**Jenkins Remoting Setup**

 A **Jenkins controller** (On Amazon EC2) with **controller executors disabled** (for safety & isolation).

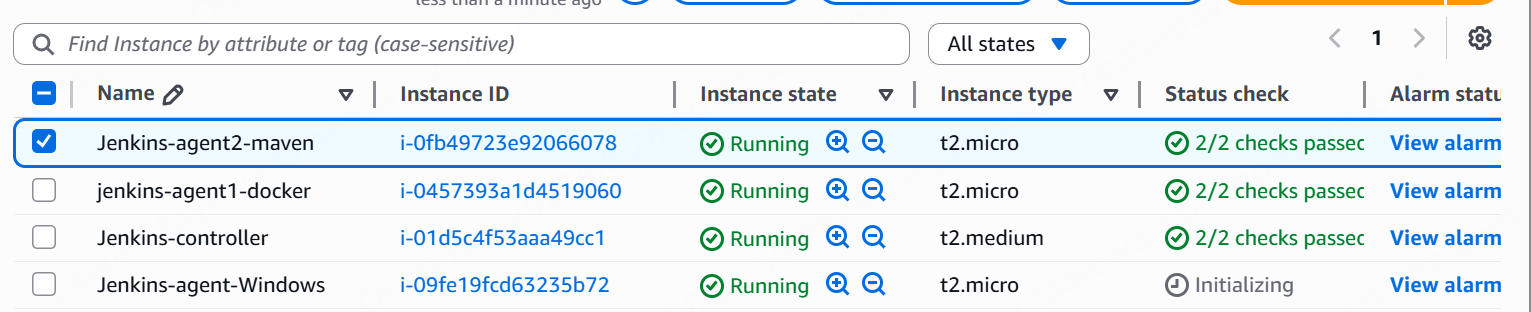
 **Three remote agents** set up via different methods:

** agent-ubuntu-docker → for Docker-based builds.**

** agent-ubuntu-maven → for Java/Maven builds.**

**windows-agent → runs Windows-specific tasks** A pipeline that **distributes builds** across these agents

 **Security hardening**: least‑privilege user, credentials hygiene, and locked‑down ports.

****

**Phase 1**

Prerequisites & Install Jenkins (Controller)

**Prerequisites.**

* Ubuntu 22.04 LTS on AWS
* Java 17
* Jenkins server

**Steps.**

1. **Launch EC2 Instance.**

 In AWS Console → EC2 → Launch Instance.

 Name: jenkins-controller.

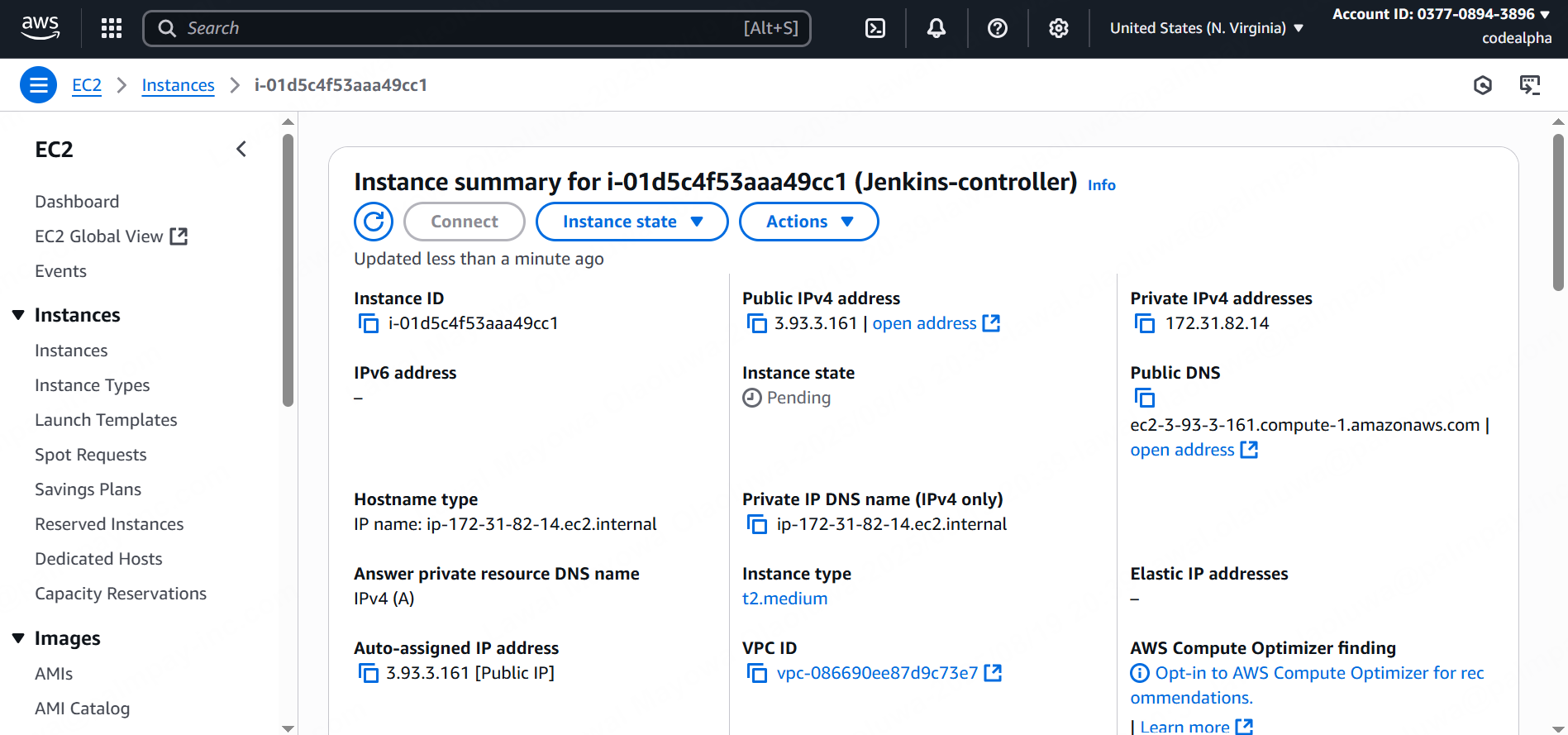
 AMI: Ubuntu 22.04 LTS.

 Instance type: t2.medium

 Security Group:

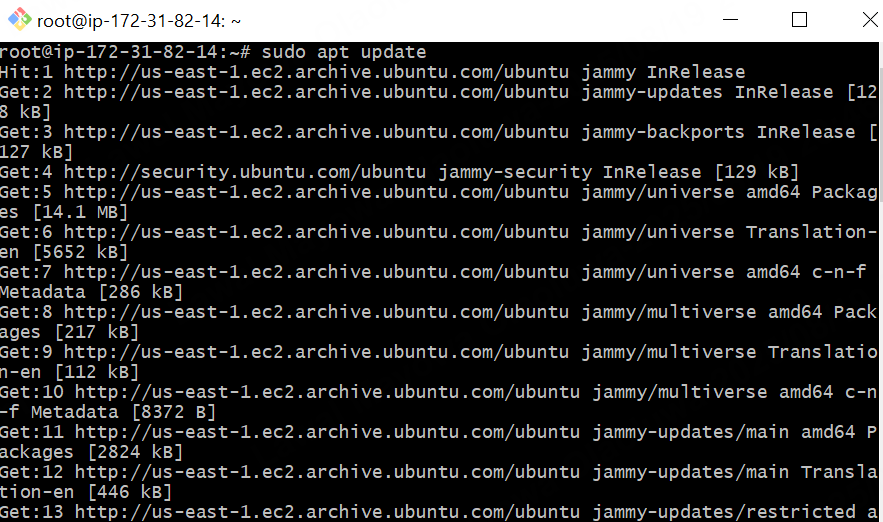
Allow **22 (SSH)** from your IP.

Allow **8080 (Jenkins web)** from your IP (or 0.0.0.0/0 if testing).

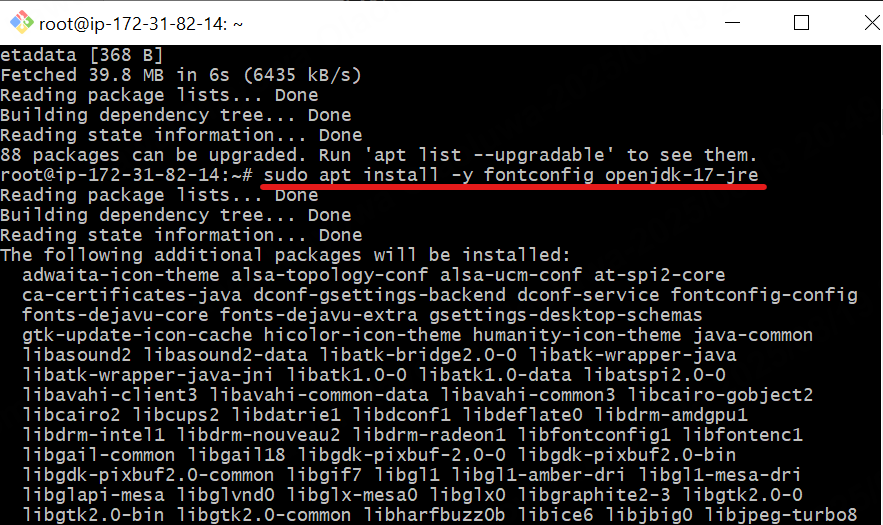
****

1. **Update packages and install java 17**

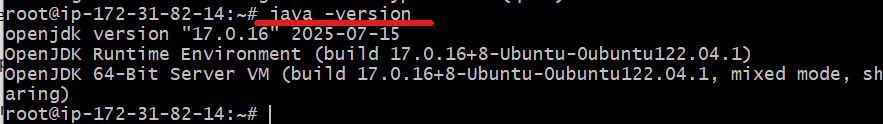
Using “sudo apt update”



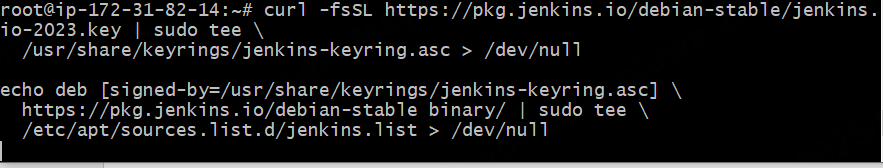
Install java 17



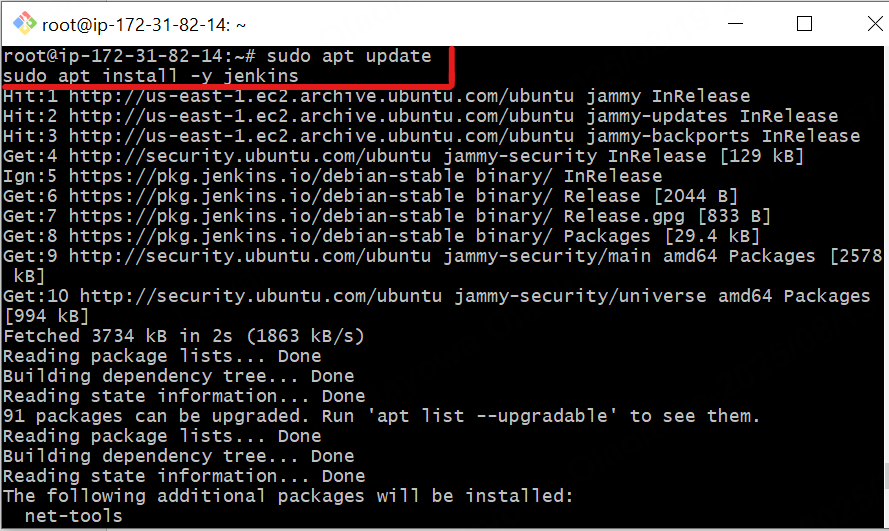
Check java version.



