**CI Pipeline**

**Pre-Requisites:**

**Server:1**

* Install Git
* Install Maven
* Install Jenkins
* Create S3 bucket in AWS
* Create IAM User and get credentials(Username and Password)
* Install Docker
* Install Mysql

**Server:2**

* Install EKS Cluster

**Install mysql:**

wget https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm

yum localinstall mysql57-community-release-el7-11.noarch.rpm -y

yum install mysql-community-server -y

systemctl start mysqld.service

# Here we get pwd

cat /var/log/mysqld.log

# Mysql Login

mysql -u root -p

# Change Password for root user

ALTER USER 'root'@'localhost' IDENTIFIED BY 'Naresh#240';

# Create User

create user 'naresh'@'localhost' IDENTIFIED BY 'Naresh#240';

# Grant all permissions for user

GRANT ALL PRIVILEGES ON \*.\* TO 'naresh'@'localhost' WITH GRANT OPTION;

# Create User

create user 'naresh'@'%' IDENTIFIED BY 'Naresh#240';

# Grant all permissions for user

GRANT ALL PRIVILEGES ON \*.\* TO 'naresh'@'%' WITH GRANT OPTION;

# Create database mysqldb

CREATE DATABASE mysqldb

# Change to database mysqldb

USE mysqldb;

# Create table with the name of employee

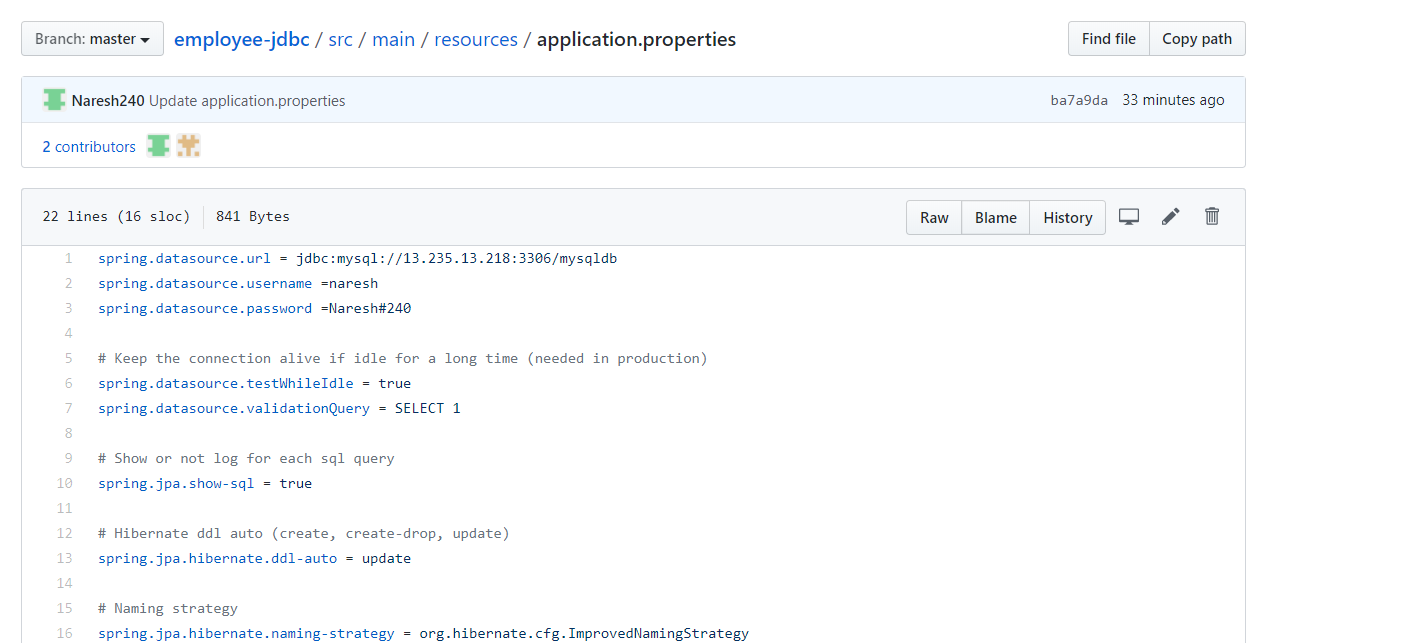
create table employee(empId varchar(40), empName varchar(40));

**Note:** Goto project and change JDBC connection:

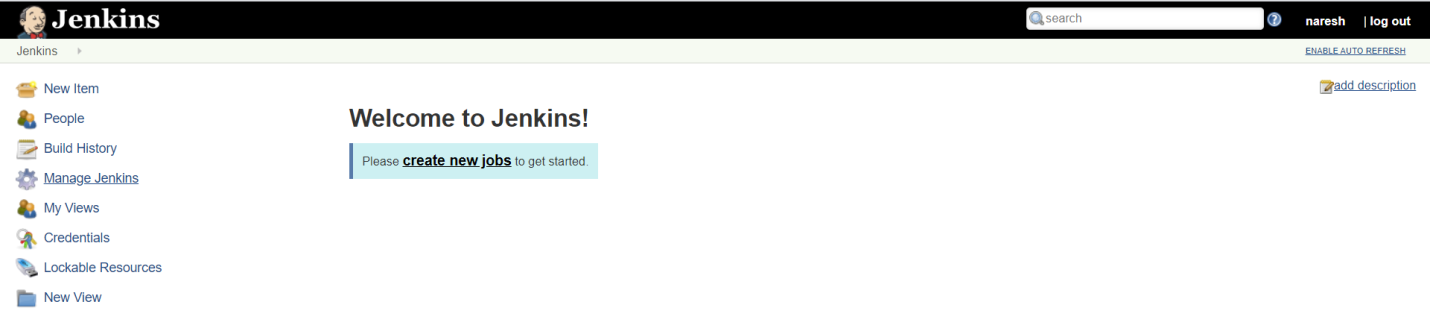
**Path:** employee-jdbc/src/main/resources/application.properties

