Jenkins 1

Users and Roles and in Jenkins

Master-Slave architecture

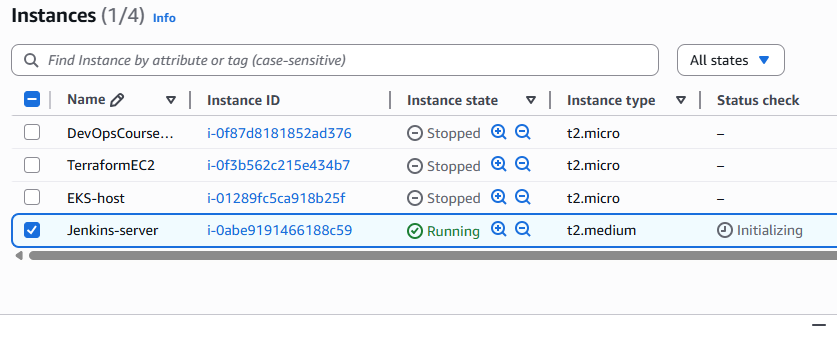
Pipeline

User and Roles in Jenkins

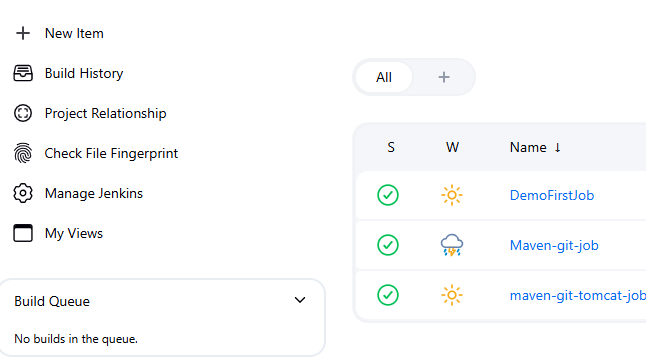
There are multiple people (Dev team, Testing team, DevOps team, ) working on the same project. Do we want to give access to everyone working on the project the entire Jenkins dashboard? The access level will be different for different teams in Jenkins. that’s where we need User management in Jenkins. Generally in every project, multiple teams (Dev, DevOps, QA) could be there and for every team Jenkins access would be provided. Every team members will have their own user account to login into Jenkins and not every team member will have complete access to Jenkins. Certain teams such as Dev, QA team will have limited access to Jenkins because they are responsible just to run the job not to create, edit, or delete the jobs whereas Operations team will have more access to Jenkins as they are responsible to create, edit or delete jobs in Jenkins.

Creating Users and managing their permissions:

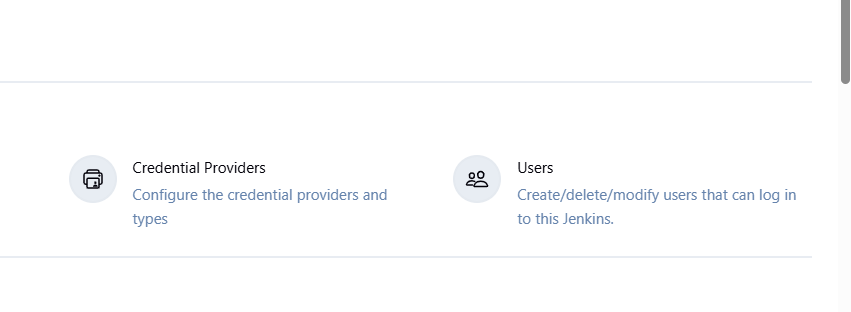
Start Jenkins-Server



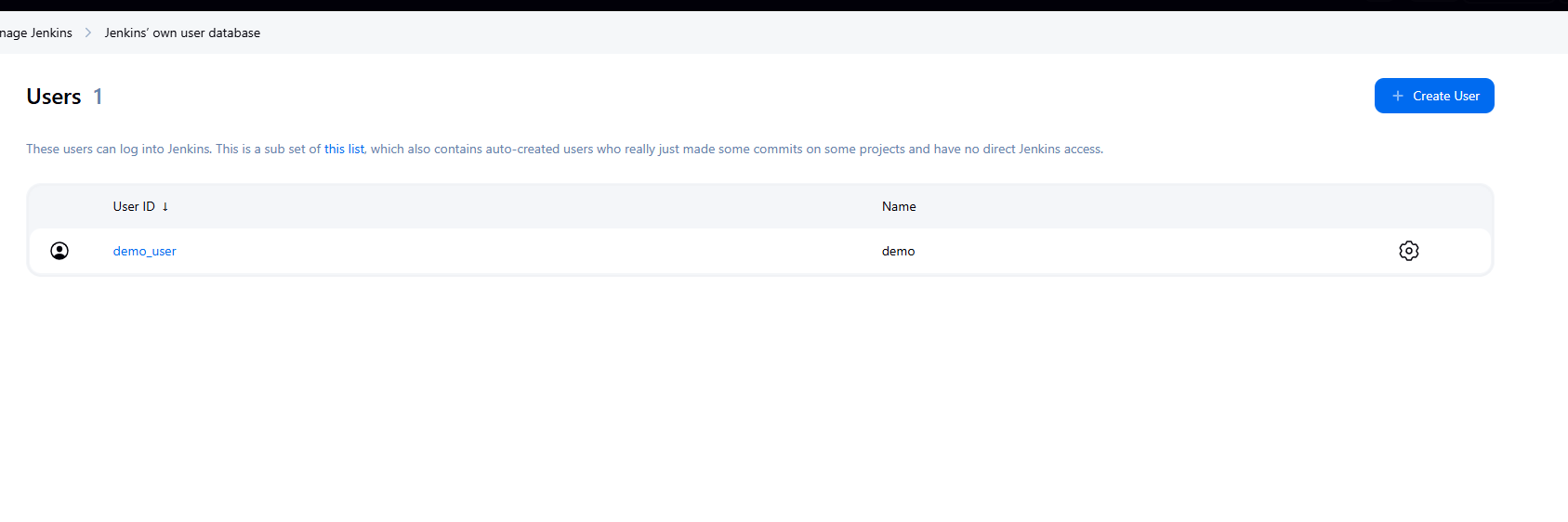
<http://16.52.83.212:8080/login?from=%2F>



