

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](#)

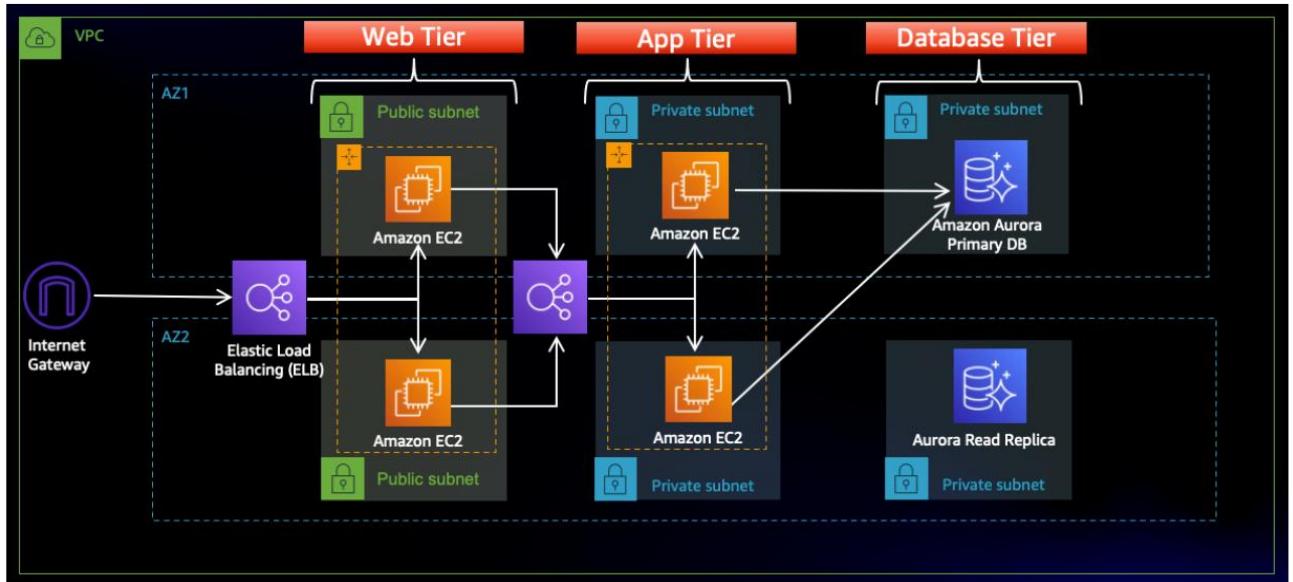
Architecture (Region-1: CloudFormation)

- **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

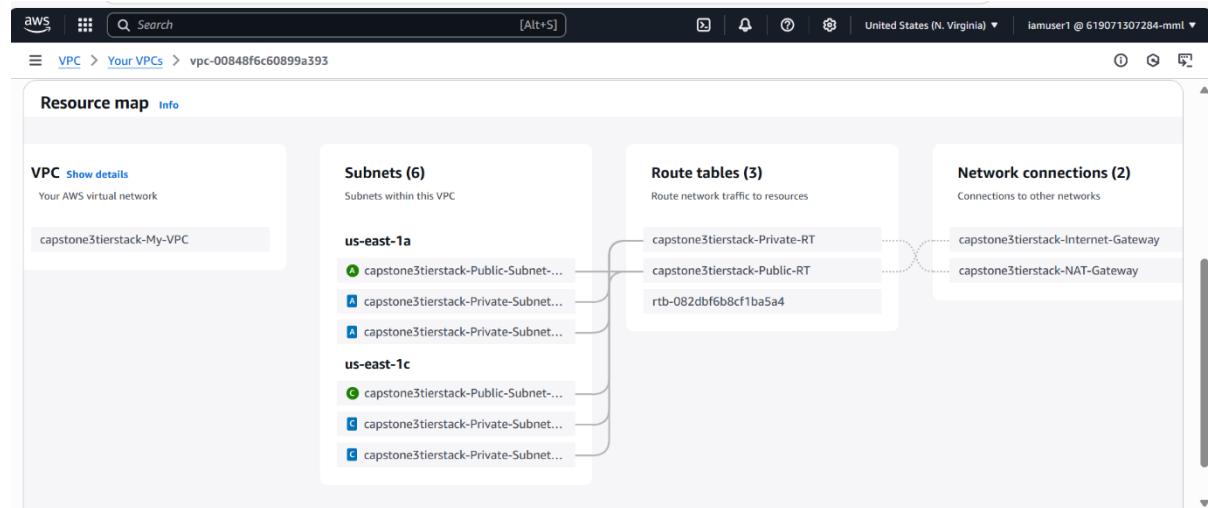
Entire infrastructure was provisioned using **AWS CloudFormation Templates** integrated into a **CI/CD pipeline**.