spring.datasource.url = jdbc:mysql://<IP-Address>:3306/mysqldb

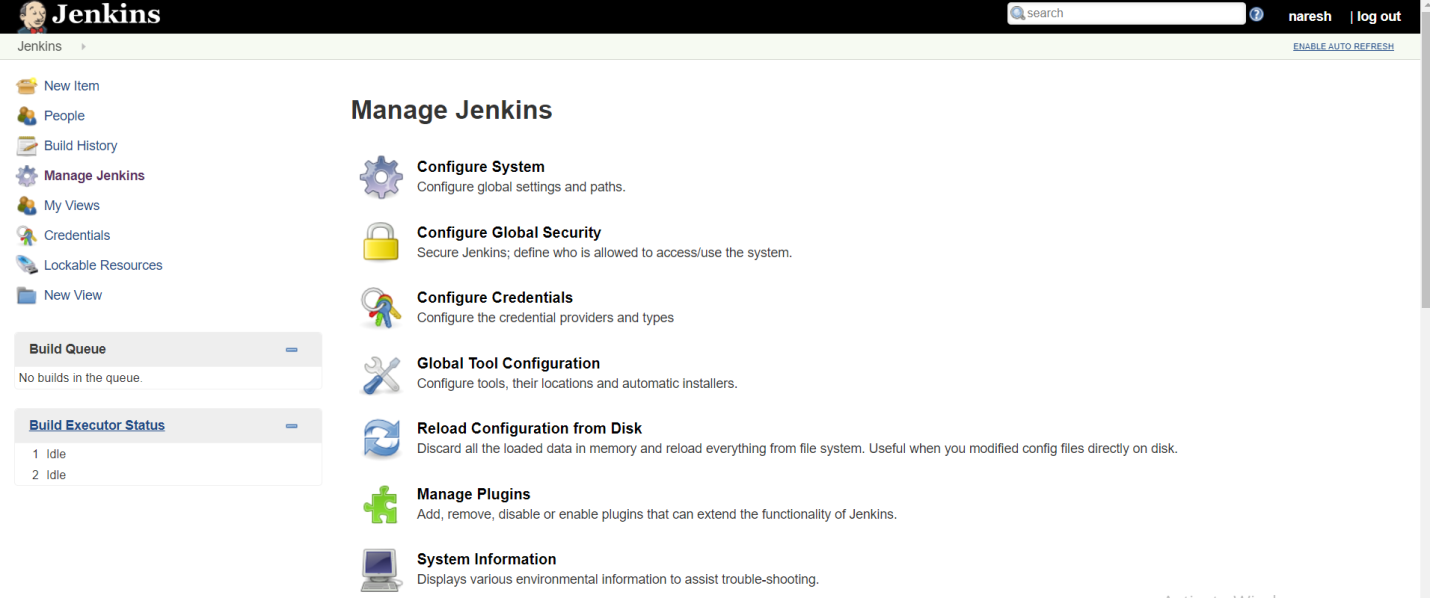
IP-Address means where we installed our mysql



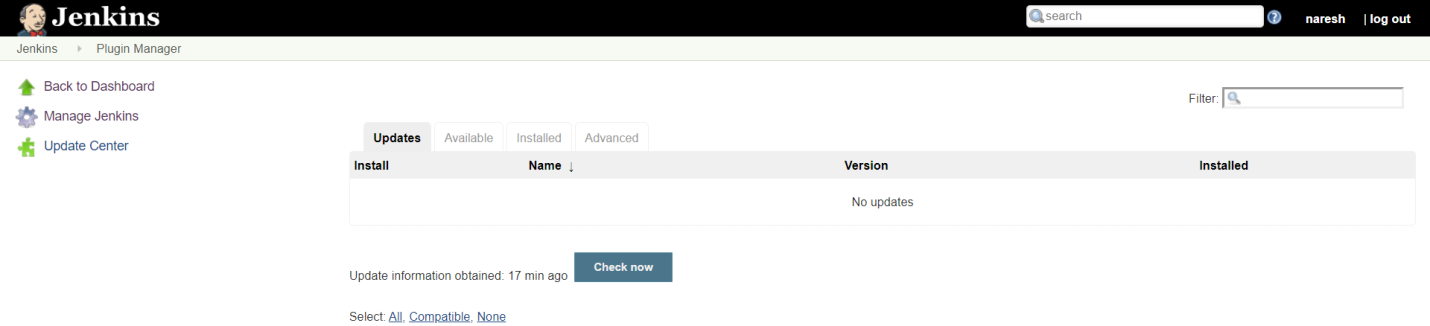
**Step 1:** Open our Jenkins



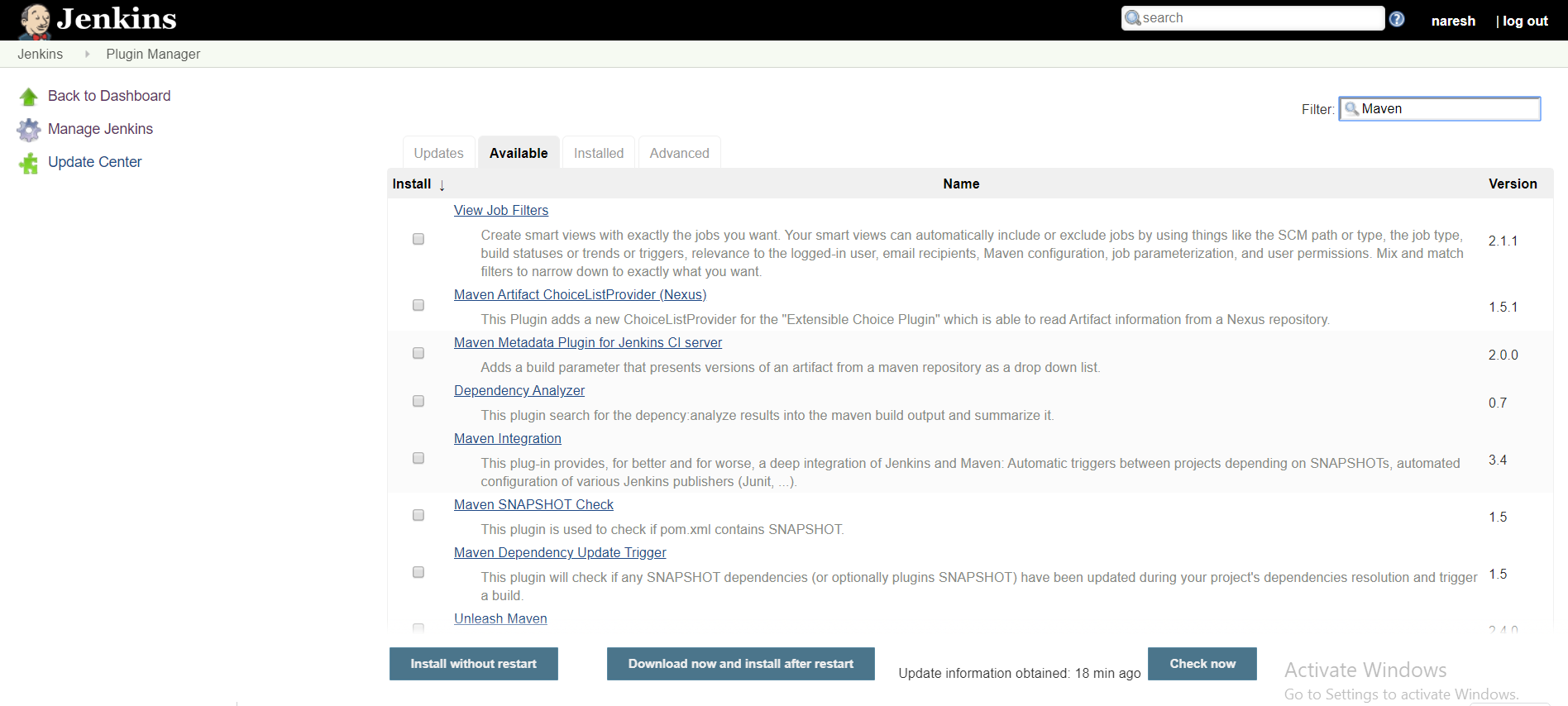
Click on Manage Jenkins



Click on Manage Plugins



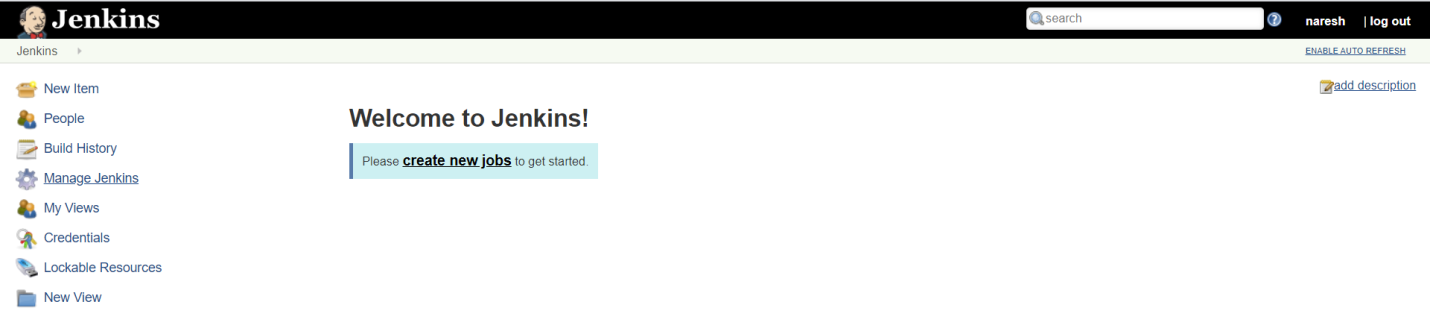
Click on Available and Search Required Pluggins

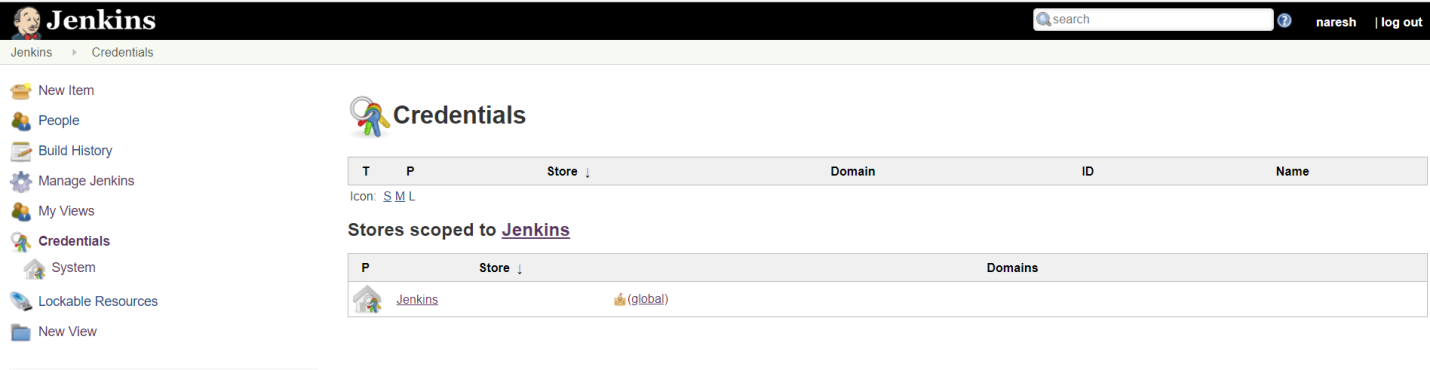


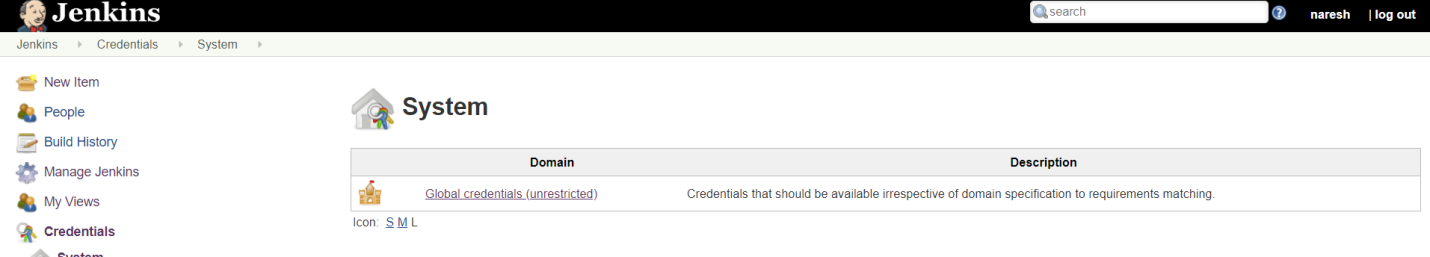
**Select below Plugins:**

* Maven Integration
* S3 publisher
* Pipeline: AWS steps
* SSH
* SSH Agent
* Kubernetes plugin
* Kubernetes CLI Plugin
* Kubernetes Credentials Provider
* Kubernetes Continuous Deploy Plugin
* Kubernetes Credentials Provider

