**(Acknowledged w3school git study site)**

**What is Git?**

Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

**What is version control?**

Version control is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

**What does Git do?**

* Manage projects with Repositories.
* Clone a project to work on a local copy.
* Control and track changes with Staging and Committing.
* Branch and Merge to allow for work on different parts and versions of a project.
* Pull the latest version of the project to a local copy.
* Push local updates to the main project.

**What is GitHub?**

* Git is not the same as GitHub.
* GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.
* GitHub makes tools that use Git.
* GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

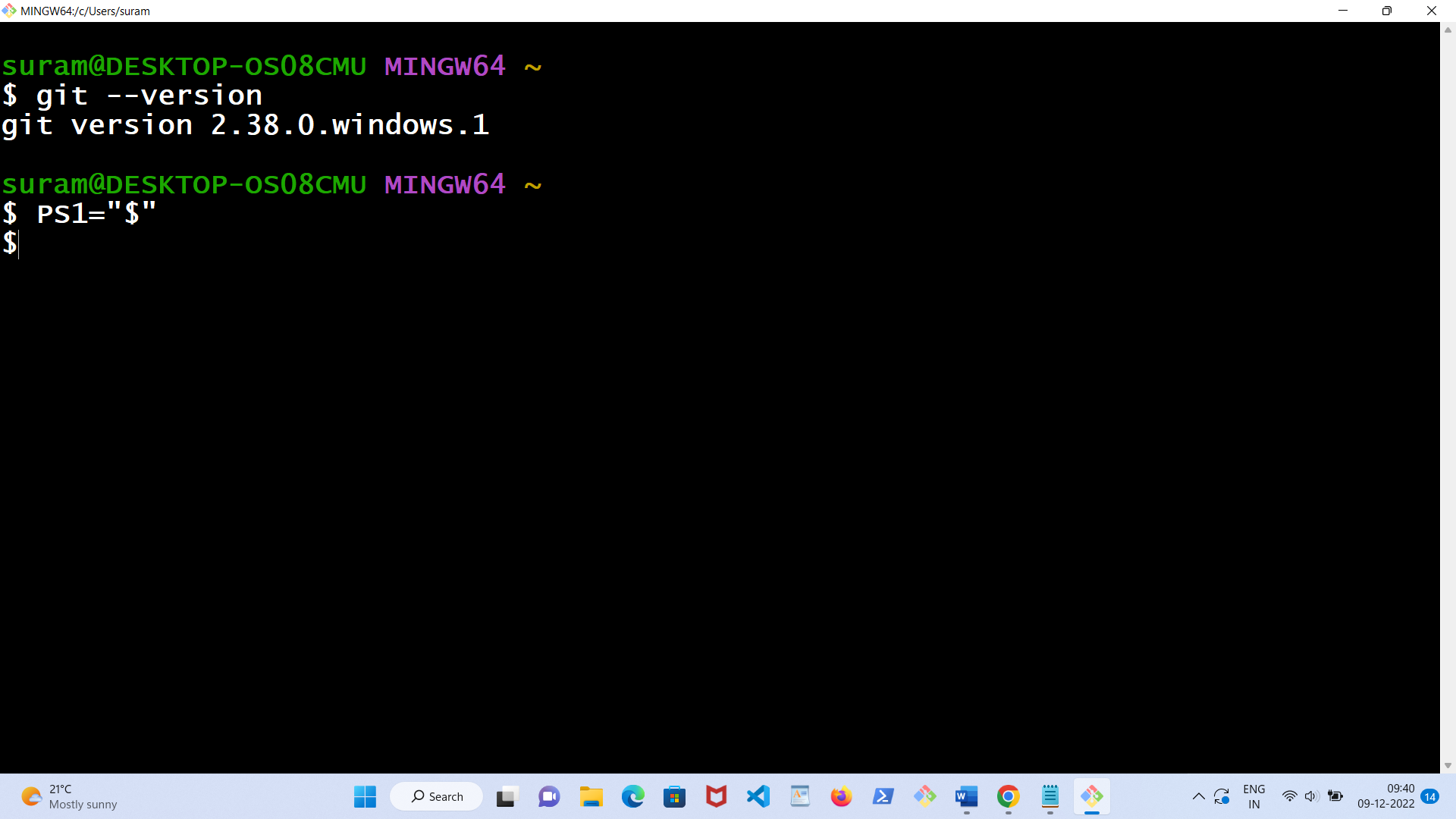
**Git Install**

You can download Git for free from the following website: [https://www.git-scm.com/](https://git-scm.com/)

## Using Git with Command Line

To start using Git, for Windows you can use Git bash, which comes included in Git for Windows.

Check the Git version there by typing the command ***git –version***.



Note: PS1=”$” command will convert GitBash ‘s prompt as “$” .

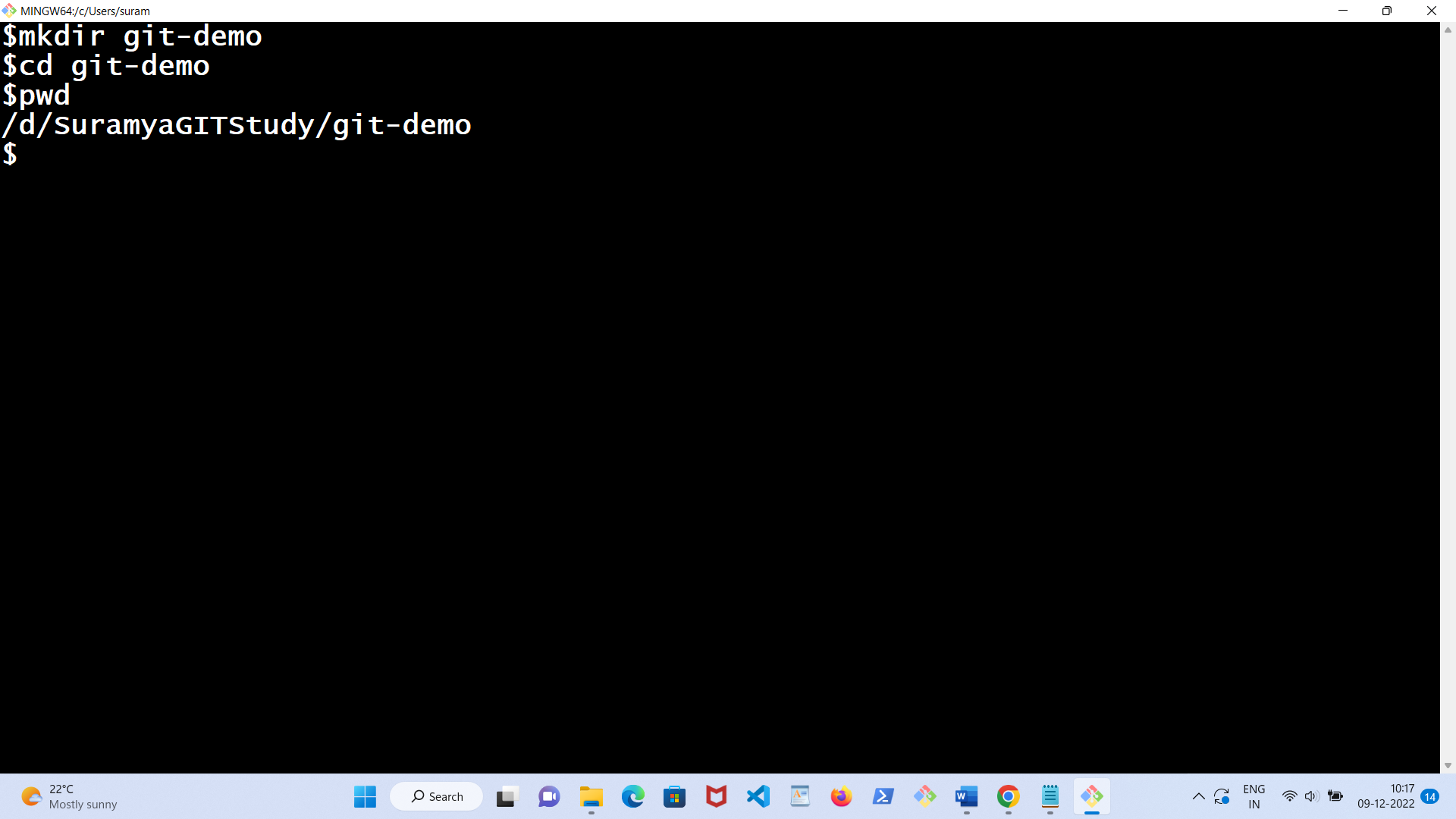
## Configure Git

$ git config --global user.name "any\_user\_name"

$ git config --global user.email "any\_email@any.com"

**Practise the following: (the prompt of GitBash is $):**

* mkdir git-demo
* cd git-demo
* pwd



* Create a file README.md inside the git-demo folder with following text:

# git-demo

GIT repository for Git tutorial

This is an example repository for the Git tutoial

This repository is built step by step in the tutorial.

It is included in GITHUB.

