DevOps Project

Problem Statement:

Create an end-to-end CI/CD pipeline in AWS platform using Jenkins as the orchestration tool, GitHub as scm, maven as the build tool, deploy in a docker instance and create a docker image, store the docker image in ECR, Kubernetes deployment using ECR image. Build sample java web app using Maven.

Approach:

Requirements:

- **CI/CD pipeline System**
- **Git** - Local version control system
- **GitHub** - Distributed version control system
- **Jenkins** - Continuous Integration tool
- **Maven** - Build Tool
- **Docker** - Containerization
- **Kubernetes** - Container Management Tool

Step-1:

1. **Set up CI/CD with GitHub, Jenkins, Maven & Tomcat:**
    - **Install Jenkins:**
        - Download and install Jenkins on your server or local machine.
        - Start Jenkins and complete the initial setup wizard.
    - **Configure Maven and Git:**
        - Install Maven and Git on the same machine where Jenkins is installed.
        - Configure the Maven and Git paths in Jenkins under “Global Tool Configuration”.
    - **Set up Tomcat Server:**
        - Download and install Apache Tomcat.
        - Configure Tomcat to run on a specific port and ensure it is running.

- **Integrate GitHub, Maven, and Tomcat Server with Jenkins:**
  - Create a GitHub repository for your project.
  - In Jenkins, create a new job and configure it to pull code from your GitHub repository.
  - Add Maven build steps to compile and package your application.
  - Configure Jenkins to deploy the packaged application to the Tomcat server.
- **Create CI and CD Jobs:**
  - Set up a Continuous Integration (CI) job to automatically build and test your code whenever changes are pushed to GitHub.
  - Set up a Continuous Deployment (CD) job to deploy the application to Tomcat after a successful build.
- **Test the Deployment:**
  - Push changes to your GitHub repository and verify that Jenkins automatically builds, tests, and deploys the application to Tomcat

Step-2:

1. **Set up CI/CD with GitHub, Jenkins, Maven & Docker:**
   - **Set up the Docker Environment:**
     - Install Docker on your server or local machine.
     - Verify the Docker installation by running a test container.
   - **Create an Image and Container on Docker Host:**
     - Write a Dockerfile to define your application's environment.
     - Build the Docker image using the Dockerfile.
     - Run a container from the Docker image to ensure it works as expected.
   - **Integrate Docker Host with Jenkins:**
     - Install the Docker plugin in Jenkins.
     - Configure Jenkins to communicate with the Docker daemon.
     - Set up credentials in Jenkins to access the Docker host.
   - **Create CI/CD Job on Jenkins to Build and Deploy on Container:**

        - Create a new Jenkins job and configure it to pull code from your GitHub repository.
        - Add build steps to use Maven for compiling and packaging the application.
        - Add steps to build the Docker image and push it to a Docker registry.
        - Configure Jenkins to deploy the Docker container using the newly built image

Step-3:

1. **Build and Deploy on Container:**
    - **CI/CD with GitHub, Jenkins, Maven & Kubernetes:**
        - Set up a CI/CD pipeline integrating GitHub, Jenkins, Maven, and Kubernetes.
    - **Set up Kubernetes (EKS):**
        - Create and configure an Amazon EKS (Elastic Kubernetes Service) cluster.
        - Ensure the cluster is properly set up and accessible.
    - **Write Pod, Service, and Deployment Manifest Files:**
        - Create Kubernetes manifest files for Pods, Services, and Deployments.
        - Define the desired state and configuration for your application in these files.
    - **CI/CD Job to Build Code on Jenkins & Deploy it on Kubernetes:**
        - Create a Jenkins job to pull code from your GitHub repository.
        - Add build steps to use Maven for compiling and packaging the application.
        - Configure Jenkins to apply the Kubernetes manifest files to deploy the application to the EKS cluster.

Step-4:

- **Deploy artifacts on Kubernetes:**

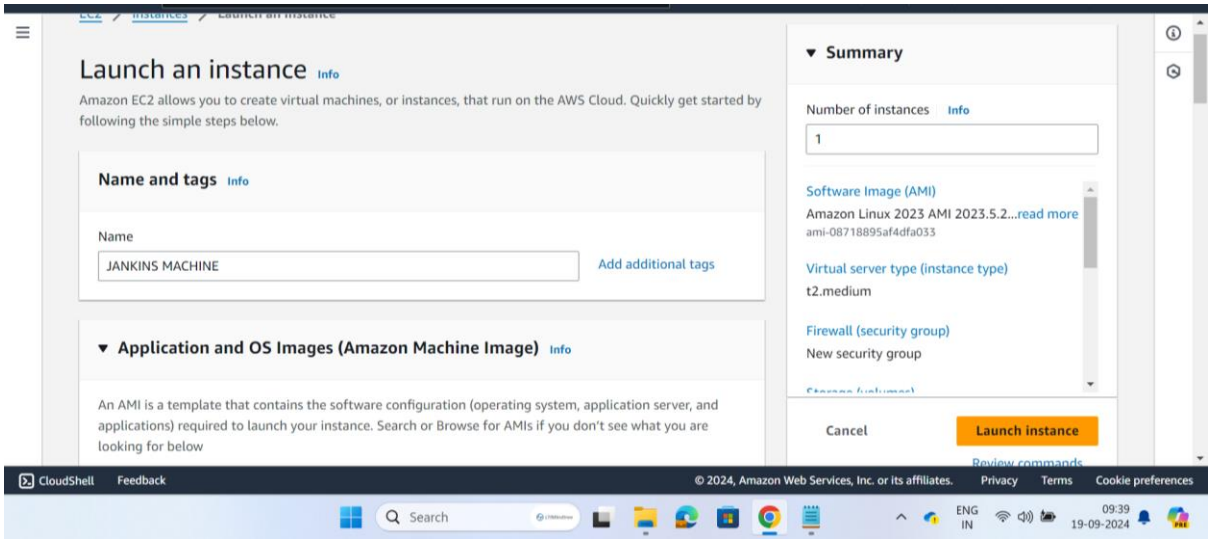
- Write codes in the artifacts of Docker and Kubernetes which we want to run.
- **Build the Code in Jenkins:**
    1. Create a Jenkins job to pull the latest code from your GitHub repository.
    2. Add build steps to compile and package the application using Maven.
    3. Ensure the build artifacts are correctly generated and stored.
- Check in Kubernetes if the pods are getting created or not.
- Now copy the service IP and paste it in the browser and check the output.

Step :

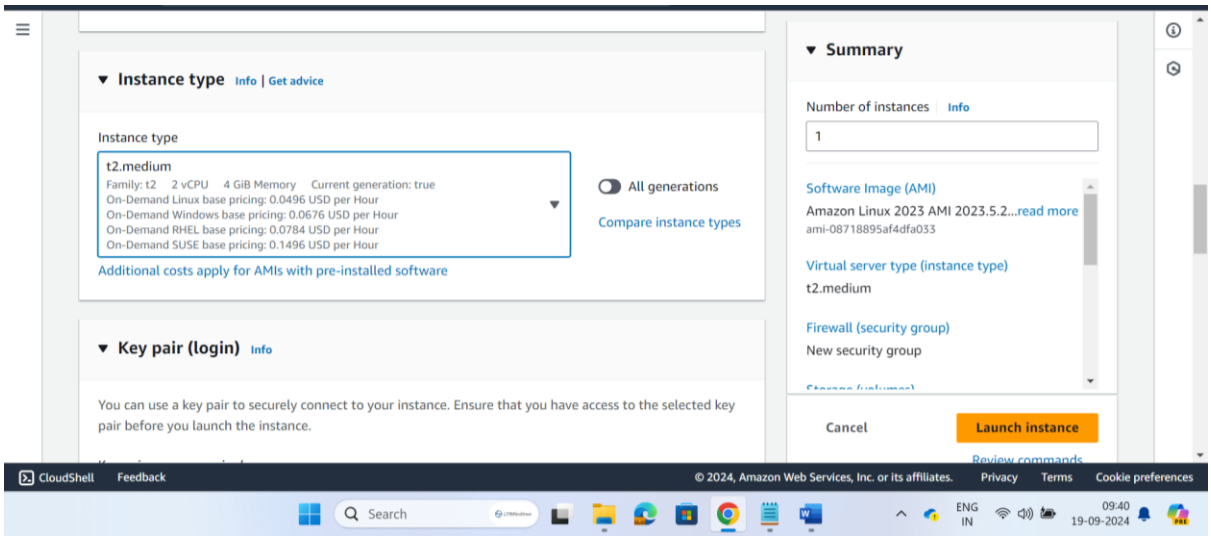
1. Setting Up EC2 Instances

1. 1.1 Creating the EC2 Instances
2. Jenkins Server: Choose an t2.medium instance type because Jenkins, being an orchestration tool, requires more resources for handling builds, running jobs, and managing plugins.
3. Developer Server and Tomcat Server: Use t2.micro instances as they will be handling lighter tasks, such as serving as the development environment and hosting the application.
4. Instance Configuration: Key Pair: Create or use an existing key pair to connect securely to these instances via SSH.

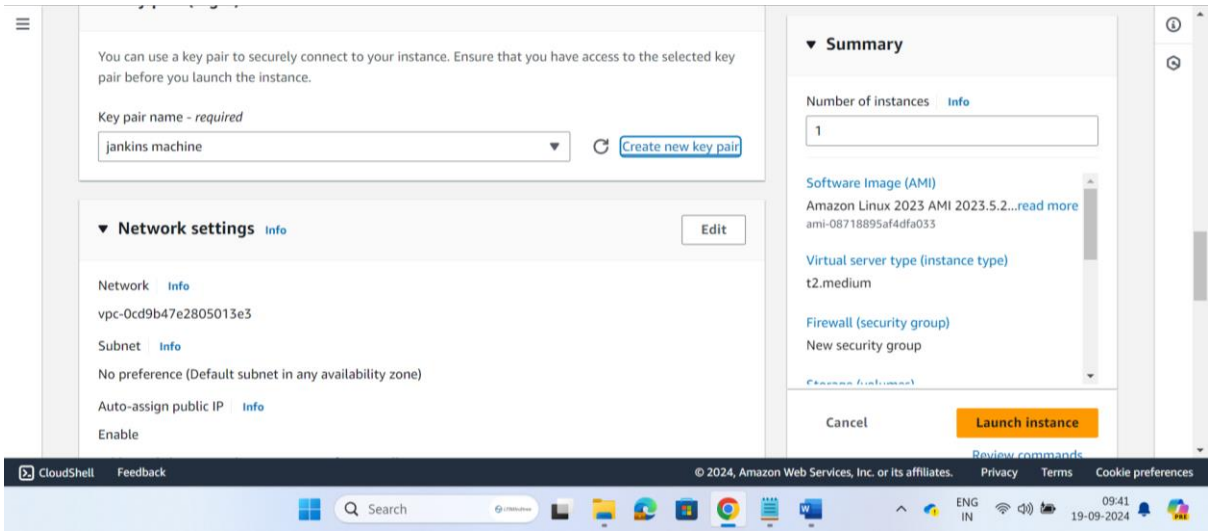
Create three instances developer,Jenkins and tomcat:

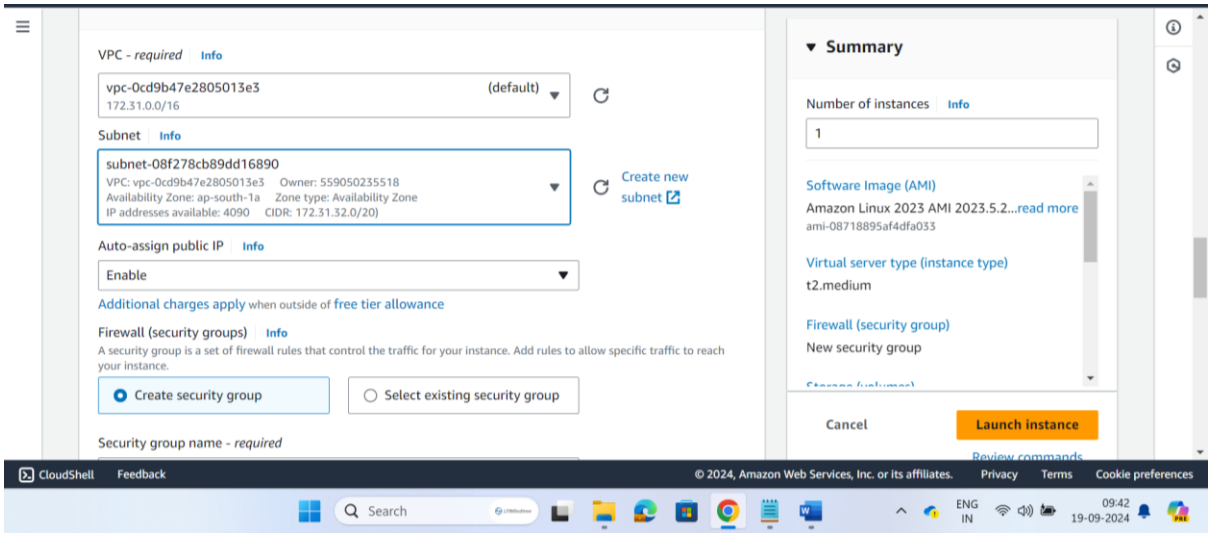


Select t2medium in jenkin machine ,t2micro for Developer and tomcat machine:

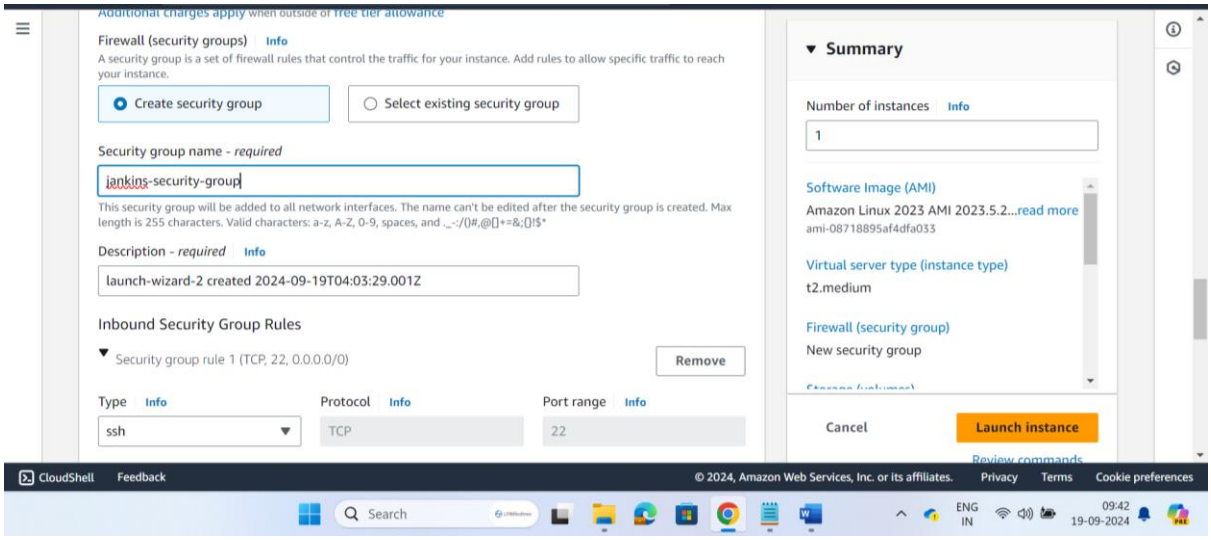


Take same key pair for all the Machine:

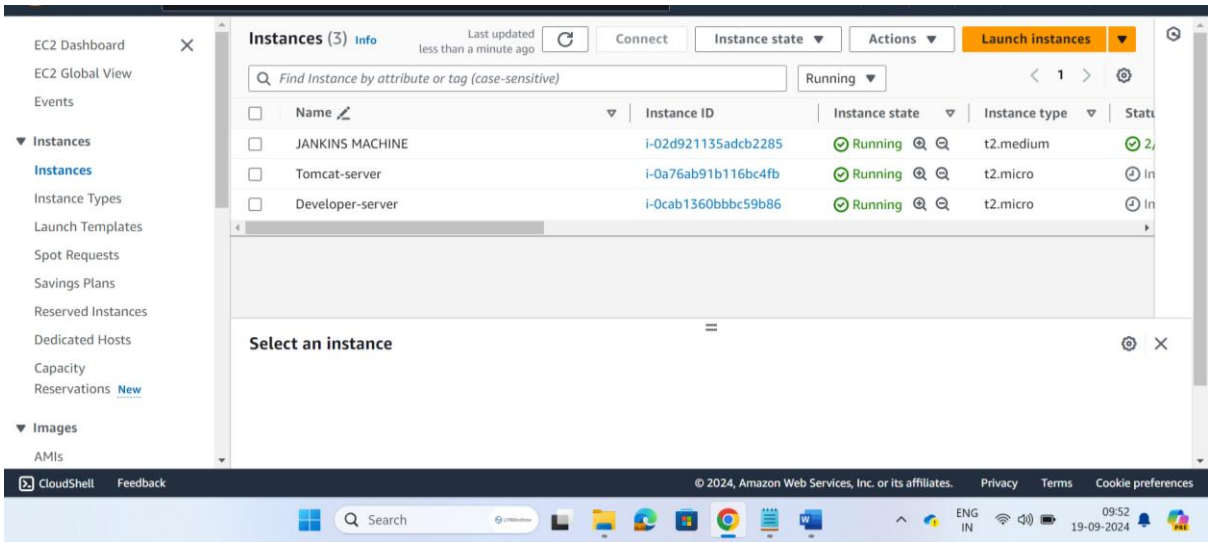




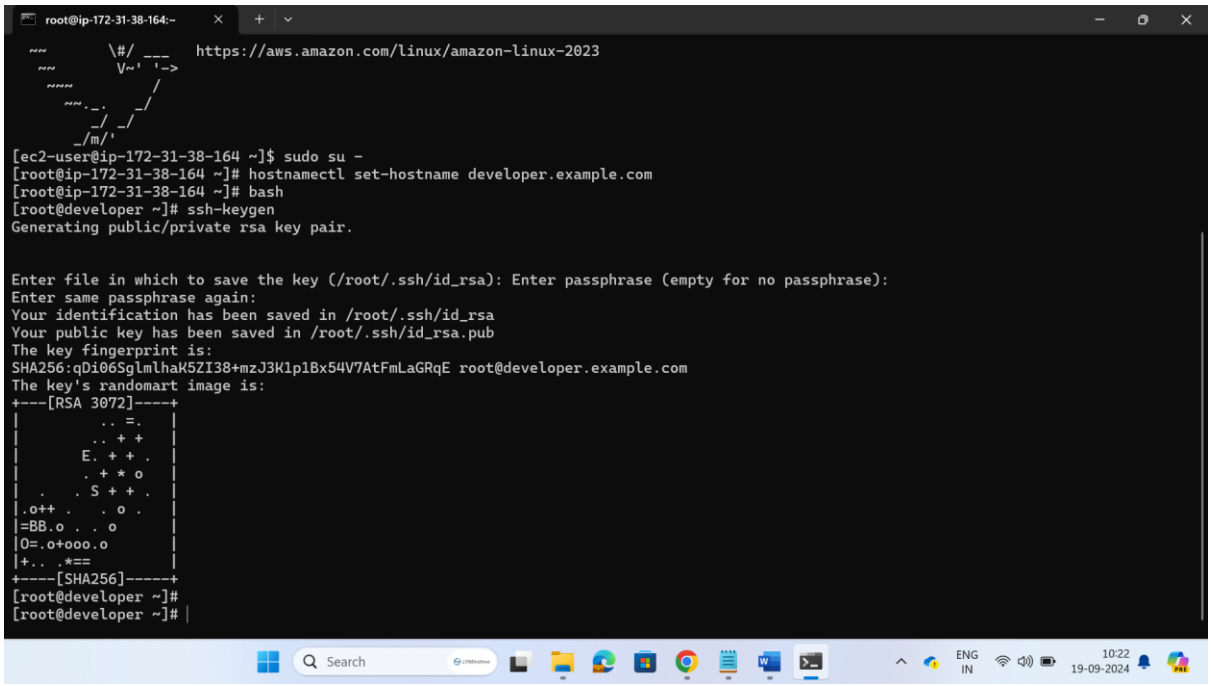
Created Security group and exposed port 8080 :



Created three instances for Developer,Jenkins and Tomcat:

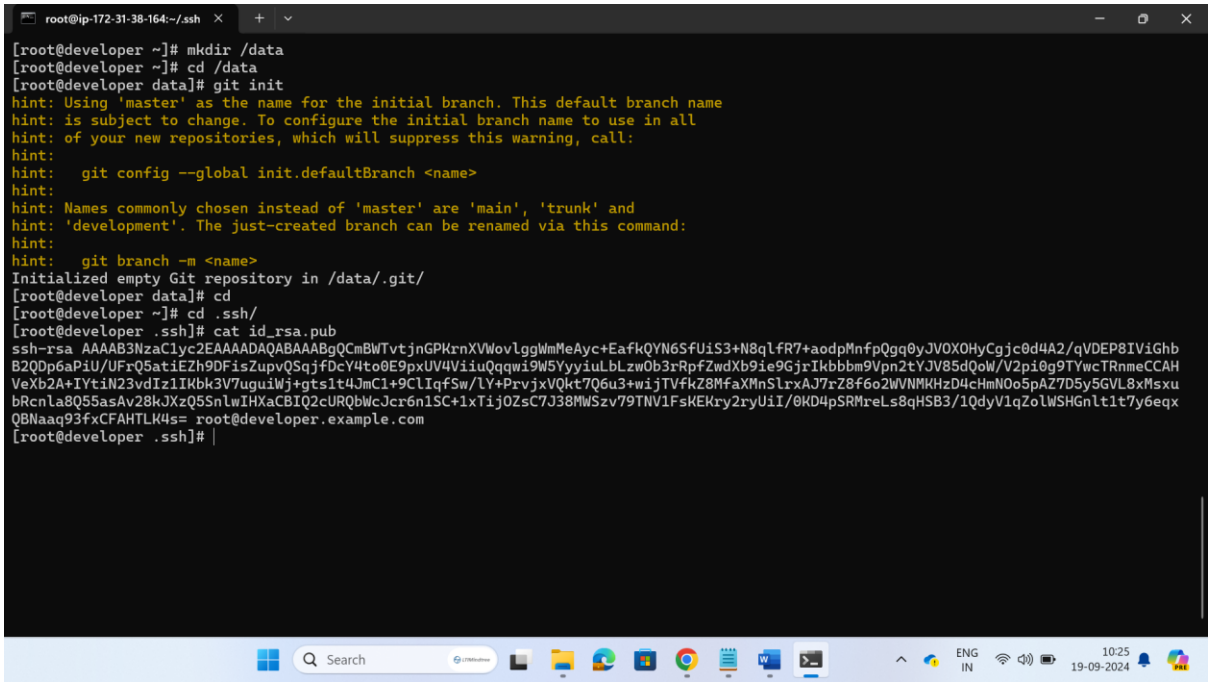


Created ssh-keygen

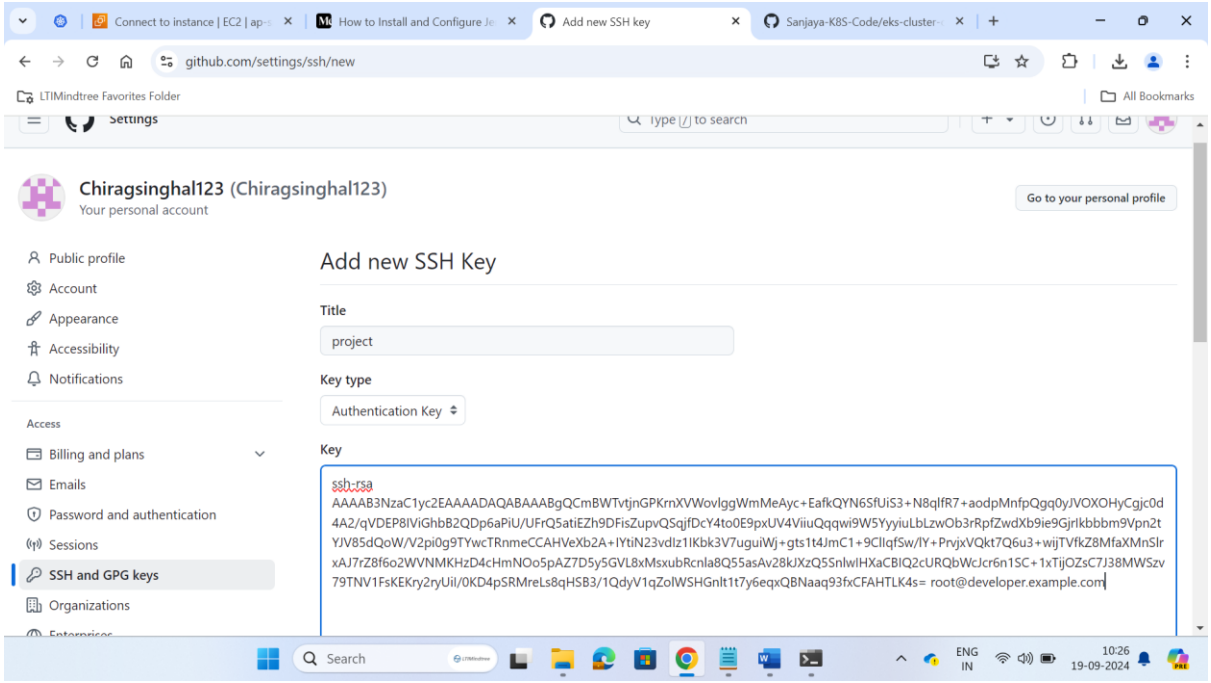


Install git

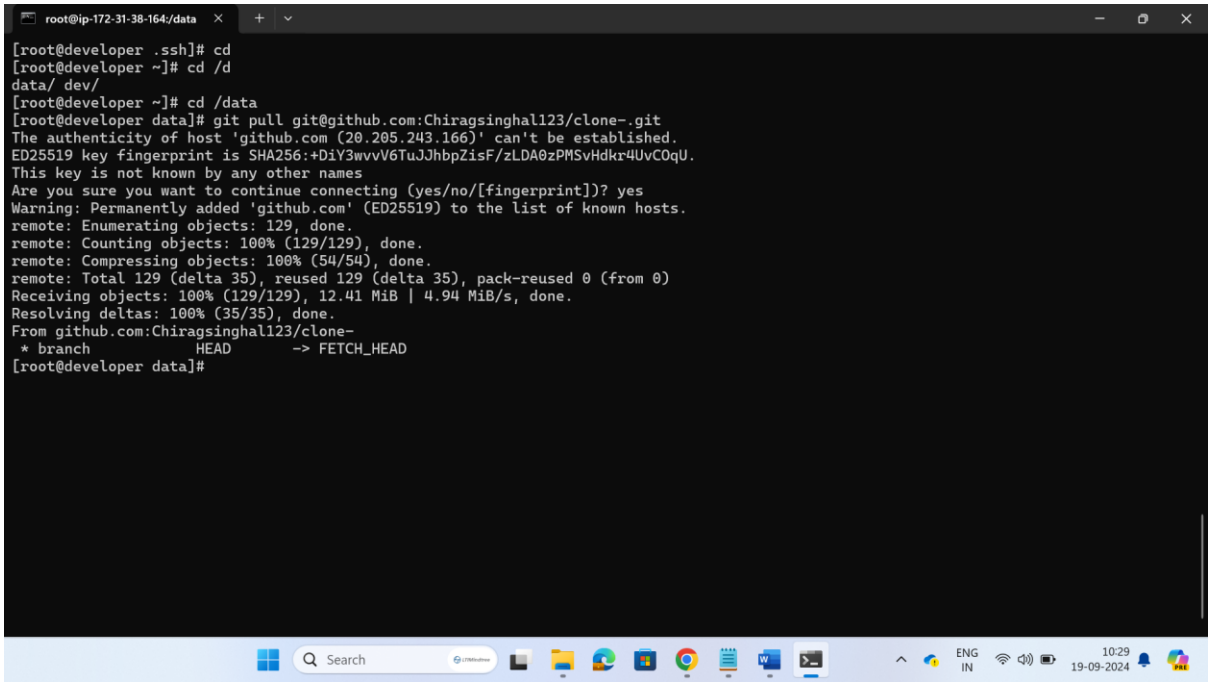
Create directory and create public key

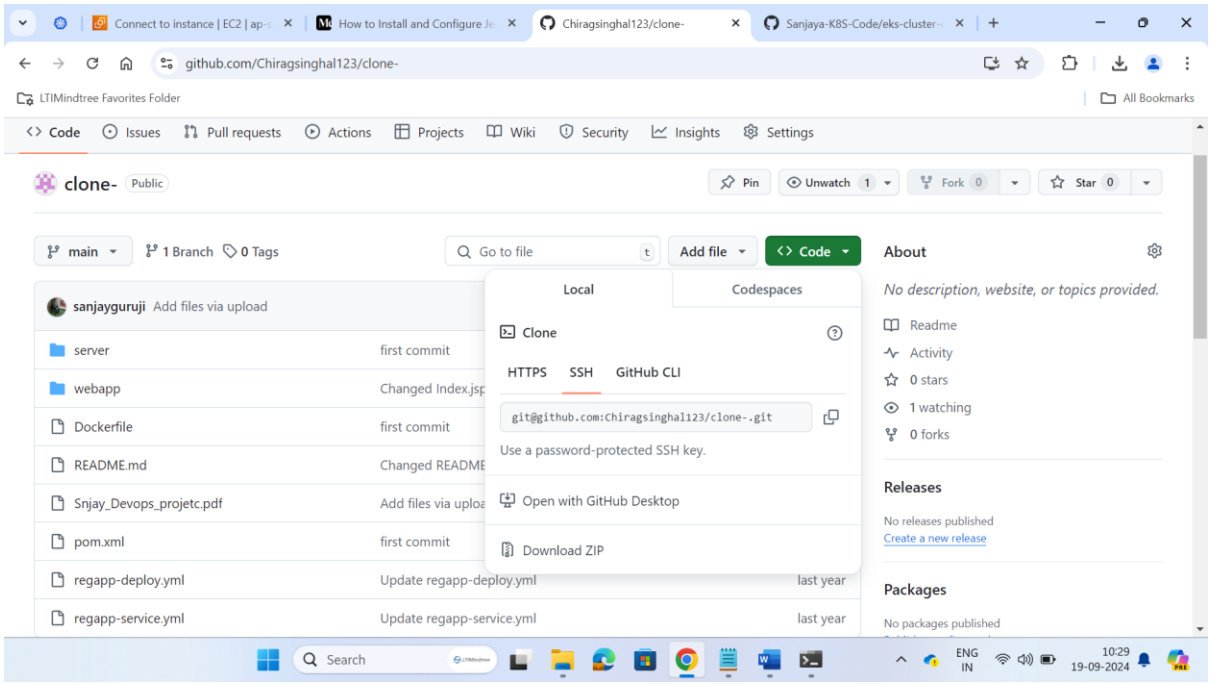


Copy jenkin machine public key and paste in github



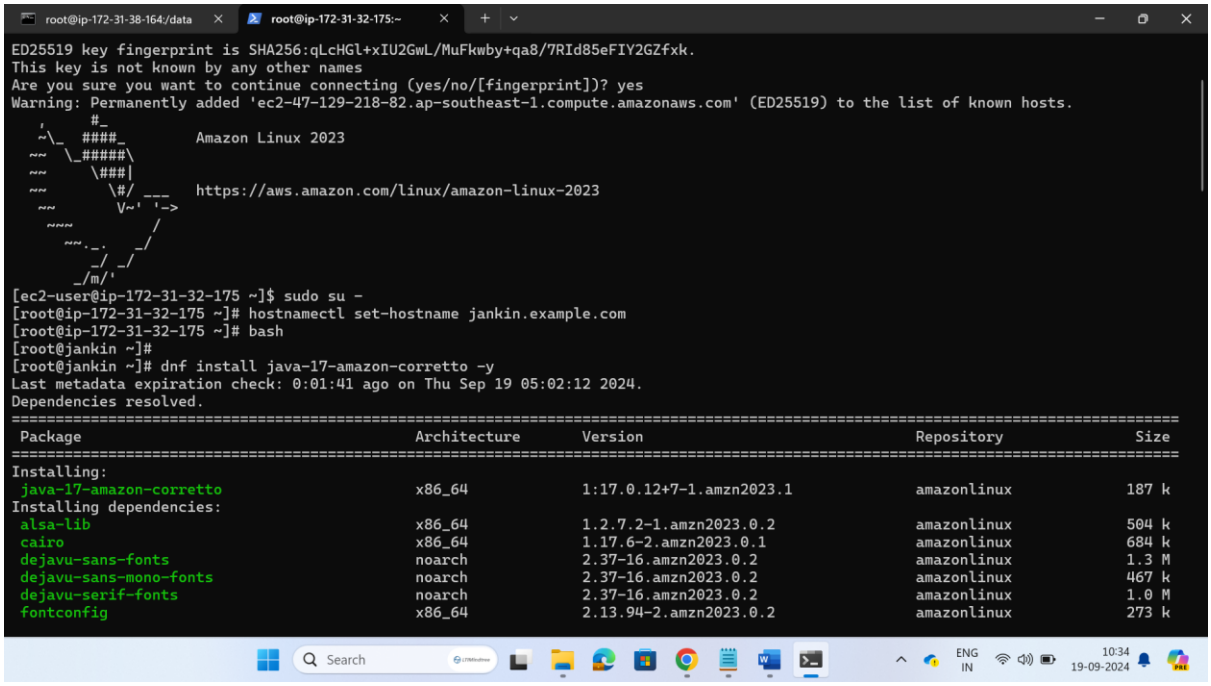
Git pull java application in my developer machine by git ssh



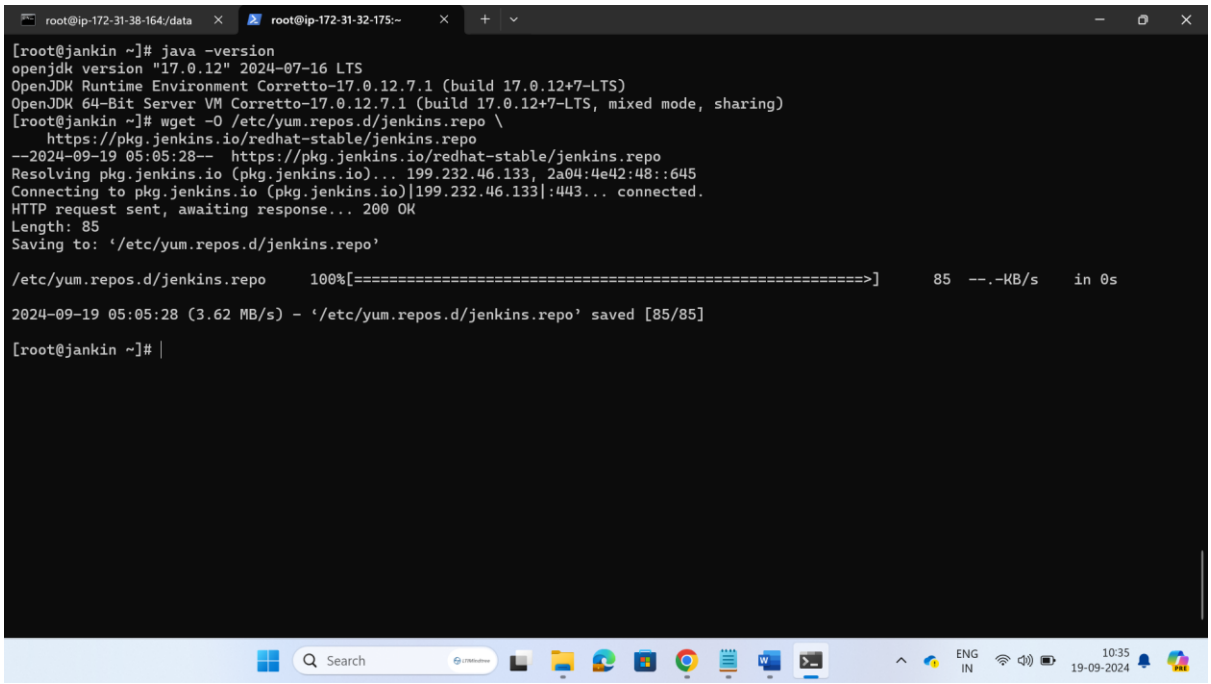


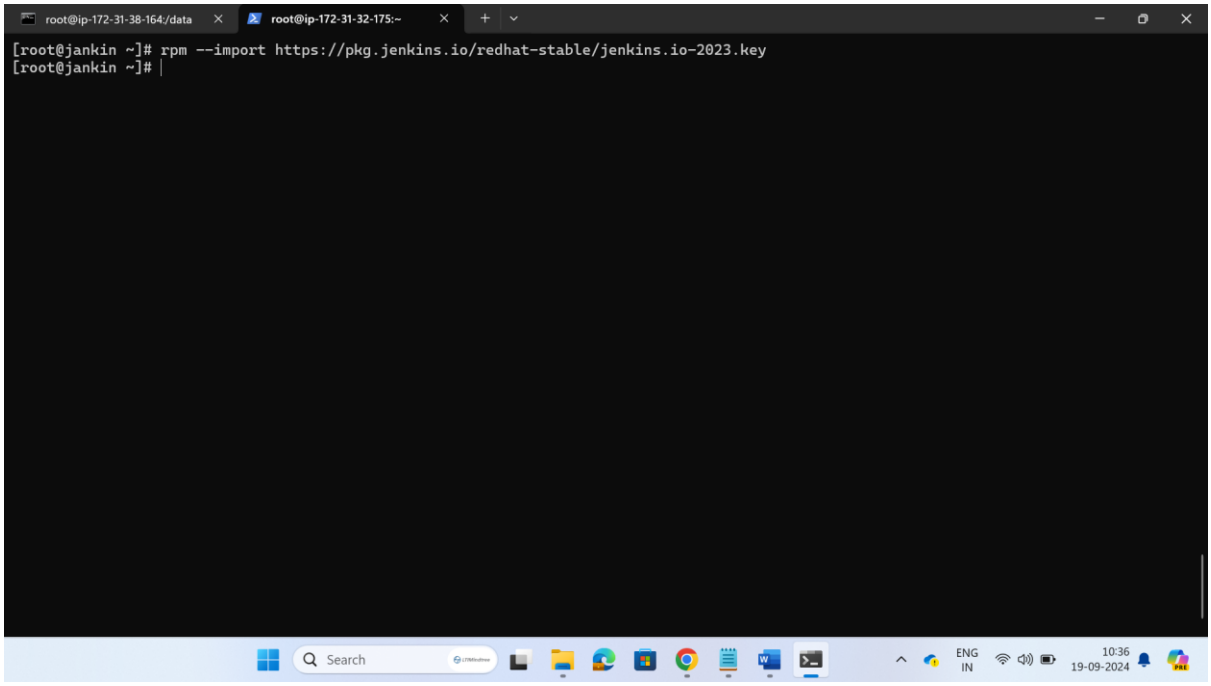
Change branch name

In jenkin machine change host name and install JAVA

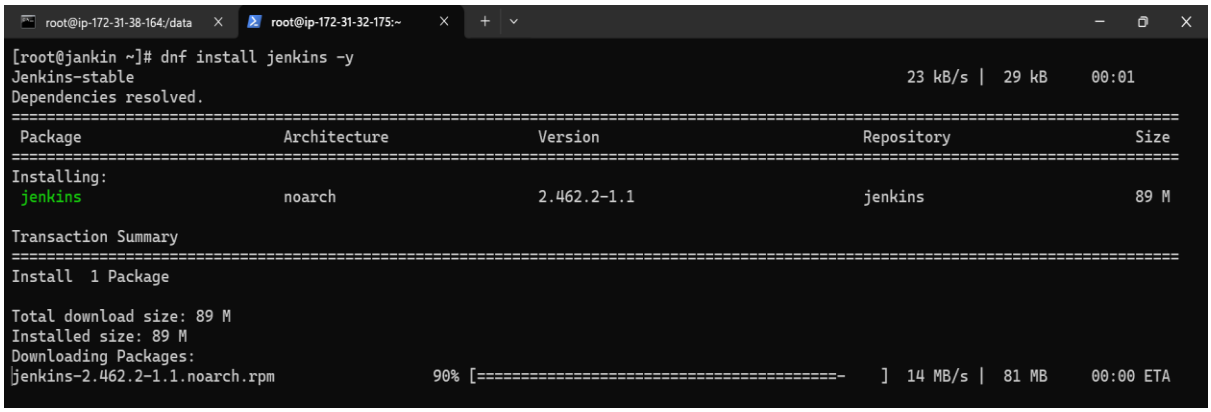


Check java version



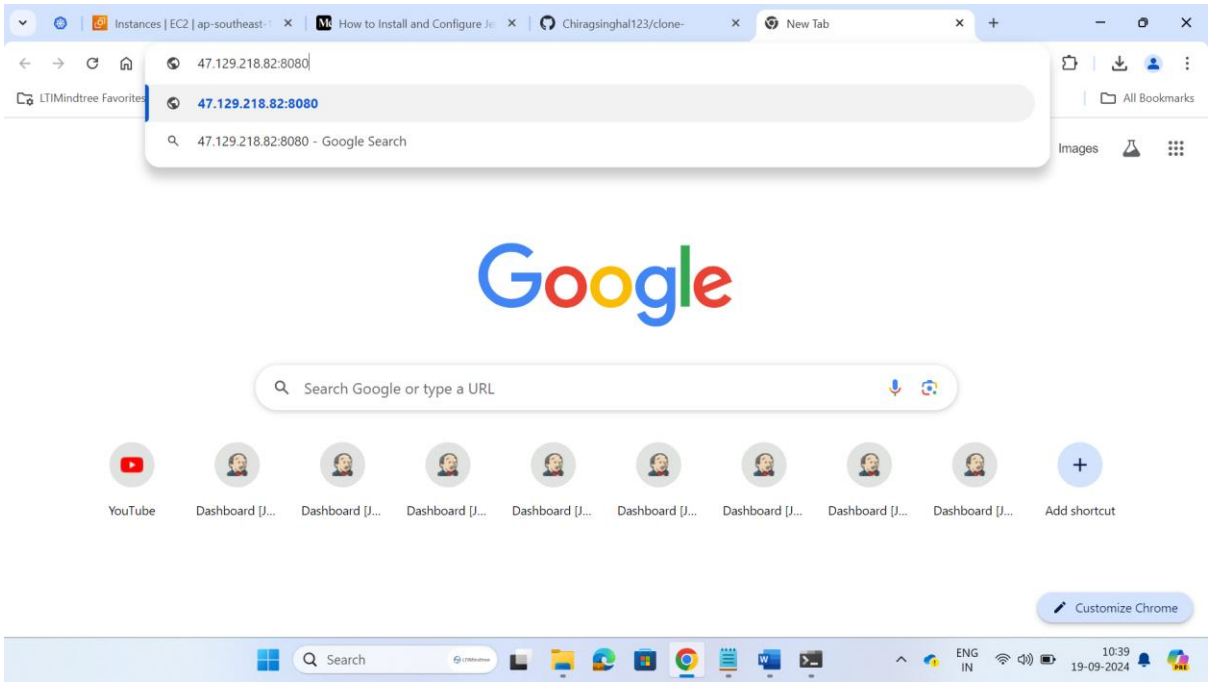


Install jenkins and java in jenkin machine

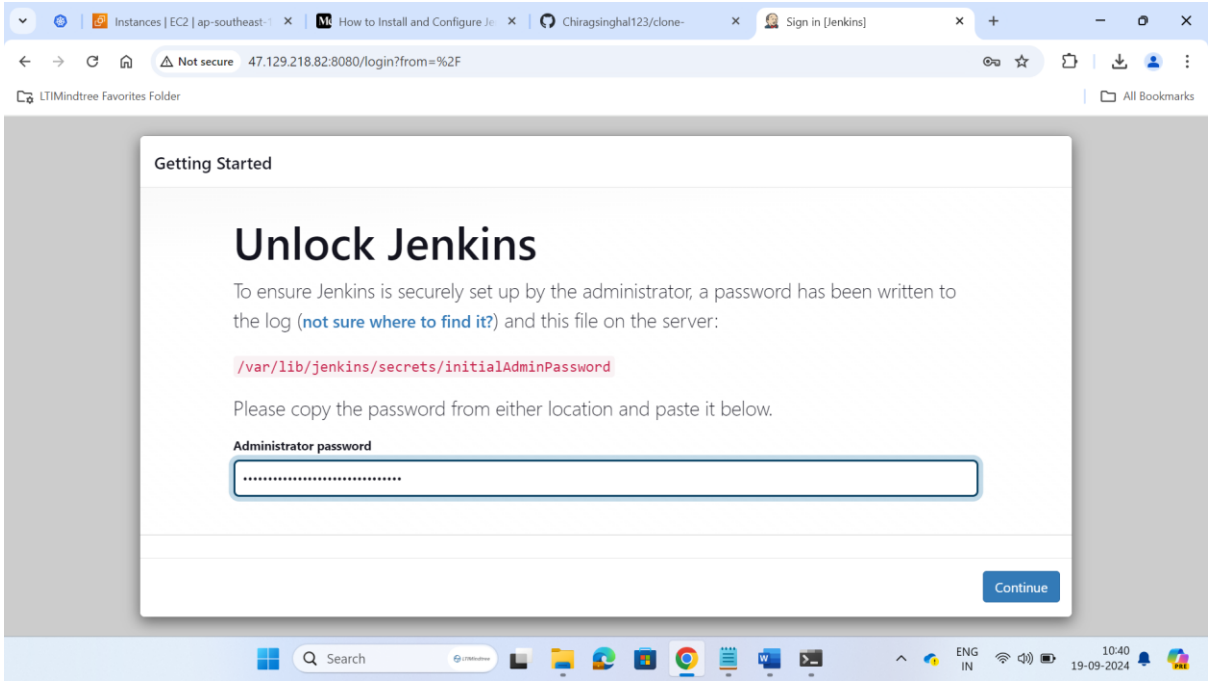


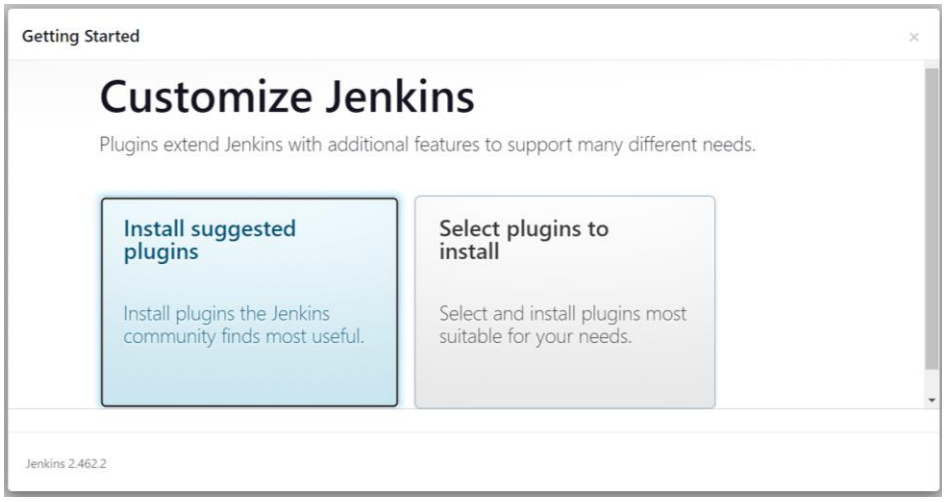
Jenkin machine is start and enable:

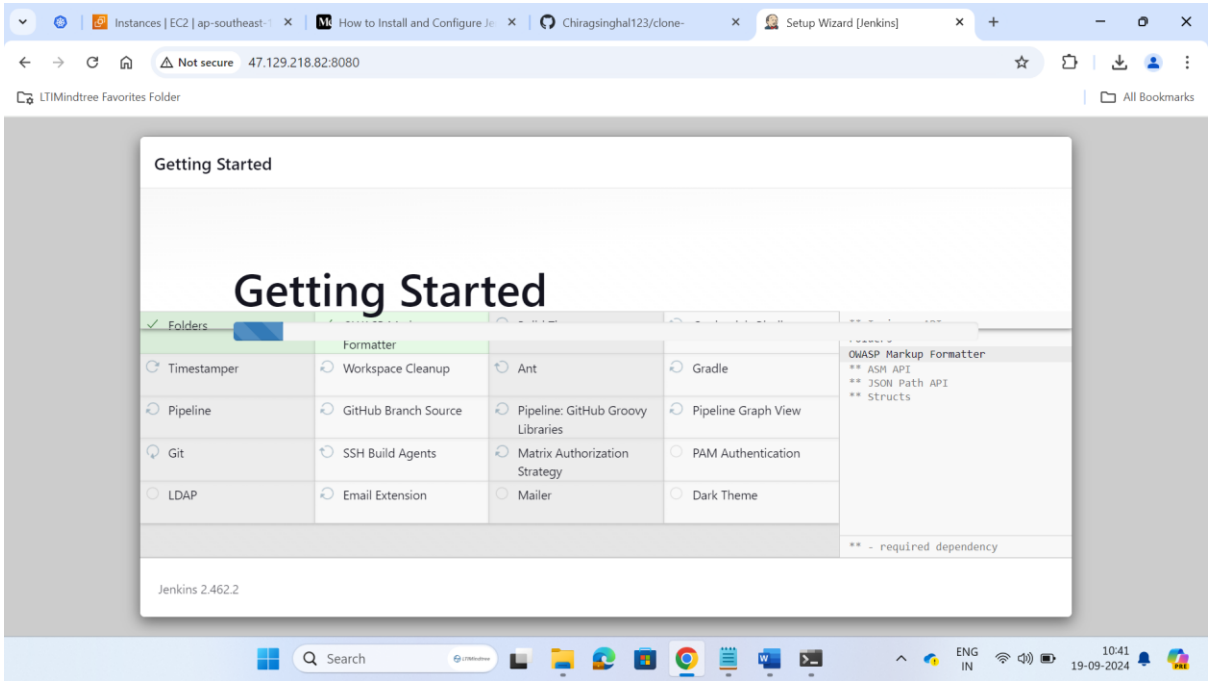
Paste jankin public ip and add :8080



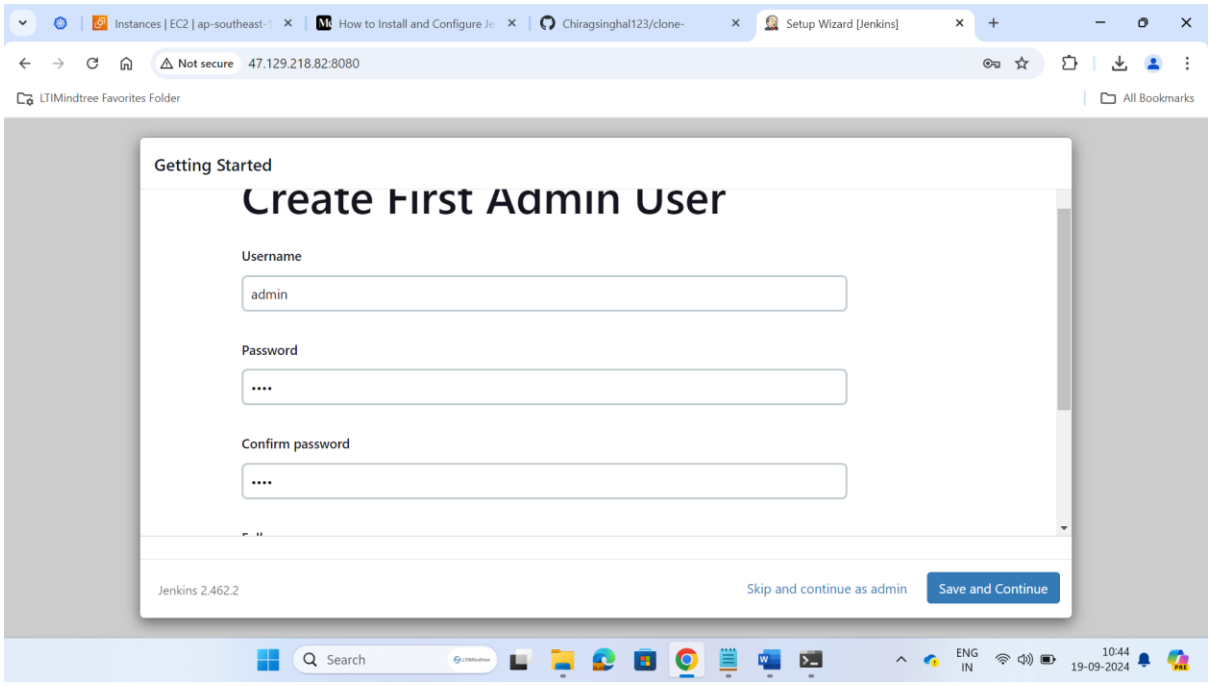
Open Jenkin by password



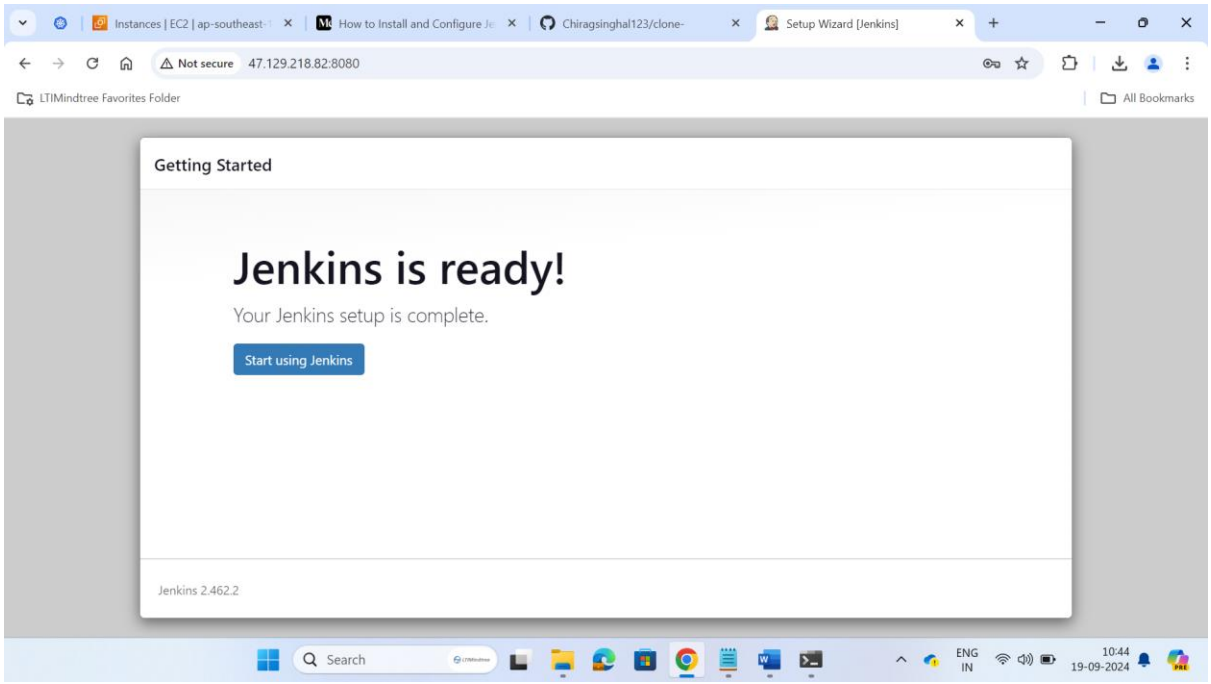

Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.462.2


Instances | EC2 | ap-southeast-1
How to Install and Configure Je
Chiragsinghal123/clone-
Setup Wizard [Jenkins]
Not secure
47.129.218.82:8080
LTIMindtree Favorites Folder
All Bookmarks
Getting Started
Getting Started
Folders
Formatter
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline Graph View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
Dark Theme
OWASP Markup Formatter
** ASM API
** JSON Path API
** Structs
** - required dependency
Jenkins 2.462.2
Search
ENG
IN
10:41
19-09-2024

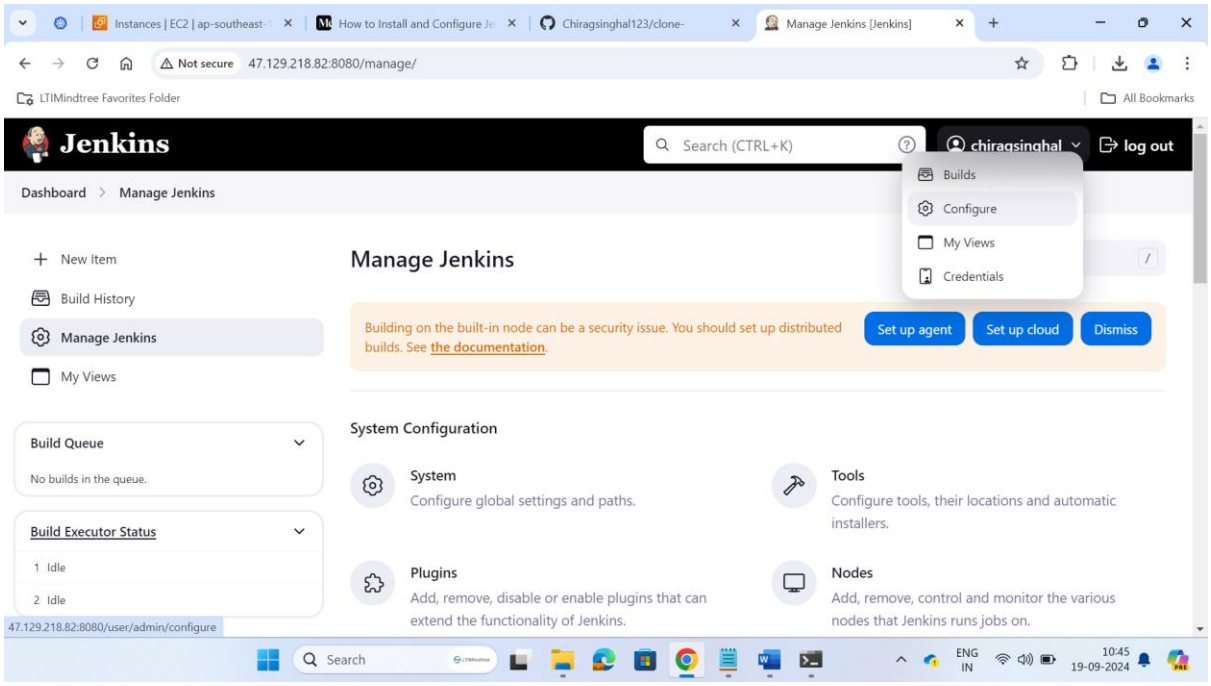
Add username,password and emailid

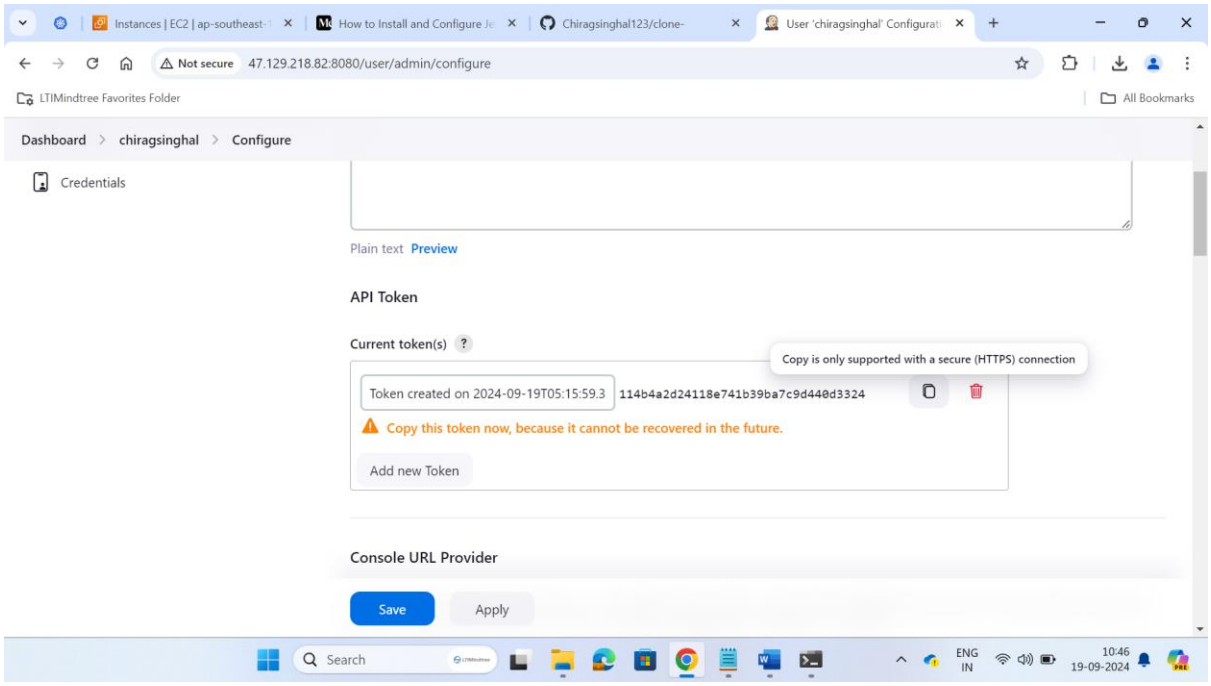


Jenkins server is ready

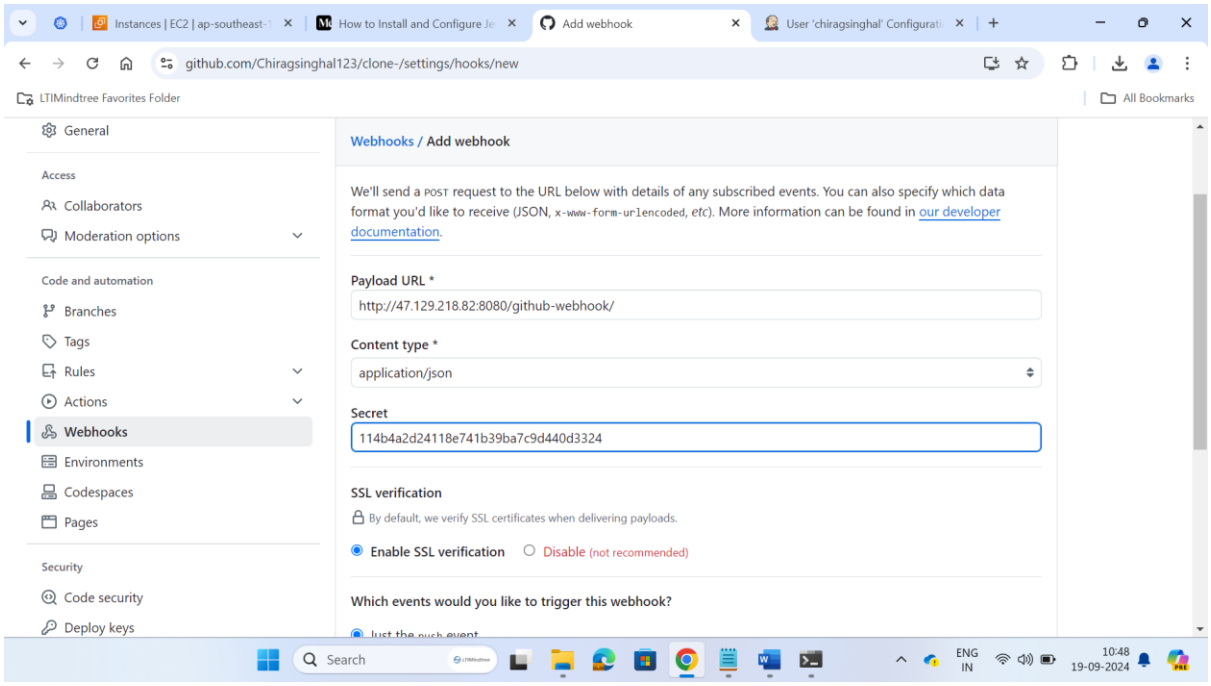


In Jenkin server select profile then select Configure and Generate Token then paste in githubs Webhook:

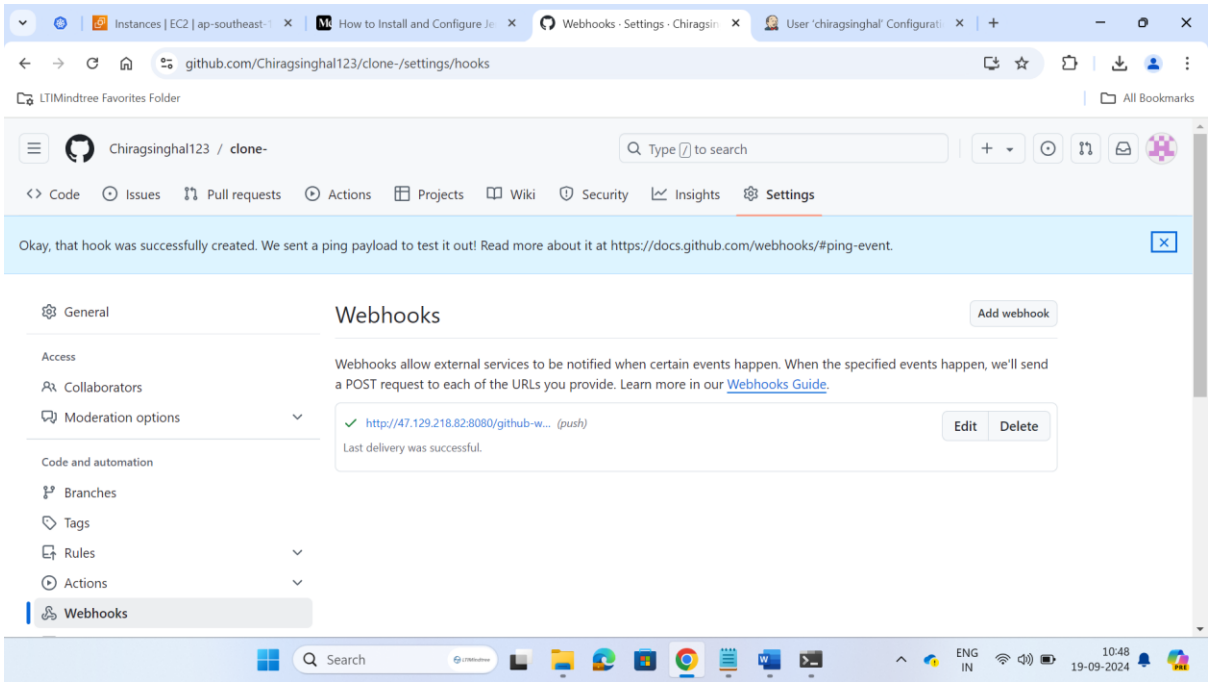




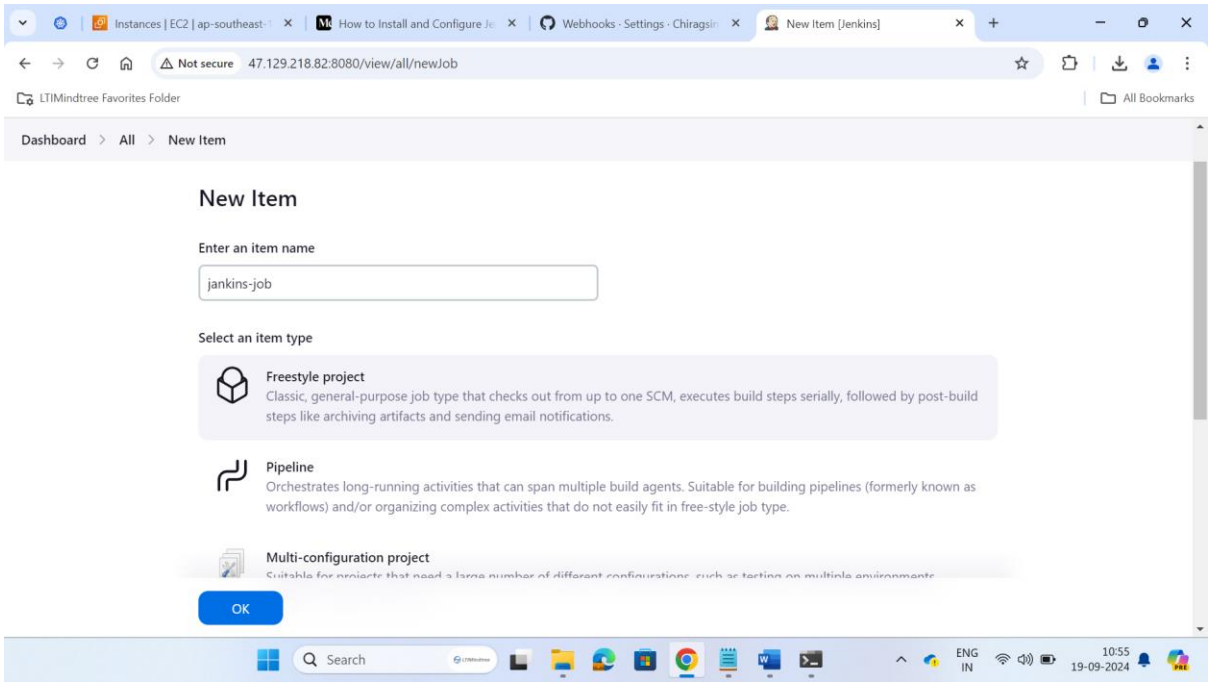
Paste Generate code in secret key

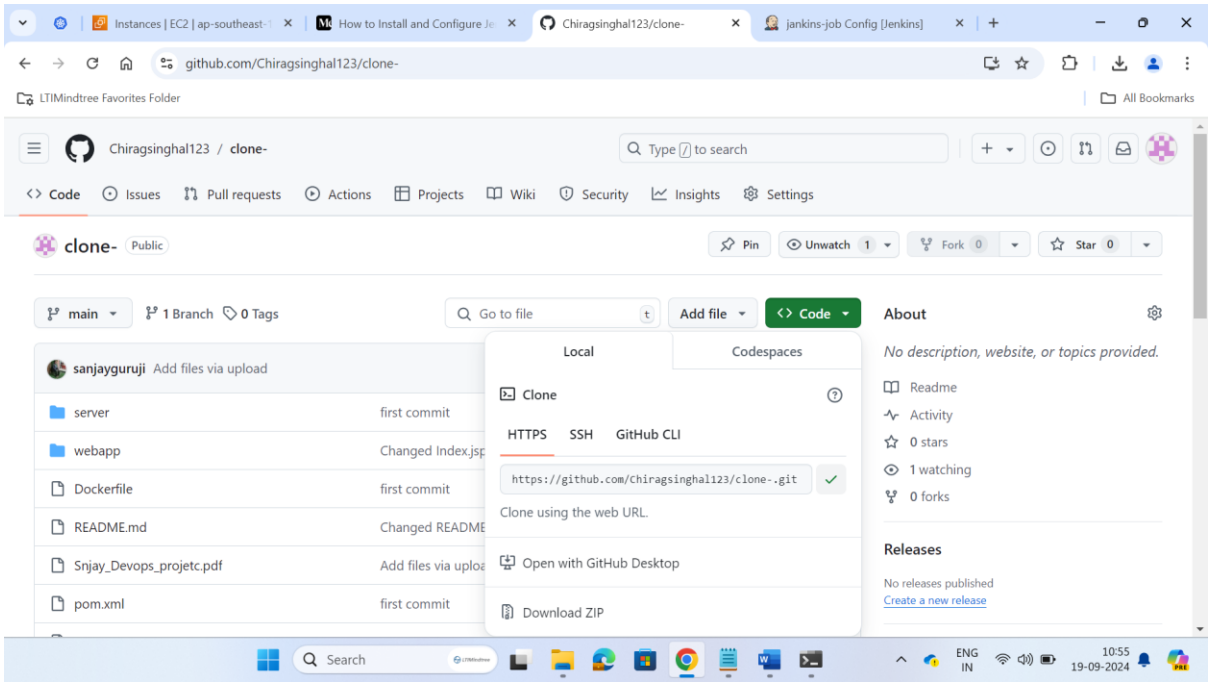


webhooks is created Successfully

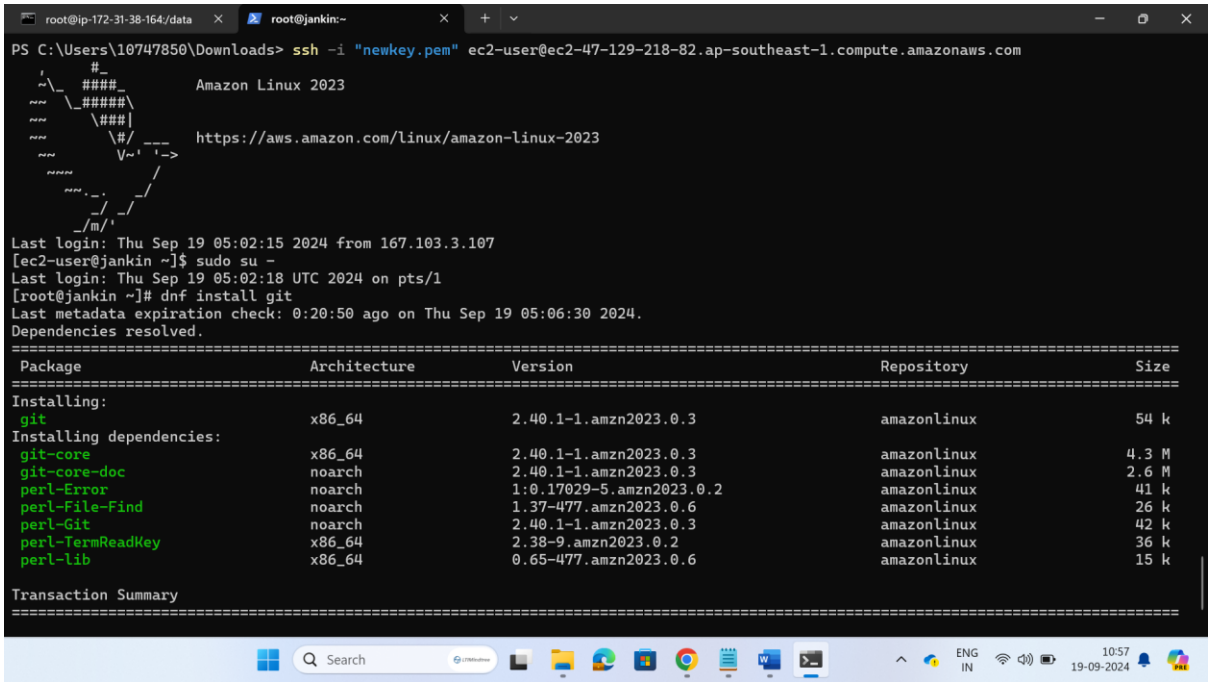


Open jenkin server click new item then select Freestyle project and check jenkin job is build or not

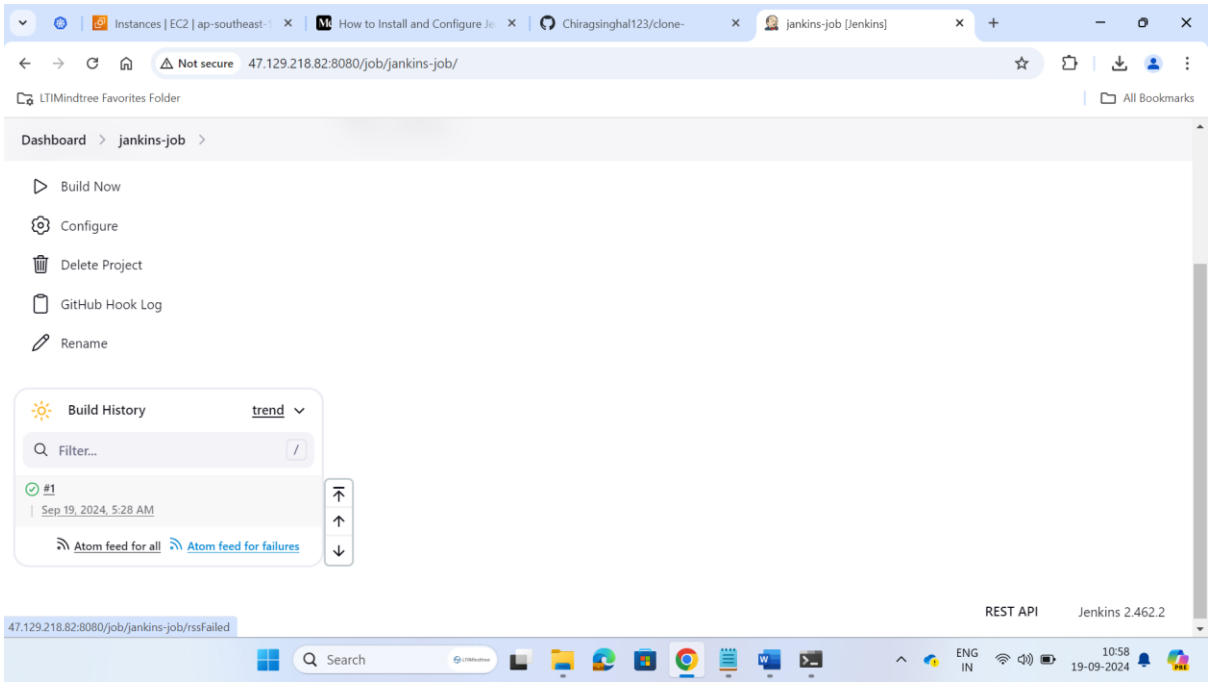




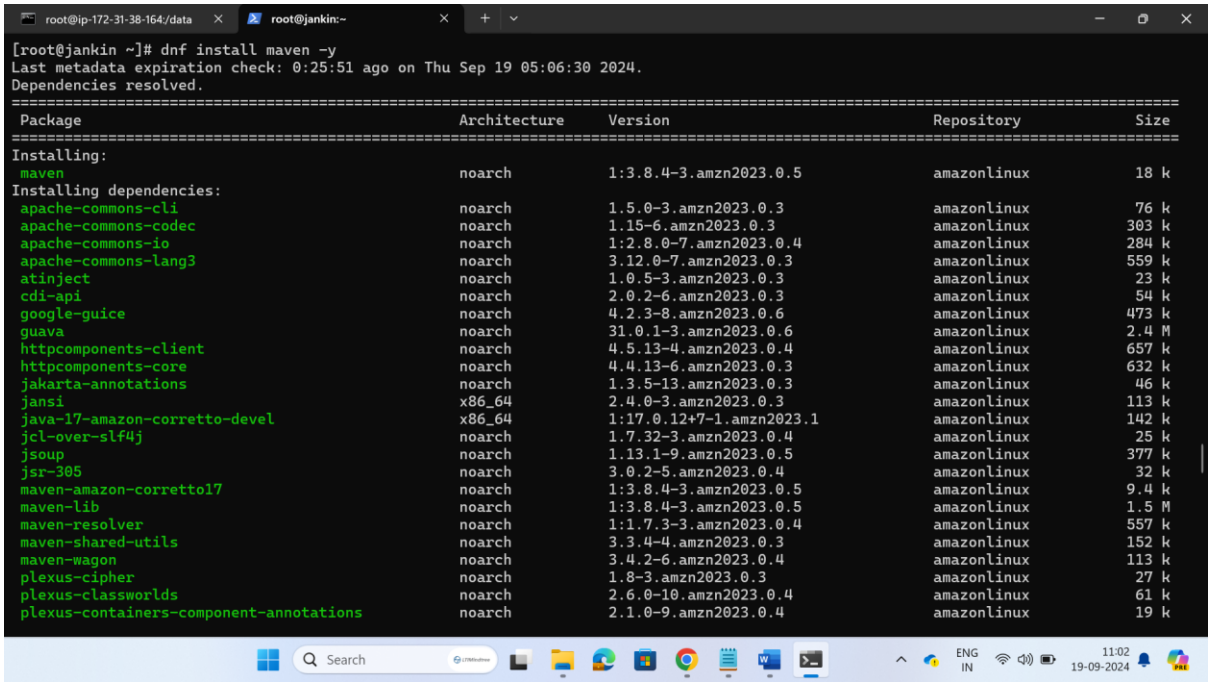
In jenkins machine install git



Jenkin job is completed



Jenkin Machine Install maven



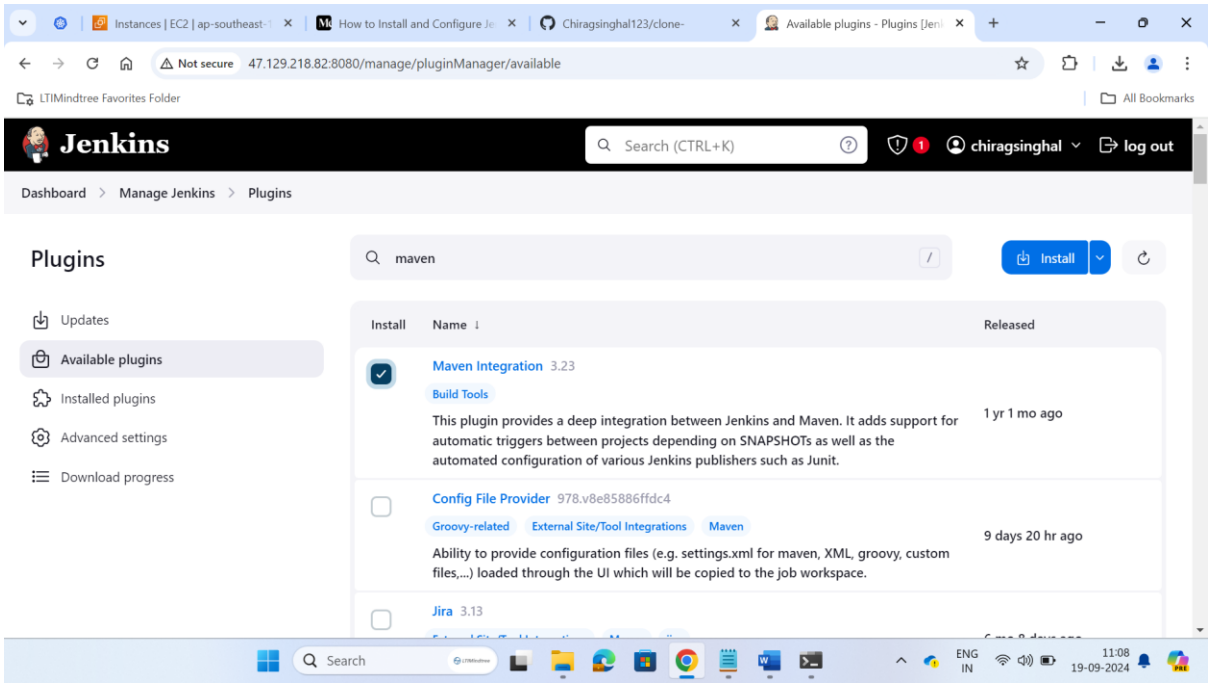
```
[root@jankin ~]# dnf install maven -y
Last metadata expiration check: 0:25:51 ago on Thu Sep 19 05:06:30 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
maven	noarch	1:3.8.4-3.amzn2023.0.5	amazonlinux	18 k
Installing dependencies:				
apache-commons-cli	noarch	1.5.0-3.amzn2023.0.3	amazonlinux	76 k
apache-commons-codec	noarch	1.15-6.amzn2023.0.3	amazonlinux	303 k
apache-commons-io	noarch	1:2.8.0-7.amzn2023.0.4	amazonlinux	284 k
apache-commons-lang3	noarch	3.12.0-7.amzn2023.0.3	amazonlinux	559 k
atinject	noarch	1.0.5-3.amzn2023.0.3	amazonlinux	23 k
cdi-api	noarch	2.0.2-6.amzn2023.0.3	amazonlinux	54 k
google-guice	noarch	4.2.3-8.amzn2023.0.6	amazonlinux	473 k
guava	noarch	31.0.1-3.amzn2023.0.6	amazonlinux	2.4 M
httpcomponents-client	noarch	4.5.13-4.amzn2023.0.4	amazonlinux	657 k
httpcomponents-core	noarch	4.4.13-6.amzn2023.0.3	amazonlinux	632 k
jakarta-annotations	noarch	1.3.5-13.amzn2023.0.3	amazonlinux	46 k
jansi	x86_64	2.4.0-3.amzn2023.0.3	amazonlinux	113 k
java-17-amazon-corretto-devel	x86_64	1:17.0.12+7-1.amzn2023.1	amazonlinux	142 k
jcl-over-slf4j	noarch	1.7.32-3.amzn2023.0.4	amazonlinux	25 k
jsoup	noarch	1.13.1-9.amzn2023.0.5	amazonlinux	377 k
jsr-305	noarch	3.0.2-5.amzn2023.0.4	amazonlinux	32 k
maven-amazon-corretto17	noarch	1:3.8.4-3.amzn2023.0.5	amazonlinux	9.4 k
maven-lib	noarch	1:3.8.4-3.amzn2023.0.5	amazonlinux	1.5 M
maven-resolver	noarch	1:1.7.3-3.amzn2023.0.4	amazonlinux	557 k
maven-shared-utils	noarch	3.3.4-4.amzn2023.0.3	amazonlinux	152 k
maven-wagon	noarch	3.4.2-6.amzn2023.0.4	amazonlinux	113 k
plexus-cipher	noarch	1.8-3.amzn2023.0.3	amazonlinux	27 k
plexus-classworlds	noarch	2.6.0-10.amzn2023.0.4	amazonlinux	61 k
plexus-containers-component-annotations	noarch	2.1.0-9.amzn2023.0.4	amazonlinux	19 k

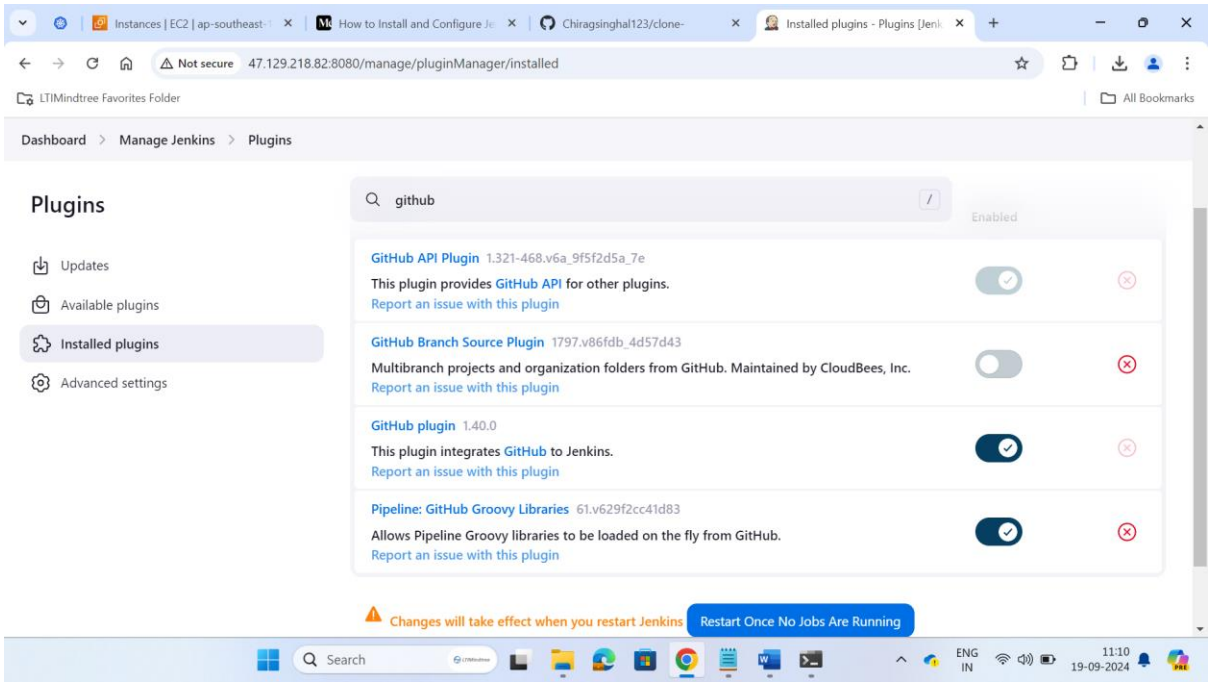
See java version and maven version

```
[root@jankin ~]# mvn -v
Apache Maven 3.8.4 (Red Hat 3.8.4-3.amzn2023.0.5)
Maven home: /usr/share/maven
Java version: 17.0.12, vendor: Amazon.com Inc., runtime: /usr/lib/jvm/java-17-amazon-corretto.x86_64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.1.109-118.189.amzn2023.x86_64", arch: "amd64", family: "unix"
[root@jankin ~]#
```

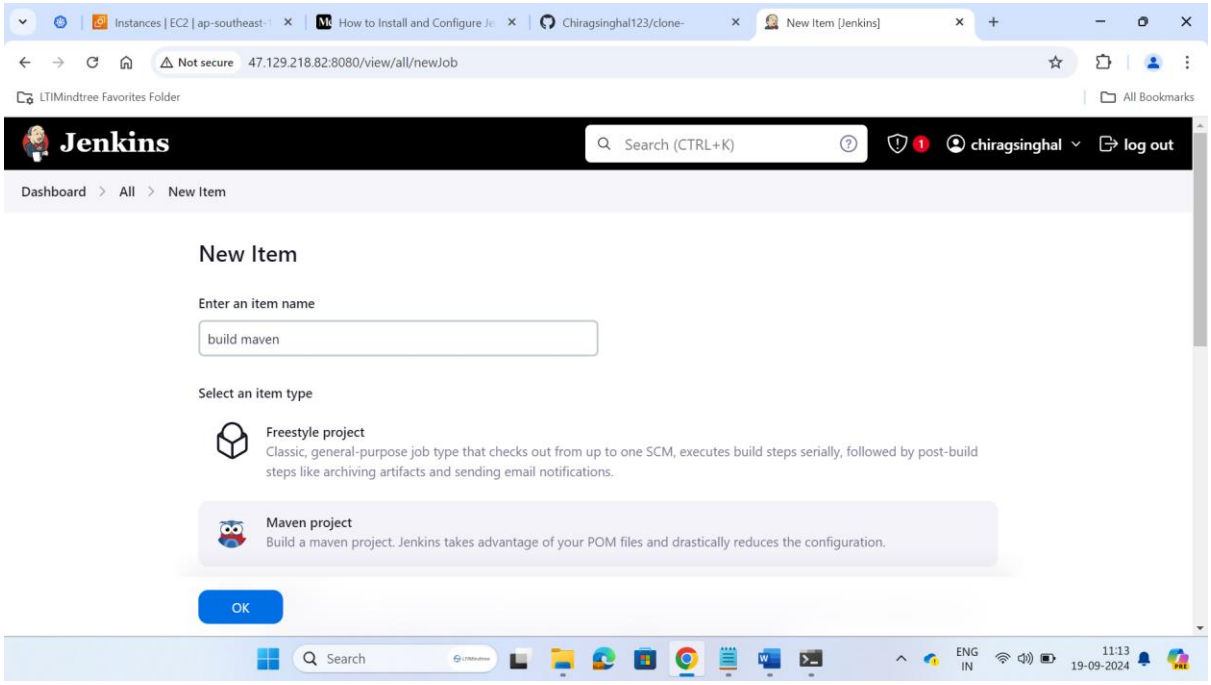
Click manage Jenkins then click plugins and install maven Integeration



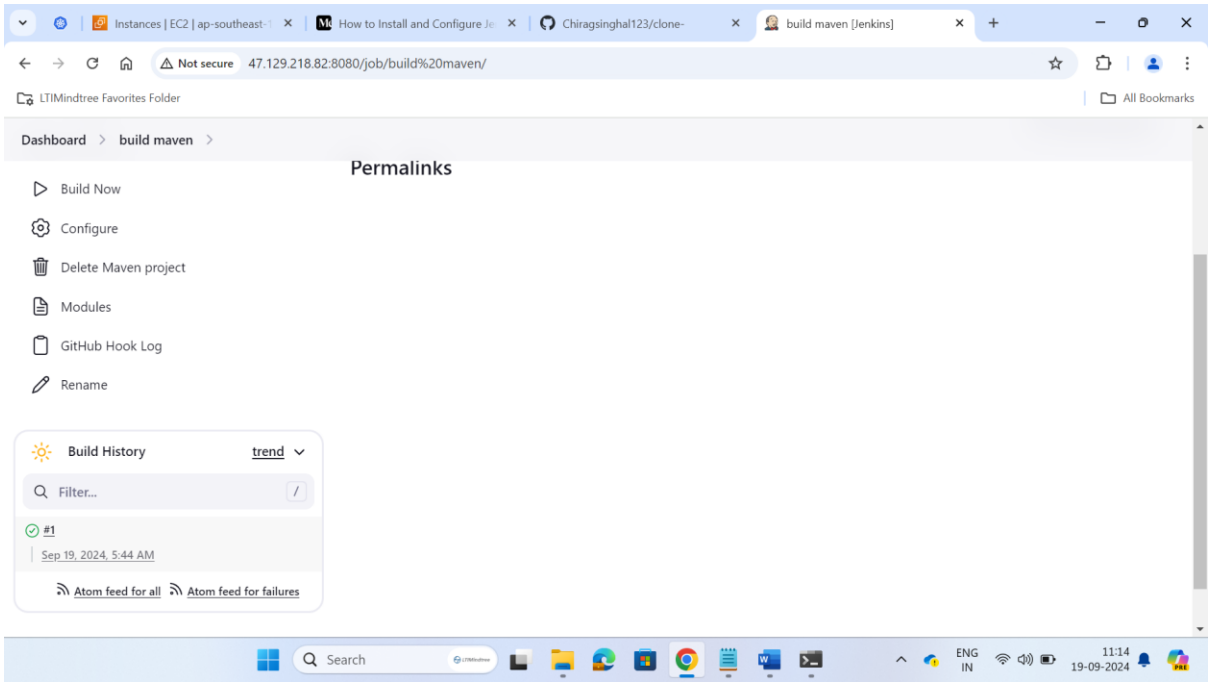
Search github and turn off github branch source plugin then automatic turn on GitHub plugin



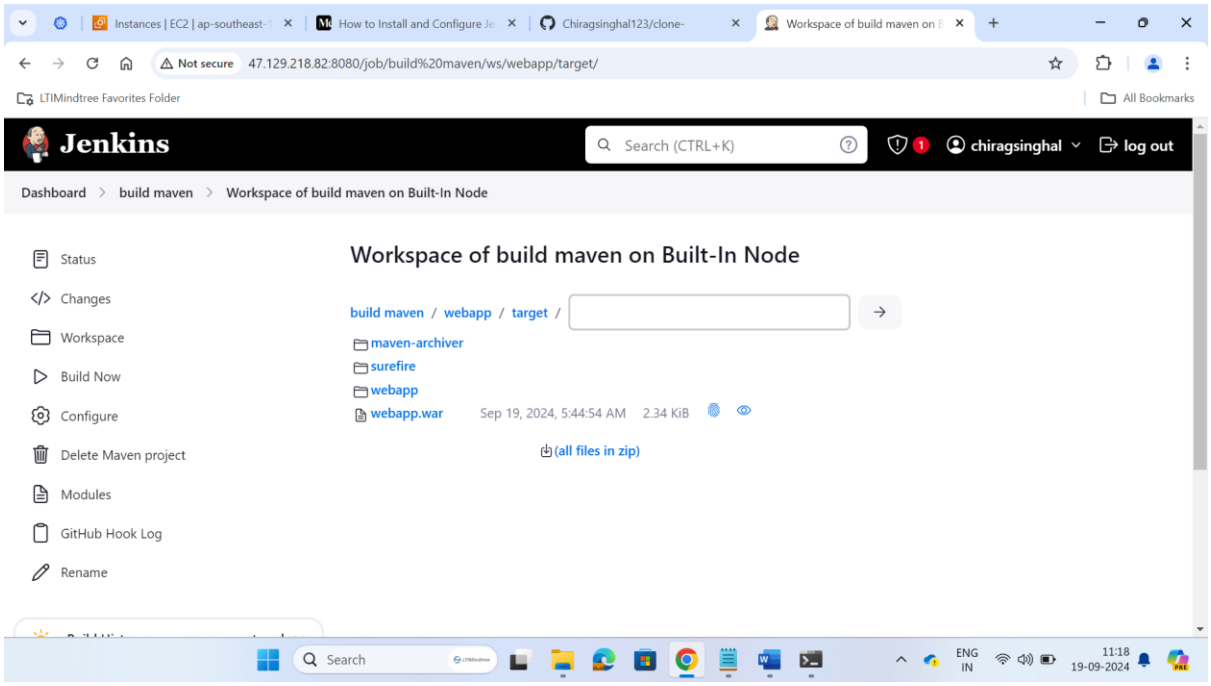
Select New Item click maven project then ok



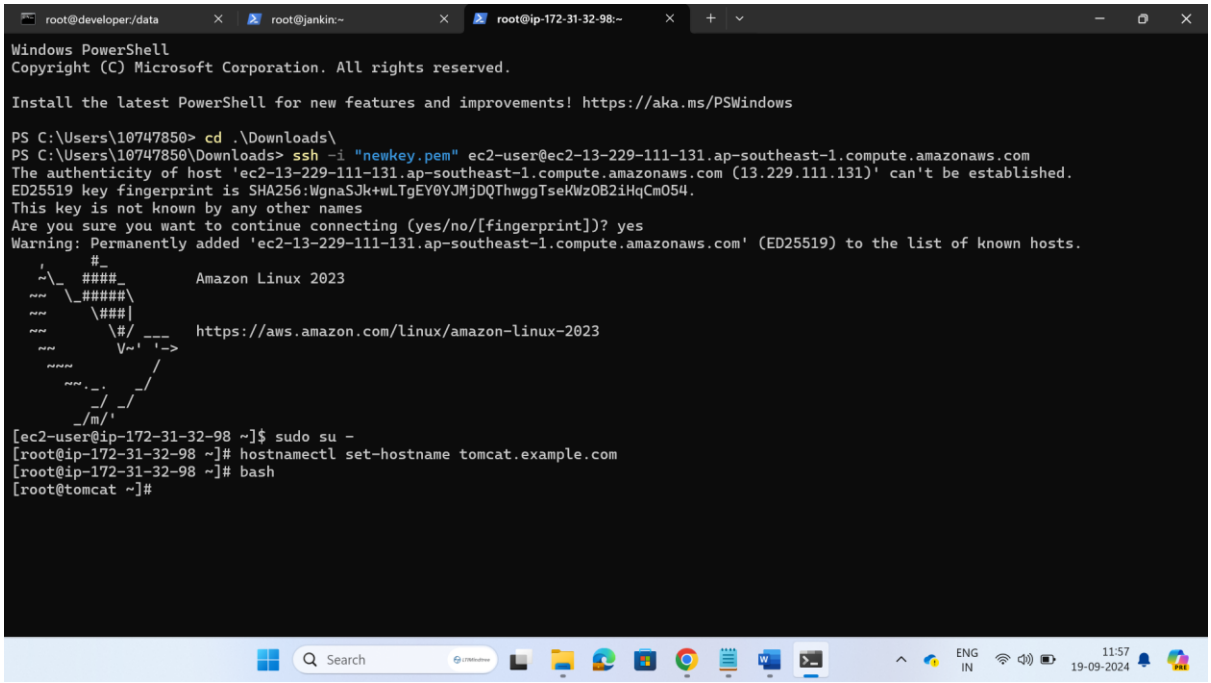
My maven project Is Successfully Build

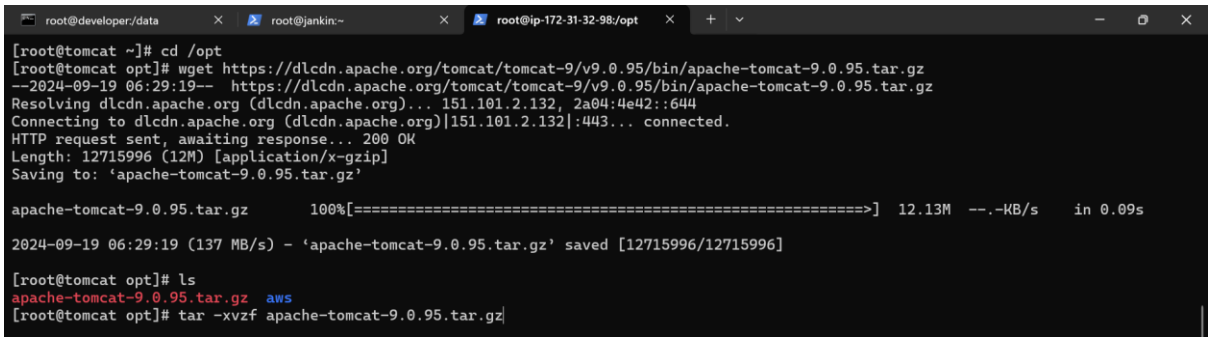


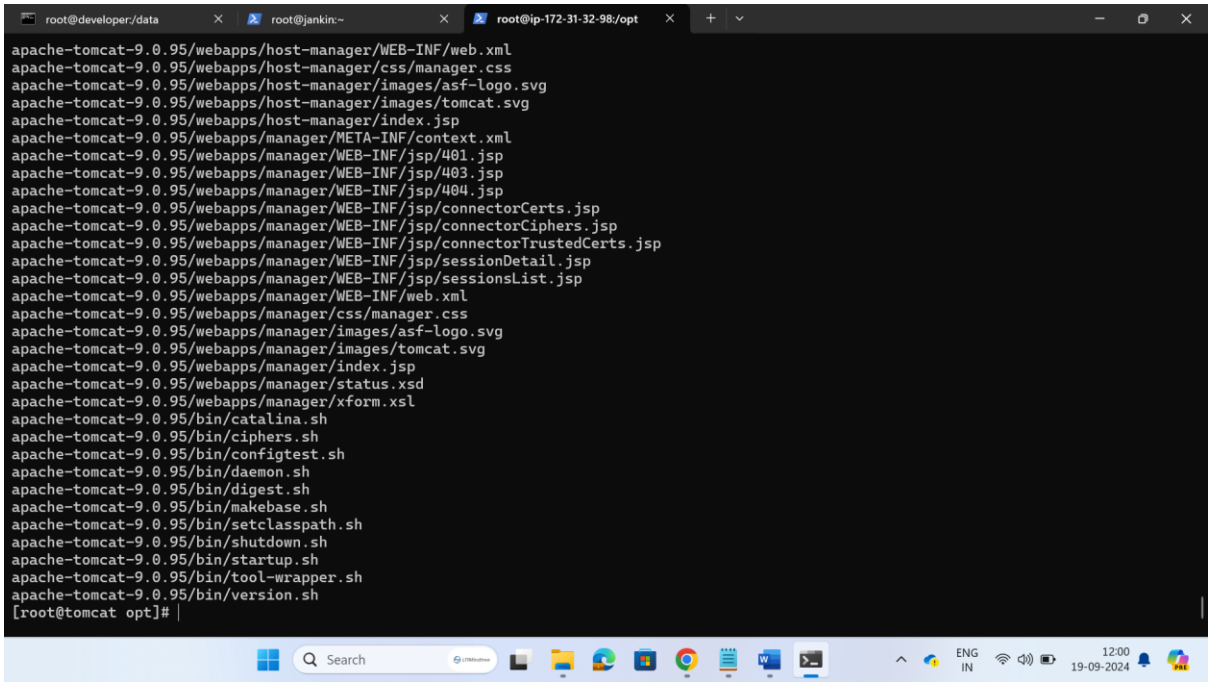
Show Webapp.war file

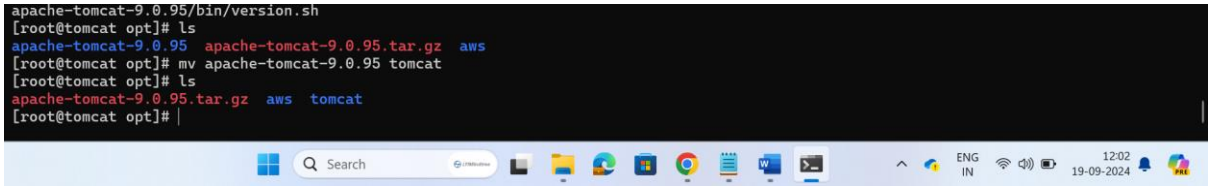


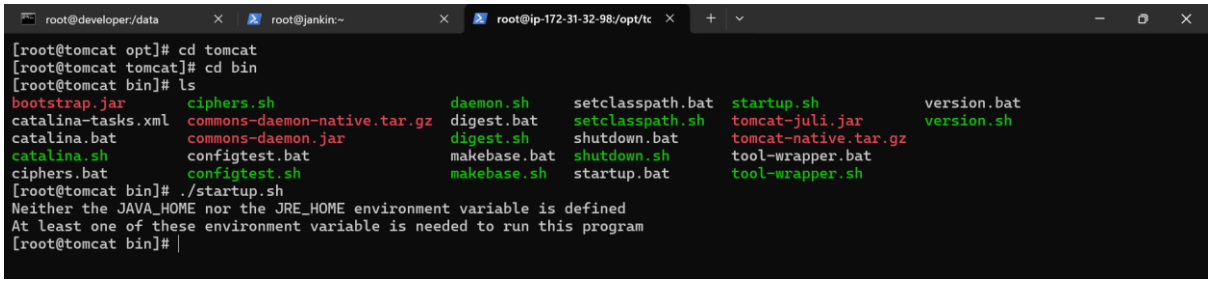
Start Tomcat Machine



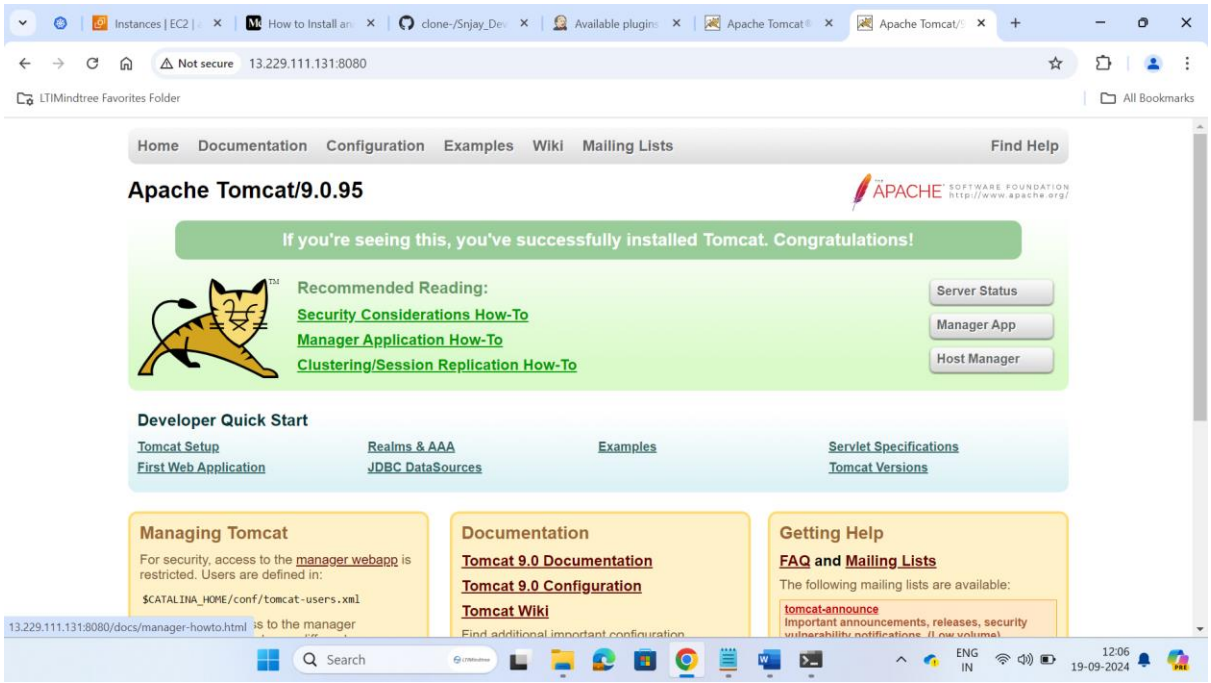


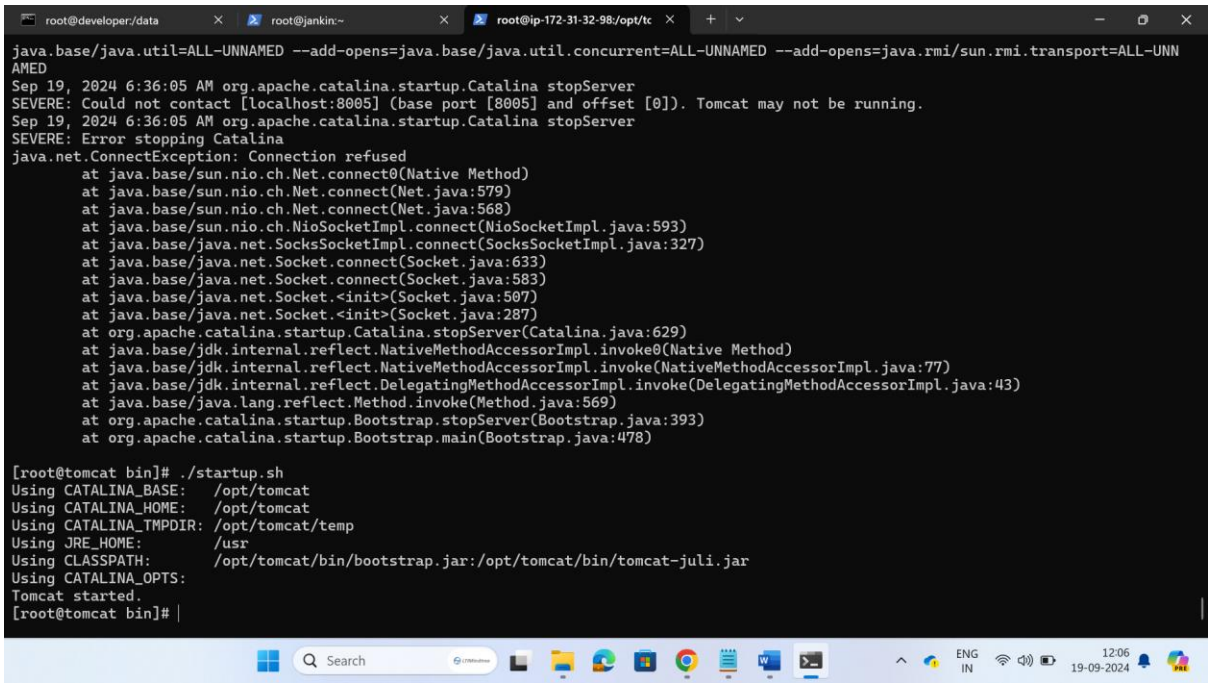

root@developer:/data
root@jankin:~
root@ip-172-31-32-98:/opt
apache-tomcat-9.0.95/webapps/host-manager/WEB-INF/web.xml
apache-tomcat-9.0.95/webapps/host-manager/css/manager.css
apache-tomcat-9.0.95/webapps/host-manager/images/asf-logo.svg
apache-tomcat-9.0.95/webapps/host-manager/images/tomcat.svg
apache-tomcat-9.0.95/webapps/host-manager/index.jsp
apache-tomcat-9.0.95/webapps/manager/META-INF/context.xml
apache-tomcat-9.0.95/webapps/manager/WEB-INF/jsp/401.jsp
apache-tomcat-9.0.95/webapps/manager/WEB-INF/jsp/403.jsp
apache-tomcat-9.0.95/webapps/manager/WEB-INF/jsp/404.jsp
apache-tomcat-9.0.95/webapps/manager/WEB-INF/jsp/connectorCerts.jsp
apache-tomcat-9.0.95/webapps/manager/WEB-INF/jsp/connectorCiphers.jsp
apache-tomcat-9.0.95/webapps/manager/WEB-INF/jsp/connectorTrustedCerts.jsp
apache-tomcat-9.0.95/webapps/manager/WEB-INF/jsp/sessionDetail.jsp
apache-tomcat-9.0.95/webapps/manager/WEB-INF/jsp/sessionsList.jsp
apache-tomcat-9.0.95/webapps/manager/WEB-INF/web.xml
