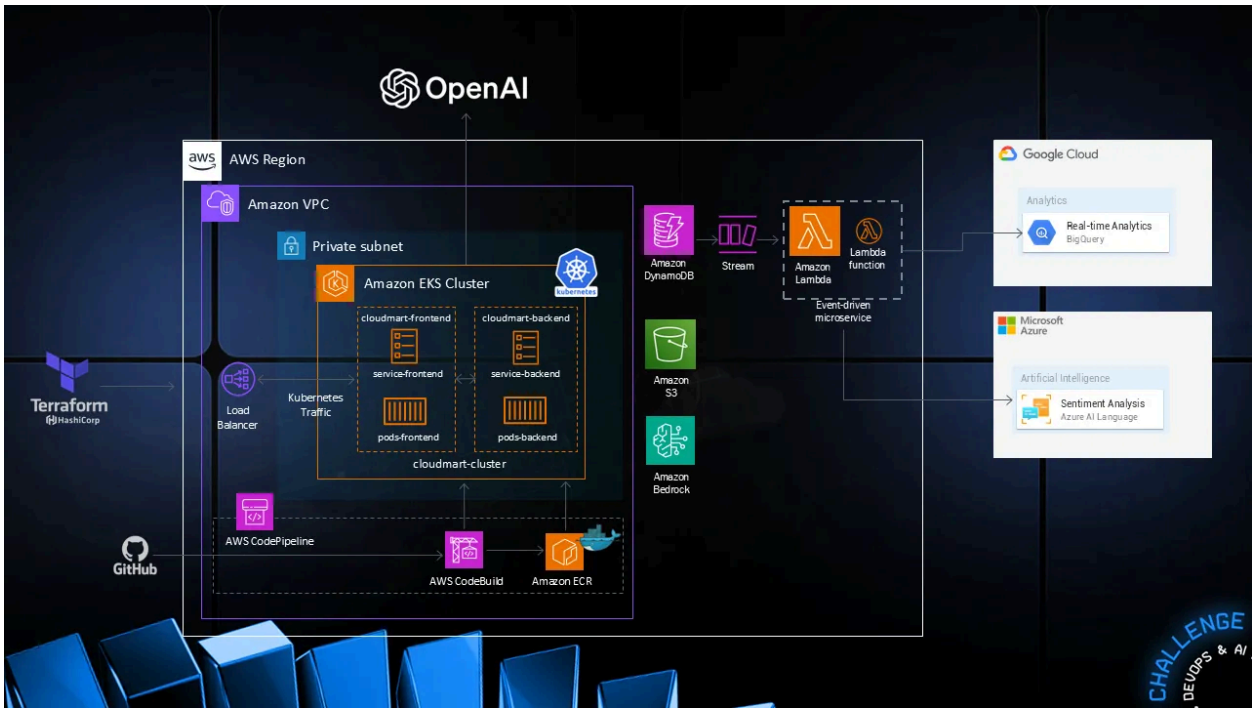
CLOUD MART PROJECT

(BUILDING AN E-COMMERCE WEBSITE)

HERE IS THE ARCHITECTURE OF OUR APPLICATION CLOUD MART:

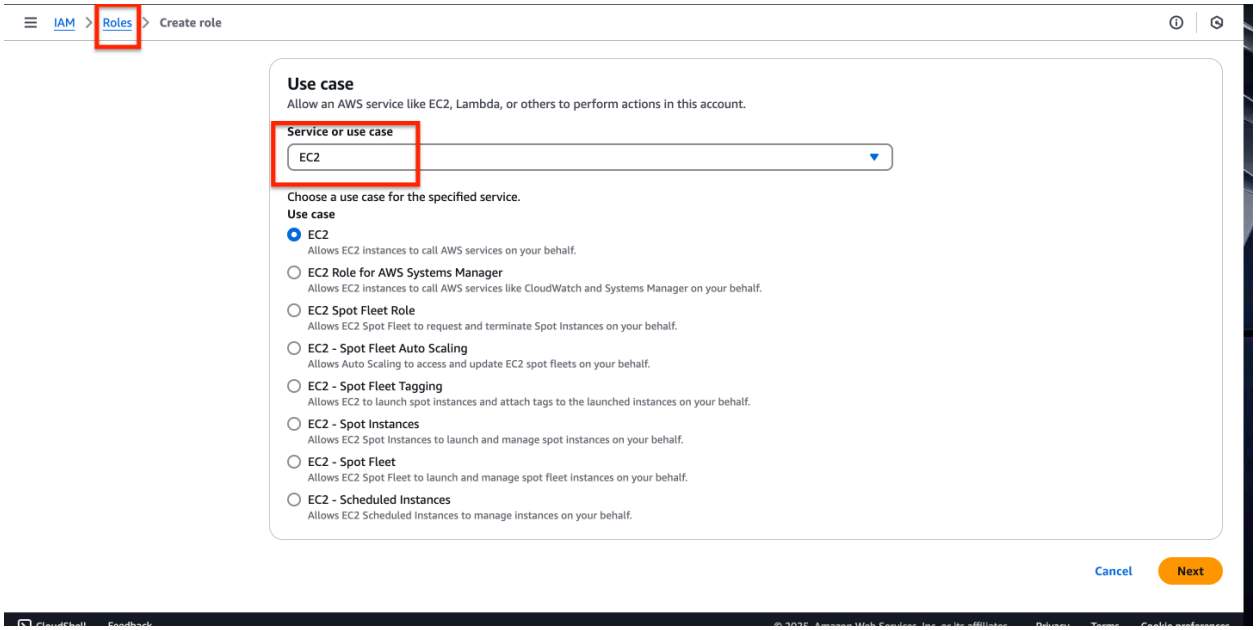


PHASE 1:

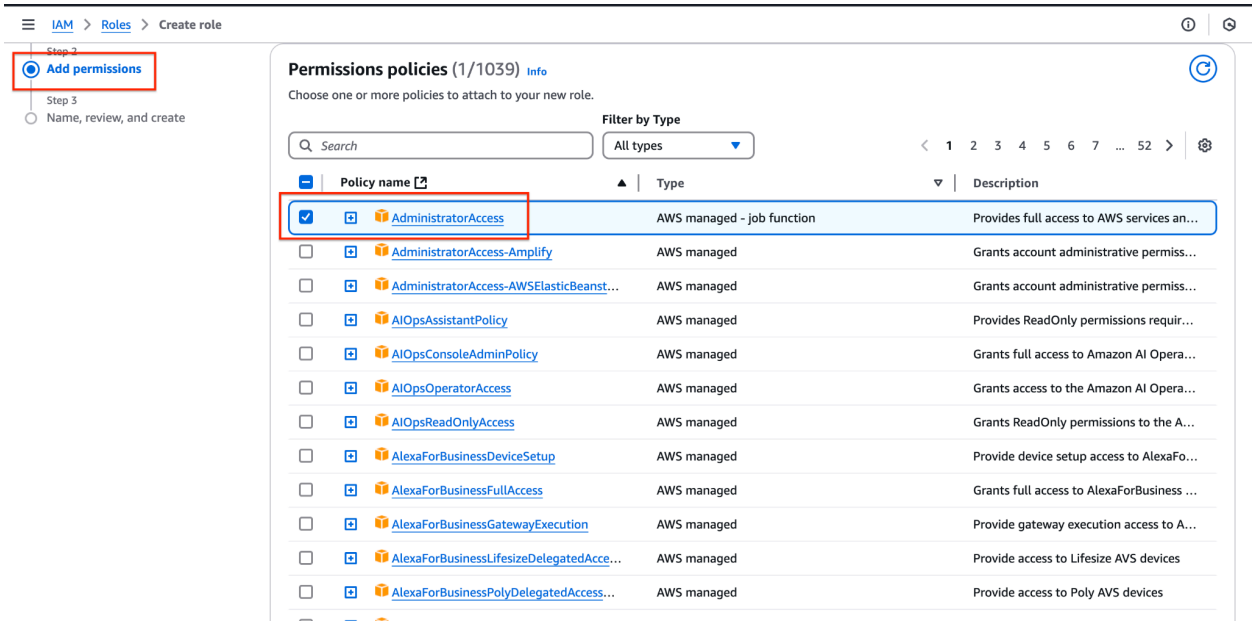
AUTOMATING AWS PROVISIONING WITH TERRAFORM USING CLAUDE

Step 1: Create IAM Role for EC2

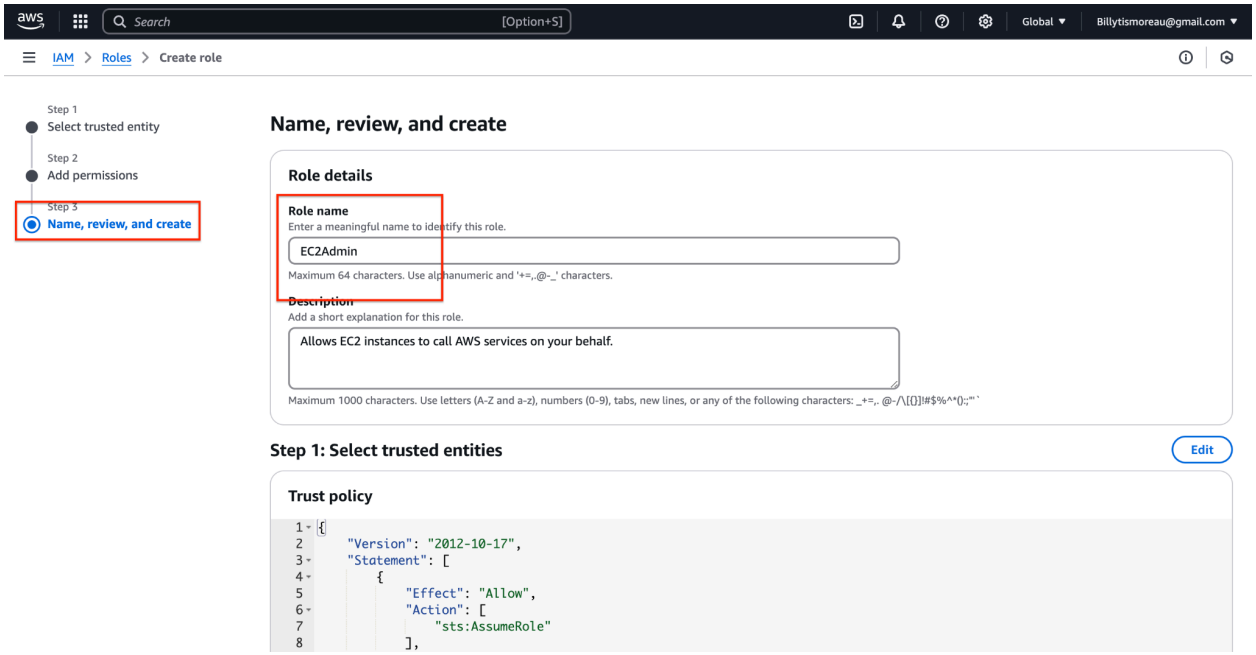
1. Log in to the AWS Management Console.
2. Navigate to the IAM dashboard.
3. Click "Roles" in the left sidebar, then "Create role".
4. Choose "AWS service" as the trusted entity type and "EC2" as the use case.

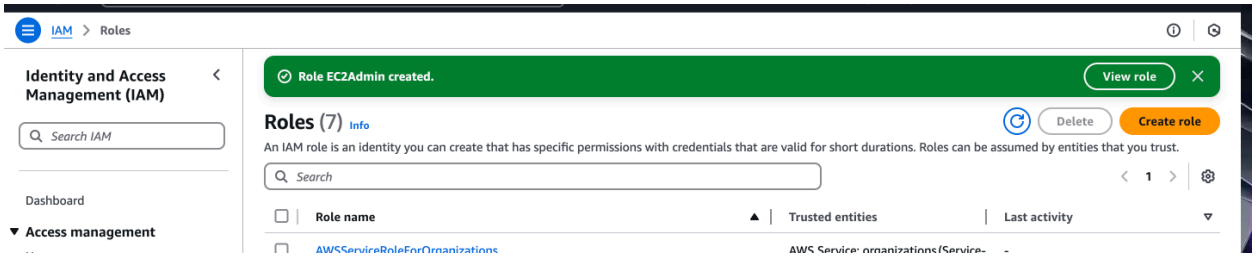


5. Search for and attach the "AdministratorAccess" policy. Note: In a production environment, use a more restricted policy.