* Create css file welstyle.css in same location with following text:

body {

background-color: yellow;

}

h1 {

color: red;

margin-left: 20px;

}

h2{

color: blue;

margin-left: 20px;

}

h3 {

color: green;

margin-left: 20px;

}

h4 {

color: orange;

margin-left: 20px;

}

* Create Welcome.html file with following text:

<!DOCTYPE html>

<html>

<head>

<title>Welcome!</title>

<link rel="stylesheet" href="welstyle.css">

</head>

<body>

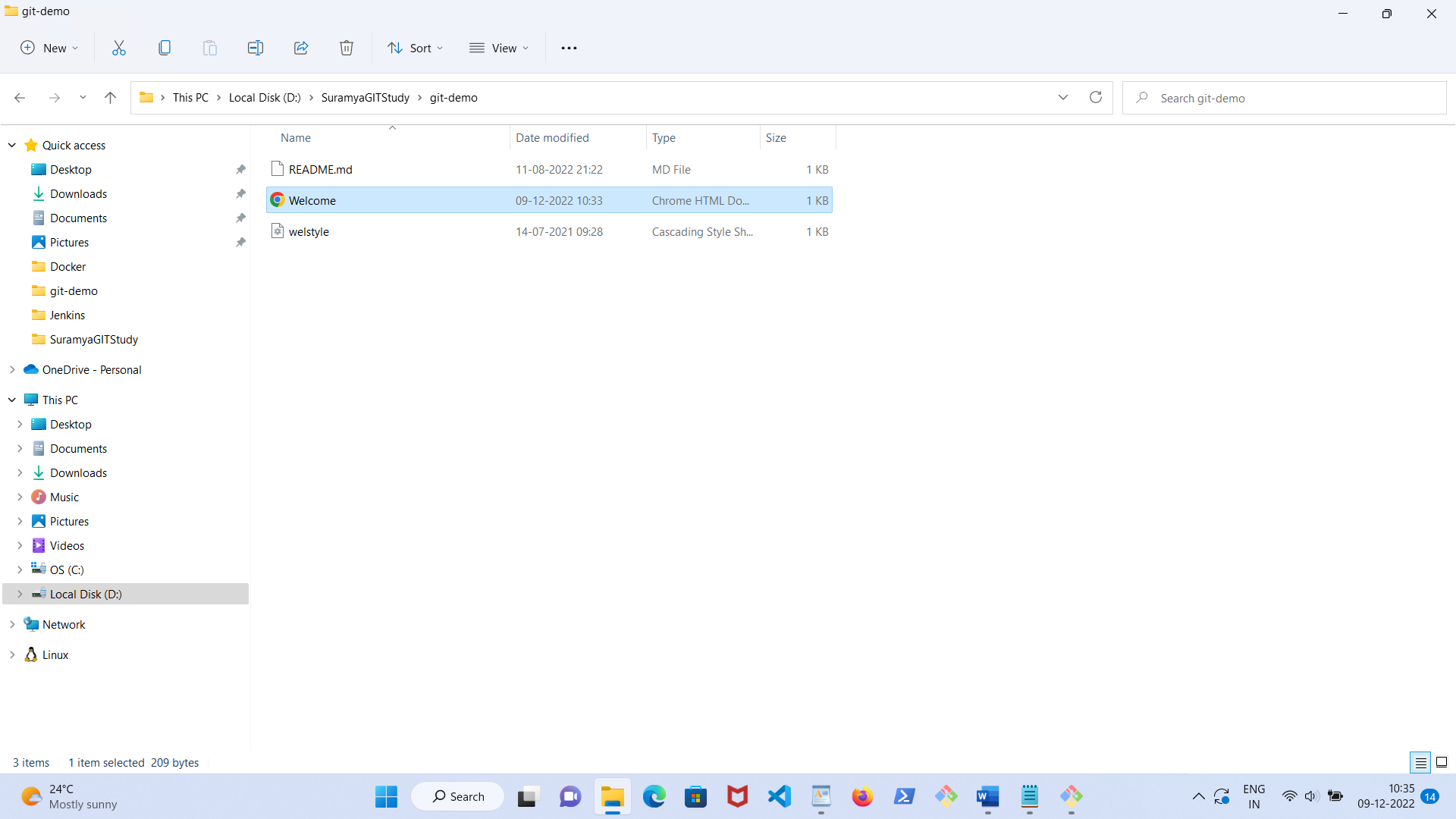
<h1>Hello people!</h1>

<h2>This is one file in git-demo folder.</h2>

</body>

</html>

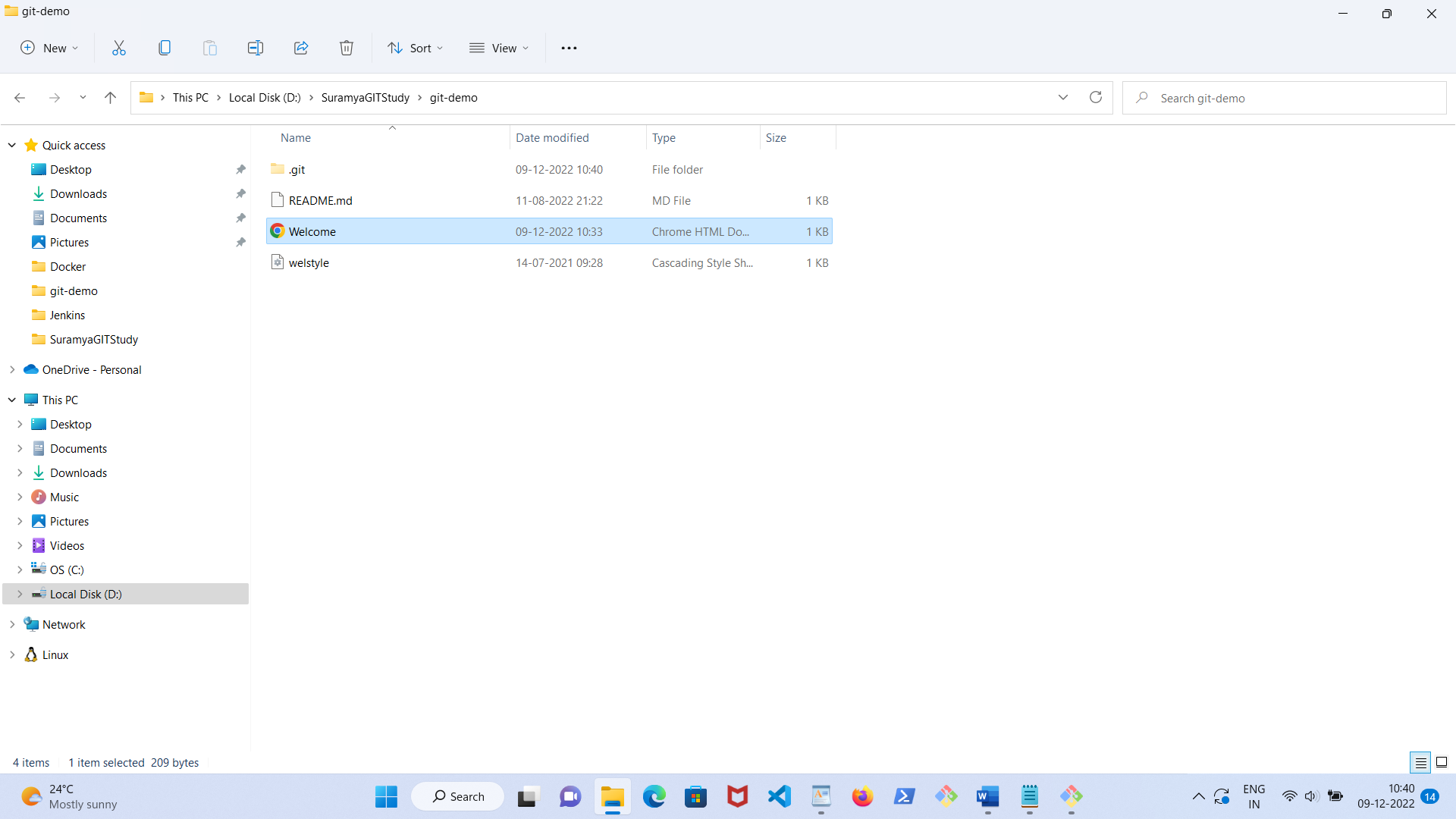
* git-demo folder looks like from windows explorer as follows:



Type following command:

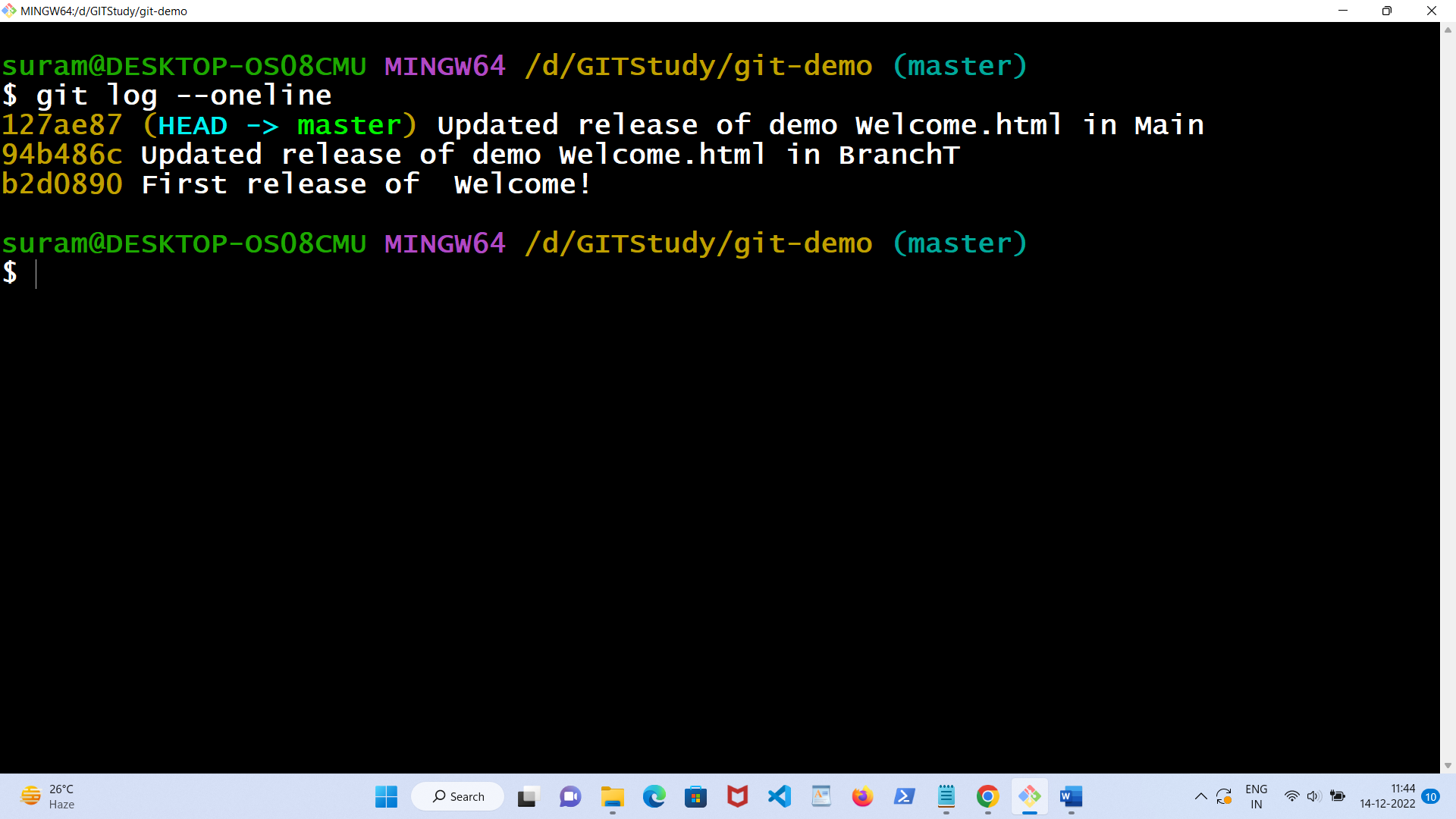
* git init
* git add --all
* git commit -m "First release of Welcome!"

Now look at the followings:

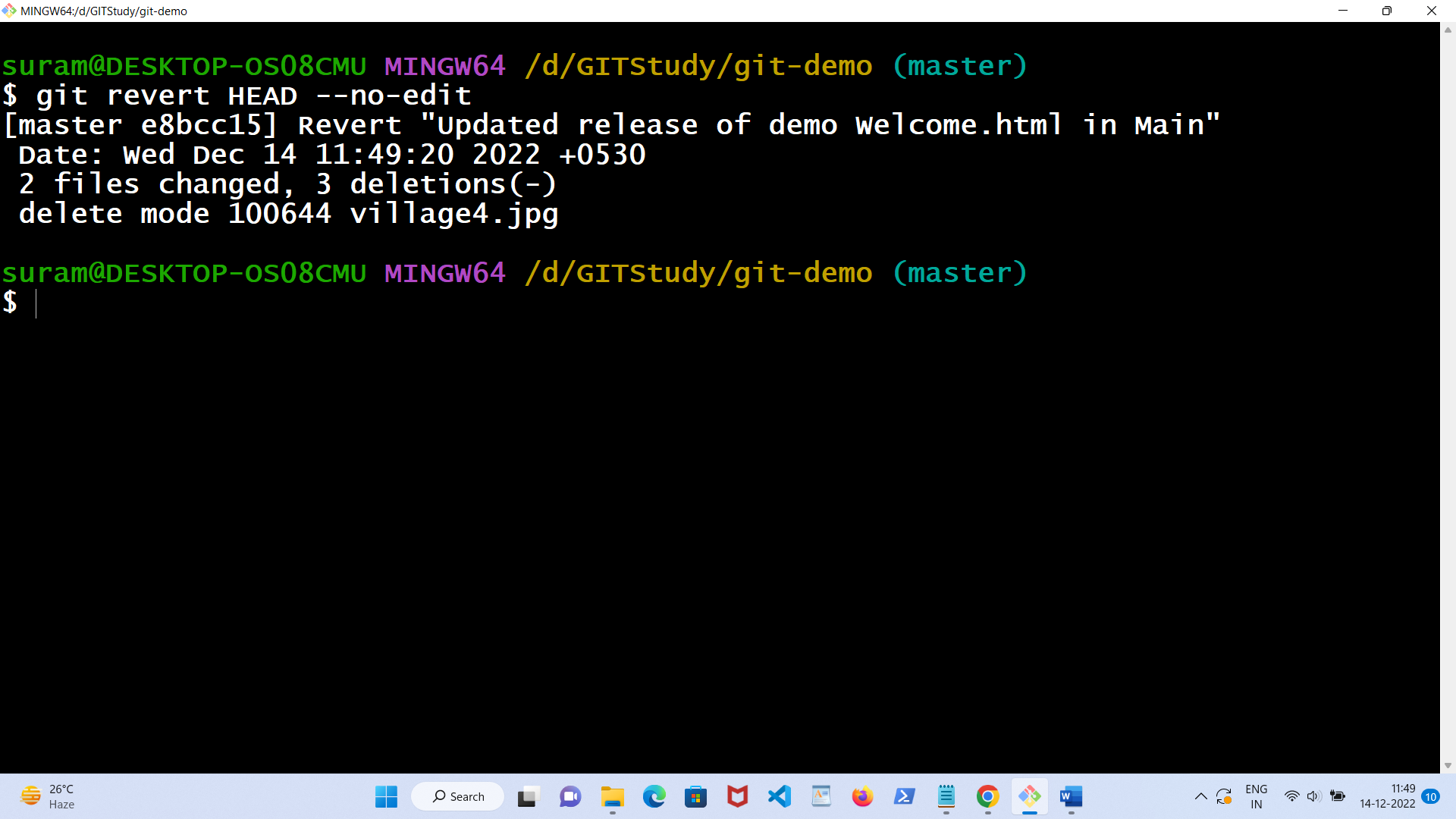


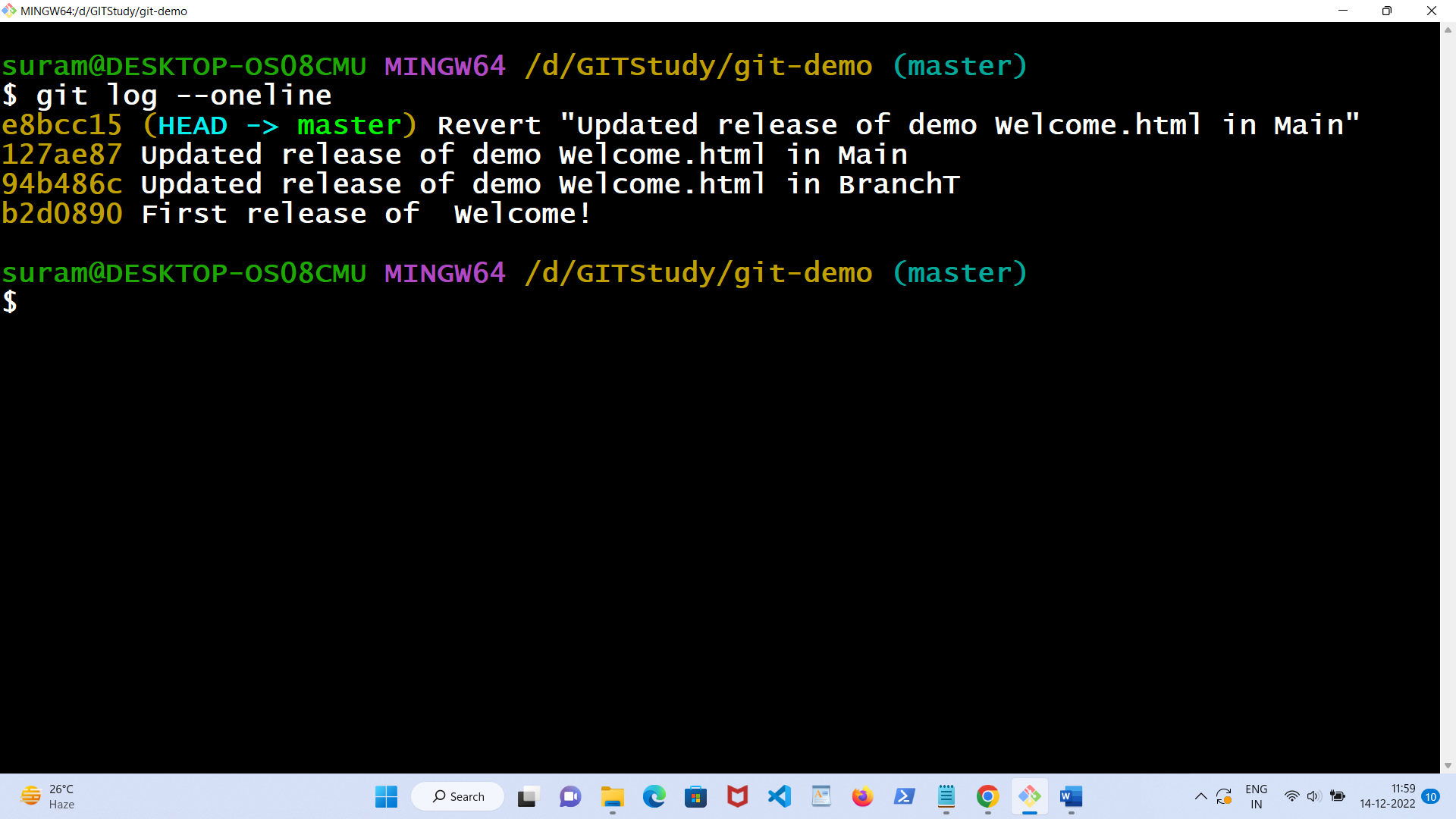
One new folder .git is created . It is required to log every commit history and every other information required for your remote repository, version control, commits etc.

