**Jenkins CI/CD Pipeline for Java Application Deployment**

**📘 Project Overview:**

This project demonstrates how to build and deploy a Java web application using a fully automated CI/CD pipeline. It integrates several DevOps tools to ensure continuous integration, automated testing, containerization, and smooth deployment. **Git Hub:** [**Link**](https://github.com/Naveezzz/java-maven-app-jenkins)

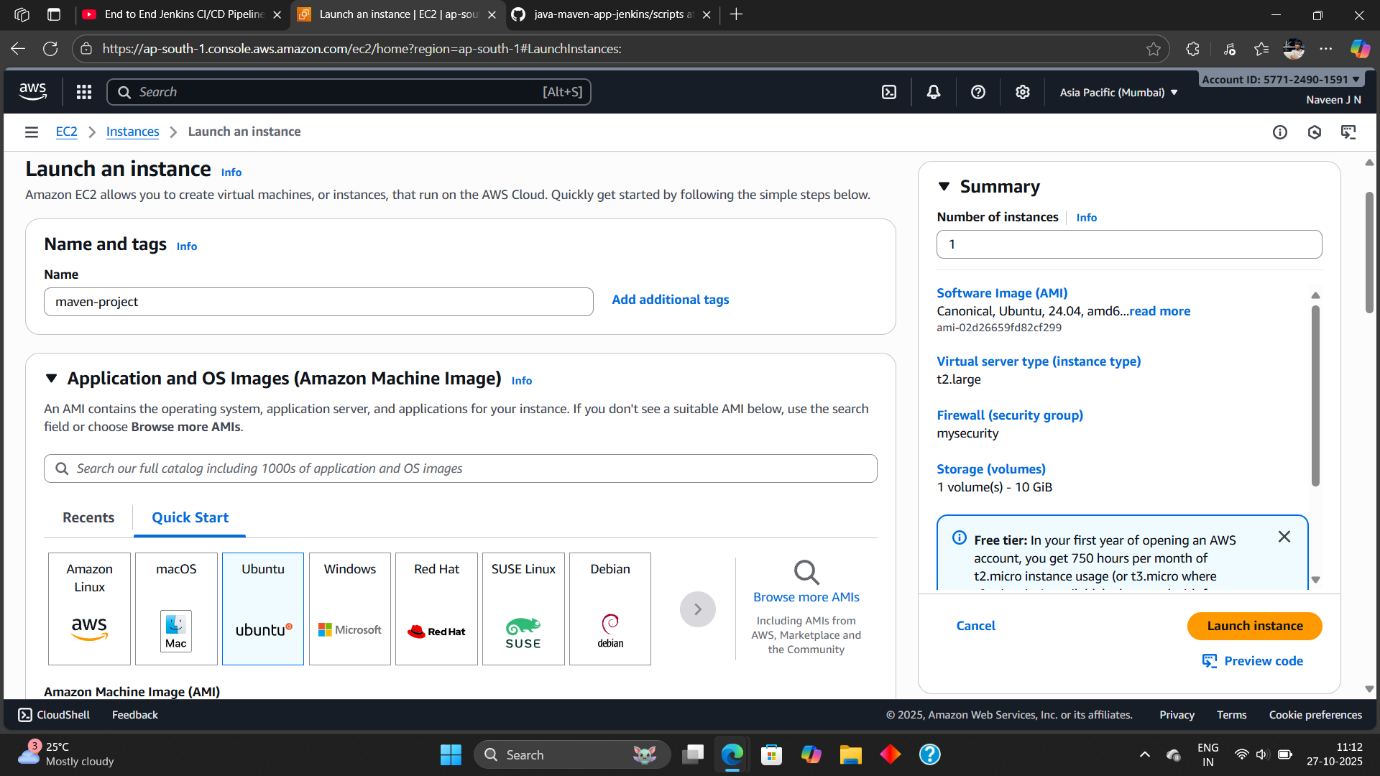
**Tools Used:**

* **Maven** — For building and packaging the Java web application (WAR file).
* **GitHub** — For source code management and version control.
* **Jenkins** — For automating the CI/CD pipeline (build, test, and deploy).
* **Docker** — For containerizing the application and pushing the image to Docker Hub.
* **Tomcat Server** — For hosting and running the deployed web application.

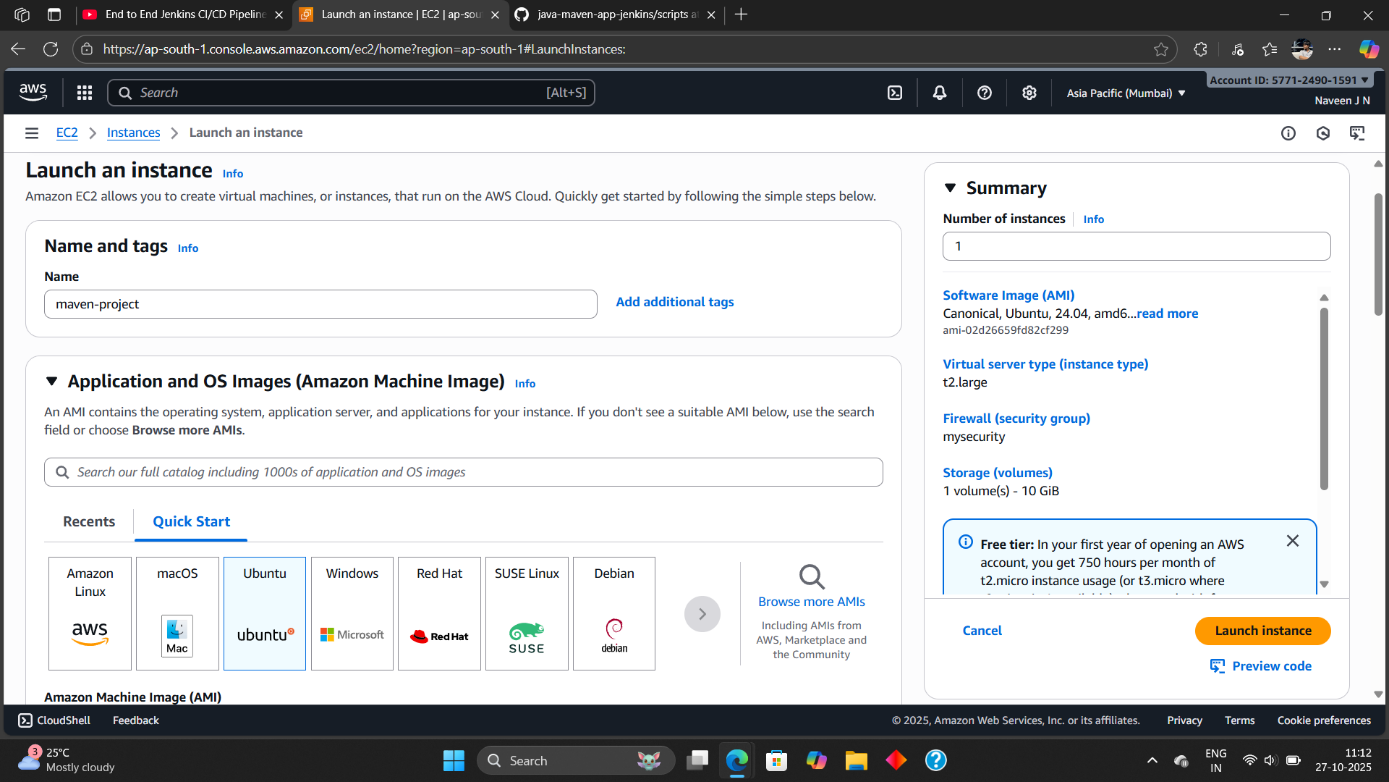
**⚙️ Setup Steps**

**Step 1: Create an Instance:**

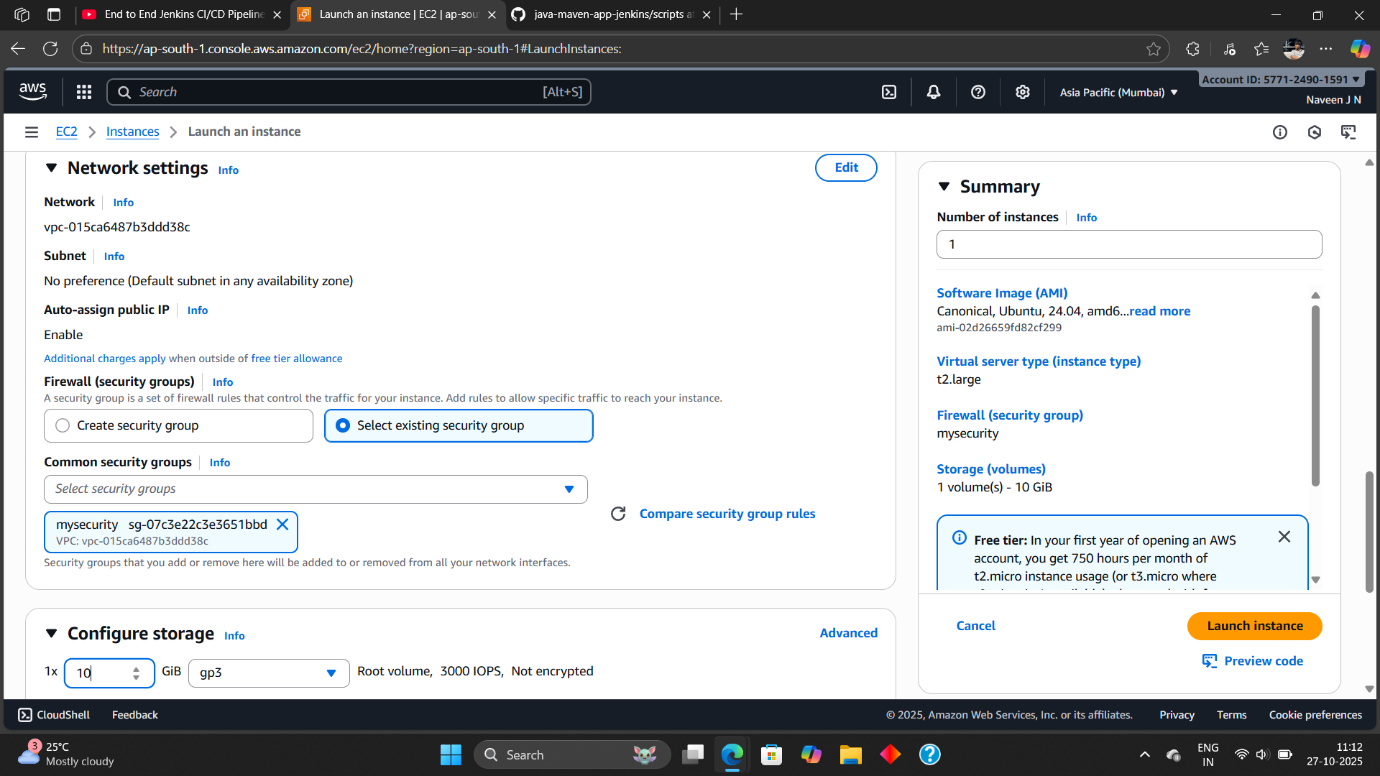
1. Create the Instance (Your\_Instance\_Name) OS (Ubuntu)



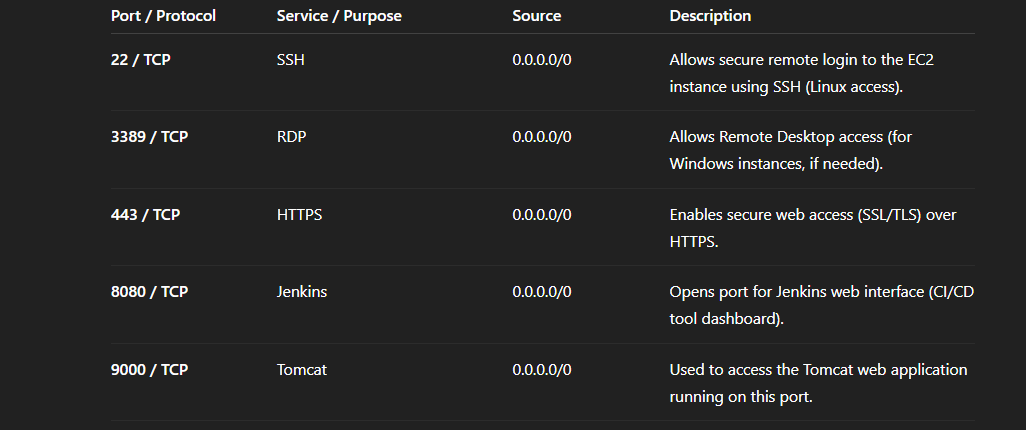
2. Instance Type (t2.large) and Select the Key Pair exist Or Create It.