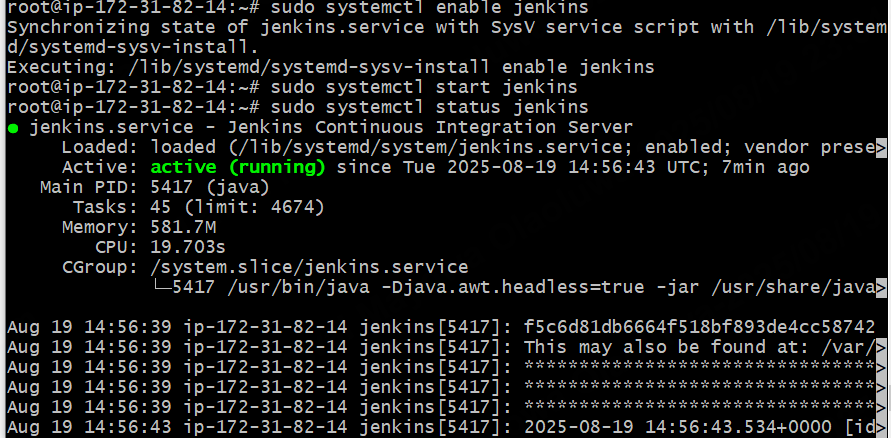
1. **Add Jenkins Repository**

****

1. **Install Jenkins**

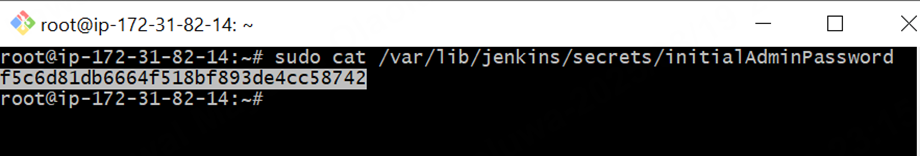
****

1. **Enable, Start and check Jenkins Status**

****

1. **Access Jenkins**

sudo cat /var/lib/jenkins/secrets/initialAdminPassword in CLI for First-time setup:

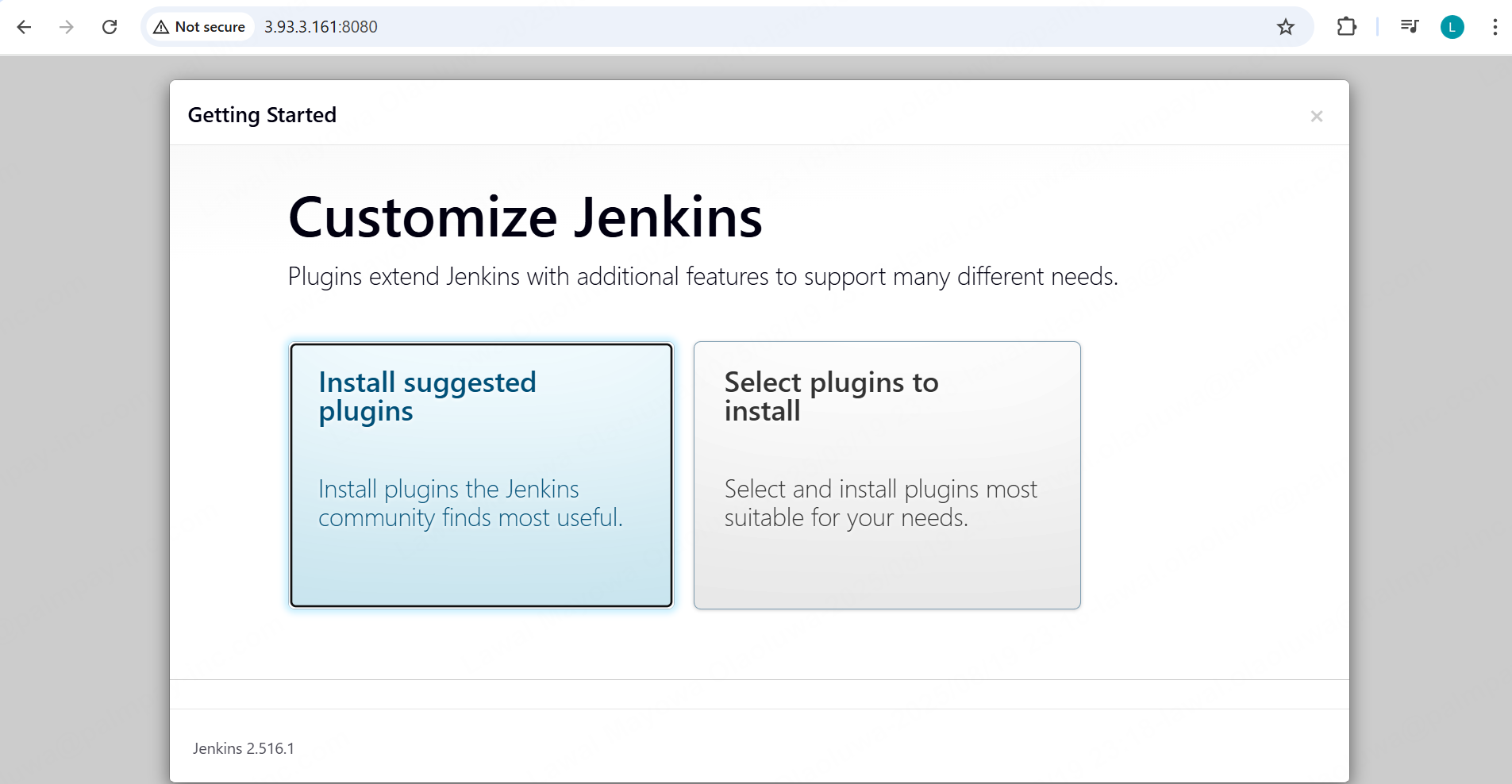
****

**In browser**: http://<controller-public-ip>:8080

****

Paste in  **“Administration password”**

**Pick : Install suggested plugins.**

****

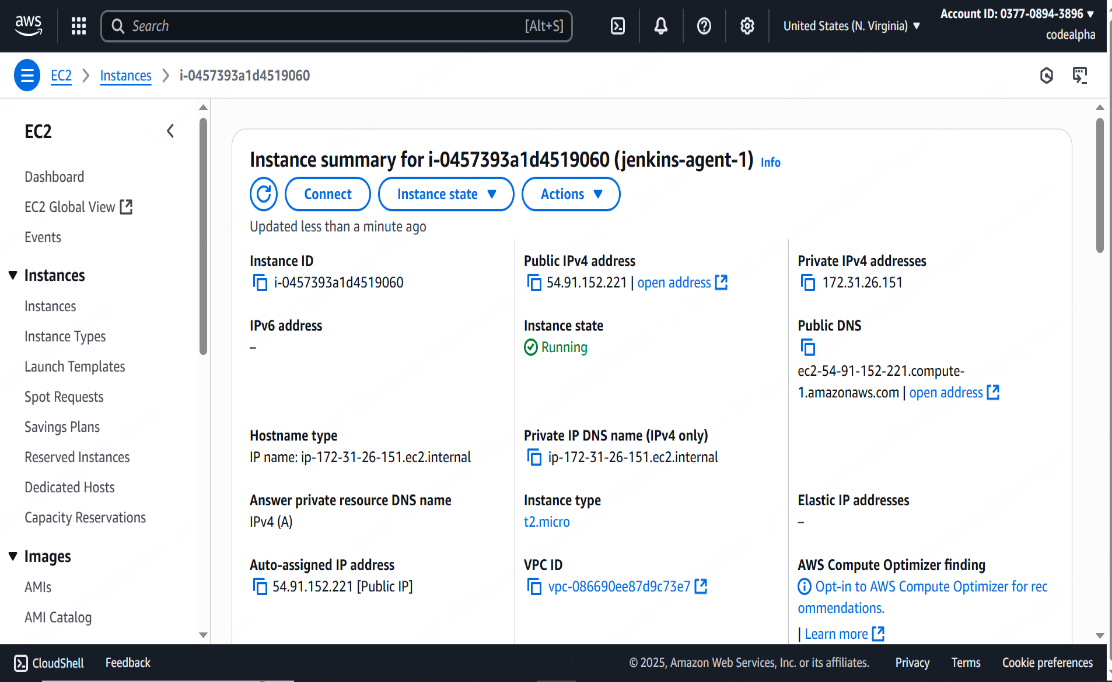
**Phase 2**

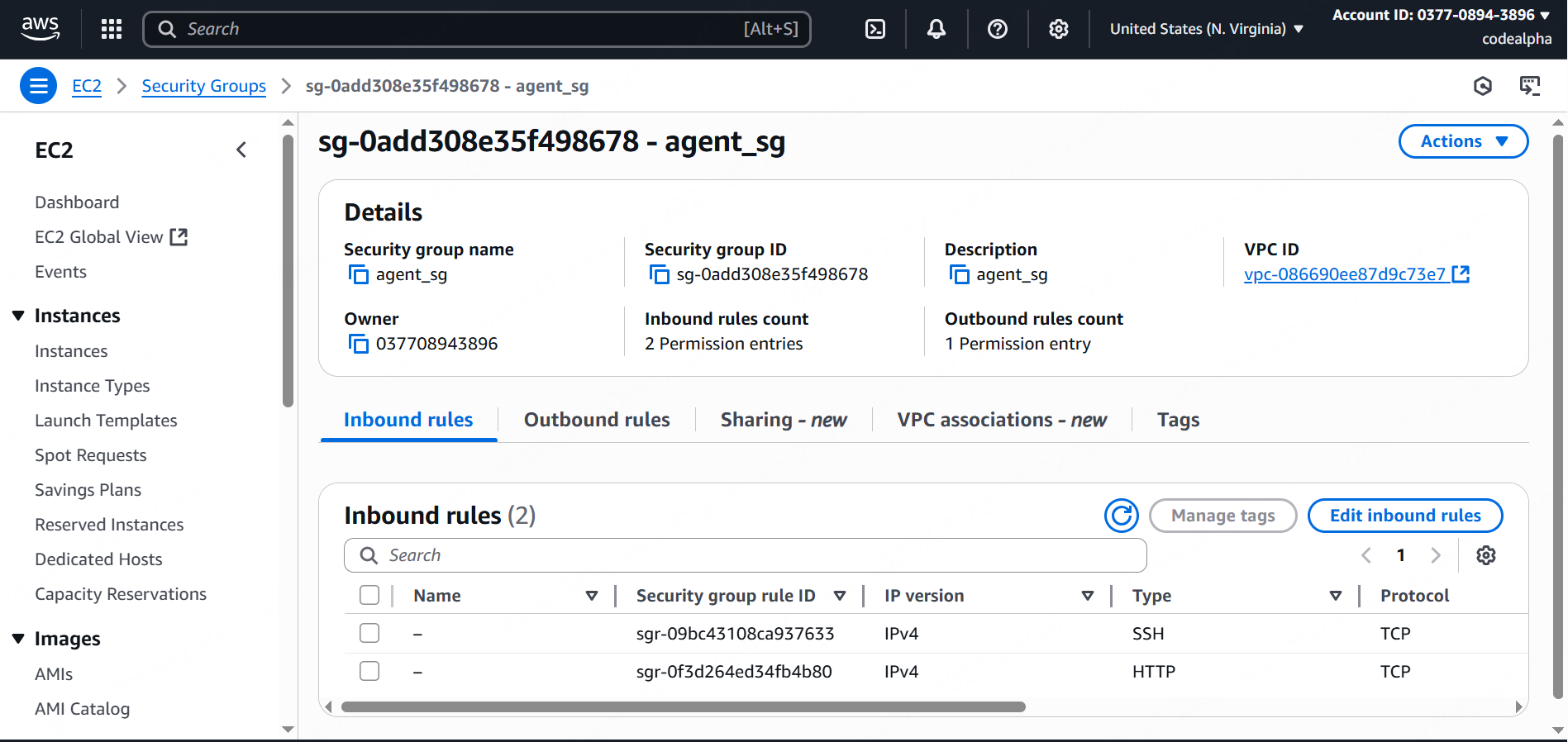
**Jenkins Agents Setup**

Option A – SSH-based linux docker Agent.

