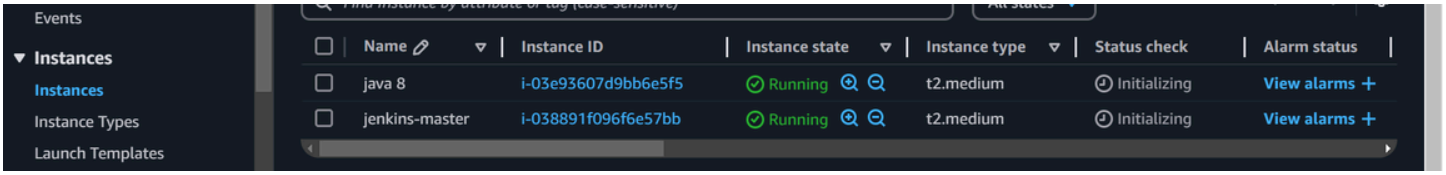
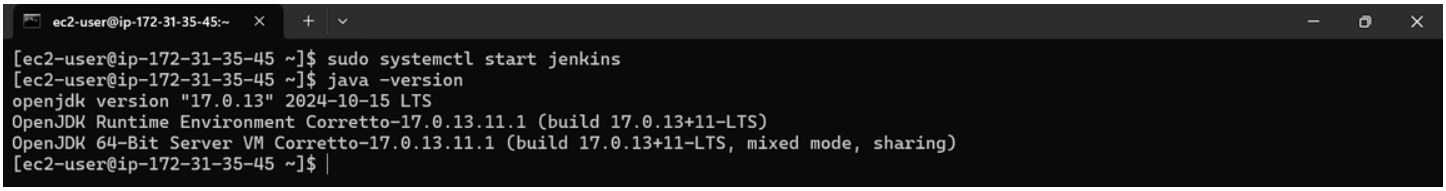
SETTING UP A JENKINS CI/CD PIPELINE FOR DOCKERIZED JAVA APPLICATIONS WITH MULTIPLE JAVA VERSIONS

1. Initial Setup:

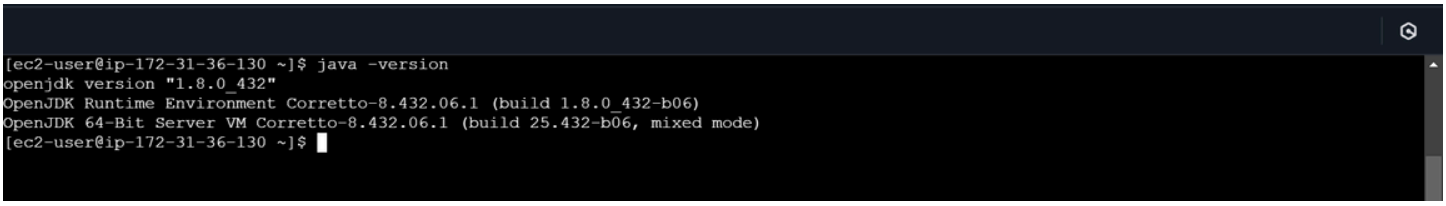
- Jenkins Master (jenkins-masters EC2): You installed Jenkins and Java 17.
- Java 8 Instance (java 8 EC2): You installed Java 1.8.0, Git, Maven, and Docker.

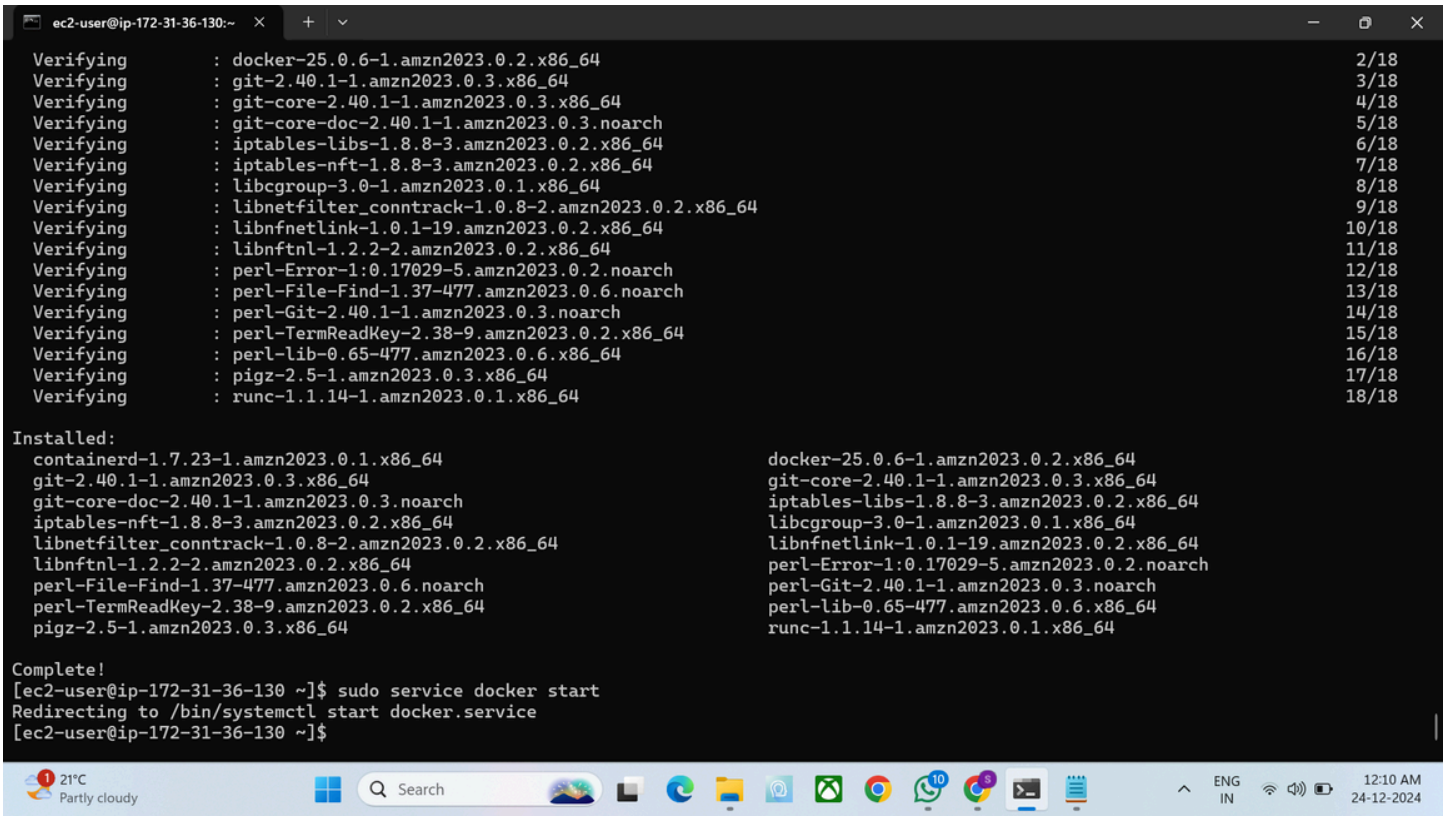


In Jenking master



In Java 8



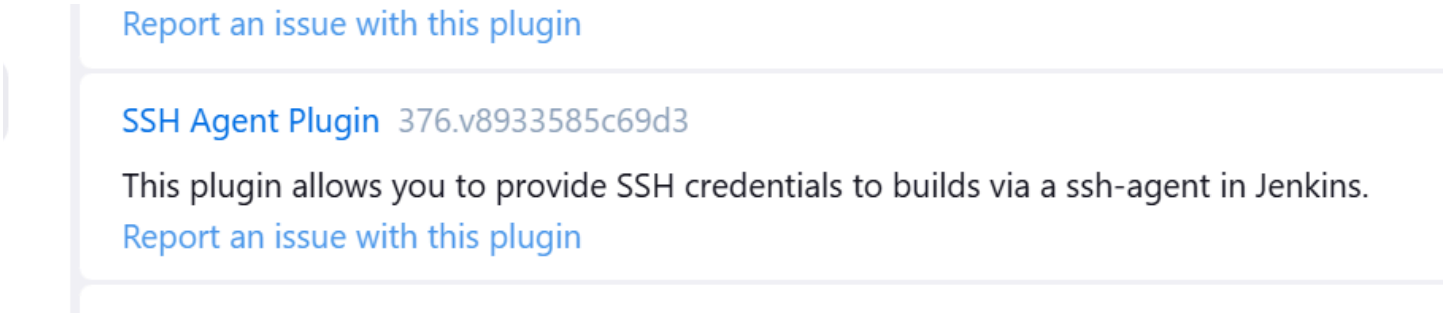


installed Java 1.8.0, Git, Maven, and Docker

---

2. Jenkins Setup:

- You connected to the Jenkins master and completed the Jenkins setup using the public IP.
- You installed the **SSH Agent Plugin** on Jenkins to manage SSH keys.



---

3. SSH Key Setup:

- You generated a key pair using **ssh-keygen** and added the public key to the authorized_keys on the Java 8 instance.
- You configured Jenkins credentials using the SSH private key to allow Jenkins to connect to the Java 8 instance.

```
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-36-130 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_rsa
Your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:5UnIJSHVW1CbAibsVONyKkp+8TdH5dMKPeUZhOt+Dlw ec2-user@ip-172-31-36-130.ap-south-1.compute.internal
The key's randomart image is:
+---[RSA 3072]----+
|      .ooOooo...  |
|       o* *. o+   |
|      o. = +o+.o  |
|       .+ +.*.+ o |
|  . o . S =.= E   |
| o . +    . o.=   |
|  o . . o ..+     |
|   .    . o  ...  |
|             o.   |
+----[SHA256]-----+
[ec2-user@ip-172-31-36-130 ~]$
```

21°C Partly cloudy | Search | ENG IN | 12:16 AM 24-12-2024

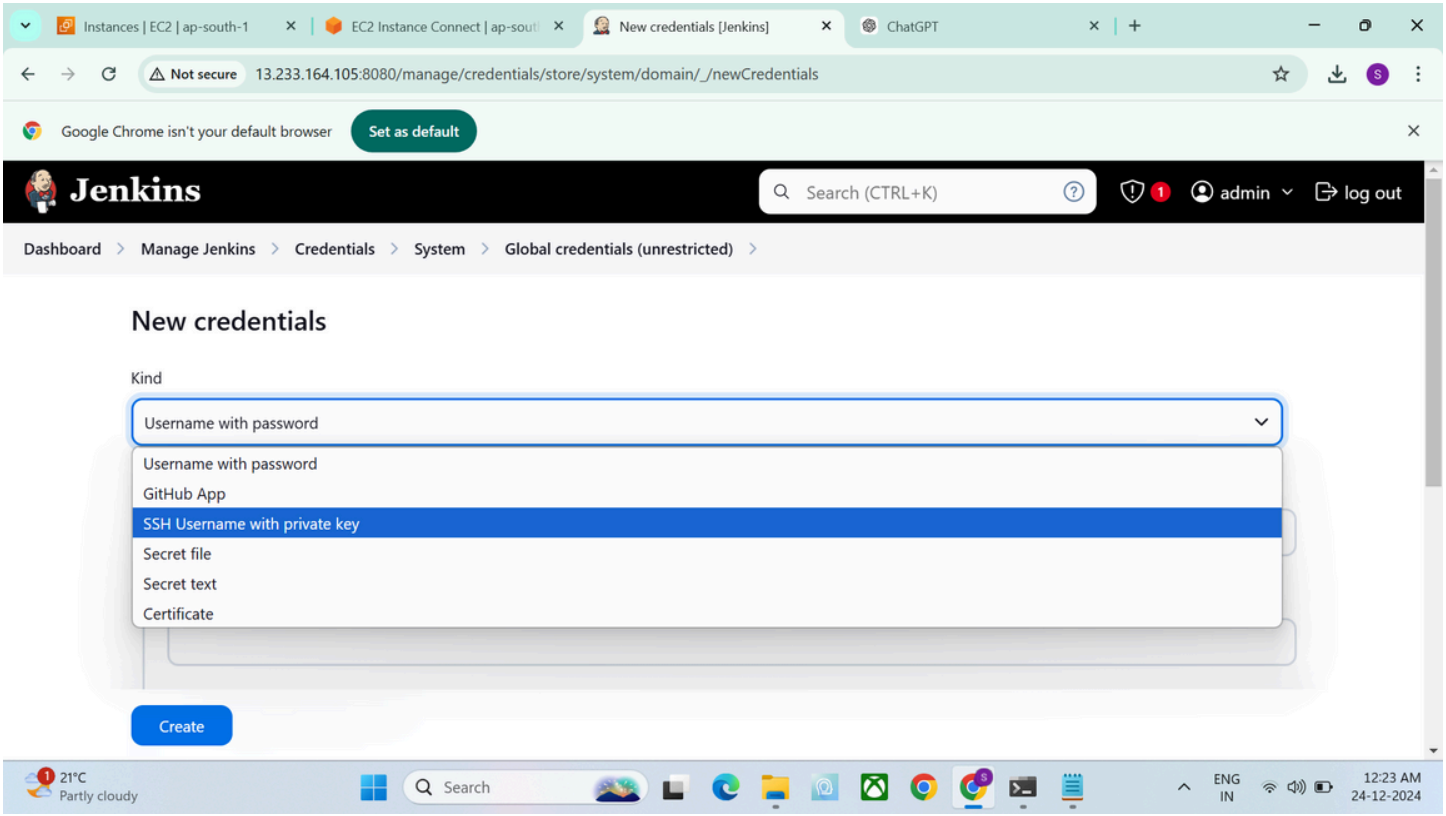
```
ec2-user@ip-172-31-36-130:~
[ec2-user@ip-172-31-36-130 ~]$ cd ~/.ssh
[ec2-user@ip-172-31-36-130 .ssh]$ ls
authorized_keys  id_rsa  id_rsa.pub
[ec2-user@ip-172-31-36-130 .ssh]$
```

