**AWS EKS Using Terraform And Deploy Sample Applications**

**Pre-requisites:**

* Install Terraform-0.11.14v
* Install Git

**Install Terraform-0.11.14v:**

wget <https://releases.hashicorp.com/terraform/0.11.14/terraform_0.11.14_linux_amd64.zip>

unzip terraform\_0.11.14\_linux\_amd64.zip

cp terraform /usr/bin

terraform --version

**Install Git:**

yum install git -y

**Clone Terraform code from GITHUB to create EKS cluster:**

git clone <https://github.com/Naresh240/terraform-eks-cluster.git>

cd terraform-eks-cluster

This terraform script will create IAM roles, VPC, EKS, and worker node, it will also create kubernetes server to configure kubectl on EKS.

Note: This terraform will also take workstation IP, so you don't have to create a Kubernetes server separately.

terraform init

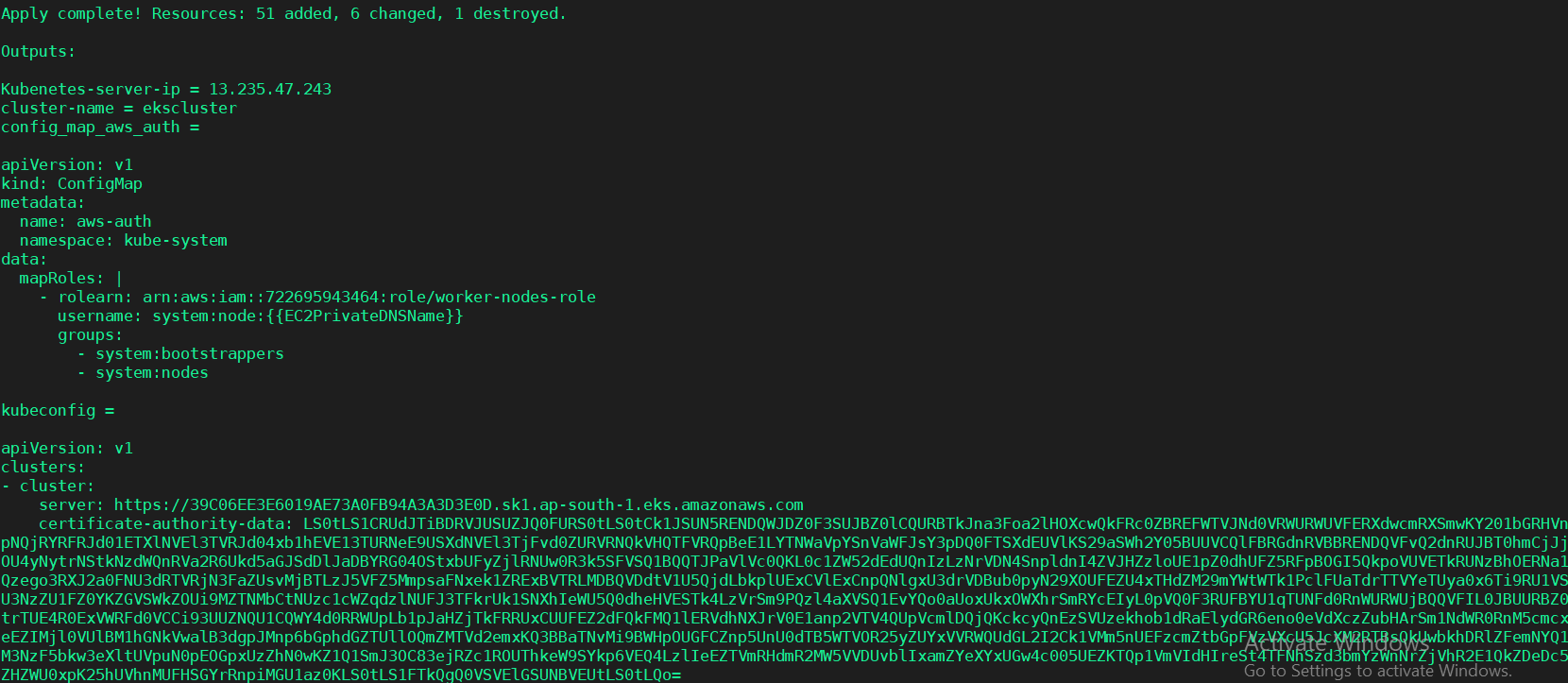
terraform plan

terraform apply

**Give details like as below:**



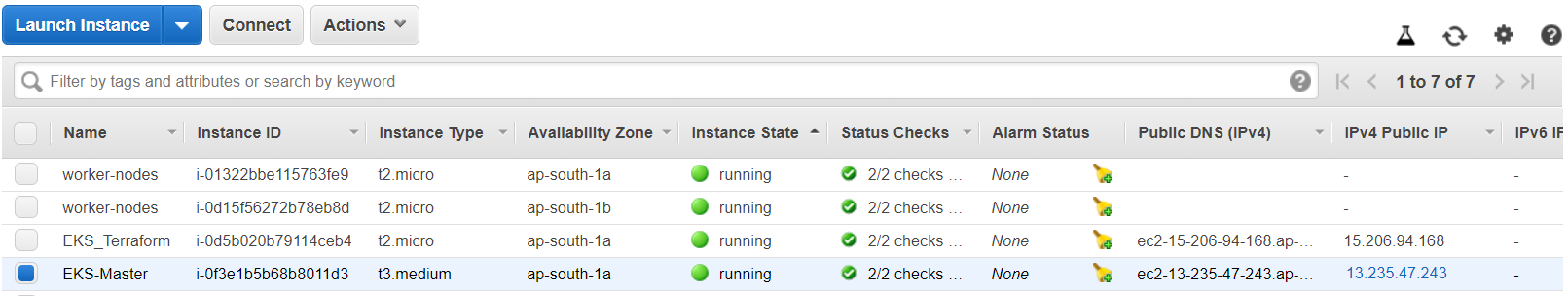
**Output:**



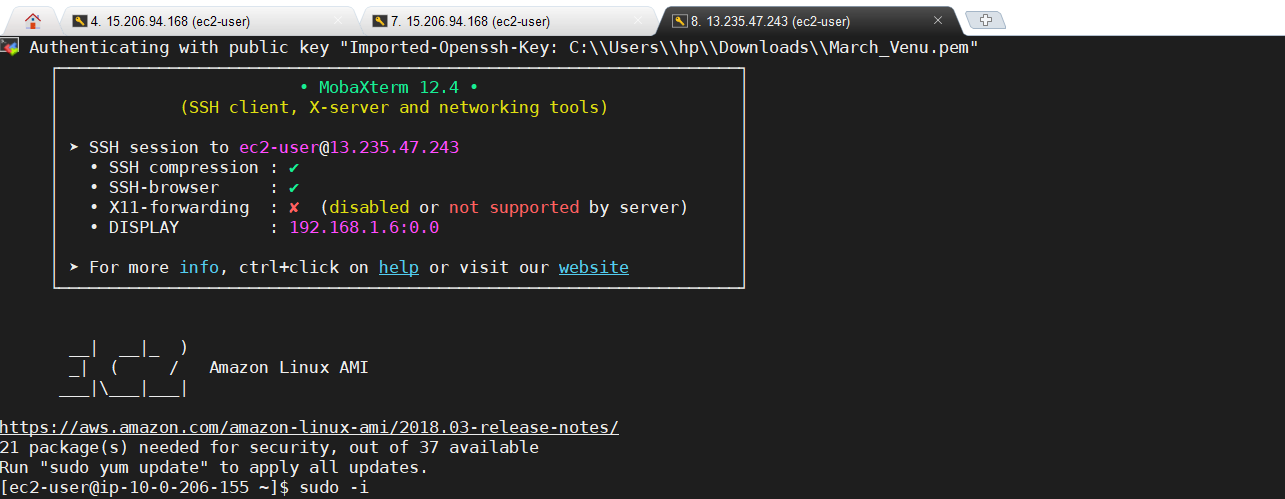
**Open Master server and Install:**

* kubectl
* aws configure
* aws-iam-authenticator
* Configure kubectl for Amazon EKS
* To enable worker nodes to join your cluster

**Open EC2 Service and check whether servers created or not:**



**Connect to Master server and Install:**



**Configure AWS cloud to master:**

aws configure



**Install kubectl:**

curl -o kubectl https://amazon-eks.s3-us-west-2.amazonaws.com/1.14.6/2019-08-22/bin/linux/amd64/kubectl

chmod +x ./kubectl

mkdir -p $HOME/bin

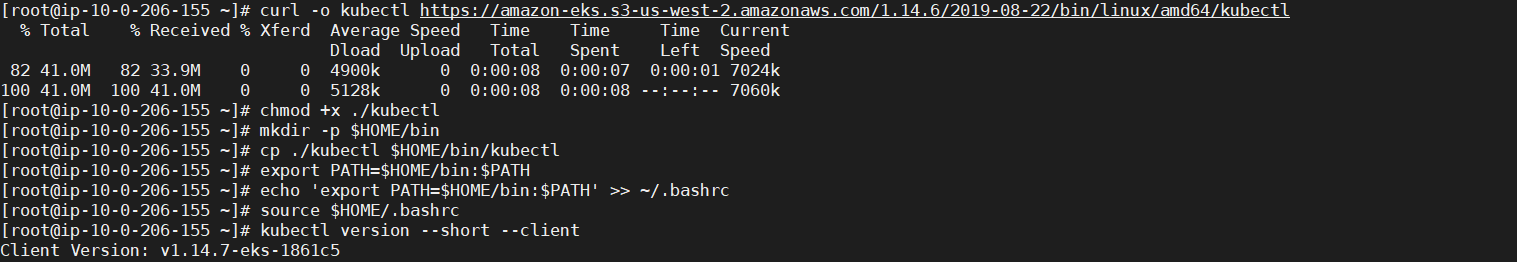
cp ./kubectl $HOME/bin/kubectl

export PATH=$HOME/bin:$PATH

echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc

source $HOME/.bashrc

kubectl version --short –client



**Install aws-iam-authenticator:**

curl -o aws-iam-authenticator https://amazon-eks.s3-us-west-2.amazonaws.com/1.14.6/2019-08-22/bin/linux/amd64/aws-iam-authenticator

chmod +x ./aws-iam-authenticator

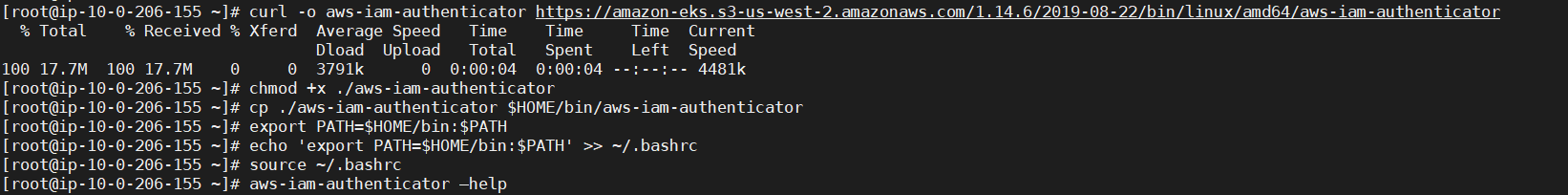
cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator

export PATH=$HOME/bin:$PATH

echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc

source ~/.bashrc

aws-iam-authenticator –help



**Configure kubectl for Amazon EKS:**

Before creating kubeconfig file use aws configure. Here you can also create a profile and add it to kubeconfig file

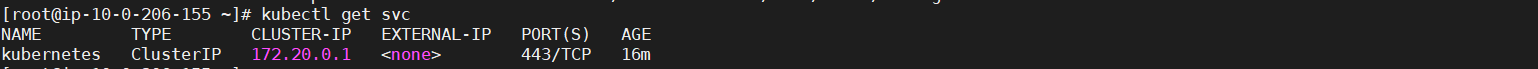
aws eks --region ap-south-1 update-kubeconfig --name eks-cluster



Test your configuration

kubectl get svc

**Output:**



**To enable worker nodes to join your cluster:**

Download, edit, and apply the AWS IAM Authenticator configuration map

curl -o aws-auth-cm.yaml https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2019-02-11/aws-auth-cm.yaml

open the aws-auth-cm.yaml file any editor. Replace the ARN of instance role (not instance profile)

Note: do not change any other line in this file

vi aws-auth-cm.yaml

apiVersion: v1

kind: ConfigMap

metadata:

name: aws-auth

namespace: kube-system

data:

mapRoles: |

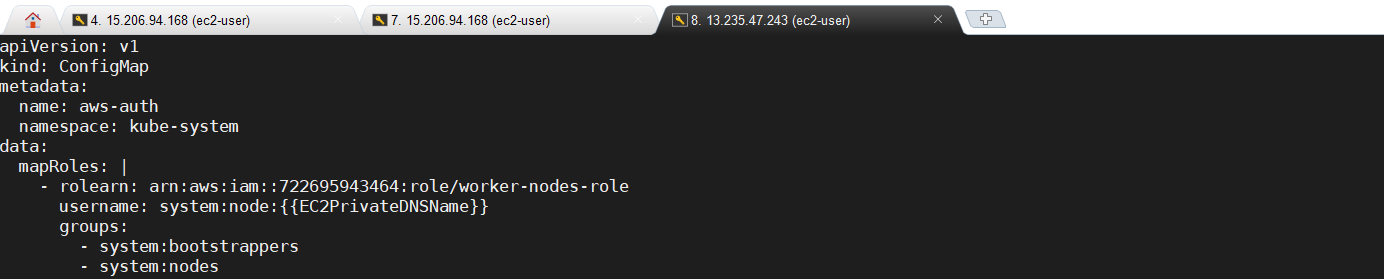
- rolearn: arn:aws:iam::722695943464:role/worker-nodes-role

username: system:node:{{EC2PrivateDNSName}}

groups:

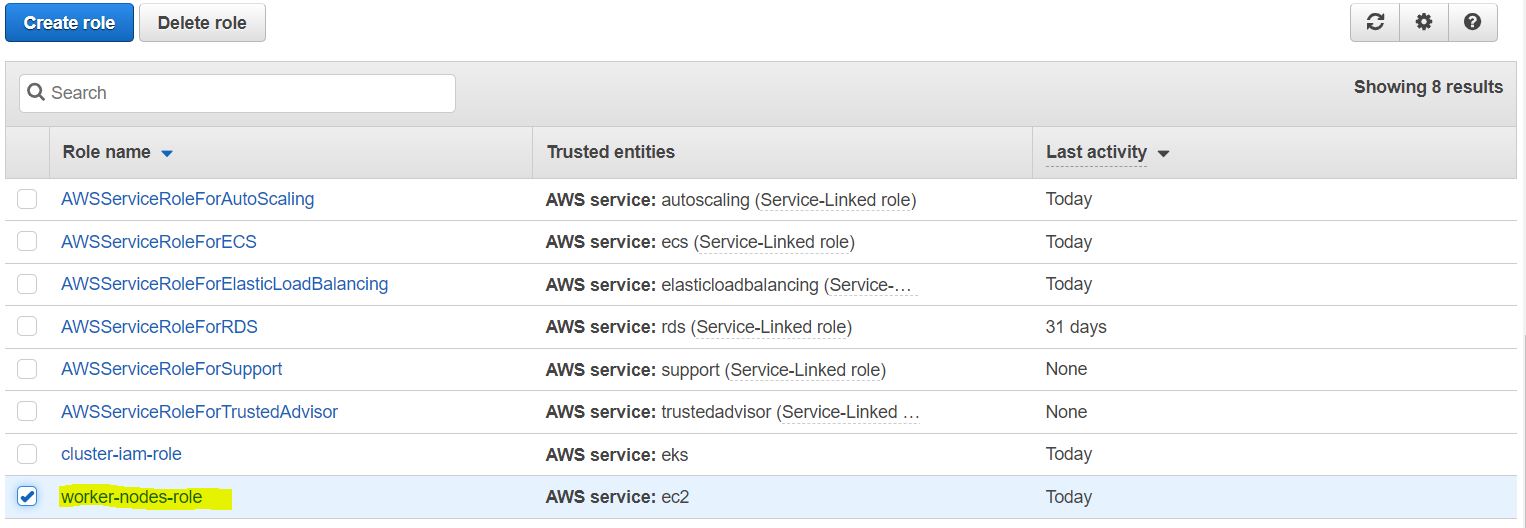
- system:bootstrappers

- system:nodes

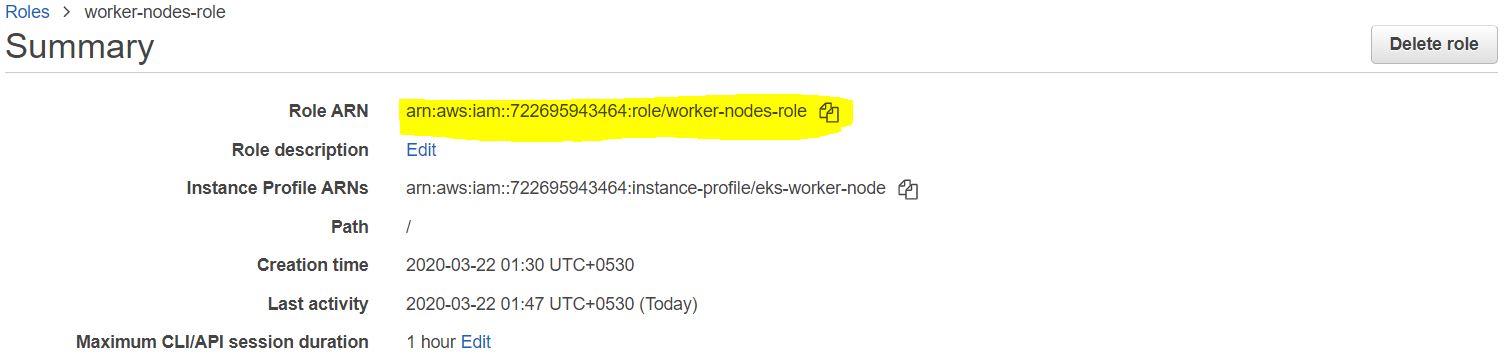


Goto IAM Service 🡪 Roles

Open worker-node-role and copy **arn**:

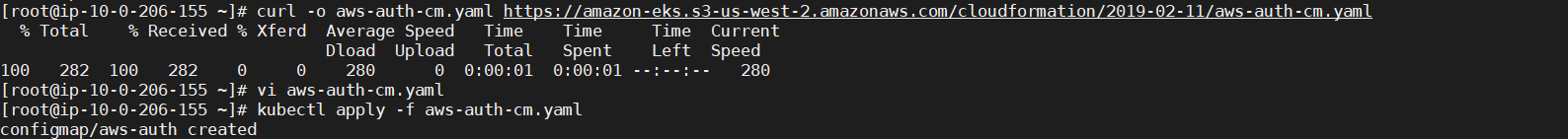


Clickon worker-nodes-role



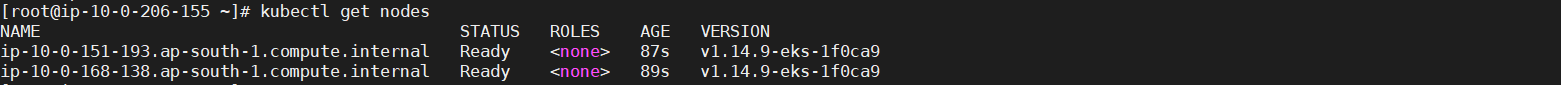
Apply the configuration. This command may take a few minutes to finish

kubectl apply -f aws-auth-cm.yaml



Watch the status of your nodes and wait for them to reach the Ready status

kubectl get nodes



**Deploy the Nginx container to the Cluster:**

Kubernetes Cluster is now ready, it’s time to deploy the Nginx container.

On the Master Node, run the following command to create an Nginx deployment:

kubectl create deployment nginx --image=nginx

Output:



You can list out the deployments with the following command:

kubectl get pods

Output:



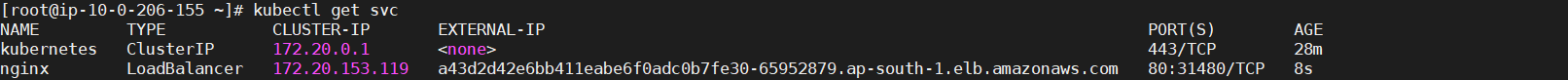
After, we will need to make the Nginx container available to the network with this command:

kubectl create service loadbalancer nginx --tcp=80:80



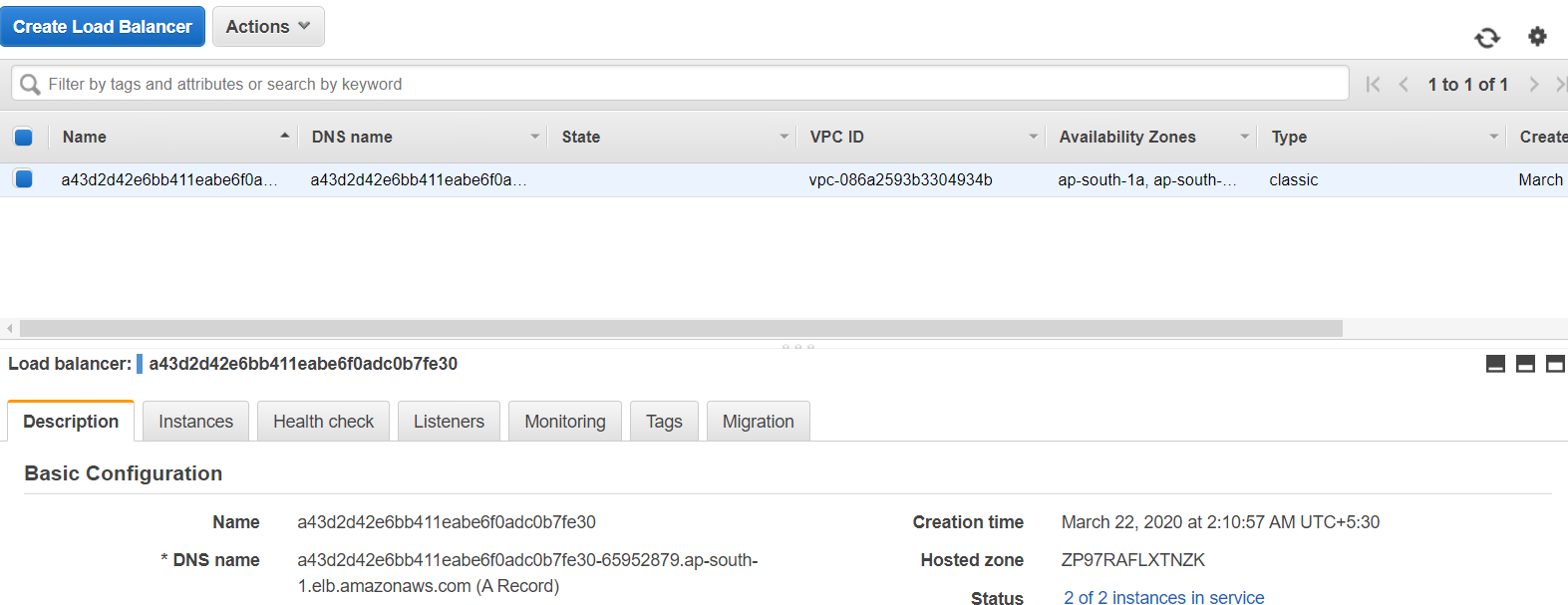
Now, list out all the services by running the following command:

kubectl get svc

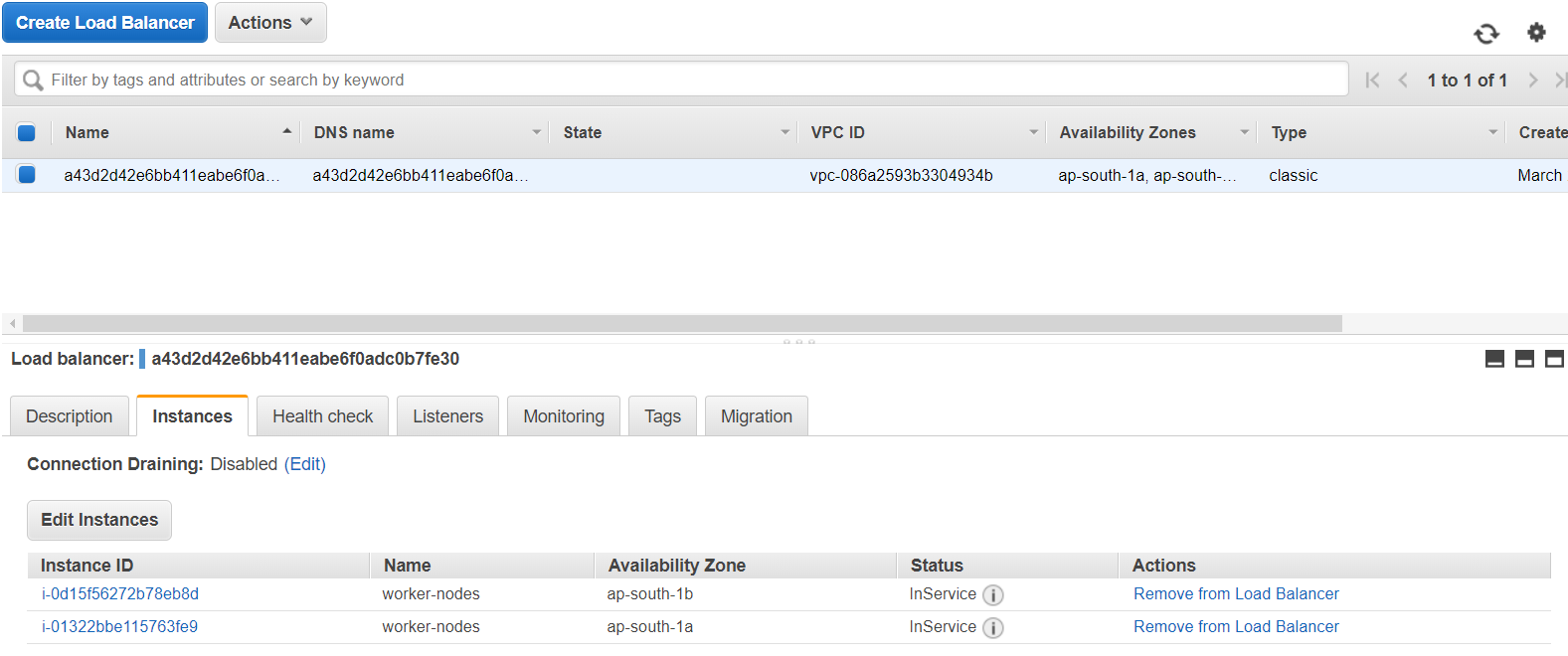


Check whether LoadBalancer created or not in UI:

Goto EC2 service 🡪 Load Balancers

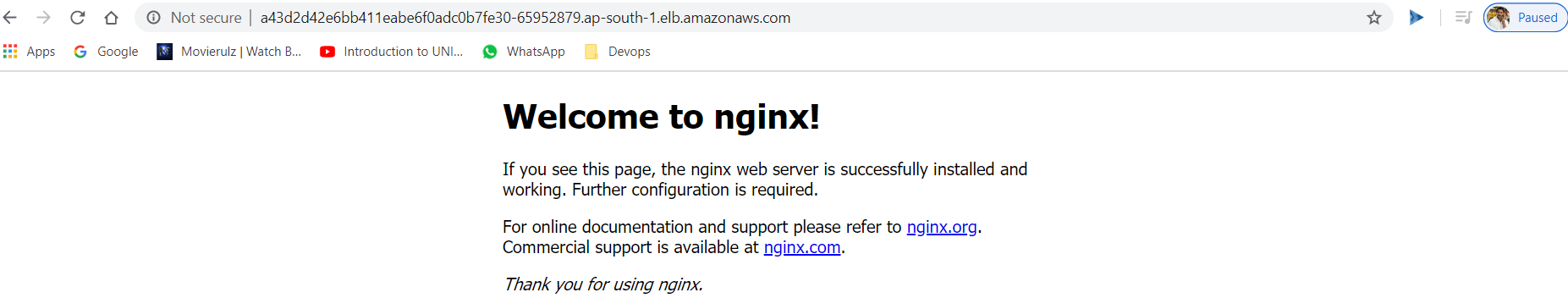


Check whether the nodes are comes InService or not



Once they comes Inservice check output in UI:

<http://a43d2d42e6bb411eabe6f0adc0b7fe30-65952879.ap-south-1.elb.amazonaws.com/>



**CleanUP:** (Goto Terraform Server)

terraform destroy