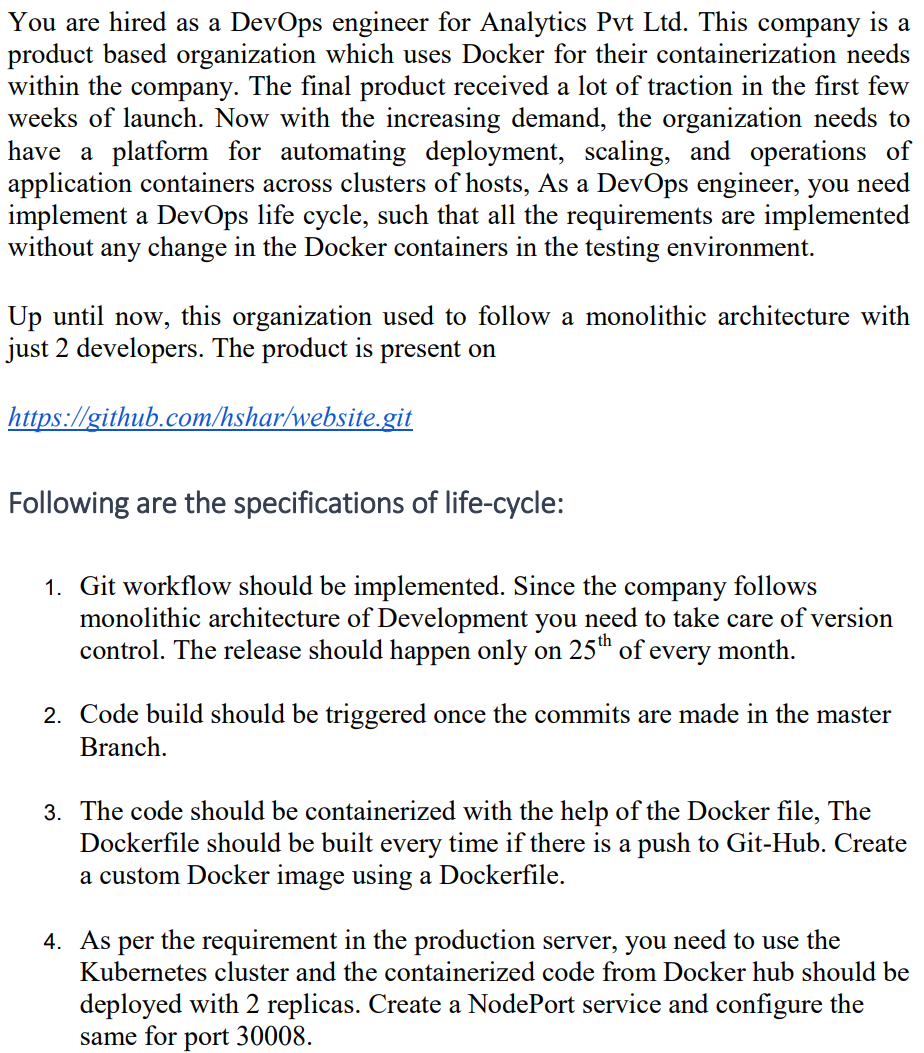
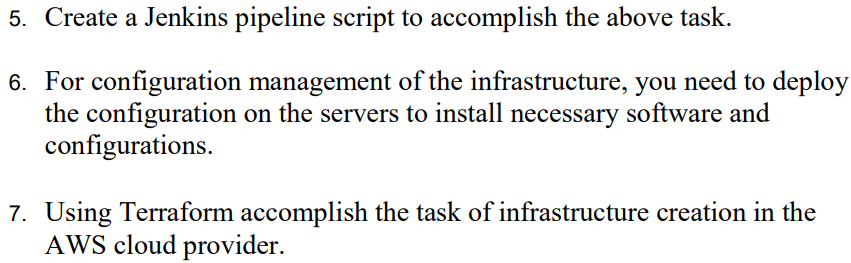
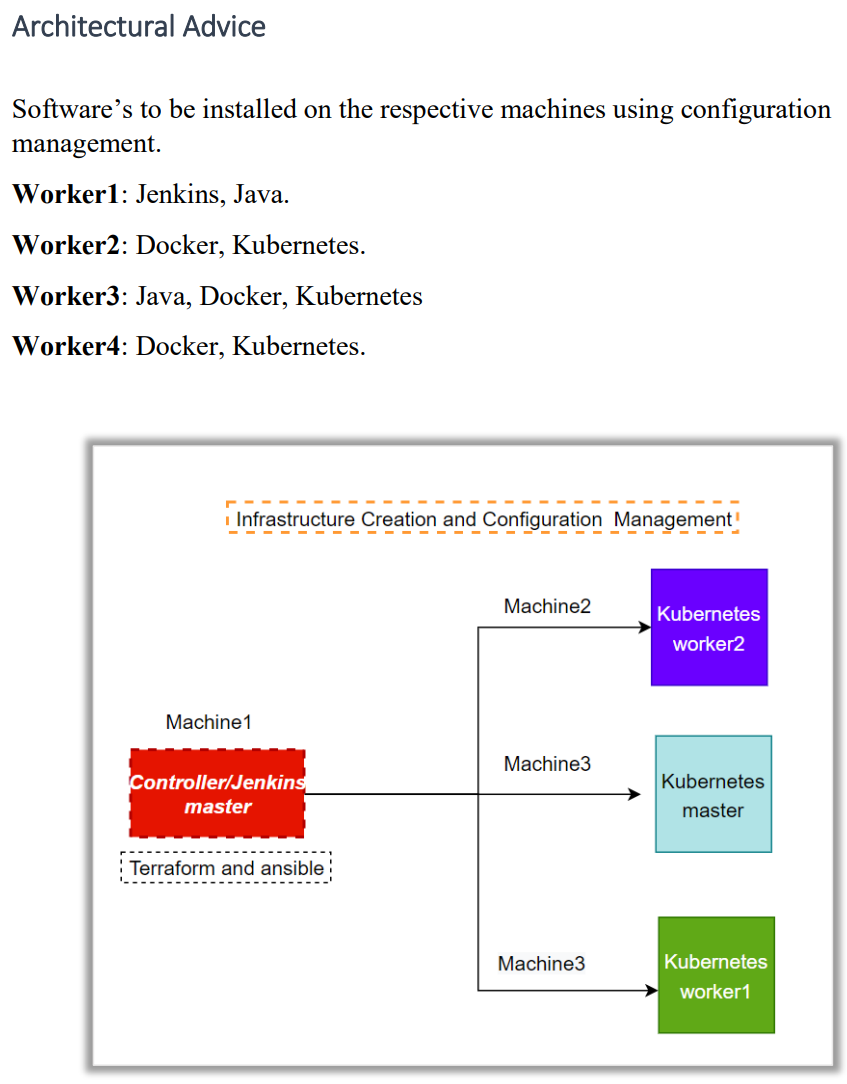
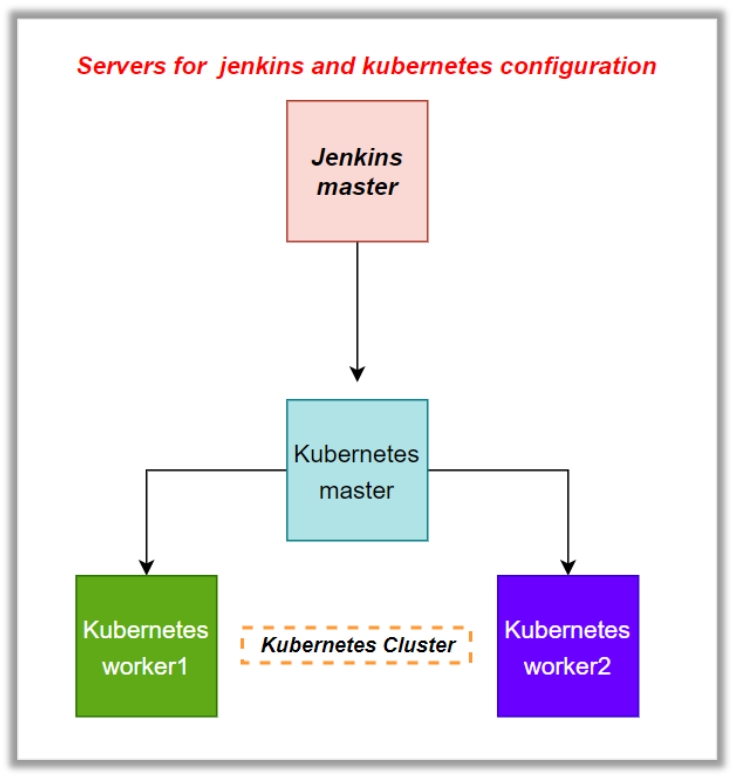
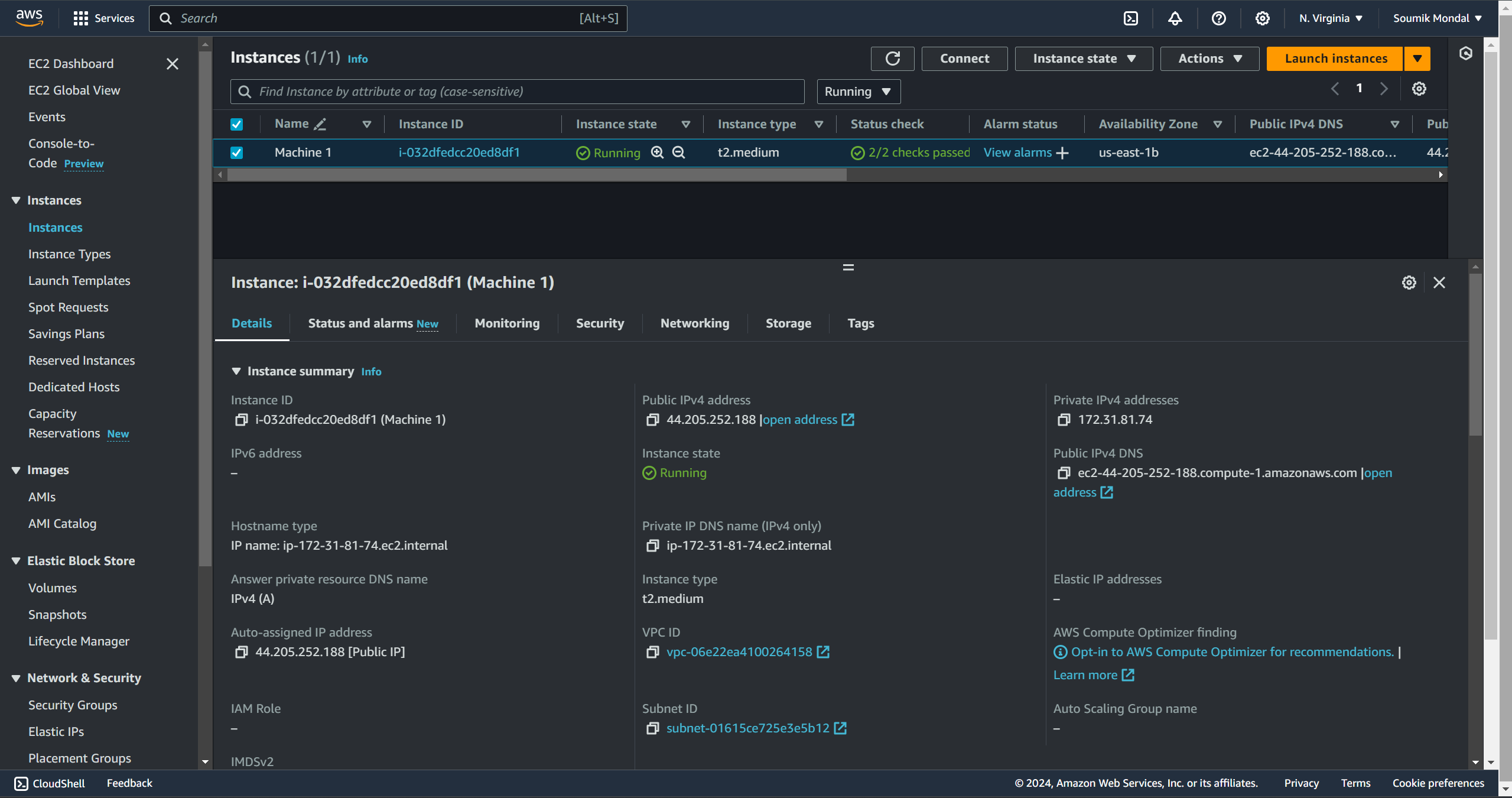
**DevOps Project**

****

****

****

-----------------------------------------------------------**Project Solution**-----------------------------------------------------

  
**Created an instance with ubuntu 20.04 and the instance type is – t2.micro**

**Terraform installation script->**

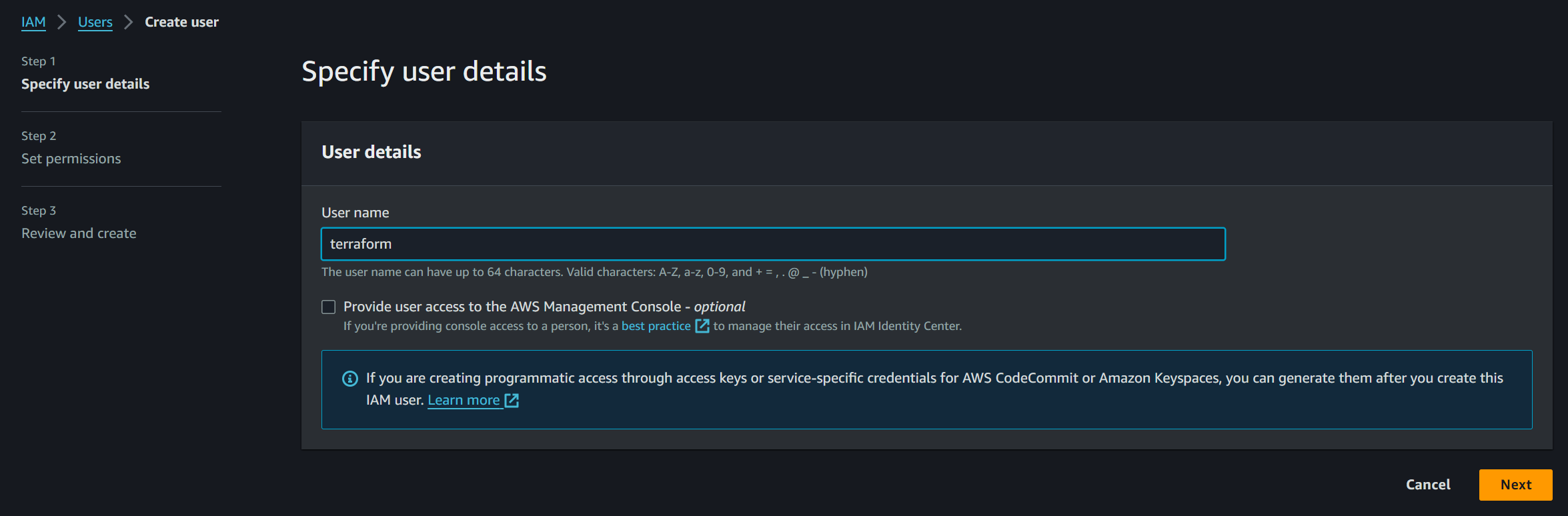
 wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

 echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

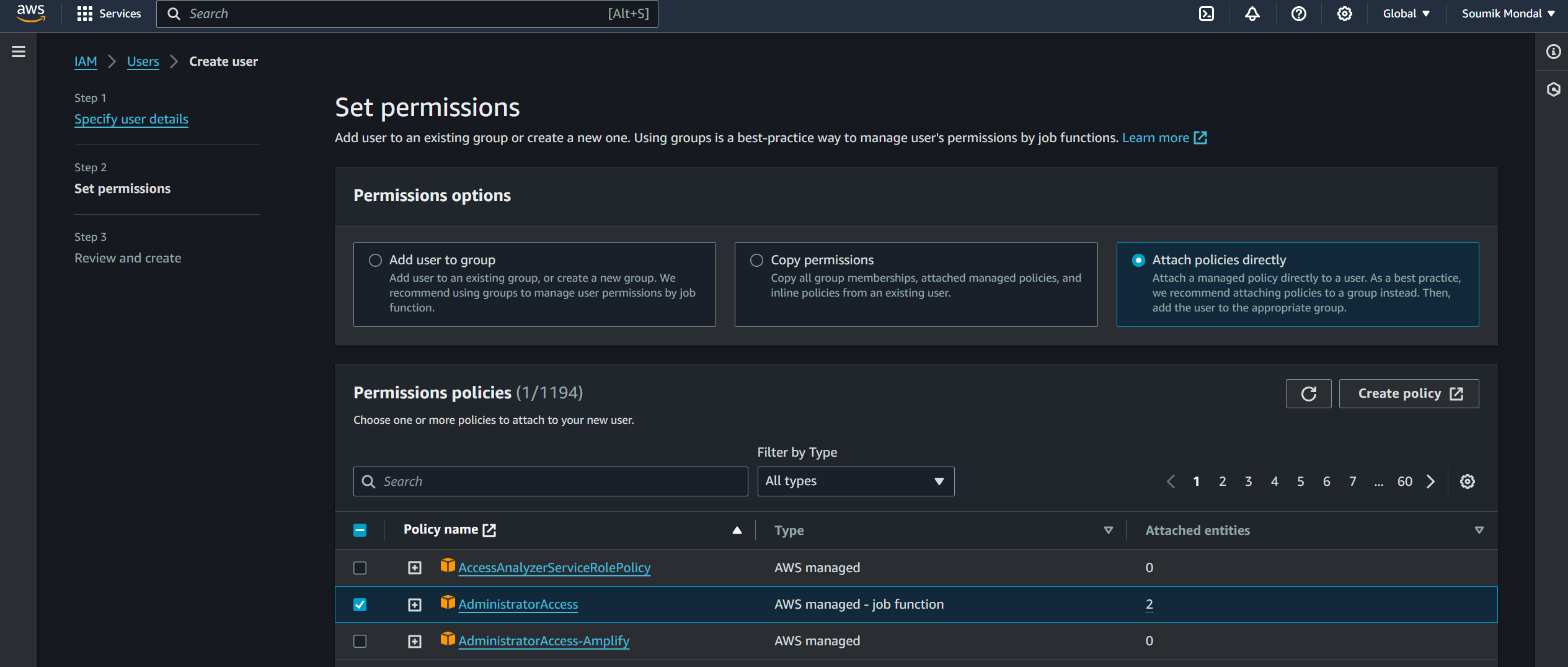
 sudo apt update && sudo apt install terraform

**Now we have to create other three instances with the help of Terraform.**

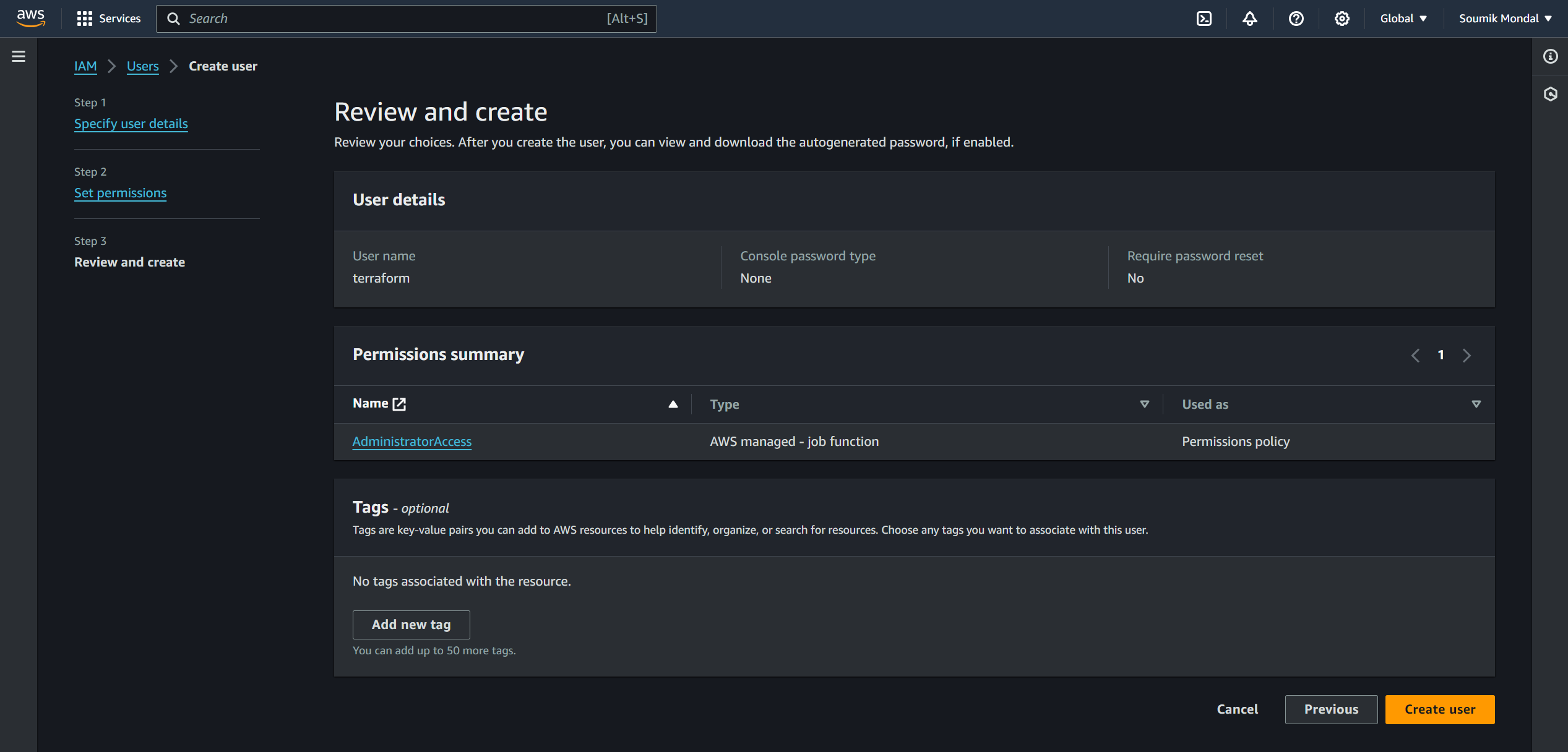
**Let’s create access and a secret key –**

****

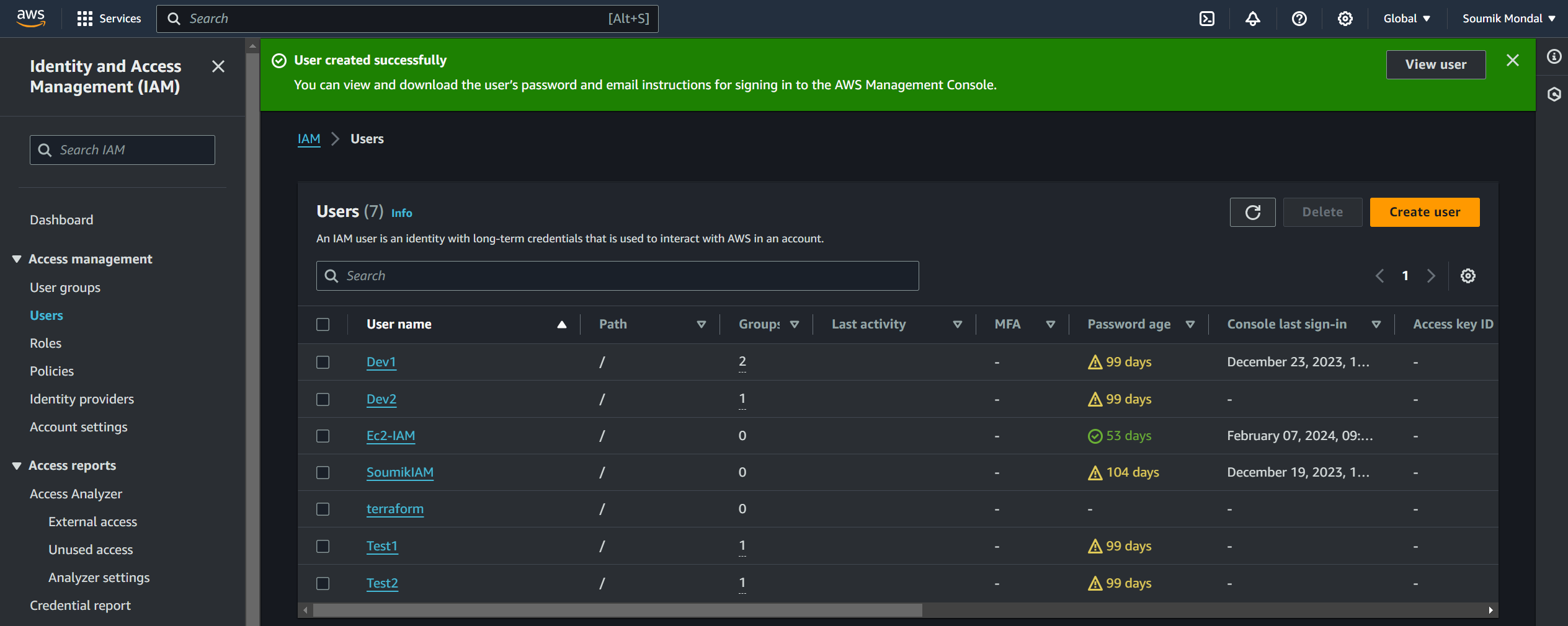


****

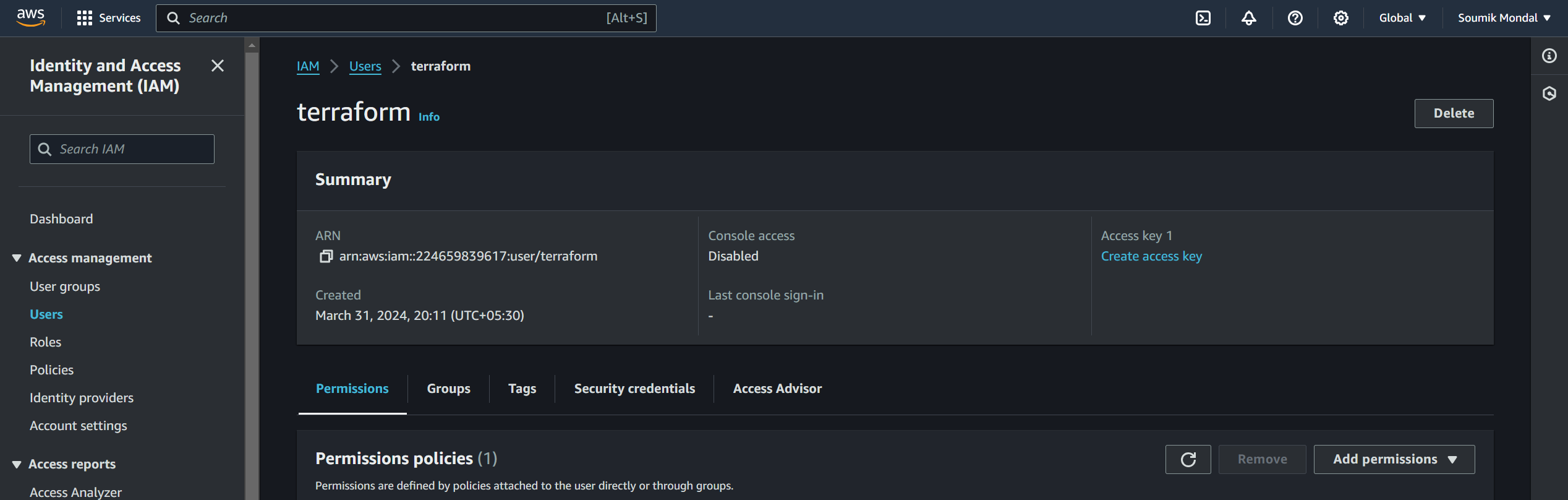


****

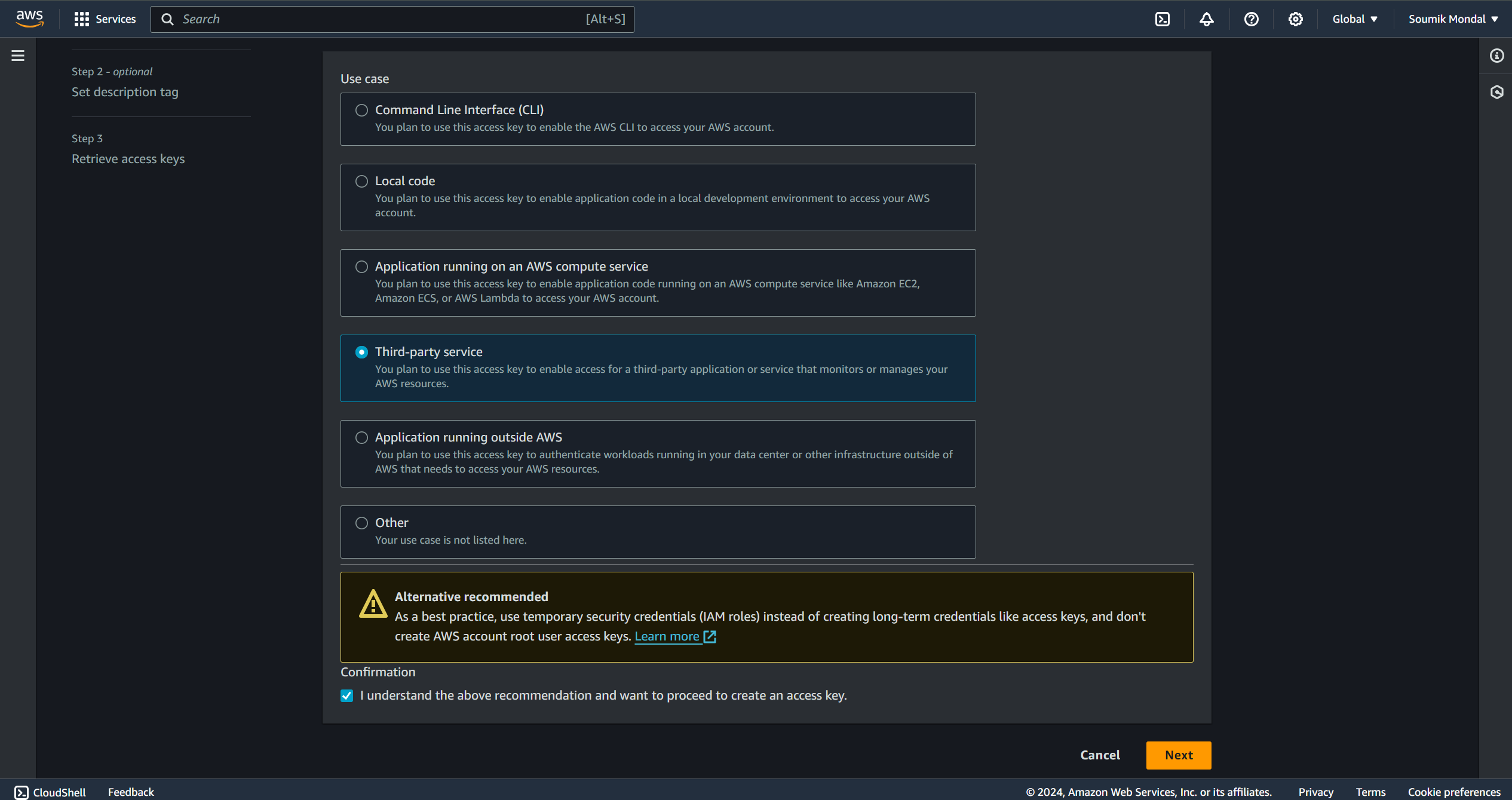


****

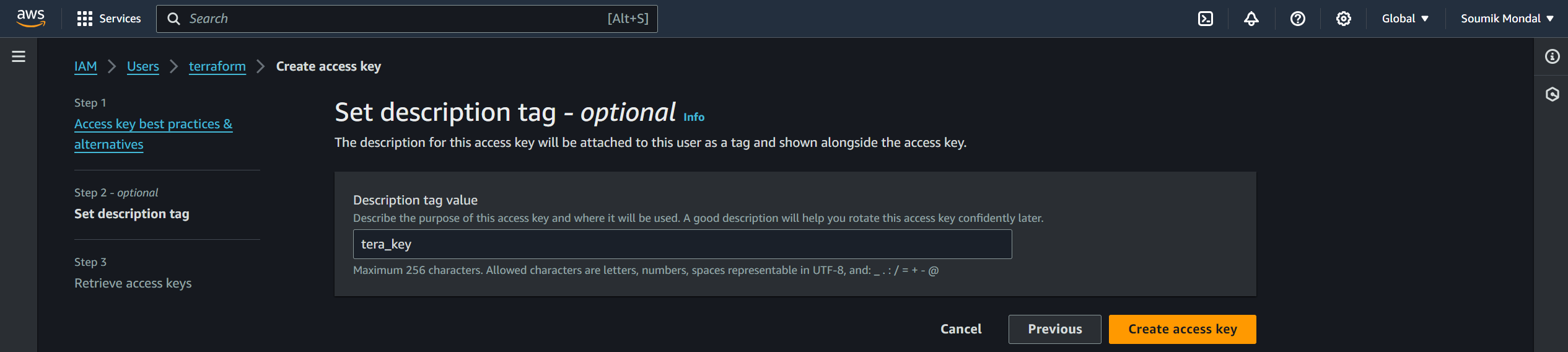


****



****



****



****

Do not share the Access key and secret key to anyone



**Terra.tf**provider "aws" {

    access\_key = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

    secret\_key = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

    region = "us-east-1"

}

resource "aws\_instance" "k8\_master" {

  ami           = "ami-0cd59ecaf368e5ccf"

  instance\_type = "t2.medium"

  key\_name = "pemKey"

  security\_groups = ["default"]

  tags = {

    Name = "Machine 3"

  }

}

resource "aws\_instance" "k8\_slave1" {

  ami           = "ami-0cd59ecaf368e5ccf"

  instance\_type = "t2.micro"

  key\_name = "pemKey"

  security\_groups = ["default"]

  tags = {

    Name = "Machine 2"

  }

}

resource "aws\_instance" "k8\_slave2" {

  ami           = "ami-0cd59ecaf368e5ccf"

  instance\_type = "t2.micro"

  key\_name = "pemKey"

  security\_groups = ["default"]

  tags = {

    Name = "Machine 4"

  }

}

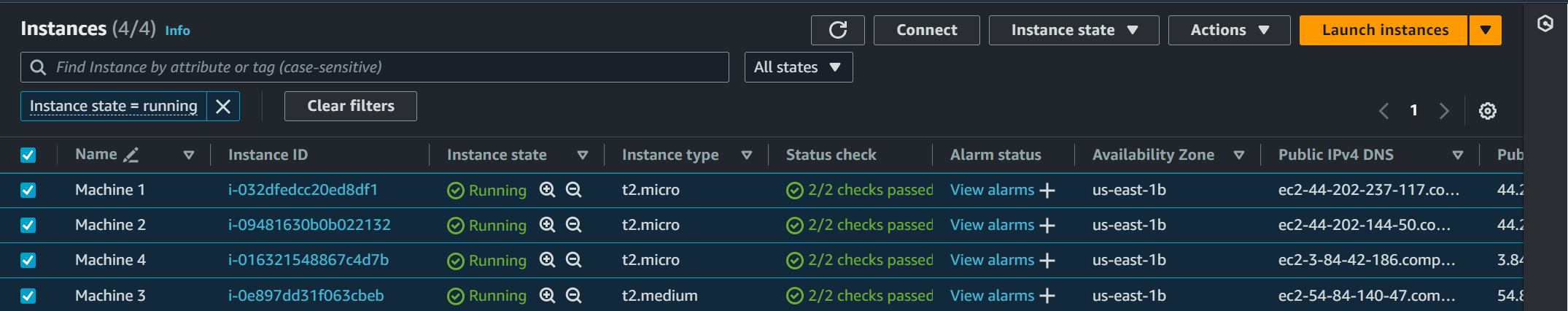
terraform init



terraform plan



terraform apply

**  
Deployed Machine 2, machine 3, and machine 4 by terraform**

**Let’s install ansible**

**Ansible installation script**

sudo apt update

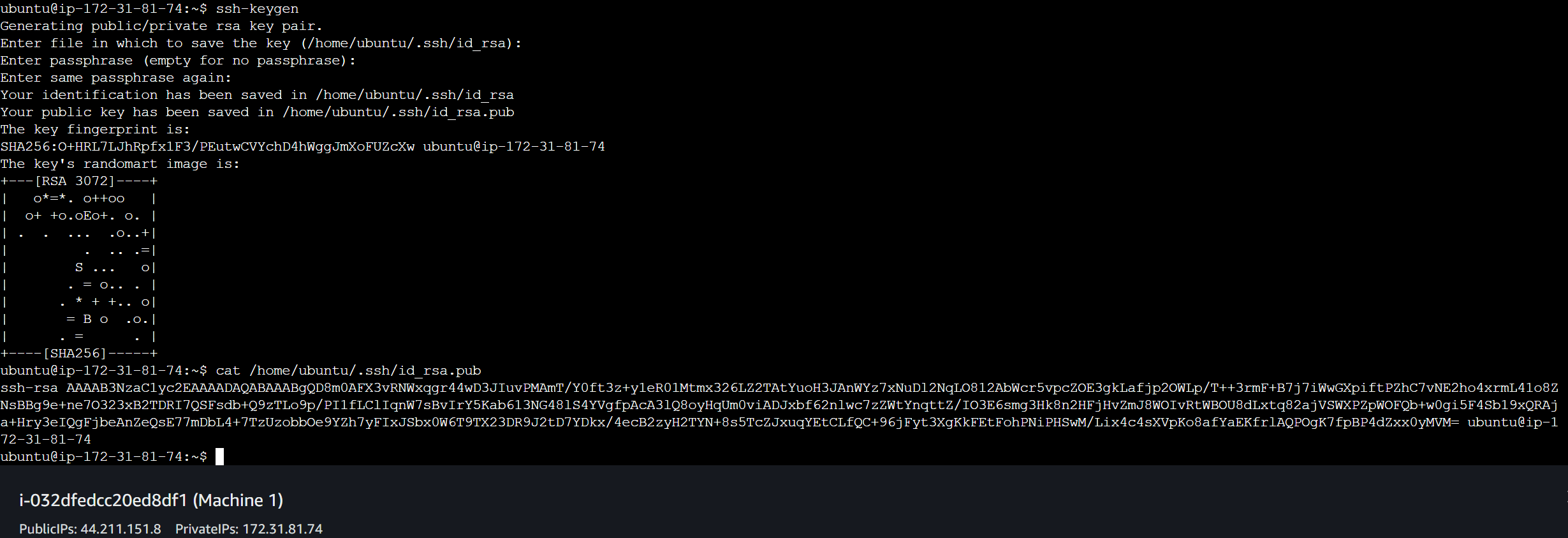
sudo apt install software-properties-common -y

sudo add-apt-repository --yes --update ppa:ansible/ansible

sudo apt install ansible -y

**Now create keygen on the master VM and authorize slave VMs**

ssh-keygen

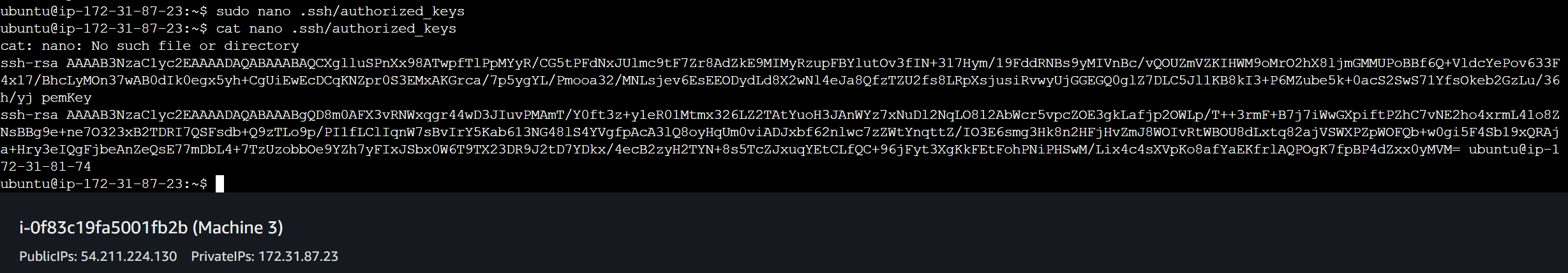
****



Copy this key

cat /home/ubuntu/.ssh/id\_rsa.pub



****

Paste the key in authorized\_keys

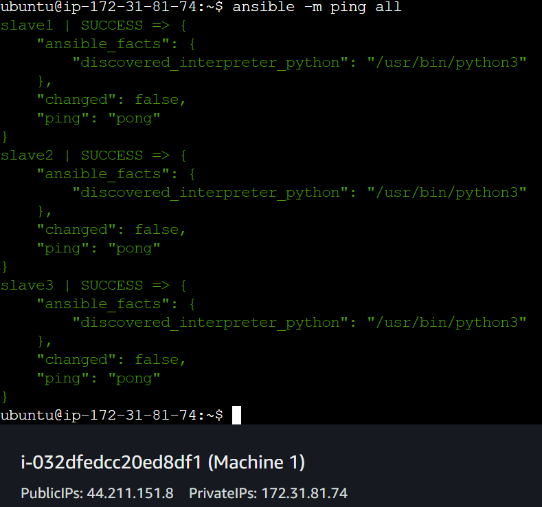
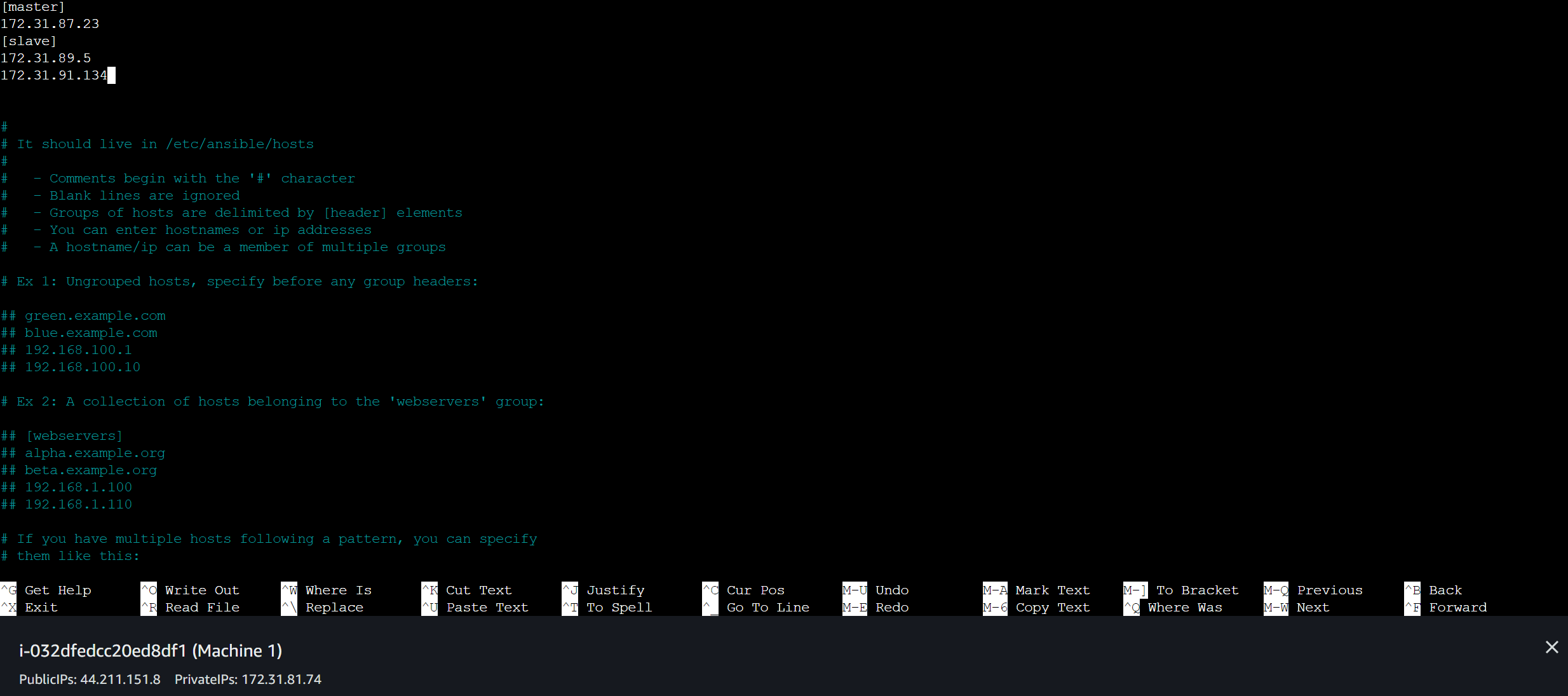
sudo nano .ssh/authorized\_keys



**Now copy the private IPs and paste them into the host file of the Master VM**

Path for ansible host file

****

**** ****

Master and slave group created with Private IPs of slave machines



**Checking the connection between all the slave machines and the master VM**

**Now we have to create an ansible playbook (play.yaml) which will run some scripts (script1, script2, and script3) into the slave machines.**

**script1.sh**

sudo apt update

sudo apt install openjdk-11-jdk -y

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

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

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

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

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

sudo apt-get update

sudo apt-get install jenkins -y

**script2.sh**

sudo apt update

sudo apt install openjdk-11-jdk -y

sudo apt install docker.io -y

sudo apt upgrade -y

sudo apt-get install -y apt-transport-https ca-certificates curl gpg

sudo mkdir -p -m 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo systemctl enable --now kubelet

**script3.sh**

sudo apt update

sudo apt install docker.io -y

sudo apt upgrade -y

sudo apt-get install -y apt-transport-https ca-certificates curl gpg

sudo mkdir -p -m 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo systemctl enable --now kubelet

**play.yaml**

---

- name: installing jenkins and java on machine 1

  hosts: localhost

  become: true

  tasks:

    - name: running script1

      script: script1.sh

- name: installing java, docker and kubernetes on machine 3

  hosts: master

  become: true

  tasks:

    - name: running script2

      script: script2.sh

- name: installing kubernetes and docker on machine 2 and machine 4

  hosts: slave

  become: true

  tasks:

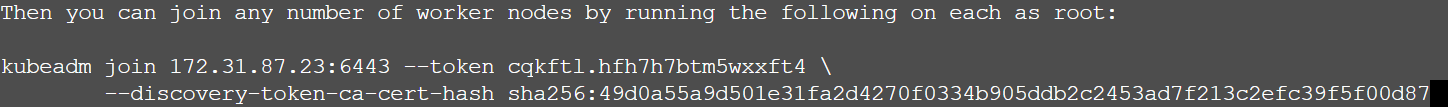
    - name: running script3

      script: script3.sh

ansible-playbook play.yaml – To run the play.yaml file by ansible

sudo kubeadm init – To initiate kubeadm and make the k8s cluster

**You will get a join token for the master VM, copy the join token**



**Copy this command**



**Now we have to run a certain set of commands on the master VM-**

mkdir -p $HOME/.kube

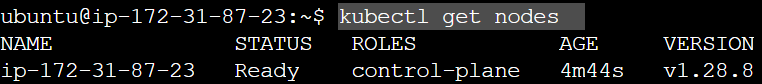
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

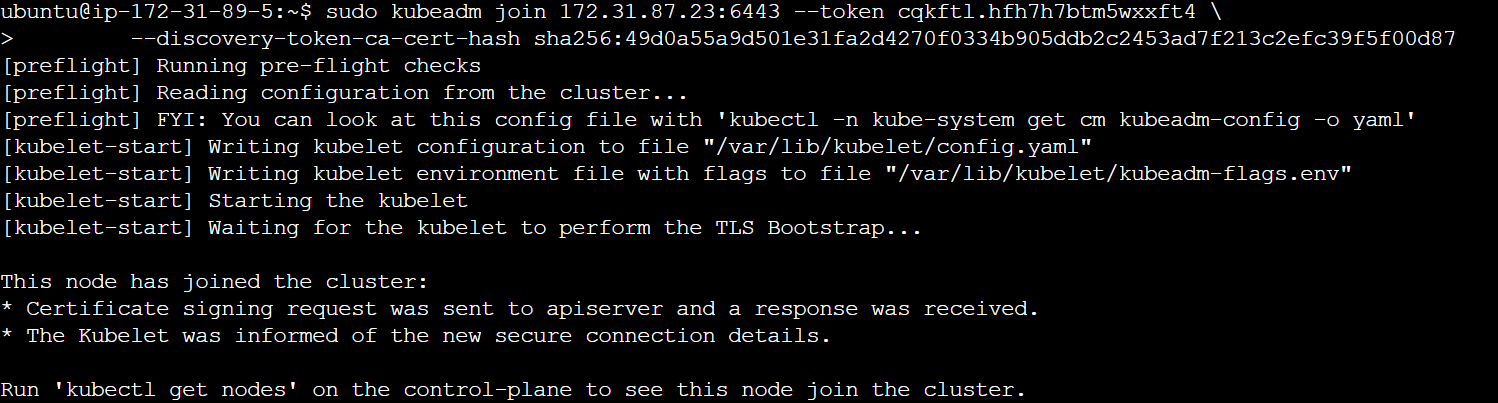
curl https://raw.githubusercontent.com/projectcalico/calico/v3.27.2/manifests/calico.yaml -O

kubectl apply -f calico.yaml

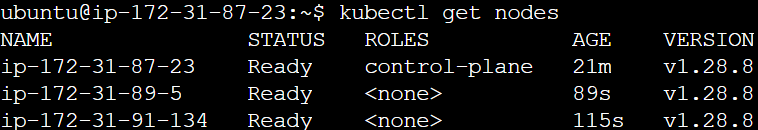
kubectl get nodes – To check whether the master node is ready to work or not



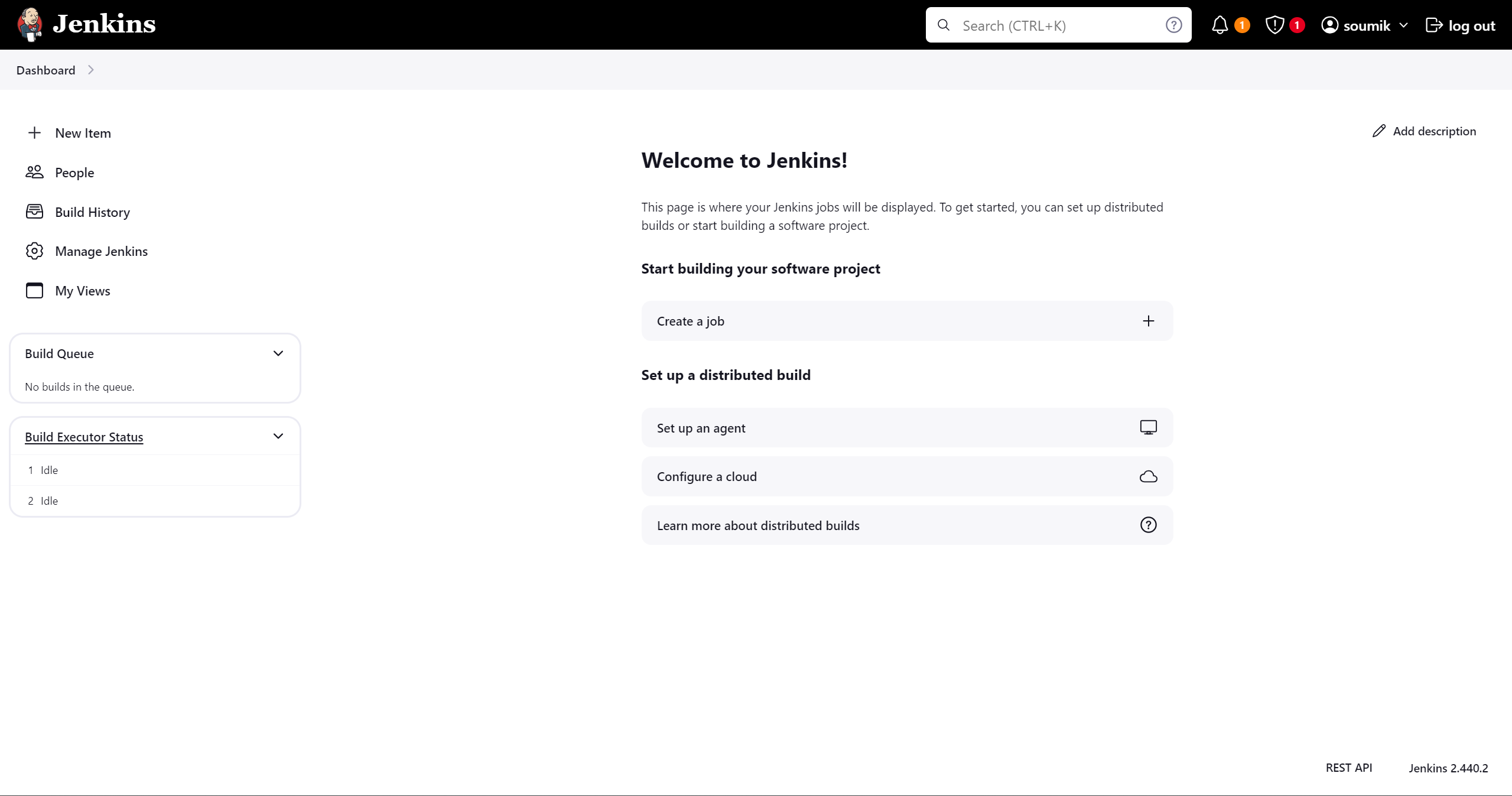
**Now paste the join token with the sudo privilege**



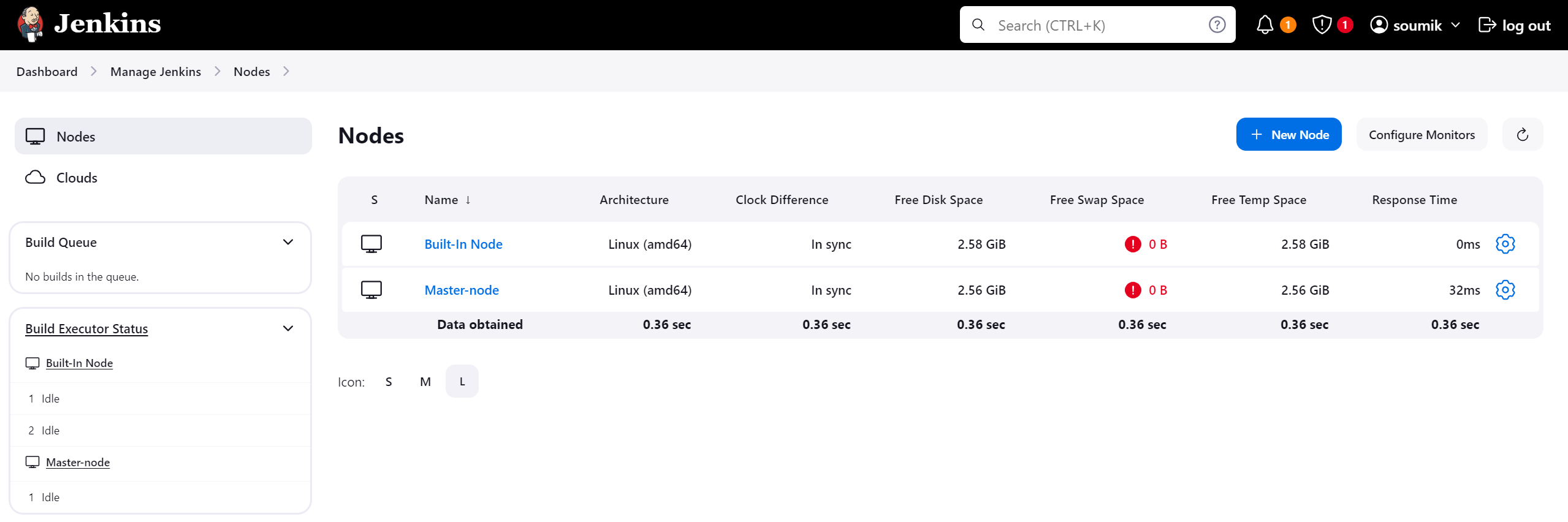
**All nodes are ready to work -**



**Run the Jenkins from Machine 1**

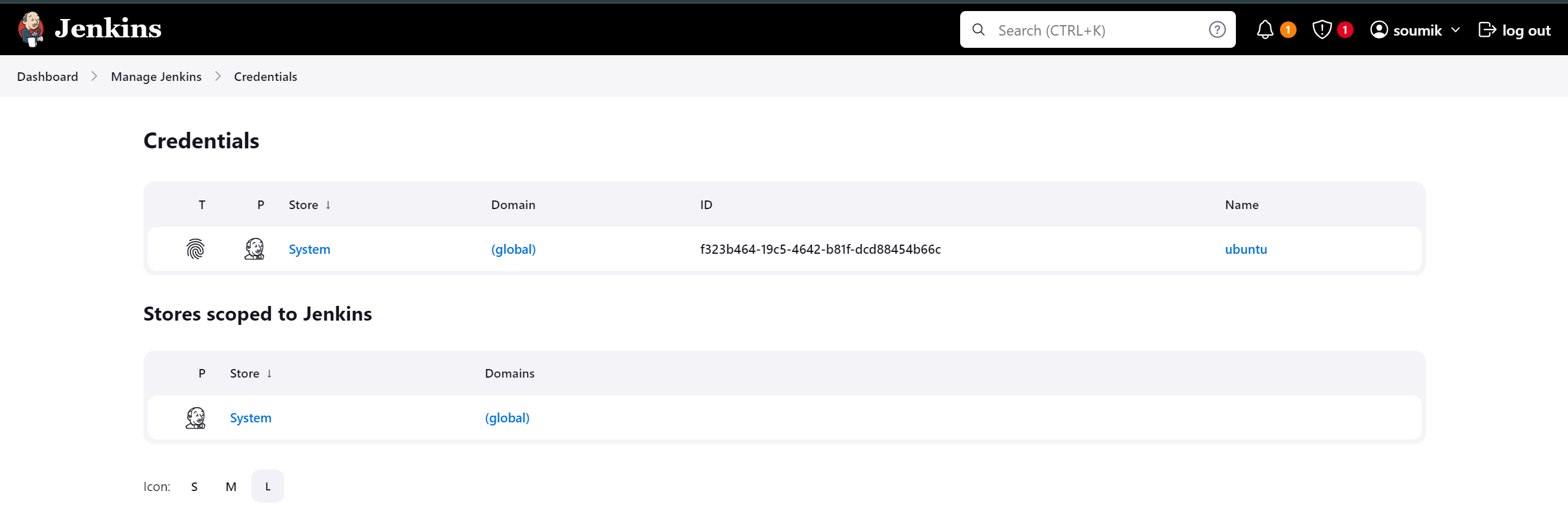
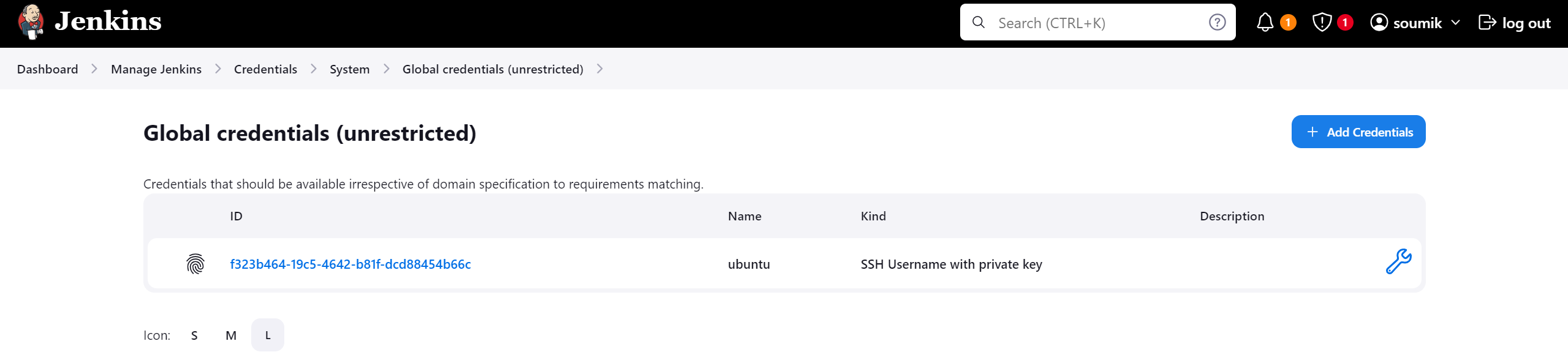
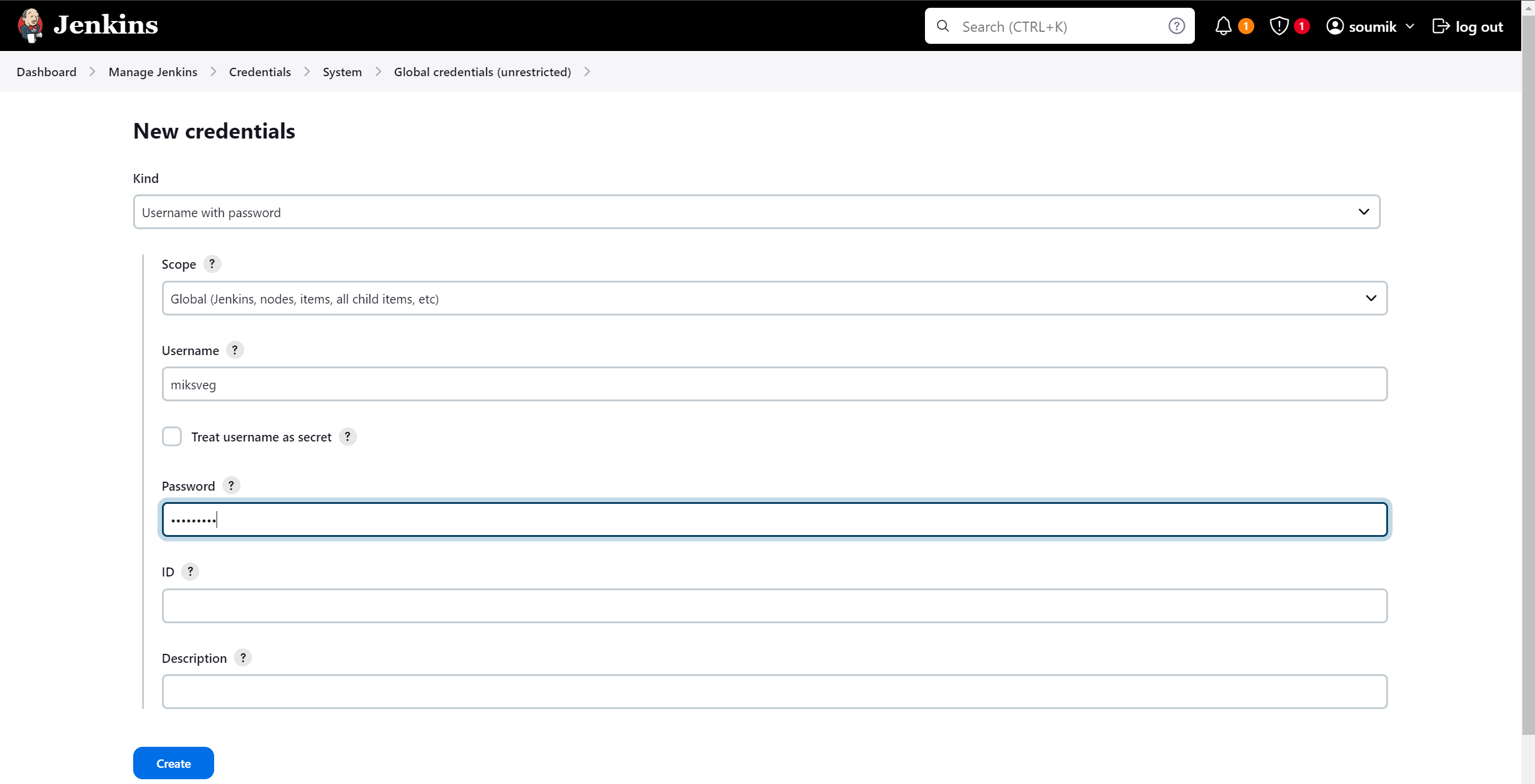
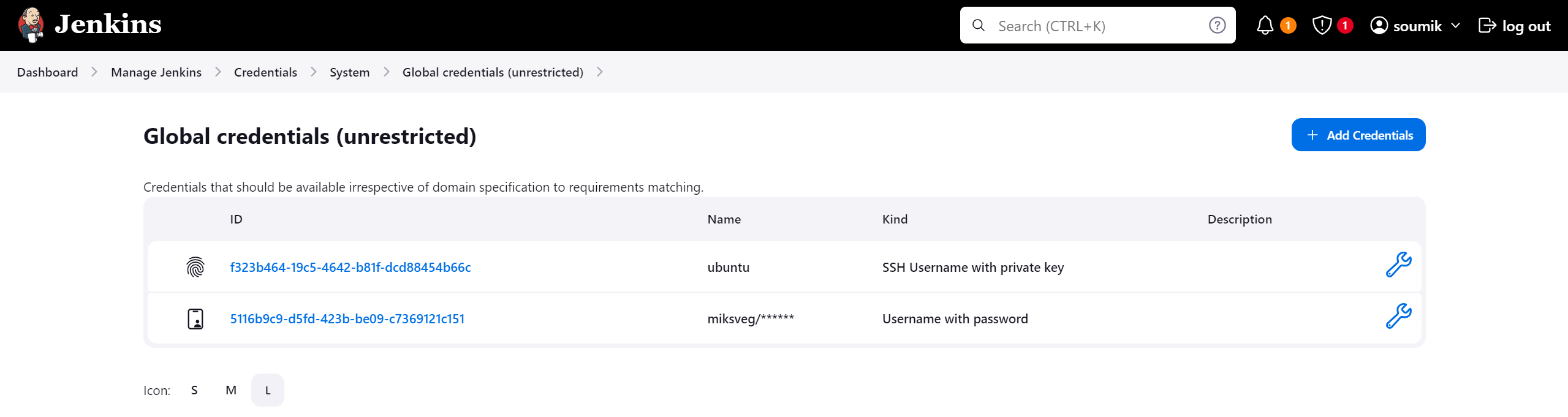


**Setup Jenkins with K8s master VM**

****

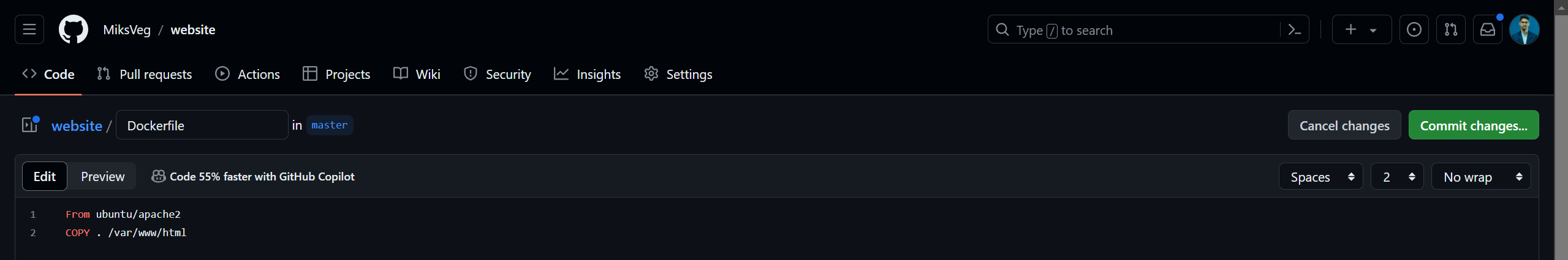
**Now let's add the credentials of Dockerhub Jenkins pipeline can access the Dockerhub**

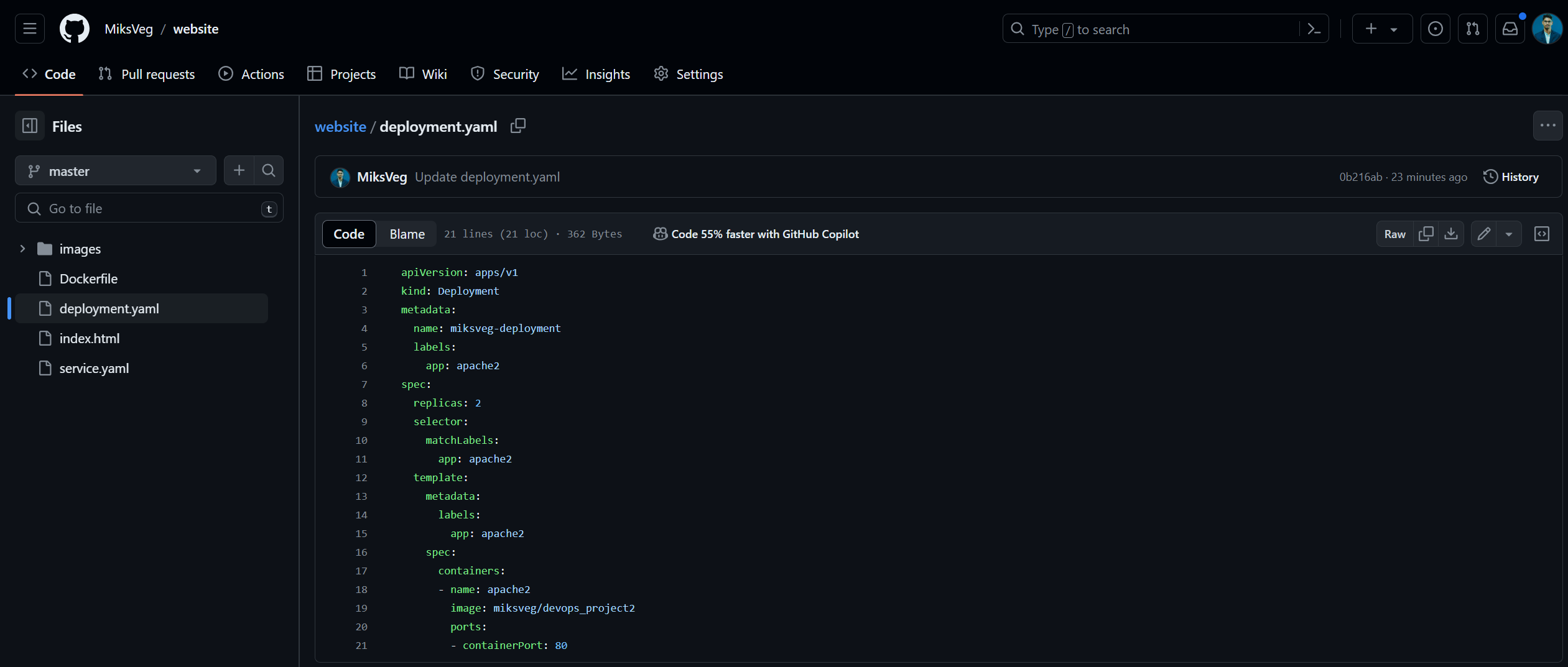
**Path –** Dashboard > Manage Jenkins > Credentials

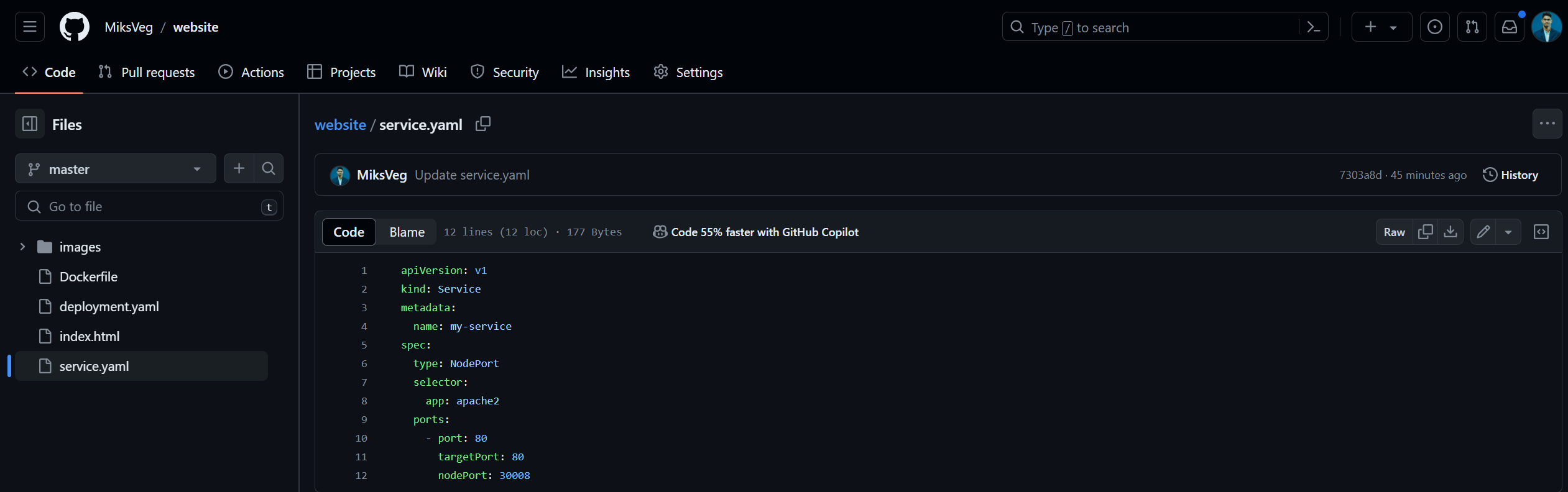
**  
  
**



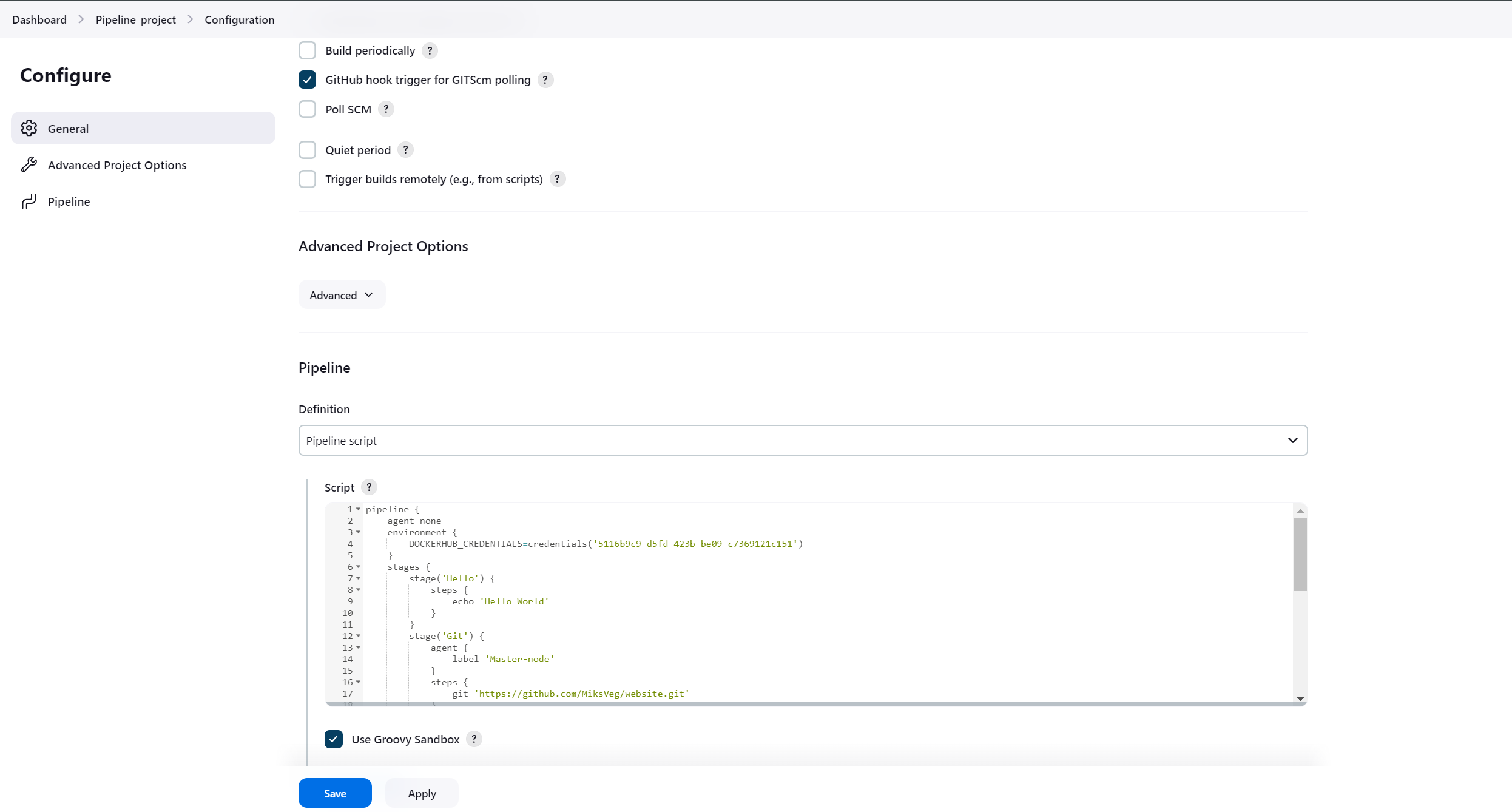
Successfully added the credentials

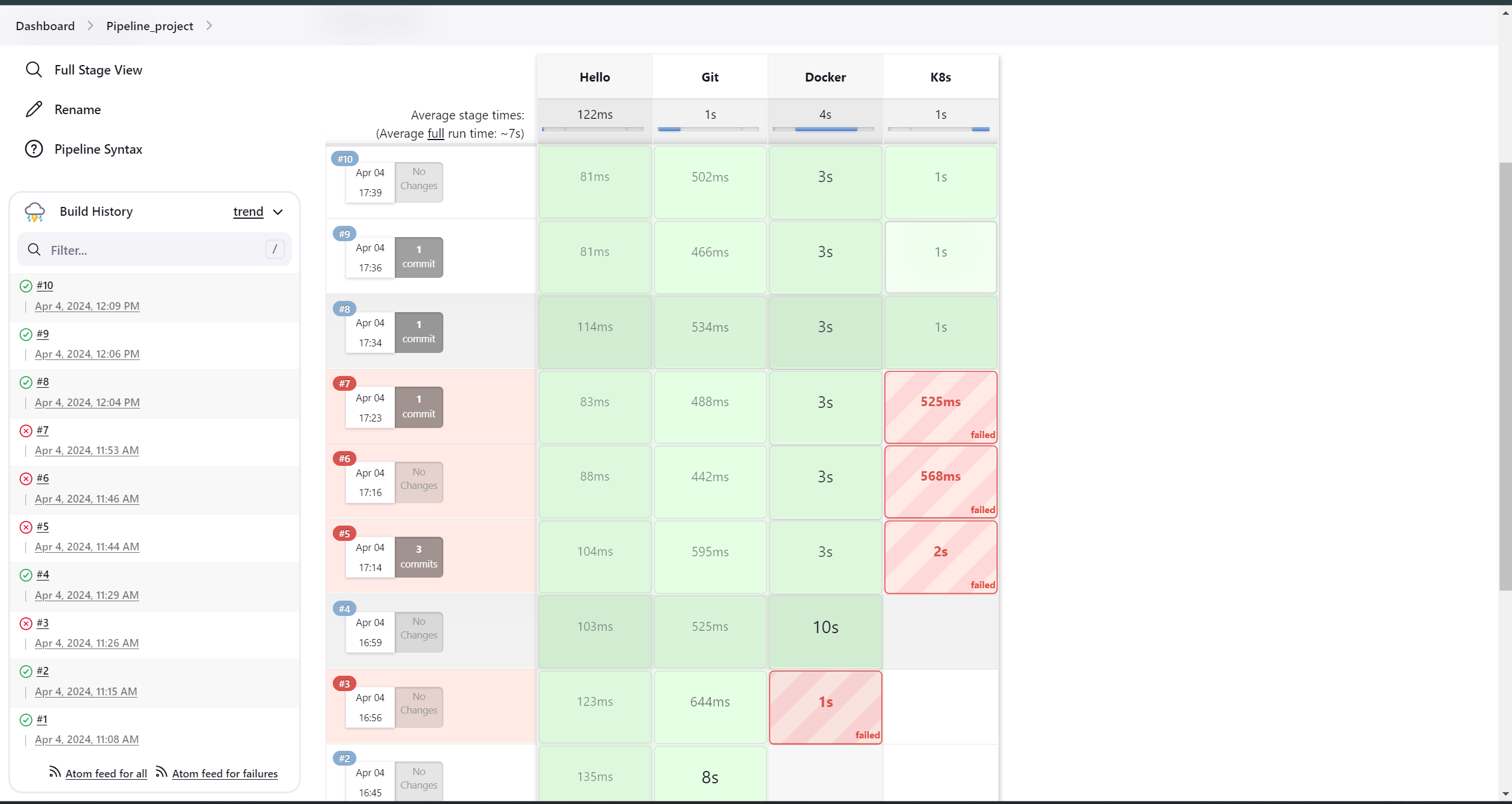
**Now fork the GitHub repo (**[**https://github.com/hshar/website.git**](https://github.com/hshar/website.git)**) and add a Dockerfile  
**

**Created a deployemt.yaml file to deploy Kubernetes pods  
**

**Created service.yaml to start a node port service on port number 30008  
**

**Created a Pipeline with groovy script and successfully run it**

****

**Successfully running after some troubleshooting **

**Pipeline Script-**

pipeline {

agent none

environment {

DOCKERHUB\_CREDENTIALS=credentials('5116b9c9-d5fd-423b-be09-c7369121c151')

}

stages {

stage('Hello') {

steps {

echo 'Hello World'

}

}

stage('Git') {

agent {

label 'Master-node'

}

steps {

git 'https://github.com/MiksVeg/website.git'

}

}

stage('Docker') {

agent {

label 'Master-node'

}

steps {

sh 'sudo docker build /home/ubuntu/jenkins/workspace/Pipeline\_project -t miksveg/devops\_project2'

sh 'sudo echo $DOCKERHUB\_CREDENTIALS\_PSW | sudo docker login -u $DOCKERHUB\_CREDENTIALS\_USR --password-stdin'

sh 'sudo docker push miksveg/devops\_project2'

}

}

stage('K8s') {

agent {

label 'Master-node'

}

steps {

sh 'kubectl delete deploy miksveg-deployment'

sh 'kubectl apply -f deployment.yaml'

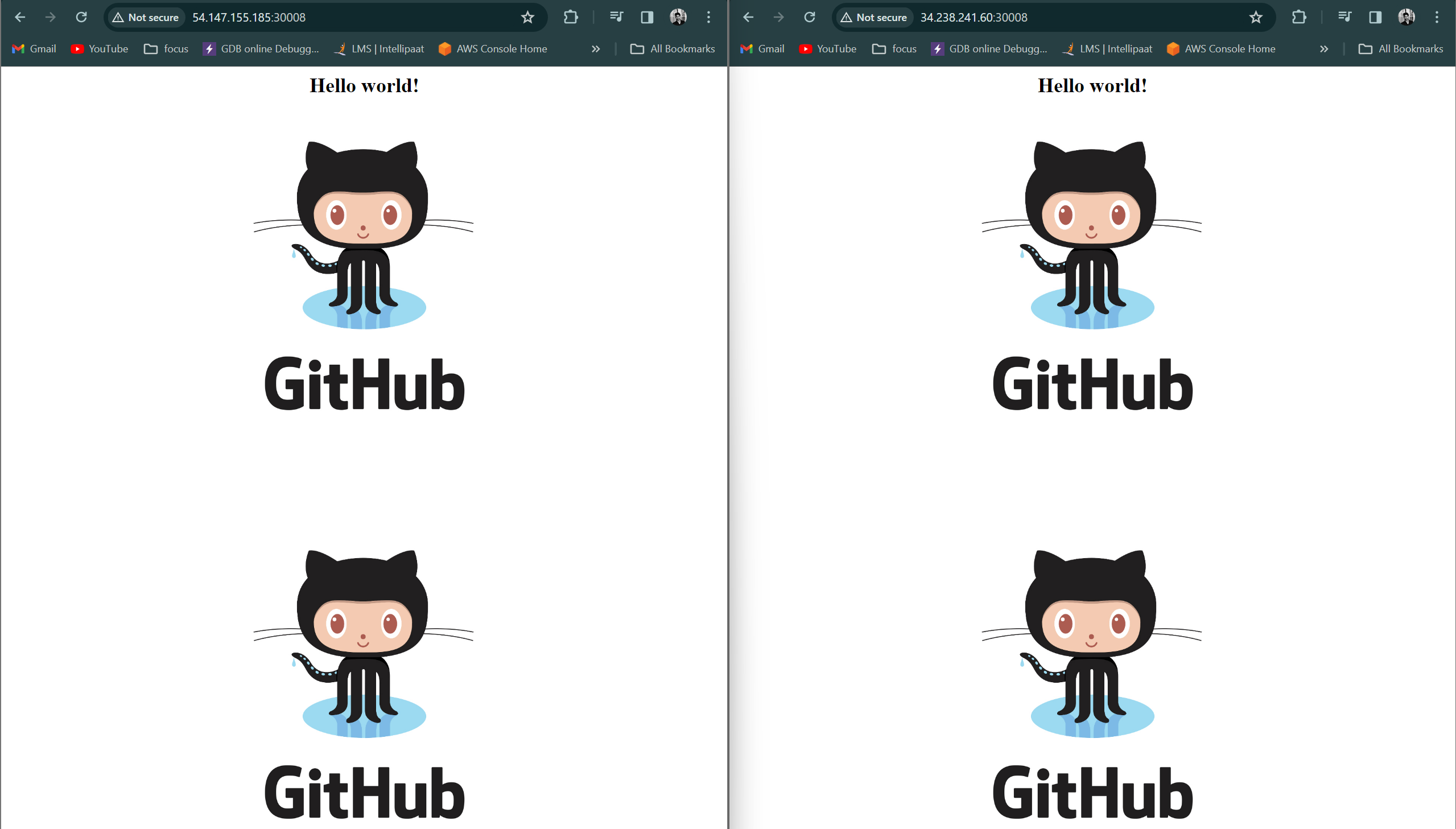
sh 'kubectl apply -f service.yaml'

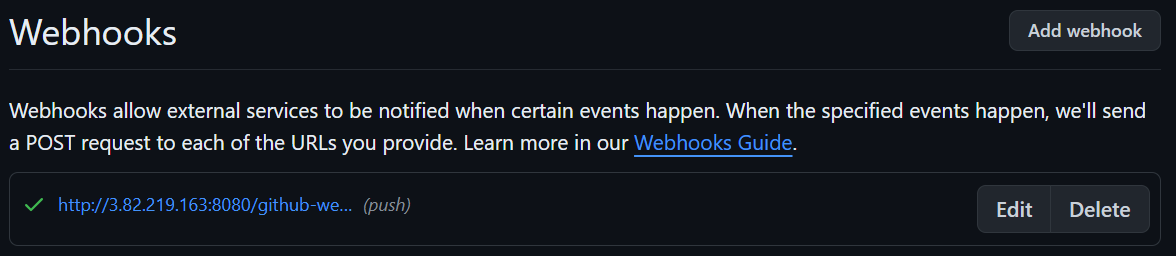
}

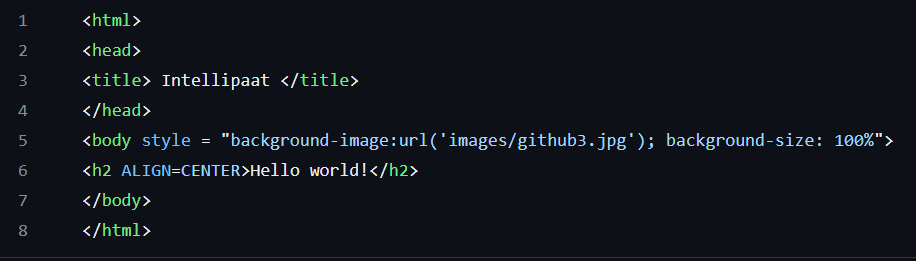
}

}

}

**Successfully running the deployment**

**At last created a webhook for automatically trigger when made a change in repo**

**Now commit a change in repo**



****



**--------------------------------------------------------end of the project ----------------------------------------------------**