* git add --all
* git commit -m "First release of Welcome!" 🡪 all files are saved in local git repository with that message
* git branch BranchA 🡪 create a branch BranchA
* git branch 🡪 shows all branches at present \*master , BranchA – means master is the active one.
* git checkout BranchA 🡪 BranchA is now active one
* (add new text & attach an image file in Welcome.html)
* git add --all 🡪 added everything in BranchA
* git commit -m “Second release of Welcome.html”
* git checkout master
* (view the Welcome.html in master branch from Windows explorer, the entries of BranchA are not present)
* git checkout -b BranchT 🡪 branch BranchT is created and it is now active branch
* git branch 🡪 displays all three branches
* (add new text & a new image in Welcome.html)
* git add - -all
* git commit -m "Updated release of Welcome.html in BranchT".
* git checkout master
* git merge BranchT 🡪 merging the contents of BranchT into master.
* (Check the contents of master)
* git branch -d BranchT 🡪 delete BranchT
* (add some text in html file)
* git add --all
* git commit -m "Updated release of demo Welcome.html in Main"
* git log –oneline



* git revert HEAD --no-edit (using git revert HEAD reverts the latest change, and then commit, adding the option --no-edit to skip the commit message editor i.e getting the default revert message).



* (check the master folder to find newly added texts & image file are missing)
* git log –oneline
* 
* (look above the id of last commit is **127ae87**).
* git reset 127ae87 🡪 reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit i.e the commit at stage of merging of BranchT with main.
* git log - -oneline
* (add new text in html file)
* git add - - all
* git commit -m “Agin updat the fil” (deliberately wrong spellings)
* git log - -oneline.
* git commit - - amend -m “Again update the file” 🡪 amends the message
* git log - -oneline
* git branch
* git status
* (open your github account and create a github repository named git-demo and remember the URL of that one)

Eg . [https://github.com/<user-name>/git-demo.git](https://github.com/%3cuser-name%3e/git-demo.git)

* git remote add origin [https://github.com/<user-name>/git-demo.git](https://github.com/%3cuser-name%3e/git-demo.git)
* git push -u origin master
* (go to the github and view the Welcome.html file)
* (add new text in github’s Welcome.html file then commit it)
* (Check Welcome.html file in local git new texts are missing)
* git pull origin 🡪 check the local git new texts are present in Welcome.html
* (add new text and image in Welcome.html of local git)
* git add --all
* git commit -m "Updated release of Welcome in local repo!"
* git push origin 🡪 (check the github’s Welcome.html , new contents are reflected there)
* (create a branch in github named BranchB)
* git pull 🡪 all remote branches are updated & reflected
* git branch -a 🡪 display all local & remote branches
* git branch -r 🡪 display only remote branches
* git branch 🡪 display only local branches
* git push origin BranchB 🡪 pushing the contents of github's master to BranchB
* git checkout BranchA
* git push -u origin BranchA 🡪 pushing the content of BranchA to github, where a new BranchA also created.
* git branch -a 🡪 you will find two BranchA are listed.
* (add new text in Welcome.html file of github’s BranchA then commit)
* git pull origin 🡪 updated contents of Github’s BranchA is available in local BranchA
* git checkout master
* git checkout remotes/origin/BranchA 🡪 moves to github's BranchA
* git checkout master
* git branch -a 🡪

**BranchA**

**BranchB**

**\* master**

**remotes/origin/BranchA**

**remotes/origin/BranchB**

**remotes/origin/master**

* git remote rename origin upstream
* git branch -a 🡪

**BranchA**

**BranchB**

**\* master**

**remotes/upstream/BranchA**

**remotes/upstream/BranchB**

**remotes/upstream/master**

* git remote rename upstream origin
* git branch BranchP 🡪 creates a local branch BranchP
* git branch -a
* git branch -m BranchP BranchQ 🡪 rename local BranchP to BranchQ. (*Branch renaming is possible only for local git, not the github one*.)
* git branch -a
* git remote -v

**origin https://github.com/<user-name>/git-demo.git (fetch)**

**origin https://github.com//<user-name>/git-demo.git (push)**

**Git Ignore**

When sharing your code with others, there are often files or parts of your project, you do not want to share.

Examples

* log files
* temporary files
* hidden files
* personal files
* etc.

Git can specify which files or parts of your project should be ignored by Git using a .gitignore file.

**To create a .gitignore file, go to the root of your local Git, and create it:**

### **Example**

touch .gitignore

Now open the file using a text editor.

We are just going to add two simple rules:

* Ignore any files with the .log extension
* Ignore everything in any directory named temp