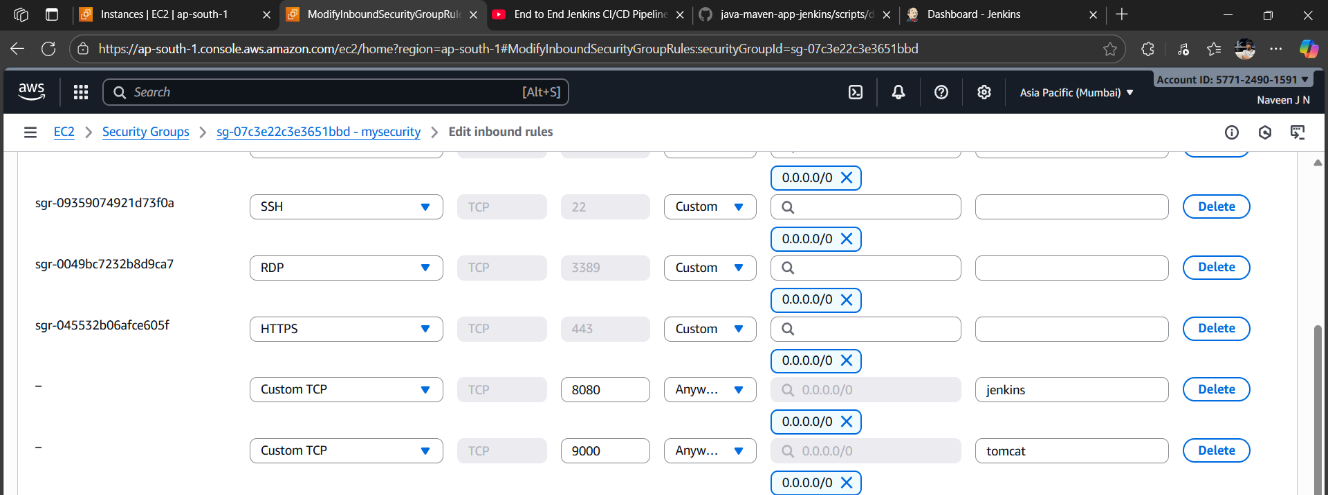


3. Add The Security Group but in this we need to add some inbound rules.



The Inbound Role:





4. Configure Storage Change as 1x 10 (Because we are download the dependencies

Of project required and launch Instance)



**Step 2: Login Through Putty And Download Jenkins, docker**

1**. Get Your EC2 Instance’s Public IP**

Go to your **AWS Management Console** → **EC2** → **Instances**.

Select your running instance.

In the Details tab (bottom pane), find:

Public IPv4 address → e.g., 13.233.45.112 (this is the IP you’ll use).

2. Convert .pem Key to .ppk (for PuTTY)

PuTTY doesn’t accept .pem keys directly — you must convert it to .ppk format.

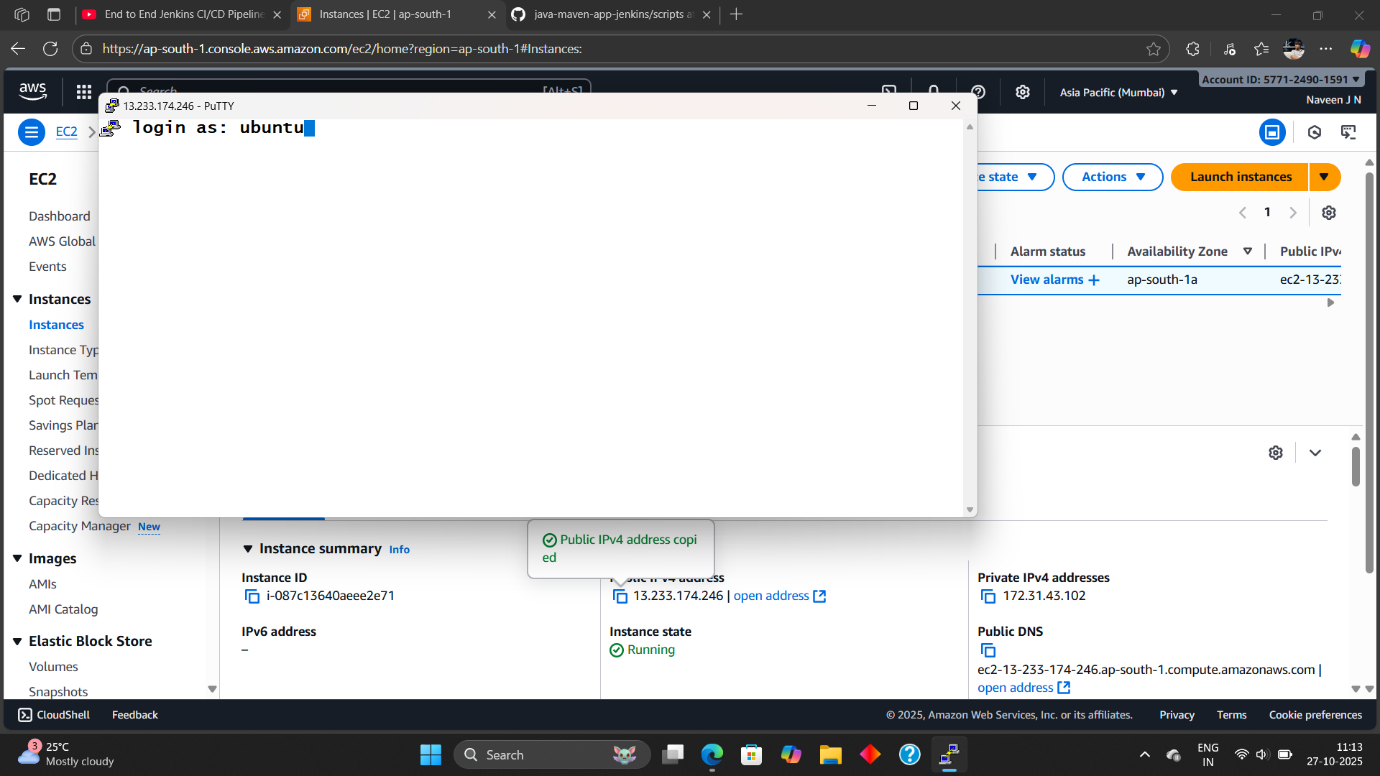
Open PuTTYgen (comes with PuTTY installation).

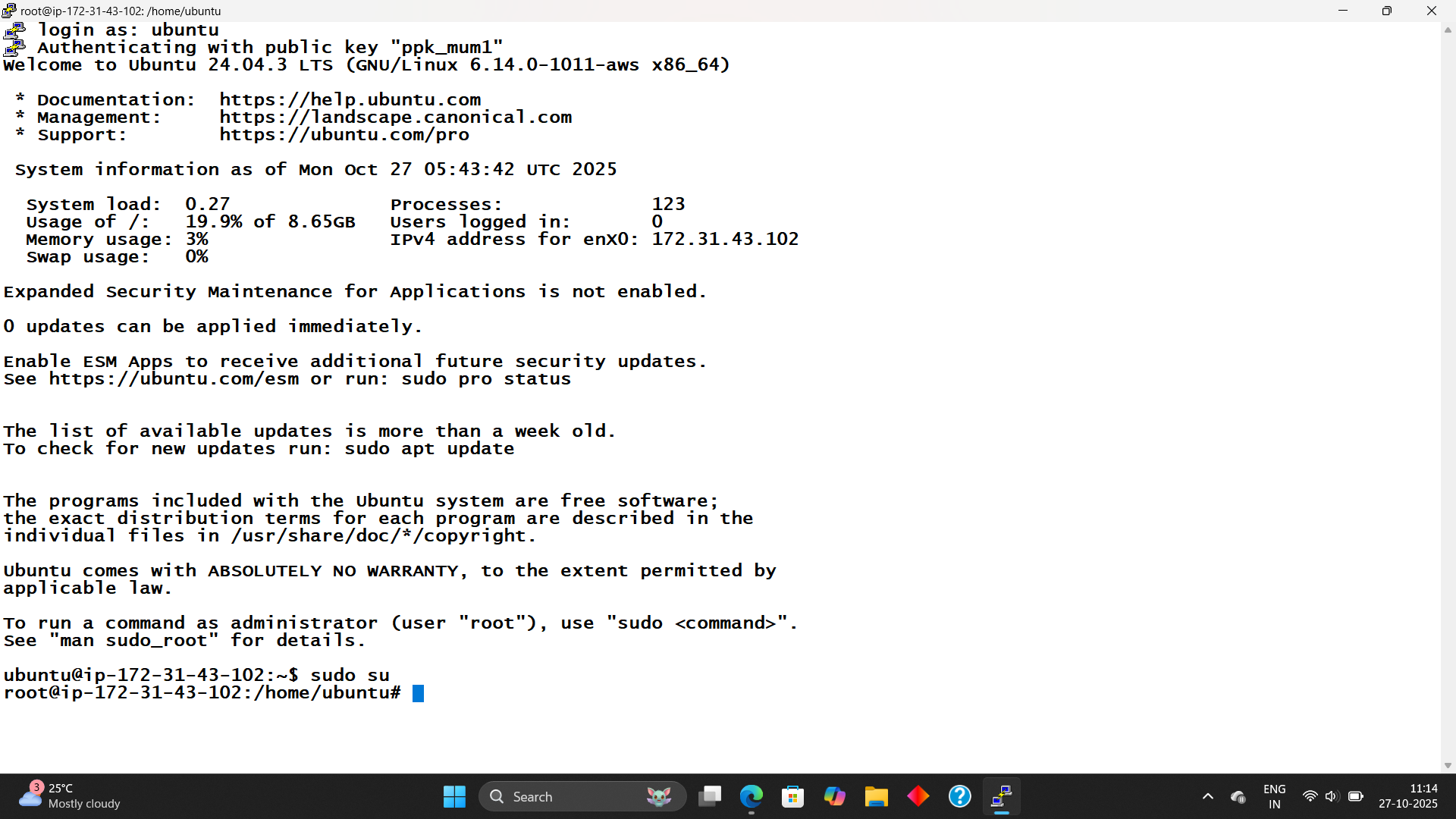
Click Load → browse and select your .pem key file downloaded from AWS.

Click Save private key → save as my-key.ppk.

When prompted for username, enter:

* For Amazon Linux / RHEL → **ec2-user**
* For Ubuntu → **ubuntu**
* For CentOS → **centos**
* For Debian → **admin** or **ec2-user**
* **After Login Convert to root user : Sudo su**





3. Install Jenkins and Docker :

Step1:

**Create Jenkins Install Script**

Create a file: **#nano jenkins.sh**

After create file paste this script then **(Save → Ctrl + O, Enter → Ctrl + X)**

#!/bin/bash

#this Script belong to Cloudaseem Youtube channel #####

# jenkins installation on ubuntu

sudo apt update -y

sudo apt install fontconfig openjdk-17-jre -y

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

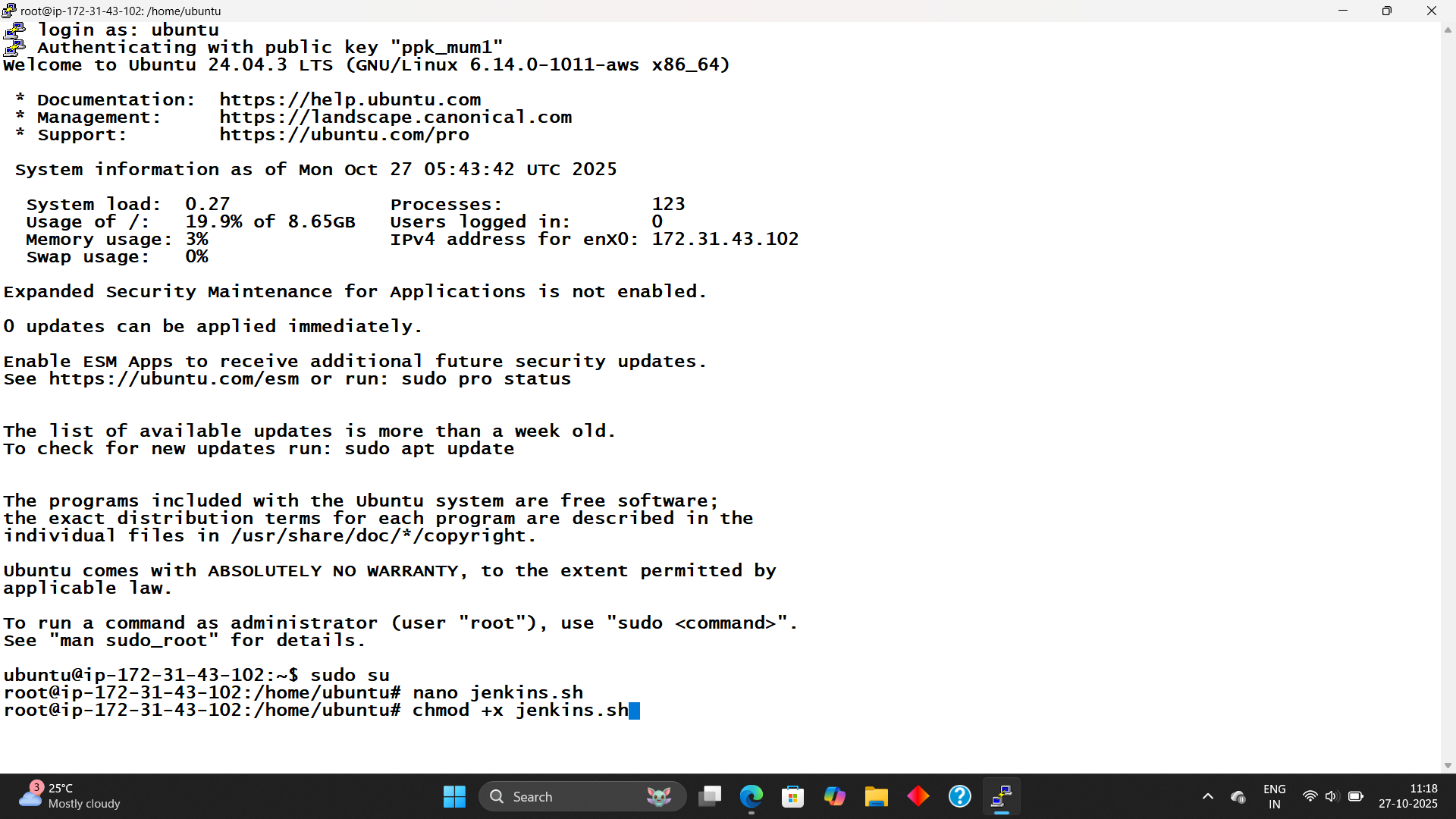
sudo apt-get update -y

sudo apt-get install jenkins -y

sudo systemctl enable jenkins

sudo systemctl start jenkins

After execute this script to download package**: #chmod +x jenkins.sh**



Step2:

**Create Docker Install Script**

Create a file: **#nano Docker.sh**

After create file paste this script then **(Save → Ctrl + O, Enter → Ctrl + X)**

#!/bin/bash

# Script to install Docker on an EC2 instance and configure permissions

# Update the package list

sudo apt-get update -y

# Install Docker

sudo apt-get install docker.io -y

# Add the 'ubuntu' and 'jenkins' users to the 'docker' group to allow running Docker without sudo

sudo usermod -aG docker ubuntu

sudo usermod -aG docker jenkins

# Apply the new group settings immediately

newgrp docker

# Set correct permissions for the Docker socket to allow 'docker' group members to access it

sudo chmod 660 /var/run/docker.sock

sudo chown root:docker /var/run/docker.sock

# Restart Docker service to apply changes

sudo systemctl restart docker

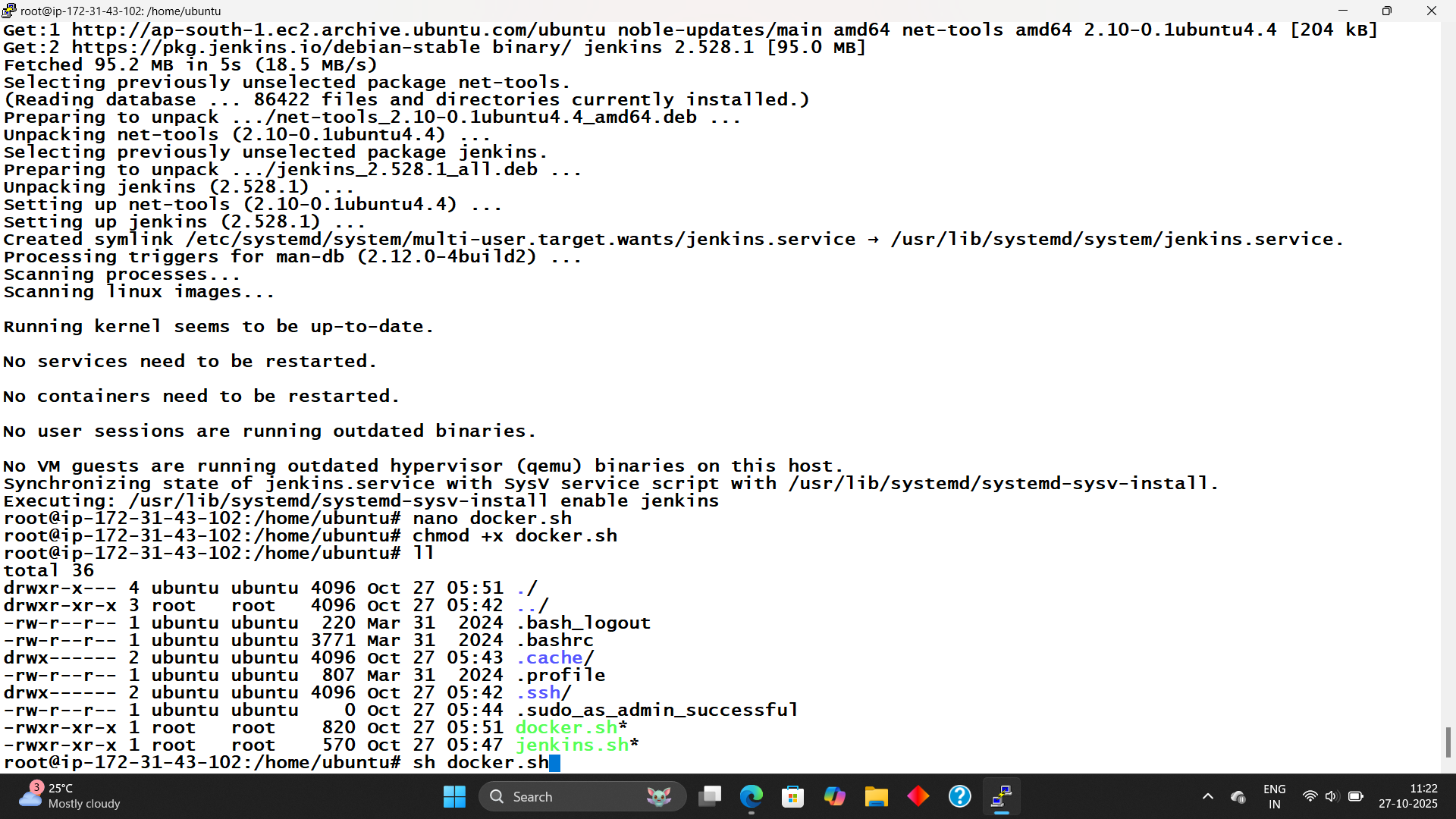
# Verify installation

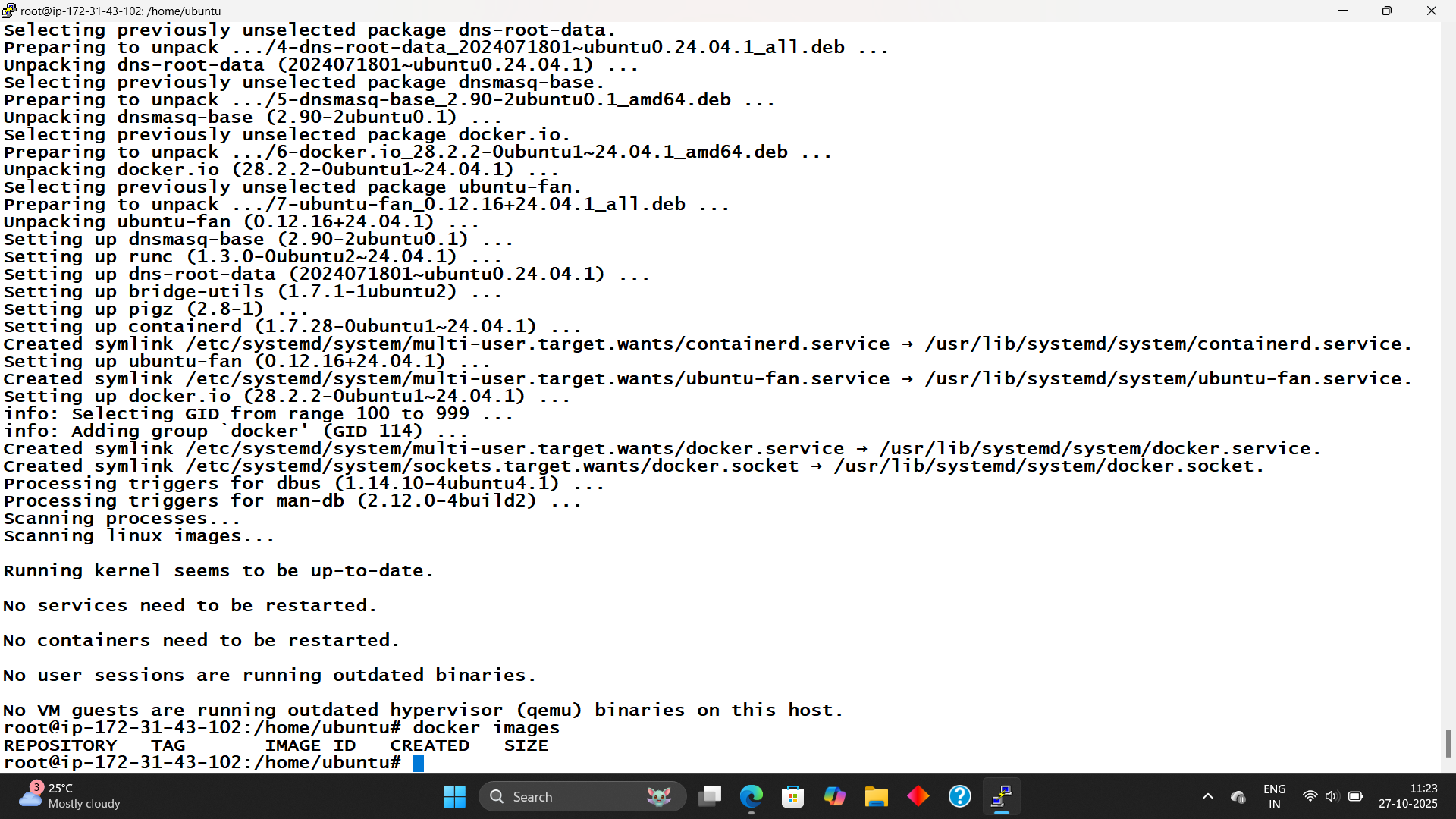
docker -version

# Run SonarQube container in detached mode with port mapping

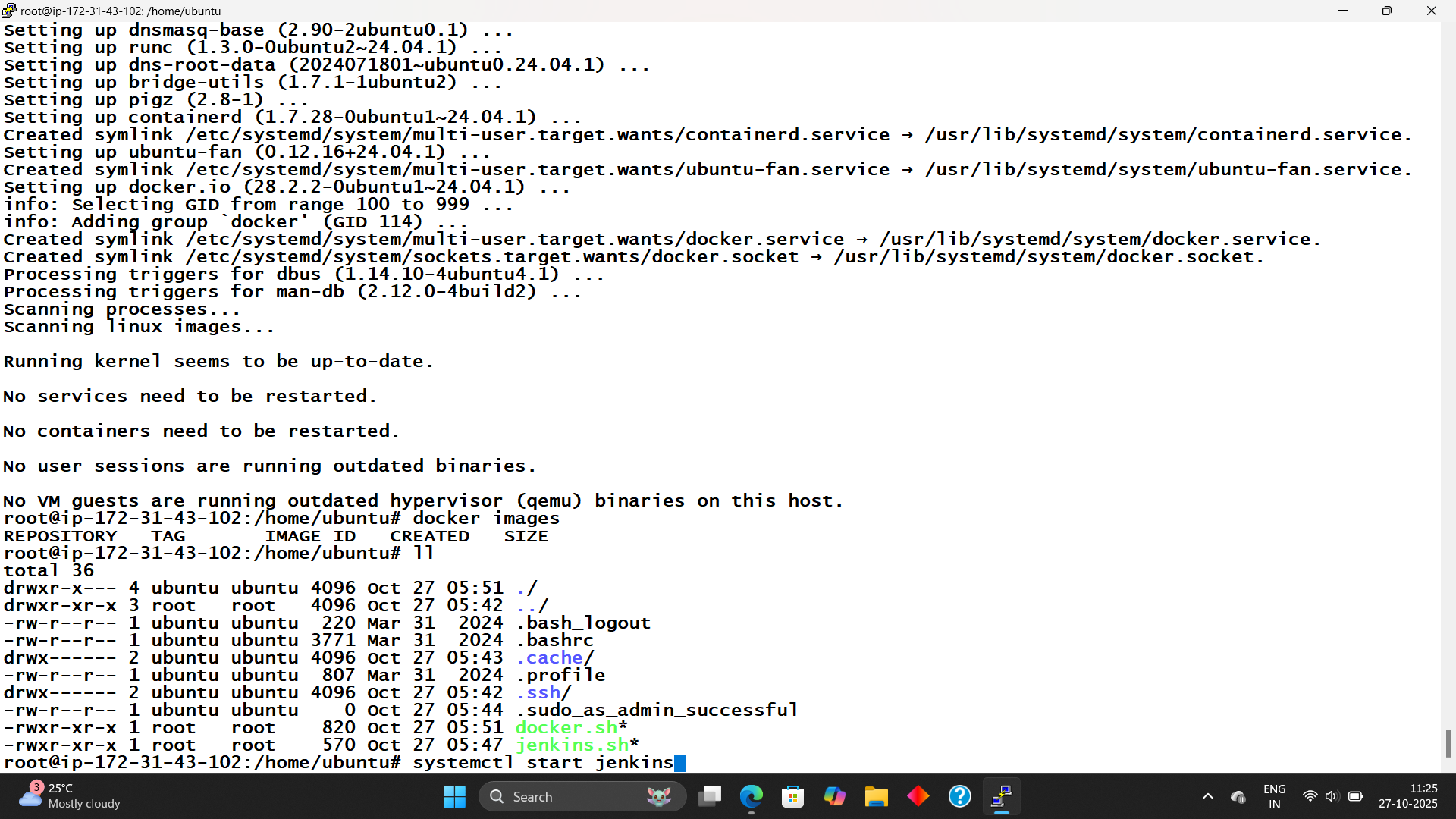
#docker run -d --name sonar -p 9000:9000 sonarqube:lts-communityAfter execute this script to download package**: #chmod +x Docker.sh**

