**Orbit Bank**

**Devops Capstone Project 2**

**Description**

Orbit Bank is one of the leading banking and financial service providers and is facing challenges in managing their monolithic applications and experiencing downtime during deployment. The company needs to develop an online banking application that provides private banks with a global accounting foundation, offering electronic banking services to all private banks, and enable private bank clients to carry out their daily transactions.

To address these issues, the company has decided to transition to a microservices architecture and implement a DevOps pipeline workflow using Jenkins, Ansible playbook, and Kubernetes cluster to deploy container on Docker Hub.

**Task (Activities)**

1. Create the Dockerfile, Jenkinsfile, Ansible playbook, and the source file of the static website

and upload it on the GitHub repository.

2. Create Jenkins pipeline to perform continuous integration and deployment for a Docker

container

3. Set up Docker Hub

4. Set up Kubernetes cluster and configure deployment stage in the pipeline

5. Configure Ansible playbook to deploy container on Docker Host

6. Execute Jenkins build

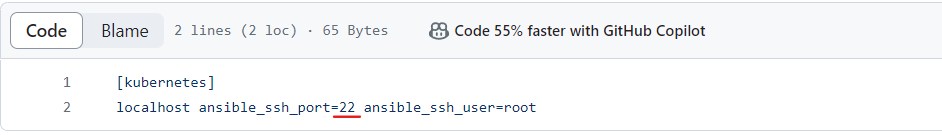
7. Access deployed application on a Docker container

**Step 1 : Update the Inventory.Yaml and Jenkinsfile**

1. Update the Jenkinsfile with the following code and commit changes

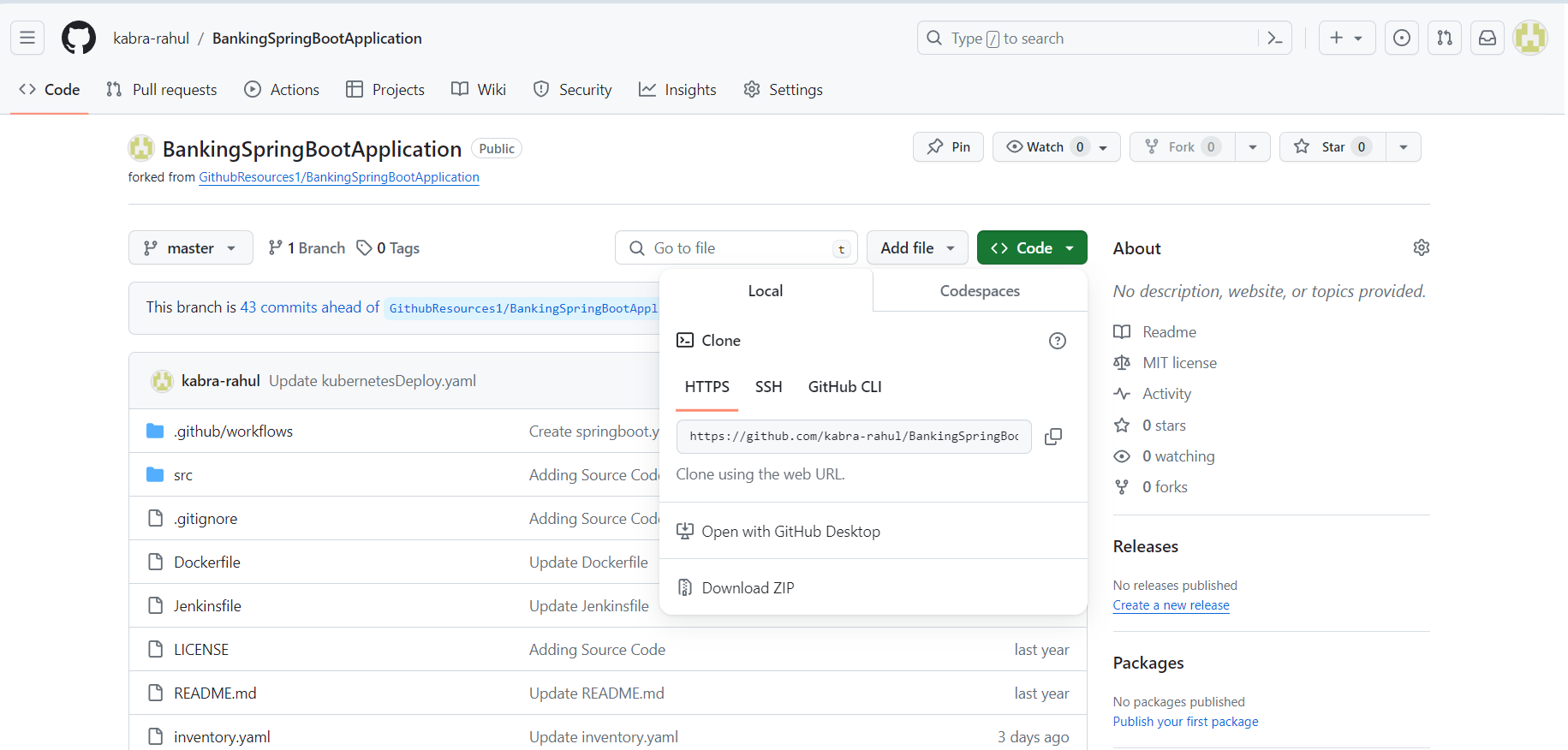


1. Update the inventory with with following code and commit the changes



1. Copy the project link

[**https://github.com/kabra-rahul/BankingSpringBootApplication.git**](https://github.com/kabra-rahul/BankingSpringBootApplication.git)



**Step 2 : Create an EC2 Instance and IAM user**

2.1 Launch and EC2 instance with the following configuration

**Name – devops-server**

**Number of Instance -1**

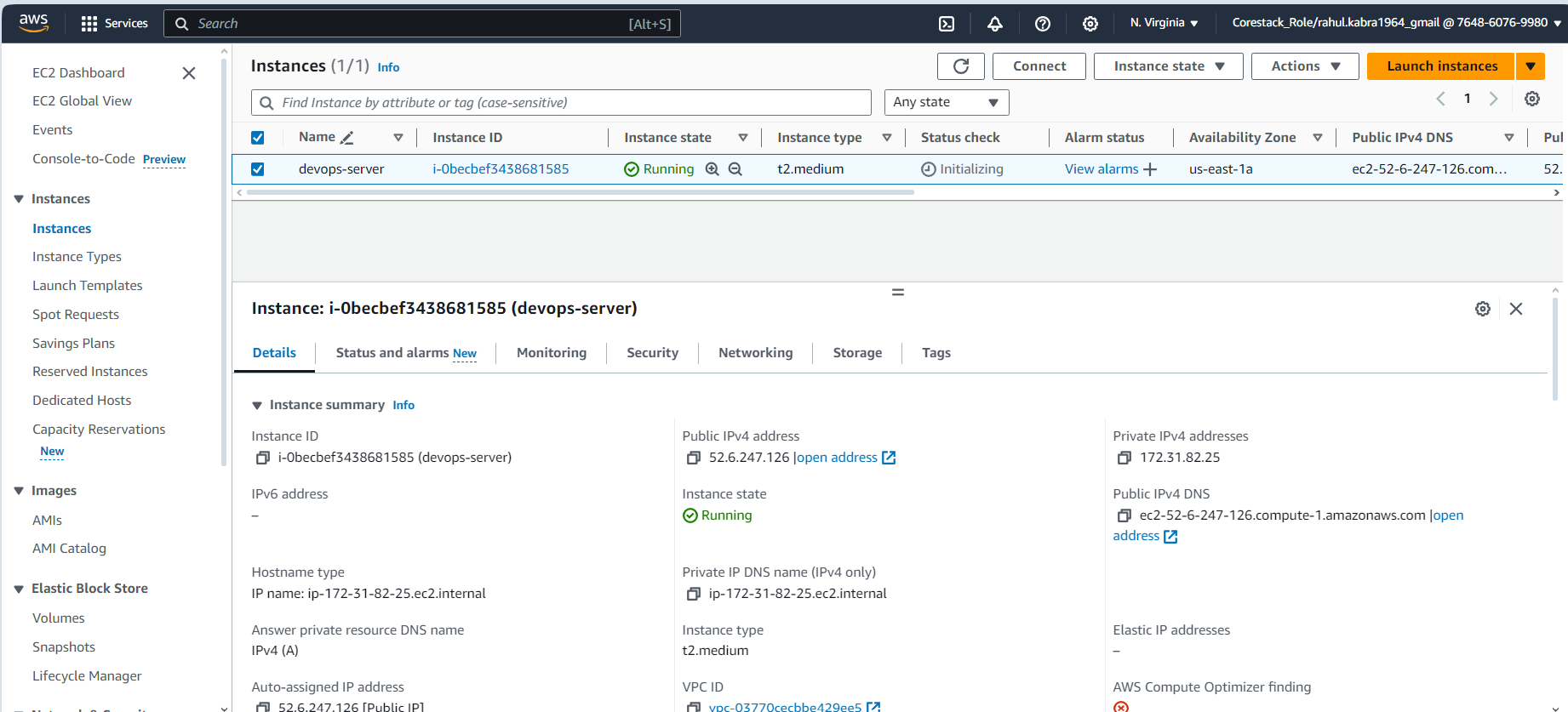
**AMI - Amazon Linux 2 Kernel 5.10 AMI 2.0.20240223.0 x86\_64 HVM gp2**

**Instance Type – t2.medium**

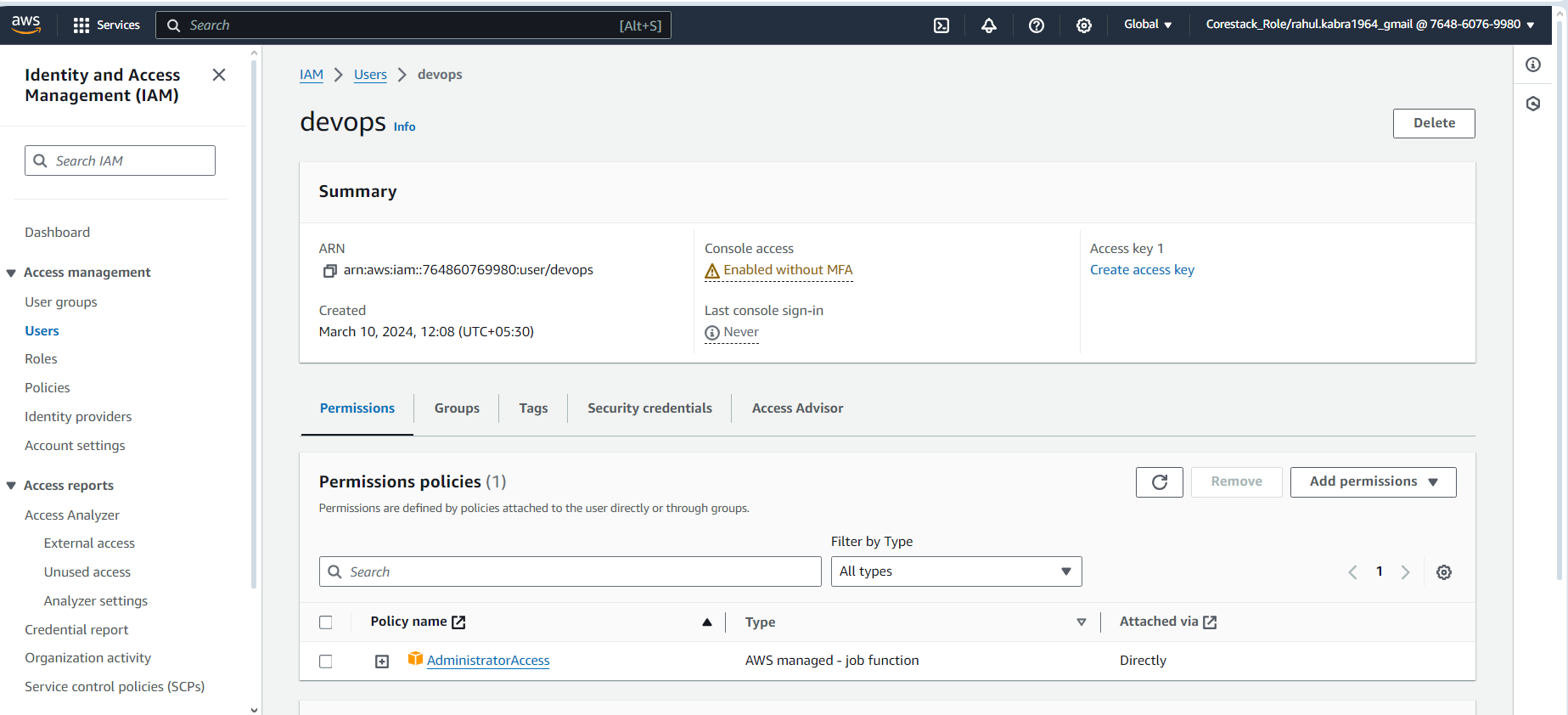
**Key pair – devops.pem**

**Select Create Security Group –> Select Allow SSH Traffic and HTTP traffic from**

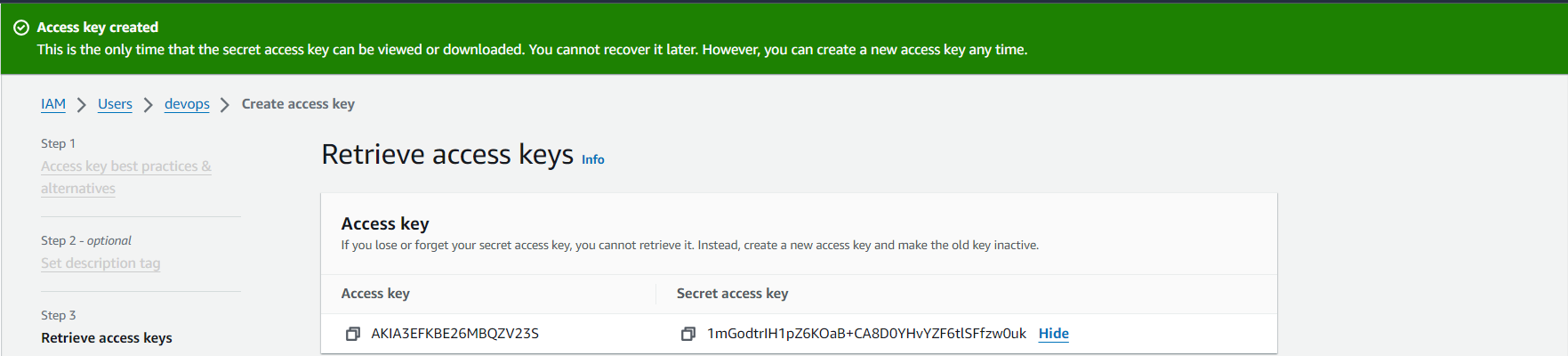
**Add Inbound Security Rule -> All Traffic (0.0.0.0/0)**

