**Java App Deployment on Kubernetes Cluster Runbook**

**Project Overview**

This runbook provides step-by-step instructions for deploying a Java application on a Kubernetes cluster. The project source is hosted on a GitHub repository.

GitHub Repository: Project Source

**Prerequisites:**

Before starting the deployment, ensure you have the following:

AWS Account: An active AWS account to create and manage resources.

Registered DNS Name: A domain name registered and configured to point to your Kubernetes cluster.

**Deployment Steps**

**Step 1: create an EC2 instance**

Log in to the AWS Management Console.

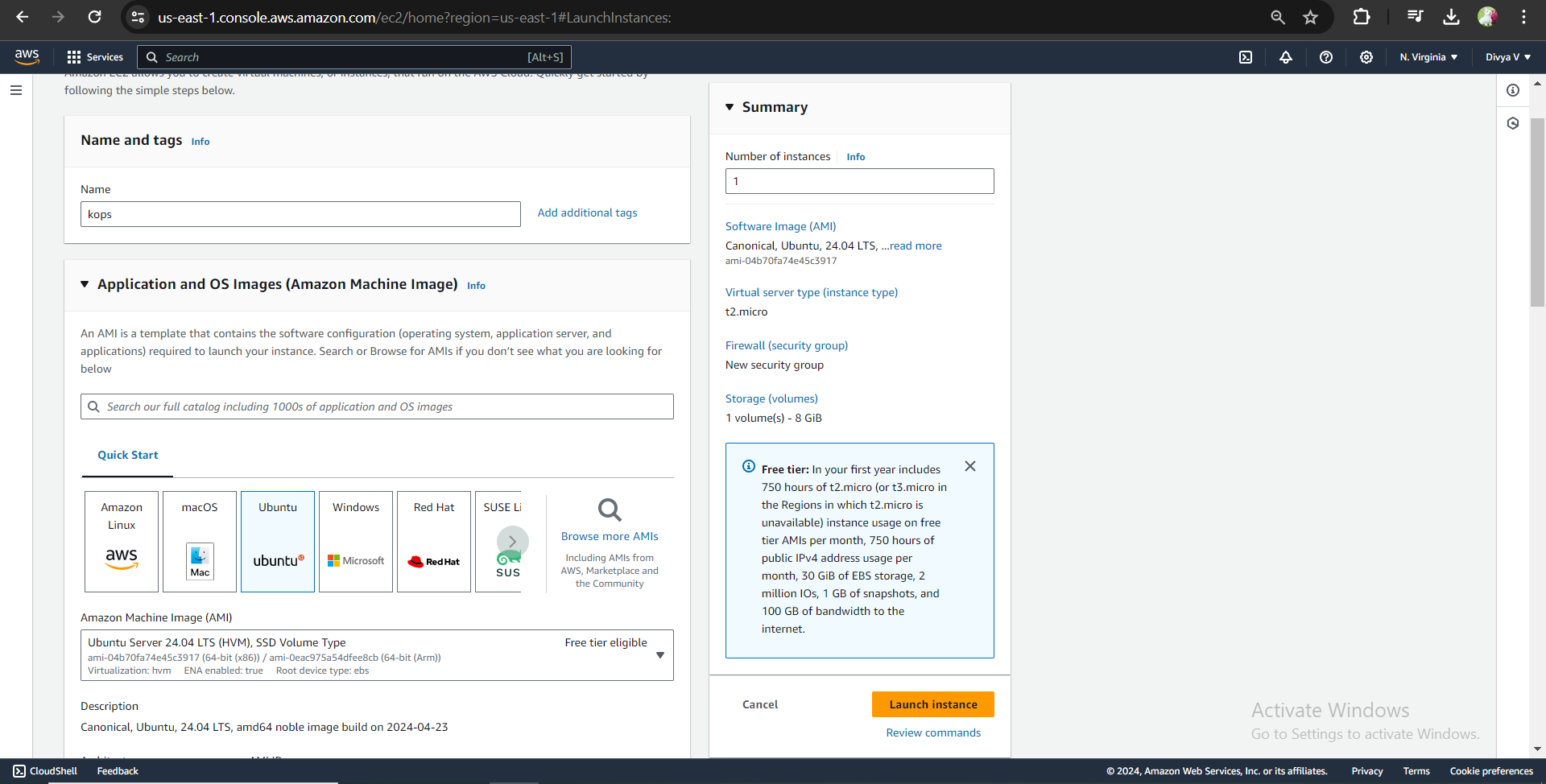
Navigate to EC2 Dashboard: Select "Launch Instance".

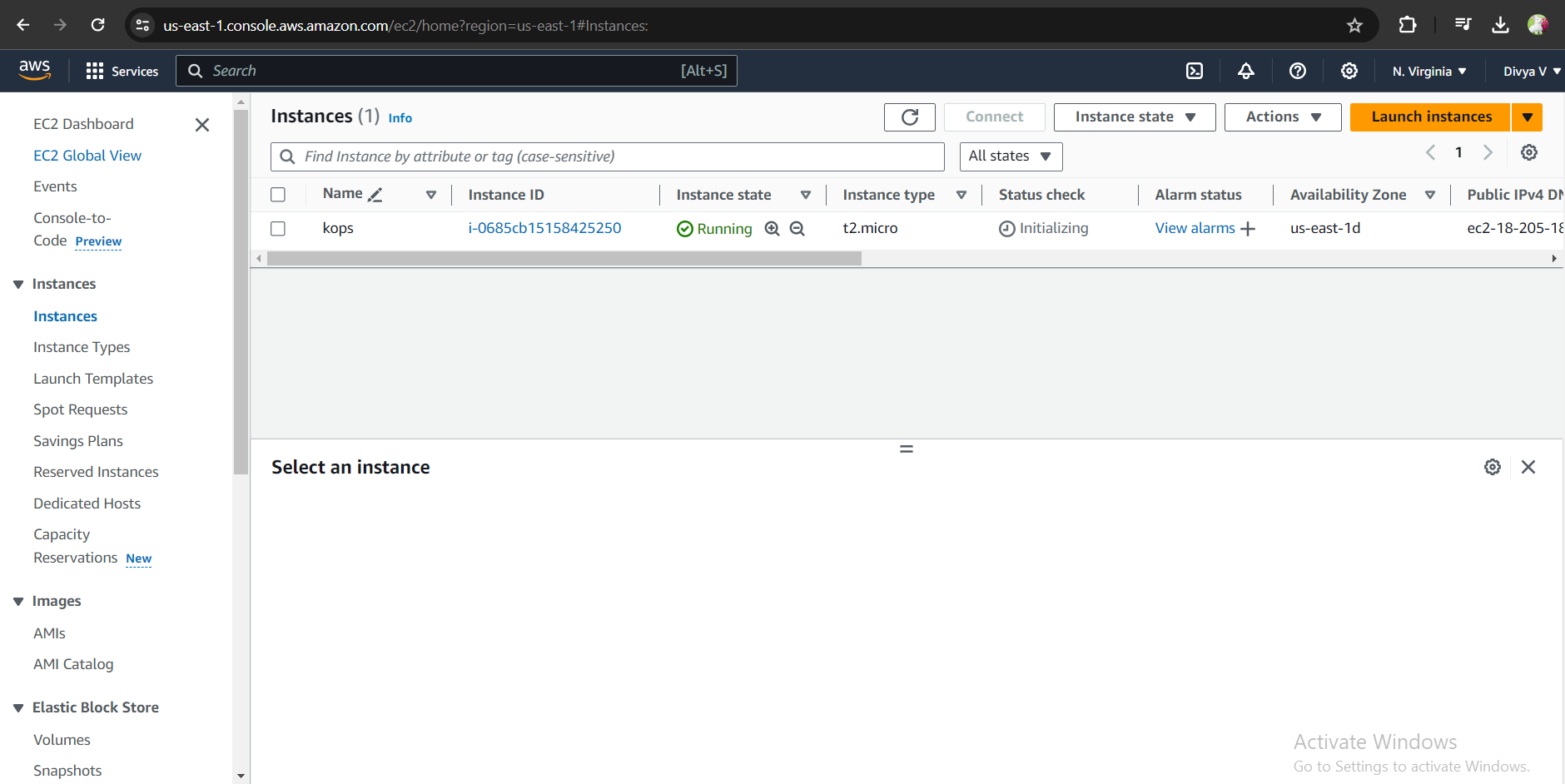
Configure Instance:

* Name: Name your instance "kops".
* AMI: Choose "Ubuntu 20.04 LTS (HVM), SSD Volume Type".
* Instance Type: Select "t2.micro".
* Key Pair: Create a new key pair, download it, and store it securely.
* Create a new security group that allows SSH access on port 22 from your IP address.

Launch the Instance: Review and launch the instance.

Snap:





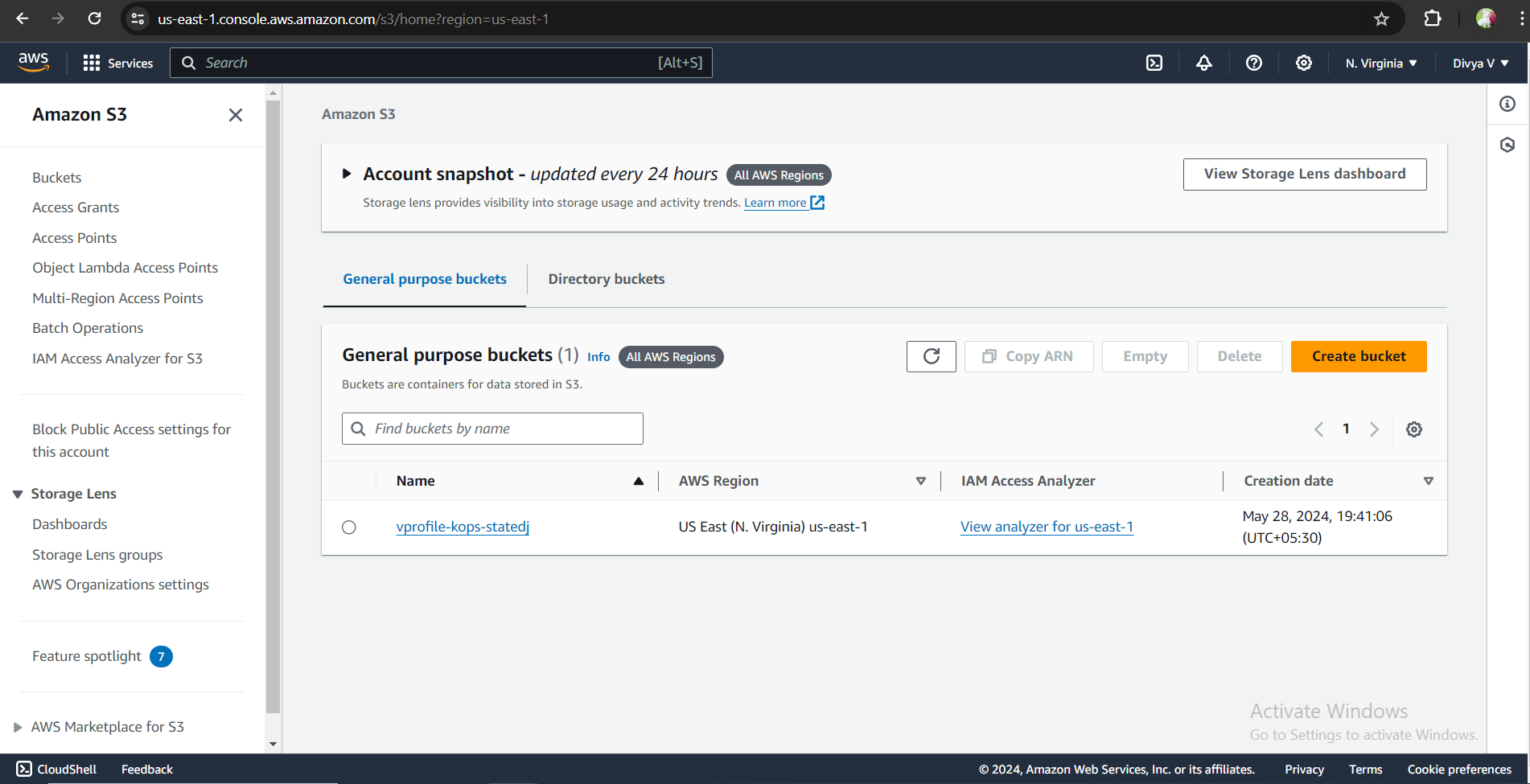
**Step 2: Create S3 Bucket**

Create an S3 bucket to store the state of kOps so we can run our kOps command from anywhere as long as it is pointed to our S3 bucket.

**Steps to create an S3 bucket:**

* Log in to the AWS Management Console.
* Navigate to the S3 Service: Select "Create Bucket".
* Configure Bucket:
* Bucket Name: Enter "vprofile-kops-statedj".
* Region: Select your preferred region.
* Create the Bucket: Review and create the bucket**.**

**Snap:**



**Step 3: Create IAM User**

Create an IAM user for our AWS CLI with the following details:

**Username: Kops-admin**

**Attach Policies: Administrator Access**

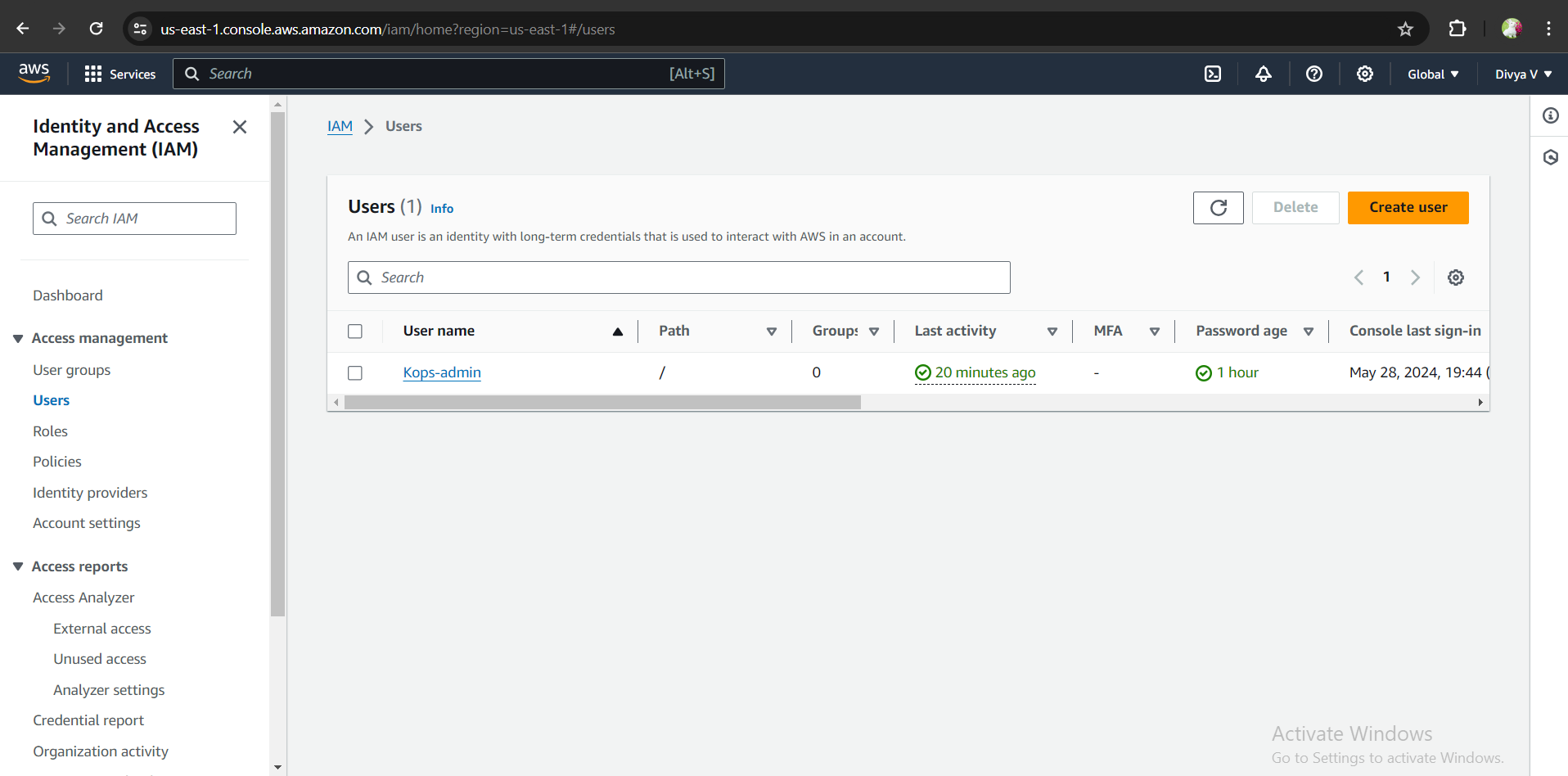
**Steps:**

* Log in to AWS Management Console.
* Navigate to IAM Service.
* **Add User:** User name: Kops-admin

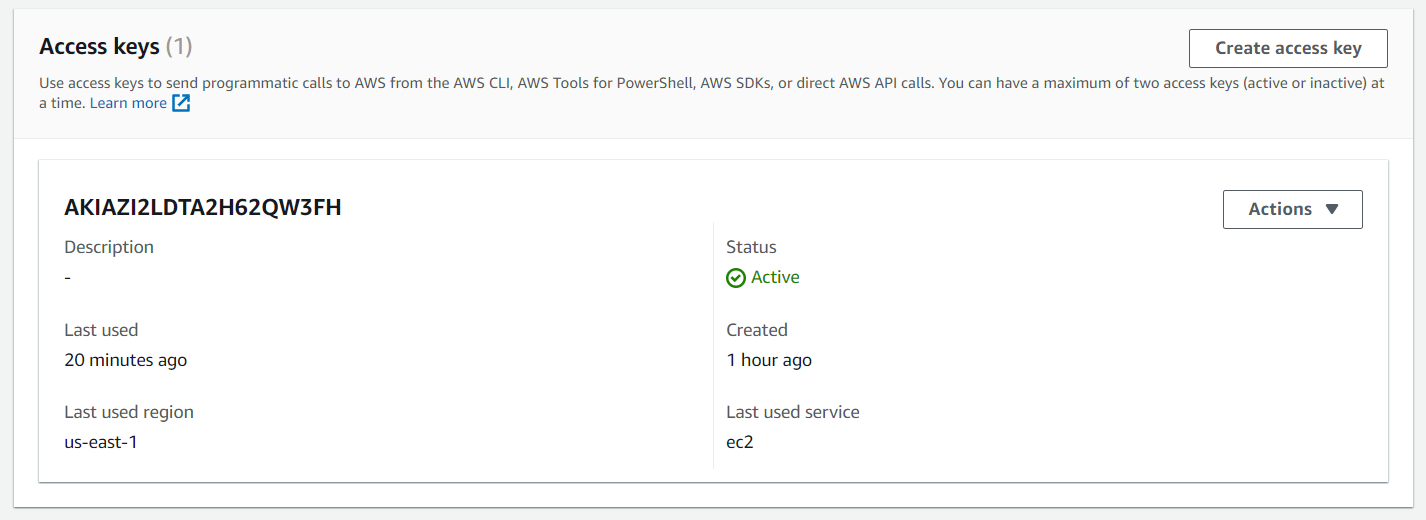
Attach Policies:

* Select "Attach policies directly".
* Attach "Administrator Access".
* Review and Create User.
* Download Credentials: Download the .csv file with access key ID and secret access key.
* And create access keys

**Snap:**



**Snap: Access keys Creation**



**Step 4 : Configure DNS in Route53:**

* Log in to the AWS Management Console.
* Navigate to Route 53 and create a hosted zone for your domain.
* Note the name servers (NS) provided by Route 53.
* Update Name Servers at Domain Provider website. (godaddy, hostinger..,)
* Update the name servers with those provided by Route 53.
* verify the setup using <https://toolbox.googleapps.com/apps/dig/>

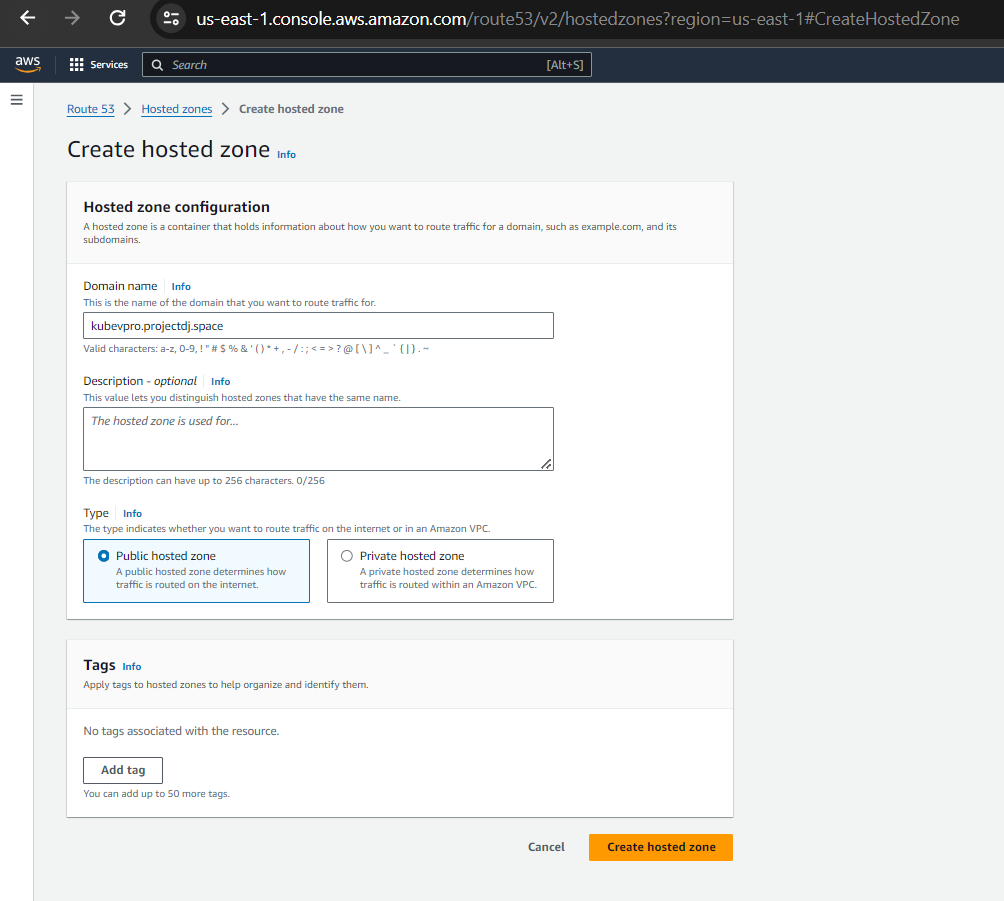
**Note: DNS changes can take up to 24 hours to propagate. It is recommended to complete this step in advance.**

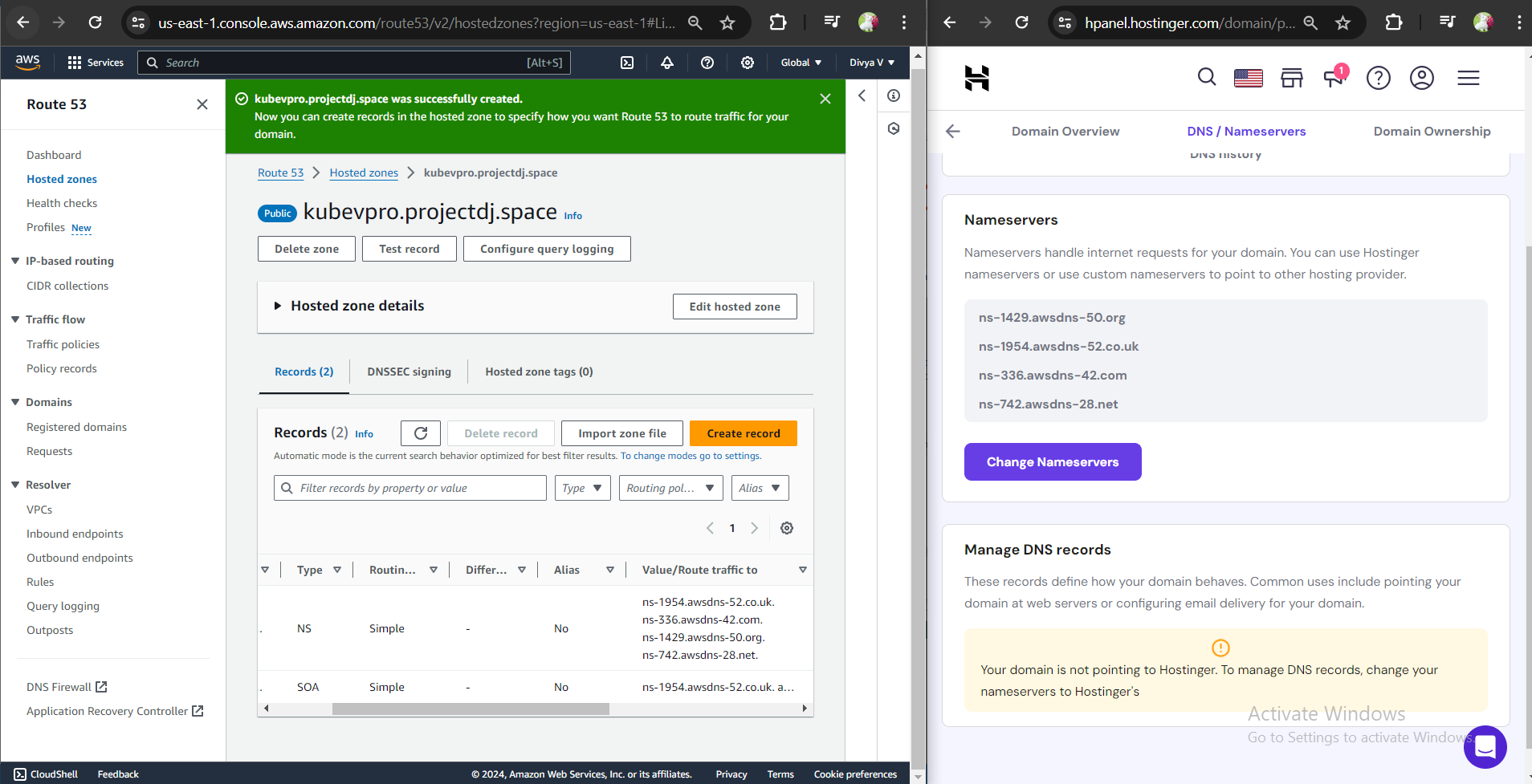
[After kops cluster creation

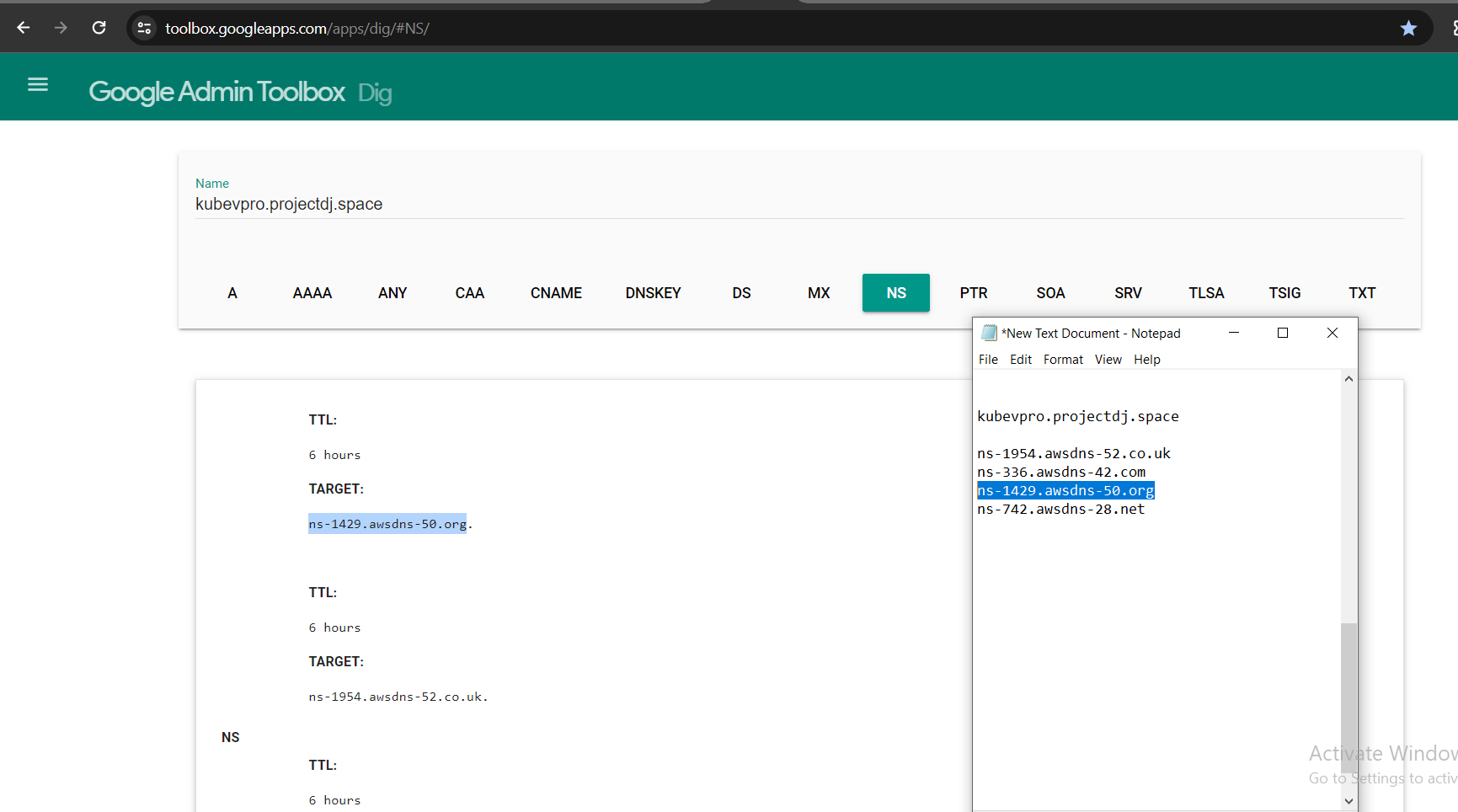
Retrieve Load Balancer URL.

Update DNS Records]

**Snap:**







**Step-5: Login Ec2 and configure awscli**

Now that we have our IAM user, S3 bucket and Domain setup, we can login and configure our ec2 instance.

Pem file permissions:

set appropriate permissions for your .pem files to ensure the security of your SSH keys.

In windows:

* Locate Your Private Key File
* Right-click on the File
* Go to the Security Tab
* Edit Permissions
* Adjust Permissions

In Linux:

**chmod 400 file\_name.pem**

**Steps to Configure EC2 Instance:**

Log in to Your EC2 Instance:

Use the private key file downloaded when you created the EC2 instance.

**ssh -i your-key-file.pem ec2-user@your-ec2-instance**

**Generate the SSH key pair:**

* Once logged in, run the following command to generate a new SSH key pair
* Navigate to the SSH Directory
* List the SSH Directory Contents

**ssh-keygen -N "" -f $HOME/.ssh/id\_rsa**

**cd ~/.ssh**

**ls**

This will generate a new SSH key pair and list the contents of the .ssh directory, ensuring that the keys are created and stored correctly. The generated SSH key will be used by kOps for cluster management.

**Update Packages and Install AWS CLI:** After logging in to your EC2 instance and generating the SSH key, update the packages and install the AWS CLI.

**Update Packages:**

sudo apt update

**Check System Architecture:**

uname –m

**Download AWS CLI Installer:**

Follow the instructions based on the system architecture from the AWS CLI installation guide:

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

**Download AWS CLI Installer:**

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

**Install Unzip Utility:**

sudo apt install unzip -y

**Unzip AWS CLI Installer:**

unzip awscliv2.zip

**Install AWS CLI:**

sudo ./aws/install

**Verify AWS CLI Installation:**

aws --version

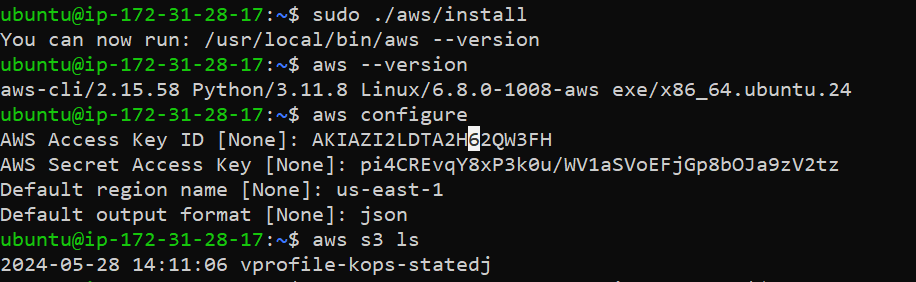
**Configure AWS CLI:**

aws configure

Enter the AWS access key ID and secret access key obtained when you created the IAM user.

Specify the default region name and json as output format when prompted.

**Snap:**



**Step 6: Setup kOps Cluster**

**Install and Setup kubectl:**

Follow the commands from the official documentation:

<https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>

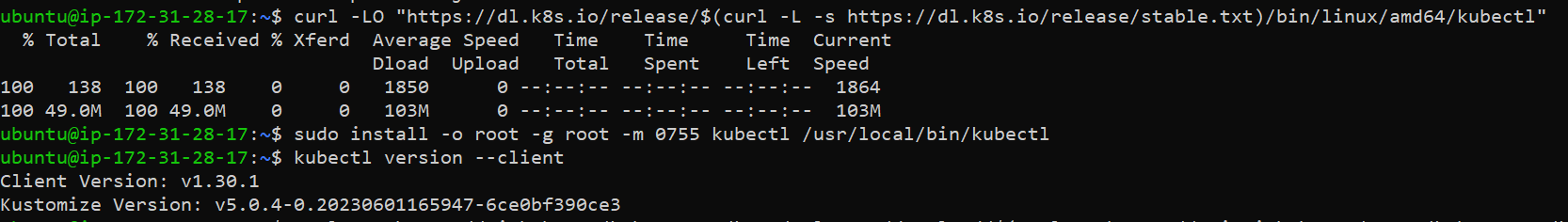
Download kubectl Installer:

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

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

this command installs the kubectl binary to /usr/local/bin/, sets its owner and group to root, and assigns it read, write, and execute permissions for the owner and read and execute permissions for the group and others. This ensures that kubectl can be executed by any user on the system but only modified by the root user.

**Snap:**



**Verify the Kubectl installation:**

kubectl version --client

**Install kOps:**

Follow the commands from the official documentation.

curl -LO <https://github.com/kubernetes/kops/releases/download/v1.23.0/kops-linux-amd64>

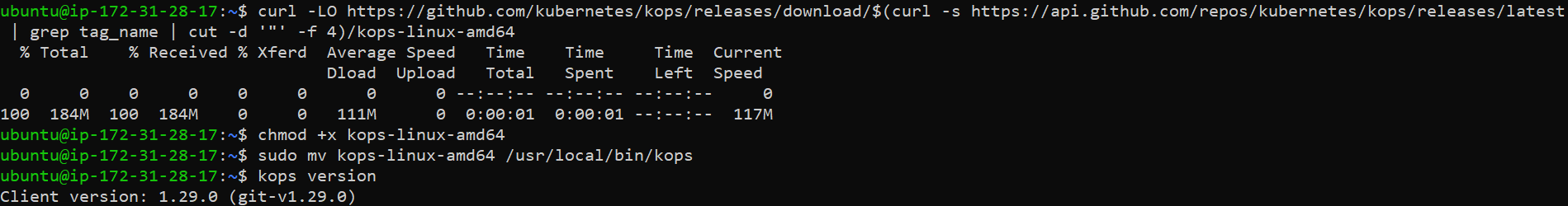
chmod +x kops-linux-amd64

sudo mv kops-linux-amd64 /usr/local/bin/kops

**Verify the Kops installation:**

kops version

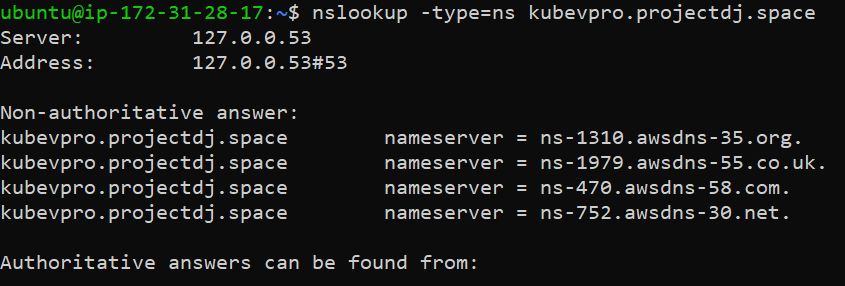
Snap:



**Verify Domain:**

nslookup --type=ns kubevpro.<yourdomain-name.com>

You should get an output similar to the following if your installations are correct:



**Set Environment Variable:**

Setting the KOPS\_STATE\_STORE environment variable is necessary for kOps to know where to store and retrieve the cluster configuration in Amazon S3. You can export this variable in your terminal session by running the following command:

**export KOPS\_STATE\_STORE=s3://vprofile-kops-statedj**

This command tells kOps to use s3://vprofile-kops-statedj as the state store for managing your Kubernetes clusters. Make sure to replace s3://vprofile-kops-statedj with the correct S3 bucket URL for your kOps state store.

**Create Kubernetes Cluster:**

Replace <yourdomain-name.com> and <s3-bucket-name> with your actual domain name and S3 bucket name:

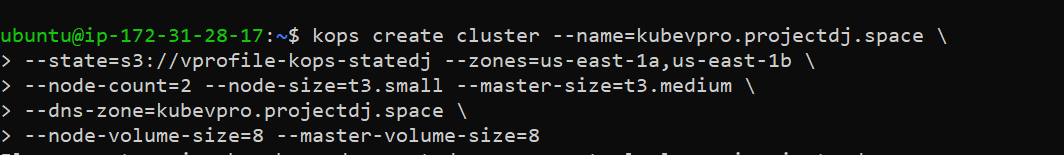
**kops create cluster --name=kubevpro.<yourdomain-name.com> \**

**--state=s3://<s3-bucket-name> --zones=us-west-1b,us-west-1c \**

**--node-count=2 --node-size=t3.small --master-size=t3.medium \**

**--dns-zone=kubevpro.<yourdomain-name.com> --node-volume-size=8 --master-volume-size=8**

Snap:

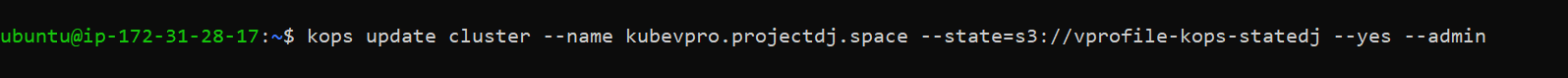


**Update Cluster:**

Run the following command to apply the cluster configuration:

**kops update cluster --name=kubevpro.<yourdomain-name.com> --state=s3://<s3-bucket-name> --yes --admin**

Snap:

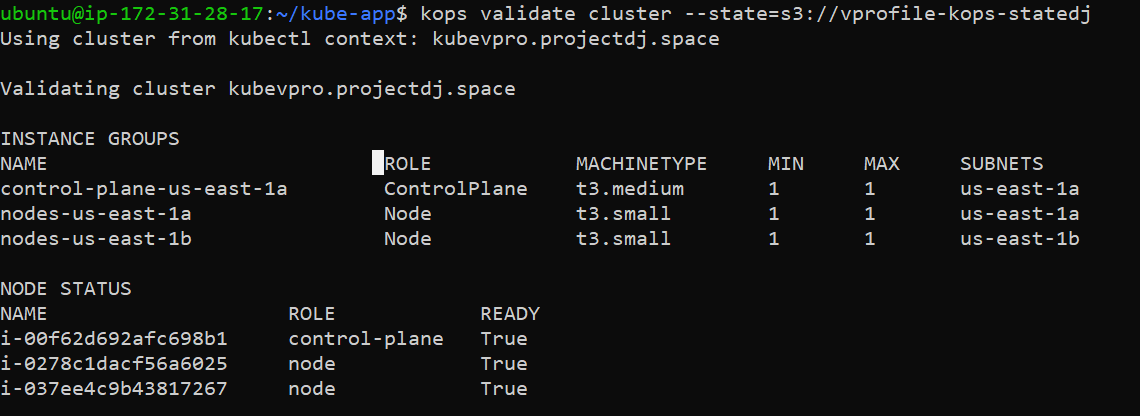


**Validate Cluster:**

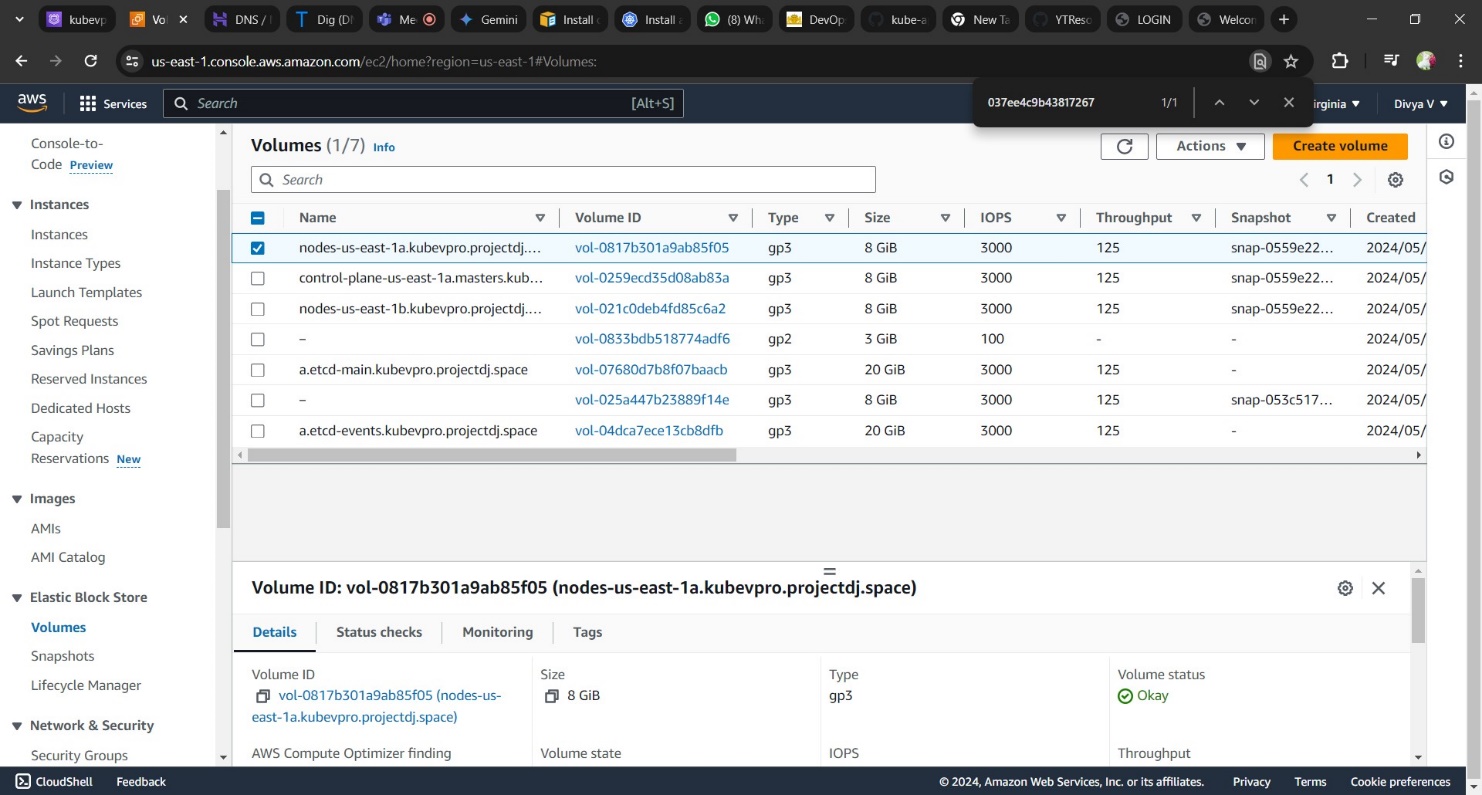
Wait for 10–15 minutes for the cluster to be fully operational, then validate:

kops validate cluster --state=s3://<s3-bucket-name>

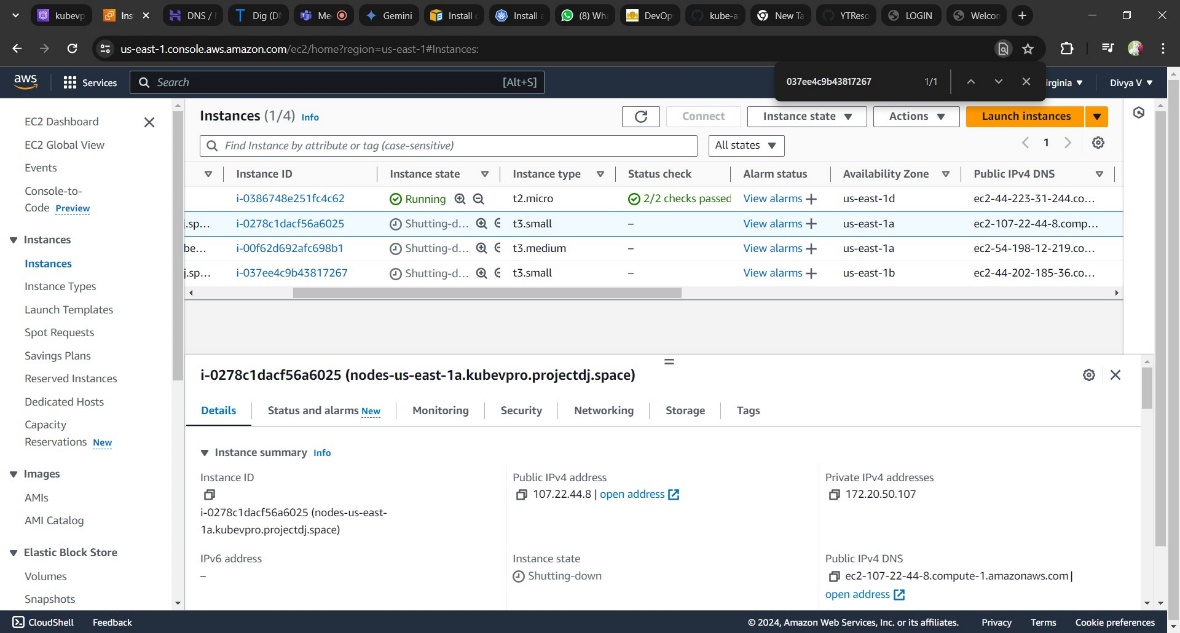
If all is well, you should see output similar to the following:



In AWS Console:



Below EC2 instances created by cluster (Snap took after cluster deletion)



**Step 7: Create Volume for DB Pod**

To store MySQL data, create an EBS volume using the command below:

**Create an EBS Volume:**

Use the following command to create an EBS volume

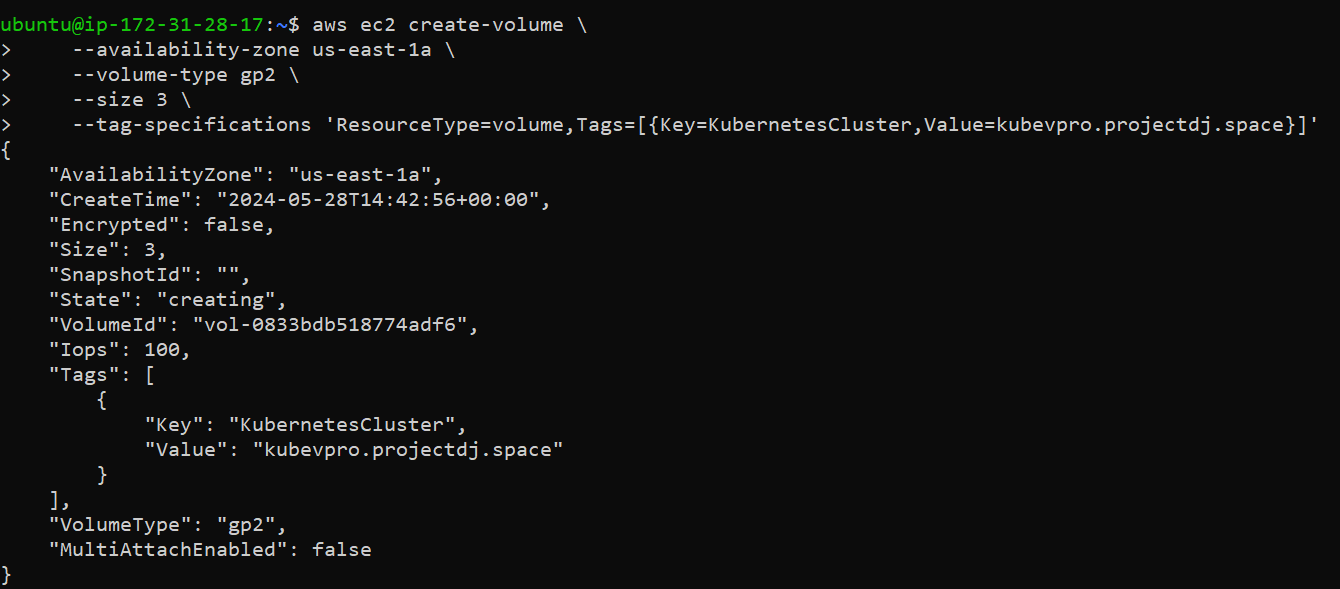
aws ec2 create-volume \

--availability-zone us-east-1a \

--volume-type gp2 \

--size 3 \

--tag-specifications 'ResourceType=volume,Tags=[{Key=KubernetesCluster,Value=kubevpro.irvingpictures.co}]'

Snap:

**Label the Nodes:**

Ensure that your DB pod runs in the same zone as your EBS volume by labeling your nodes accordingly.

Steps:

* Describe Node
* Identify the node in the specific zone

**kubectl describe node i-0b181a16c01f0a1d4 | grep us-west-1**

Label Node:

Label the node in the us-west-1c zone:

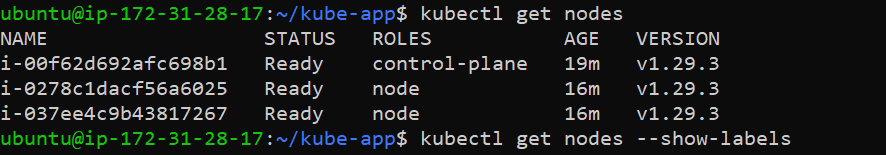
**kubectl label nodes i-09a3d0bfbab45b066 zone=us-west-1c**

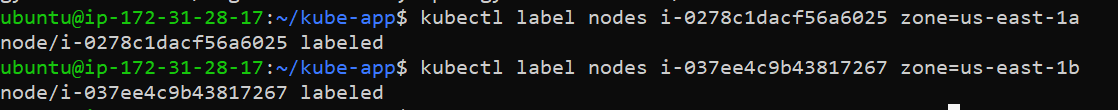
**Verify Labels:**

Verify that the nodes have been labeled correctly:

**kubectl get nodes --show-labels**

Snap:





* Create an EBS volume in the required availability zone to store MySQL data.
* Label the nodes to ensure that your DB pod runs in the same zone as the EBS volume, ensuring proper data locality and performance

**Step 8: Source Code Review**

You can find all Kubernetes manifest files by cloning the following repository. Follow these steps to review the source code and make necessary changes.

Steps to Review Source Code:

Clone the Repository:

git clone https://github.com/rumeysakdogan/kube-app.git

Navigate to the Cloned Repository:

cd kube-app



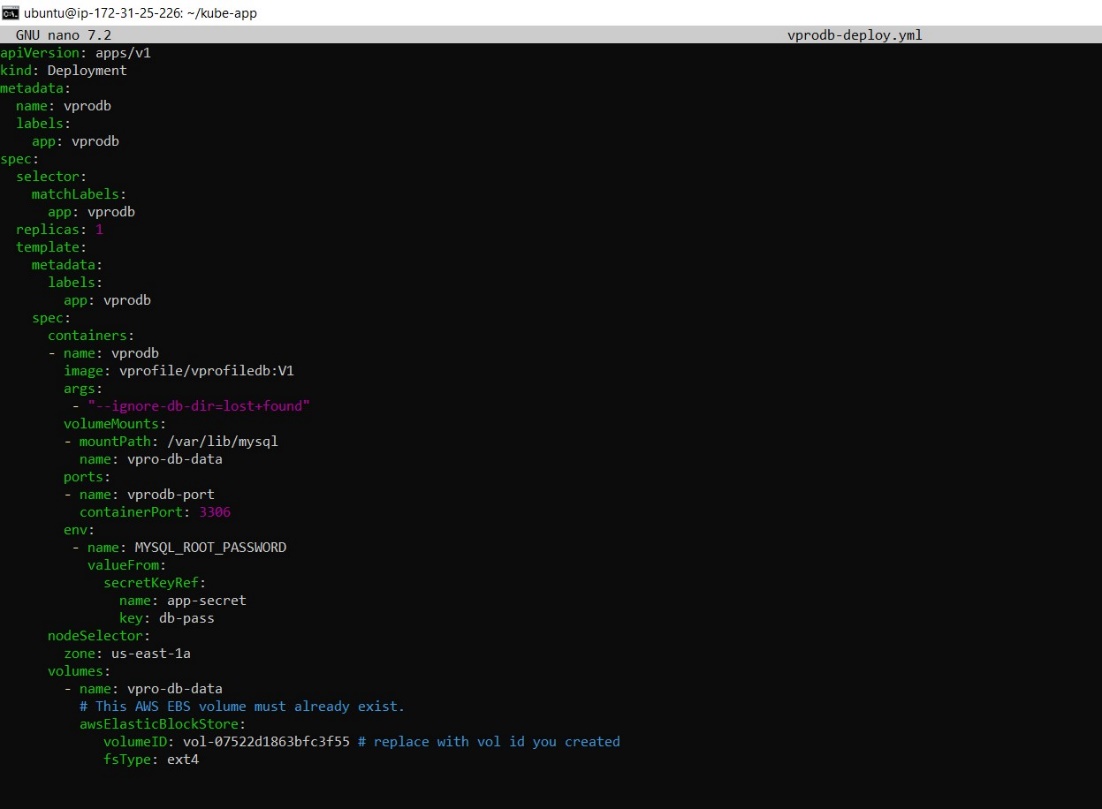
**Review and Edit Manifest Files:**

Use your preferred text editor to open and edit the Kubernetes manifest files as needed.

Here we need to update our EBS voulume ID in the DB deployment Yaml file.

nano vprodb-deploy.yml





save CTRL+o

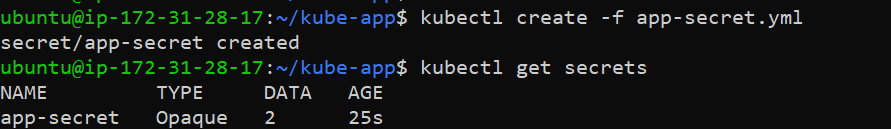
exit CTRL+x

**Step 9: Create Secret**

To create a secret in Kubernetes, you can use the kubectl create secret command. Secrets are used to store sensitive information such as passwords, API keys, and certificates.

Here we stored DB password and rabbitMQ password

**kubectl create -f app-secret.yml**

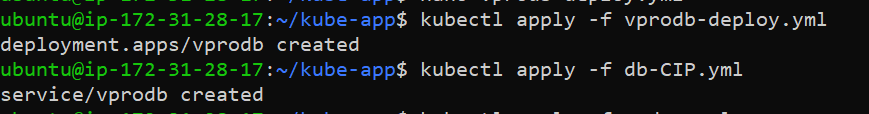


**Step 10: DB Deployment & Service Definition**

Create a Kubernetes deployment and service definition for your database.

kubectl apply -f vprodb-deploy.yml

kubectl apply -f db-CIP.yml



**Step 11: Memcached Deployment & Service Definition**

Create a Kubernetes deployment and service definition for Memcached.

**kubectl apply -f mcdep.yml**

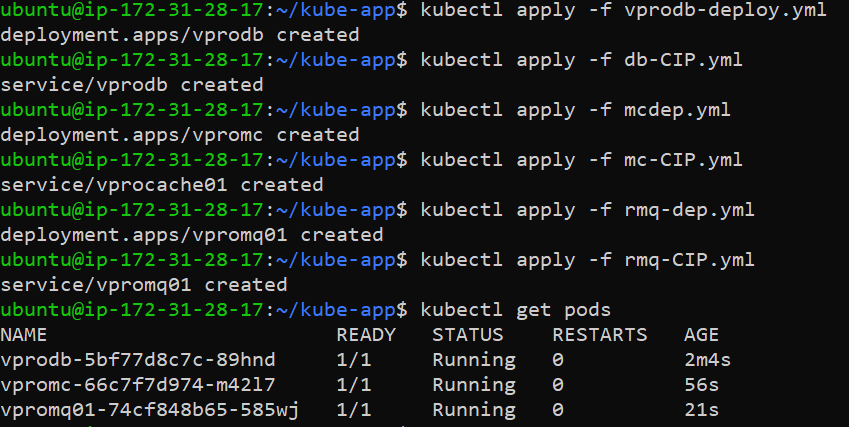
**kubectl apply -f mc-CIP.yml**

**Step 12: RabbitMQ Deployment & Service Definition**

Create a Kubernetes deployment and service definition for RabbitMQ.

**kubectl apply -f rmq-dep.yml**

**kubectl apply -f rmq-CIP.yml**

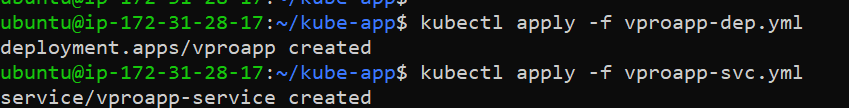


**Step 13: Application Deployment & Service Definition**

Create a Kubernetes deployment and service definition for your application.

kubectl apply -f vproapp-dep.yml

kubectl apply -f vproapp-svc.yml

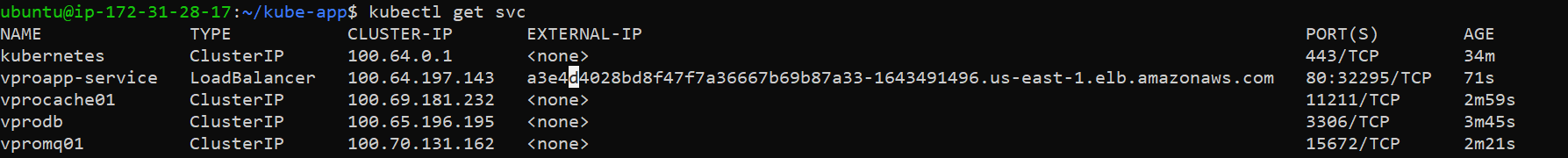


After running these commands, you can check the resources created with the following command:

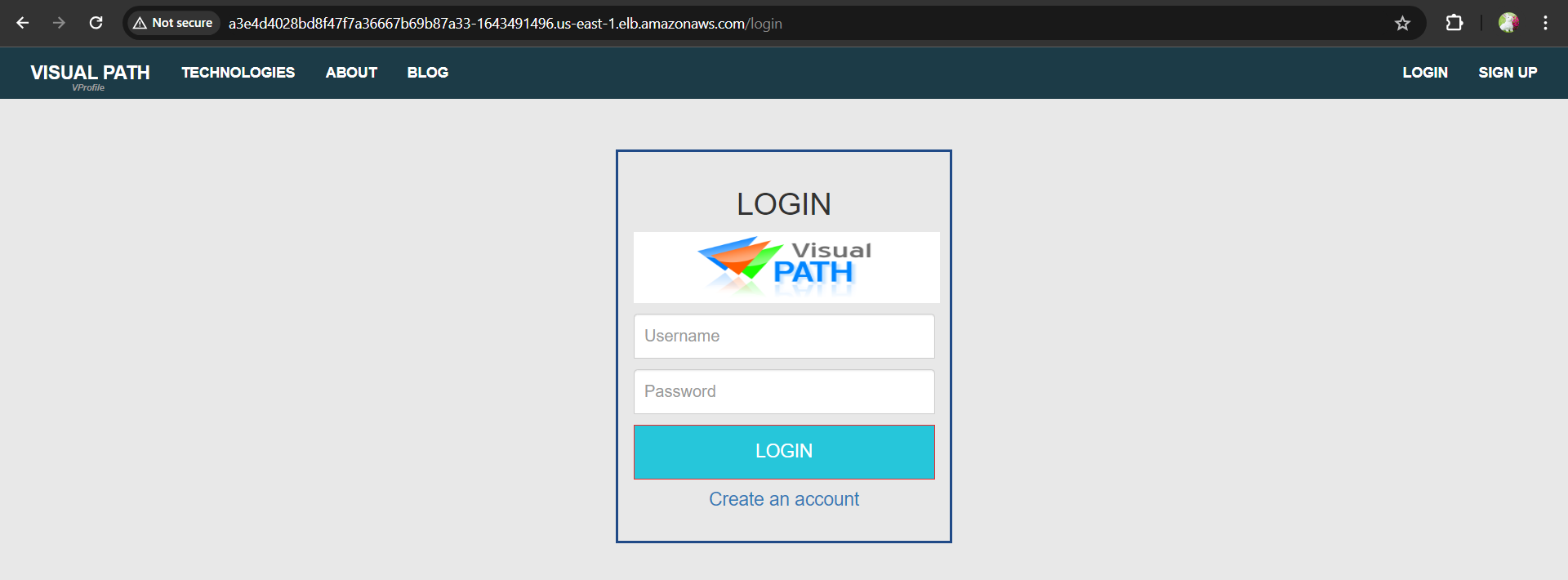
kubectl get all

**Load balance IP:**

External IP created for app service, which Is AWS Load Balancer DNS



Using this we can access the application.



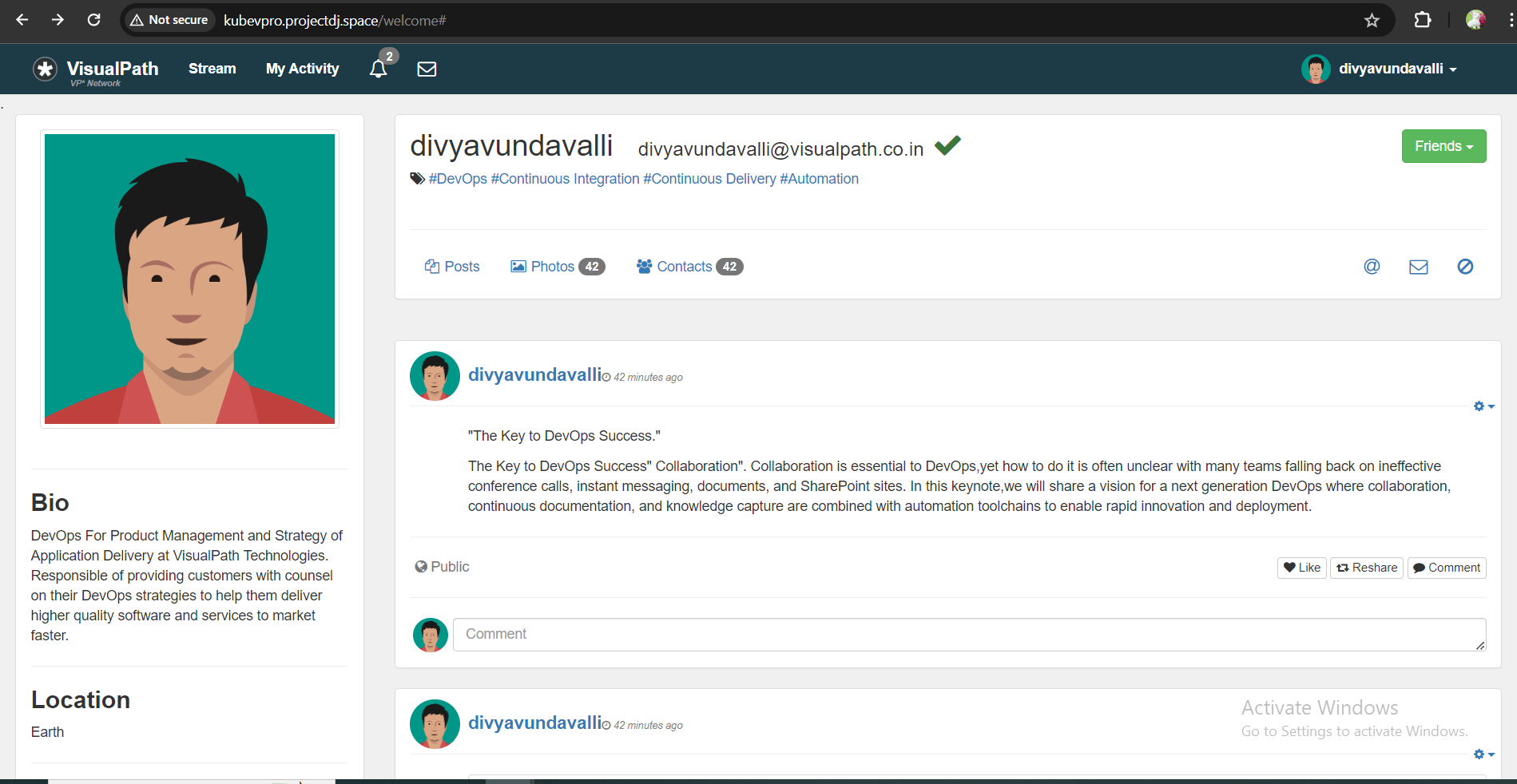
**Step 13: Create Route53 Record for Application Load Balancer**

To create a Route53 record for your application load balancer, follow these steps:

* Navigate to the Route53 service.
* Create a Record: Type A
* Select Region
* Load balancer type
* Select our load balancer

Using our domain <http://kubevpro.projectdj.space> we can get the Application:

Signup and log on to the Application:

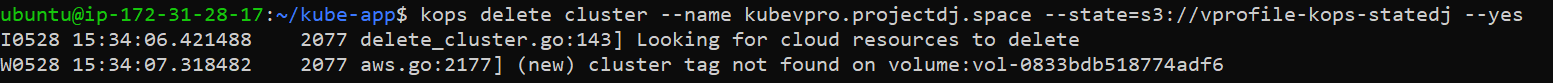


Explore all the functionalities of application.

**Step-14: Cleanup**

We will start with deleting our kubernetes services first.

kops delete cluster --name kubevpro.projectdj.space --state=s3://vprofile-kops-statedj --yes

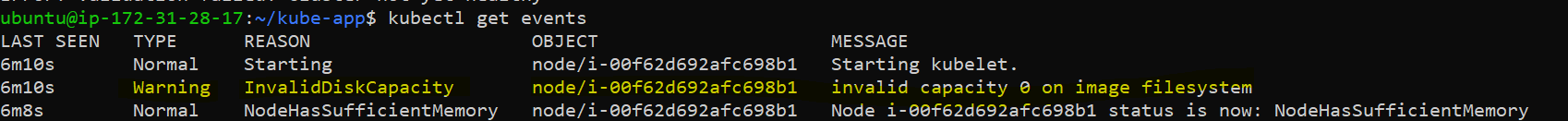


Then we will delete our cluster, s3 bucket, Route53 Hosted zone,EC2 instance

.

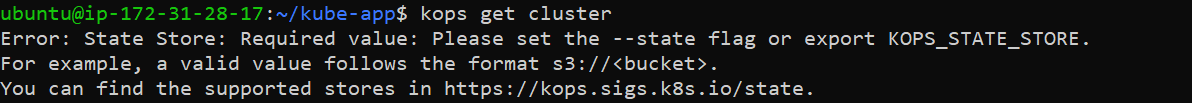
**Troubleshooting:**

**Issue1**: Cluster failed to set due to In-sufficient space



Consider launching the cluster with Medium or small EC2 instances to handle required amount of load

**Issue2:** Error at state Store



Ensure that the required environment variables are correctly set.

**Issue 3:**

If Cluster didn’t ready even after 15 mins then

Execute “**kubectl get events”** to gather information about any events or issues that might be preventing the cluster from becoming ready.

**Issue 4:**

parsing Error while applying yaml files



Review the YAML files for syntax errors or formatting issues. YAML is sensitive to indentation and syntax, so even a small space can cause parsing errors.