Base architecture:

Architecture Overview



Vpc: subnets and route tables and network connections



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

CloudFormation > Stacks > capstone3tierstack

Stacks (9)

Events (91)

Timestamp	Logical ID	Status	Detailed status
2025-06-24 09:45:00 UTC+0530	capstone3tierstack	CREATE_COMPLETE	-
2025-06-24 09:44:58 UTC+0530	MySQLDatabase	CREATE_COMPLETE	-
2025-06-24 09:39:30	MultiNodeGroup	CREATE_COMPLETE	-

Mostly eks cluster and rds(database is created successfully)

Amazon Elastic Kubernetes Service > Clusters > MyEKSCluster

MyEKSCluster

Cluster info

Status	Kubernetes version	Support period	Provider
Active	1.32	Standard support until March 21, 2026	EKS

Compute

Aurora and RDS > Databases

Databases (1)

DB identifier	Status	Role	Engine	Region ...	Size
mysqldatabase	Available	Instance	MySQL Co...	us-east-1a	db.t3.small

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:

The screenshot shows the AWS CodeBuild configuration interface for a new action named 'Build'. The 'Action provider' is set to 'AWS CodeBuild'. The 'Region' is set to 'United States (N. Virginia)'. Under 'Input artifacts', there is a 'SourceArtifact' entry defined by 'Source'. The 'Project name' field contains 'cap_3tier_codebuild'. A checkbox for 'Define buildspec override - optional' is checked, with a note below it stating 'Buildspec file or definition that overrides the latest one defined in the build project for this build only'. At the bottom right, there are 'Cancel' and 'Done' buttons.

Action name
Choose a name for your action
Build
No more than 100 characters

Action provider
AWS CodeBuild

Region
United States (N. Virginia)

Input artifacts
Choose an input artifact for this action. [Learn more](#)

SourceArtifact
Defined by: Source
No more than 100 characters

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.
Q cap_3tier_codebuild or

Define buildspec override - optional
Buildspec file or definition that overrides the latest one defined in the build project for this build only

AWS Services Search [Alt+S] United States (N. Virginia) iamuser1 @ 619071307284-mm1

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

cap_3tier_codebuild:3c1f4c5f-2852-406a-a659-ff21de5992c0

Stop build Debug build Retry build

Build status

Status	Initiator	Build ARN	Resolved source version
Succeeded	codepipeline/capstone_3tier_pipeline	arn:aws:codebuild:us-east-1:19071307284:build/cap_3tier_codebuild:3c1f4c5f-2852-406a-a659-ff21de5992c0	b175b0a1d98ab0af4b868e25995d5579723aa9a2
Start time	End time	Build number	
Jun 25, 2025 3:08 PM (UTC+5:30)	Jun 25, 2025 3:10 PM (UTC+5:30)	16	

Build logs Phase details Reports Environment variables Build details Resource utilization

AWS Services Search [Alt+S] United States (N. Virginia) iamuser1 @ 619071307284-mm1

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

The screenshot shows the AWS IAM Roles Permissions page. The left sidebar includes sections for Identity and Access Management (IAM), Access management, and Access reports. The main area displays a table of attached policies:

Policy name	Type	Attached entities
AmazonEC2ContainerRegistryFullAccess	AWS managed	5
AmazonEC2ContainerRegistryPowerUser	AWS managed	5
CodeBuildBasePolicy-cap_3tier_codebuild-service-role	Customer managed	1
CodeBuildCodeConnectionsSourceCredentialsProvider	Customer managed	1

Below the table are sections for 'Permissions boundary (not set)' and 'Generate policy based on CloudTrail events'.