****

2.2 Create a new IAM user devops with Administrator role

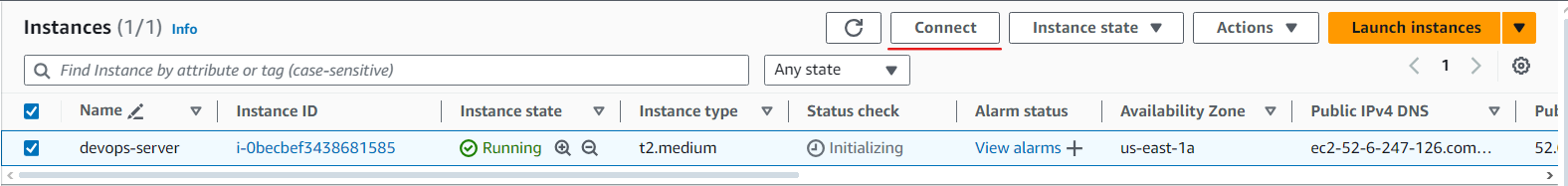


2.3 Create the access key and secret access key and save them

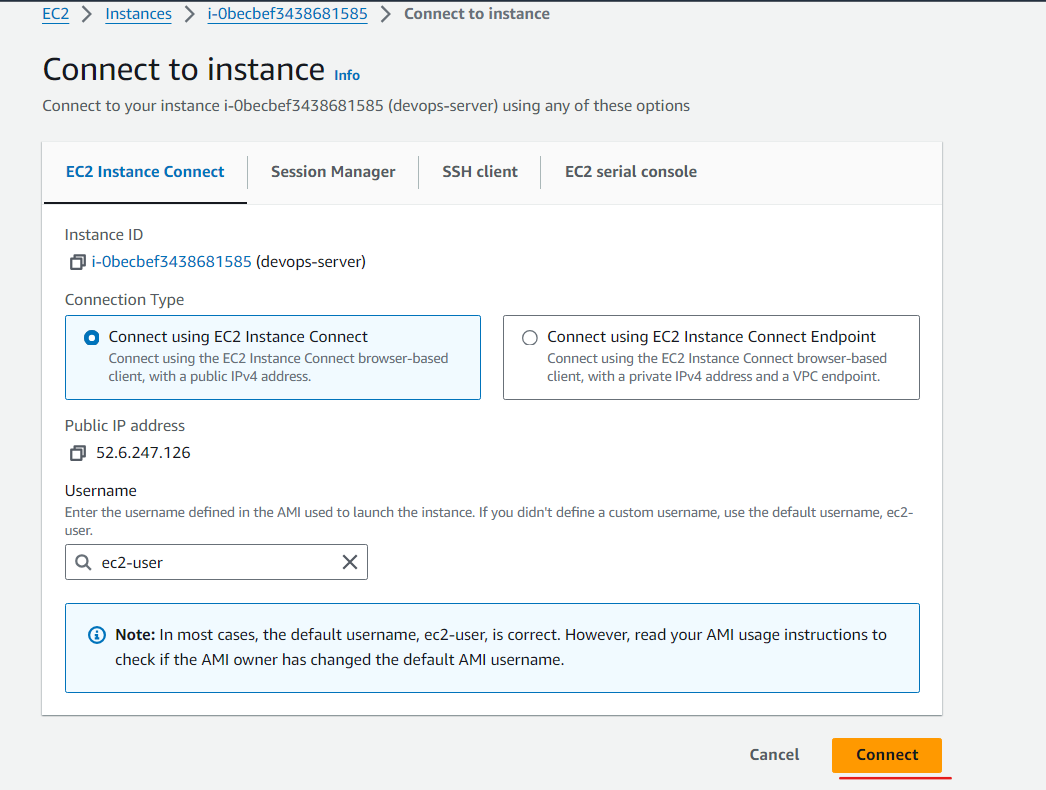


**Step 3 : Connecting to EC2 Machine and Installing package**

3.1 Go to EC2 instance home and select devops-server and click on **Connect**



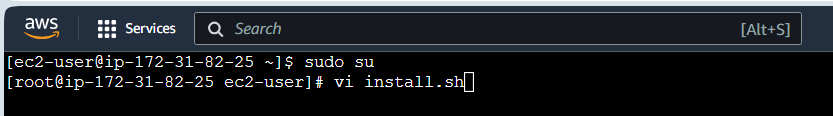
3.2 Click on **Connect**



3.3 Run the below commands in the terminal and create a file install.sh

**sudo su**

**vi install.sh**

****

3.4 Type the below command in the install.sh

**#!/bin/bash**

**#Installing Java and Jenkins**

**sudo yum update –y**

**sudo amazon-linux-extras install java-openjdk11 -y**

**sudo wget -O /etc/yum.repos.d/jenkins.repo \https://pkg.jenkins.io/redhat-stable/jenkins.repo**

**sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key**

**sudo yum upgrade**

**sudo yum install jenkins -y**

**sudo systemctl enable jenkins**

**sudo systemctl start jenkins**

**#Installing NodeJs**

**curl -fsSL https://rpm.nodesource.com/setup\_16.x | sudo bash -**

**sudo yum install -y nodejs**

**#Installing Maven**

**sudo yum install -y maven**

**#Installing Docker**

**sudo yum install -y docker**

**sudo systemctl start docker**

**sudo systemctl enable docker**

**#Installing python and Ansible**

**sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm -y**

**sudo yum update -y**

**sudo yum install git python python-level python-pip openssl ansible -y**

**#Installing Terraform**

**sudo yum install -y yum-utils shadow-utils**

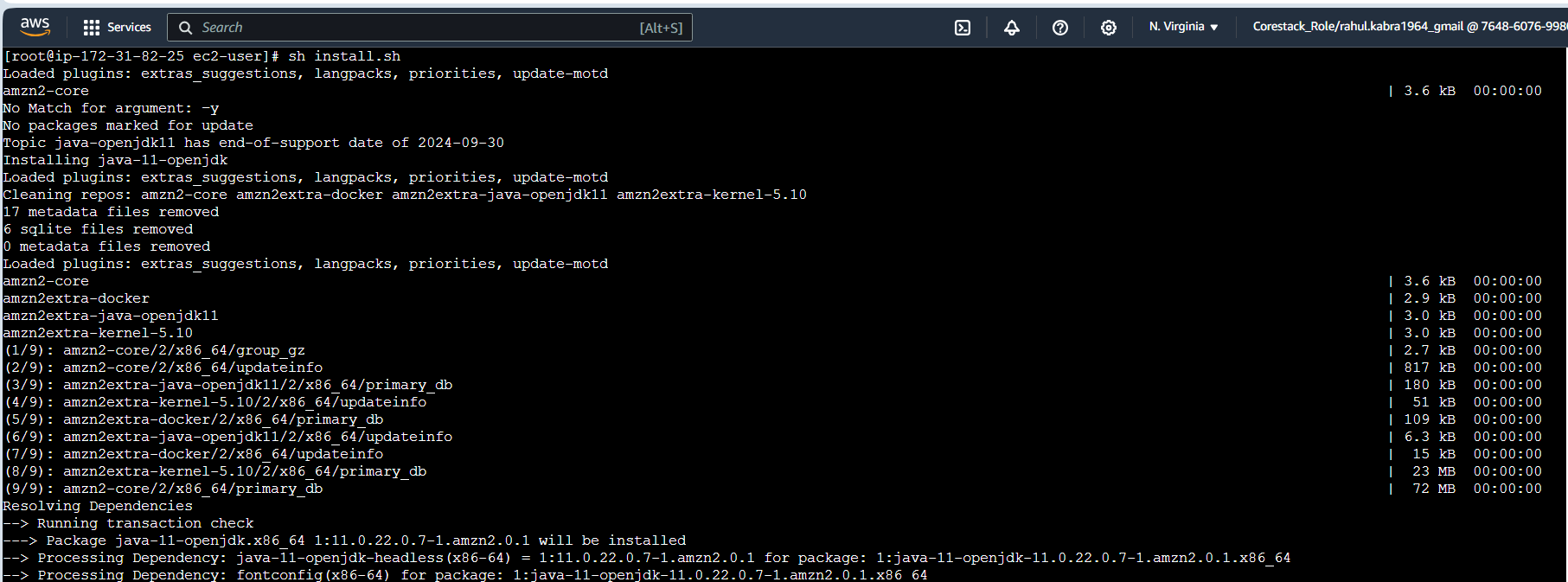
**sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo**

**sudo yum -y install terraform**



3.5 Run below command to execute

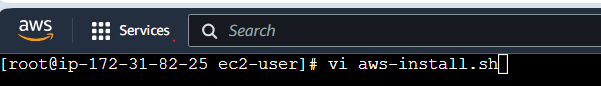
**sh install.sh**

****

****

3.6 Create file install aws-install.sh

**vi aws-install.sh**

****

3.7 Type the below command in the aws-install.sh

**#!/bin/bash**

**# Check if the required parameters are provided**

**if [ $# -lt 3 ]; then**

**echo "Usage: $0 <access\_key> <secret\_key> <region>"**

**exit 1**

**fi**

**#Install asw cli**

**sudo yum install awscli**

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

**sudo yum install unzip**

**unzip awscliv2.zip**

**sudo ./aws/install --update**

**# Access and use the runtime parameters**

**access\_key=$1**

**secret\_key=$2**

**region=$3**

**# Set AWS credentials and default region**

**aws configure set aws\_access\_key\_id $access\_key**

**aws configure set aws\_secret\_access\_key $secret\_key**

**aws configure set default.region $region**

**#Install Kubectl**

**curl -LO "https://dl.k8s.io/release/v1.23.6/bin/linux/amd64/kubectl"**

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

**chmod +x kubectl**

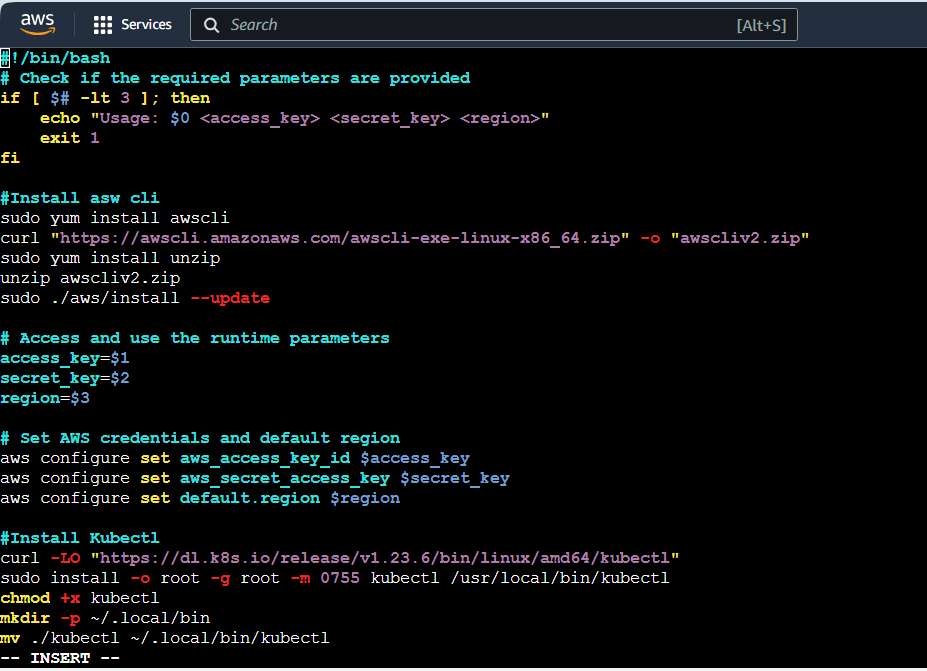
**mkdir -p ~/.local/bin**

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

**#Set env variable**

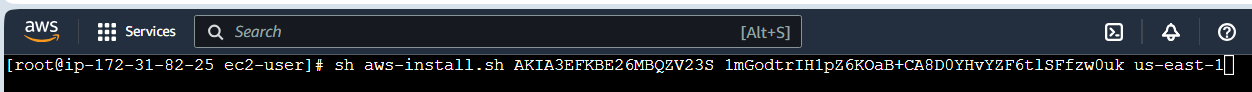
**export PATH="$PATH:~/.local/bin/"**

**echo "export PATH=$PATH:~/.local/bin/" >> ~/.bashrc**

****

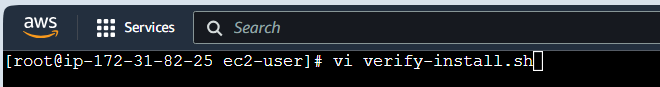
3.8 Run below command to execute

**sh aws-install.sh <access-key> <secret-access-key> us-east-1**

****

3.9 Create file install verify-install.sh

**vi verify-install.sh**

****

3.10 Type the below command in the verify-install.sh , to verify if installed correctly

**#!/bin/bash**

**#Verify installations**

**echo “Java version : “ && java -version**

**echo “======================================”**

**echo “Jenkins Status : “ && sudo systemctl status jenkins**

**echo “======================================”**

**echo “Docker version : “ && docker--version**

**echo “======================================”**

**echo “Maven version : “ && mvn --version**

**echo “======================================”**

**echo “Ansible version : “ && ansible --version**

**echo “======================================”**

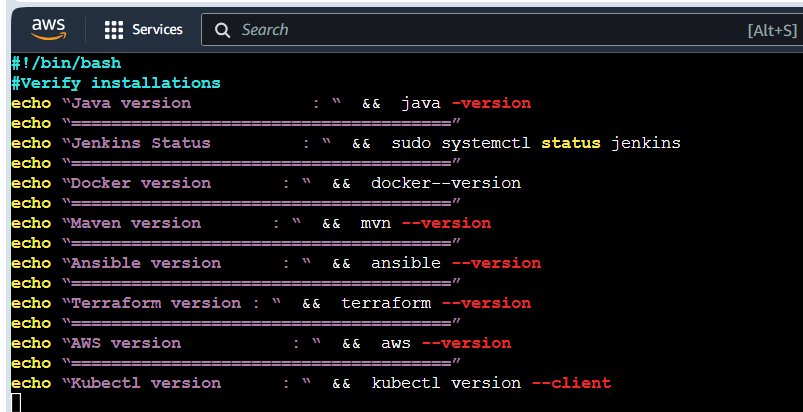
**echo “Terraform version : “ && terraform --version**

**echo “======================================”**

**echo “AWS version : “ && aws --version**

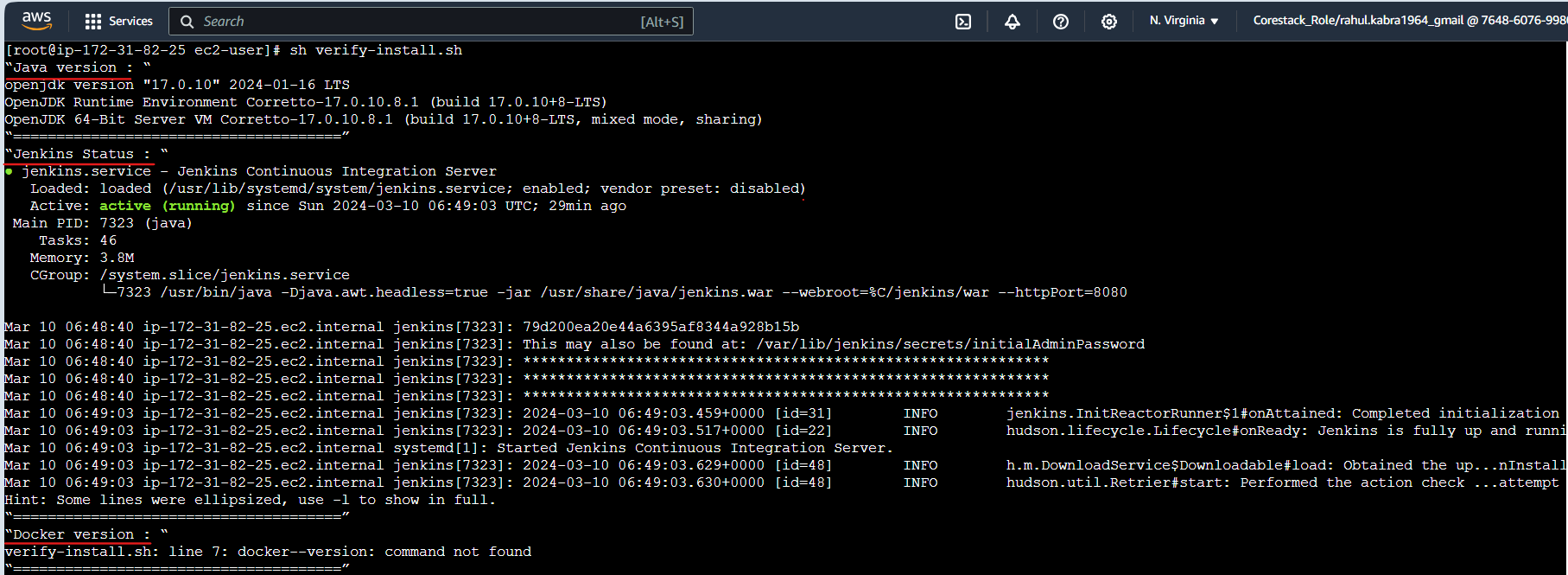
**echo “======================================”**

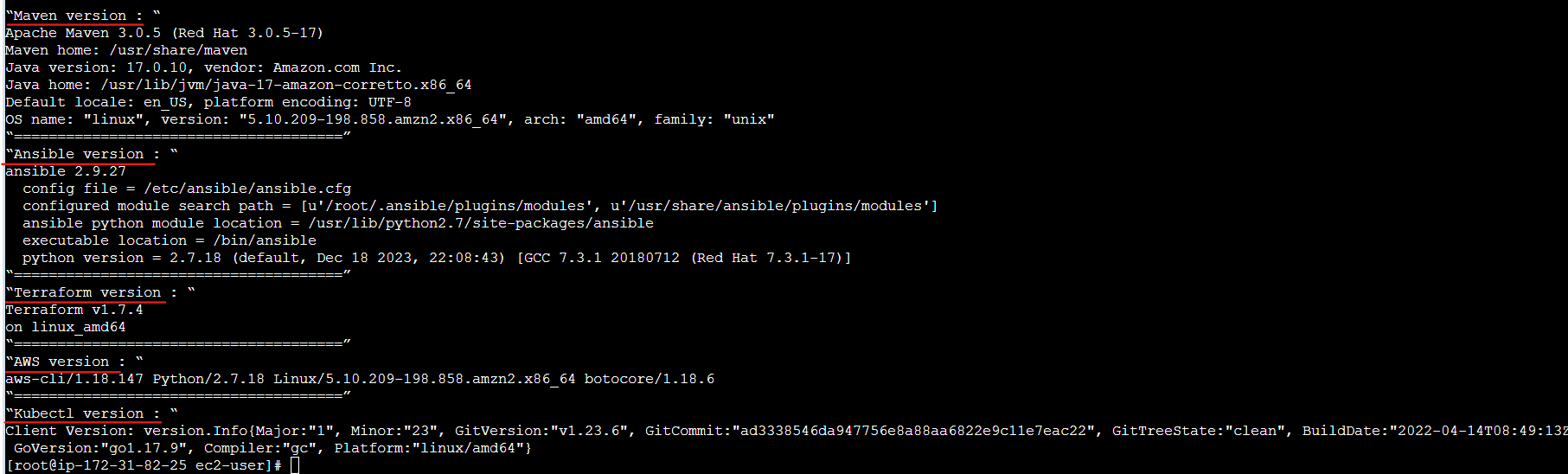
**echo “Kubectl version : “ && kubectl version --client**

****

3.11 Run below command to execute

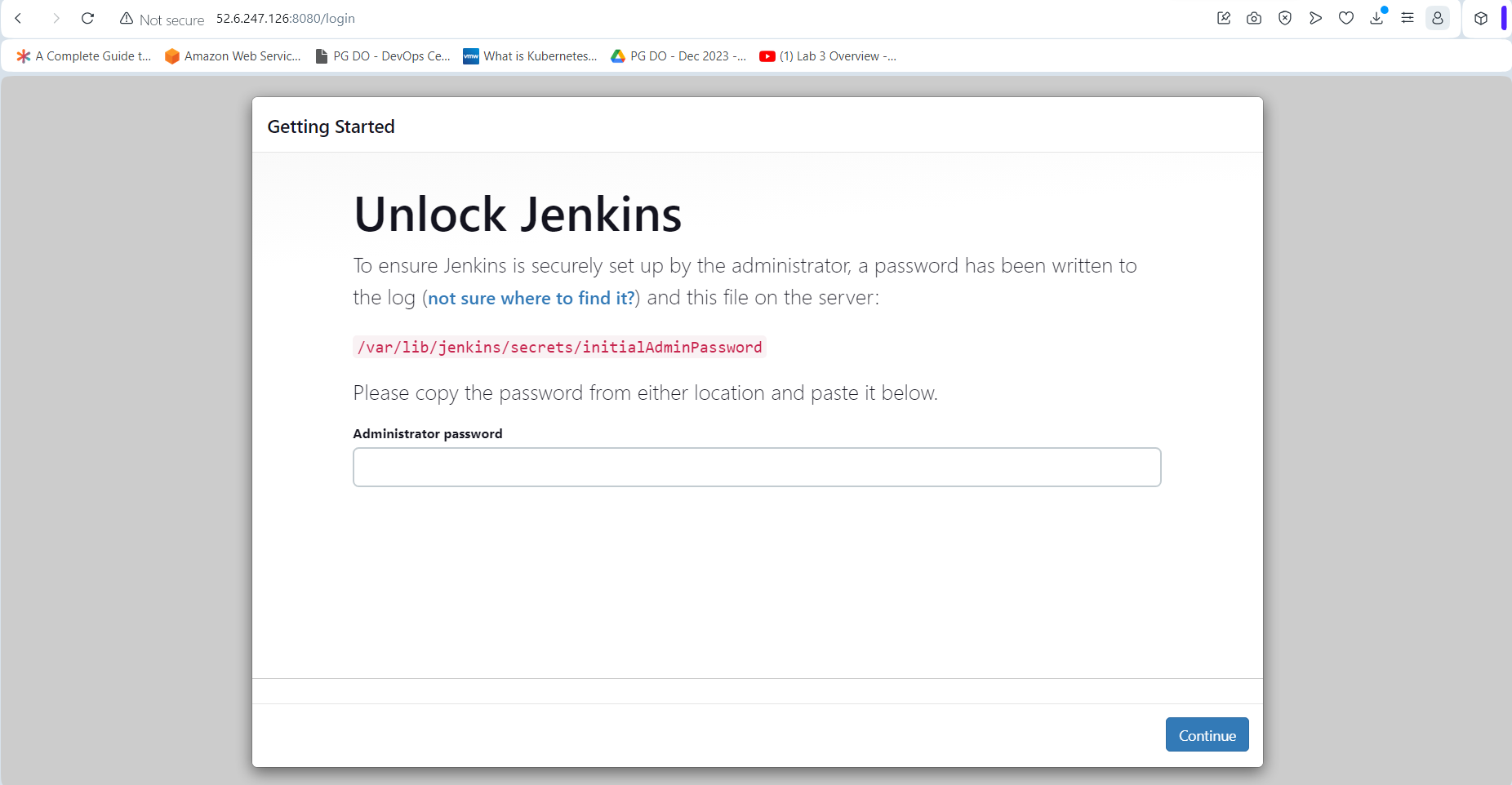
**sh verify-install.sh**

****

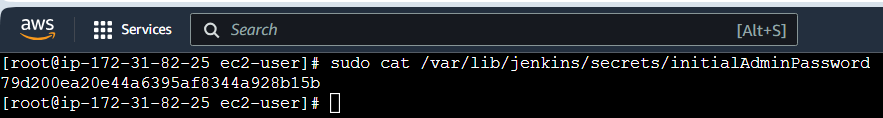
****

**Step 4 : Accessing Jenkins**

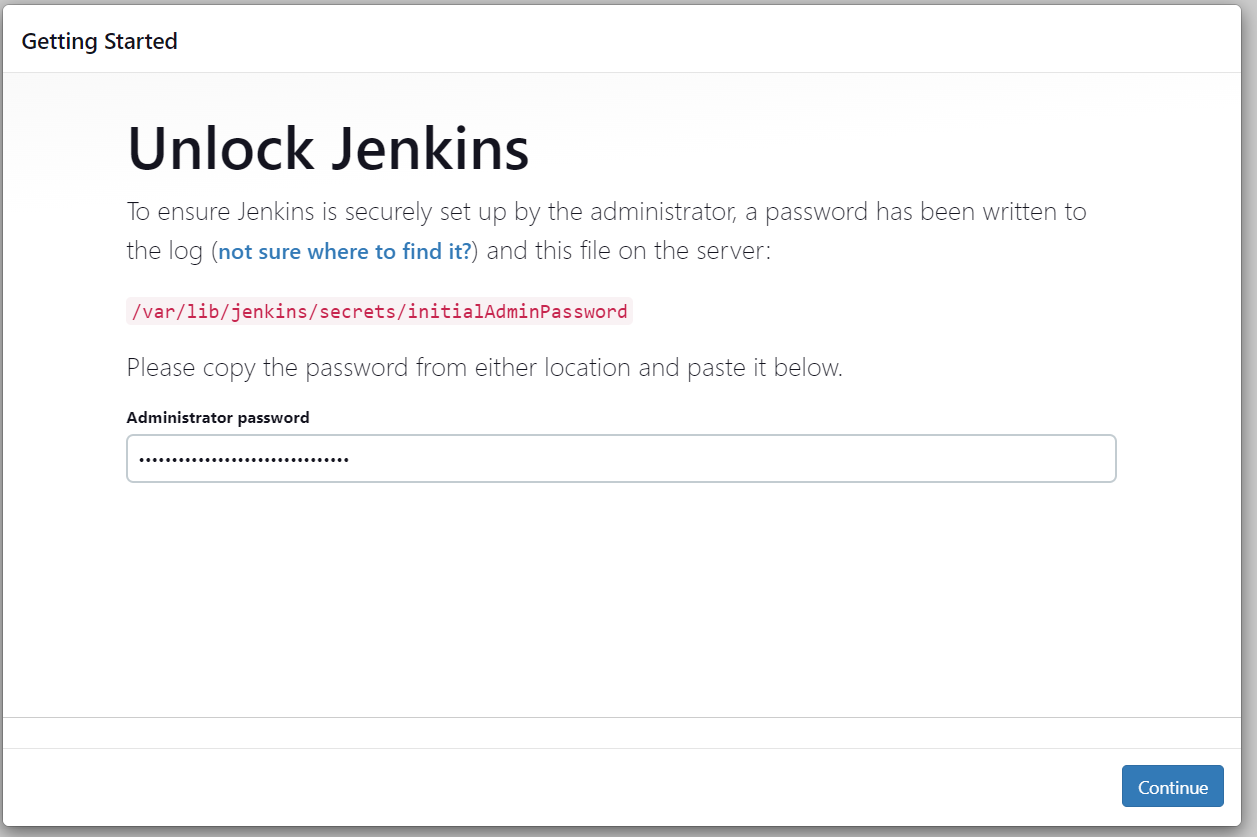
* 1. Open  [**http://52.6.247.126:8080**](http://3.88.187.77:8080/)**/** in the browser



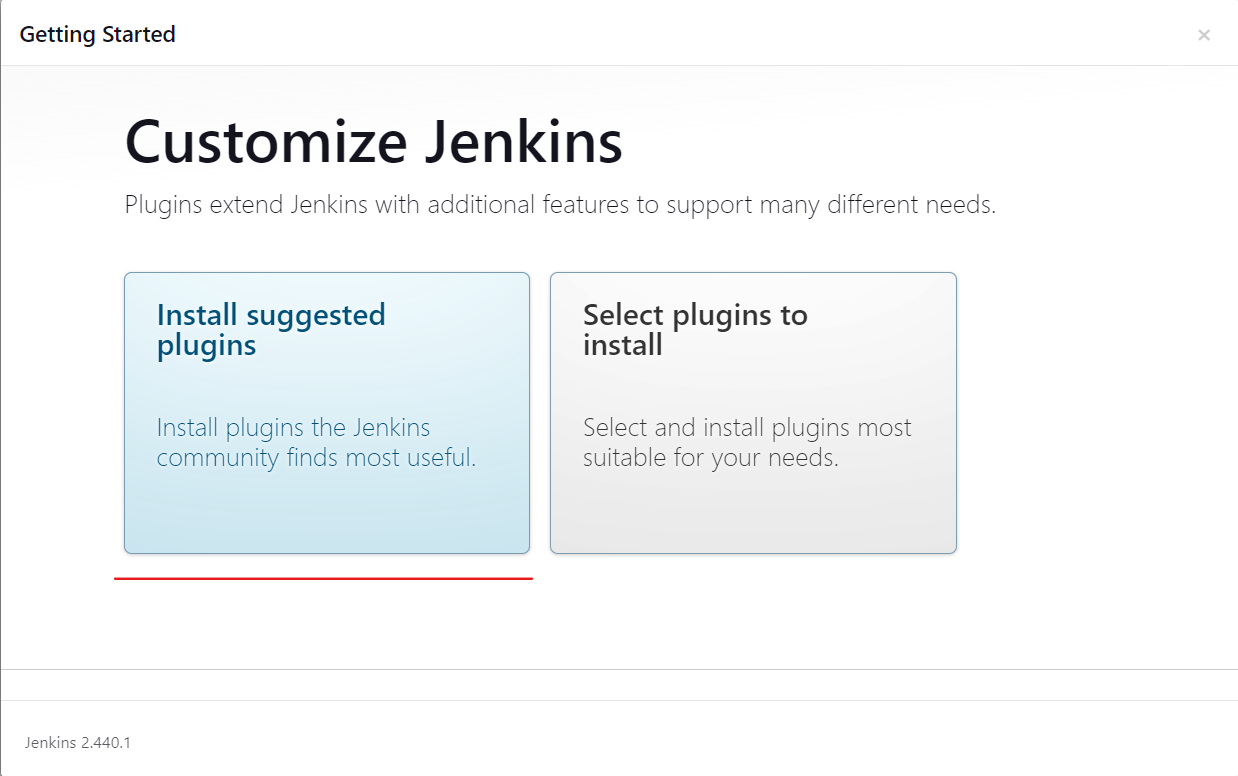
* 1. In your terminal run the following command: ***sudo cat /var/lib/jenkins/secrets/initialAdminPassword***

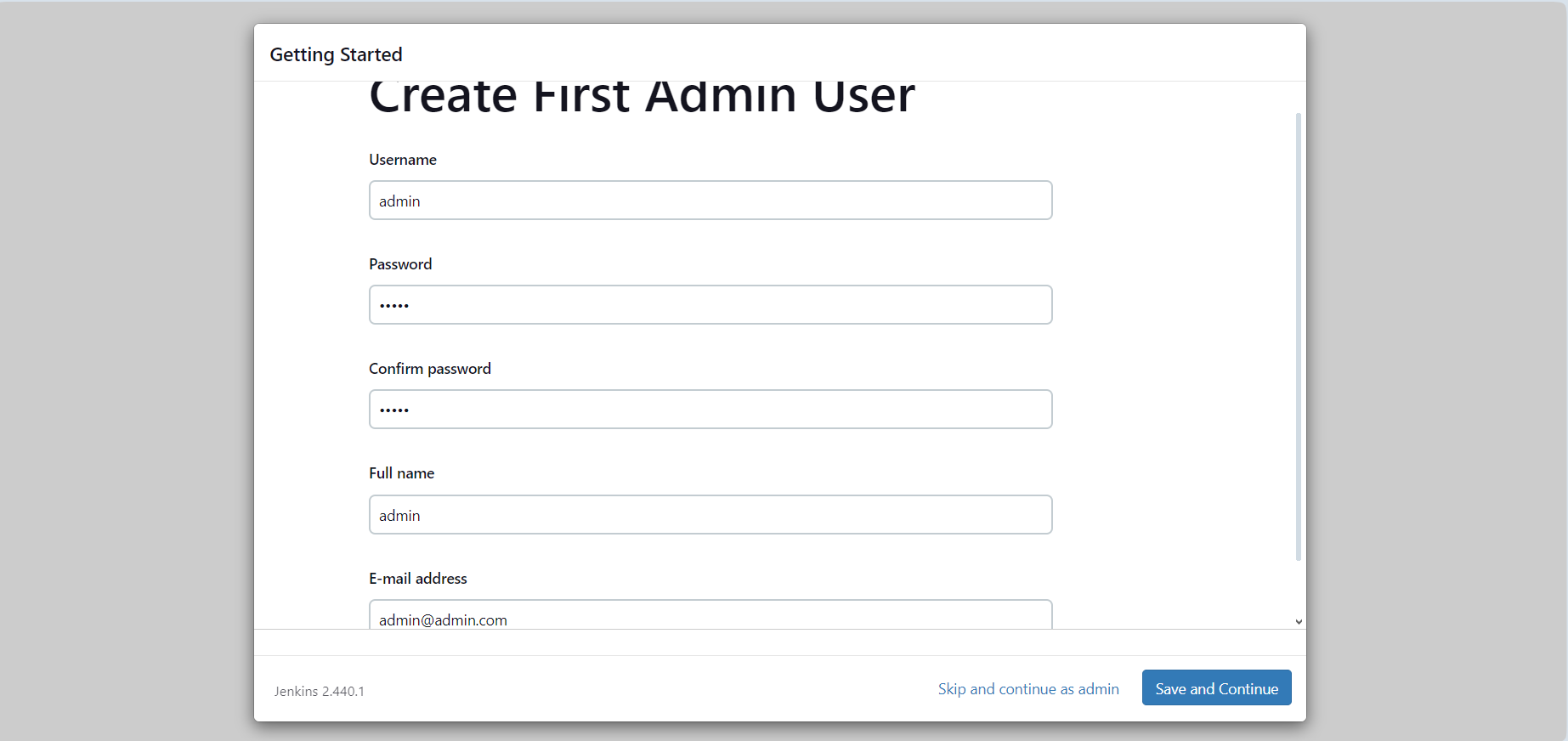


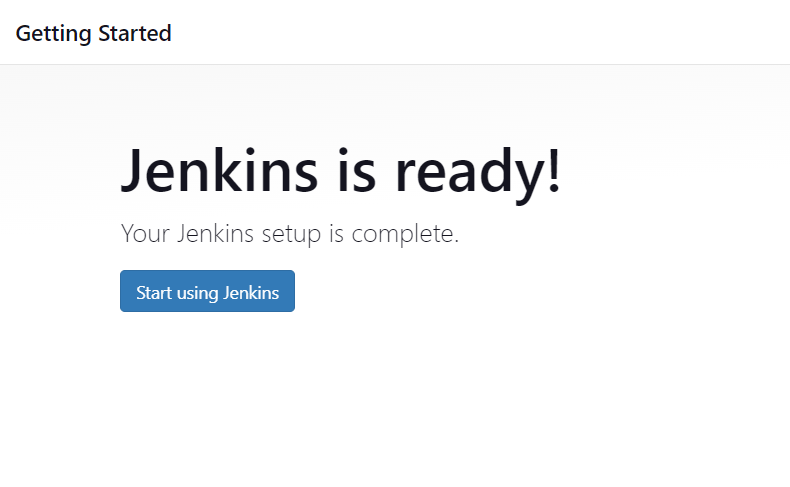
* 1. Copy this password and paste it your Jenkins page in the browser and click continue



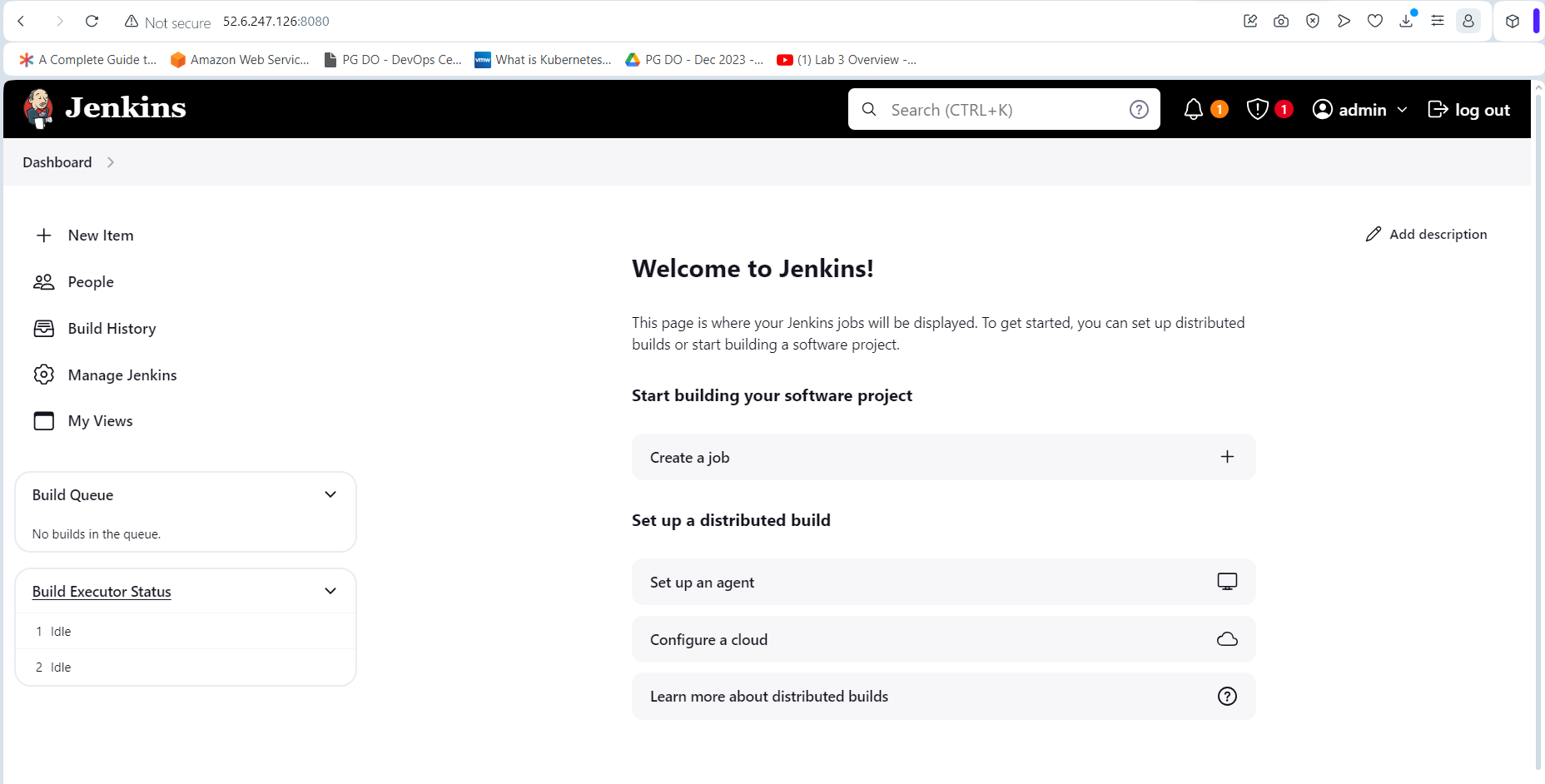
* 1. Now, click on I**nstall the suggested plugins**



* 1. You can create an admin user by filling below details
  2. In the Instance configuration page, click on the **Start using Jenkins** button.

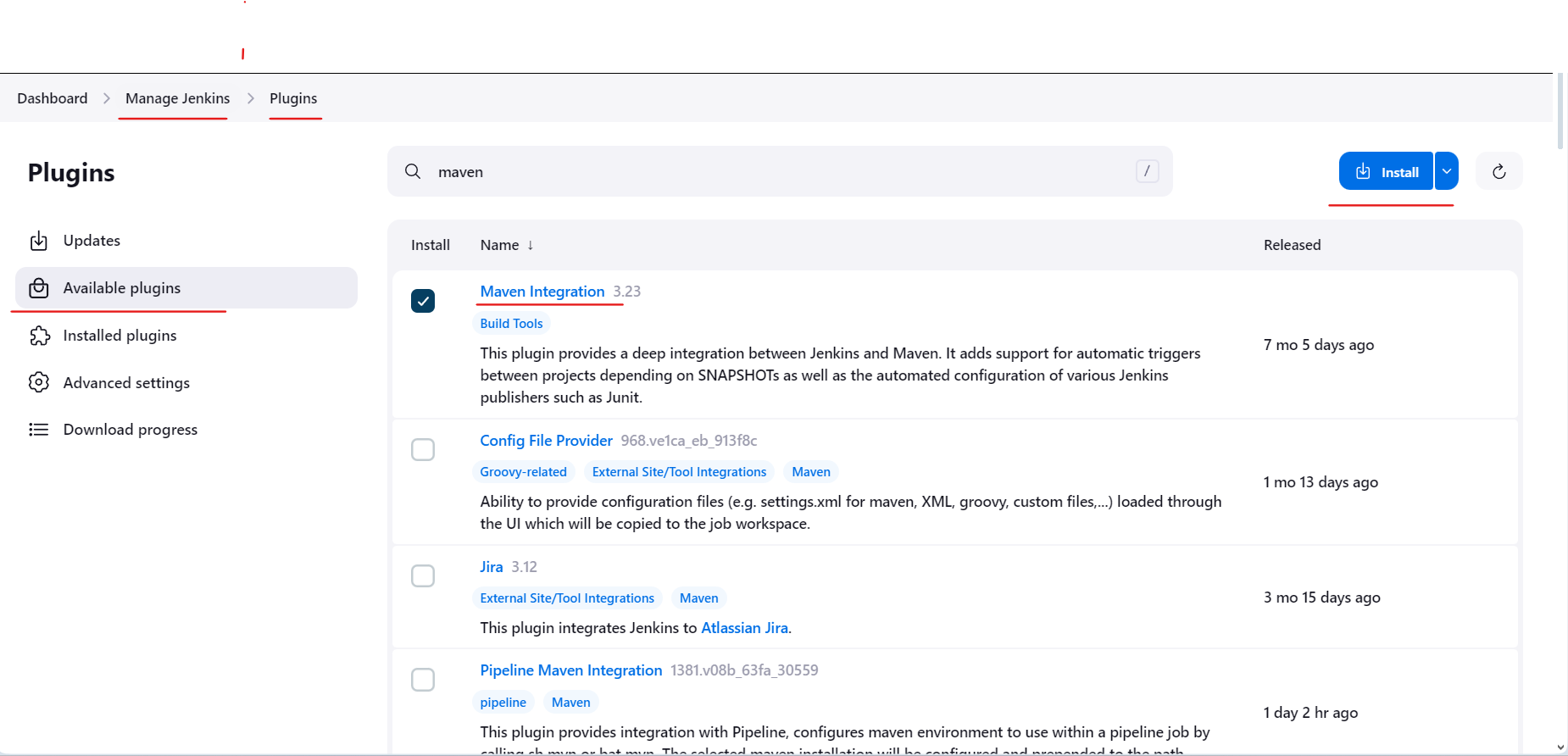


* 1. Now, you can work with Jenkins as shown in the screenshot below



**Step 5 : Installing Plugins and configuring Jenkins**

1. Go to Manage Jenkins -> Plugins -> Available Plugins -> Maven and then click install



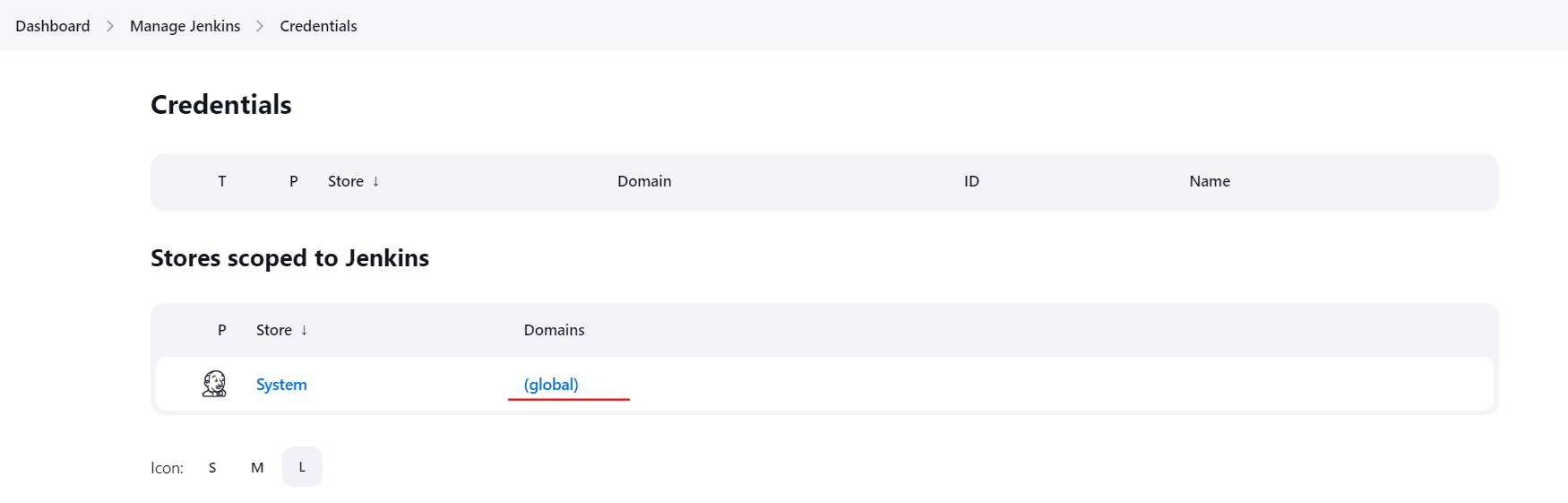
1. Similarly install the following plugins

**Ansible**

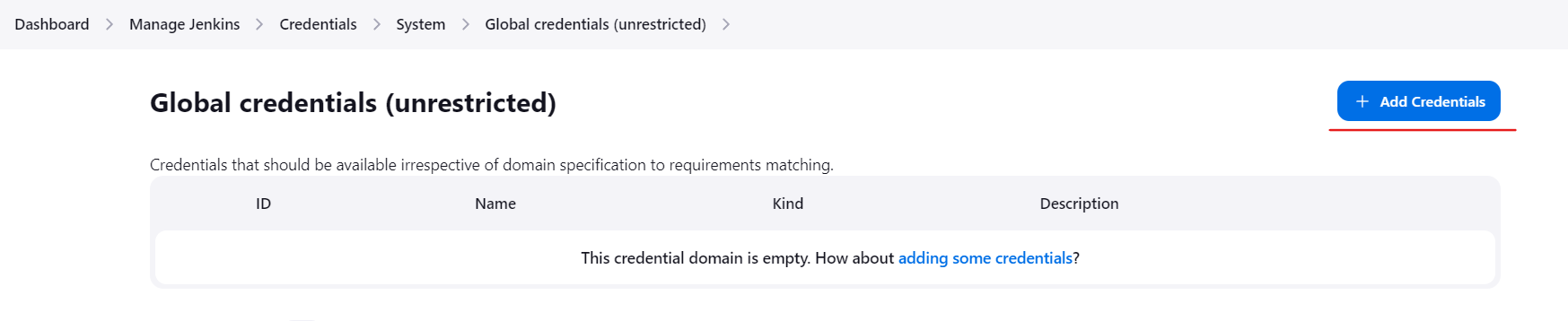
**Docker Pipeline / Docker**

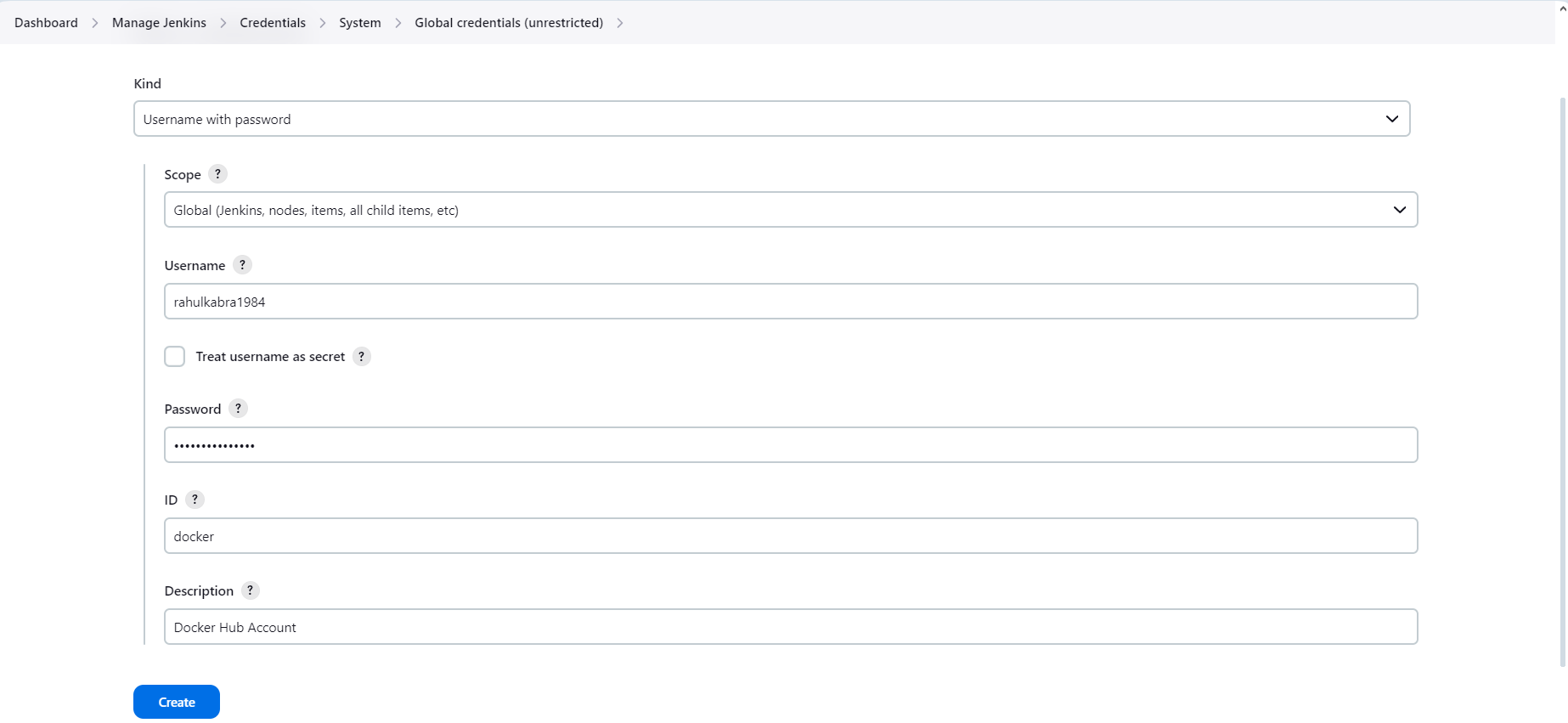
**HTML Publisher**

1. Now navigate to Manage Jenkins 🡪 Credentials and then select global domain.



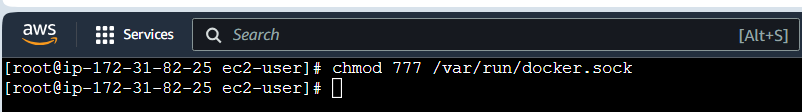
1. Click on new Credential by clicking on Add Credentials to create a new Docker hub credentials as per below details: (credential id should match the one you mentioned in jenkinsfile





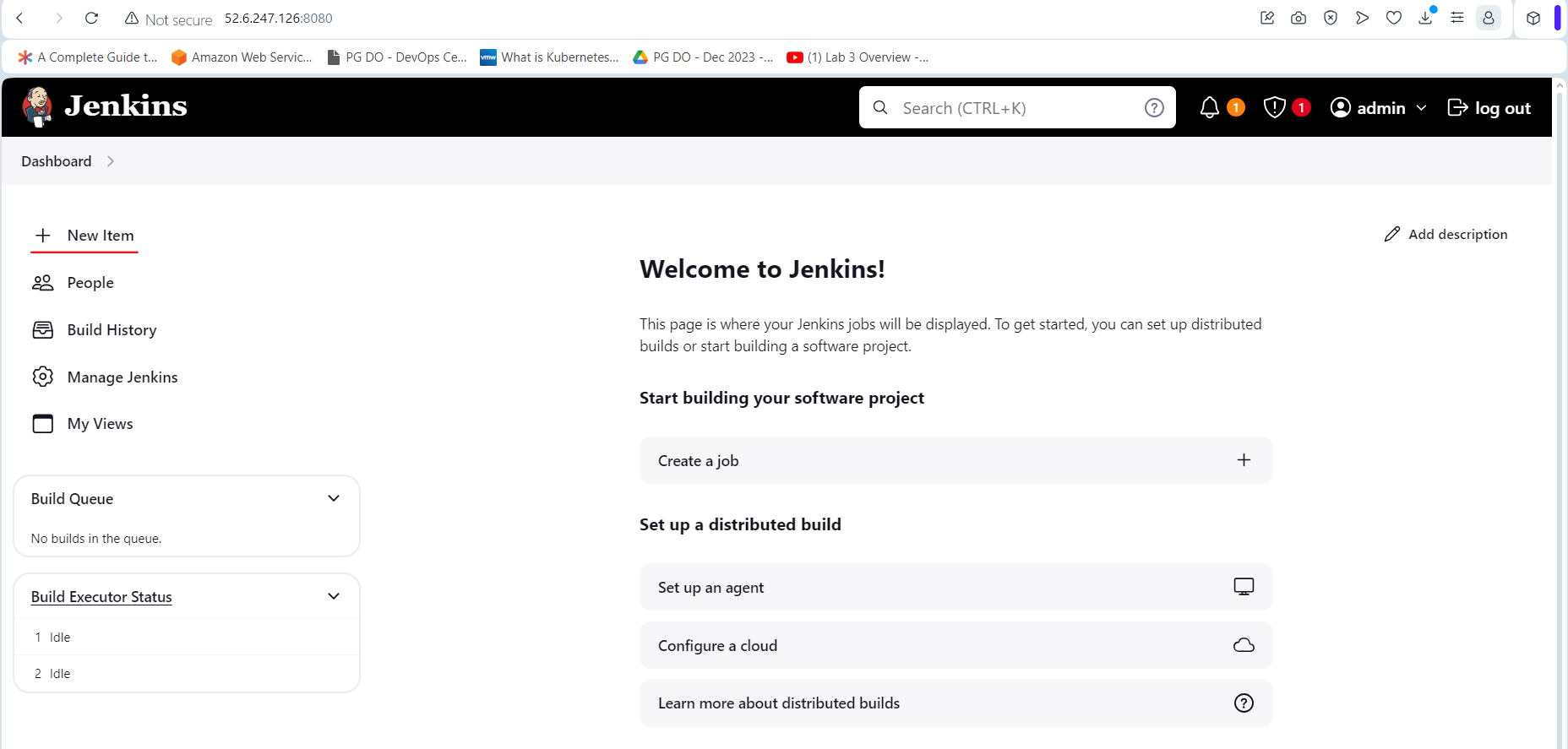
1. Next provide full access to Docker Sock file using below command:

**chmod 777 /var/run/docker.sock**

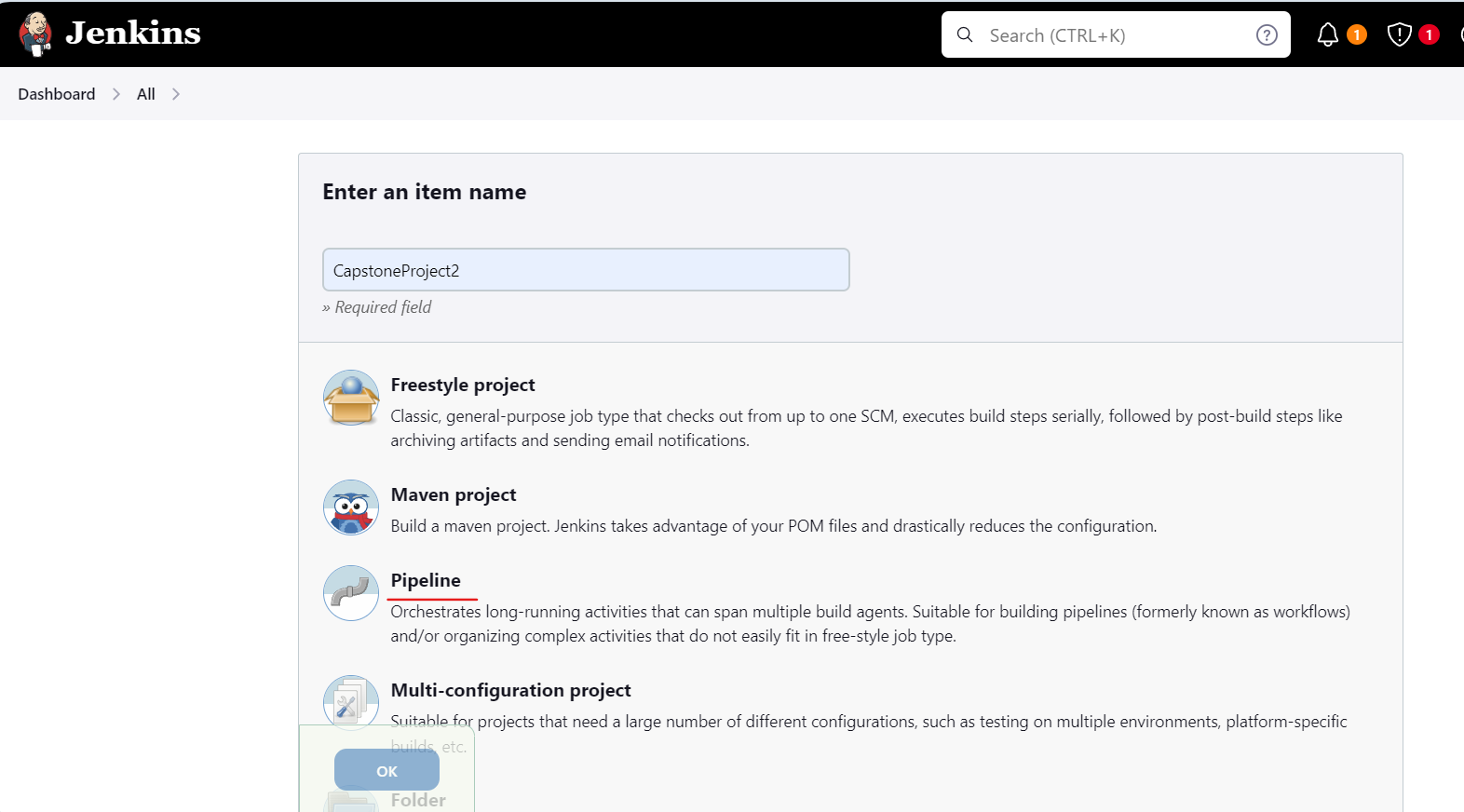
****

**Step 6 : Create Jenkins Pipeline**

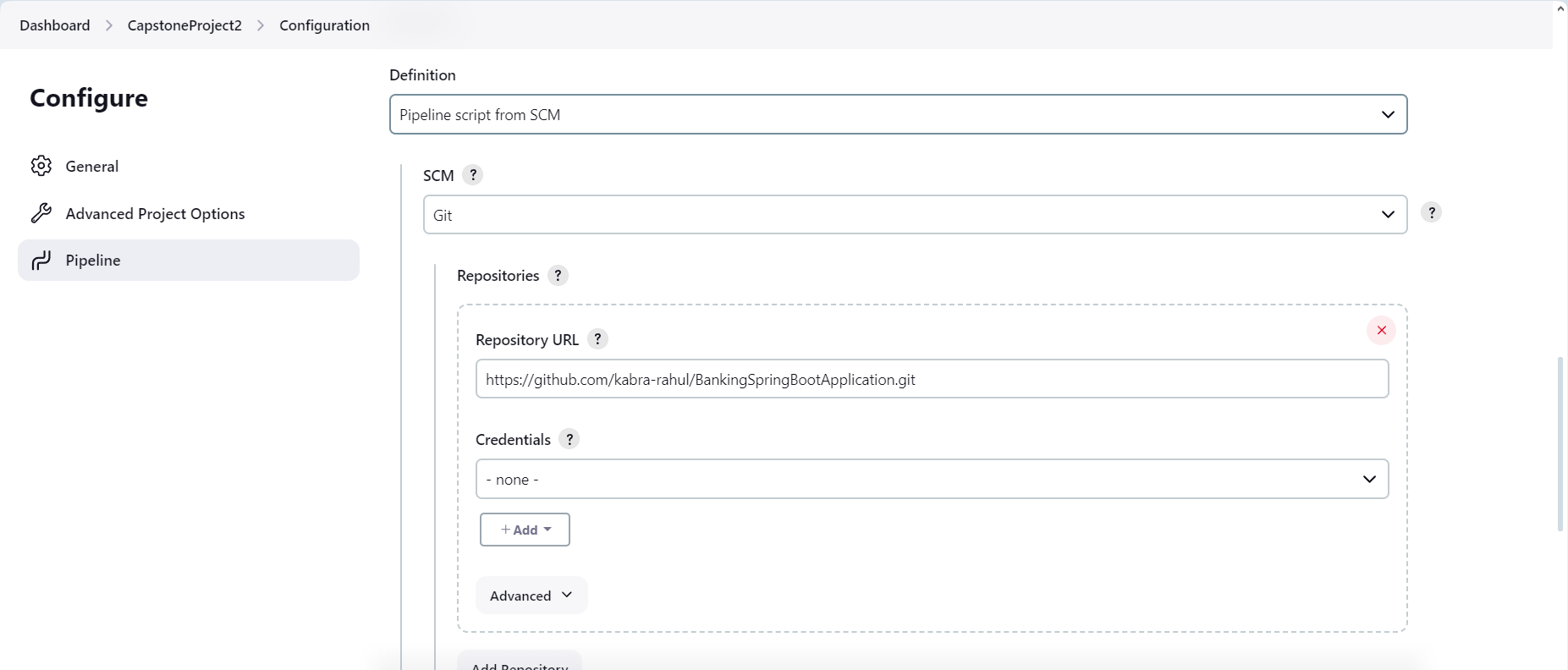
1. Access Jenkins application and click on New Item to create a new Jenkins job:

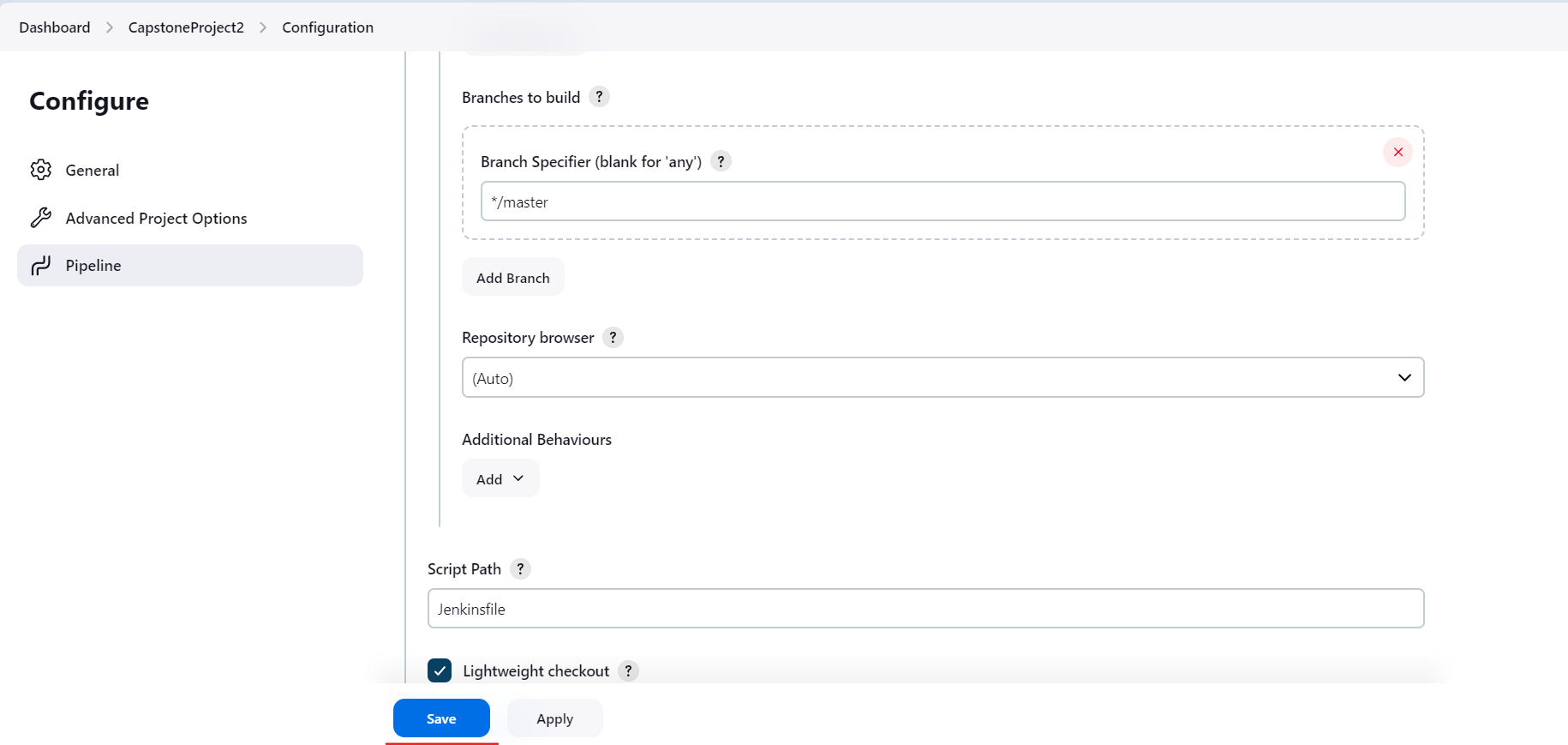


1. Select desired Jenkins pipeline job type and fill in job name



1. Once Clicked on Ok you will be navigated to Jenkins job configuration page where we can provide Jenkins job details such as parameters, Jenkins pipeline configuration etc.

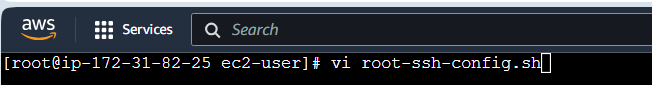




**Step 7 Configuring SSH Connection in EC2 Instance**

7.1 In the EC2 instance create file install verify-install.sh

**vi root-ssh-config.sh**

****

7.2 Type the below command in the verify-install.sh

**#!/bin/bash**

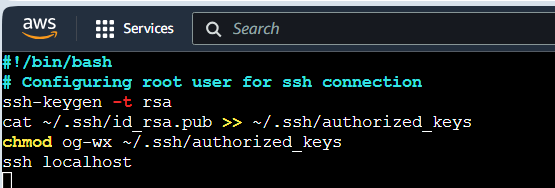
**# Configuring root user for ssh connection**

**ssh-keygen -t rsa**

**cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys**

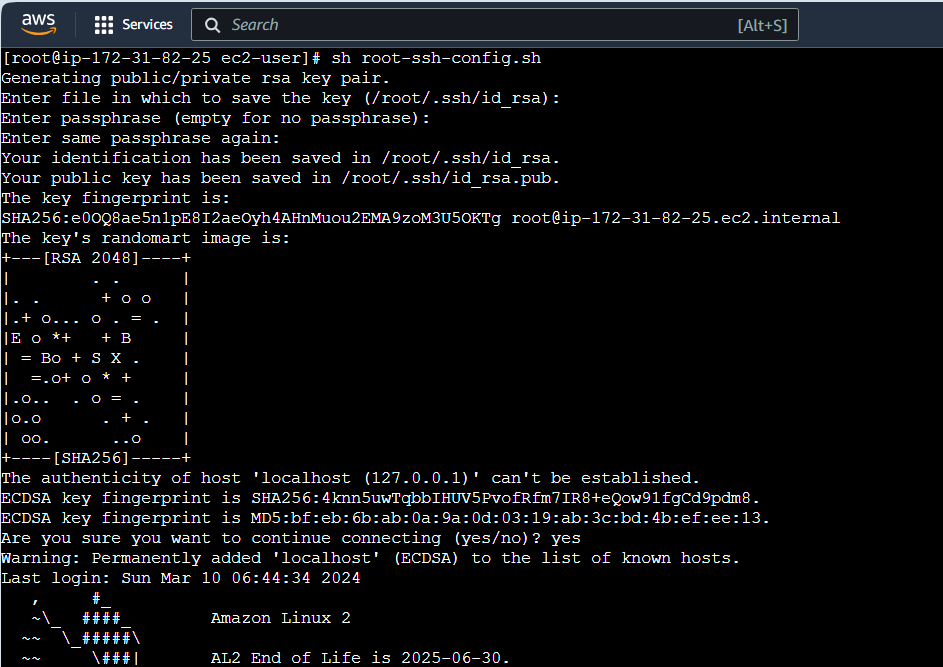
**chmod og-wx ~/.ssh/authorized\_keys**

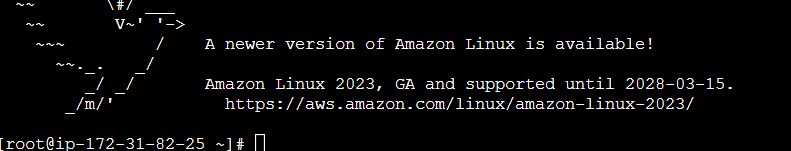
**ssh localhost**

****

7.3 Run below command to execute

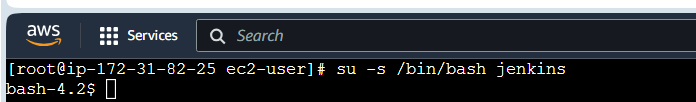
**sh root-ssh-config.sh**

****

****

7.4 Login as Jenkins user by executing the below command

**su -s /bin/bash jenkins**

****

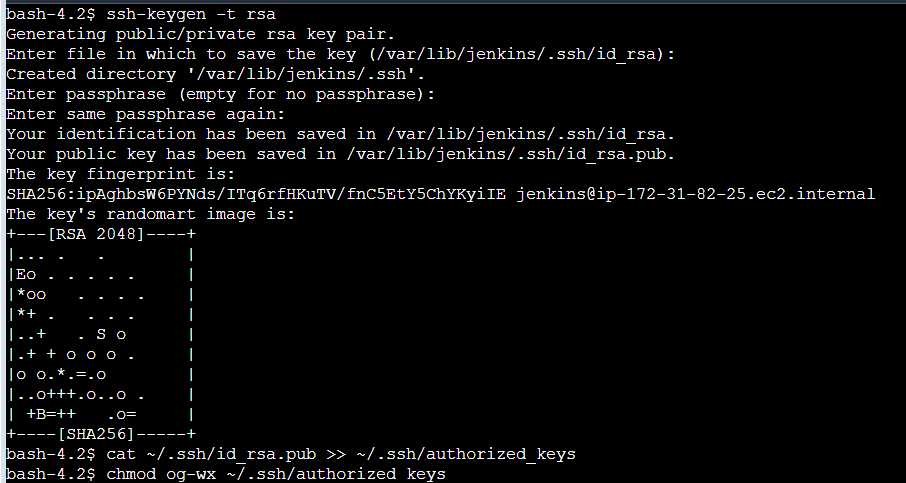
7.5 Execute the below commands in the bash shell

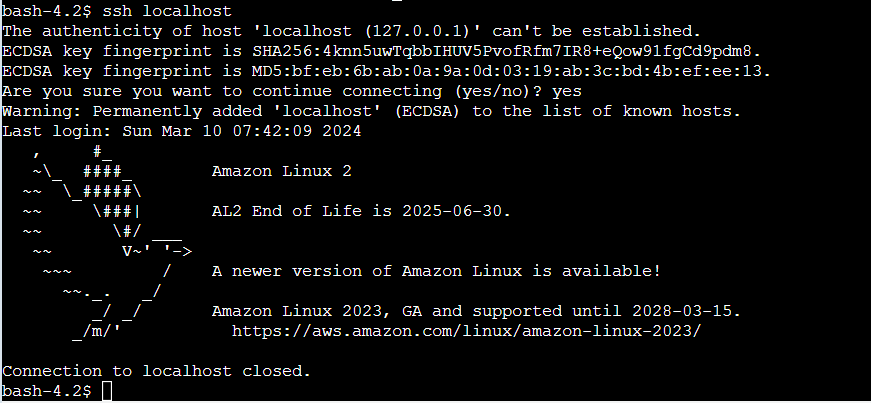
**ssh-keygen -t rsa**

**cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys**

**chmod og-wx ~/.ssh/authorized\_keys**

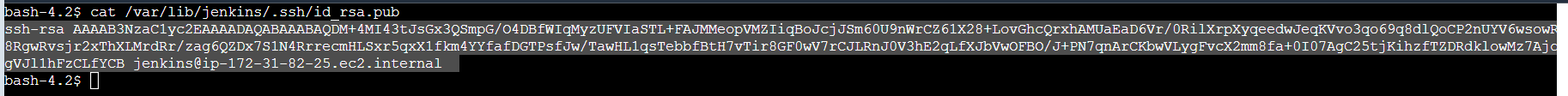
**ssh localhost**

****

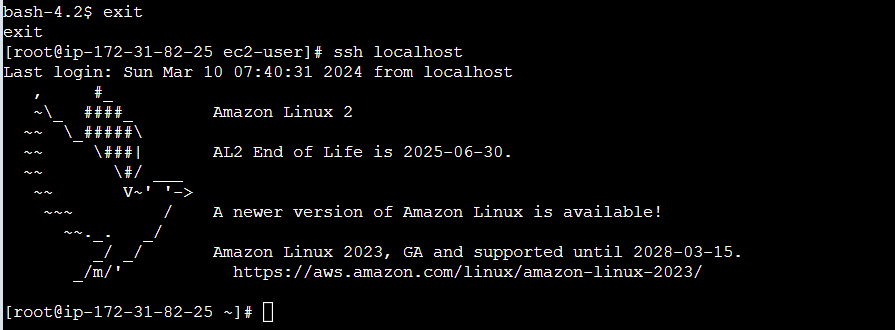
****

7.6 Copy key from id\_rsa.pub

**cat /var/lib/jenkins/.ssh/id\_rsa.pub**

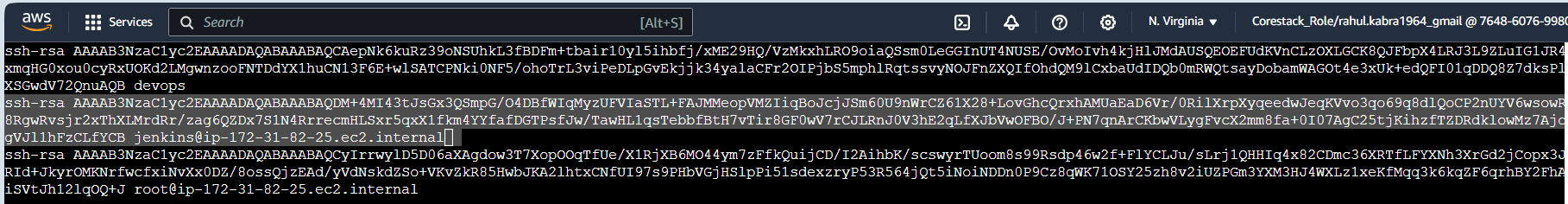


7.7 Exit from bash back to root user and ssh to localhost



7.8 Paste the copied key into authorized\_keys file using below command

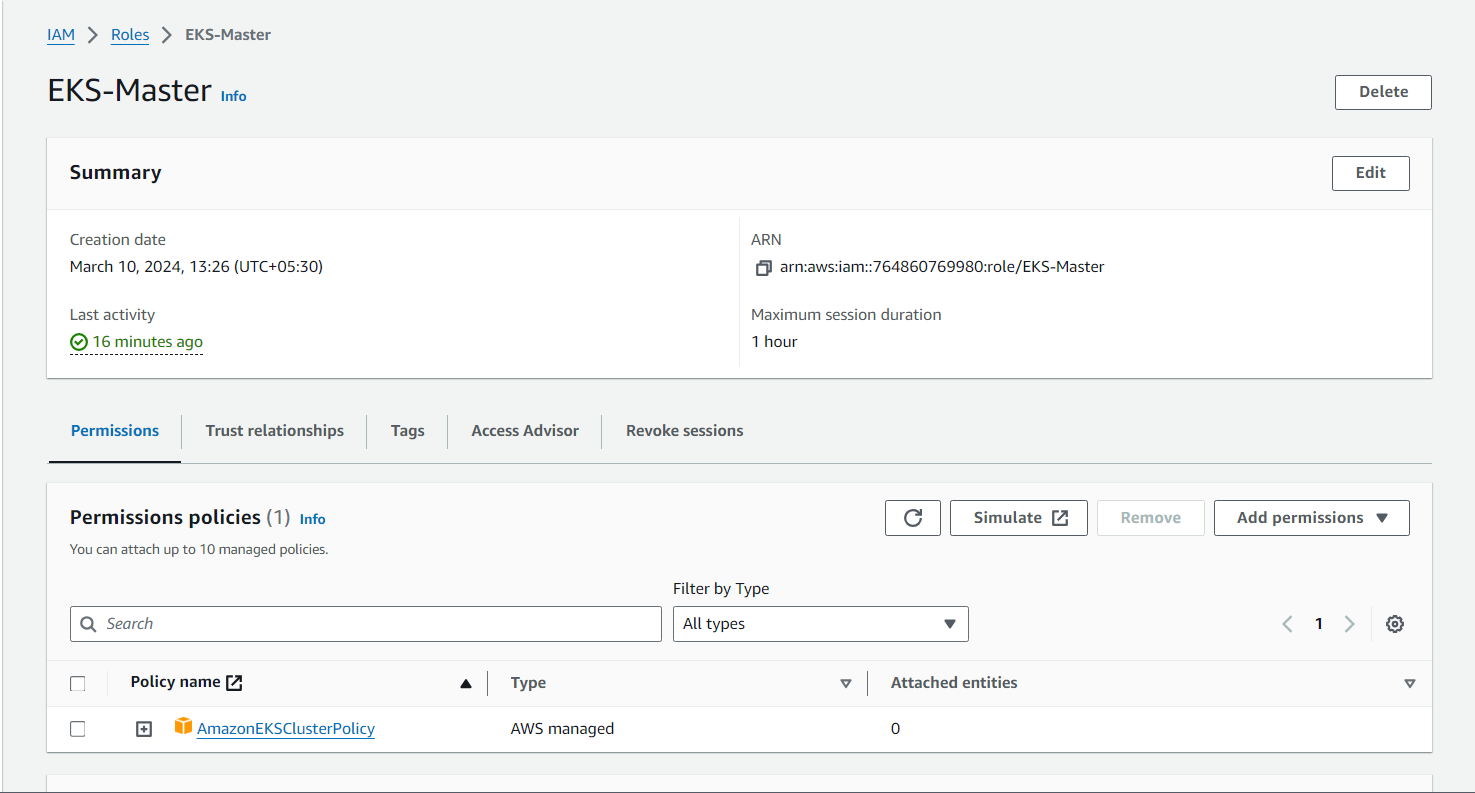
**vi .ssh/authorized\_keys**

****

**Step 8 : Creating EKS Cluster**

8.1 Create a new IAM role **EKS-Master** for EKS cluster with following policies

**AmazonEKSClusterPolicy**

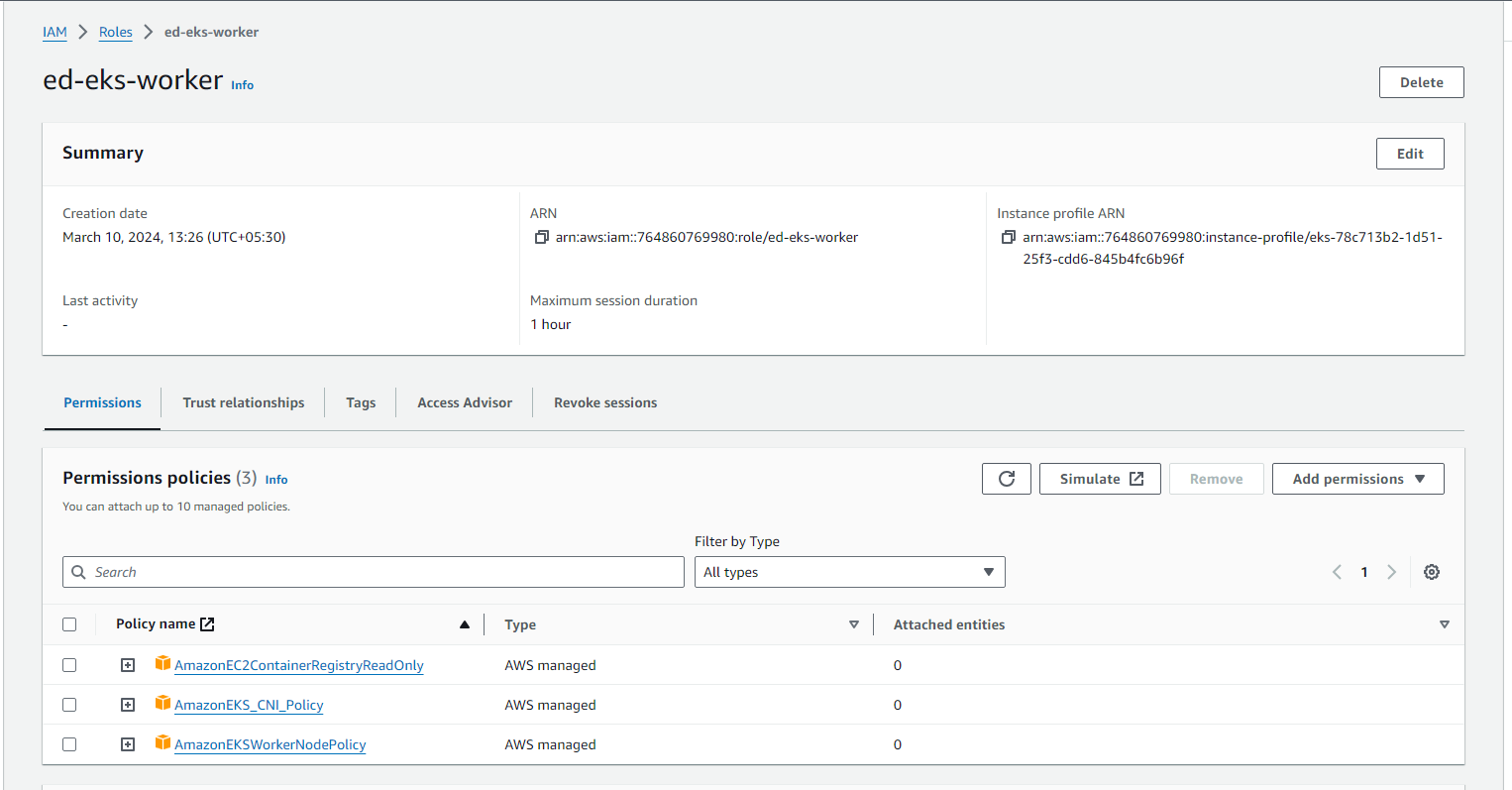
****

8.2 Create a new IAM **ed-eks-worker** role for the node group with following policies

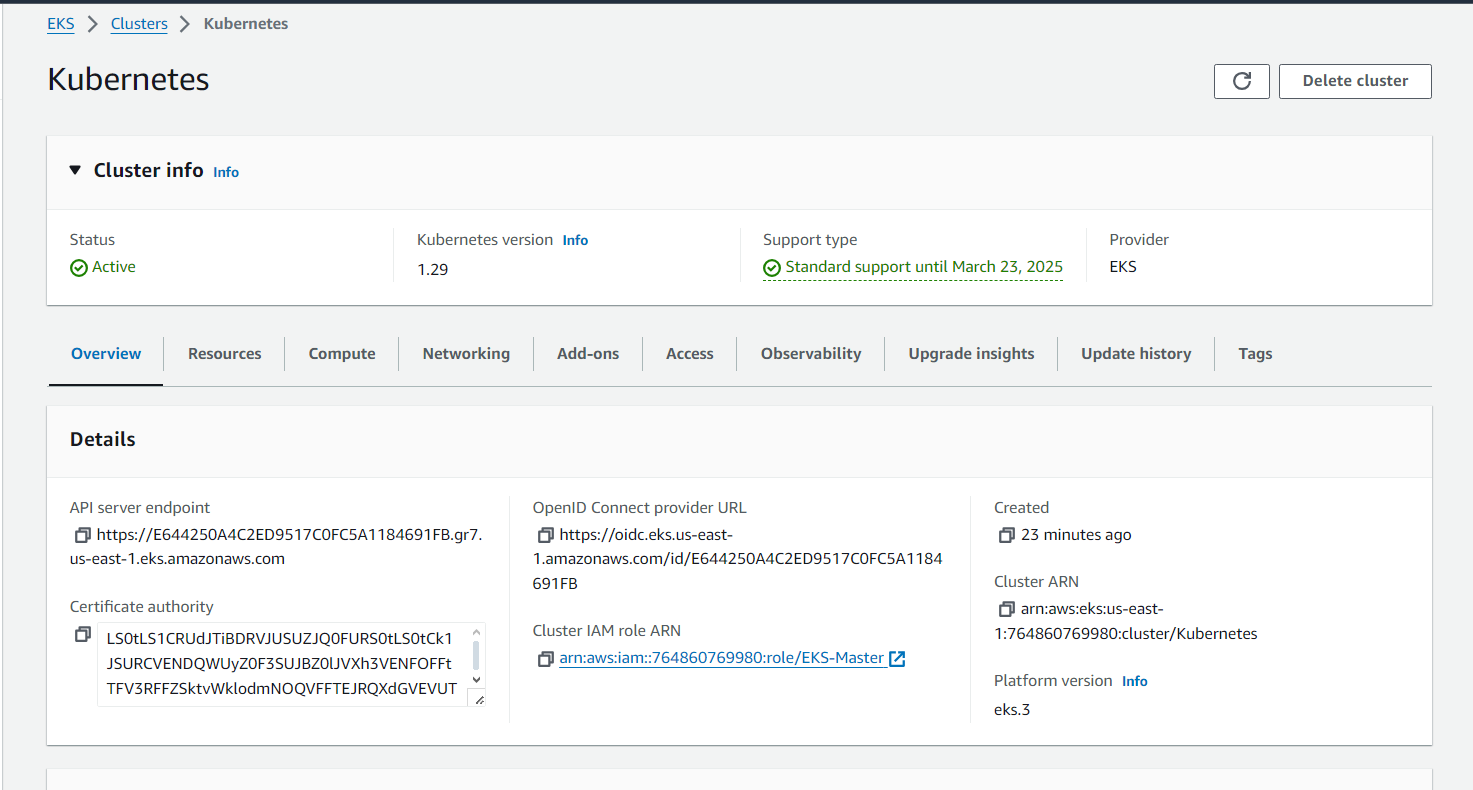
**AmazonEKSWorkerNodePolicy**

**AmazonEKS\_CNI\_Policy**

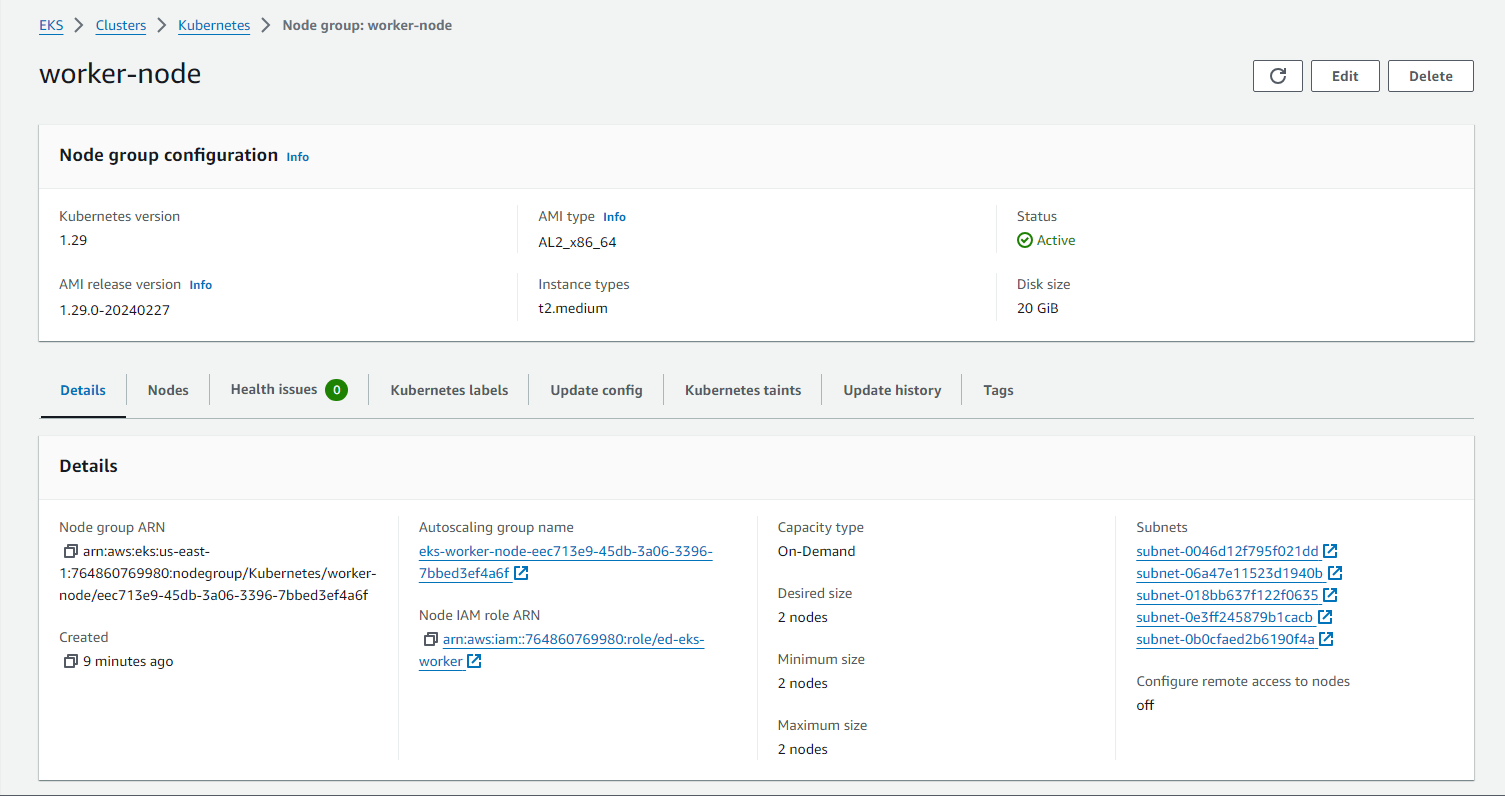
**AmazonEC2ContainerRegistryReadOnly**

****

8.3 Now using the IAM role **EKS-Master**, create a EKS cluster named **Kubernetes**

****

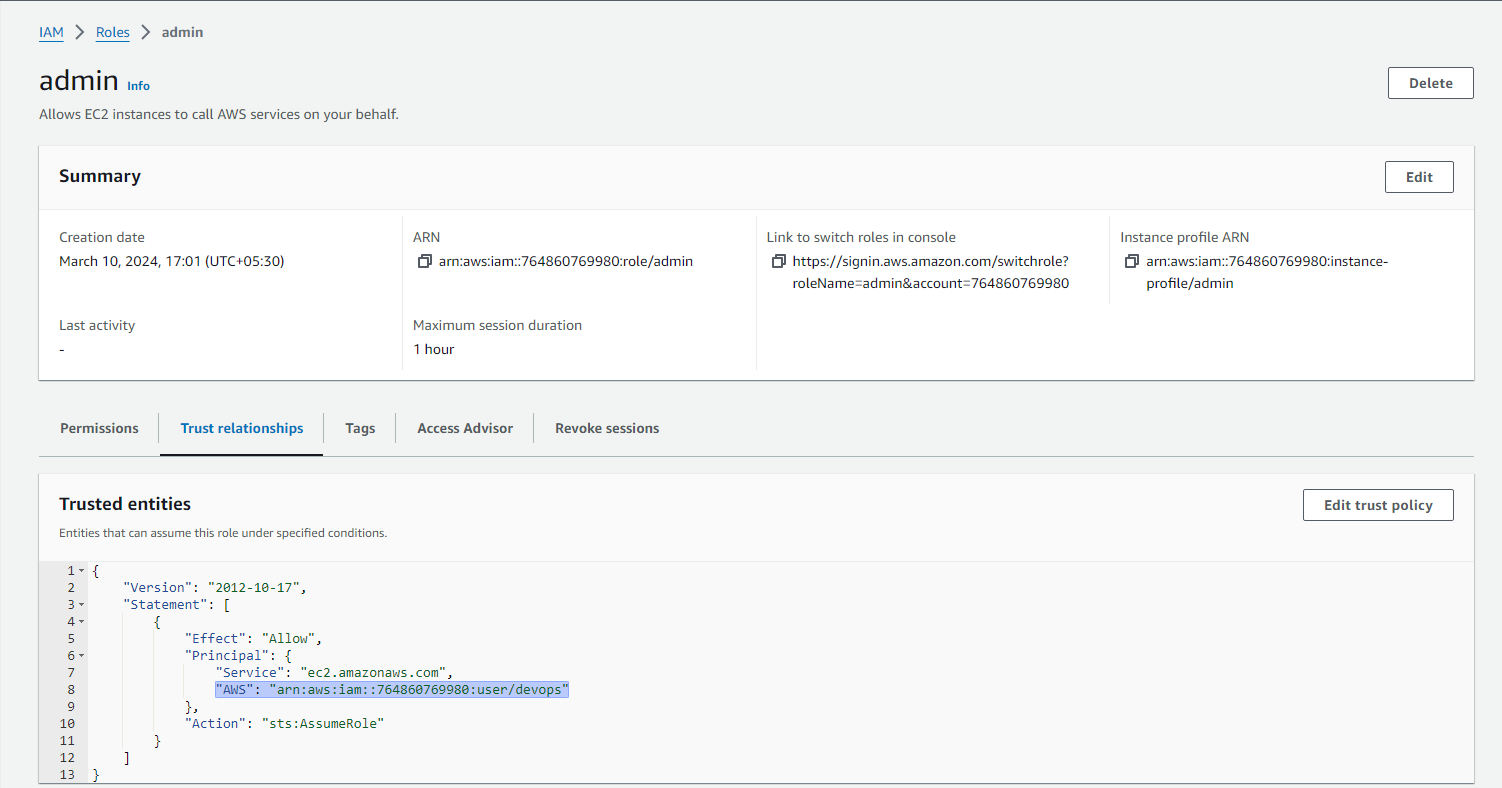
8.4 Now using the IAM role **ed-eks-worker**, create a node group **worker-node**



8.5 Now create a new IAM role **admin** for EKS cluster with following policy

**AdministratorAccess** and add **"AWS": "arn:aws:iam::764860769980:user/devops"**

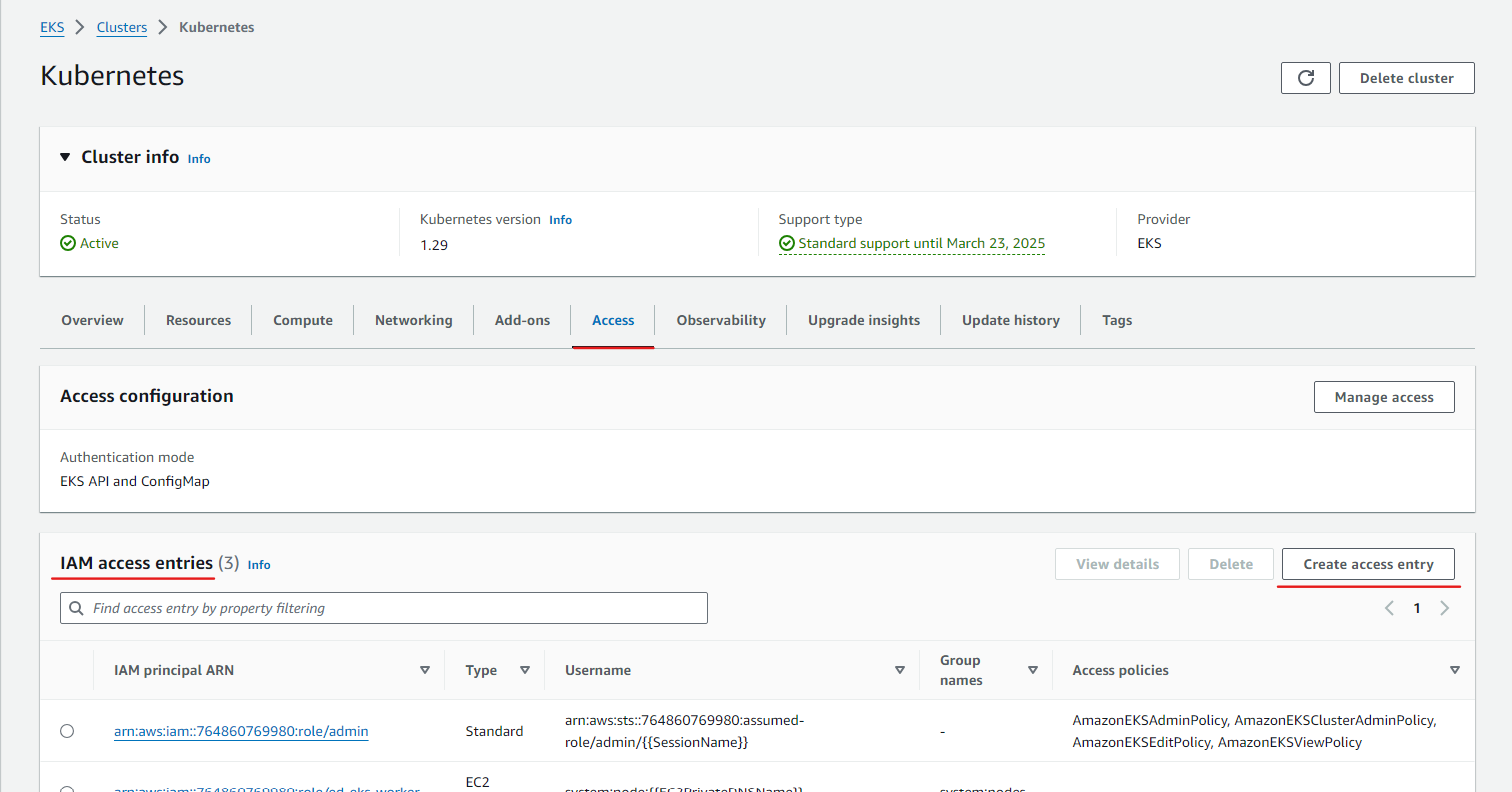
In the trust relationships of it.



8.6 In the EKS cluster , go to access tab and in IAM access entries , click on Create access

entry and ad the following arole entry in it

**arn:aws:iam::764860769980:role/admin**



8.7 Now configure the kubeconfig file to connect to the EKS cluster by executing below

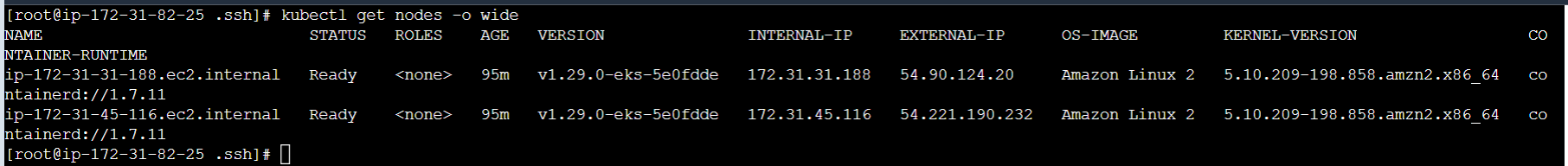
command

**aws eks update-kubeconfig --region us-east-1 --name Kubernetes --role-arn "arn:aws:iam::764860769980:role/admin"**

****

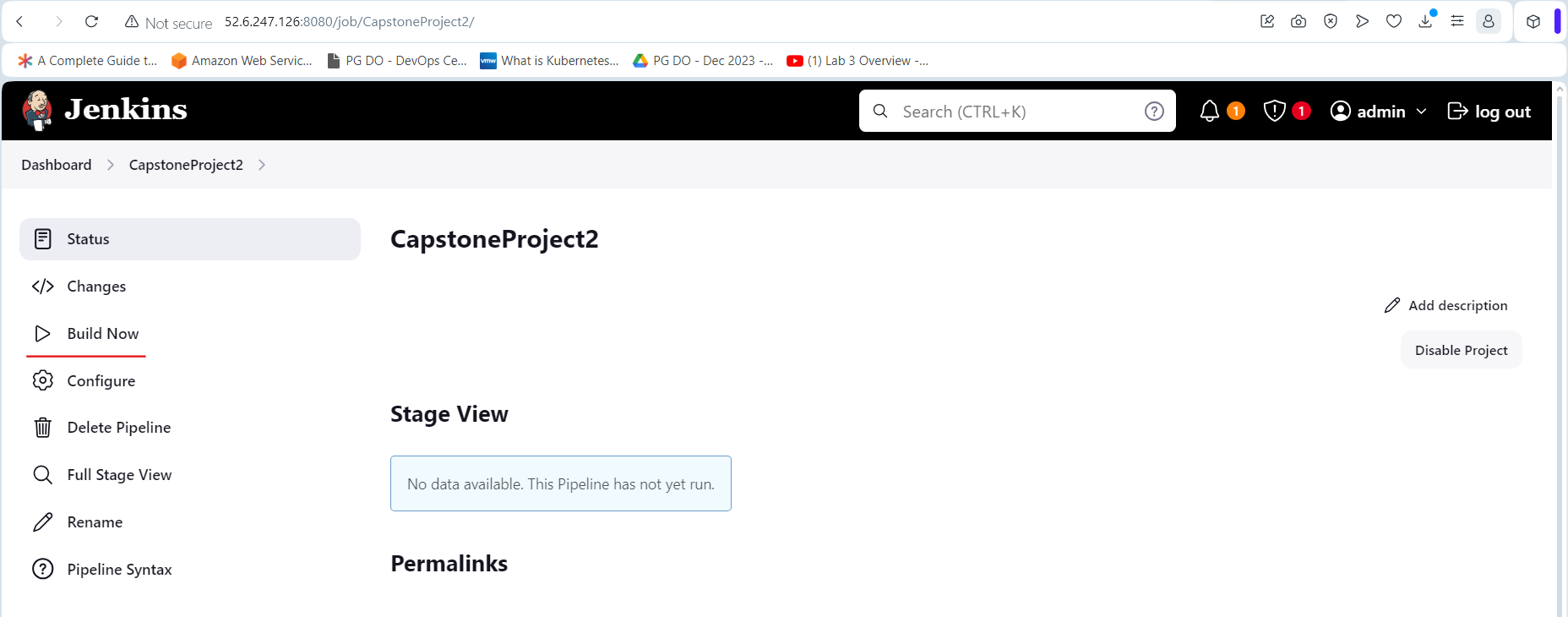
8.7 Now verify the EKS cluster and nodes in the terminal

**kubectl get nodes -o wide**

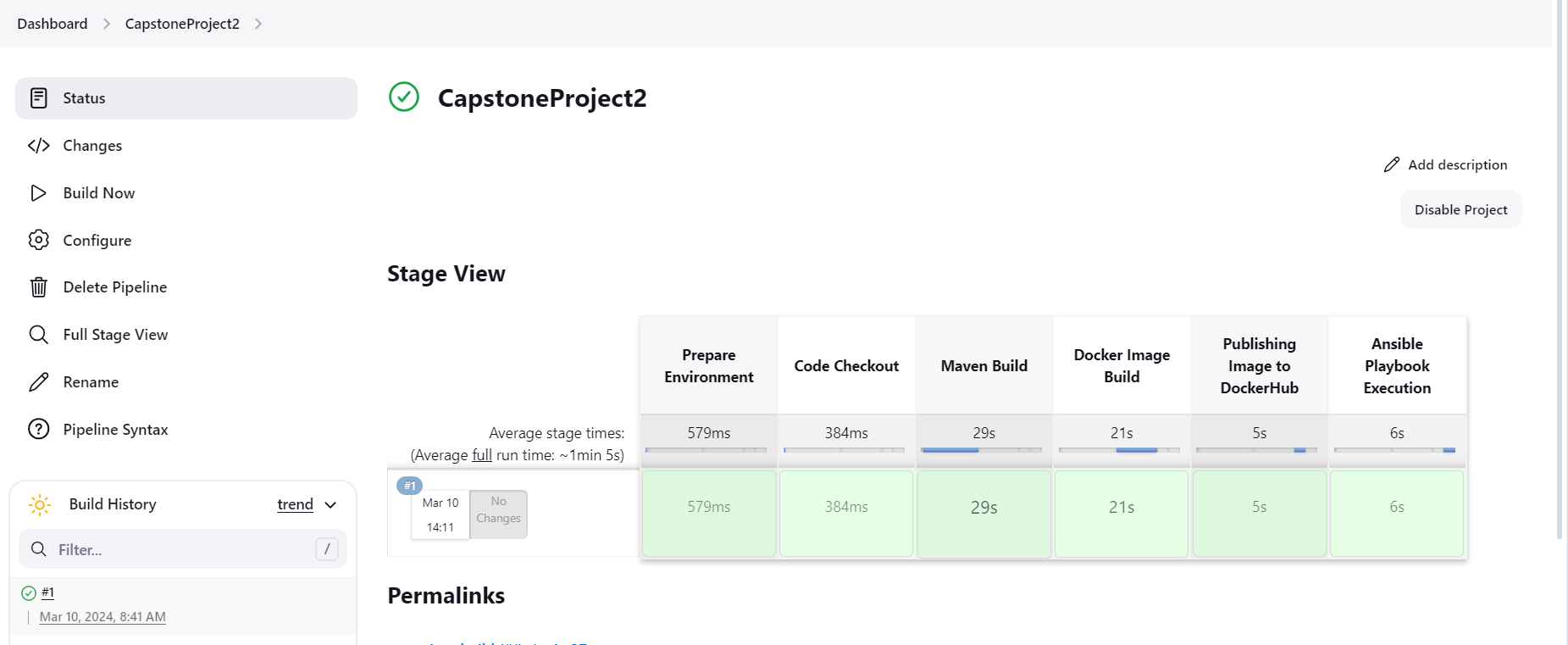


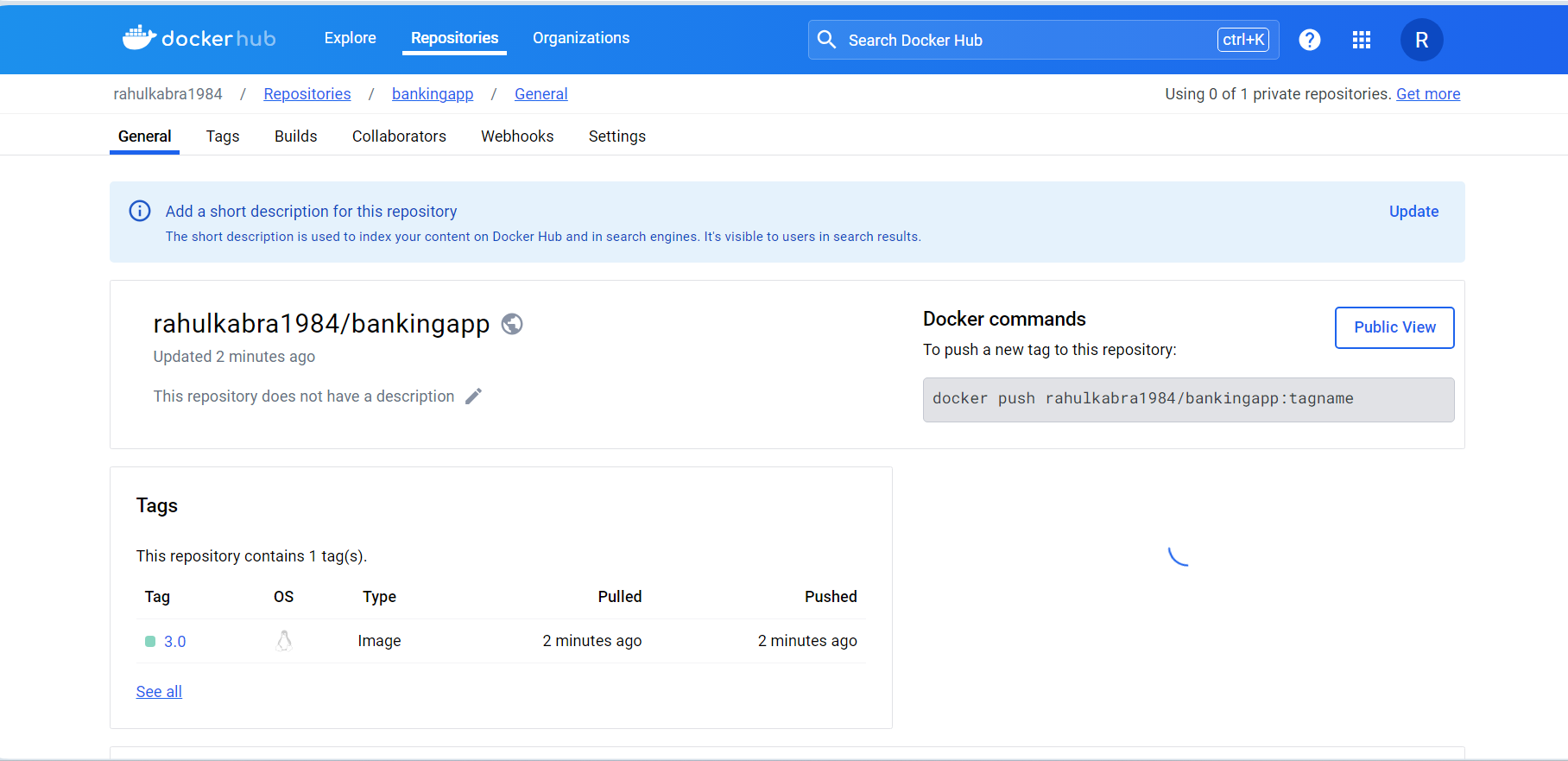
**Step 9 : Execute Jenkins Build and Validate it**

9.1 Navigate to Jenkins job created in above step and click on Build now to start running build for Jenkins job created.



9.2 Once Jenkins build is completed validate if Docker image really gets uploaded to Docker hub or not.



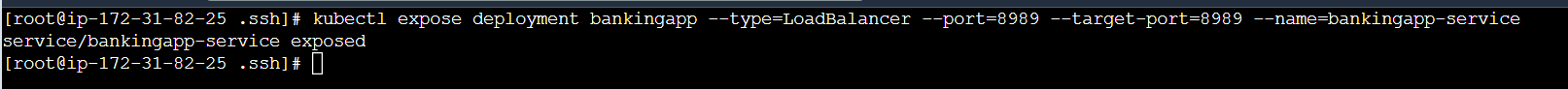


9.3 Now execute the below script in the shell script to create loadbalancer service and

verify it.

**kubectl expose deployment bankingapp --type=LoadBalancer --port=8989 --target-**

**port=8989 --name=bankingapp-service**

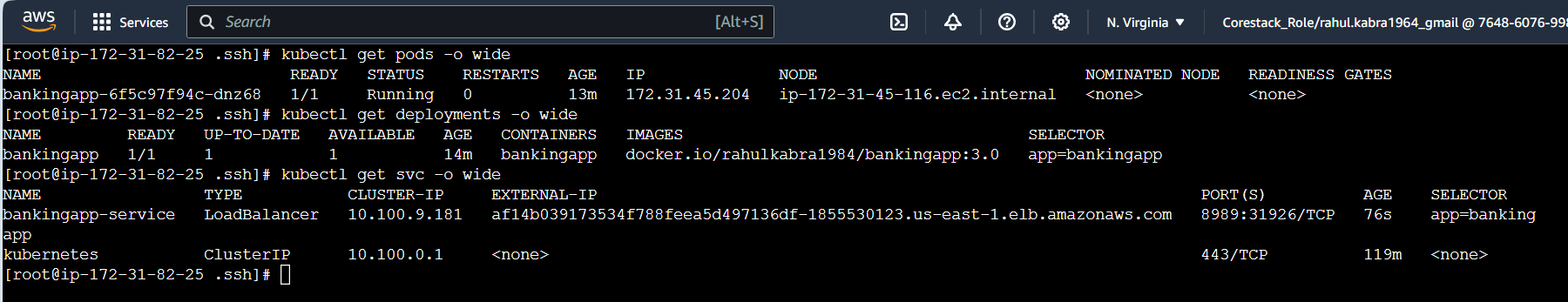


9.4 Verify in the terminal using below commands the pods created

**kubectl get pods -o wide**

**kubectl get deployments -o wide**

**kubectl get svc -o wide**

****

9.5 Now verify if the wesite is opening on the below address

<http://af14b039173534f788feea5d497136df-1855530123.us-east-1.elb.amazonaws.com:8989/bank-api/swagger-ui.html>

