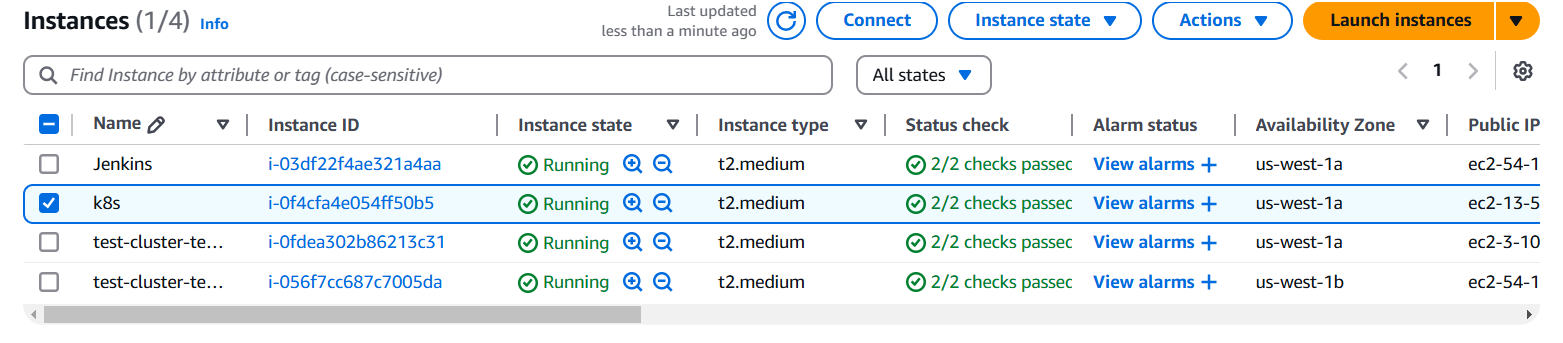
# **How to deploy Retail-app-Kubernetes on k8s cluster using Jenkins**

**Eks cluster**

* Firstly, we need to set up k8s cluster in aws
* Create a linux server with ubuntu os, instance type is t2.medium and storage is 30gb for setting up eks cluster



* Connect your ec2 instance with putty using shh client
* After that you have to install following cli tools on your server 👇

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



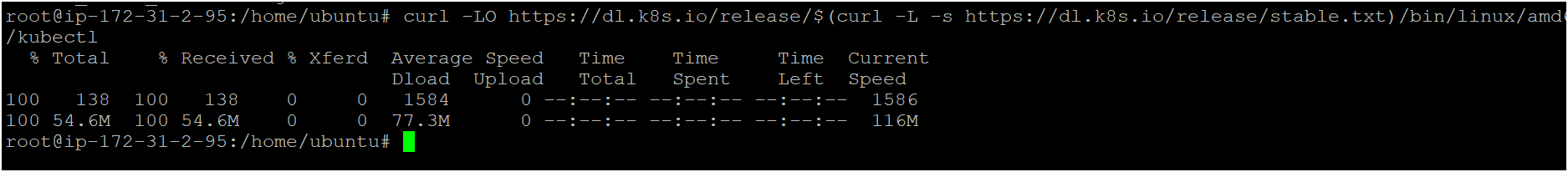
1. **Kubectl**

**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

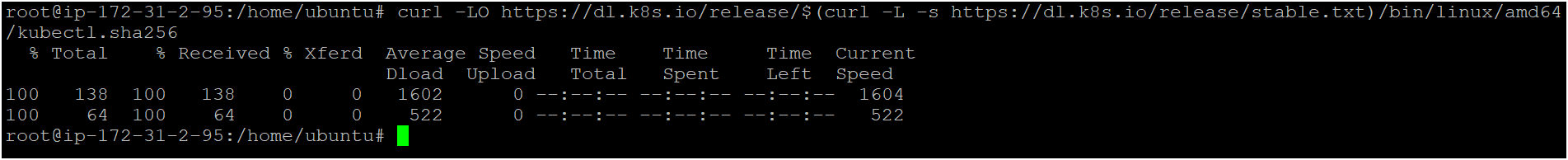
curl -LO [https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)



* Validate the binary (optional)

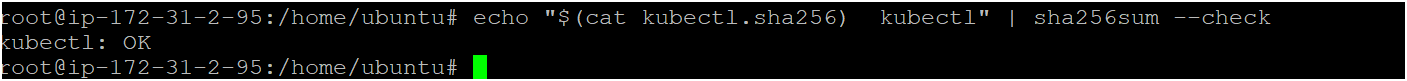
Download the kubectl checksum file:

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



* 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

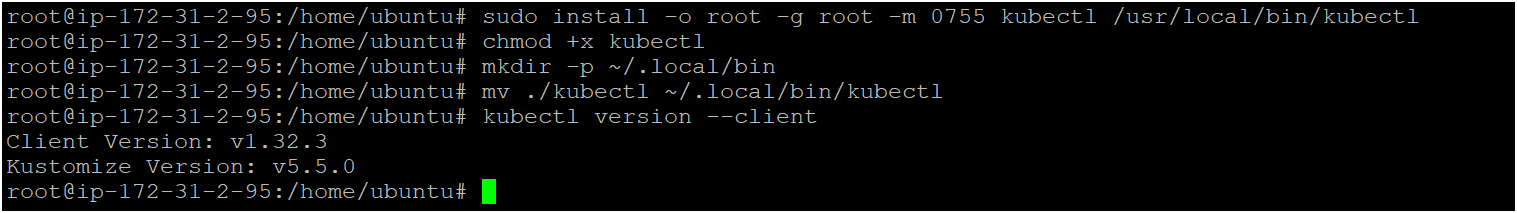
chmod +x kubectl

mkdir -p ~/.local/bin

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

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

kubectl version –client

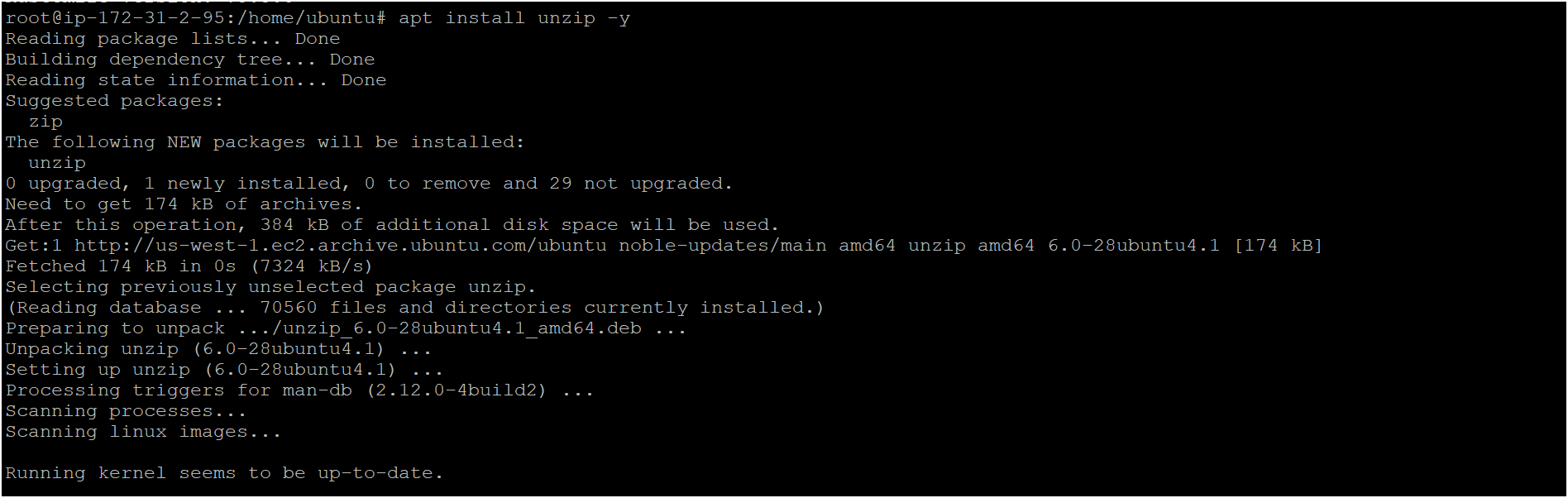


1. **AWS CLI**

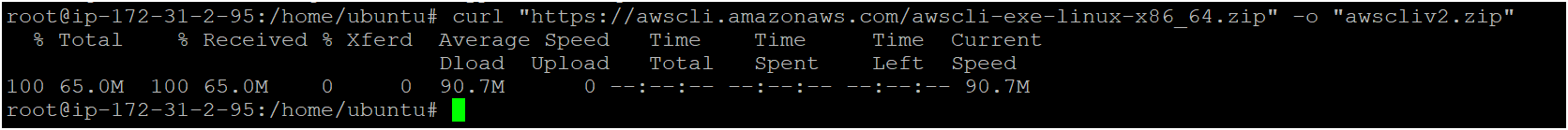
**Description: -** The **AWS Command Line Interface (CLI)** is a tool provided by Amazon Web Services that enables users to interact with AWS services from a terminal or command prompt. It allows you to perform various AWS tasks directly from the command line, including managing resources, configuring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

**Command: -**

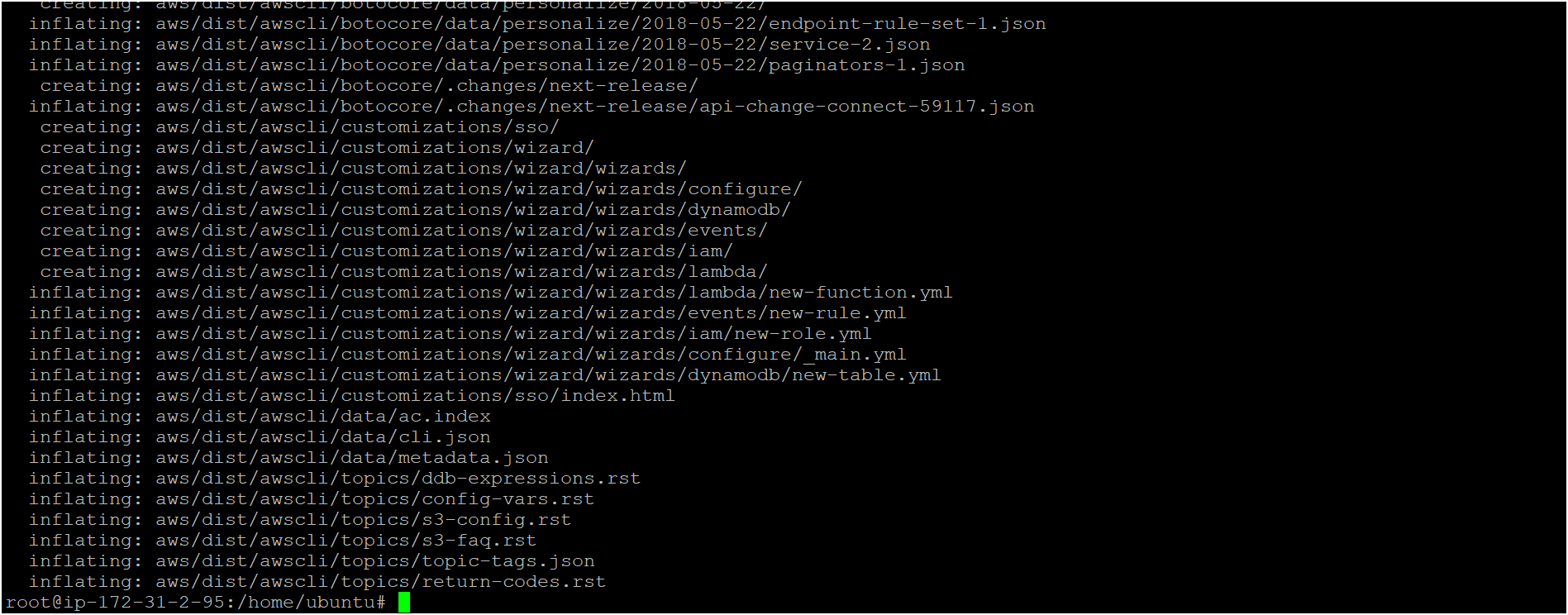
apt install unzip -y



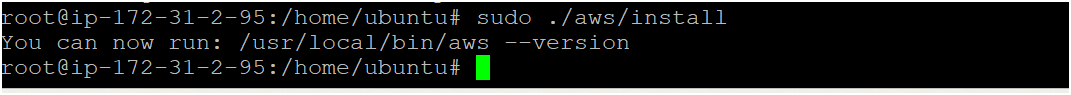
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"



unzip awscliv2.zip

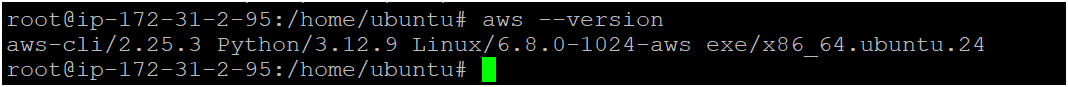


sudo ./aws/install



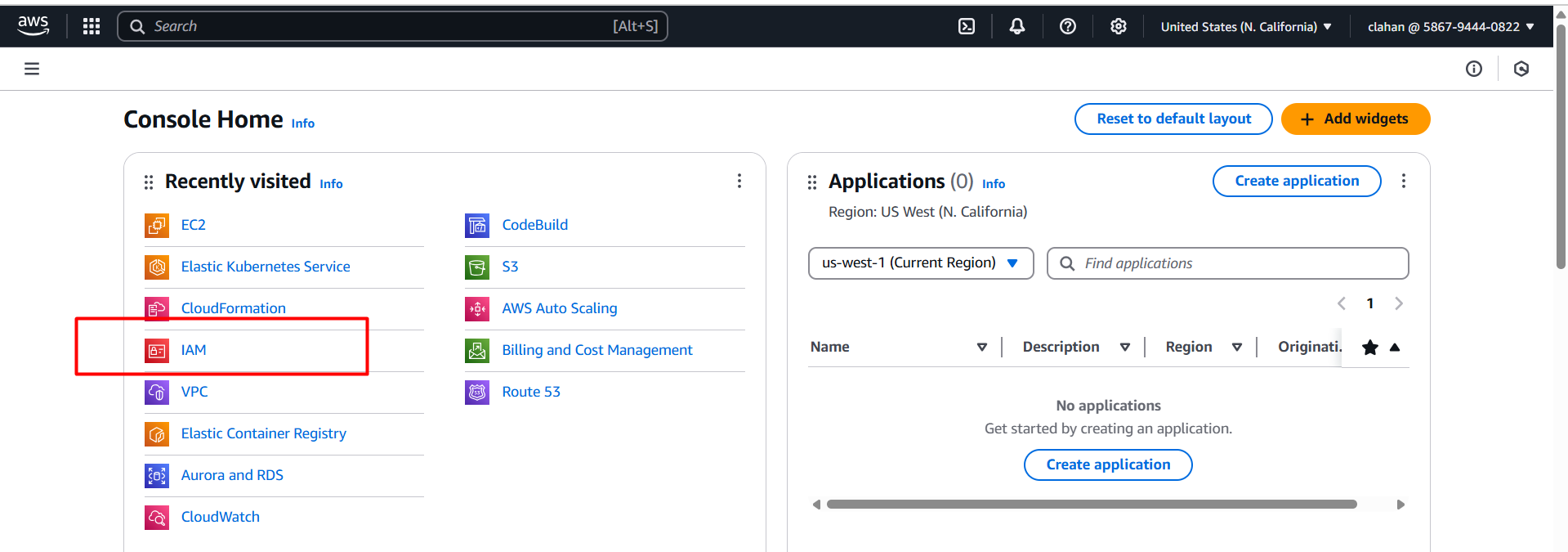
**Confirm the installation with the following command: -**

**aws --version**

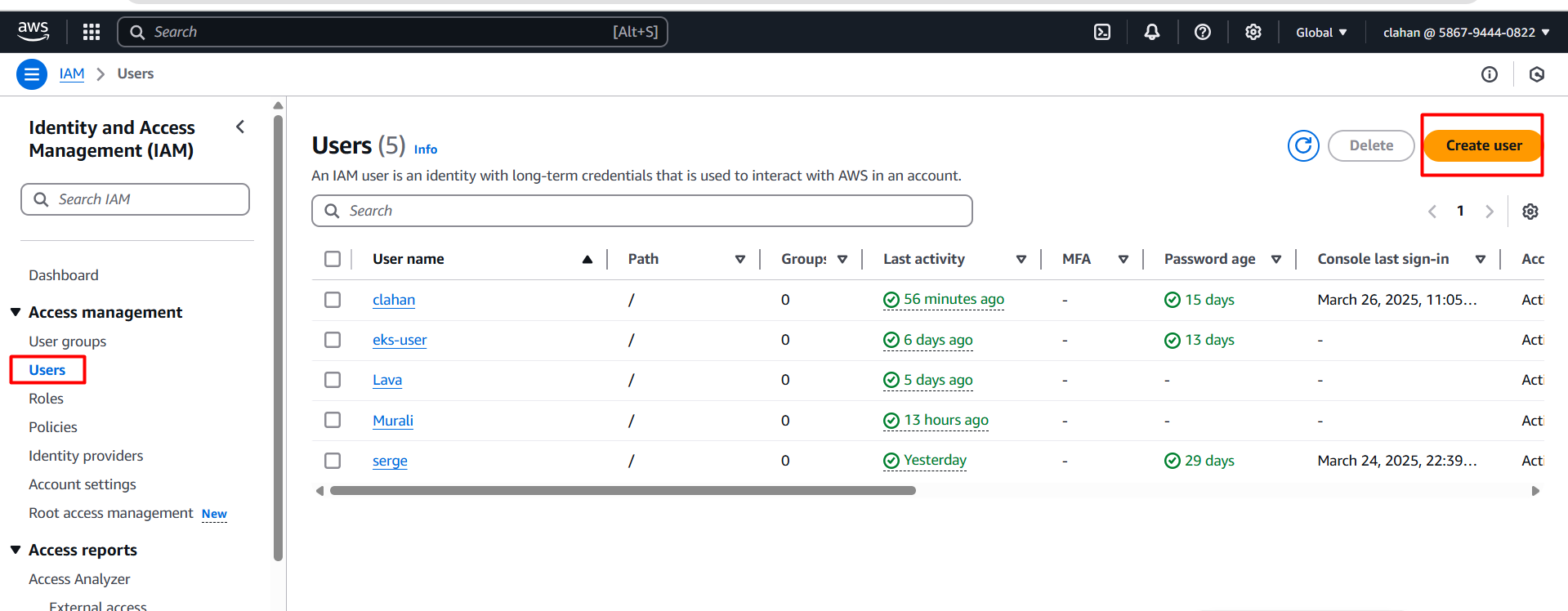
****

**IAM USER: -**

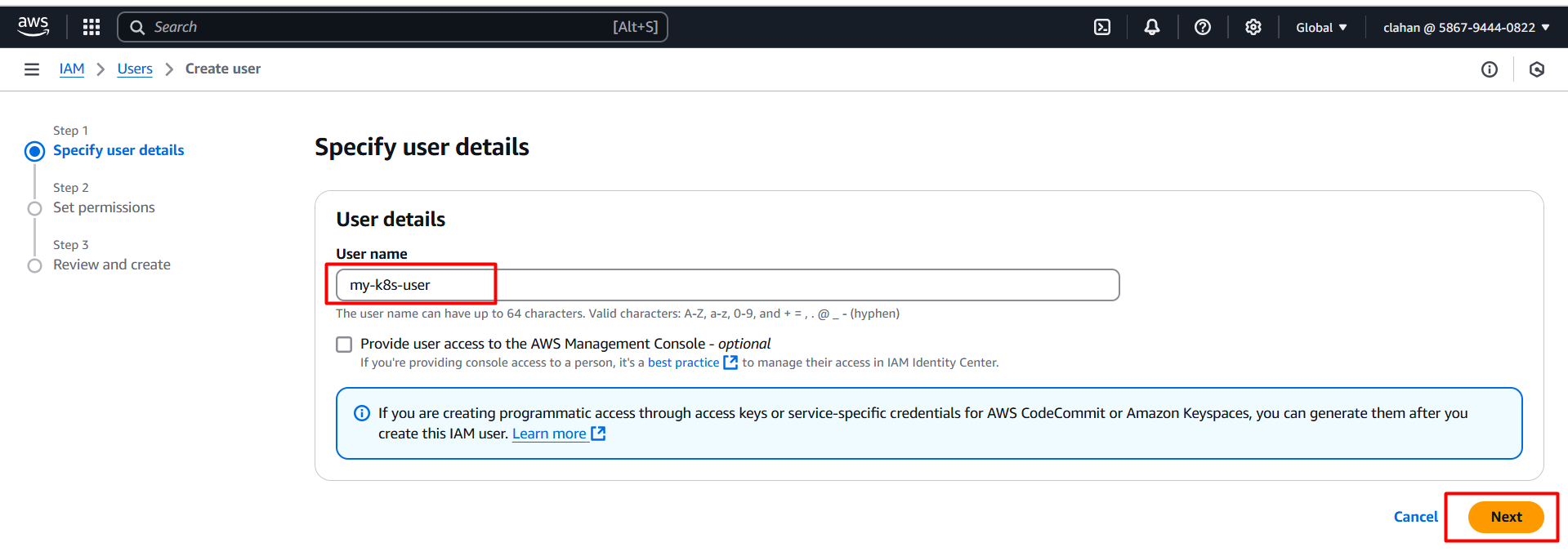
* Create one iam user as following below steps 👇
* Click on IAM service or search for IAM in aws console



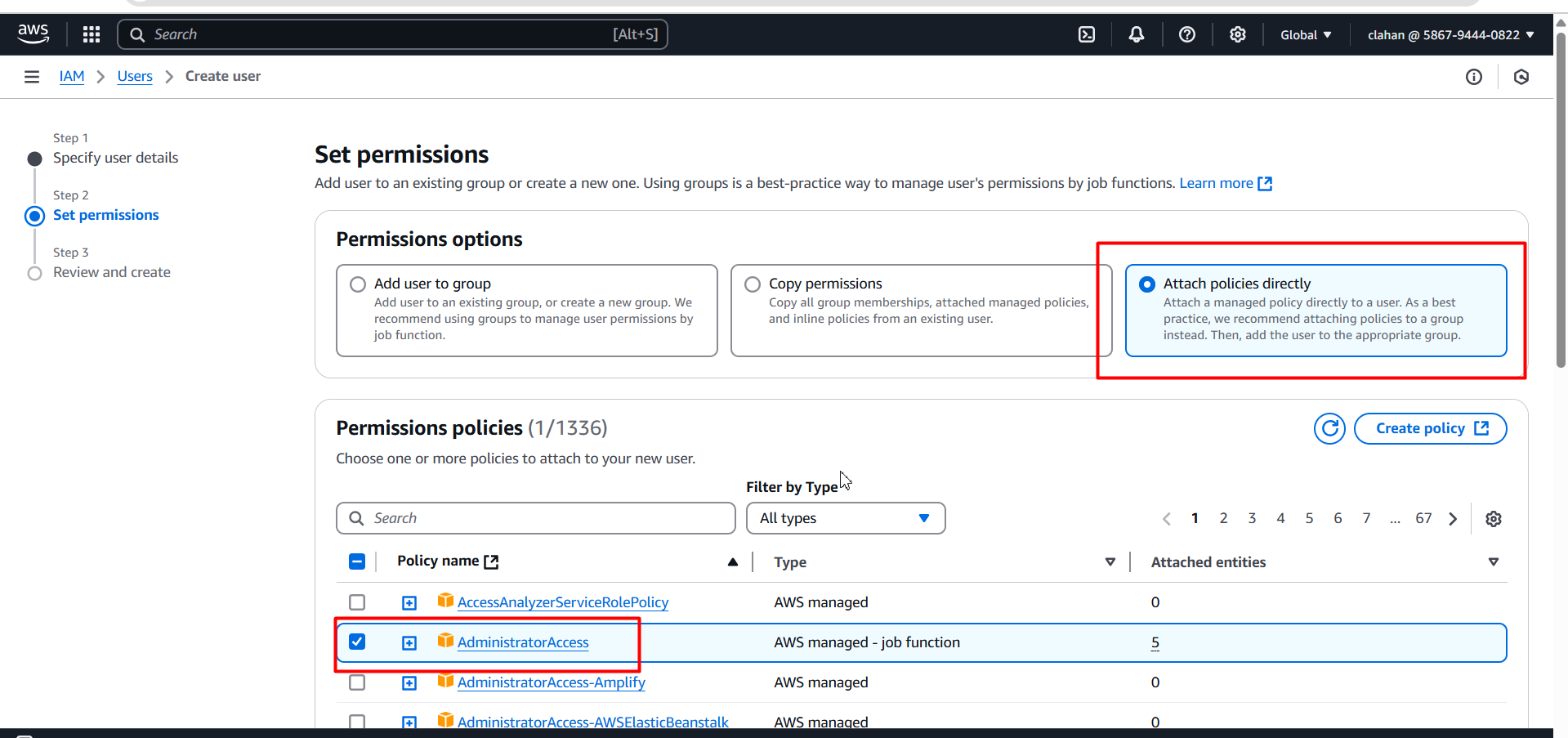
* Click on users 🡪 create user



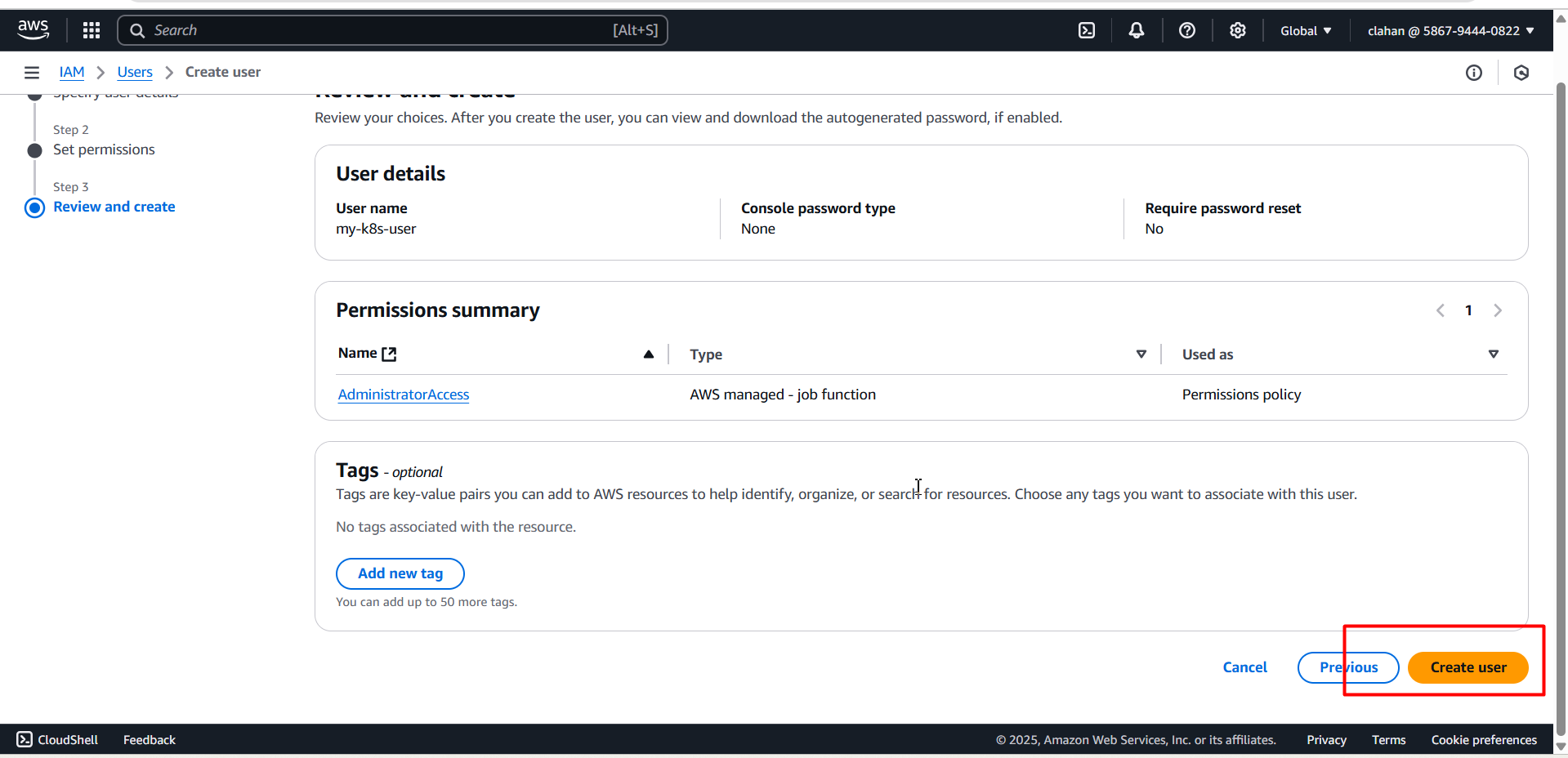
* Give a name and click on next



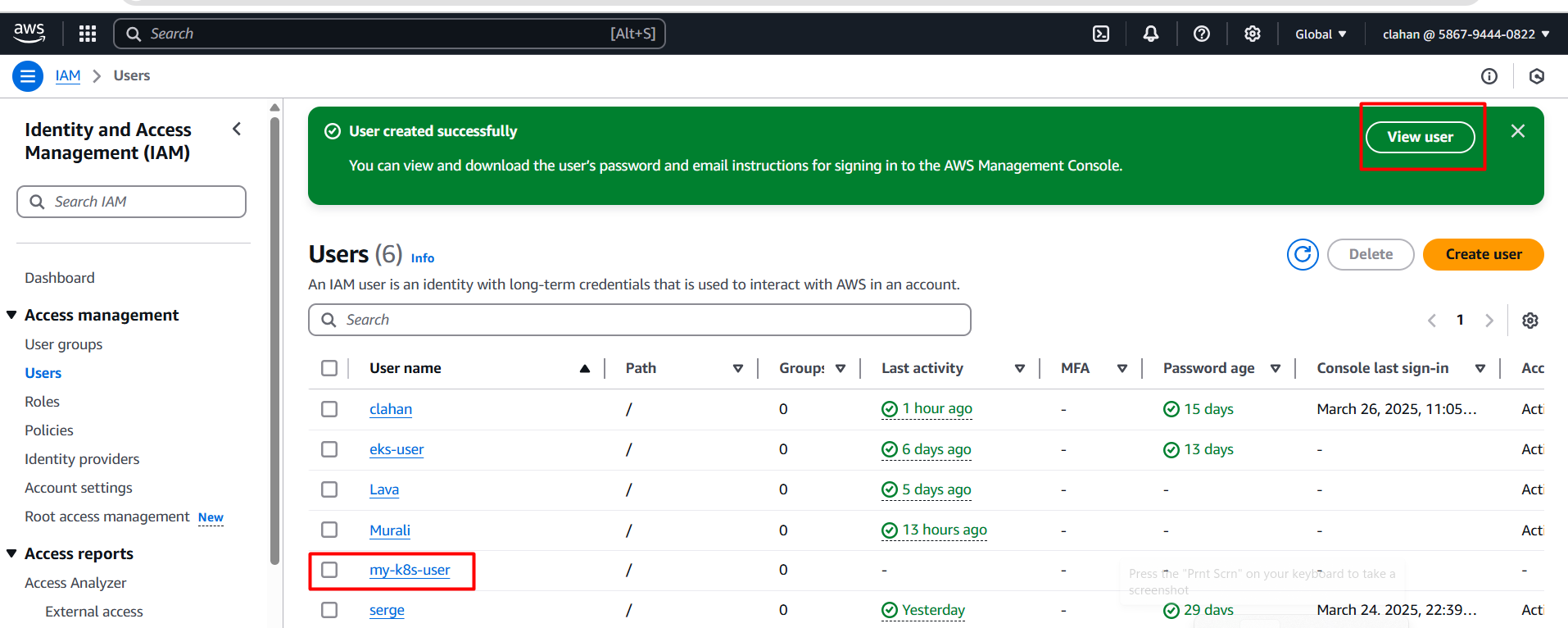
* Choose attach policies directly and select permission “Administrator access” and click on next



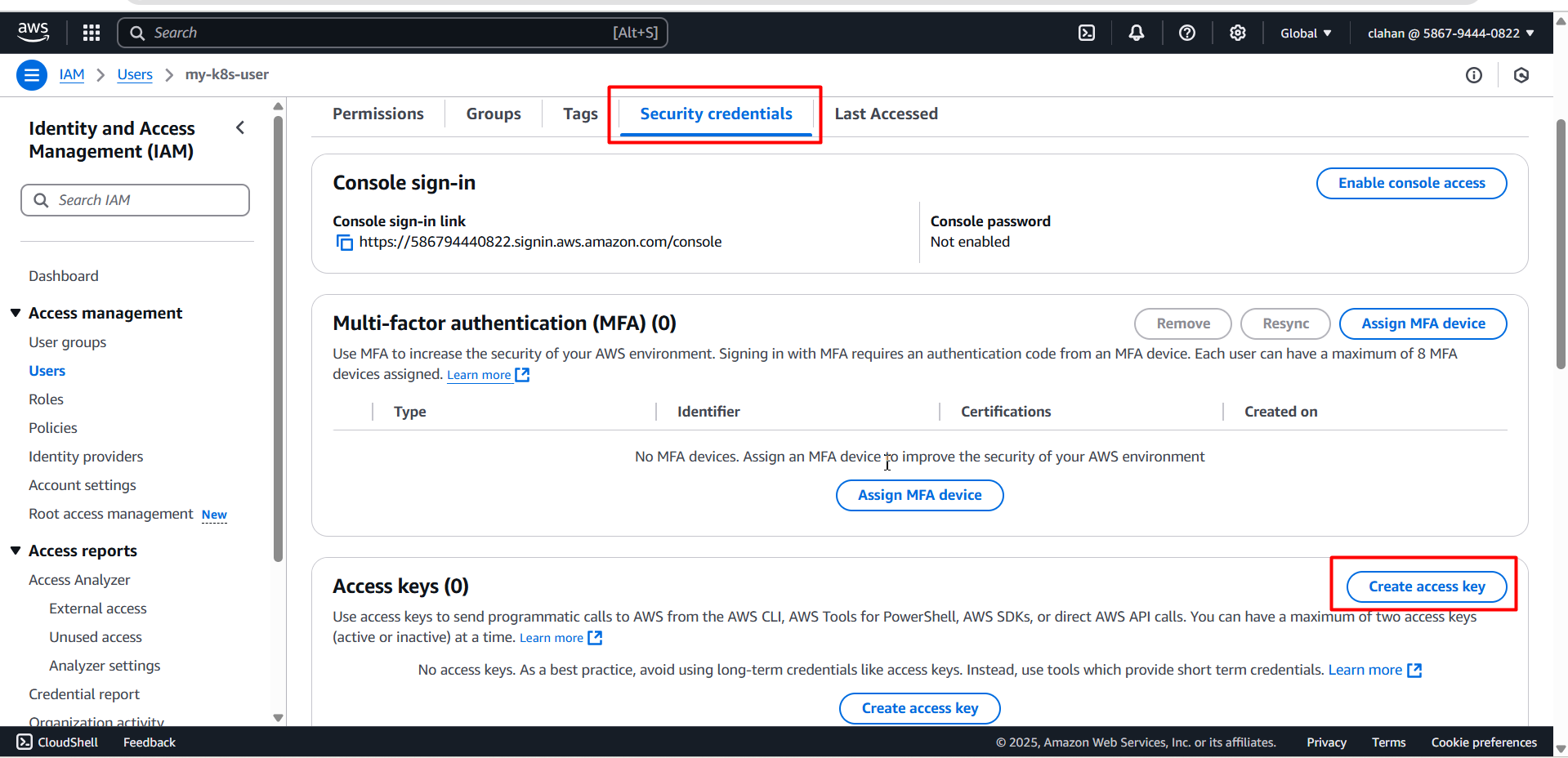
* Review the details then click on create user



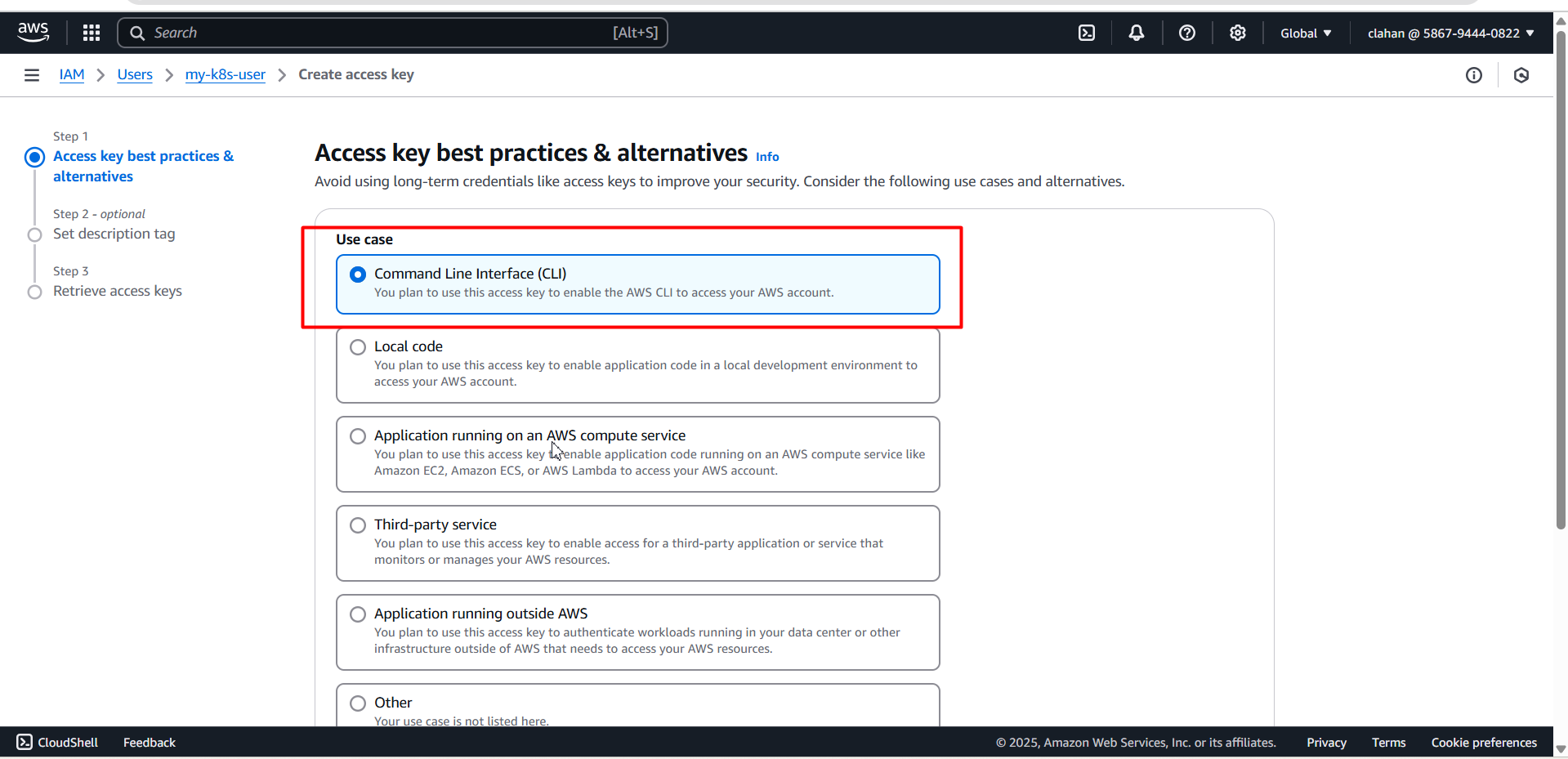
* IAM user created and click on view

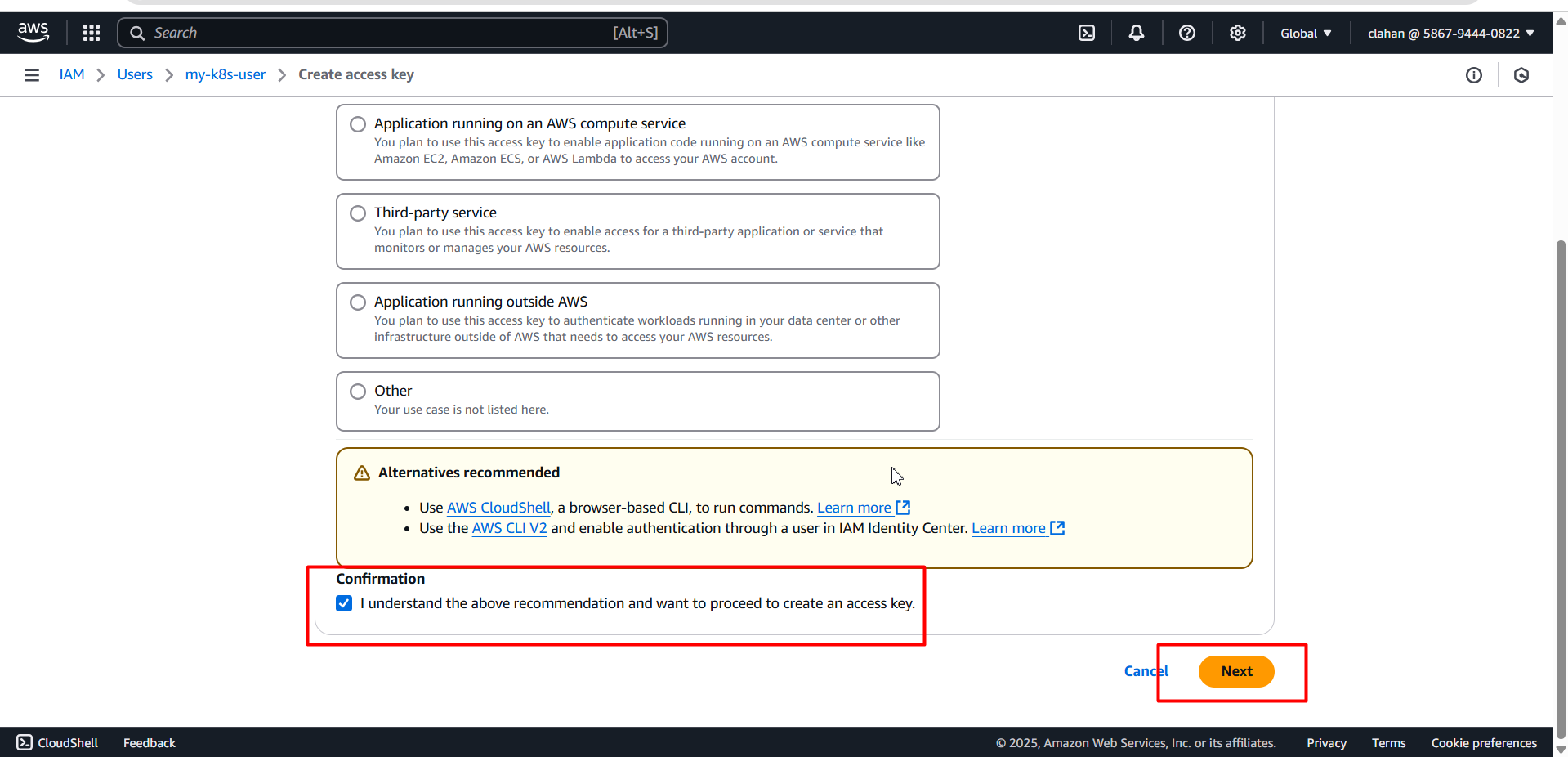


* Inside the iam user click on security credentials 🡪 click create access key (we need generate access key id & secret access key id for providing in cluster)

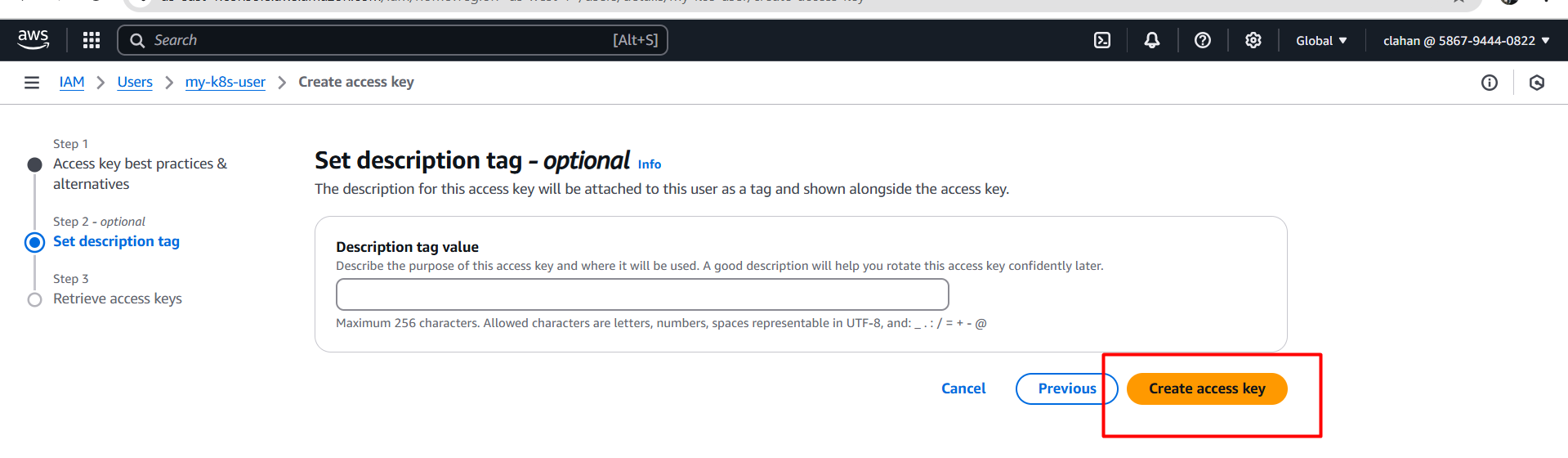


* Next choose command line interface (cli) 🡪 confirm the check box (below) 🡪 click on next

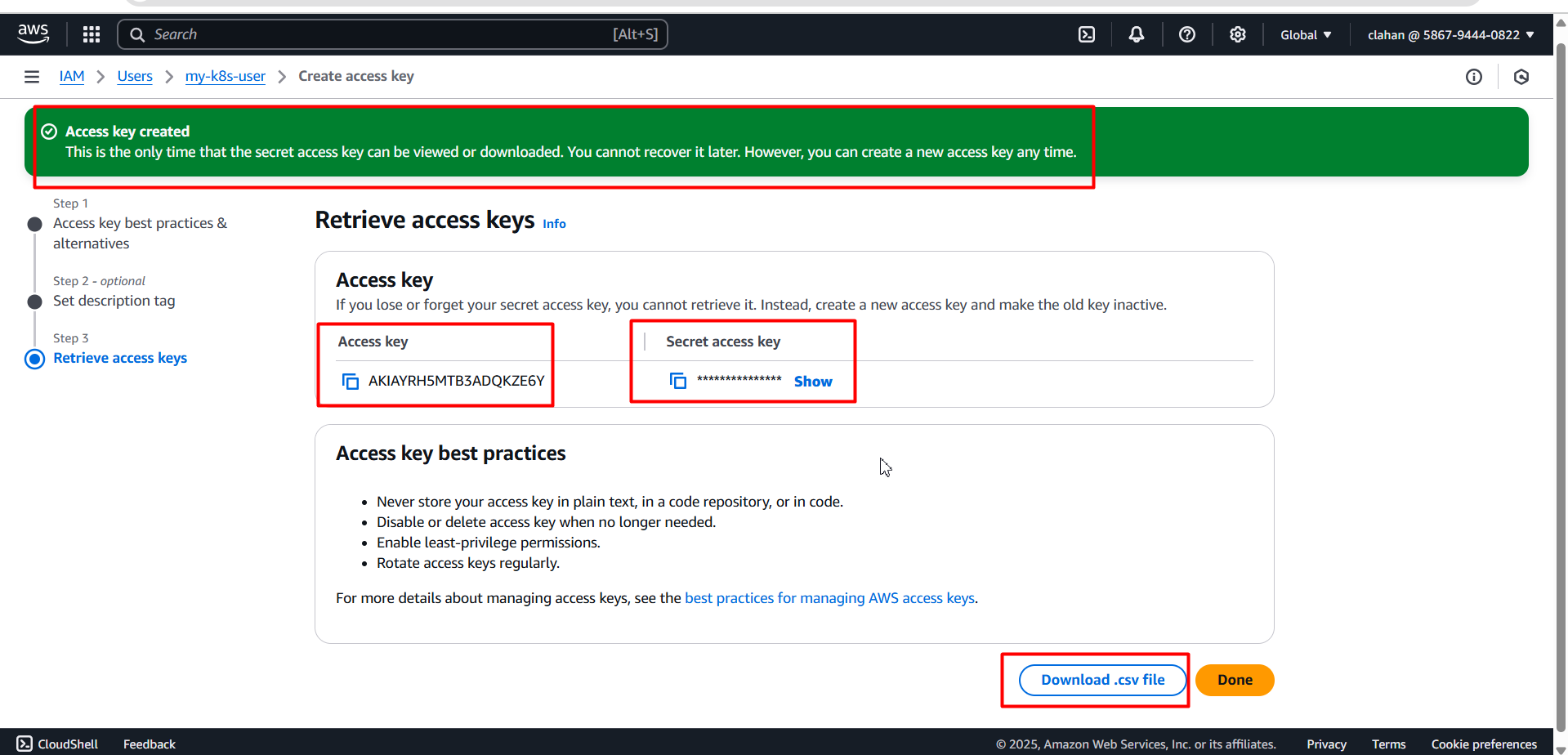




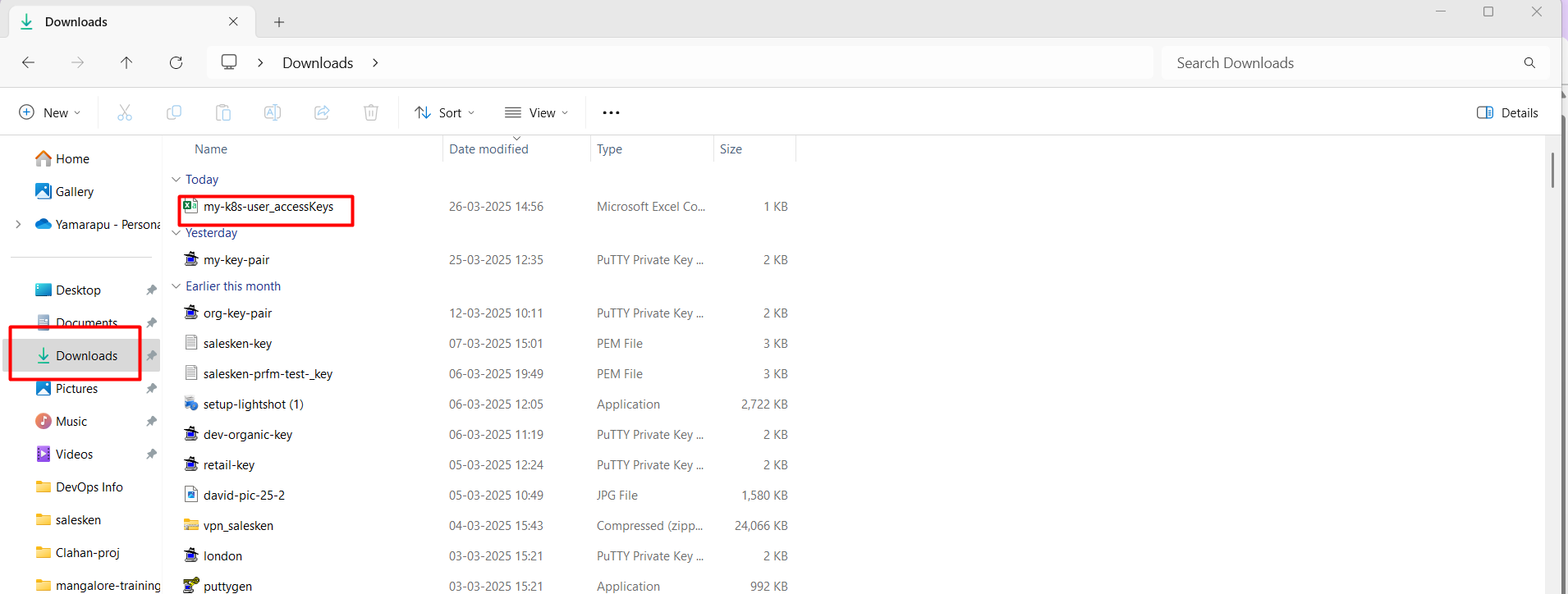
* Click on create access key



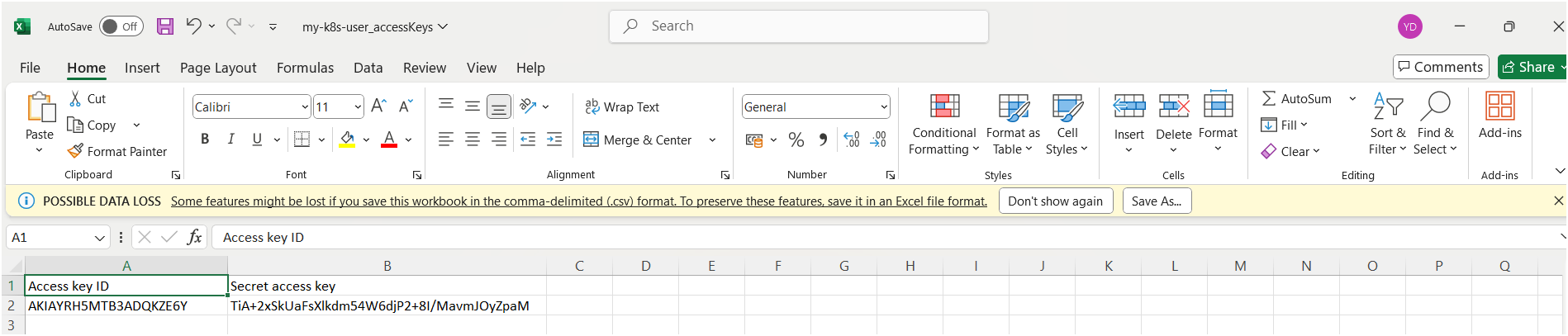
* Access keys are only visible one time so download them locally by clicking on download.csv file



* Click on Download.csv file it will download into your file manager

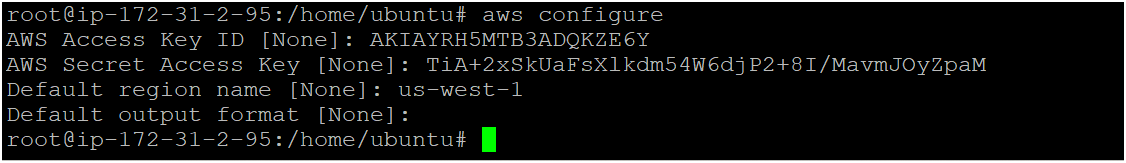


* Now you have to provide/configure those access keys and secret key id on your server



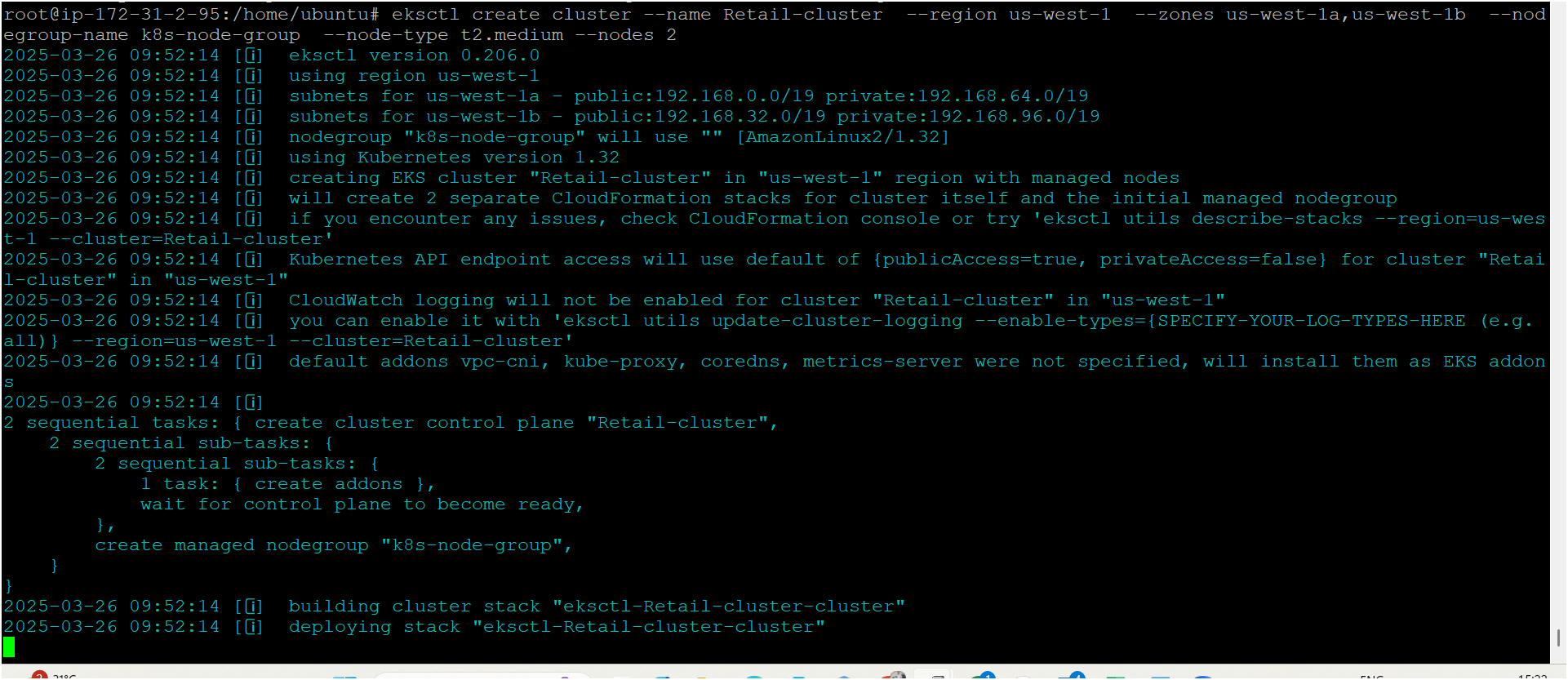
aws configure (execute this command on your terminal)

* Specify your using region

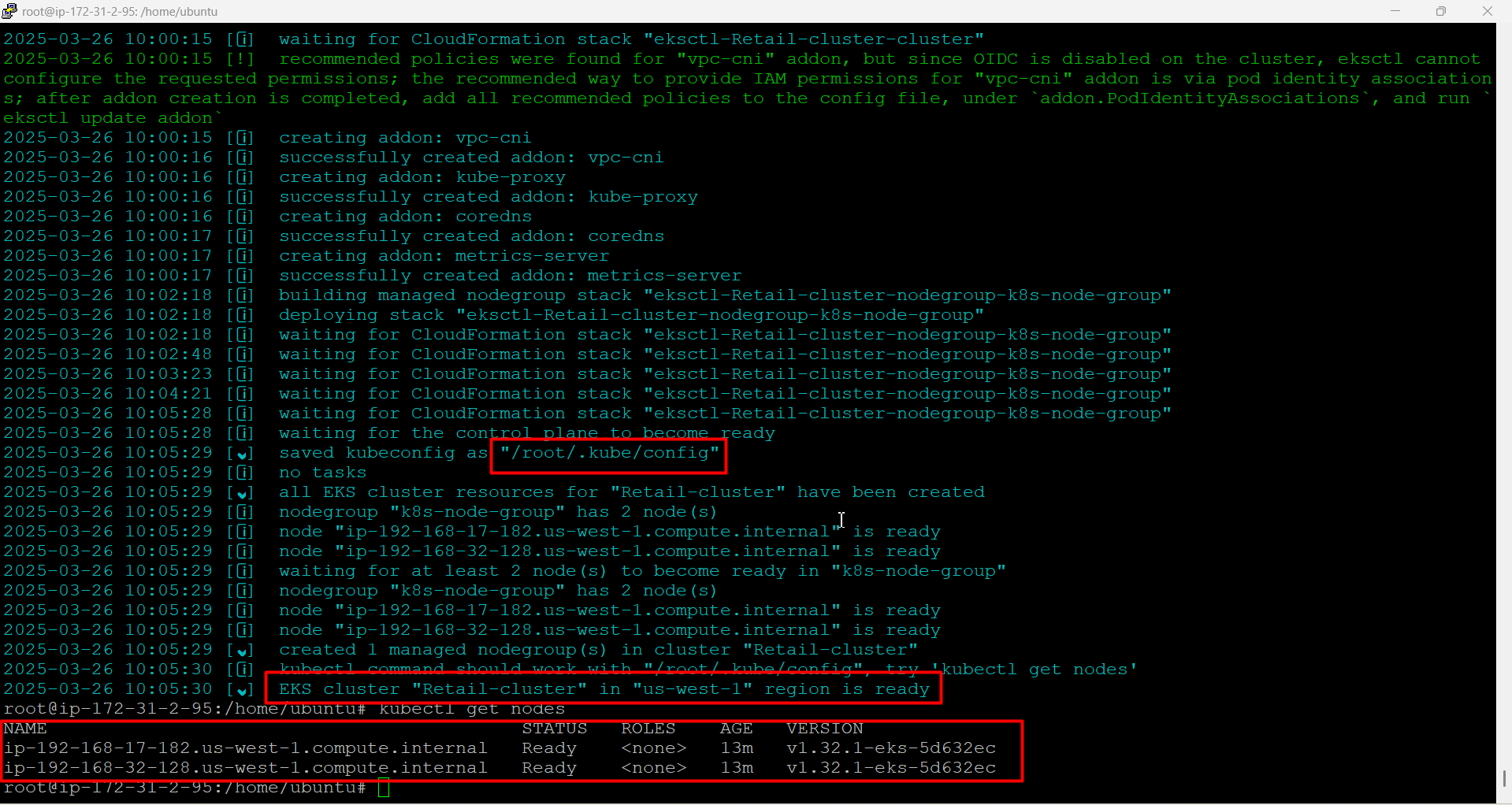


* Now you can create eks cluster by executing following command

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

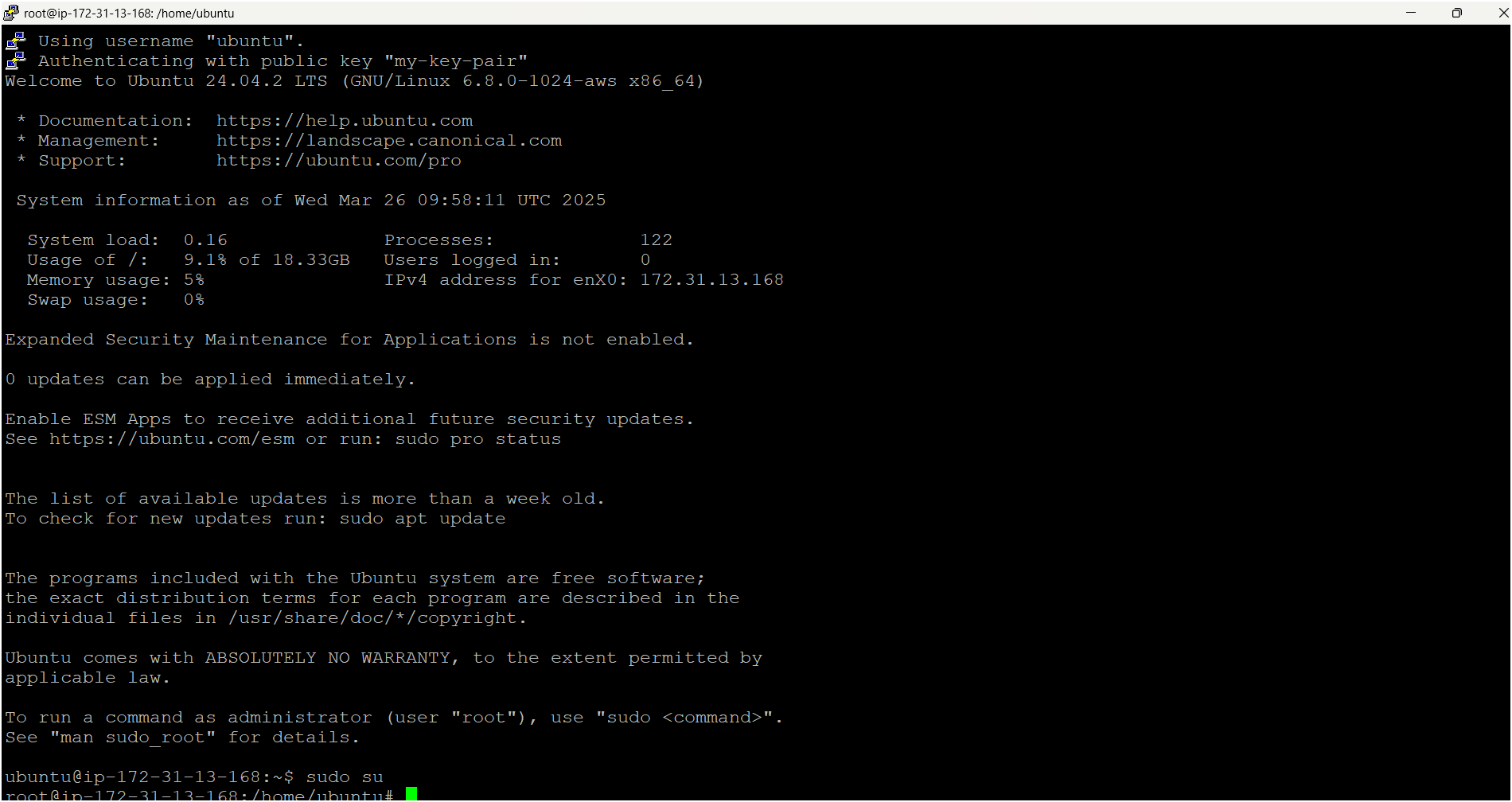


* Now your cluster is ready as shown below image 👇



**Jenkins**

* Create one ec2 instance with ubuntu os, instance type t2.medium and storage 25 (in same region where you created your k8s cluster
* Connect instance to putty server
* Please make sure you have to install kubectl and aws cli on Jenkins server (you can get command above where I mentioned for k8s cluster)
* Now install Jenkins on your terminal by executing following commands 👇



apt update -y

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

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

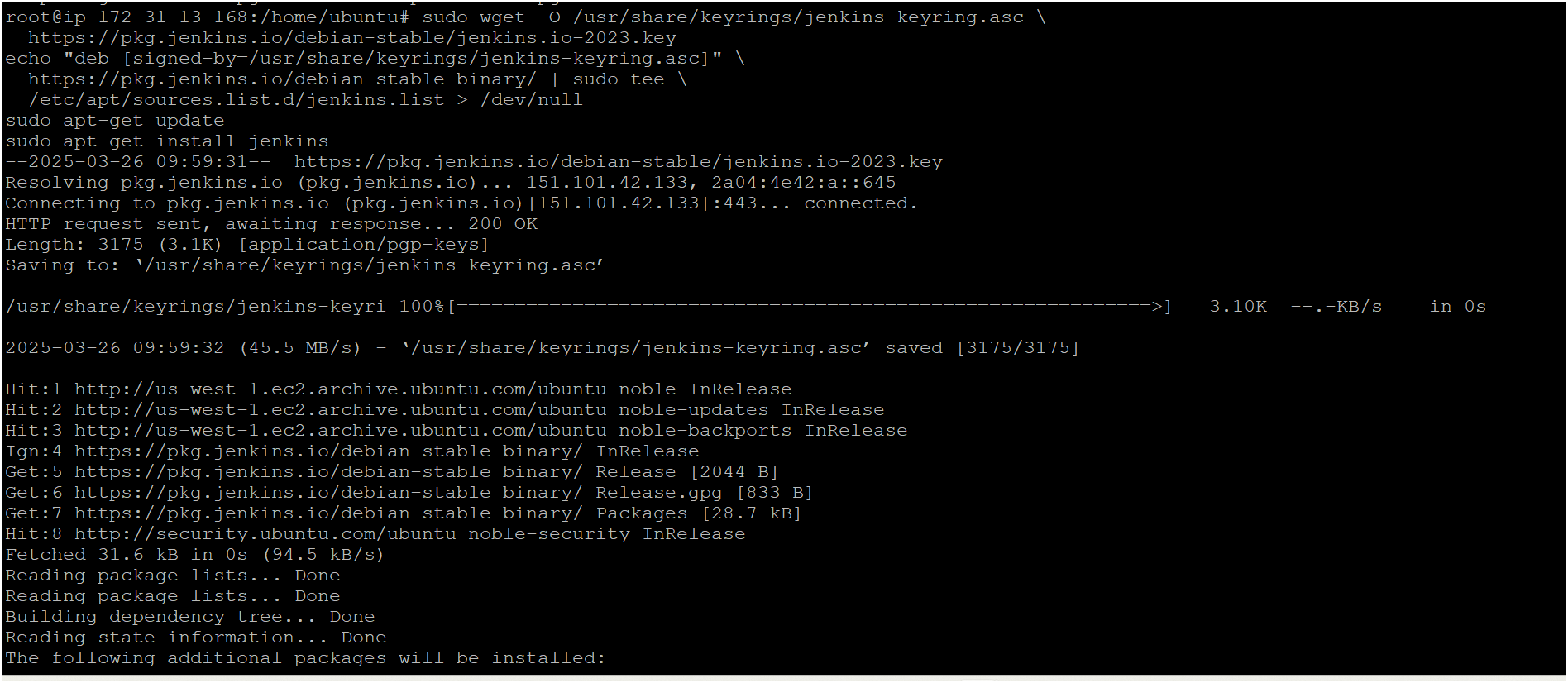
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

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

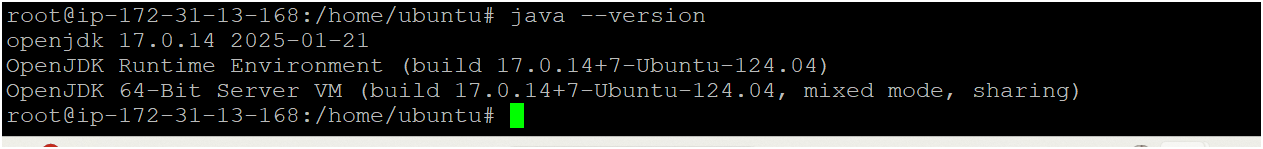


* Install java on your terminal where are installing Jenkins becz java is prerequisite for Jenkins

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

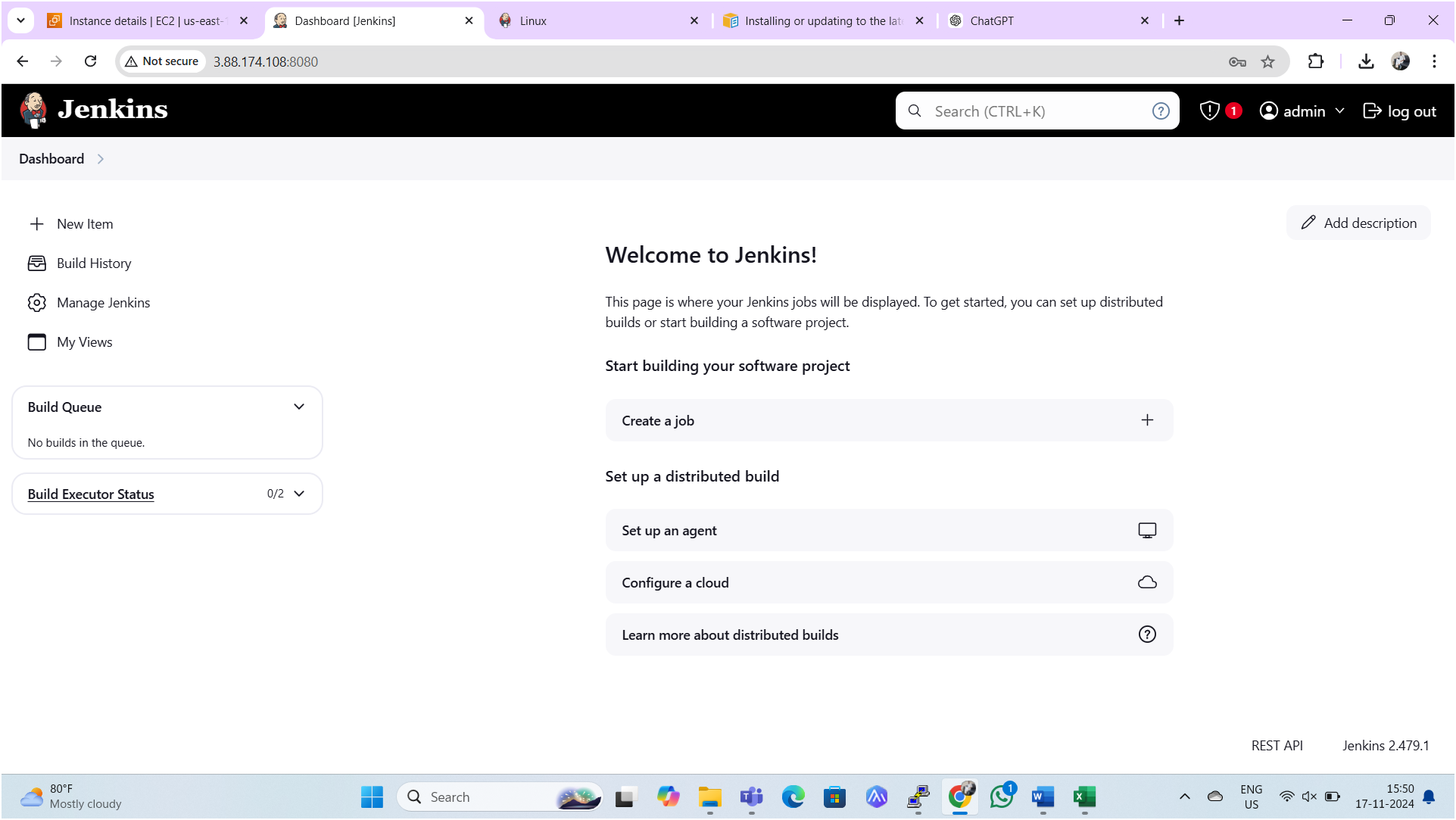
sudo systemctl start Jenkins

sudo systemctl status Jenkins

* Now your Jenkins server is ready



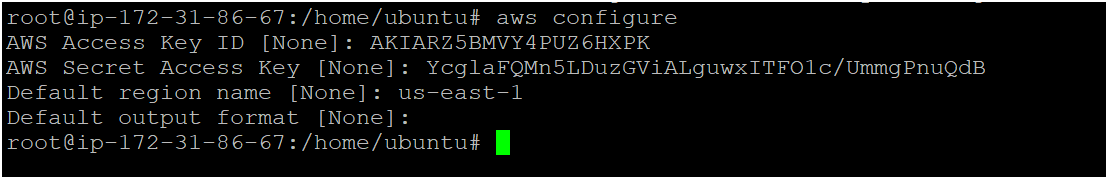
* Access your Jenkins server on browser using instance public ip address with port number as shown below image 👇
* Open port number 8080 in your security group for Jenkins



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

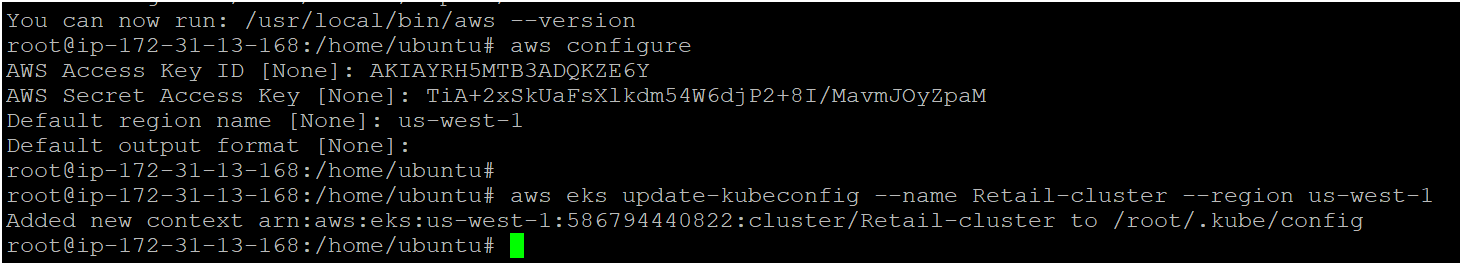


* Firstly, we have to provide aws access keys on Jenkins server



* 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

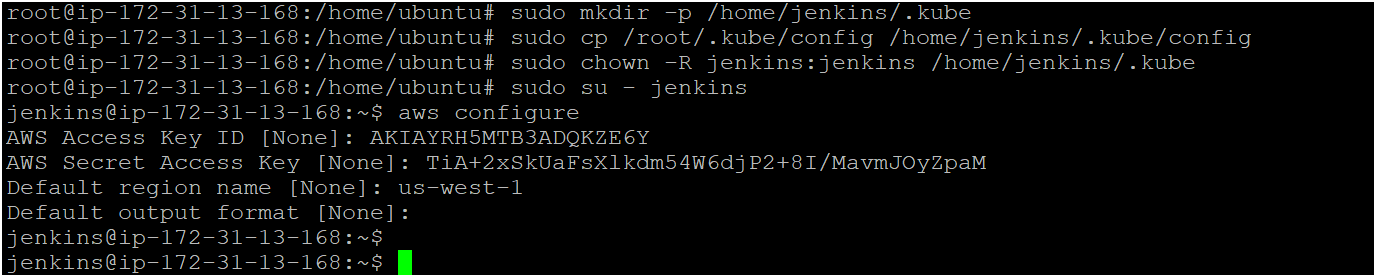
sudo mkdir -p /home/jenkins/.kube

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

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

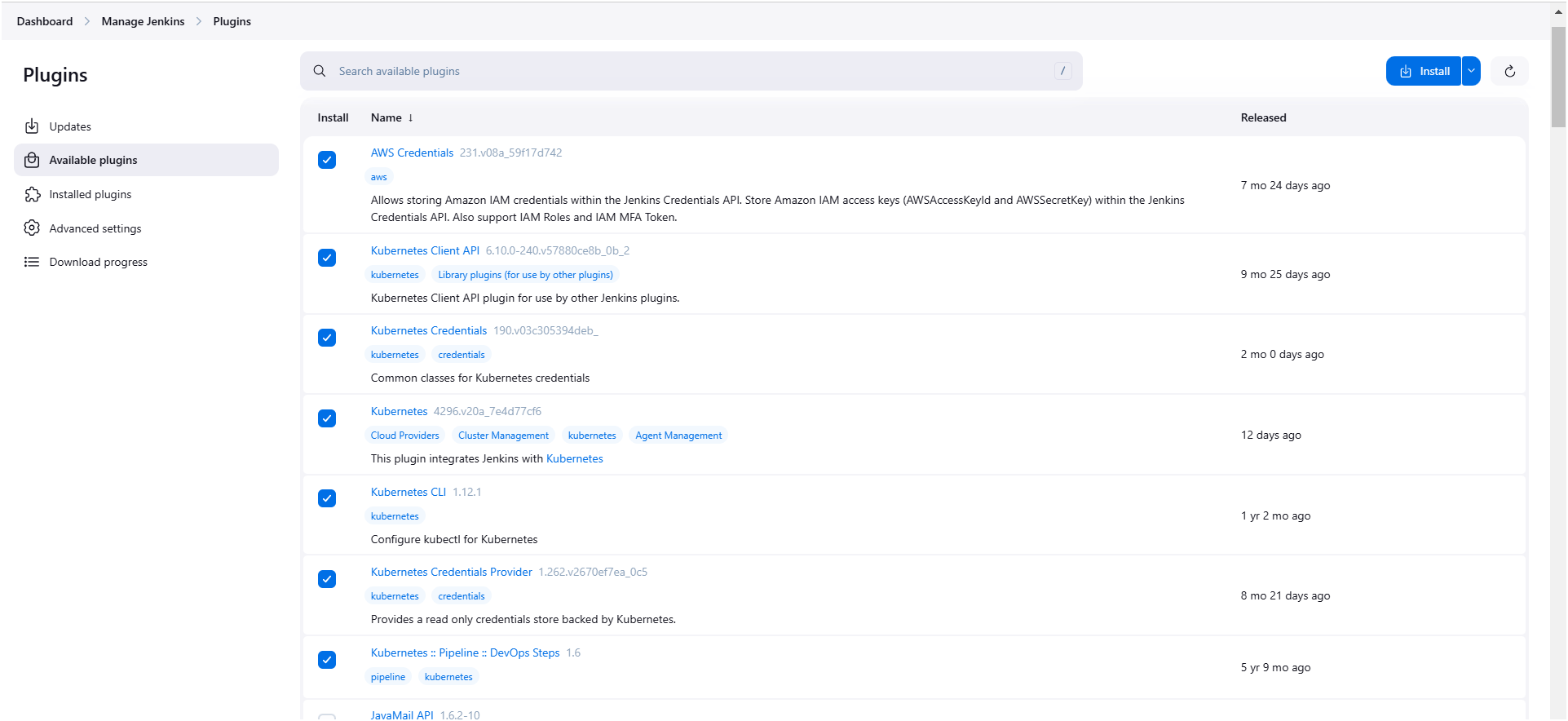
sudo su - jenkins

aws configure (provide access keys on Jenkins server)



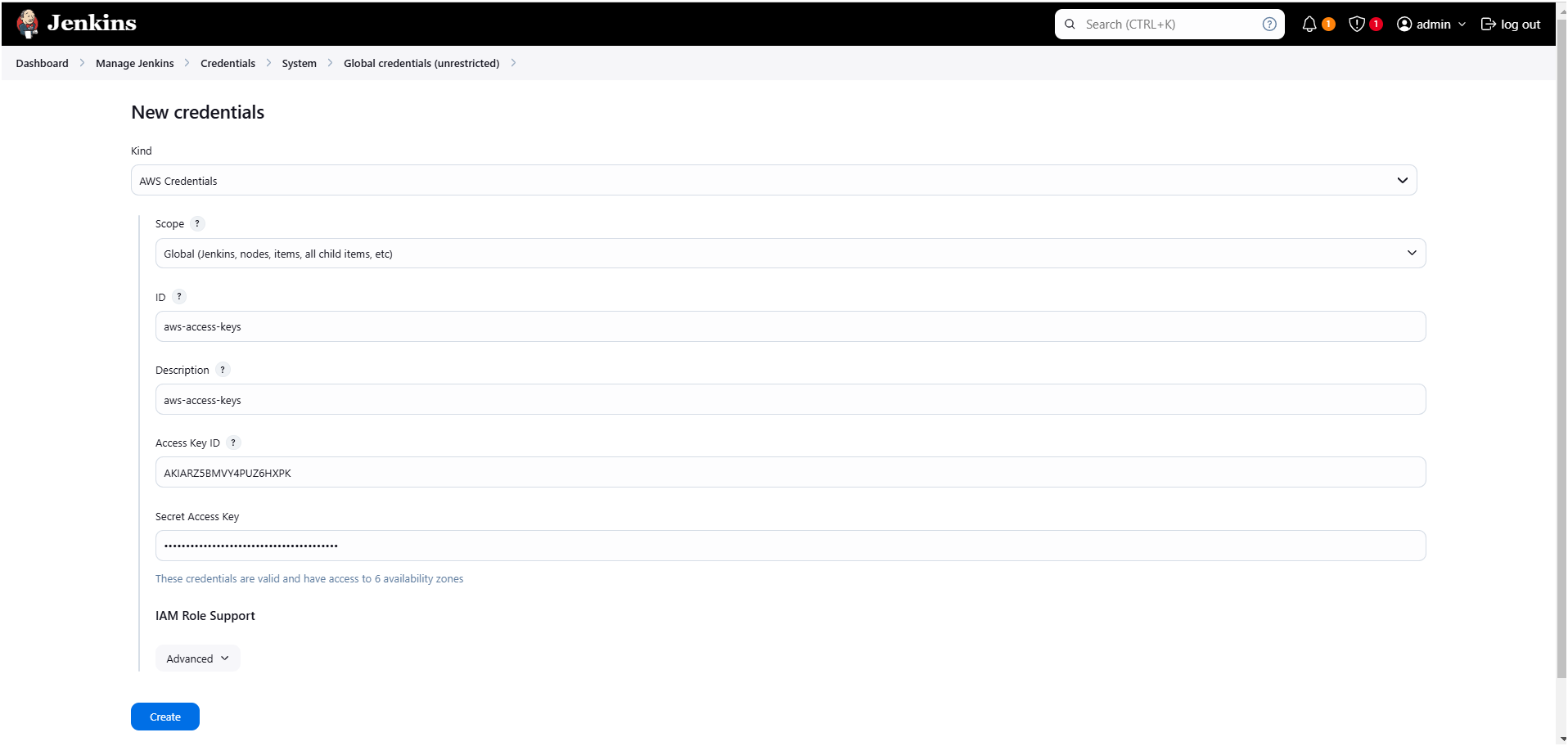
* Install following plugins on Jenkins

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)
8. OWASP dependence check



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

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



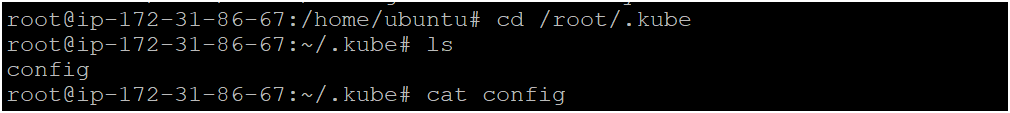
* Now we have to provide Kube-config file in Jenkins web server
* Firstly, we need to copy and save it in a document or file in your local as shown below image
* Execute below two commands on Jenkins server for copying config file

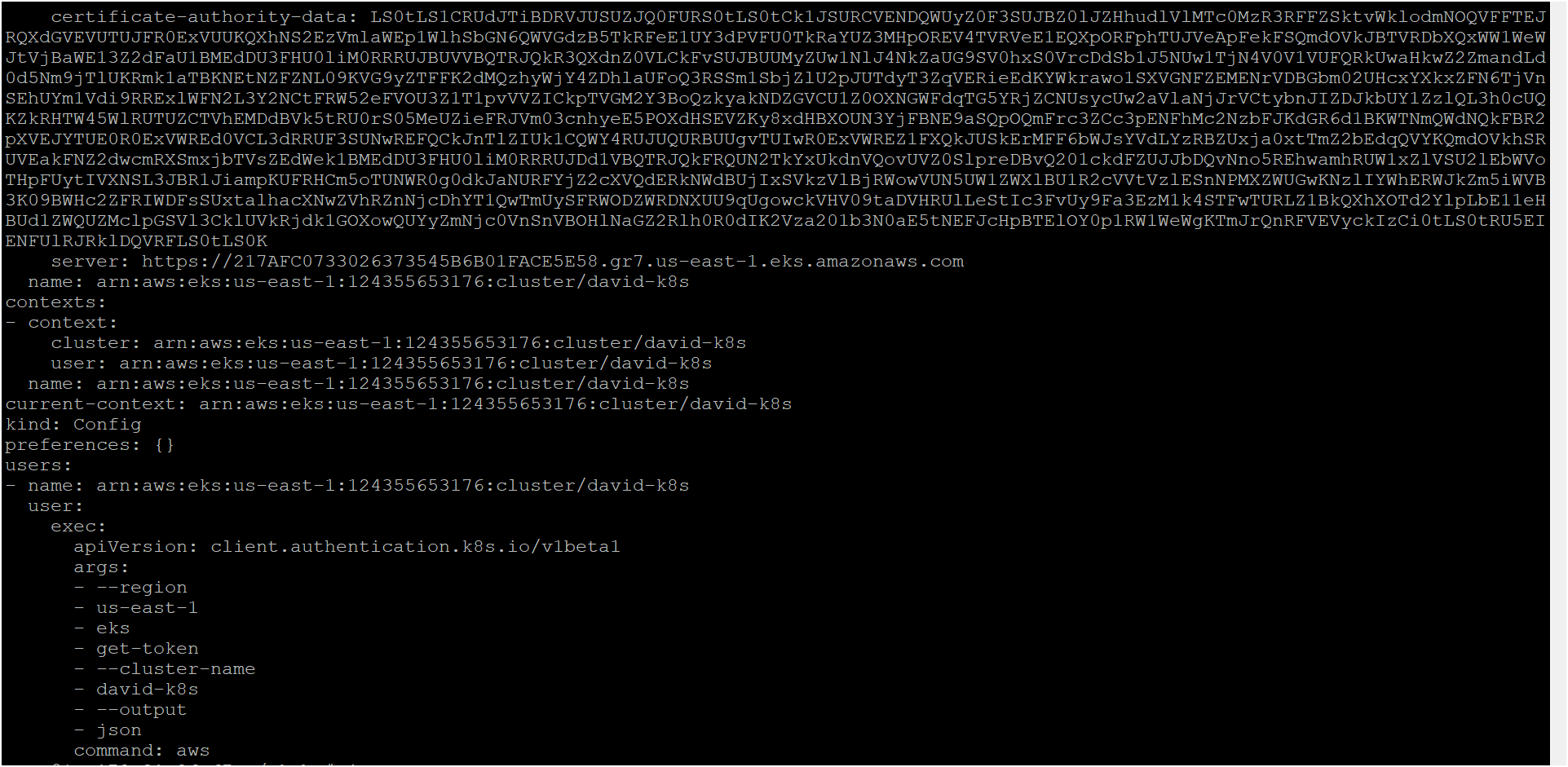
cd /root/.kube

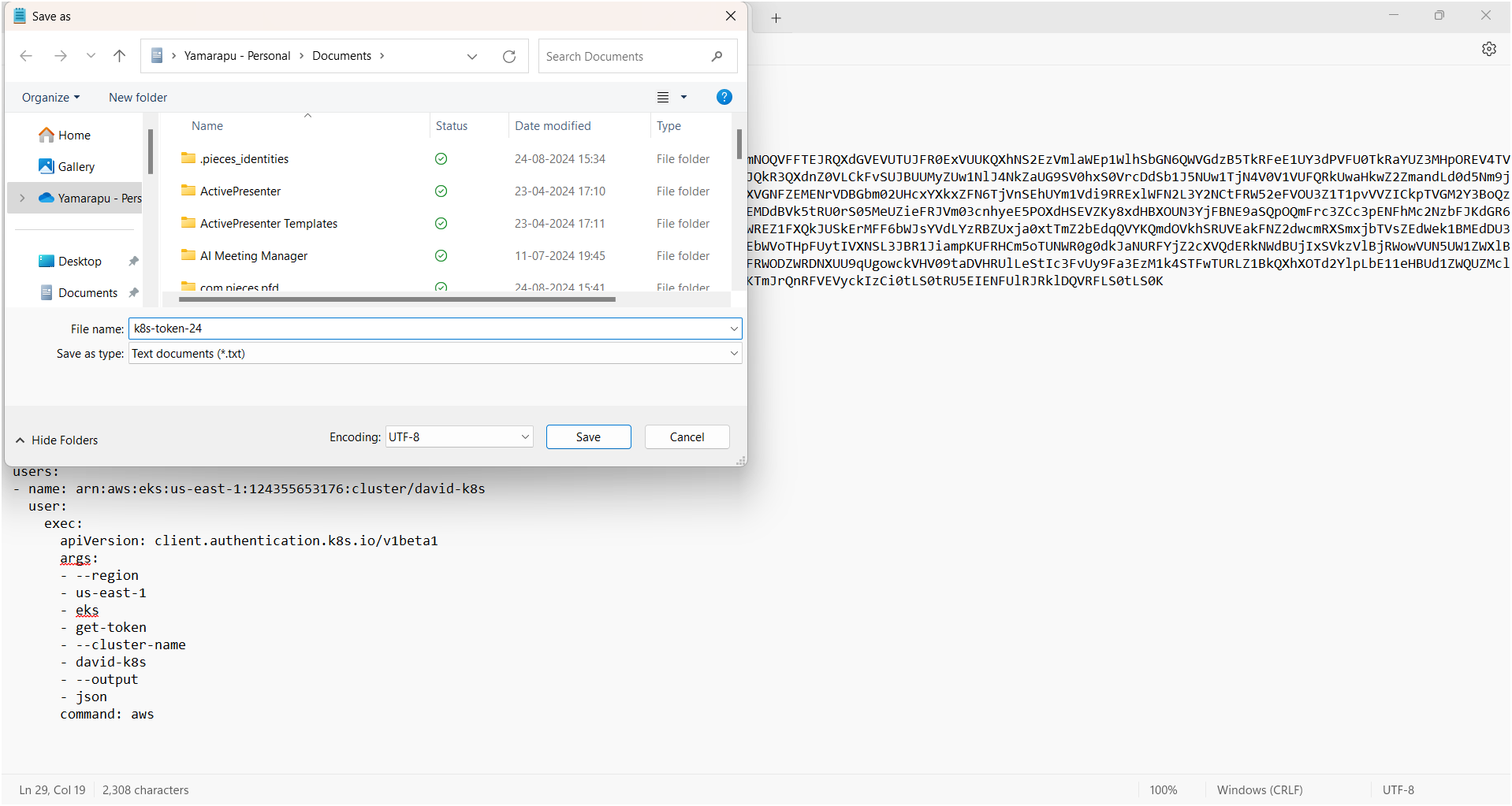
ls

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

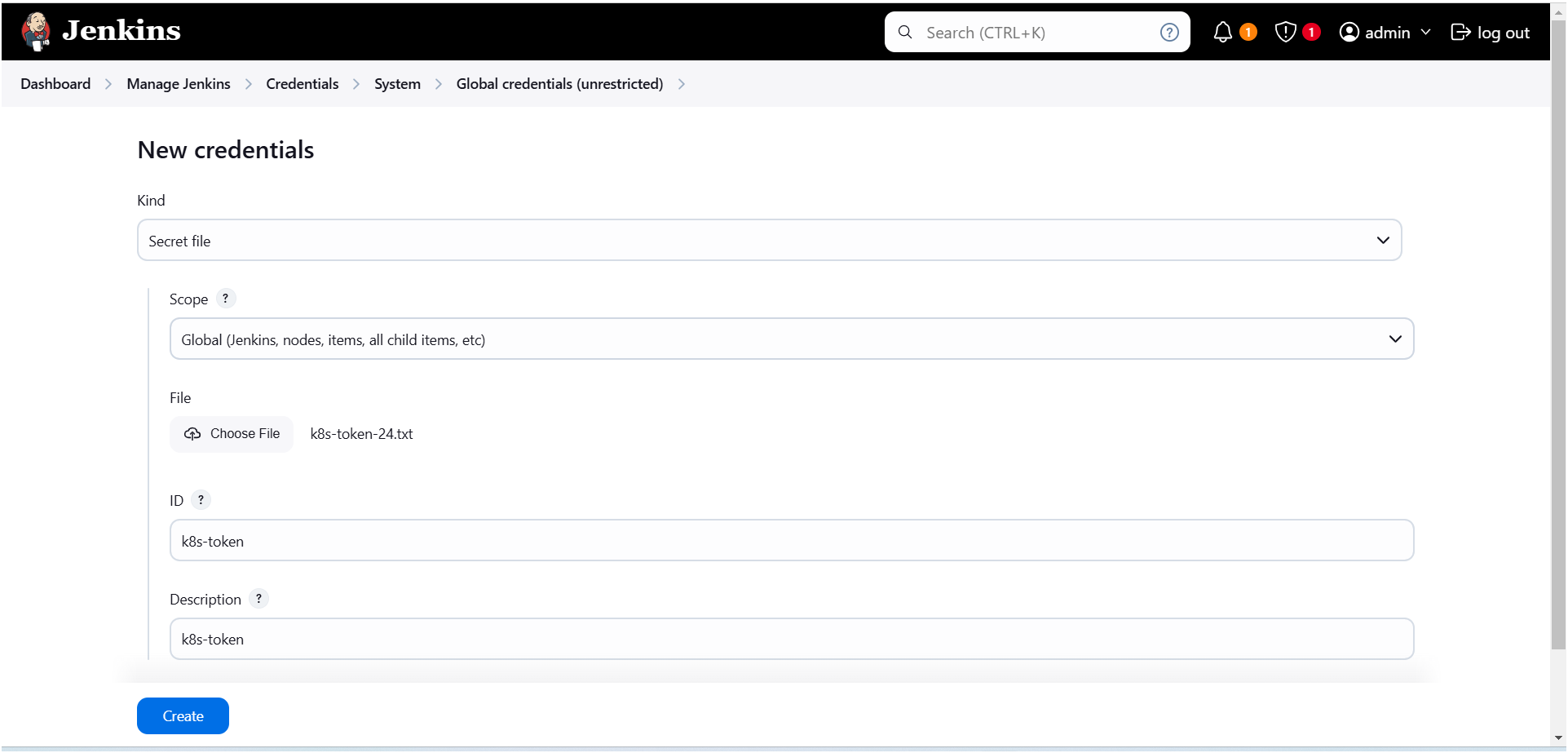
* Please make sure you have to copy this config file from Jenkins server only not from cluster.



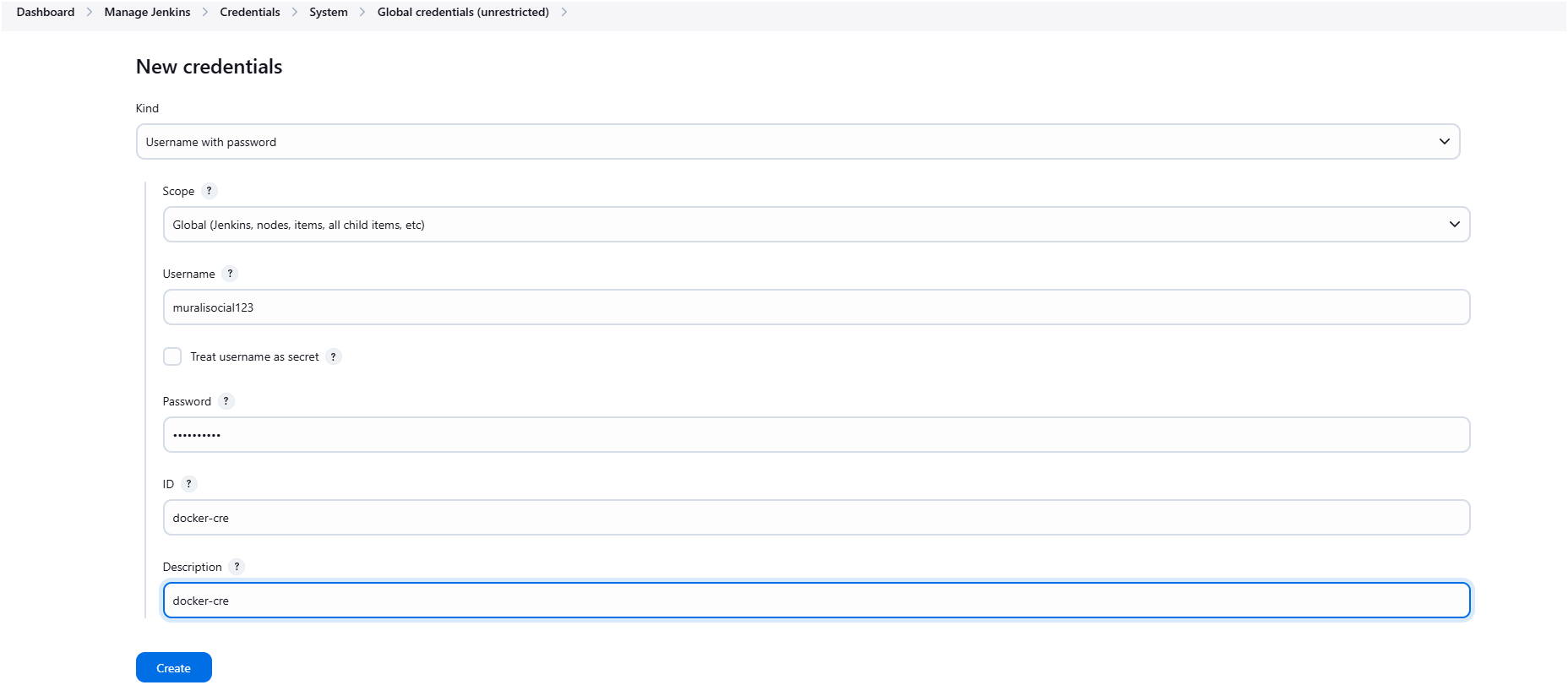




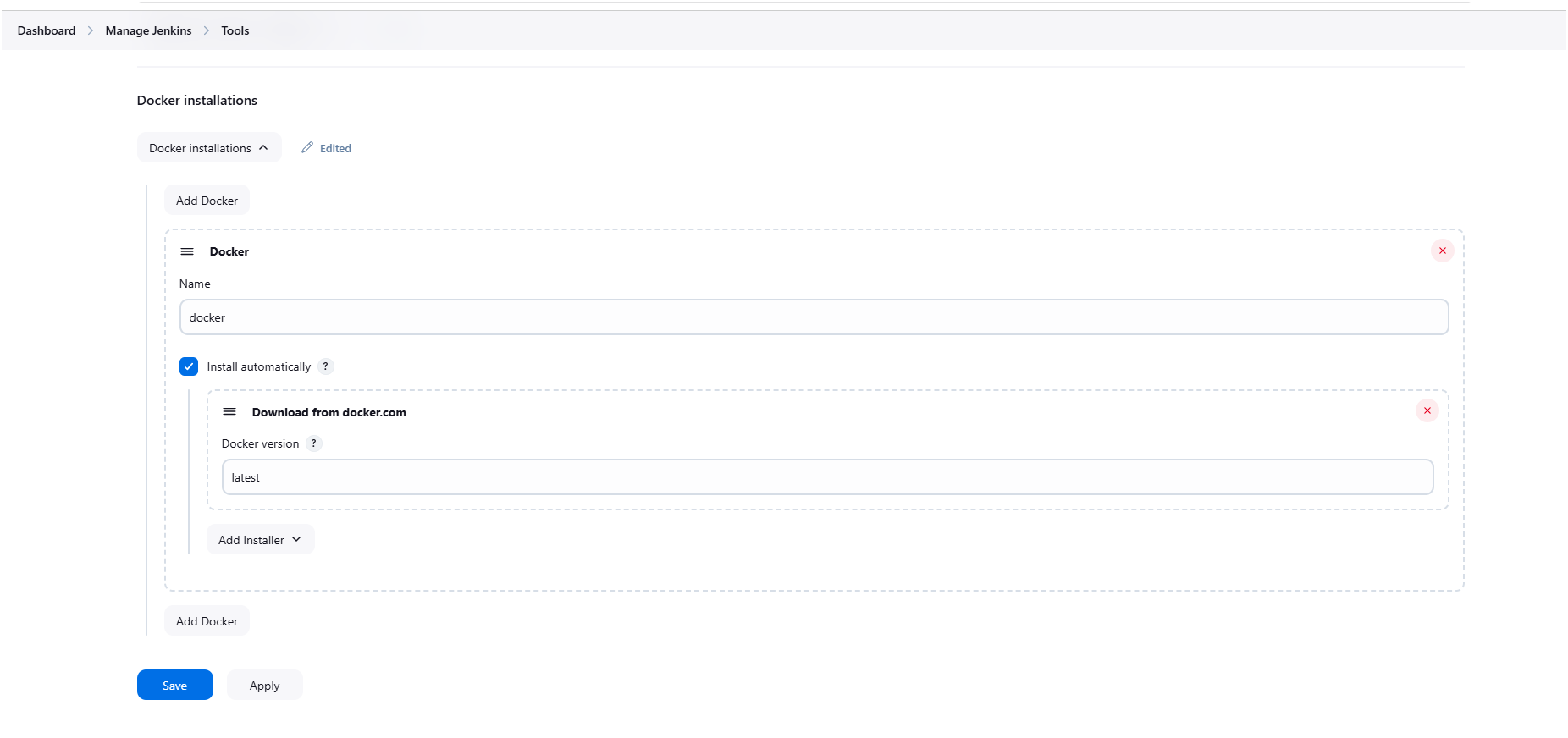
* Now we have to provide this file in Jenkins web server in credentials as follows



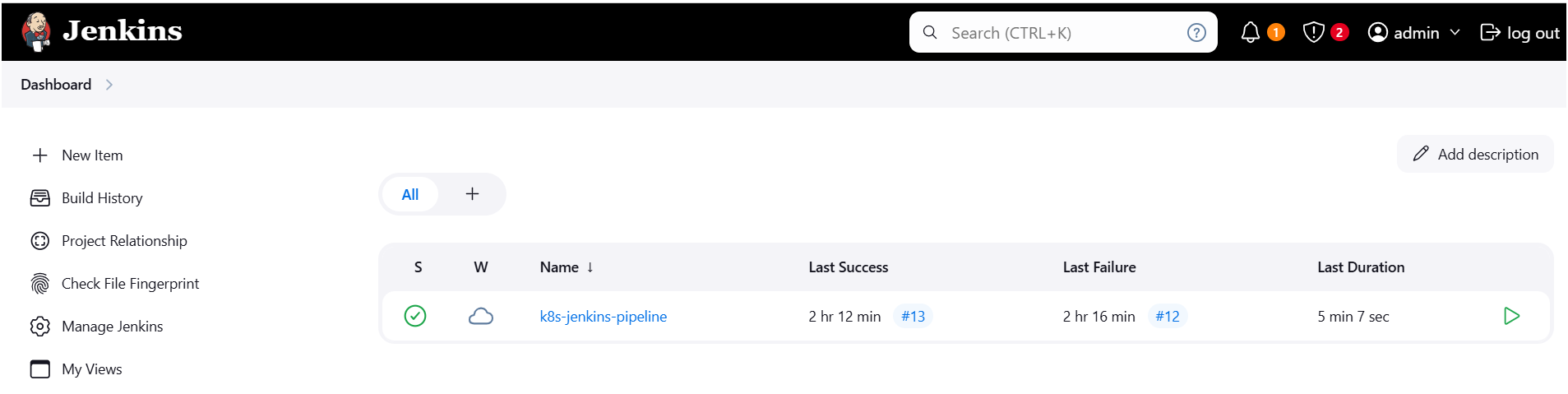
* Provide docker credentials as shown below image 👇



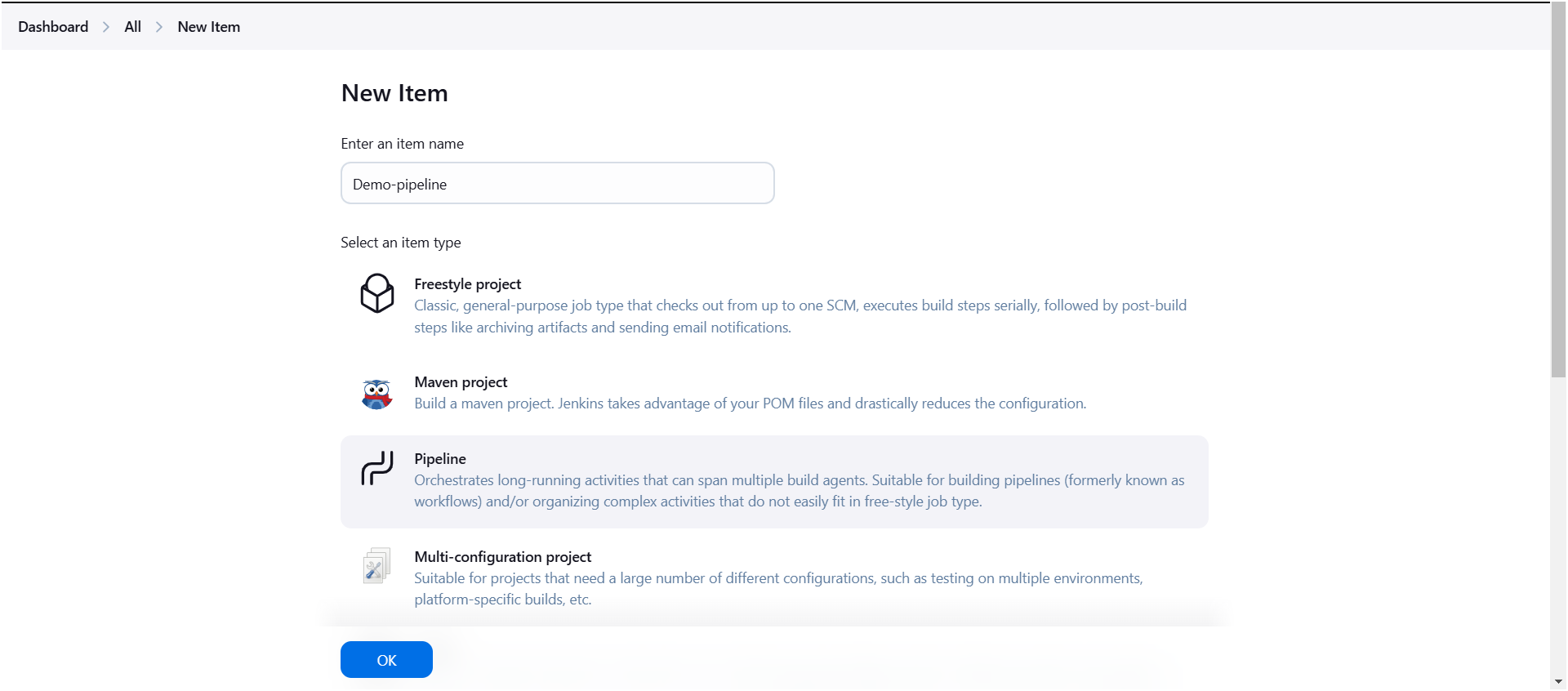
* Configure docker in tools as shown below image 👇



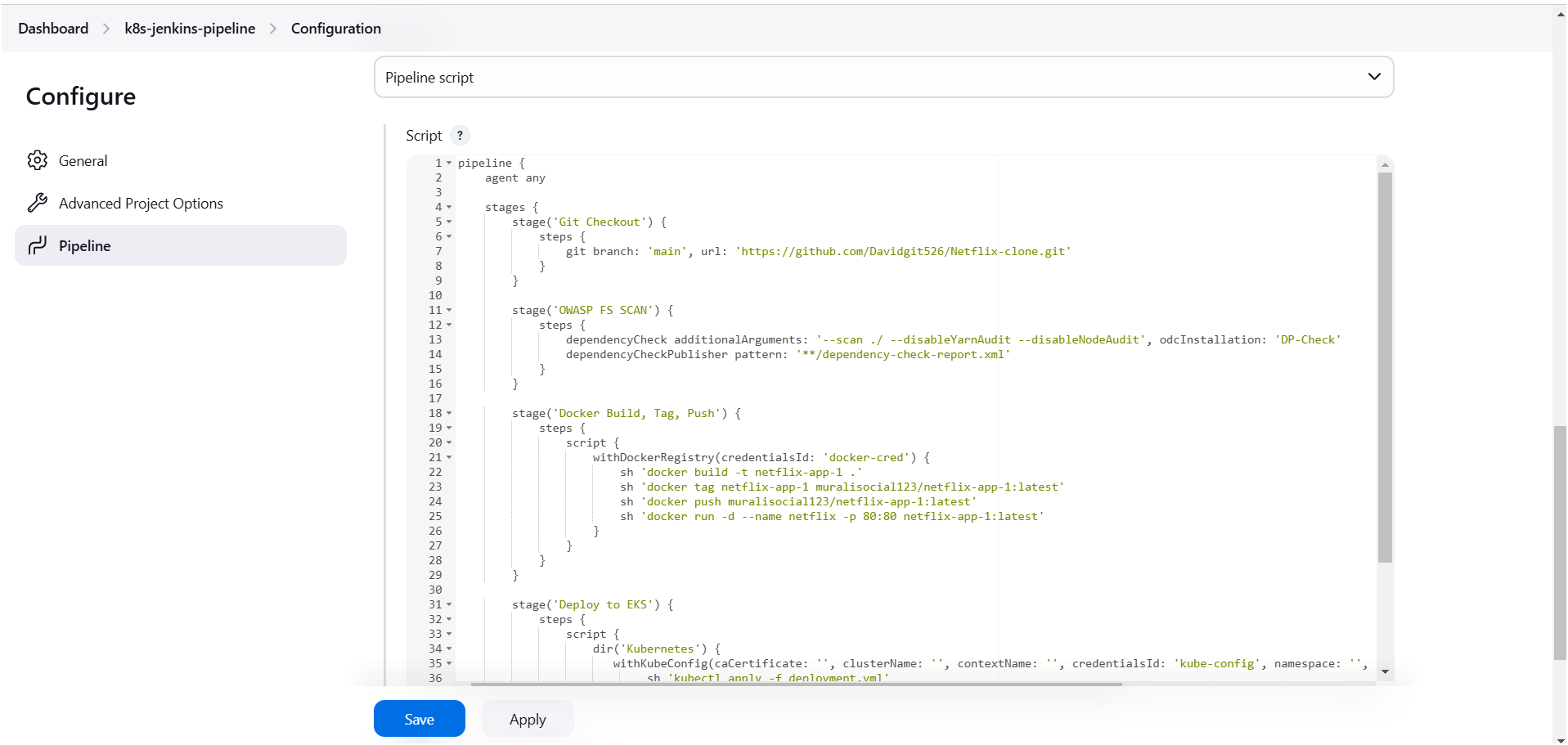
* Create a new item (pipeline) as shown below image 👇



* 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 'https://github.com/Msocial123/Retail-App\_kubernetes.git'

}

}

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

steps {

script {

withDockerRegistry(credentialsId: 'docker-cred') {

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

sh 'docker tag retail-app-1 muralisocial123/ retail-app-1 latest'

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

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

sh ‘docker compose up -d’

}

}

}

}

stage('Deploy to EKS') {

steps {

script {

withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s-token', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

sh 'kubectl apply -f mongodb-deployment.yml'

sh 'kubectl apply -f mongodb-service.yml'

sh 'kubectl apply -f userprofile-deployment.yml'

sh 'kubectl apply -f usernode-js-service.yml'

}

}

}

}

}

}

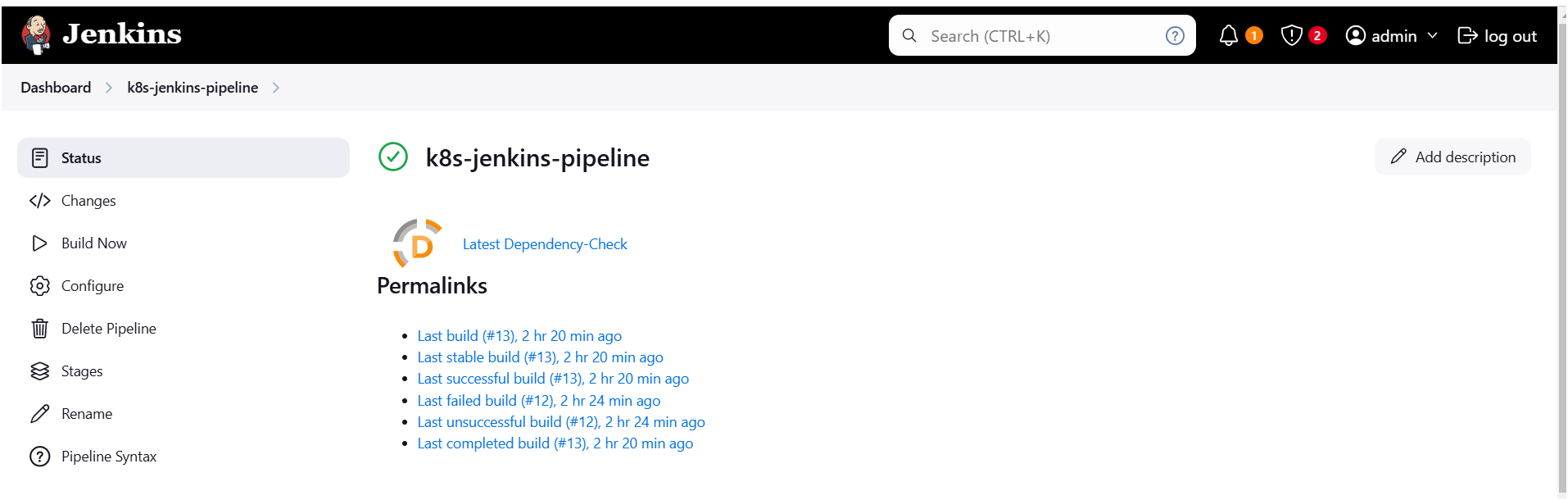
**Note:** before deploying app on k8s cluster must delete containers (if u get any errors execute following commands on jenkins server)

sudo chown -R jenkins:jenkins /var/lib/jenkins/workspace/Retail-app

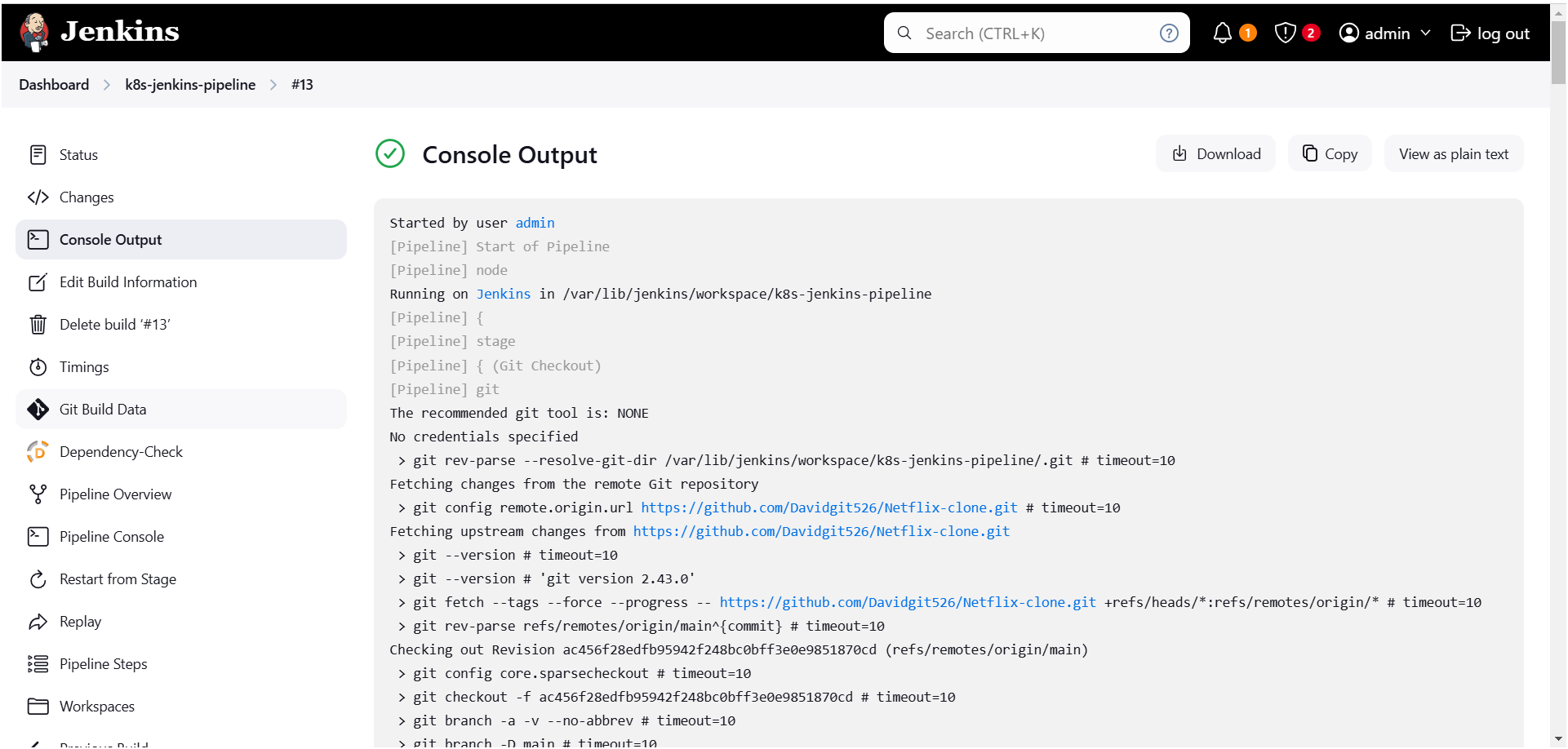
sudo chmod -R 777 /var/lib/jenkins/workspace/Retail-app

sudo rm -rf /var/lib/jenkins/workspace/Retail-app/data

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

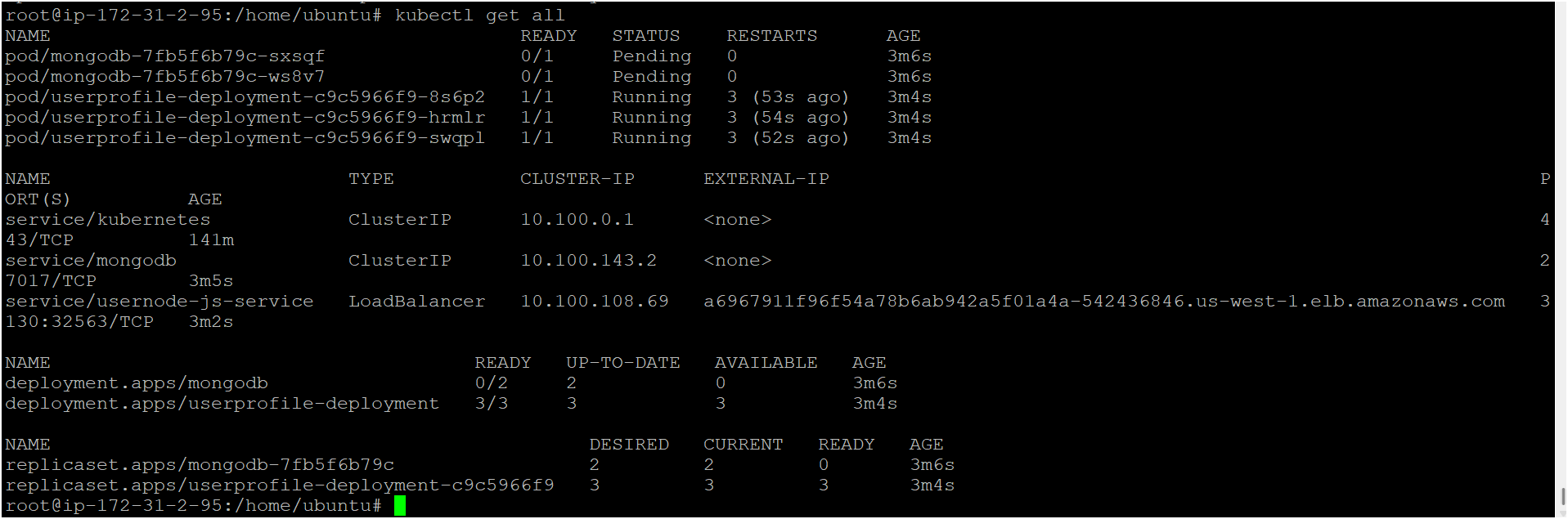


* Open console output by clicking on build number 👇





* Go to your cluster check application has deployed or not 👇
* My Retail-app-kubernetes application successfully has deployed on eks cluster



* Am able to access my application