Code build is success...

Code deploy :

AWS CodePipeline for continuous integration and deployment

Application deployed to EKS via CodePipeline

The screenshot shows the AWS CodePipeline Deploy Action configuration page. The fields include:

- Action name: Deploy
- Action provider: Amazon EKS
- Region: United States (N. Virginia)
- Input artifacts: BuildArtifact (Defined by: Build)
- EKS Cluster Name: MyEKSCluster
- Deploy configuration type: Kubectl (selected)

Buttons at the bottom right include 'Cancel' and 'Done'.

aws X

Deploy configuration type
Please select deploy configuration type.

Helm
Helm configuration type

Kubectl
Kubectl configuration type

Manifest file paths
Enter comma-separated manifest file paths.
`deployment.yaml,service.yaml`

Kubernetes namespace - optional
You can provide a name for the Kubernetes namespace to override the default.

Subnet IDs
Specify the subnet IDs that your compute action will use.

subnet-043c25ae366aa1a77 X subnet-0a0f14b87ae627a95 X subnet-09dbc3b9913a32c1f X subnet-0242fb3034306bfac X
subnet-0702ef91c16eb3f4b X subnet-0974927f44b4701bb X

Security group IDs
Specify the security group IDs that your compute action will use.

Pipeline name
`capstone_3tier_pipeline`

Pipeline ARN
`arn:aws:codepipeline:us-east-1:619071307284:capstone_3tier_pipeline`

Service role ARN
`arn:aws:iam::619071307284:role/service-role/AWSCodePipelineServiceRole-us-east-1-capstone_3tier_pipeline`

Version
2

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

IAM access entries (4) [Info](#)

View details Edit Delete Create access entry < 1 >

IAM principal ARN	Type	Username	Group names
<code>arn:aws:iam::619071307284:role/service-role/AWSCodePipelineServiceRole-us-east-1-capstone_3tier_pipeline</code>	Standard	<code>arn:aws:sts::619071307284:assumed-role/AWSCodePipelineServiceRole-us-east-1-capstone_3tier_pipeline/{{SessionName}}</code>	-
<code>arn:aws:iam::619071307284:role/EKSNodeGroupRole</code>	EC2 Linux	<code>system:node:{{EC2PrivateDNSName}}</code>	system:nodes
<code>arn:aws:iam::619071307284:role/aws-service-role/eks.amazonaws.com/AWSServiceRoleForAmazonEKS</code>	Standard	<code>arn:aws:sts::619071307284:assumed-role/AWSServiceRoleForAmazonEKS/{{SessionName}}</code>	-
<code>arn:aws:iam::619071307284:user/iamuser1</code>	Standard	<code>arn:aws:iam::619071307284:user/iamuser1</code>	-

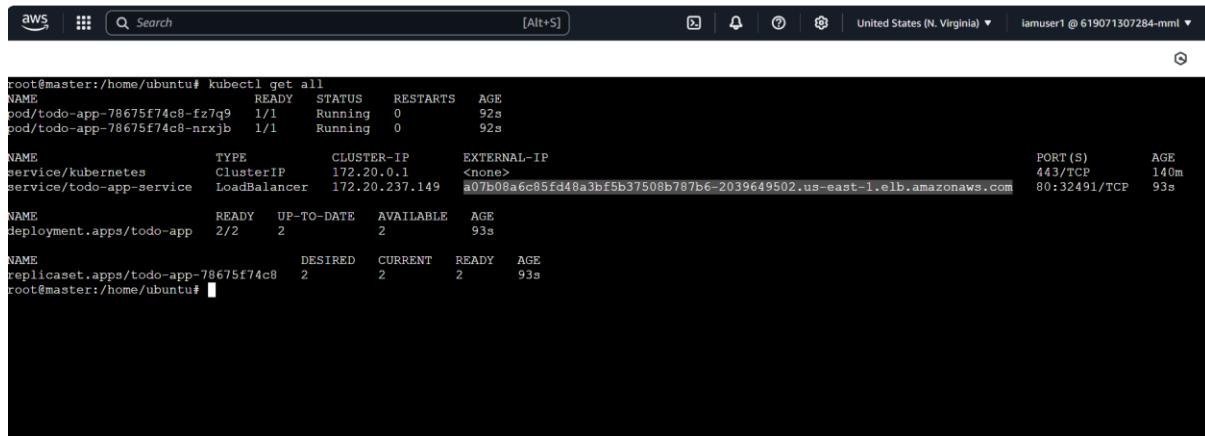
In Region-1, integrated SonarQube for static code analysis and quality checks during the build phase.

```
- 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=sq...  
# - echo Building Docker image...
```