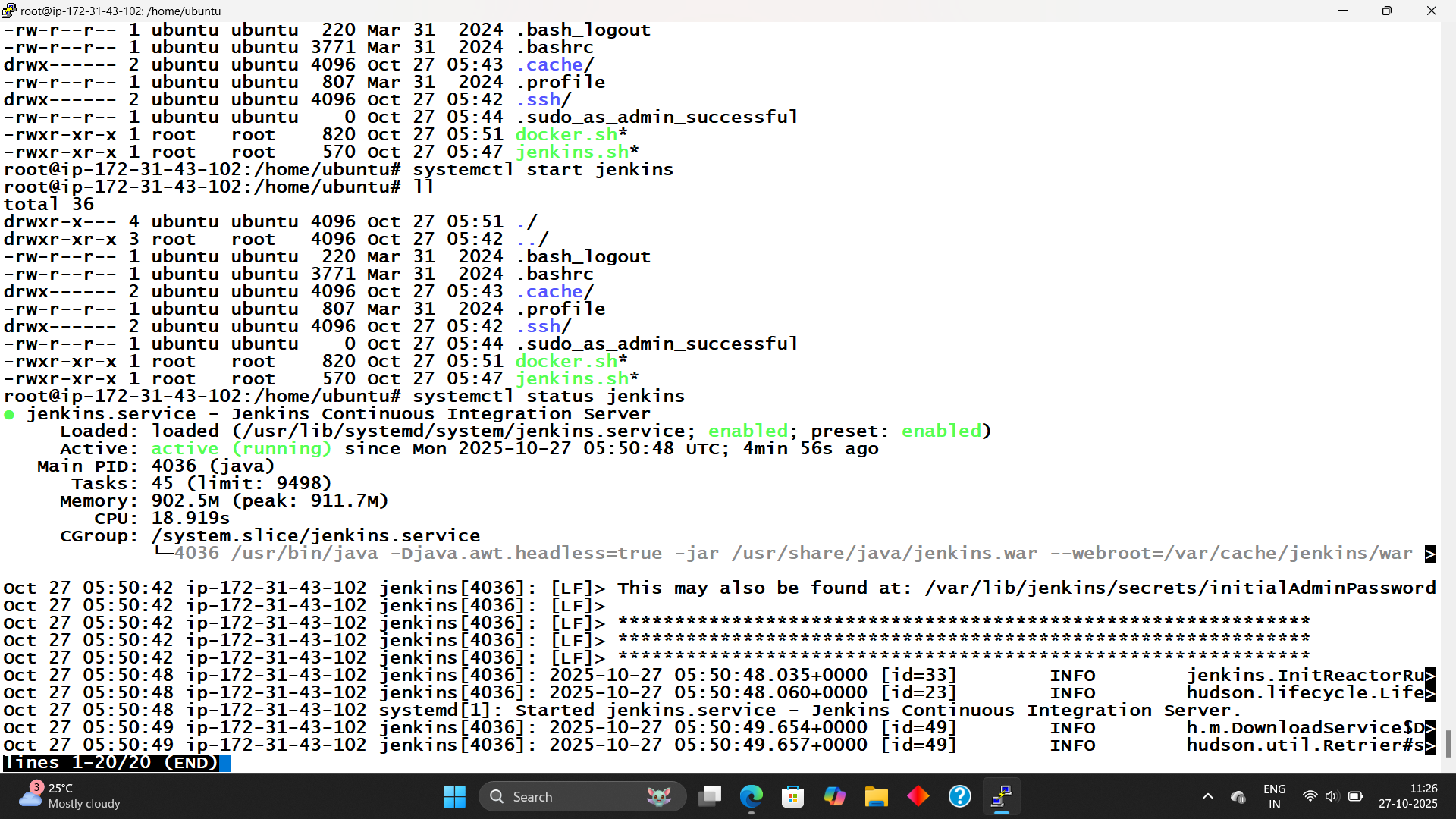
**#sh Docker.sh**

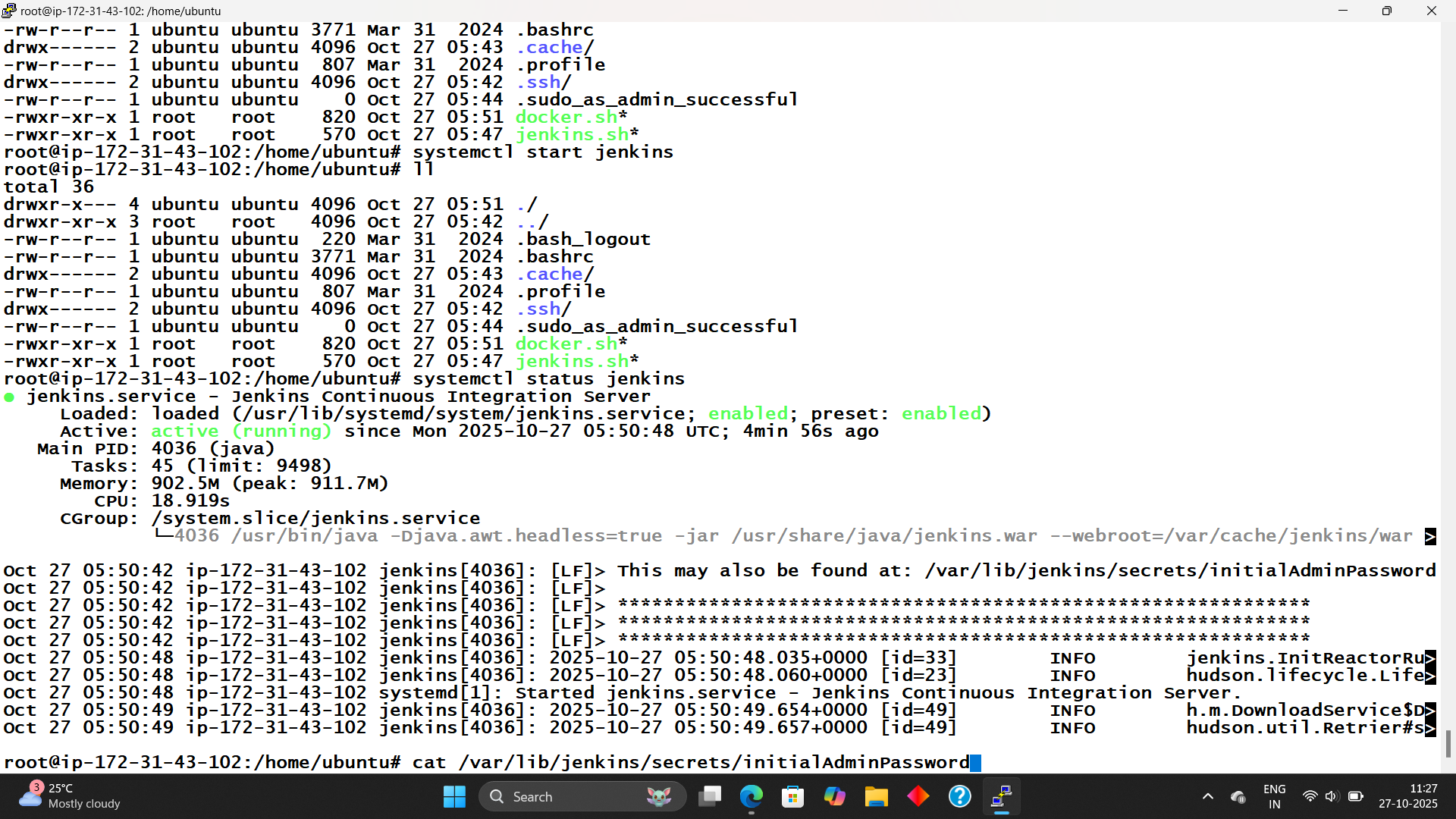
****

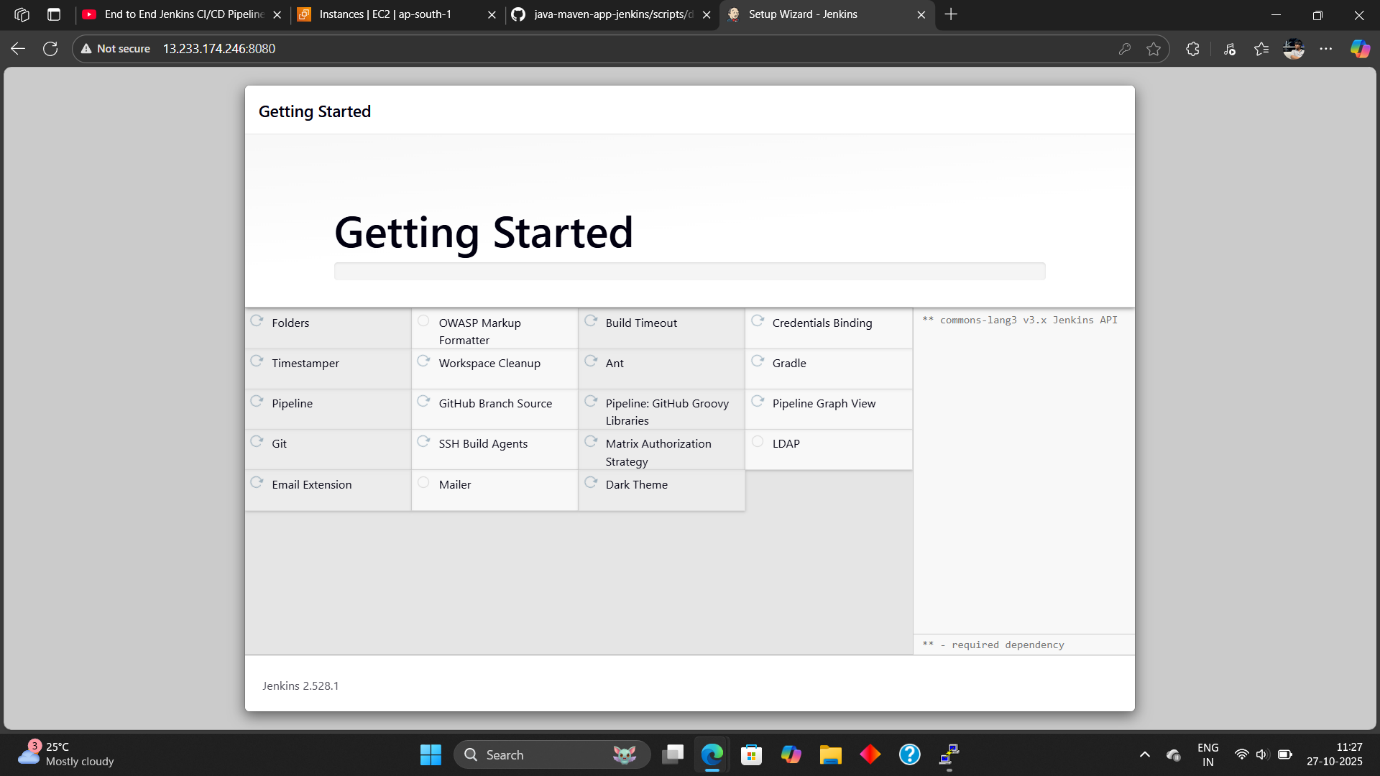
****

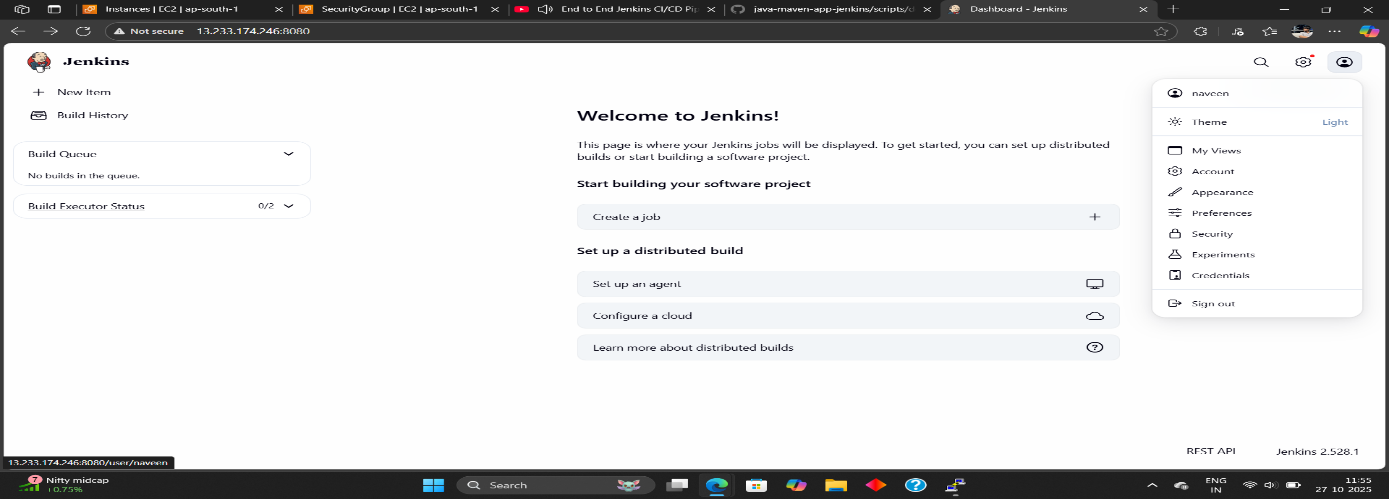
**Step 3: Start the Jenkins Server And Login:**







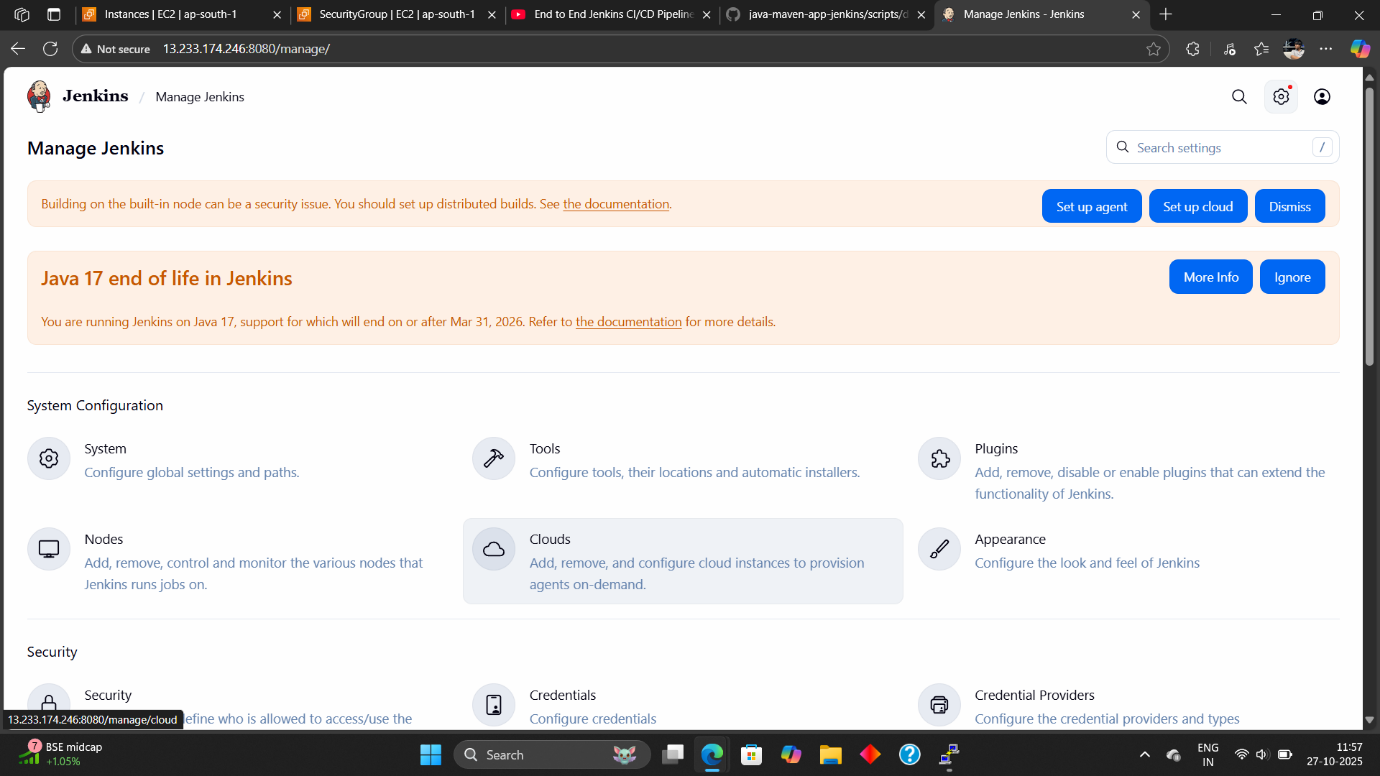


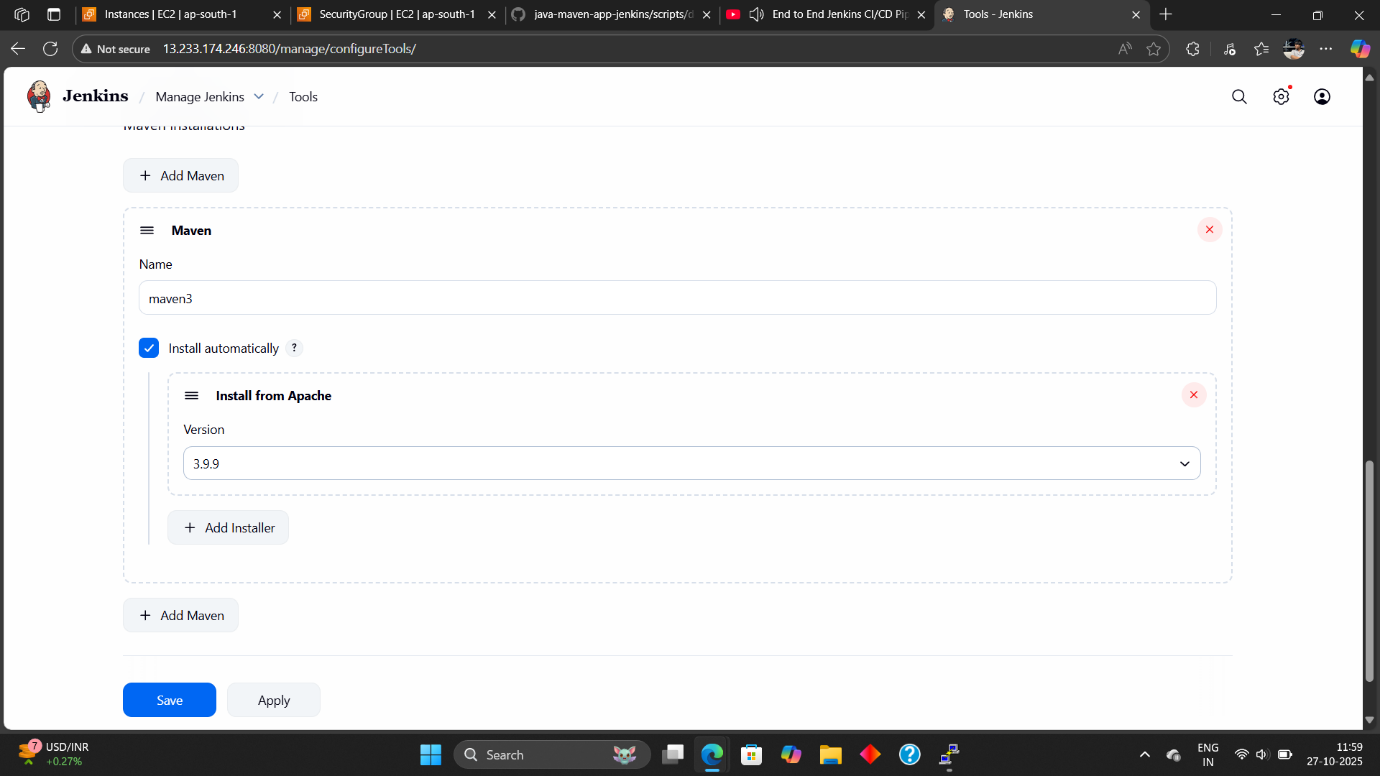


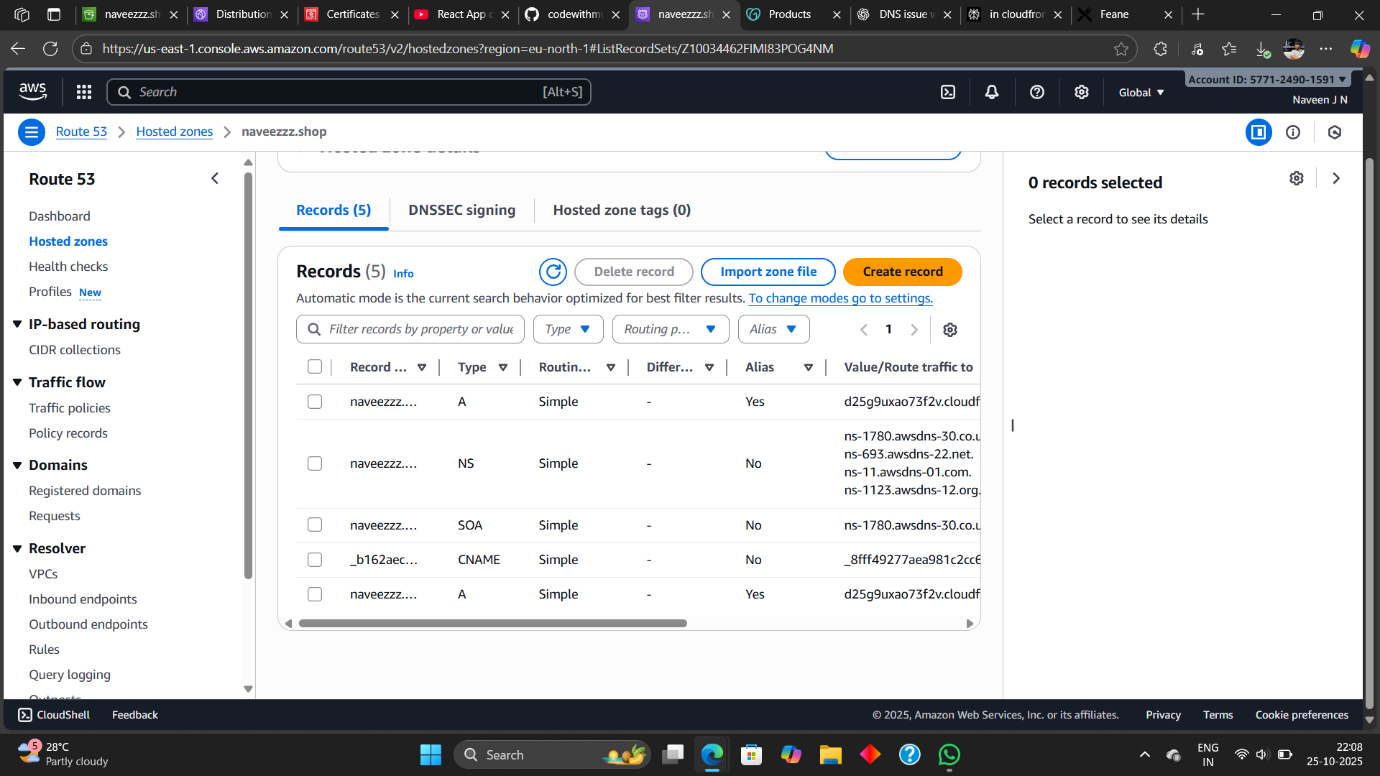
**Step 4: Add Maven And Its Version:**

1. Go to Settings → Manage Jenkins → Tools → Add Maven Name and Version.

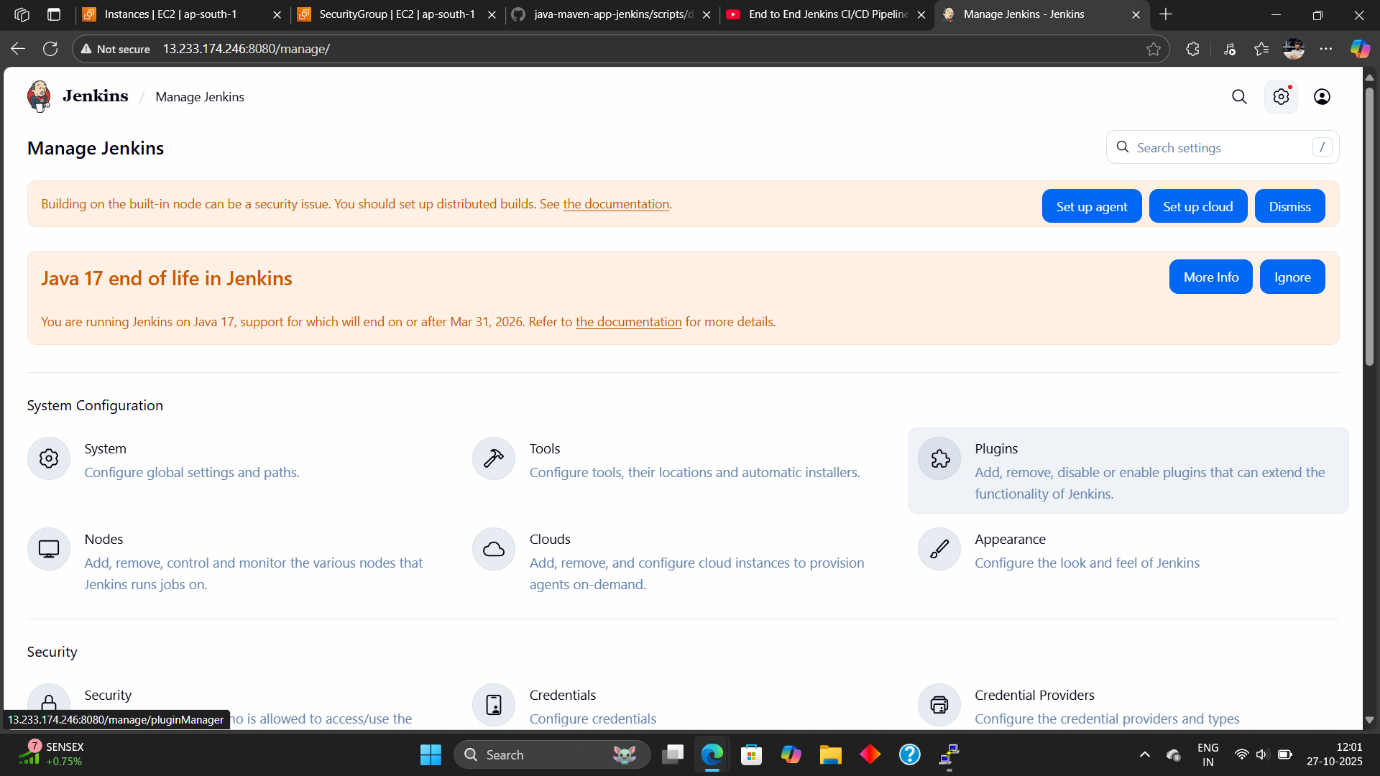
2.Create the job.

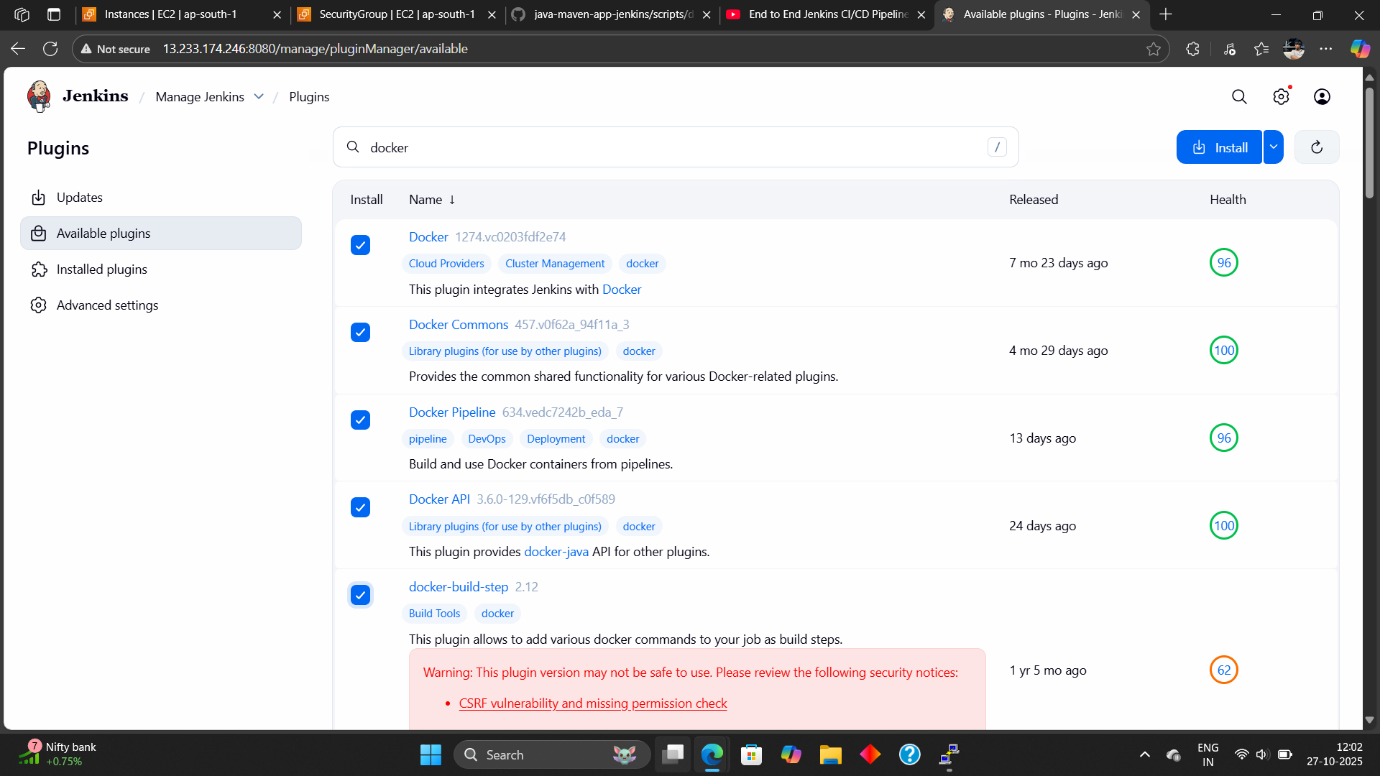






2. Manage Jenkins → Plugins → Install Docker Plugins.



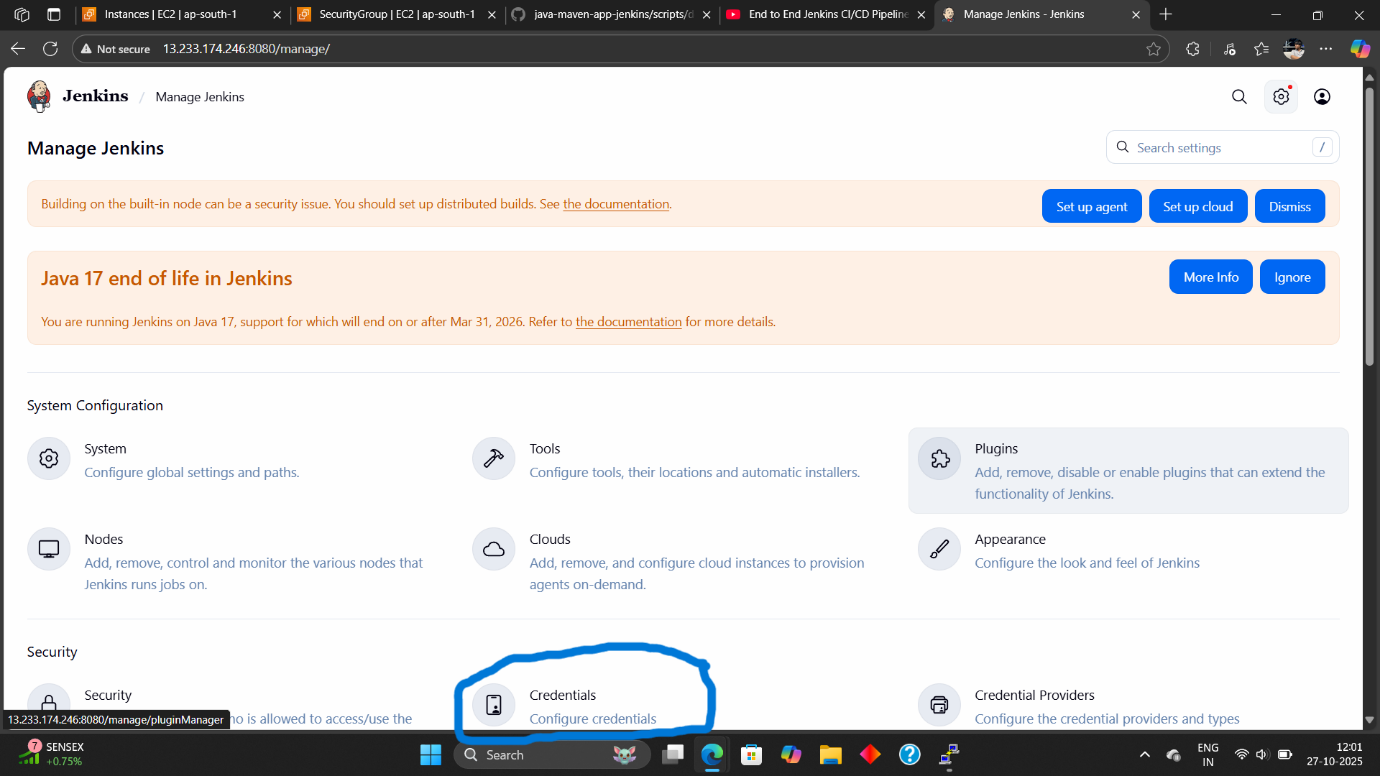


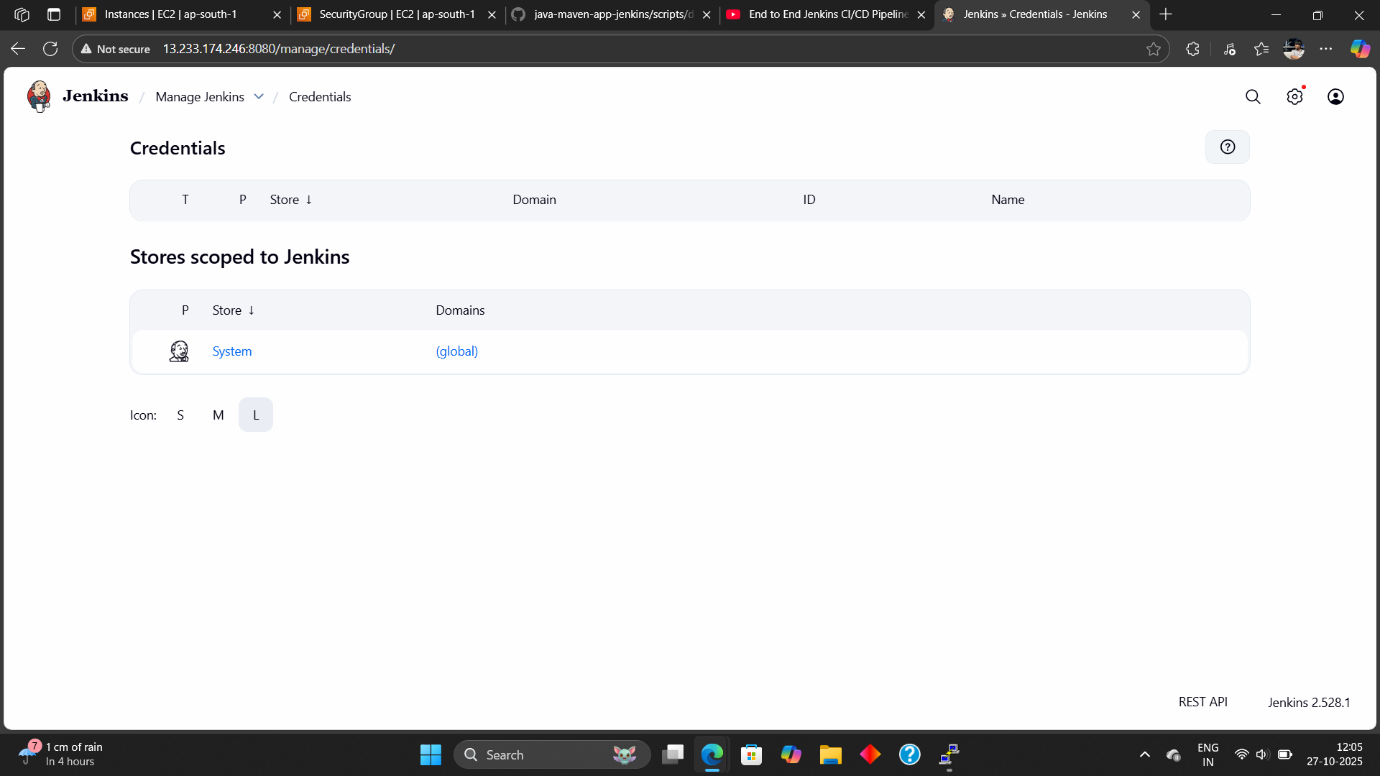
**3. Docker Credentials in Jenkins:**

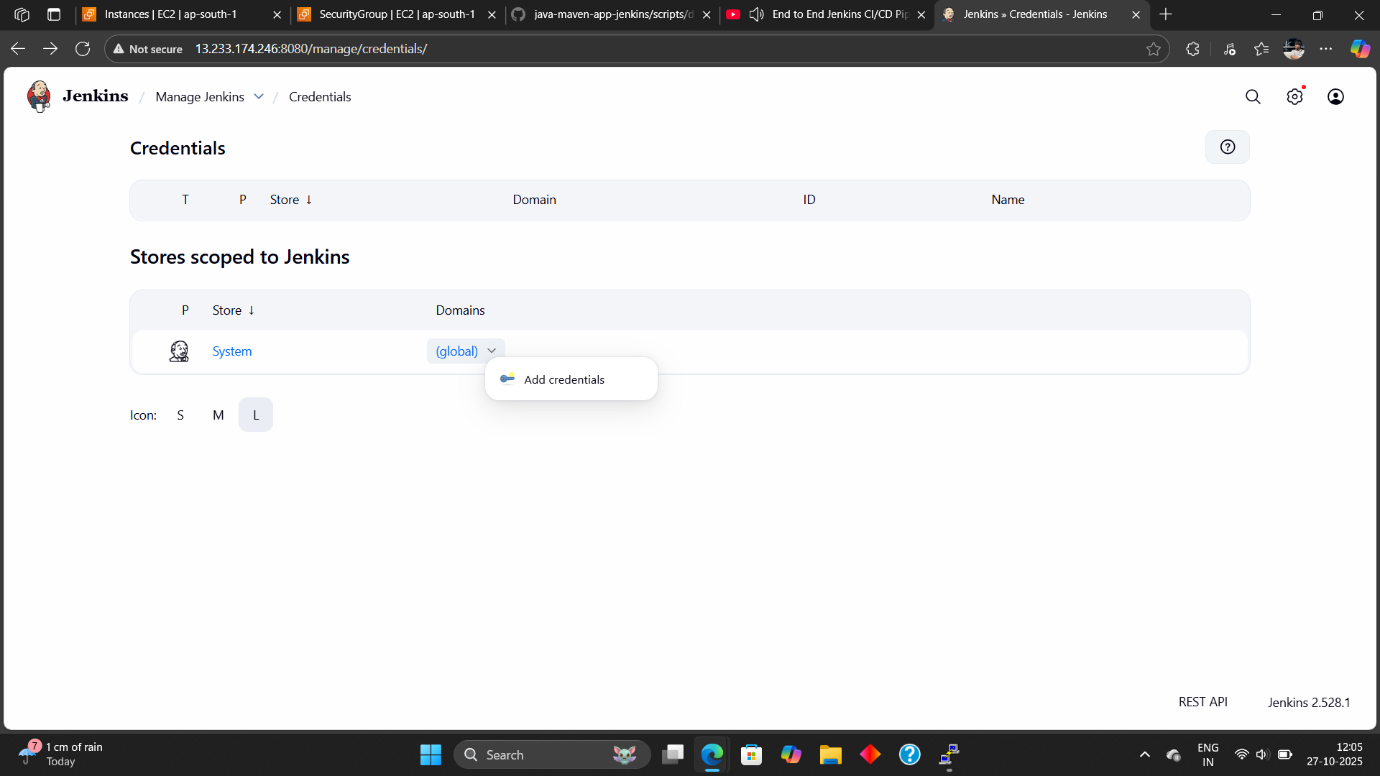
When you use Jenkins for a **CI/CD pipeline** that