SCRIPTED PIPELINE

***USE CASE : Build pipeline will spin dynamic slave ec2 instance and will run the pipeline , do deployment of the artifact in Nexus repository manager . After uploading the artifact slave instance will get automatically terminated.***

***Pipeline – Gilt clone 🡪 mvn compile 🡪 mvn clean package 🡪 deploy to Nexus***

# Pre-requisites

* Custom IAM Role
* Custom images (installed with needed software’s. Java 11, maven, git…. Etc), AMLinux used here
* Create setting.xml file (used to store the 3’rd part repository credentials for authentication)
* Custom security group for mapping with node

# Creating IAM Role

Roles> Create role > AWS service use case > Amazonec2fullaccess > next > Permission: ec2 full access > give a role name > create role

Create a custom AMI image, which will be used to spin dynamically

Create ec2 instance (Linux, ubuntu, RHEL) > connect to the same > install the below software

1. $ sudo yum update -y
2. $ sudo yum install docker -y
3. $ sudo amazon-linux-extras install java-openjdk11
4. $ sudo yum install git -y
5. $ sudo yum install maven -y
6. $ cd m2 ( if .m2 directory is not there then please run the mvn clean compile or any mvn command)
7. $ vi settings.xml

Paste the below code

<settings>

<servers>

<server>

<id>nexus\_repo-snapshot</id> -- nexus repository id

<username>admin</username> ---- nexus username

<password>admin123</password> -----nexus password

</server>

</servers>

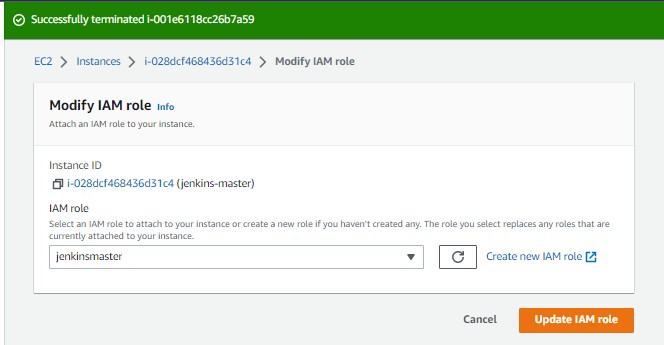
</settings>

Build the image and save it

# Create Jenkins master Server

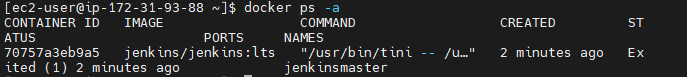
1. Create a Ec2 instance ( Jenkins master node)
2. Login to instance and run the needed commands
   1. $ sudo yum update -y
   2. $sudo yum install docker -y
   3. $sudo systemctl start docker
   4. $sudo usermod -aG docker $USER – (adding ec2-user to the group docker)
   5. \*\*\*\*restart the connection to the instance\*\*\*\*\*
   6. $docker pull Jenkins/Jenkins:lts --( pulling the Jenkins image form dockerhub)
   7. $docker images

# Attach the IAM role to Jenkins Master Server



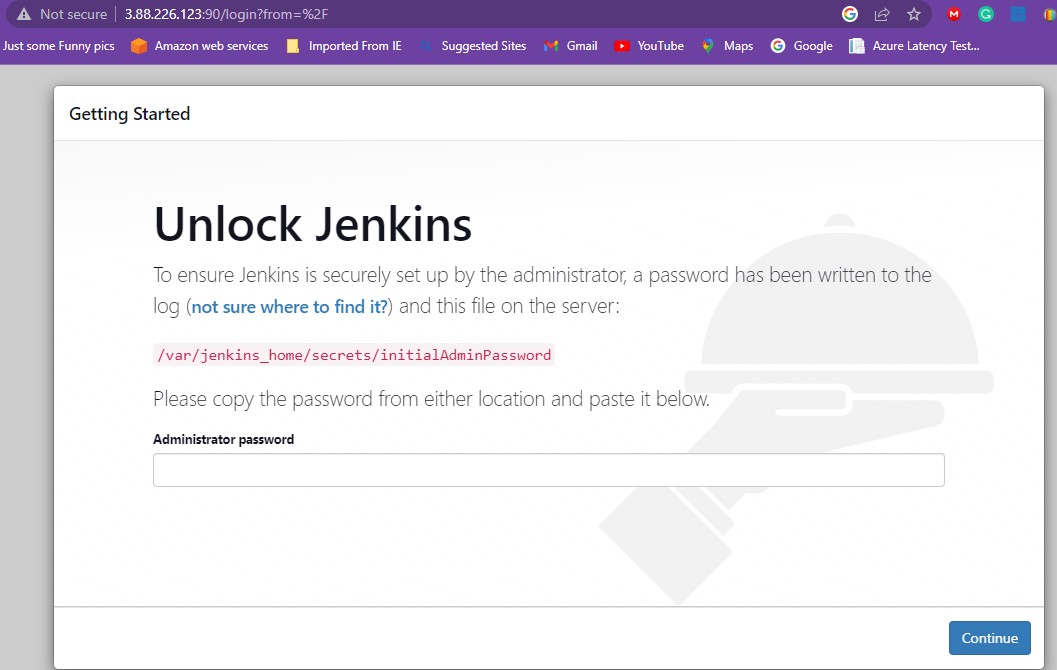
# Start the container from the downloaded image

1. docker run -d --name=jenkinsmaster -p 90:8080 -p 50000:50000 -v ${HOME}/jenkins\_home:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock jenkins/jenkins:lts
2. docker ps -a (see the container running )



# Configure Jenkins

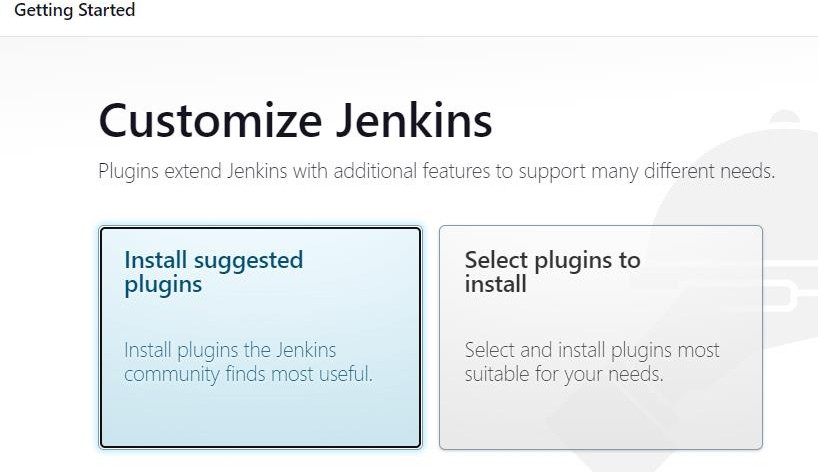
1. Open browser, copy the instance ip address and pate it with port 90



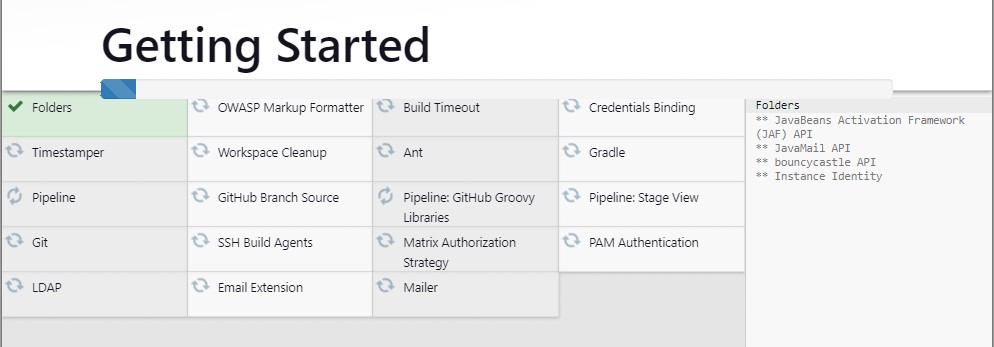
1. Navigate to the path mentioned in the page and fetch the one-time password
   1. docker exec -it <container name> bash

You can fetch the same from the external volume attached also .

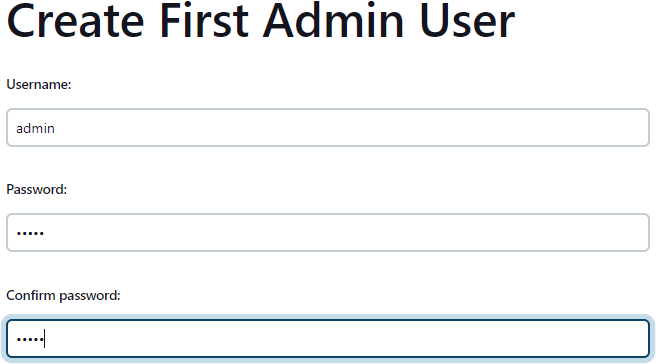
1. Copy the password and paste it on Jenkins home page and



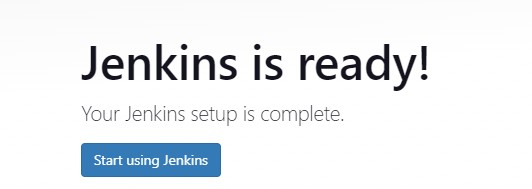
install all the suggested plugins



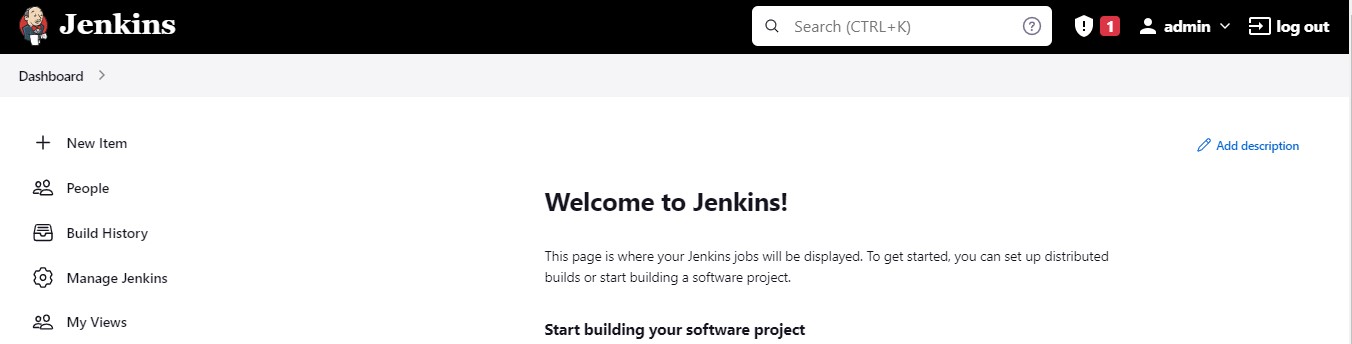
1. Create the default user id , password and login to Jenkins master node



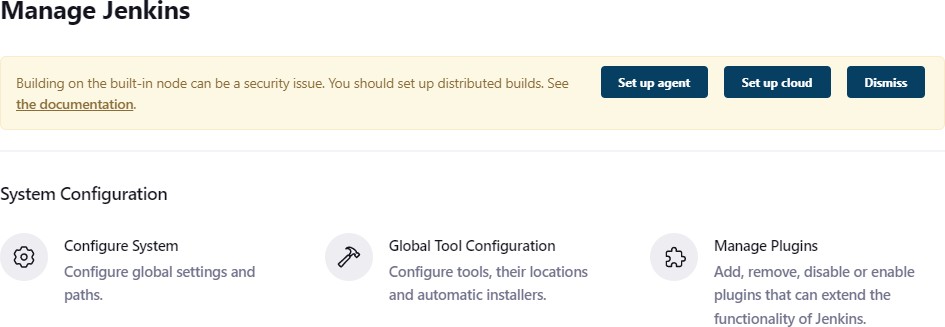
# Start using Jenkins



1. Jenkins Home page

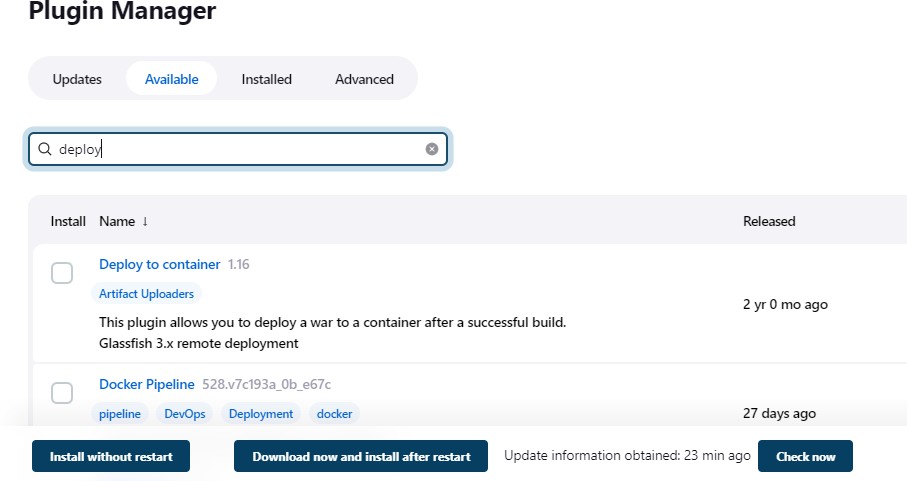


1. Navigate to Manage Jenkins > manage plugins



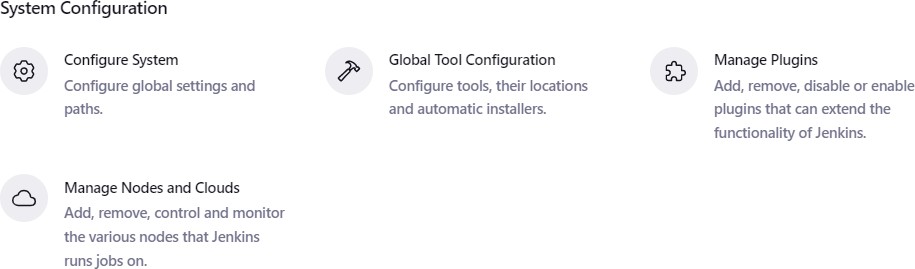
**Install the suggested plugins**

* Deploy to container
* Docker
* Amazon ec2
* [CloudBees AWS Credentials](https://plugins.jenkins.io/aws-credentials)
* SSH agent
* Pipeline utility
* Sonarqube scanner & utilities
* Blue ocean

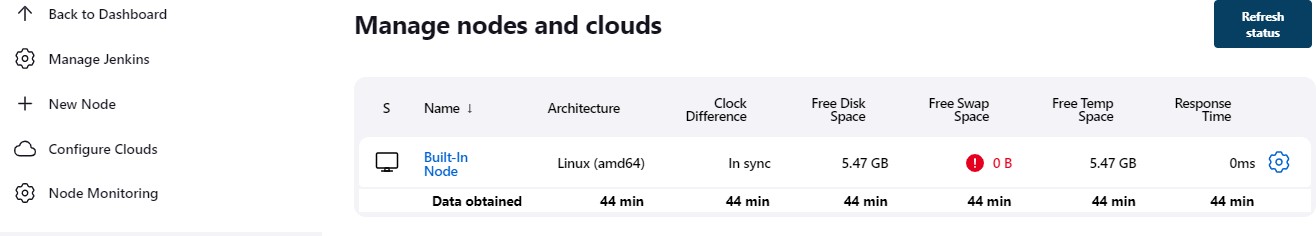


# Configure Slave Instance

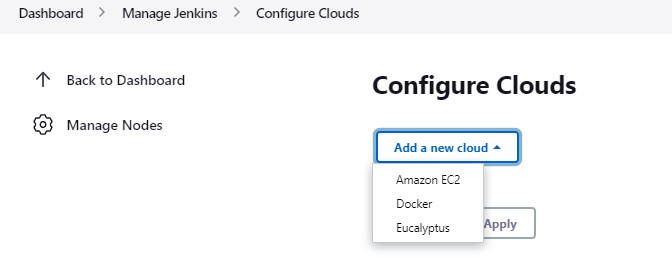
After all the plugins are installed, time to create the slave medium node >manage node & cloud



1. Configure cloud



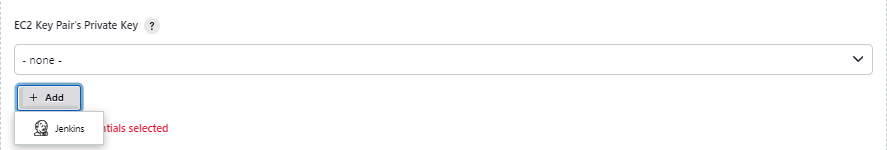
1. Choose Amazon Ec2



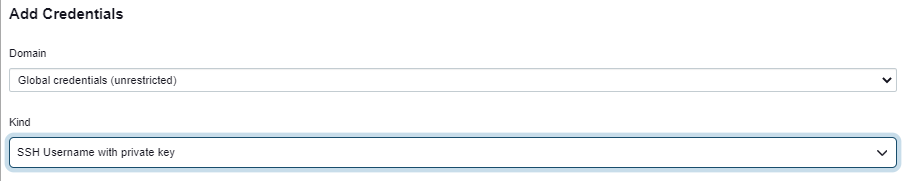
1. Give a name <medium>; > choose the use ec2 instance profile to obtain credentials



1. Register the AWS – ec2-user credentials with pair key



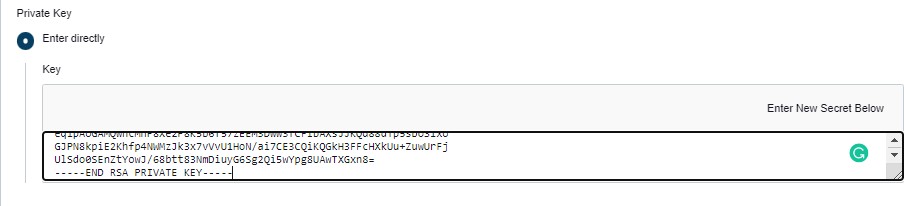
1. Choose the SSH username with private key



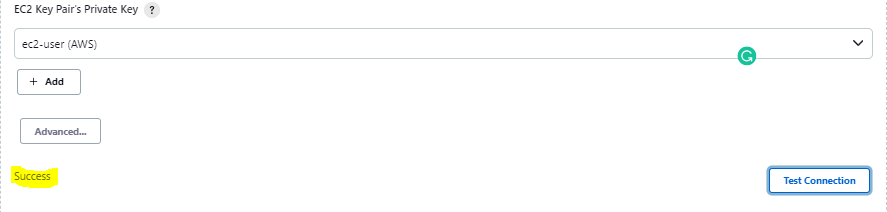
1. Enter the fields as appropriately



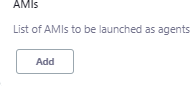
1. Enter the private key ( PEM file which was used to connect the EC2- copy the contents fully)



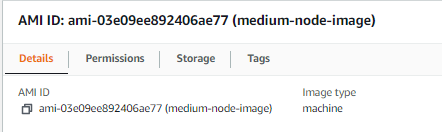
1. Test the connection on the Jenkins



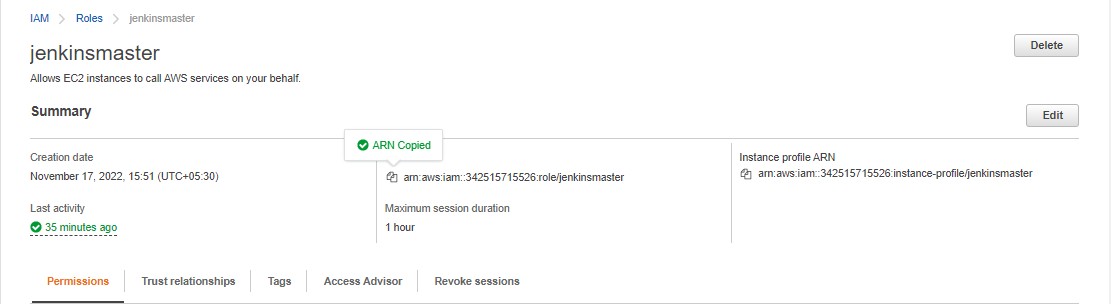
1. Now add image which needs to be used for slave instance (our-case, AMI created will be used)



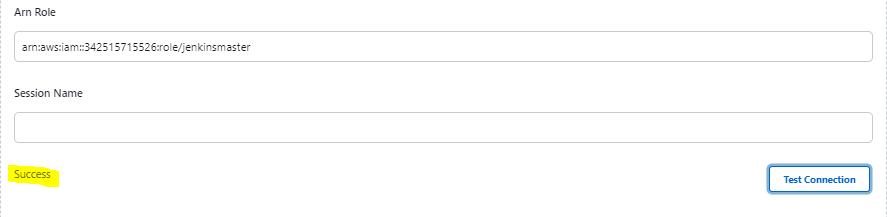
1. Copy the AMI ID and paste the same



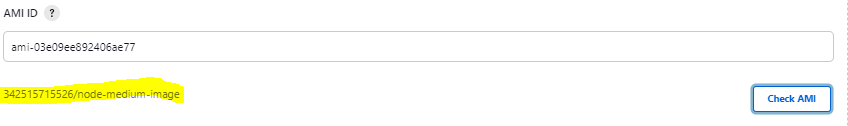
1. Enter the ARN id ( copy the same from the Role created)



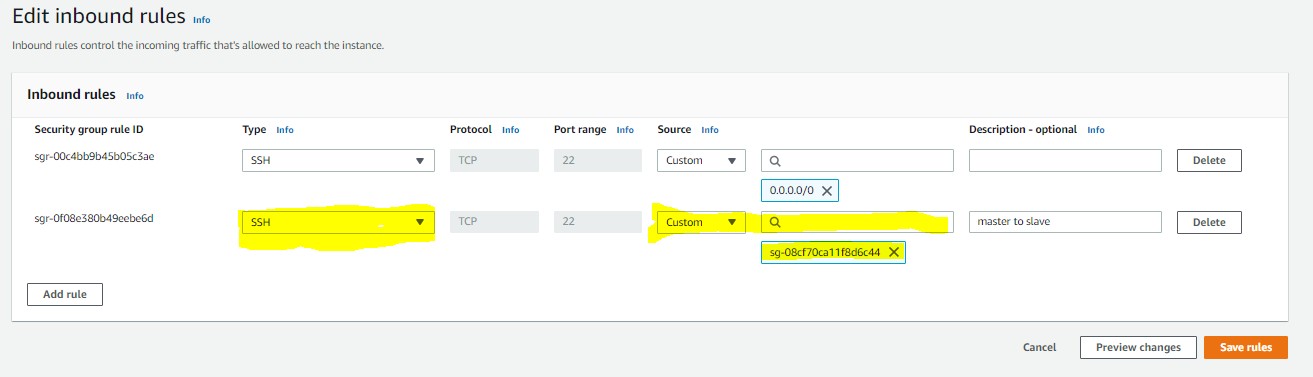
1. Validate the same



1. Check the AMI validity ( do not forget to give the region) and the type accoridngly(T2-micro)



1. Map the security group\*\* security connections slave security group to be mapped \*\*



type is TCP port 22 with Source as master server security group

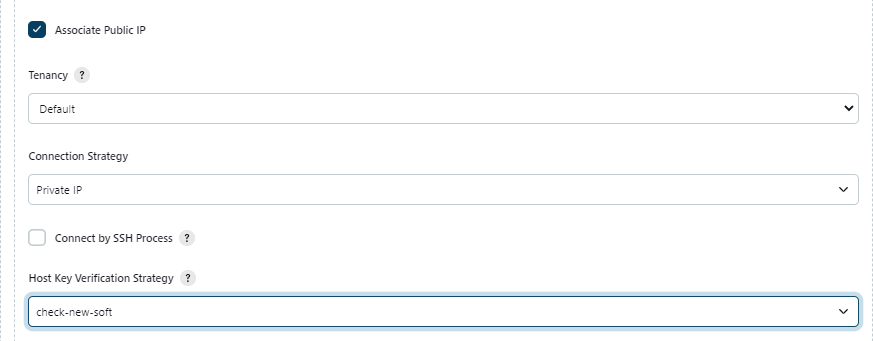
1. Enter the remaining fields as applicable ( see screen shot)



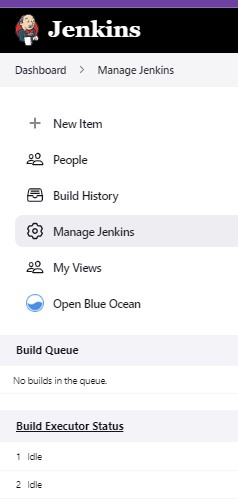
1. Map the label and usage accordingly



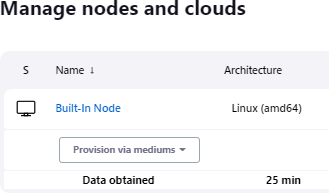
1. Click advanced – ensure public-ip is associated and verification strategy is soft



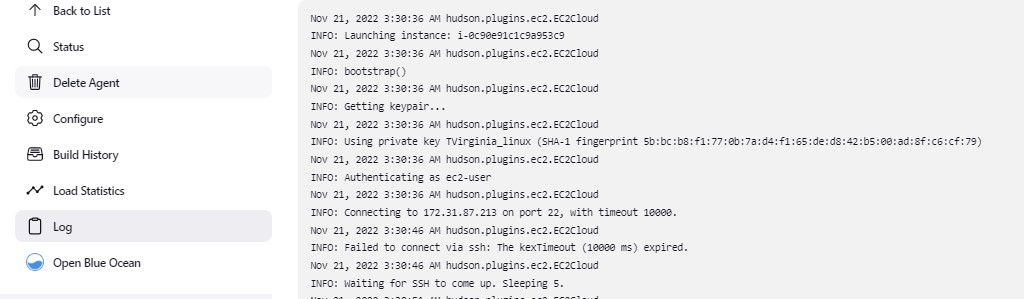
1. Apply and Save >>> Now check whether the node is spinning correctly> go to Build executor status

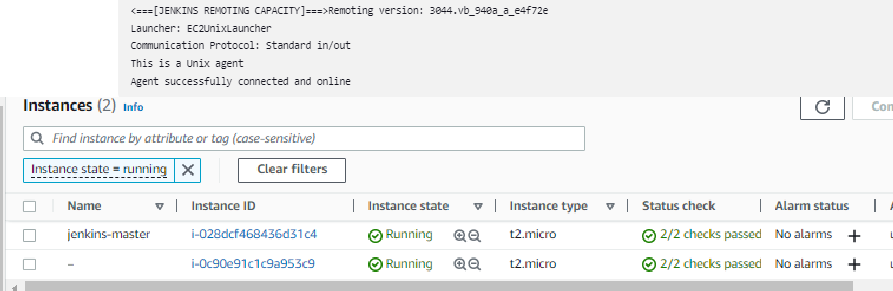


1. You should be able to see the medium connection we just created



1. Open the medium drop down and see the progress



1. Successfully connected and online
2. Write a pipeline to connect to your github repository and build the code using the medium node

node ("medium")

{

stage ("git-clone")

{

echo " Pulling changes from branch"

checkout([$class: 'GitSCM', branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[credentialsId: 'github', url: ['https://github.com/Mururadh/Ashokit-project.git']]](https://ind01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2FMururadh%2FAshokit-project.git%27%5D%5D&data=05%7C01%7CMurugesh.R%40TechMahindra.com%7C210a869171a34ca7ccf308dacb7231eb%7Cedf442f5b9944c86a131b42b03a16c95%7C0%7C0%7C638045988575335507%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=0sLQCLbCHR6PSn9OnIfVNi4QxXIyfzIuT2cvxJe5Z%2Bs%3D&reserved=0)])

}

stage ("Maven build code")

{

echo "\*\* Testing the code\*\*\*" sh 'mvn -v'

sh 'mvn clean package'

}

stage("push to nexus")

{

echo "pulling code from medium slave " stage ("Push Nexus")

{

echo " Pushing to nexus Now "

sh """mvn -U org.apache.maven.plugins:maven-deploy-plugin:2.8.1:deploy-file -DgroupId=in.ashokit \

-DartifactId=01-maven-web-app \

-Dversion=1.0-SNAPSHOT \

-Dpackaging=war \

-Dfile=target/01-maven-web-app.war \

-DrepositoryId=nexus\_repo-snapshot \

-Durl=[http://3.87.215.204:85/nexus/content/repositories/nexus\_repo-snapshot/](https://ind01.safelinks.protection.outlook.com/?url=http%3A%2F%2F3.87.215.204%3A85%2Fnexus%2Fcontent%2Frepositories%2Fnexus_repo-snapshot%2F&data=05%7C01%7CMurugesh.R%40TechMahindra.com%7C210a869171a34ca7ccf308dacb7231eb%7Cedf442f5b9944c86a131b42b03a16c95%7C0%7C0%7C638045988575335507%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=nS639PptTT%2Buz2KXhLQ%2FV5kaQDIDqfp0QGT60D%2FSTis%3D&reserved=0)"""

}

}

}

https://github.com/Mururadh/Ashokit-project.git