apache-tomcat-9.0.95/webapps/manager/css/manager.css
apache-tomcat-9.0.95/webapps/manager/images/asf-logo.svg
apache-tomcat-9.0.95/webapps/manager/images/tomcat.svg
apache-tomcat-9.0.95/webapps/manager/index.jsp
apache-tomcat-9.0.95/webapps/manager/status.xsd
apache-tomcat-9.0.95/webapps/manager/xform.xsl
apache-tomcat-9.0.95/bin/catalina.sh
apache-tomcat-9.0.95/bin/ciphers.sh
apache-tomcat-9.0.95/bin/configtest.sh
apache-tomcat-9.0.95/bin/daemon.sh
apache-tomcat-9.0.95/bin/digest.sh
apache-tomcat-9.0.95/bin/makebase.sh
apache-tomcat-9.0.95/bin/setclasspath.sh
apache-tomcat-9.0.95/bin/shutdown.sh
apache-tomcat-9.0.95/bin/startup.sh
apache-tomcat-9.0.95/bin/tool-wrapper.sh
apache-tomcat-9.0.95/bin/version.sh
[root@tomcat opt]#
Search
ENG
IN
12:00
19-09-2024

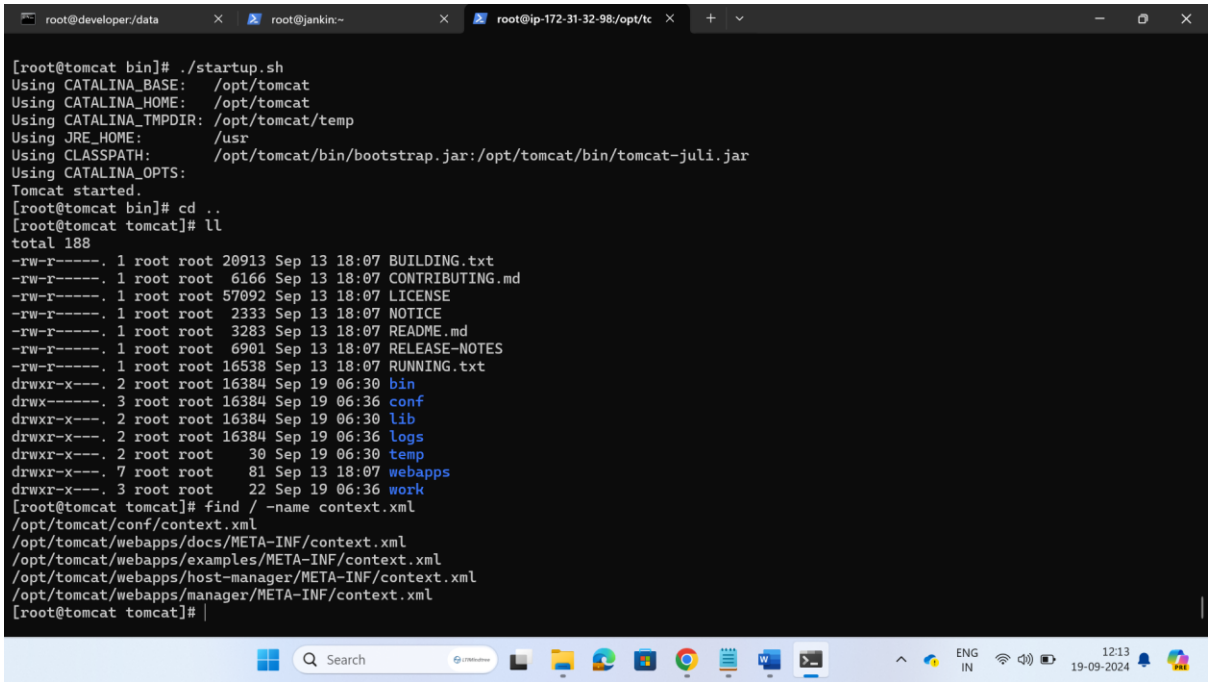

apache-tomcat-9.0.95/bin/version.sh
[root@tomcat opt]# ls
apache-tomcat-9.0.95 apache-tomcat-9.0.95.tar.gz aws
[root@tomcat opt]# mv apache-tomcat-9.0.95 tomcat
[root@tomcat opt]# ls
apache-tomcat-9.0.95.tar.gz aws tomcat
[root@tomcat opt]#
Search
ENG
IN
12:02
19-09-2024


root@developer:/data
root@jankin:~
root@ip-172-31-32-98:/opt/tc
[root@tomcat opt]# cd tomcat
[root@tomcat tomcat]# cd bin
[root@tomcat bin]# ls
bootstrap.jar ciphers.sh daemon.sh setclasspath.bat startup.sh version.bat
catalina-tasks.xml commons-daemon-native.tar.gz digest.bat setclasspath.sh tomcat-juli.jar version.sh
catalina.bat commons-daemon.jar digest.sh shutdown.bat tomcat-native.tar.gz
catalina.sh configtest.bat makebase.bat shutdown.sh tool-wrapper.bat
ciphers.bat configtest.sh makebase.sh startup.bat tool-wrapper.sh
[root@tomcat bin]# ./startup.sh
Neither the JAVA_HOME nor the JRE_HOME environment variable is defined
At least one of these environment variable is needed to run this program
[root@tomcat bin]#

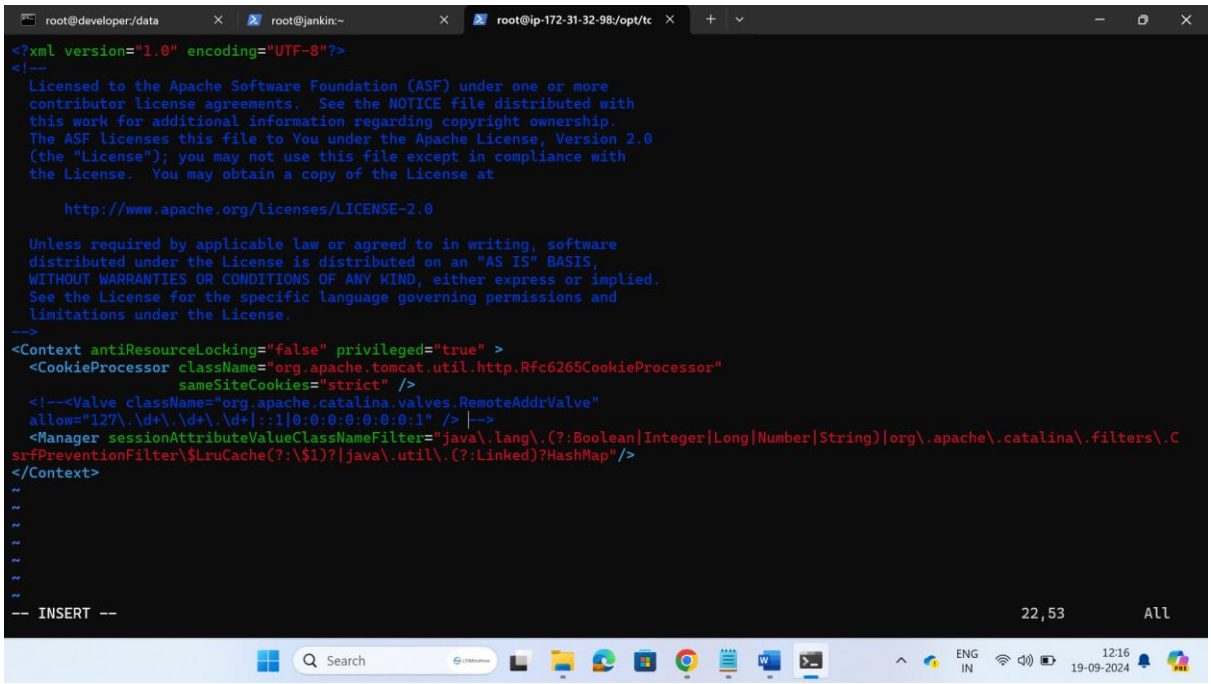
Copy Tomcat Public key and Port 8080



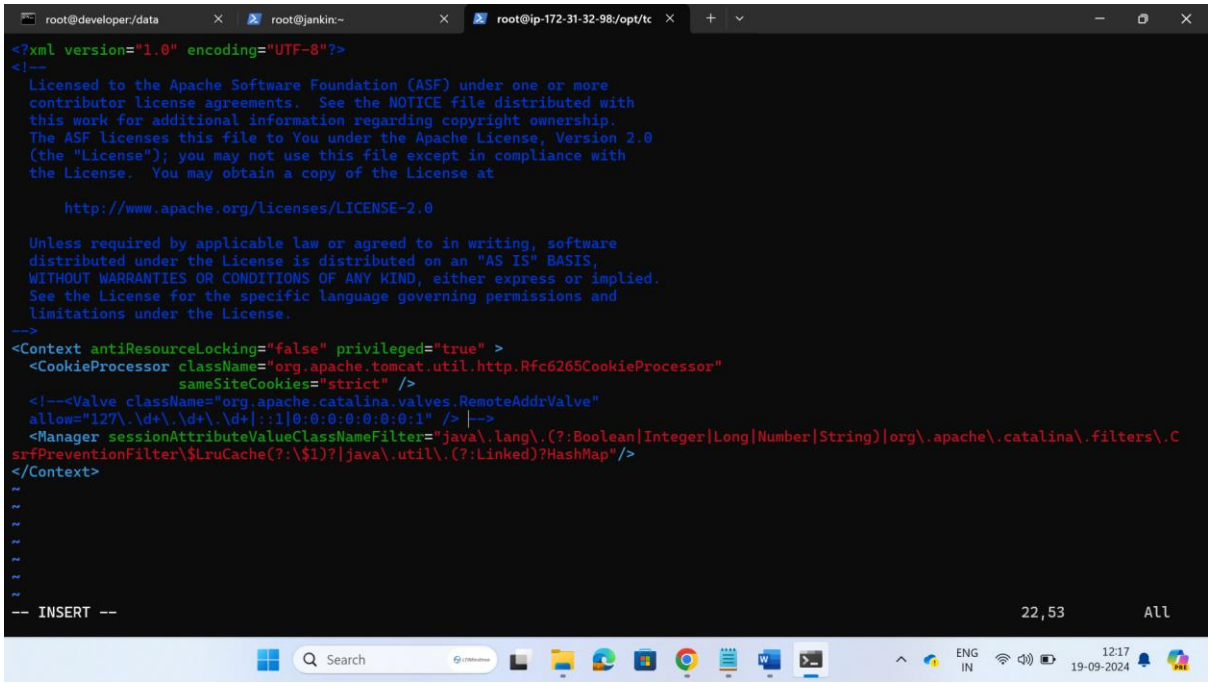




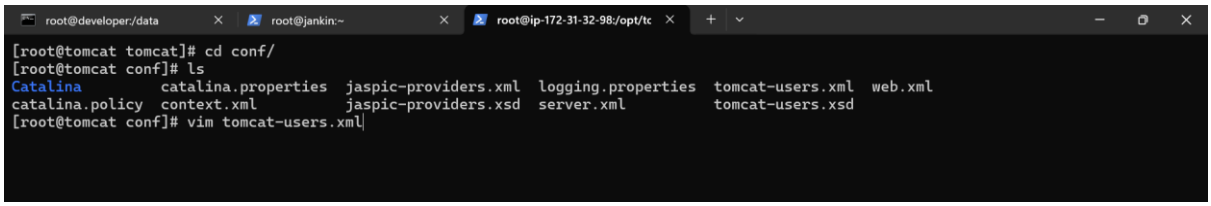
Comment last two lines



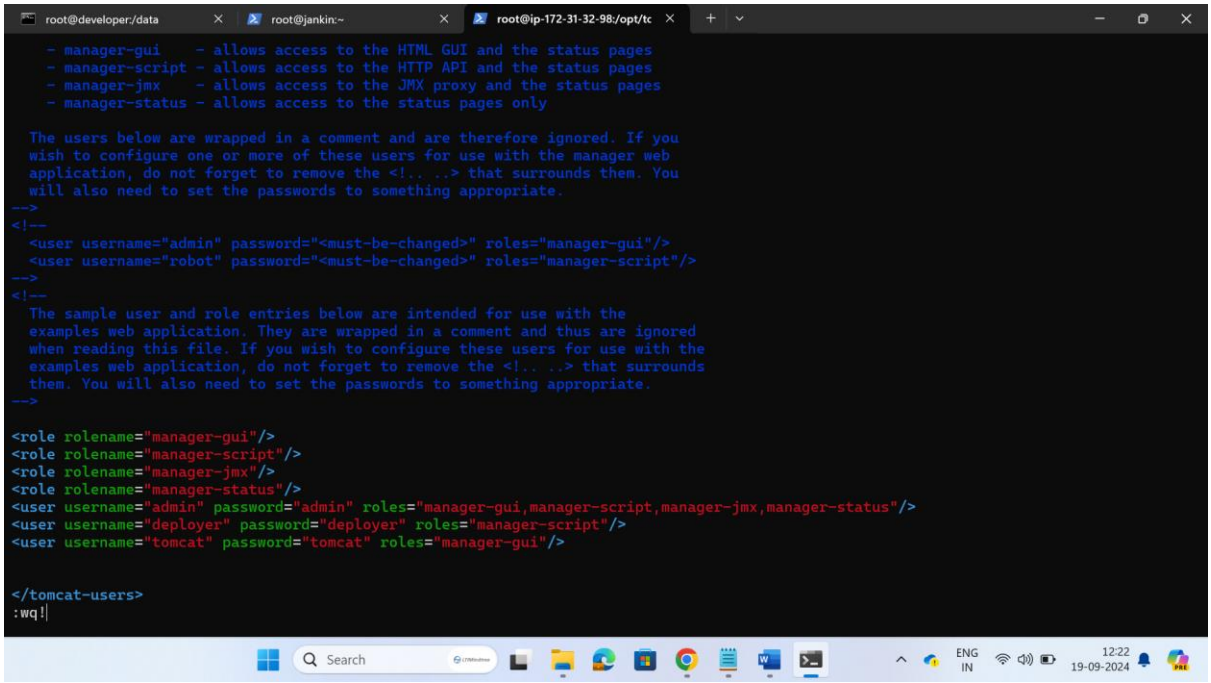
Comment last two lines

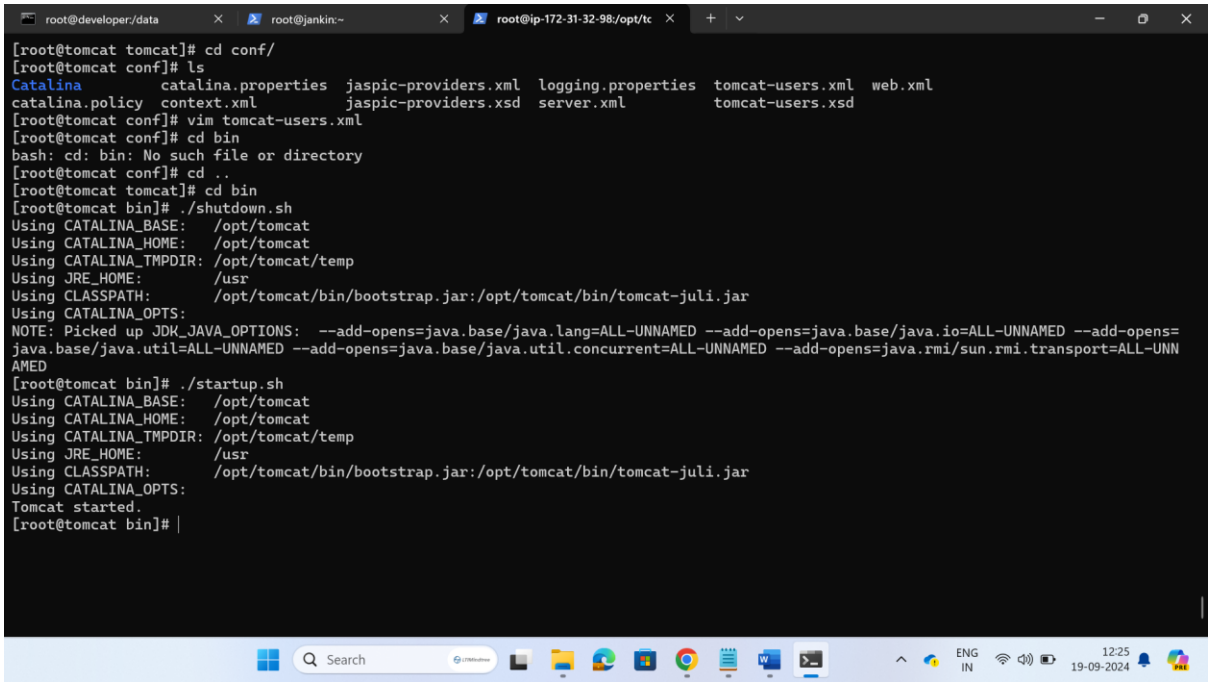


Open tomcat-users.xml file and Changes this file

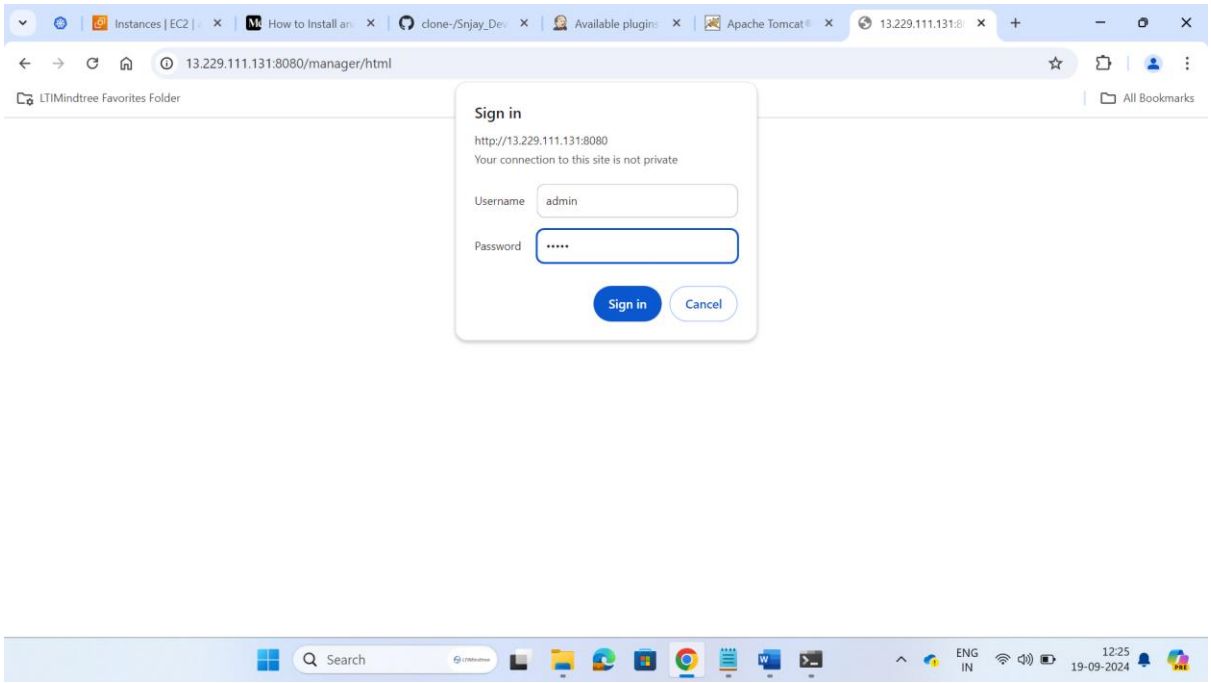


Change last six lines then save this file

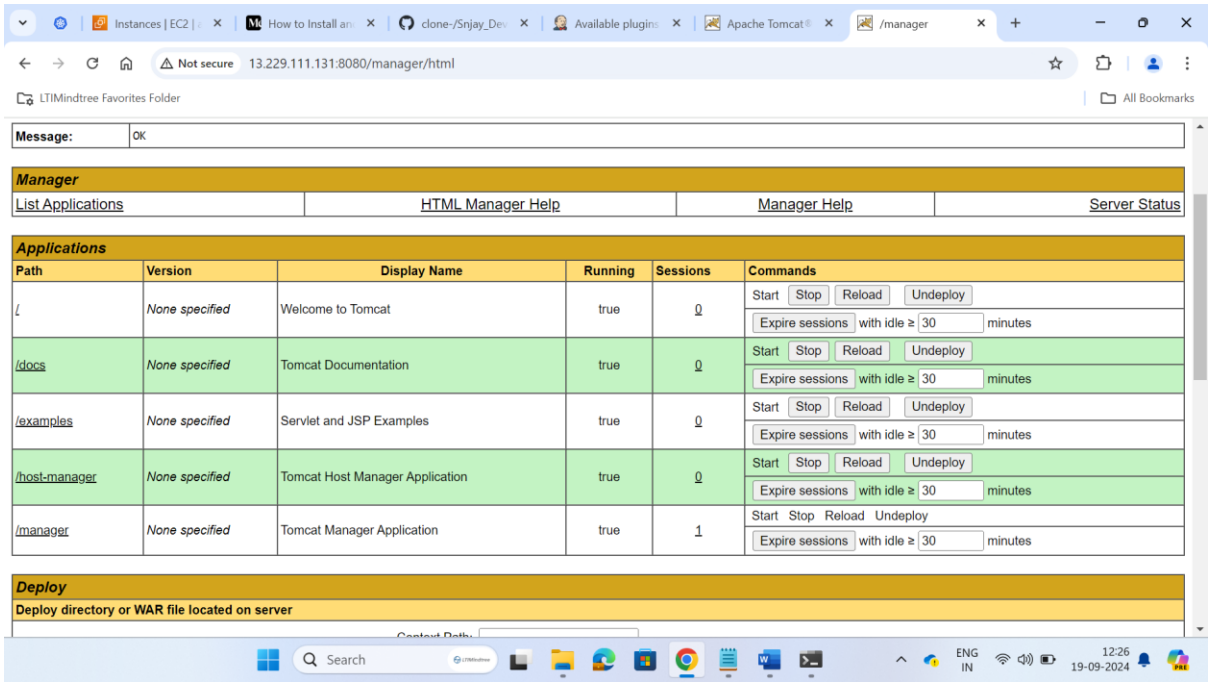




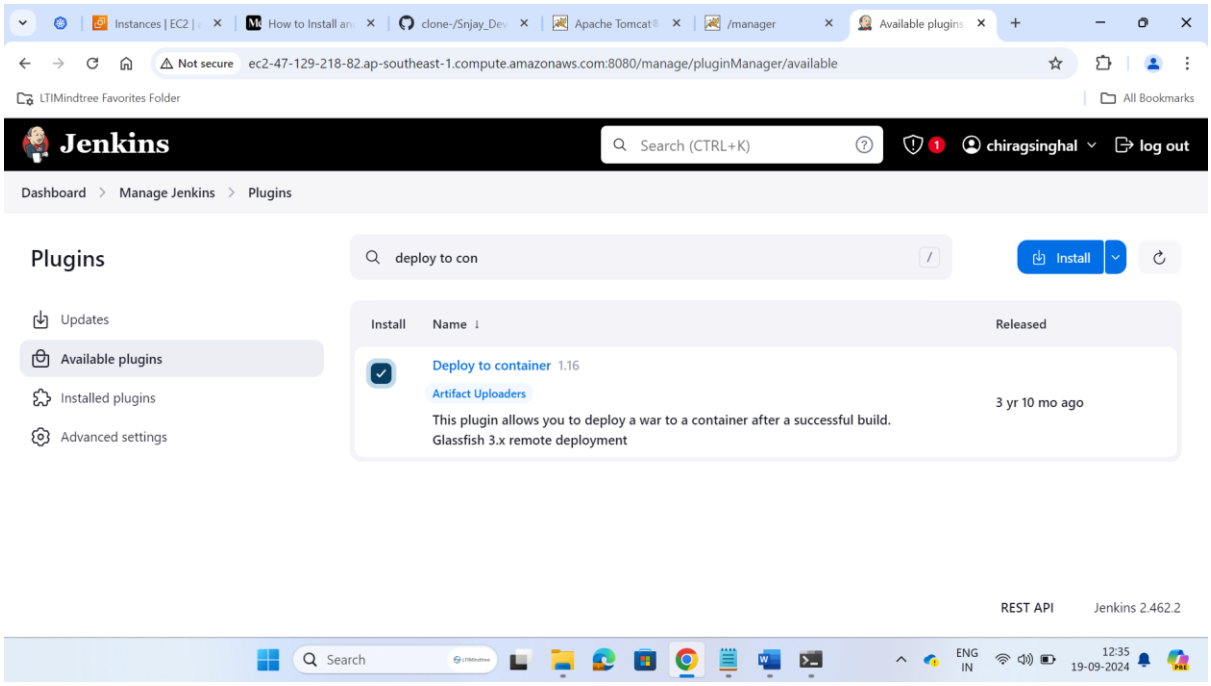
Open Tomcat Server: set username and passwrod by using file



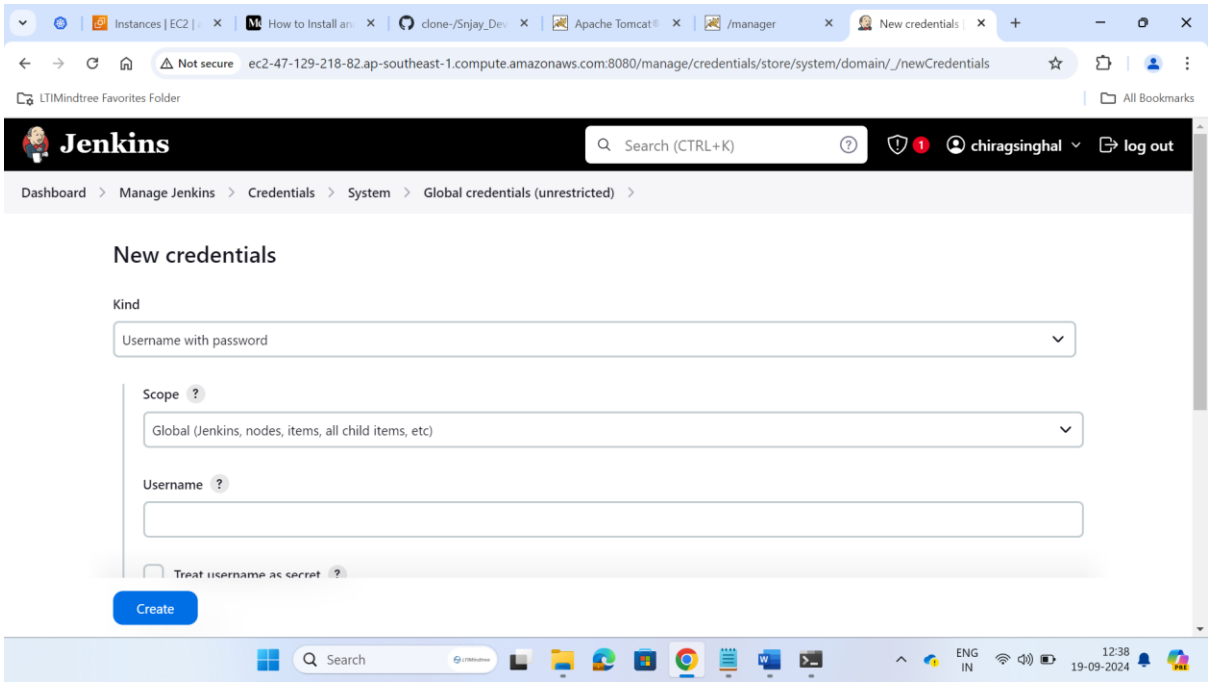
Show this page

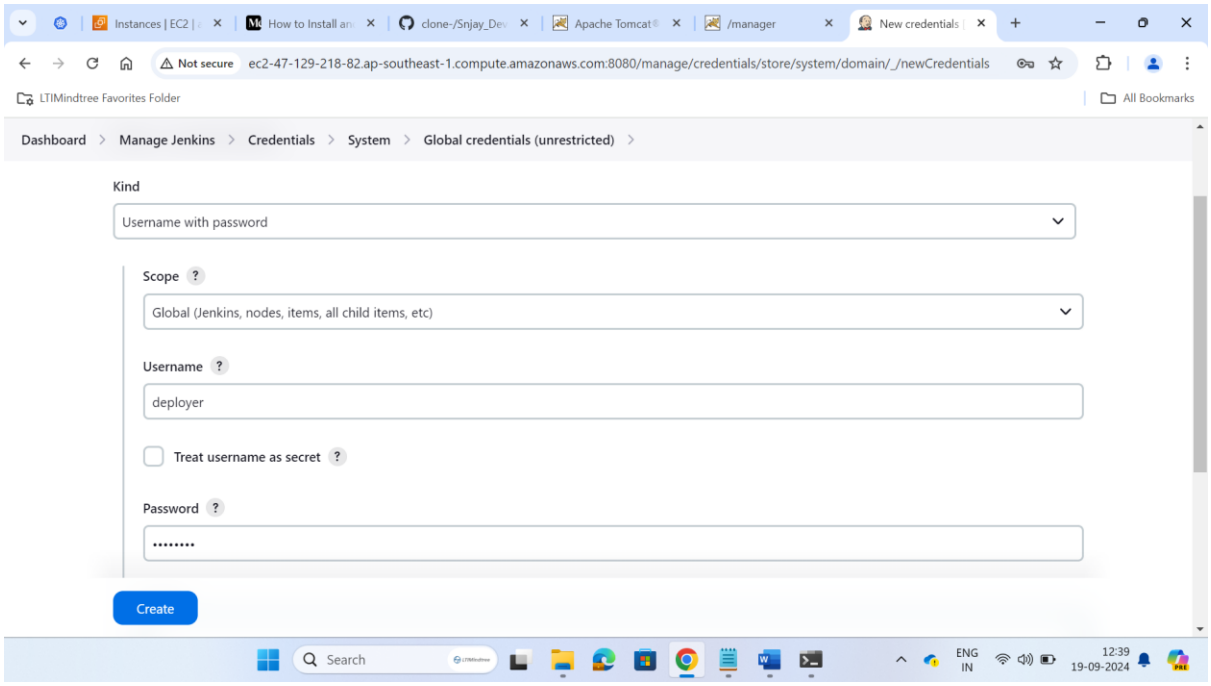


Install Deploy to Container by using Tomcat

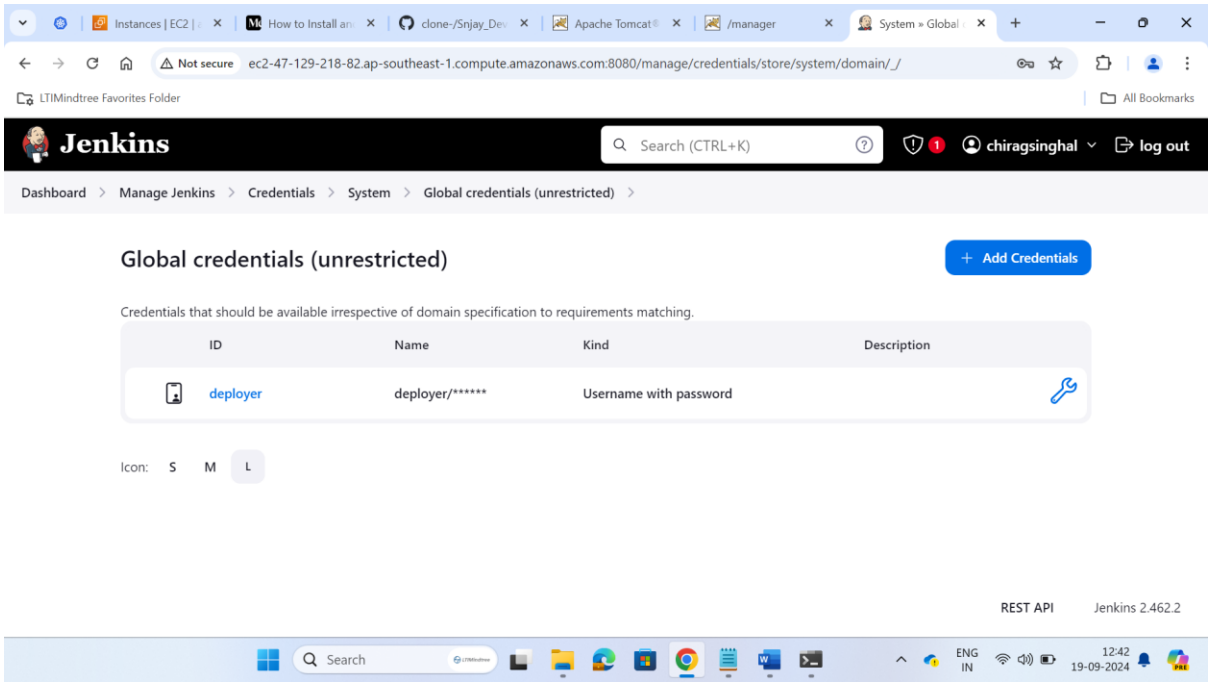


Goto Dashboard > Manage Jenkins > Credentials > System > Global Credentials then click new credentials

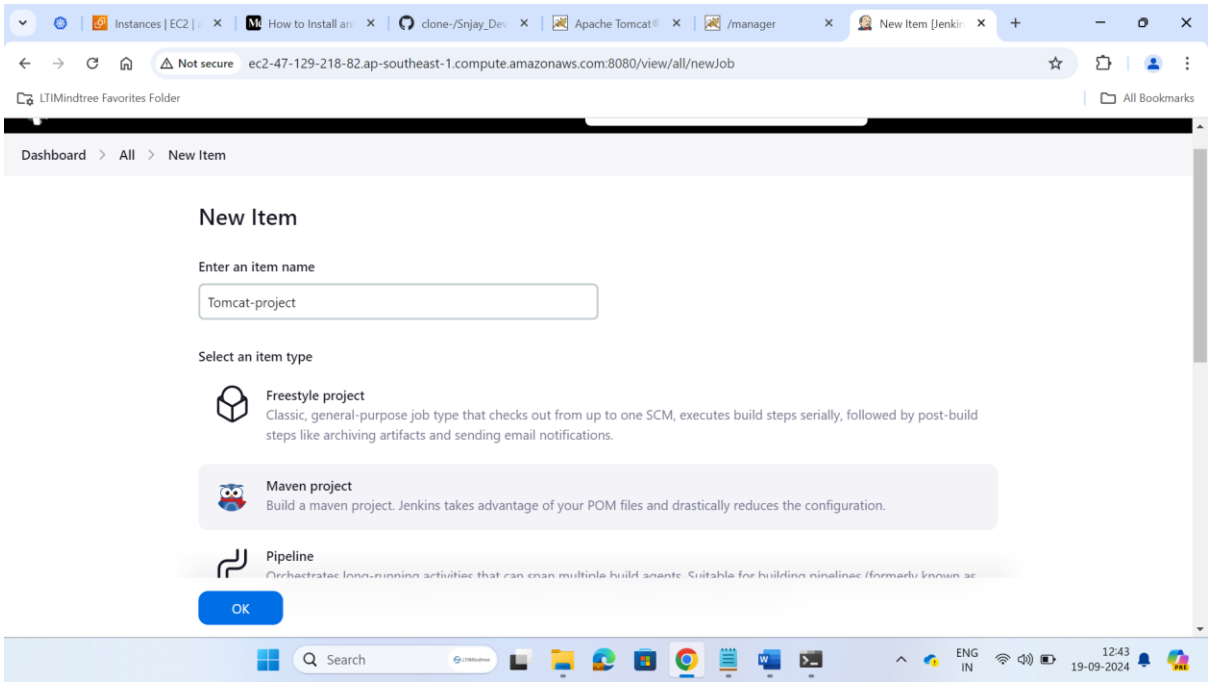




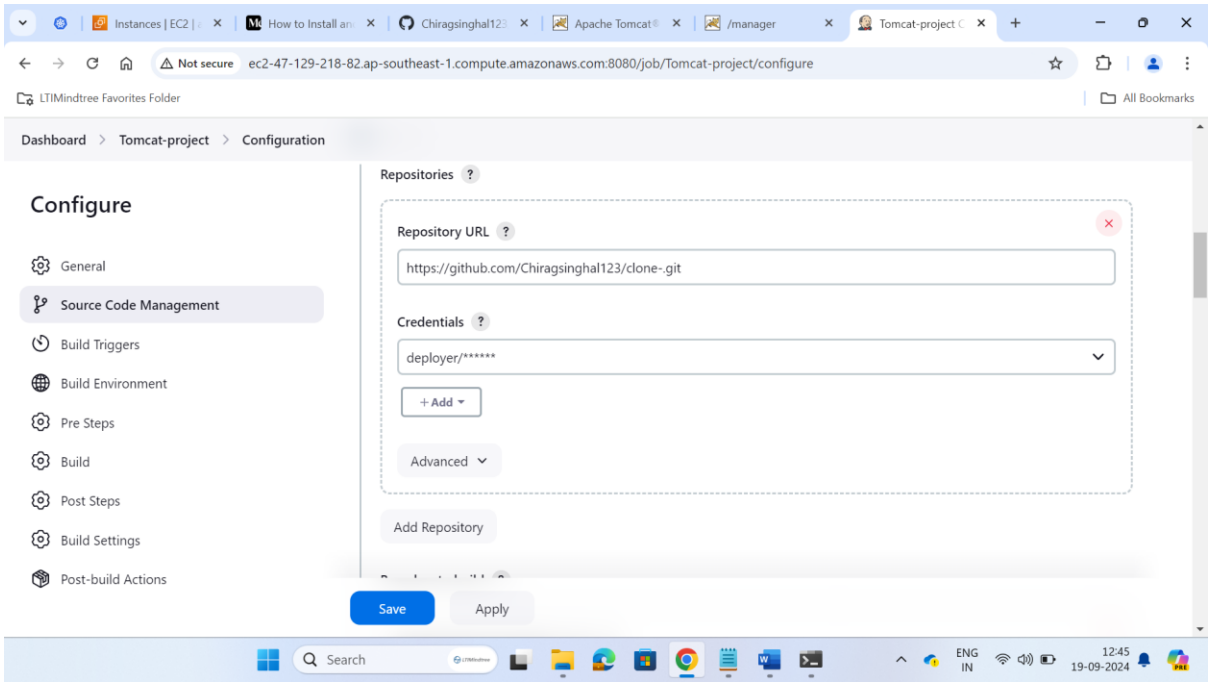
Add Credentials

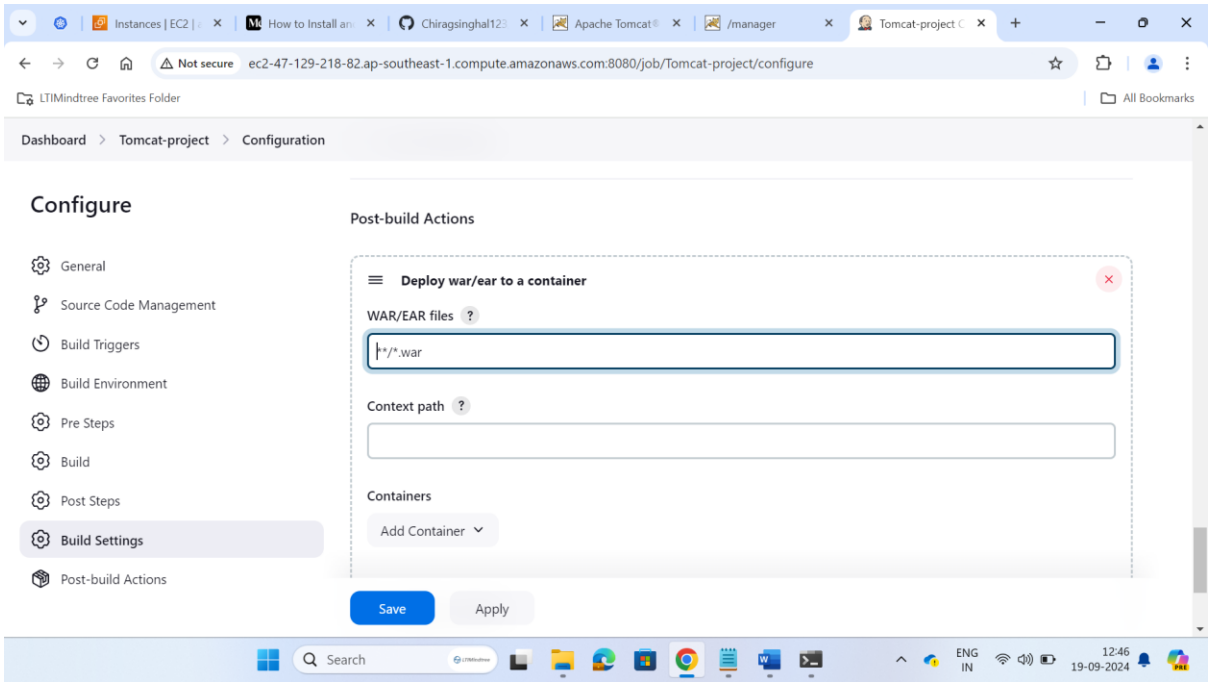


Click New Item and build Tomcat Project select Maven project then ok

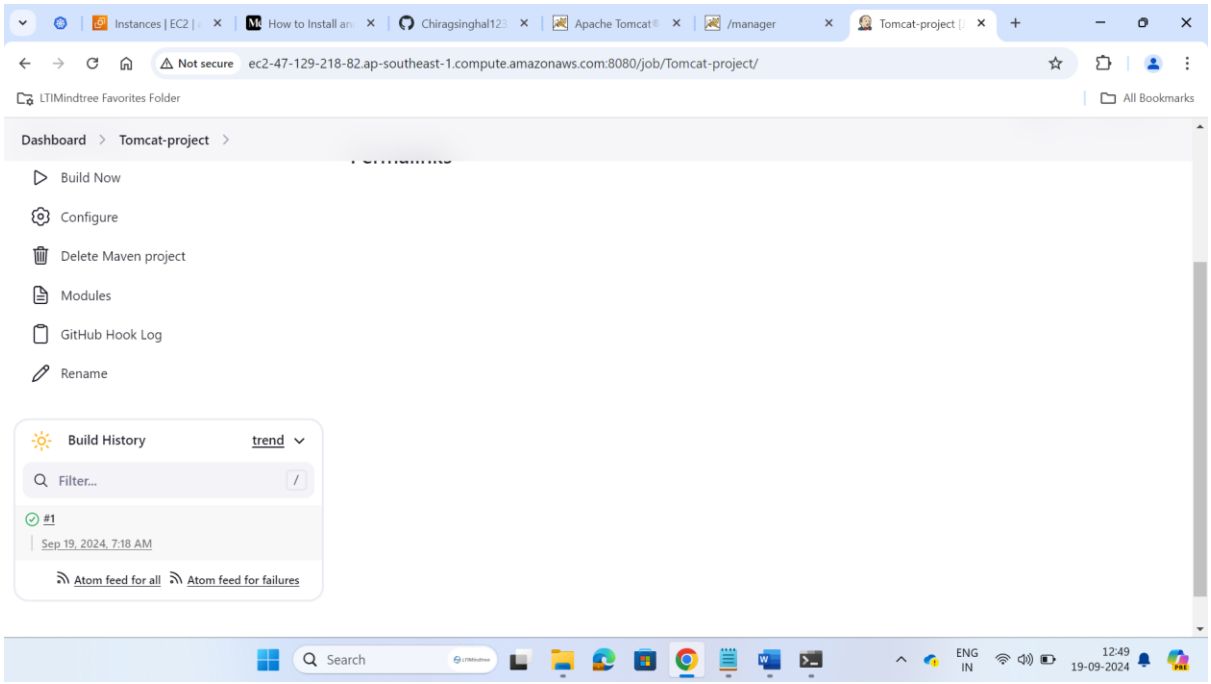


Add GitHub url < Add Credentials

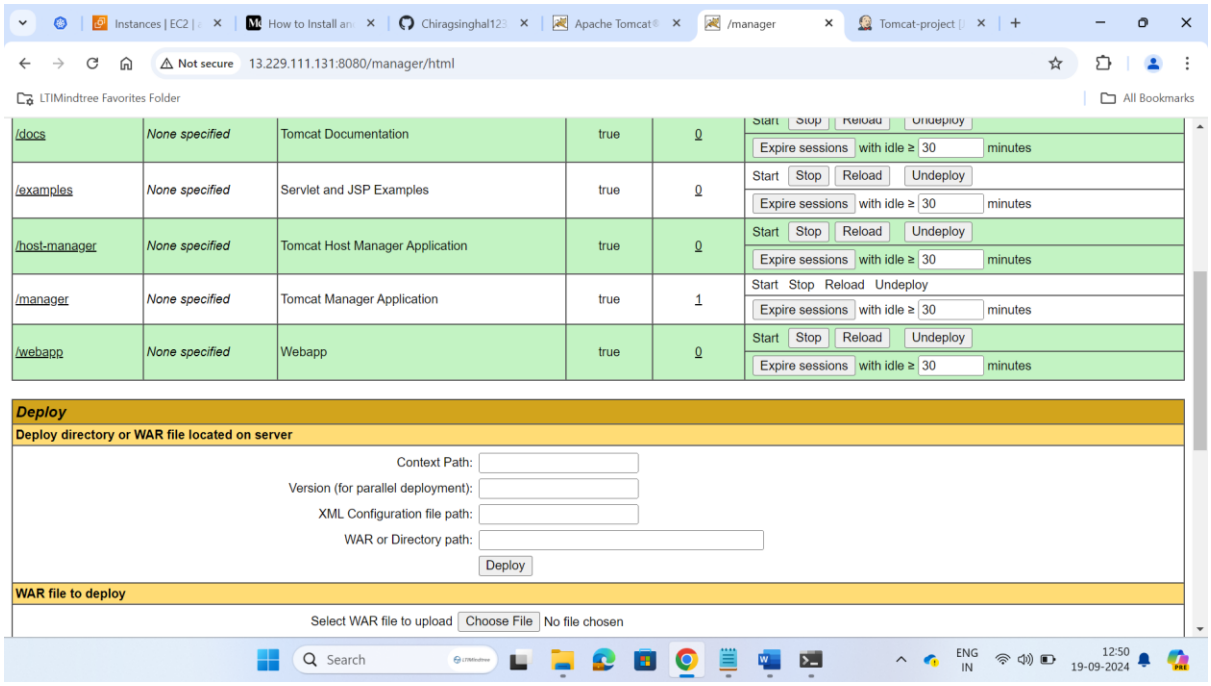


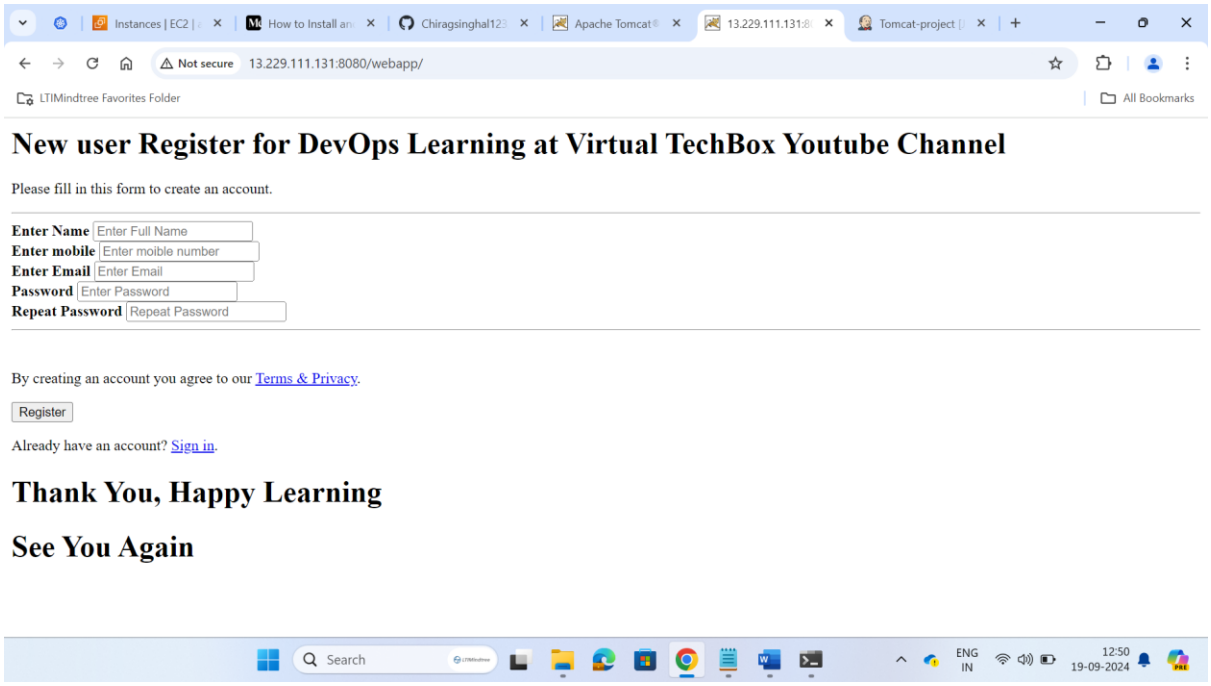


Build Tomcat Project Sucessfully

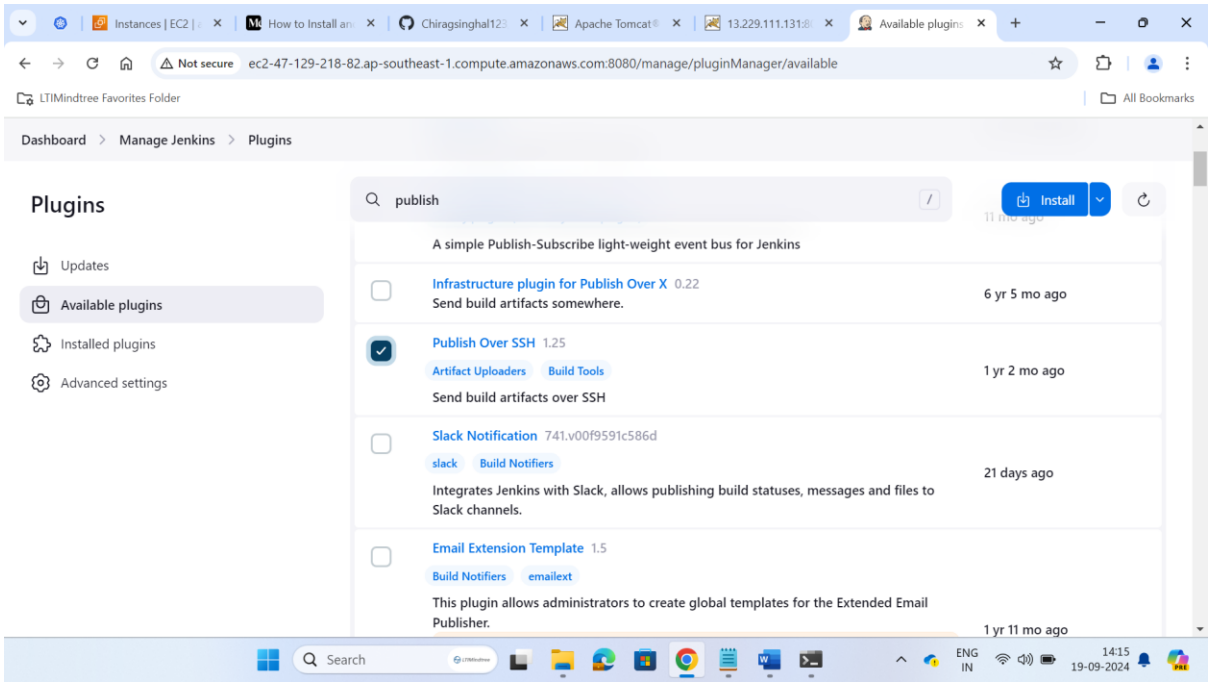


Webapp Aplication is Visible in Tomcat Server

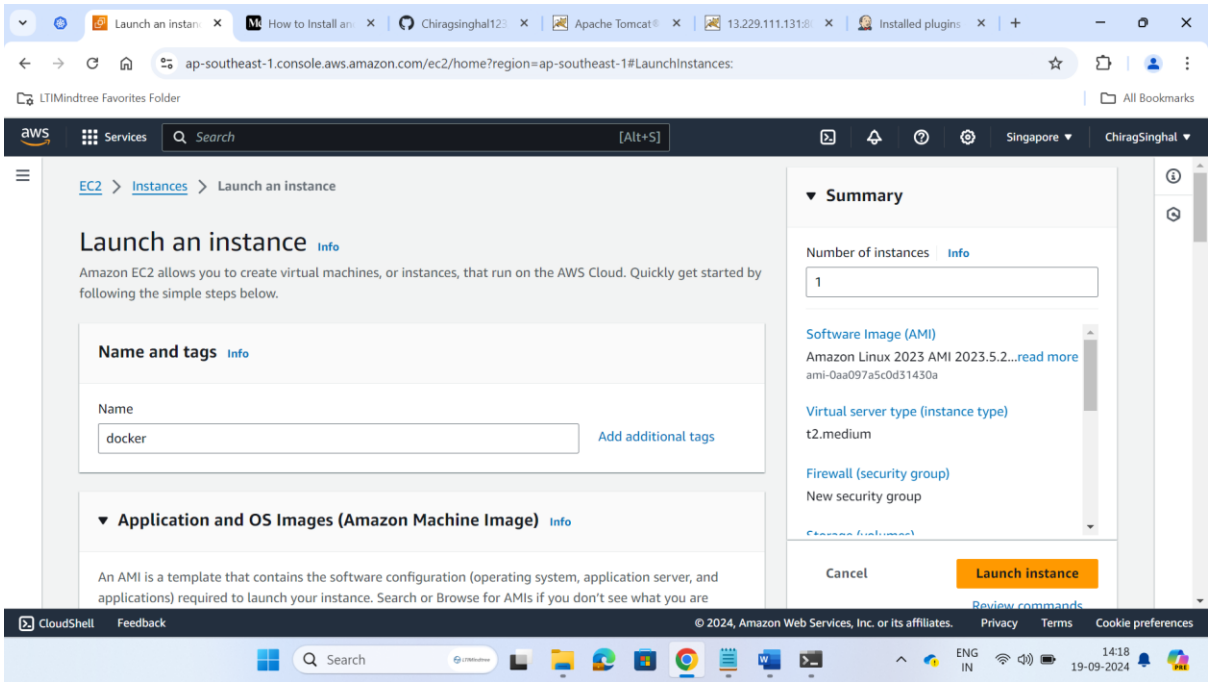




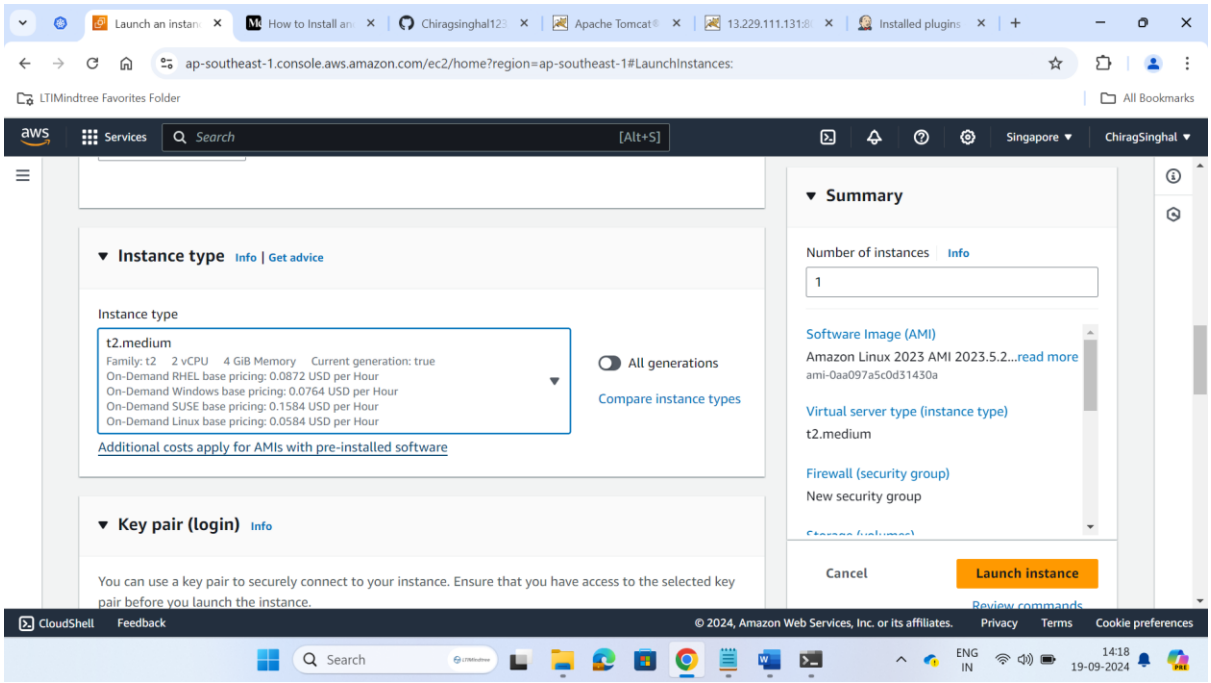
Click Manage Jenkin install publish over SSH by using Docker



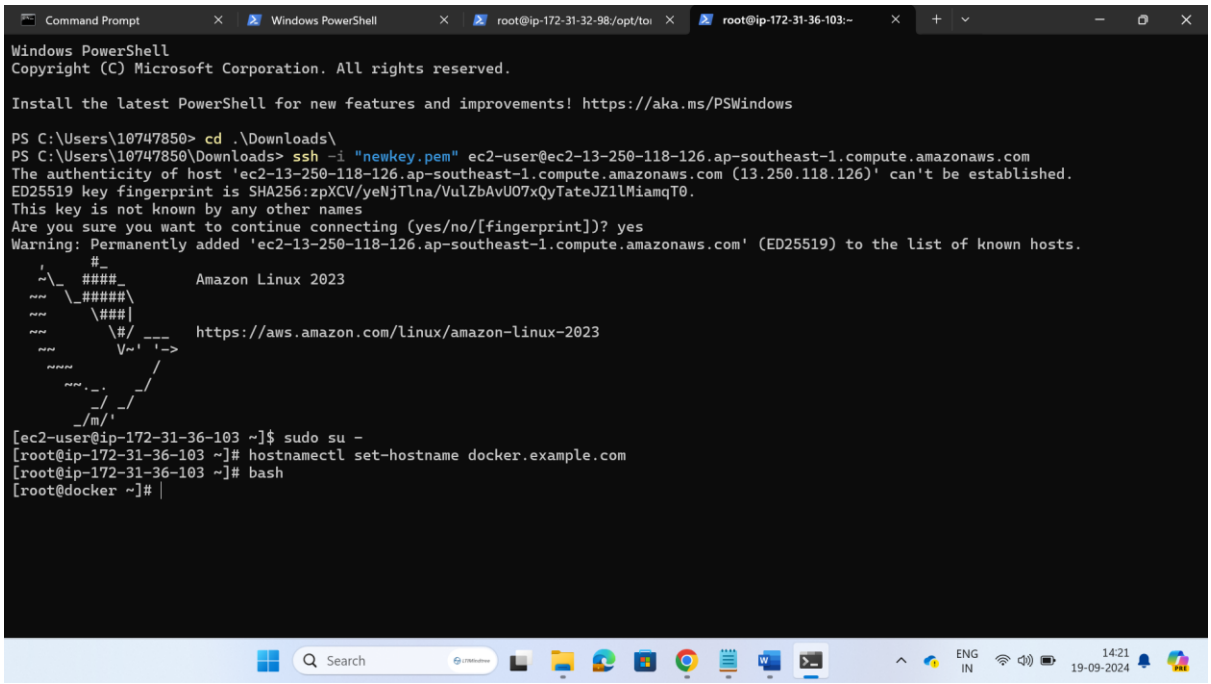
Launch instance Docker



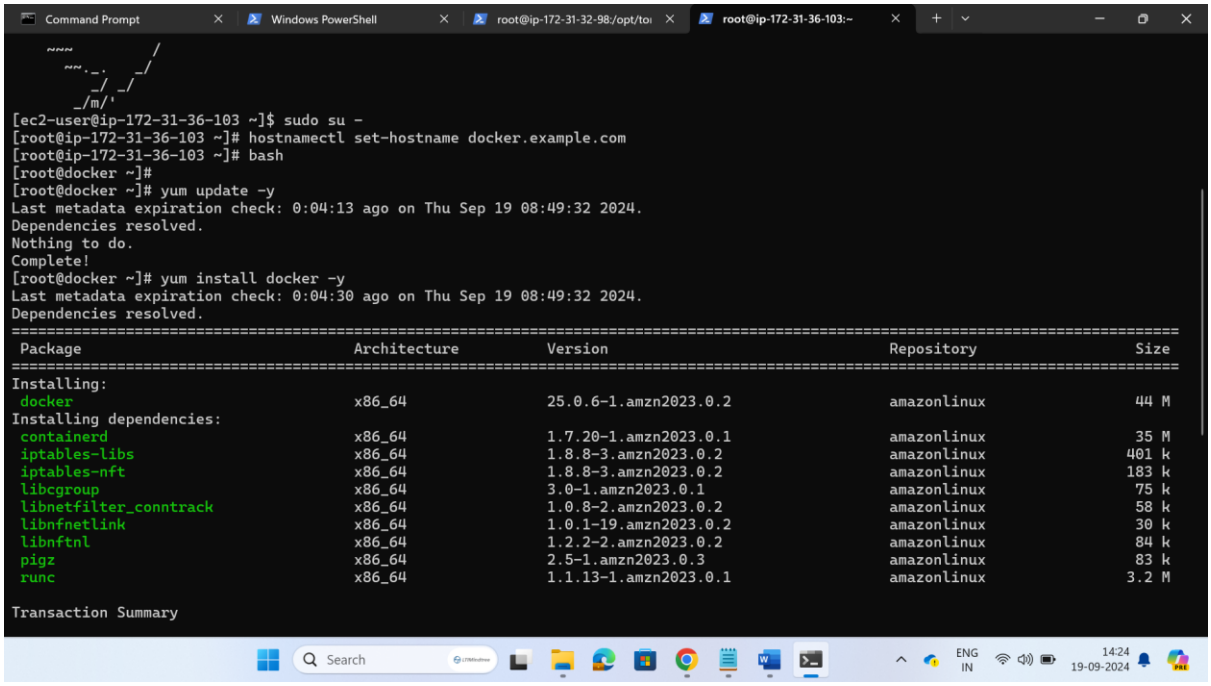
Select t2medium because the minimum reqiurement to run Docker is of 2 core CPU.



Docker machine is ready and change host name

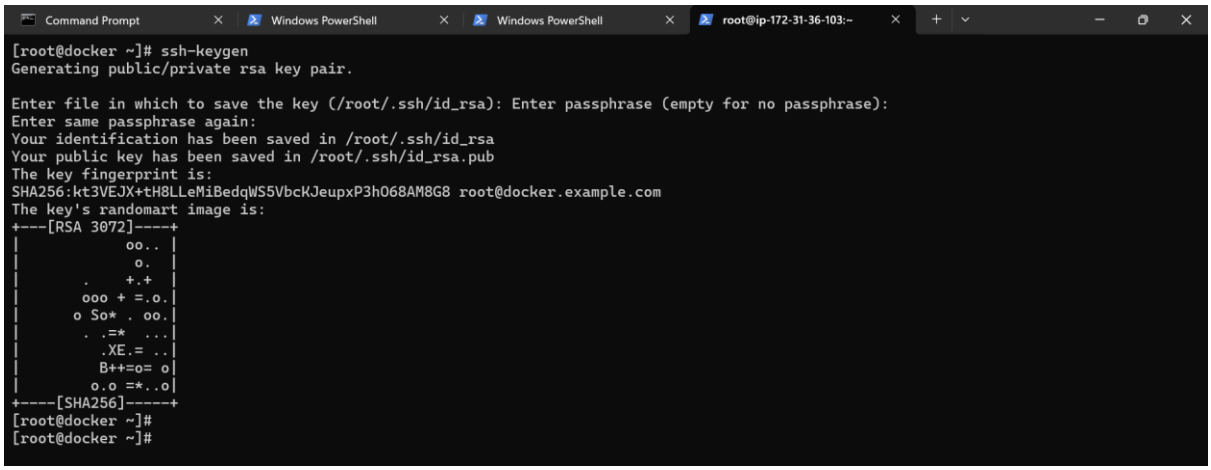


Docker Machine install Docker

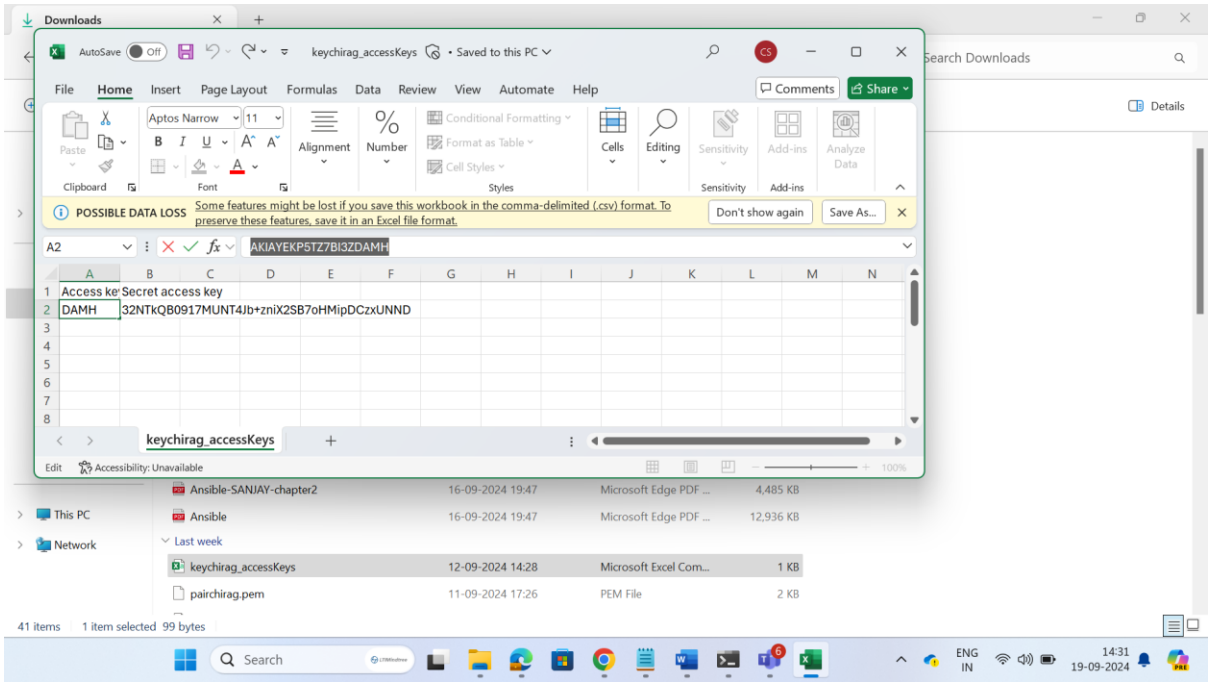


Docker service start

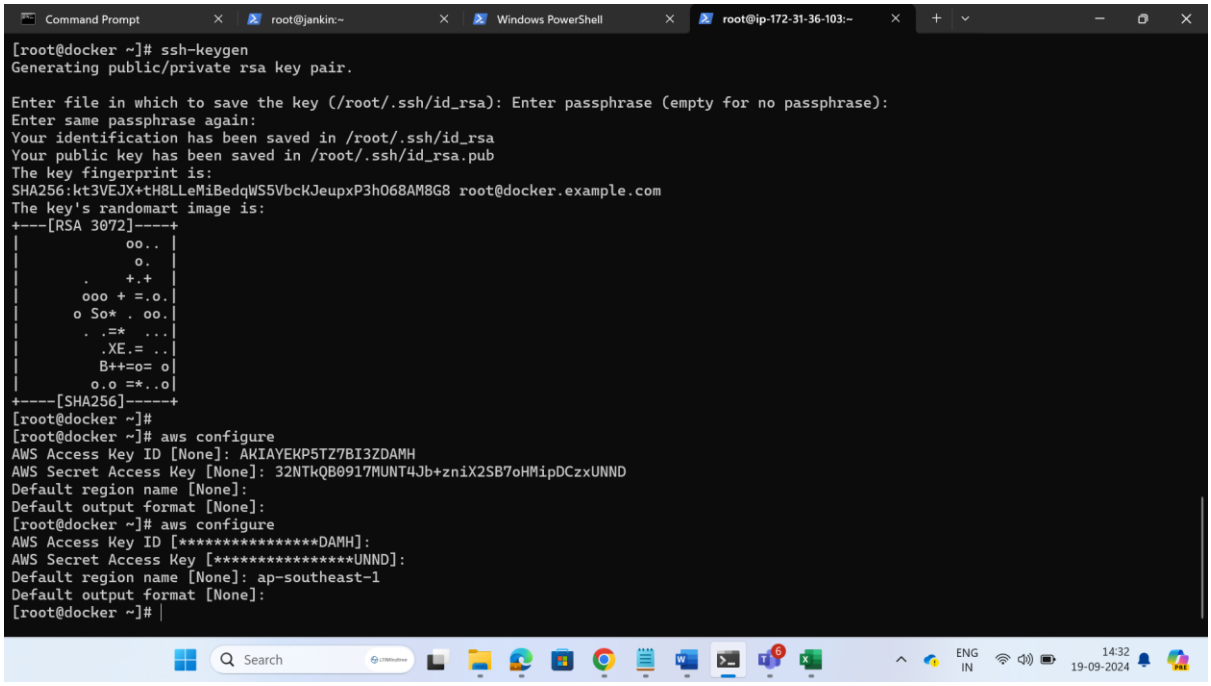
Create ssh-keygen



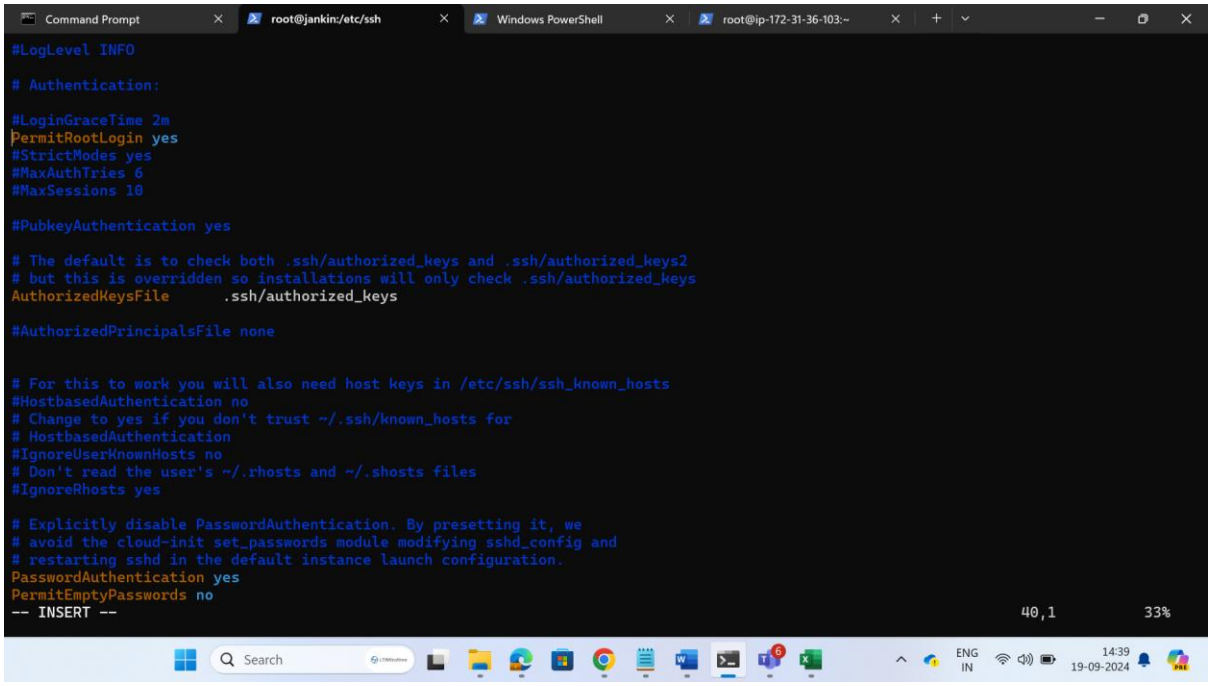
Generate user key and Secret access key:

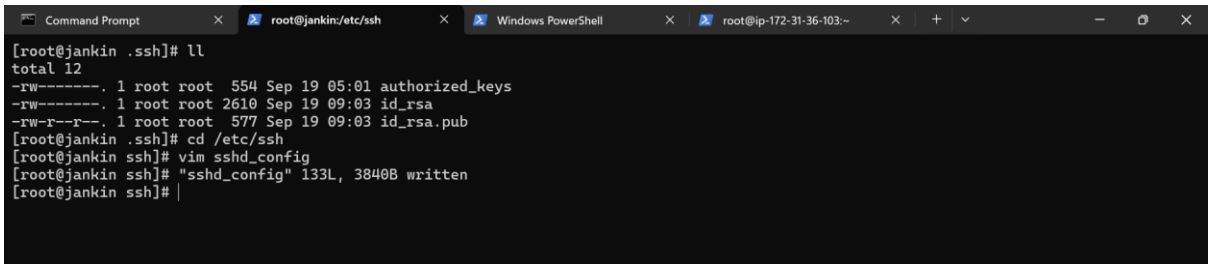


Paste Access key ,Secret Access Key and Region name :

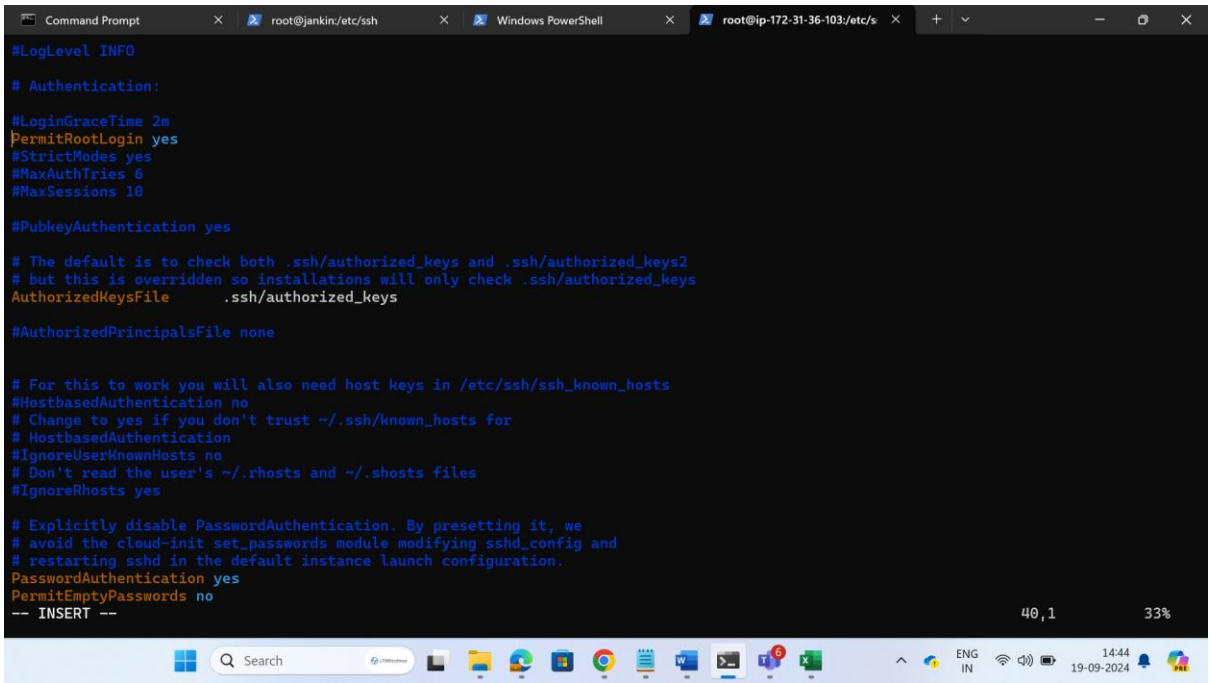


Command : vim /etc/ssh/sshd_confg [allow permission in Jenkin machine]

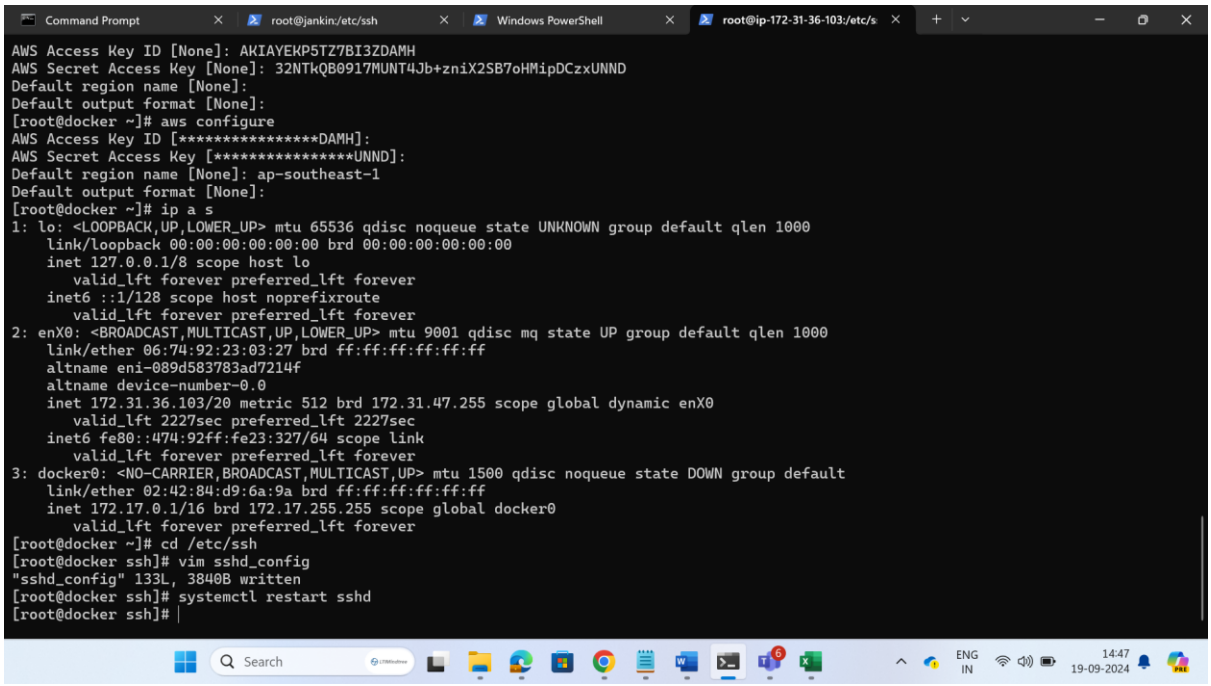


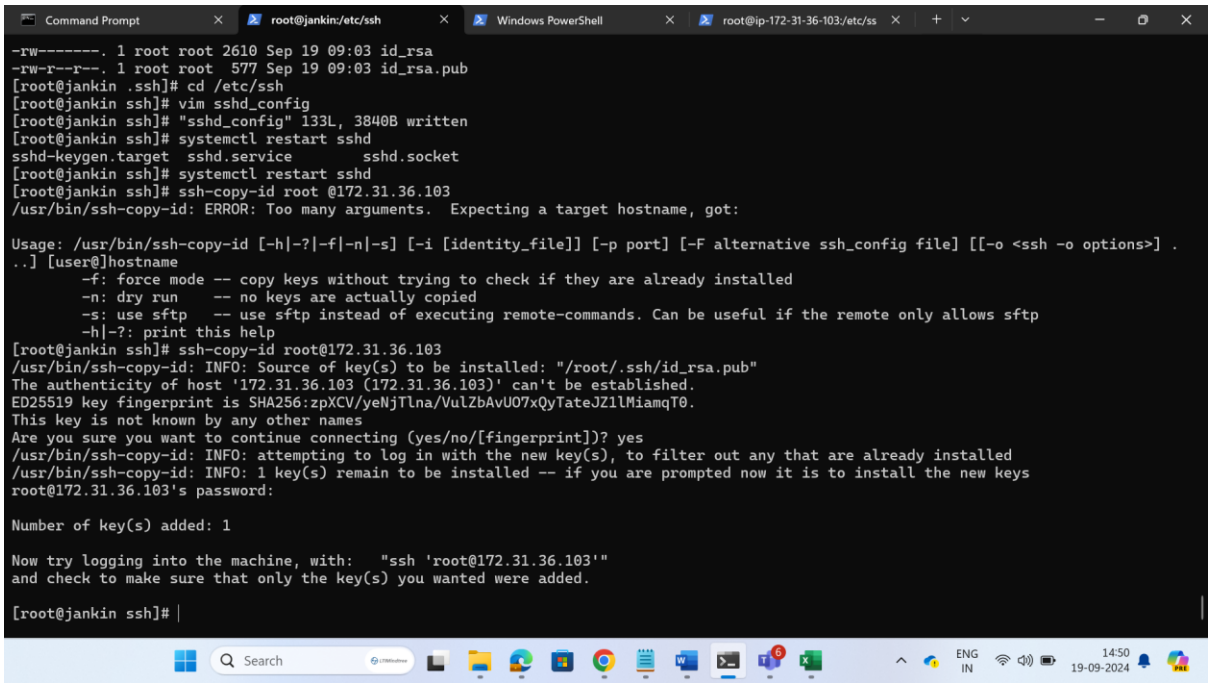


Allow Permission in Docker Machine

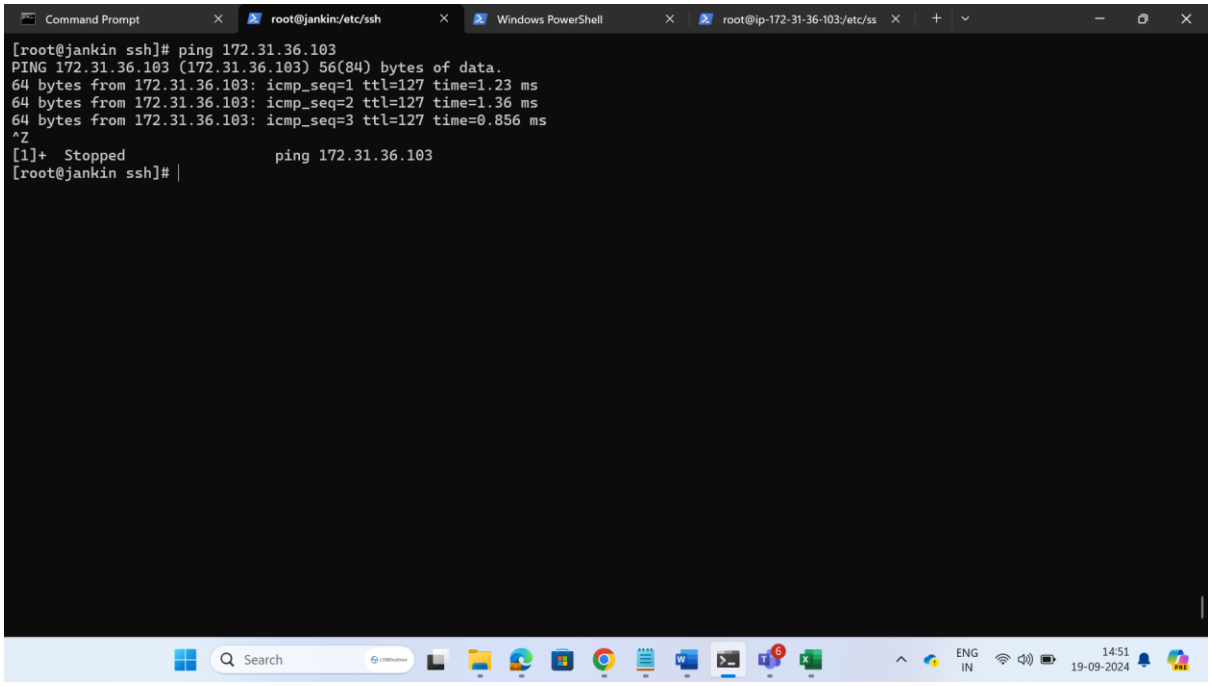


Aws Configure in Docker Machine:

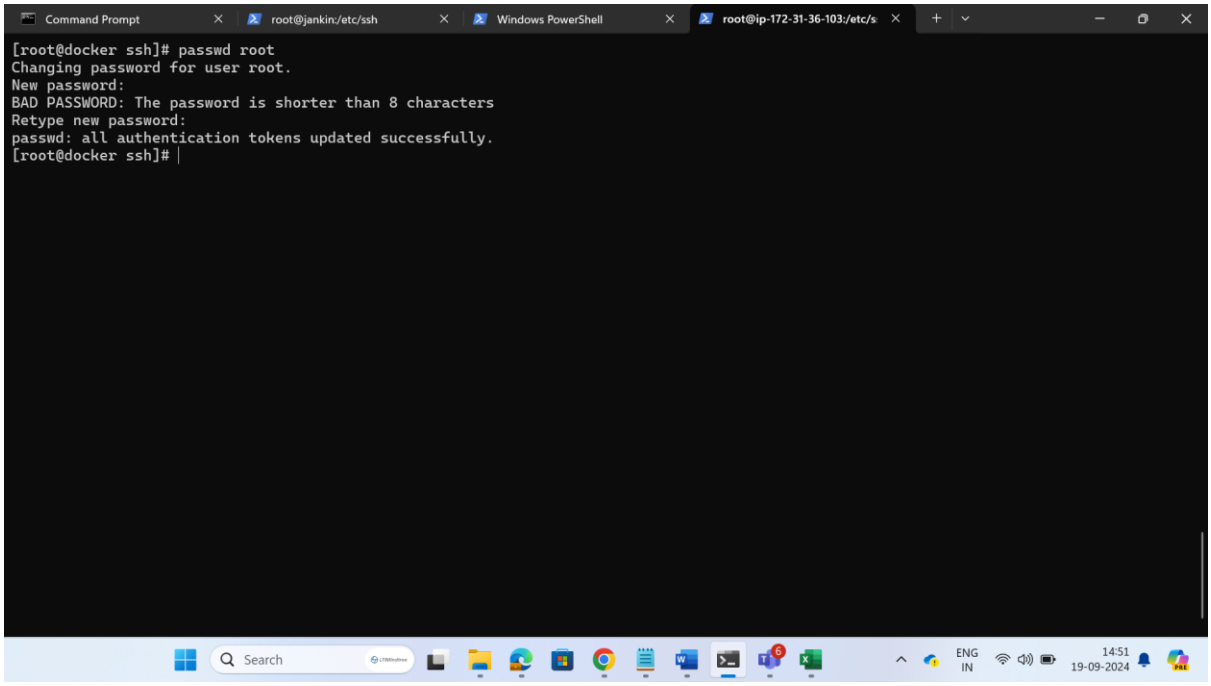


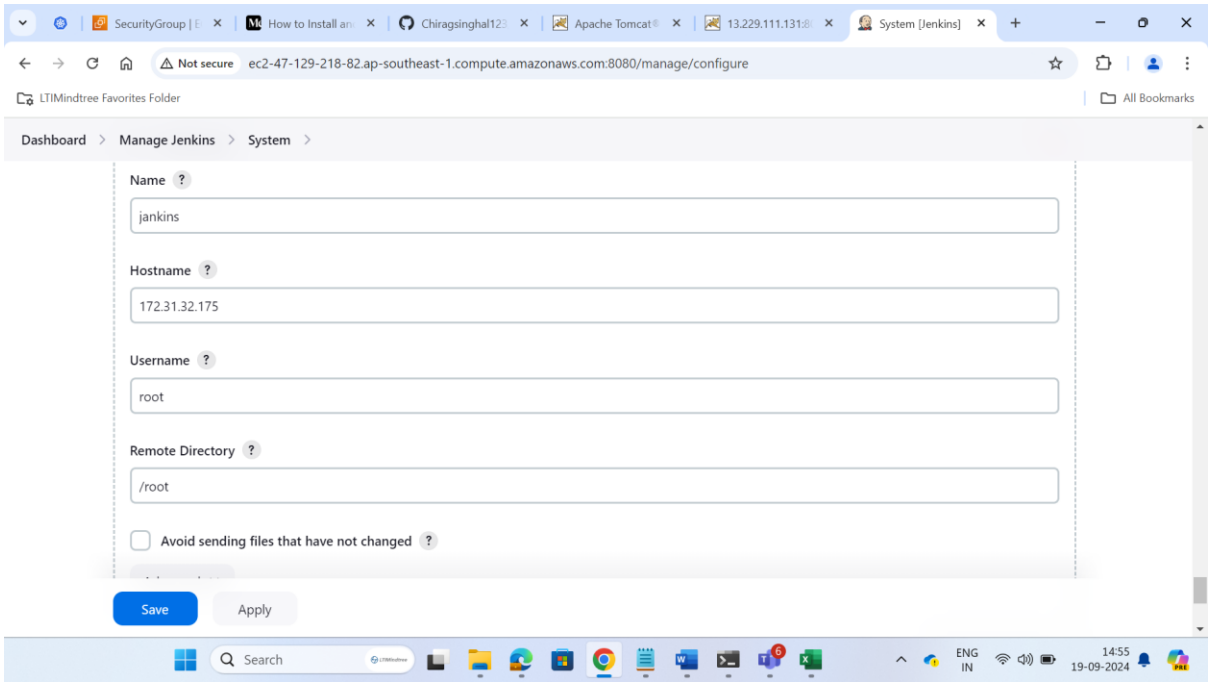


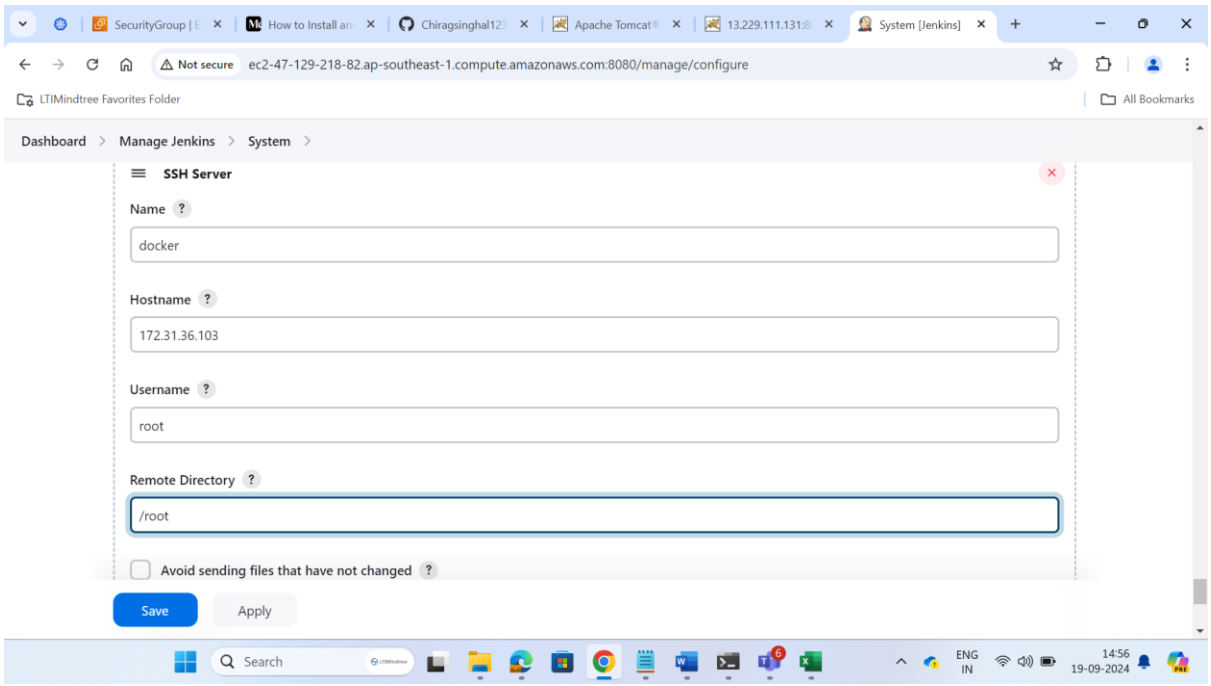
Check ping Docker ip Address in Jenkins Machine



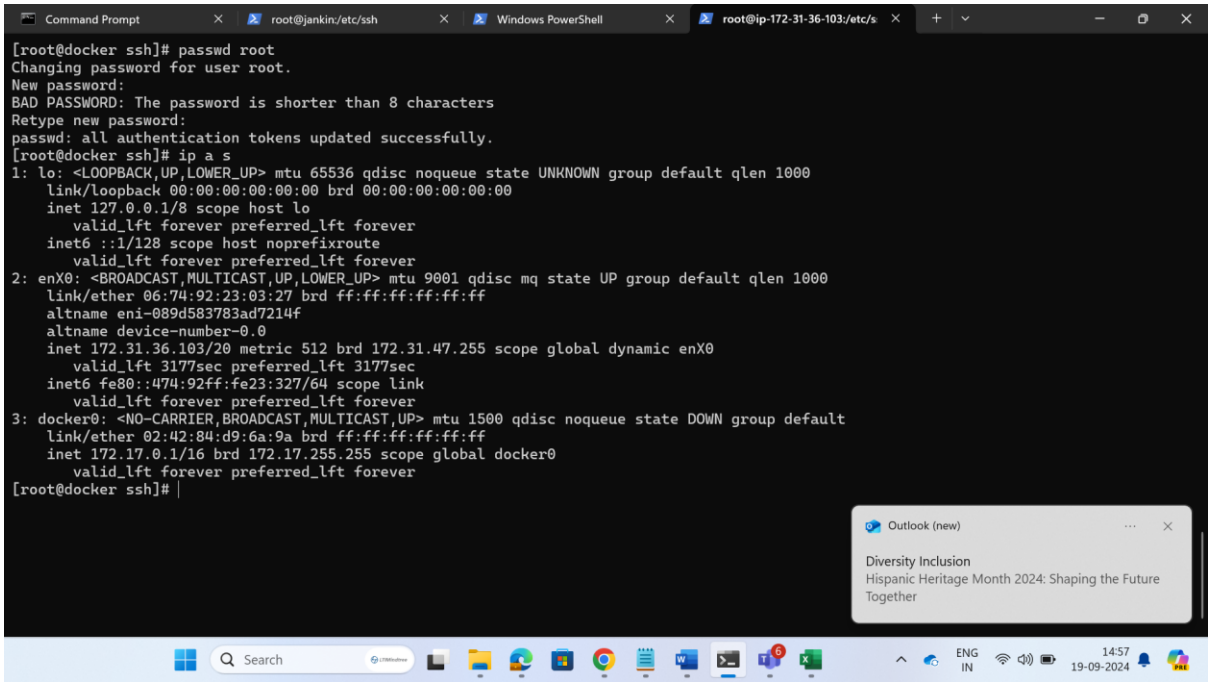
Set Password in Docker Machine :



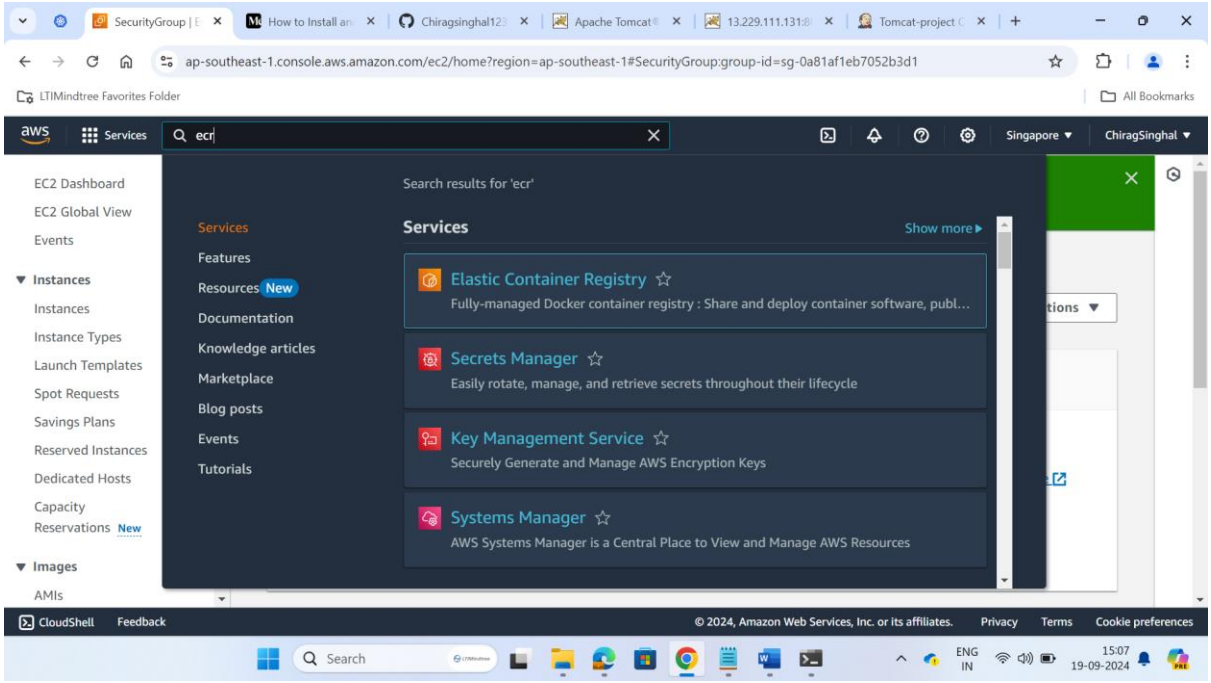

System [Jenkins]
Not secure
ec2-47-129-218-82.ap-southeast-1.compute.amazonaws.com:8080/manage/configure
LTIMindtree Favorites Folder
All Bookmarks
Dashboard
Manage Jenkins
System
Name
jankins
Hostname
172.31.32.175
Username
root
Remote Directory
/root
Avoid sending files that have not changed
Save
Apply
Search
ENG
IN
14:55
19-09-2024


System [Jenkins]
Not secure
ec2-47-129-218-82.ap-southeast-1.compute.amazonaws.com:8080/manage/configure
LTIMindtree Favorites Folder
All Bookmarks
Dashboard
Manage Jenkins
System
SSH Server
Name
docker
Hostname
172.31.36.103
Username
root
Remote Directory
/root
Avoid sending files that have not changed
Save
Apply
Search
ENG
IN
14:56
19-09-2024

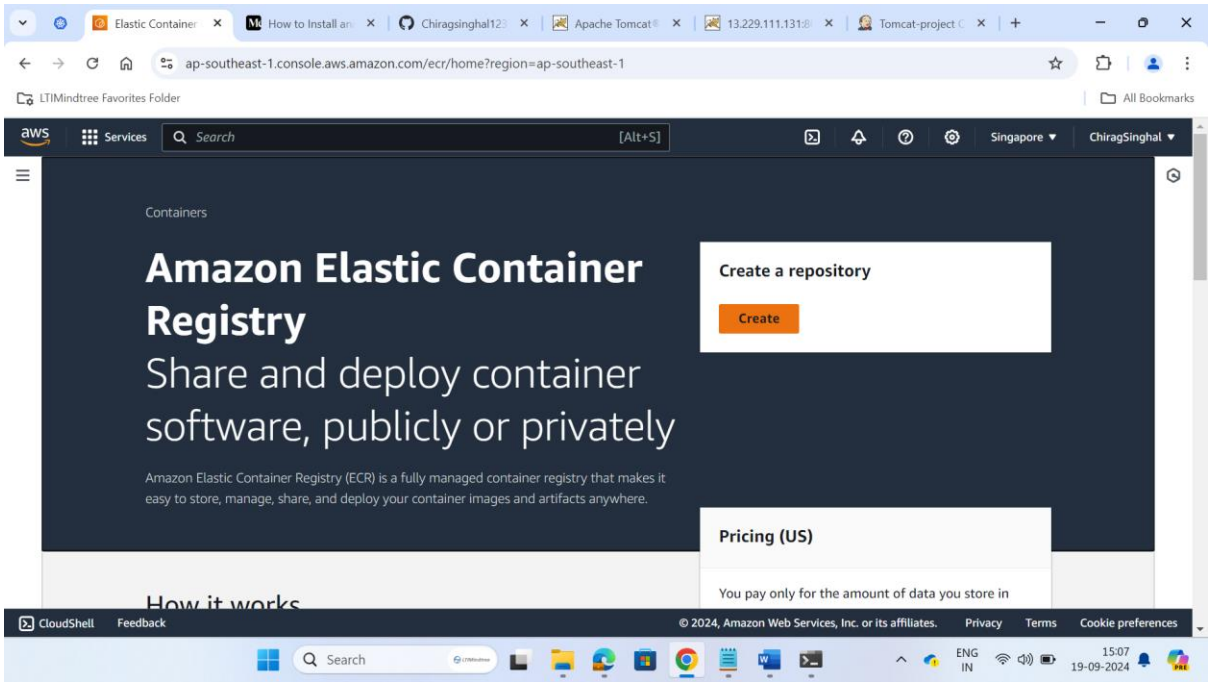
Show Docker Machine IP Address

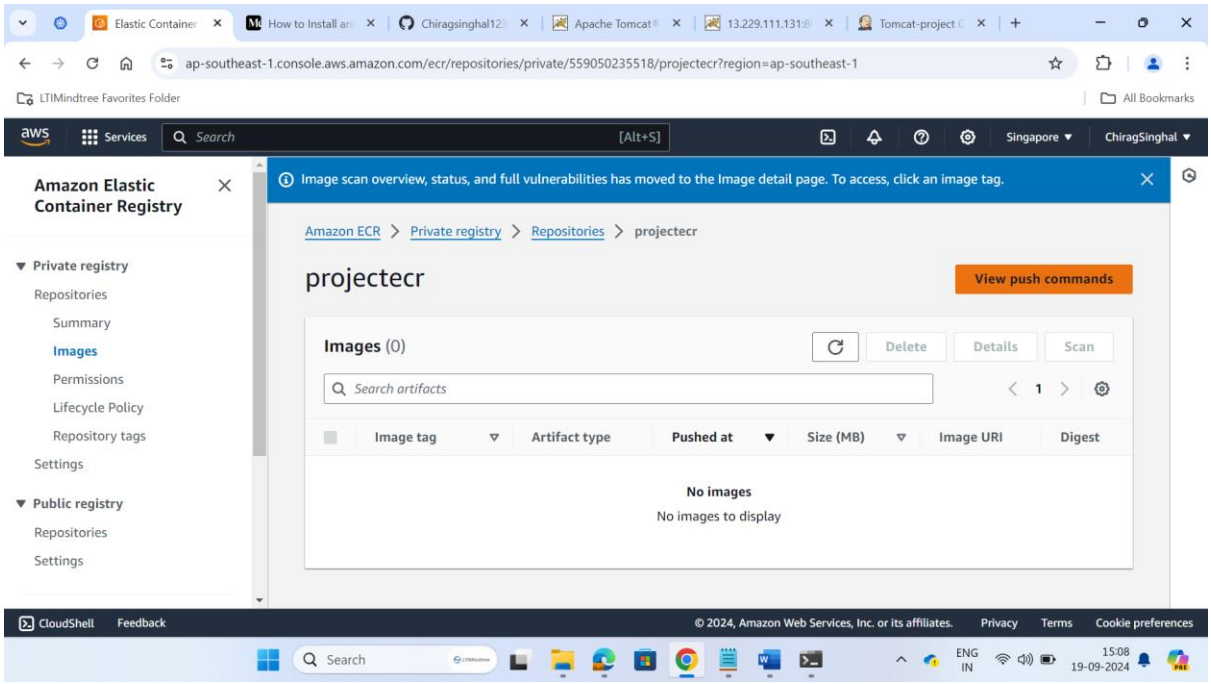


Cerate ECR [Elastic Container Registry]

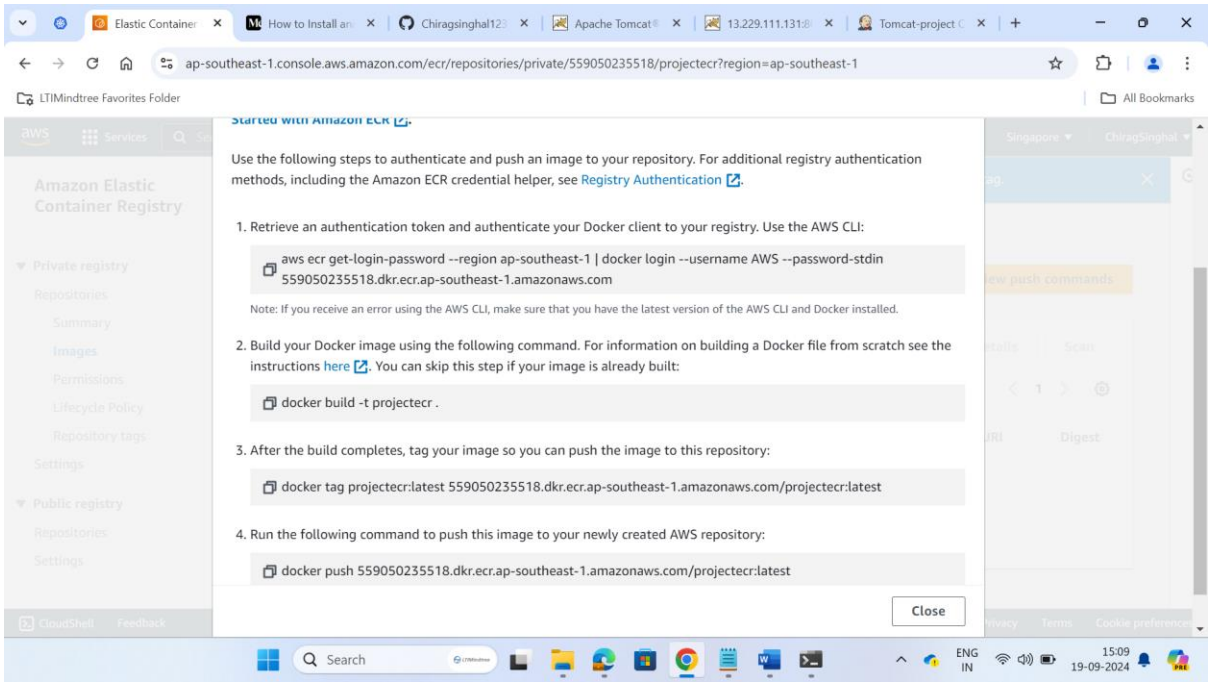


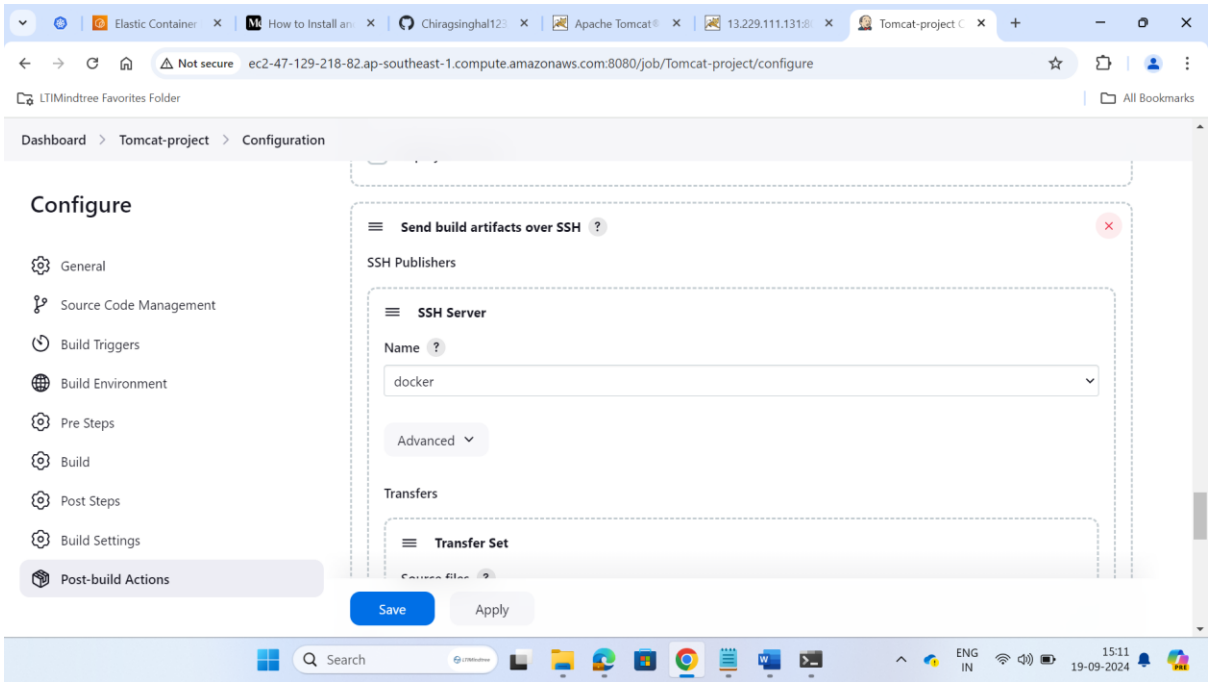
Then Click Create

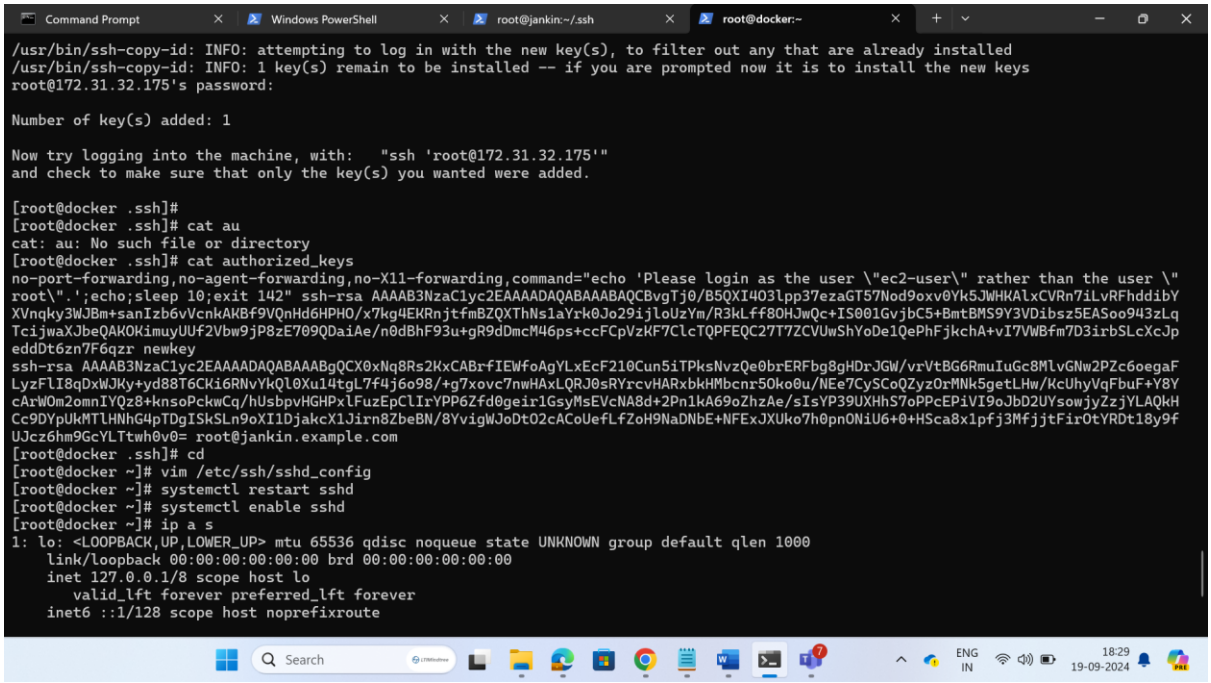




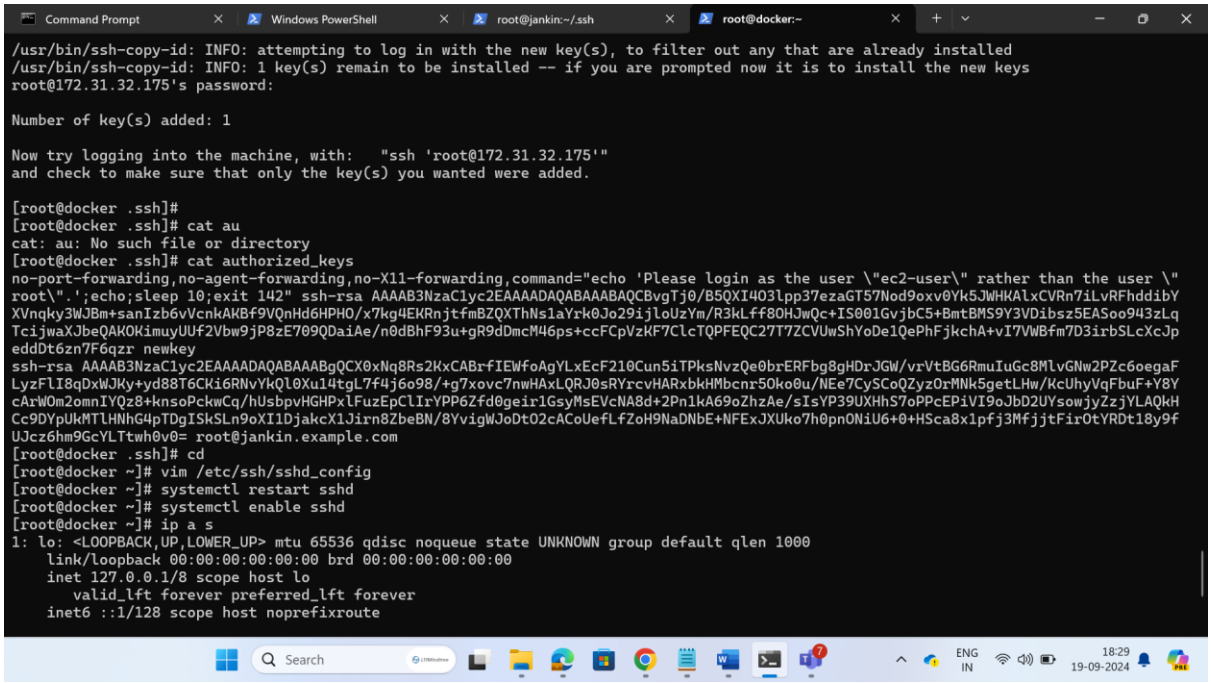
Click view Push Command



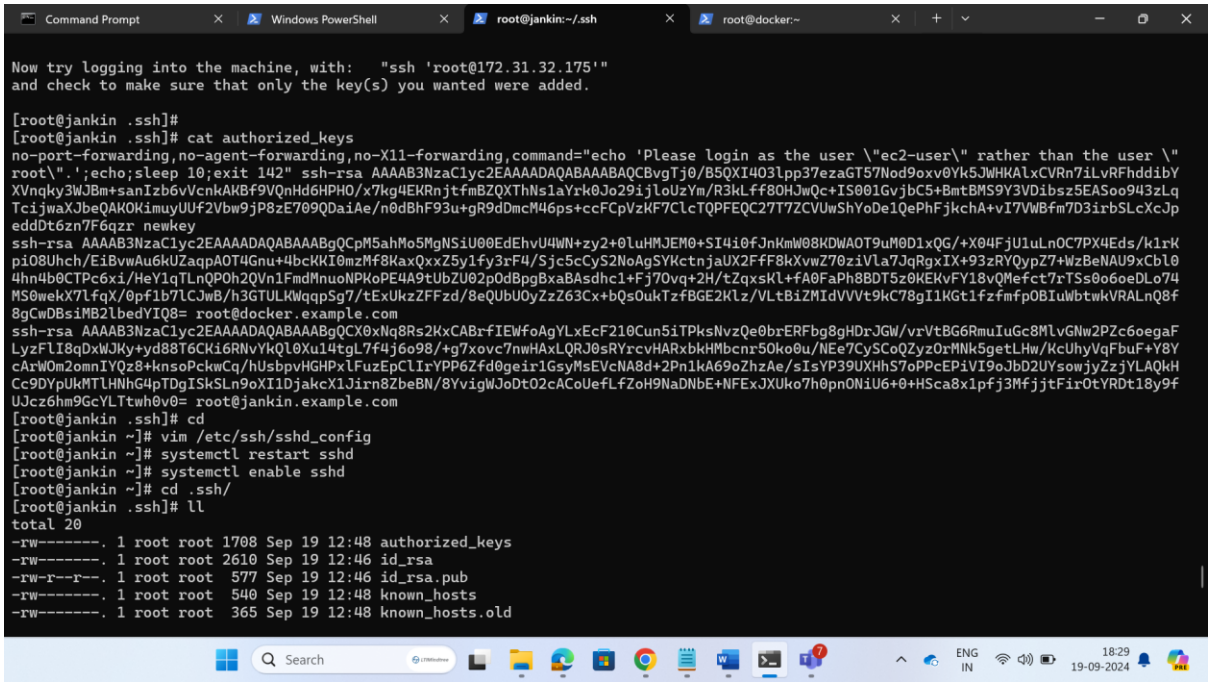


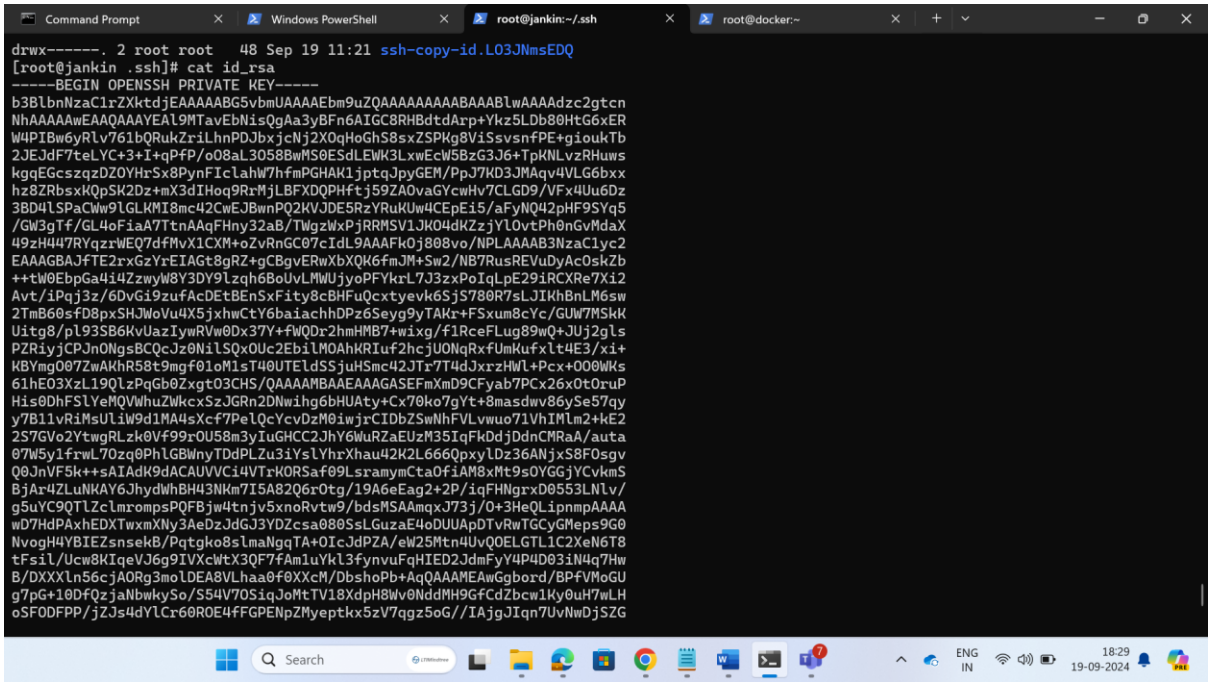


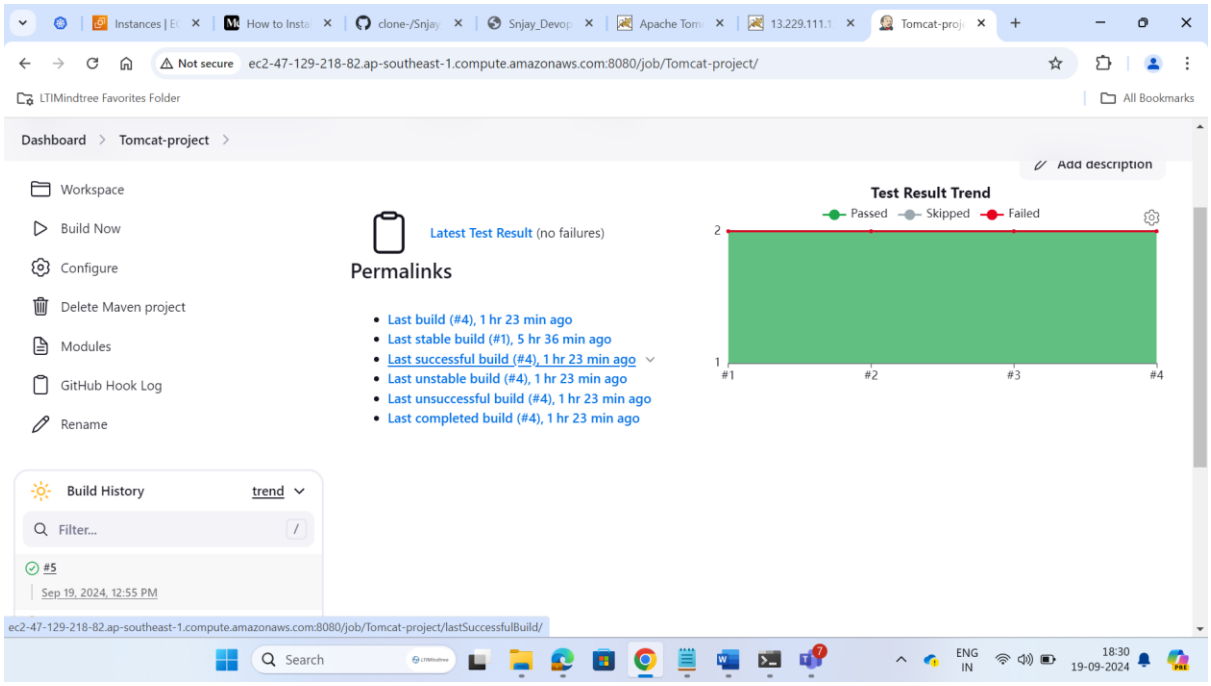
Add the Docker Machine's public key to the Jenkins Machine



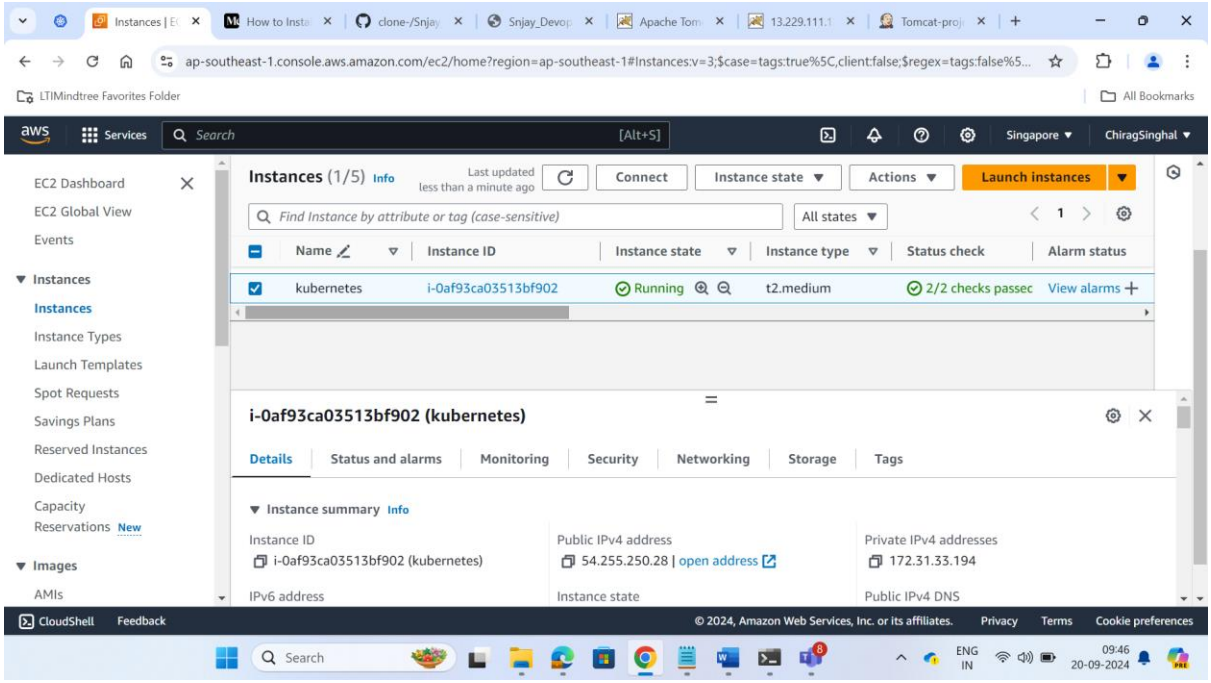
“Add the Jenkin Machine’s public key to the Docker Machine



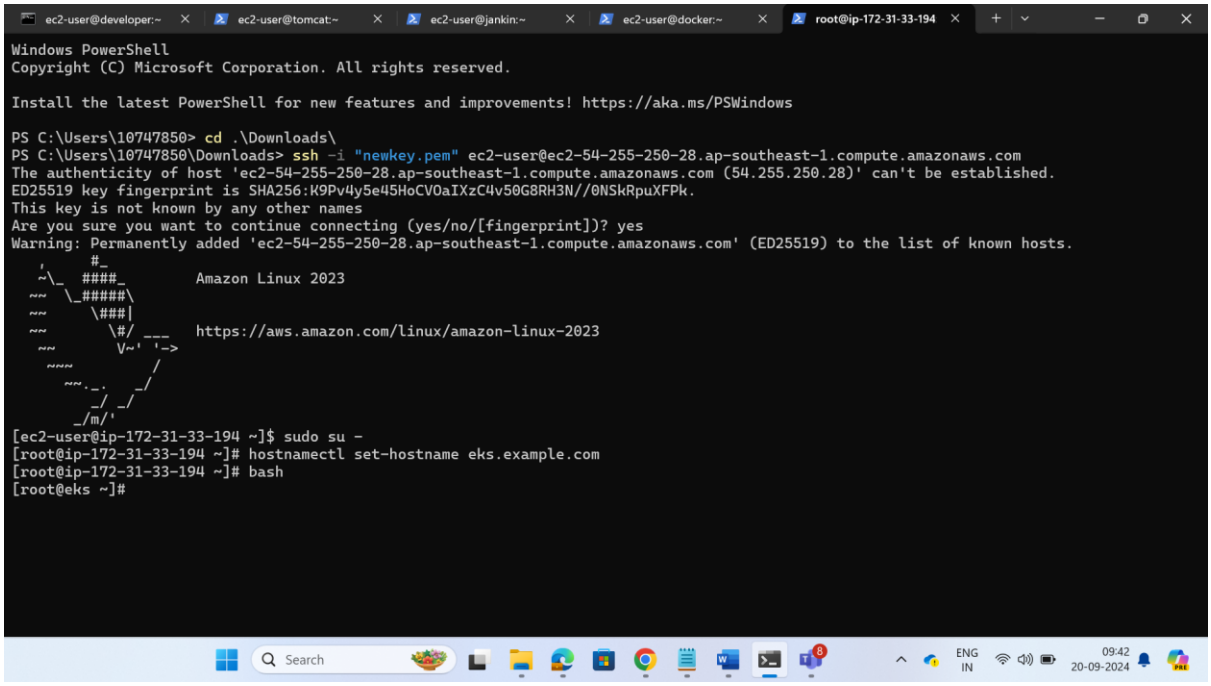




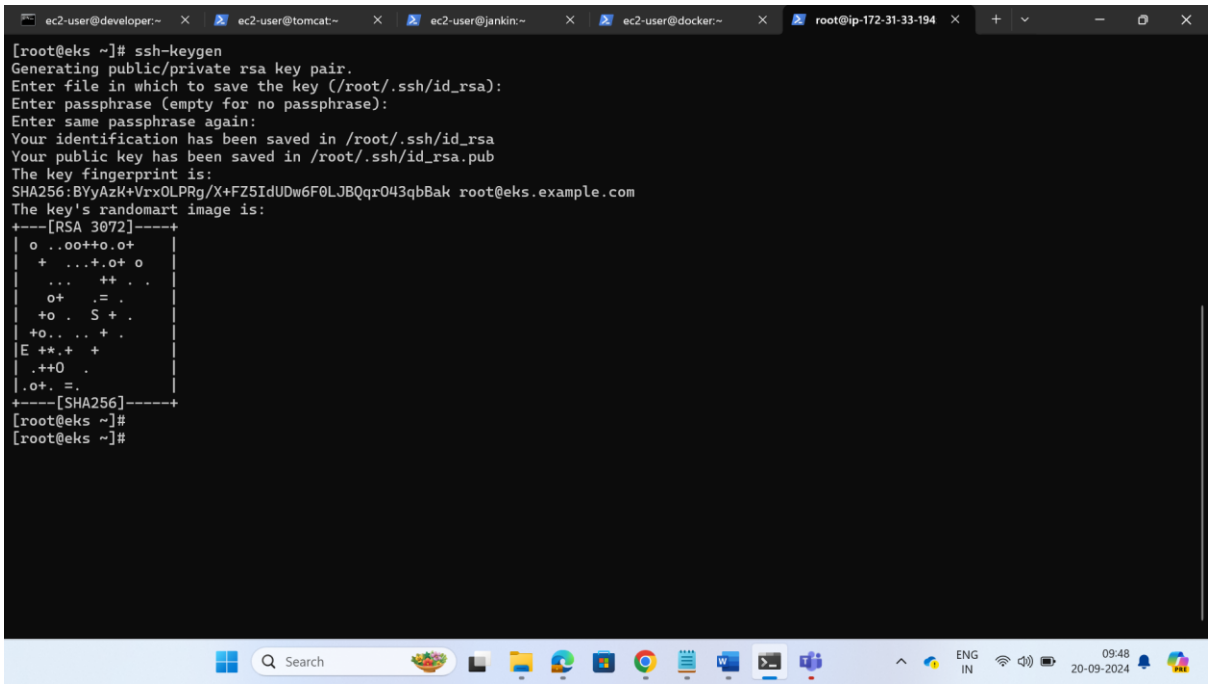
Create Kubernetes Cluster machine



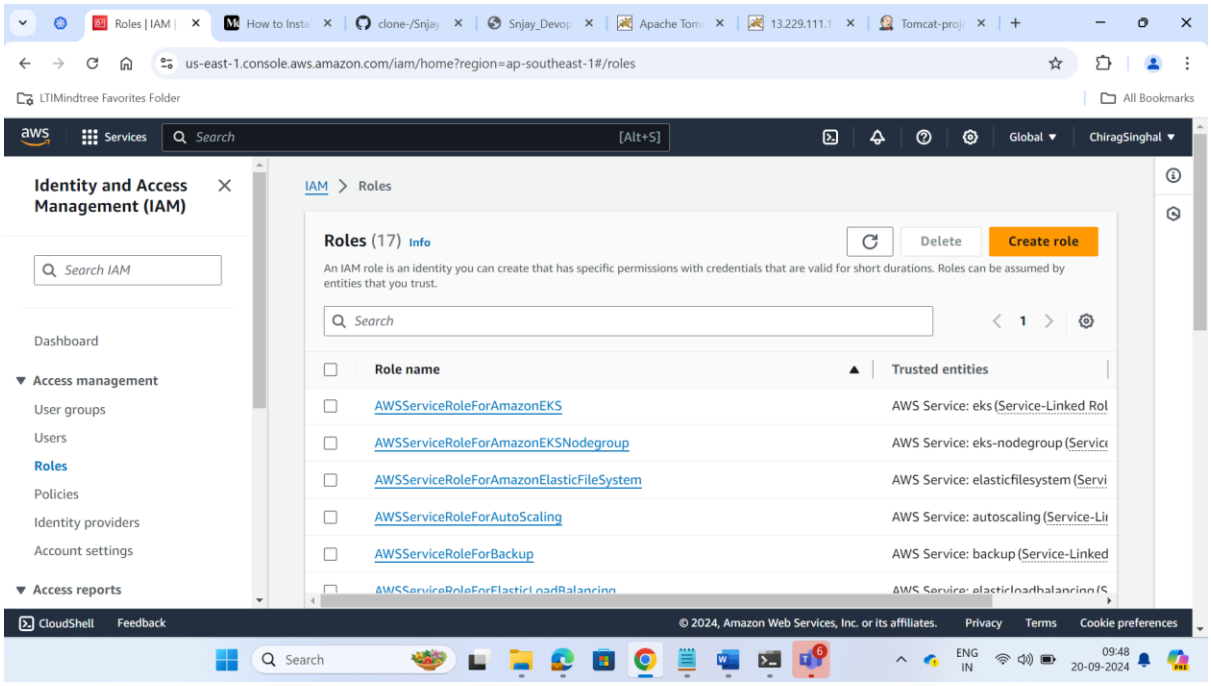
Kubernetes Machine is Ready and Change Hostname



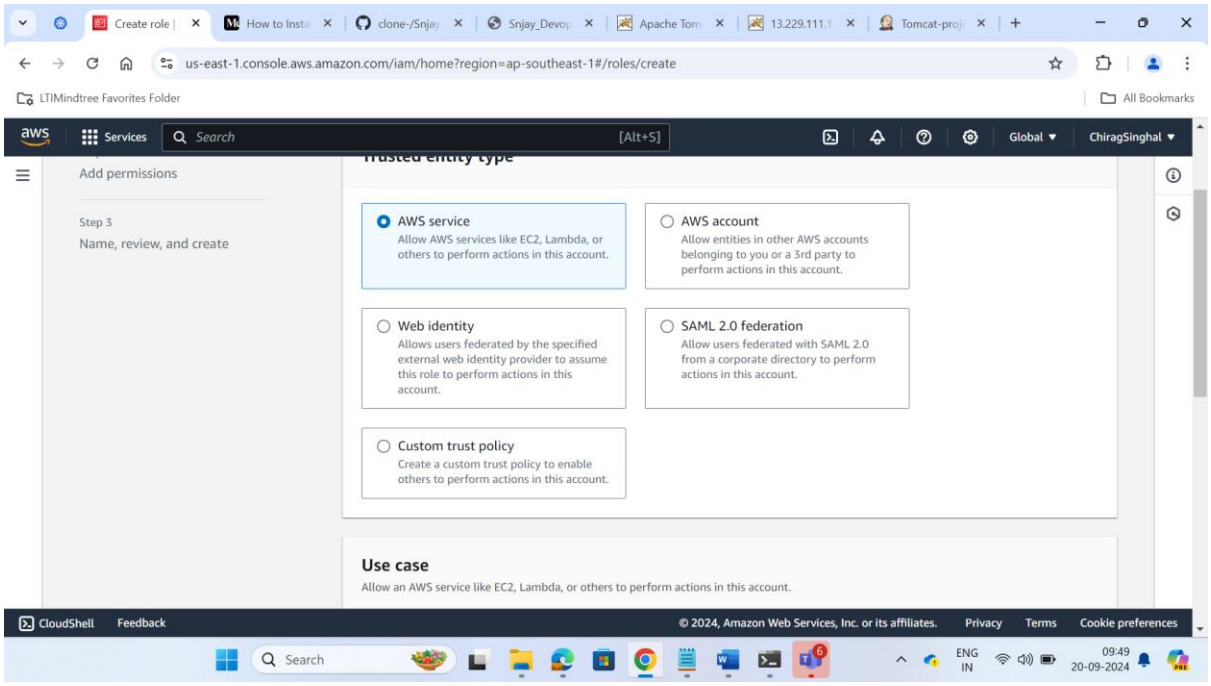
Create ssh-keygen [Mean create public,private IP Address]



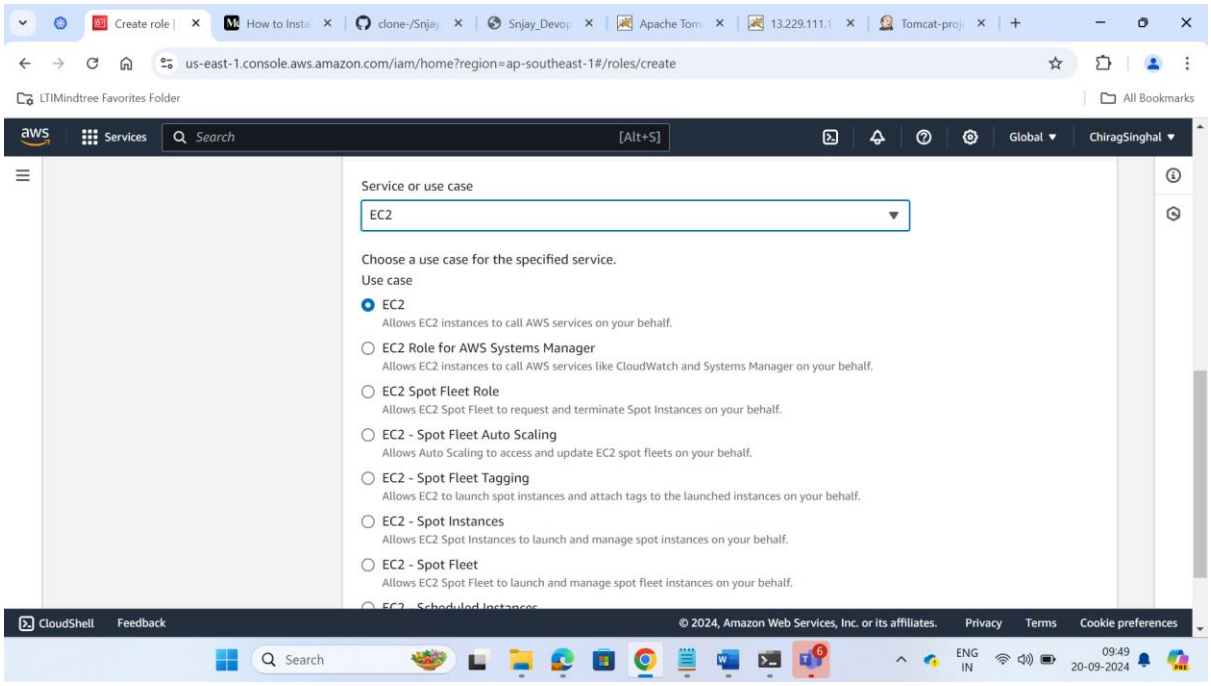
Create Roles :-



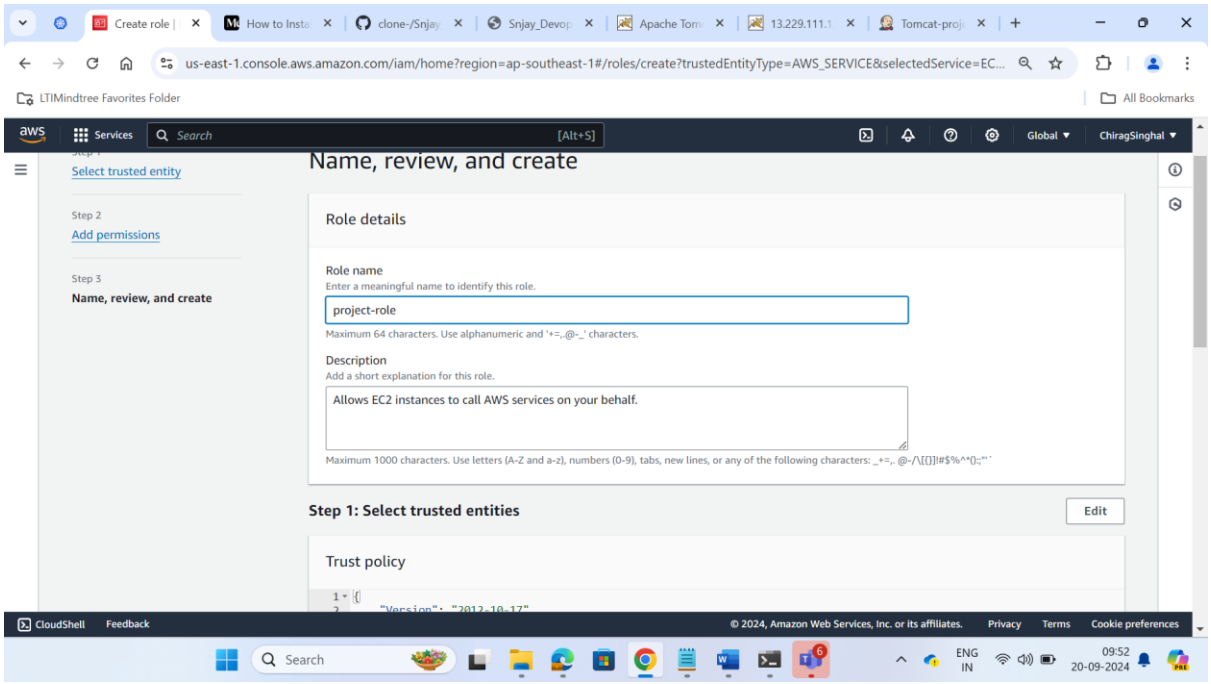
Select AWS service



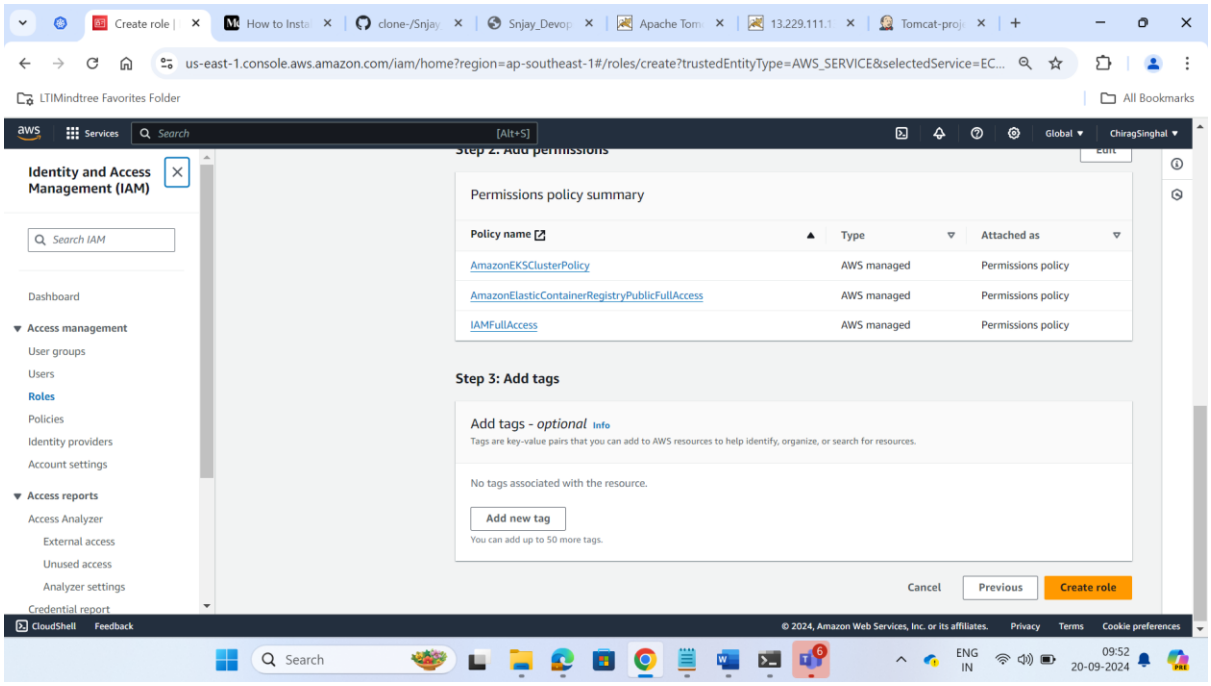
Select EC2



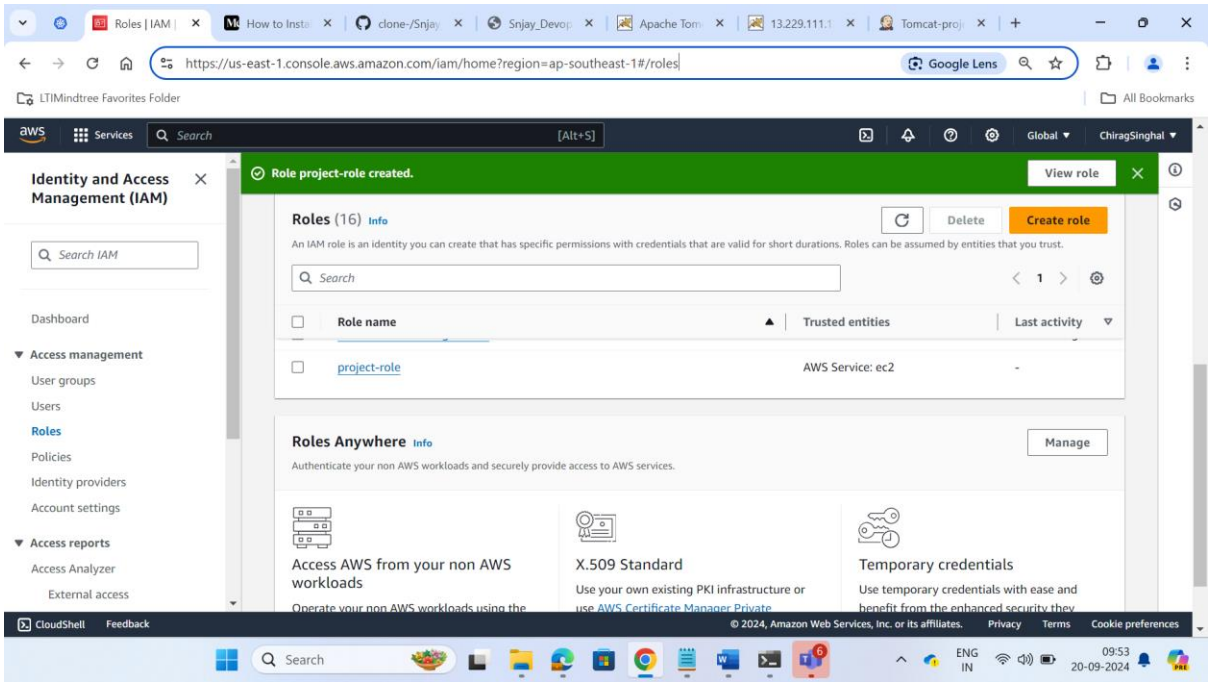
Then Name of the role



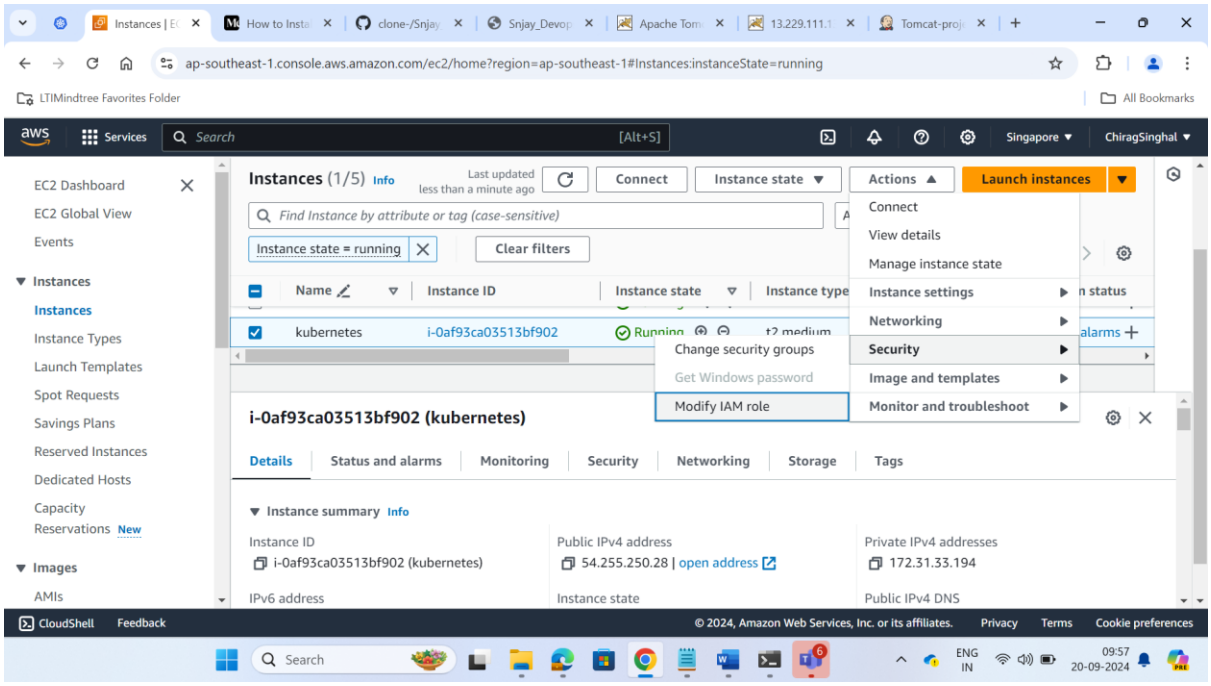
Allow Permission << AmazonEKSClusterPolicy < IAMFullAccess < AmazonElasticContainerRegistryPublicFullAccess >>



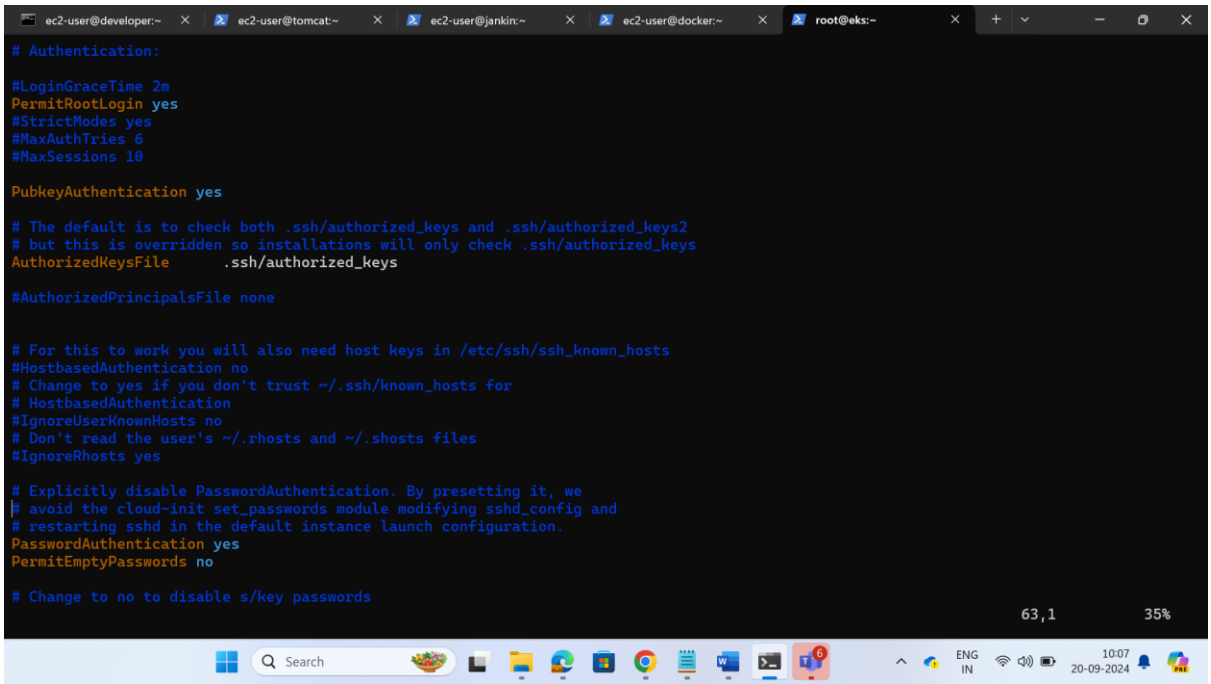
Ready for Role



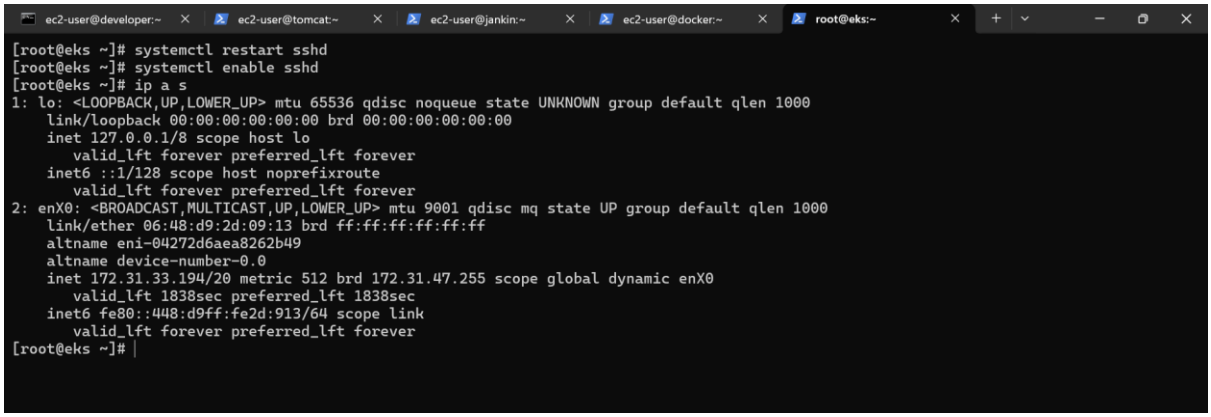
Kubernetes instance add role then goto action and then modify IAM Role



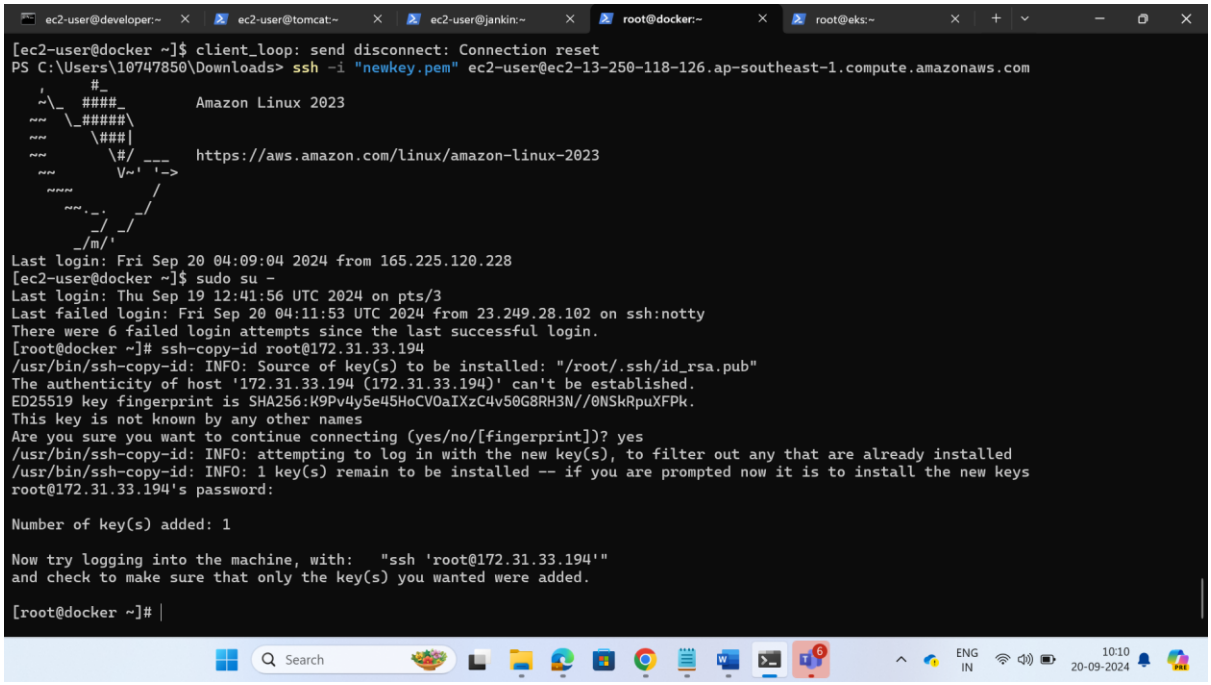
Allow permission in PermitRootLogin and PasswordAuthentication



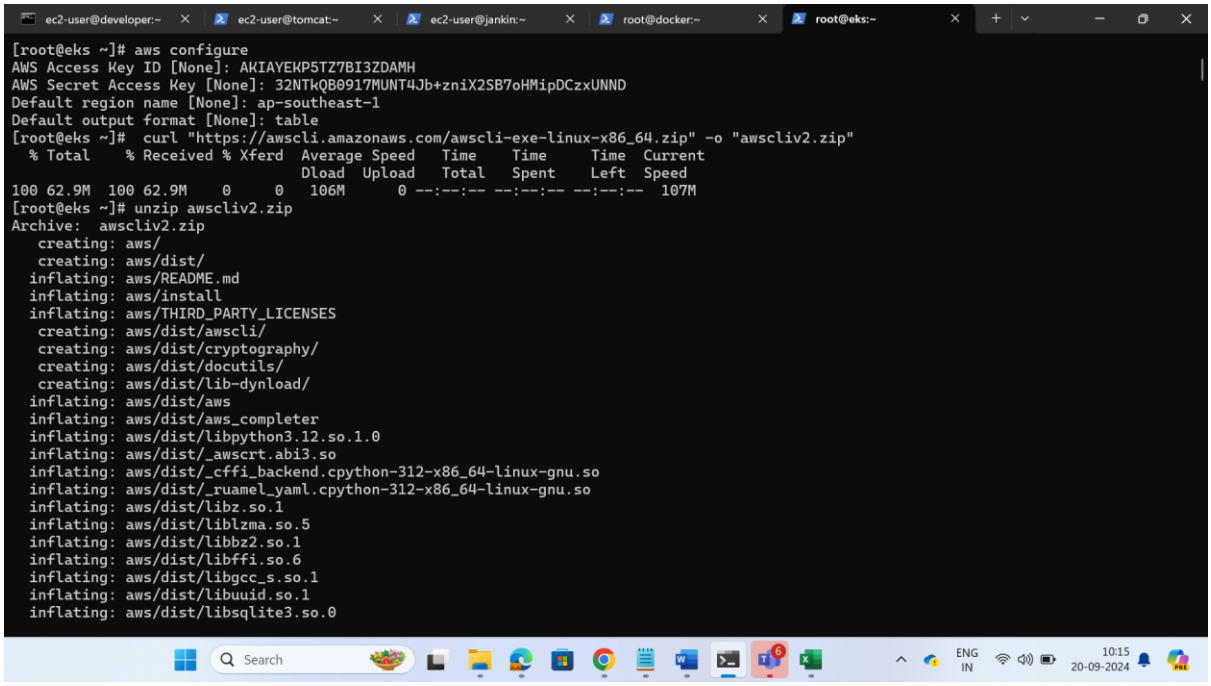
Restart and enable the sshd



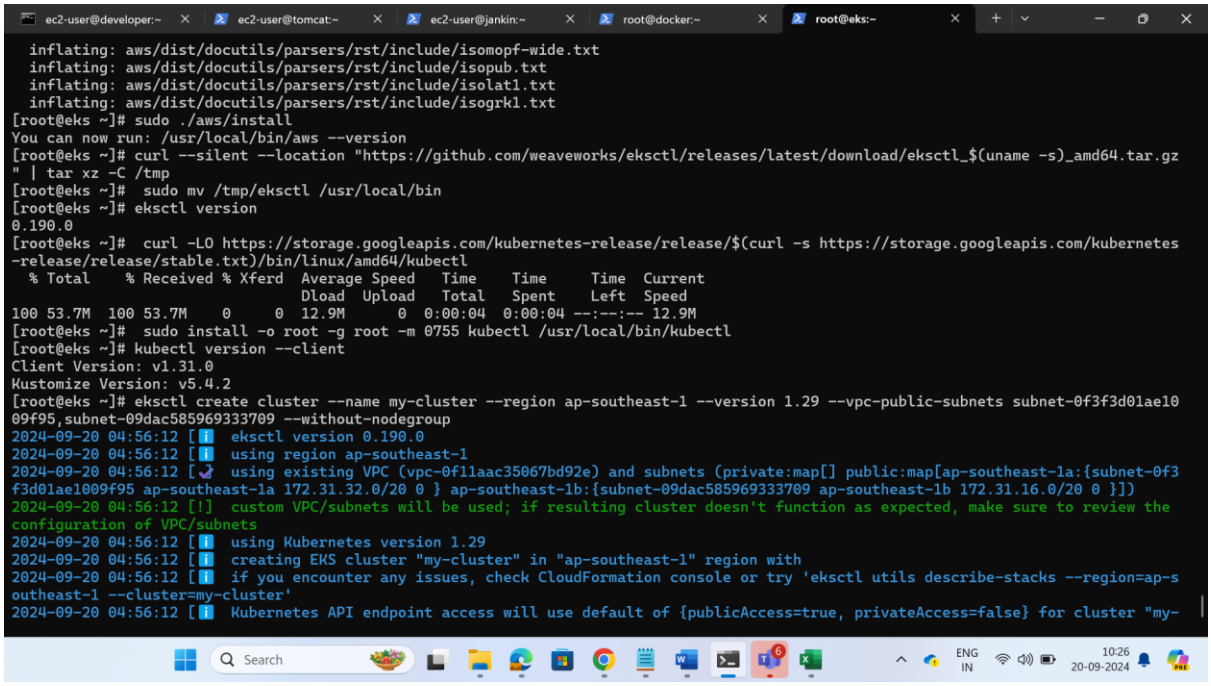
Add the Docker Machine's public key to the Kubernetes Machine."

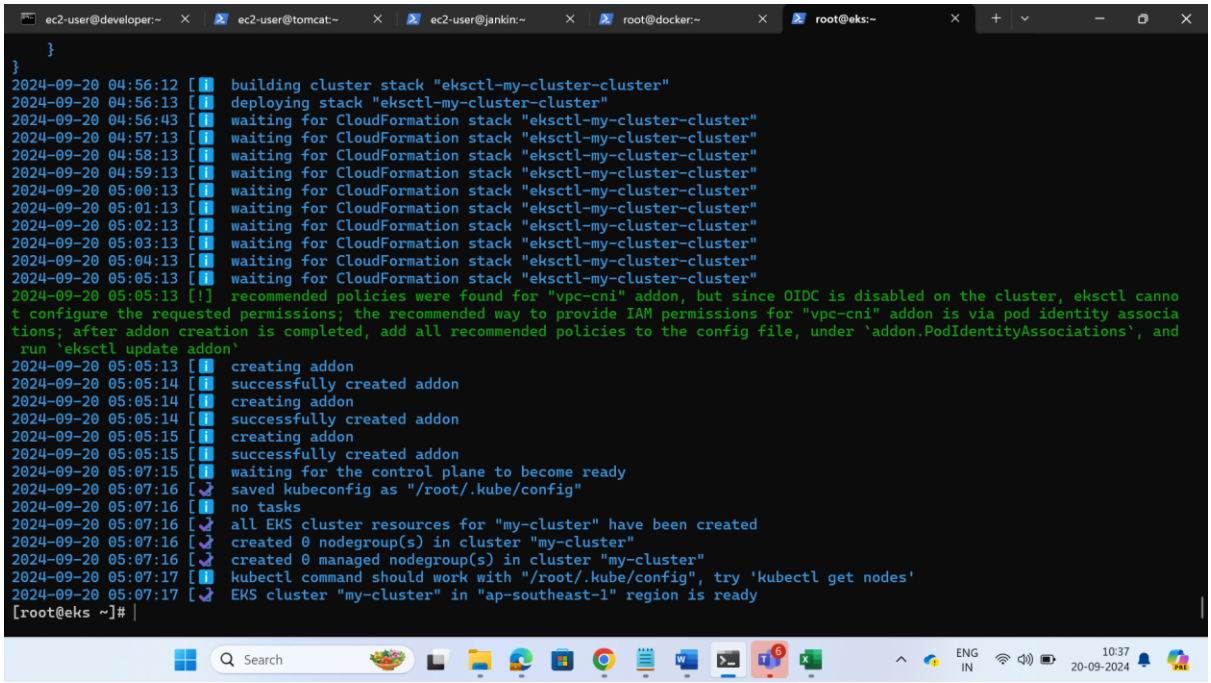


Configure the AWS:



We Created a Cluster





After Creating a Cluster I created a node group

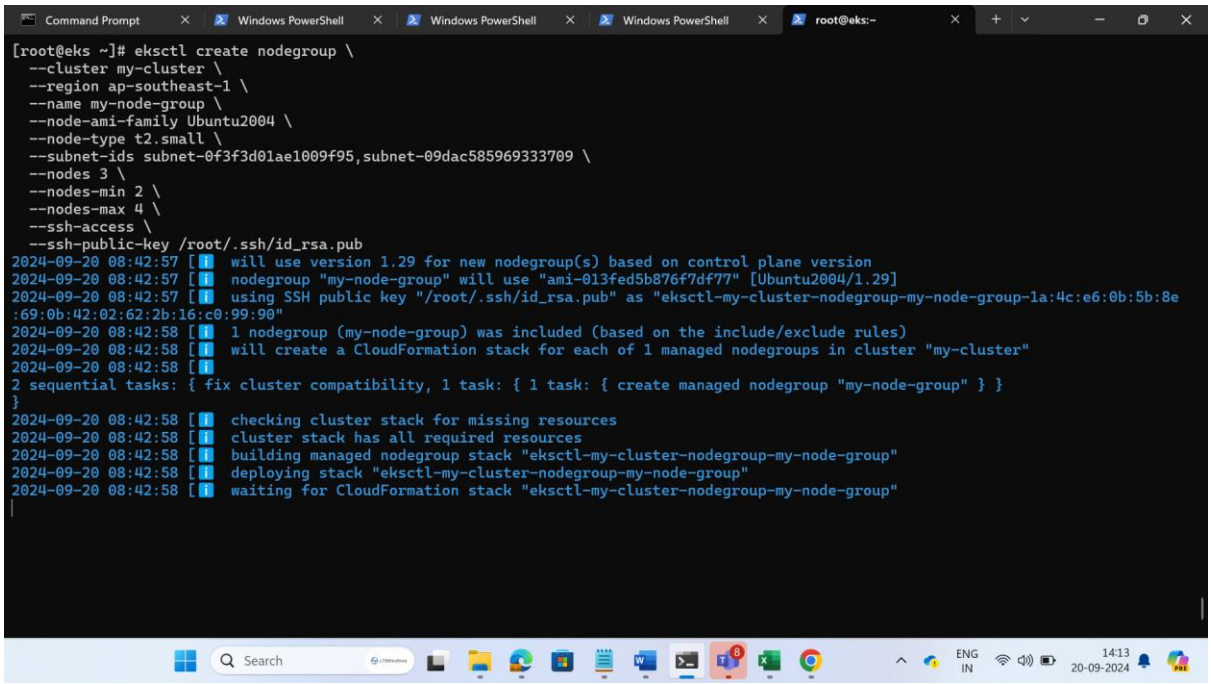
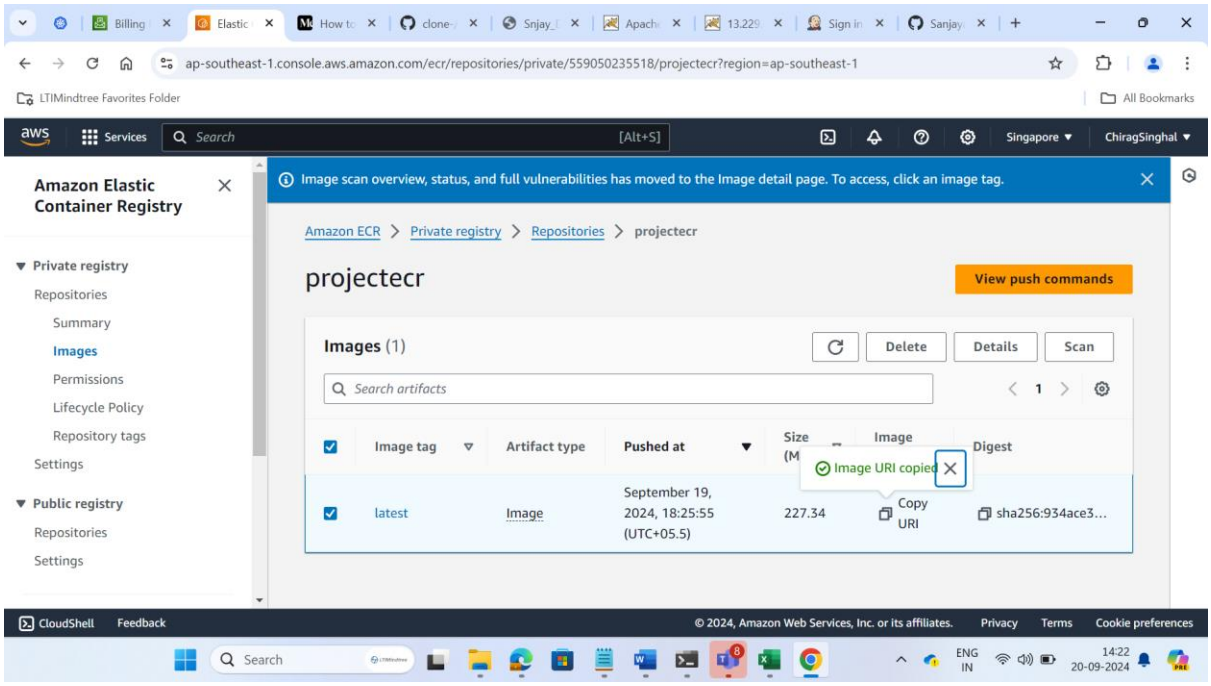
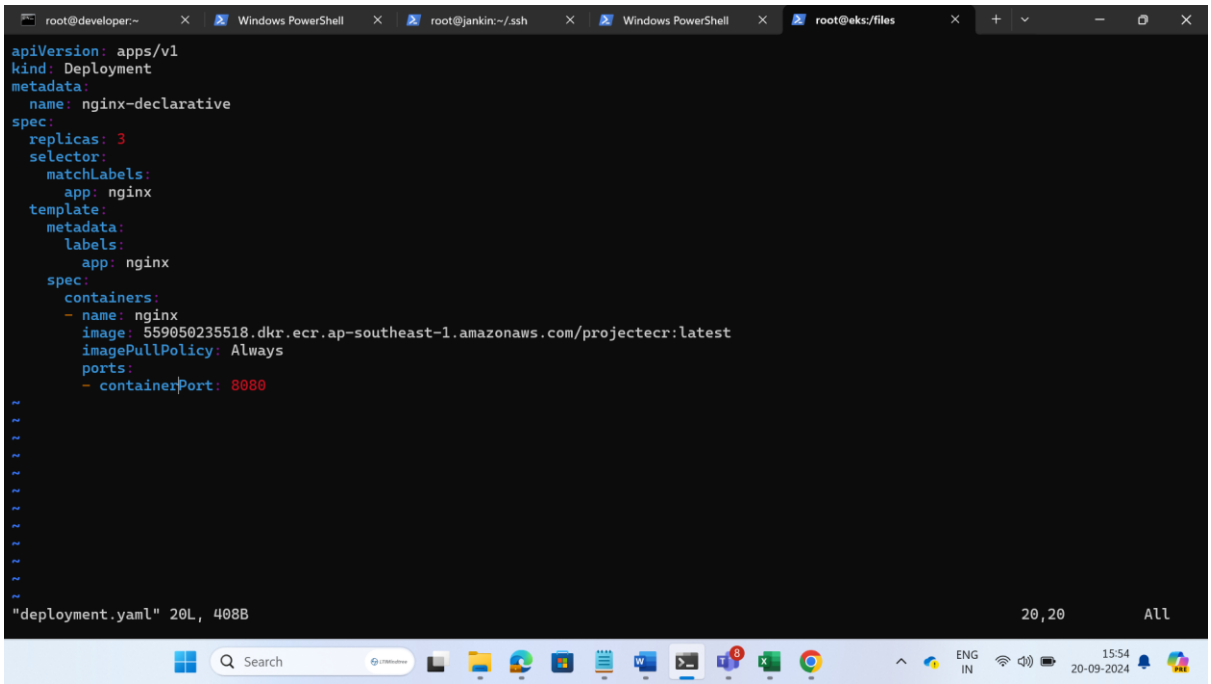


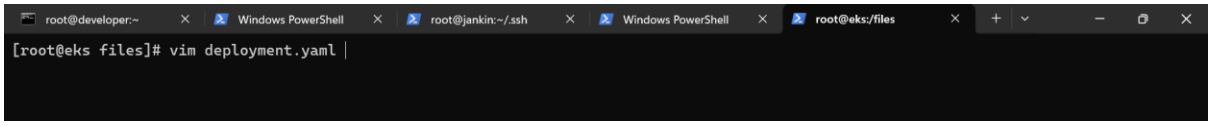
Image is Created by Docker



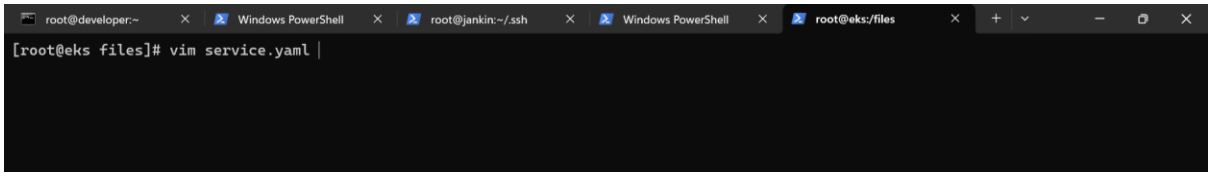
This is the Deployment file



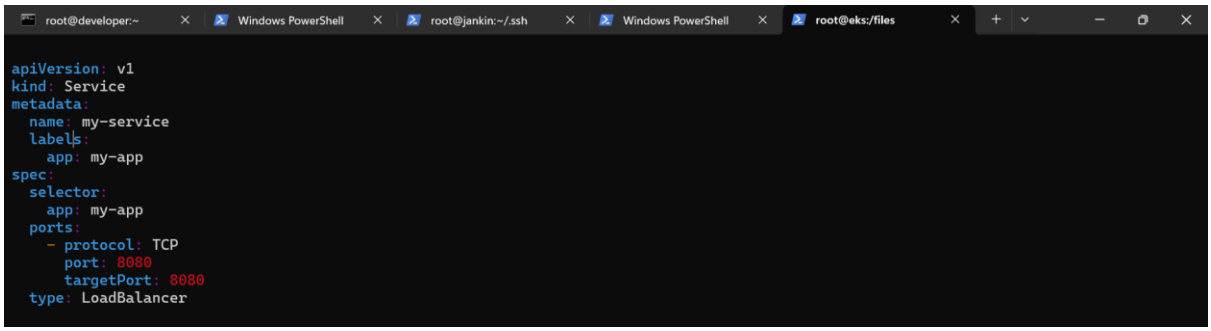
Create Deployment file

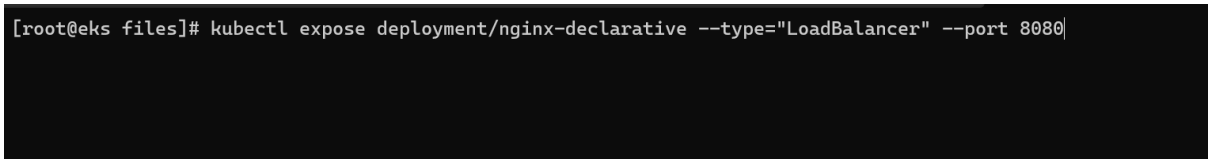


Create service file

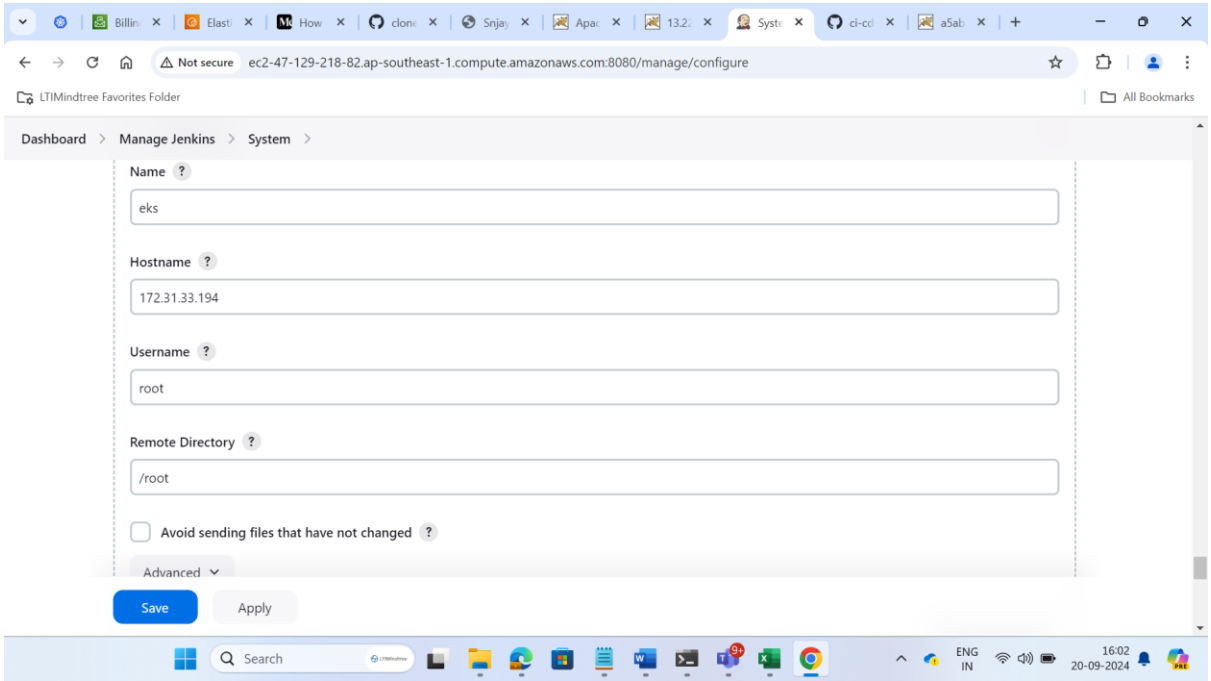


This is ther Server file

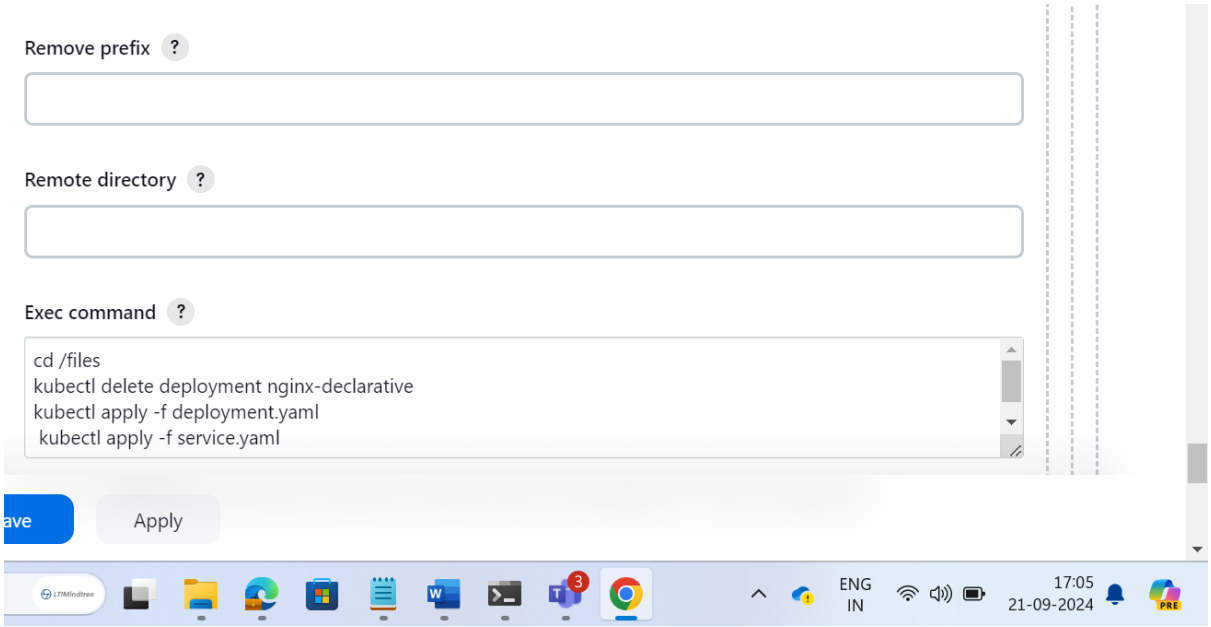




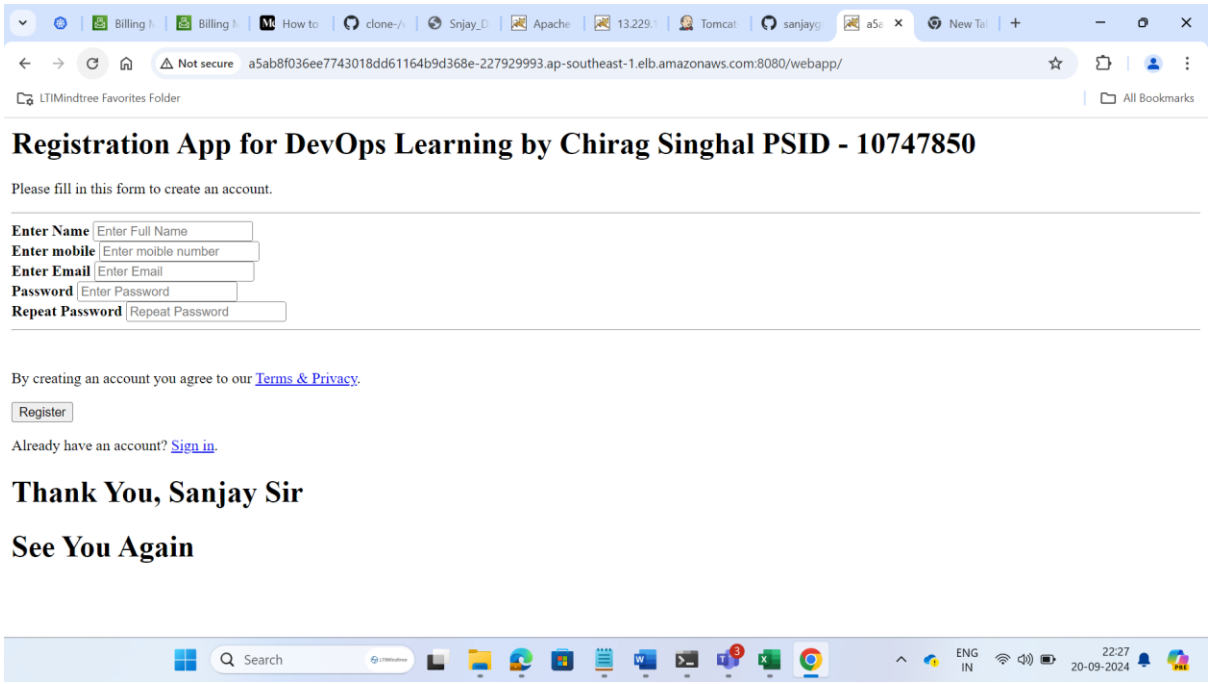
Configure Kubernetes and paste it public IP Address



In Configure Automate the process to delete the old deployment and apply new deploymemt whenever there is a commit :



Deployment is Completed



Final output is displayed by taking the servive ip:8080/webapp/ which is being hosting in the Tomcat server.