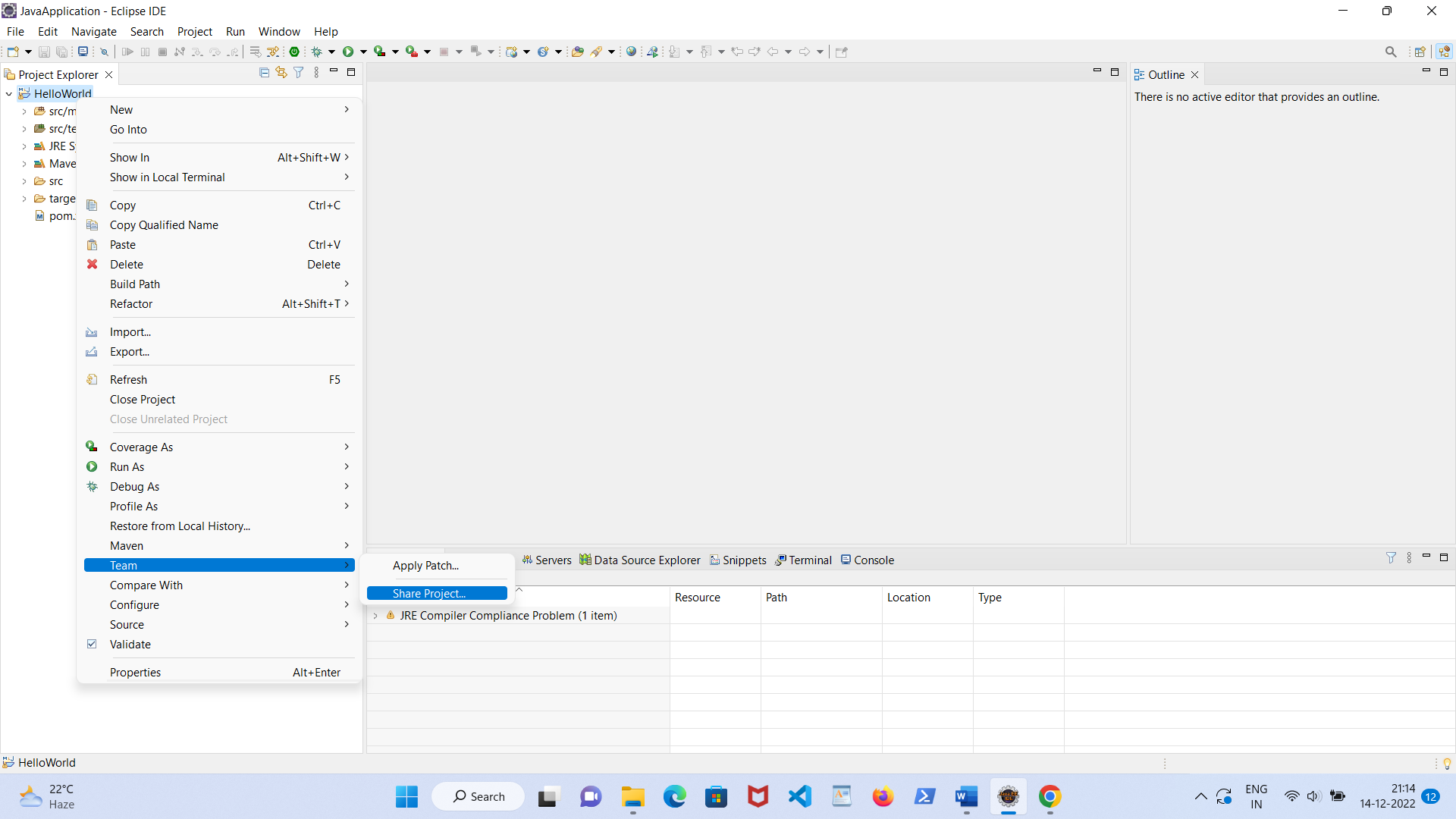
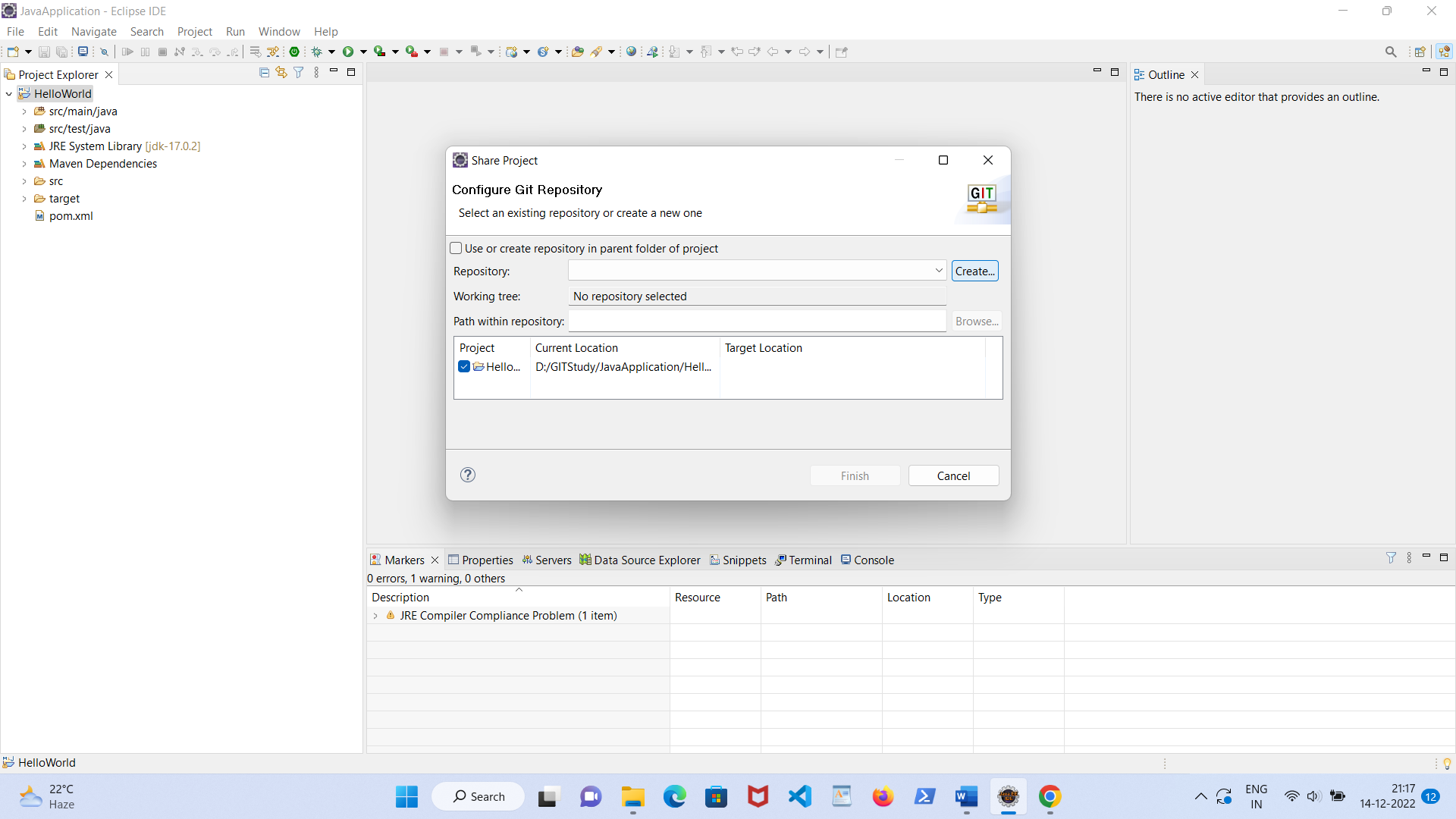
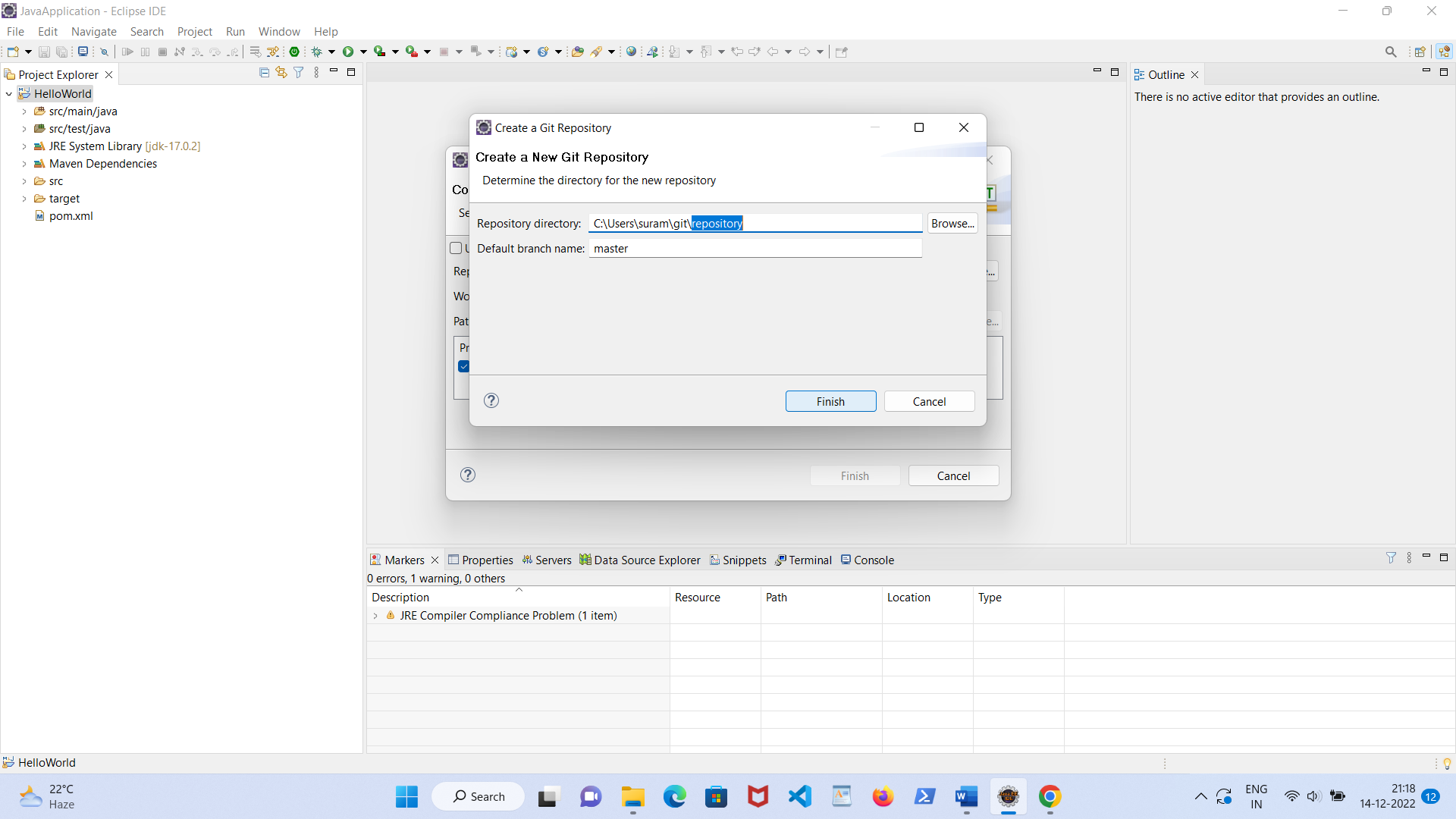
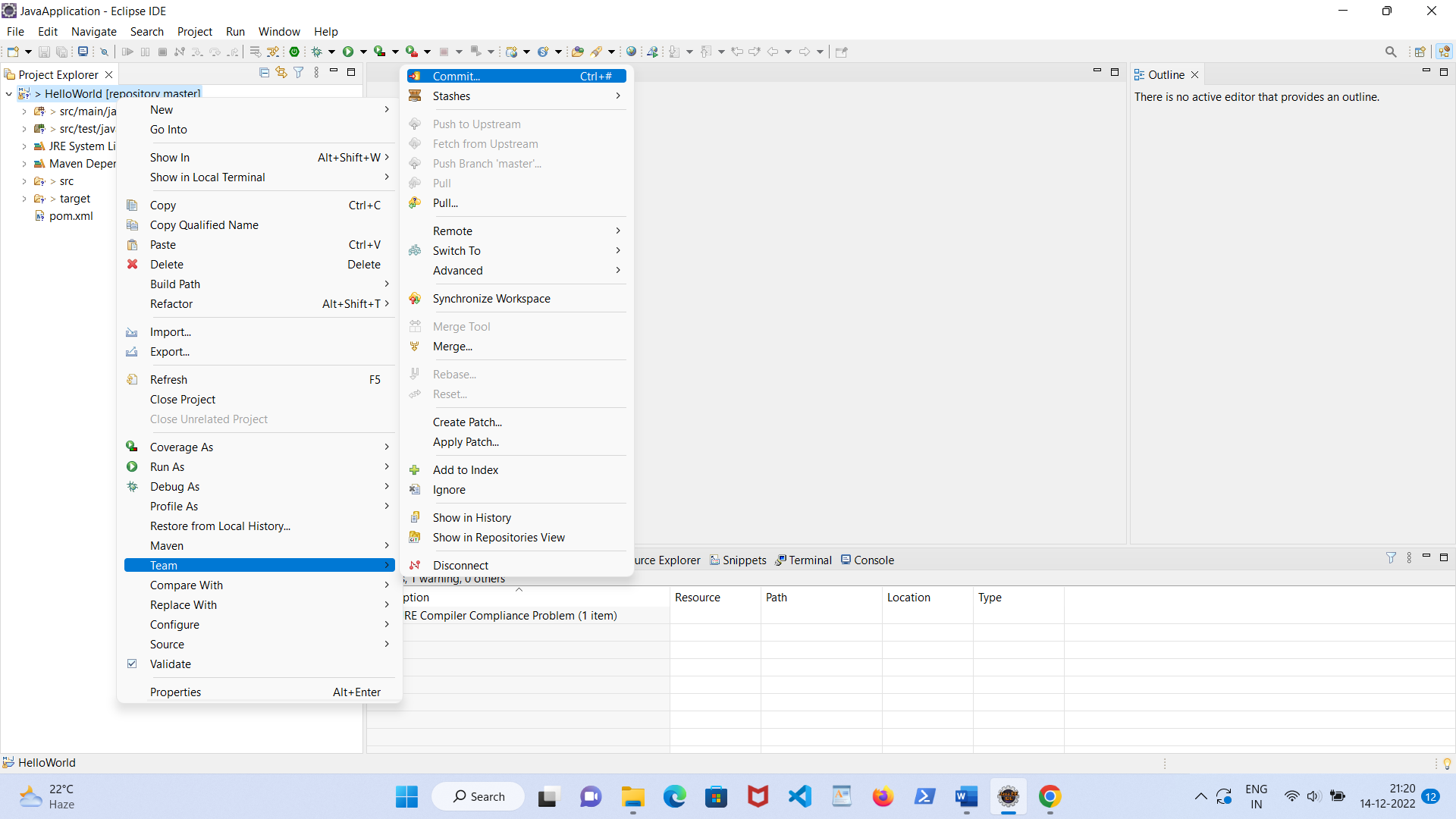
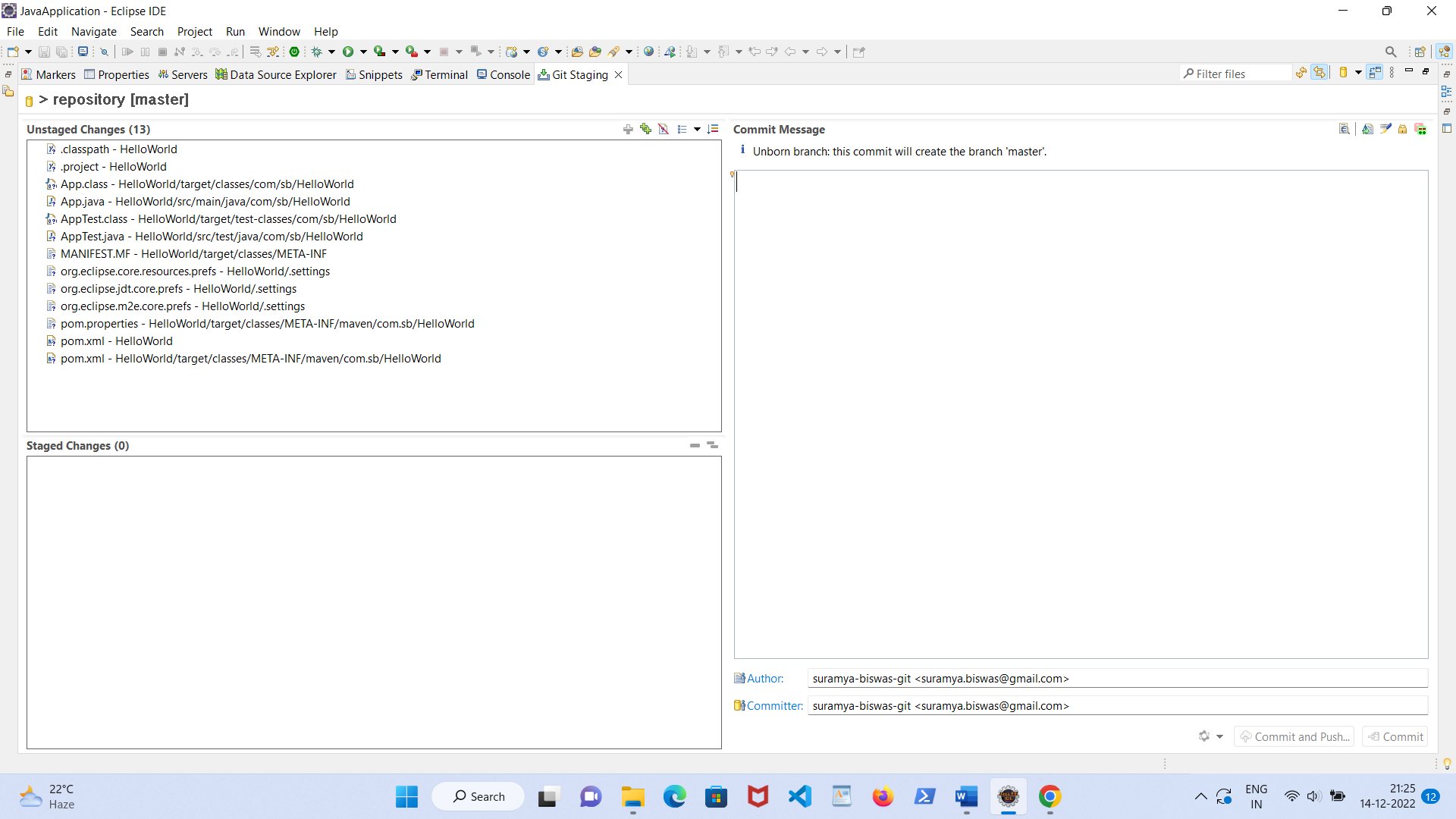
### **Example**

