Create a simple script file and push it to repo. Create a project in Jenkins connected to your GitHub repository. When a commit is made to your repo, automatically build must get triggered from Jenkins and the output must be shared to me via email.

Techstacks needs to be used

AWS EC2

Github

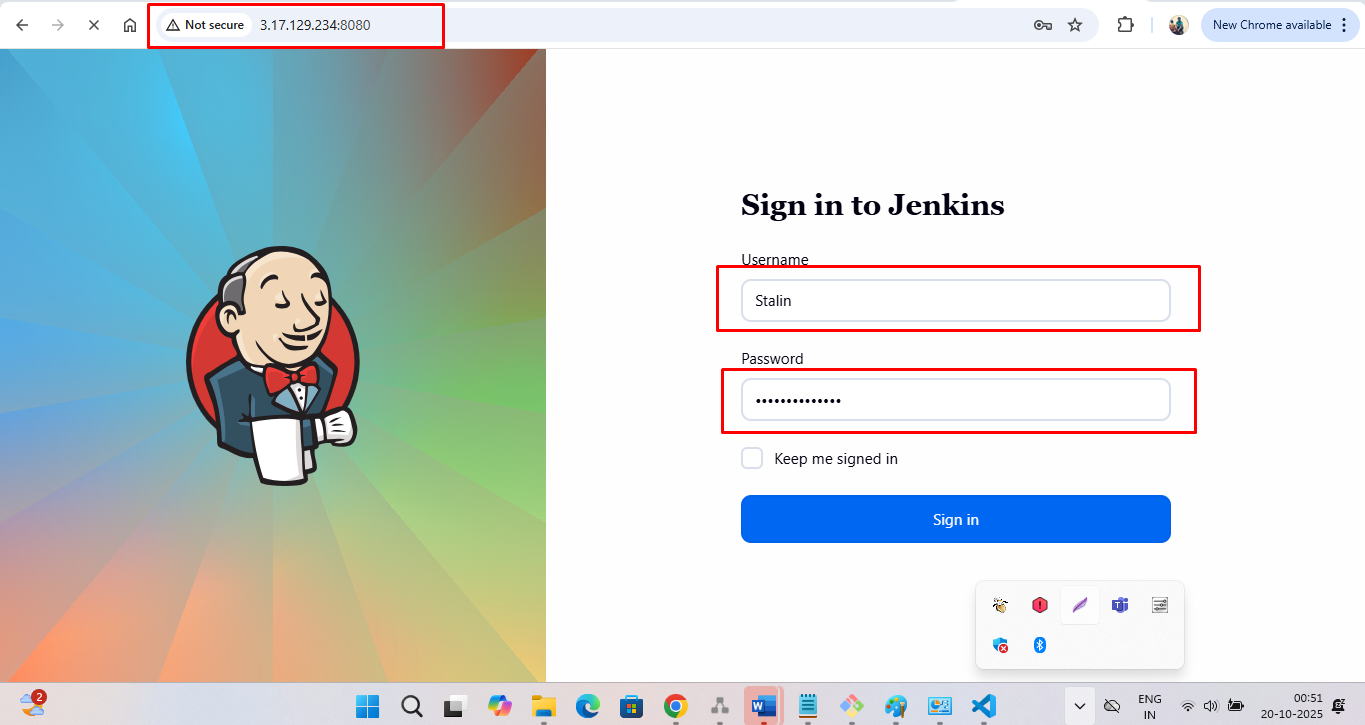
Jenkins

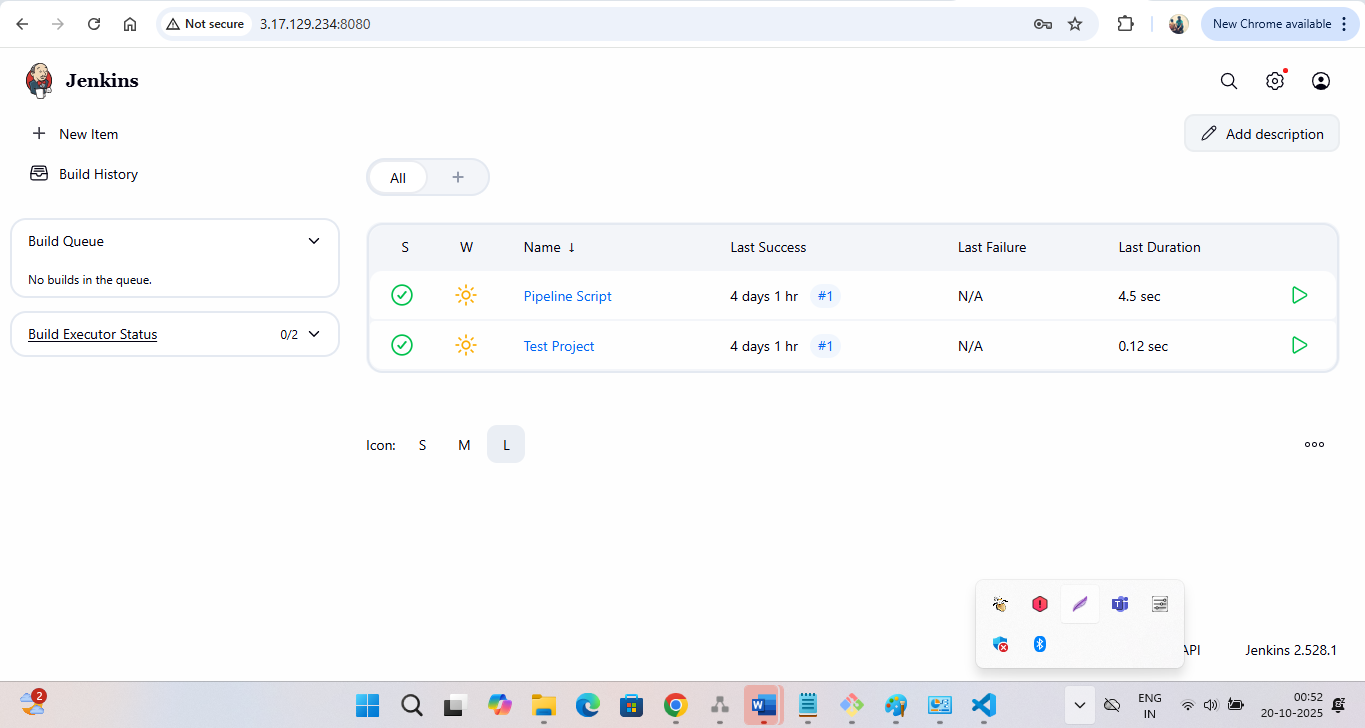
Note: I already installed Jenkins on my EC2 instance hosted in AWS and I am connecting through its public IP (refer Jenkins Task 1 for installation of the same.)

Syntax: http://<EC2-Public-IP>:8080

http:// 3.17.129.234:8080

Step 1: Login to your Jenkins with the username and password created at the beginning.





Step 2: Create a GitHub repository.

Create two files locally → hello.sh and Jenkinsfile

In the hello.sh, enter the below script.

echo '#!/bin/bash

echo "Hello from Jenkins on AWS EC2!"

date

' > hello.sh

Note: chmod +x hello.sh - Give a write permission to the same.

In Jenkinsfile, enter the below script.

pipeline {

    agent any

    stages {

        stage('Build') {

            steps {

                sh './hello.sh > output.txt'

            }

        }

    }

    post {

        success {

            emailext (

                subject: "Jenkins Build Successful - ${env.JOB\_NAME}",

                body: "Build completed successfully.\n\nOutput attached.",

                to: "your-email@example.com",

                attachmentsPattern: "output.txt"

            )

        }

        failure {

            emailext (

                subject: "Jenkins Build Failed - ${env.JOB\_NAME}",

                body: "Build failed. Check Jenkins logs.",

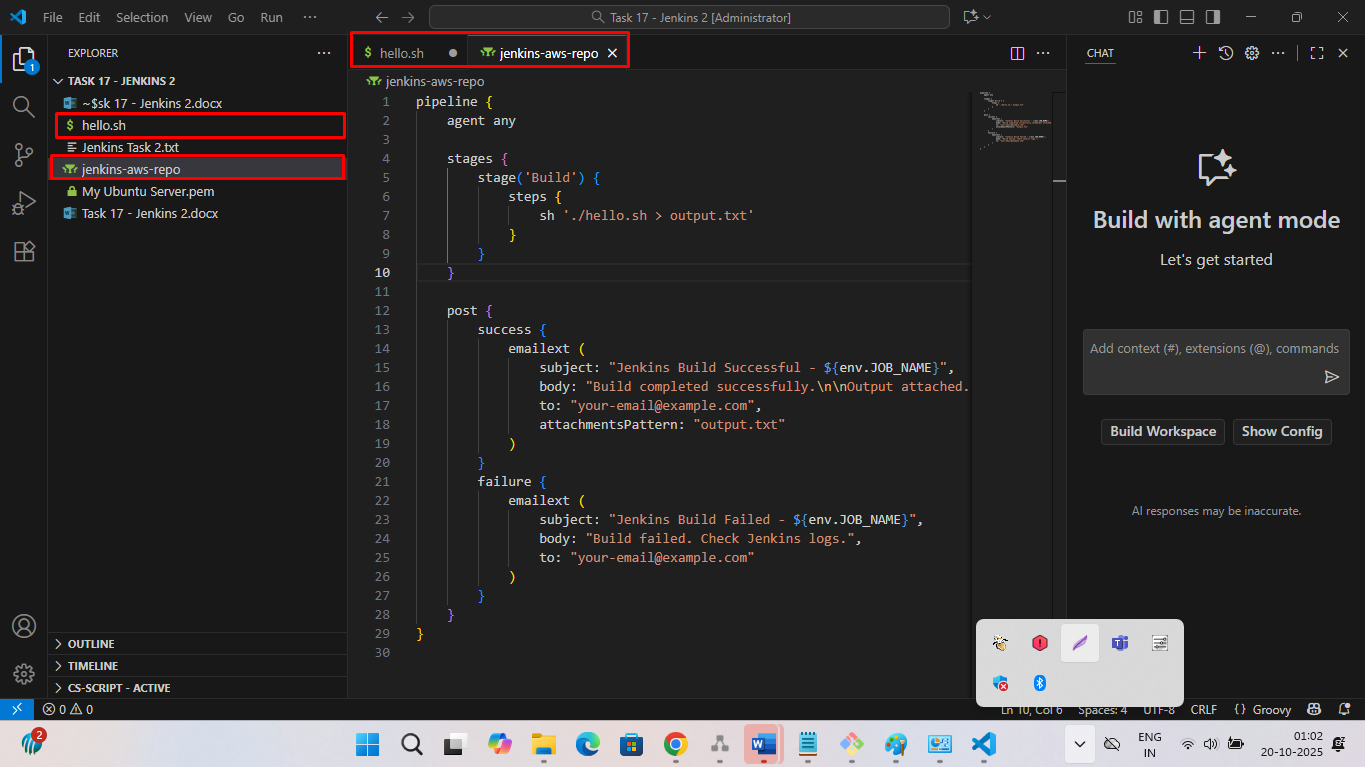
                to: "your-email@example.com"

            )

        }

    }

}



Step 3: Push these codes to GitHub repository.

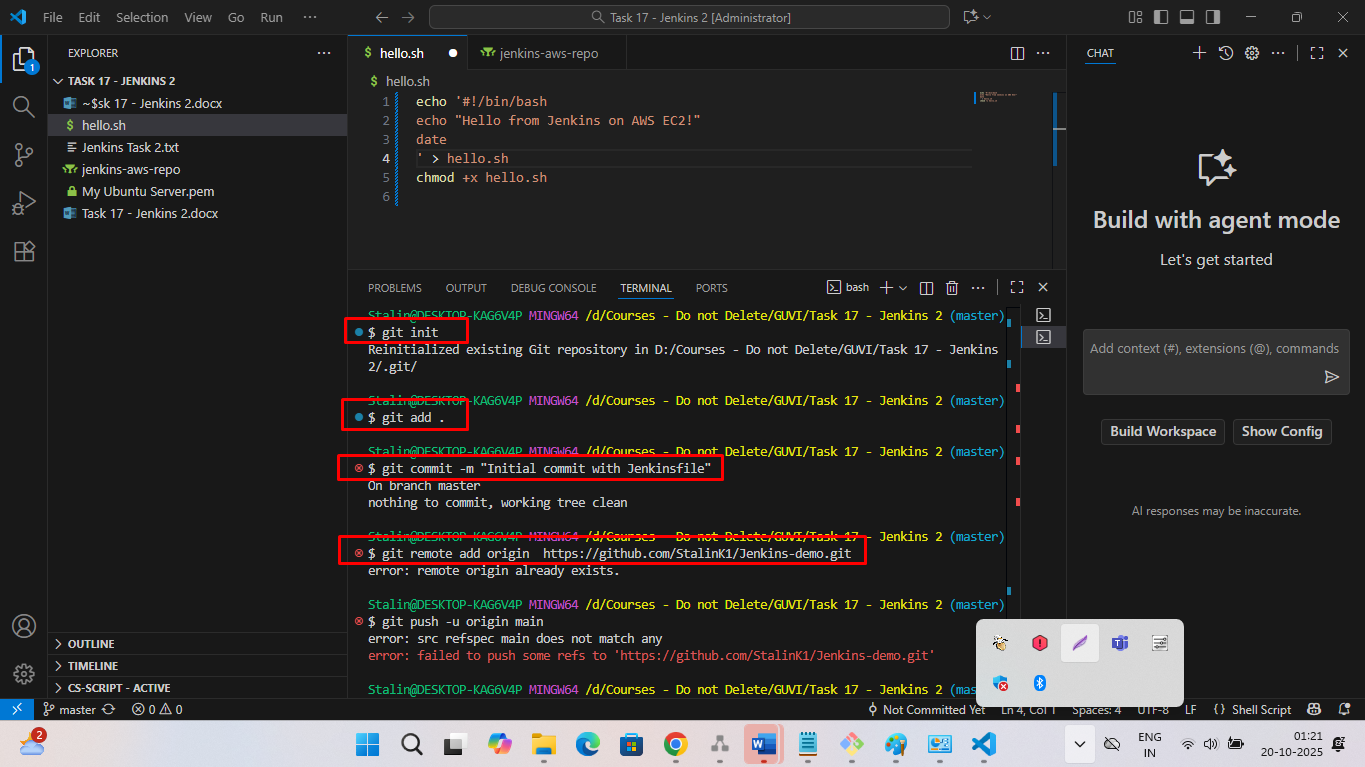
cmd: git init - Initializes a new Git repository

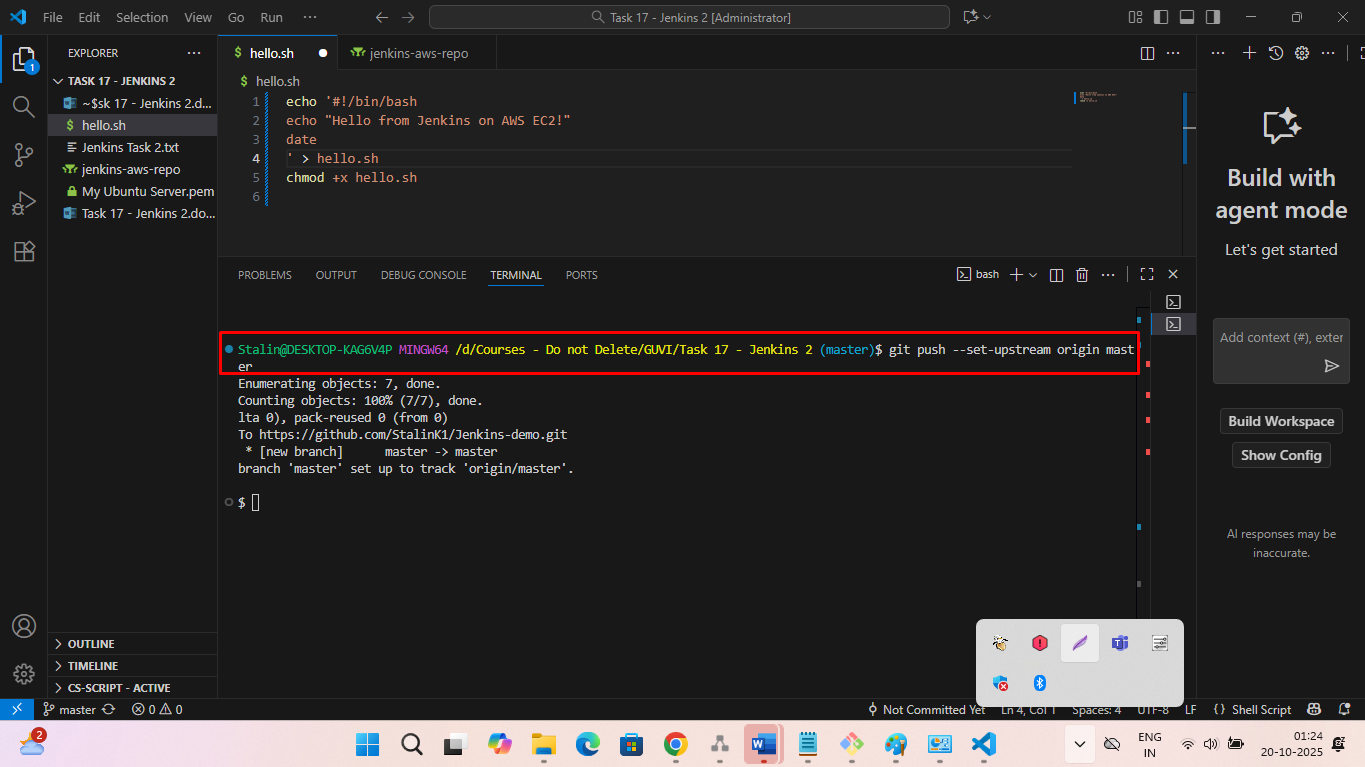
cmd: git add . - Adds **all files** in the current directory

cmd: git commit -m "Initial commit with Jenkinsfile" - Saves the staged changes in Git

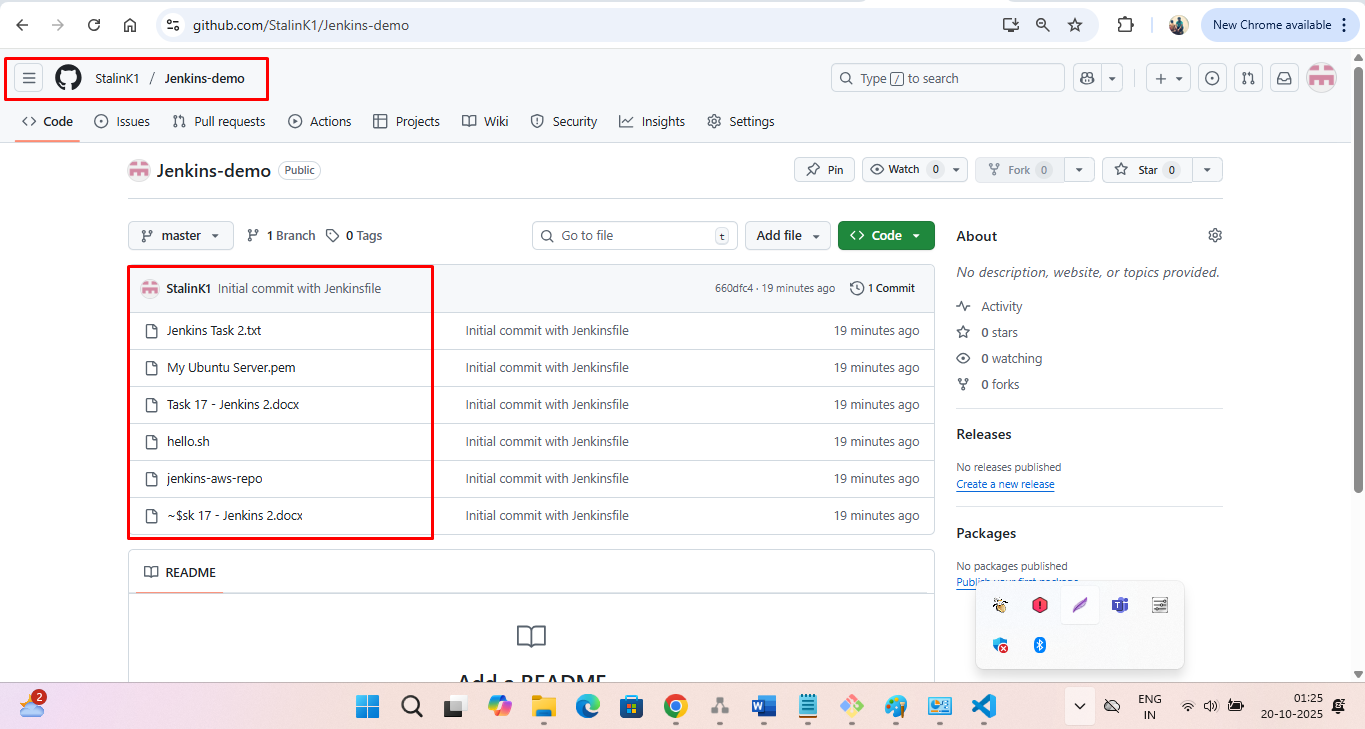
cmd: git remote add origin <https://github.com/StalinK1/Jenkins-demo.git> - Links your local repository to a **remote GitHub repository**.

cmd: git push --set-upstream origin master - Pushes your local master branch to the remote repository (origin) and sets it as the default upstream branch.



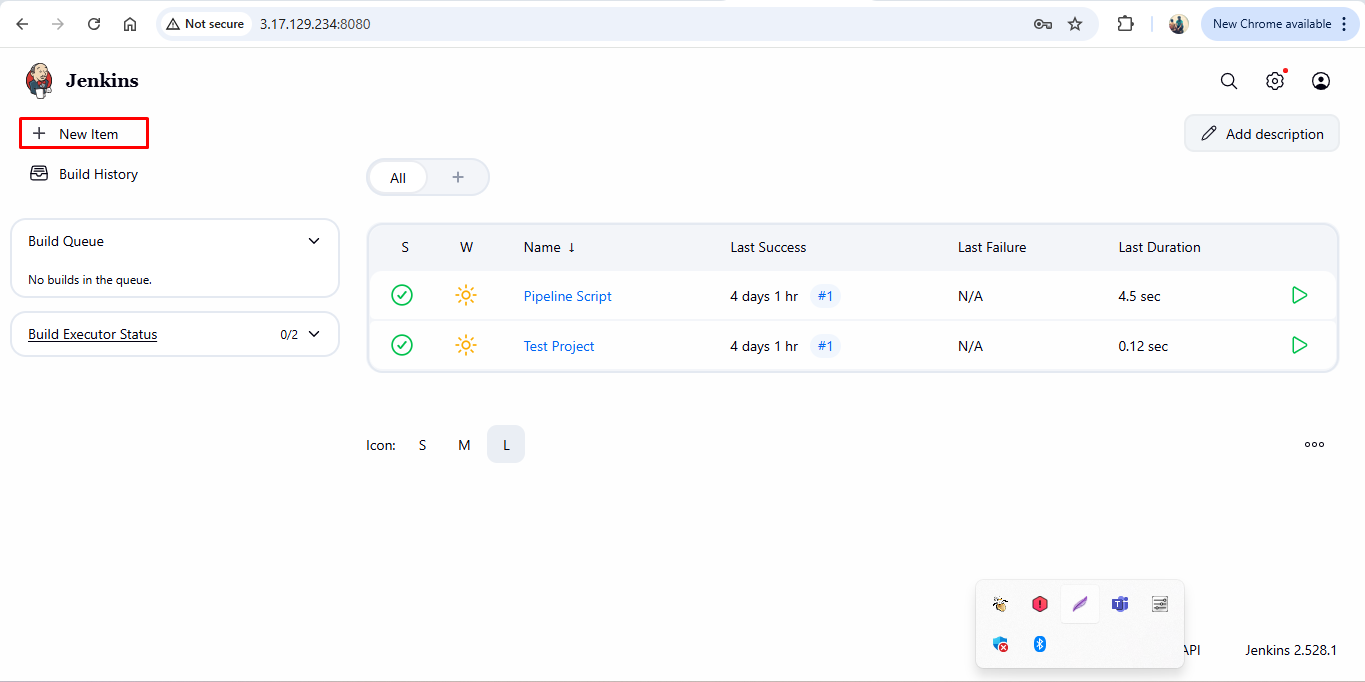


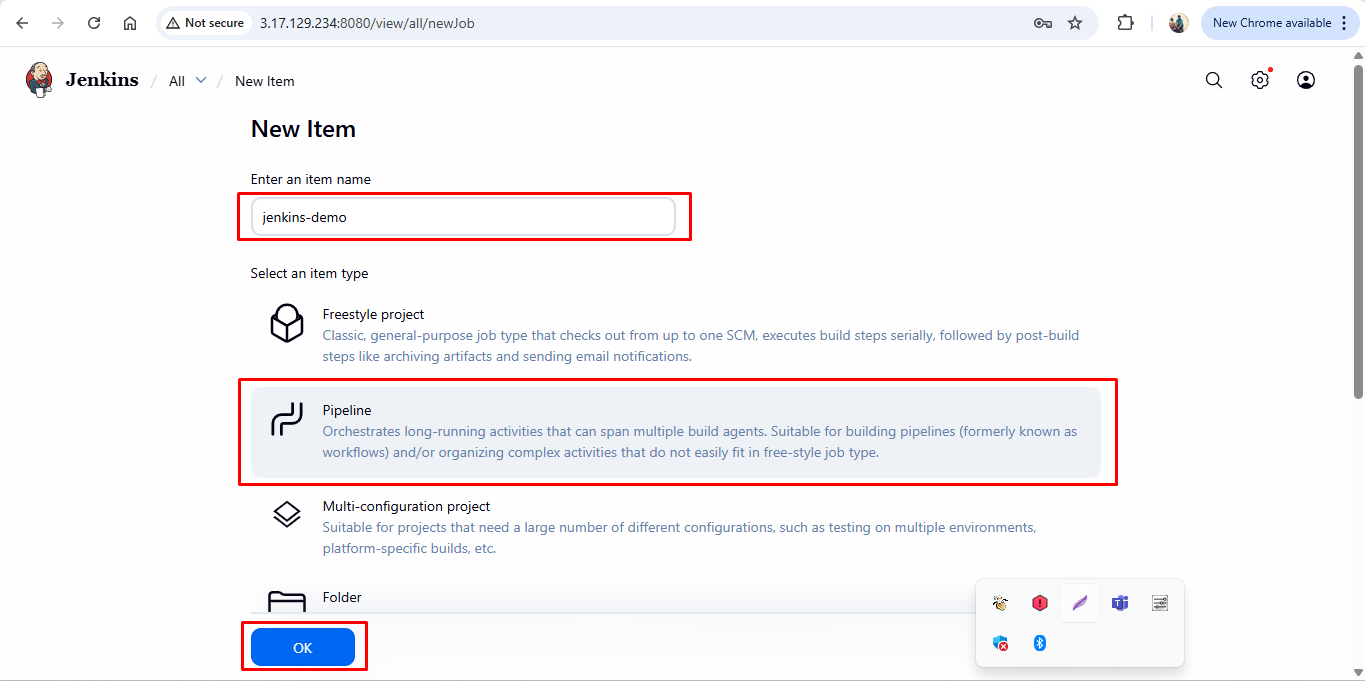
Note : Codes pushed to your Git Hub Repository (I already created a Git Hub repository.)



Step 4: Connect GitHub to Jenkins.

4.1 : In Jenkins Dashboard → New Item → Pipeline and click OK.





4.2:

In Pipeline Definition, choose Pipeline script from SCM

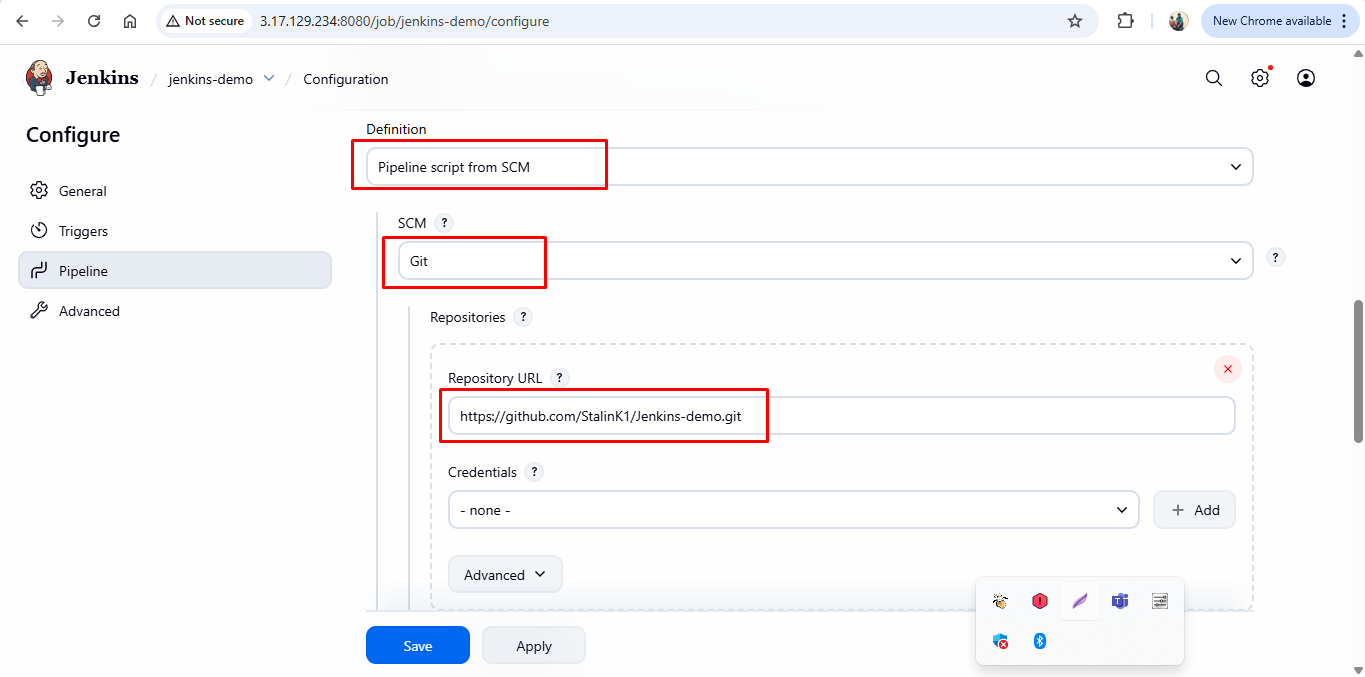
SCM: Git

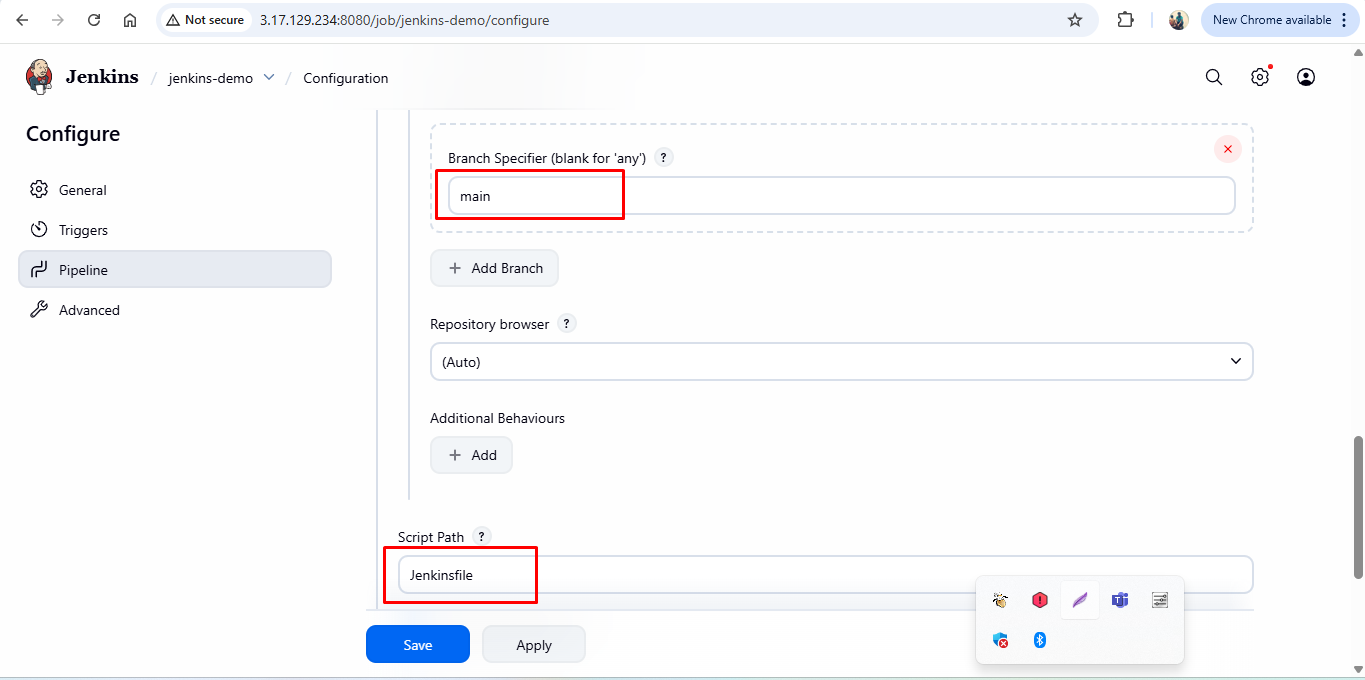
Repo URL: <https://github.com/StalinK1/Jenkins-demo.git>

Branch: \*/master

Script path: Jenkinsfile

Save the build.

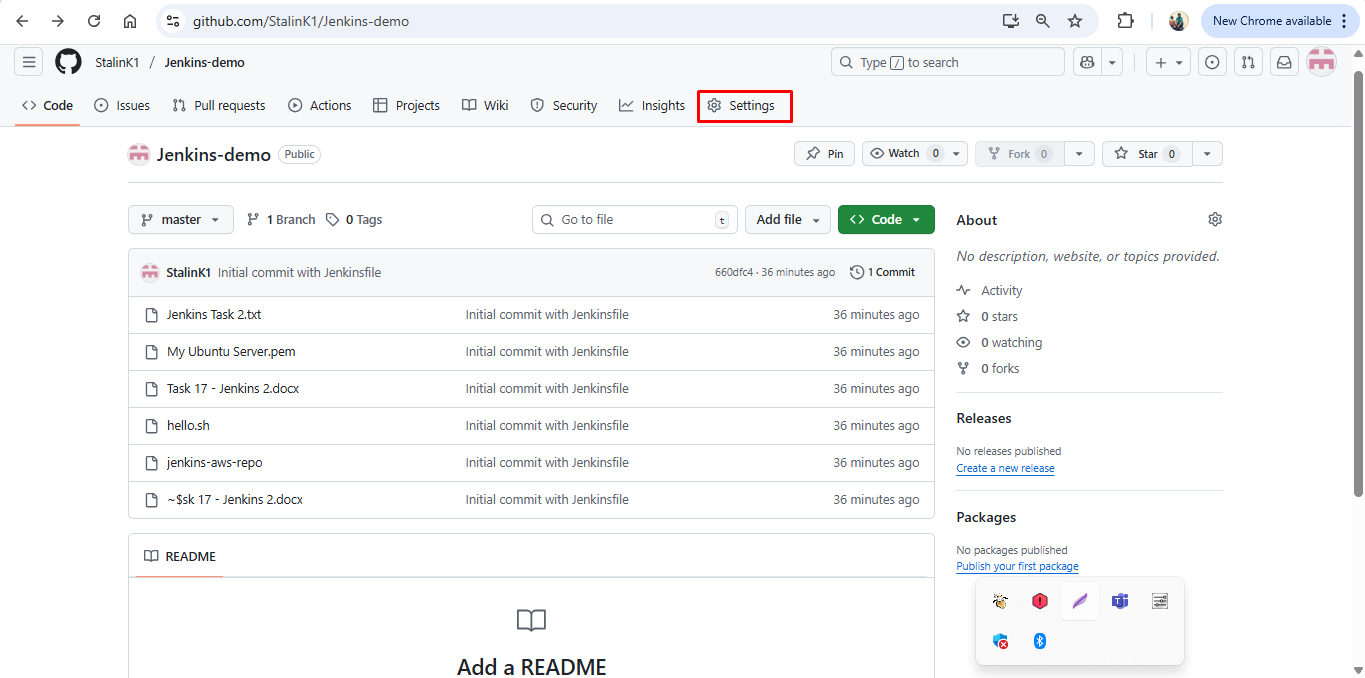


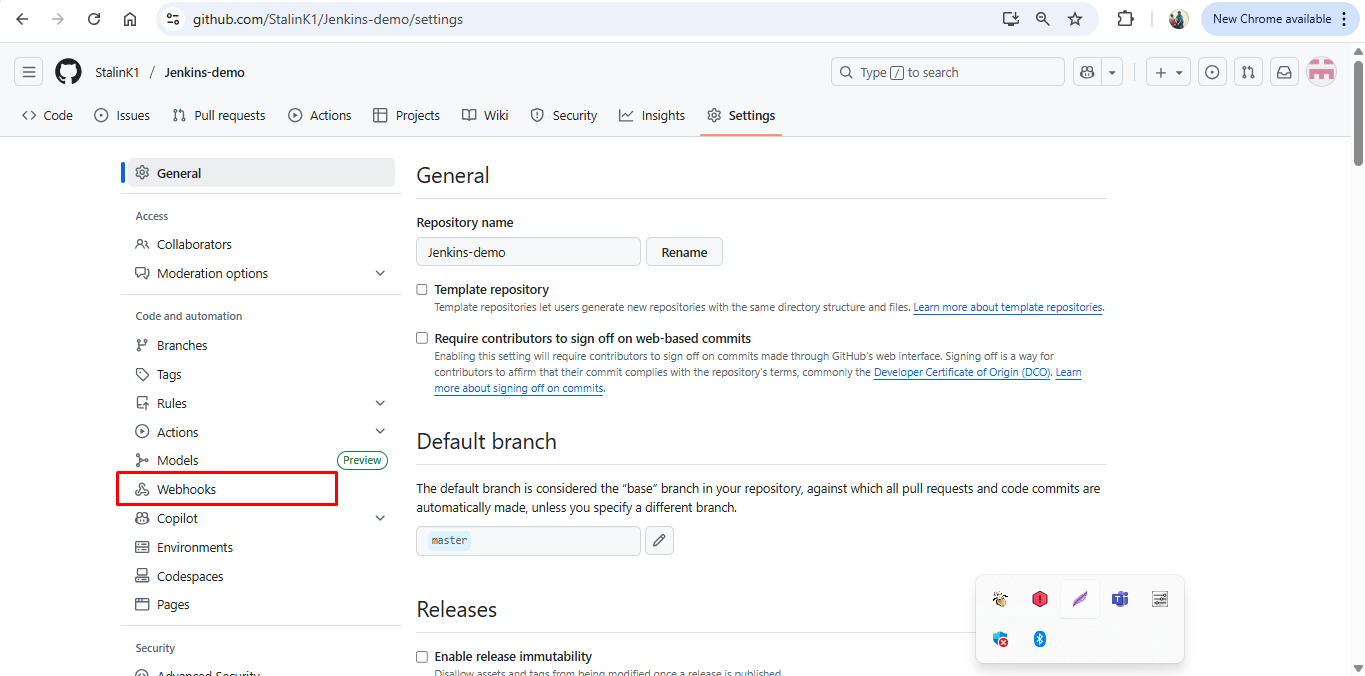


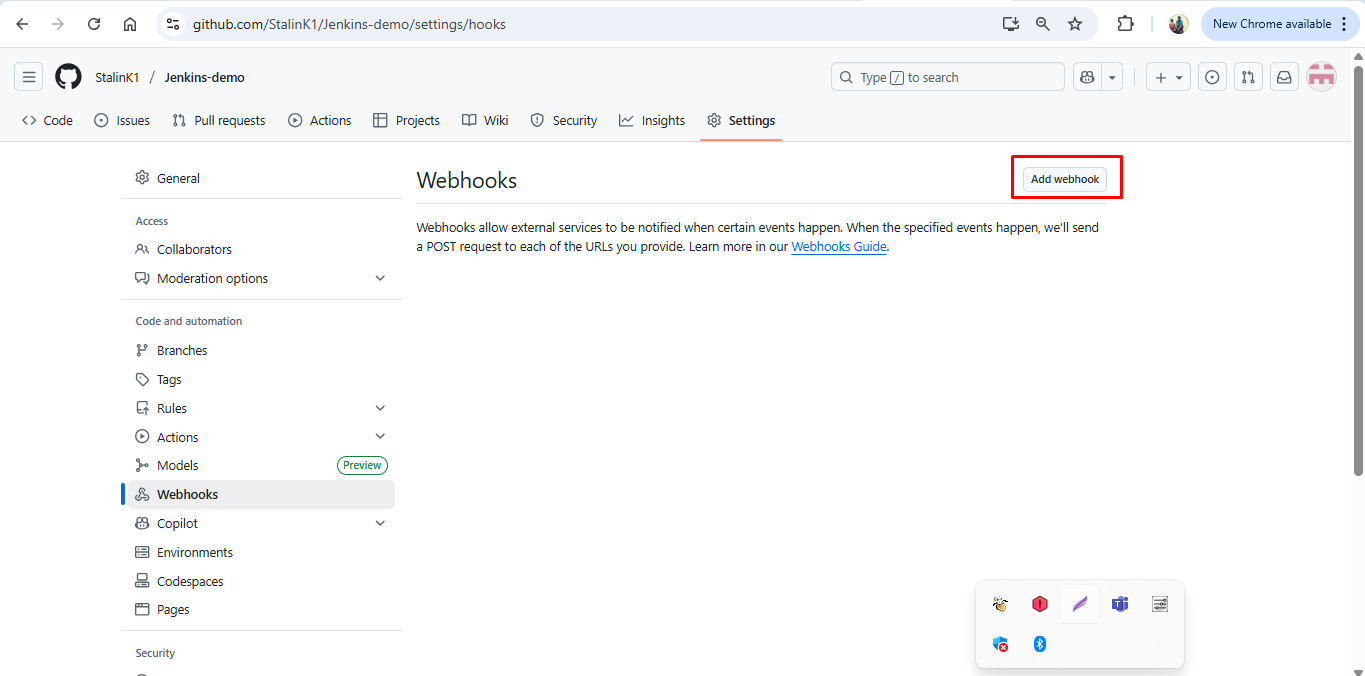
Step 5: Auto Trigger from GitHub

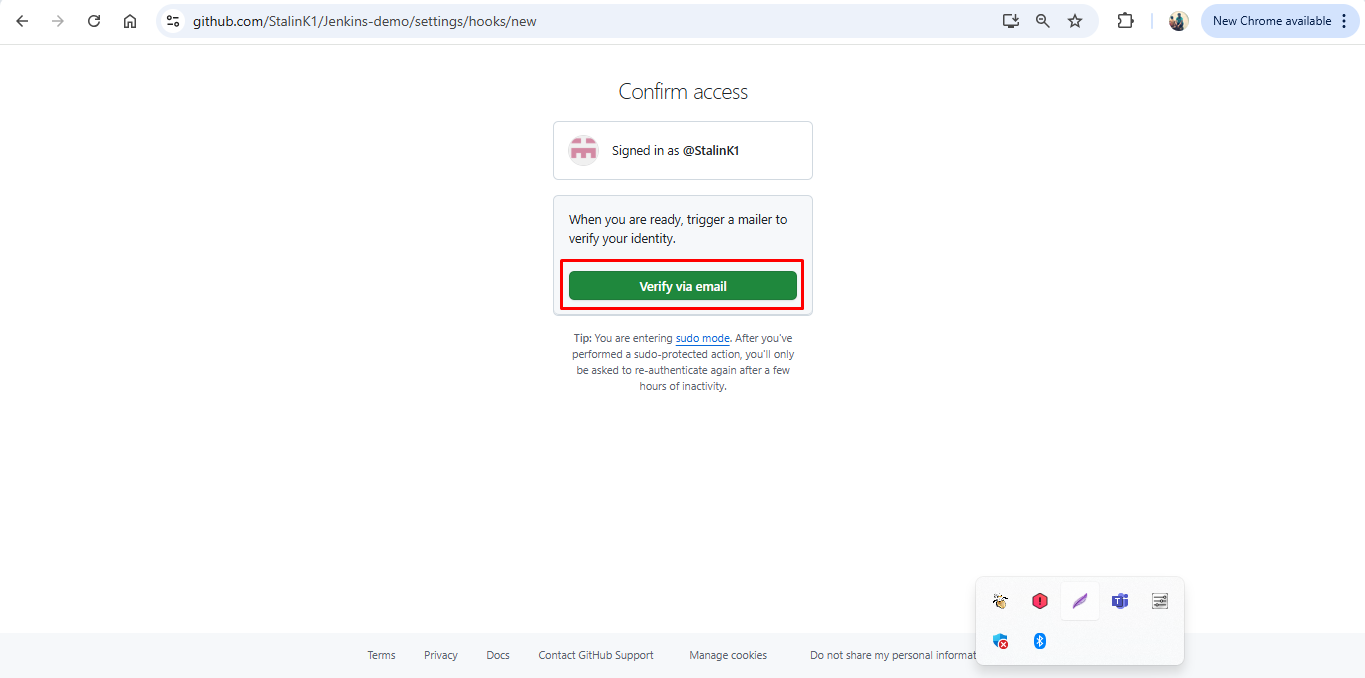
In GitHub Repo → Settings → Webhooks → Add Webhook

You need to verify you e-mail.









Step 5.1:

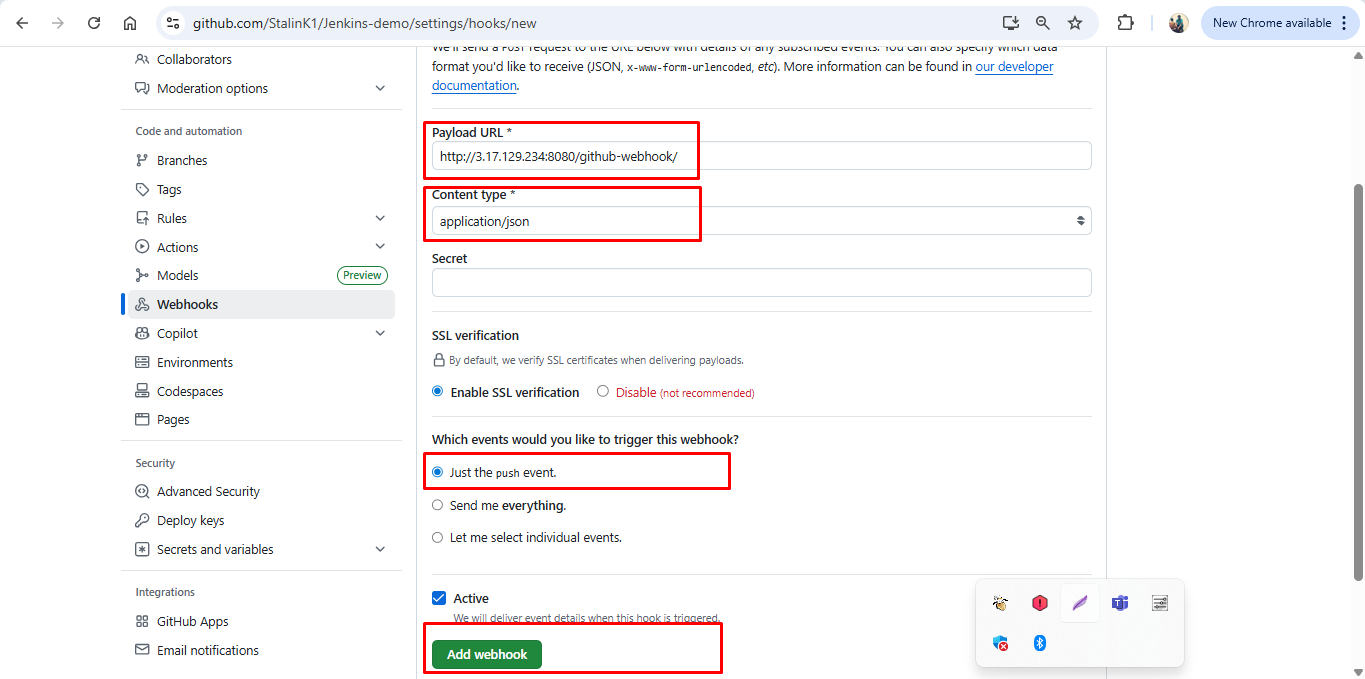
Give the Payload URL: http://<EC2-Public-IP>:8080/github-webhook/

<http://3.17.129.234:8080/github-webhook/>

Content type: application/json

Trigger: Just the push event

Click Add Webhook

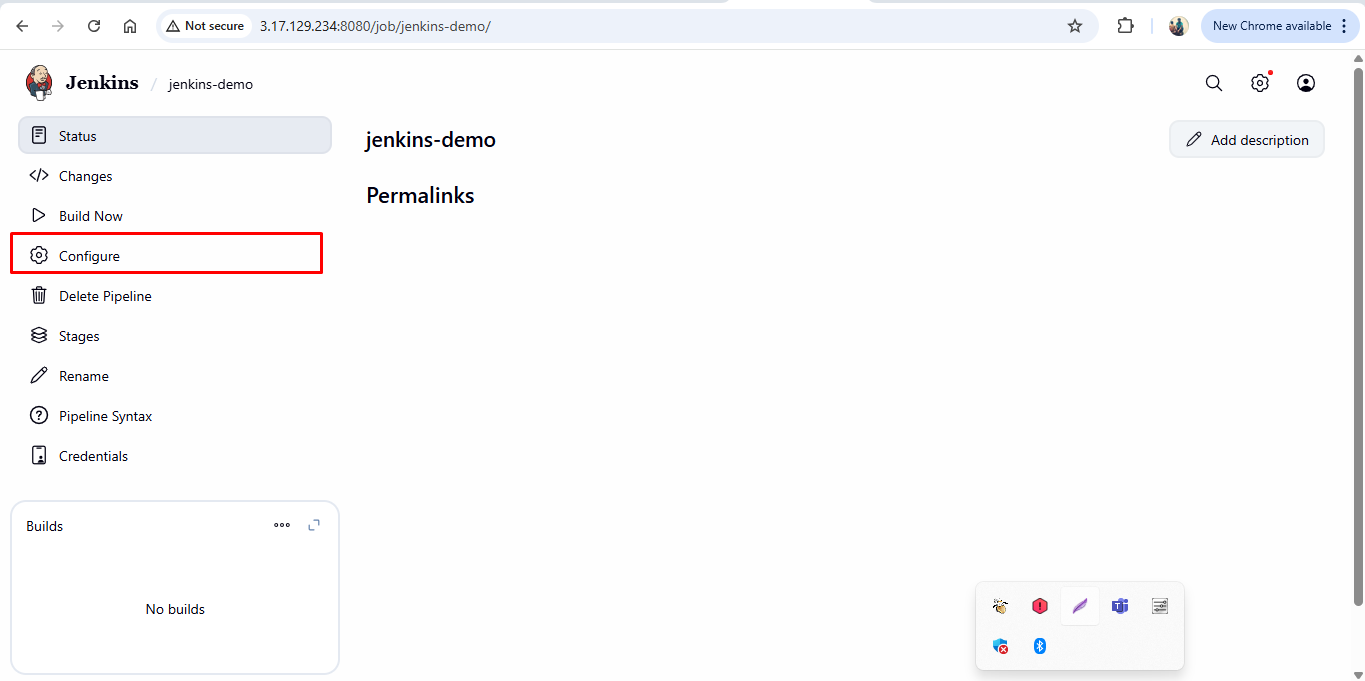


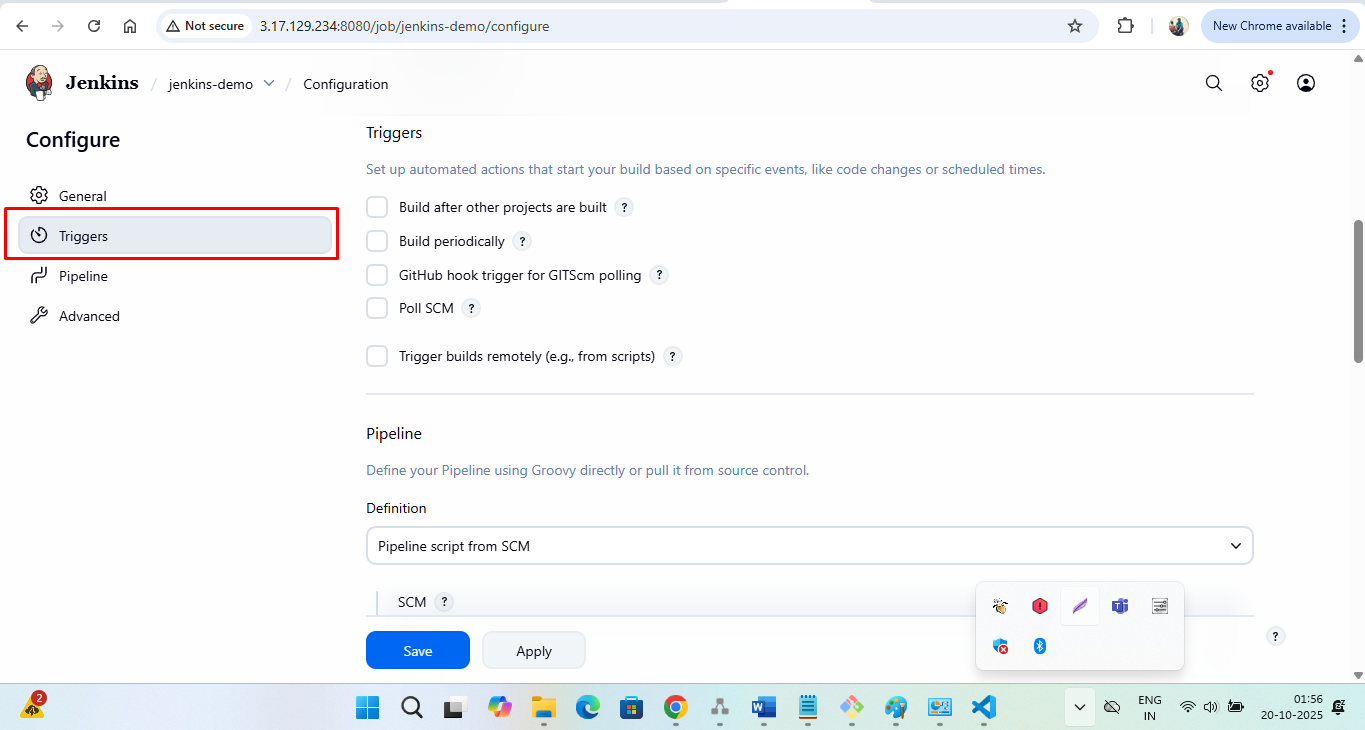
5.2

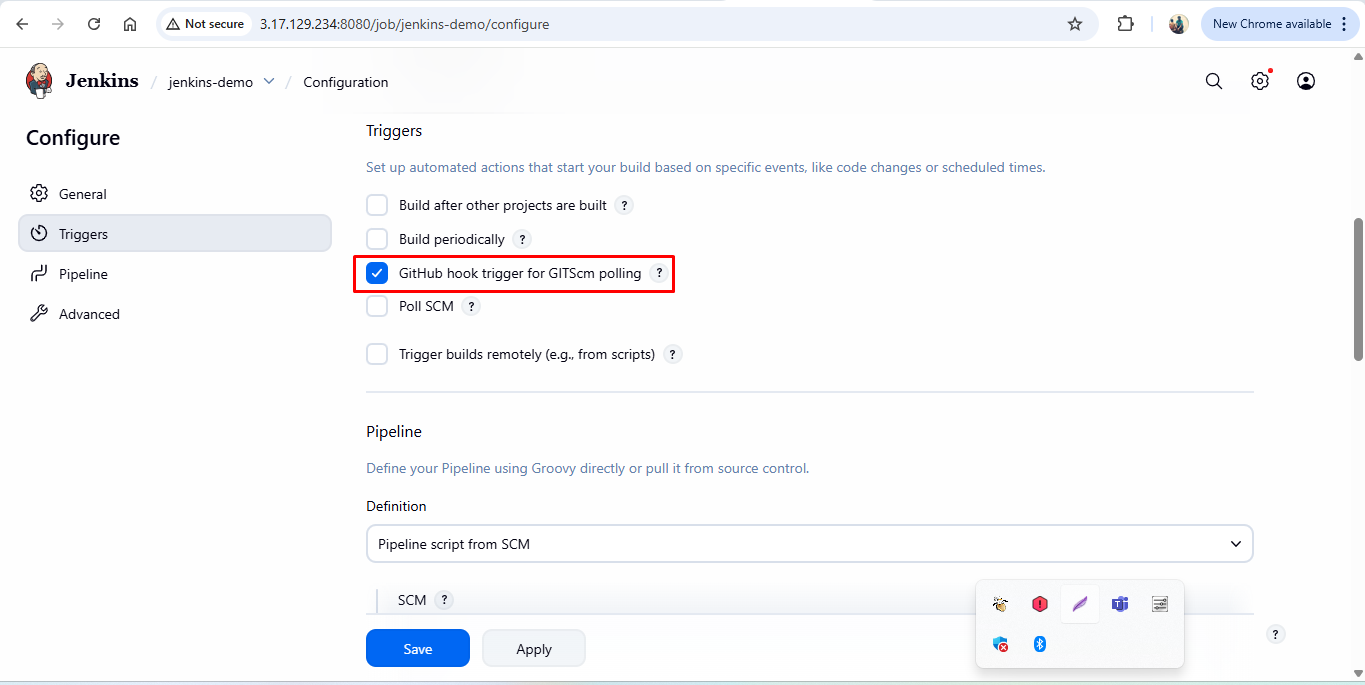
In Jenkins, Go to job → Configure → Build Triggers

Check -> GitHub hook trigger for GITScm polling

Save the configuration.



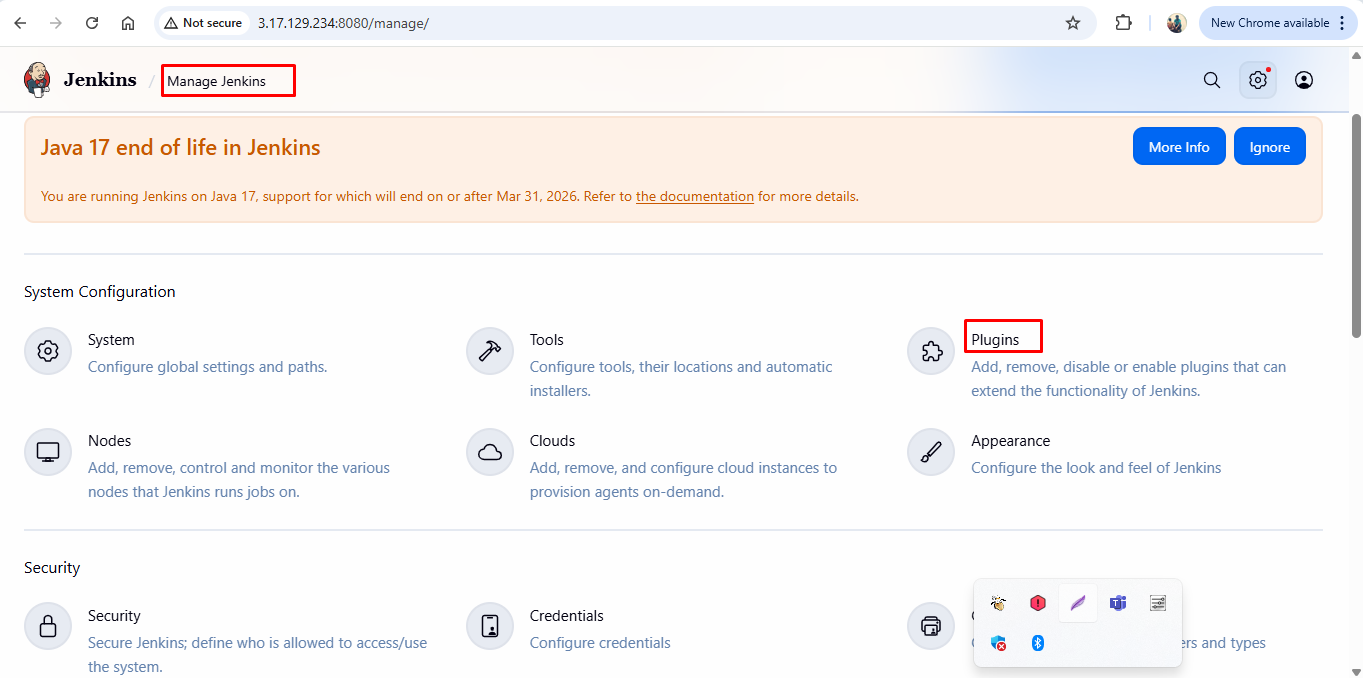


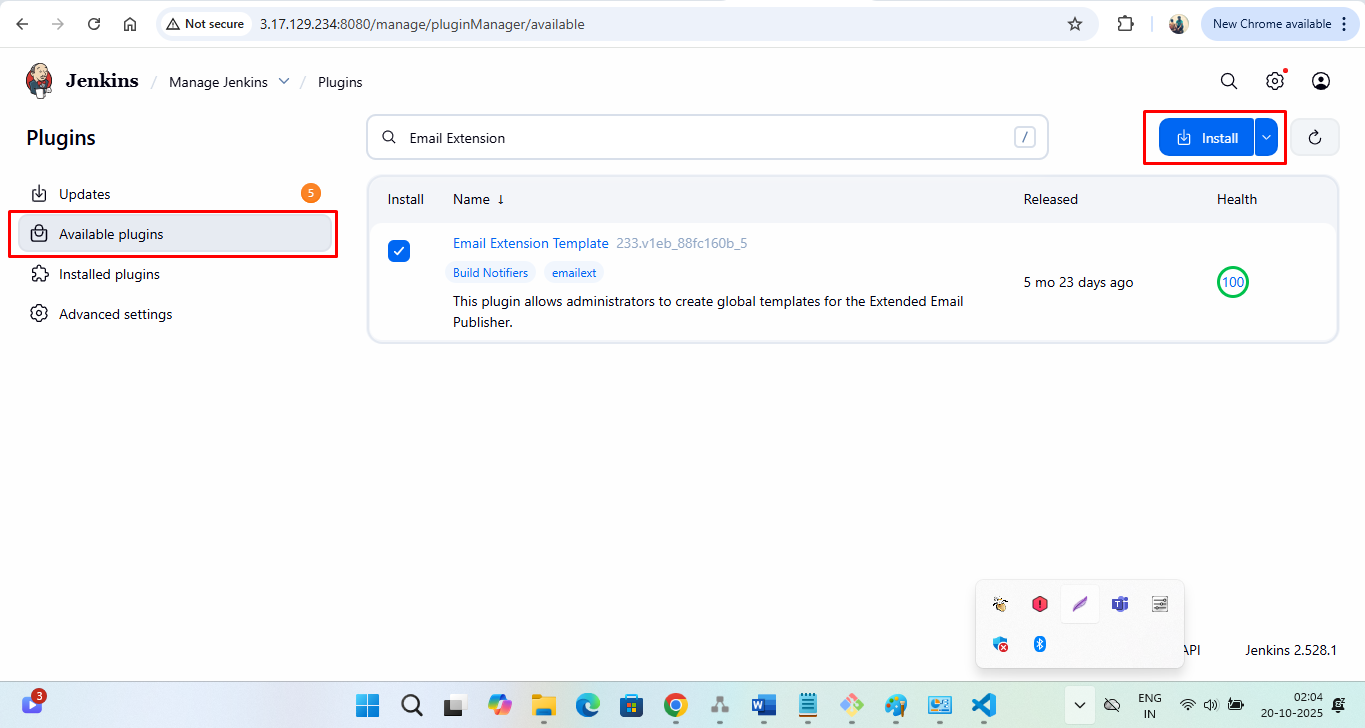


Step 6: Configure Email in Jenkins

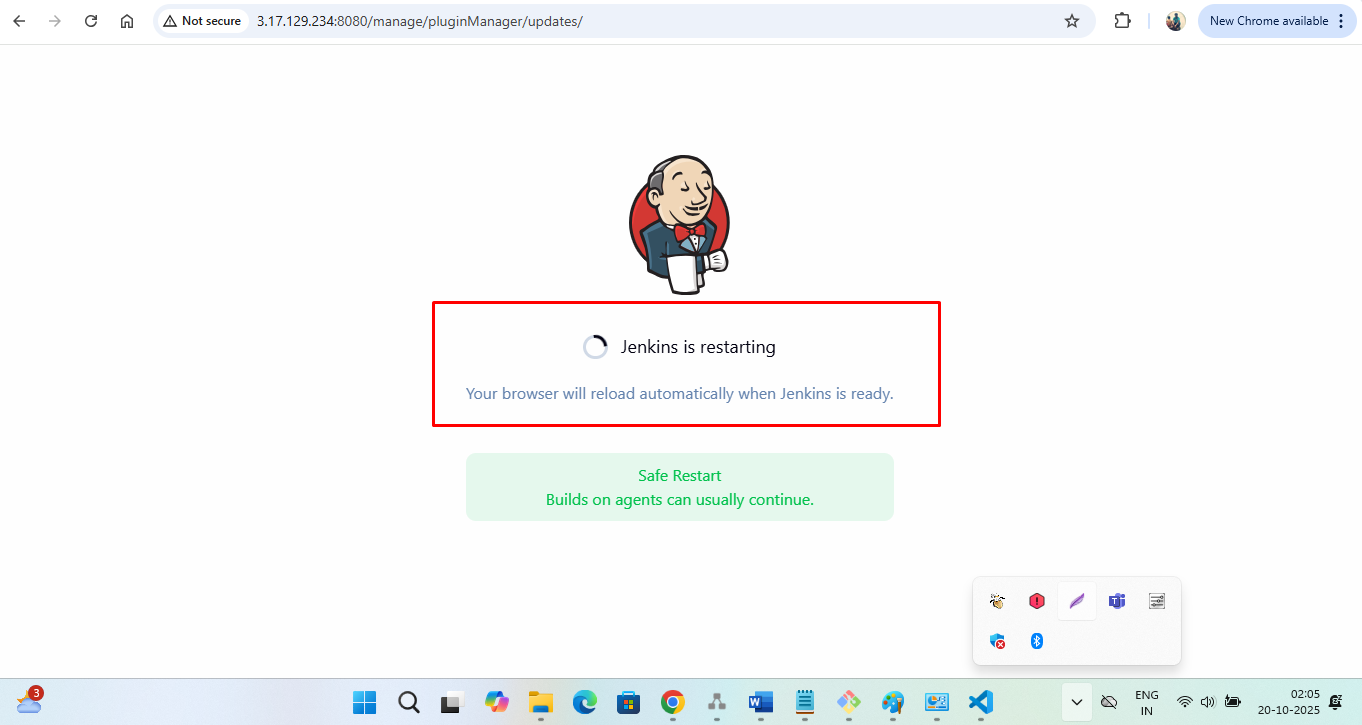
Go to Manage Jenkins → Plugins → Available

Install Email Extension Plugin

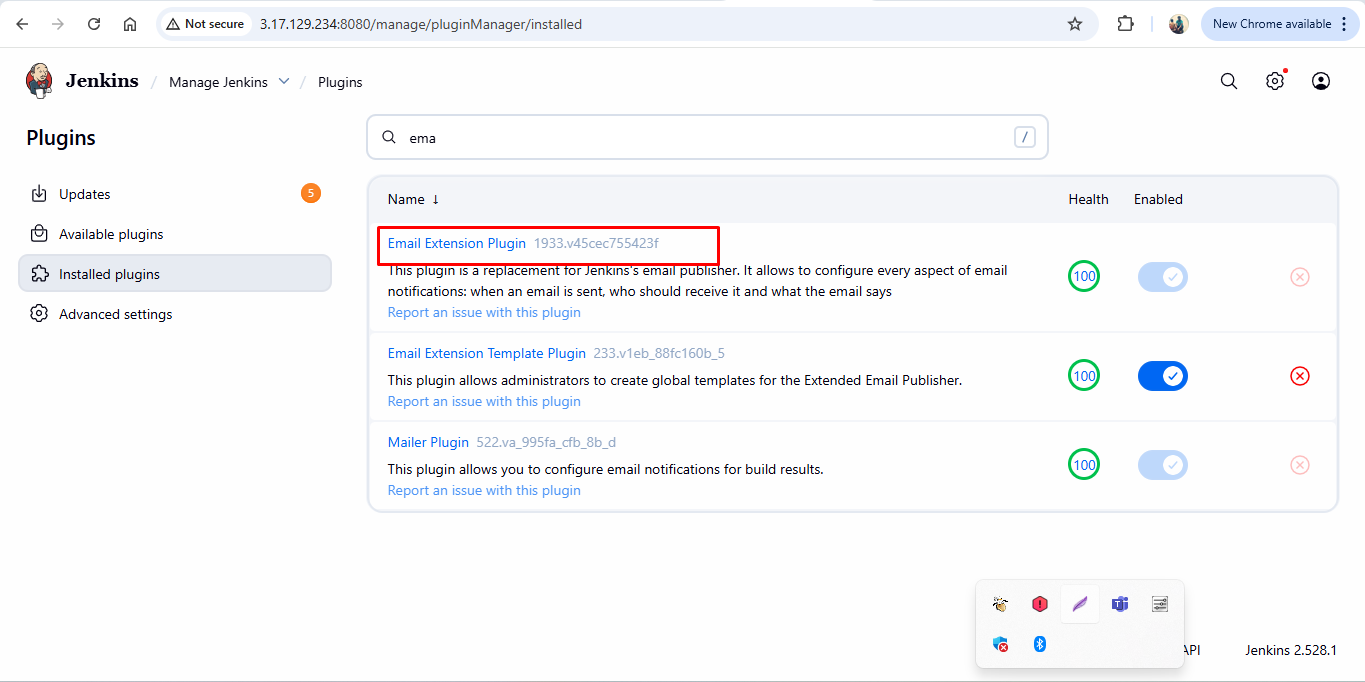




After installing plugins, restart Jenkins.



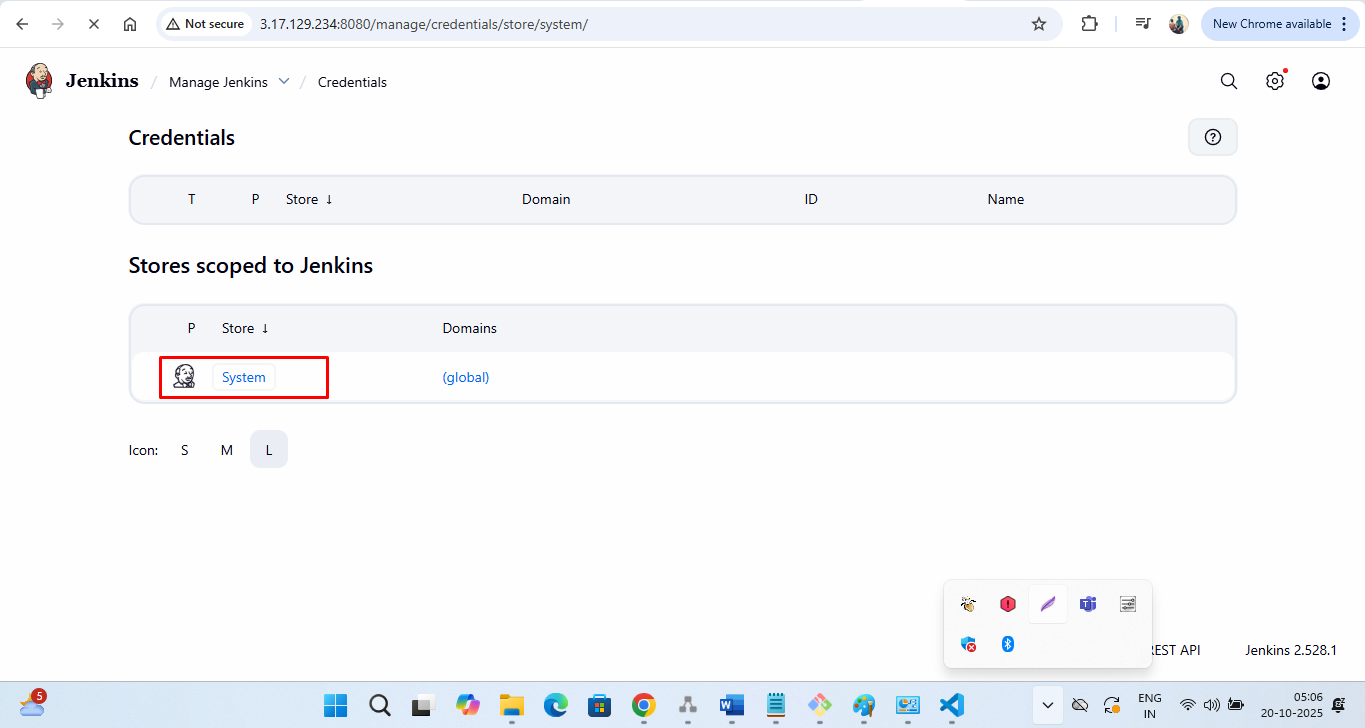
6.1: Verify the installed Plugin.

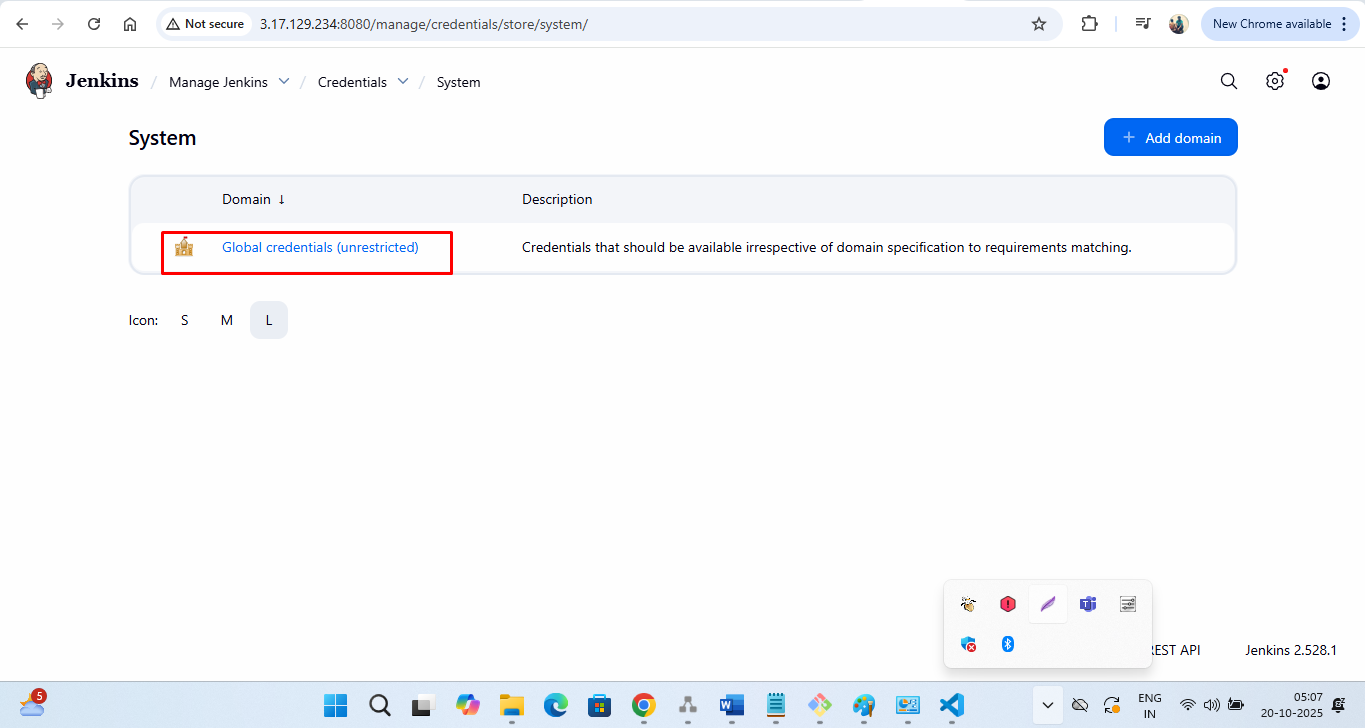


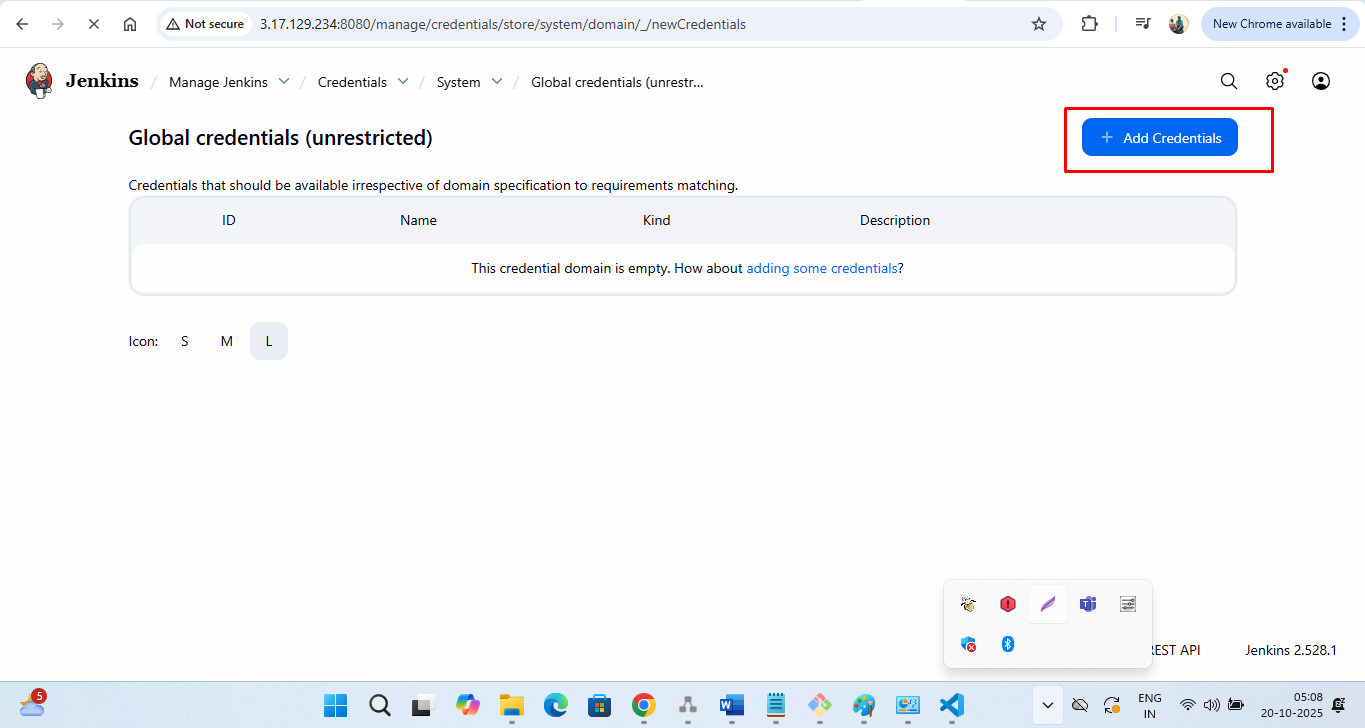
6.2:

Add Gmail credentials in Jenkins

Jenkins Dashboard → Manage Jenkins → Credentials → System → Global credentials → Add Credentials







Kind: Username with password

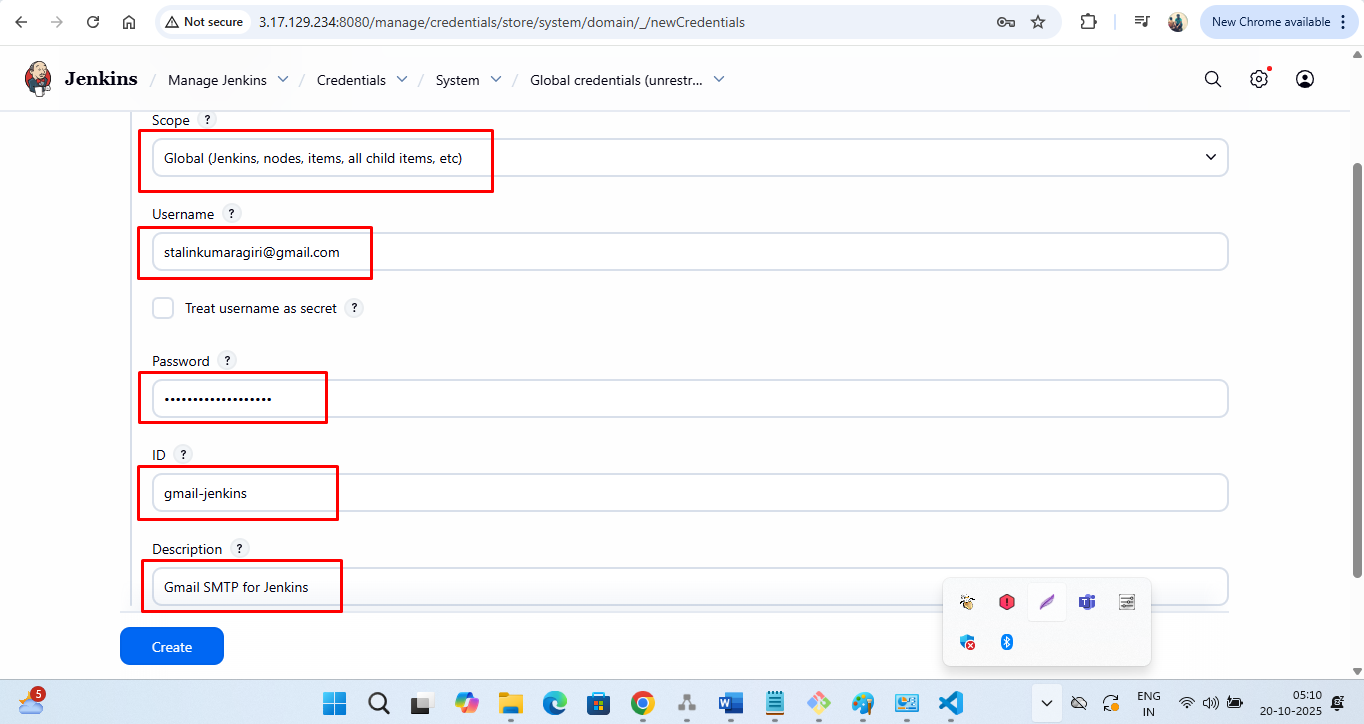
Username: your full Gmail address (your-email@gmail.com)

Password: the App Password you just generated

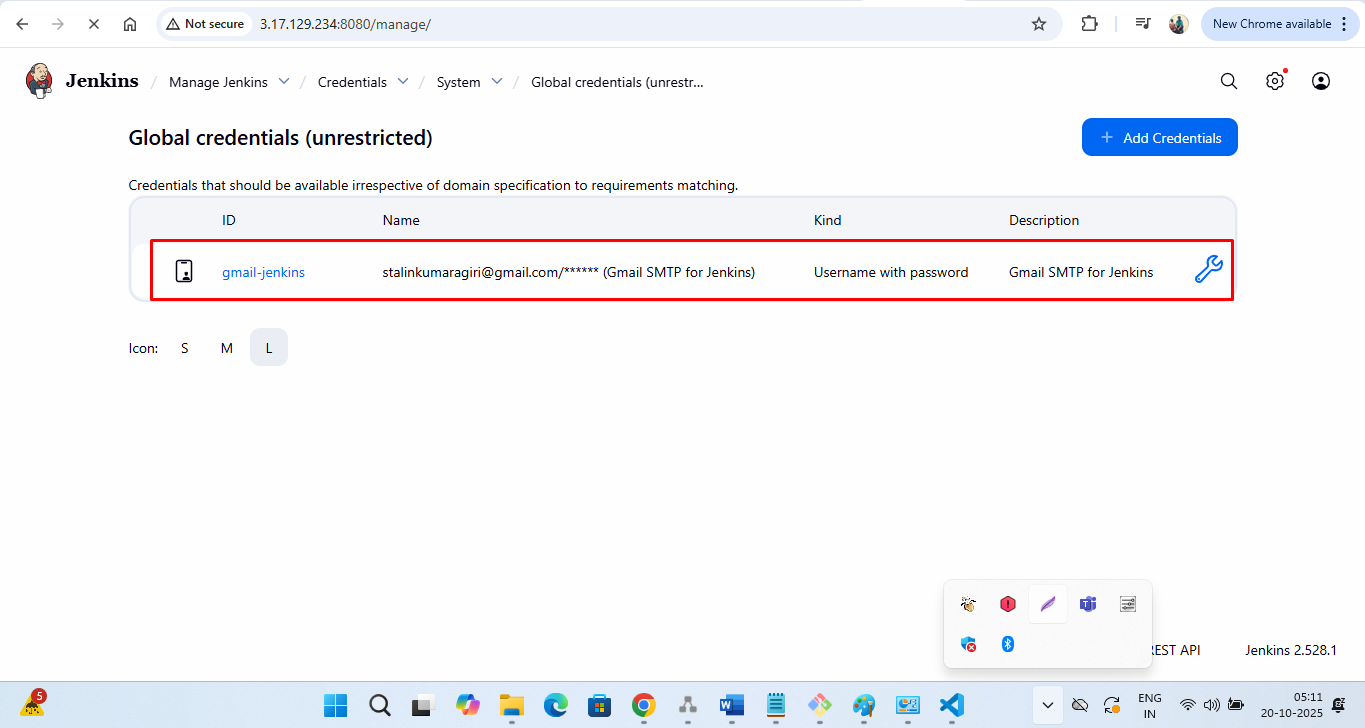
ID: give it a simple ID, e.g., gmail-jenkins

Description: Gmail SMTP for Jenkins

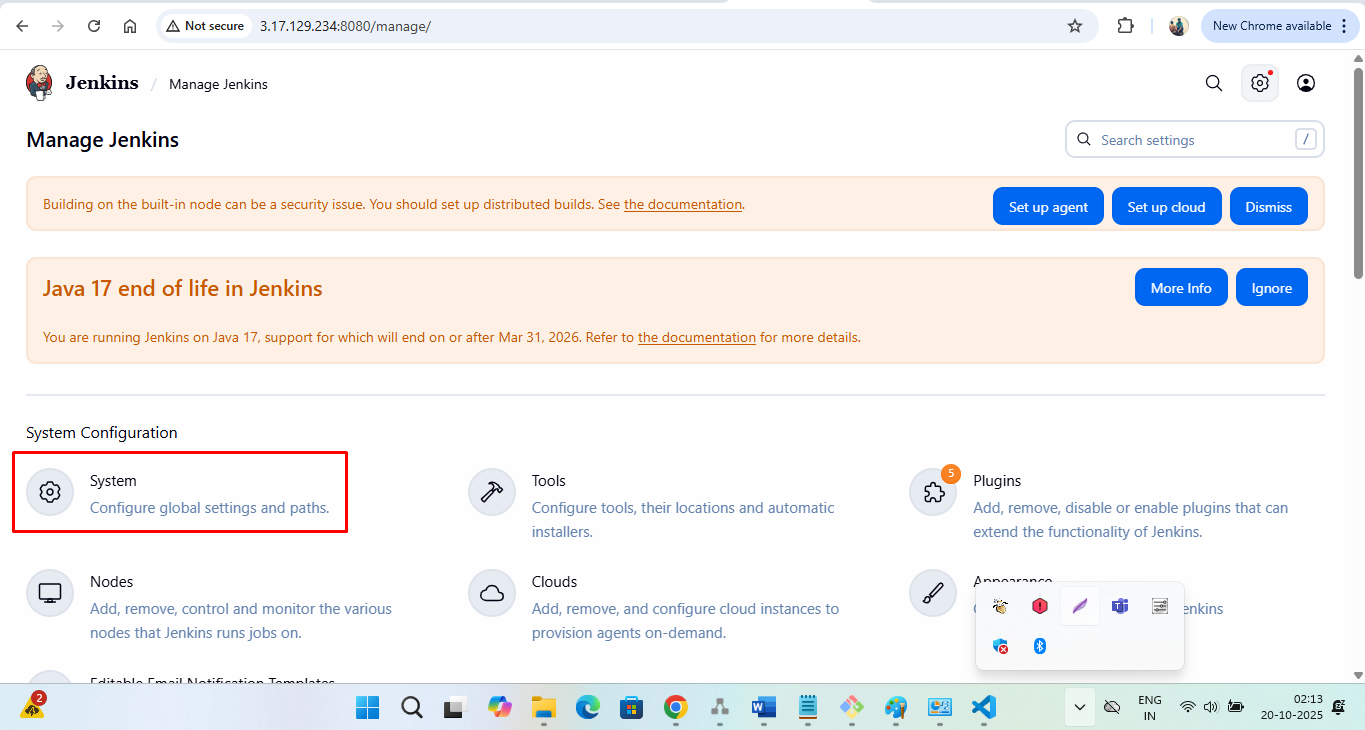
Create



Global Credential has been updated.



Manage Jenkins → System → E-mail Notification



SMTP Server - smtp.gmail.com

Default user e-mail suffix - @gmail.com

Use SMTP Authentication - (checked)

User Name: stalin.kumaragiri@gmail.com

Password: Paste the App Password you generated

Use SSL - (checked)

SMTP Port - 465

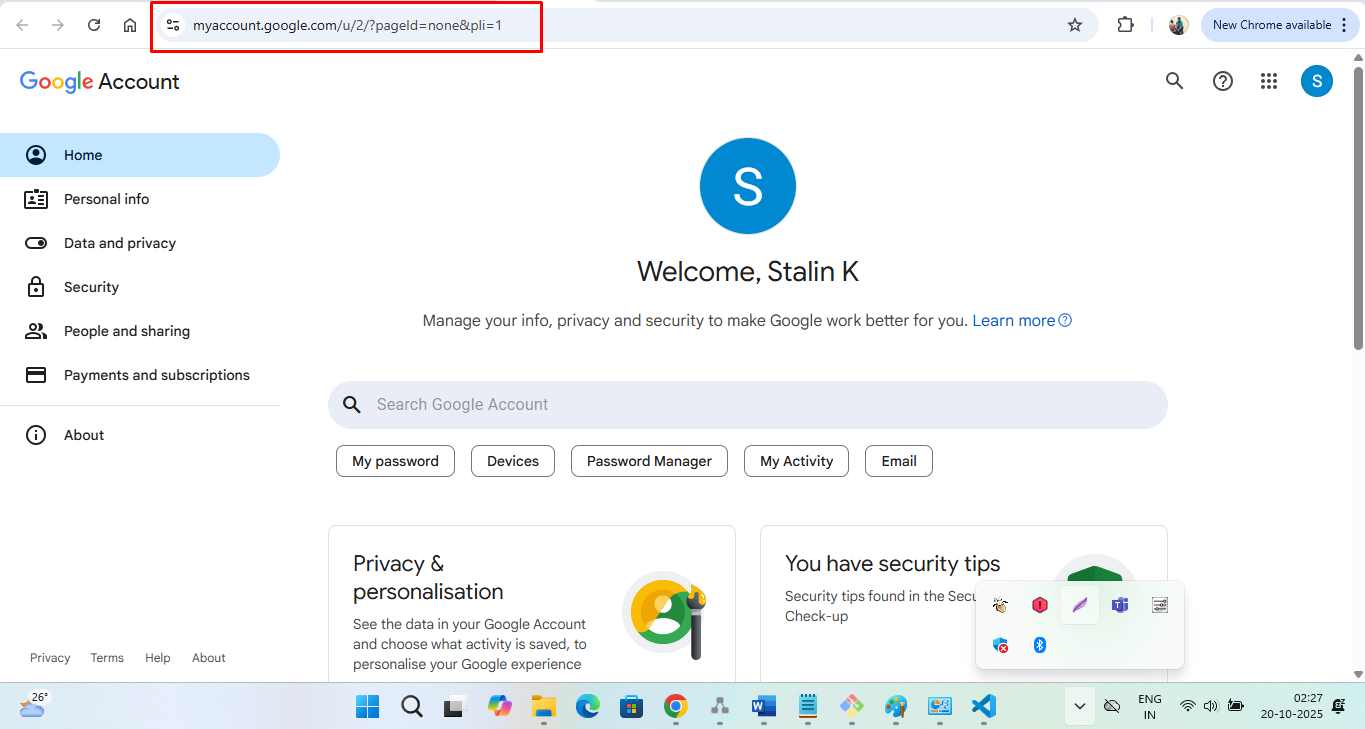
Reply-To Address (optional): same Gmail ID

Charset UTF-8

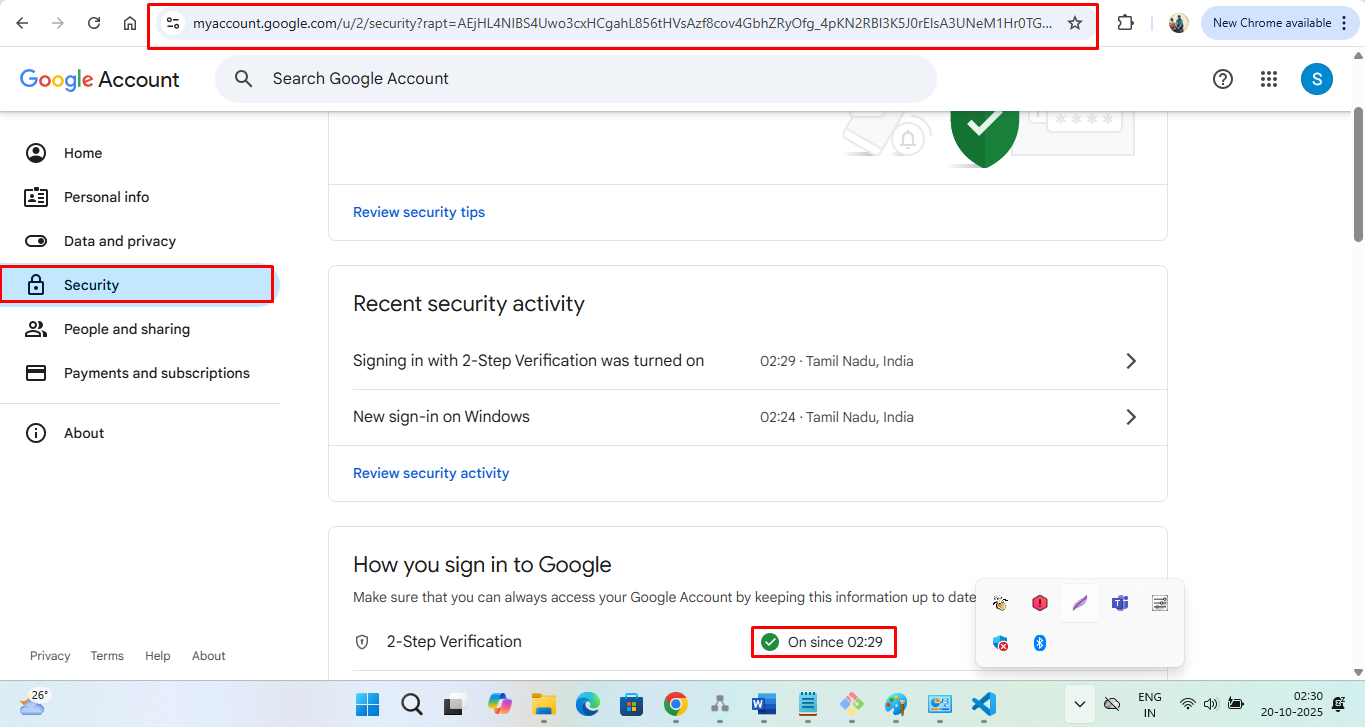
**How to create App Password:**

Goto your account.

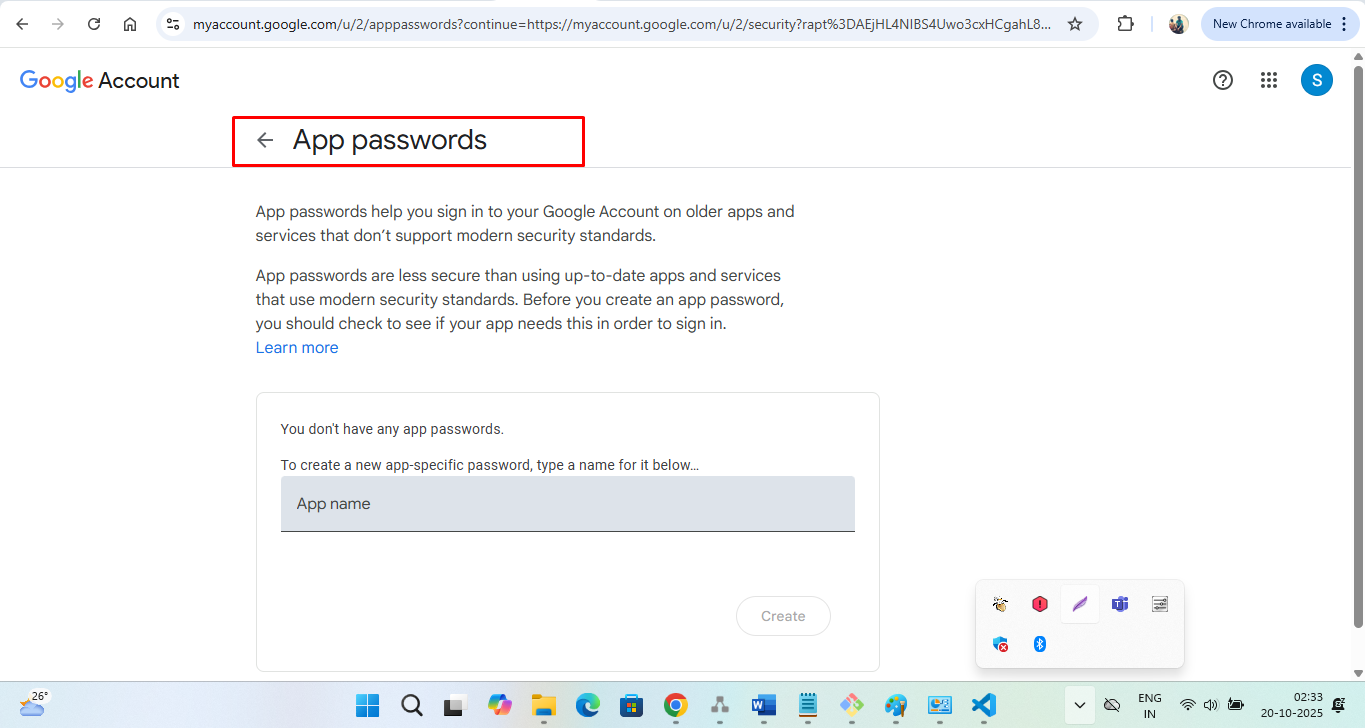
<https://myaccount.google.com/u/2/?pageId=none&pli=1>



Enable **2-Step Verification** (if not already done)

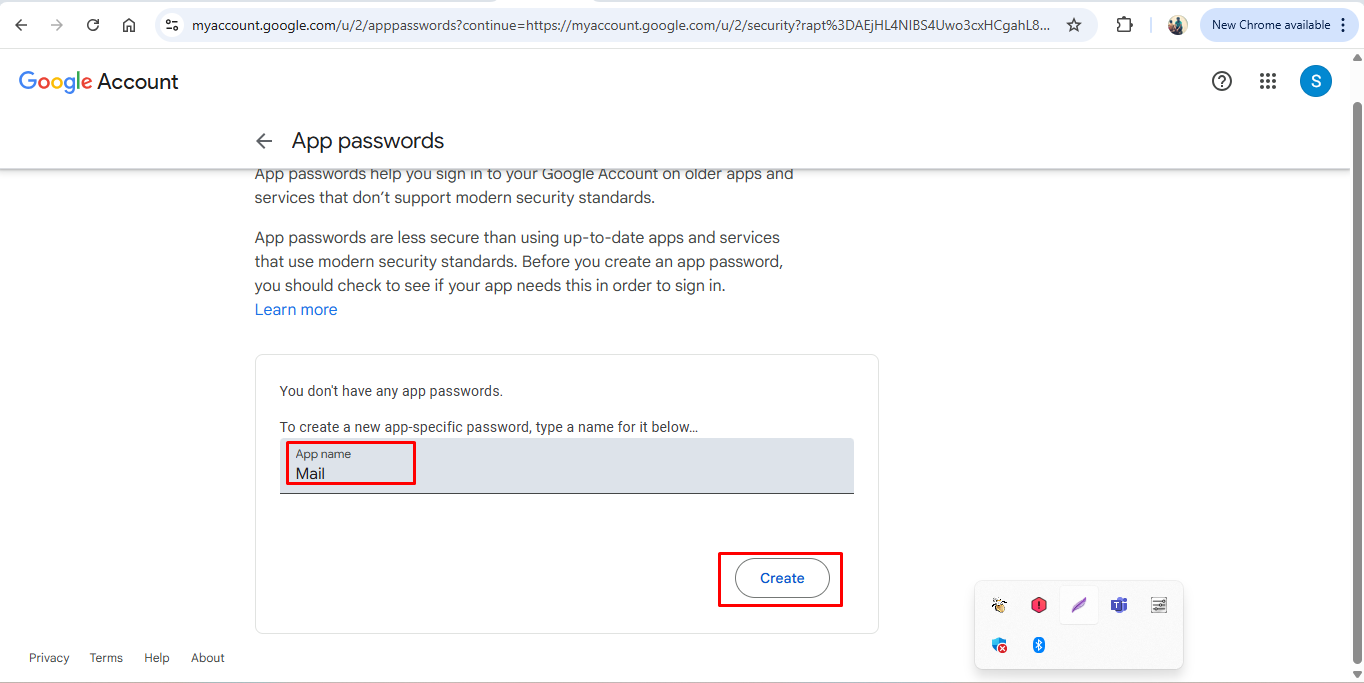


Navigate to:  
Security → App passwords

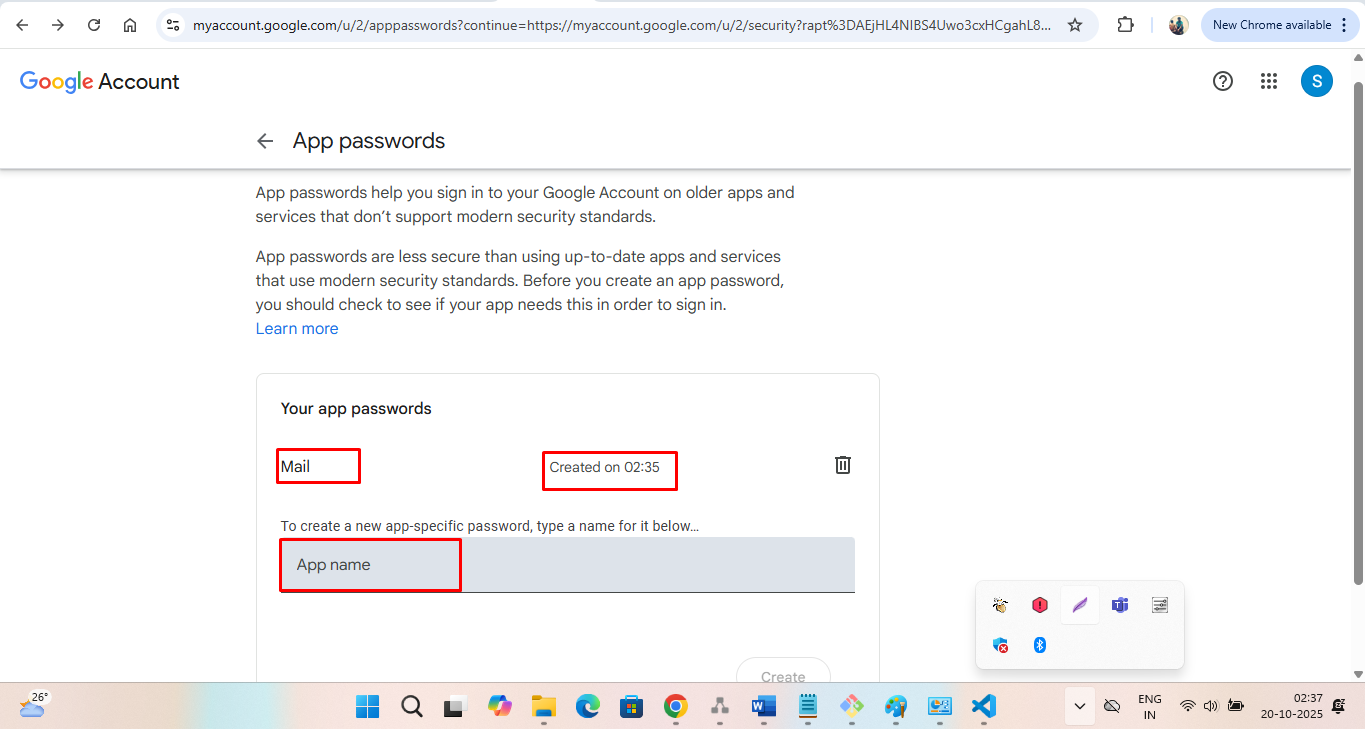


Choose:

App: Mail



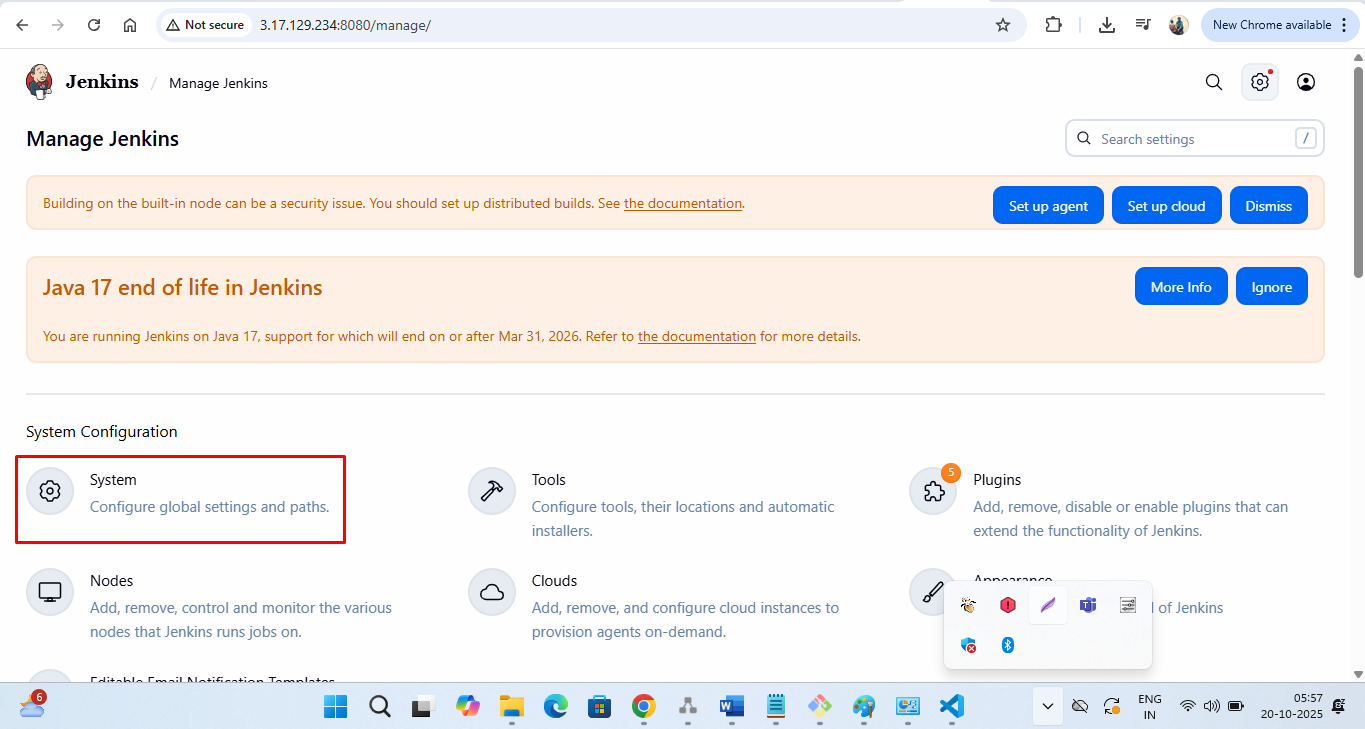
Google generates a 16-character App Password (e.g., abcd efgh ijkl mnop)  
Copy it and save it safely — you’ll use this in Jenkins credentials.



After Generating App Password - Returning to Step 6.2:

Configure Extended Email Notification

1. Jenkins → Manage Jenkins → Configure System → Extended E-mail Notification

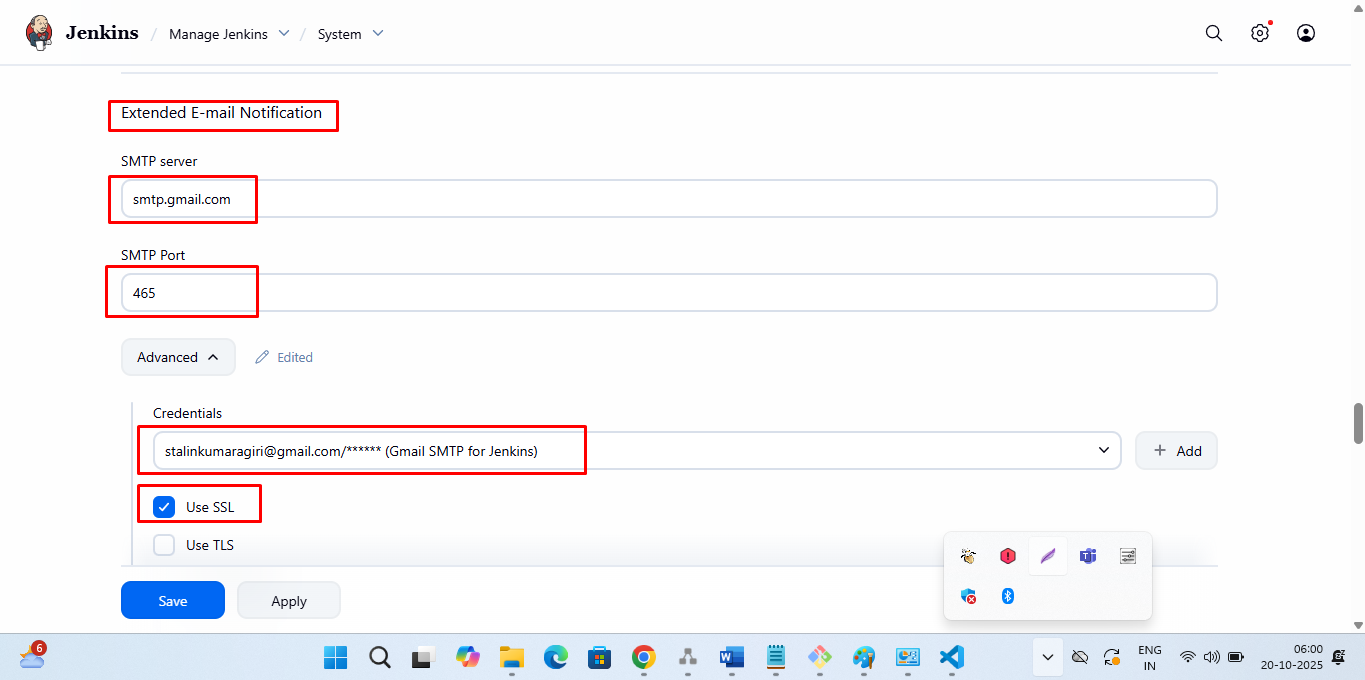


SMTP Server: smtp.gmail.com

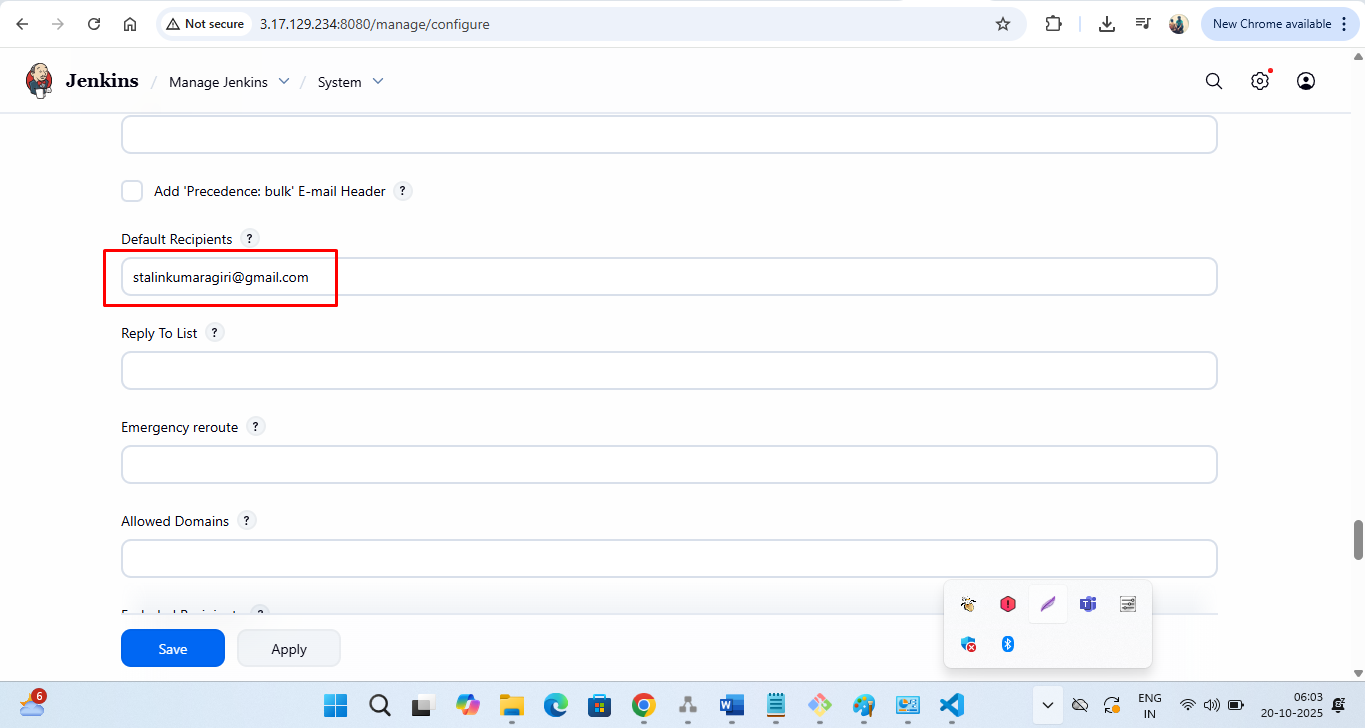
SMTP Port: 465

SSL: Checked

Use “Credentials” dropdown → select the credentials you just created (gmail-jenkins)



Default Recipients: your email address



Save

Step 7: Build the project now and to check whether pipeline has been created or not.

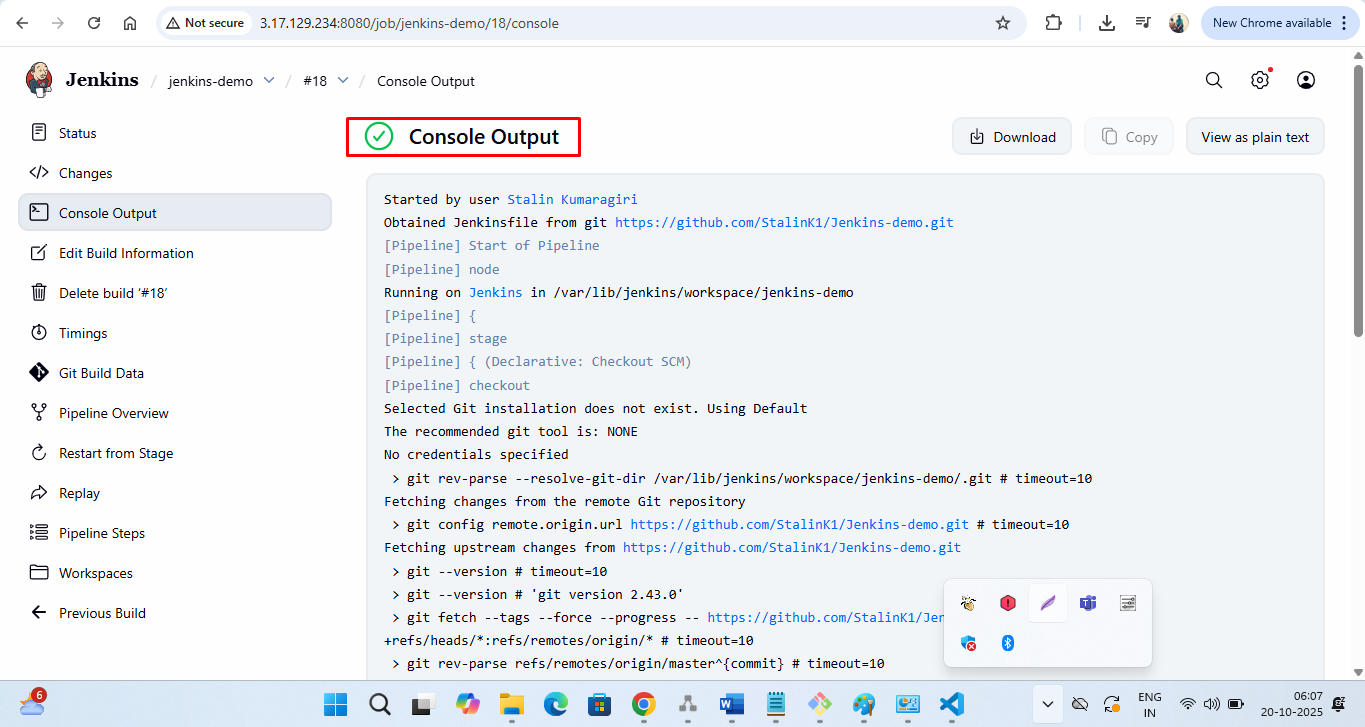
Note: I faced an issue while building the script. I have enabled write permission and pushed the code again using these commands.

git add hello.sh --chmod=+x

git commit -m "Add execute permission to hello.sh"

git push origin master

After the updated the command, build is working fine and we also received the email alert.

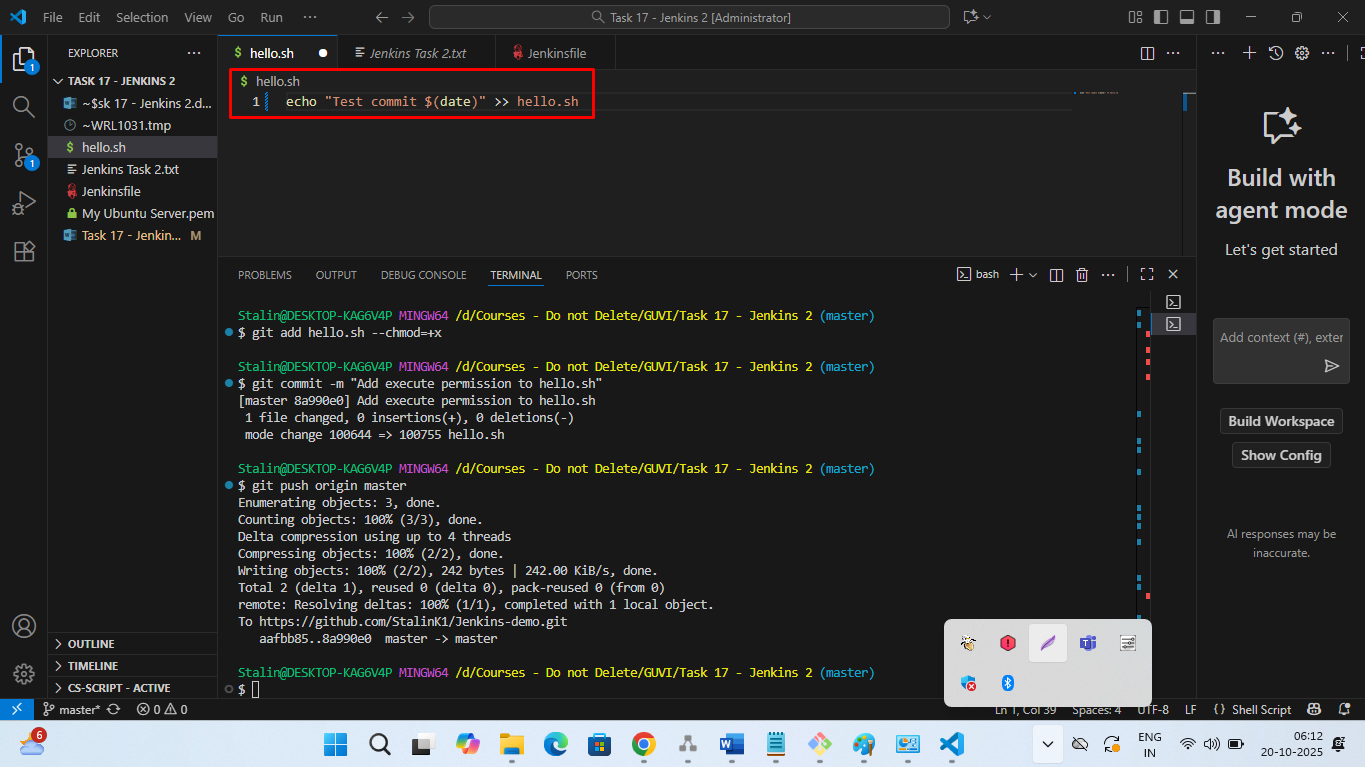






Step 7: Testing deployment.

Make a change in your script:



git add hello.sh

git commit -m "Trigger build"

git push