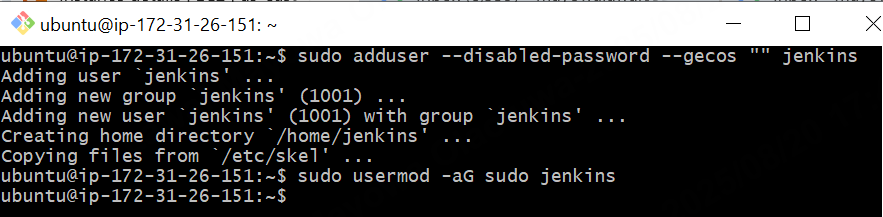
Steps

1. Prepare the Remote Machine (Agent Node)  
   Ubuntu agent Machine on AWS

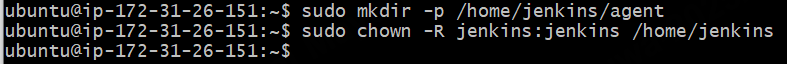
****

****

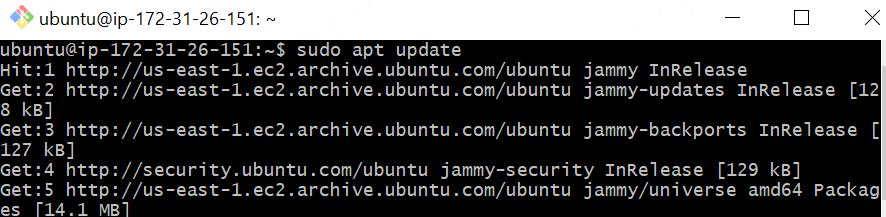
1. Prep into the agent machine and add Jenkins user. And also to Sudo group.

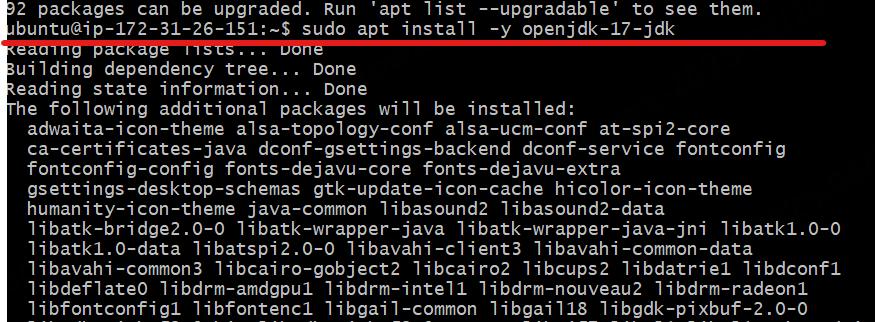


1. Create a clean work dir for Jenkins agent

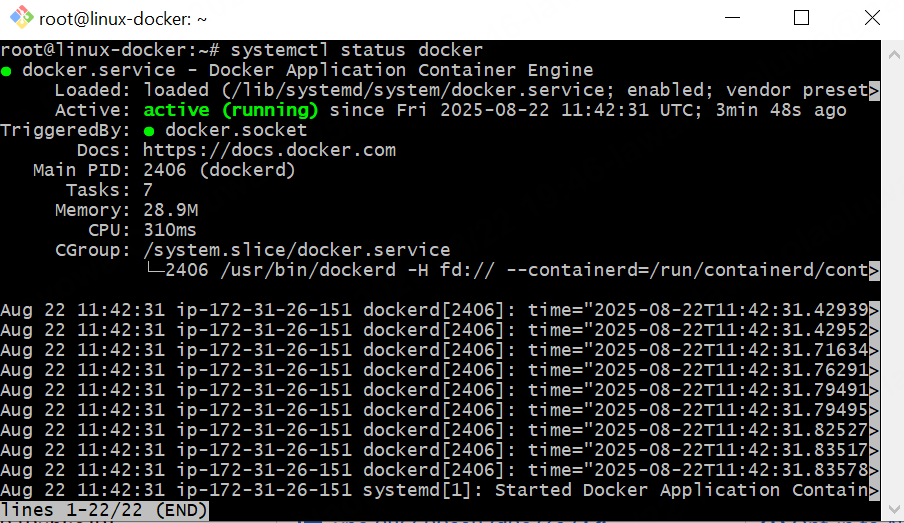


1. Install Java (OpenJDK 17)

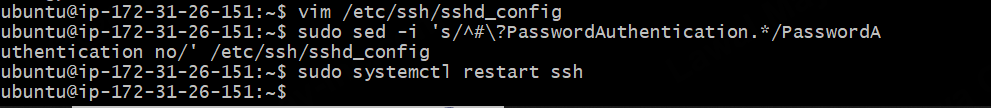




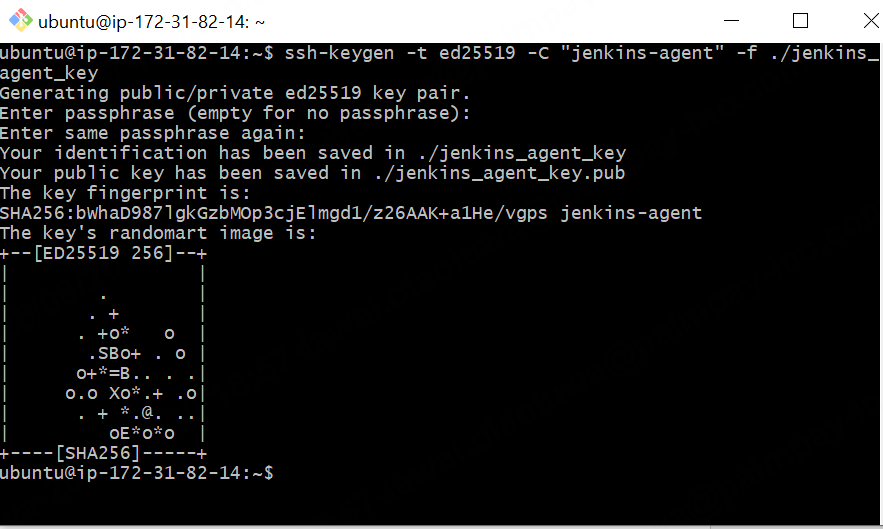
1. Install Docker and enable. (use docker documentation on any of your websites)



1. Harden SSH a bit (optional but good)



1. Create an SSH key for Jenkins Controller to agent.

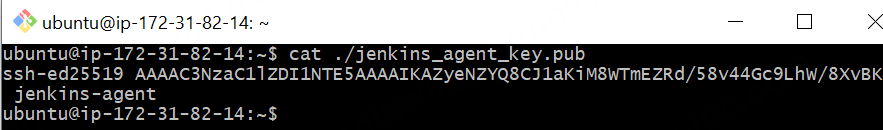


**This creates: jenkins\_agent\_key (private) and jenkins\_agent\_key.pub (public)**

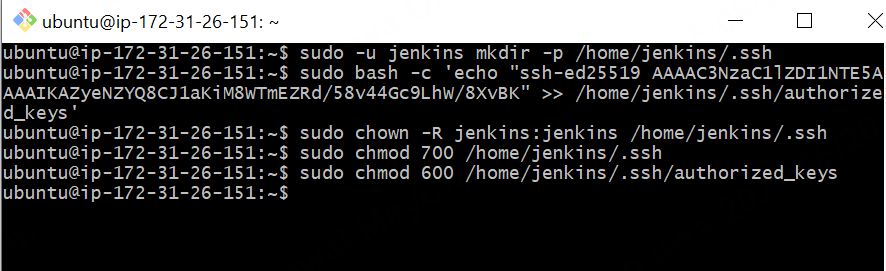
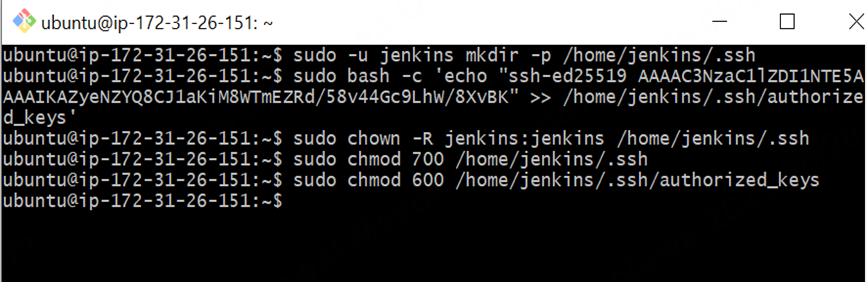
****

1. Add the **public** key to the agent’s jenkins user

* **Copy the public key from controller**

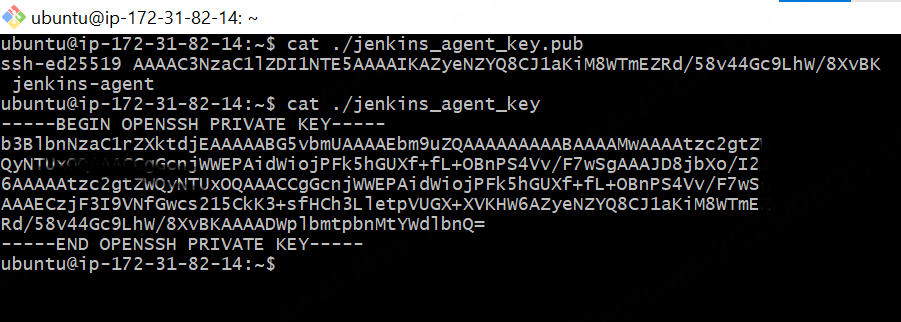
****

* **Paste in Agent**

****

1. Add the private key to Jenkins Controller(UI)

* **Copy private key from Jenkins Controller CLI**

****

* **Paste in Jenkins Controller UI**

****

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

**Credentials**

**Kind:** SSH Username with private key

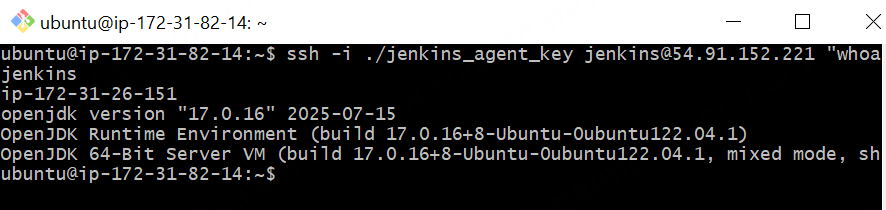
**Username:** jenkins

**Private Key:** “Enter directly” → paste contents of jenkins\_agent\_key

**ID/Description:** agent-ssh-jenkins-ed25519

1. Test SSH manually (from controller)

**ssh -i ./jenkins\_agent\_key jenkins@<AGENT\_IP> "whoami && hostname && java -version"**

****

1. Add Remote Node in Jenkins UI

* Create Node

Jenkins dashboard→ **Manage Jenkins**→ Nodes →click new Node→ Input name and select permanent agent→ Create