Finally application is deployed in eks

For manual check : in ec2

For service type: load balancer ..its created successfully



```
aws | Search [Alt+S] | United States (N. Virginia) | iamuser1 @ 619071307284-mml | Q

root@master:/home/ubuntu# kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/todo-app-78675f74c8-fz7q9     1/1     Running   0          92s
pod/todo-app-78675f74c8-nrxjb    1/1     Running   0          92s

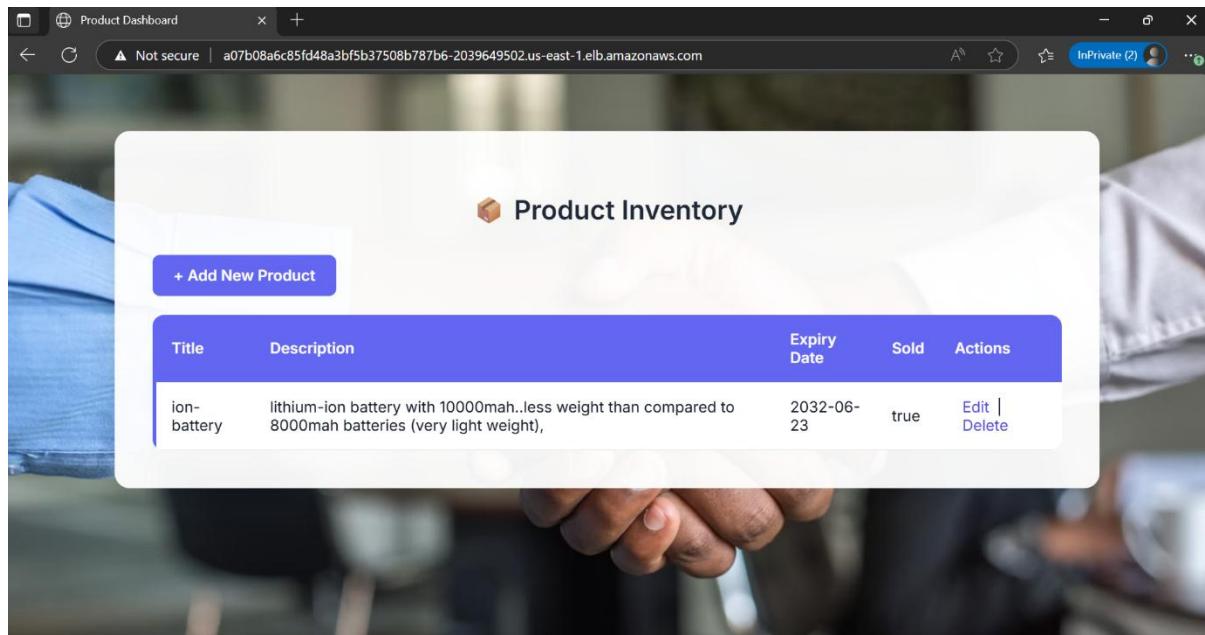
NAME                TYPE      CLUSTER-IP       EXTERNAL-IP
service/kubernetes  ClusterIP  172.20.0.1      <none>
service/todo-app-service  LoadBalancer  172.20.237.149  a07b08a6c85fd48a3bf5b37508b787b6-2039649502.us-east-1.elb.amazonaws.com  PORT(S)        AGE
                                                               443/TCP        140m
                                                               80:32491/TCP  93s

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/todo-app  2/2     2           2           93s

NAME            DESIRED   CURRENT   READY   AGE
replicaset.apps/todo-app-78675f74c8  2         2         2         93s
root@master:/home/ubuntu#
```