6. Name the role "EC2Admin" and provide a description.
7. Review and create the role.

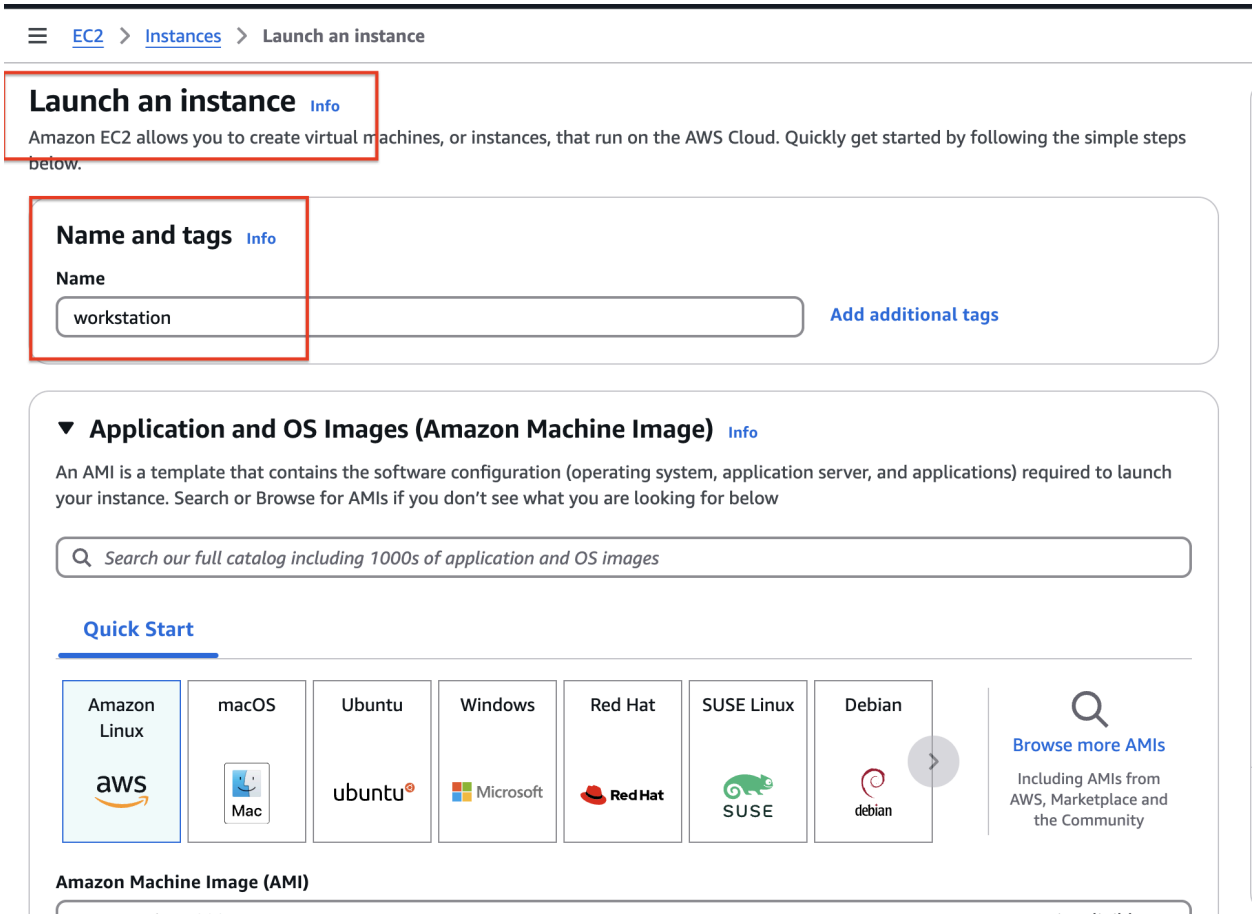




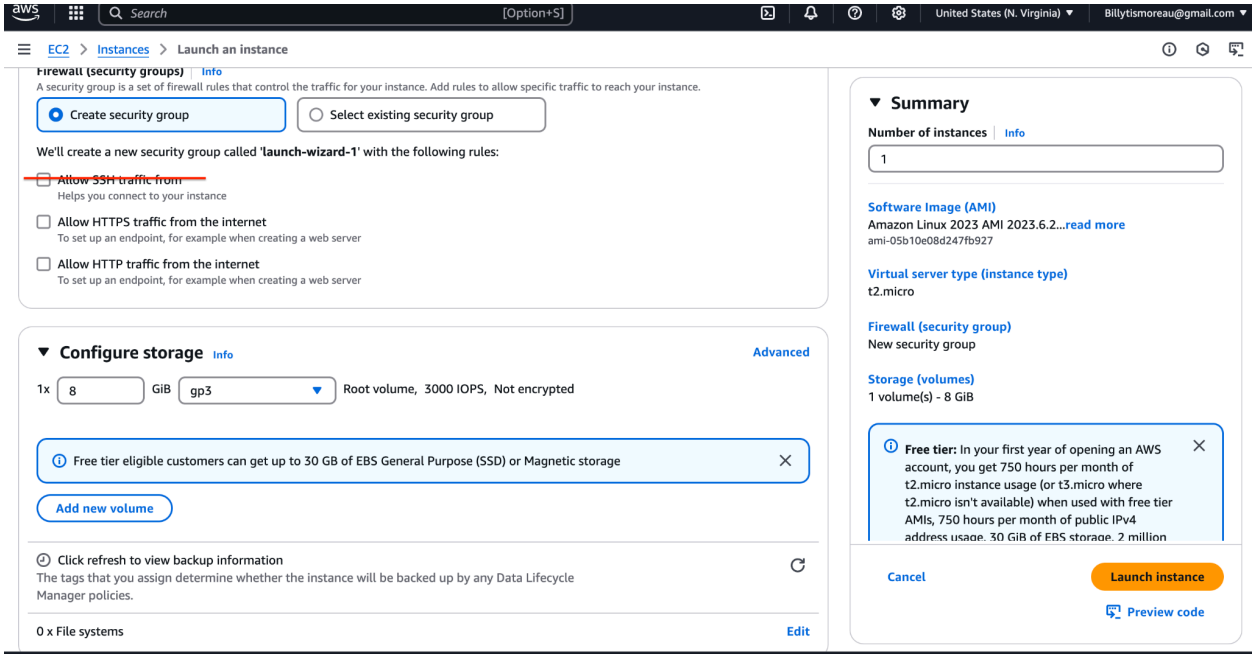
The Role been created:

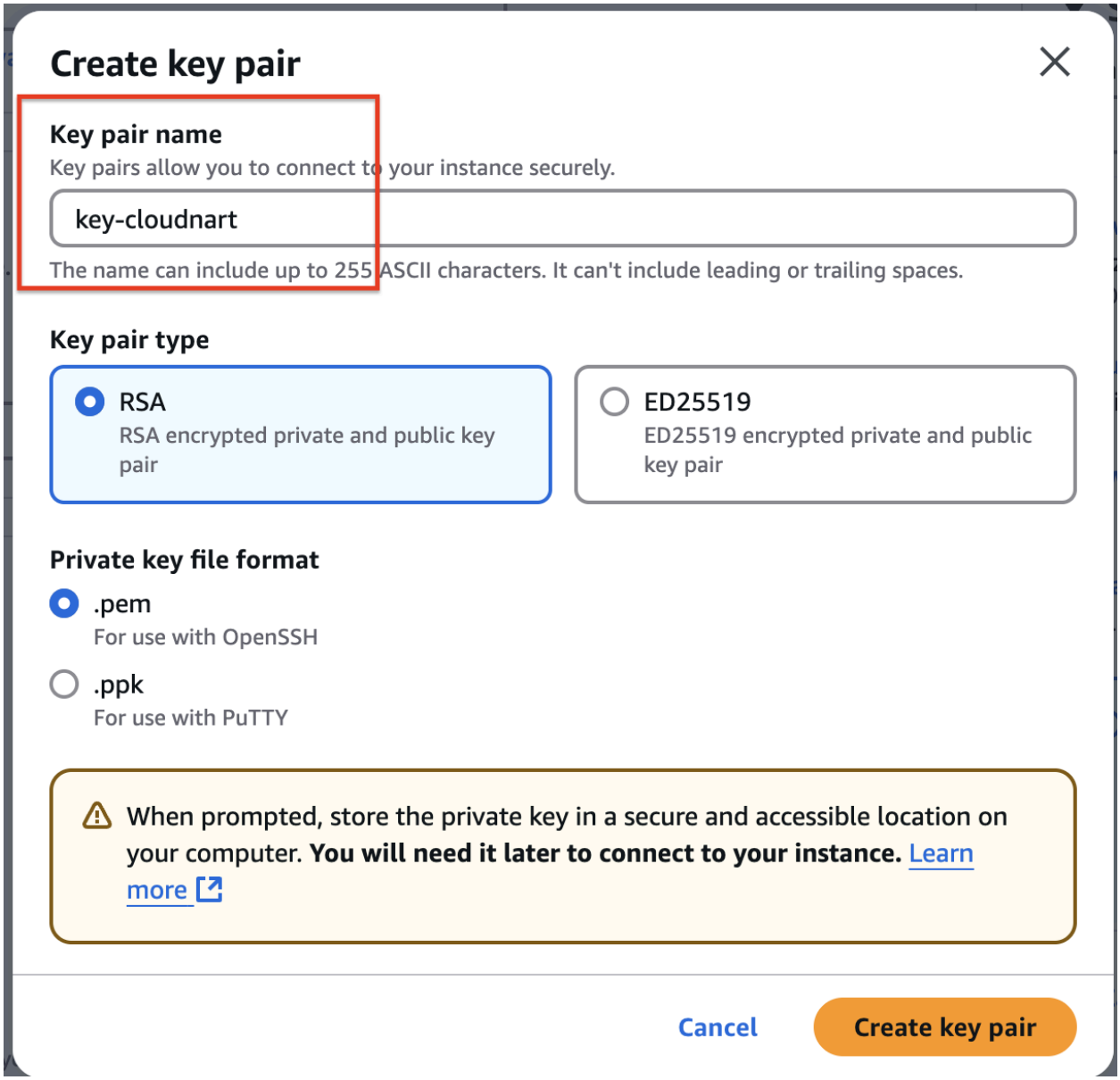
Step 2: Launch EC2 Instance

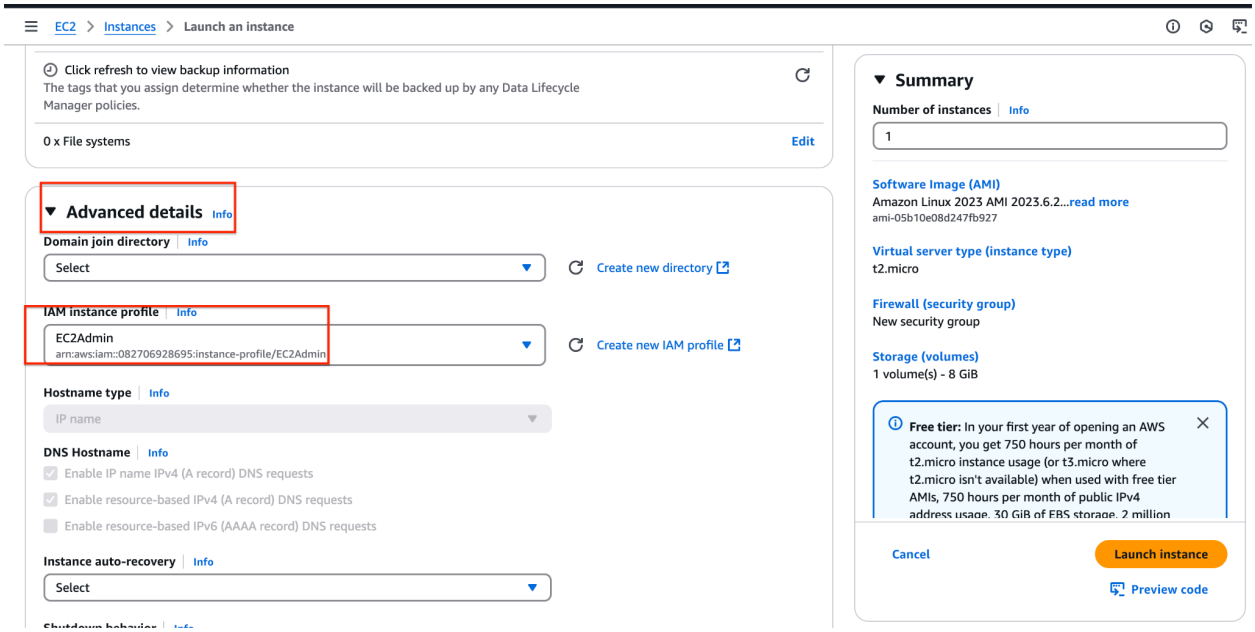
1. Go to the EC2 dashboard in the AWS Management Console.
2. Click "Launch Instance".



3. Choose an Amazon Linux 2 AMI.
4. Select a t2.micro instance type.
5. Configure instance details:
    - Network: Default VPC
    - Subnet: Any available
    - Auto-assign Public IP: Enable
    - IAM role: Select "EC2Admin"
6. Keep default storage settings.
7. Add a tag: Key="Name", Value="workstation".
8. Create a security group allowing SSH access from EC2 Connect IP.
9. Review and launch, selecting or creating a key pair.


aws
Search
[Option+S]
United States (N. Virginia)
Billytismoreau@gmail.com
EC2 > Instances > Launch an instance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
We'll create a new security group called 'launch-wizard-1' with the following rules:
Allow SSH traffic from
Helps you connect to your instance
Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server
Configure storage Info
Advanced
1x 8 GiB gp3 Root volume, 3000 IOPS, Not encrypted
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage
Add new volume
Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.
0 x File systems
Edit
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2...read more
ami-05b10e08d247fb927
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million
Cancel
Launch instance
Preview code