* Configure Node Details

**Name**: agent-ubuntu-docker

**# of executors**: 1

**Remote root directory**: /home/jenkins/agent

**Labels**: linux-docker

**Usage**: Only build jobs with label expressions

**Launch method:** Launch agents via SSH

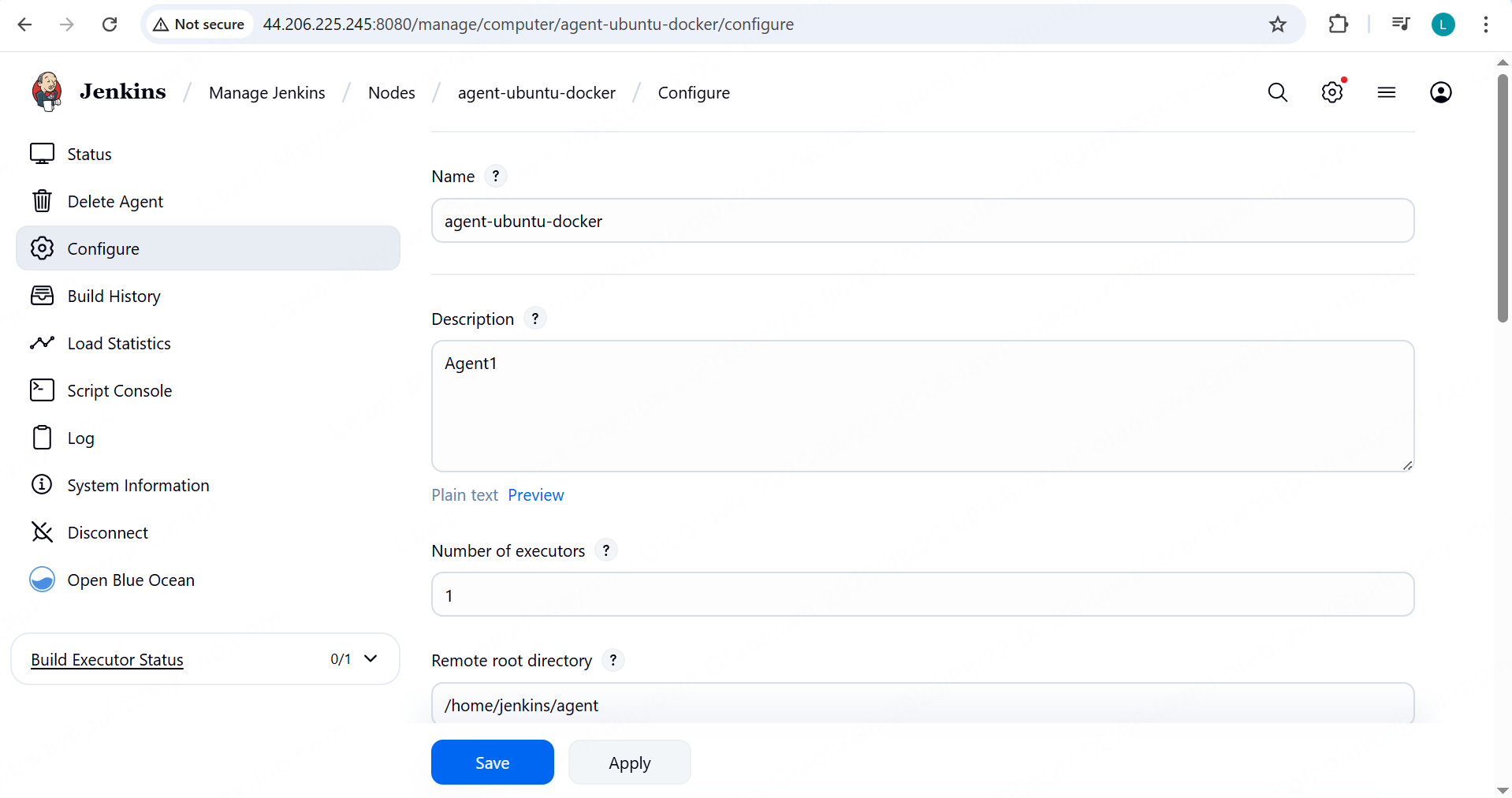
**Host:** Agent Private IP

**Credentials**: agent-ssh-jenkins-ed25519

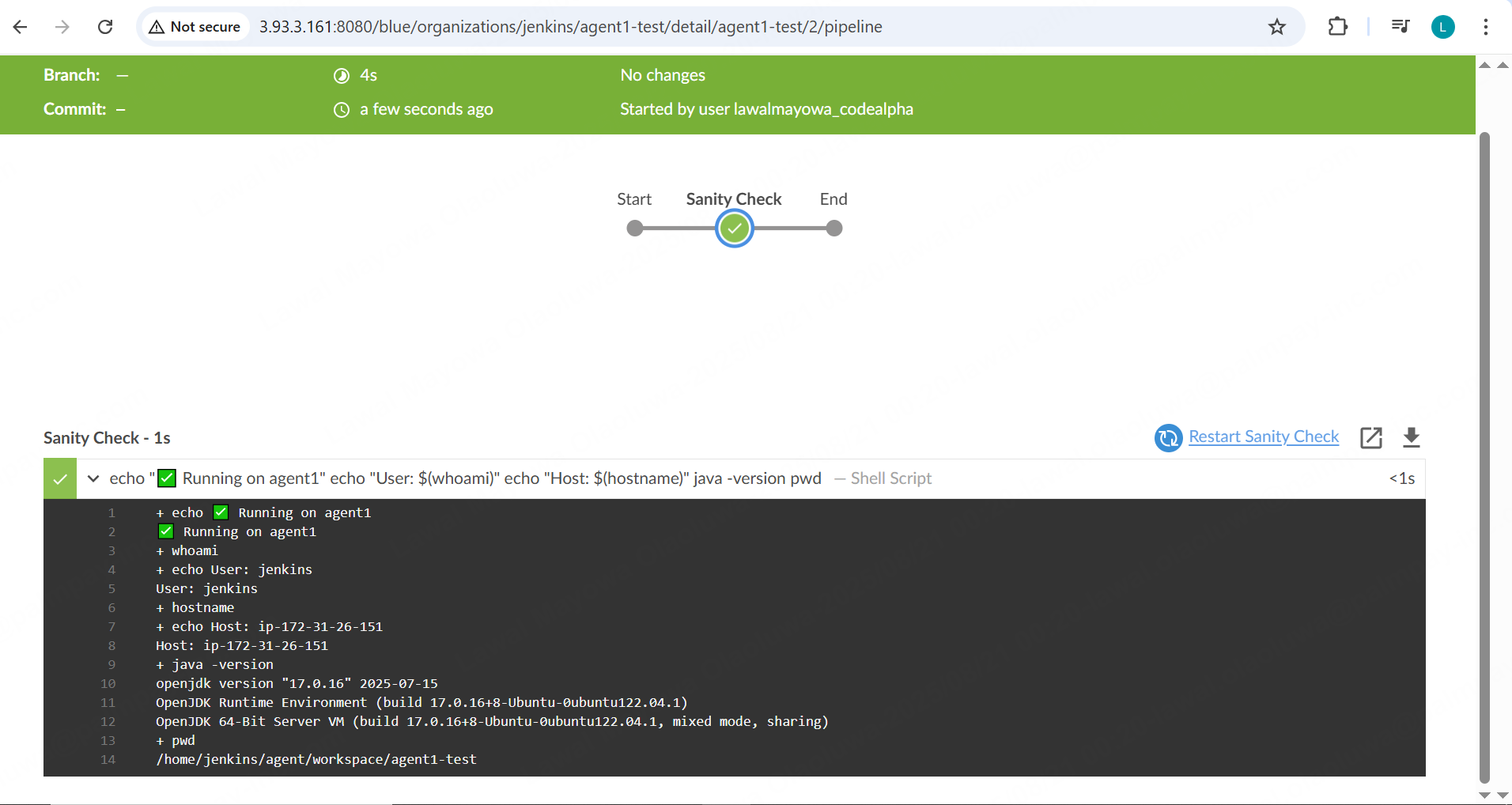
**Host Key Verification Strategy**: Manually trusted key verification once

**Availability:** Keep this agent online as much as possible

**Save and Connect**

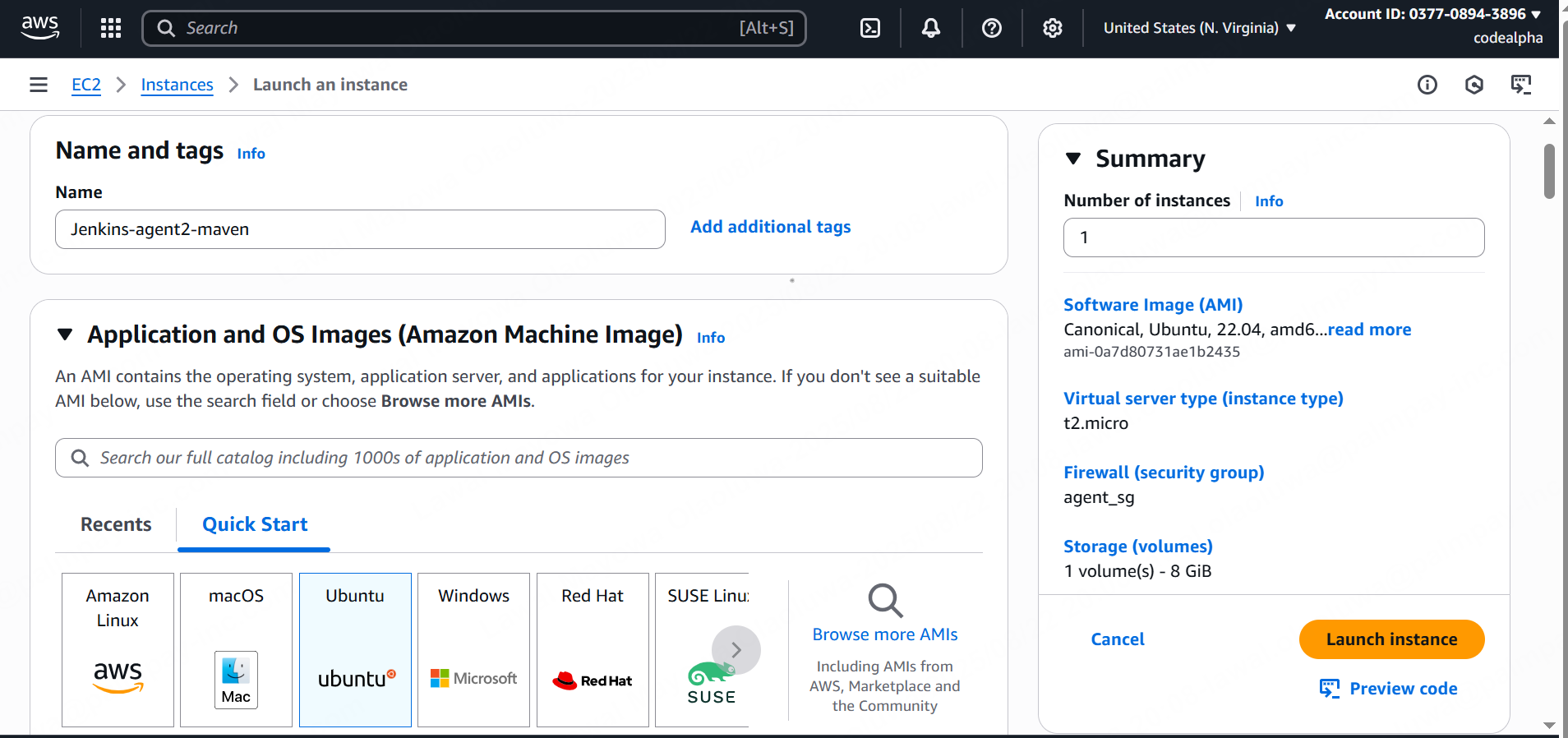


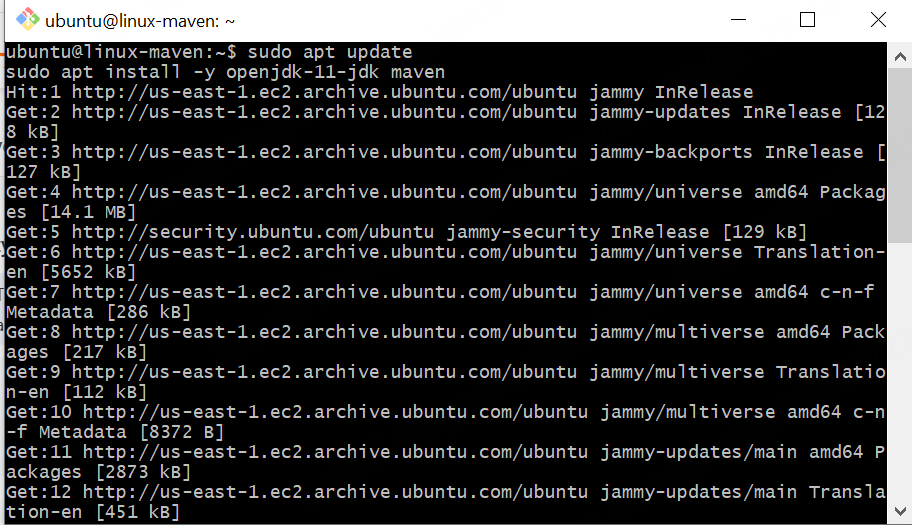
**Test successful**

****

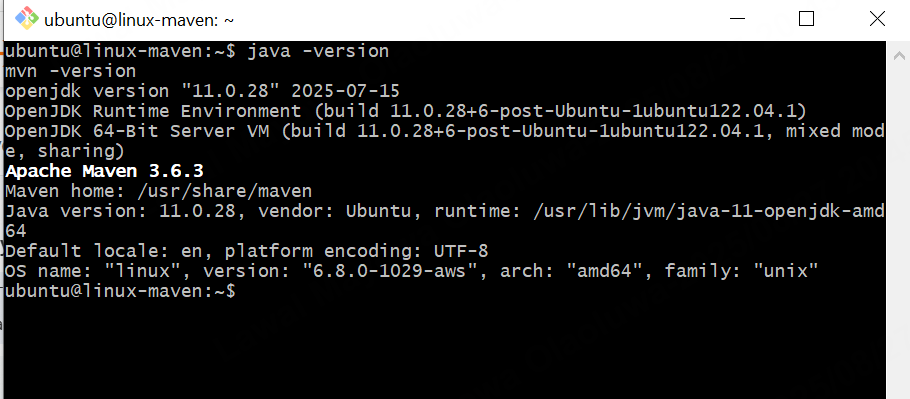
**Second agent**

linux-maven (Ubuntu 22.04)



