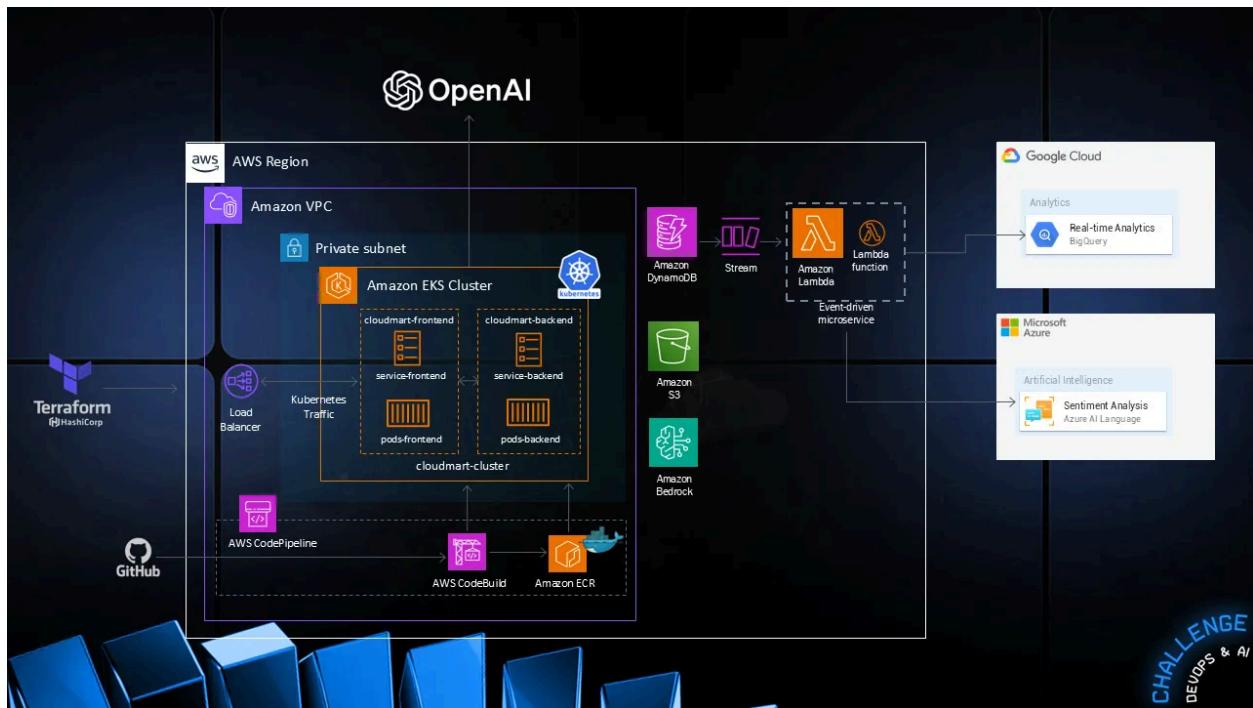


CLOUD MART PROJECT

(BUILDING AN E-COMMERCE WEBSITE)

HERE IS THE ARCHITECTURE OF OUR APPLICATION CLOUD MART:

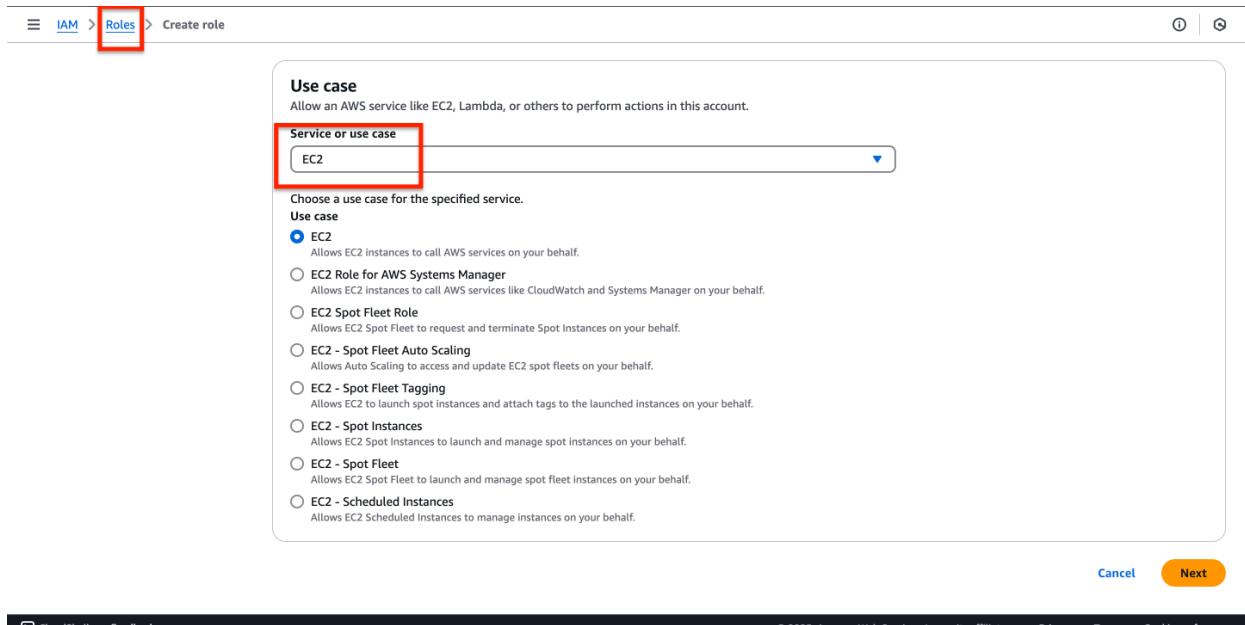


PHASE 1:

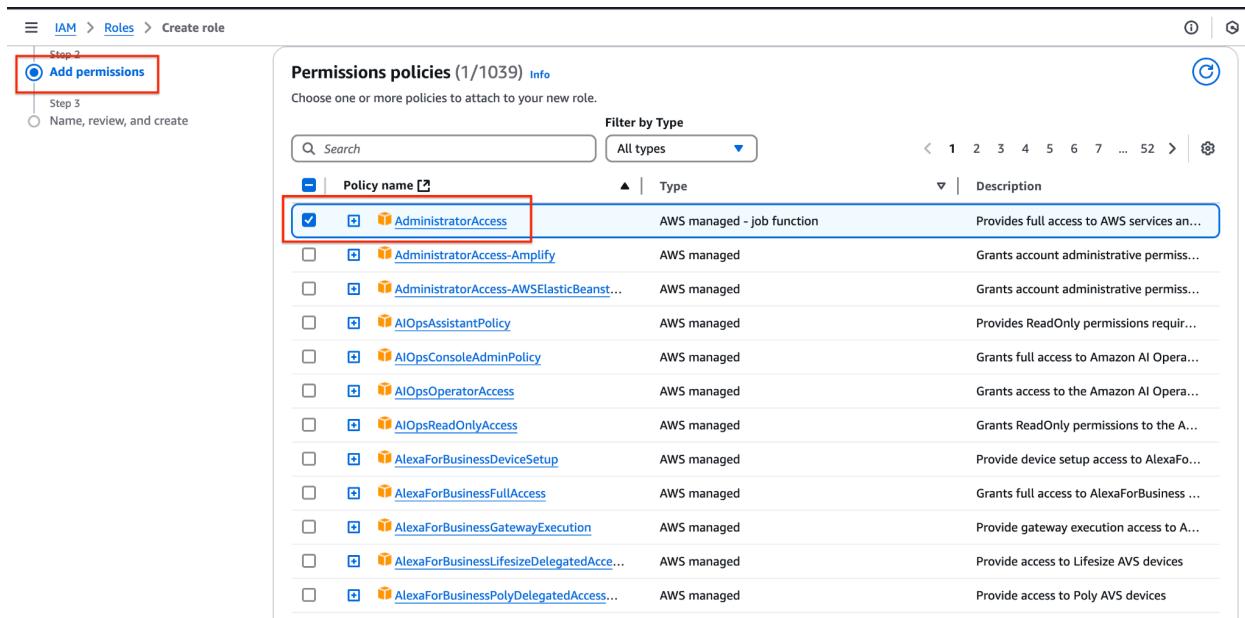
AUTOMATING AWS PROVISIONING WITH TERRAFORM USING CLAude

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 screenshot shows the AWS IAM 'Create role' wizard at Step 3: Name, review, and create. The left sidebar lists steps: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create). The main area is titled 'Name, review, and create' and contains 'Role details'. It includes fields for 'Role name' (set to 'EC2Admin'), 'Description' (set to 'Allows EC2 instances to call AWS services on your behalf.'), and a note about the maximum 1000 characters allowed. A red box highlights the 'Role name' field.

Step 1: Select trusted entities

Edit

Trust policy

```
1 [ {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Allow",  
6             "Action": [  
7                 "sts:AssumeRole"  
8             ],  
9         }  
10    ]  
11 } ]
```

The screenshot shows the AWS IAM Roles page. At the top, a green banner displays the message "Role EC2Admin created." with a "View role" button and a close "X" button. Below the banner, the title "Roles (7) Info" is shown, along with a search bar and buttons for "Delete" and "Create role". A descriptive text states: "An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust." The main table lists seven roles, including "EC2Admin" and "AWSServiceRoleForOrganizations". The columns are "Role name", "Trusted entities", and "Last activity".

The Role been created:

Step 2: Launch EC2 Instance

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

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

workstation

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux



macOS



Ubuntu



Windows



Red Hat



SUSE Linux



Debian



Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

3. Choose an Amazon Linux 2 AMI.
4. Select a t2.micro instance type.
5. Configure instance details:
 - o Network: Default VPC
 - o Subnet: Any available
 - o Auto-assign Public IP: Enable
 - o 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

(i) Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

[Add new volume](#)

(i) 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

(i) 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.

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](#)

EC2 > Instances > Launch an instance

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

Advanced details

Domain join directory | Info

Select | Create new directory

IAM instance profile | Info

EC2Admin | arn:aws:iam::082706928695:instance-profile/EC2Admin

Create new IAM profile

Hostname type | Info

IP name

DNS Hostname | Info

Enable IP name IPv4 (A record) DNS requests

Enable resource-based IPv4 (A record) DNS requests

Enable resource-based IPv6 (AAAA record) DNS requests

Instance auto-recovery | Info

Select

Shutdown behavior |

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

EC2 > Instances > Launch an instance

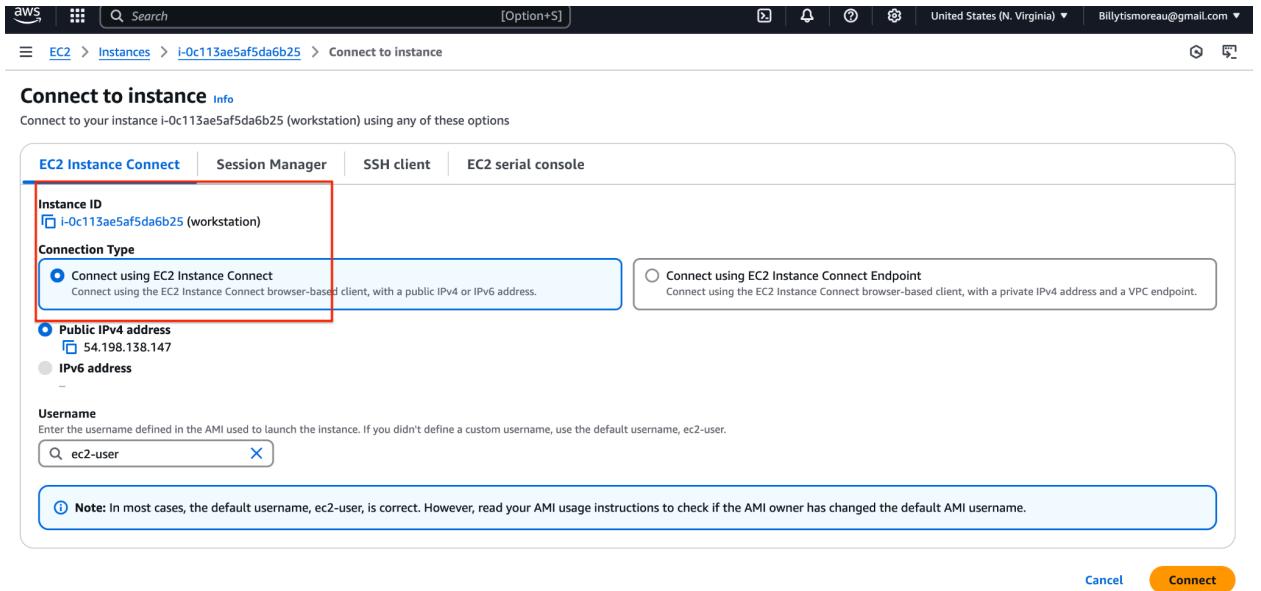
Success
Successfully initiated launch of instance (i-0c113ae5af5da6b25)

▶ Launch log

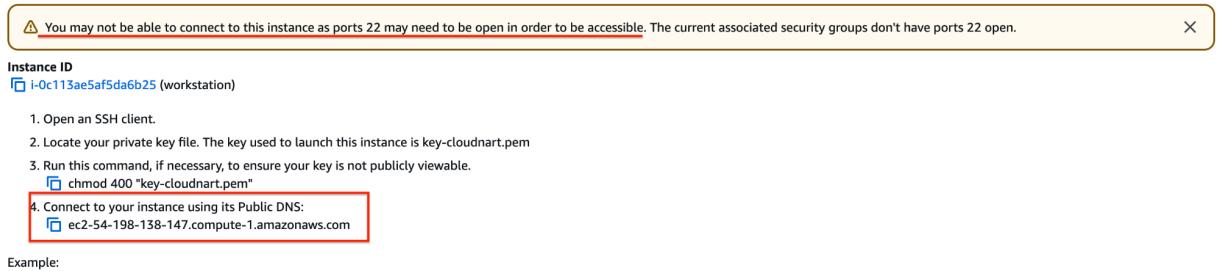
Step 3: Connect to EC2 Instance

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



EC2 > Security Groups > sg-0c8a9c42fabbef1b4 - launch-wizard-1

Inbound security group rules successfully modified on security group (sg-0c8a9c42fabbef1b4 | launch-wizard-1)

Details

sg-0c8a9c42fabbef1b4 - launch-wizard-1

Actions ▾

Details

Security group name launch-wizard-1	Security group ID sg-0c8a9c42fabbef1b4	Description launch-wizard-1 created 2025-03-06T07:37:44.576Z	VPC ID vpc-0886e0aa53db2bf82
Owner 082706928695	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

Inbound rules Outbound rules Sharing - new VPC associations - new Tags

Inbound rules (1)

<input type="checkbox"/> Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-0f954db95fd28ae1e	IPv4	SSH	TCP	22	

Manage tags Edit inbound rules

1 / 1

Finally we were connected to our EC2 instance

The screenshot shows a terminal window in AWS CloudShell. The prompt is '\$' followed by a multi-line ASCII art logo of a tree. Below the logo, the text 'Amazon Linux 2023' is displayed. A URL 'https://aws.amazon.com/linux/amazon-linux-2023' is shown with an arrow pointing to it. At the bottom of the terminal, the message 'Last login: Thu Mar 6 22:03:18 2025 from 18.206.107.27 [ec2-user@ip-172-31-19-87 ~]\$' is visible.

i-0c113ae5af5da6b25 (workstation)

Public IPs: 54.198.138.147 Private IPs: 172.31.19.87

Step 4: Install Terraform

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

```
sudo yum update -y
```

Install yum-utils:

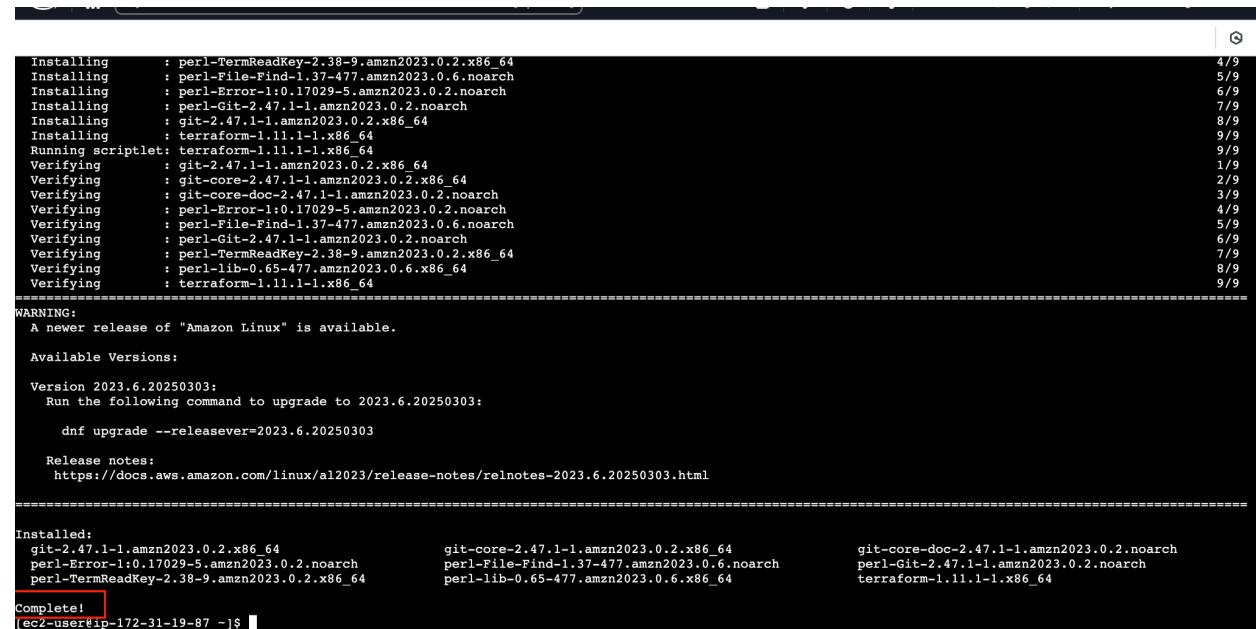
```
sudo yum install -y yum-utils
```

Add HashiCorp repository:

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

Install Terraform:

```
sudo yum -y install terraform
```



```
Installing      : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64          4/9
Installing      : perl-File-Find-1.37-477.amzn2023.0.6.noarch           5/9
Installing      : perl-Error-1:0.17029-5.amzn2023.0.2.noarch            6/9
Installing      : perl-Git-2.47.1-1.amzn2023.0.2.noarch             7/9
Installing      : git-2.47.1-1.amzn2023.0.2.x86_64                  8/9
Installing      : terraform-1.11.1-1.x86_64                      9/9
Running scriptlets: terraform-1.11.1-1.x86_64                   9/9
Verifying       : git-2.47.1-1.amzn2023.0.2.x86_64                  1/9
Verifying       : git-core-2.47.1-1.amzn2023.0.2.x86_64                2/9
Verifying       : git-core-doc-2.47.1-1.amzn2023.0.2.noarch            3/9
Verifying       : perl-Error-1:0.17029-5.amzn2023.0.2.noarch            4/9
Verifying       : perl-File-Find-1.37-477.amzn2023.0.6.noarch           5/9
Verifying       : perl-Git-2.47.1-1.amzn2023.0.2.noarch             6/9
Verifying       : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64          7/9
Verifying       : perl-lib-0.65-477.amzn2023.0.6.x86_64                 8/9
Verifying       : terraform-1.11.1-1.x86_64                      9/9
=====
WARNING:
A newer release of "Amazon Linux" is available.

Available Versions:

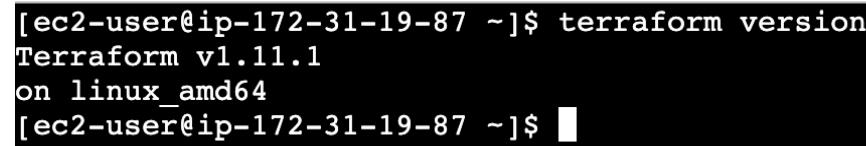
Version 2023.6.20250303:
Run the following command to upgrade to 2023.6.20250303:
dnf upgrade --releasever=2023.6.20250303

Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.6.20250303.html
=====

Installed:
git-2.47.1-1.amzn2023.0.2.x86_64          git-core-2.47.1-1.amzn2023.0.2.x86_64          git-core-doc-2.47.1-1.amzn2023.0.2.noarch
perl-Error-1:0.17029-5.amzn2023.0.2.noarch    perl-File-Find-1.37-477.amzn2023.0.6.noarch    perl-Git-2.47.1-1.amzn2023.0.2.noarch
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64    perl-lib-0.65-477.amzn2023.0.6.x86_64        terraform-1.11.1-1.x86_64
=====
Complete!
[ec2-user@ip-172-31-19-87 ~]$
```

Verify installation:

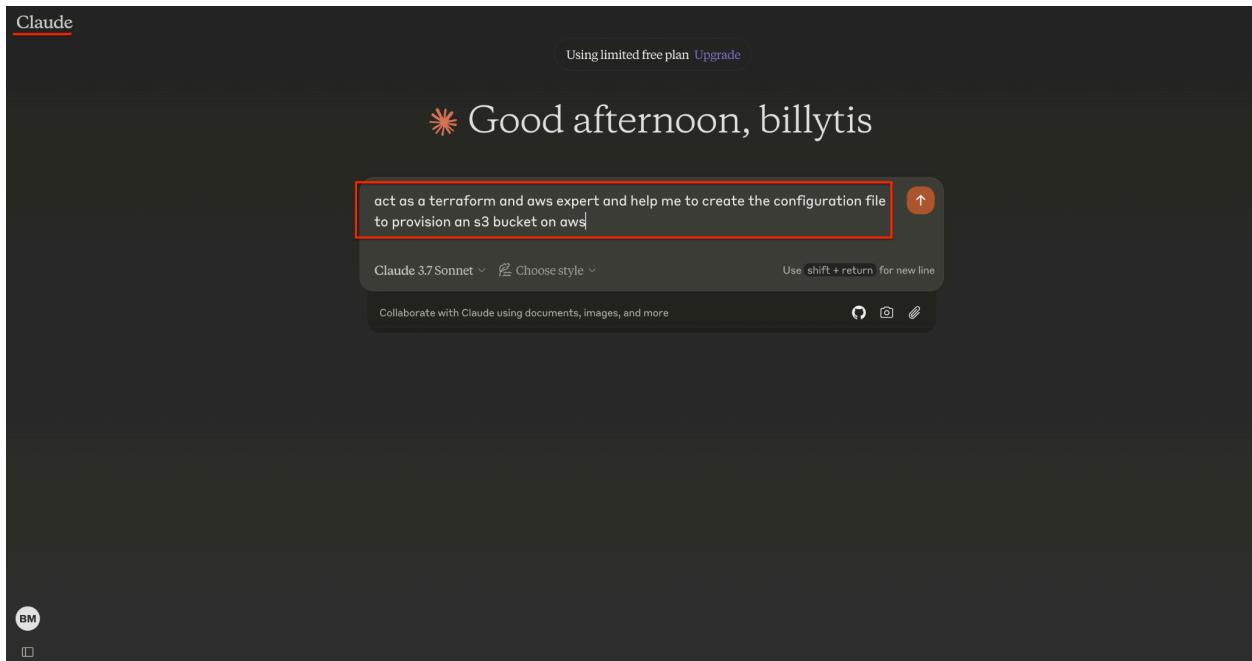
```
terraform version
```



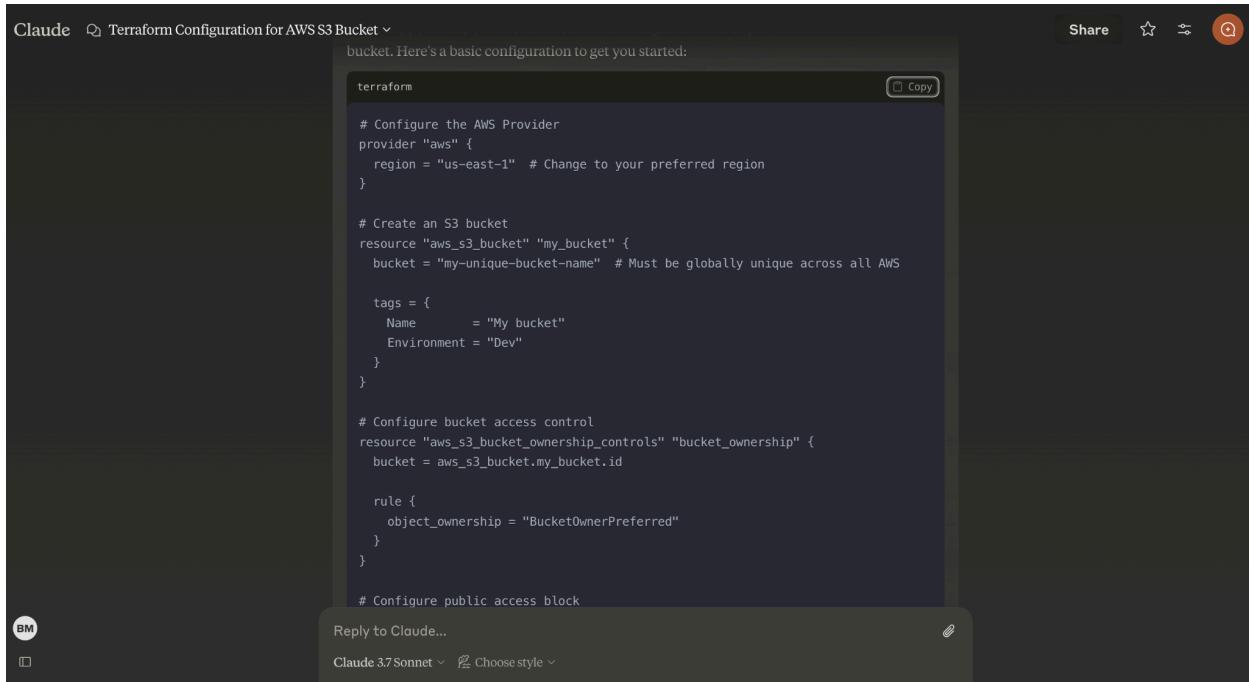
```
[ec2-user@ip-172-31-19-87 ~]$ terraform version
Terraform v1.11.1
on linux_amd64
[ec2-user@ip-172-31-19-87 ~]$
```

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:



Claude Q Terraform Configuration for AWS S3 Bucket Share ⚡ ⚡ ⚡

bucket. Here's a basic configuration to get you started:

```
terraform
# Configure the AWS Provider
provider "aws" {
    region = "us-east-1" # Change to your preferred region
}

# Create an S3 bucket
resource "aws_s3_bucket" "my_bucket" {
    bucket = "my-unique-bucket-name" # Must be globally unique across all AWS

    tags = {
        Name      = "My bucket"
        Environment = "Dev"
    }
}

# Configure bucket access control
resource "aws_s3_bucketOwnershipControls" "bucket_ownership" {
    bucket = aws_s3_bucket.my_bucket.id

    rule {
        object_ownership = "BucketOwnerPreferred"
    }
}

# Configure public access block
```

Reply to Claude... Edit

Claude 3.7 Sonnet Choose style

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.

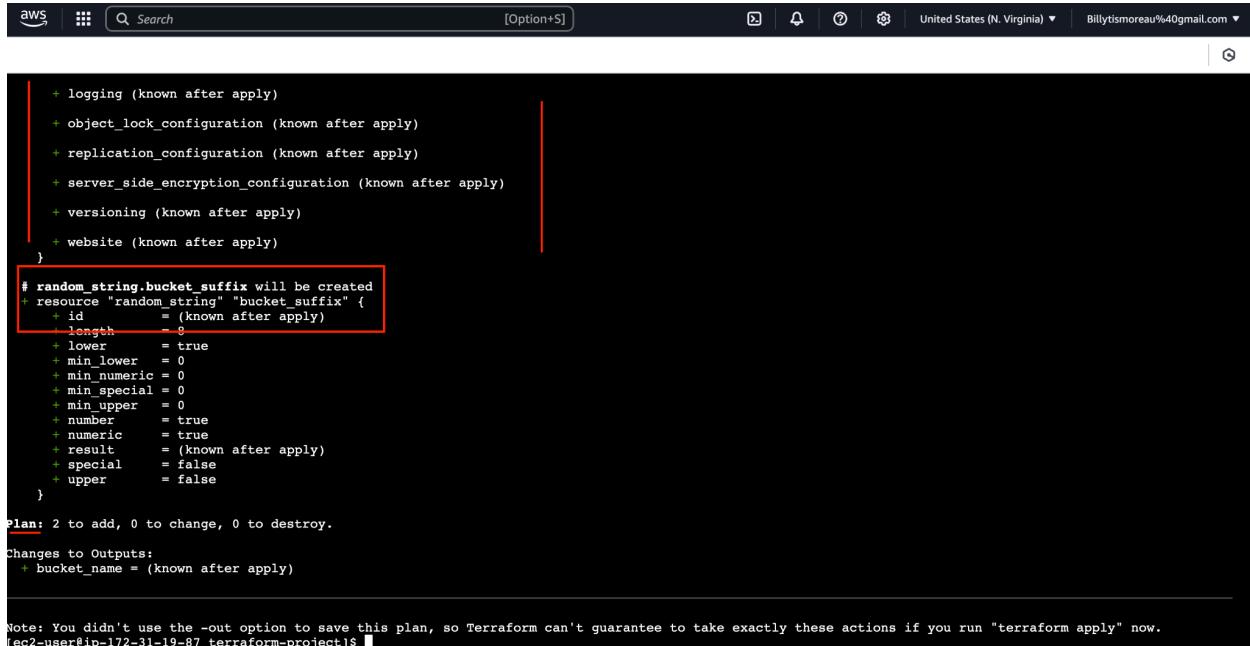
Save and exit the editor (in nano, press Ctrl+X, then Y, then Enter).

Initialize Terraform:

```
terraform init
```

Review the plan:

```
terraform plan
```



```
+ logging (known after apply)
+ object_lock_configuration (known after apply)
+ replication_configuration (known after apply)
+ server_side_encryption_configuration (known after apply)
+ versioning (known after apply)
+ website (known after apply)
}

# 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.

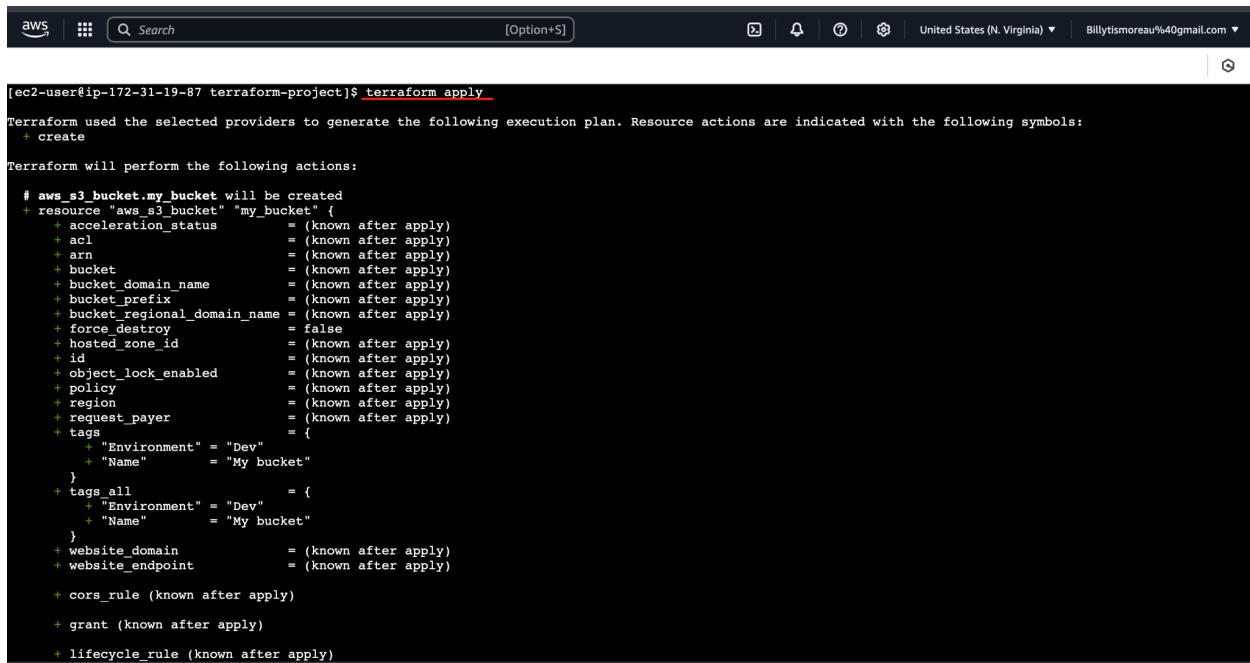
Changes to Outputs:
+ bucket_name = (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
```

[ec2-user@ip-172-31-19-87 terraform-project]\$

Apply the configuration:

Terraform apply



```
[ec2-user@ip-172-31-19-87 terraform-project]$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_s3_bucket.my_bucket will be created
resource "aws_s3_bucket" "my_bucket" {
+ acceleration_status      = (known after apply)
+ acl                      = (known after apply)
+ arn                      = (known after apply)
+ bucket                   = (known after apply)
+ bucket_domain_name       = (known after apply)
+ bucket_prefix             = (known after apply)
+ bucketRegionalDomainName = (known after apply)
+ force_destroy            = false
+ hosted_zone_id           = (known after apply)
+ id                       = (known after apply)
+ object_lock_enabled       = (known after apply)
+ policy                   = (known after apply)
+ region                   = (known after apply)
+ request_payer             = (known after apply)
+ tags                     = {
+   "Environment" = "Dev"
+   "Name"        = "My bucket"
+ }
+ tags_all                 = {
+   "Environment" = "Dev"
+   "Name"        = "My bucket"
+ }
+ website_domain           = (known after apply)
+ website_endpoint          = (known after apply)
+ cors_rule                (known after apply)
+ grant                    (known after apply)
+ lifecycle_rule            (known after 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"
[redacted] terraform project]$
```

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with navigation links like 'Amazon S3', 'General purpose buckets', 'Directory buckets', etc. The main area displays an 'Account snapshot' with an 'updated every 24 hours' message. Below it, under 'General purpose buckets', there are two entries:

Name	AWS Region	IAM Access Analyzer	Creation date
my-bucket-a4j2vnse	US East (N. Virginia) us-east-1	View analyzer for us-east-1	March 6, 2025, 15:43:07 (UTC-08:00)
mypollybucket-1986	US East (N. Virginia) us-east-1	View analyzer for us-east-1	March 4, 2025, 13:18:58 (UTC-08:00)

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.

The screenshot shows the AWS DynamoDB console. The left sidebar has a 'Tables' section selected. The main area displays a table titled 'Tables (3) info' with three rows:

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Recent
cloudmart-orders	Active	id (\$)	-	0	0	Off	☆	On-req
cloudmart-products	Active	id (\$)	-	0	0	Off	☆	On-req
cloudmart-tickets	Active	id (\$)	-	0	0	Off	☆	On-req

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
```

```
Installed:
  containerd-1.7.25-1.amzn2023.0.1.x86_64           docker-25.0.8-1.amzn2023.0.1.x86_64           iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64         libcgROUP-3.0-1.amzn2023.0.1.x86_64          libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  libnftnl-link-1.0.1-19.amzn2023.0.2.x86_64       libnftnl-1.2.2-2.amzn2023.0.2.x86_64          pigz-2.5-1.amzn2023.0.3.x86_64
  runc-1.2.4-1.amzn2023.0.1.x86_64

Complete!
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344bla5: Pull complete
Digest: sha256:bfb0cc14f13f9ed1ae86abc2b9f11181dc50d779807ed3a3c5e55a6936dbdd5
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Docker version 25.0.8, build 0bab007
[ec2-user@ip-172-31-19-87 ~]$
```

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.zip
```

```
inflating: __MACOSX/src/services/._orderService.js
inflating: package.json
inflating: __MACOSX/.package.json
[ec2-user@ip-172-31-19-87 backend]$ ls
__MACOSX cloudmart-backend.zip package.json src
[ec2-user@ip-172-31-19-87 backend]$
```

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"]
```

```
inflating: _MACOSX/.vite.config.js
[ec2-user@ip-172-31-19-87 frontend]$ ls
_MACOSX      eslint.config.js  package-lock.json  postcss.config.js  src          vite.config.js
cloudmart-frontend.zip  index.html    package.json     public        tailwind.config.js
[ec2-user@ip-172-31-19-87 frontend]$ nano dockerfile
[ec2-user@ip-172-31-19-87 frontend]$
```

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

The screenshot shows the AWS IAM Users page. At the top, there is a header with the title 'Users (1)' and a 'Info' link. Below the header, a message states: 'An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.' There is a search bar labeled 'Search'. To the right of the search bar are navigation icons: a left arrow, a right arrow, and a refresh icon. Below the search bar is a table with the following columns: 'User name', 'Path', 'Group', 'Last activity', 'MFA', 'Password age', and 'Console last sign-in'. The table contains one row for the user 'eksuser', which is highlighted in blue. The 'eksuser' row has a checkbox in the first column, a path of '/', a group of '0', and other details like 'Last activity' and 'Console last sign-in'.

I used aws configure to make sure that the user “`eksuser`” is actually configured on our ec2 workstation virtual machine

```
[ec2-user@ip-172-31-19-87 frontend]$ aws configure
AWS Access Key ID [None]: A[REDACTED]+
AWS Secret Access Key [None]: [REDACTED]
Default region name [None]: us-east-1
Default output format [None]:
[ec2-user@ip-172-31-19-87 frontend]$
```

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 [i] 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] 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 18:24:08 [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 [*] no tasks
2025-03-08 18:26:48 [*] all EKS cluster resources for "cloudmart" have been created
2025-03-08 18:26:48 [*] nodegroup "standard-workers" has 1 node(s)
2025-03-08 18:26:48 [*] node "ip-192-168-24-198.ec2.internal" is ready
2025-03-08 18:26:48 [*] waiting for at least 1 node(s) to become ready in "standard-workers"
2025-03-08 18:26:48 [*] nodegroup "standard-workers" has 1 node(s)
2025-03-08 18:26:48 [*] 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 [*] 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 [i] 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 18:51:04 [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

The screenshot shows the AWS ECR Public Registry interface. On the left, there's a sidebar with navigation links for Amazon Elastic Container Registry, Private registry, Public registry, ECR public gallery, Amazon ECS, Amazon EKS, Getting started, and Documentation. The main area has a search bar, a 'Created public repository' notification (cloudmart-backend has been successfully created in public registry), and a table titled 'Public repositories (1)'. The table includes columns for Repository name, URI, and Created at. A single row is shown for 'cloudmart-backend' with the URI 'public.ecr.aws/j8h5o7q1/cloudmart-backend' and the creation date 'March 08, 2025, 14:06:29 (UTC-08)'. The 'cloudmart-backend' link in the Repository name column is highlighted with a red box.

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

Screenshot of the AWS Amazon Elastic Container Registry (ECR) Public Registry interface. The left sidebar shows navigation for 'Amazon Elastic Container Registry' and 'Public registry'. The main area displays 'Public repositories (1)' with a single entry: 'cloudmart-backend' (URI: public.ecr.aws/j8h5o7q1/cloudmart-backend, Created at: March 08, 2025, 14:06:29 (UTC-08)).

Screenshot of the AWS Amazon Elastic Container Registry (ECR) Public Registry interface, showing the details for the 'cloudmart-backend' repository. The left sidebar shows navigation for 'Amazon Elastic Container Registry' and 'Public registry'. The main area displays 'Images (0)' with a message: 'Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag.' A red box highlights the 'View push commands' button.

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

```
ec2-user@ip-172-31-19-87 backend]$ docker build -t cloudmart-backend .
[+] Building 53.1s (10/10) FINISHED
--> [internal] load build definition from dockerfile
--> [internal] load metadata for docker.io/library/node:18
--> [internal] load .dockerignore
--> [internal] transfer context: 2B
--> [1/5] FROM docker.io/library/node:18@sha256:ba756f198b4b1e0114b53b23121c8ae27f7ae4d5d95ca4a0554b0649cc9c7dcf
--> [internal] resolve docker.io/library/node:18@sha256:ba756f198b4b1e0114b53b23121c8ae27f7ae4d5d95ca4a0554b0649cc9c7dcf
--> sha256:ba756f198b4b1e0114b53b23121c8ae27f7ae4d5d95ca4a0554b0649cc9c7dcf 21.7s
--> sha256:515ad54a8b2812a0ec58ff82c06f0f0dddb337a226b1879f09e15f67b69fc 0.0s
--> sha256:515ad54a8b2812a0ec58ff82c06f0f0dddb337a226b1879f09e15f67b69fc 0.0s
--> sha256:1281e50d2e435595c266df06531a7e8c2eb00195622c8ab2eed8d760e6576b 0.0s
--> sha256:4043794e81923d1f62d46c8fa0c23701def3e07b615f9de1a0c43ce174f107fb 2.49KB 0.0s
--> sha256:515ad54a8b2812a0ec58ff82c06f0f0dddb337a226b1879f09e15f67b69fc 1.7s
--> sha256:1281e50d2e435595c266df06531a7e8c2eb00195622c8ab2eed8d760e6576b 64.39MB 0.0s
--> sha256:4043794e81923d1f62d46c8fa0c23701def3e07b615f9de1a0c43ce174f107fb 2.49KB
--> sha256:515ad54a8b2812a0ec58ff82c06f0f0dddb337a226b1879f09e15f67b69fc 6.39KB 0.0s
--> sha256:8031108f3da87bb32f090262d0109c8ad0d599168050967becefaf5d02e9a681b 24.06MB 0.0s
--> sha256:447713e77b4fc3658cfba0c1e816b70ff6d9bf06563d8cfcfb0459406aed33b4 211.34MB 4.68
--> sha256:447713e77b4fc3658cfba0c1e816b70ff6d9bf06563d8cfcfb0459406aed33b4 211.34MB
--> extracting sha256:515ad54a8b2812a0ec58ff82c06f0f0dddb337a226b1879f09e15f67b69fc 3.98
--> sha256:447713e77b4fc3658cfba0c1e816b70ff6d9bf06563d8cfcfb0459406aed33b4 3.98
--> sha256:396e9c5702ad97405a485ee635a41c2c4f8bde46e611f0af68 1.25MB 2.5s
--> sha256:396e9c5702ad97405a485ee635a41c2c4f8bde46e611f0af68 1.25MB
--> sha256:c5702ad97405a485ee635a41c2c4f8bde46e611f0af68 45.70MB 3.4s
--> sha256:c5702ad97405a485ee635a41c2c4f8bde46e611f0af68 45.70MB
--> sha256:8728cb9d9a1bb84c95cbaa0eae129482b9428acce138dcbab3176854f8d9 447B 2.6s
--> sha256:8728cb9d9a1bb84c95cbaa0eae129482b9428acce138dcbab3176854f8d9 447B
--> extracting sha256:8031108f3da87bb32f090262d0109c8ad0d599168050967becefaf5d02e9a681b 0.8s
--> extracting sha256:1d281e50d4e35595c266df06531a7e8c2eb00185622c8ab2eed8d760e6576b 3.2s
--> extracting sha256:1d281e50d4e35595c266df06531a7e8c2eb00185622c8ab2eed8d760e6576b 3.2s
--> extracting sha256:447713e77b4fc3658cfba0c1e816b70ff6d9bf06563d8cfcfb0459406aed33b4 8.7s
--> extracting sha256:f87facc2c491970afc16e1d15bc7a2be960b00caa9467613ba0f87d4e7bdf 0.0s
--> extracting sha256:f87facc2c491970afc16e1d15bc7a2be960b00caa9467613ba0f87d4e7bdf 0.0s
--> extracting sha256:396e9c5702ad97405a485ee635a41c2c4f8bde46e611f0af68 2.5s
--> extracting sha256:396e9c5702ad97405a485ee635a41c2c4f8bde46e611f0af68 0.1s
--> extracting sha256:8728cb9d9a1bb84c95cbaa0eae129482b9428acce138dcbab3176854f8d9 0.0s
--> [internal] load build context
--> transferring context: 49.88kB
--> [2/5] WORKDIR /usr/src/app
--> [3/5] COPY package*.json .
--> [4/5] RUN npm install
--> [5/5] COPY .
--> exporting to image
--> exporting layers
--> writing image sha256:82a14c3e85e64d4c6c90c8f26bc58b719fb8c8eae47e13be541e84a92dc74ceb
--> naming to docker.io/library/cloudmart-backend
ec2-user@ip-172-31-19-87 backend]$
```

The docker push command has been launch:

```
>> => sha256:447713e77b4fc3658cfbaba0c1e816b70ff6d9bf06563dc8cfc0459406aed33b4 211.34MB / 211.34MB
=> => extracting sha256:155ad54a852812a0ec559ff82c06f0f0dddb337a226b11879f09e15f67b69fc 3.98
=> => sha256:f87facc2c91970afc16ec1d15bc7a2be960b0caaa9467613ba0f87d4e7bdf 3.32kB 3.98
=> => sha256:396e9c5702ad97405a8485ee635a41c2c4fdbb636cec4df8bde466e11f0af68 1.25MB / 1.25MB 2.48
=> => sha256:2f9475d0583b480d241fbcc3e81dccb2c8328edcbe28553bdcf241b4ae3c3edc 45.70MB / 45.70MB 2.58
=> => sha256:c8728cb69dcelbba4cb95cbaa0e0ae129482b9428acce138dcbab3176854f8d9 447B / 447B 2.68
=> => extracting sha256:8031108f3cda87b7bb2f090262d0109c8a0db99168050967becefafad502e9a681b 0.08
=> => extracting sha256:d281e50d3e435595c266df06531a7e8c2eb0c185622c8ab2eed8d760e6576b 3.28
=> => extracting sha256:447713e77b4fc3658cfbaba0c1e816b70ff6d9bf06563dc8cfc0459406aed33b4 8.78
=> => extracting sha256:447713e77b4fc3658cfbaba0c1e816b70ff6d9bf06563dc8cfc0459406aed33b4 0.08
=> => extracting sha256:2f9475d0583b480d241fbcc3e81dccb2c8328edcbe28553bdcf241b4ae3c3edc 2.58
=> => extracting sha256:396e9c5702ad97405a8485ee635a41c2c4fdbb6636cec4df8bde466e11f0af68 0.18
=> => extracting sha256:c8728cb69dcelbba4cb95cbaa0e0ae129482b9428acce138dcbab3176854f8d9 0.08
=> [internal] load build context 0.08
=> => transferring context: 49.88kB 0.08
[2/5] WORKDIR /usr/src/app 0.18
[3/5] COPY package*.json . 0.18
[4/5] RUN npm install 0.08
[5/5] COPY . 9.48
=> => exporting to image 9.48
=> => exporting layers 0.08
=> => writing image sha256:82a14c3e85e64d4c6c90c8f26bc58b719fb8c8eae47e13be541e84a92dc74ceb 0.08
=> => naming to docker.io/library/cloudmart-backend 0.08
[ec2-user@ip-172-31-19-87 backend]$ docker tag cloudmart-backend:latest public.ecr.aws/j8h5o7q1/cloudmart-backend:latest
[ec2-user@ip-172-31-19-87 backend]$ docker push public.ecr.aws/j8h5o7q1/cloudmart-backend:latest
The push refers to repository [public.ecr.aws/j8h5o7q1/cloudmart-backend]
8857ac116e31: Pushed
8cac98dd4db54: Pushing [=====>] 202MB/204.3MB
f98109d3b5f8: Pushed 21.2s
cf571127328b: Pushed 0.18
e129ba4f1574: Pushed 0.08
f270dee6385: Pushed 0.08
780658909072: Pushing [=====>] 157MB 0.08
aea0a4b6ca2: Pushed 0.08
4b017a36fd9c: Pushing [=====>] 197.3MB/587.6MB 0.08
20a9b386e10e: Pushing [=====>] 169.5MB/177.1MB 0.08
f8217d7865d2: Pushed 0.08
d1c9aza5f237: Pushing [=====>] 114MB/116.5MB 0.08
```

The push is done

```
>> => extracting sha256:155ad54a852812a0ec559ff82c06f0f0dddb337a226b11879f09e15f67b69fc 3.98
=> => sha256:f87facc2c91970afc16ec1d15bc7a2be960b0caaa9467613ba0f87d4e7bdf 3.32kB 2.48
=> => sha256:396e9c5702ad97405a8485ee635a41c2c4fdbb636cec4df8bde466e11f0af68 1.25MB / 1.25MB 2.58
=> => sha256:2f9475d0583b480d241fbcc3e81dccb2c8328edcbe28553bdcf241b4ae3c3edc 45.70MB / 45.70MB 3.48
=> => sha256:c8728cb69dcelbba4cb95cbaa0e0ae129482b9428acce138dcbab3176854f8d9 447B / 447B 2.68
=> => extracting sha256:8031108f3cda87b7bb2f090262d0109c8a0db99168050967becefafad502e9a681b 0.08
=> => extracting sha256:d281e50d3e435595c266df06531a7e8c2eb0c185622c8ab2eed8d760e6576b 3.28
=> => extracting sha256:447713e77b4fc3658cfbaba0c1e816b70ff6d9bf06563dc8cfc0459406aed33b4 8.78
=> => extracting sha256:447713e77b4fc3658cfbaba0c1e816b70ff6d9bf06563dc8cfc0459406aed33b4 0.08
=> => extracting sha256:2f9475d0583b480d241fbcc3e81dccb2c8328edcbe28553bdcf241b4ae3c3edc 2.58
=> => extracting sha256:396e9c5702ad97405a8485ee635a41c2c4fdbb6636cec4df8bde466e11f0af68 0.18
=> => extracting sha256:c8728cb69dcelbba4cb95cbaa0e0ae129482b9428acce138dcbab3176854f8d9 0.08
=> [internal] load build context 0.08
=> => transferring context: 49.88kB 0.08
[2/5] WORKDIR /usr/src/app 0.18
[3/5] COPY package*.json . 0.18
[4/5] RUN npm install 0.08
[5/5] COPY . 9.48
=> => exporting to image 9.48
=> => exporting layers 0.08
=> => writing image sha256:82a14c3e85e64d4c6c90c8f26bc58b719fb8c8eae47e13be541e84a92dc74ceb 0.08
=> => naming to docker.io/library/cloudmart-backend 0.08
[ec2-user@ip-172-31-19-87 backend]$ docker tag cloudmart-backend:latest public.ecr.aws/j8h5o7q1/cloudmart-backend:latest
[ec2-user@ip-172-31-19-87 backend]$ docker push public.ecr.aws/j8h5o7q1/cloudmart-backend:latest
The push refers to repository [public.ecr.aws/j8h5o7q1/cloudmart-backend]
8857ac116e31: Pushed
8cac98dd4db54: Pushed 21.2s
f98109d3b5f8: Pushed 0.18
cf571127328b: Pushed 0.08
e129ba4f1574: Pushed 0.08
f270dee6385: Pushed 0.08
780658909072: Pushed 0.08
aea0a4b6ca2: Pushed 0.08
4b017a36fd9c: Pushed 9.48
20a9b386e10e: Pushed 9.48
f8217d7865d2: Pushed 0.08
d1c9aza5f237: Pushed 0.08
latest: digest: sha256:5050664314ee2b52d7a42dd893041f59a06a13533f54bbf714bb4764a011fc91 size: 2839 0.08
[ec2-user@ip-172-31-19-87 backend]$
```

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

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

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

```
[ec2-user@ip-172-31-19-87 backend]$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
cloudmart-backend-app-55db84697b-zr6h9  1/1     Running   0          4m36s
[ec2-user@ip-172-31-19-87 backend]$ kubectl get deployment
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
cloudmart-backend-app        1/1     1           1           5m
[ec2-user@ip-172-31-19-87 backend]$ kubectl get service
NAME            TYPE      CLUSTER-IP   EXTERNAL-IP
cloudmart-backend-app-service LoadBalancer   10.100.240.29   a3ed30030eae945e797be912f746f78e-936372663.us-east-1.elb.amazonaws.com
kubernetes       ClusterIP   10.100.0.1    <none>
[ec2-user@ip-172-31-19-87 backend]$
```

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

Private repositories (1)		 View push commands	Delete	Actions ▾	Create repository
Repository name	URI	Created at	Tag immutability	Encryption type	
cloudmart-frontend	 082706928695.dkr.ecr.us-east-1.amazonaws.com/cloudmart-frontend	March 08, 2025, 15:15:03 (UTC-08)	Mutable	AES-256	

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

Push commands for cludmart-frontend

X

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 get-login-password --region us-east-1 | docker login --username AWS --password-stdin 082706928695.dkr.ecr.us-east-1.amazonaws.com
```

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 cludmart-frontend .
```

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

```
 docker tag cludmart-frontend:latest 082706928695.dkr.ecr.us-east-1.amazonaws.com/cludmart-frontend:latest
```

 **Code copied**

4. Command to push this image to your newly created AWS repository:

```
 docker push 082706928695.dkr.ecr.us-east-1.amazonaws.com/cludmart-frontend:latest
```

[Close](#)

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

nano cludmart-frontend.yaml

And i pasted the code:

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

```

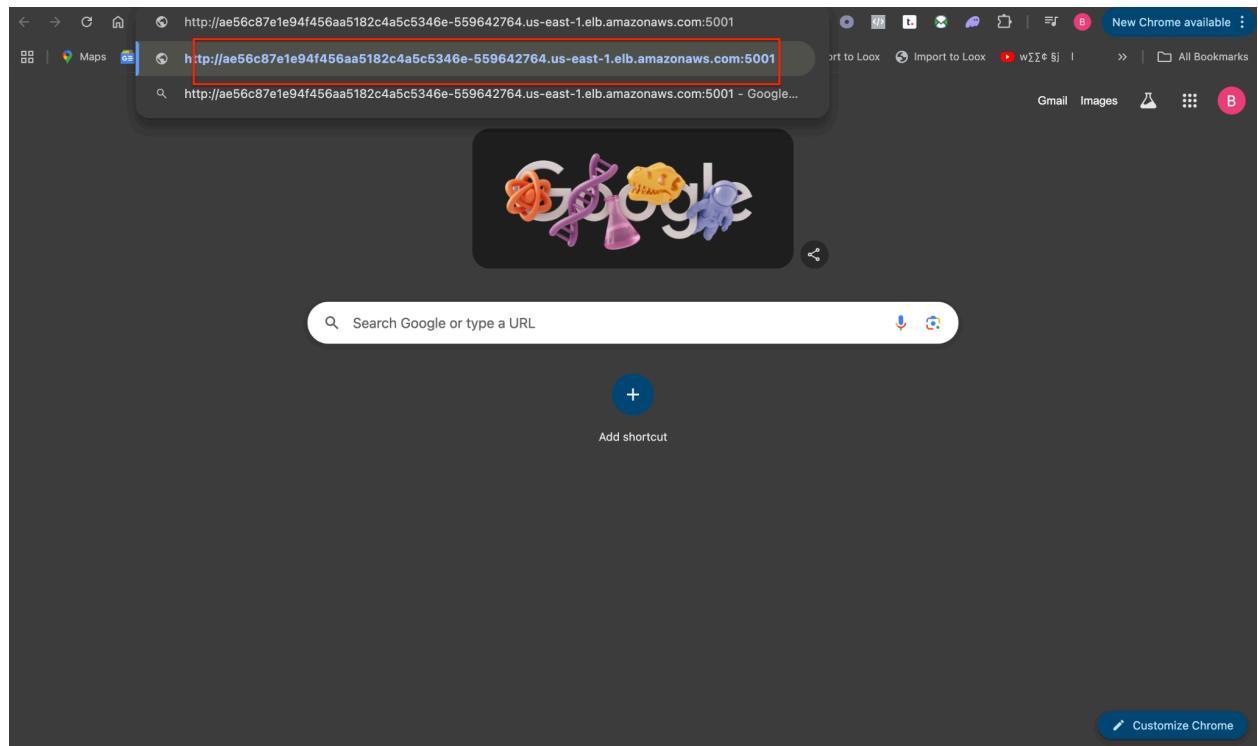
kubectl get pods
kubectl get deployment
kubectl get service
```

```
[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-service   LoadBalancer   10.100.240.29   a3ed30030eae945e797be912f746f78e-936372663.us-east-1.elb.amazonaws.com   5000:30485/TCP   40m
cloudmart-frontend-service   LoadBalancer   10.100.175.52   ae56c87e1e94f456aa5182c4a5c5346e-559642764.us-east-1.elb.amazonaws.com   5001:32019/TCP   2m1s
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   5h11m
[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-service   LoadBalancer   10.100.240.29   a3ed30030eae945e797be912f746f78e-936372663.us-east-1.elb.amazonaws.com   5000:30485/TCP   46m
cloudmart-frontend-service   LoadBalancer   10.100.175.52   ae56c87e1e94f456aa5182c4a5c5346e-559642764.us-east-1.elb.amazonaws.com   5001:32019/TCP   7m30
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   5h16m
[ec2-user@ip-172-31-19-87 frontend]$
```



AND VOILA!!!

The screenshot shows a web browser window with the URL `ae56c87e1e94f456aa5182c4a5c5346e-559642764.us-east-1.elb.amazonaws.com:5001`. The page title is "CloudMart". The main content area is titled "Featured Products" and contains the message "No products found matching your search.". At the bottom, there is a dark footer bar with the CloudMart logo, the tagline "Your AI-powered shopping destination", navigation links for Home, Cart, My Orders, About, and Support, and a copyright notice: "© 2024 CloudMart. All rights reserved.".

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

The screenshot shows an AWS CloudShell terminal window. The user has run the command `kubectl apply -f cloudmart-frontend.yaml`, which applies a deployment and service configuration. The user then runs `kubectl get service` to list the services. The output shows two services: `cloudmart-backend-app-service` and `cloudmart-frontend-app-service`. Both are LoadBalancers with ClusterIPs of `10.100.240.29` and `10.100.175.52` respectively. They have external IP addresses: `a3ed30030ea945e797be912f746f78e-936372663.us-east-1.elb.amazonaws.com` and `ae56c87e1e94f456aa5182c4a5c5346e-559642764.us-east-1.elb.amazonaws.com`. The ports listed are `5000:30485/TCP` and `5001:32019/TCP`. The age of the services is `46m` and `7m30s` respectively. The user also lists a ClusterIP service named `ubernetes` with IP `10.100.0.1` and port `443/TCP`.

I typed this URL:

<http://ae56c87e1e94f456aa5182c4a5c5346e-559642764.us-east-1.elb.amazonaws.com:5001/admin>

AND VOILA!!! its working

The screenshot shows the CloudMart Product Management interface. At the top, there is a blue header bar with the CloudMart logo, a user icon labeled "Anonymous", and a shopping cart icon. Below the header is a title bar with "Product Management" and a "Add Product" button. A table header row is visible with columns for PRODUCT, DESCRIPTION, PRICE, and ACTIONS. The main content area below the table is currently empty. At the bottom of the page is a dark footer bar containing the CloudMart logo, the tagline "Your AI-powered shopping destination", and links to Home, Cart, My Orders, About, and Support. A copyright notice at the bottom of the footer states "© 2024 CloudMart. All rights reserved."

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

The screenshot shows a web browser window for 'CloudMart'. The title bar indicates it's not secure. The main content area is titled 'Product Management' with a '+ Add Product' button. A table lists one product: 'white T-shirt for men' with a price of '\$100.00'. The bottom of the page features a dark footer with the CloudMart logo, a search bar, and links for Home, Cart, My Orders, About, and Support. A copyright notice at the bottom states '© 2024 CloudMart. All rights reserved.'

And here is the frontend

The screenshot shows the CloudMart frontend for a product. The top navigation bar includes a menu icon, 'CloudMart' logo, 'Anonymous' status, and a shopping cart icon. Below the header, a 'Featured Products' section displays a white T-shirt with its details: 'white T-shirt for men', 'white T-shirt for men', '\$100.00', and a blue 'Add to Cart' button. To the right of the product card is a search bar with the placeholder 'Search products...'. The overall layout is clean and modern.

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