Manage Jenkins

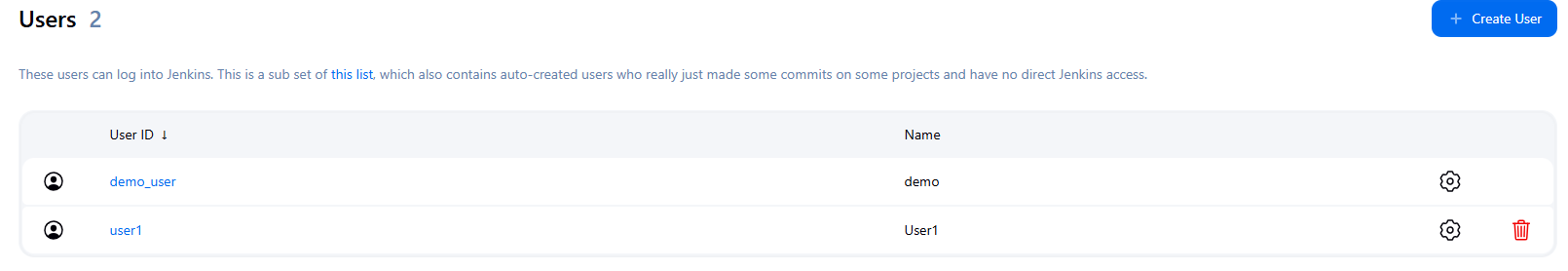


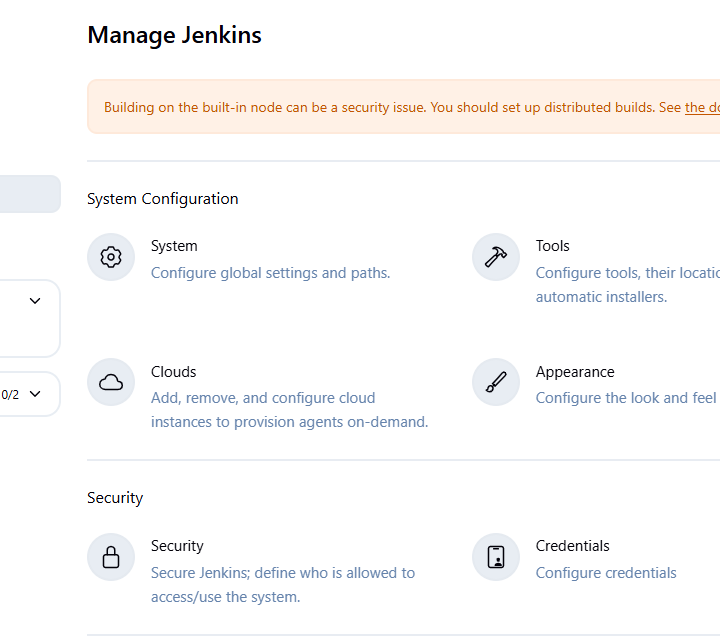
Users

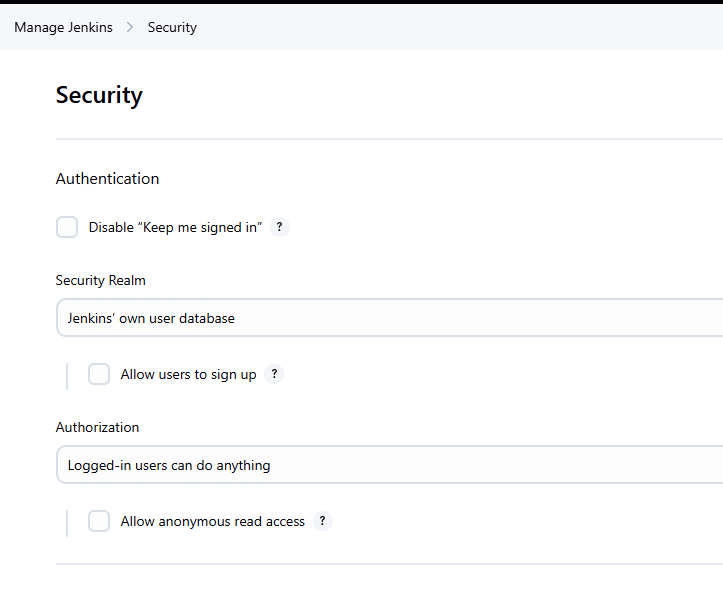


Create User

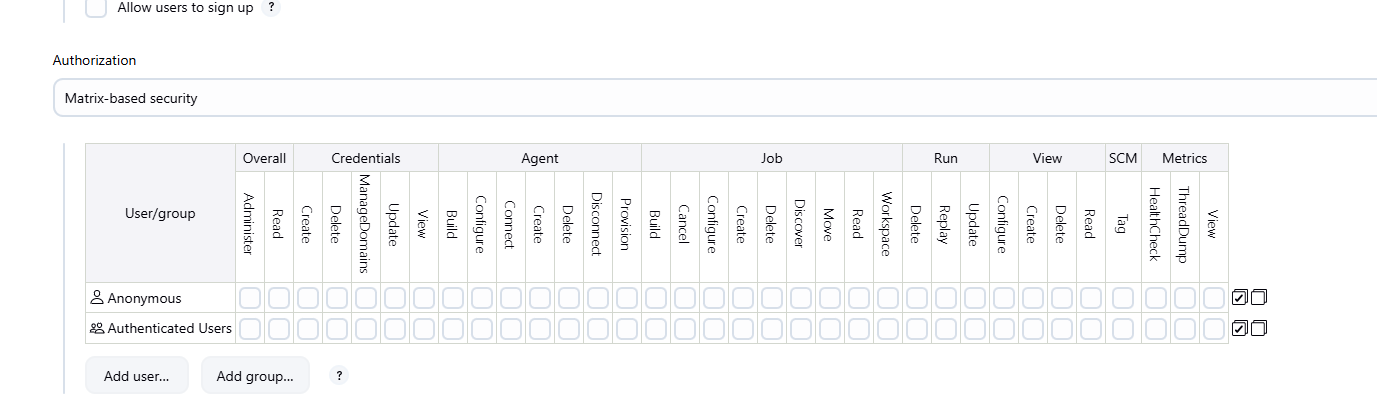


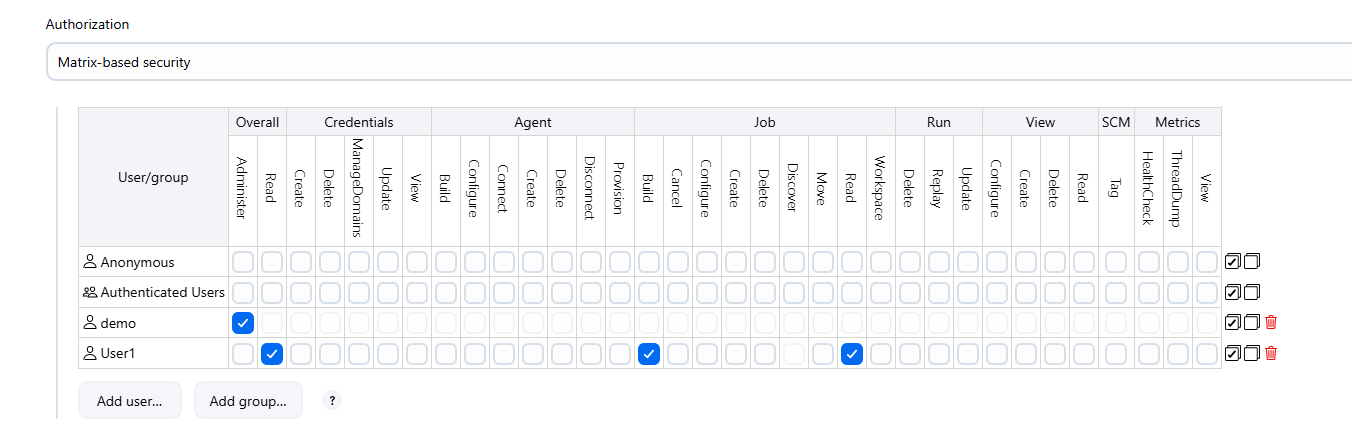






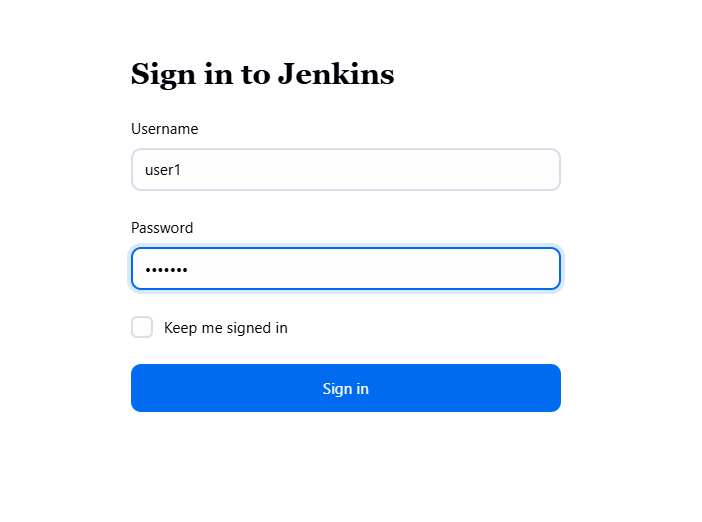
In Authorization select matrix-based security



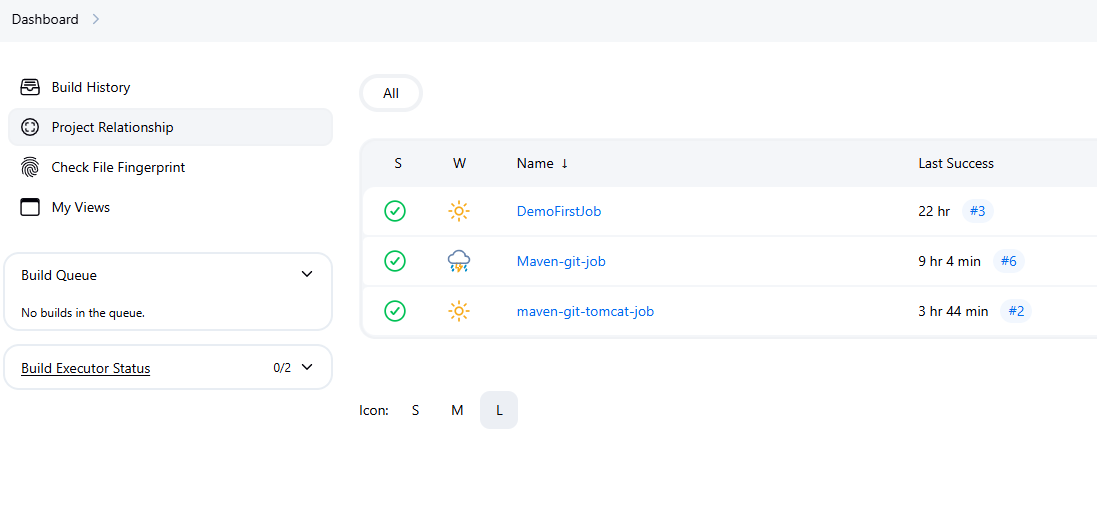


Apply then Save

Logout



For example, user1 cannot have access to Configure



Say we have 200 users are there, and it is cumbersome to give role-based access to entire 200 users imagine. The solution is to create User-Roles, Developer-Role then we will assign to specific teams

Logout then login as admin user

Creating Users and managing their Permissions

--> Jenkins dashboard

Dashboard --> Manage Jenkins --> Configure Security --> Security Realm (Jenkins own user database) -> Authorization (Matrix base security) -> Add users (demo\_user, user1) Add permission based on requirements. Enable API token stats --> Apply and Save

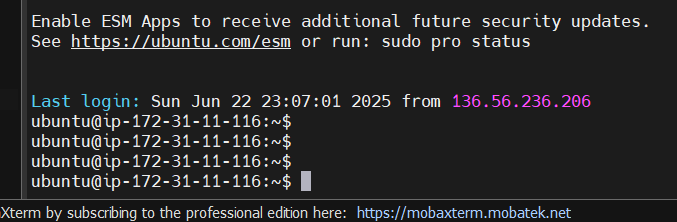
39:00

User Roles in Jenkins

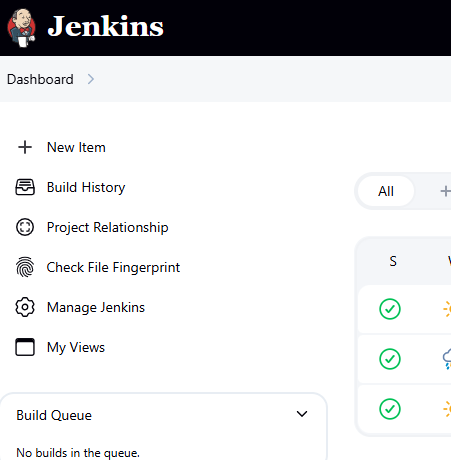
We can assign roles and in Role we can configure what that role-assigned user can do with Jenkins

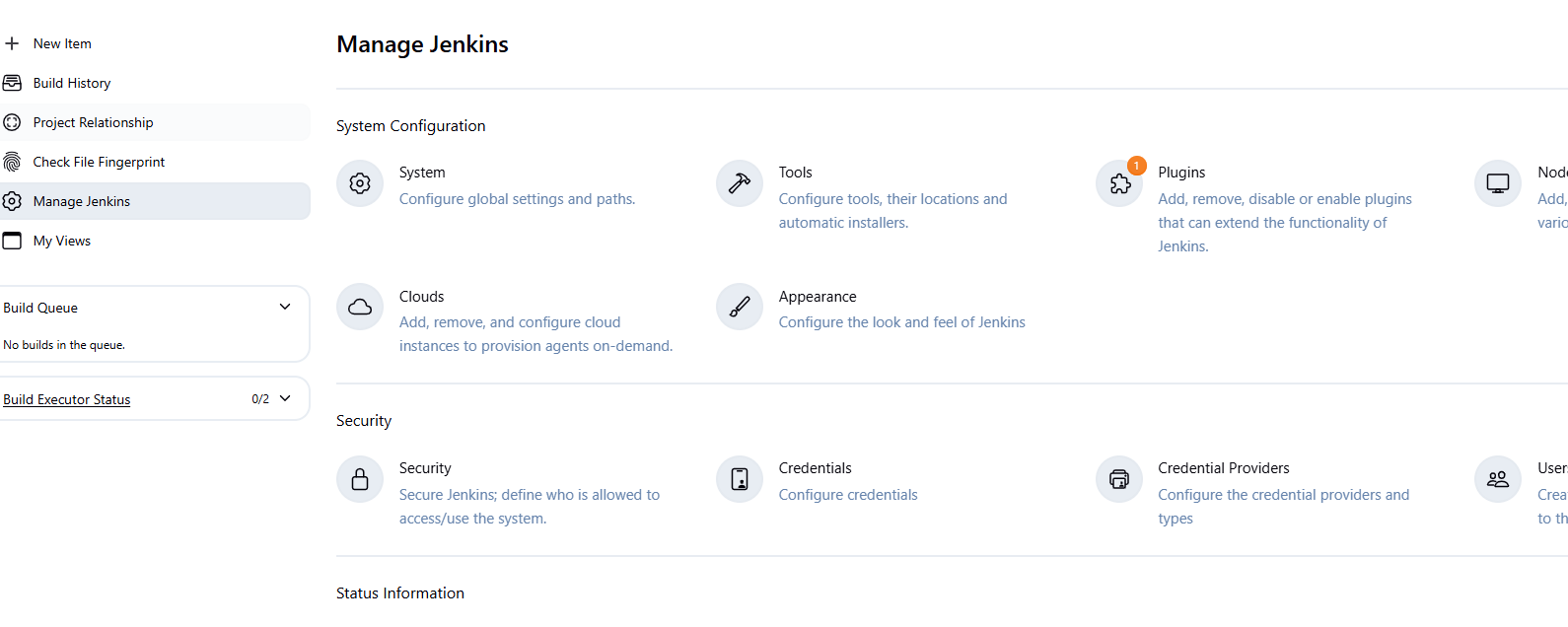
We create a Role then we add Users into that Role