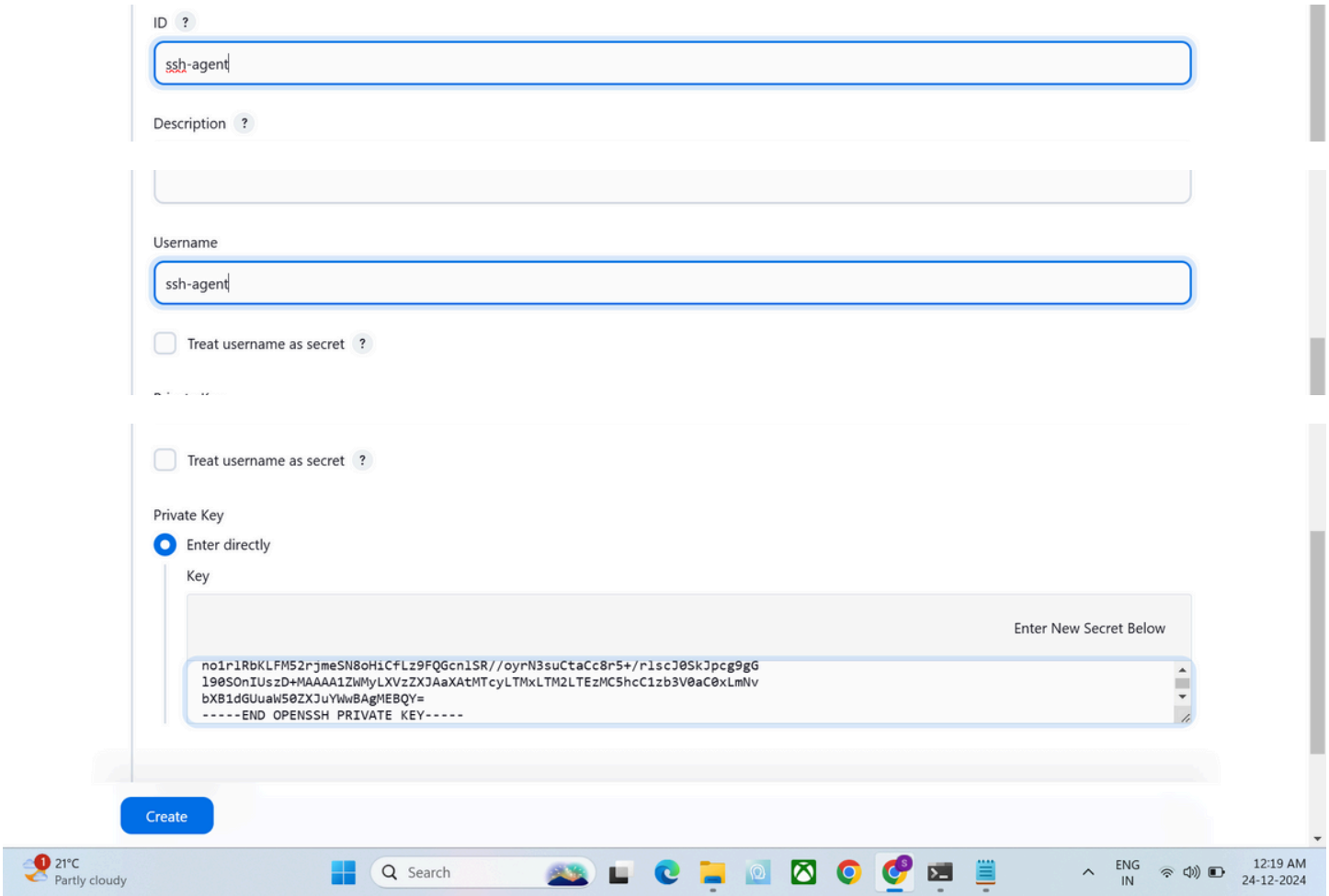
vi id_rsa

```
ec2-user@ip-172-31-36-130:~
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAgcn79kAaOw8+O8SfOJymCzIl4ZV4UznhYFVC16YN6qVSzZ6+0nN5
cE/J5+ehmJUlQnkmNvAx3/6vfx4TX/40tBrbFPwOlmXswmaHdsdD8Q2E2kc90e3ne1aBtp
vRw2CiTXLti8MoerzAV2pPWBTZIhD/zZ1tXfB88NXf+hwM8a740haSTVqJc7E0eDLwcOP5
UWrdachdb9yAuKJEQG0QgbSGBadgggsRzU+0SKh/YoHqKbXbb+Z31gMUYfim1JLm7ghfrG
K1nqsSgV5j1DGZ+GvciZLuRnCNjuAXiwY8VRBL+IHZJ4wehp2yPolV5a1TBKYW/lnMphI8
+Y7COE81XE2XNHifFEPA7BNVgF8mC1L86oVABQG3tanaT8HjRtvCP7AmY5TnvV+6bPJa6W
iB2OfepP2/aVI6TuAG6/BSrVpD7YR/hpfmP/+OZ6SedDU2Am6g19nekAj3W6NycSeBC7Uh
naexEwHJkkFunbnxe30Y3/BxpuY23mE5Mf5h6sOJAAAFsJFF/MiRRfzIAAAAB3NzaC1yc2
EAAAGBAIHJ+/ZAGjsPPjvEnzicpgsyJeGVeFM54WBVQtemDeqlUs2evtJzeXBPyefnoZiV
JUJ5JjbwMd/+r38eE1/+NLQa2xT8DpZl7MJmh3bHQ/ENhNpHPdHt53tWgbab0cNgok1y7Y
vDKHq8wFdqT1gU2SIQ/82dbV3wfPDV3/ocDPGu+NIWkk1aiXOxNHgy8HDj+VFq3WnIXW/c
gLiiREBtEIG0hgWnYILEc1PtEiof2KB6im122/md9YDFGH4ptSS5u4IX6xitZ6rEoFeY9
Qxmfhr3ImS7kZwjY7gF4sGPFUQS/iB2SeMHoadsj6JVeWtUwSmFv5ZzKYSPPmOwjhPNVxN
lzR4nxRDwOwTVYBfJgtS/OqFQAUBt7Wp2k/B40bbwj+wJmOU571fumzyWulogdjn3qadv2
lSOk7gBuvwUq1aQ+2Ef4aX5j//jmeknnQ1NgJuoNfZ3pAI91ujcnEngQu1IZ2nsRMByZJB
bp258Xt9GN/wcabmNt5hOTH+YerDiQAAAAMBAAEAAAGAIzNTZU2sb5EqSU6B+not5IJ02L
JPwI7Fq/lj2BX+10GAbCkeSvirmoThV0awwq/Mu7Mc5LCPbldmZJ0VFydcRu6ztYTJNgwt
wunI2rOfnyyR+BtzTlDuQ7rKEms/YHSAgG3ntEy5WVSA/BRmsZZx73XZ0vvb8irrz+vJw8