builds **Docker images** and pushes them to **Docker Hub**, **Jenkins** needs **permission** to **log in to Docker Hub.**

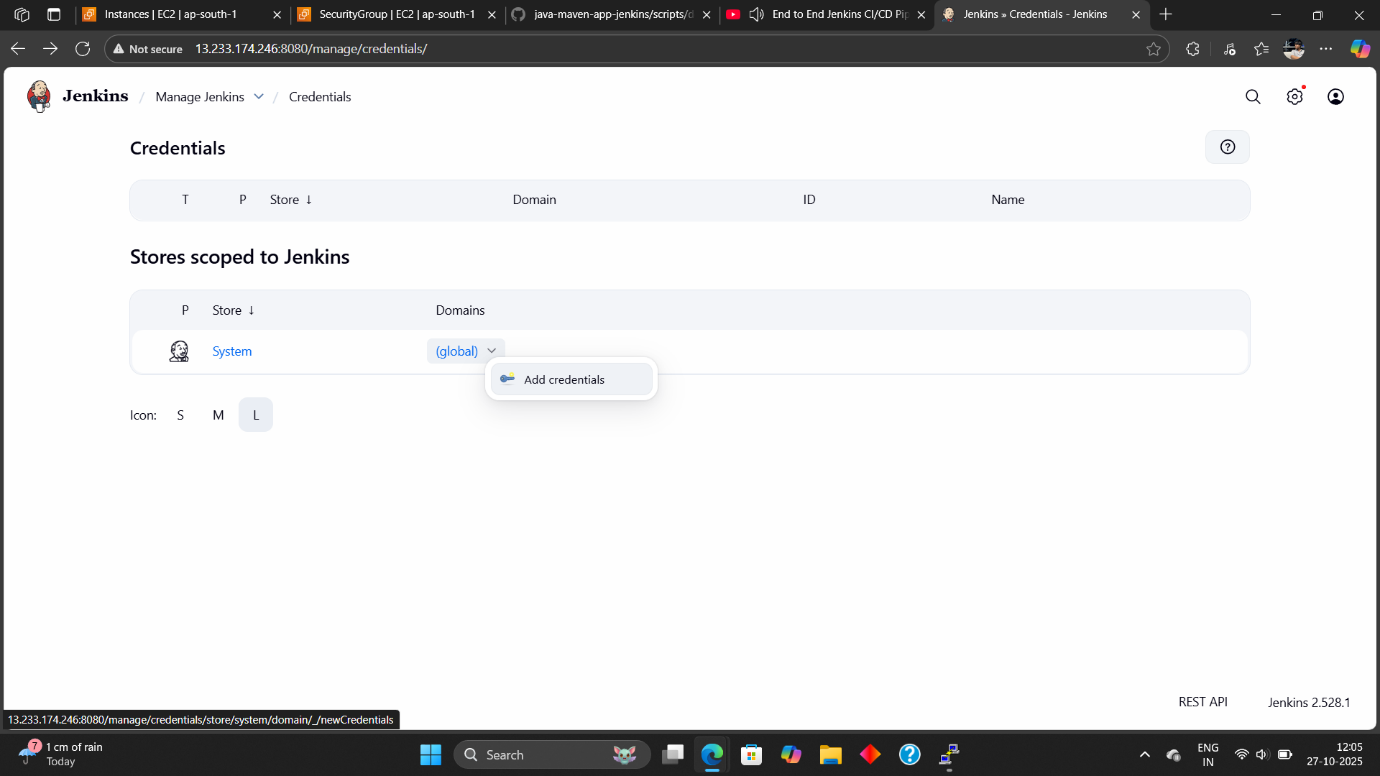
Manage Jenkins → Credentials → Domain (Global)

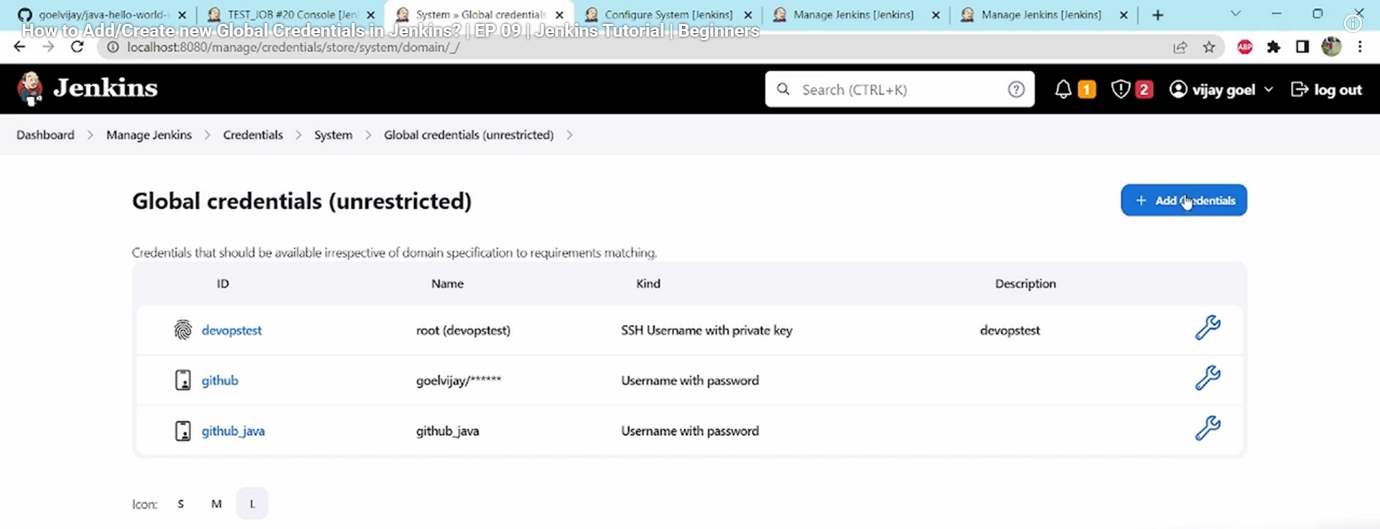
Add Your **(Docker\_Name)** and **(Password)** For Credentials