--> Install “Role-based Authorization Strategy” plugin (this plugin allows us to define roles and assign users to them)



<http://99.79.49.73:8080/login?from=%2F>

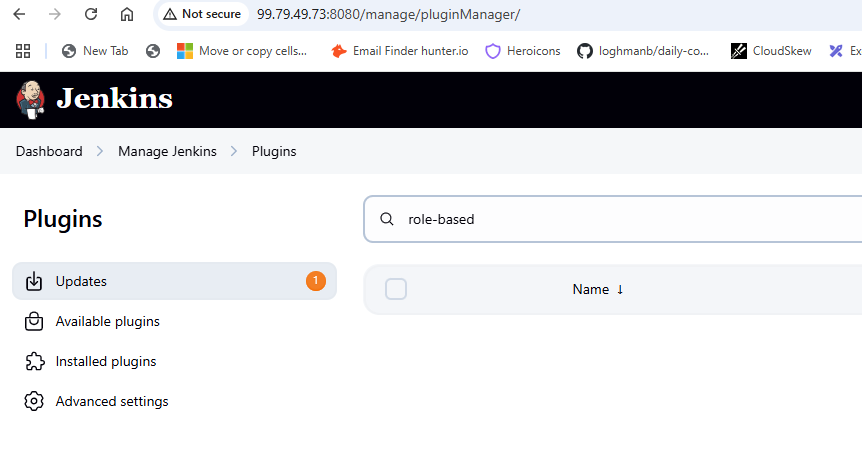




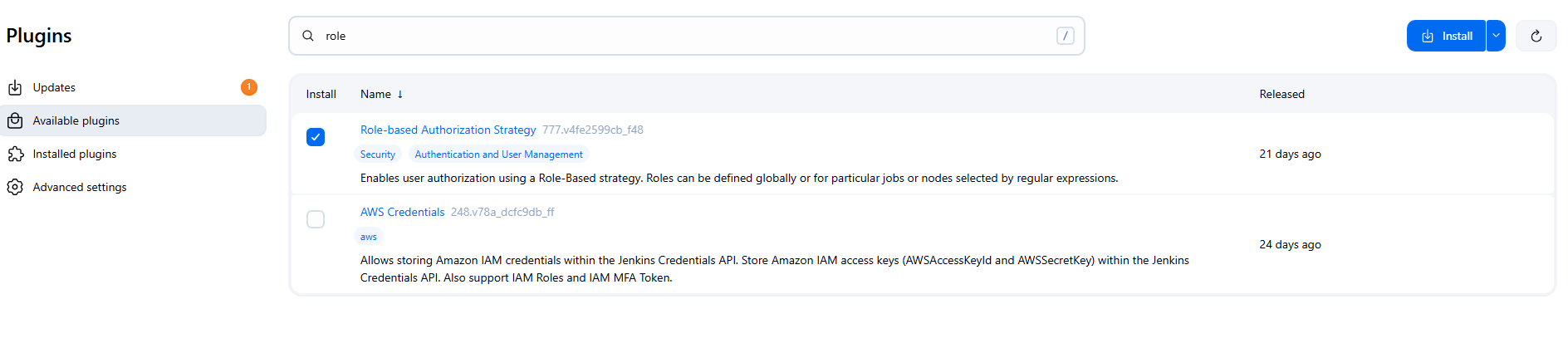
<http://99.79.49.73:8080/manage/configureSecurity/>

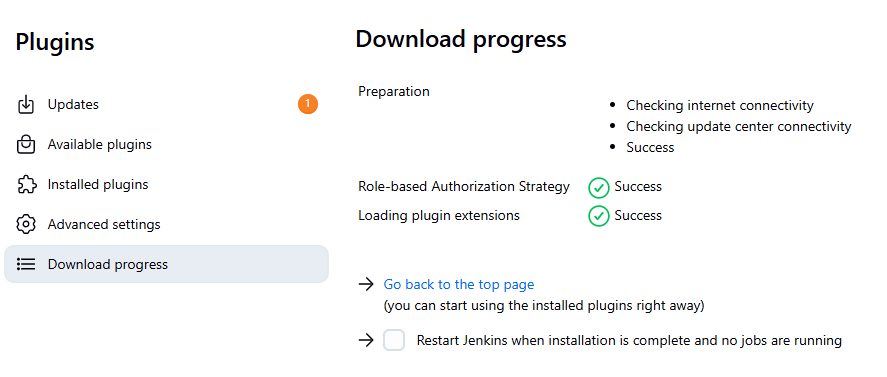
Right now Role-based option is not available

<http://99.79.49.73:8080/manage/pluginManager/>

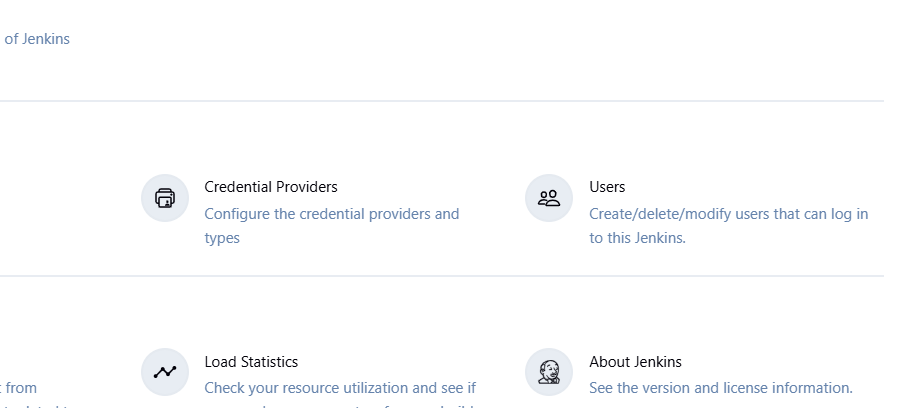


Select Role-based Authorization Strategy





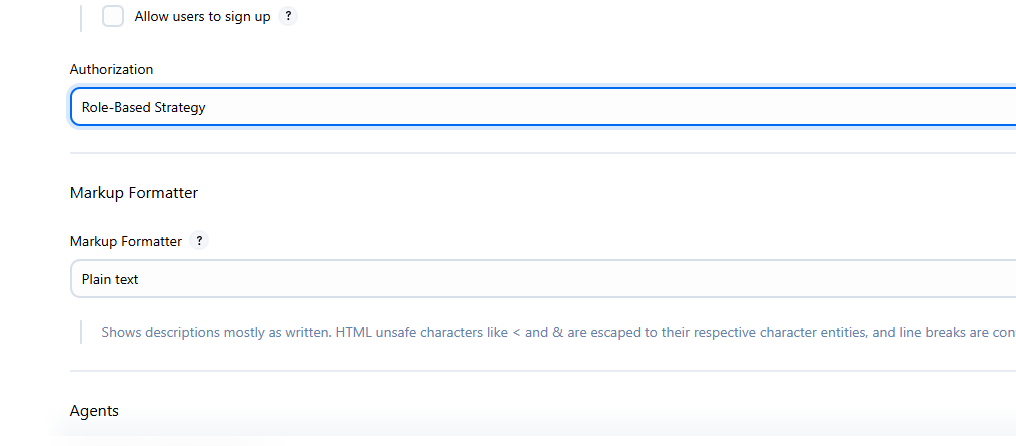
Go back to Manage Jenkins --> Users





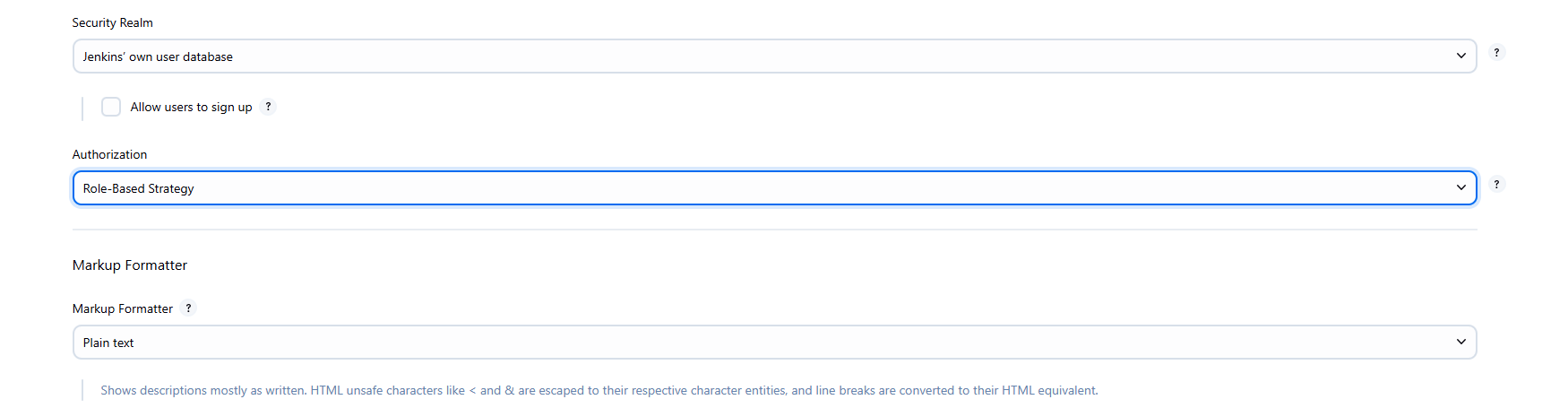
Manage --> Security

Now we can see Role-Based Strategy here



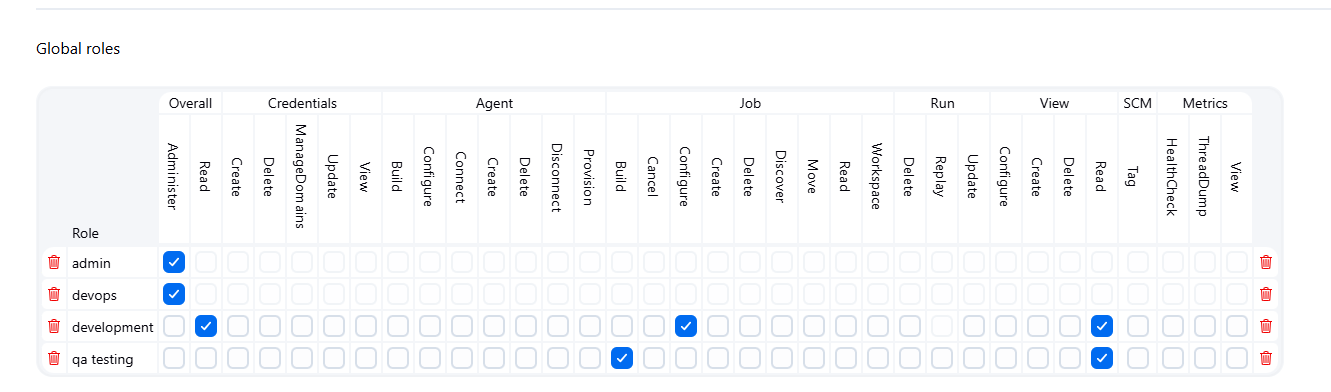
54:00

<http://3.98.56.215:8080/login?from=%2F>

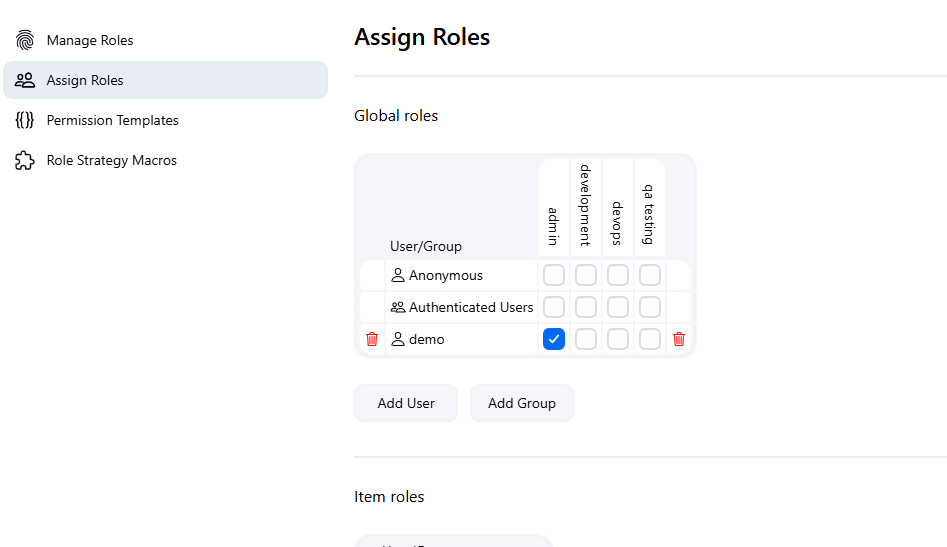


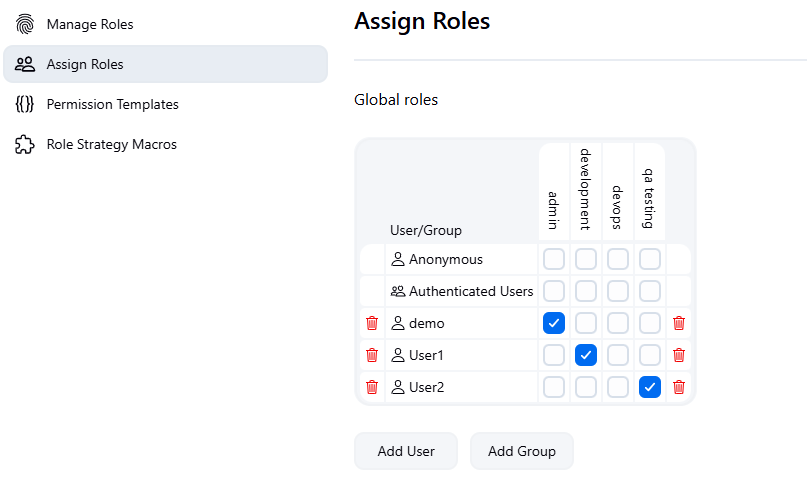


admin has entire Administrator access



Different roles like devops, development, qa testing





--> Manage Jenkins --> Configure Security --> Authorization --> “Role-based Strategy” --> Save and Apply

--> Create User Roles --> Manage Jenkins --> Manage and Assign Roles --> Manage Roles and define roles based on our requirements (admin, QA, development, ), later we went to assign role --> Add to create a new role --> Add user to the roles

(Dashboard -> Manage Jenkins -> Manage and Assign the roles -> Assign roles (refer live class recordings)

Master and Slave Architecture in Jenkins

Currently we have only one machine in the entire eco-system

So much of burden into one machine is not recommended. We can divide the burden by following Master-Slave Architecture. If you are running a pipeline, one pipeline consists of multiple jobs. So many jobs running in one machine will be a huge burden. A machine that’s already there, that’s considered as Master machine. I will create different jobs on Master machine then running jobs will be divided onto different machines (many worker nodes or slave machines), which are responsible to run the jobs. Burden will not be there on one machine.

--> If we use one machine for Jenkins, then the moment we run multiple jobs it may burden the machine, which can lead to system crash to decrease burden on Jenkins server/VM, we can go with Master-Slave configuration. It could be used to reduce the burden on Jenkins Server/VM by distributing tasks/load

Jenkins Master vs Jenkins Slave:

Jenkins Master

--> The machine or server, where Jenkins tool is installed can be referred as Master machine, which is used to create jobs, schedule jobs, and manage the jobs

--> It will also distribute jobs execution into Slave machines/servers

(We can Run/Execute jobs on Jenkins Master machine too however, it is not recommended)

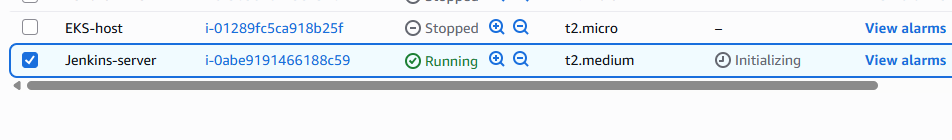
Jenkins Slave

It is a machine which is connected to Jenkins Master machine and execute the jobs/tasks assigned by Master machine/server

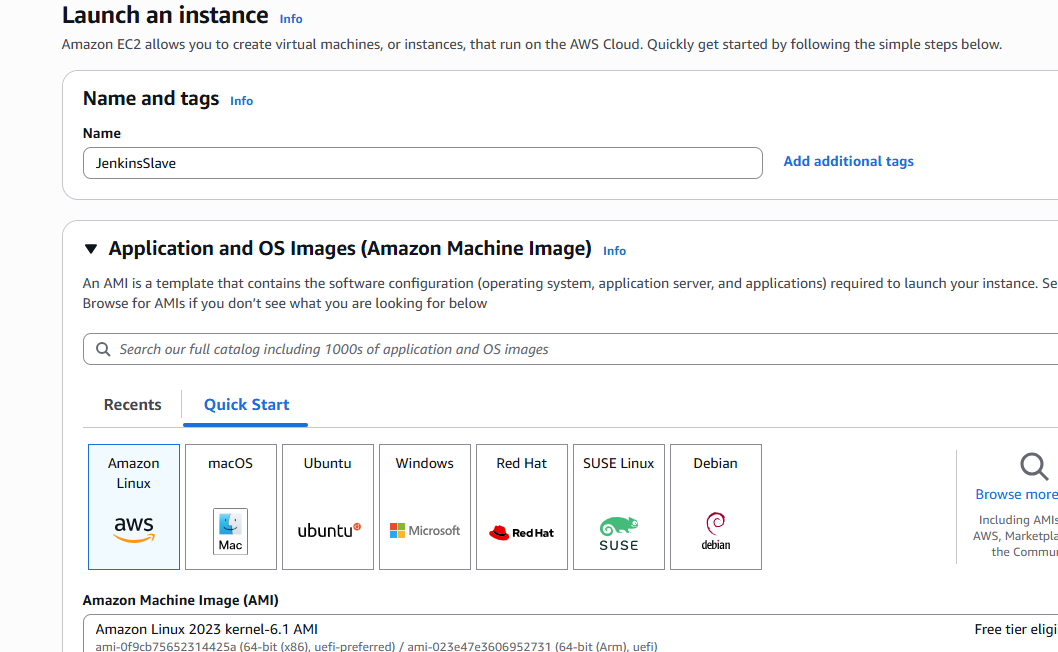
Slave server/machine will receive tasks from Master machine for the job execution

1:14:30

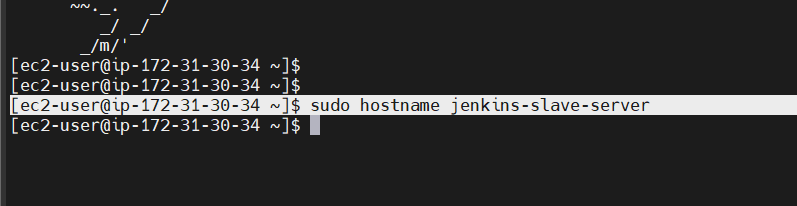
Lets Jenkins-server be the Master machine



Create JenkinsSlave VM



[ec2-user@ip-172-31-30-34 ~]$ sudo hostname jenkins-slave-server

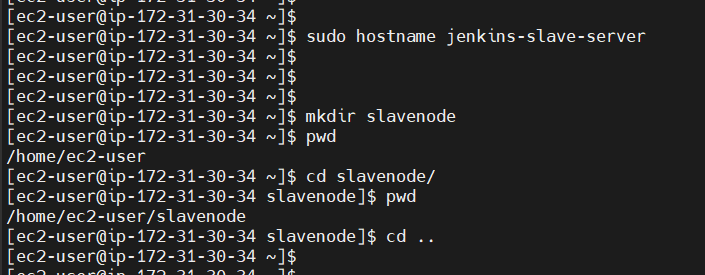


sudo rpm --import https://yum.corretto.aws/corretto.key

sudo curl -Lo /etc/yum.repos.d/corretto.repo https://yum.corretto.aws/corretto.repo

sudo yum install -y java-21-amazon-corretto-devel

We need Java for Jenkins



[ec2-user@ip-172-31-30-34 ~]$ javac -version

javac 21.0.7

1. Create Master VM

Launch Linux VM (recommended Ubuntu -> t2.medium)

Install Java

Install Jenkins

(Follow the initial steps and commands of this notes to do the same)

1. Create Jenkins Slave VM

Created Linux VM (Ubuntu -> t2.micro)

Change hostname for readability

(sudo hostname jenkins-slave-server)