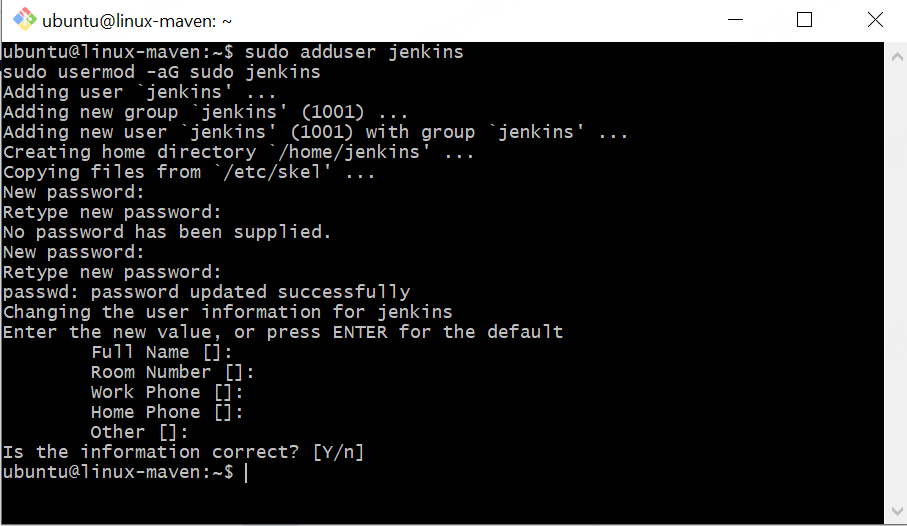
**Ssh to the linux-maven agent ( the second agent).  
  
**

**Update** and install **open jdk** and **Maven.**

****

**Check jdk and Maven Version.**

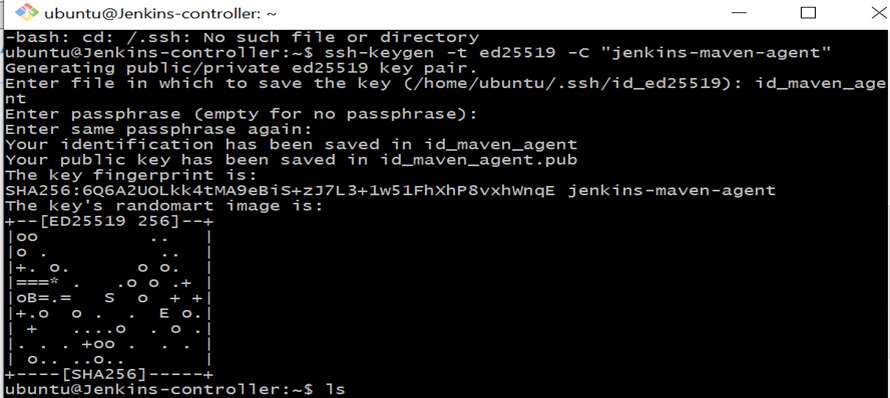
**Create Jenkins User on EC2**

****

**Password:** admin

**Set Up SSH from Jenkins Controller**

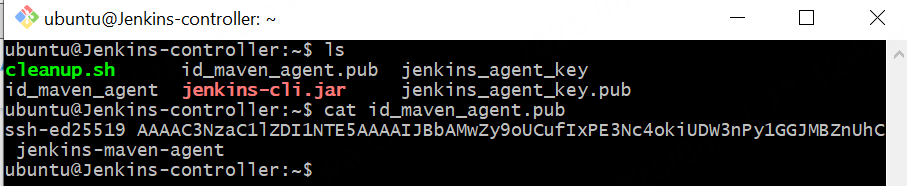
* **On your Jenkins Controller:**

****

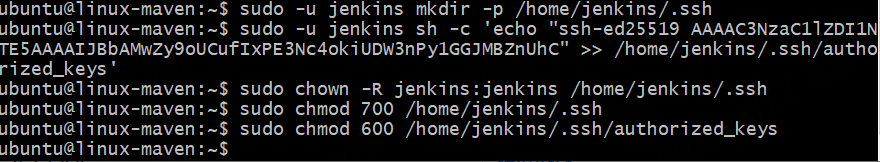
**Creating another key pair for ssh login for the second agent.**