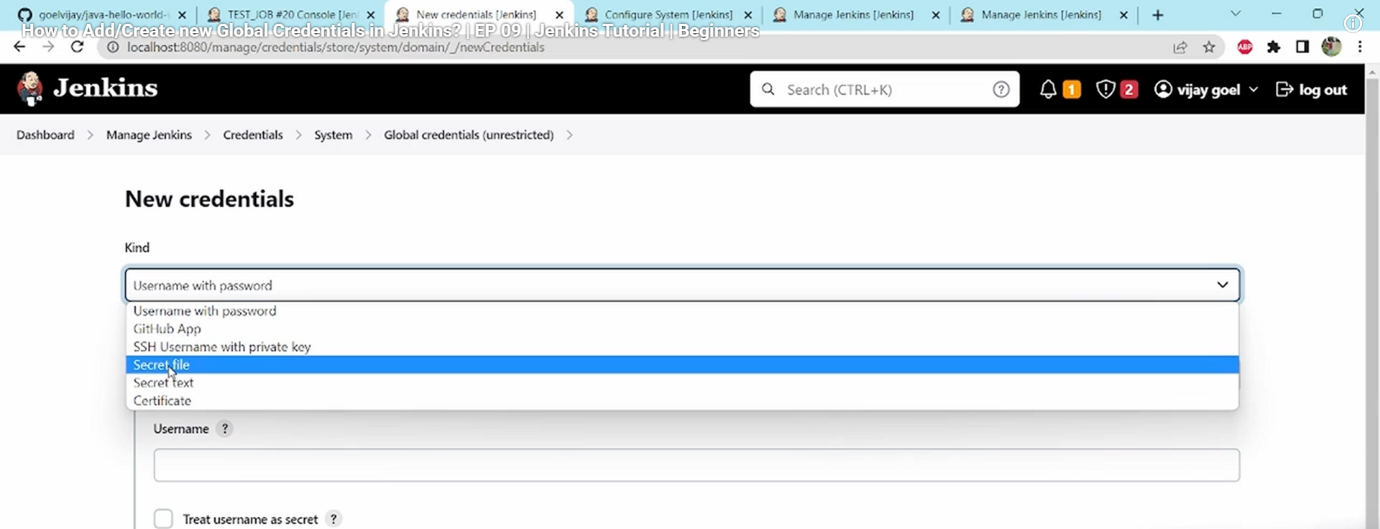


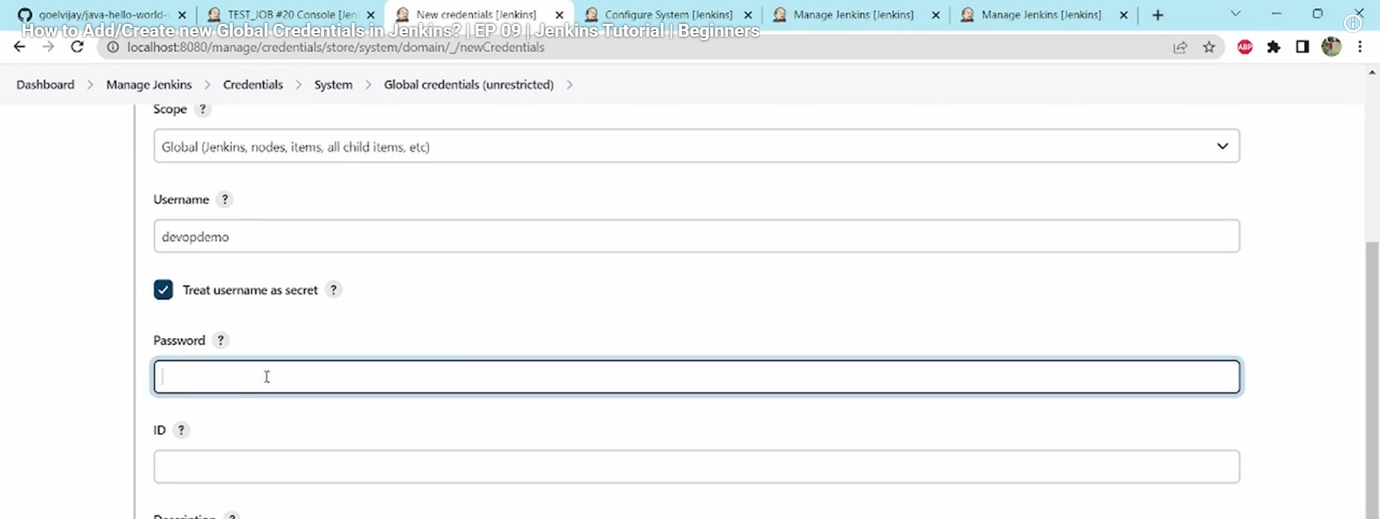








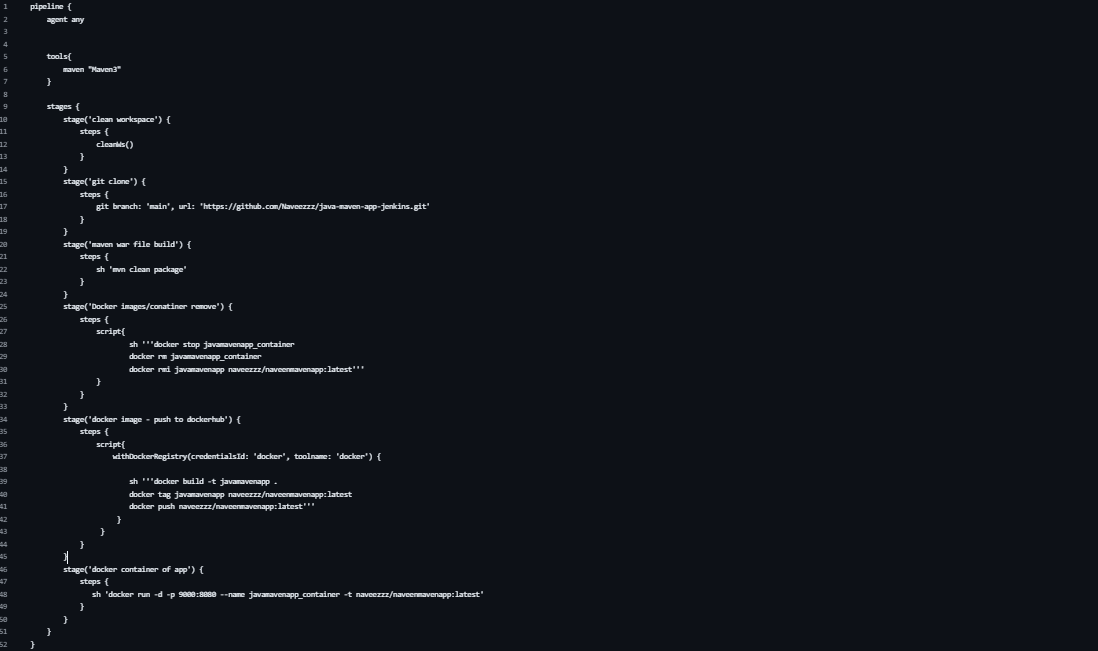




**Step 4: Add Pipeline Script For CI/CD:**

1. Go to you Job click → Configure → Pipeline → Add Pipeline Script → Apply

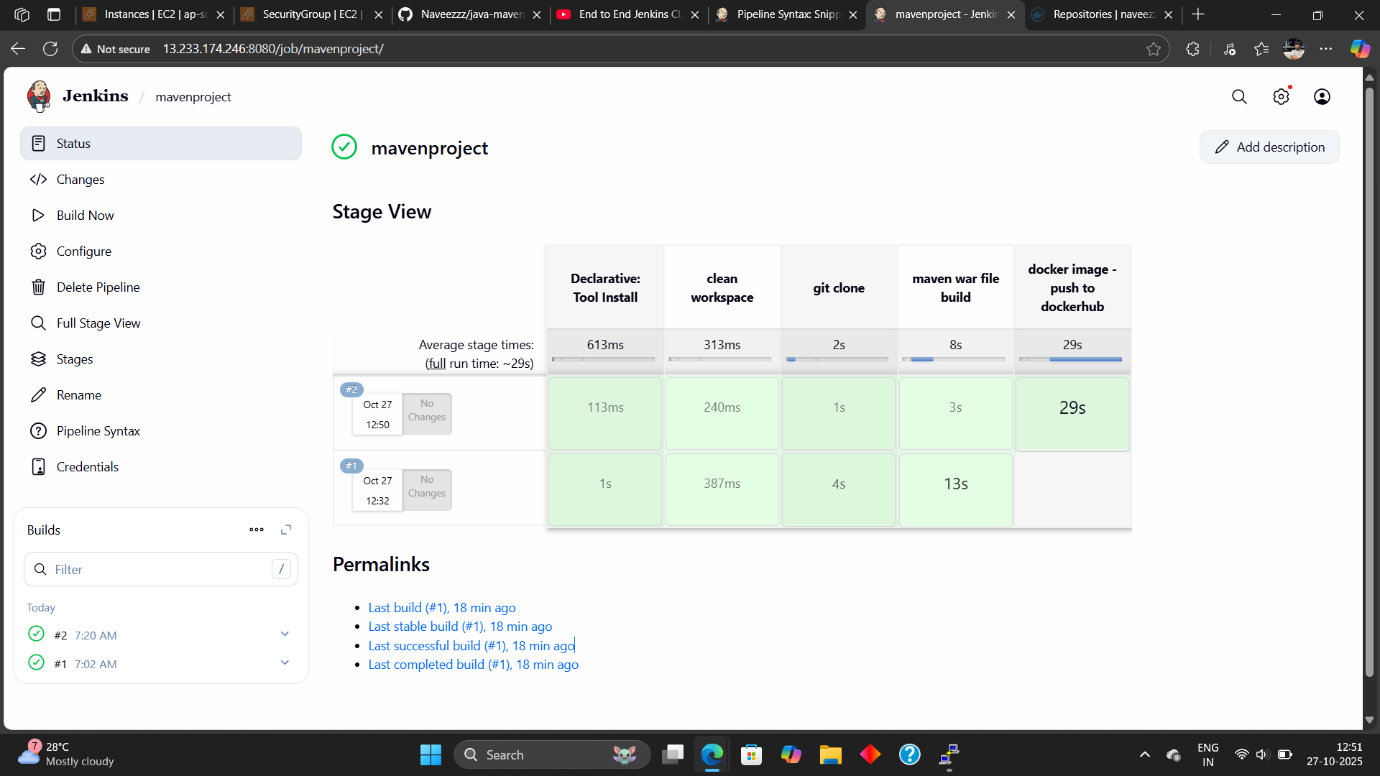
→ Save.

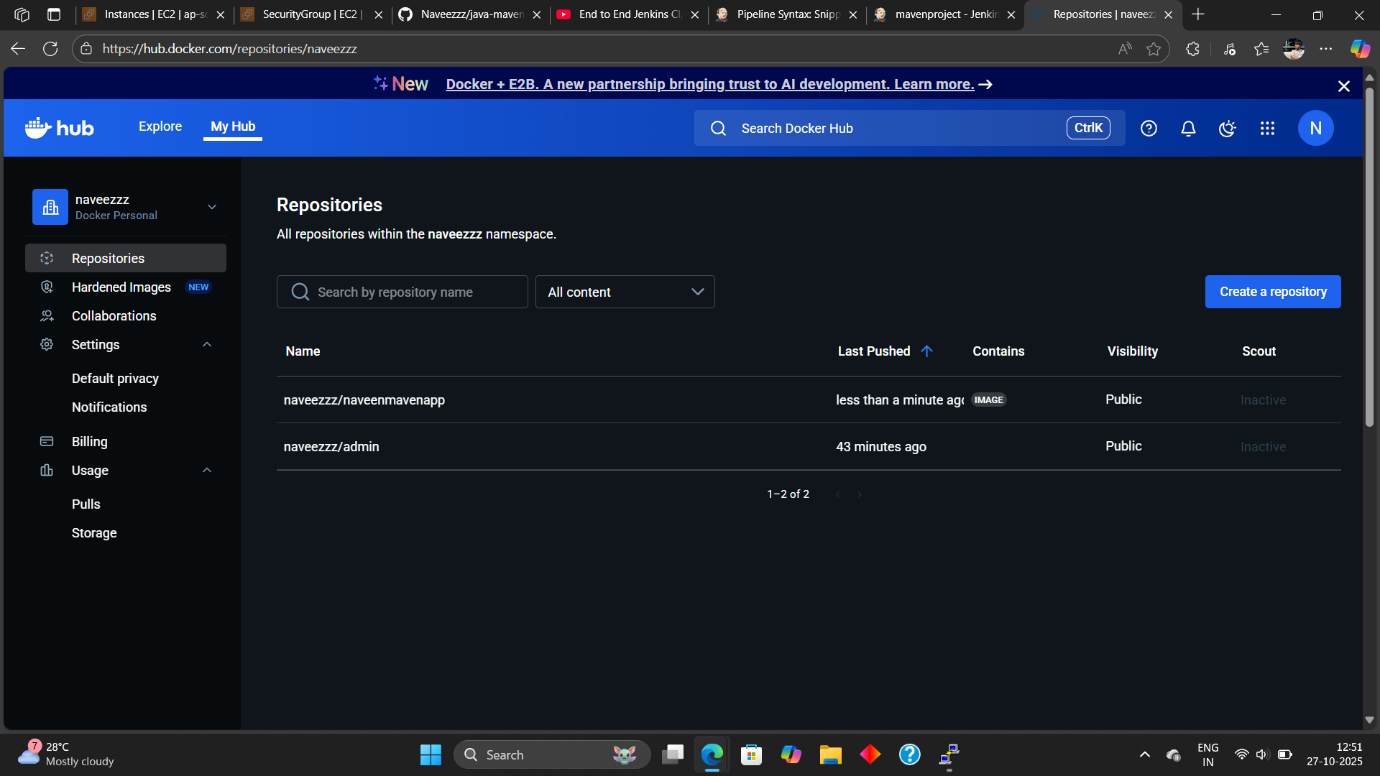


**Step 5: Add Pipeline Script For CI/CD:**

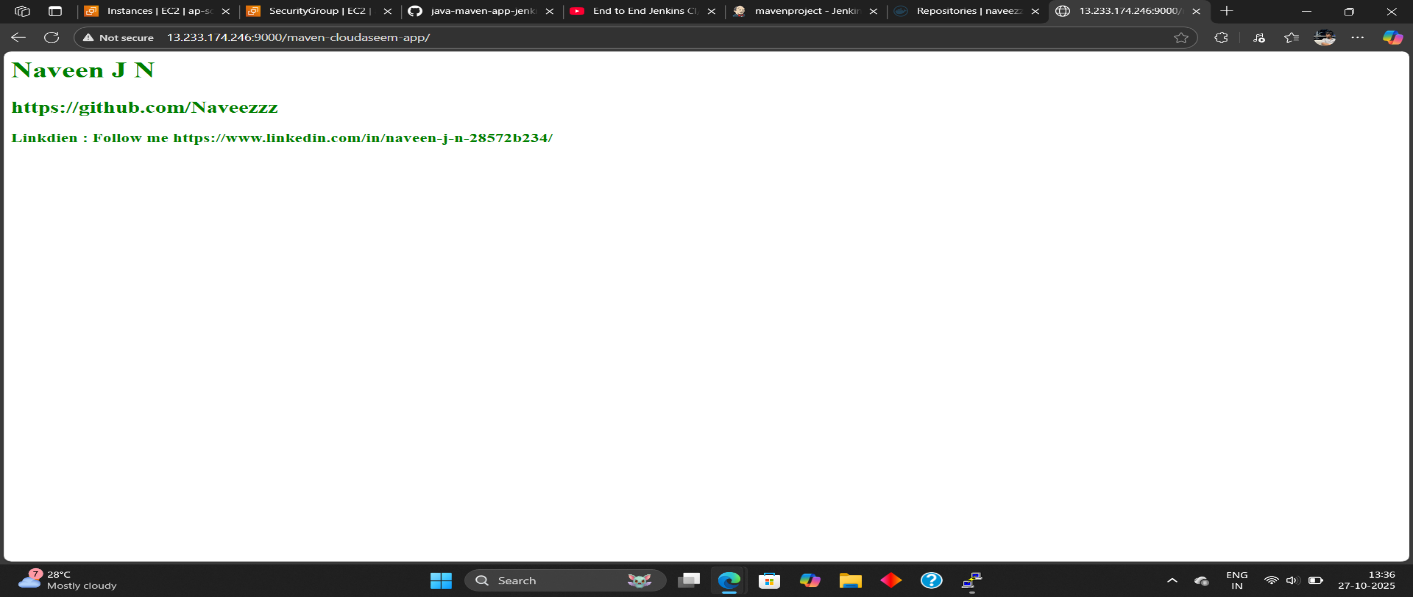
1. Go to you Job and click Job → Click **(Build Now)** → Now Job Runs And update

In Docker Hub and Tomcat website.





**Output:**

****