# ignore ALL .log files  
\*.log  
# ignore ALL files in ANY directory named temp  
temp/

Now all .log files and anything in temp folders will be ignored by Git.

* Git checkout master
* touch .gitignore
* go to the Windows explorer and open the .gitignore
* type the above code and save
* create a demo.log file
* create a mydemo.txt file
* git add - - all
* git commit -m “Add log & text files”
* git push -u origin master
* (check at github, you will see that mydemo.txt file is present but demo.log is absent )

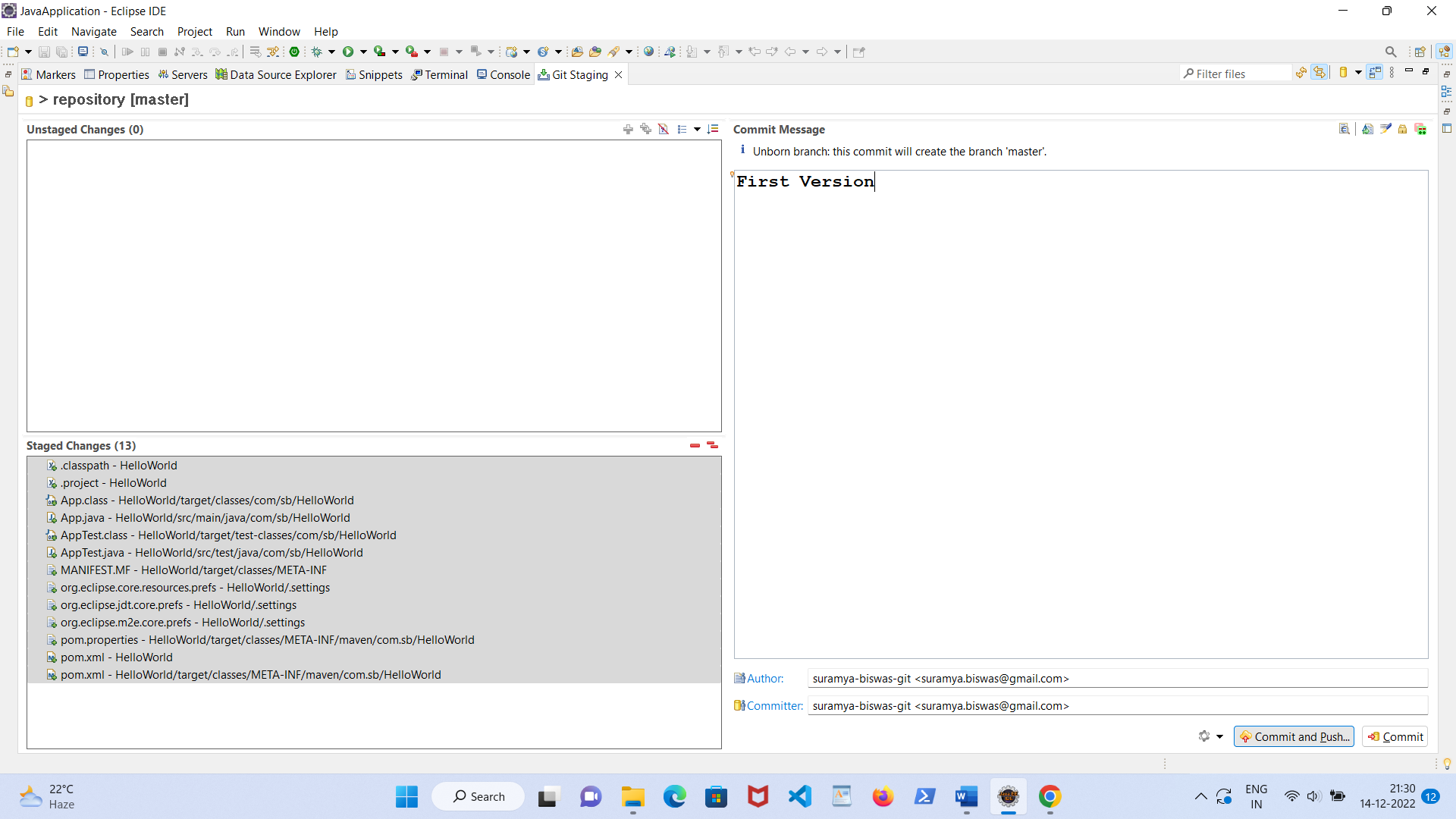
**Steps to send a Java Maven Application to Github from Eclipse**

* login to your GitHub account
* go to <https://github.com/settings/tokens>
* click on "Generate new token"
* make necessary selections (but must select repo)
* click on "save"
* System will have a token
* use this token instead of a password in the eclipse.
* Open Eclipse & create a Java Application in Maven -> HelloWorld
* Maven Update & Maven Build.
* Open your github & create a github repository “*HelloWorld*” 🡪 [https://github.com/<user-name>/HelloWorld.git](https://github.com/%3cuser-name%3e/HelloWorld.git).
* Go to eclipse select *HelloWorld,* right click then select Team->Share Project . 
* 
* 
* After clicking Finish , select *HelloWorld* and right click now select Team ->Commit.
* 
* 

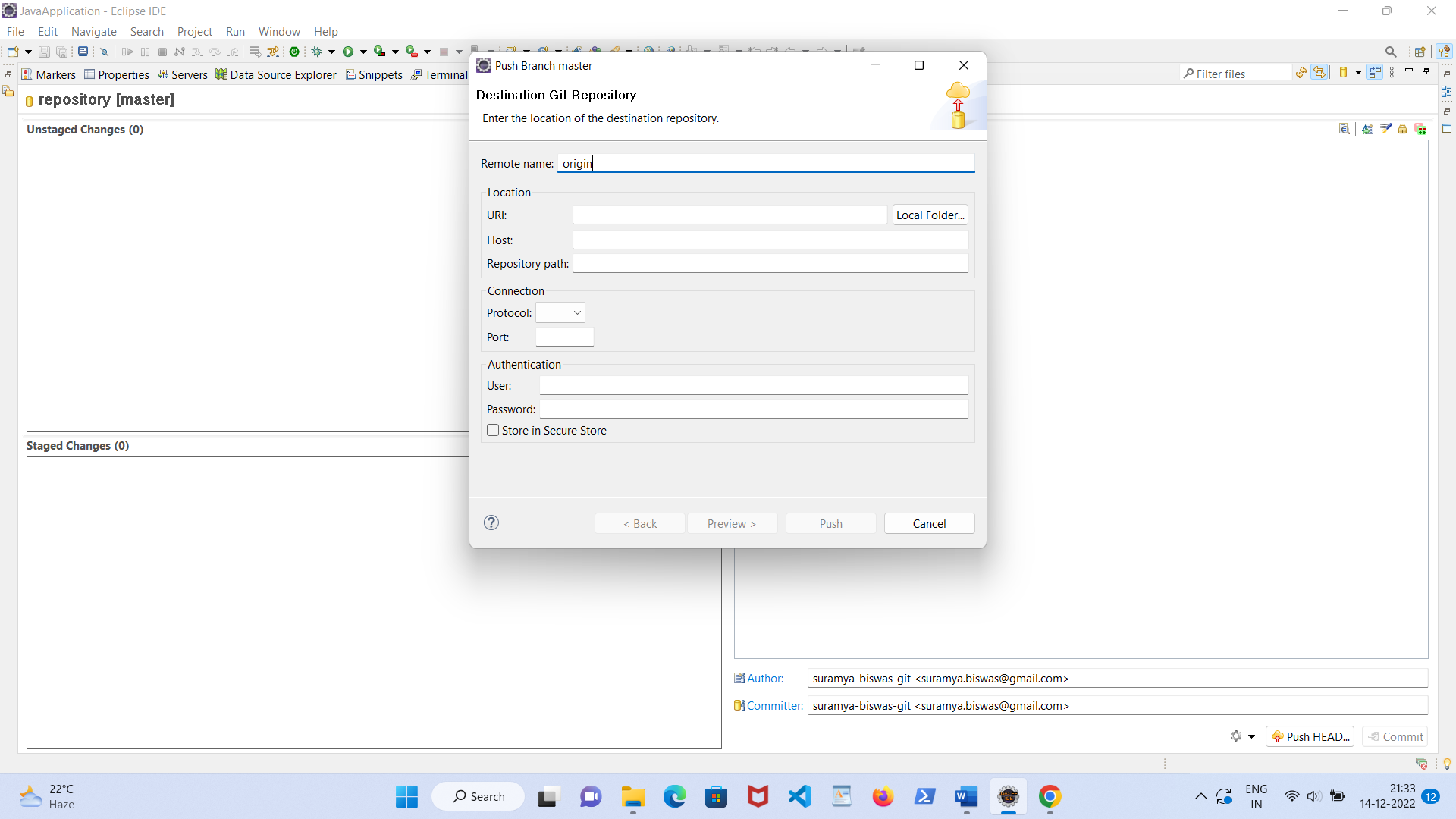


Select then drag and drop the *Unstaged Changes* box texts to *Staged Changes* box.

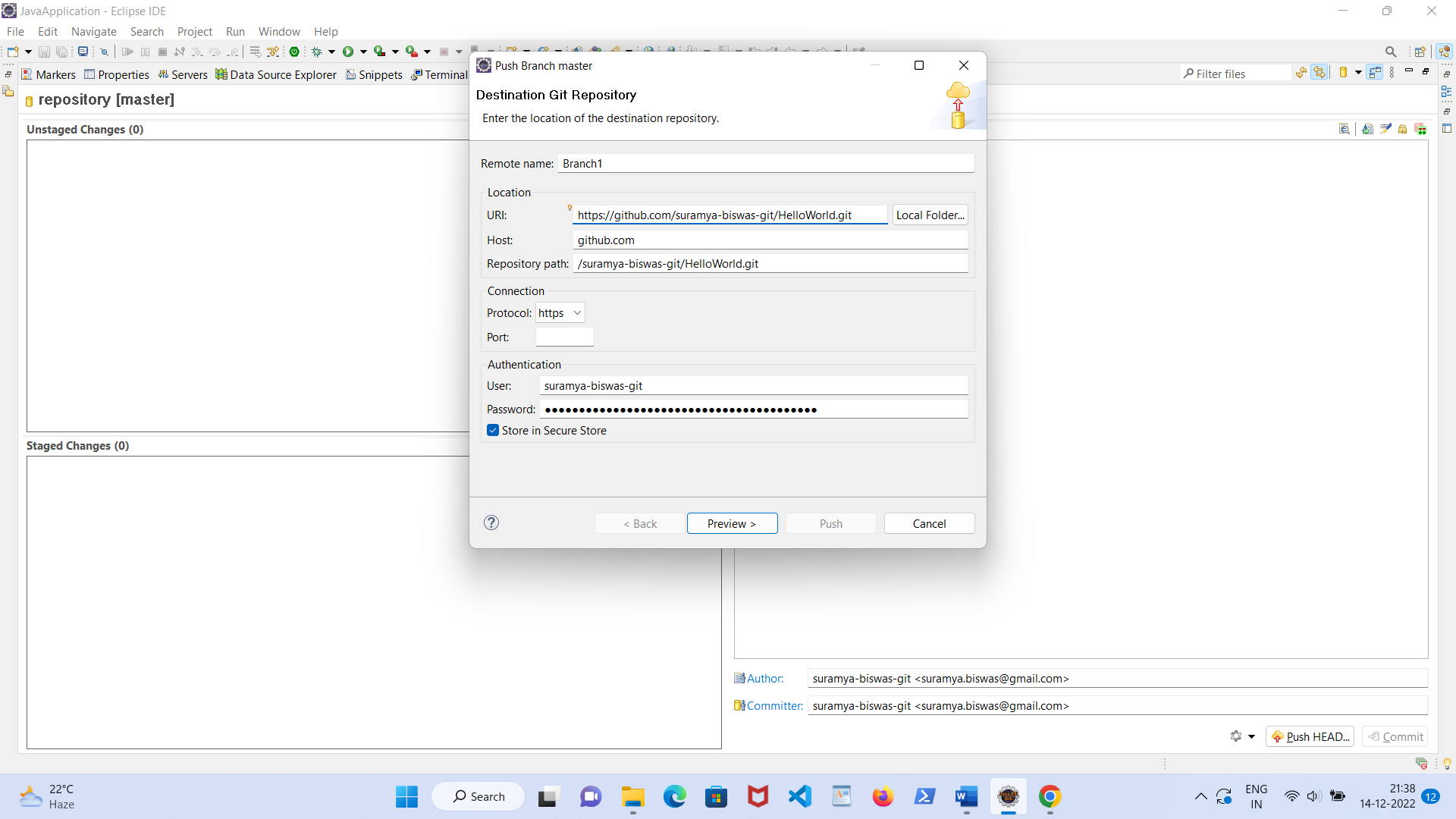
Write any text in Commit Message box.

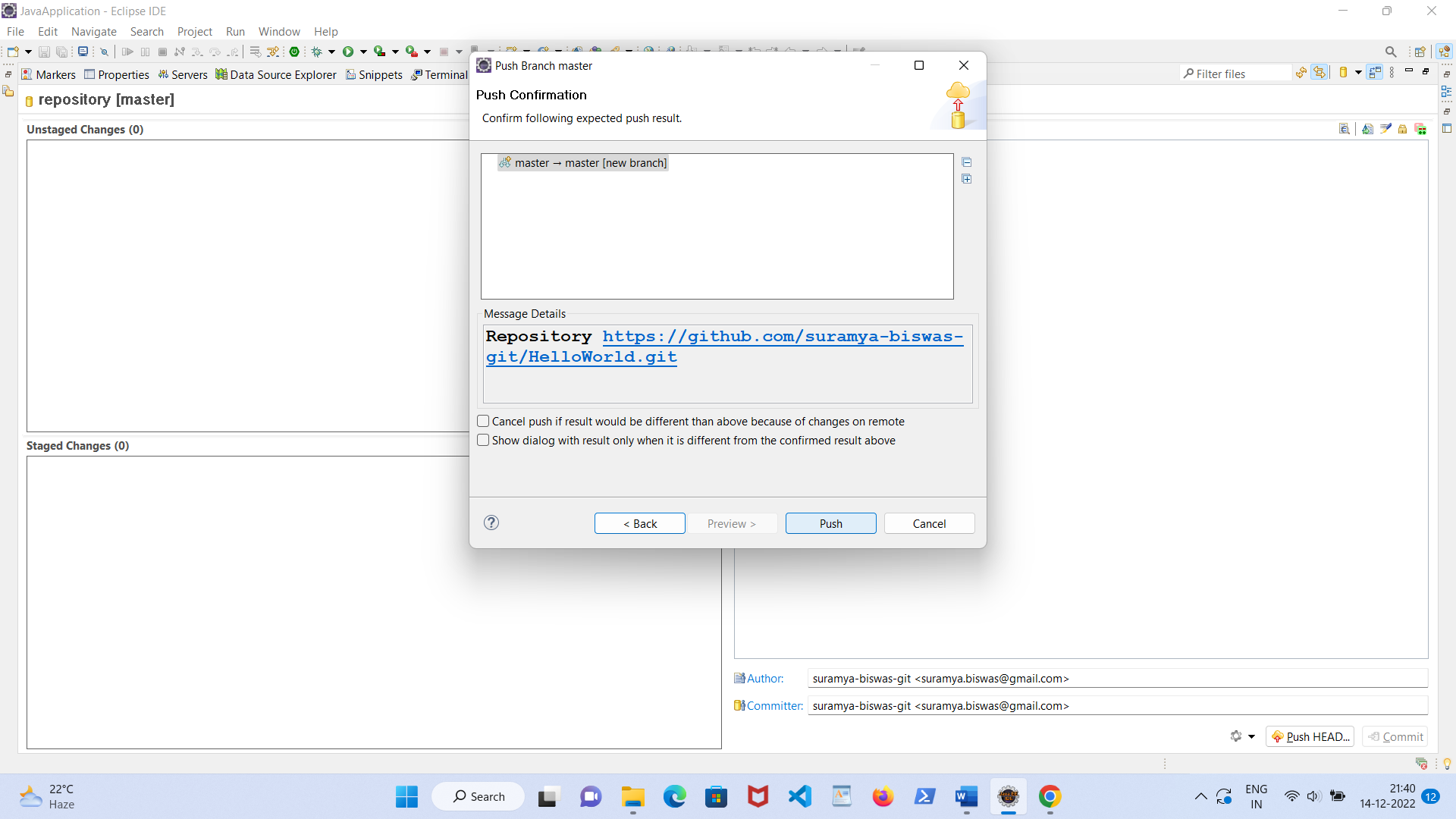
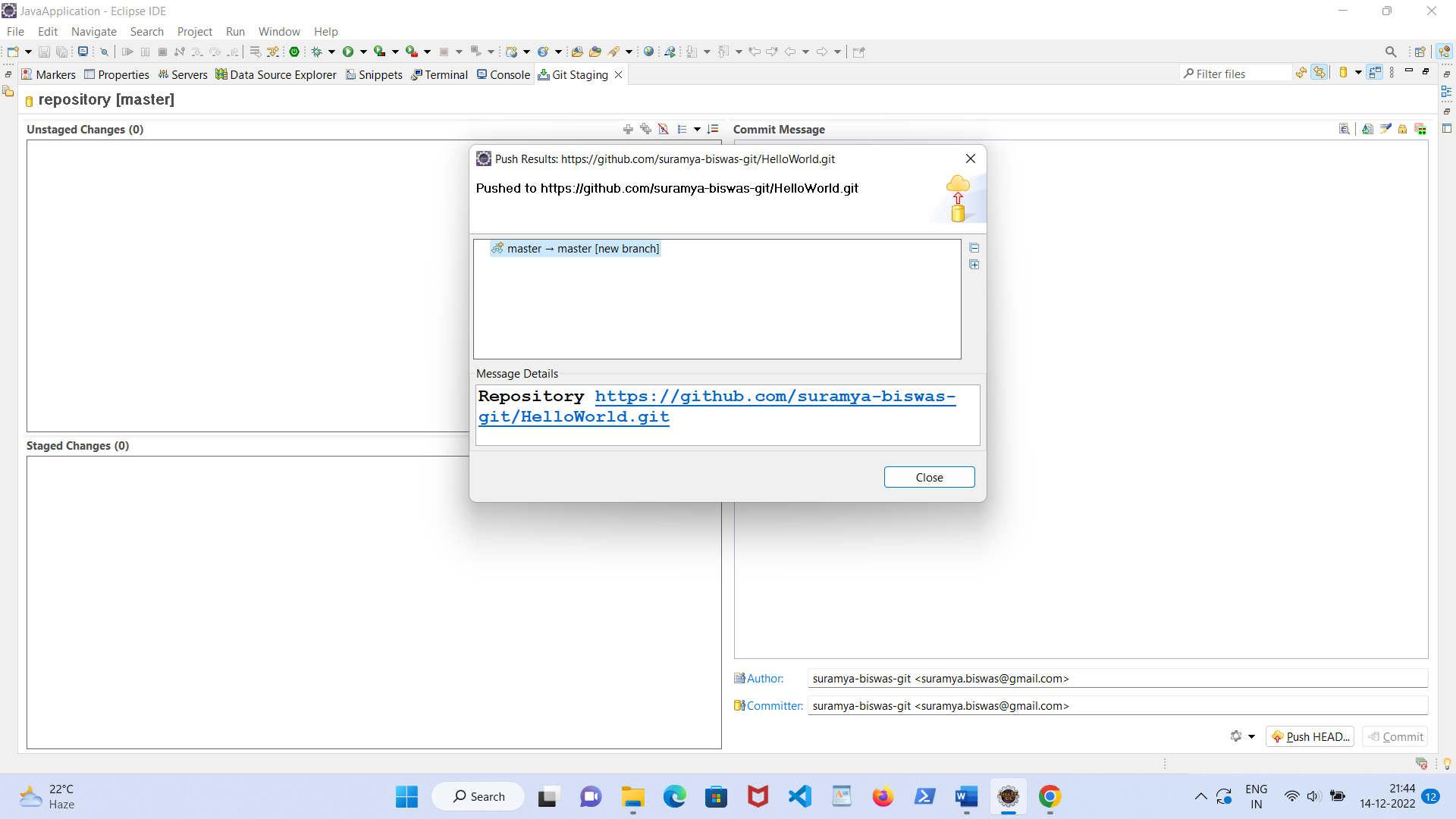


Click Commit and Push button. If any dialog box appears ignore it. Again click Push HEAD button.



* Change Remote name Origin to Branch1, Type github repository URL i.e [https://github.com/<user-name>/HelloWorld.git](https://github.com/%3cuser-name%3e/HelloWorld.git). Type the GitHub user id & token value in password then click Preview button



* Click Preview button again then Push Confirmation window will appear. Click Push button.
* 
* 
* Now check your GitHub repository where you will find all the classes & packages.
* For future modification you can delinked your Java application in the following way:
* 