* **Copy Public Key from Controller to Maven Agent EC2**

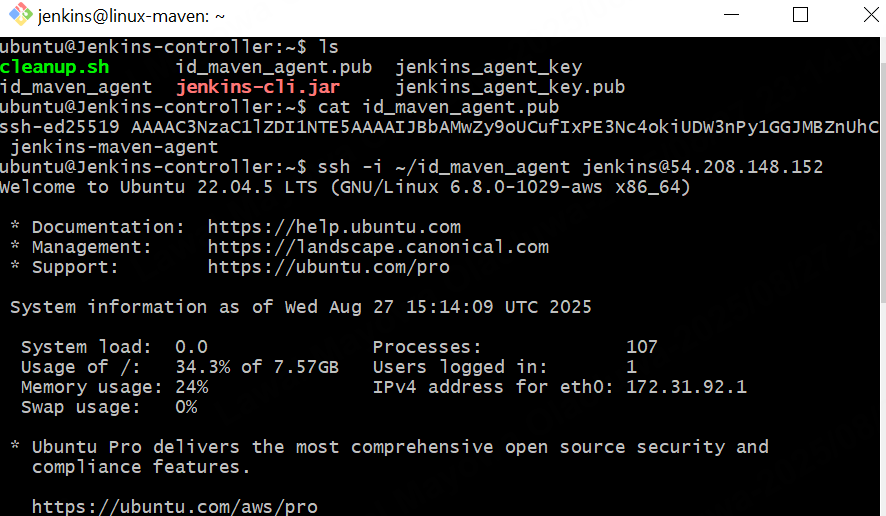
**Copy from the Jenkins controller**

****

**Paste the public key in agent.**

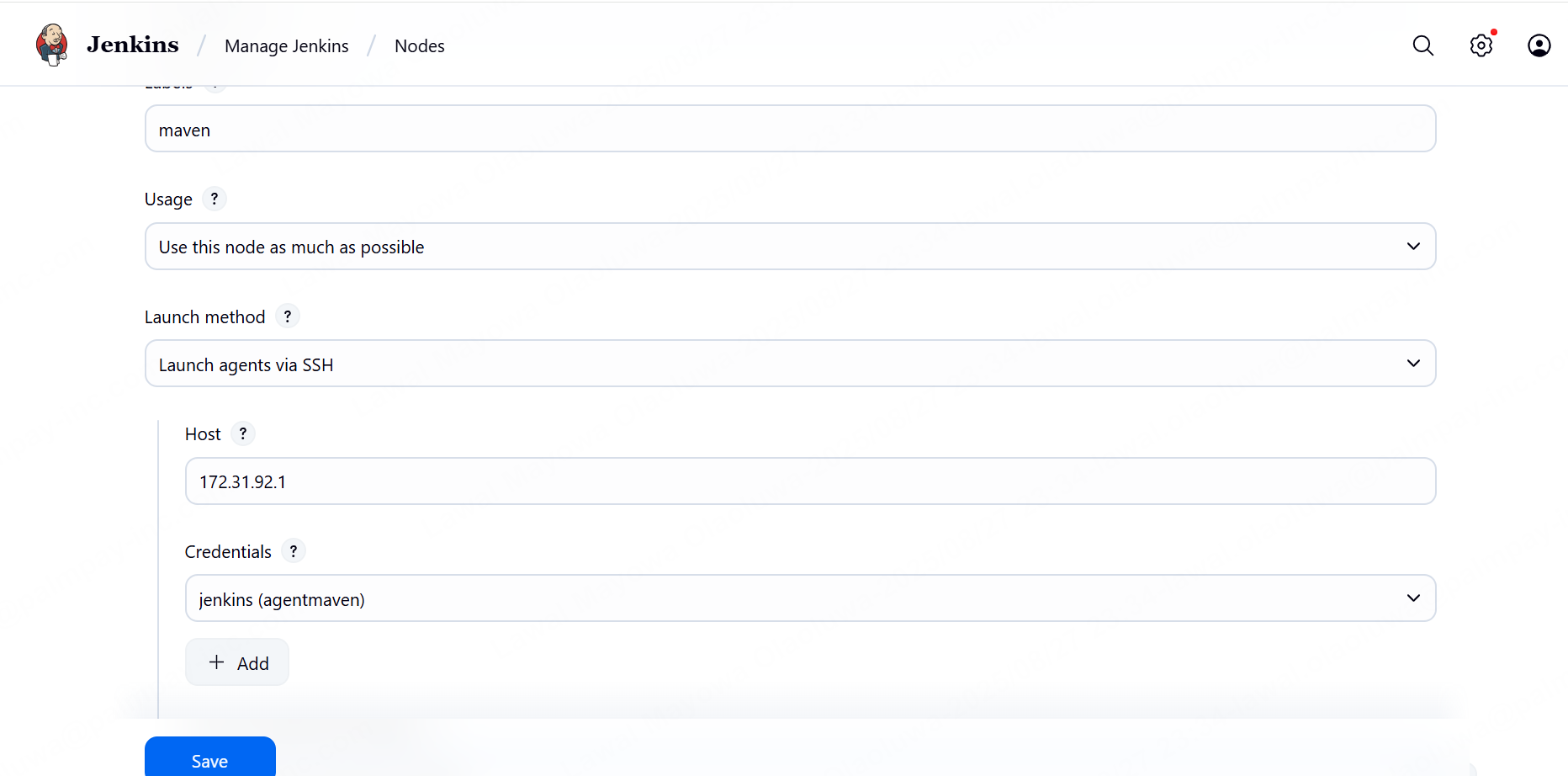
****

**Test remove login**

****

**Successful: Controller Jenkins can login maven Jenkins agent.**

* **Paste private key From Cli to Jenkins UI (see example 1)**

****

**Go to Manage Jenkins → Nodes → New Node.**

**Enter a name (e.g., maven-agent), select Permanent Agent, then click OK.**

**Configure the node:**

**Remote root directory: /home/jenkins**

**Labels:** maven (this helps when targeting the agent in a pipeline).

**Usage:** "Use this node as much as possible".

**Launch method:** Select Launch agents via SSH.

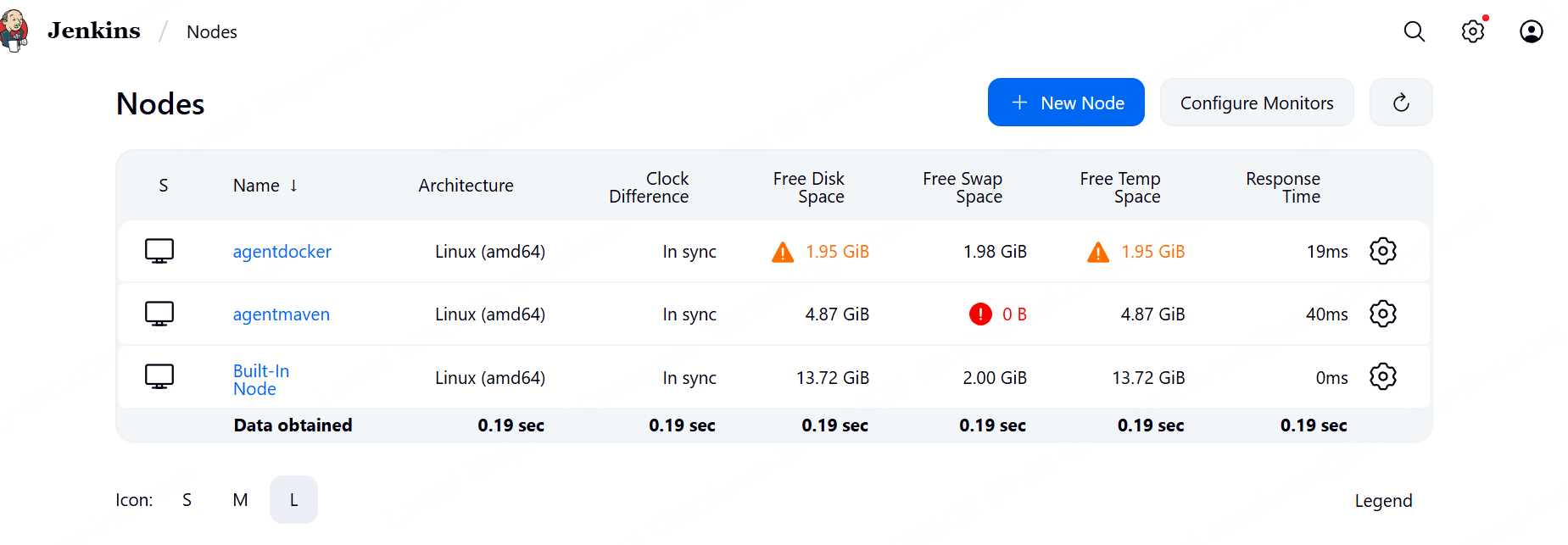
**Host:** Use the private IP of the Maven Agent instance.

**Credentials:**

**Add credentials** (kind = SSH Username with private key).

**Username =** jenkins (the one you set up on the agent).

**Private key =** paste the private key that matches the public key in the agent’s authorized\_keys.

**Host Key Verification Strategy →** set to “Non verifying”   
  
Node configure.

Test Node

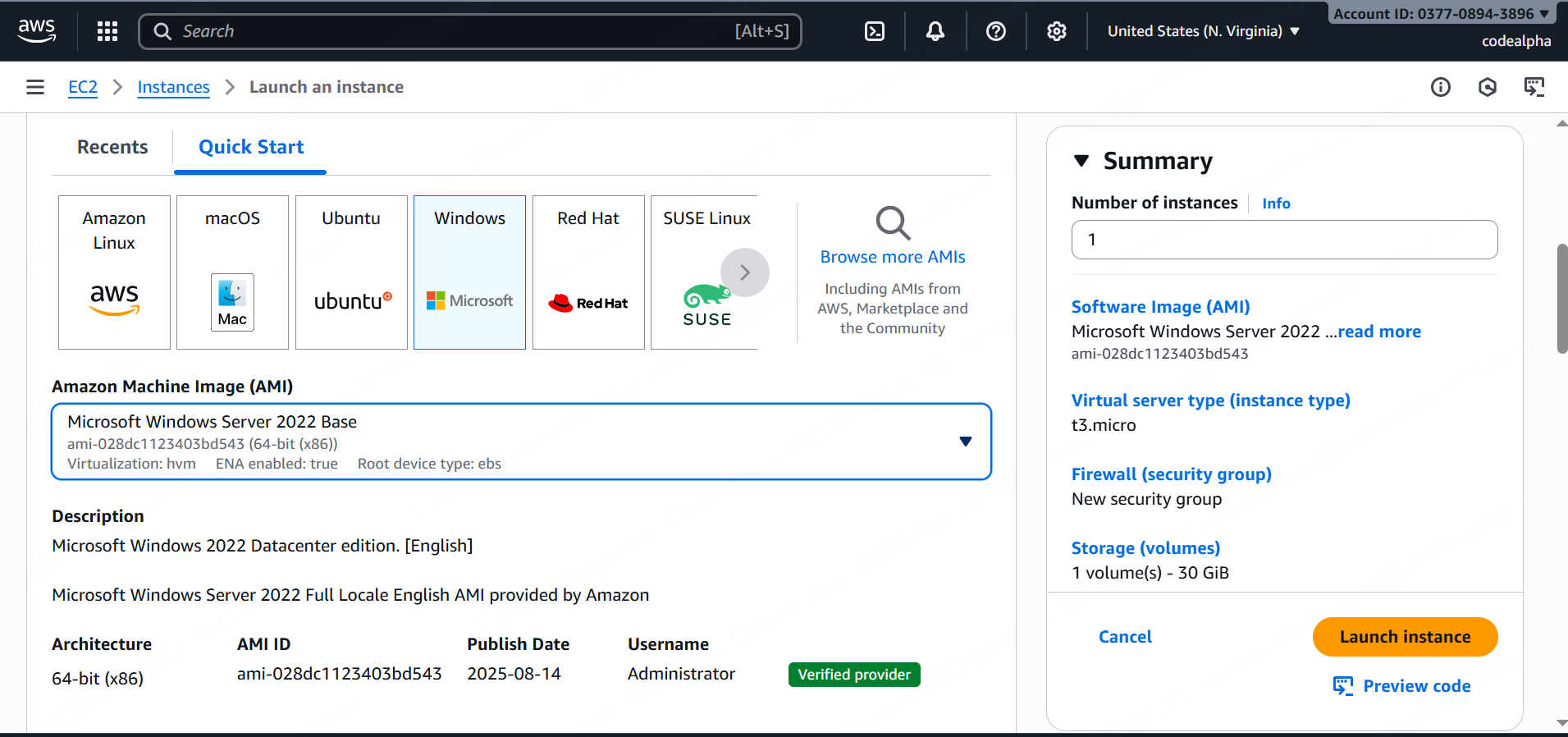


Test successful.

**Third Agent**

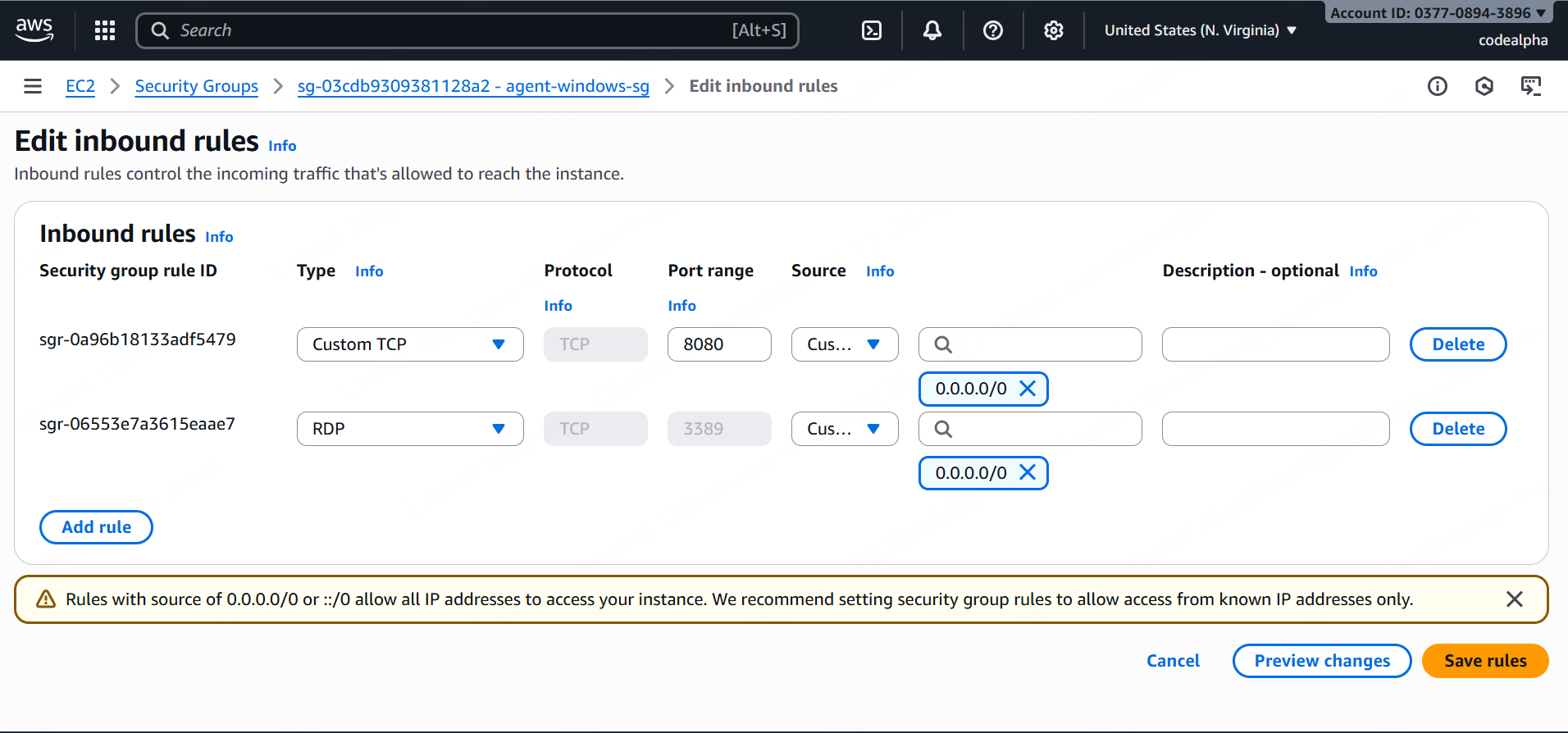
**agent-windows → for Windows builds & PowerShell/.NET jobs.**

* Launch an EC2 windows instance on Aws

****

* Allow RDP (TCP 3389) from your IP for remote login.

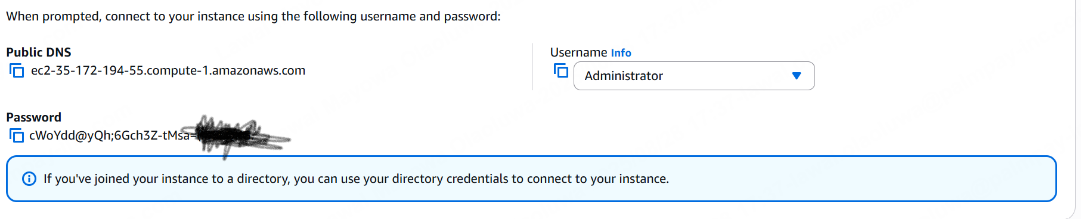
Allow outbound TCP to Jenkins controller IP on port 8080 (or HTTPS).



* Log in via RDP using the key pair.

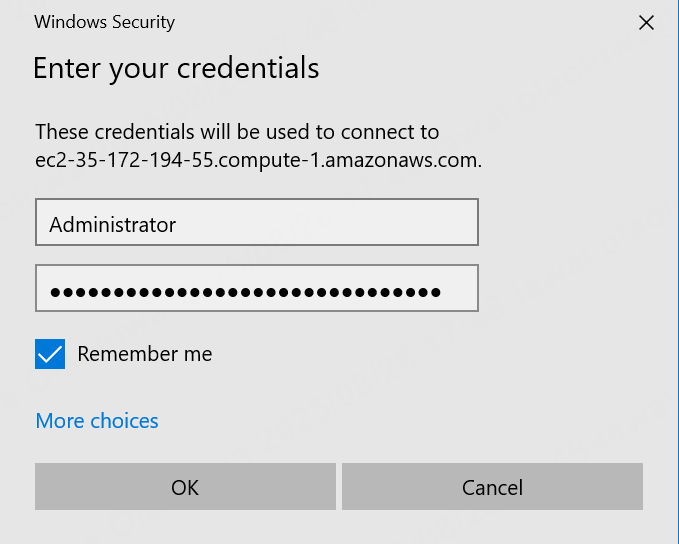
AWS Management Console → EC2 → Instances→ **Windows EC2 instance**→ Connect → RDP client→ get password→ Upload your **private key (.pem)** → Decrypt password

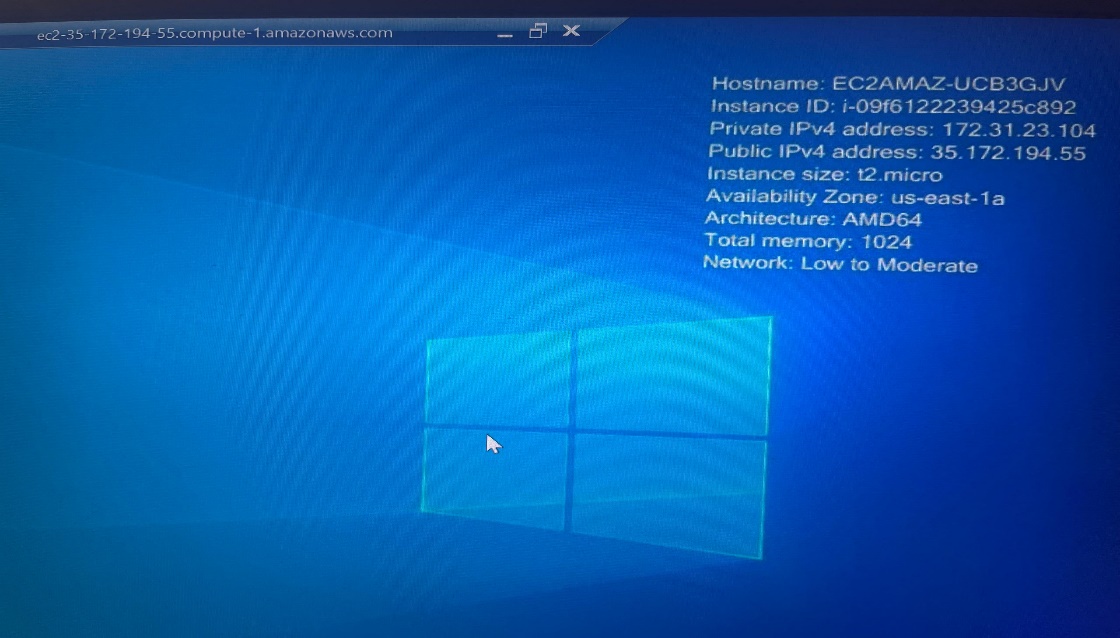
You’ll see the **Administrator password** (copy it securely)



* Open **Remote Desktop Connection (RDP)** on your local machine and paste the dns of the EC2 and the saved password







Connected successfully

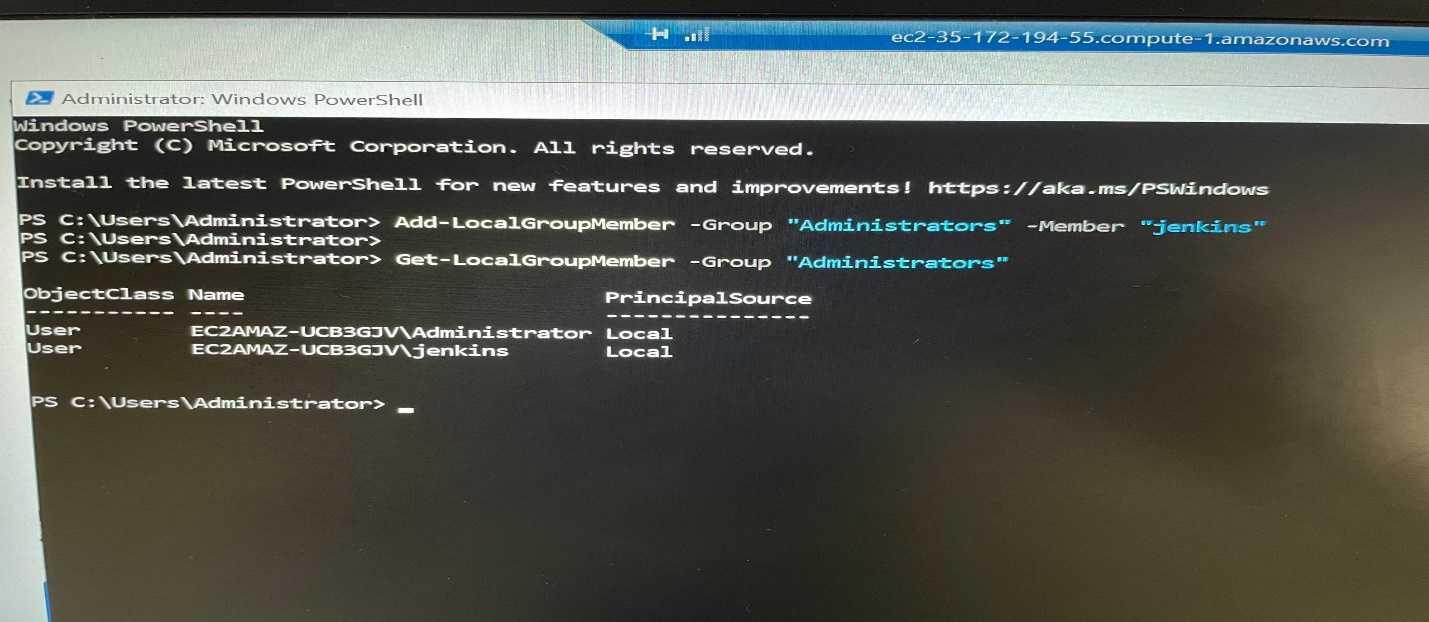
* Prepare Windows Agent for Jenkins

Computer Management → Local Users and Groups → Users → New User

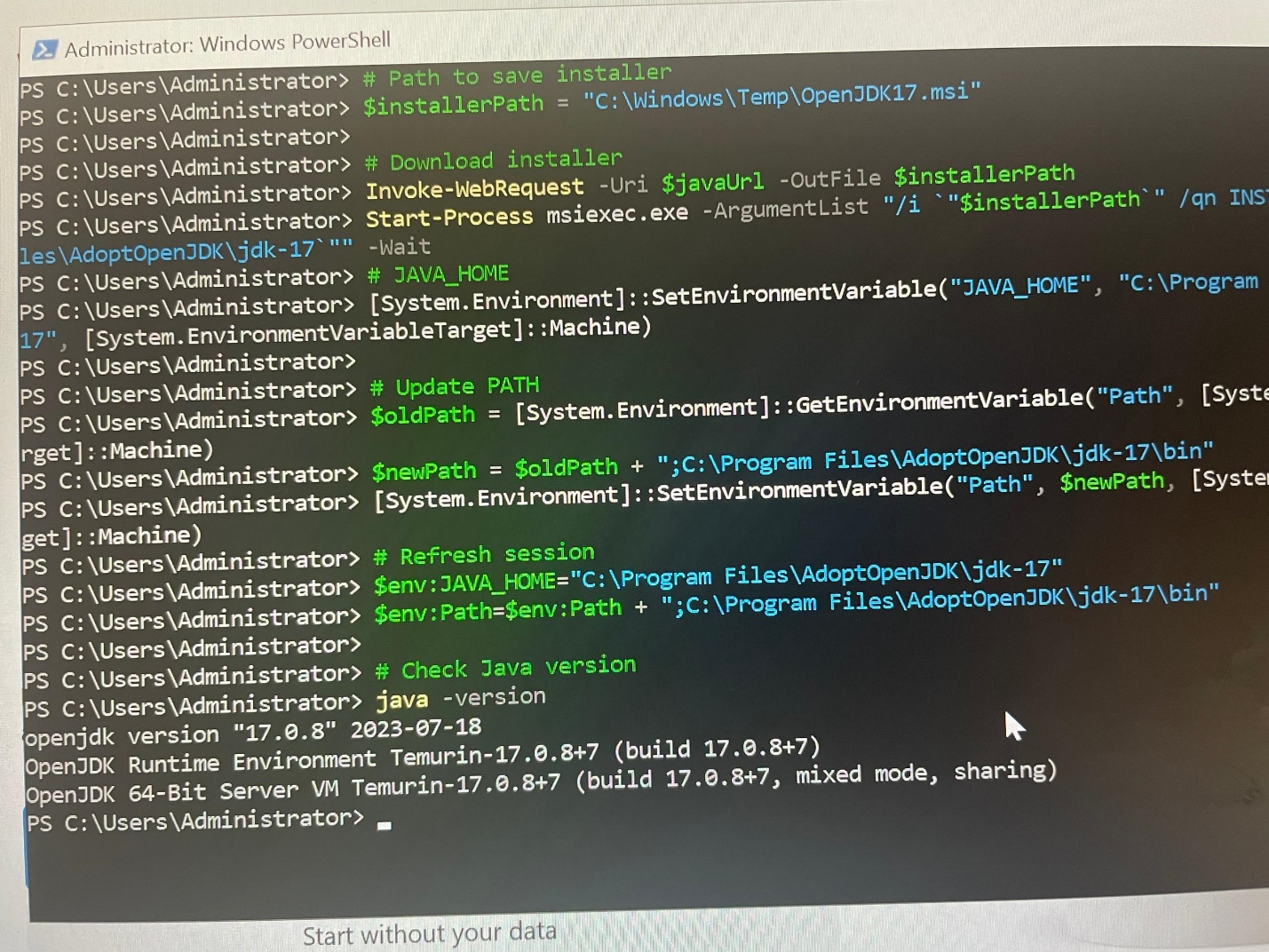
Username: Jenkins

Password: admin

Add the user to **Administrators group (using powershell)**



* **Create a folder for Jenkins agent files**:
* **Install Java 17**



* **Add Windows Node in Jenkins Controller**

**Node Name:** windows-agent

**Type:** Permanent Agent

**Remote Root Directory:** C:\jenkins

**Labels:** windows

**Usage:** Only build jobs with label expressions matching this node

**Launch Method:** Launch agent via Java Web Start (JNLP)

* Go to the Node Page in Jenkins  
    
  Navigate to **Manage Jenkins → Manage Nodes and Clouds → your Windows agent node→** Click on the **node name** to open its page

**Make it permanent**

Stop the current running agent

* **Download WinSW (Windows Service Wrapper)**

**Place it in your Jenkins work directory:**

**cd C:\jenkins**

**Invoke-WebRequest -Uri https://github.com/winsw/winsw/releases/latest/download/WinSW-x64.exe -OutFile C:\jenkins\jenkins-agent.exe**

* **Create a service config file  
  In the same folder (C:\jenkins), create jenkins-agent.xml with:**

**<service>**

**<id>jenkins-agent</id>**

**<name>Jenkins Agent</name>**

**<description>Jenkins Agent running as a Windows Service</description>**

**<executable>java</executable>**

**<arguments>-jar C:\jenkins\agent.jar -url http://3.91.80.33:8080/ -secret 8d1cc3b954acd87d8b7e73609859d5e76676f8251bc14c8d945c2a8c78446526 -name windows-agent -workDir C:\jenkins -webSocket</arguments>**

**<logpath>C:\jenkins\logs</logpath>**

**<log mode="roll"></log>**

**</service>**

* **Install the service**

**.\jenkins-agent.exe install**

* **Start the service**

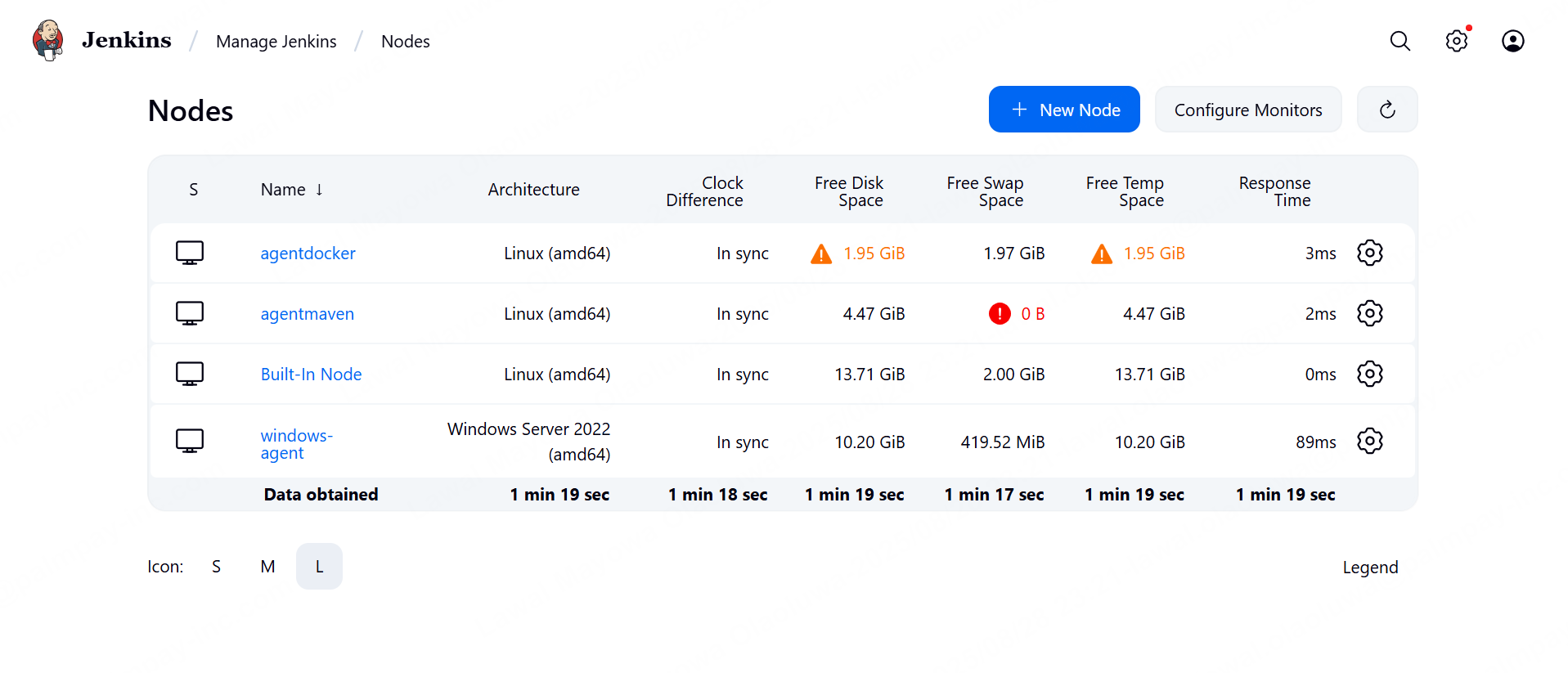
**.\jenkins-agent.exe start**

* **Verify**

**Get-Service jenkins-agent**

**Jenkins agent running successfully.**

**All Agents running successfully.**

****

**REMOTE JENKIN SCRIPT.**

* **agentmaven → builds Java with Maven**
* **windows-agent → runs Windows-specific tasks**
* **agentdocker → builds & deploys Docker images**

**Stage 1: Build with Maven**

stage('Build with Maven') {

agent { label 'agentmaven' } // Run only on Maven agent

steps {

git branch: 'main', url: 'https://github.com/spring-projects/spring-petclinic.git'

sh 'mvn clean package -DskipTests'

stash includes: 'target/\*.jar', name: 'app-jar'

}

}

**Stage 2: (Tests skipped)**// stage('Test with Maven') { ... }

**Stage 3: Run on Windows Agent**

stage('Run on Windows Agent') {

agent { label 'windows' } // Runs on Windows node

steps {

echo "Running a task on the Windows agent..."

bat 'echo Hello from Windows Agent > windows\_task.txt'

bat 'type windows\_task.txt'

}

post {

always {

archiveArtifacts artifacts: 'windows\_task.txt', fingerprint: true

echo "Windows agent task output archived."

}

}

}

**Stage 4: Docker Build & Push**

stage('Docker Build & Push') {

agent { label 'agentdocker' } // Runs on Docker-capable agent

steps {

unstash 'app-jar' // Get back the JAR built on Maven agent

// Create Dockerfile and copy app.jar

sh '''

cp target/\*.jar app.jar

echo "FROM openjdk:11-jre-slim" > Dockerfile

echo "COPY app.jar /app.jar" >> Dockerfile

echo 'ENTRYPOINT ["java","-jar","/app.jar"]' >> Dockerfile

'''

// Build & push to Docker Hub

withCredentials([usernamePassword(credentialsId: 'dockerhub-creds', usernameVariable: 'USER', passwordVariable: 'PASS')]) {

sh '''

docker build -t $USER/jenkinsjob:latest .

echo $PASS | docker login -u $USER --password-stdin

docker push $USER/jenkinsjob:latest

'''

}

}

}

**Stage 5: Deploy on Docker Agent**

stage('Deploy on Docker Agent') {

agent { label 'agentdocker' }

steps {

sh '''

docker stop jenkinsjob || true

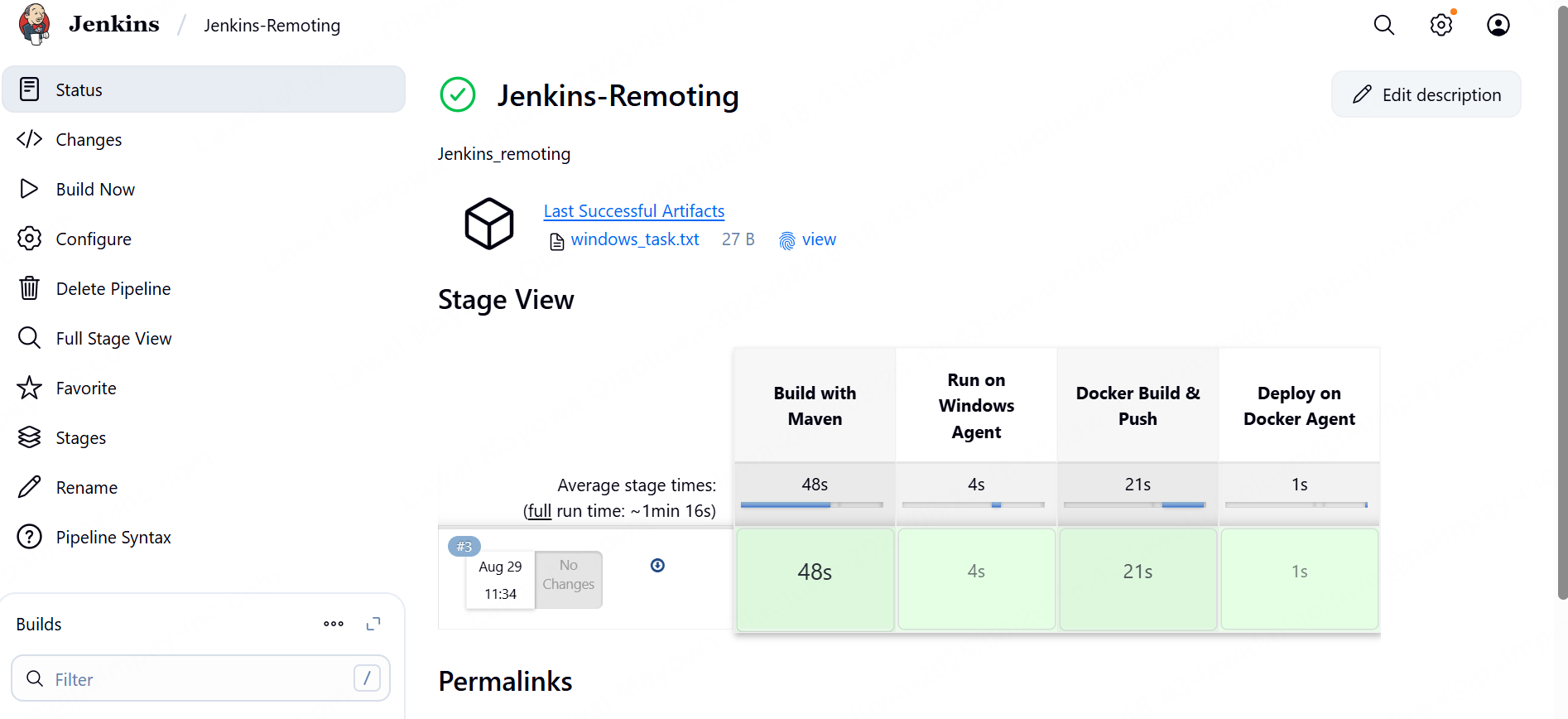
docker rm jenkinsjob || true

docker run -d --name jenkinsjob -p 8080:8080 pmayors/jenkinsjob:latest

'''

}

}

  
  
  
FULL JENKINS FILE.  
  
pipeline {

agent none

environment {

DOCKER\_IMAGE = "pmayors/jenkinsjob:latest"

}

stages {

stage('Build with Maven') {

agent { label 'agentmaven' }

steps {

// Pull Spring PetClinic project from GitHub (public repo)

git branch: 'main', url: 'https://github.com/spring-projects/spring-petclinic.git'

// Build the project (skip tests)

sh 'mvn clean package -DskipTests'

// Save jar artifact for later stages

stash includes: 'target/\*.jar', name: 'app-jar'

}

}

// Skipping Test stage as per your setup

// stage('Test with Maven') {

// agent { label 'agentmaven' }

// steps {

// sh 'mvn test'

// }

// post {

// always {

// junit 'target/surefire-reports/\*.xml'

// }

// }

// }

stage('Run on Windows Agent') {

agent { label 'windows' }

steps {

echo "Running a task on the Windows agent..."

bat 'echo Hello from Windows Agent > windows\_task.txt'

bat 'type windows\_task.txt'

}

post {

always {

// Archive the output from Windows agent for verification

archiveArtifacts artifacts: 'windows\_task.txt', fingerprint: true

echo "Windows agent task output archived."

}

}

}

stage('Docker Build & Push') {

agent { label 'agentdocker' }

steps {

// Retrieve the JAR file from Maven build

unstash 'app-jar'

// Create Dockerfile dynamically (Java 17 & correct jar name)

sh '''

cp target/spring-petclinic-\*.jar app.jar

echo "FROM openjdk:17-jdk-slim" > Dockerfile

echo "COPY app.jar /app.jar" >> Dockerfile

echo 'ENTRYPOINT ["java","-jar","/app.jar"]' >> Dockerfile

'''

// Build and push Docker image

withCredentials([usernamePassword(credentialsId: 'dockerhub-creds', usernameVariable: 'USER', passwordVariable: 'PASS')]) {

sh '''

docker build -t $DOCKER\_IMAGE .

echo $PASS | docker login -u $USER --password-stdin

docker push $DOCKER\_IMAGE

'''

}

}

}

stage('Deploy on Docker Agent') {

agent { label 'agentdocker' }

steps {

// Stop & remove old container if it exists, then run new one

sh '''

docker stop jenkinsjob || true

docker rm jenkinsjob || true

docker run -d --name jenkinsjob -p 8080:8080 $DOCKER\_IMAGE

echo "=== Running Containers ==="

docker ps

echo "=== Last 50 container logs ==="

docker logs --tail=50 jenkinsjob || true

'''

}

}

}

}

**📂 Execution Flow**

1. **Build on agentmaven**
   * Pulled Spring PetClinic from GitHub.
   * Compiled JAR using Maven (skipping tests).
   * Stashed artifact for later stages.
2. **Run task on Windows agent**
   * Executed a batch script (echo Hello from Windows Agent).
   * Archived output file.
3. **Docker build & push on agentdocker**
   * Unstashed JAR.
   * Created Dockerfile dynamically.
   * Built and pushed image to **Docker Hub** using Jenkins credentials.
4. **Deploy on agentdocker**
   * Stopped old container (if running).
   * Deployed new container from latest Docker image.
   * Published on port **8080**.