$ exit

Install java

$ sudo apt install openjdk-21-jdk

Create one directory in home/ubuntu

$ mkdir slavenode

3 --> Configure Slave server/node in Jenkins Master server/node

Jenkins Dashboard --> Manage Jenkins --> Select Node option --> Assign Name and Select Permanent Agent --> Enter remote root directory (/home/ubuntu/slavenode) --> Label name (assign a name of your own choice) --> Select launch method “Launch Agent Via SSH” --> Give Host as “Slave VM DNS URL” --> Add credentials --> Select kind as SSH username with Private key and --> User as : Ubuntu --> Select Private key as enter directly and add Private key

(Open Pem file and the content of Pem file copied and added into this configuration --> Save)

Host key verification strategy --> Manually trusted key verification strategy --> Apply and Save

(Watch Live class recordings to configure by following above steps)

Go to Dashboard and create any Jenkins job and run the Jenkins job --> Console output we can verify that the Jenkins jobs are getting executed in Jenkins slave node

Build & Deployment

--> Limitation & Challenges in manual build and deployment process

--> Automating build and deployment (Jenkins)

--> CICD --> Jenkins set up in Linux (for Windows watch Youtube of link attached)

--> Job creation

--> Git + Maven job + Git + Maven + Tomcat job

--> User and Role management (Role-based access)

--> Master-slave configuration

--> Plugins management (Deploy to container plugin, Role-based strategy plugin)

Jenkins pipeline:

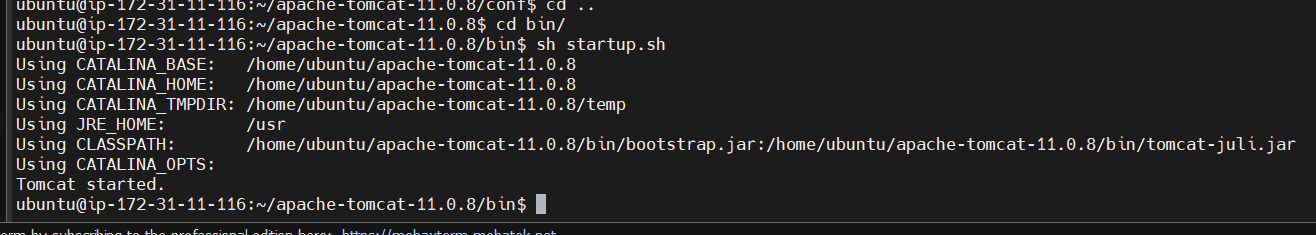
Two ways:

Declarative pipeline

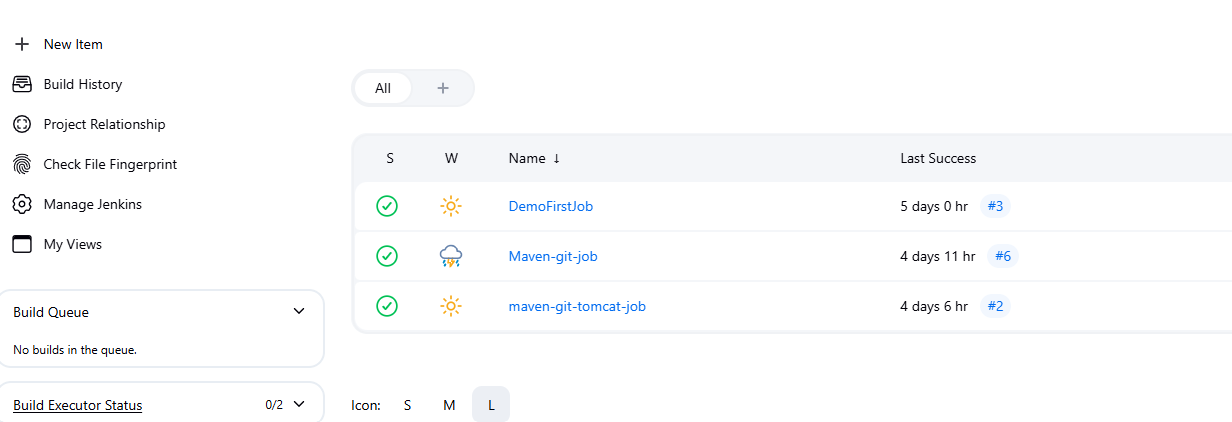
Scripted pipeline (Groovy script)

Entire build & deployment is pipeline

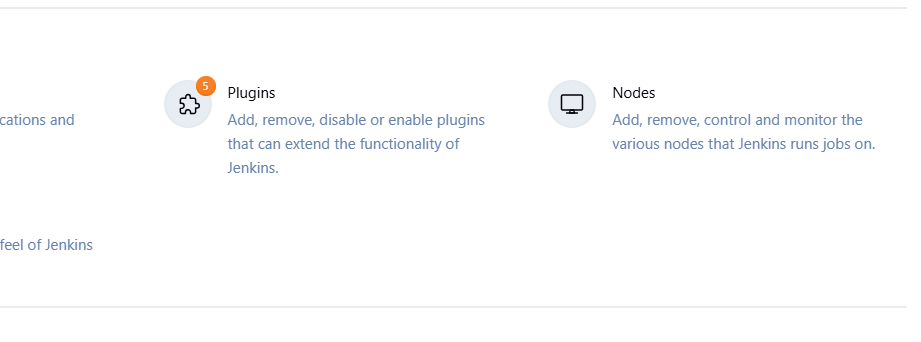
Go back to Jenkins Master



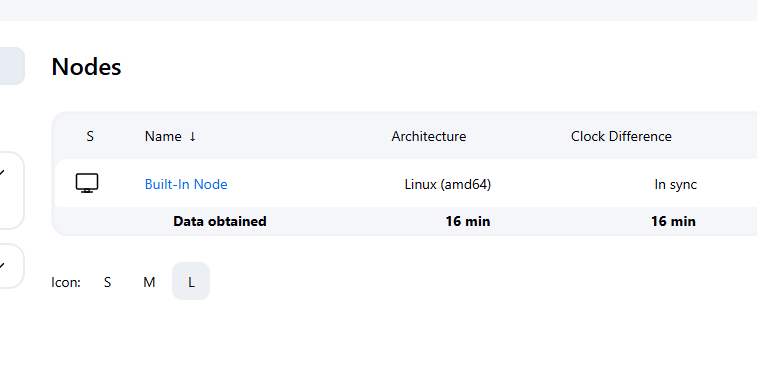
<http://35.182.80.43:8080/login?from=%2F>



Manage Jenkins

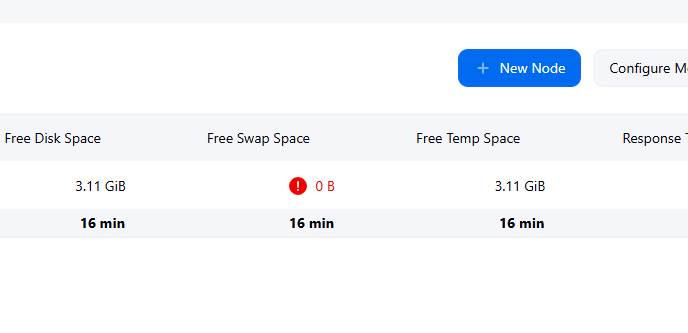


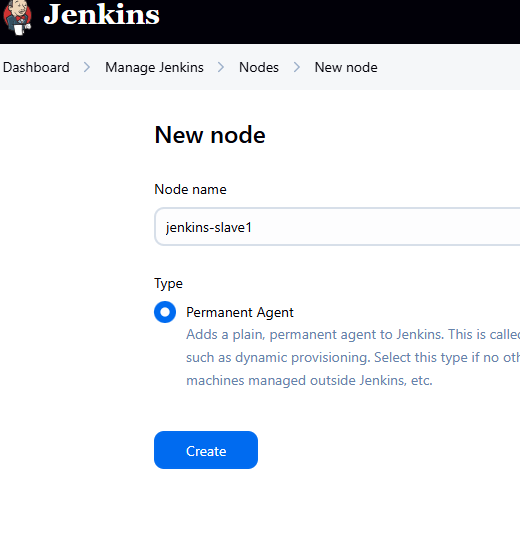
Click Nodes



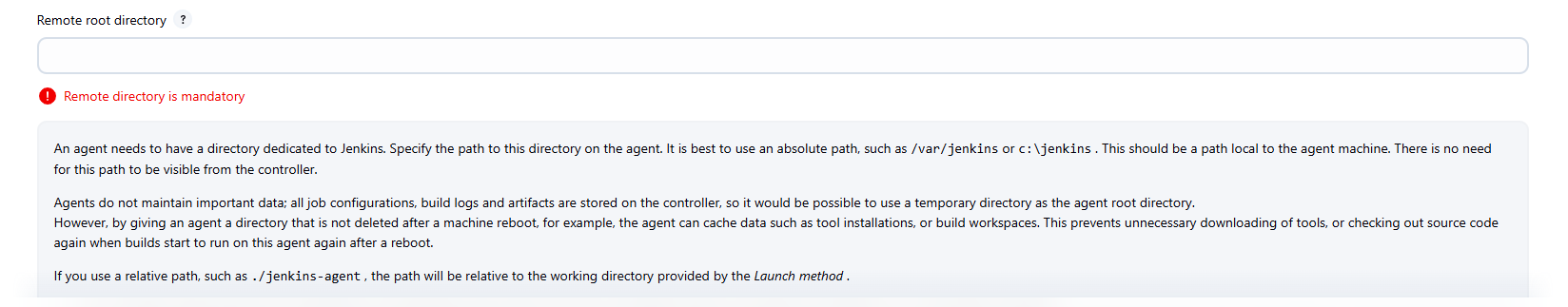
We can see one Built-In Node

Click on the New Node





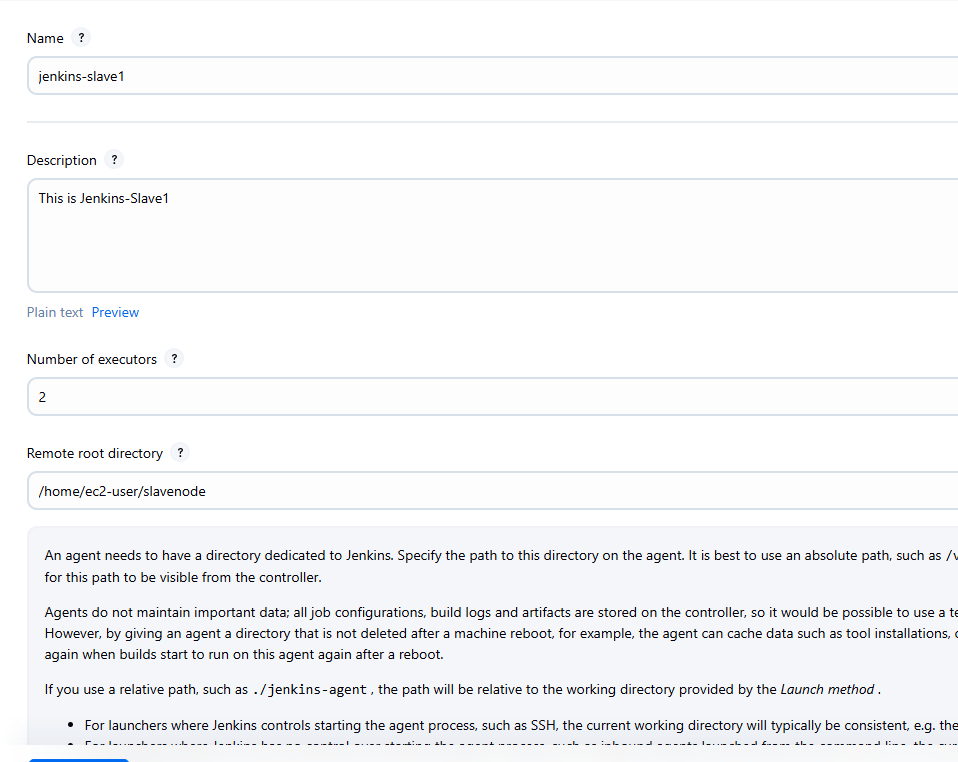
Click Create

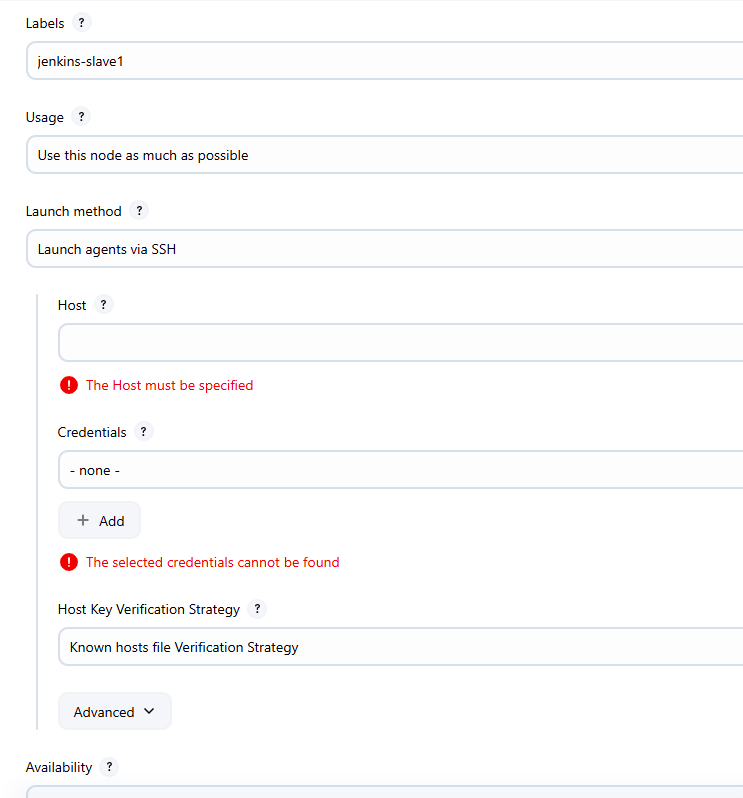


[ec2-user@ip-172-31-30-34 ~]$ cd slavenode/

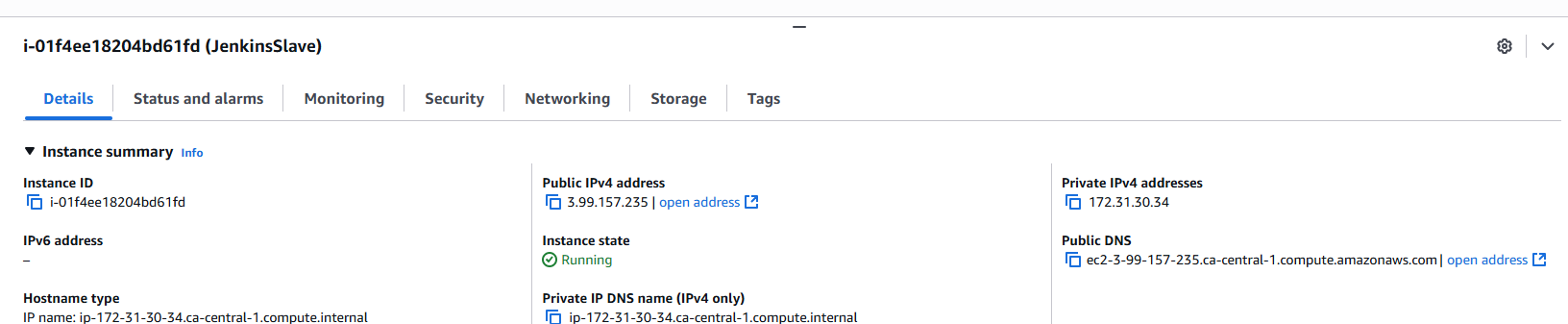
[ec2-user@ip-172-31-30-34 slavenode]$ pwd

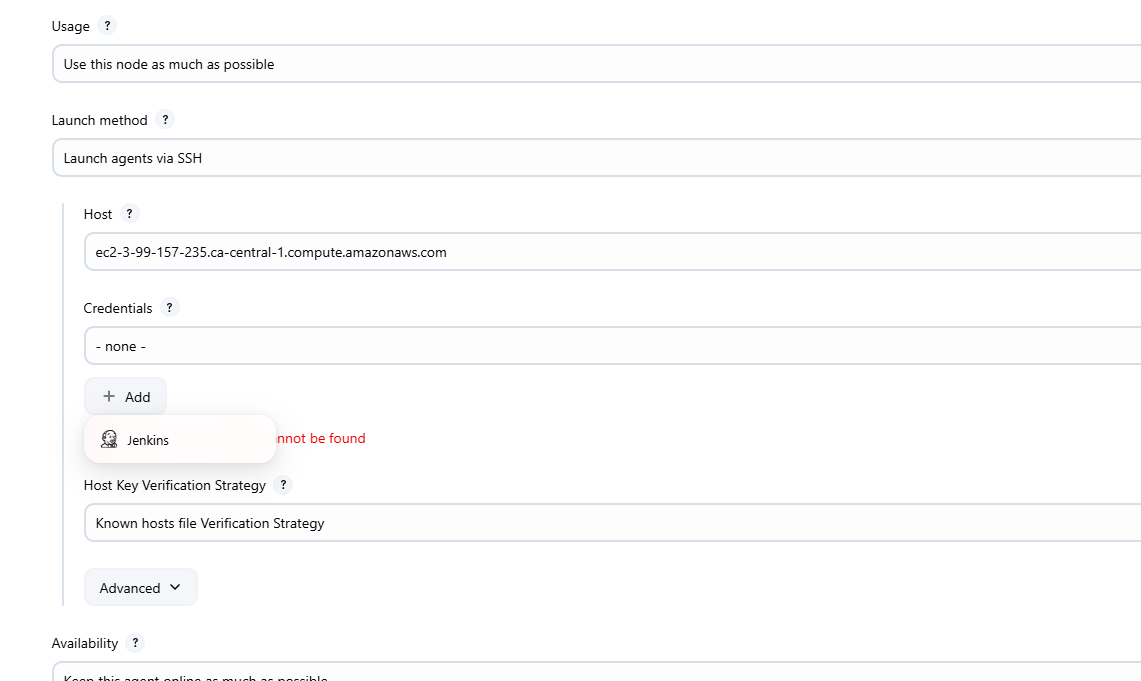
/home/ec2-user/slavenode

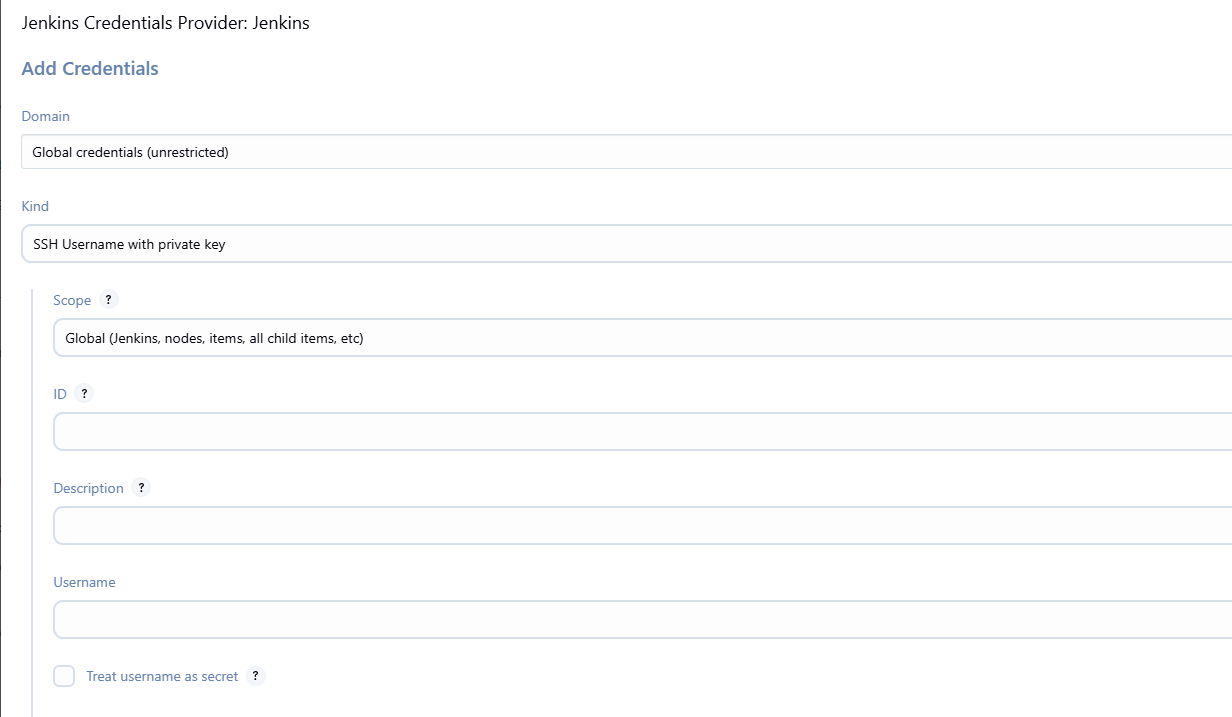


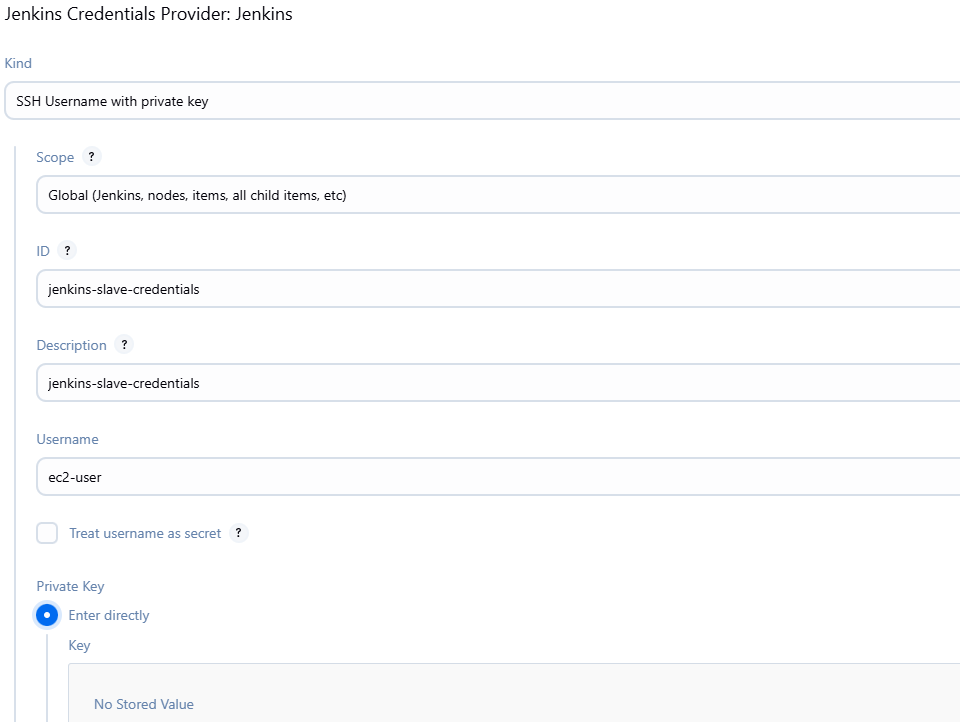


Copy Public DNS



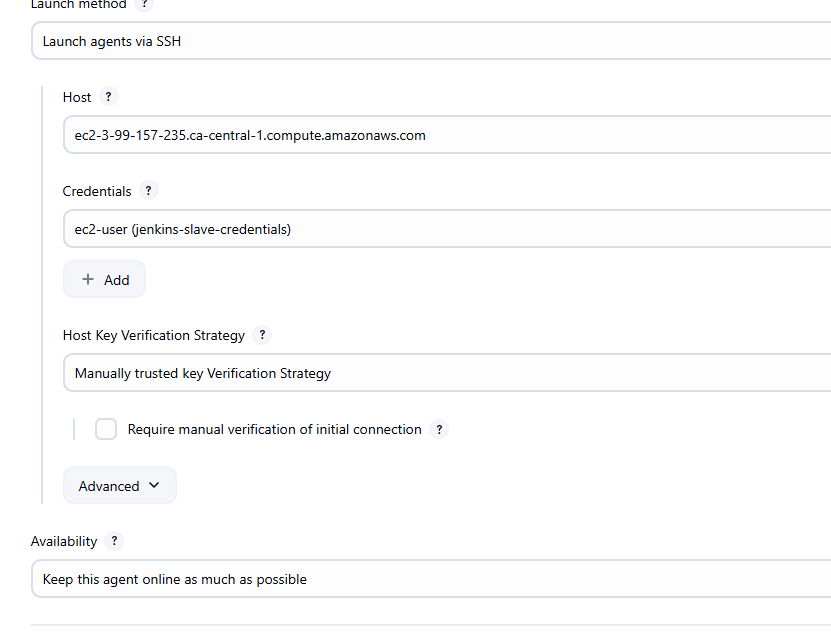


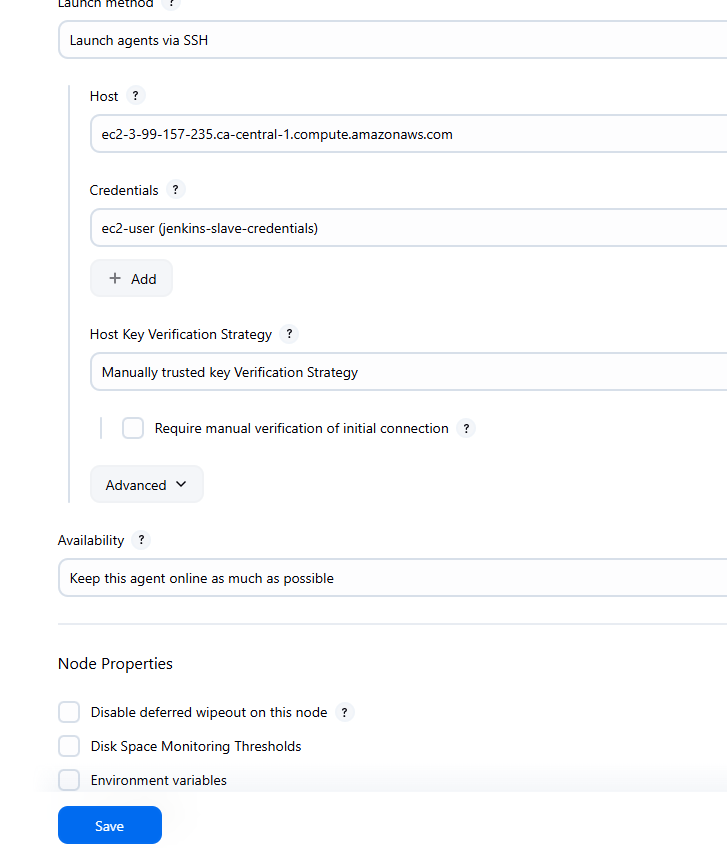




Open DevOpsMar30.pem file copy contents and paste into Private Key

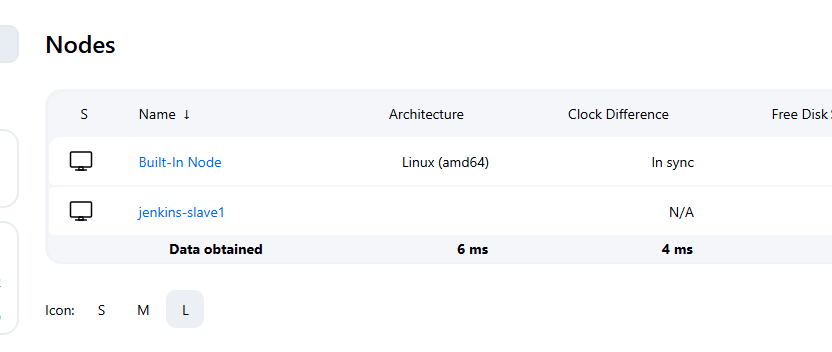
Click Add





Click Save

Now we can see jenkins-slave1 node



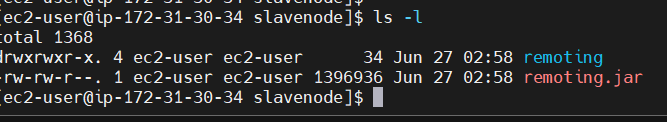
[ec2-user@ip-172-31-30-34 slavenode]$ ls -l

total 1368

drwxrwxr-x. 4 ec2-user ec2-user 34 Jun 27 02:58 remoting

-rw-rw-r--. 1 ec2-user ec2-user 1396936 Jun 27 02:58 remoting.jar

Now we can see two extra files



[ec2-user@ip-172-31-30-34 slavenode]$ cd remoting/

[ec2-user@ip-172-31-30-34 remoting]$ ls -l

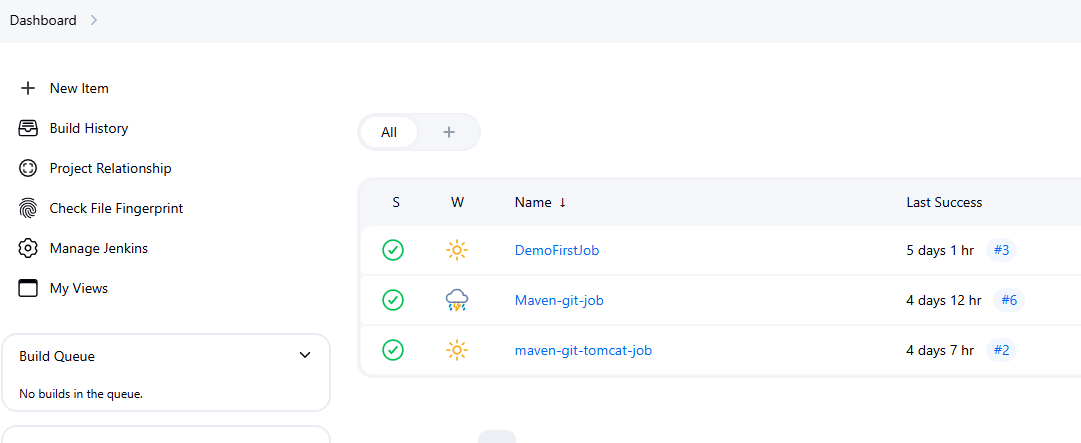
total 0

drwxrwxr-x. 11 ec2-user ec2-user 96 Jun 27 02:59 jarCache

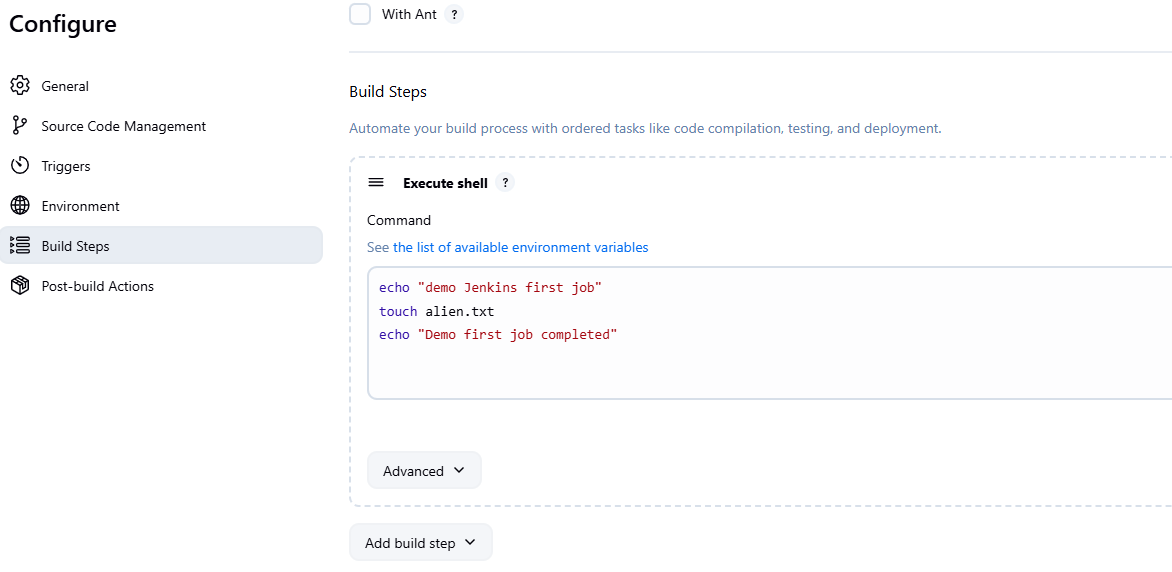
drwxrwxr-x. 2 ec2-user ec2-user 54 Jun 27 02:58 logs

[ec2-user@ip-172-31-30-34 remoting]$

Go back to Dashboard

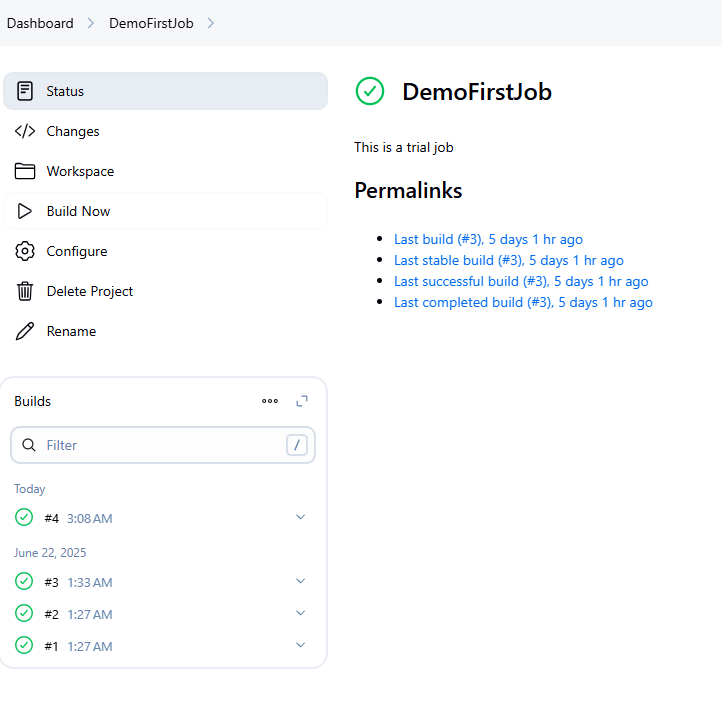


Click Configure



It is creating alien.txt

I click on Build Now

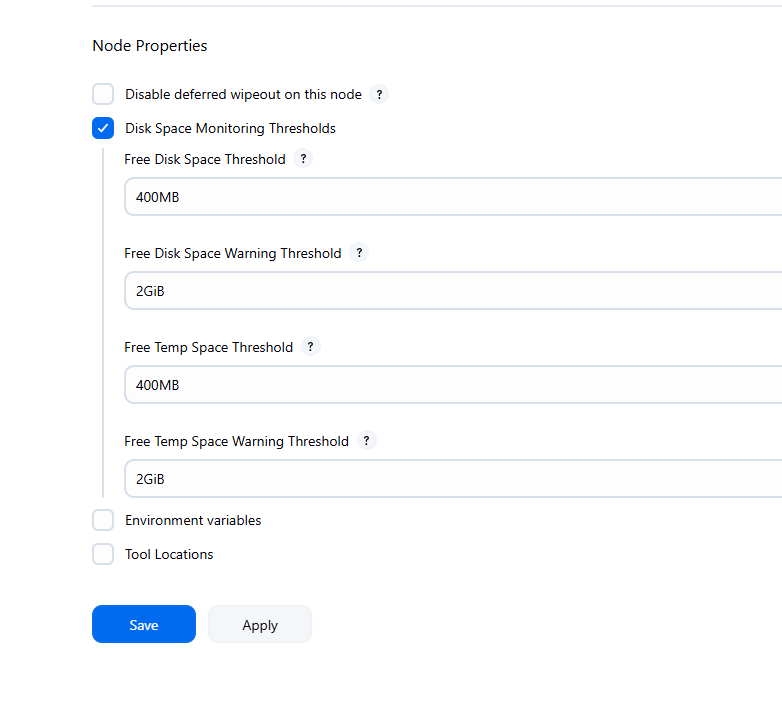


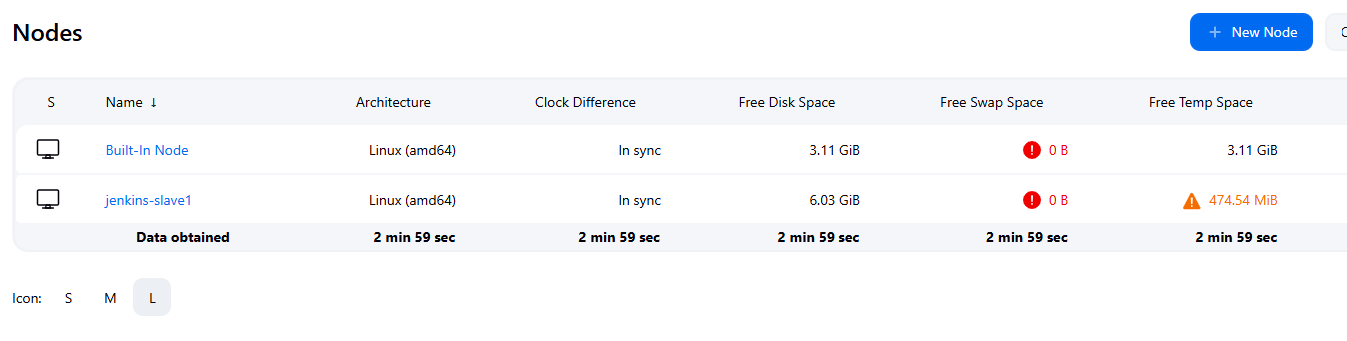
Console output

Building on the built-in node in workspace /var/lib/jenkins/workspace/DemoFirstJob



In slave-node properties





Building remotely on [jenkins-slave1](http://35.182.80.43:8080/computer/jenkins-slave1/) in workspace /home/ec2-user/slavenode/workspace/DemoFirstJob

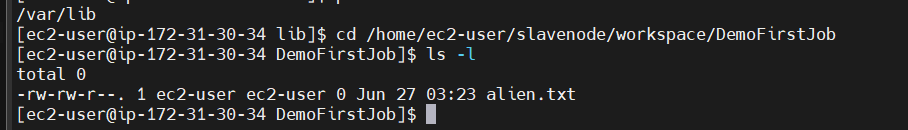


[ec2-user@ip-172-31-30-34 lib]$ cd /home/ec2-user/slavenode/workspace/DemoFirstJob

[ec2-user@ip-172-31-30-34 DemoFirstJob]$ ls -l

total 0

-rw-rw-r--. 1 ec2-user ec2-user 0 Jun 27 03:23 alien.txt

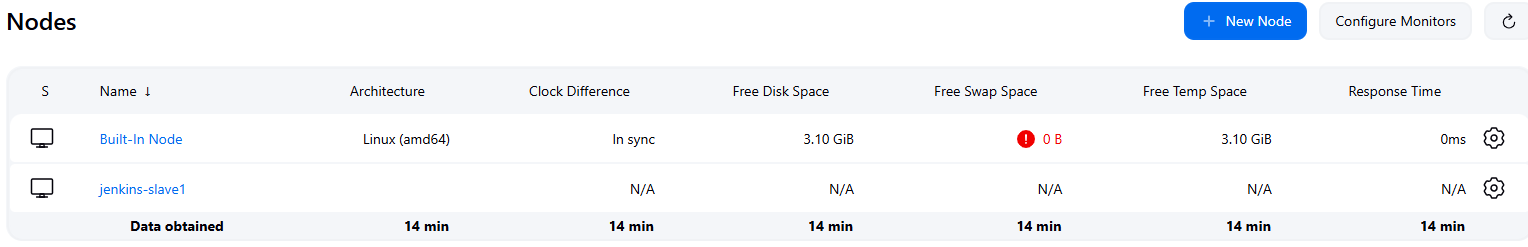


Now we can see in the slave-node that alien.txt is created

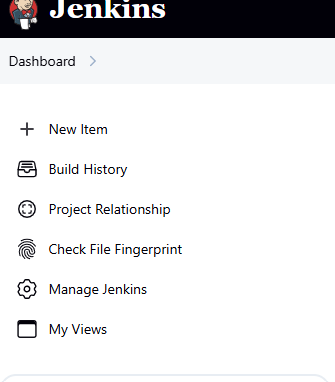
Better to create an Ubuntu machine instead of Amazon Linux VM

1:51

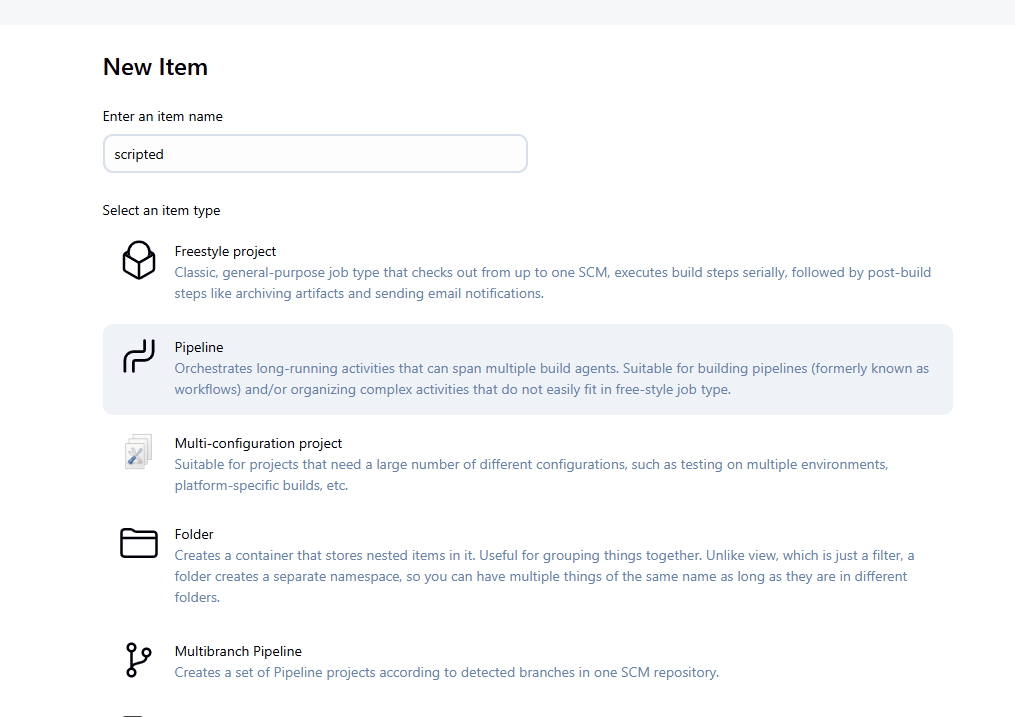
Not needed this thing



Declarative approach pipeline



New Item --> Pipeline



Click Triggers

