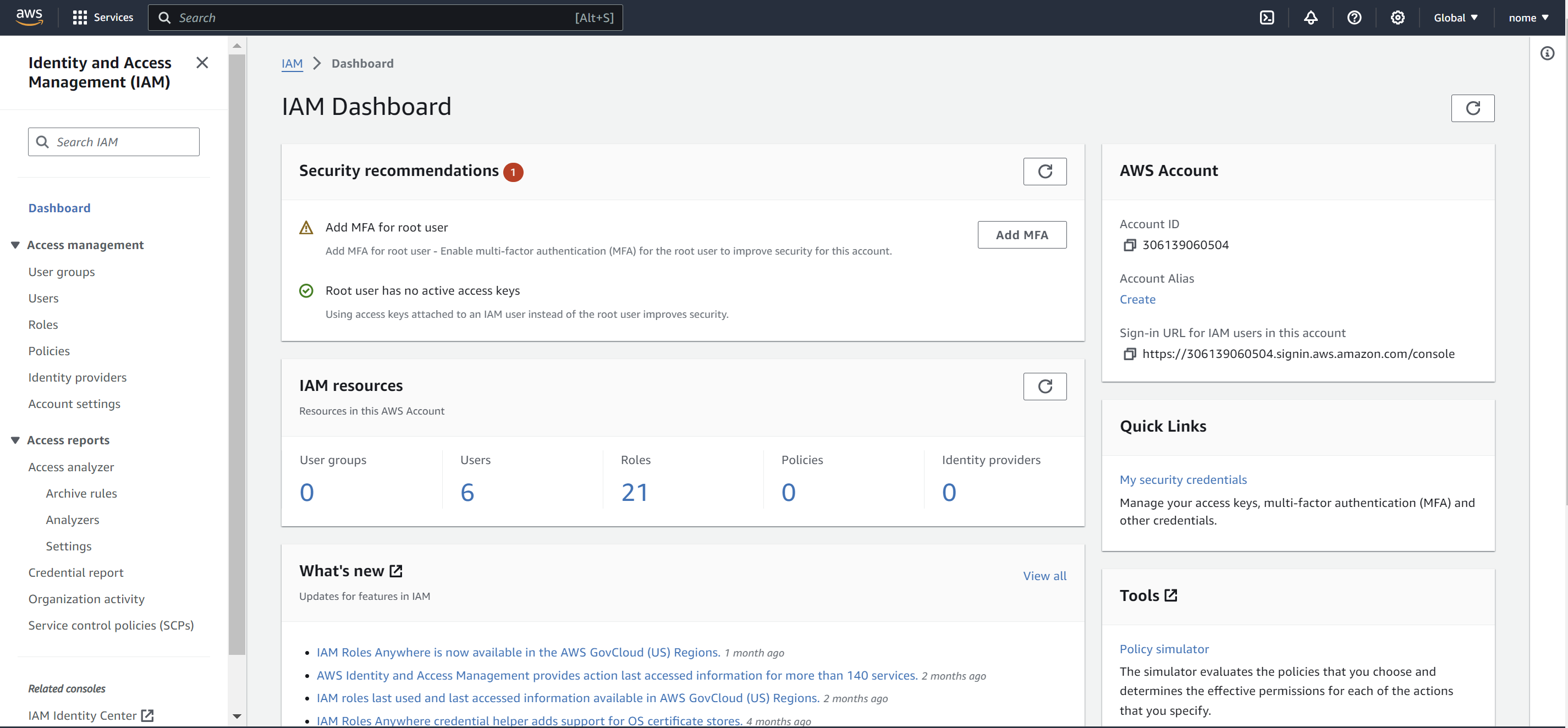
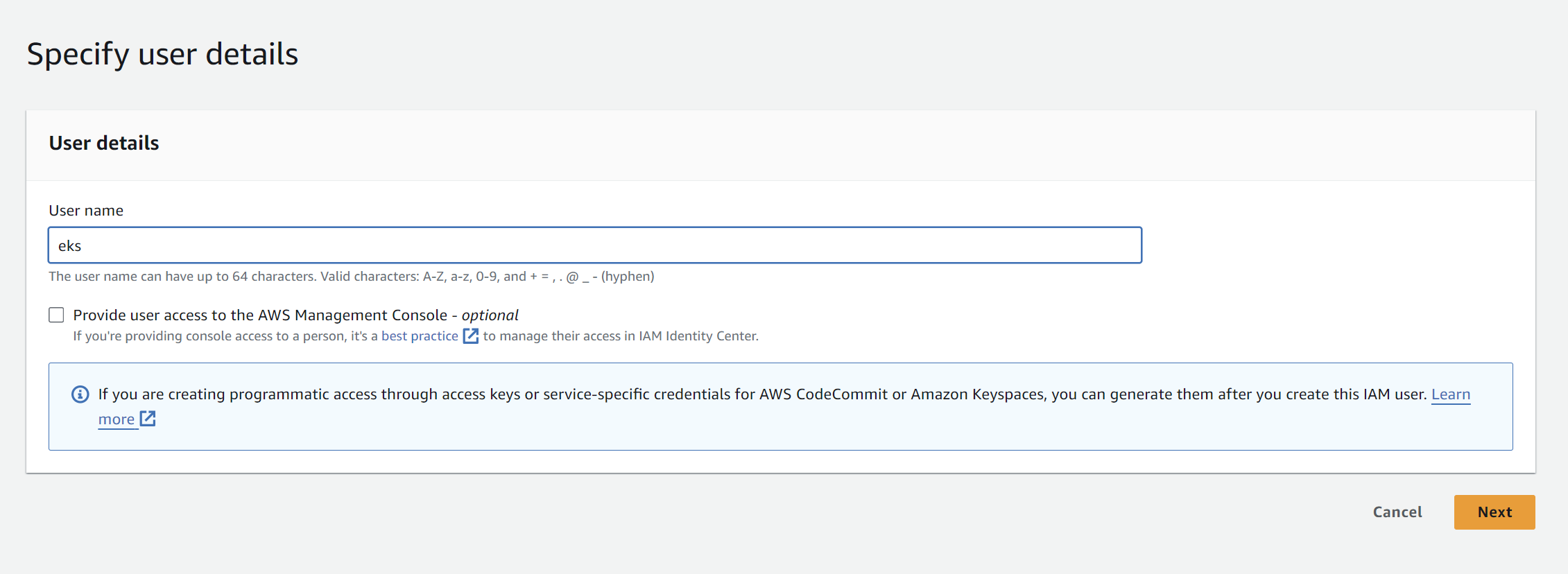
**AWS EKS**

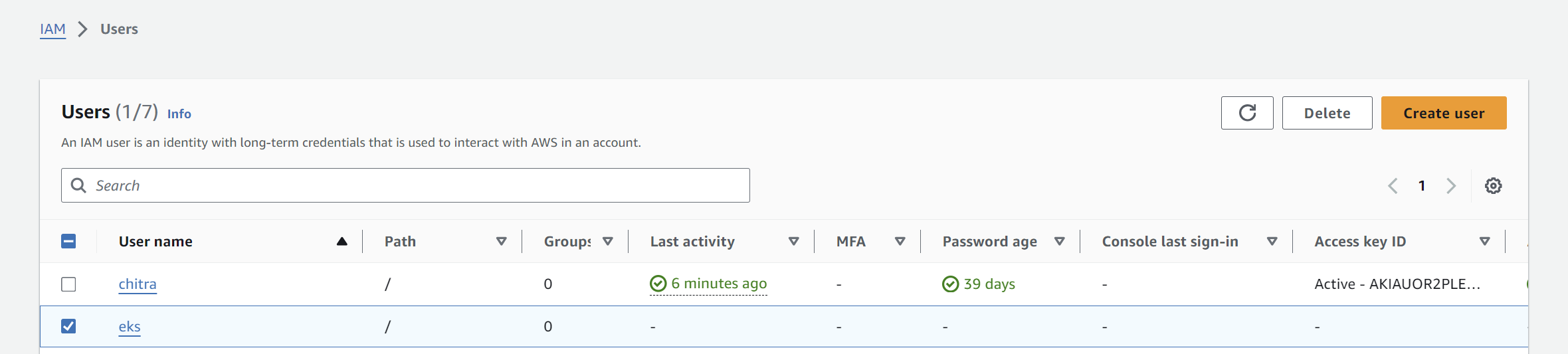
**Step-1:** Go to **IAM console** and create a user.



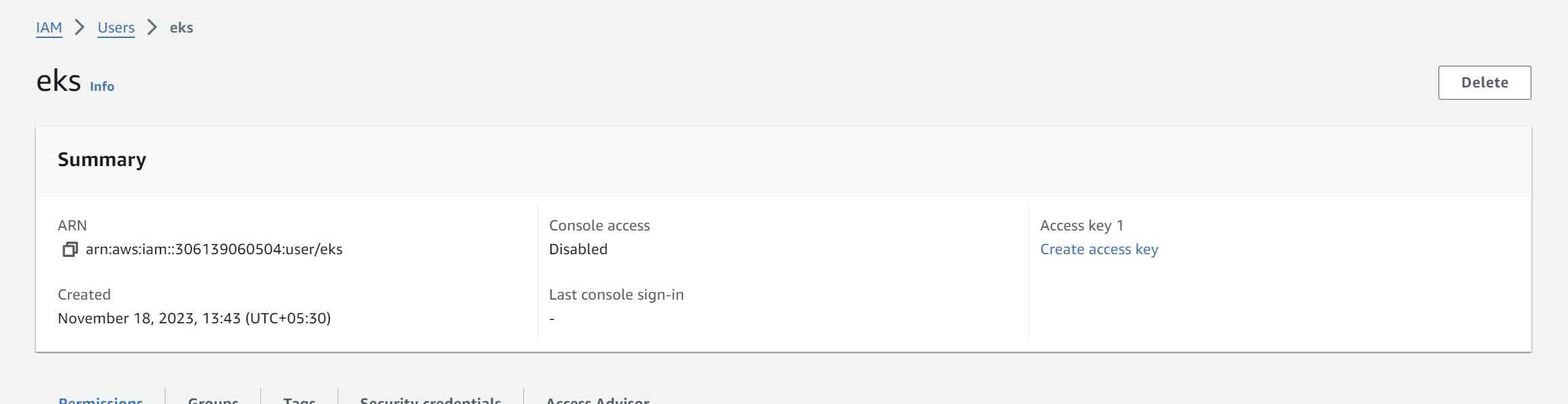
**Step-2:** Name the user.



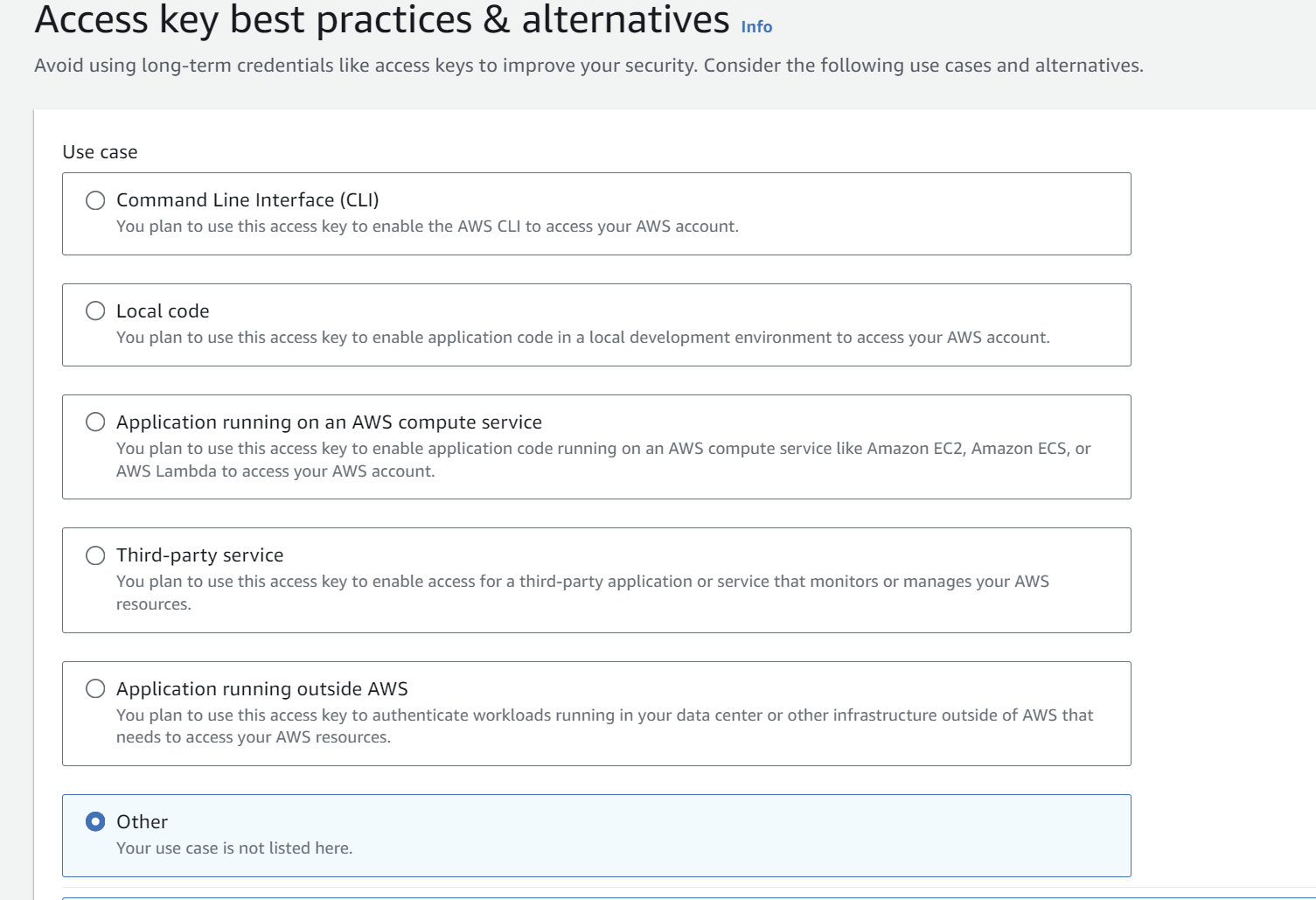
**Step-3:** Now click on the user that you just created. Here it is “**eks**”.



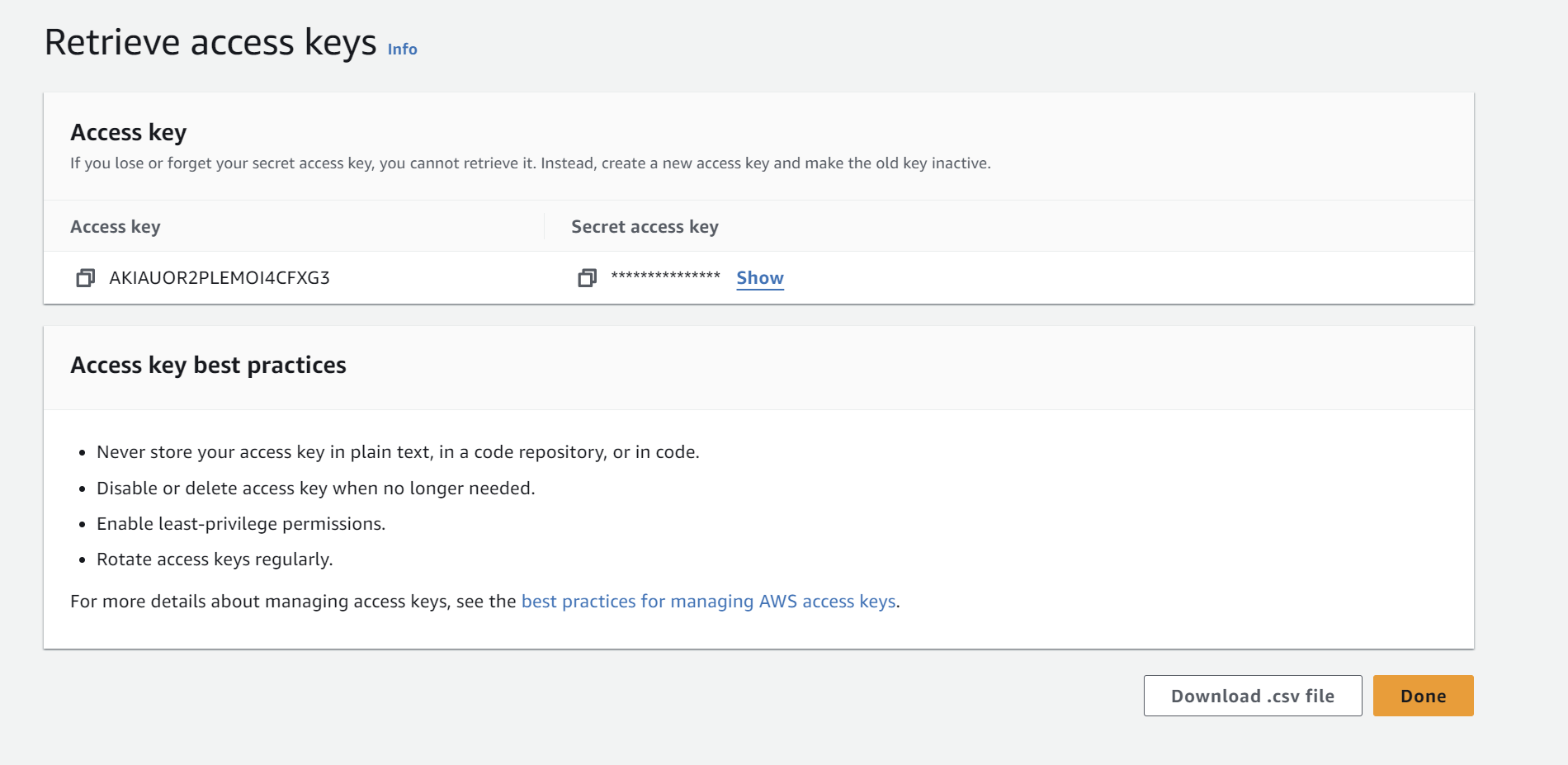
**Step-4:** Click on “**Create access key**”.



**Step-5:** Choose the option “**Other**” and click on next. Leave the other details as default.

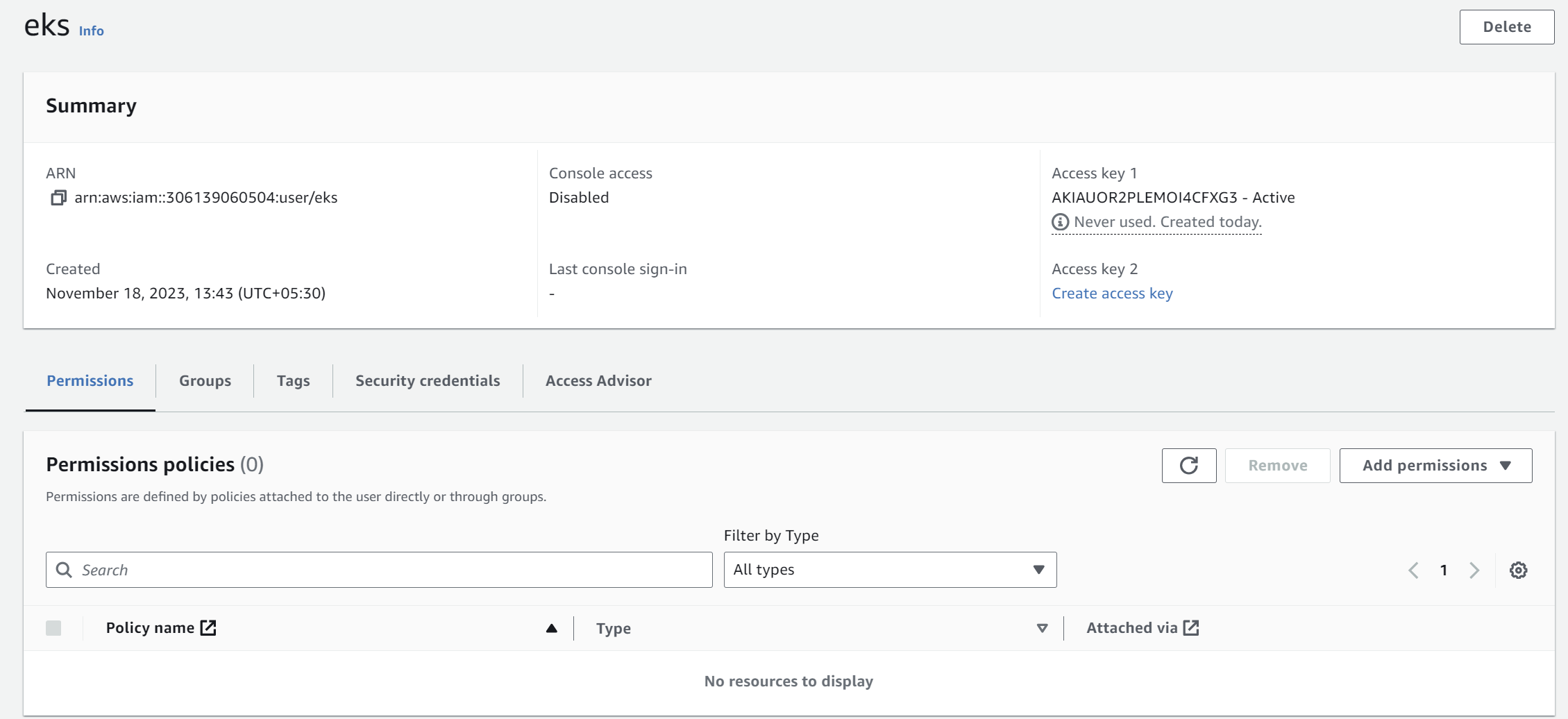


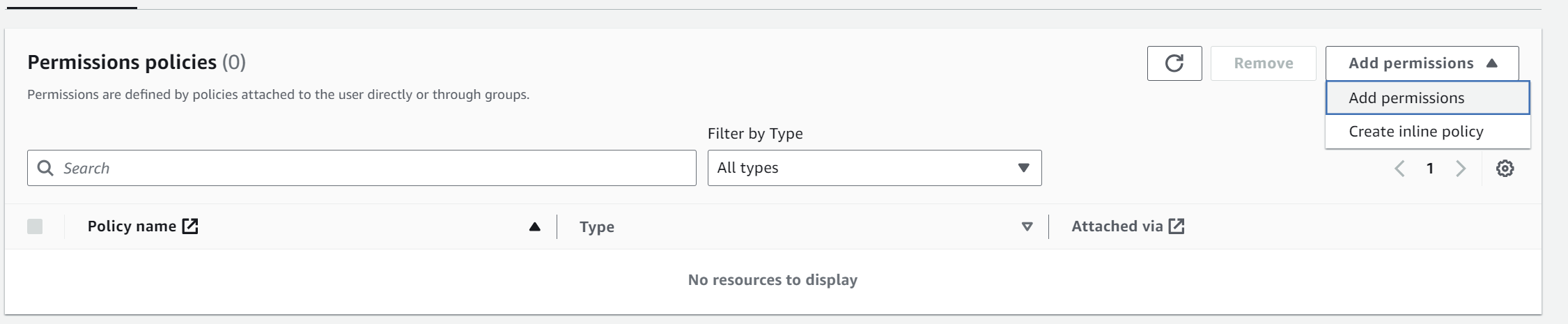
**Step-6:** Now copy the “**Access key**” and “**Secret Access Key**” and save it somewhere as it can’t be retrieved later.



**Step-7:** Now go to “**Add Permissions**”.

**Note:** You can attach the policies while creating the user or after creating the user as well.



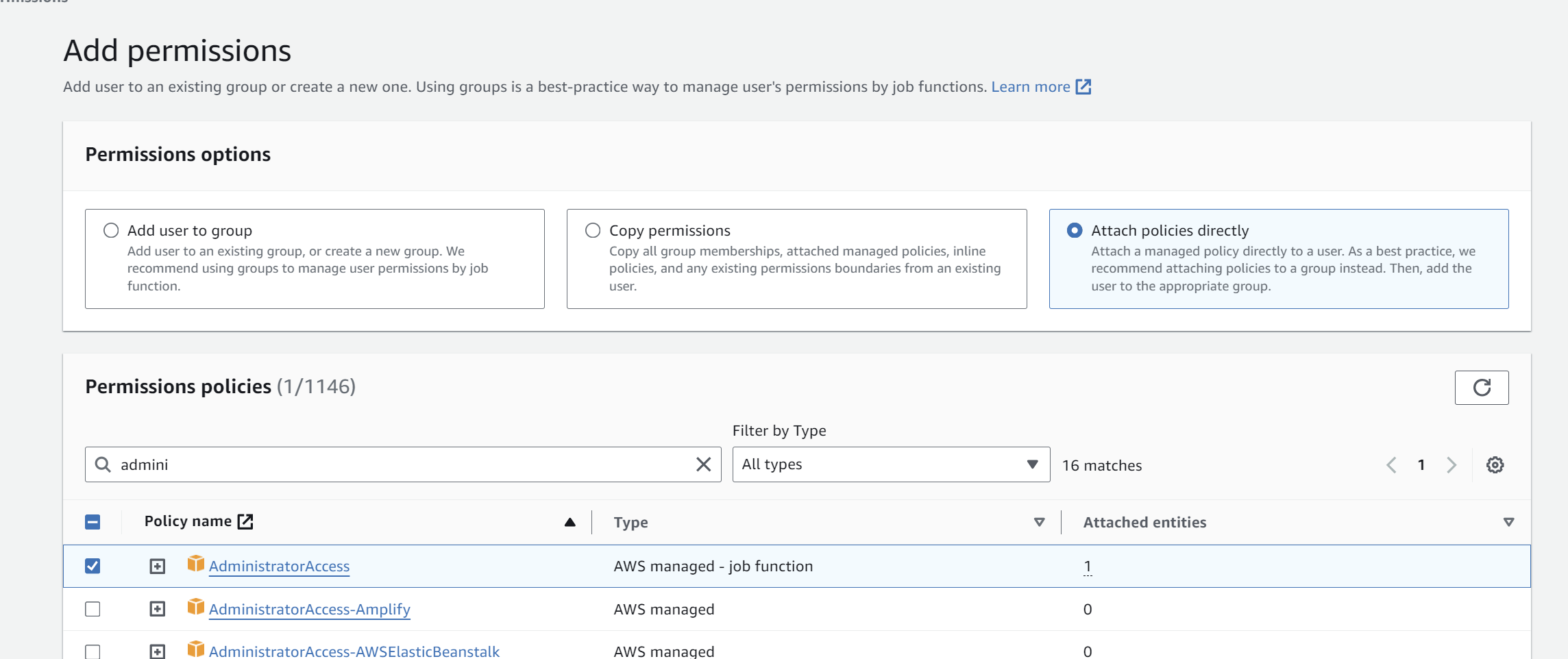


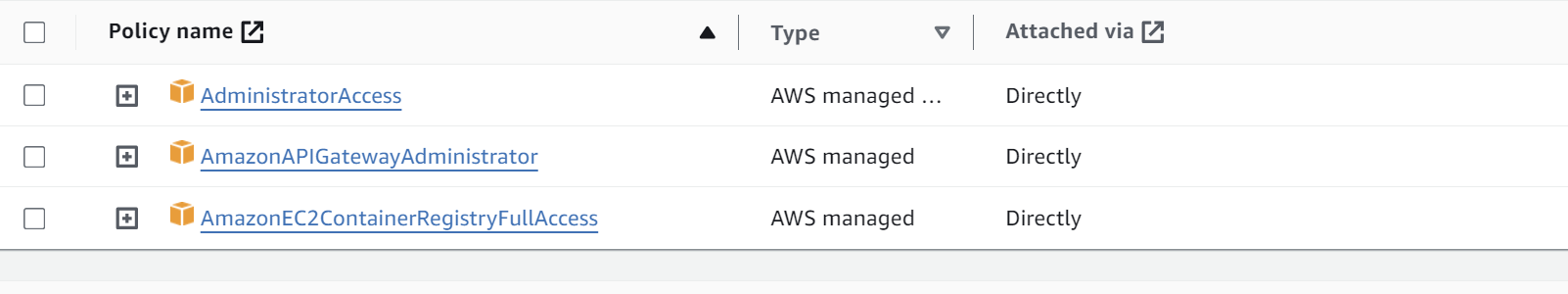
**Step-8:** Click on “Attach policies directly” and choose the following permissions and click on next and save them.

**AdministratorAccess**

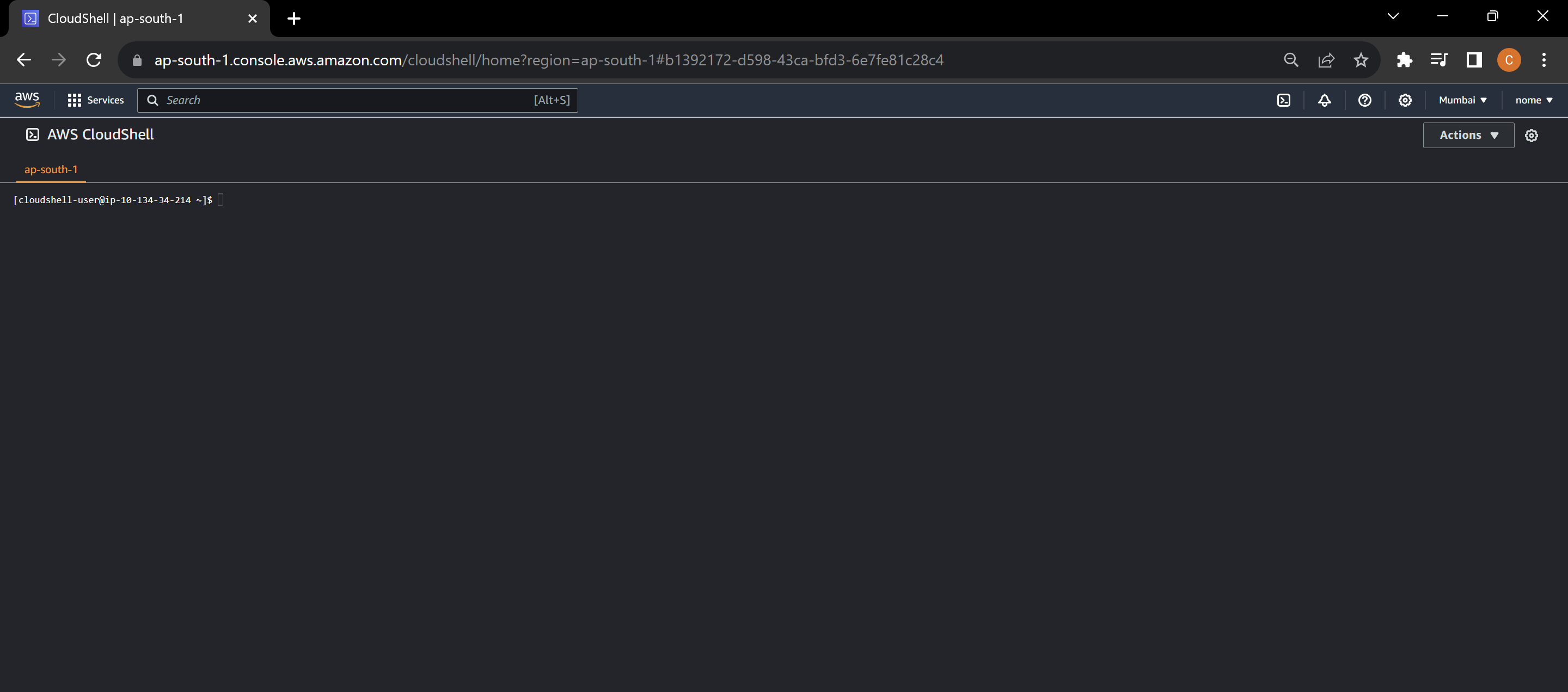
**AmazonAPIGatewayAdministrator**

**AmazonEC2ContainerRegistryFullAccess**

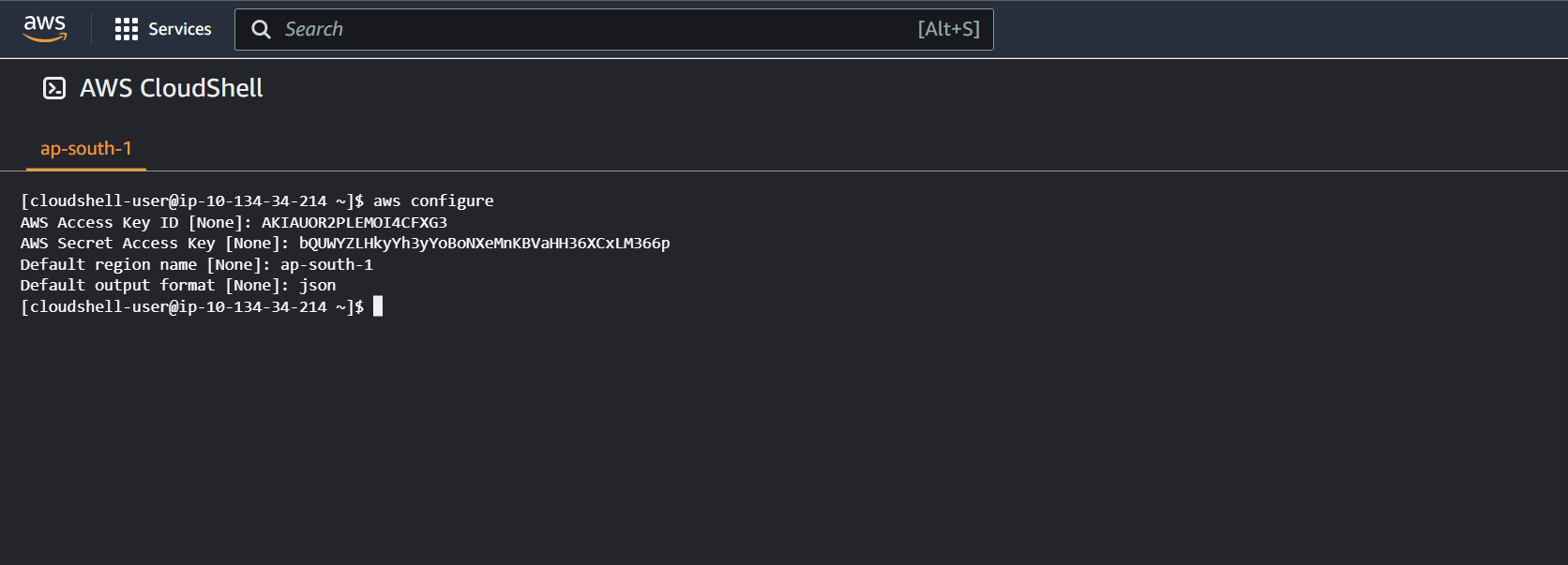




**Step-9:** Now go to AWS Console and search “**AWS Cloudshell**”.



**Step-10:** First step is to configure AWS CLI. Write the command “**aws configure**”. After that provide the access key and secret access key that you have copied earlier. Write your region and the format. Keep the format in “**json**” for example.

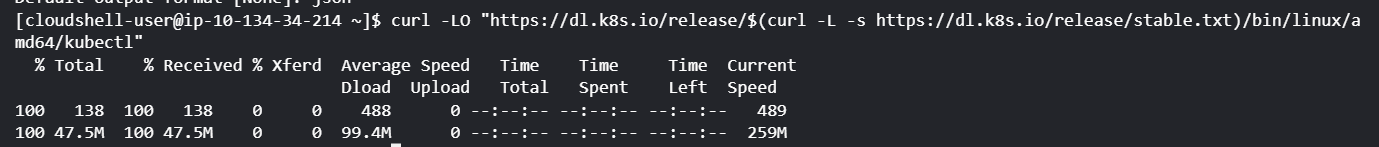


**Installing the required tools**

**Step-11:** Now install **kubectl** using the following command:

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)

* **kubectl is the command-line tool used to interact with Kubernetes clusters**.



**Step-12:** After the installation is complete, run the following command:

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

* **This command is installing the kubectl binary into the /usr/local/bin directory with root ownership and executable permissions, allowing users to run kubectl commands globally on the system.**



**Step-13:** In order to install the “**eksctl**” command line tool, we need to create a script. Create a file called “**eksctl.sh**” using vim or nano or any other test editor.

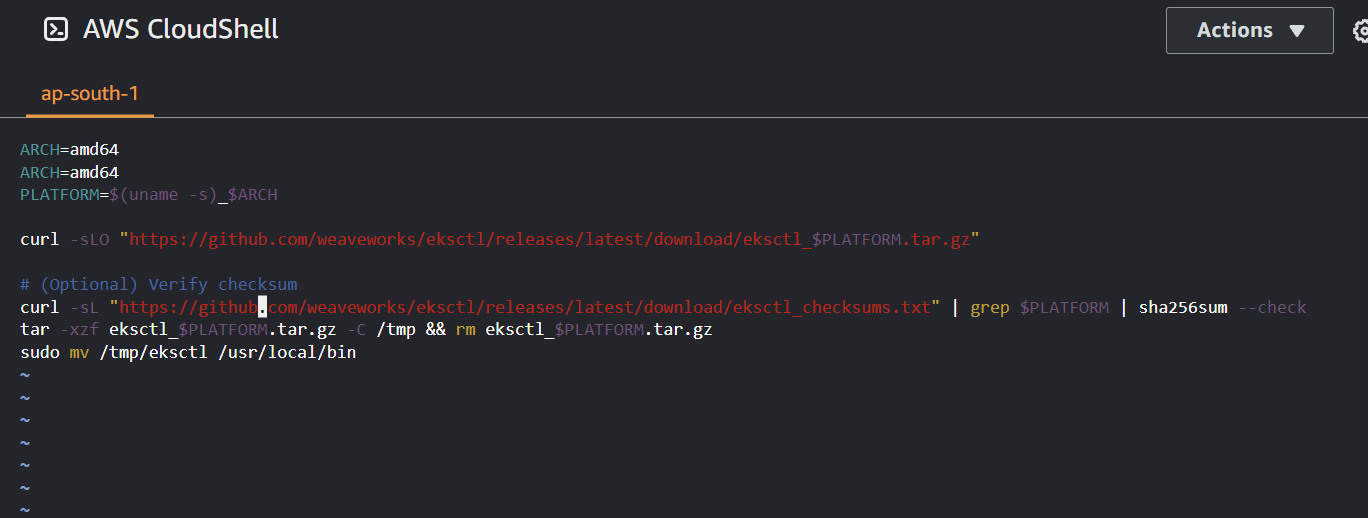
vim eksctl.sh

* eksctl is a command-line utility for creating, managing, and interacting with Amazon Elastic Kubernetes Service (EKS) clusters.



**Step-14:** Now in order to insert text, press **“I”** and then copy the below given commands inside it.

ARCH=amd64PLATFORM=$(uname -s)\_$ARCHcurl -sLO "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$PLATFORM.tar.gz"# (Optional) Verify checksumcurl -sL "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_checksums.txt" | grep $PLATFORM | sha256sum --checktar -xzf eksctl\_$PLATFORM.tar.gz -C /tmp && rm eksctl\_$PLATFORM.tar.gzsudo mv /tmp/eksctl /usr/local/bin

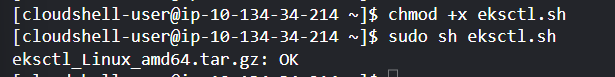


**Step-15:** Run the below commands:

chmod +x eksctl.sh

sudo sh eksctl.sh

* The command **chmod +x eksctl.sh** is used to grant execute permissions (+x) to the file named eksctl.sh.
* The command **sudo sh eksctl.sh** is attempting to run the script named **eksctl.sh** that we created.



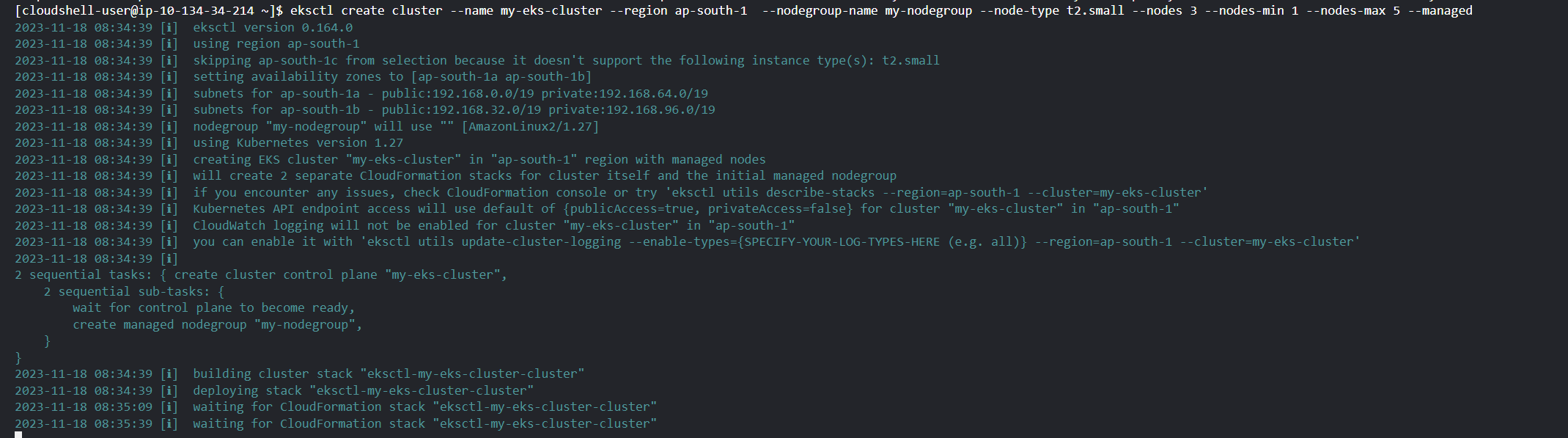
**Creating an EKS Cluster**

**Step-16:** Now create a new Amazon EKS cluster using the `**eksctl**` command:

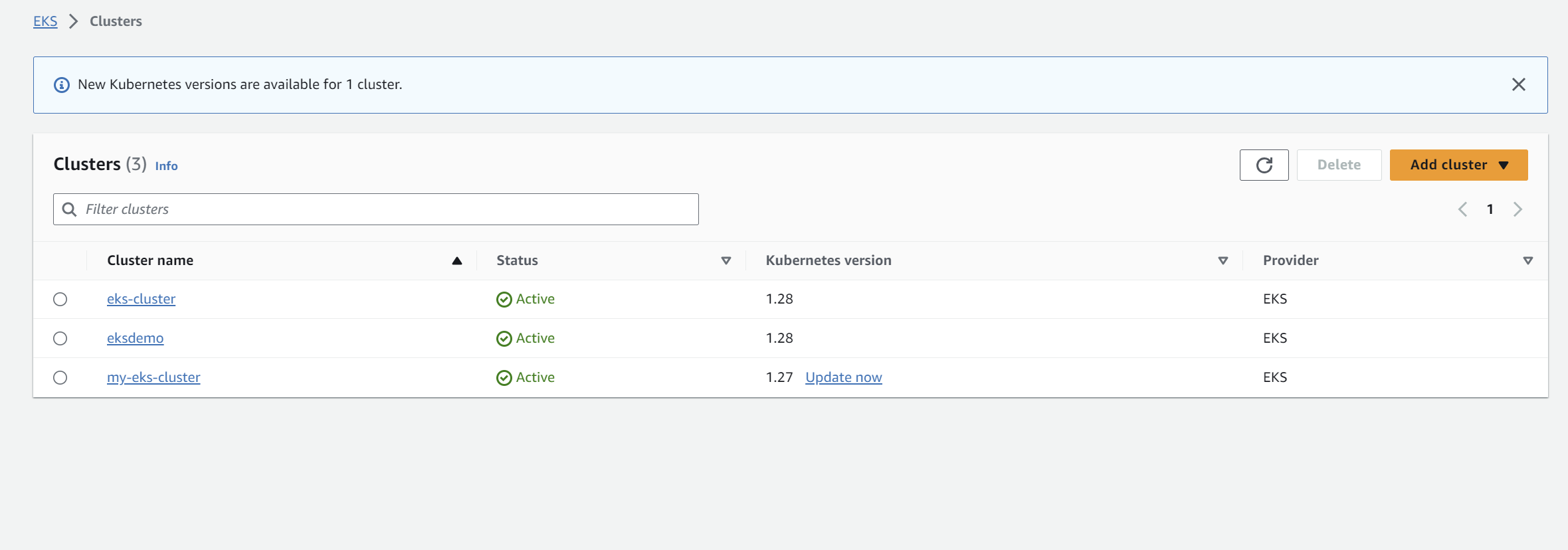
eksctl create cluster --name my-eks-cluster --region ap-south-1 --nodegroup-name my-nodegroup --node-type t2.small --nodes 3 --nodes-min 1 --nodes-max 5 –managed

* The **eksctl create cluster** command is creating an Amazon EKS cluster named "**my-eks-cluster**" in the **Asia Pacific (Mumbai) region** with a managed node group named "**my-nodegroup**," using **t2.small** instances with a **desired** count of **3 nodes**, a **minimum** of **1 node**, and a **maximum** of 5 nodes.

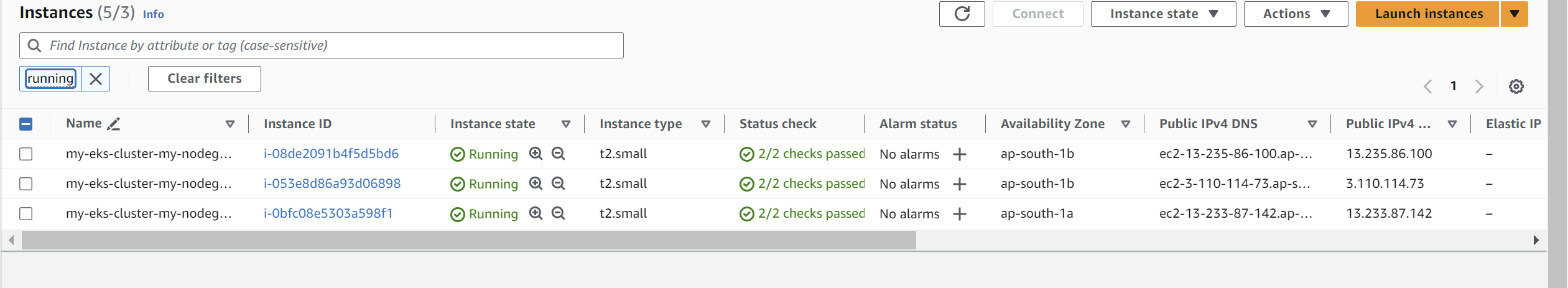
**Note:** Make sure to change the region name to your specific region in the command portion that is highlighted.



**Step-17:** Now after the creation is complete in the above step, go to **AWS console** and search **“EKS”.** You can see that the cluster is created.



**Step-18:** Also go to **EC2** Dashboard and click on instances. We can see that the specified number of node instances are created i.e. 3.



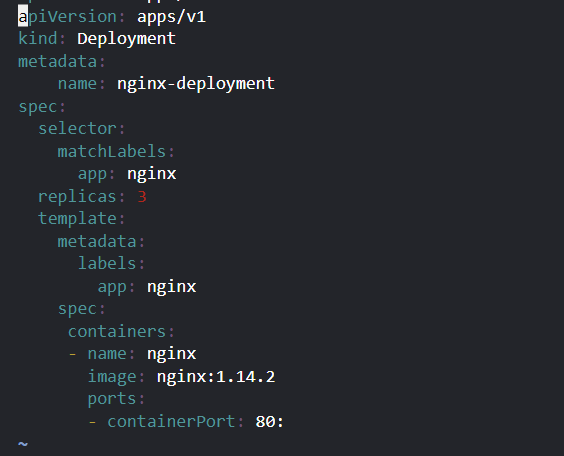
**Deploying a sample application**

**Step-19:** Now create a “**deployment.yaml”** file using any editor and copy the following yaml code.

NOTE: MAKE SURE THE INDENTATION OF THE YAML FILE IS SAME AS SHOWN OTHERWISE IT WILL THROW MAPPING ERROR.

apiVersion: apps/v1kind: Deploymentmetadata: name: nginx-deploymentspec: selector: matchLabels: app: nginx replicas: 3 template: metadata: labels: app: nginx spec: containers: - name: nginx image: nginx:1.14.2 ports: - containerPort: 80





**Step-20:** Deploy the sample application to your EKS cluster:

kubectl apply -f deployment.yaml



**Exposing the Application**

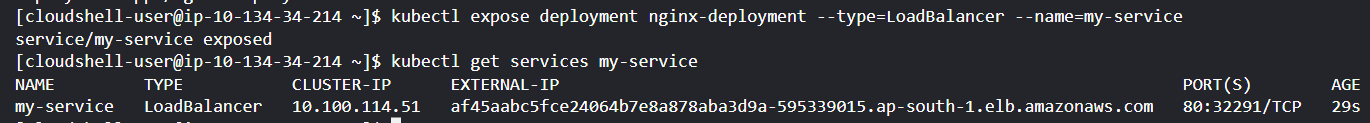
**Step-21:** Now expose the application using a Kubernetes service and get the external IP address of the LoadBalancer:

kubectl expose deployment nginx-deployment --type=LoadBalancer --name=my-service

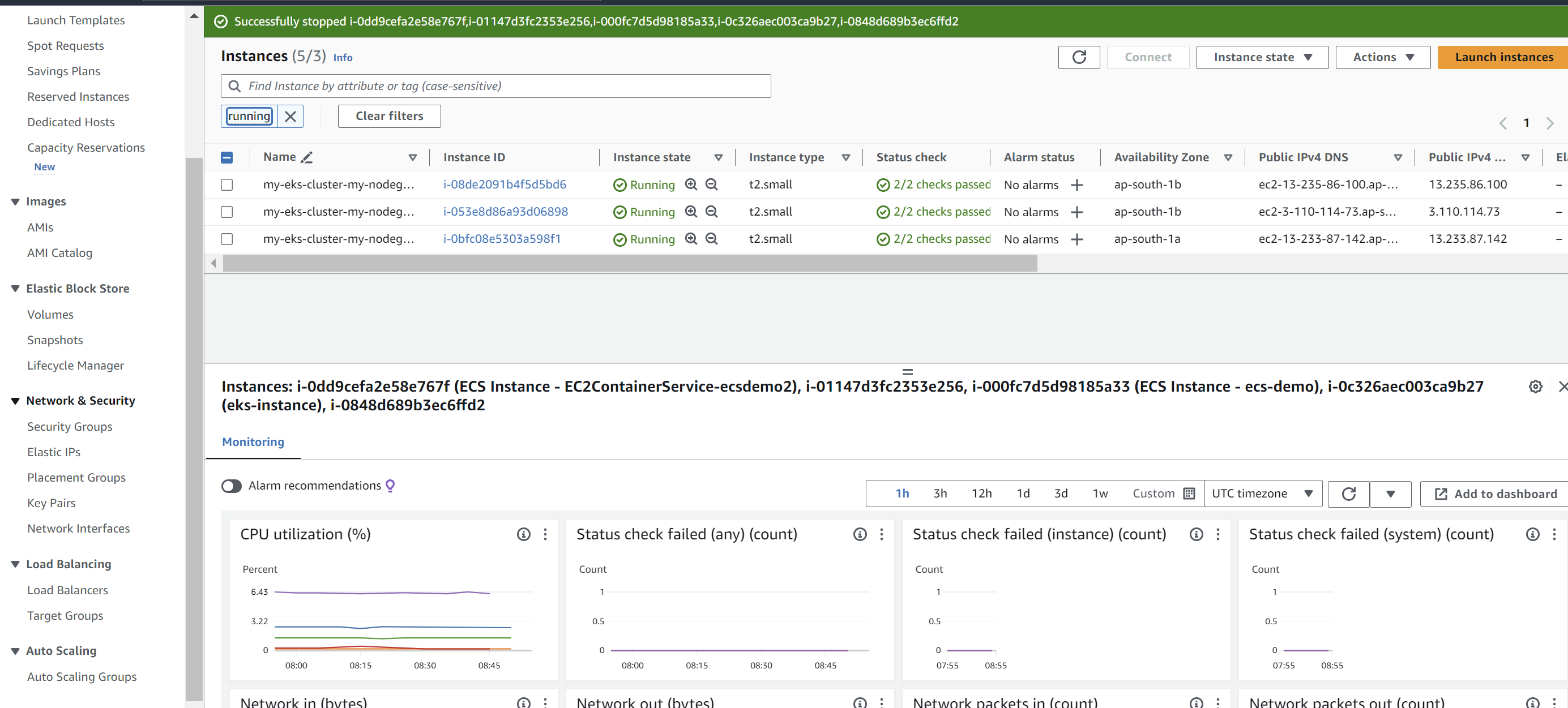
* The **kubectl expose command** is creating a new Kubernetes Service named "**my-service**" and exposing the "**nginx-deployment**" Deployment to the external network using a **LoadBalancer** type service.

kubectl get services my-service

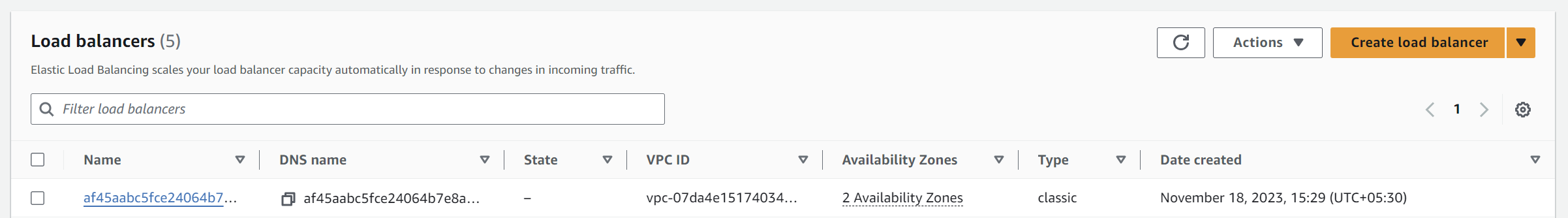
* The **kubectl get services my-service** command is used to retrieve information about the Kubernetes Service named "**my-service**," displaying details such as the **service's IP address, ports**, and other relevant information.



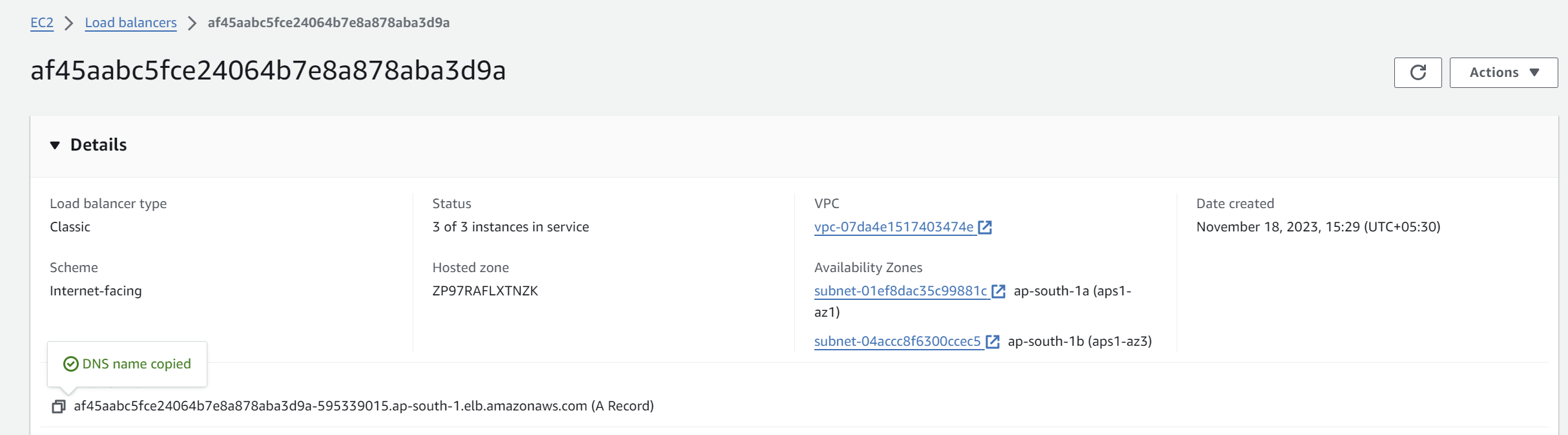
**Step-22:** Now go to EC2 dashboards and click on load balancers.



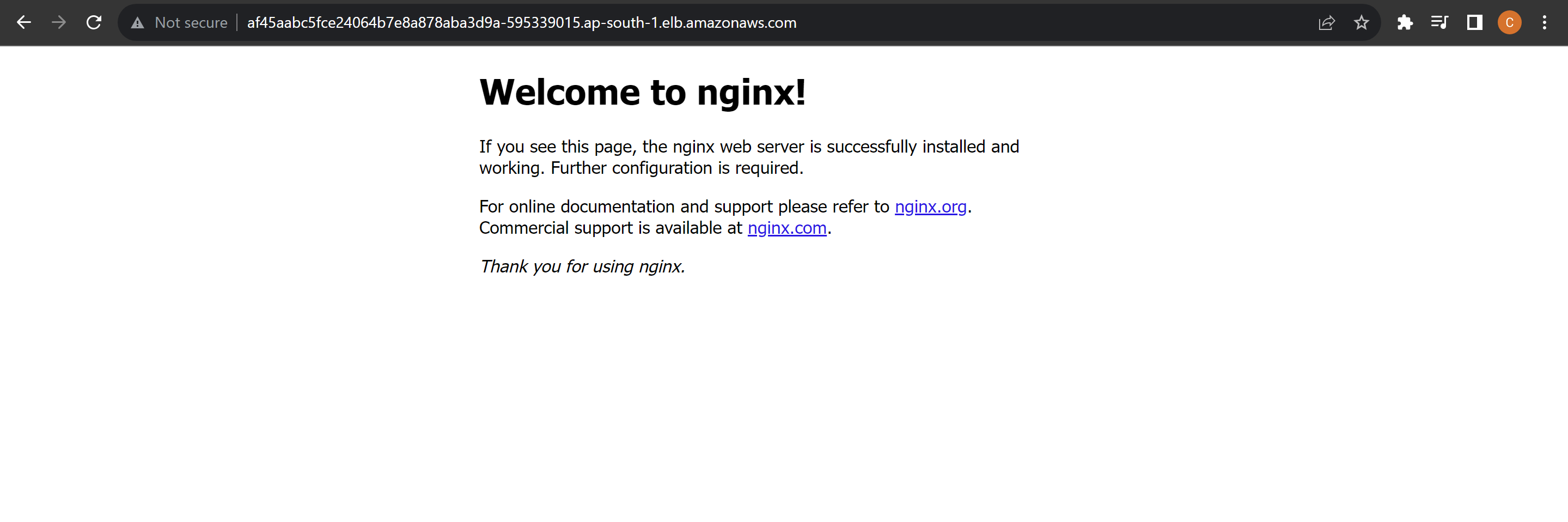
**Step-23:** We can see on the dashboard a “**classic**” load balancer that is made by default after running the command.



**Step-24:** Copy the **DNS name** and paste it in browser to check whether the service is accessible or not.



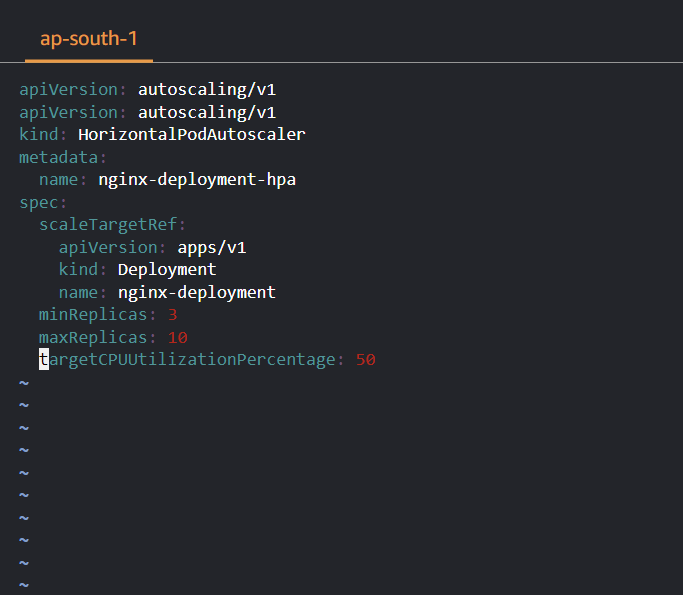
We can see that it is accessible on browser.



**Setting up Horizontal Pod Autoscaler**

Step-25: Now create a new file named `**hpa.yaml**`(just like we made the deployment.yaml) with the following content:

apiVersion: autoscaling/v1kind: HorizontalPodAutoscalermetadata: name: nginx-deployment-hpaspec: scaleTargetRef: apiVersion: apps/v1 kind: Deployment name: nginx-deployment minReplicas: 3 maxReplicas: 10 targetCPUUtilizationPercentage: 50



**Step-26:**

kubectl apply -f hpa.yaml



Step-27: After performing the above commands, you can delete the service and cluster using the following commands:

kubectl delete svc my-serviceeksctl delete cluster --name my-eks-cluster