1Ay3mvnVcm1dj0x45rkyAJqyyBbY4EwBBqGxEcTjcgOLL705nQ8Y6cM5jp+puf40bjyaU/
i5F5kudYYBBUwl+Ma8HtgghRMqkmQZqK97p17rNUHeQAh/Rf2OV4WFKmH9OahB/JHXRznu
TbGt6fv8tegBtKkQ77mPip5STbanhU5MZDC1KIK6lEihHIgA0XweVIK+RiFbtP9tUXnQNb
1f76k7YBHv+yYII83/Fko+suIfY9UnmFmjvzWiH0zH/G+6xqaNZxy+/RpNRlisPtEtGWJi
O8WjY926HBxueePDYqOIE10m3mh0dPk3dcesZlEy5t5y6NQWd5rj6X8vkT/r2ITOw7AAAA
wBo/y2FatlPvRad69FjSmrF6/d+b0uENM9probOat0y8QuL/qhAOfdejjf/wLst4XHyccv
C0/e99p2oCUhwhYpI9TUj6FMv7i6F2RZlWJnmy/vCkTDUrnhout92LLL0JtTDHYtmPcm7K
t1fd5FXToSfymXKu4uzkOvdYgM860AKm6QSYOlUPtwuonirRM/dd4KsC1I07rinbJEOVOM
v5OUa3lXL5n0TyvDBBDZ+Cdc2z5cSYdHsqVU0xSXe5FIs2ogAAAMEAtZaX3ESjR3JVQoEY
1DppjmdT5ZFDDxT8symCc3+0VzRh6IMzz5NYn6XOMG8XfJQZqTw/b7Nm/lolxRHQ6vgi7p
S6hOnMZSfAm4dLwyuv8laHZWbHLT73kioKwsJW1g2FkpVBIVzXX957f3YgOj1poqP+IM/D
0l5BOLHJ+qHBCNBr90+NUh9FmlfwhsBmCy72Y0m5YJFBc7U6J274UHfhtHXyzaKfGJTjfY
v//p2LP/f9sWy64tvHYKYbNI4pXqKjAAAAwQC2+WvhkYgfOh0ZOV/AdAxh+rR/UwZqys0h
D+TeWdPgPipwWs6wTbhNiJXdPzuWhTLBcO7eW5PEnUlUKcMTRDFV4nbXgSqXzR3r80lGy9
VPoQndnhzdiQ+9mdL5NxsU8b3yemMOQoAVhDuwlhL1rao67xDeYyvjRKwMJ7/6ilw/mXui
no1rlRbKLFM52rjmeSN8oHiCfLz9FQGcnlSR//oyrN3suCtaCc8r5+/rlscJ0SkJpcg9gG
l90SOnIUszD+MAAAA1ZWMyLXVzZXJAaXAtMTcyLTMxLTM2LTEzMC5hcC1zb3V0aC0xLmNv
bXB1dGUuaW50ZXJuYWwBAgMEBQY=
-----END OPENSSH PRIVATE KEY-----
~
~
~
~
                                                                1,1           All
```

21°C Partly cloudy | Search | ENG IN | 12:18 AM 24-12-2024

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

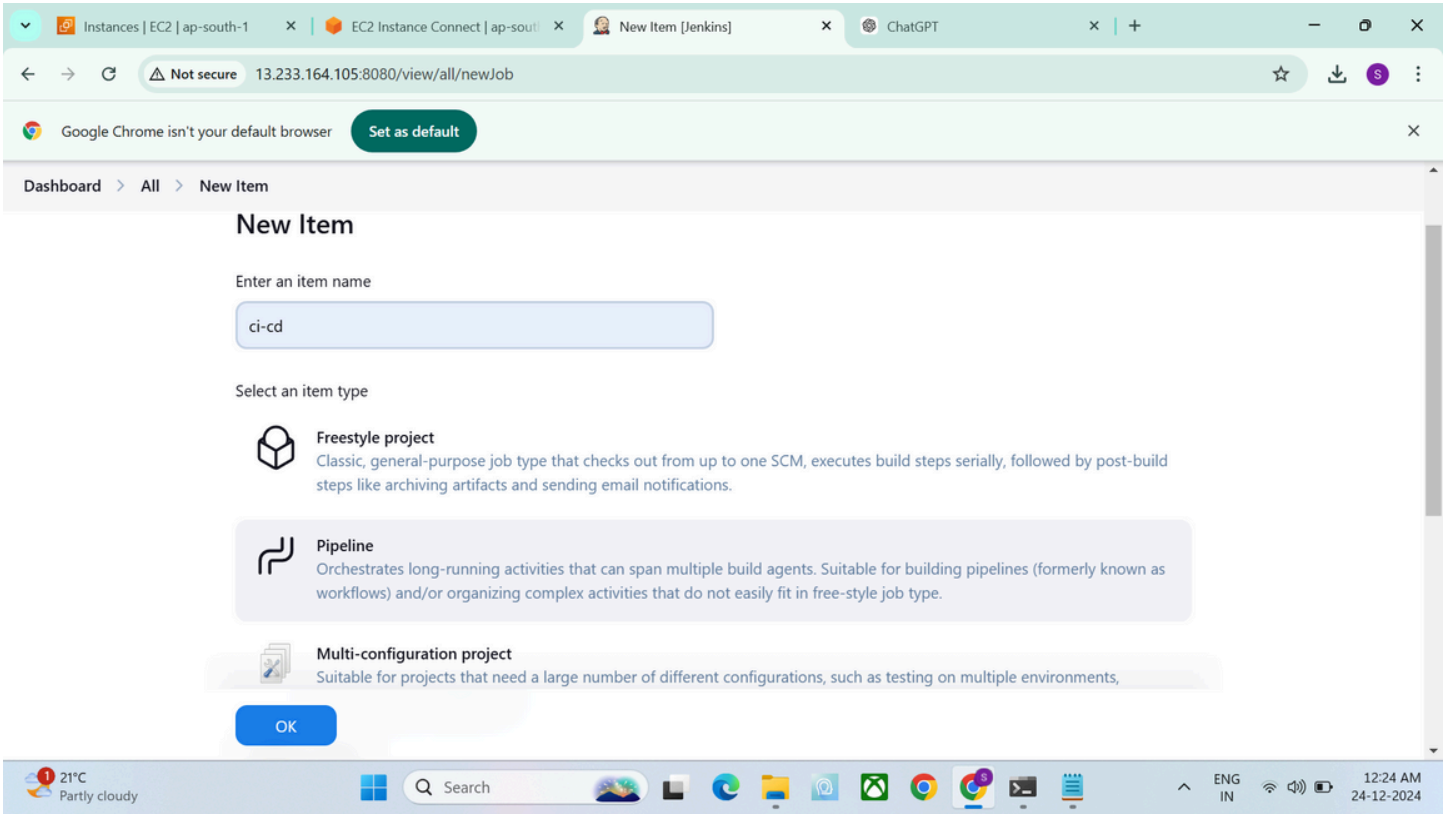


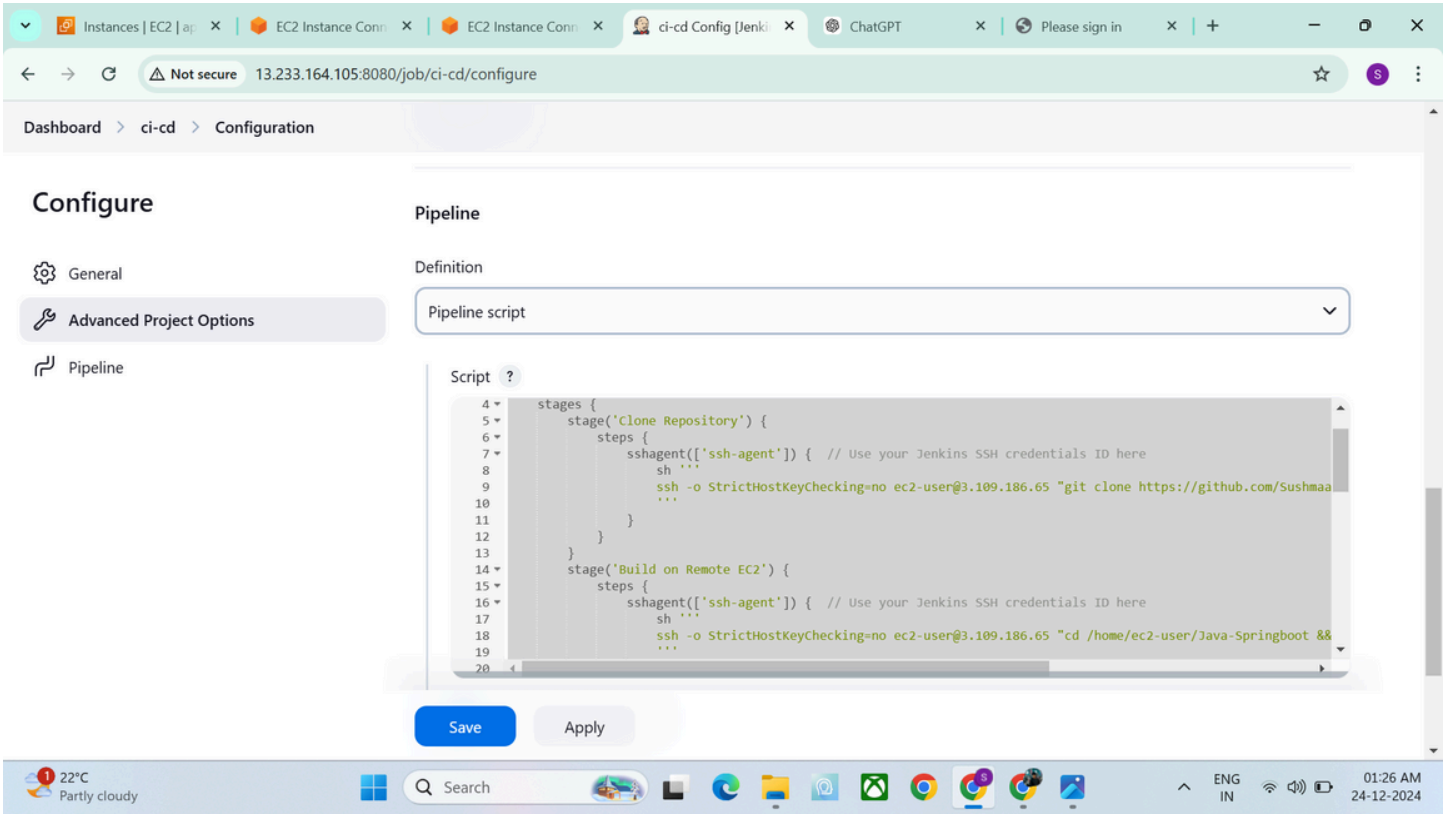


4. Pipeline Configuration:

You created a Pipeline project in Jenkins, and your pipeline consists of four stages:

- Clone Repository: Clones the Git repository from GitHub to the Java 8 instance.
- Build on Remote EC2: Runs a Maven build on the Java 8 instance to compile the project.
- Build Docker Image: Builds a Docker image on the Java 8 instance using the Dockerfile in the repository.
- Build Docker Image and Run Container on Remote EC2: Builds the Docker image again (if needed), removes any previous containers, and runs a new container from the built image.



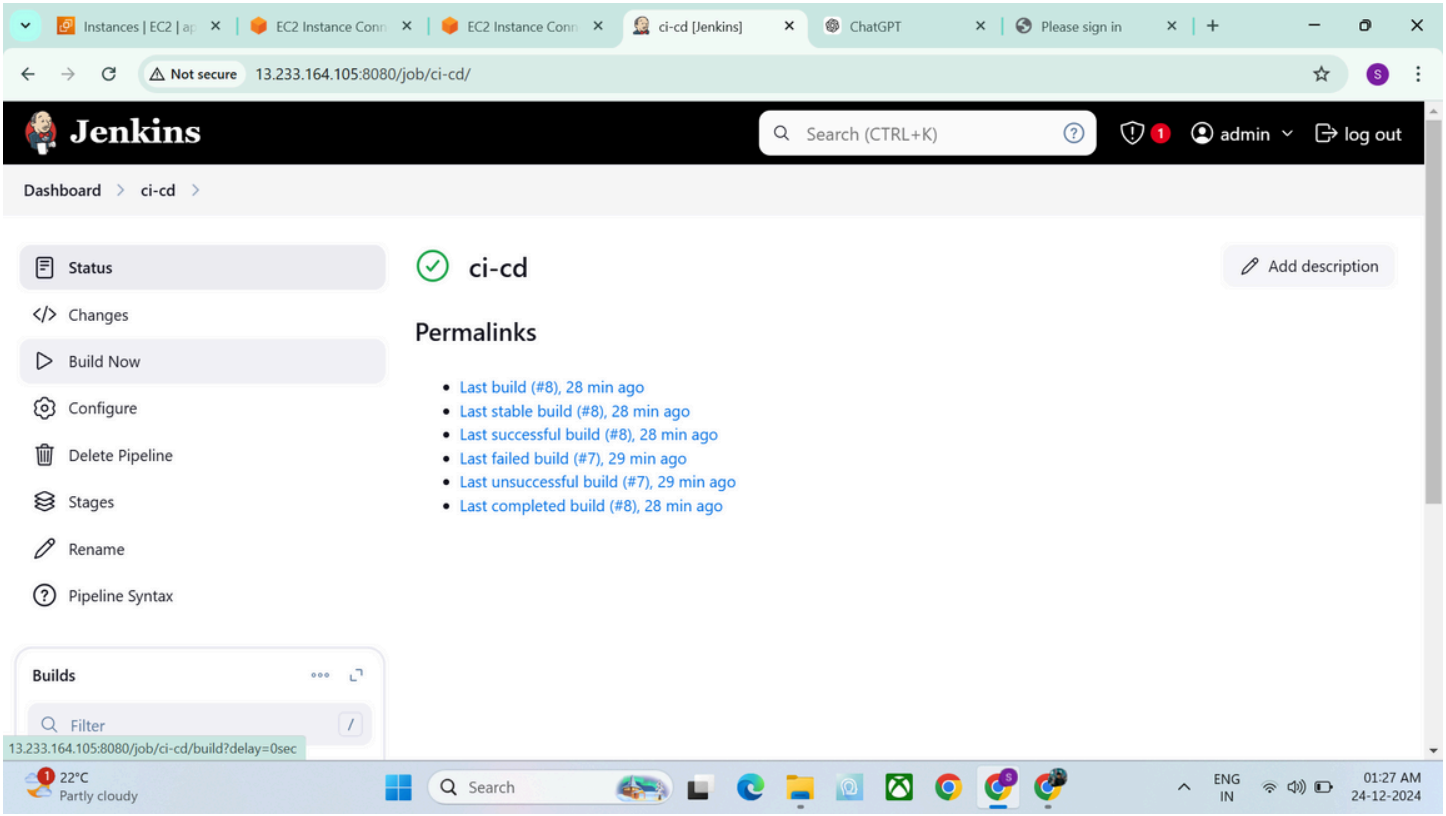


5. Pipeline Script:

The pipeline script uses the SSH Agent to connect to the Java 8 instance and execute the necessary commands remotely.

```
pipeline {

  agent any



  stages {

    stage('Clone Repository') {

      steps {

        sshagent(['ssh-agent']) {

          sh '''

          ssh -o StrictHostKeyChecking=no ec2-user@3.109.186.65 "git clone https://github.com/Sushmaa123/Java-Springboot.git /home/ec2-user/Java-Springboot"

          '''

        }

      }

    }

    stage('Build on Remote EC2') {

      steps {

        sshagent(['ssh-agent']) {

          sh '''

          ssh -o StrictHostKeyChecking=no ec2-user@3.109.186.65 "cd /home/ec2-user/Java-Springboot && mvn clean install"

          '''

        }

      }

    }

    stage('Build Docker Image') {
```

```
        steps {
          sshagent(['ssh-agent']) {
            sh '''
            ssh -o StrictHostKeyChecking=no ec2-user@3.109.186.65 "cd /home/ec2-user/Java-Springboot && sudo docker build -t java-springboot-image ."
            '''
          }
        }
      }
      stage('Build Docker Image and Run Container on Remote EC2') {
        steps {
          sshagent(['ssh-agent']) {
            sh '''
            ssh -o StrictHostKeyChecking=no ec2-user@3.109.186.65 "
            cd /home/ec2-user/Java-Springboot
            sudo docker build -t java-springboot-image .
            sudo docker stop java-springboot-container || true
            sudo docker rm java-springboot-container || true
            sudo docker run -d --name java-springboot-container -p 8081:8080 java-springboot-image
            "
            '''
          }
        }
      }
    }
}
```



6. Result:

After running the pipeline, Jenkins will create a Docker container on the Java 8 instance, and you should be able to access the container through port 8081 on the instance.