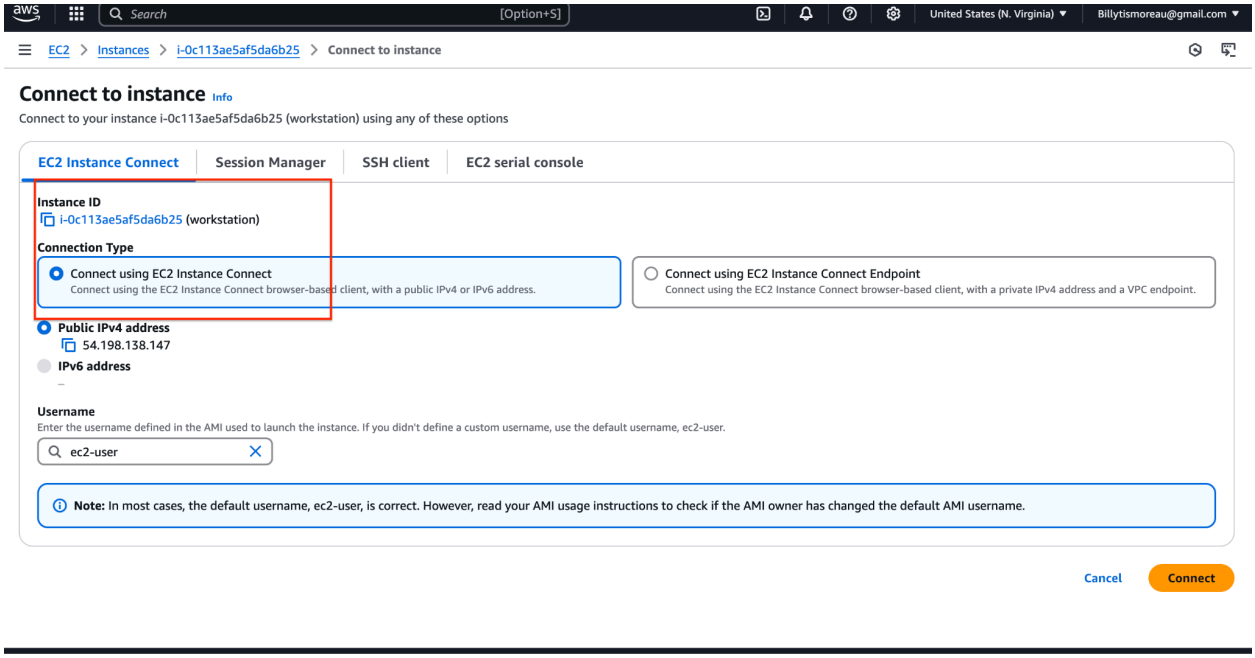

Create key pair
Key pair name
Key pairs allow you to connect to your instance securely.
key-cloudnart
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.
Key pair type
RSA
RSA encrypted private and public key pair
ED25519
ED25519 encrypted private and public key pair
Private key file format
.pem
For use with OpenSSH
.ppk
For use with PuTTY
When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. Learn more
Cancel
Create key pair



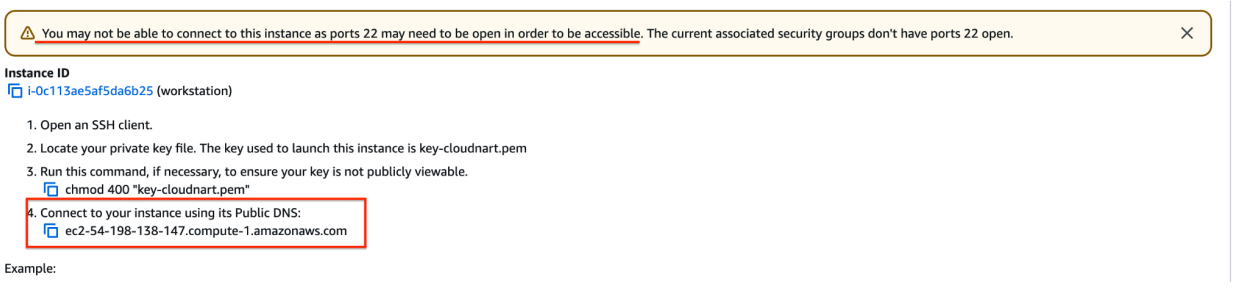
Step 3: Connect to EC2 Instance

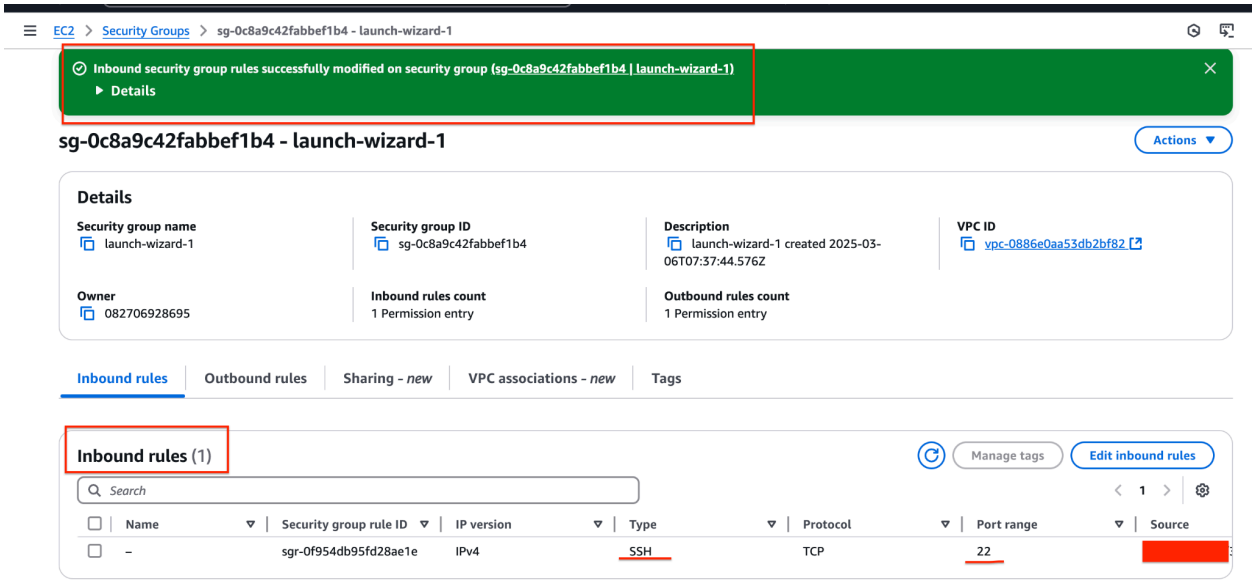
From the EC2 dashboard, select your "workstation" instance.

1. Click "Connect" and use the "EC2 Instance Connect" method.
2. In the browser-based SSH session, update system packages:

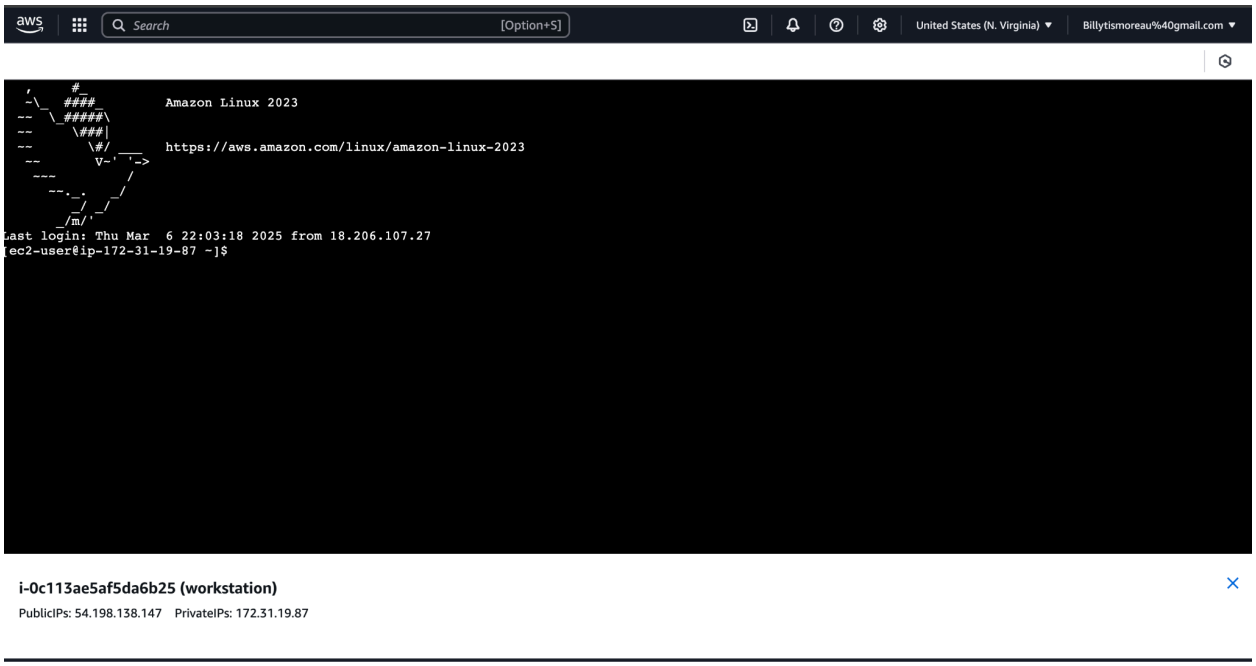


We had to open the port 22 of this instance in order to connect via SSH to it





Finally we were connected to our EC2 instance



Step 4: Install Terraform

In the browser-based SSH session, update system packages:

sudo yum update -y

Install yum-utils:

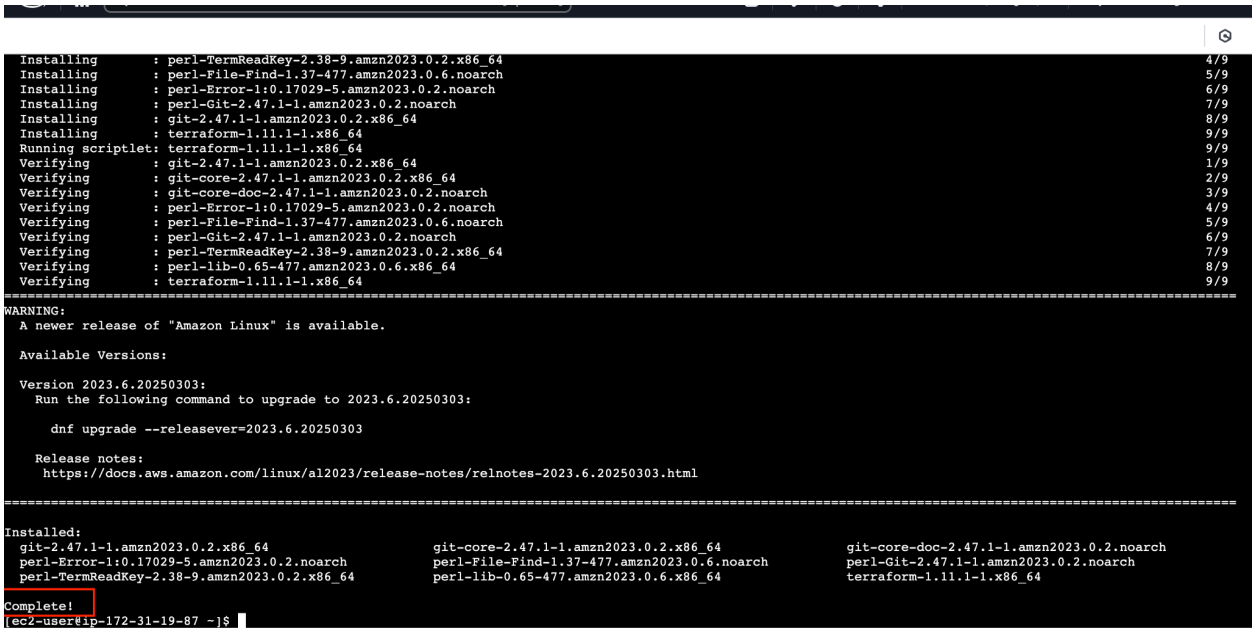
sudo yum install -y yum-utils

Add HashiCorp repository:

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

Install Terraform:

sudo yum -y install terraform

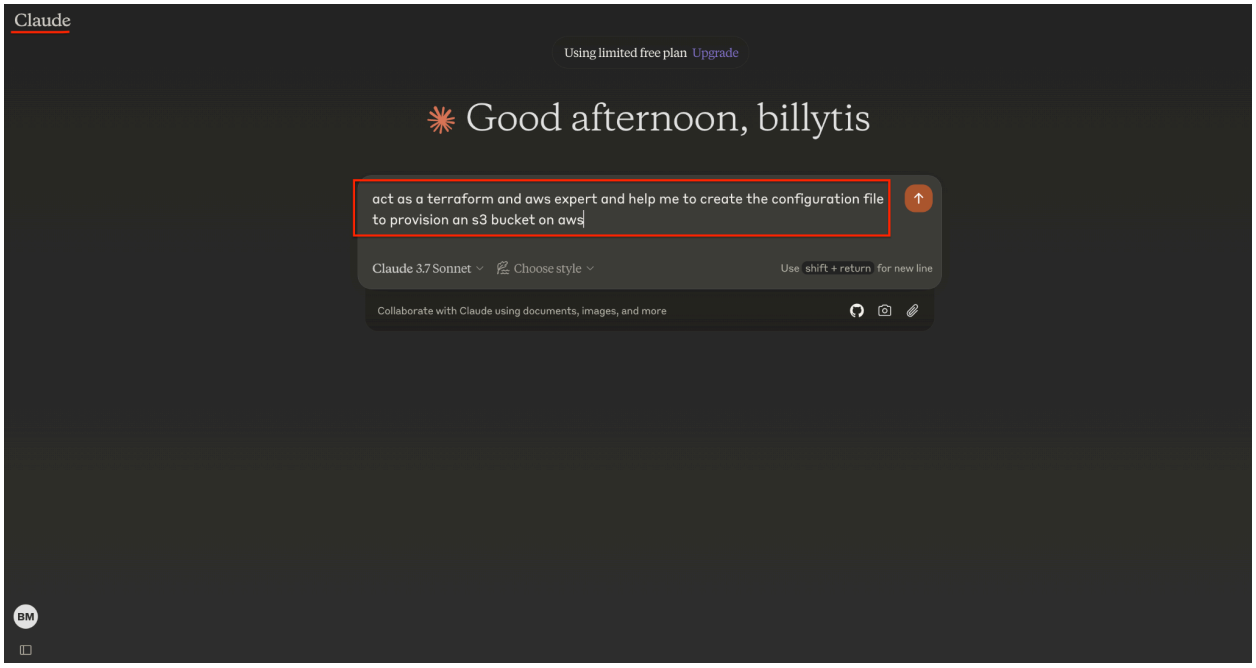


Verify installation:

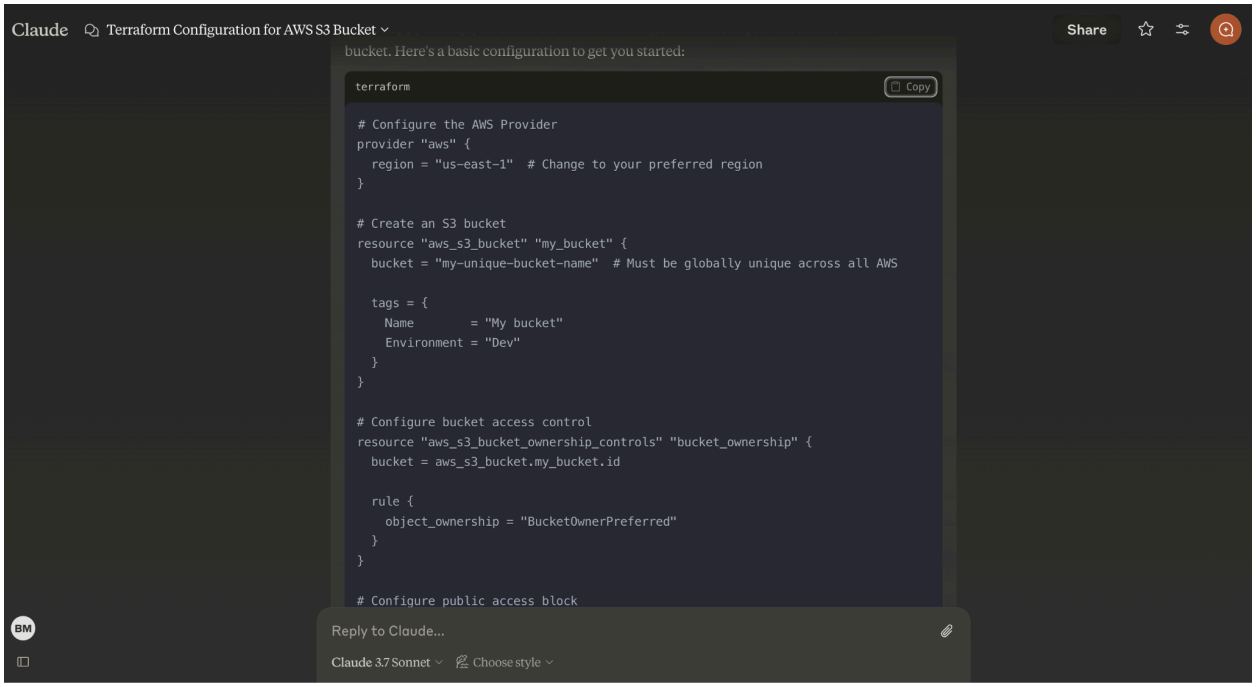
terraform version

Step 5: Create a resource using Claude as AI Assistant to Terraform

Ask Claude to create Terraform code for an S3 bucket



And the code been generated:



I asked Claude to change the region of the bucket and generate a random and unique name for the bucket:

Step 6 : Apply terraform and navigate to it:

Create a new directory and navigate to it:

mkdir terraform-project && cd terraform-project

Create and open main.tf

nano main.tf

Paste the Terraform **code generated** by Claude in Step 5.

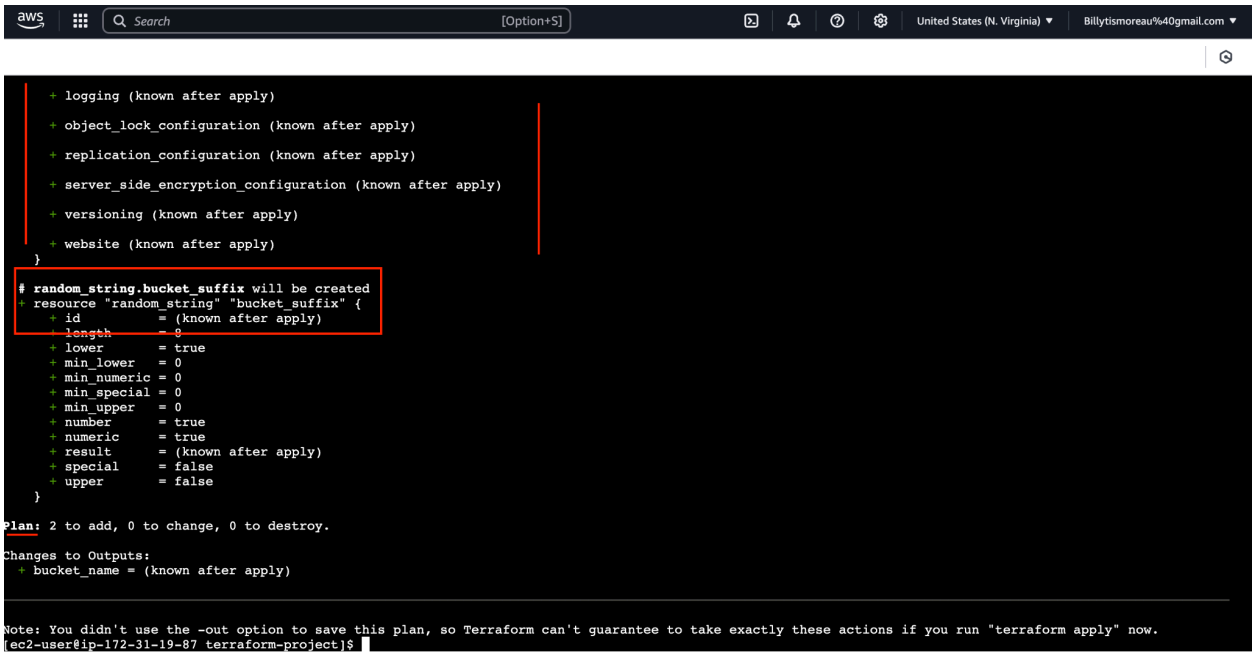
Save and exit the editor (in nano, press Ctrl+X, then Y, then Enter).

Initialize Terraform:

terraform init

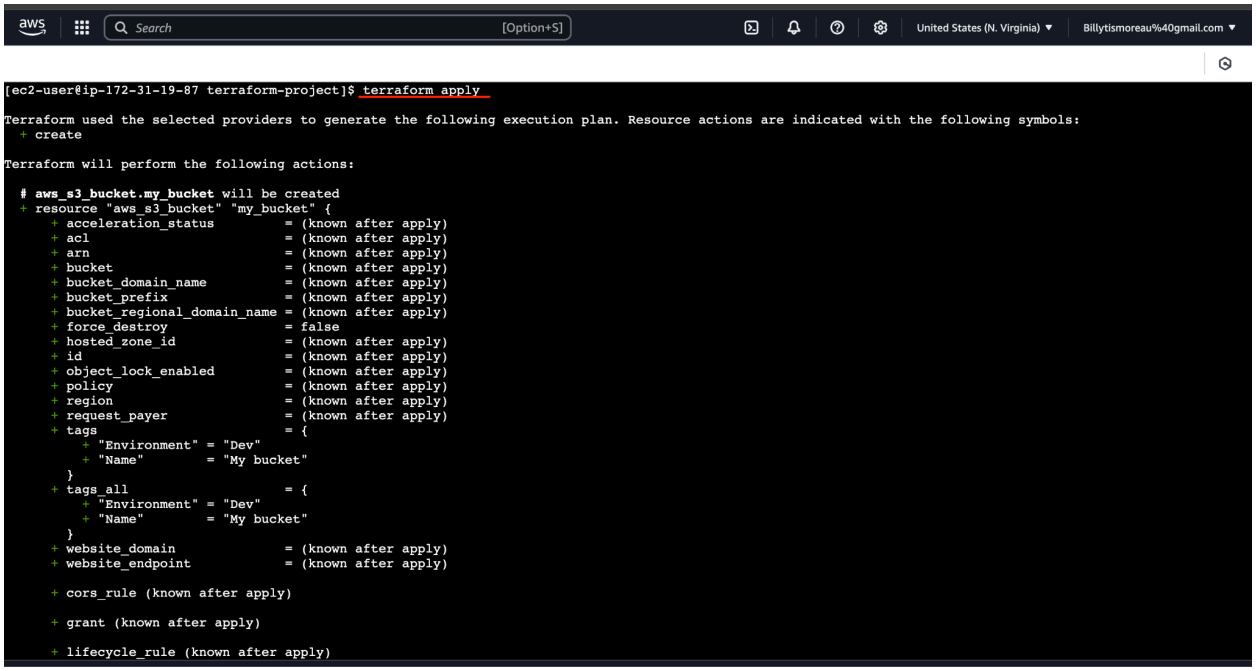
Review the plan:

terraform plan



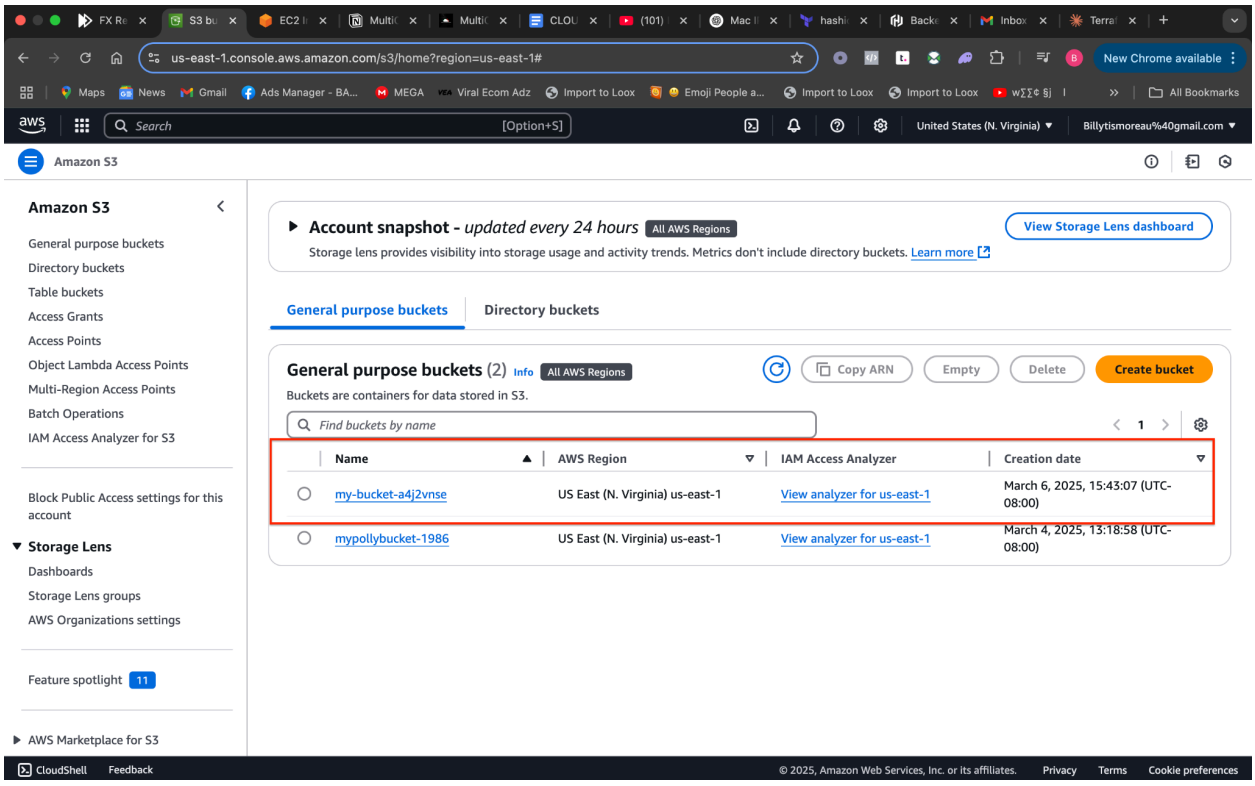
Apply the configuration:

Terraform apply



And the bucket been created

```
  # random_string.bucket_suffix will be created
  + resource "random_string" "bucket_suffix" {
      + id          = (known after apply)
      + length      = 8
      + lower       = true
      + min_lower   = 0
      + min_numeric = 0
      + min_special = 0
      + min_upper   = 0
      + number      = true
      + numeric     = true
      + result      = (known after apply)
      + special     = false
      + upper       = false
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + bucket_name = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

random_string.bucket_suffix: Creating...
random_string.bucket_suffix: Creation complete after 0s [id=a4j2vnse]
aws_s3_bucket.my_bucket: Creating...
aws_s3_bucket.my_bucket: Creation complete after 0s [id=my-bucket-a4j2vnse]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

bucket_name = "my-bucket-a4j2vnse"
[ec2-user@ip-172-31-19-87 terraform-project]$
```



Step 7: Create the Cloud DynamoDB tables

i created dynamoDB tables for our e-commerce application:

1 table for products

1 table for orders

1 table for tickets (this is for customer service, an automated chat bot will be created for this)

I generated the code with claude and then Remove the S3 lines and add the lines below to create the DynamoDB tables used by CloudMart,

Here is the code:

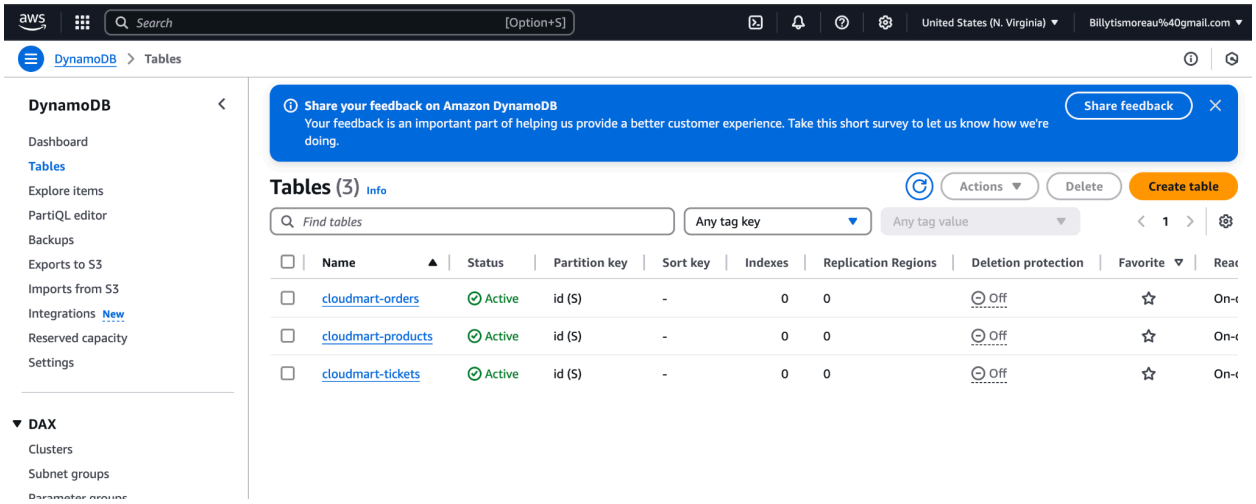
```
provider "aws" {
  region = "us-east-1"
}
# Tables DynamoDB
resource "aws_dynamodb_table" "cloudmart_products" {
  name           = "cloudmart-products"
  billing_mode   = "PAY_PER_REQUEST"
  hash_key       = "id"
  attribute {
    name = "id"
    type = "S"
  }
}
resource "aws_dynamodb_table" "cloudmart_orders" {
  name           = "cloudmart-orders"
  billing_mode   = "PAY_PER_REQUEST"
  hash_key       = "id"
  attribute {
    name = "id"
    type = "S"
  }
}
resource "aws_dynamodb_table" "cloudmart_tickets" {
  name           = "cloudmart-tickets"
  billing_mode   = "PAY_PER_REQUEST"
  hash_key       = "id"
  attribute {
    name = "id"
    type = "S"
```

```
  }
}
```

Then i applied the configuration with the command:

Terraform apply

Type **"yes"** when prompted to create the resources.



I successfully used Claude to generate Terraform code, set up an EC2 workstation, installed Terraform, and created an S3 bucket.

PHASE 2:

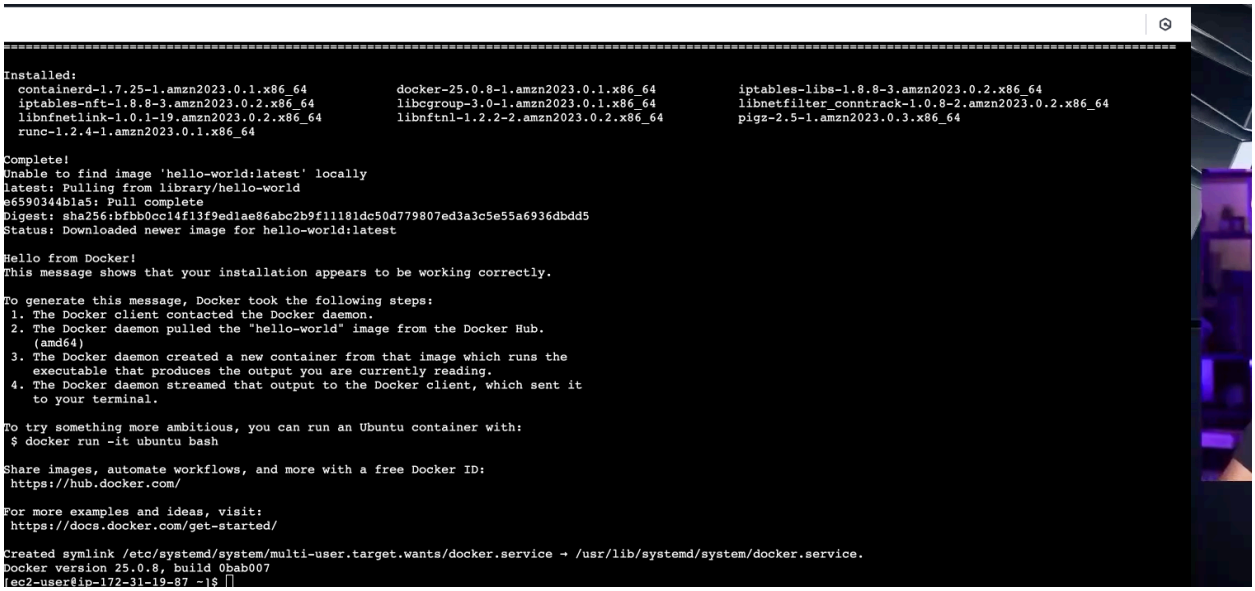
DEPLOYING THE APP WITH DOCKER & KUBERNETES ON AWS

(THE APP CONTAIN A BACKEND AND A FRONTEND)

Step 1: Install Docker on EC2

I executed the following commands in my EC2 instance:

sudo yum update -y
sudo yum install docker -y
sudo systemctl start docker
sudo docker run hello-world
sudo systemctl enable docker
docker --version
sudo usermod -a -G docker $(whoami)
newgrp docker



Docker is installed

Step 2: Create Docker image for CloudMart

(For the Backend of the application)

I created a folder and downloaded the source code:

mkdir -p challenge-day2/backend && cd challenge-day2/backend wget https://tcb-public-events.s3.amazonaws.com/mdac/resources/day2/cloudmart-backend.zip unzip cloudmart-backend.zi

Create .env file:

nano .env

Content of .env:

PORT=5000

AWS_REGION=us-east-1

BEDROCK_AGENT_ID=<your-bedrock-agent-id>

BEDROCK_AGENT_ALIAS_ID=<your-bedrock-agent-alias-id>

OPENAI_API_KEY=<your-openai-api-key>

OPENAI_ASSISTANT_ID=<your-openai-assistant-id>

Create Dockerfile:

nano Dockerfile

Content of Dockerfile:

```
FROM node:18
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5000
CMD ["npm", "start"]
```

(For the Frontend of the application)

I created a folder and download the source code:

```
cd ..
mkdir frontend && cd frontend
wget https://tcb-public-events.s3.amazonaws.com/mdac/resources/day2/cloudmart-frontend.zip
unzip cloudmart-frontend.zip
```

Create Dockerfile:

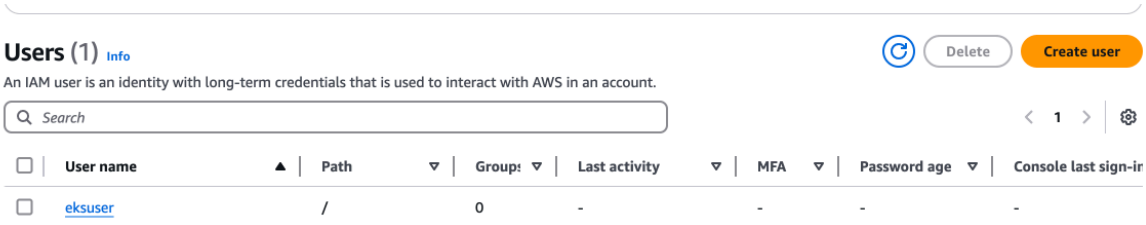
nano Dockerfile

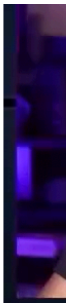
Content of Dockerfile:
FROM node:16-alpine as build
WORKDIR /app
COPY package*.json ./
RUN npm ci
COPY . .
RUN npm run build
FROM node:16-alpine
WORKDIR /app
RUN npm install -g serve
COPY --from=build /app/dist /app
ENV PORT=5001
ENV NODE_ENV=production
EXPOSE 5001
CMD ["serve", "-s", ".", "-l", "5001"]

STEP 3 (kubernetes)

Cluster Setup on AWS Elastic Kubernetes Services (EKS)

1) I created a user named `eksuser` with Admin privileges and authenticate with it





I used aws configure to make sure that the user ***"eksuser"*** is actually configure on our ec2 workstation virtual machine

2) Install the CLI tool `eksctl`

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname
-s)_amd64.tar.gz" | tar xz -C /tmp
sudo cp /tmp/eksctl /usr/bin
eksctl version
```

Install the CLI tool `kubectl` **In order to interact with the cluster**

```
curl -o kubectl
https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kube
ctl
chmod +x ./kubectl
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export
PATH=$PATH:$HOME/bin
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
kubectl version --short --client
```

3) Create an EKS Cluster

```
eksctl create cluster \
  --name cloudmart \
  --region us-east-1 \
  --nodegroup-name standard-workers \
  --node-type t3.medium \
  --nodes 1 \
  --with-oidc \
  --managed
```

```
2025-03-08 18:19:22 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-cluster"
2025-03-08 18:20:22 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-cluster"
2025-03-08 18:21:22 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-cluster"
2025-03-08 18:21:23 [!]  recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl cannot configure the requested
 permissions; the recommended way to provide IAM permissions for "vpc-cni" addon is via pod identity associations; after addon creation is completed, add all
recommended policies to the config file, under `addon.PodIdentityAssociations`, and run `eksctl update addon`
2025-03-08 18:21:23 [ℹ]  creating addon: vpc-cni
2025-03-08 18:21:23 [ℹ]  successfully created addon: vpc-cni
2025-03-08 18:21:24 [ℹ]  creating addon: kube-proxy
2025-03-08 18:21:24 [ℹ]  successfully created addon: kube-proxy
2025-03-08 18:21:24 [ℹ]  creating addon: coredns
2025-03-08 18:21:25 [ℹ]  successfully created addon: coredns
2025-03-08 18:21:25 [ℹ]  creating addon: metrics-server
2025-03-08 18:21:25 [ℹ]  successfully created addon: metrics-server
2025-03-08 18:23:26 [ℹ]  addon "vpc-cni" active
2025-03-08 18:23:27 [ℹ]  deploying stack "eksctl-cloudmart-addon-vpc-cni"
2025-03-08 18:23:27 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-addon-vpc-cni"
2025-03-08 18:23:57 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-addon-vpc-cni"
2025-03-08 18:23:57 [ℹ]  updating addon
2025-03-08 18:24:07 [ℹ]  addon "vpc-cni" active
2025-03-08 18:24:07 [ℹ]  building managed nodegroup stack "eksctl-cloudmart-nodegroup-standard-workers"
2025-03-08 18:24:08 [ℹ]  deploying stack "eksctl-cloudmart-nodegroup-standard-workers"
2025-03-08 18:24:08 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-nodegroup-standard-workers"
2025-03-08 18:24:38 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-nodegroup-standard-workers"
2025-03-08 18:25:35 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-nodegroup-standard-workers"
2025-03-08 18:26:47 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-nodegroup-standard-workers"
2025-03-08 18:26:47 [ℹ]  waiting for the control plane to become ready
2025-03-08 18:26:48 [✔]  saved kubeconfig as "/home/ec2-user/.kube/config"
2025-03-08 18:26:48 [ℹ]  no tasks
2025-03-08 18:26:48 [✔]  all EKS cluster resources for "cloudmart" have been created
2025-03-08 18:26:48 [ℹ]  nodegroup "standard-workers" has 1 node(s)
2025-03-08 18:26:48 [ℹ]  node "ip-192-168-24-198.ec2.internal" is ready
2025-03-08 18:26:48 [ℹ]  waiting for at least 1 node(s) to become ready in "standard-workers"
2025-03-08 18:26:48 [ℹ]  nodegroup "standard-workers" has 1 node(s)
2025-03-08 18:26:48 [ℹ]  node "ip-192-168-24-198.ec2.internal" is ready
2025-03-08 18:26:48 [✔]  created 1 managed nodegroup(s) in cluster "cloudmart"
2025-03-08 18:26:49 [ℹ]  kubectl command should work with "/home/ec2-user/.kube/config", try 'kubectl get nodes'
2025-03-08 18:26:49 [✔]  EKS cluster "cloudmart" in "us-east-1" region is ready
[ec2-user@ip-172-31-19-87 frontend]$
```

After 15 minutes the cluster has been created and now its time to connect to the cluster and interact

4) Connect to the EKS cluster using the `kubectl` configuration

aws eks update-kubeconfig --name cloudmart

5) Verify Cluster Connectivity

kubectl get svc kubectl get nodes

```
[ec2-user@ip-172-31-19-87 frontend]$ aws eks update-kubeconfig --name cloudmart
Added new context arn:aws:eks:us-east-1:082706928695:cluster/cloudmart to /home/ec2-user/.kube/config
[ec2-user@ip-172-31-19-87 frontend]$ kubectl get svc
kubectl get nodes
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   18m
NAME                                STATUS   ROLES    AGE   VERSION
ip-192-168-24-198.ec2.internal   Ready    <none>   10m   v1.30.9-eks-5d632ec
[ec2-user@ip-172-31-19-87 frontend]$
```

So everything works and as you can see we have 1 node, now in order to access and interact with other services with aws i had to create a role and 1 iamservice account

7) Create a Role & Service Account to provide pods access to services used by the application (DynamoDB, Bedrock, etc).

```
eksctl create iamserviceaccount \
  --cluster=cloudmart \
  --name=cloudmart-pod-execution-role \
  --role-name CloudMartPodExecutionRole \
  --attach-policy-arn=arn:aws:iam::aws:policy/AdministratorAccess\
```

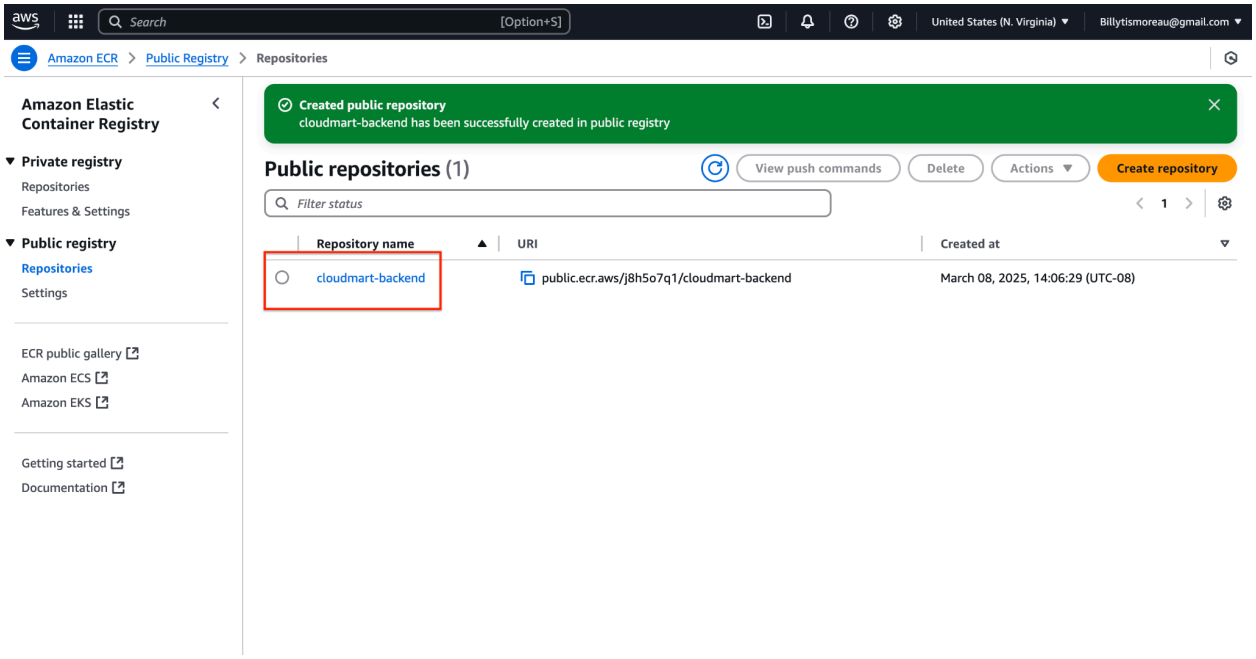
**--region us-east-1 **
--approve

```
[ec2-user@ip-172-31-19-87 frontend]$ aws eks update-kubeconfig --name cloudmart
Added new context arn:aws:eks:us-east-1:082706928695:cluster/cloudmart to /home/ec2-user/.kube/config
[ec2-user@ip-172-31-19-87 frontend]$ kubectl get svc
kubectl get nodes
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   18m
NAME                           STATUS   ROLES    AGE   VERSION
ip-192-168-24-198.ec2.internal   Ready    <none>   10m   v1.30.9-eks-5d632ec
[ec2-user@ip-172-31-19-87 frontend]$ eksctl create iamserviceaccount \
  --cluster=cloudmart \
  --name=cloudmart-pod-execution-role \
  --role-name CloudMartPodExecutionRole \
  --attach-policy-arn=arn:aws:iam::aws:policy/AdministratorAccess\
  --region us-east-1 \
  --approve
2025-03-08 18:50:33 [ℹ]  1 iamserviceaccount (default/cloudmart-pod-execution-role) was included (based on the include/exclude rules)
2025-03-08 18:50:33 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
2025-03-08 18:50:33 [ℹ]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "default/cloudmart-pod-execution-role",
        create serviceaccount "default/cloudmart-pod-execution-role",
    } }2025-03-08 18:50:33 [ℹ]  building iamserviceaccount stack "eksctl-cloudmart-addon-iamserviceaccount-default-cloudmart-pod-execution-role"
2025-03-08 18:50:34 [ℹ]  deploying stack "eksctl-cloudmart-addon-iamserviceaccount-default-cloudmart-pod-execution-role"
2025-03-08 18:50:34 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-addon-iamserviceaccount-default-cloudmart-pod-execution-role"
2025-03-08 18:51:04 [ℹ]  waiting for CloudFormation stack "eksctl-cloudmart-addon-iamserviceaccount-default-cloudmart-pod-execution-role"
2025-03-08 18:51:04 [ℹ]  created serviceaccount "default/cloudmart-pod-execution-role"
[ec2-user@ip-172-31-19-87 frontend]$
```

The cluster is ready for deployment now

Step 4) Backend Deployment on Kubernetes

Create an ECR Repository for the Backend and upload the Docker image to it:
Repository name: cloudmart-backend



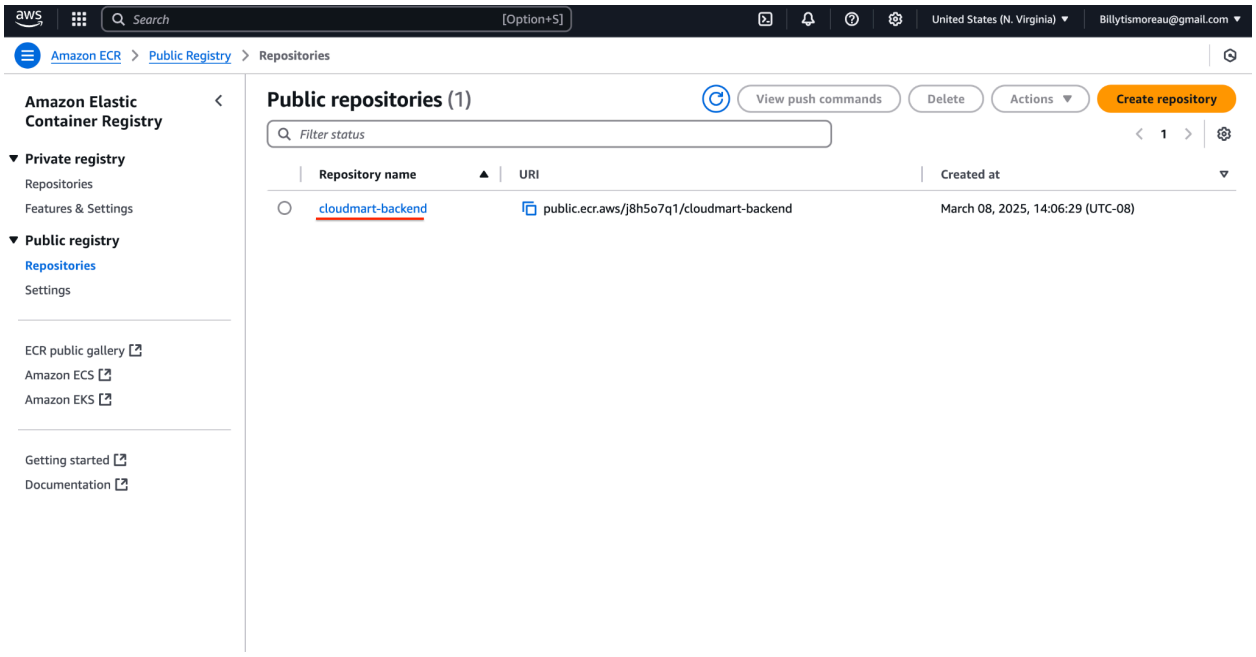
Switch to backend folder

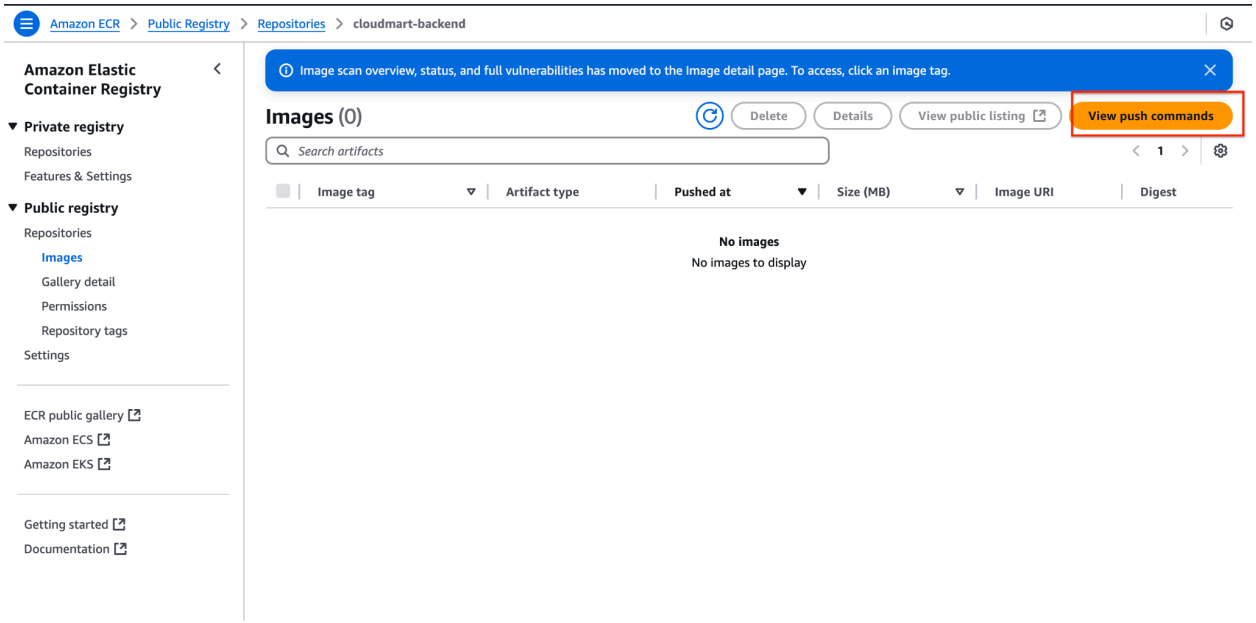
cd ../..
cd challenge-day2/backend

Create a Kubernetes deployment file (YAML) for the Backend

cd ../.. cd challenge-day2/backend nano cloudmart-backend.yaml

Then i followed the ECR steps provided in the aws console


Search
[Option+S]
United States (N. Virginia)
Billytismoreau@gmail.com
Amazon ECR
Public Registry
Repositories
Amazon Elastic Container Registry
Private registry
Repositories
Features & Settings
Public registry
Repositories
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Documentation
Public repositories (1)
View push commands
Delete
Actions
Create repository
Filter status
Repository name
URI
Created at
cloudmart-backend
public.ecr.aws/j8h5o7q1/cloudmart-backend
March 08, 2025, 14:06:29 (UTC-08)


Amazon ECR
Public Registry
Repositories
cloudmart-backend
Amazon Elastic Container Registry
Private registry
Repositories
Features & Settings
Public registry
Repositories
Images
Gallery detail
Permissions
Repository tags
Settings
ECR public gallery
Amazon ECS
Amazon EKS
Getting started
Documentation
Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag.
Images (0)
Delete
Details
View public listing
View push commands
Search artifacts
Image tag
Artifact type
Pushed at
Size (MB)
Image URI
Digest
No images
No images to display

Push commands for cloudmart-backend

macOS / Linux | Windows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see Getting Started with Amazon ECR.

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry Authentication.

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:

   ```
   aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/j8h5o7q1
   ```

   Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions here. You can skip this step if your image is already built:

   ```
   docker build -t cloudmart-backend .
   ```

3. After the build completes, tag your image so you can push the image to this repository:

   ```
   docker tag cloudmart-backend:latest public.ecr.aws/j8h5o7q1/cloudmart-backend:latest
   ```

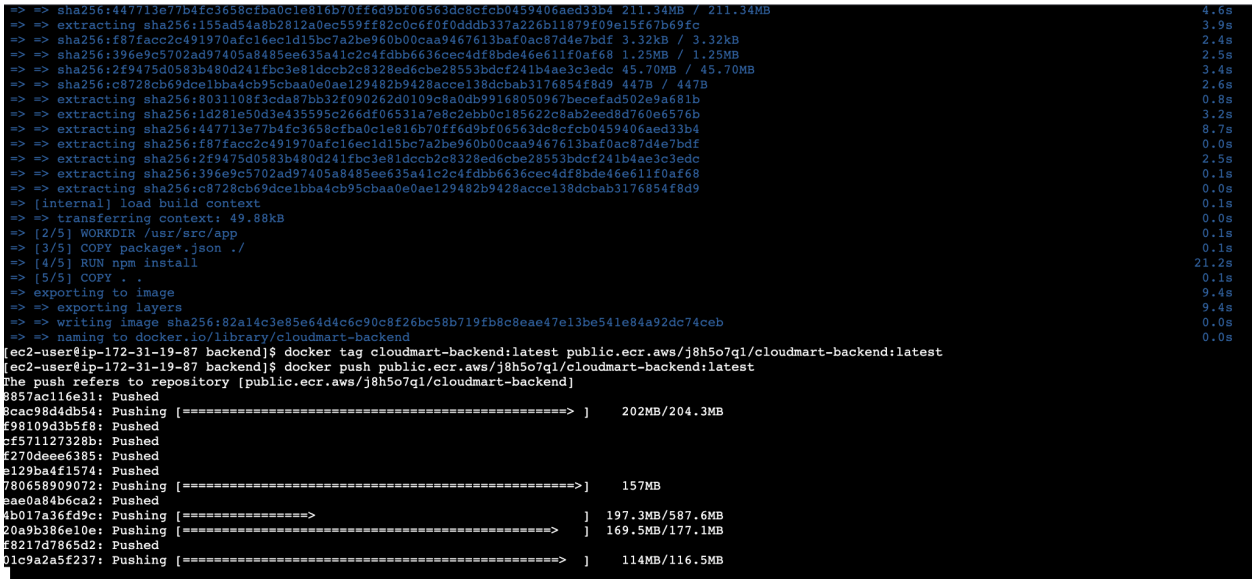
4. Run the following command to push this image to your newly created AWS repository:

   ```
   docker push public.ecr.aws/j8h5o7q1/cloudmart-backend:latest
   ```

Close

Now it is done building the image in the docker file

The docker push command has been launch:



The push is done

I created a Kubernetes deployment file (YAML) for the Backend

nano cloudmart-backend.yaml

And pasted the code

apiVersion: apps/v1
kind: Deployment
metadata:
*** name: cloudmart-backend-app***
spec:

```
  replicas: 1
  selector:
    matchLabels:
      app: cloudmart-backend-app
  template:
    metadata:
      labels:
        app: cloudmart-backend-app
    spec:
      serviceAccountName: cloudmart-pod-execution-role
      containers:
      - name: cloudmart-backend-app
        image: public.ecr.aws/j8h5o7q1/cloudmart-backend:latest
        env:
        - name: PORT
          value: "5000"
        - name: AWS_REGION
          value: "us-east-1"
        - name: BEDROCK_AGENT_ID
          value: "xxxxxx"
        - name: BEDROCK_AGENT_ALIAS_ID
          value: "xxxx"
        - name: OPENAI_API_KEY
          value: "xxxxxx"
        - name: OPENAI_ASSISTANT_ID
          value: "xxxx"
---
apiVersion: v1
kind: Service
metadata:
  name: cloudmart-backend-app-service
spec:
  type: LoadBalancer
  selector:
    app: cloudmart-backend-app
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
```

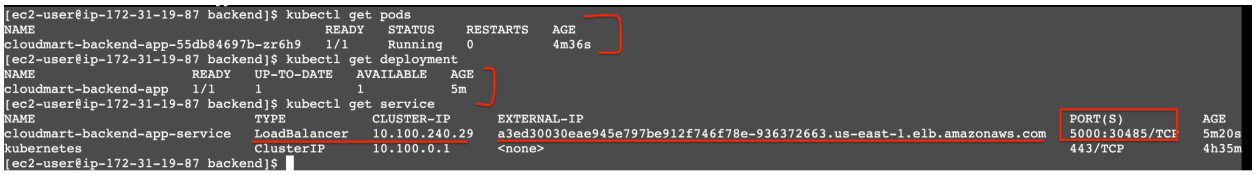
I deployed the Backend on Kubernetes:

kubectl apply -f cloudmart-backend.yaml

Done

Then i monitored the status of objects being created and obtain the public IP generated for the API

kubectl get pods kubectl get deployment kubectl get service



Step 5) frontend Deployment on Kubernetes

Change the folder to the frontend:

cd ../challenge-day2/frontend

I created a env file for the frontend

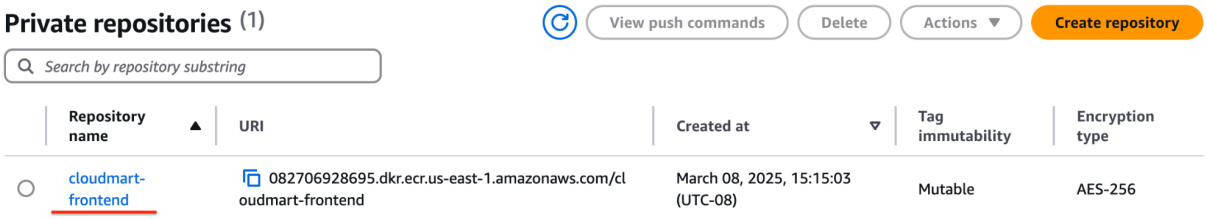
nano .env

I insert the environment variable containing the API URL created within Kubernetes

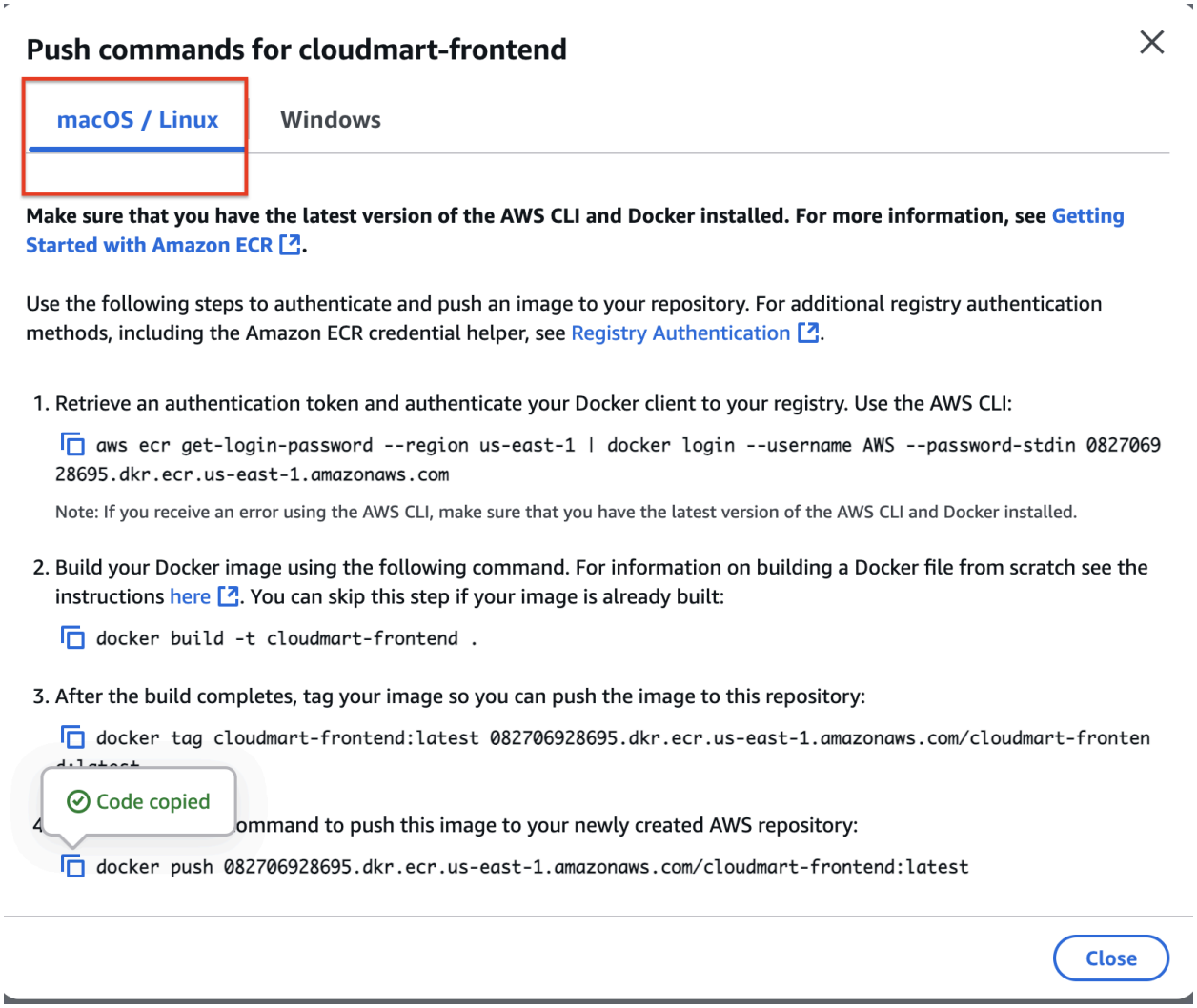
VITE_API_BASE_URL=http://a3ed30030eae945e797be912f746f78e-936372663.us-east-1.elb.amazonaws.com:5000/api

I created an ECR Repository for the Frontend and upload the Docker image to it

Cloudmart-frontend



I Followed the steps from the ECR documentation on the aws console



I create a Kubernetes deployment file (YAML) for the Frontend

nano cloudmart-frontend.yaml

And i pasted the code:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cloudmart-frontend-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: cloudmart-frontend-app
  template:
    metadata:
```

```
    labels:
      app: cloudmart-frontend-app
  spec:
    serviceAccountName: cloudmart-pod-execution-role
    containers:
    - name: cloudmart-frontend-app
      image: 082706928695.dkr.ecr.us-east-1.amazonaws.com/cloudmart-frontend:latest
---
apiVersion: v1
kind: Service
metadata:
  name: cloudmart-frontend-app-service
spec:
  type: LoadBalancer
  selector:
    app: cloudmart-frontend-app
  ports:
    - protocol: TCP
      port: 5001
      targetPort: 5001
```

And i then i deployed the frontend on Kubernetes

kubectl apply -f cloudmart-frontend.yaml

Its now deployed:

```
[ec2-user@ip-172-31-19-87 frontend]$ nano cloudmart-frontend.yaml
[ec2-user@ip-172-31-19-87 frontend]$ kubectl apply -f cloudmart-frontend.yaml
deployment.apps/cloudmart-frontend-app created
service/cloudmart-frontend-app-service created
[ec2-user@ip-172-31-19-87 frontend]$
```

I monitored the status of objects being created and obtain the public IP generated for the API

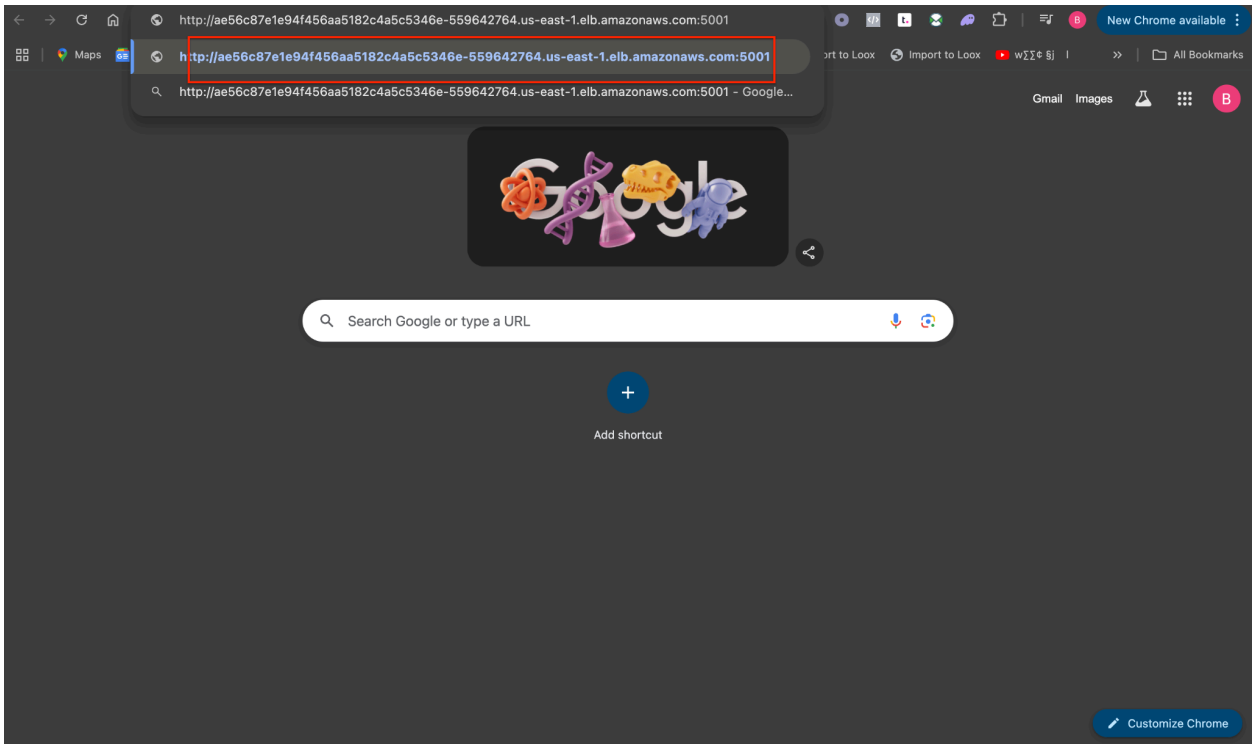
kubectl get pods
kubectl get deployment
kubectl get service

```
[ec2-user@ip-172-31-19-87 frontend]$ kubectl get pods
kubectl get deployment
kubectl get service
NAME                                      READY   STATUS    RESTARTS   AGE
cloudmart-backend-app-55db84697b-zr6h9    1/1     Running   0          40m
cloudmart-frontend-app-64cd9589fc-j7hzk   1/1     Running   0          2m
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
cloudmart-backend-app   1/1     1            1           40m
cloudmart-frontend-app  1/1     1            1           2m
NAME                                TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)          AGE
cloudmart-backend-app-service       LoadBalancer   10.100.240.29   a3ed30030eae945e797be912f746f78e-936372663.us-east-1.elb.amazonaws.com   5000:30485/TCP   40m
cloudmart-frontend-app-service      LoadBalancer   10.100.175.52   ae56c87e1e94f456aa5182c4a5c5346e-559642764.us-east-1.elb.amazonaws.com   5001:32019/TCP   2mls
kubernetes                          ClusterIP      10.100.0.1      <none>                                                                   443/TCP          5hll
m
[ec2-user@ip-172-31-19-87 frontend]$
```

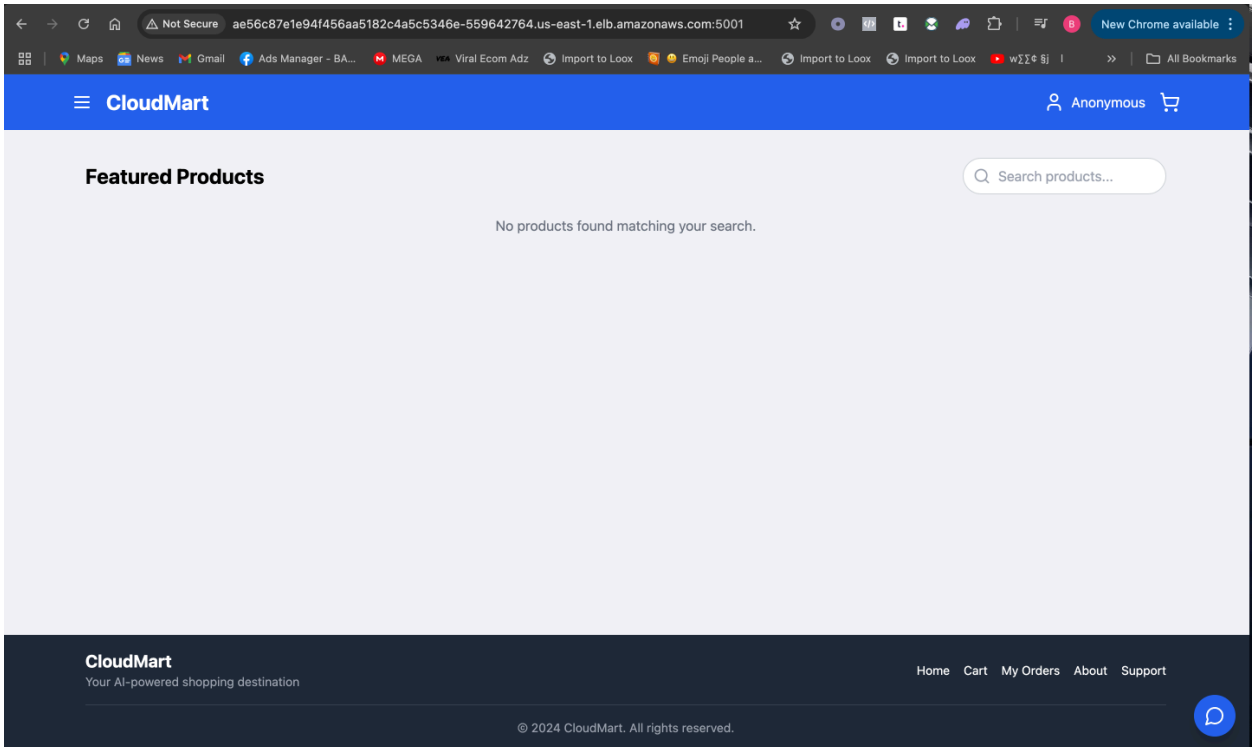
So now its time to check if everything is working the way its supposed to

I pasted the URL backend in a browser with the correct port which 5001

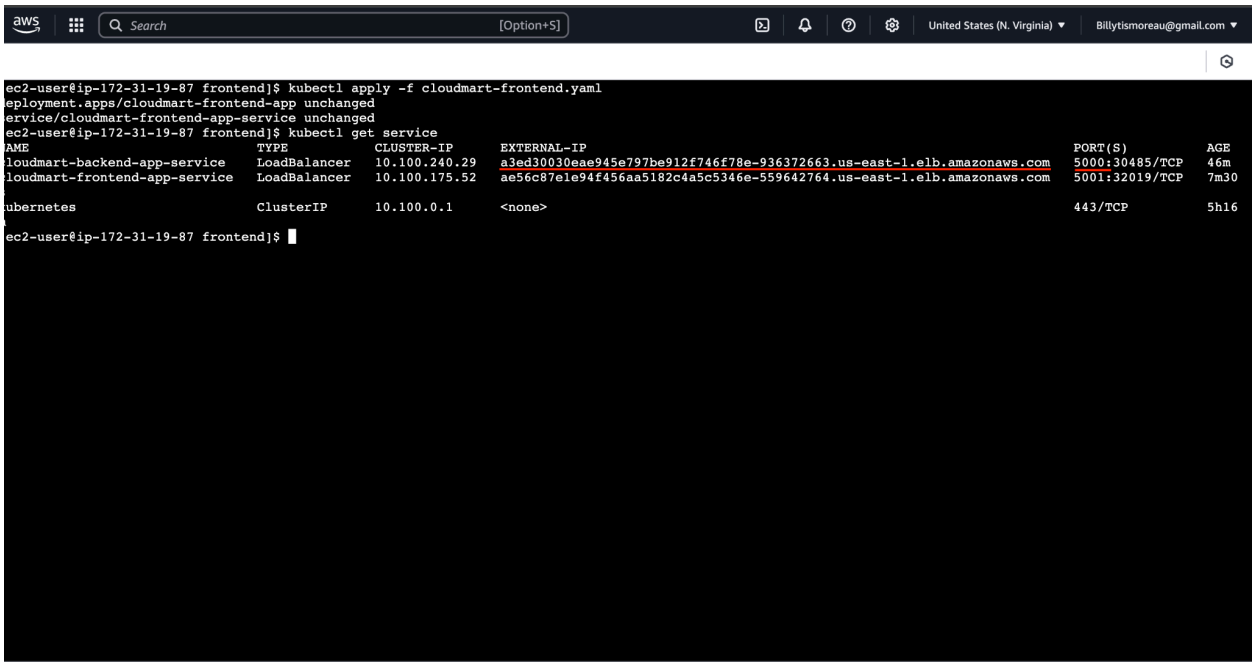
```
[ec2-user@ip-172-31-19-87 frontend]$ kubectl apply -f cloudmart-frontend.yaml
deployment.apps/cloudmart-frontend-app unchanged
service/cloudmart-frontend-app-service unchanged
[ec2-user@ip-172-31-19-87 frontend]$ kubectl get service
NAME                                TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)          AGE
cloudmart-backend-app-service       LoadBalancer   10.100.240.29   a3ed30030eae945e797be912f746f78e-936372663.us-east-1.elb.amazonaws.com   5000:30485/TCP   46m
cloudmart-frontend-app-service      LoadBalancer   10.100.175.52   ae56c87e1e94f456aa5182c4a5c5346e-559642764.us-east-1.elb.amazonaws.com   5001:32019/TCP   7m30
s
kubernetes                          ClusterIP      10.100.0.1      <none>                                                                   443/TCP          5h16
m
[ec2-user@ip-172-31-19-87 frontend]$
```



AND VOILA!!!

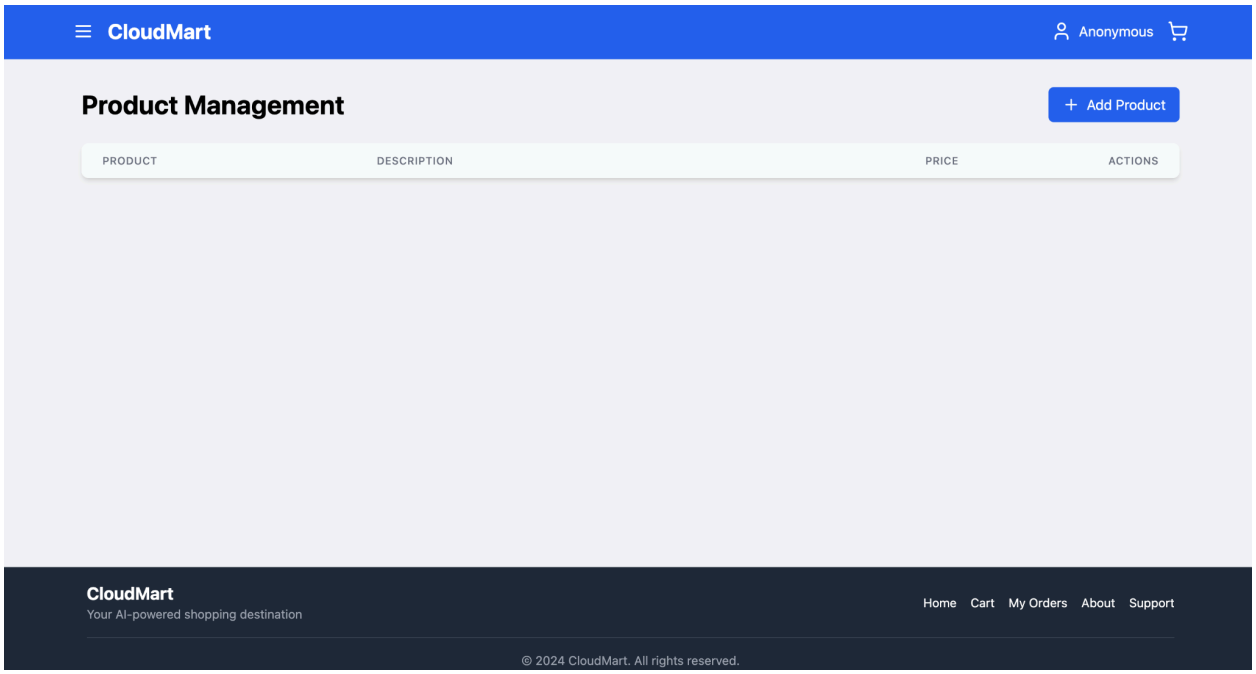


Let's check if the admin panel is working as well

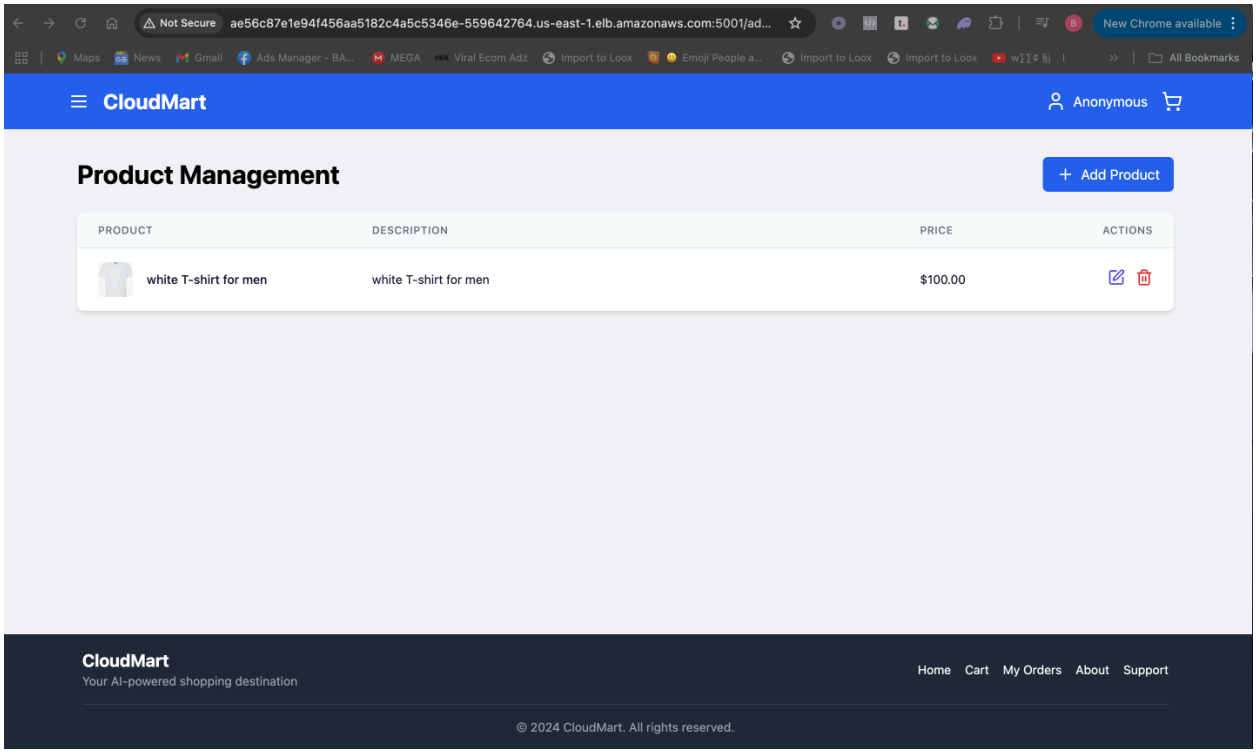


I typed this URL:
http://ae56c87e1e94f456aa5182c4a5c5346e-559642764.us-east-1.elb.amazonaws.com:5001/admin

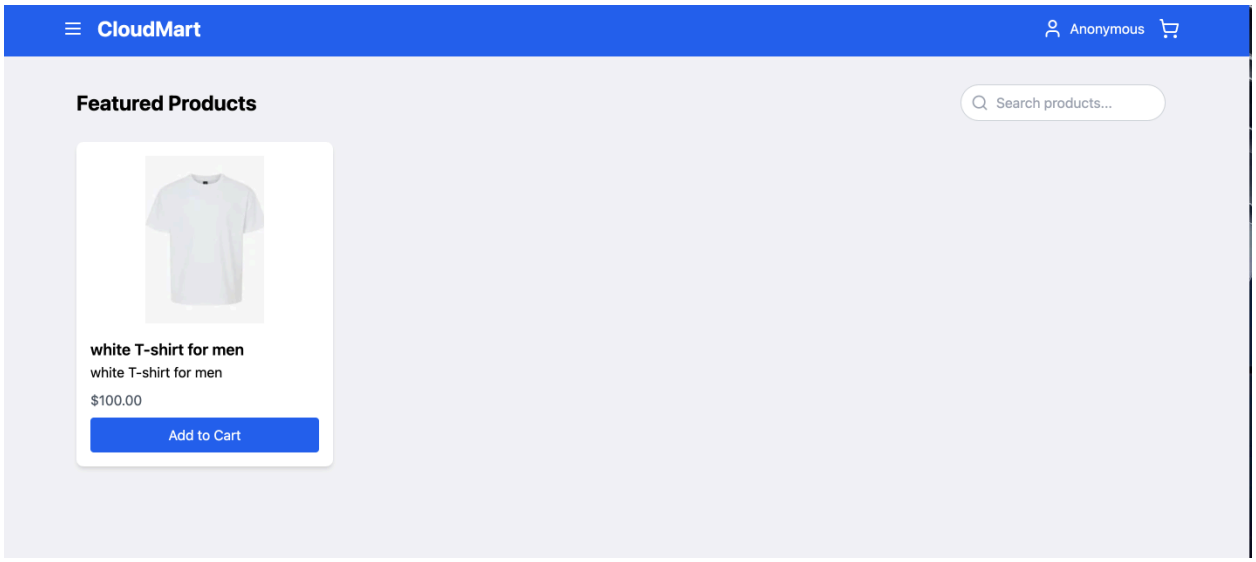
AND VOILA!!! its working



I even created a product to see if everything is working fine:



And here is the frontend



The application side is done (frontend and Backend) the architecture is almost complete: