Capstone Project: Production Storage Management

Project Overview

The goal of this capstone project was to design, implement, and deploy a scalable **Product Inventory Store** application using a **3-tier architecture (**app,web,database tiers) on AWS. The infrastructure was built to be highly available, secure, and automated across **two regions**, ensuring fault tolerance and enhanced performance.

Git hub links :

Region-1(N.Virginia) : https://github.com/vinayz7/3-tier-app-feature

Region-2(N.California): vinayz7/Capstone-Final-tf

Argo cd : vinayz7/argo_cd

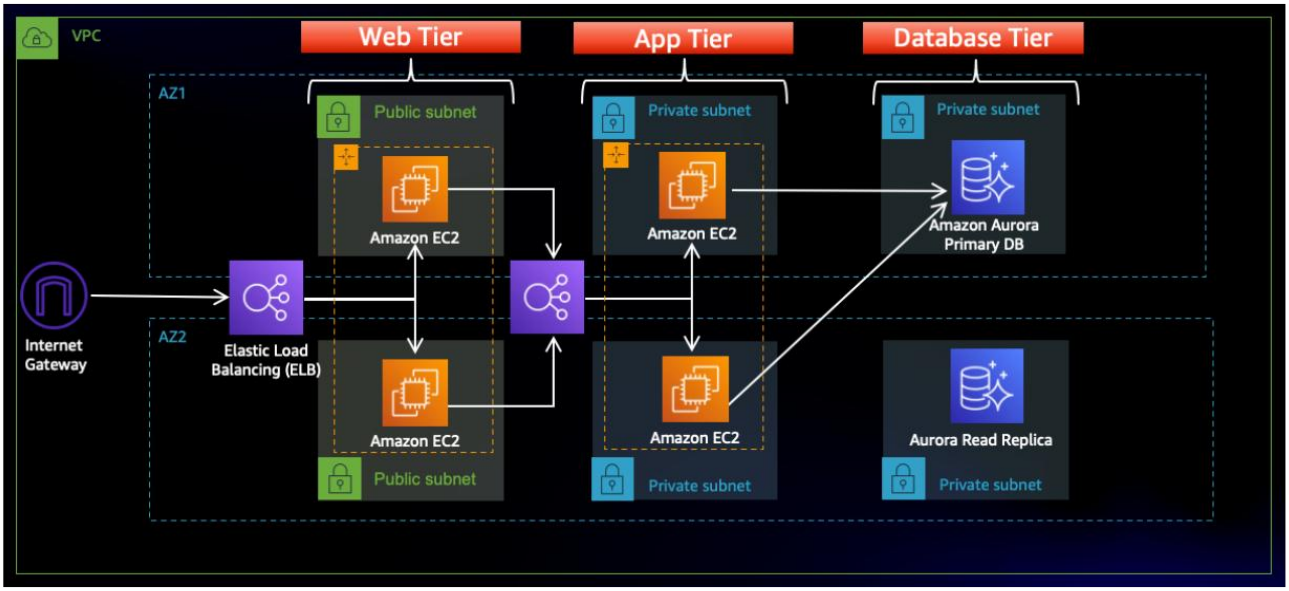
Architecture (Region-1: CloudFormation)

- **VPC with 2 Availability Zones**
- **6 Subnets**
    - Web Tier: 2 subnets
    - App Tier: 2 subnets
    - Database Tier: 2 subnets
- **Internet Gateway**, **Route Tables** with proper associations
- **Security Groups** for each layer
- **Amazon EKS Cluster** for container orchestration
- **Amazon RDS** for backend database storage

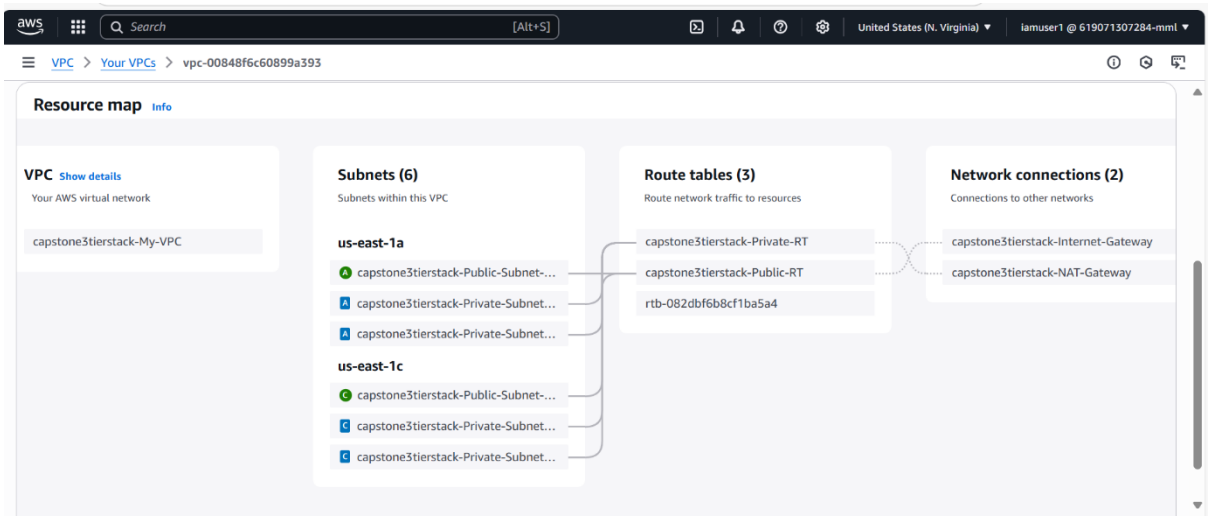
Entire infrastructure was provisioned using **AWS CloudFormation Templates** integrated into a **CI/CD pipeline**.

Base architecture:

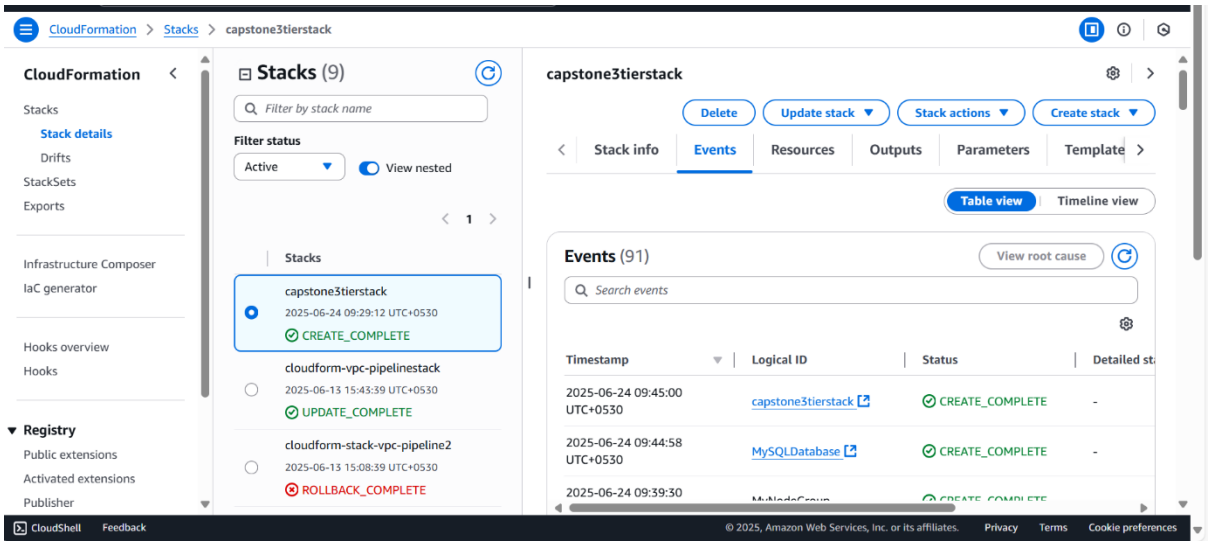
Architecture Overview



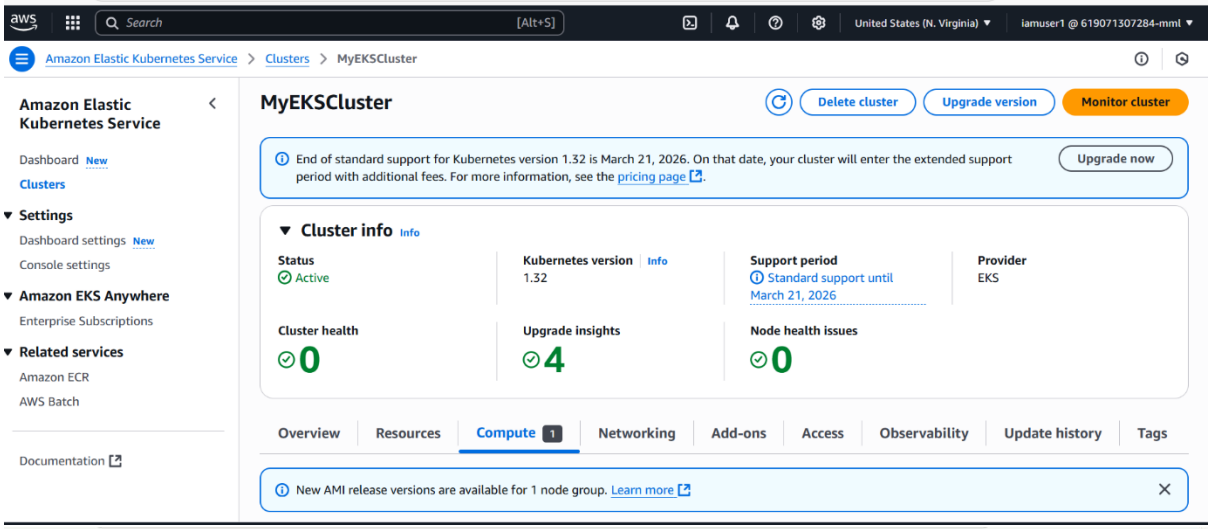
Vpc: subnets and route tables and network connections

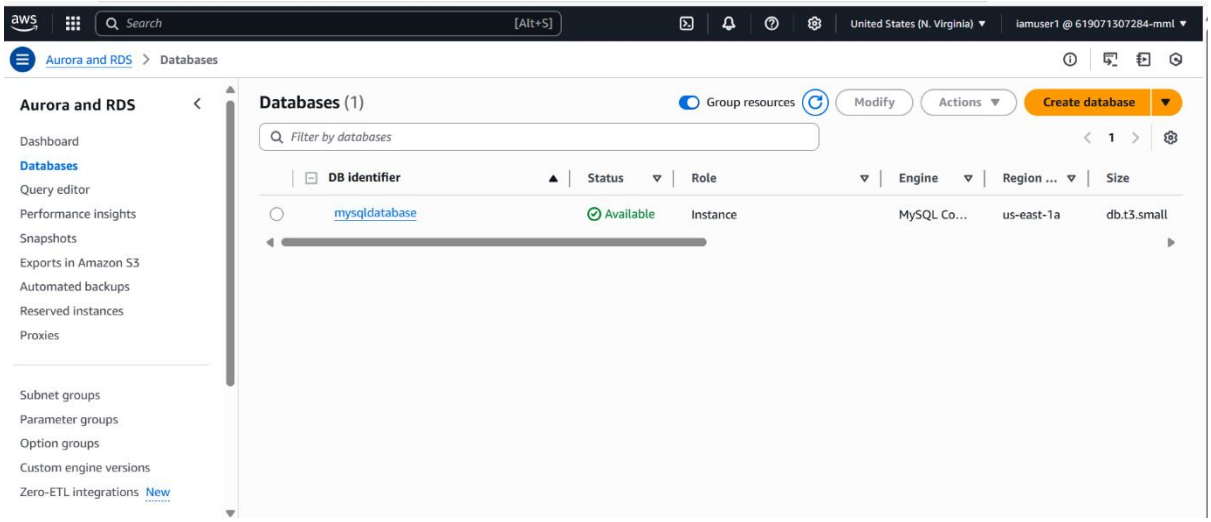


Cloud formation stack: I deployed a core infrastructure using cloud formation



Mostly eks cluster and rds(database is created successfully)





CI/CD Pipeline and Code Quality:

CodePipeline Integration:

Created separate AWS CodePipeline setups in both regions for automated build and deployment to EKS.

In Region-1, integrated SonarQube for static code analysis and quality checks during the build phase.

Kubernetes Deployment:

An Amazon EKS Cluster was provisioned within the application tier.

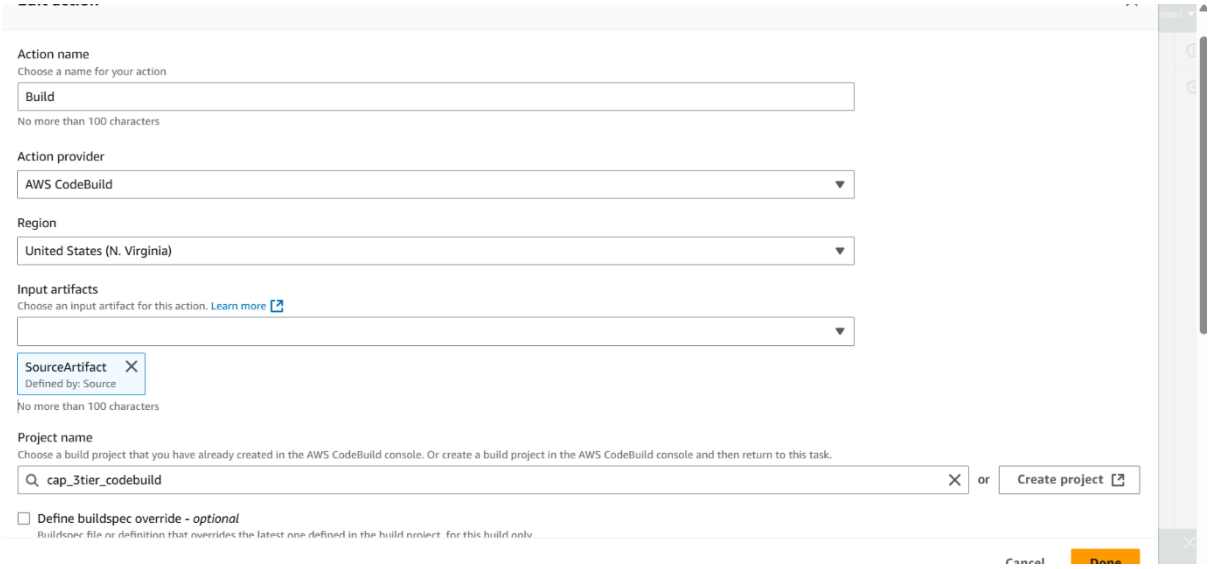
CI/CD Integration (Region-1)

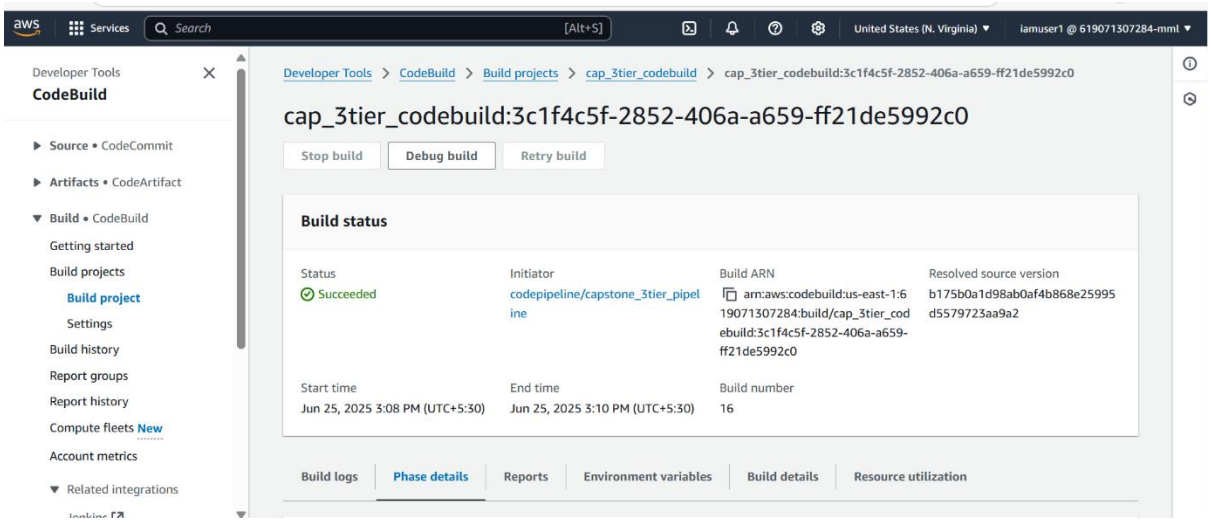
AWS CodePipeline for continuous integration and deployment

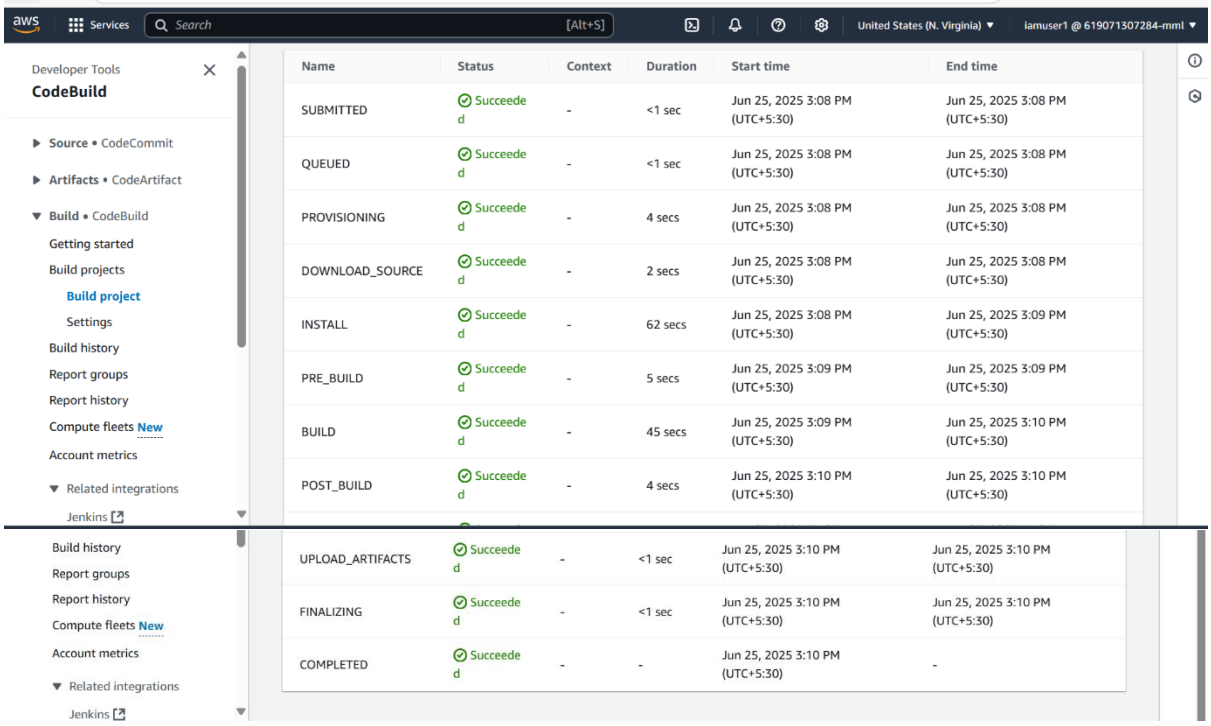
Application deployed to EKS via CodePipeline

Code pipeline: source ,code build ,deploy

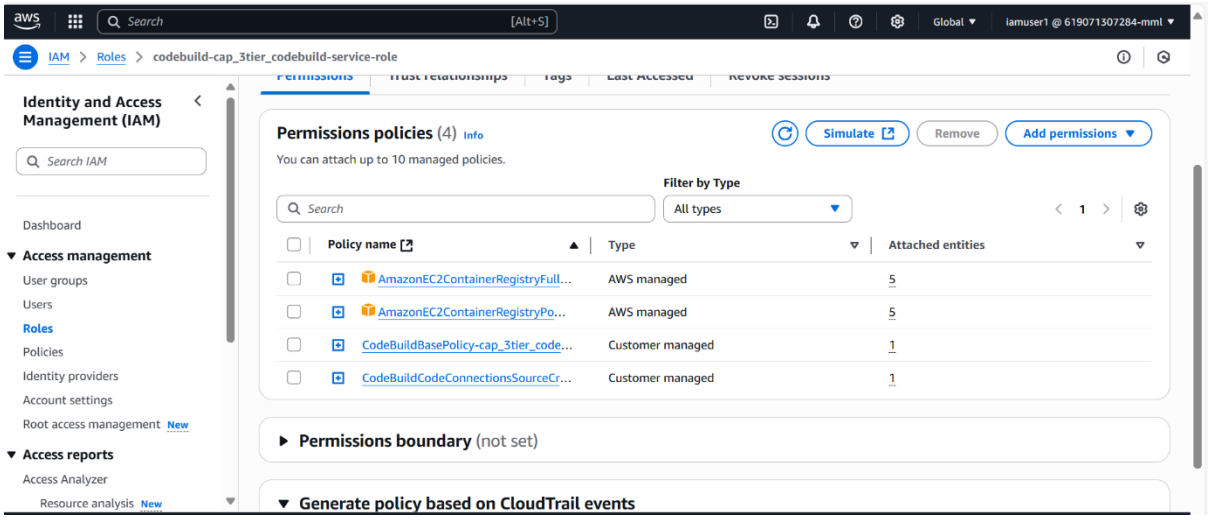
Code build:




aws
Services
Search
[Alt+S]
United States (N. Virginia)
iamuser1 @ 619071307284-mml
Developer Tools
CodeBuild
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Getting started
Build projects
Build project
Settings
Build history
Report groups
Report history
Compute fleets New
Account metrics
Related integrations
Developer Tools > CodeBuild > Build projects > cap_3tier_codebuild > cap_3tier_codebuild:3c1f4c5f-2852-406a-a659-ff21de5992c0
cap_3tier_codebuild:3c1f4c5f-2852-406a-a659-ff21de5992c0
Stop build
Debug build
Retry build
Build status
Status
Succeeded
Initiator
codepipeline/capstone_3tier_pipeline
Build ARN
arn:aws:codebuild:us-east-1:619071307284:build/cap_3tier_codebuild:3c1f4c5f-2852-406a-a659-ff21de5992c0
Resolved source version
b175b0a1d98ab0af4b868e25995d5579723aa9a2
Start time
Jun 25, 2025 3:08 PM (UTC+5:30)
End time
Jun 25, 2025 3:10 PM (UTC+5:30)
Build number
16
Build logs
Phase details
Reports
Environment variables
Build details
Resource utilization


aws
Services
Search
[Alt+S]
United States (N. Virginia)
iamuser1 @ 619071307284-mml
Developer Tools
CodeBuild
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Getting started
Build projects
Build project
Settings
Build history
Report groups
Report history
Compute fleets New
Account metrics
Related integrations
Jenkins
Name
Status
Context
Duration
Start time
End time
SUBMITTED
Succeeded
-
<1 sec
Jun 25, 2025 3:08 PM (UTC+5:30)
Jun 25, 2025 3:08 PM (UTC+5:30)
QUEUED
Succeeded
-
<1 sec
Jun 25, 2025 3:08 PM (UTC+5:30)
Jun 25, 2025 3:08 PM (UTC+5:30)
PROVISIONING
Succeeded
-
4 secs
Jun 25, 2025 3:08 PM (UTC+5:30)
Jun 25, 2025 3:08 PM (UTC+5:30)
DOWNLOAD_SOURCE
Succeeded
-
2 secs
Jun 25, 2025 3:08 PM (UTC+5:30)
Jun 25, 2025 3:08 PM (UTC+5:30)
INSTALL
Succeeded
-
62 secs
Jun 25, 2025 3:08 PM (UTC+5:30)
Jun 25, 2025 3:09 PM (UTC+5:30)
PRE_BUILD
Succeeded
-
5 secs
Jun 25, 2025 3:09 PM (UTC+5:30)
Jun 25, 2025 3:09 PM (UTC+5:30)
BUILD
Succeeded
-
45 secs
Jun 25, 2025 3:09 PM (UTC+5:30)
Jun 25, 2025 3:10 PM (UTC+5:30)
POST_BUILD
Succeeded
-
4 secs
Jun 25, 2025 3:10 PM (UTC+5:30)
Jun 25, 2025 3:10 PM (UTC+5:30)
UPLOAD_ARTIFACTS
Succeeded
-
<1 sec
Jun 25, 2025 3:10 PM (UTC+5:30)
Jun 25, 2025 3:10 PM (UTC+5:30)
FINALIZING
Succeeded
-
<1 sec
Jun 25, 2025 3:10 PM (UTC+5:30)
Jun 25, 2025 3:10 PM (UTC+5:30)
COMPLETED
Succeeded
-
-
Jun 25, 2025 3:10 PM (UTC+5:30)
-

Permissions for code build: for this code build have to go 2 permissions for this iam role

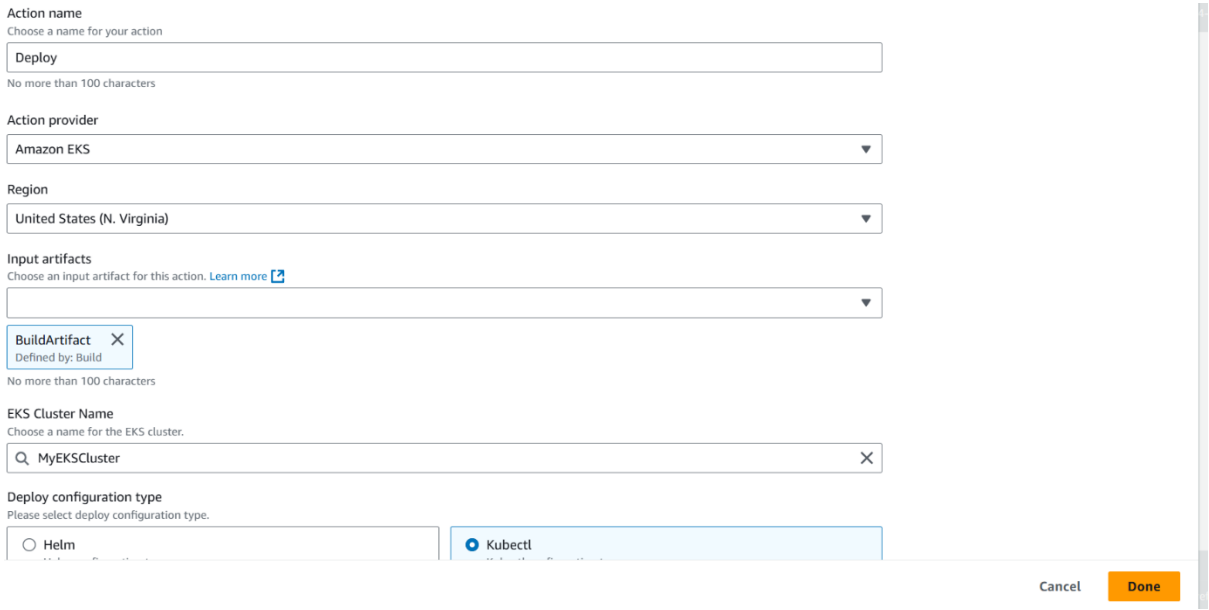


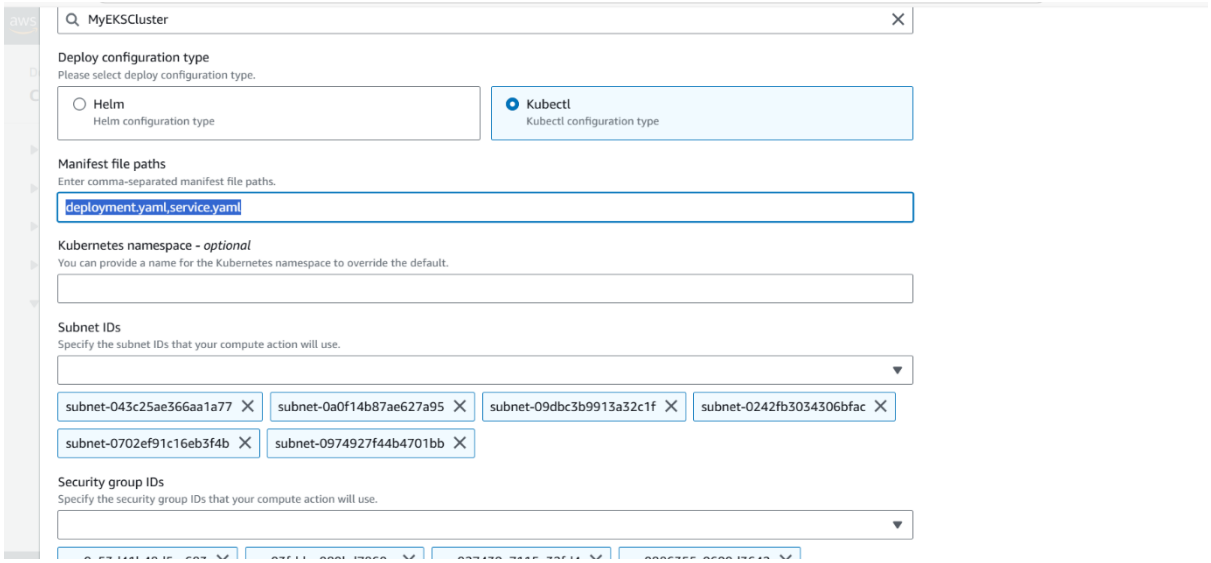
Code build is success...

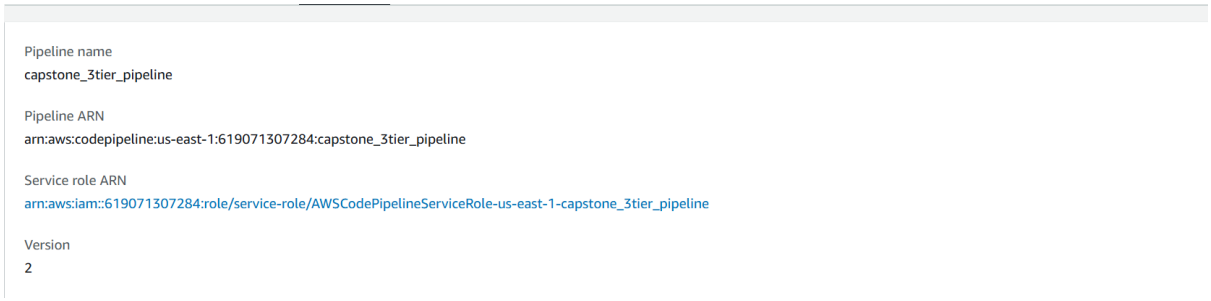
Code deploy :

AWS CodePipeline for continuous integration and deployment

Application deployed to EKS via CodePipeline



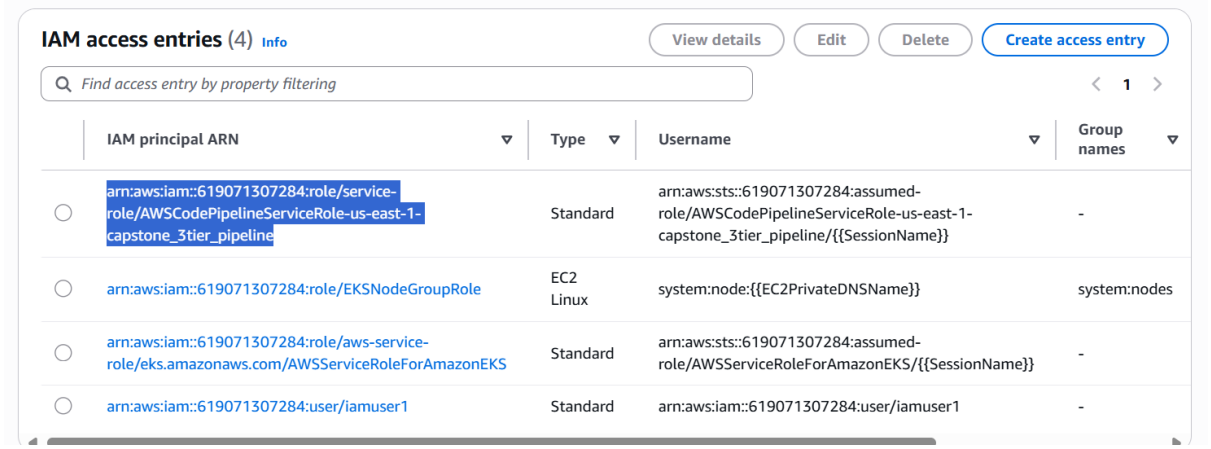




We have to give a permissions to eks cluster for this pipeline to deploy

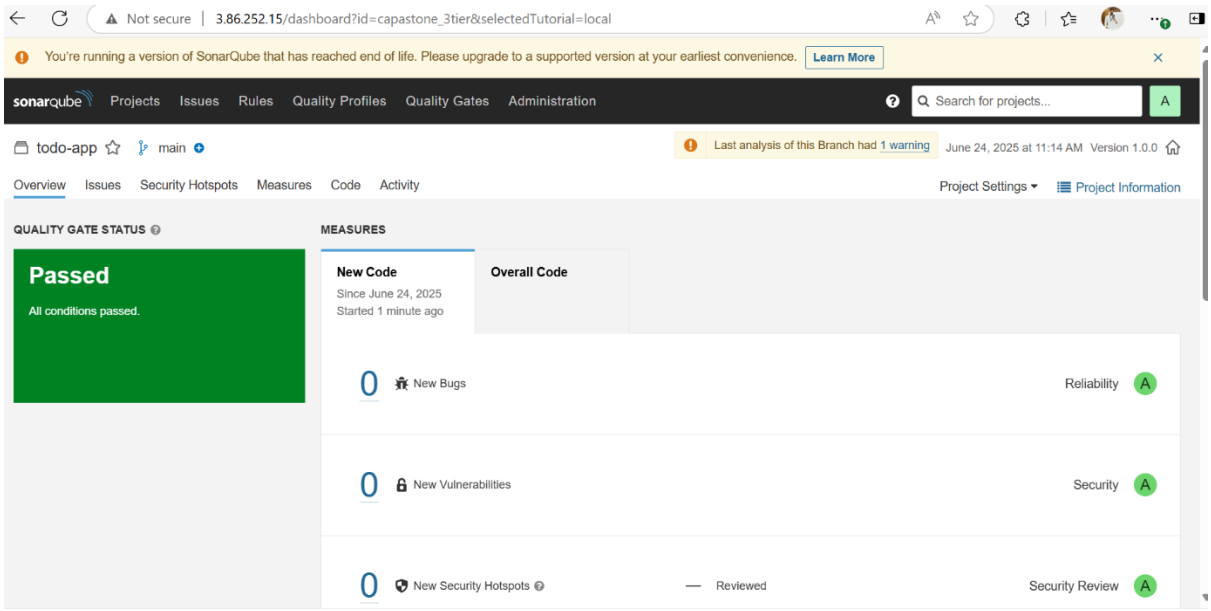
1.i have taken this pipeline service role ARN

2.now in eks cluster in access entries add this role

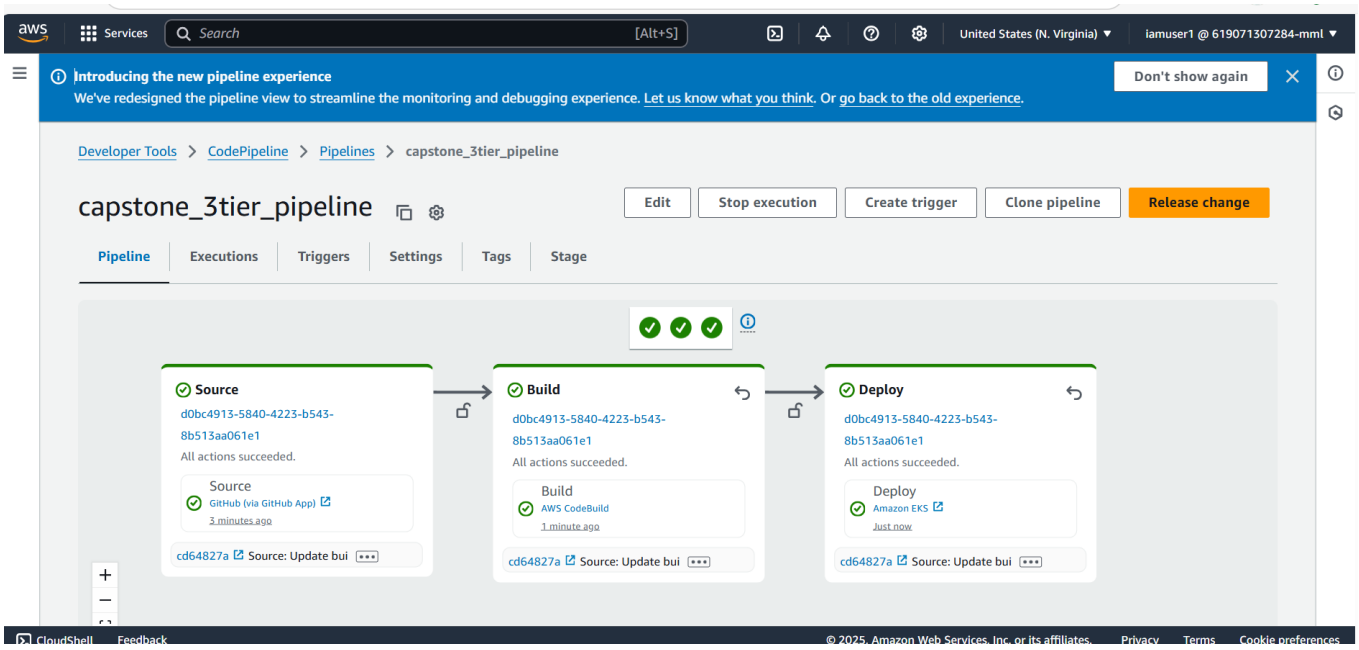


In **Region-1**, integrated **SonarQube** for static code analysis and quality checks during the build phase.

```
- mvn clean package -DskipTests
# - echo Running SonarQube analysis...
# - mvn clean verify sonar:sonar -Dsonar.projectKey=TODO -Dsonar.host.url=http://54.165.220.18 -Dsonar.login=sqp_9aef180c6bef97fc3f3a94b811ec6ae0e54dbb06
# - echo Building Docker image...
```

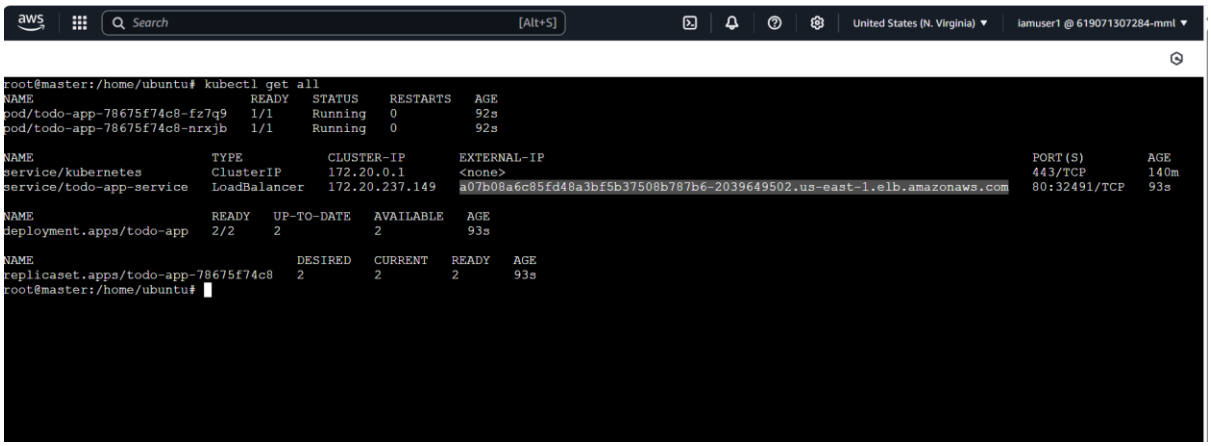


Finally application is deployed in eks



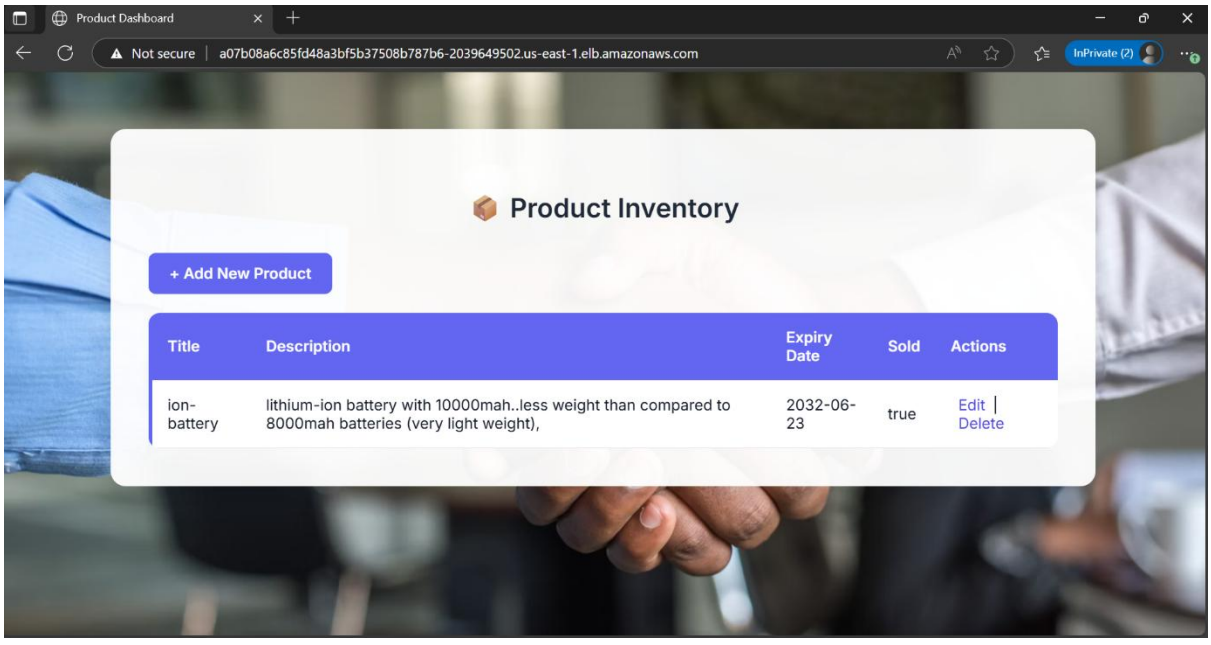
For manual check : in ec2

For service type: load balancer ..its created successfully



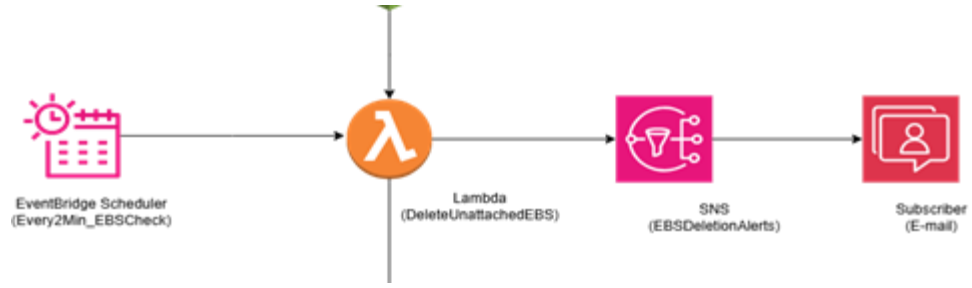
External ip is provided by aws ...copy the load balancer and access into the external browser

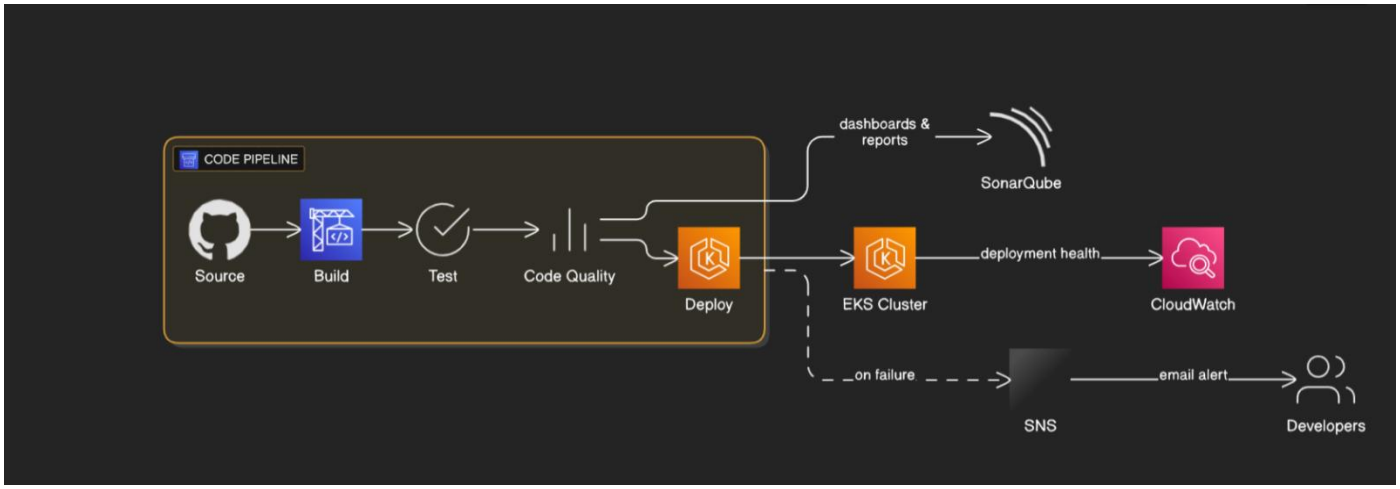
Successfully deployed the application rds database also connected successfully



For this pipeline I have added event **bridge rule** (target to **lamba** and **sns**)

If pipeline fails immediately we can get email notification





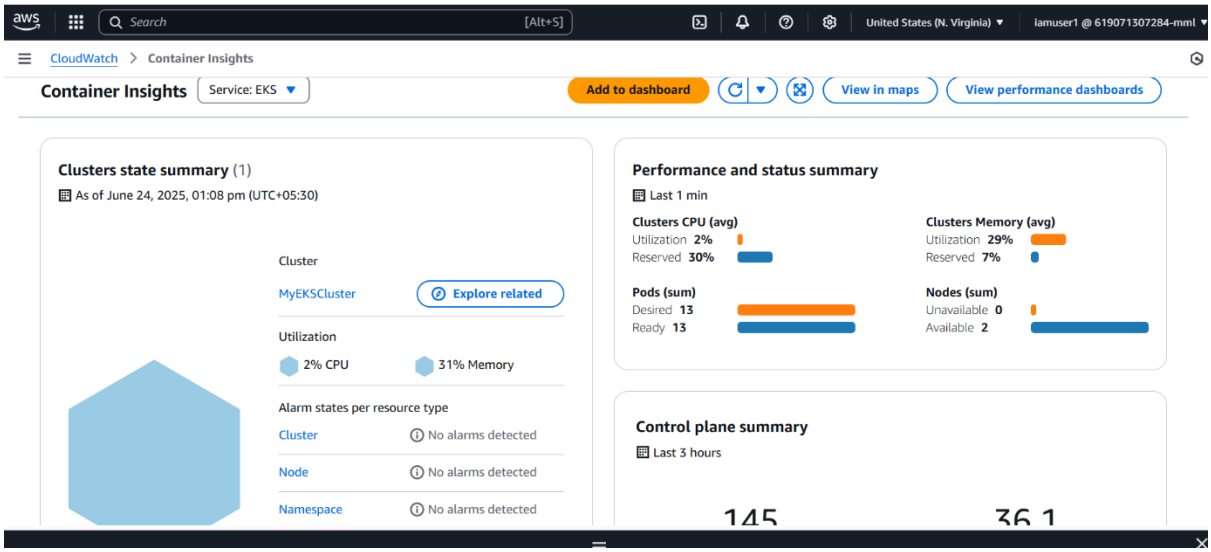
Any developers makes any changes ..pipeline will trigger and run this all process continues integration and continues deployment..

Cloud watch monitoring :

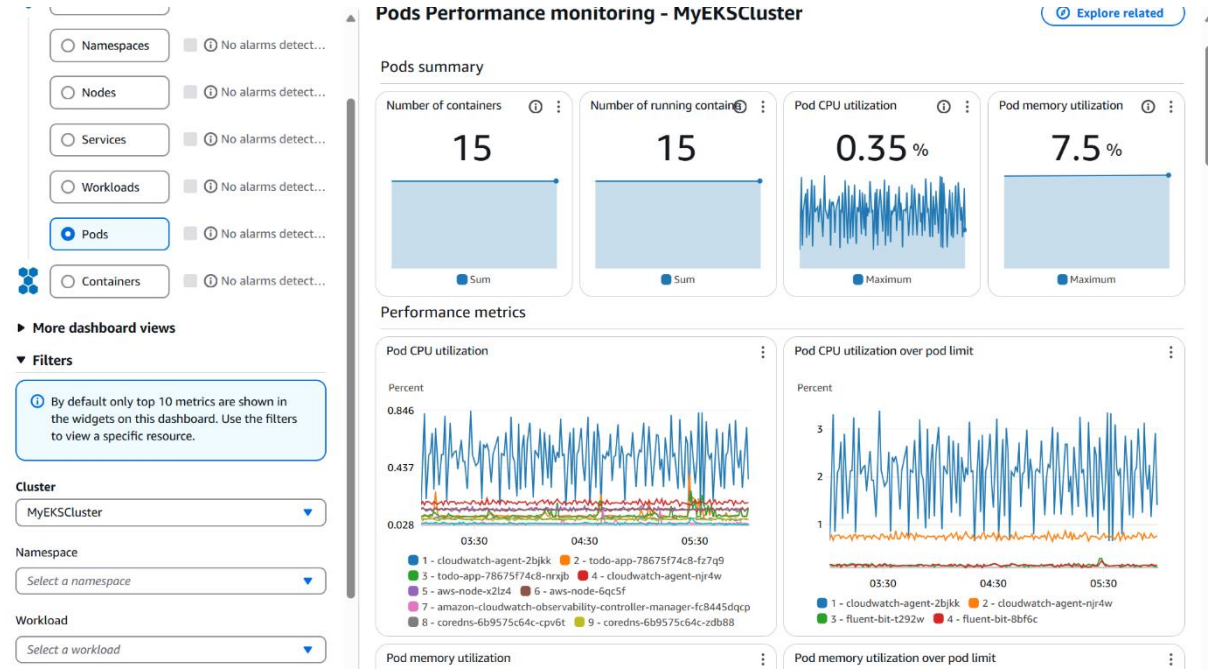
I have configured a **eks cluster** into cloud watch monitoring

We check the logs and metrics as well ..and health check we can monitor

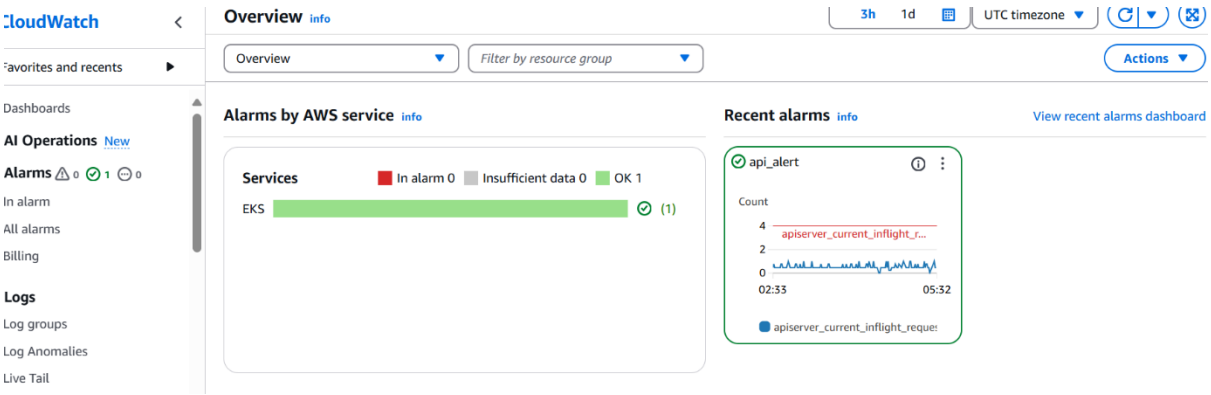
We can set alarms for this resource



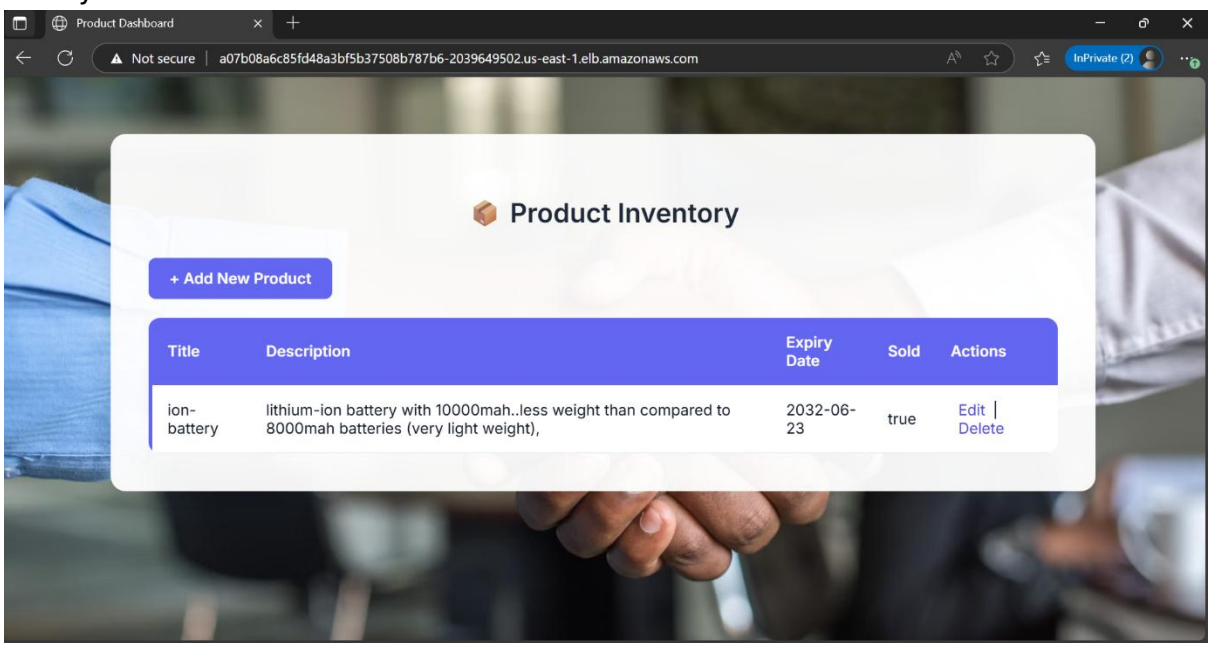
es



I have set an alarm for **api pod:**



Finally :



----------in One region is completed (N.Virginia) ----------

Region-2((N.California):

The same setup was recreated in Region-2 using Terraform, showcasing multi-region deployment capability.

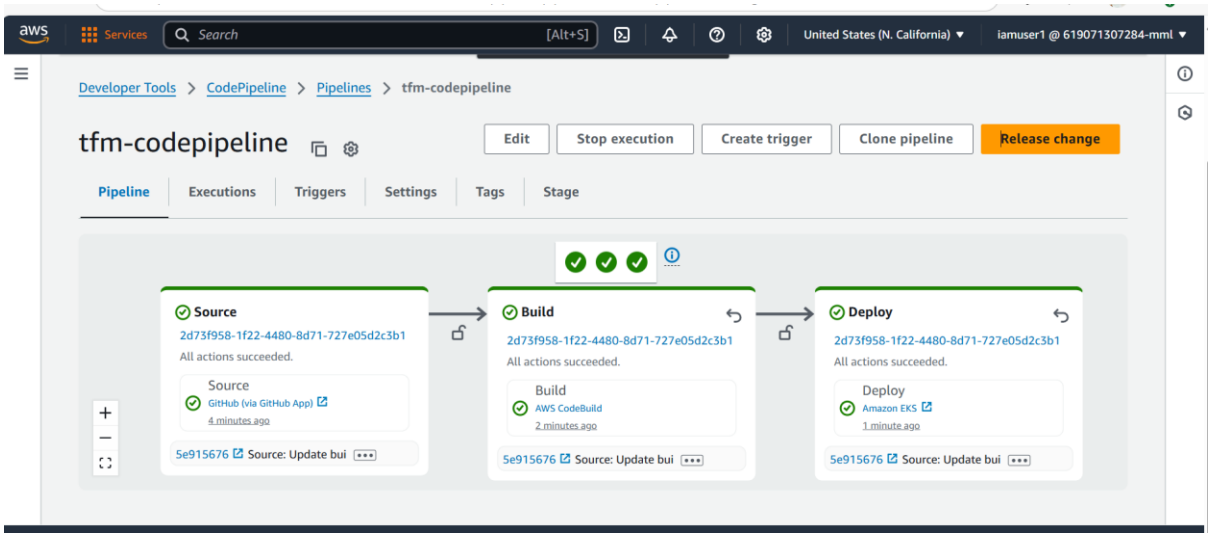
In Region-2, integrated Trivy for container image vulnerability scanning before deployment.

Vpc ,subnets,and network connections ,eks cluster ,database created successfully.

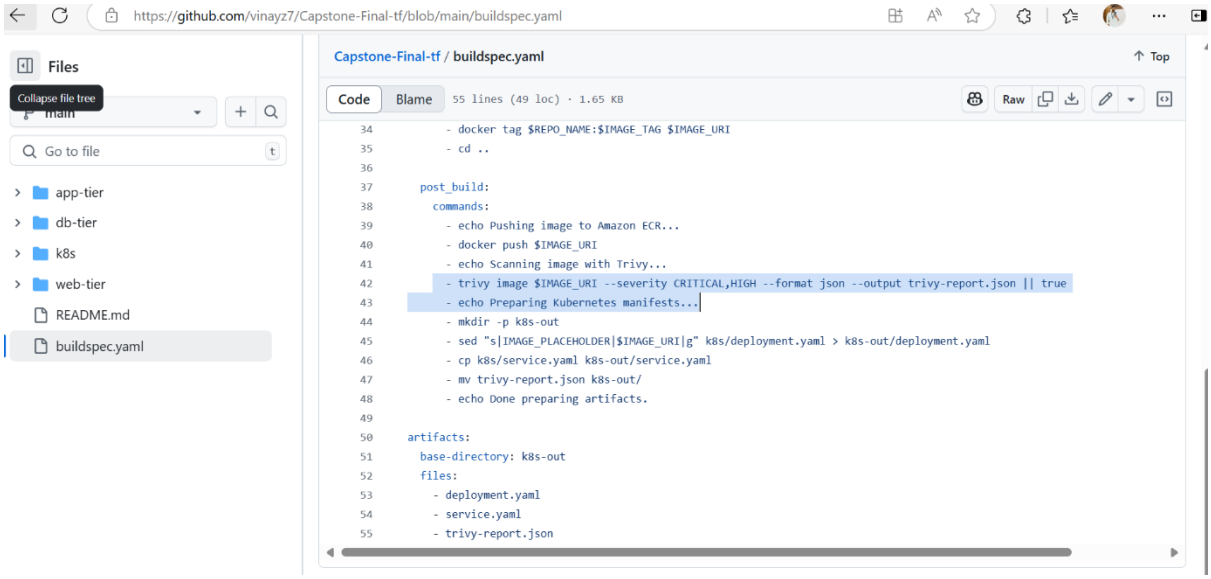
Core architecture is deployed using terraform

```
20250624095611546200000004]
aws_iam_role_policy_attachment.eks_node_role_attachments["AmazonEC2ContainerRegistryReadOnly"]: Refreshing state... [id=EKSNodeGroupRoleT
erraform-20250624095611389300000001]
aws_iam_role_policy_attachment.eks_cluster_role_attachments["AmazonEKSClusterPolicy"]: Refreshing state... [id=EKSClusterRoleTerraform-20
250624095611390300000002]
aws_iam_role_policy_attachment.eks_cluster_role_attachments["AmazonEKSVPCResourceController"]: Refreshing state... [id=EKSClusterRoleTerr
aform-20250624095611611100000006]
aws_iam_role_policy_attachment.eks_cluster_role_attachments["AdministratorAccess"]: Refreshing state... [id=EKSClusterRoleTerraform-20250
624095611635200000007]
aws_route_table.public: Refreshing state... [id=rtb-02cbf094f0f60c287]
aws_nat_gateway.nat: Refreshing state... [id=nat-06640a090a3856c94]
aws_db_subnet_group.rds_subnet: Refreshing state... [id=three-tier-db-subnet-group]
aws_eks_cluster.cluster: Refreshing state... [id=three-tier-cluster]
aws_route_table_association.public[1]: Refreshing state... [id=rtbassoc-00fcd8619e45b1067]
aws_route_table_association.public[0]: Refreshing state... [id=rtbassoc-04099c52cddcb8b52]
aws_route_table.private: Refreshing state... [id=rtb-0f40dba4cc57de429]
aws_route_table_association.private[0]: Refreshing state... [id=rtbassoc-0e385b9771b4358c7]
aws_db_instance.mysql: Refreshing state... [id=db-OLDKPH77TPNWBSARDAMMHJ5EOQ]
aws_route_table_association.private[1]: Refreshing state... [id=rtbassoc-0dc538a7367bacba0]
aws_eks_node_group.nodes: Refreshing state... [id=three-tier-cluster:three-tier-nodegroup]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
root@ip-172-31-25-104:/home/ubuntu#
```

CodePipeline for deployment to Region-2 EKS:

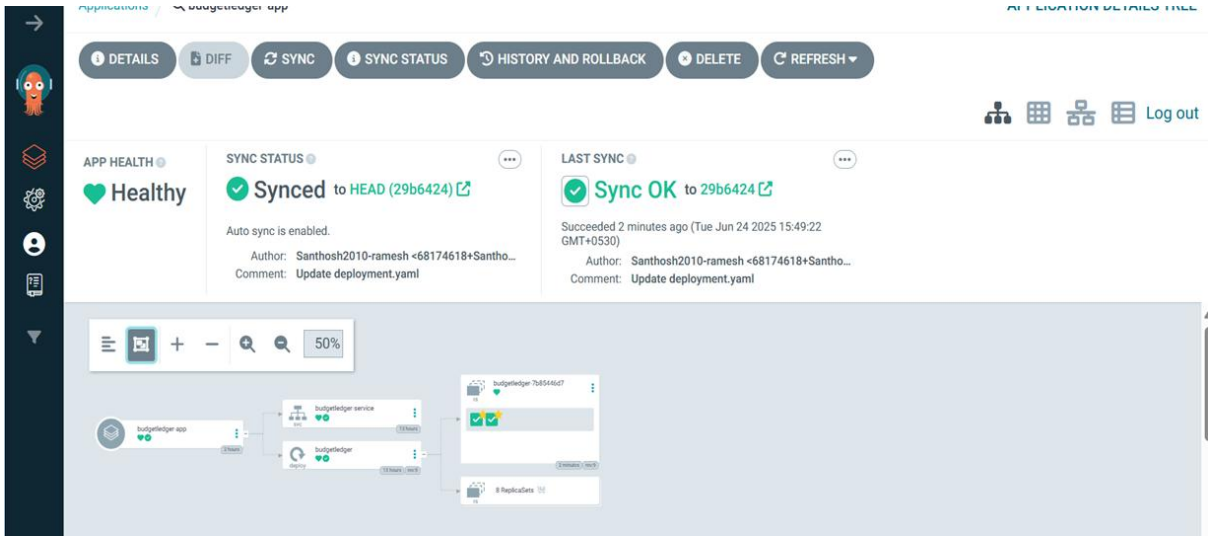


Trivy integrated for container image vulnerability scanning

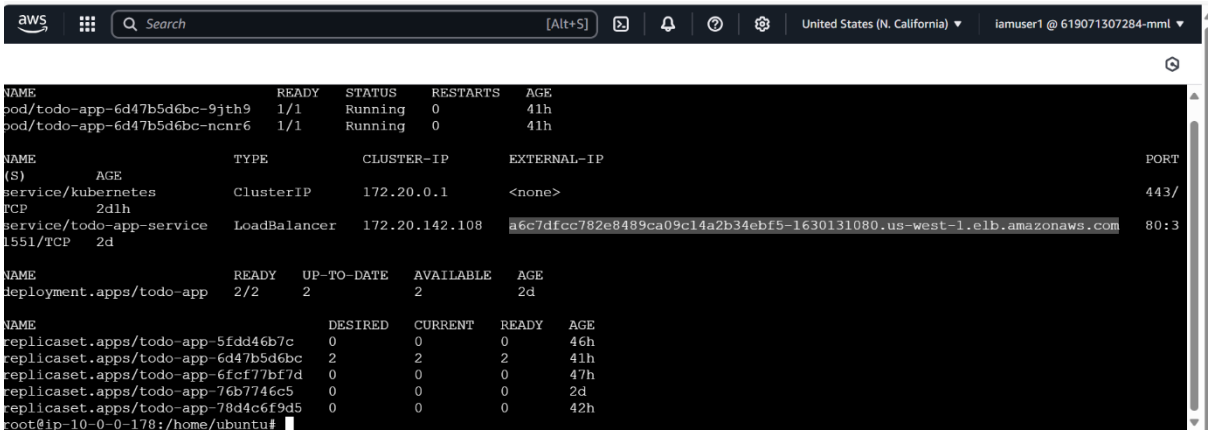


Trivy logs will save into trivy-reports.json (in Json format)

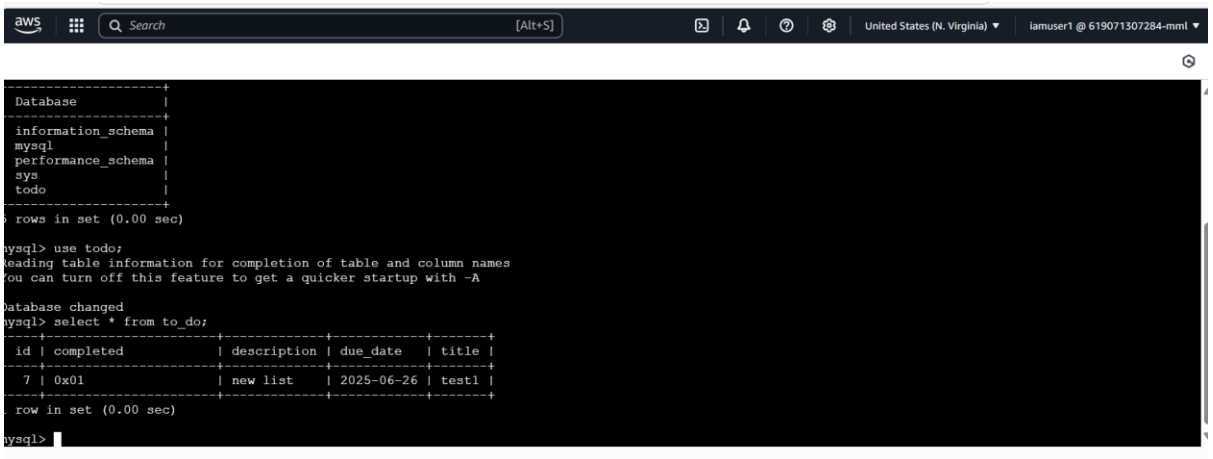
Argo CD set up independently for GitOps-style deployment:



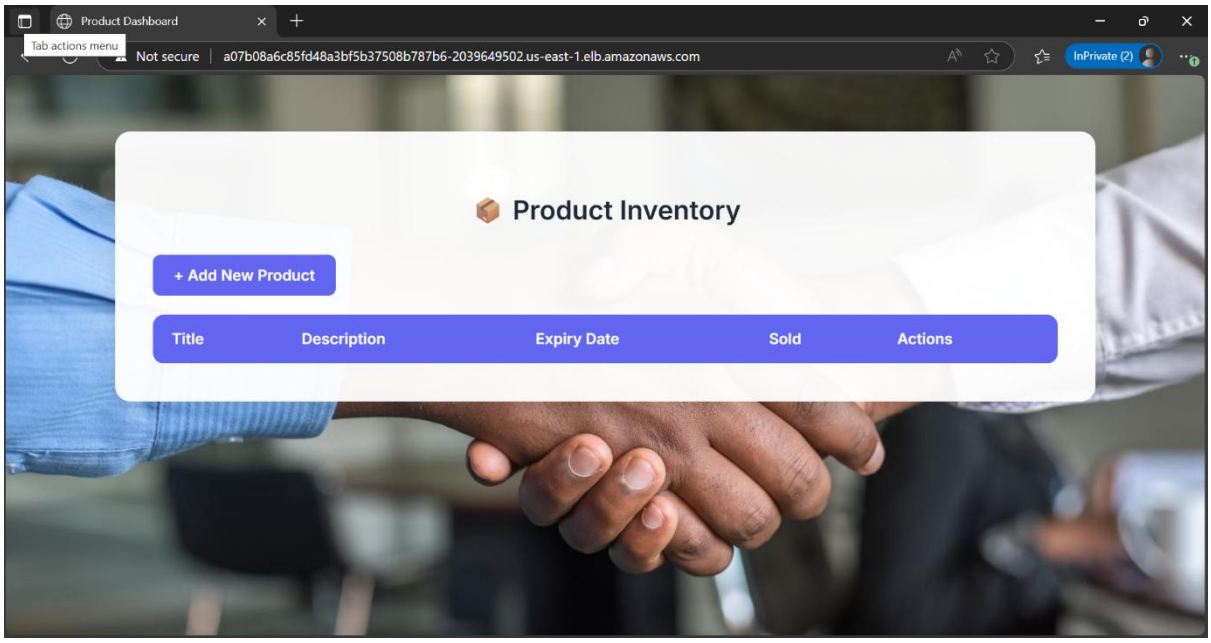
For manual check:



For **rds(database)** I've checked data base connected successfully



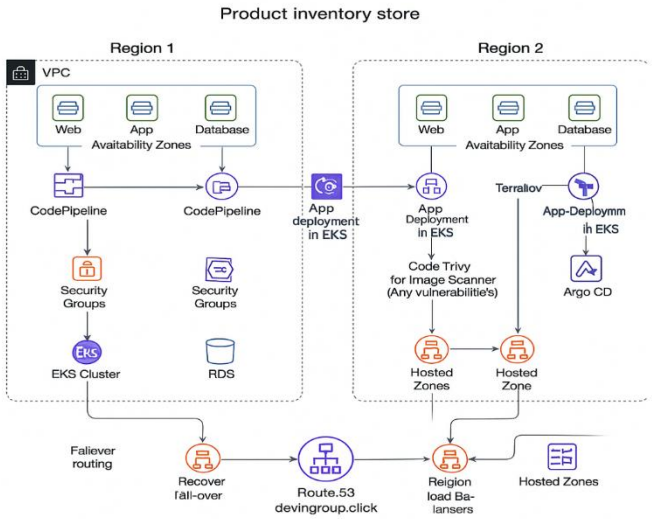
Accessing into externe browser:



In region 2 also deployed successfully

DNS and High Availability:

- Route 53 Configuration:
    - Configured Amazon Route 53 with two domains.
    - Created public hosted zones with health checks and failover routing policies.
    - Ensured high availability and resilience by directing traffic to the healthy region in case of a failure in the primary region.

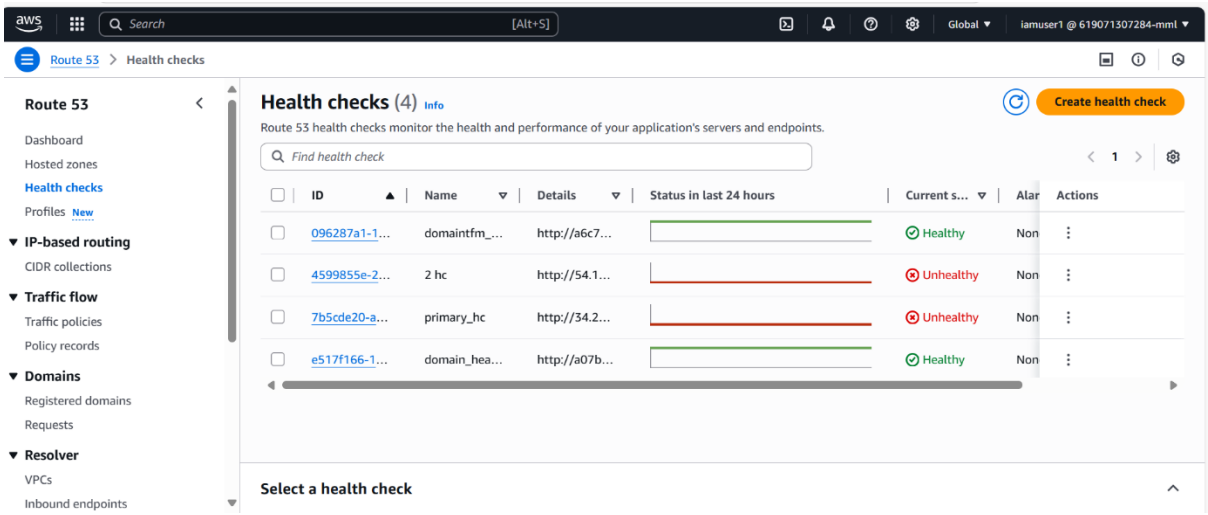


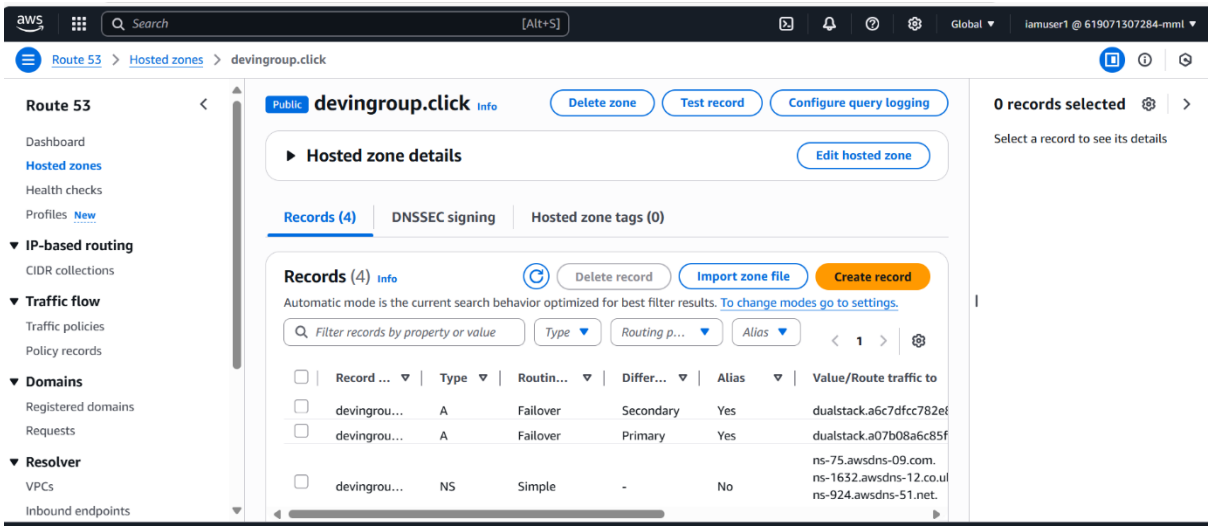
Route 53 (devingroup.click)
Configured with two hosted zones for each region.
Integrated with regional load balancers.
Failover routing policy ensures high availability—if one region goes down, traffic is automatically routed to the healthy region.

I have created a hosted zone (devingroup.click)

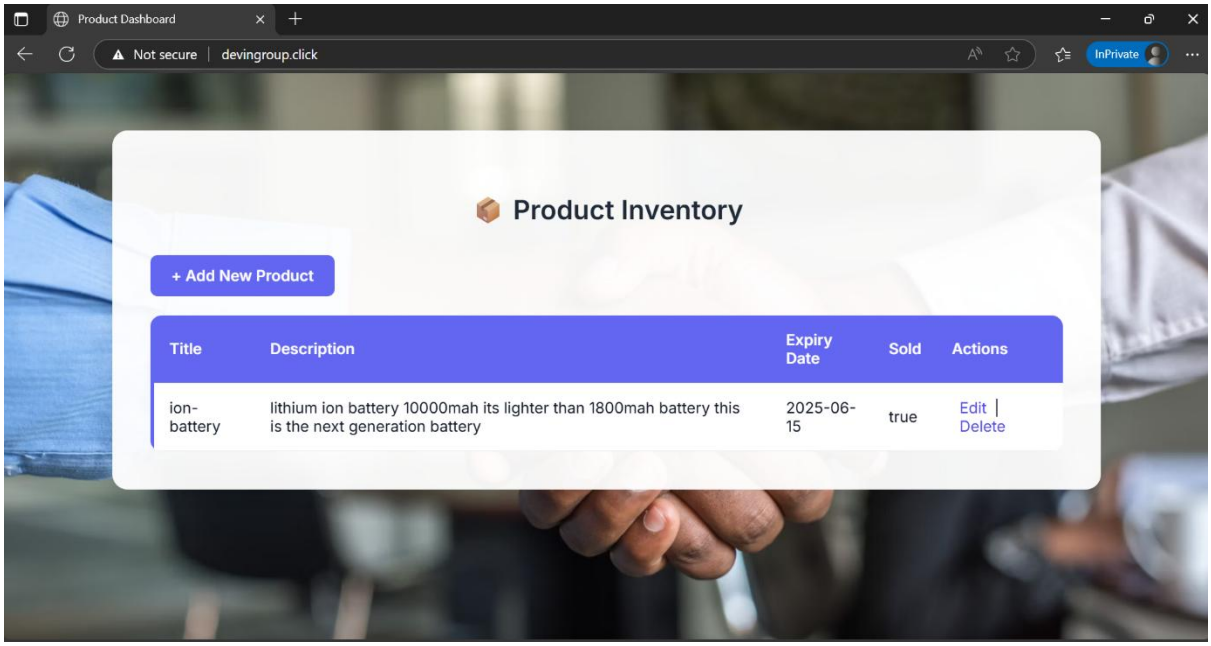
Created health check for 2 region load balancers

And created a records for this 2 load balancer end points





Access into external browser:



⭐ Key Advantages:

- **Infrastructure as Code (IaC):** Automated provisioning using CloudFormation and Terraform enhances repeatability and reduces human error.
- **High Availability & Resilience:** Multi-AZ and multi-region deployment ensures zero downtime during regional failures.
- **Security:** Integrated tools like Trivy for image scanning and SonarQube for code quality help enforce secure and reliable deployments.
- **Scalability:** Kubernetes on EKS enables horizontal scaling of application workloads based on demand.

- **Monitoring & Observability:** CloudWatch integration provides deep visibility into application and infrastructure metrics.
- **DevOps Best Practices:** Full CI/CD pipeline setup shortens release cycles and promotes continuous delivery.
- **Cost Optimization:** Segregated subnets and resource allocation help manage cost efficiently while maintaining performance.

Outcome

The project demonstrates a real-world, production-grade deployment scenario using:

- **Infrastructure as Code (IaC)**
- **CI/CD automation**
- **Security & compliance checks**
- **Monitoring & alerting**
- **Multi-region fault-tolerant architecture**

The solution is scalable, maintainable, and aligns with best practices in cloud infrastructure engineering.