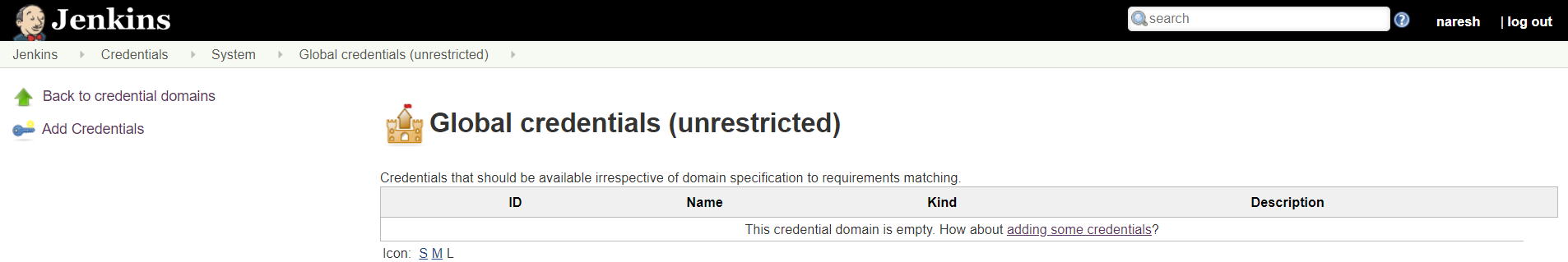
**Step 2:** Create Credentials with in Jenkins

Click on Credentials

Click on Jenkins

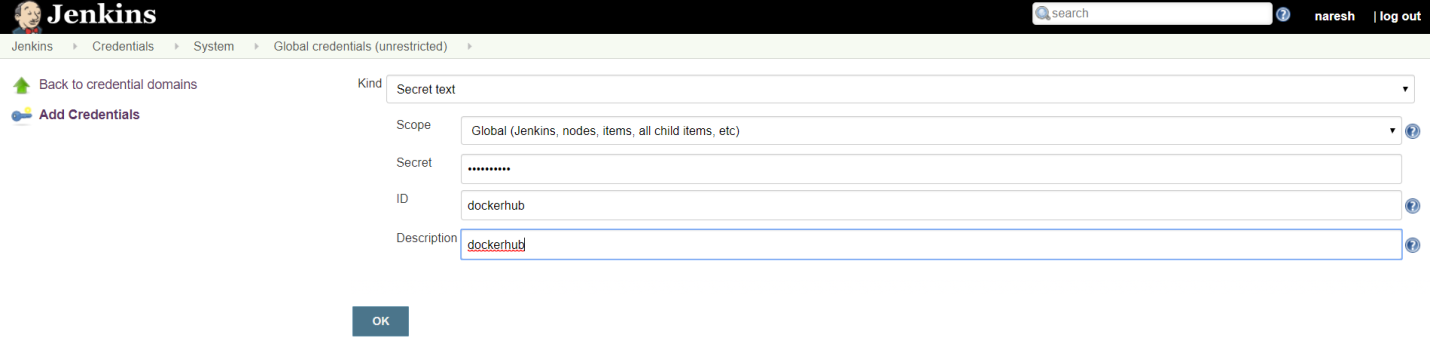


Click on Global credentials(unrestricted)



Click on Add Credentials

1. Create dockerhub credentials



Click on OK

1. Create AWS Credentials:

For these we get Username and Password from IAM User



Give Details like as above

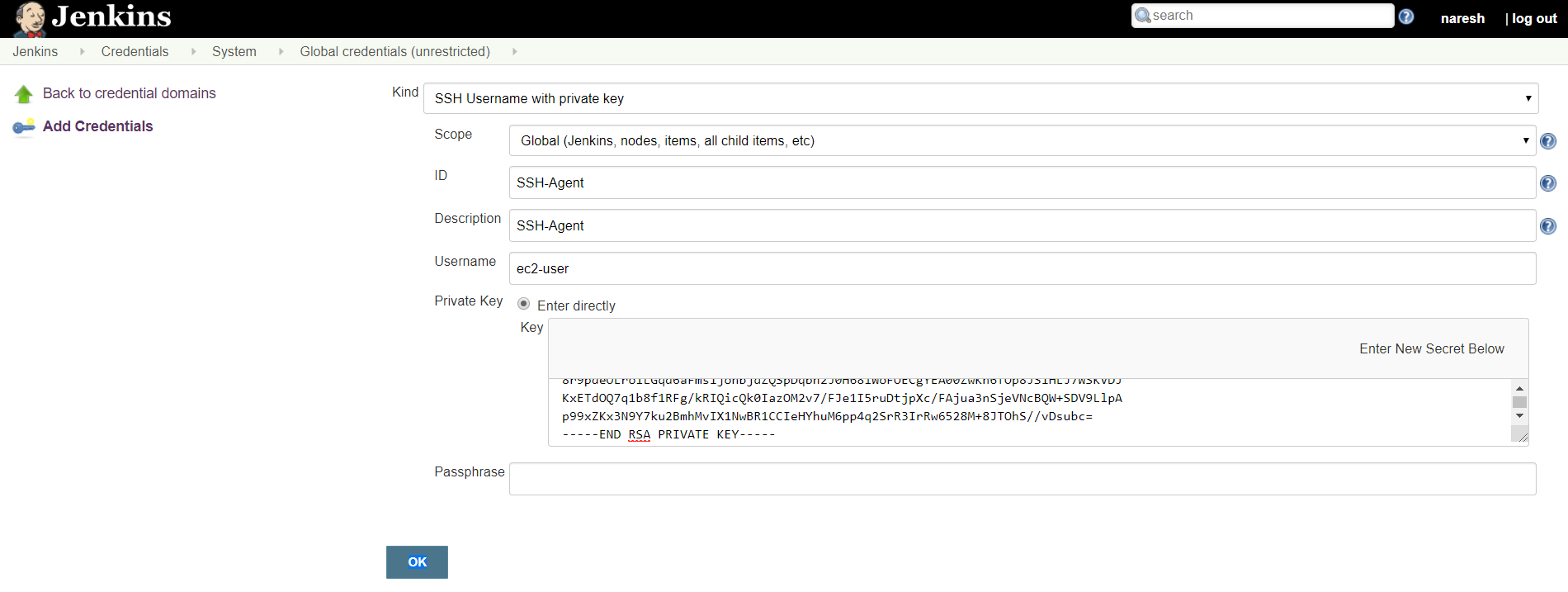
Click on ok

1. Create Credentials for SSH-Agent:

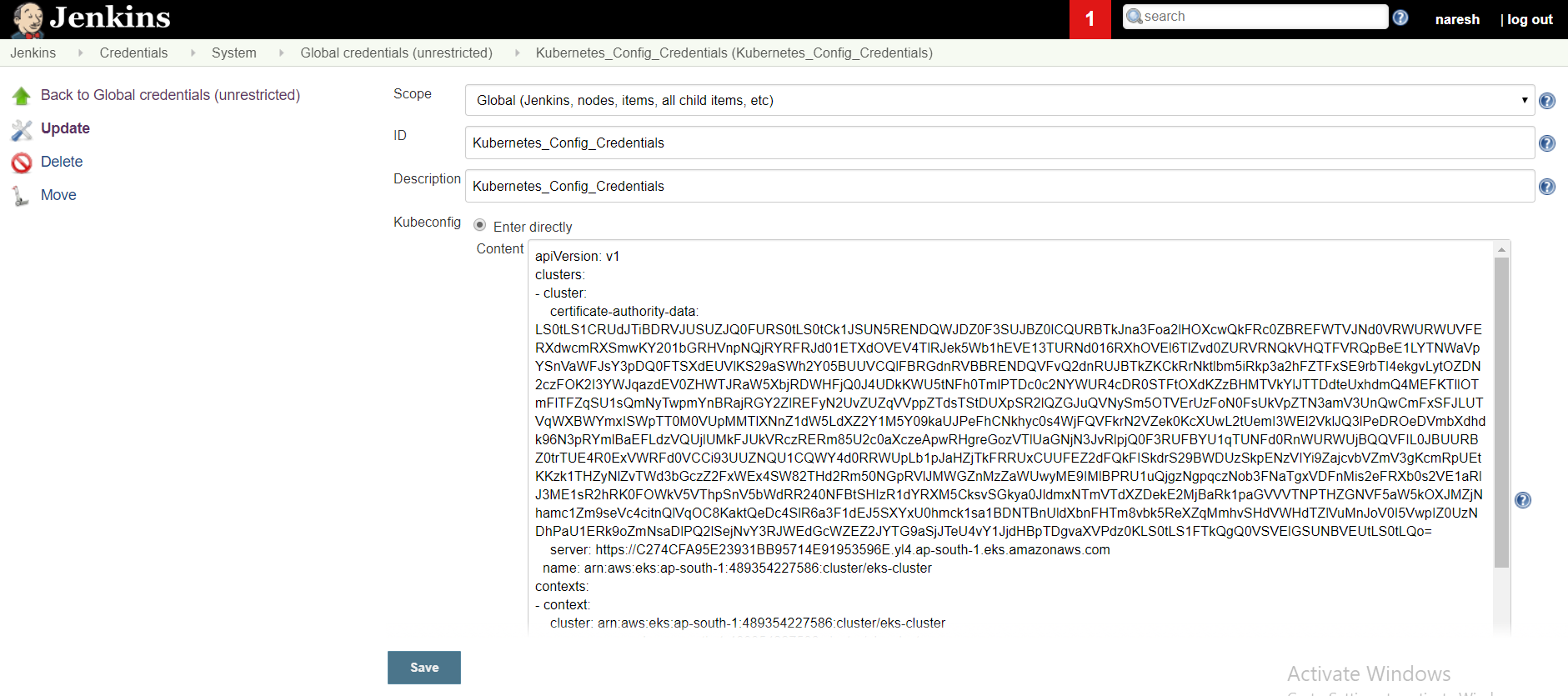
Credentials 🡪 Jenkins 🡪 Global credentials (unrestricted)

Username: ec2-user

Private Key: Pem file data



1. Create Credentials for KubeConfig:



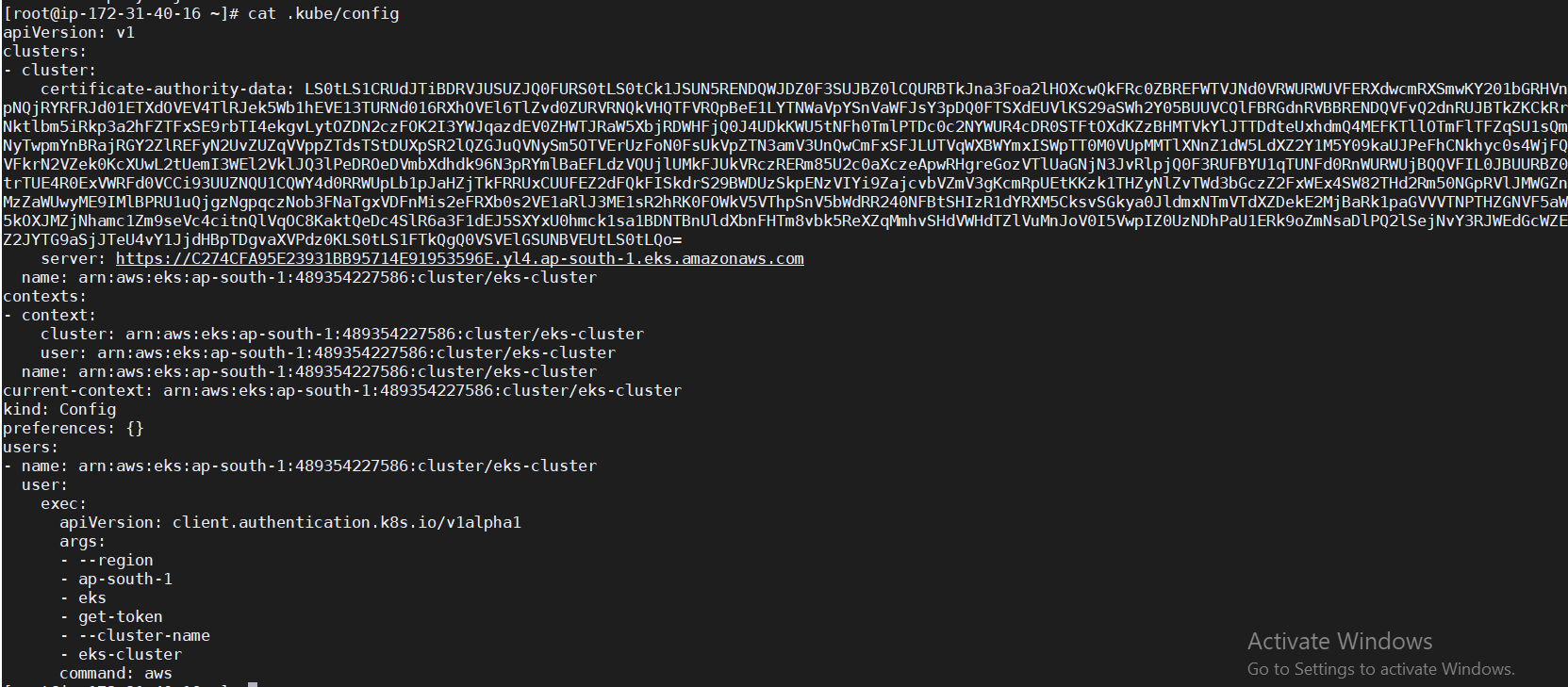
Give Details like as above

Click on ok

For above Kubeconfig data:

Goto Kubernetes Cluster and give command

cat .kube/config



**Step 3:** Integrate Maven with Jenkins

Manage Jenkins 🡪 Global Tool Configuration 🡪 Add Maven

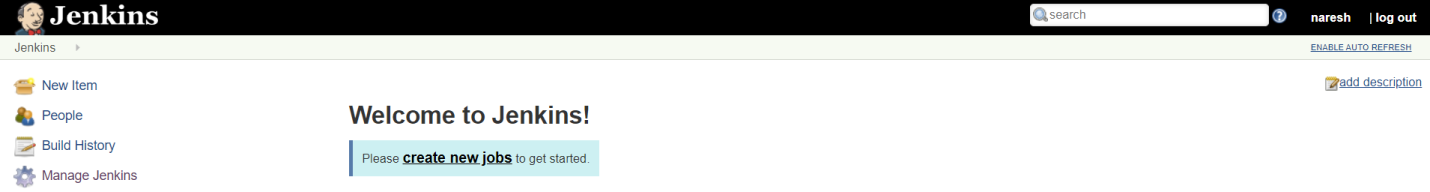


Give details where we installed maven in our machine

Click on Save

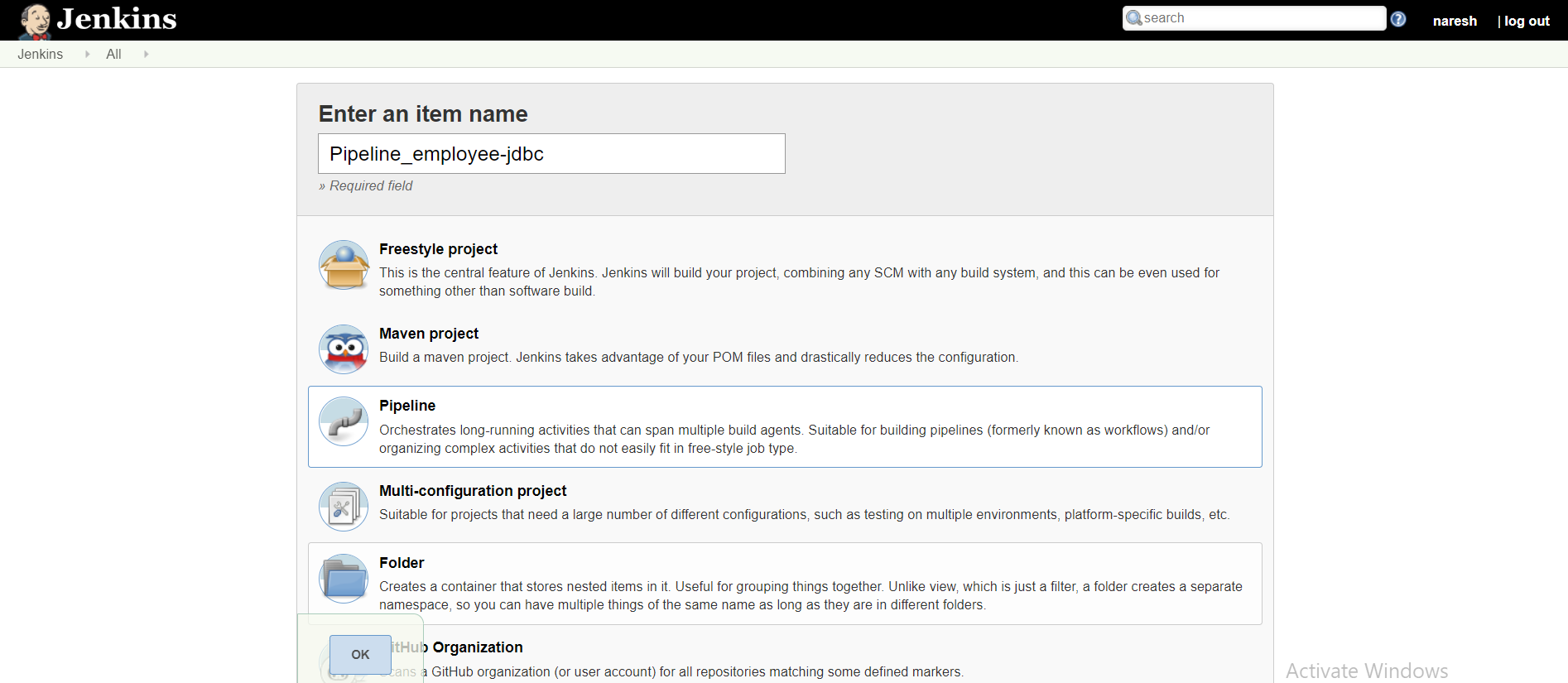
**Step 4:**

Create new Job in Jenkins

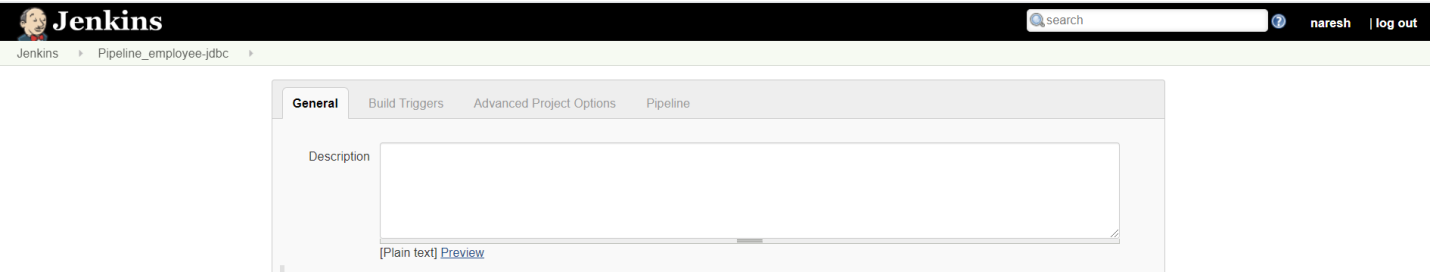


Click on **New Item**

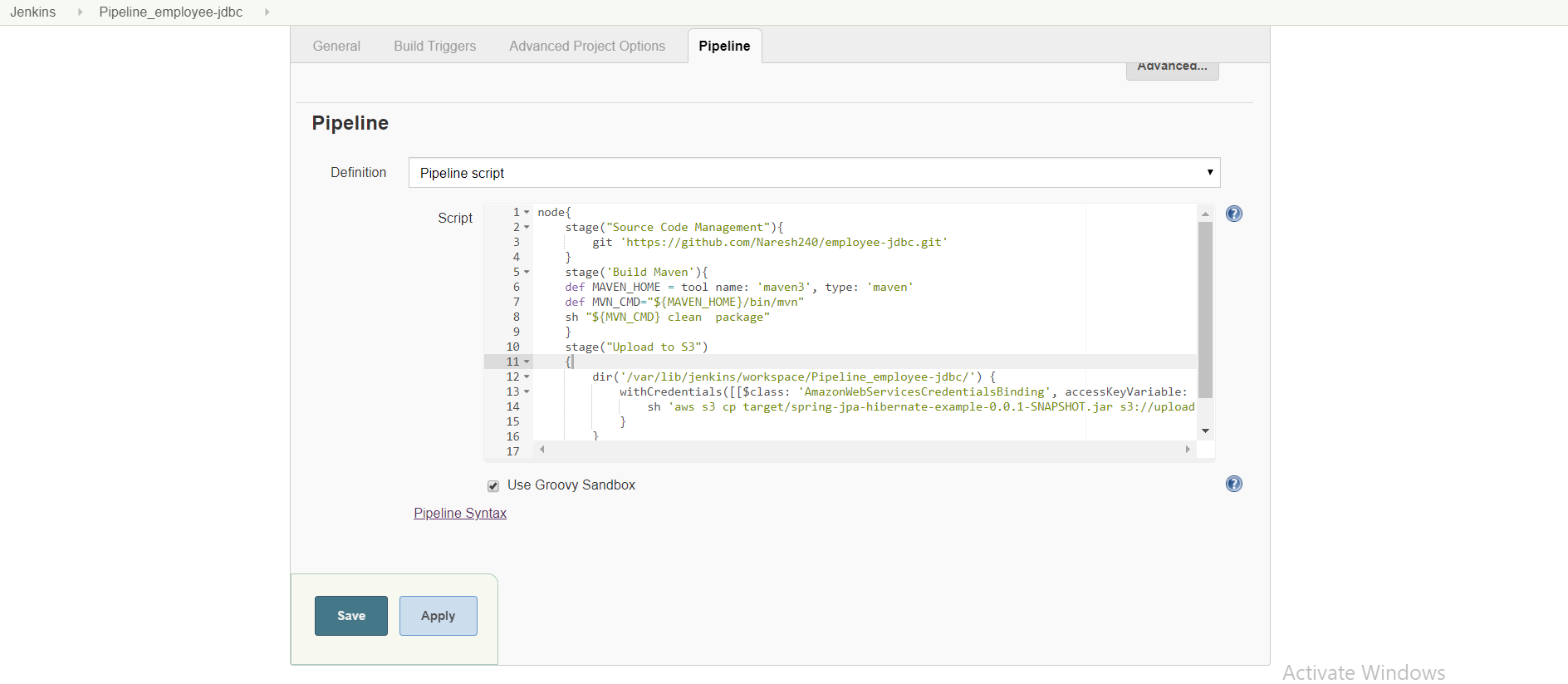
Enter Item Name and Select Pipeline



Click on ok



Click on Pipeline



Click on Save

**Pipeline Script:**

node{

stage("Source Code Management"){

git 'https://github.com/Naresh240/employee-jdbc.git'

}

stage('Build Maven'){

def MAVEN\_HOME = tool name: 'maven3', type: 'maven'

def MVN\_CMD="${MAVEN\_HOME}/bin/mvn"

sh "${MVN\_CMD} clean package"

}

stage("Upload to S3")

{

dir('/var/lib/jenkins/workspace/Pipeline\_employee-jdbc/') {

withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', accessKeyVariable: 'AWS\_ACCESS\_KEY\_ID', credentialsId: 'AWS\_Credentials', secretKeyVariable: 'AWS\_SECRET\_ACCESS\_KEY']]) {

sh 'aws s3 cp target/spring-jpa-hibernate-example-0.0.1-SNAPSHOT.jar s3://upload-artifacts-pipeline'

}

}

}

stage("Build Docker Image"){

sh 'docker build -t naresh240/employee-jdbc .'

}

stage('Push to Docker Hub'){

withCredentials([string(credentialsId: 'dockerhub', variable: 'dockerpwd')]) {

sh 'docker login -u naresh240 -p ${dockerpwd}'

}

sh 'docker push naresh240/employee-jdbc'

}

stage("Deploy with EKS"){

withCredentials([kubeconfigFile(credentialsId: 'Kubernetes\_Config\_Credentials', variable: 'KUBECONFIG')]) {

withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', accessKeyVariable: 'AWS\_ACCESS\_KEY\_ID', credentialsId: 'AWS\_Credentials', secretKeyVariable: 'AWS\_SECRET\_ACCESS\_KEY']]) {

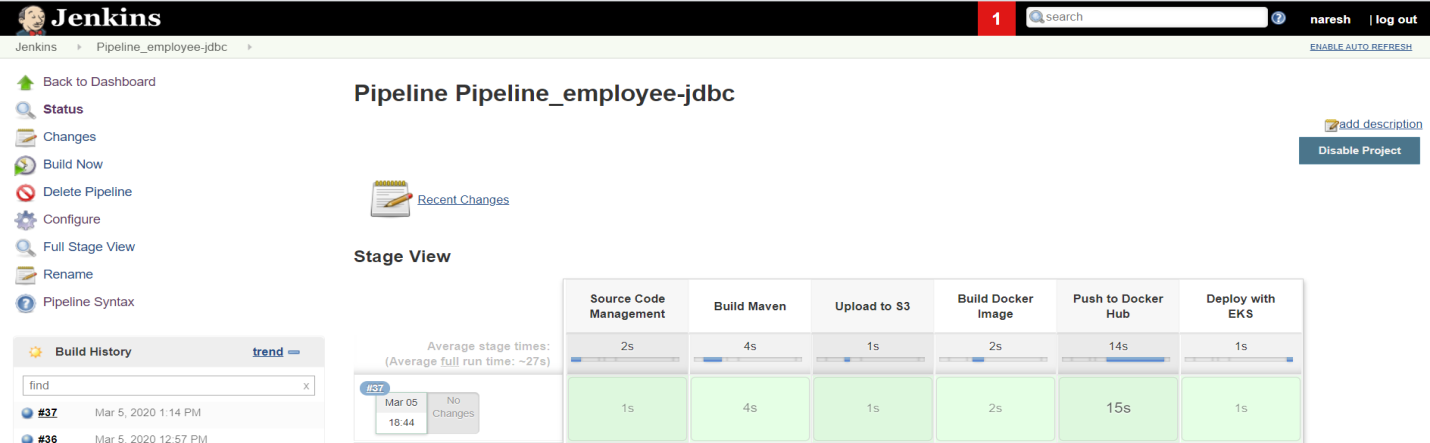
sh 'kubectl apply -f empjdbc.yml'

}

}

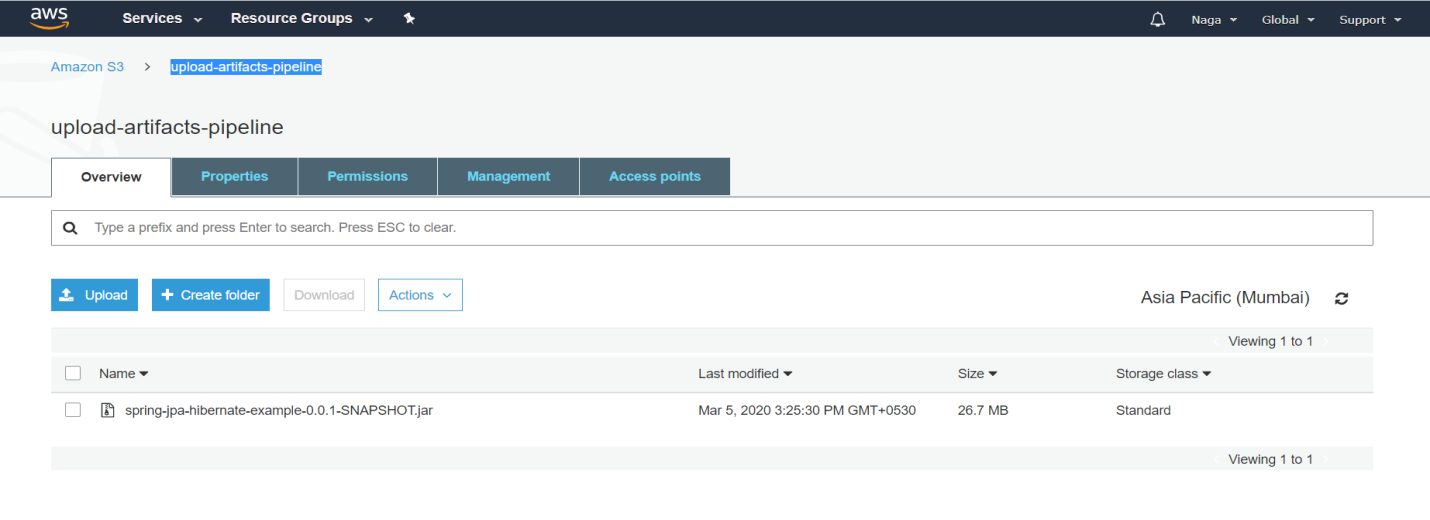
}

}



Click on build

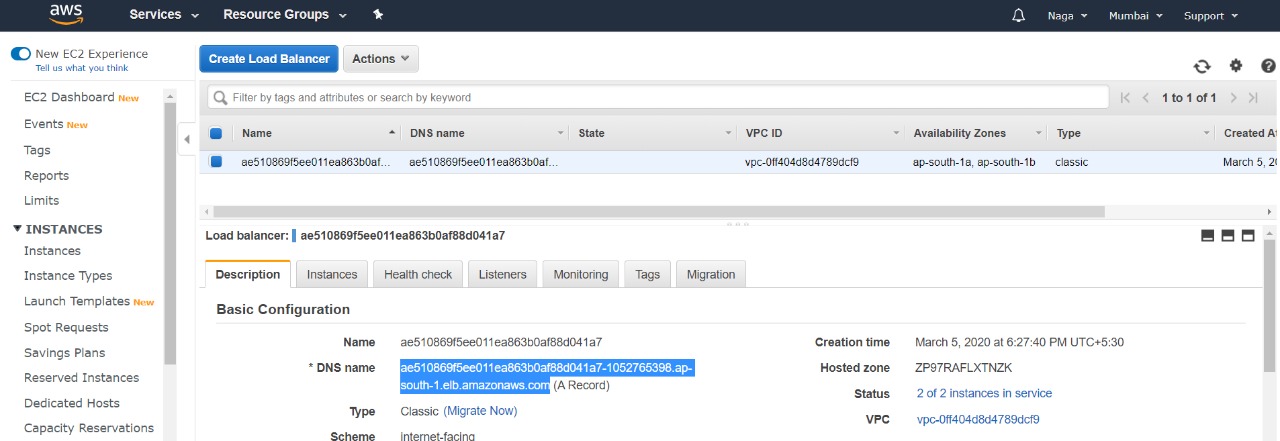
Check whether Artifact uploaded to S3 Bucket or not:



Check whether the pods created in our second Server or not:

kubectl get pods

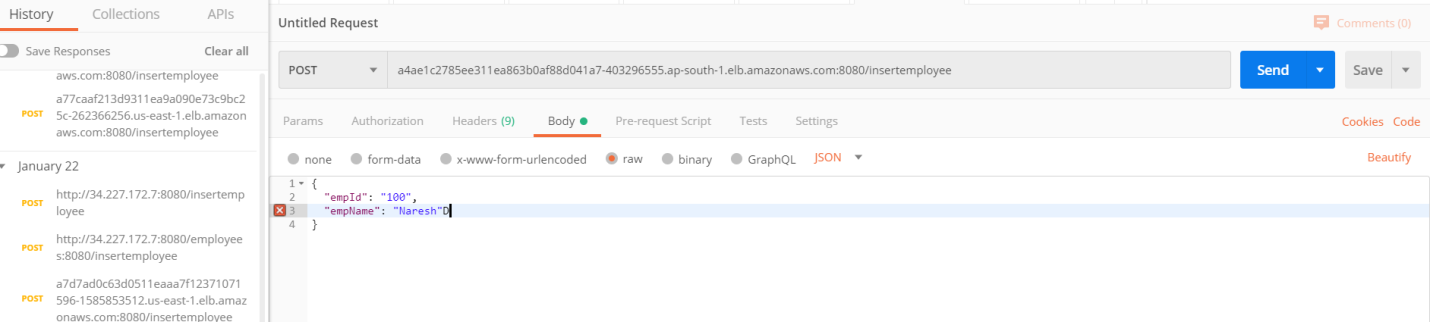
Check whether the LoadBalancer created for this deployment or not:



Insert data into our application by using Postman app:

Copy DNS Name from LoadBalancer once instaces comes InService.

a4ae1c2785ee311ea863b0af88d041a7-403296555.ap-south-1.elb.amazonaws.com:8080/insertemployee



Click on Send Button

Check whether values added in mysql or not:

<http://a4ae1c2785ee311ea863b0af88d041a7-403296555.ap-south-1.elb.amazonaws.com:8080/employees>

