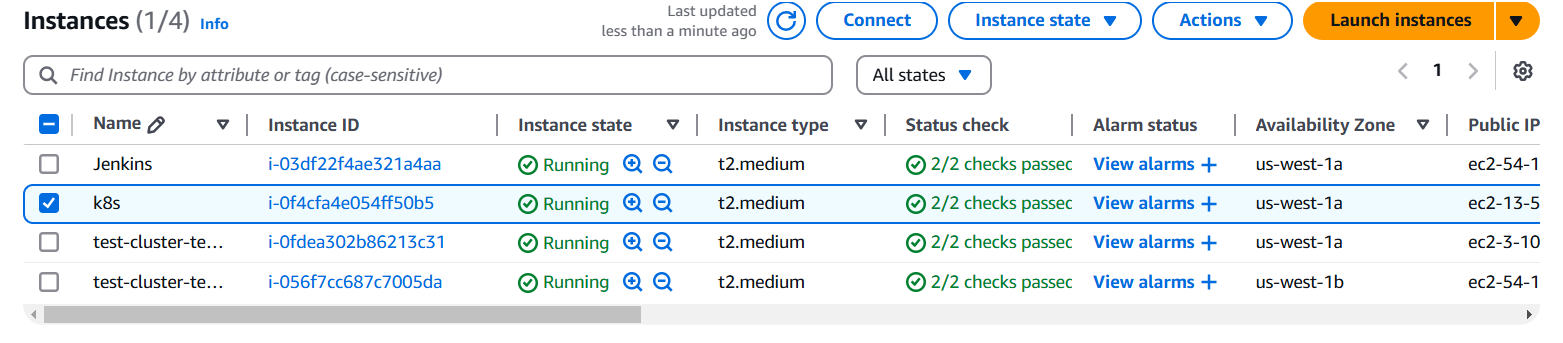
# **How to connect eks cluster to Jenkins**

**Eks cluster**

* Create a linux server with ubuntu os for setting up eks cluster



* Connect your ec2 instance with putty or mobaxterm using shh client
* After that you have to install following cli tools

1. **Eksctl**

**Description: -** eksctl is a simple CLI tool for creating and managing clusters on EKS - Amazon's managed Kubernetes service for EC2. It is written in Go, uses CloudFormation

**Command:**  # for ARM systems, set ARCH to: `arm64`, `armv6` or `armv7`

ARCH=amd64

PLATFORM=$(uname -s)\_$ARCH

curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl\_$PLATFORM.tar.gz"

# (Optional) Verify checksum

curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl\_checksums.txt" | grep $PLATFORM | sha256sum --check

tar -xzf eksctl\_$PLATFORM.tar.gz -C /tmp && rm eksctl\_$PLATFORM.tar.gz

sudo mv /tmp/eksctl /usr/local/bin

1. **Kubectl**

**Description: -** kubectl is the command-line tool for interacting with Kubernetes clusters. It allows users to manage Kubernetes resources and perform various tasks such as deploying applications, inspecting and modifying cluster resources, and troubleshooting cluster issues.

**Command: -**

* Install kubectl binary with curl on Linux

curl -LO [https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)

* Validate the binary (optional)

Download the kubectl checksum file:

curl -LO [https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl.sha256](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256)

* Validate the kubectl binary against the checksum file:

echo "**$(**cat kubectl.sha256**)** kubectl" | sha256sum –check

* Install kubectl

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

chmod +x kubectl

mkdir -p ~/.local/bin

mv ./kubectl ~/.local/bin/kubectl

* Test to ensure the version you installed is up-to-date:

kubectl version --client

1. **AWS CLI**

**Description: -** The **AWS Command Line Interface (CLI)** is a tool provided by Amazon Web Services that enables users to interact with AWS services from a terminal or command prompt. It allows you to perform various AWS tasks directly from the command line, including managing resources, configuring settings, and automating workflows.

**Command: -**

apt install unzip -y

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

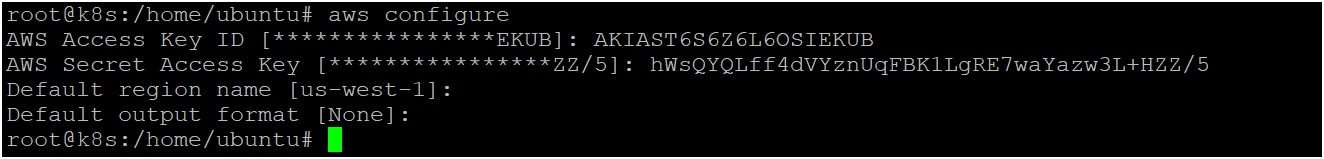
sudo ./aws/install

**Confirm the installation with the following command: -**

**aws –version**

1. **AWS access key and secret key** you have to provide before create one IAM user with administrator access permission then generate access key and secret key and you configure these keys on your terminal

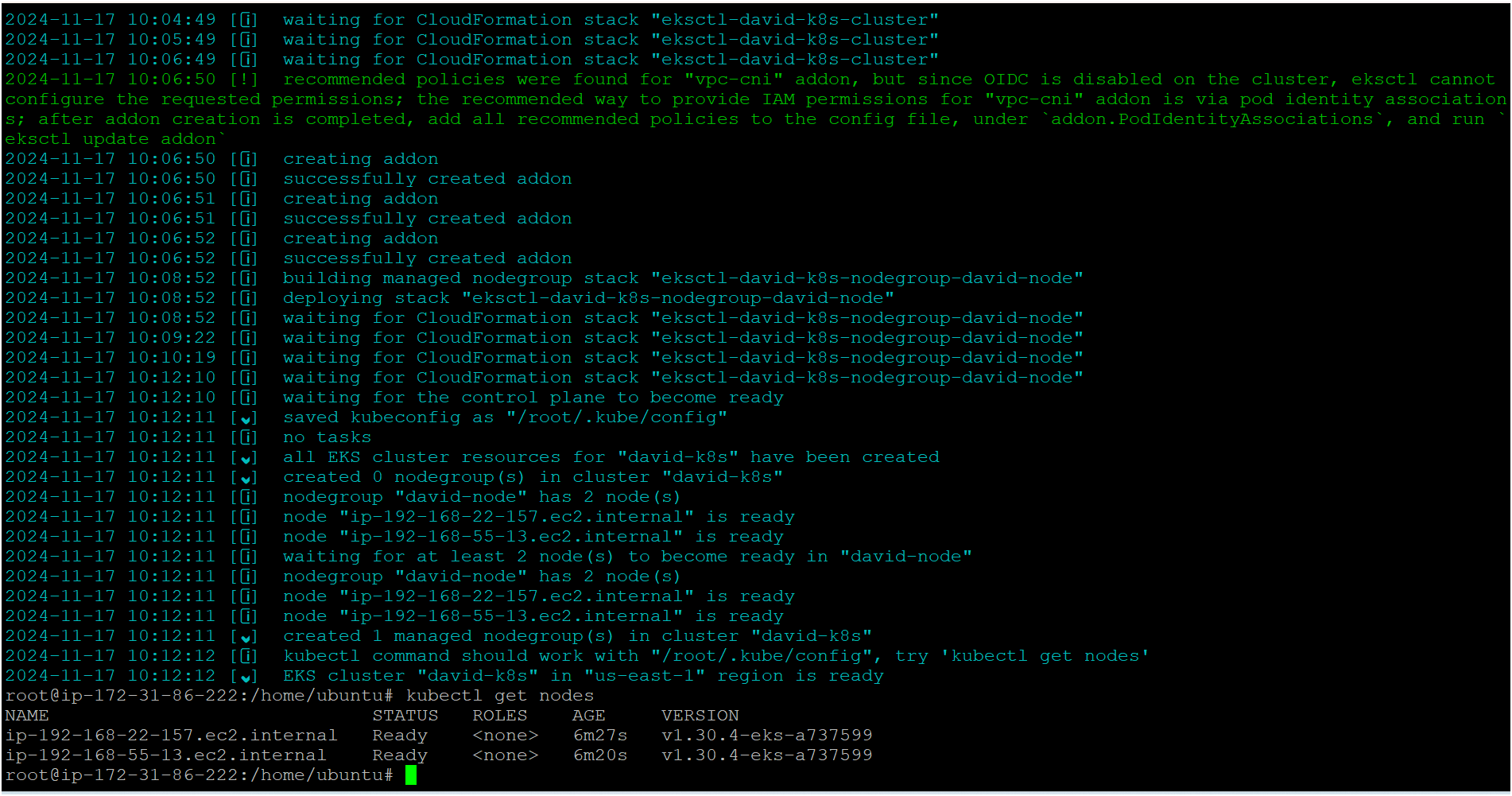
aws configure (execute this command on your terminal)



* Now you can create eks cluster by executing following command

eksctl create cluster --name <cluster\_name> --region <your\_region> --zones <availability\_zones> --nodegroup-name <nodegroip\_name> --node-type <instance\_type> --nodes <no.of.nodes>

* Now your cluster is ready as shown below image 👇



**Jenkins**

* Connect your ec2 instance with putty or mobaxterm using shh client
* Please make sure you have to install kubectl and aws cli on Jenkins server (you can get commands above where I mentioned for k8s cluster)
* Now install Jenkins on your terminal by executing following commands 👇

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins

* Install java on your terminal where are installing Jenkins becz java is prerequisite for Jenkins

sudo apt update

sudo apt install fontconfig openjdk-17-jre

java -version

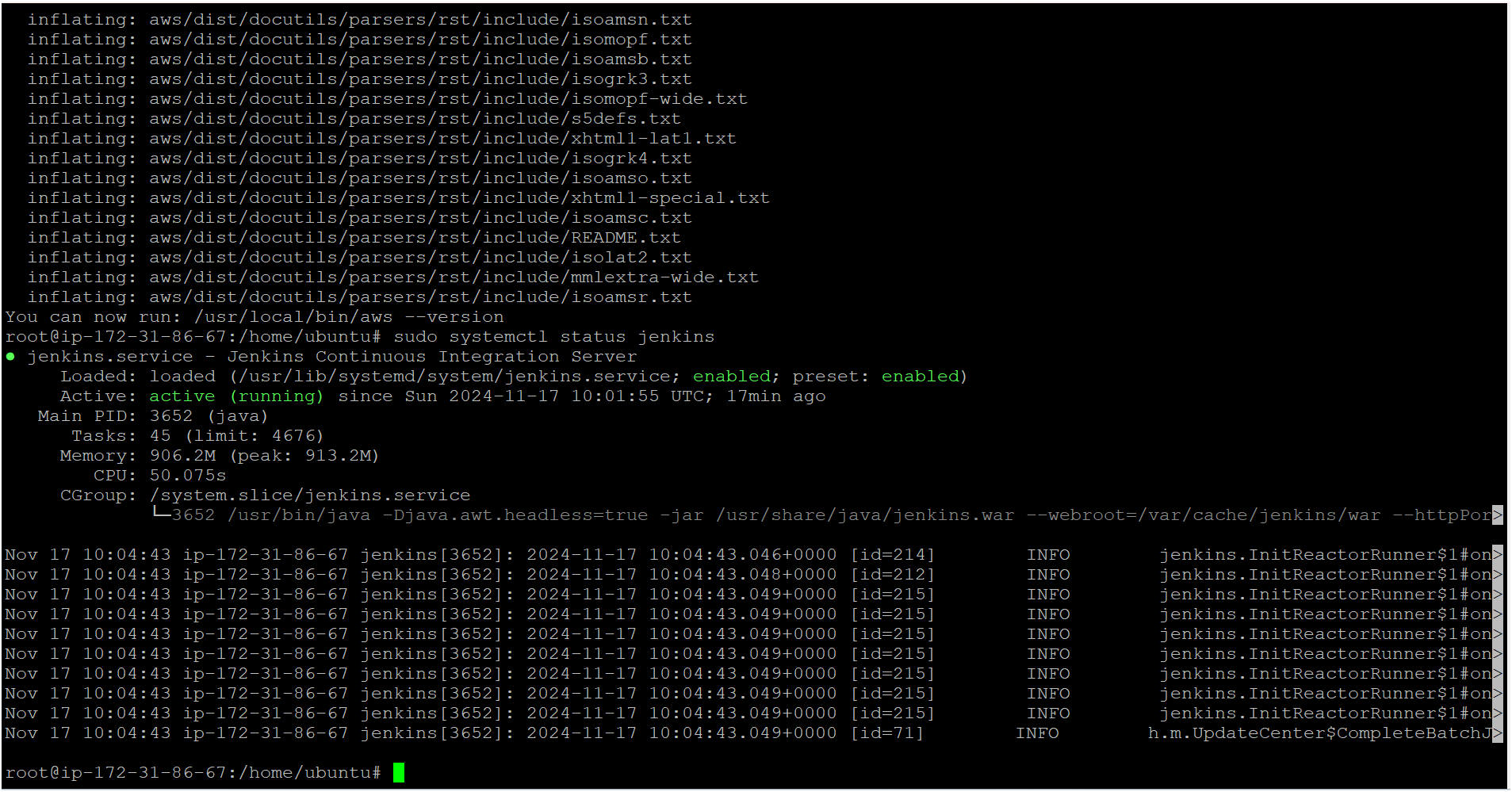
* Now you have to start your Jenkins by following commands 👇

sudo systemctl enable jenkins

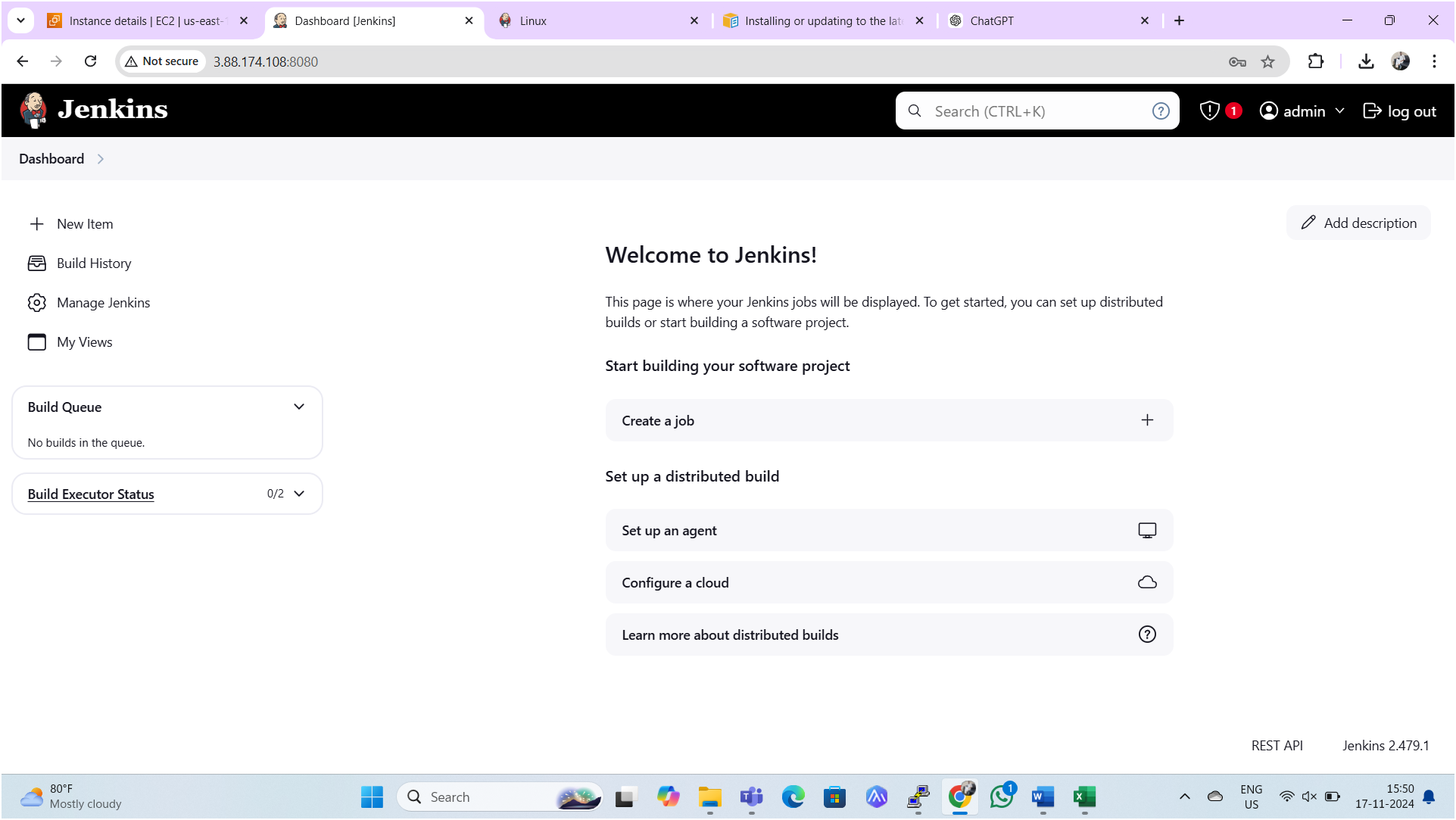
sudo systemctl start Jenkins

sudo systemctl status Jenkins

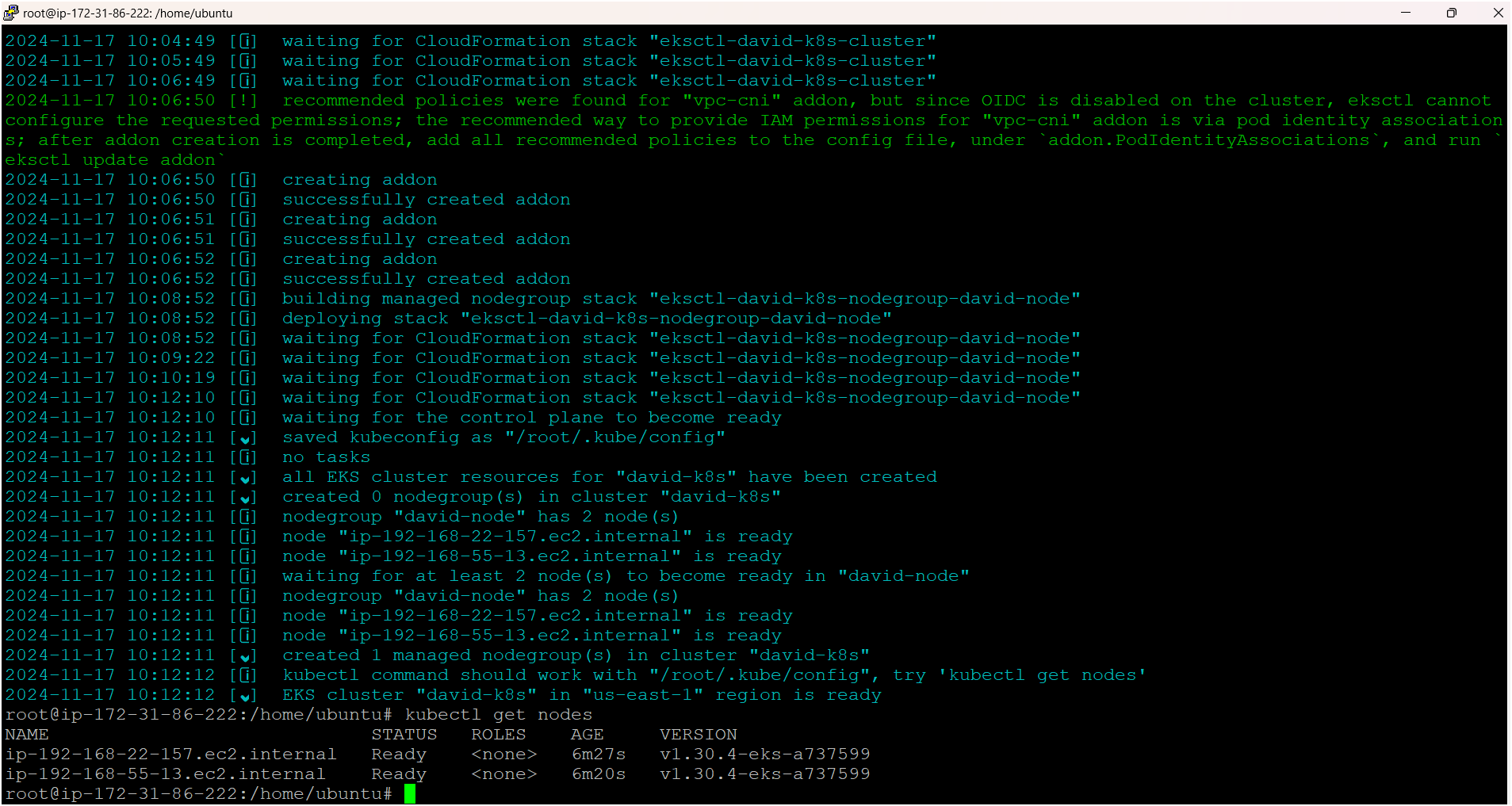
* Now your Jenkins server is ready



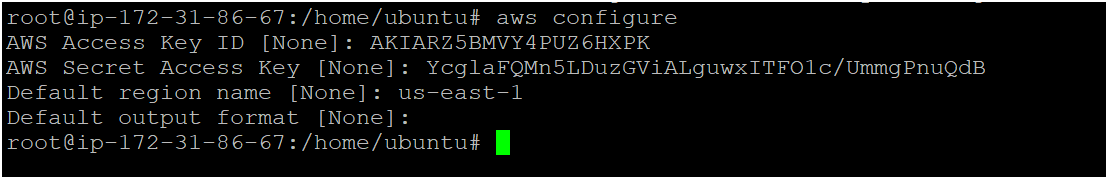
* Access your Jenkins server on browser using instance public ip address with port number as shown below image 👇
* Open port number 8080 in your security group for Jenkins



* Next, we need to copy Kube config file into the Jenkins server from k8s master node as shown below image 👇

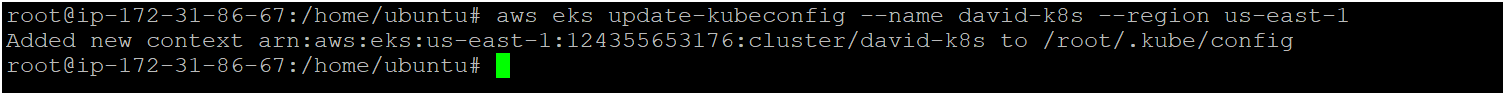


* Firstly, we have to provide aws access keys on Jenkins server



* Execute following command on Jenkins server

aws eks update-kubeconfig --name <cluster-name> --region <region>



* As seen above image Kube-config file has copied to Jenkins serve from master node i.e. cluster.
* Now execute following commands in Jenkins server one by one

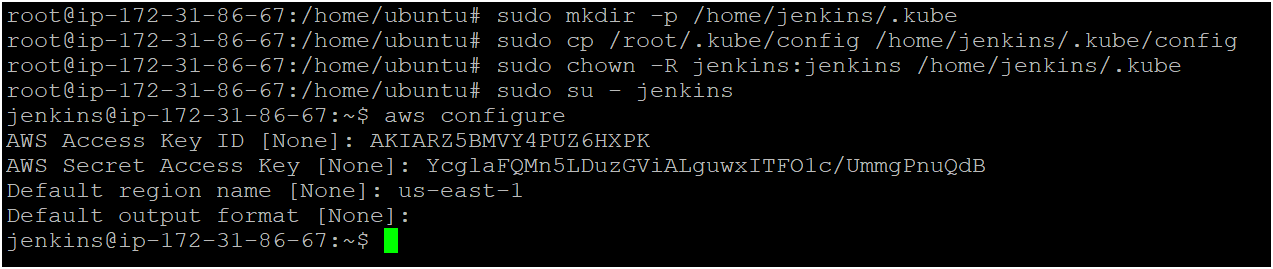
sudo mkdir -p /home/jenkins/.kube

sudo cp /root/.kube/config /home/jenkins/.kube/config

sudo chown -R jenkins:jenkins /home/jenkins/.kube

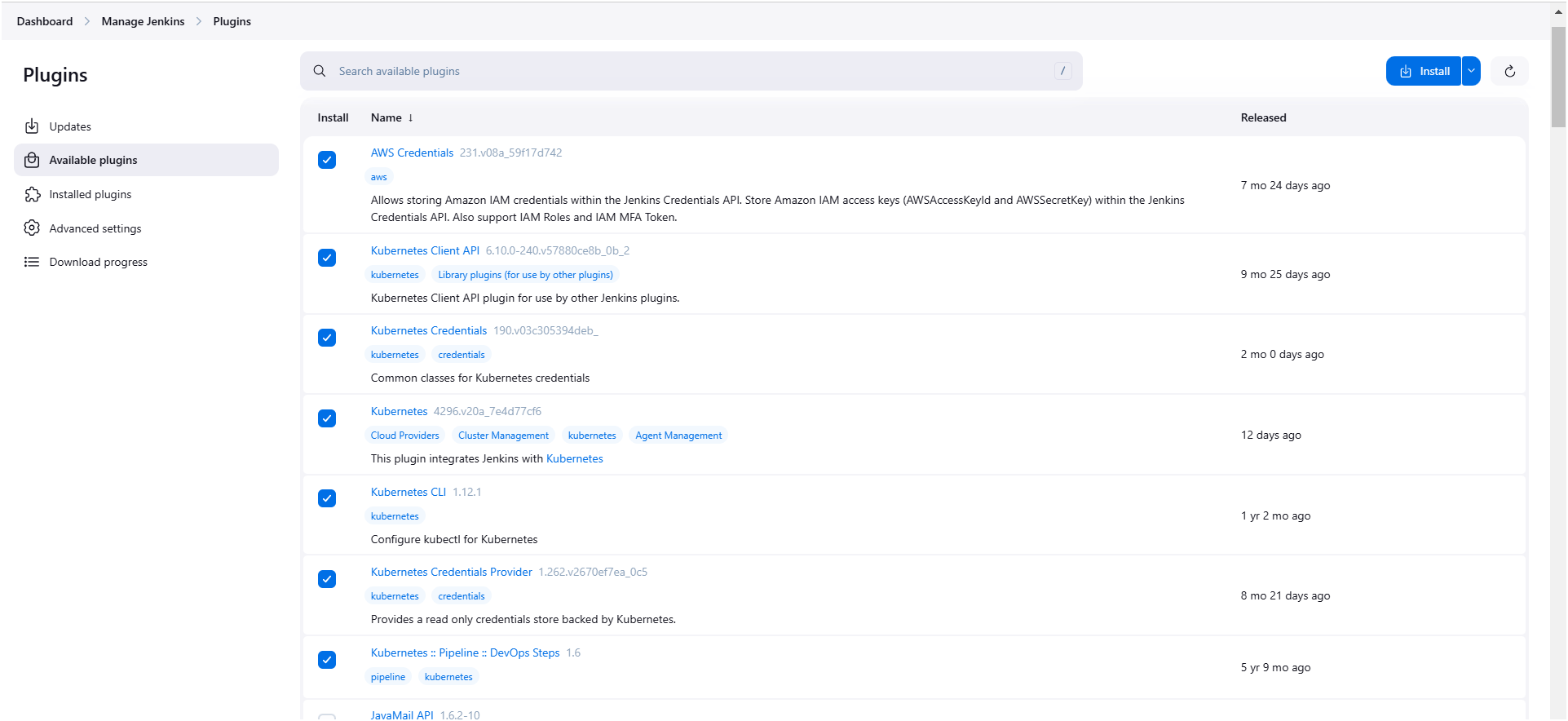
sudo su - jenkins

aws configure (provide access keys on Jenkins server)



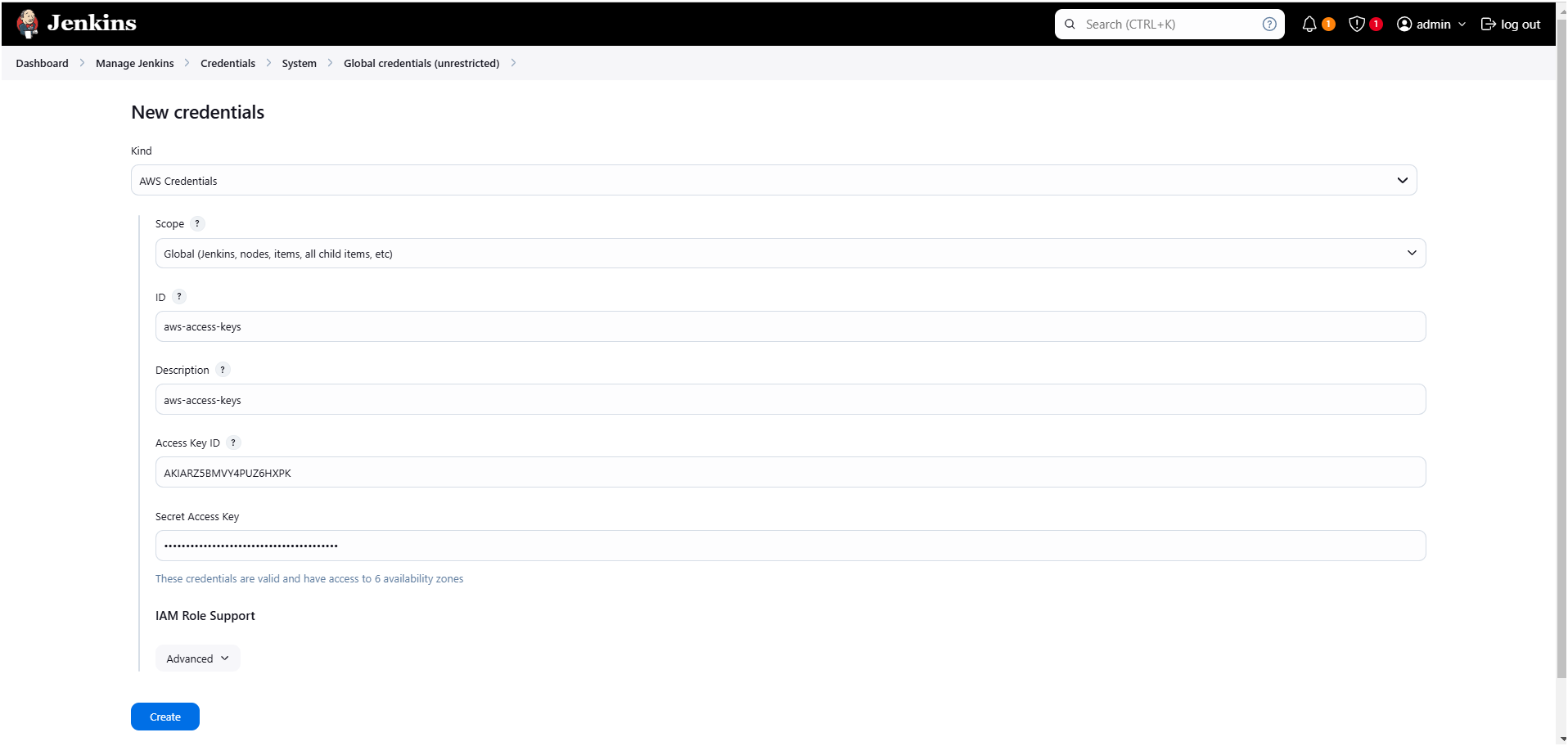
* Install following plugins on Jenkins

1. [AWS Credentials](https://plugins.jenkins.io/aws-credentials)
2. [Kubernetes Client API](https://plugins.jenkins.io/kubernetes-client-api)
3. Kubernetes CredentialsVersion
4. [Kubernetes](https://plugins.jenkins.io/kubernetes)
5. [Kubernetes CLI](https://plugins.jenkins.io/kubernetes-cli)
6. [Kubernetes Credentials Provider](https://plugins.jenkins.io/kubernetes-credentials-provider)
7. [Kubernetes :: Pipeline :: DevOps Steps](https://plugins.jenkins.io/kubernetes-pipeline-devops-steps)



* We have to provide aws access keys in Jenkins in credentials

Go to manage Jenkins 🡪 go to credentials 🡪 system 🡪 global credentials



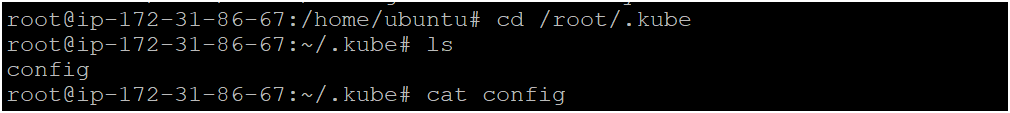
* Now we have to provide Kube-config file in Jenkins web server
* Firstly, we need to copy and save it in a document or file in your local as shown below image
* Execute below two commands on Jenkins server for copying config file

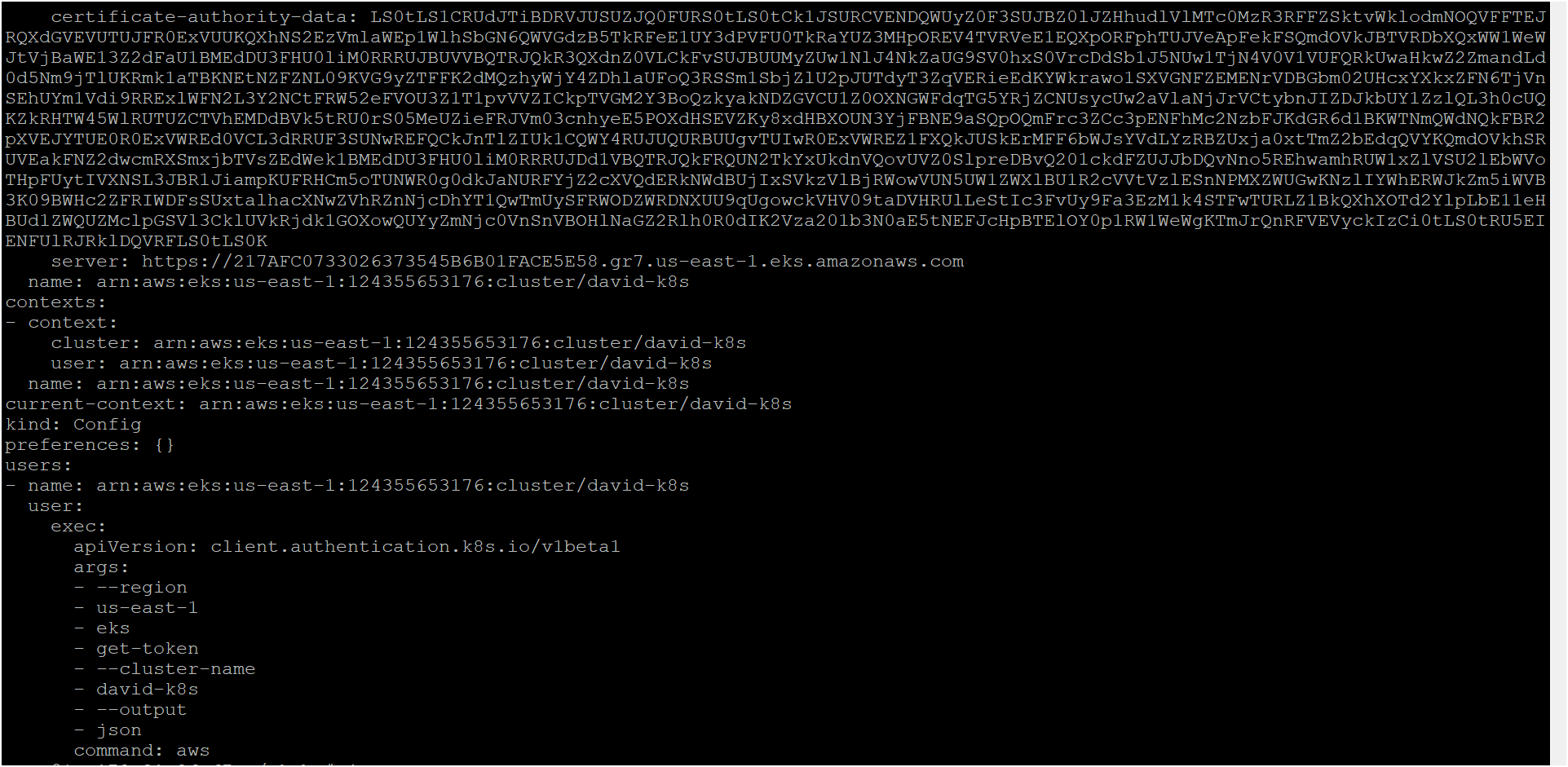
cd /root/.kube

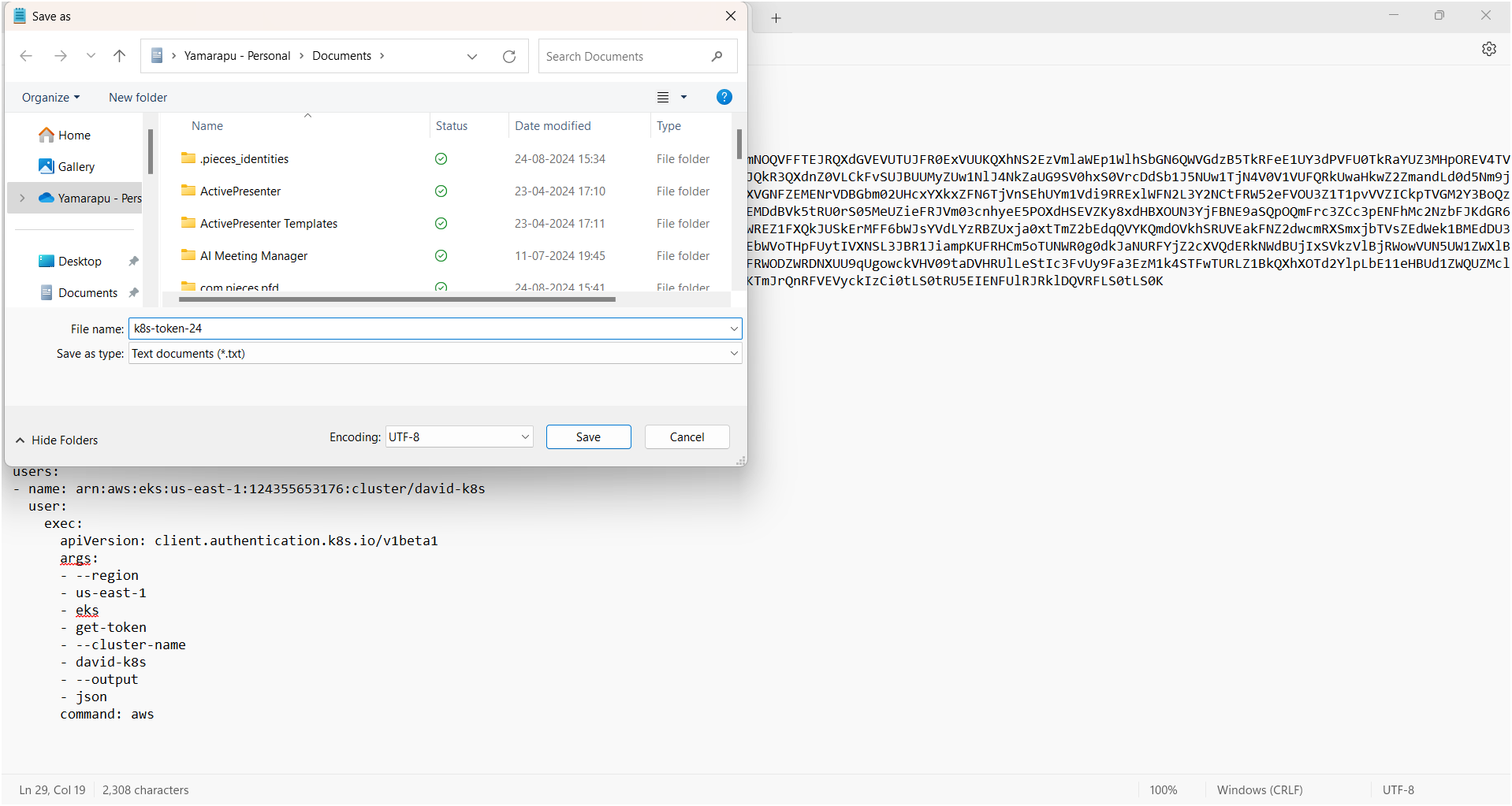
ls

cat config (now copy the entire token and save it in a document or file)

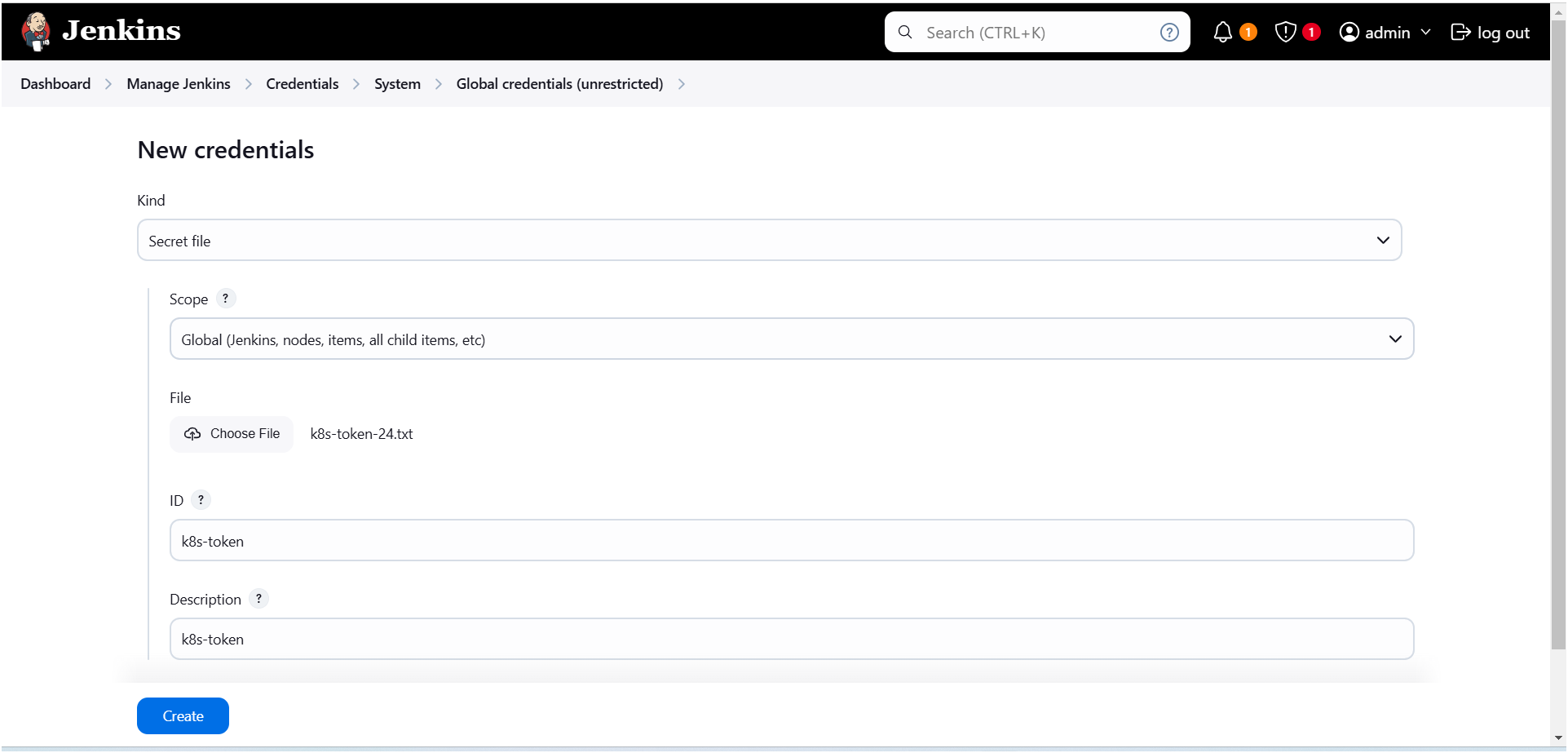
* Please make sure you have to copy this config file from Jenkins server only not from cluster.



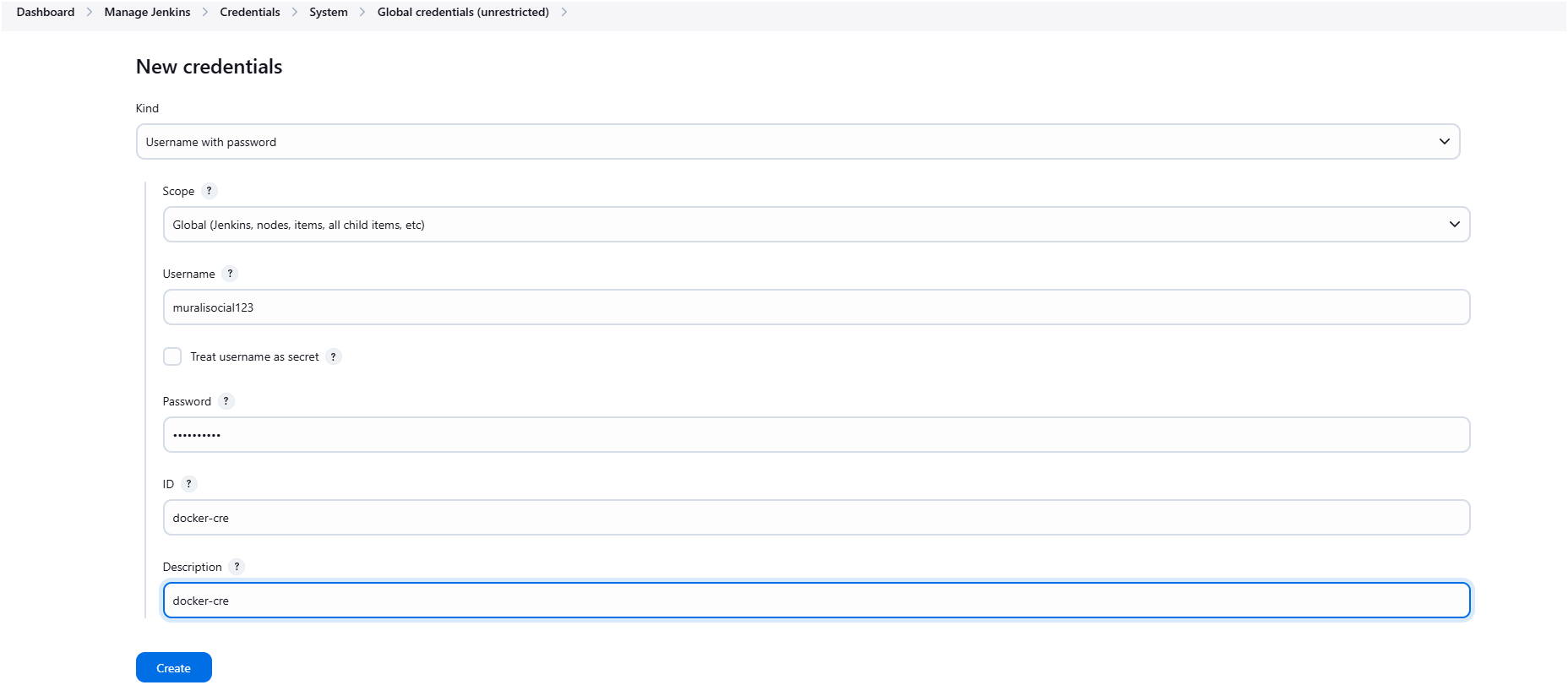




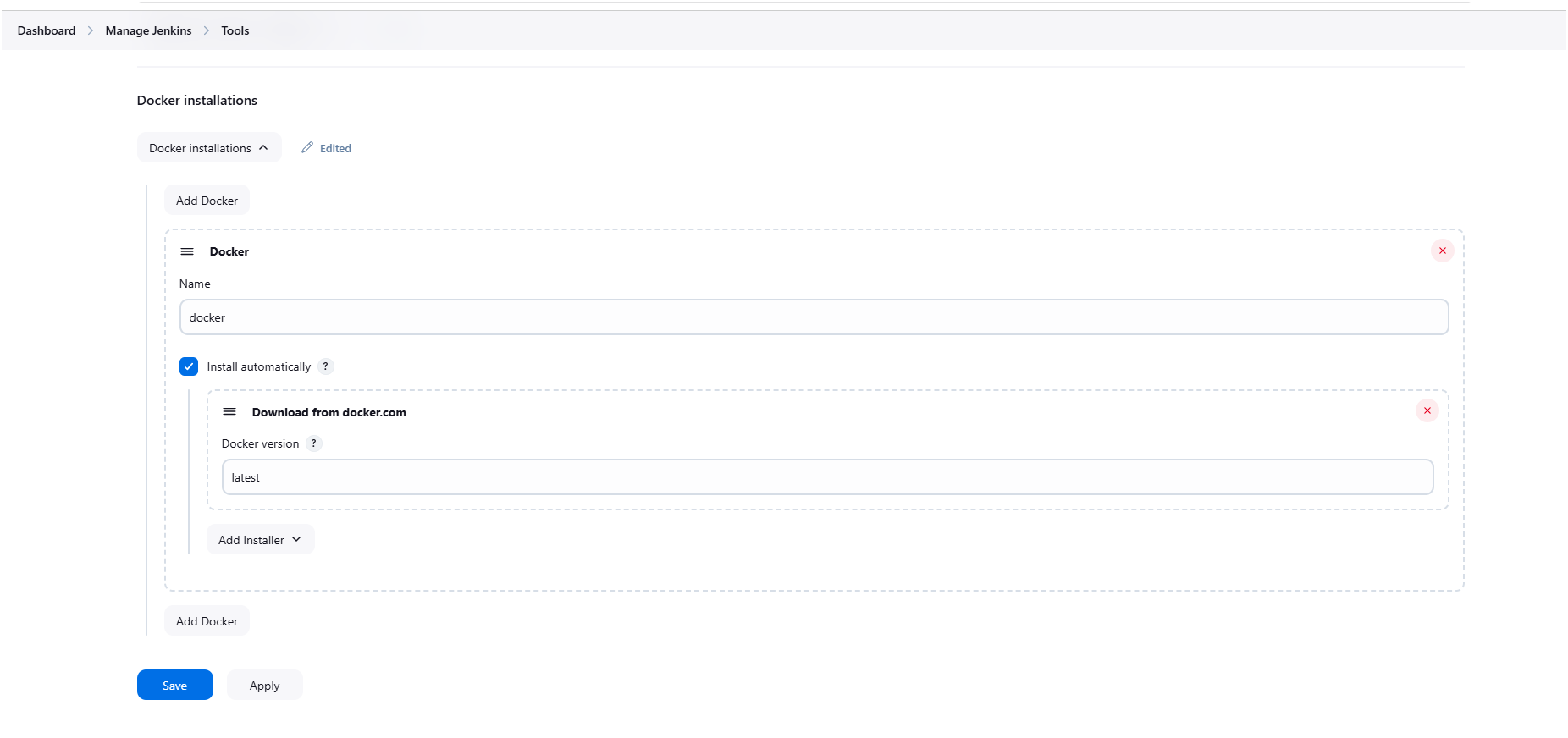
* Now we have to provide this file in Jenkins web server in credentials as follows



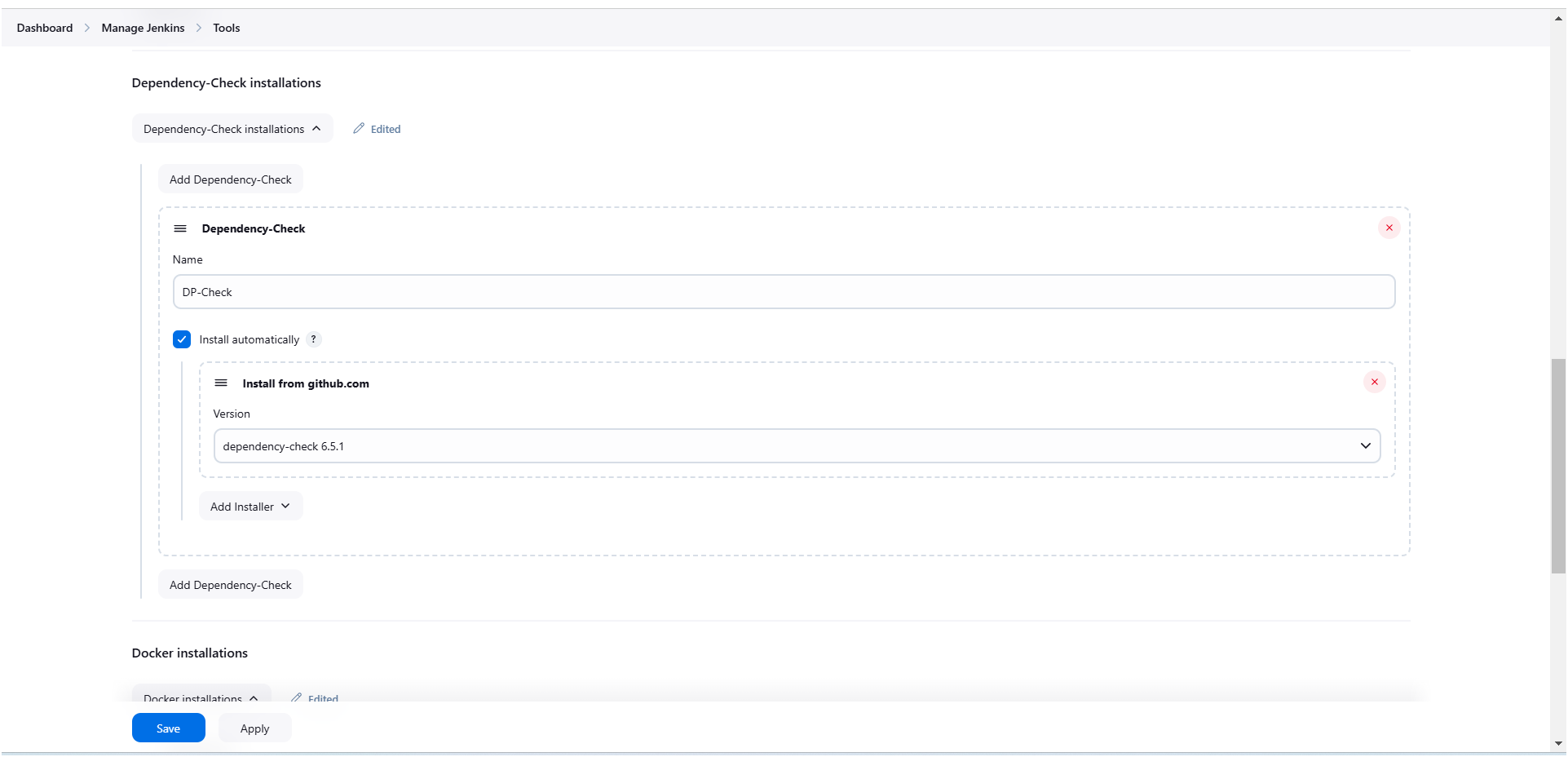
* Provide docker credentials as shown below image 👇



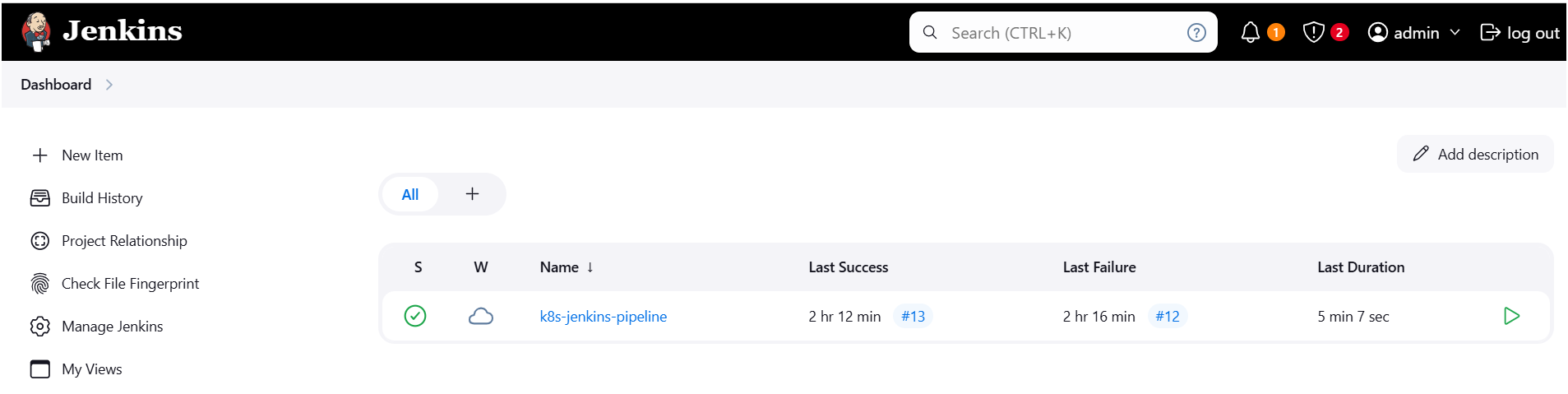
* Configure docker in tools as shown below image 👇



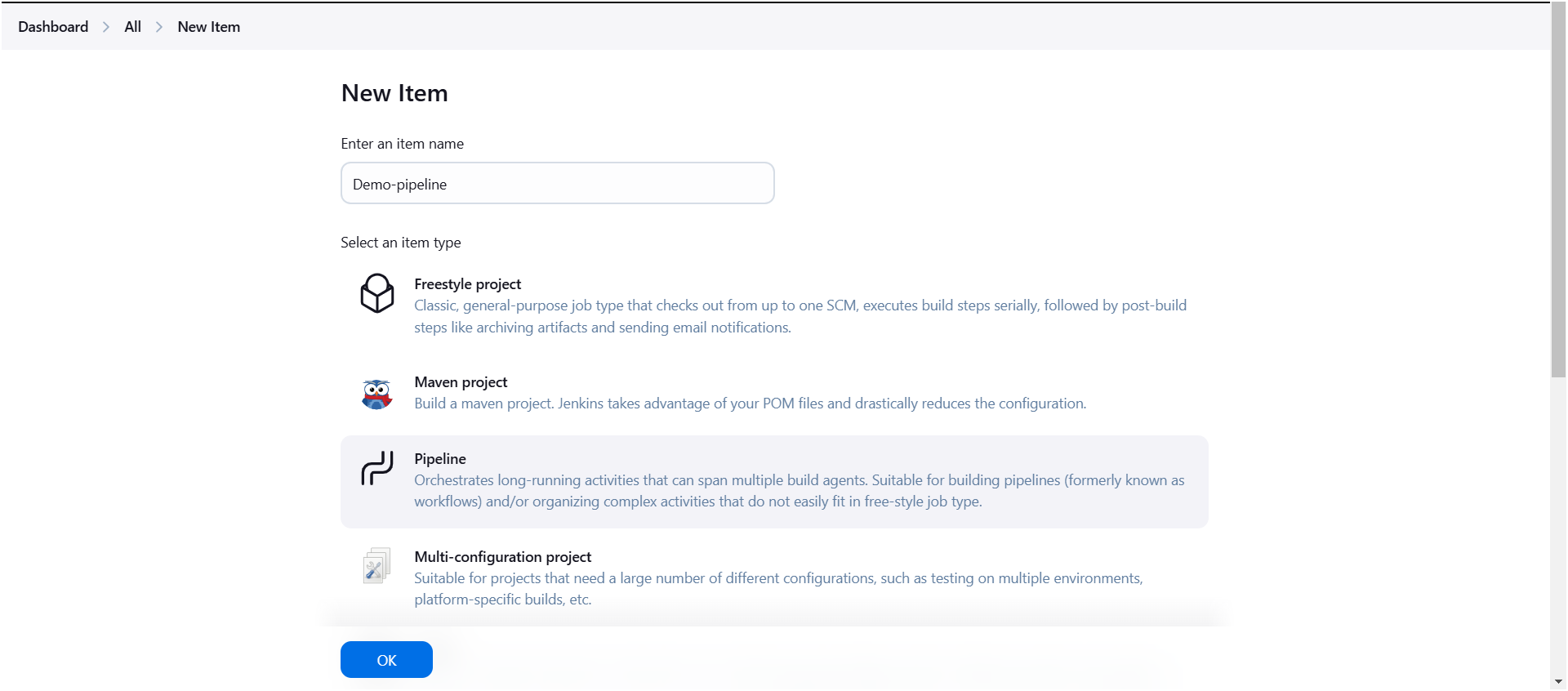
* Configure owasp dependence check as shown below image 👇



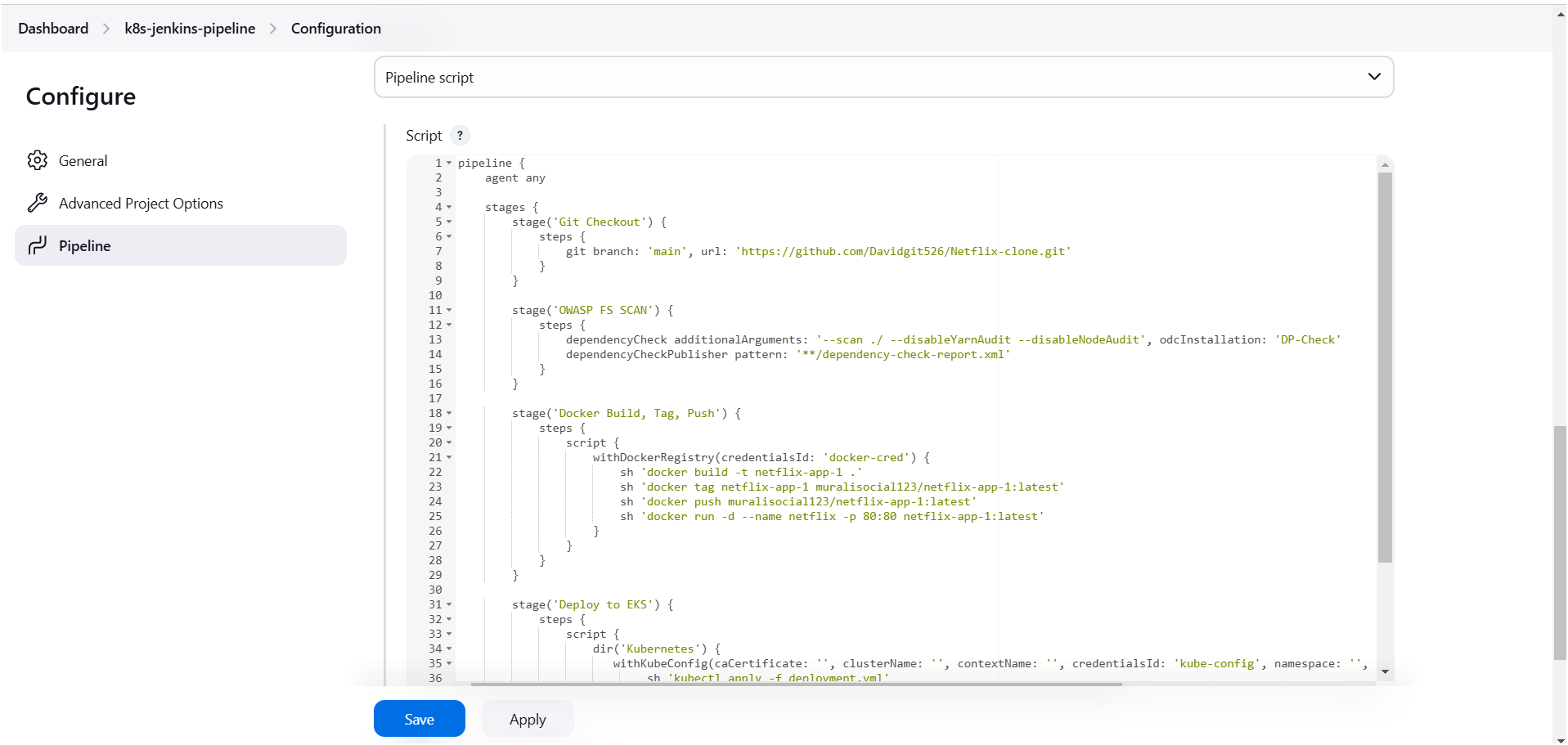
* Create a new item (pipeline) as shown below image 👇



* Click on “New item” for creating pipeline ☝️
* Provide a name for pipeline and select pipeline then click on “ok” 👇



* Now create a pipeline by writing script 👇



**Pipeline Script:**

pipeline {

agent any

stages {

stage('Git Checkout') {

steps {

git branch: 'main', url: 'https://github.com/Davidgit526/Netflix-clone.git'

}

}

stage('OWASP FS SCAN') {

steps {

dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-Check'

dependencyCheckPublisher pattern: '\*\*/dependency-check-report.xml'

}

}

stage('Docker Build, Tag, Push') {

steps {

script {

withDockerRegistry(credentialsId: 'docker-cred') {

sh 'docker build -t netflix-app-1 .'

sh 'docker tag netflix-app-1 muralisocial123/netflix-app-1:latest'

sh 'docker push muralisocial123/netflix-app-1:latest'

sh 'docker run -d --name netflix -p 80:80 netflix-app-1:latest'

}

}

}

}

stage('Deploy to EKS') {

steps {

script {

dir('Kubernetes') {

withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'kube-config', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

sh 'kubectl apply -f deployment.yml'

sh 'kubectl apply -f service.yml'

}

}

}

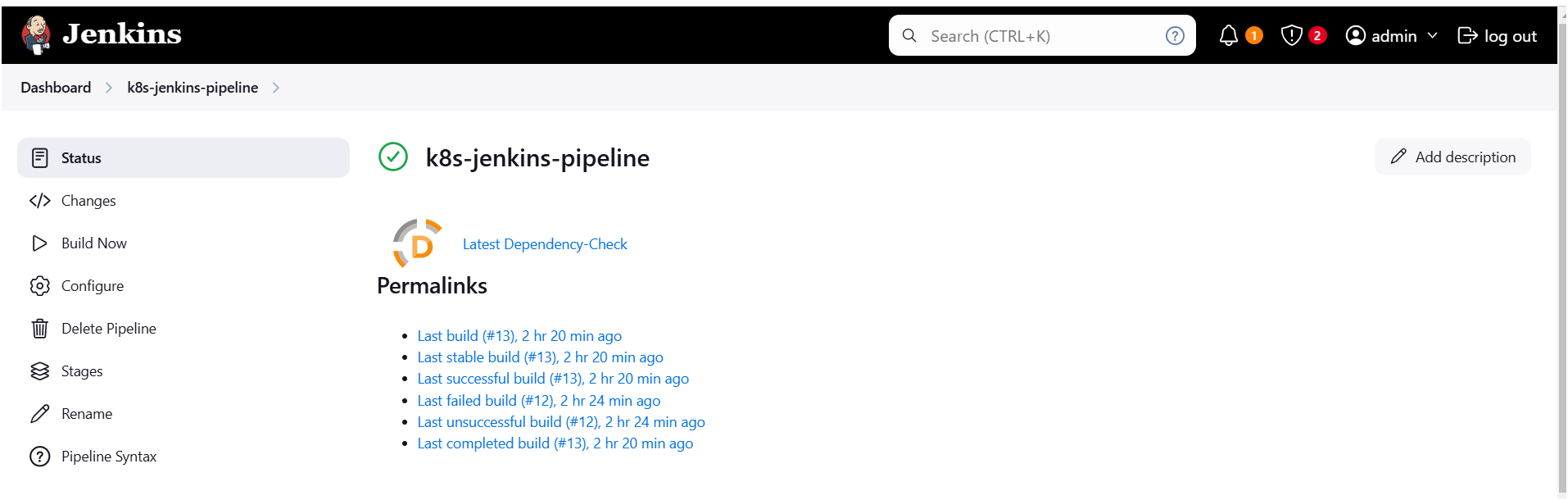
}

}

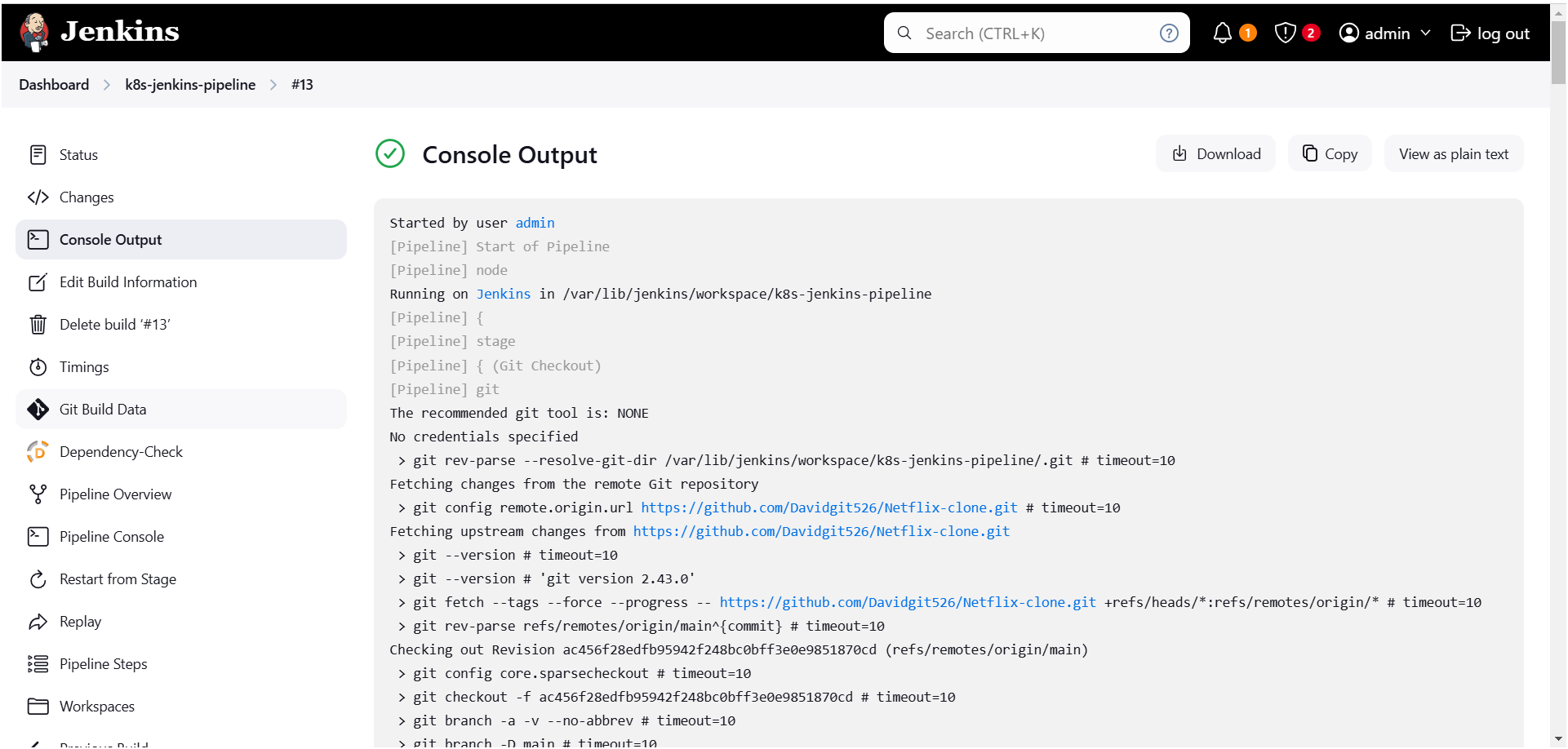
}

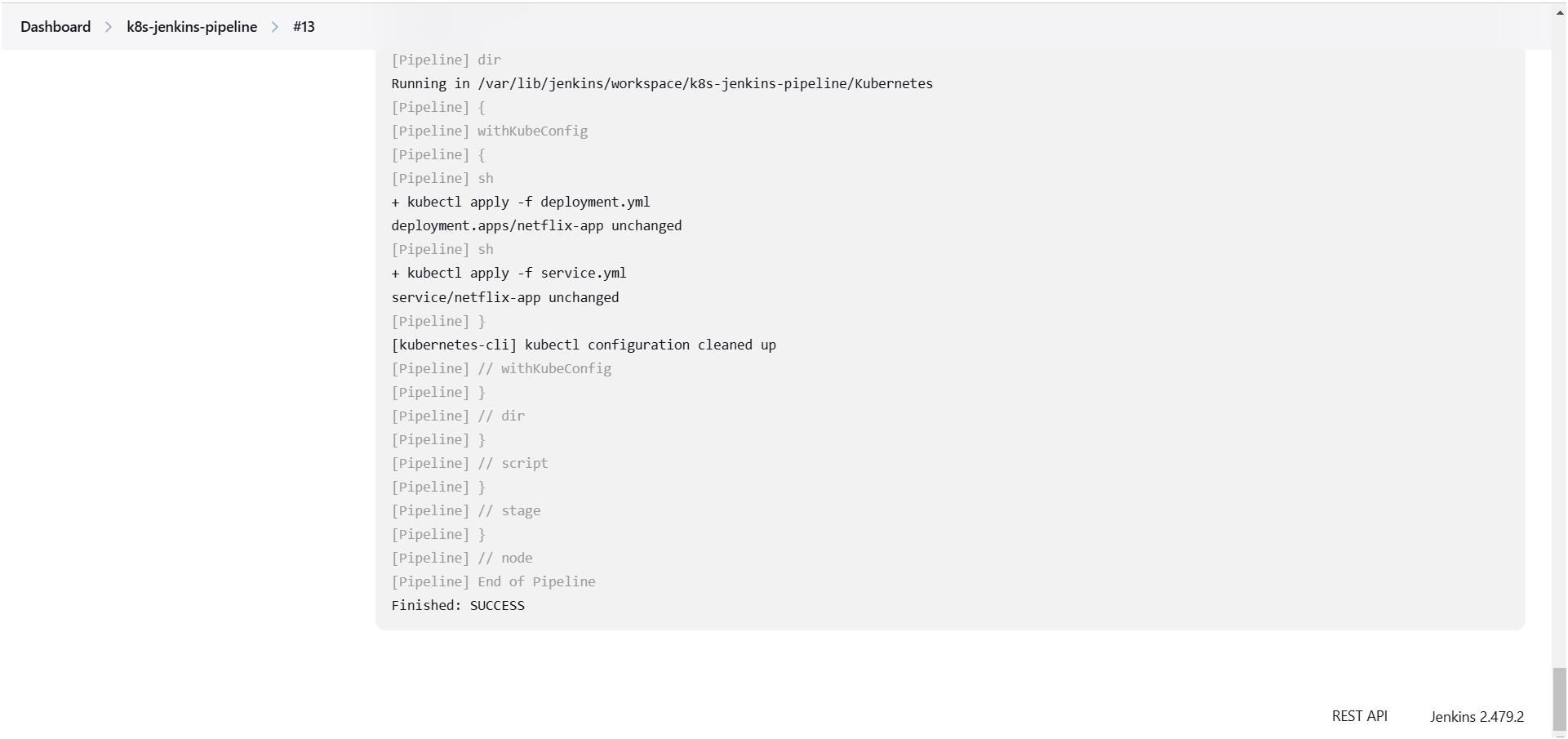
}

* After saving your script come back to your pipeline and click on “Build Now” to running your pipeline 👇

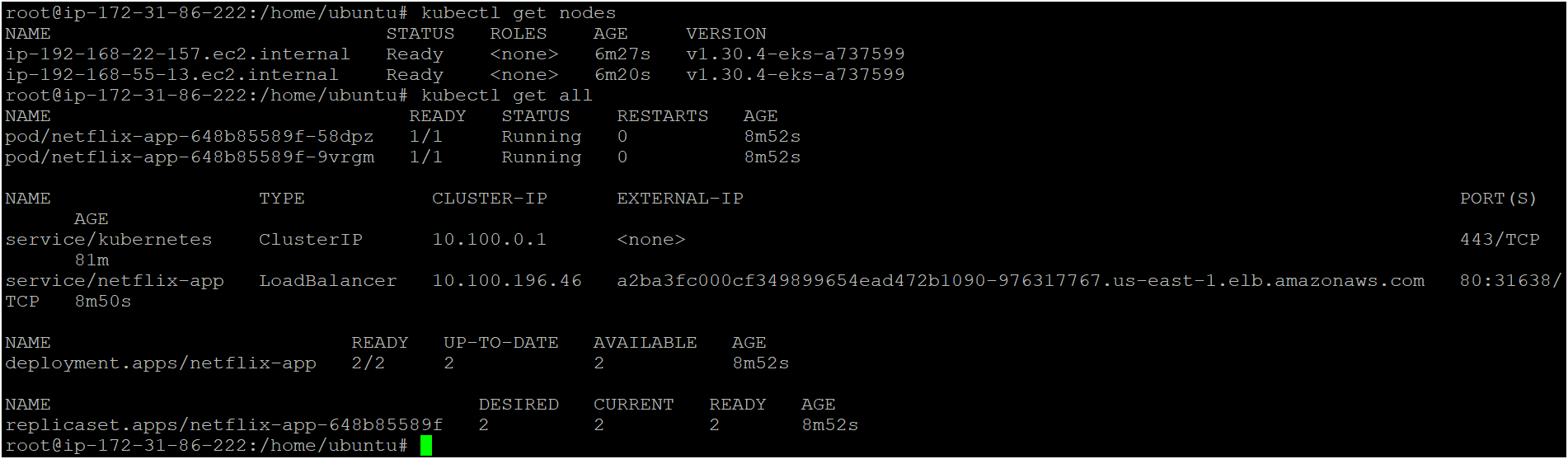


* Open console output by clicking on build number 👇

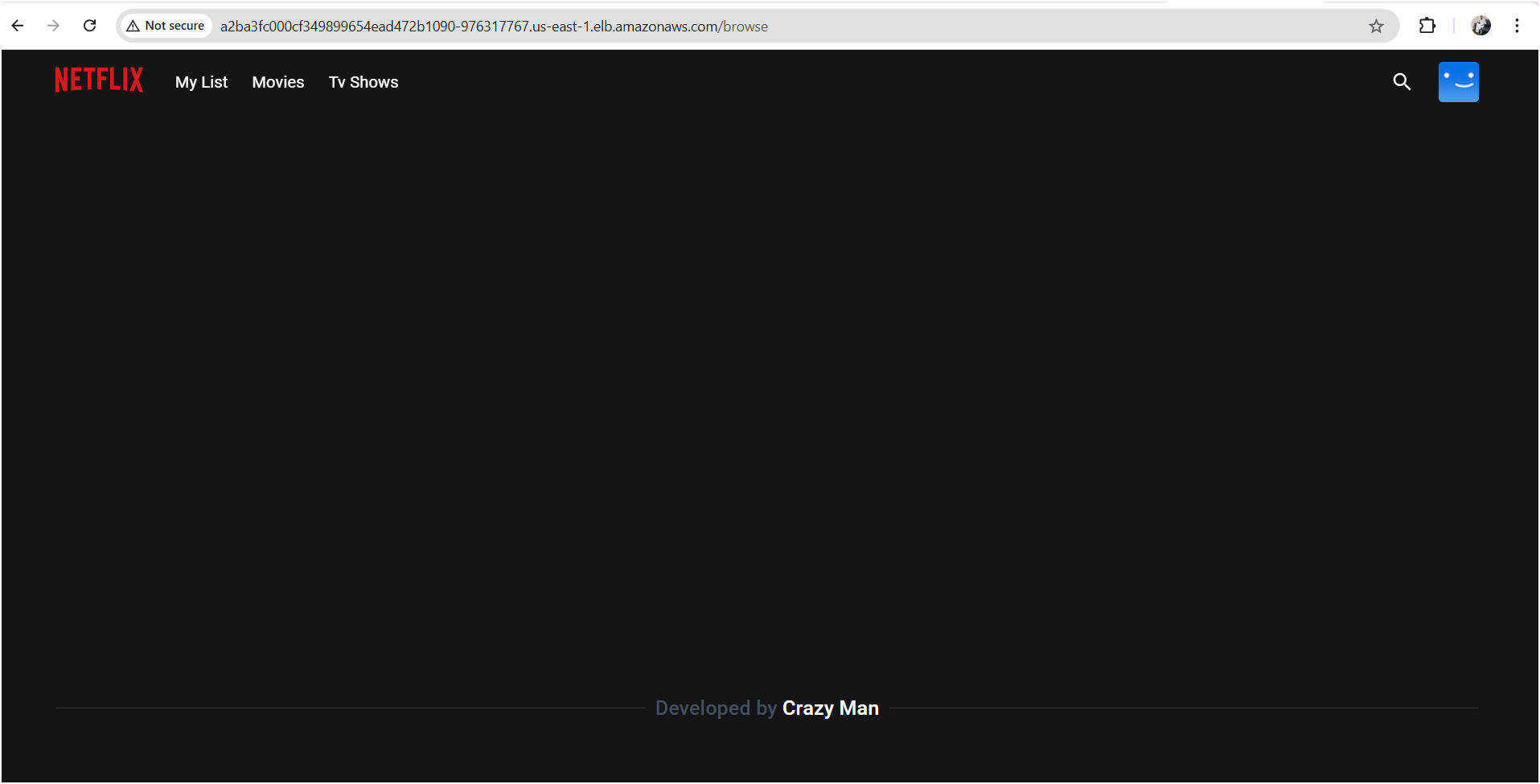




* Go to your cluster check application has deployed or not 👇
* My Netflix application successfully has deployed on eks cluster



* Am able to access my application



**Monitoring**

* Set up Prometheus and Grafana to monitor your application.

**Installing Prometheus:**

First, create a dedicated Linux user for Prometheus and download Prometheus: