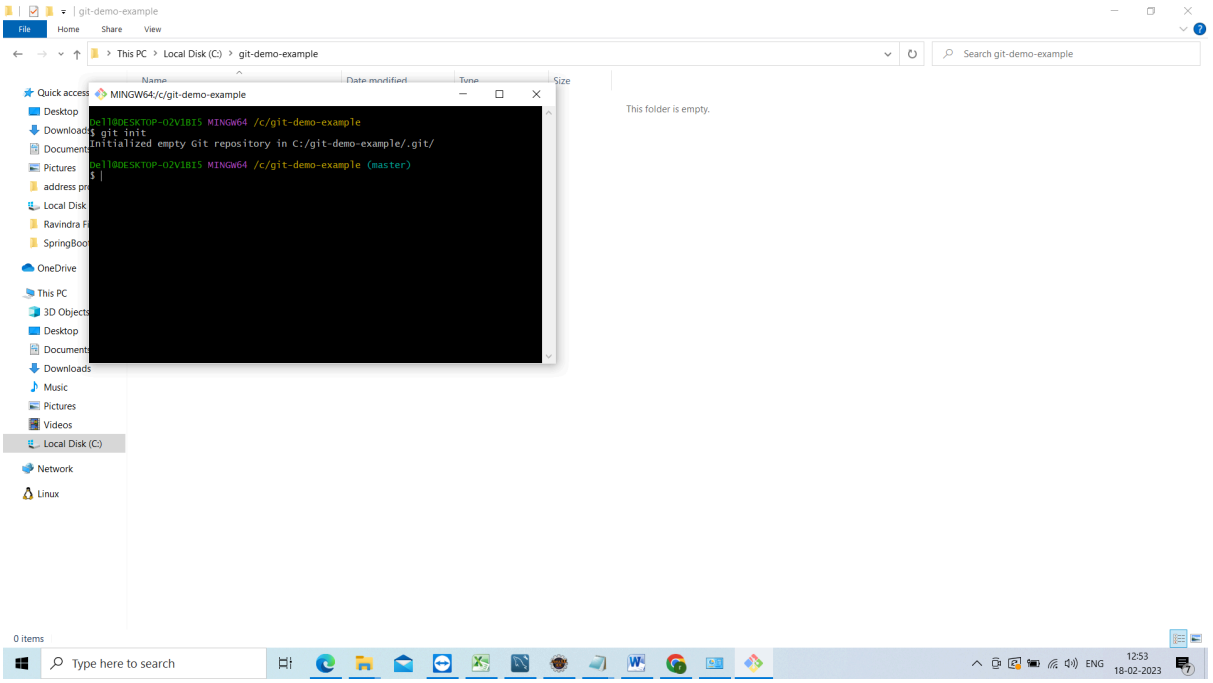


Now am going to show u how to checkout the code and how to update the code

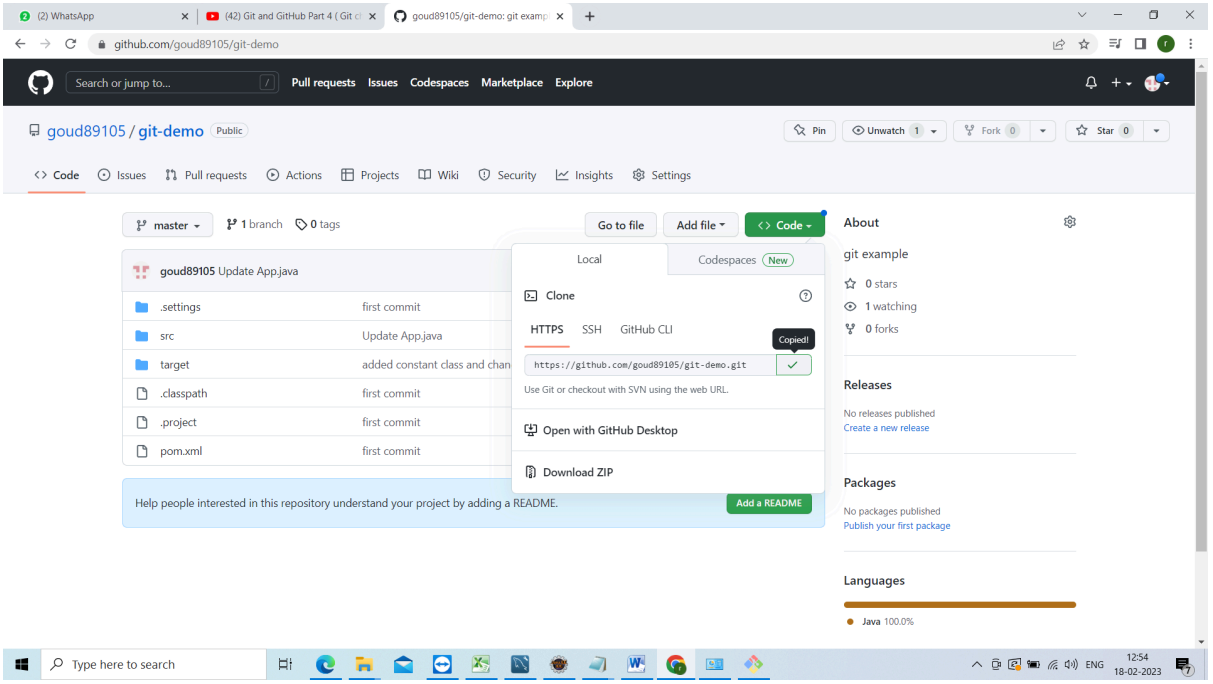
Assume u have joined the project newly they have given git link I will show how to checkout the code from github

Frist go to the any drive and create the folder and open the git bash from that folder and do git init



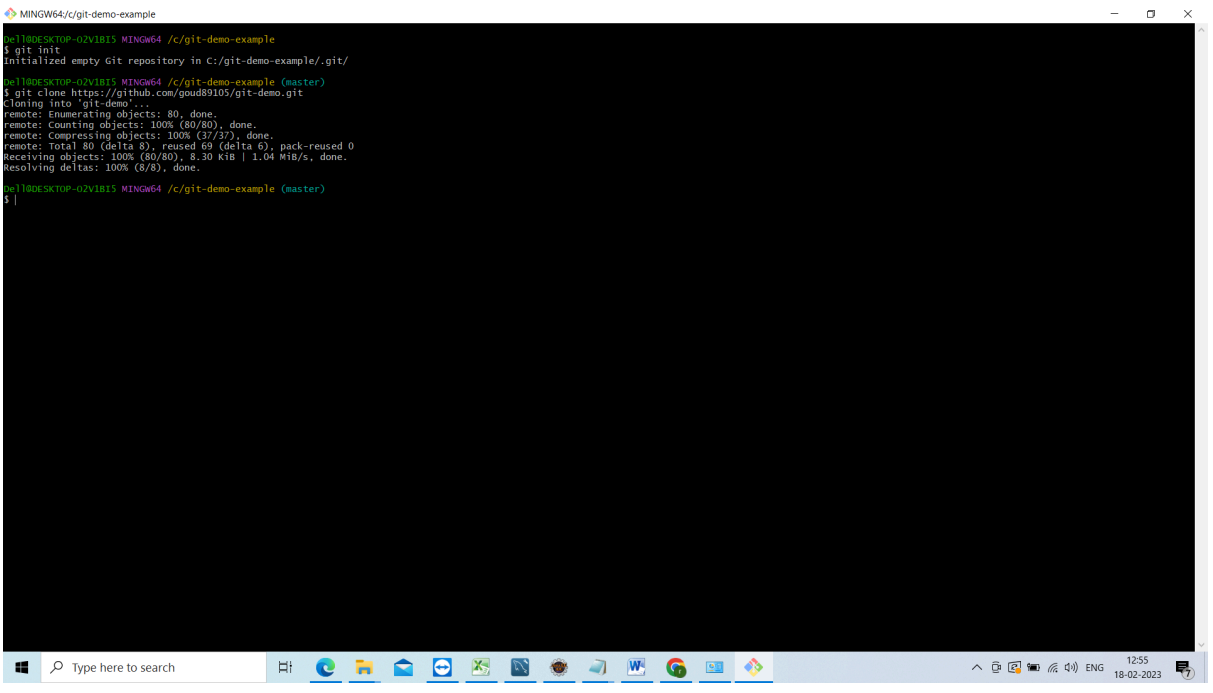


And now go to git hub and copy the link from below https



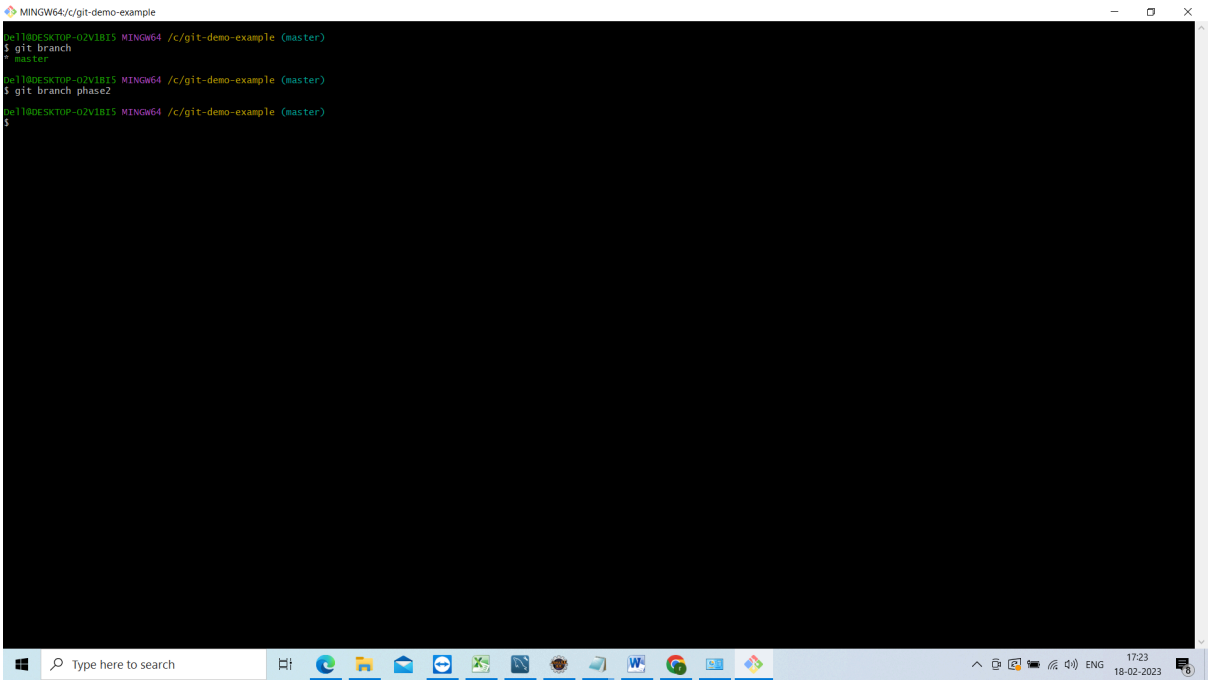
Now go to git bash execute the below command
Git clone URL

https://github.com/goud89105/git-demo.git



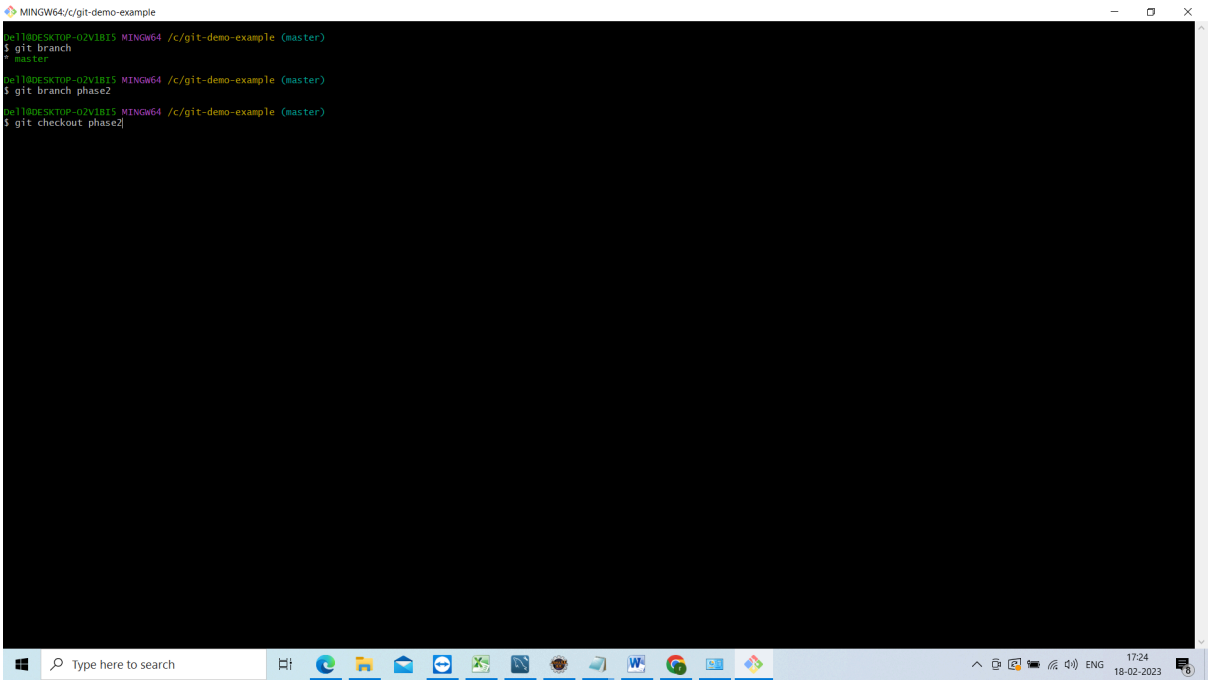
Create the new branch

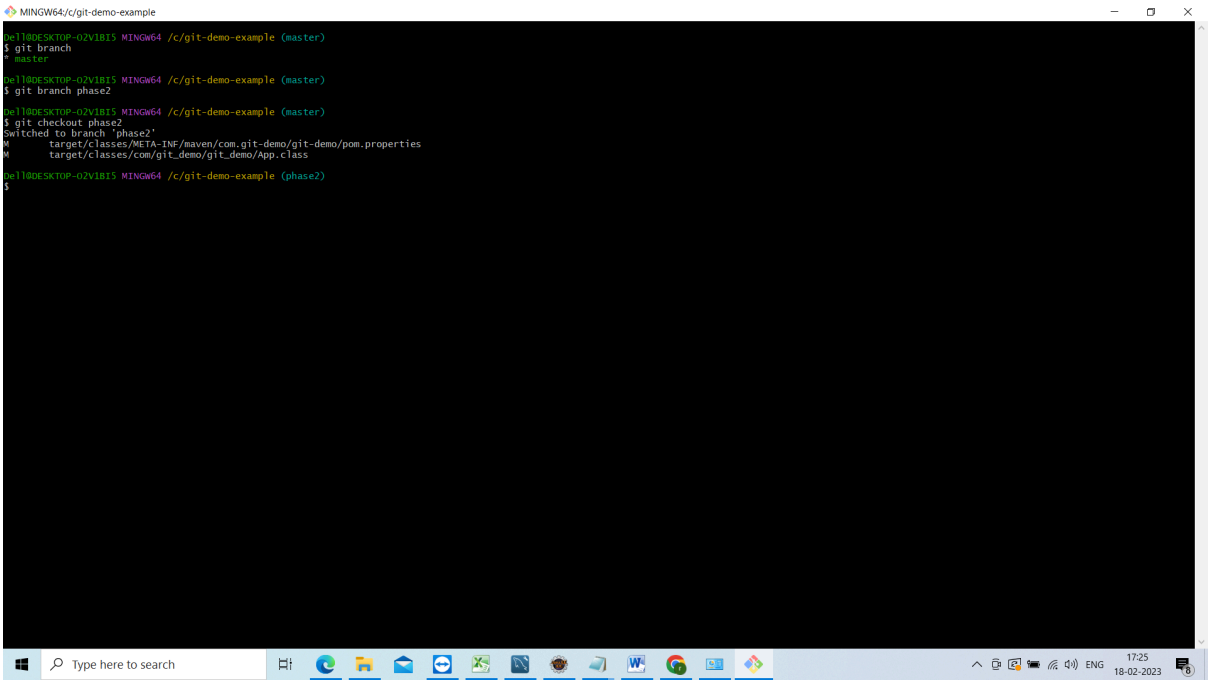
Git branch name of the branch



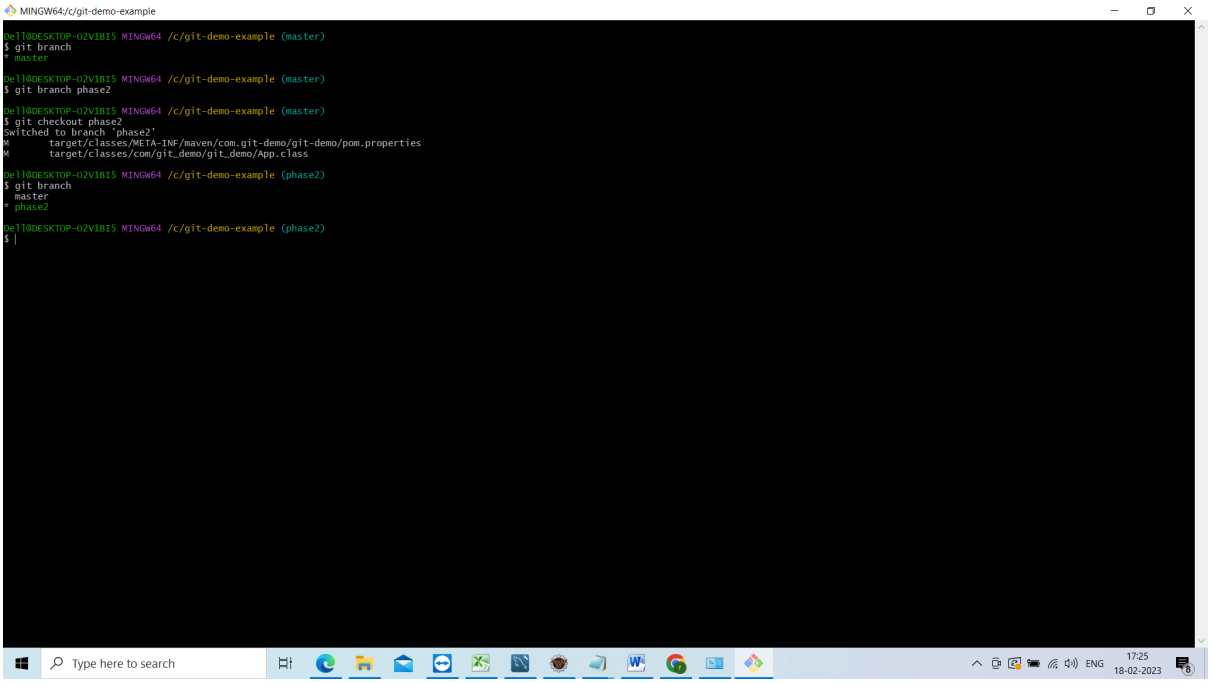
Now lets move to the

Git checkout phase2

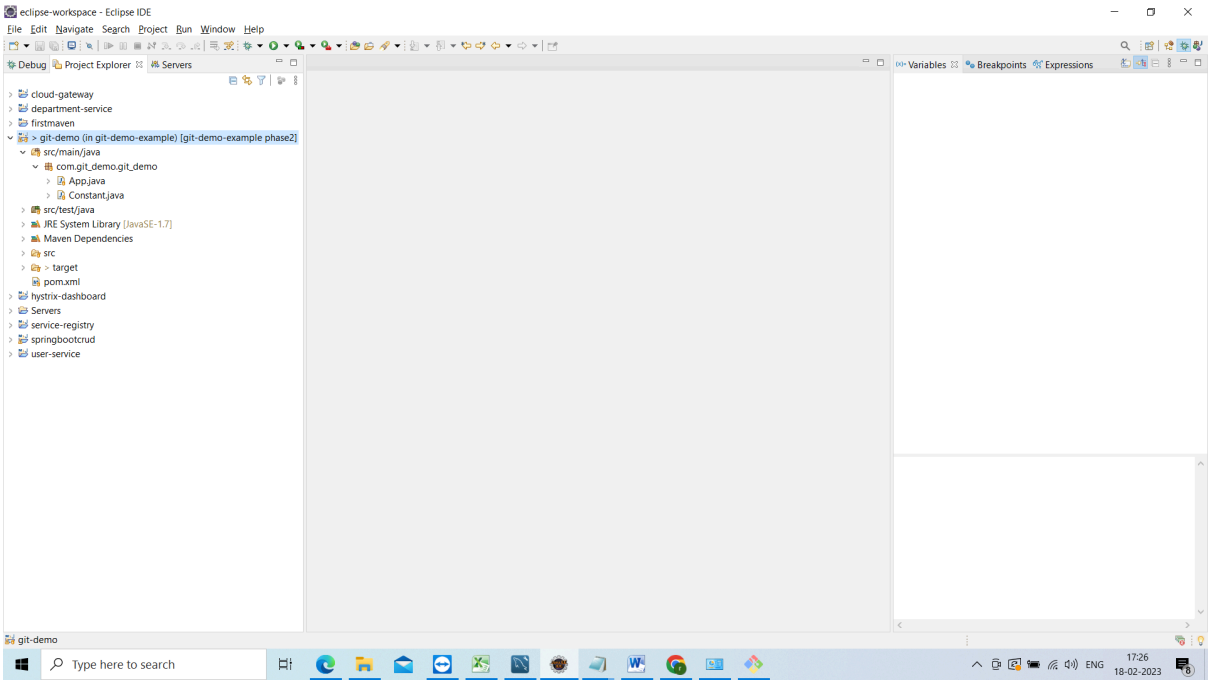




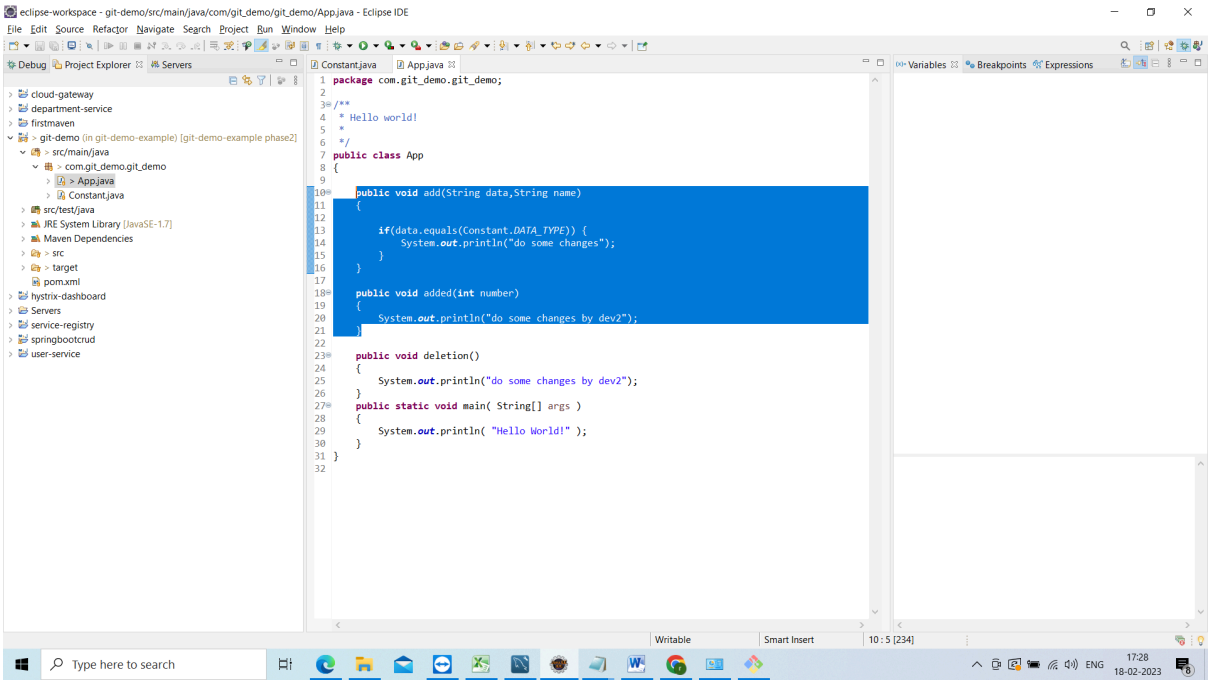
Now check the branch



Now go to eclipse see now we are in phase2 branch

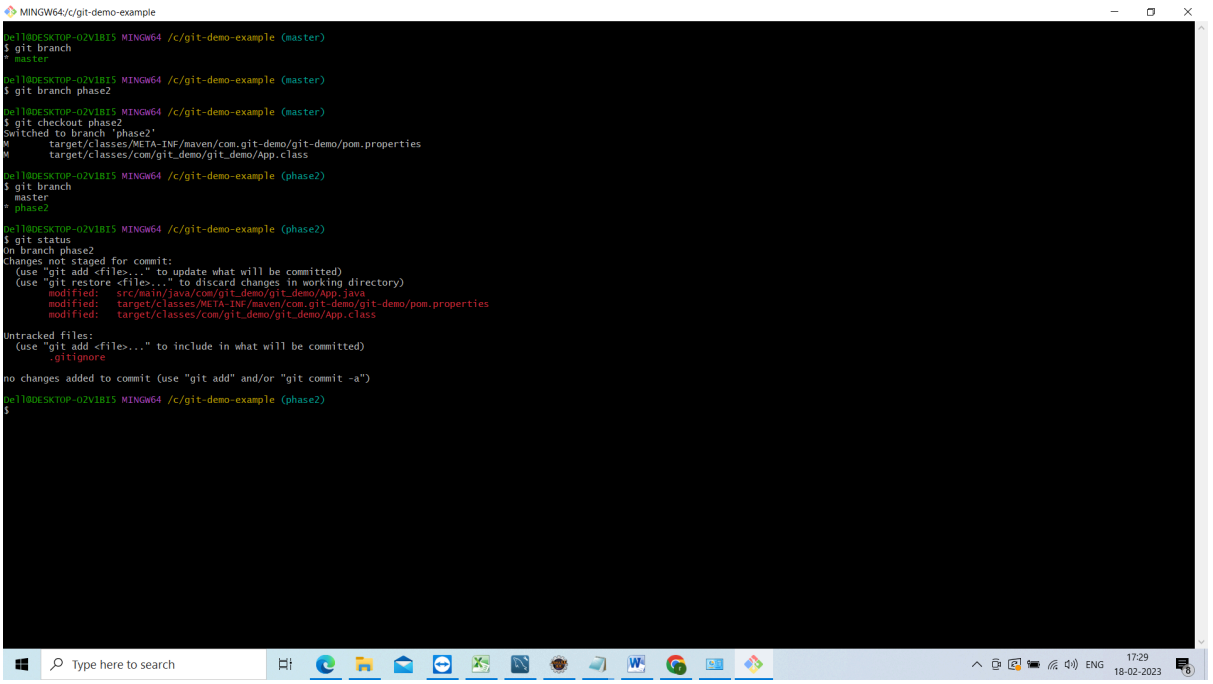


Now do some changes those changes will be in phase2 branch



Now these changes not available in master branch

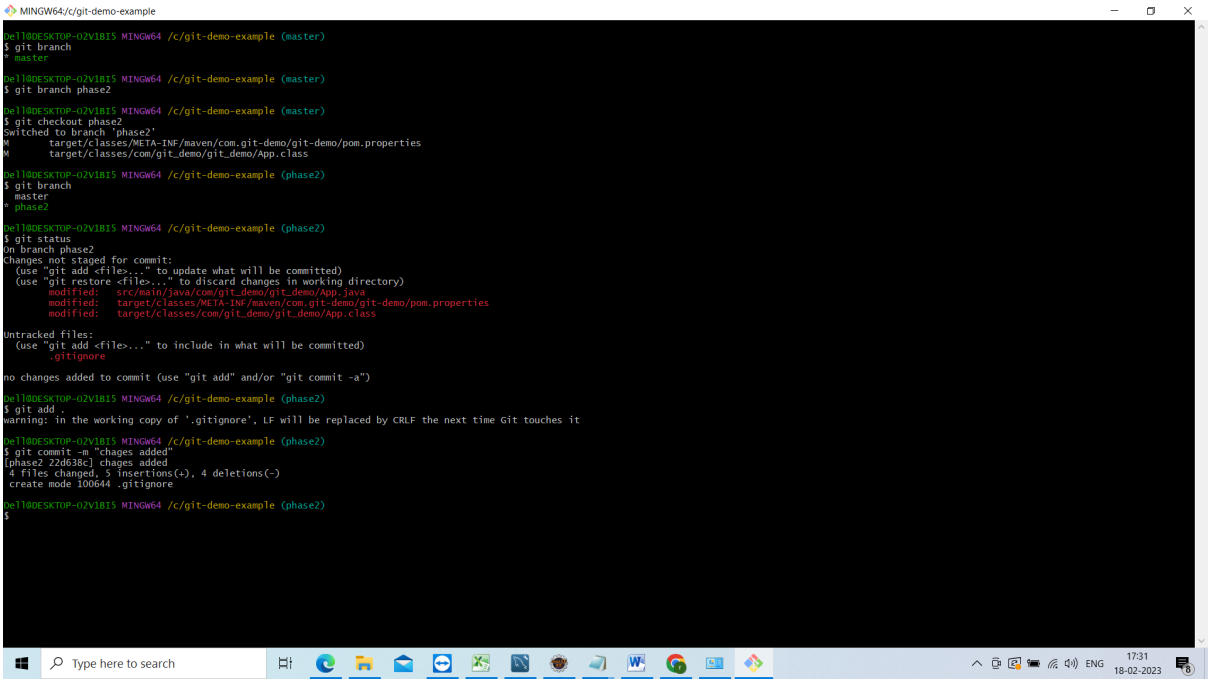
Go to git bash and see the changes in phase2



Now add it and commit

Git add .

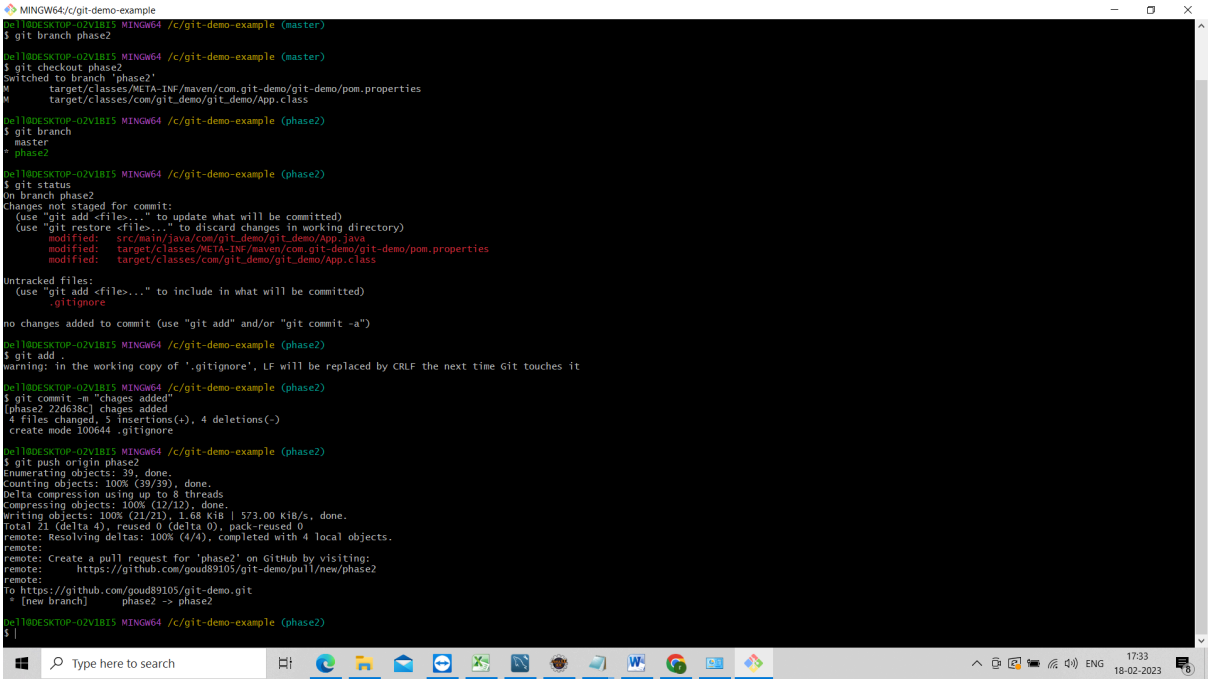
Git commit –m “changes added”



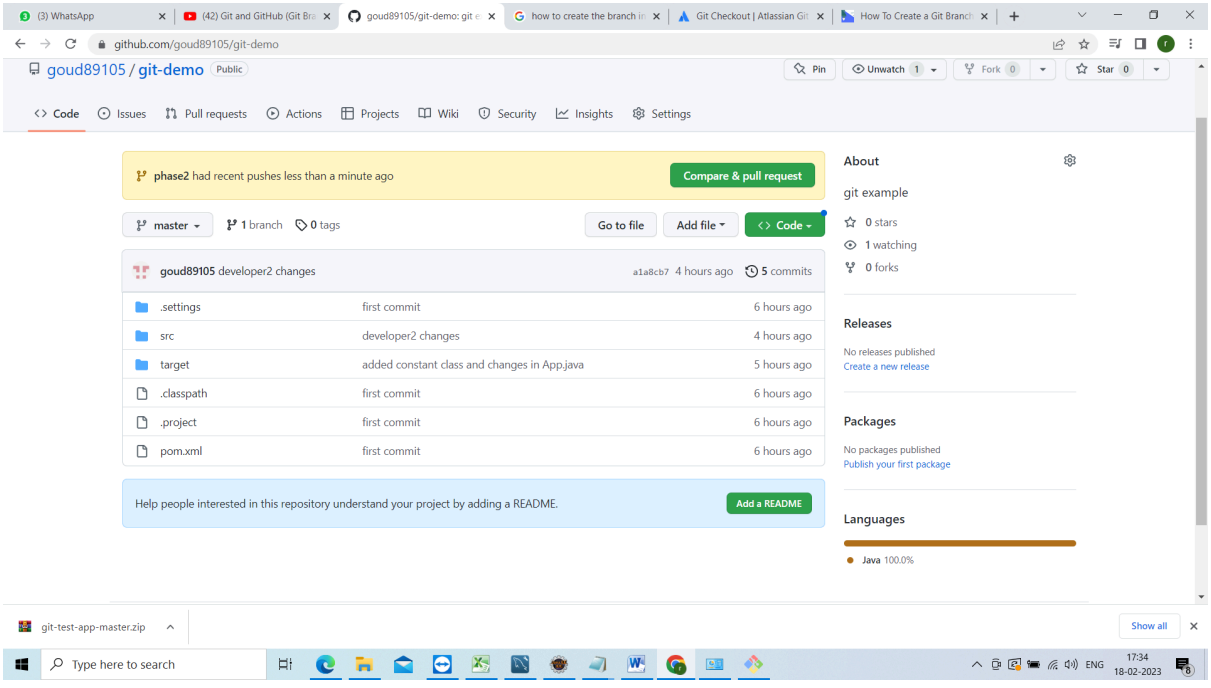
Now commited code in staging not in master and not in phase2

Now commit these changes to phase2

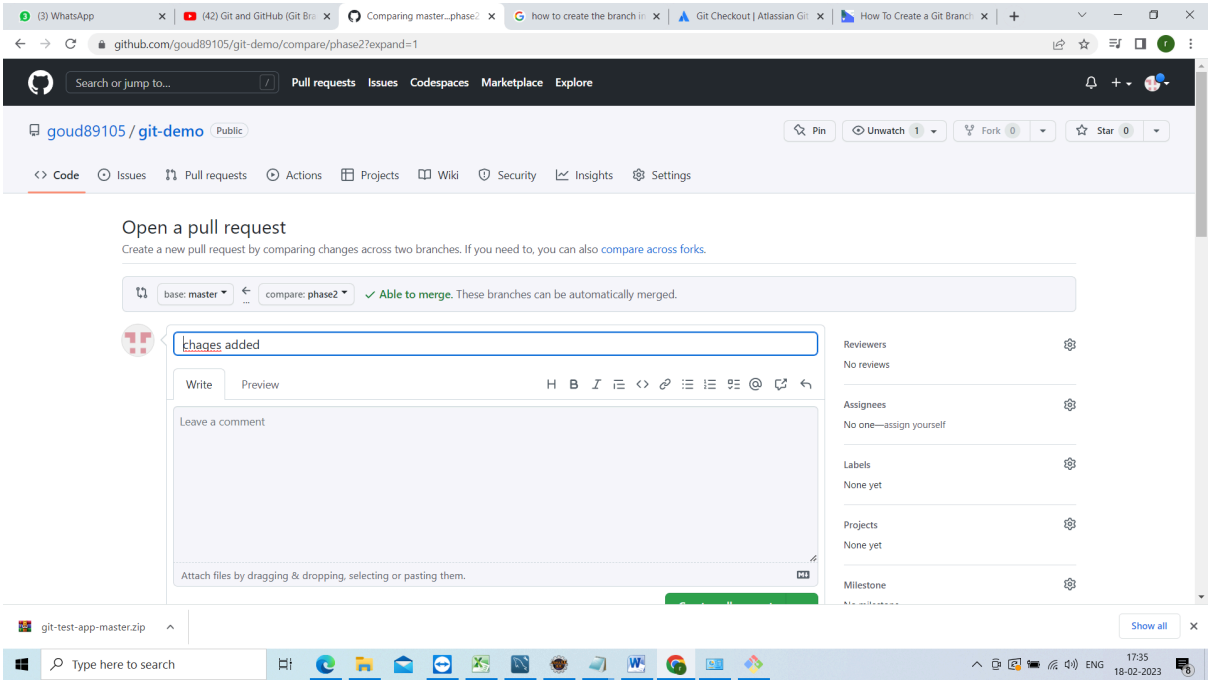
Git push origin phase2



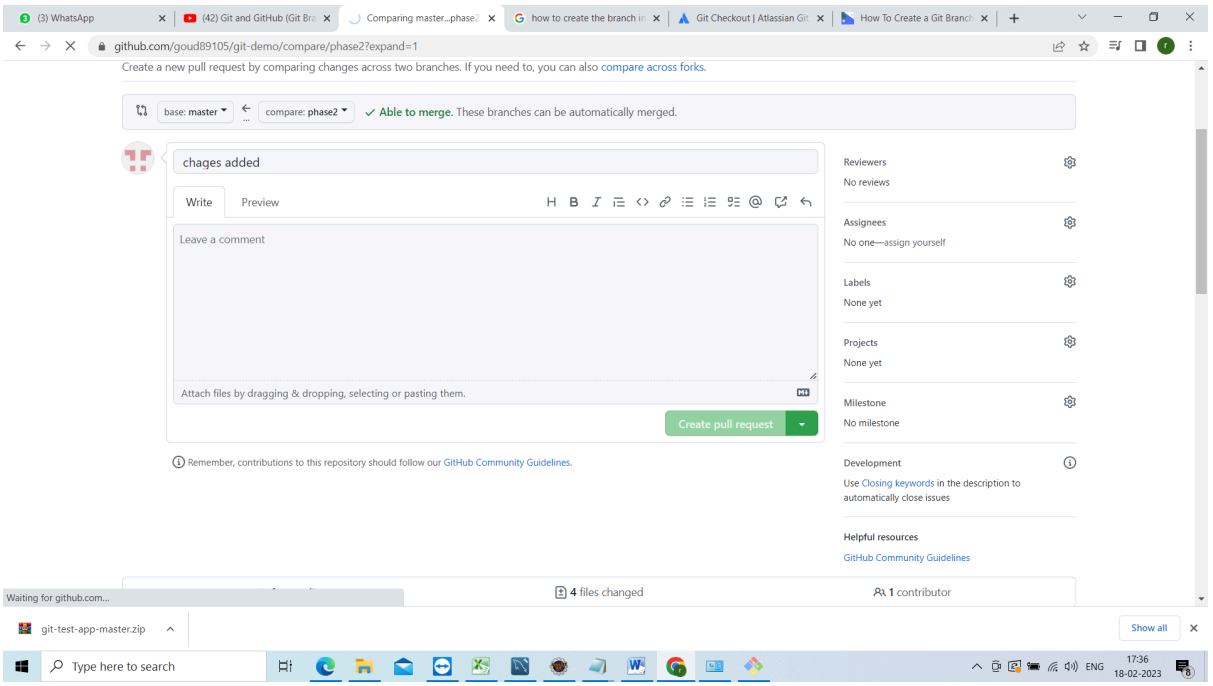
Now in git hub it will show the new branch phase2 go to git hub and see

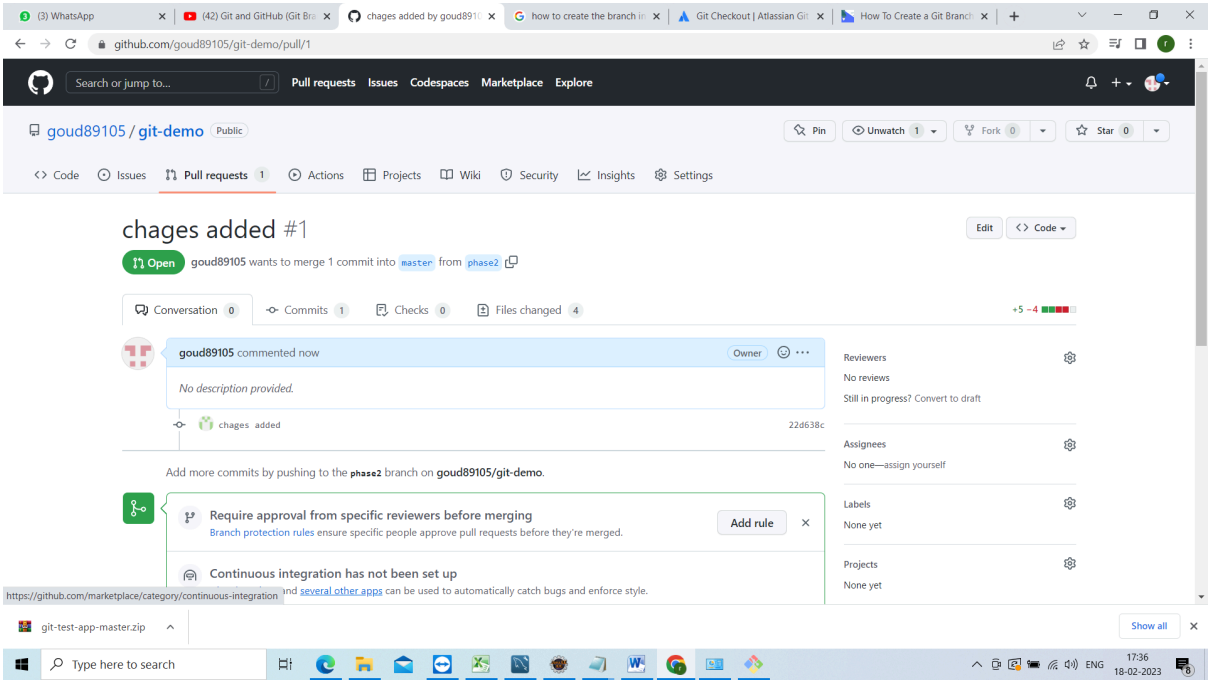


Now compare and create the pull request

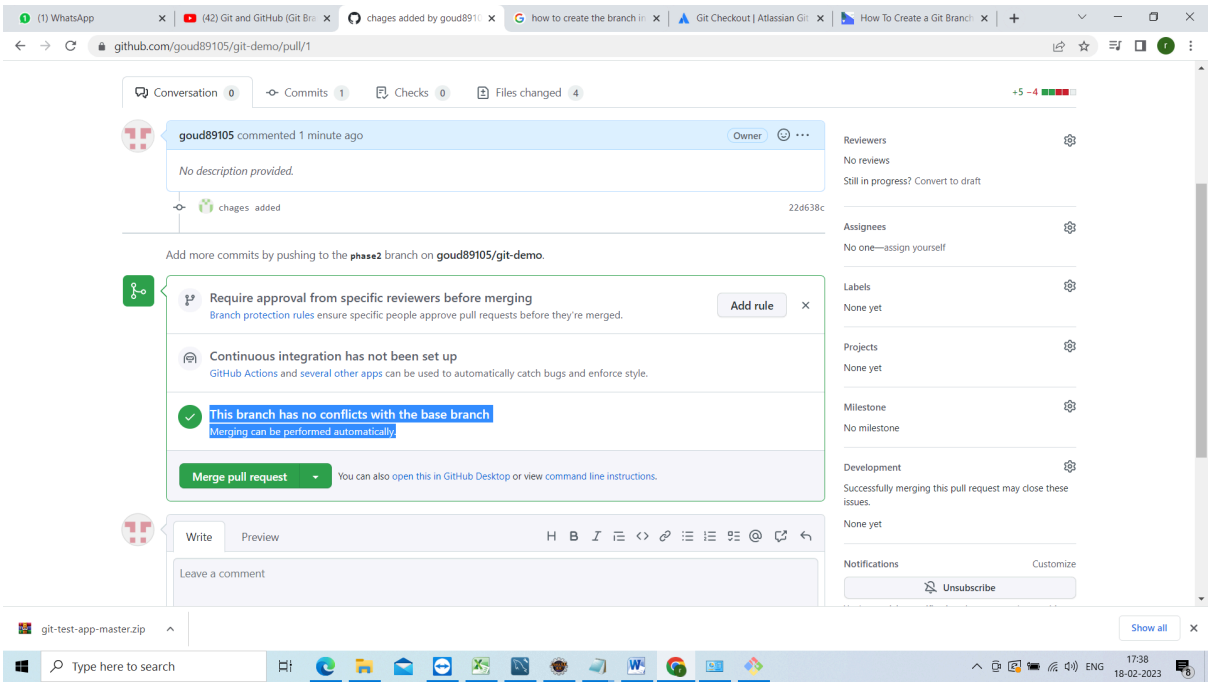


Now click on pull request

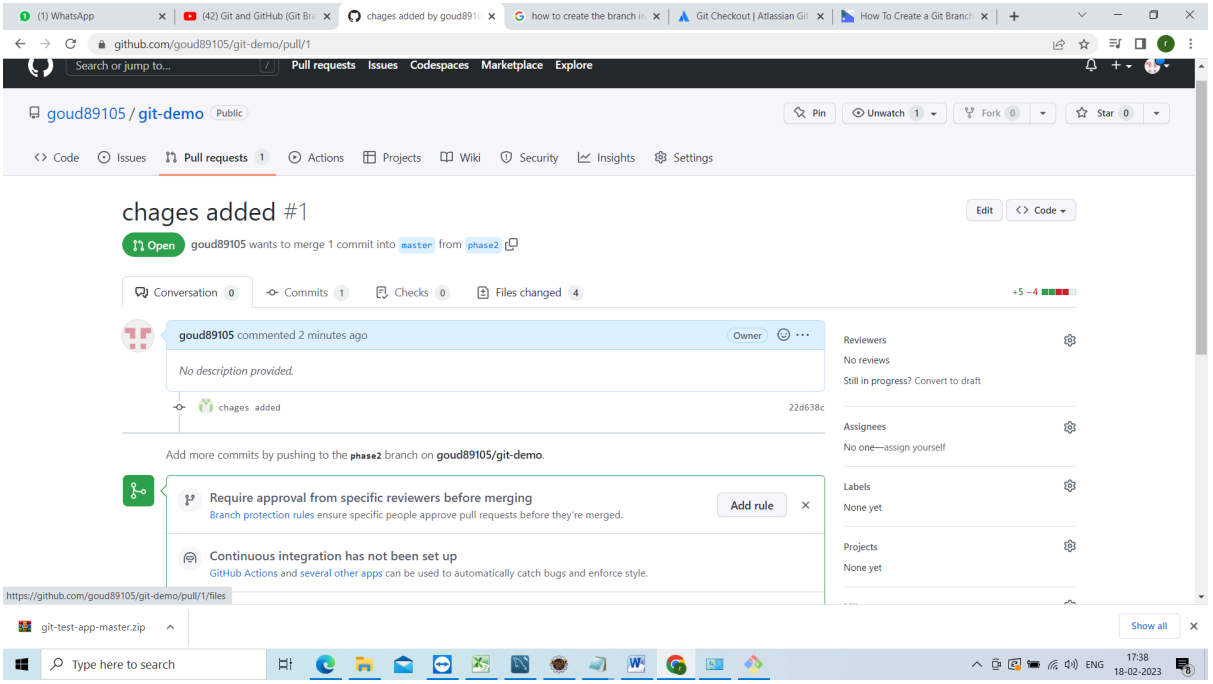


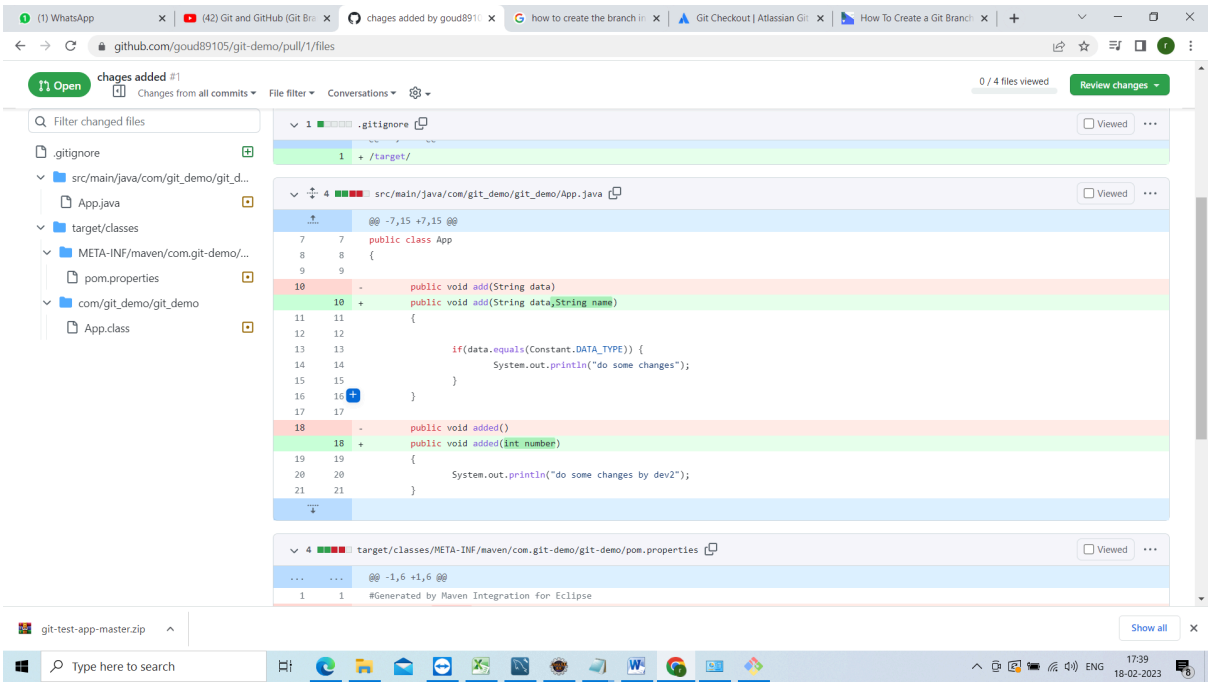


Now see this branch has no conflicts with the base branch



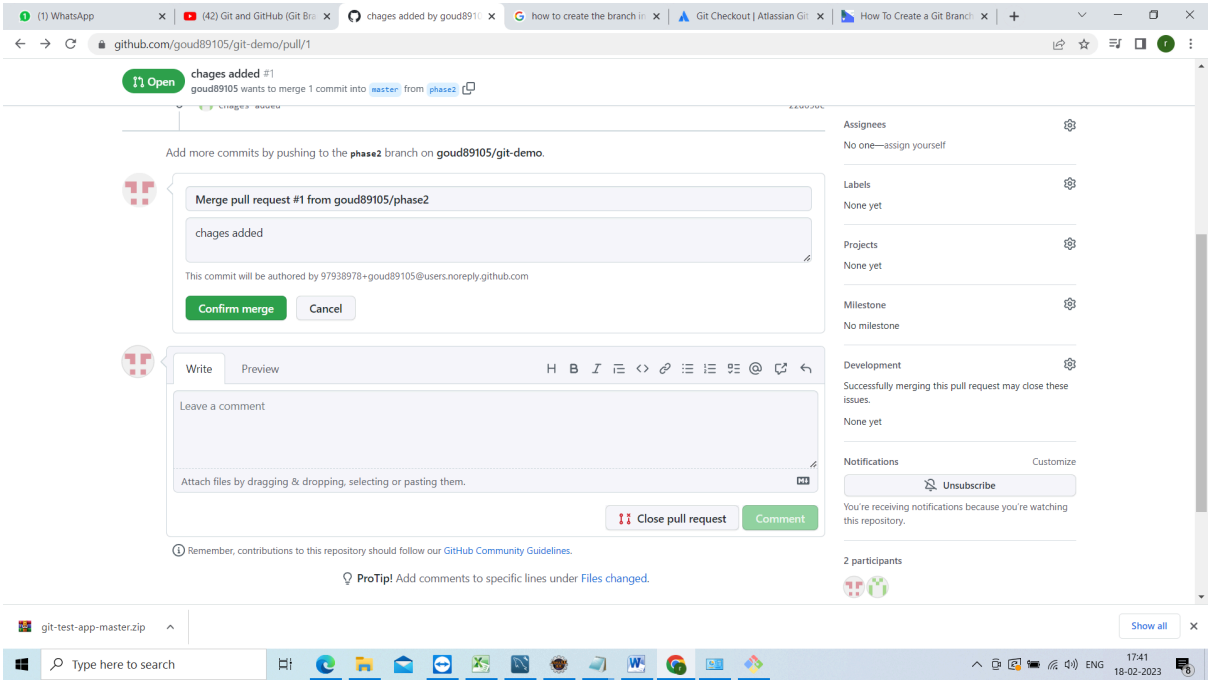
Before that lets check the changes click on files chaged



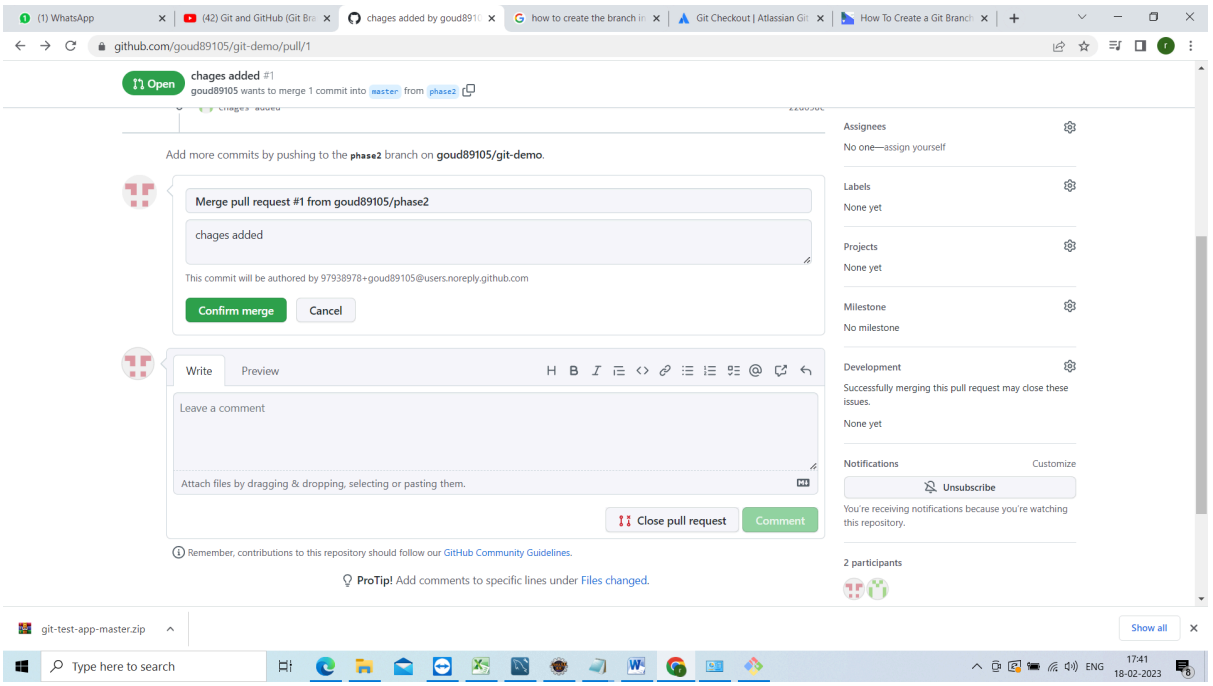


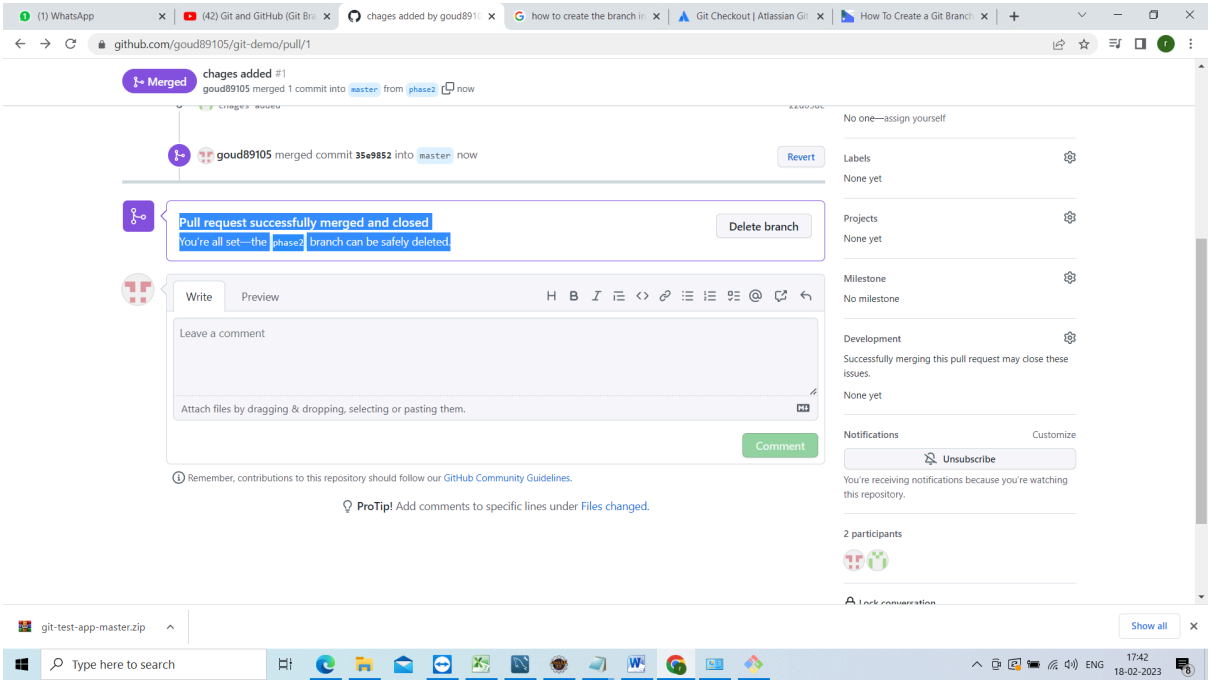
So that u can compare and can the review there

Now go back and click on merge pull request



Click on confirm merge

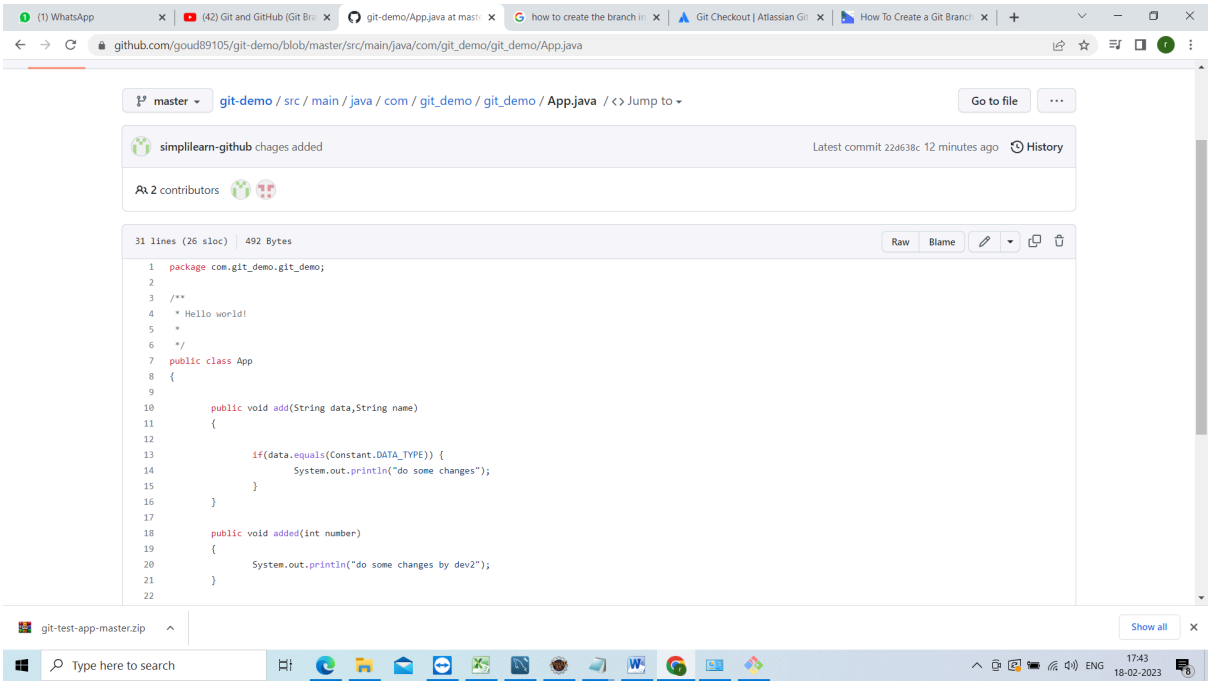




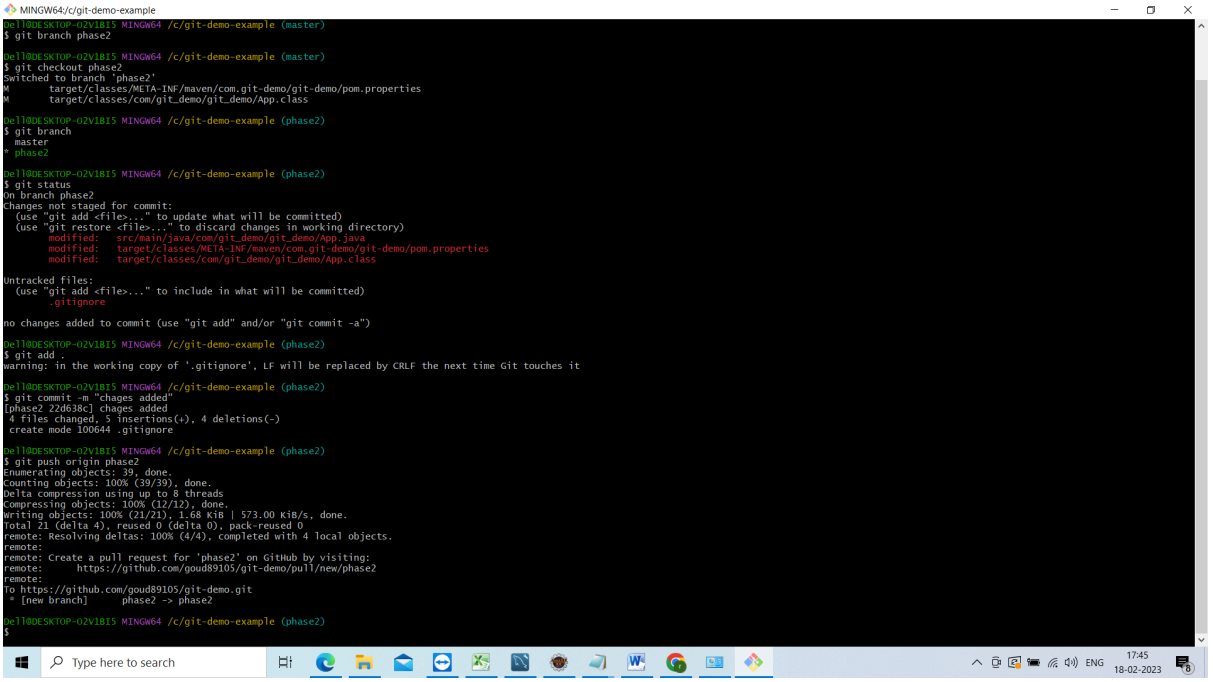
Now u have to give this pull request URL to ur team lead

https://github.com/goud89105/git-demo/pull/1

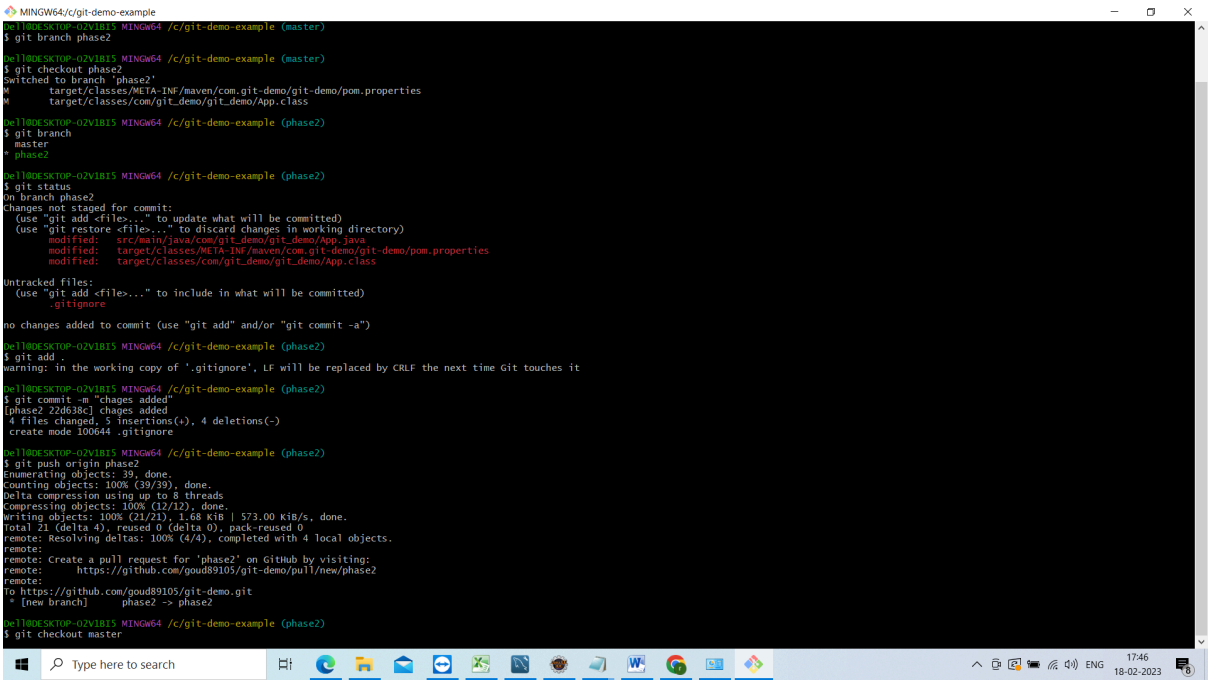
no go back and see this chages refleted or not

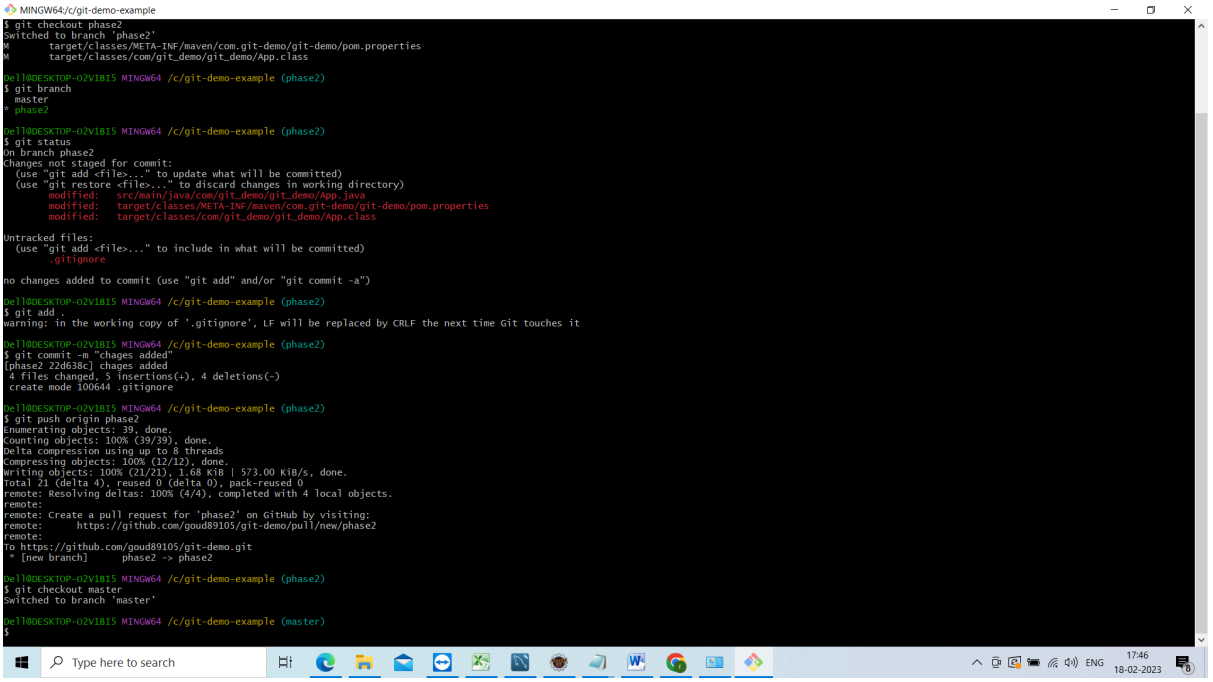


Now go to git bash changes are there in phase2 not in master changes wont come to master until u take pull

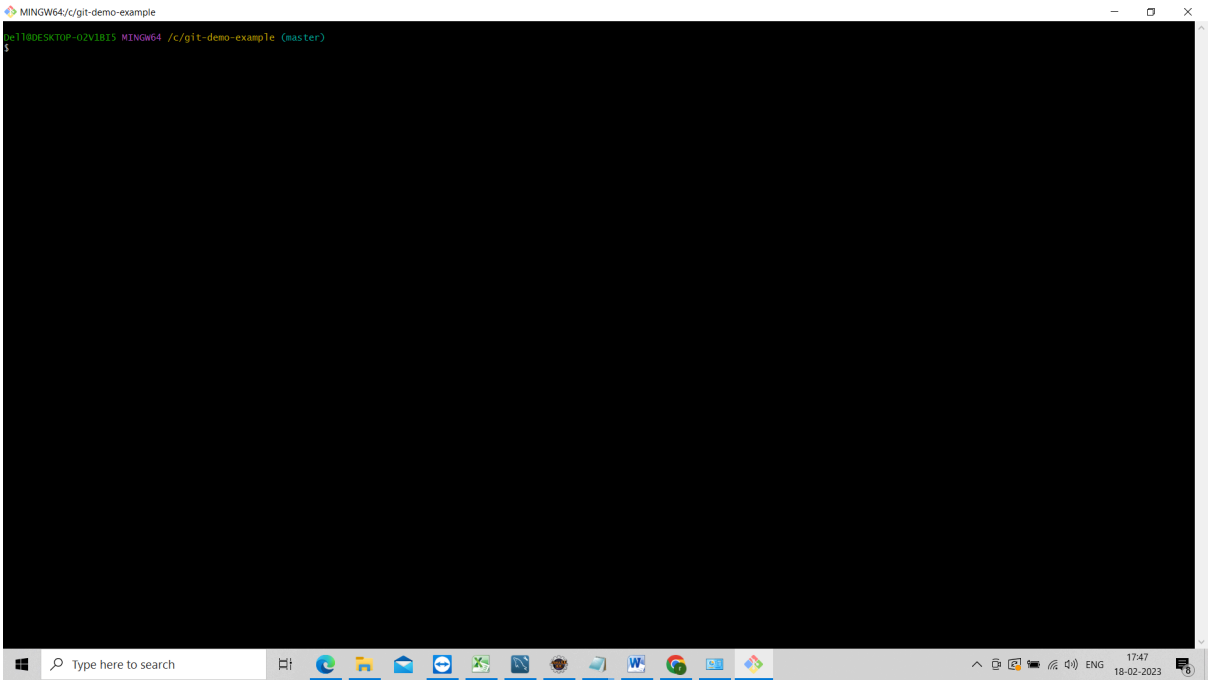


Switch to master branch

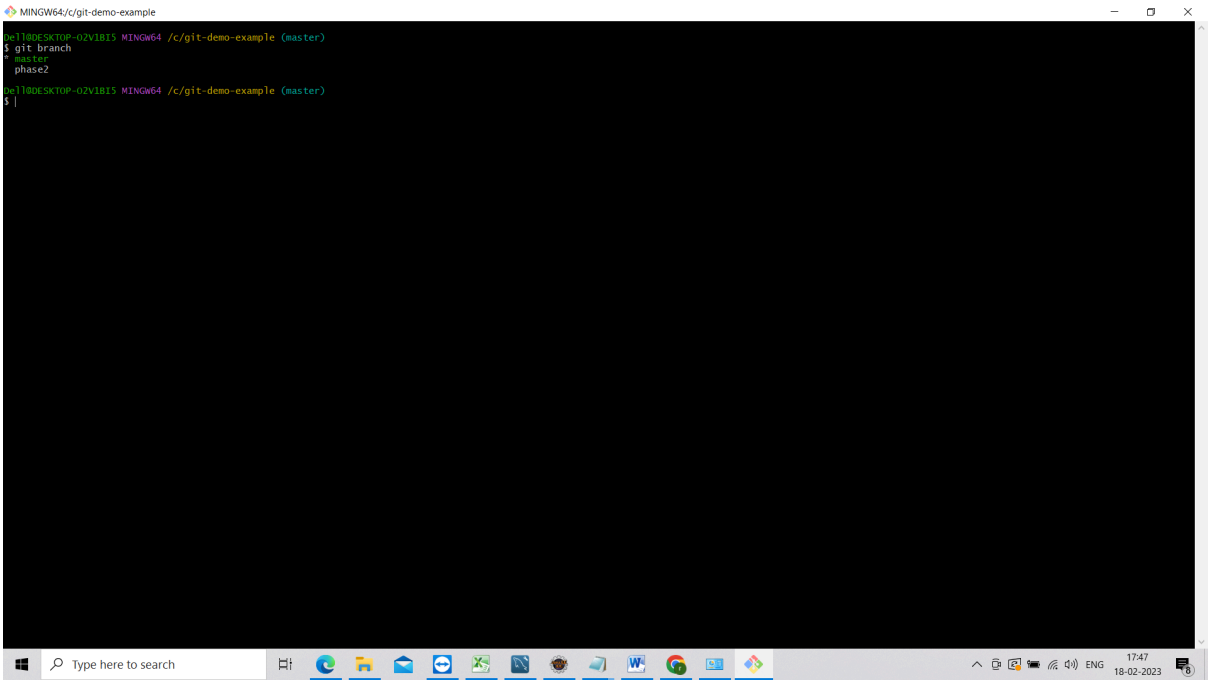




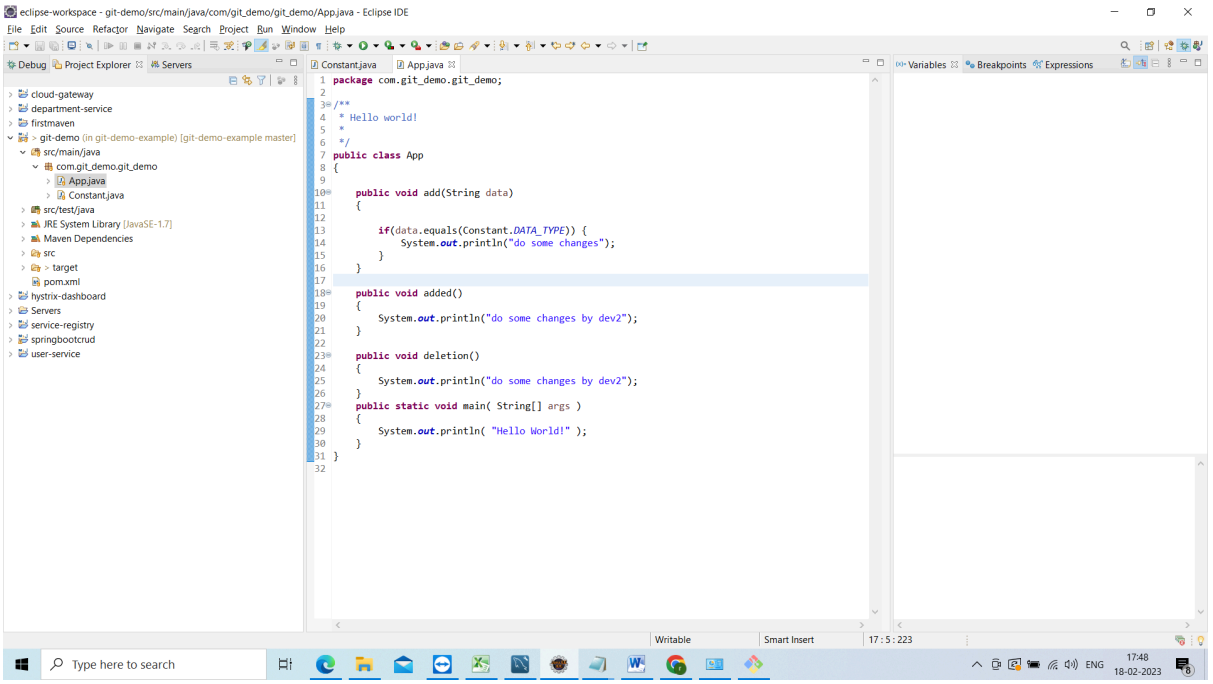
Now we are in master branch just clear the screen enter clear



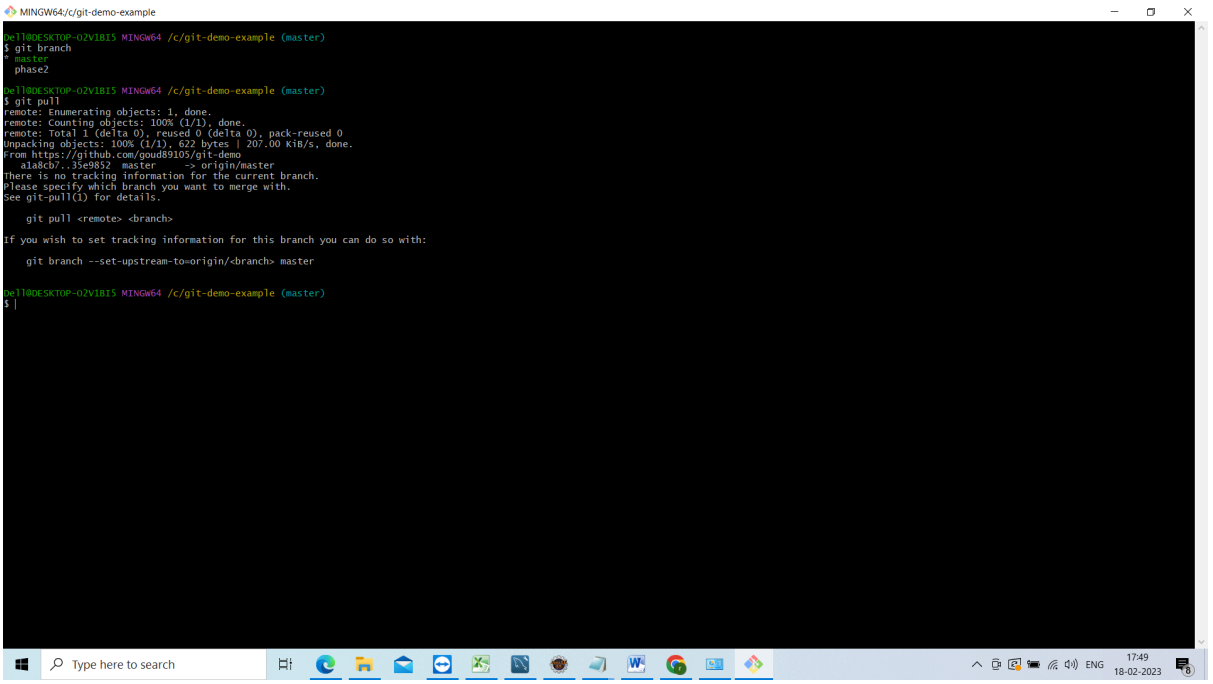
Now check git branch



Now go to eclipse and see we are in master branch chages not refleted to the master branch



Now go to git bash and do pull



Now go to eclipse and see ur chages will be reflected in the master branch

Git Introduction

Git is a **distributed version control system** that helps developers track changes in their code, collaborate with others, and manage different versions of a project efficiently.

---

1. Create & Manage Repositories

A Git repository is a storage location where the history of changes in a project is maintained.

Create a New Repository

- **Initialize a repository in an existing project**

bash

```
git init
```

- **Clone an existing repository**

bash

```
git clone <repository_url>
```

Manage Repositories

- **Check repository status**

bash

```
git status
```

- **View commit history**

bash

```
git log
```

---

2. Repository Settings

Git repositories can be configured using settings like user credentials and ignored files.

- **Set user details**

bash

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

- **Ignore specific files** (Create a .gitignore file and add patterns)

bash

```
*.log
node_modules/
```

- **View Git configurations**

bash

```
git config --list
```

---

3. Adding & Committing

Git tracks changes in **staging area** before committing.

- **Add files to staging**

bash

```
git add <file>
git add .
```

- **Commit changes**

bash

```
git commit -m "Commit message"
```

- **Amend last commit**

bash

git commit --amend -m "Updated commit message"

---

4. Branches

Branches allow multiple versions of a project to exist simultaneously.

- **Create a new branch**

bash

git branch <branch_name>

- **Switch to a branch**

bash

git checkout <branch_name>

- **Create and switch to a branch**

bash

git checkout -b <branch_name>

- **List all branches**

bash

git branch

- **Delete a branch**

bash

git branch -d <branch_name>

---

5. Stashing

Stashing temporarily saves changes without committing them.

- **Stash current changes**

bash

git stash

- **Apply stashed changes**

bash

CopyEdit

git stash pop

- **List stash entries**

bash

git stash list

- **Drop a stash entry**

bash

git stash drop

---

6. GitHub

GitHub is a cloud-based platform for hosting Git repositories.

- **Push local repository to GitHub**

bash

CopyEdit

git remote add origin <repository_url>

git push -u origin main

- **Clone a repository from GitHub**

bash

git clone <repository_url>

---

7. Fetching & Pulling

Fetching and pulling bring updates from remote repositories.

- **Fetch changes without merging**

bash

CopyEdit

git fetch

- **Pull changes and merge automatically**

bash

git pull

---

8. Git Collaboration

Collaboration involves multiple contributors working on a project.

- **Push changes to remote**

bash

CopyEdit

git push origin <branch_name>

- **Merge branches**

bash

git merge <branch_name>

- **Resolve merge conflicts manually in affected files, then commit changes.**

---

9. Git Tags

Tags mark specific points in a project’s history.

- **Create a tag**

bash

CopyEdit

git tag <tag_name>

- **Push tags to remote**

bash

CopyEdit

git push origin <tag_name>

- **List all tags**

bash

git tag

---

10. Git Clone

Cloning copies a repository from a remote source.

- **Clone a repository**

bash

git clone <repository_url>

---

11. Git Forking

Forking creates a personal copy of another user's repository on GitHub.

- **Steps to Fork a repository:**
    1. Open the repository on GitHub.
    2. Click the "Fork" button.
    3. Clone the forked repository to your local system:

bash

git clone <your_forked_repo_url>

---

12. Merge Requests

A Merge Request (or Pull Request on GitHub) is used to merge changes from one branch into another.

- **Steps to create a Merge Request:**
    1. Push your changes to a branch.
    2. On GitHub, navigate to your repository.
    3. Click **New Pull Request**.
    4. Select branches and submit.