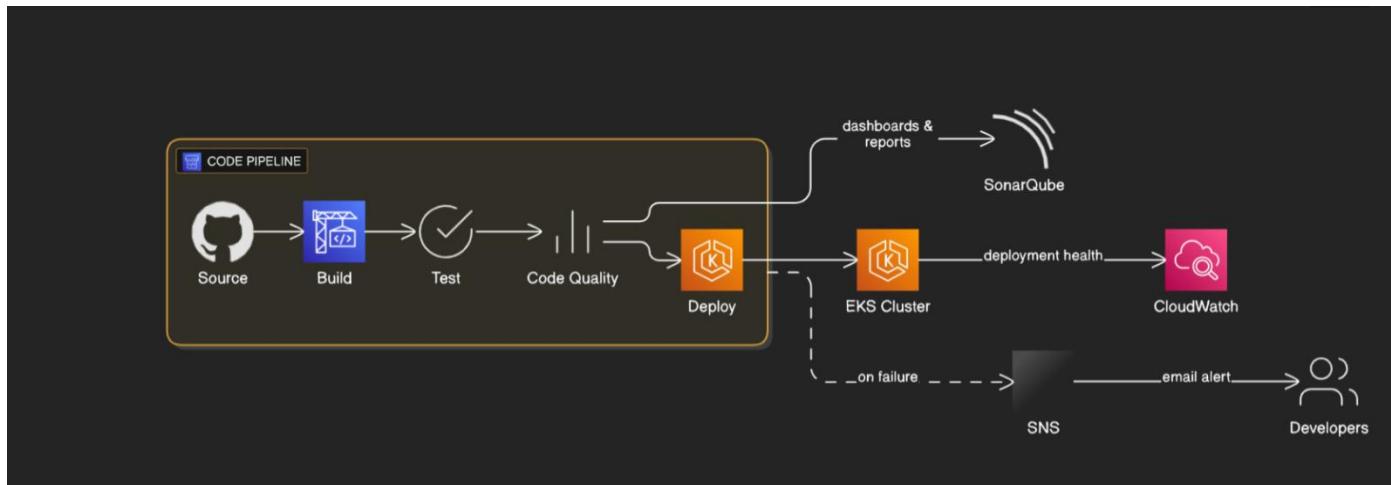
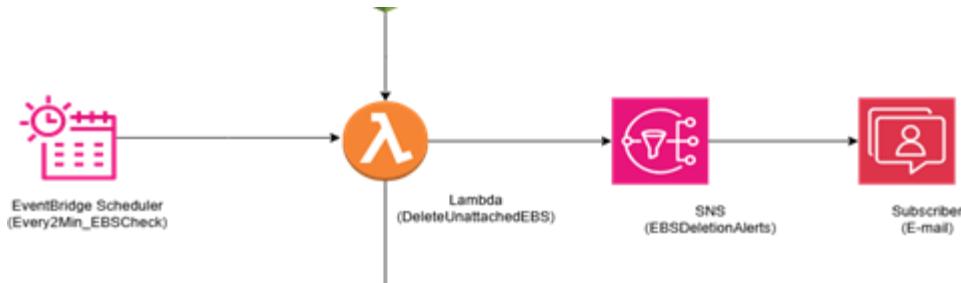
External ip is provided by aws ...copy the load balancer and access into the external browser

Successfully deployed the application rds database also connected successfully



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

If pipeline fails immediately we can get email notification



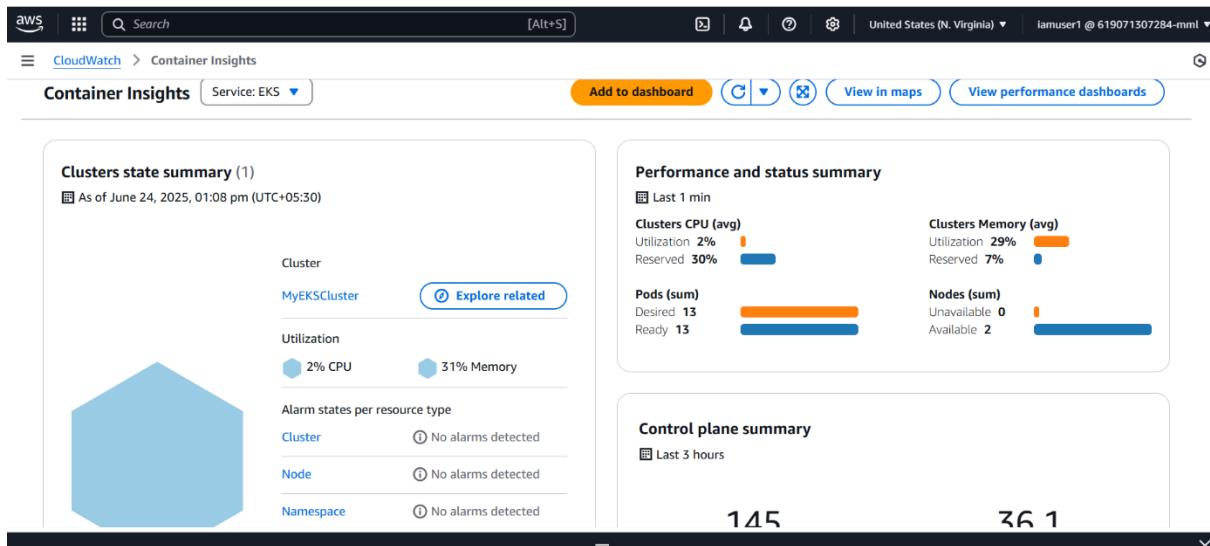
Any developers makes any changes ..pipeline will trigger and run this all process continues integration and continues deployment..

Cloud watch monitoring :

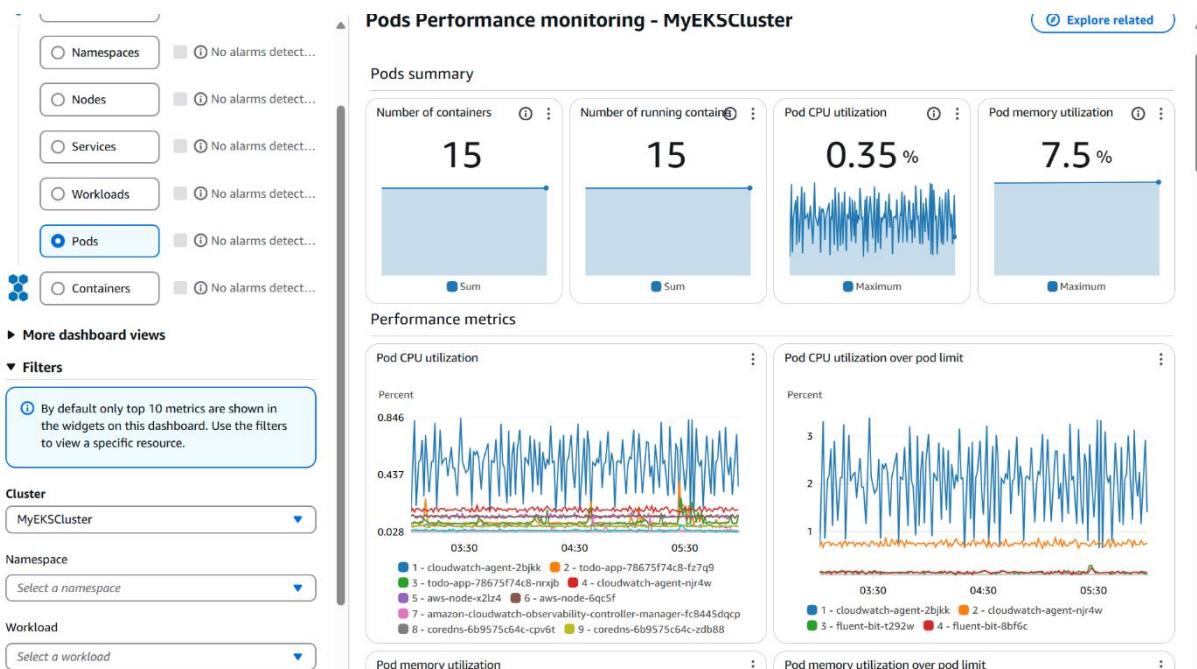
I have configured a **eks cluster** into cloud watch monitoring

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

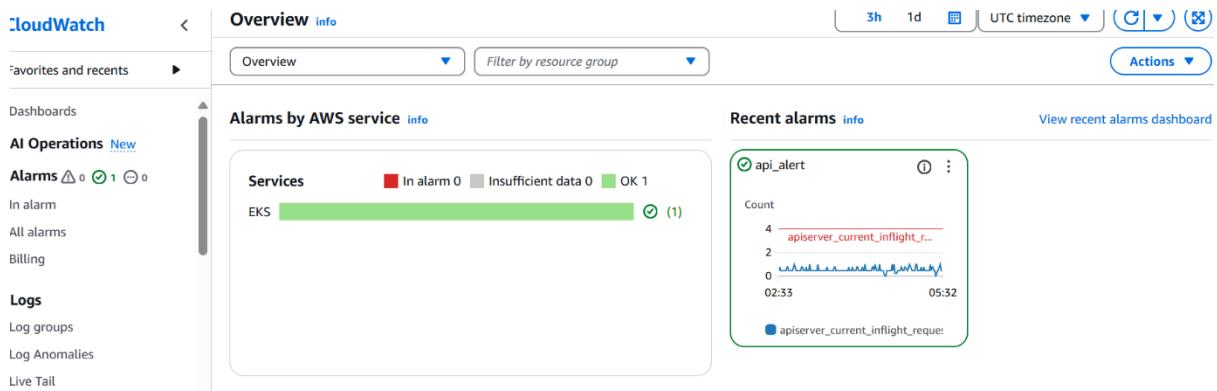
We can set alarms for this resource



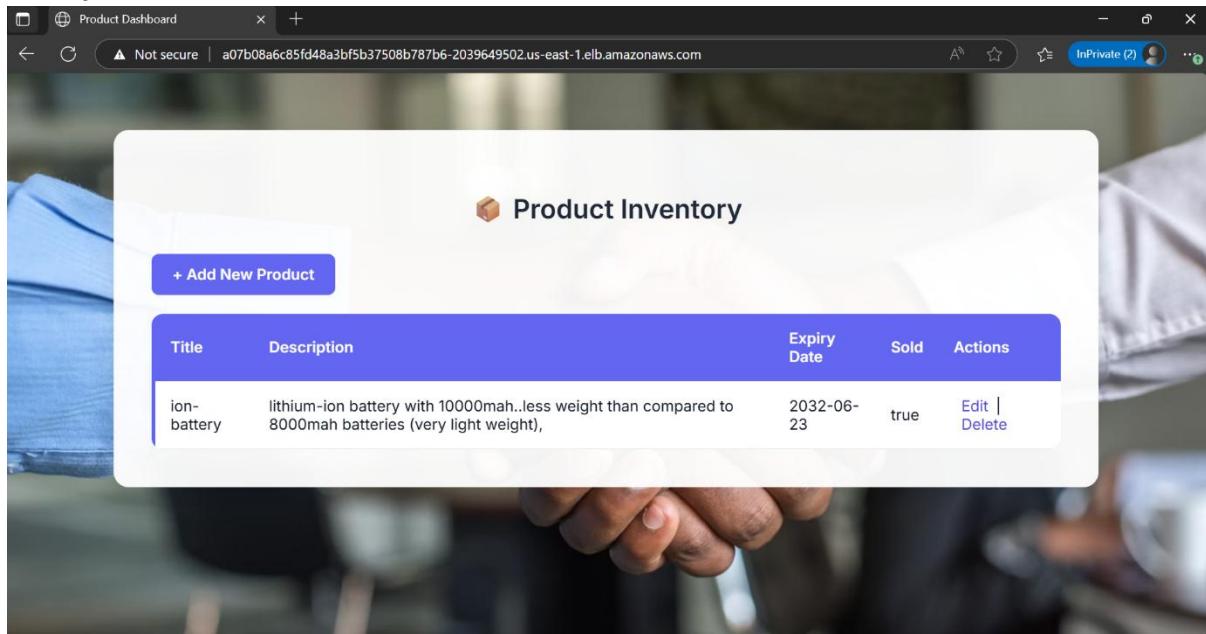
es



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.

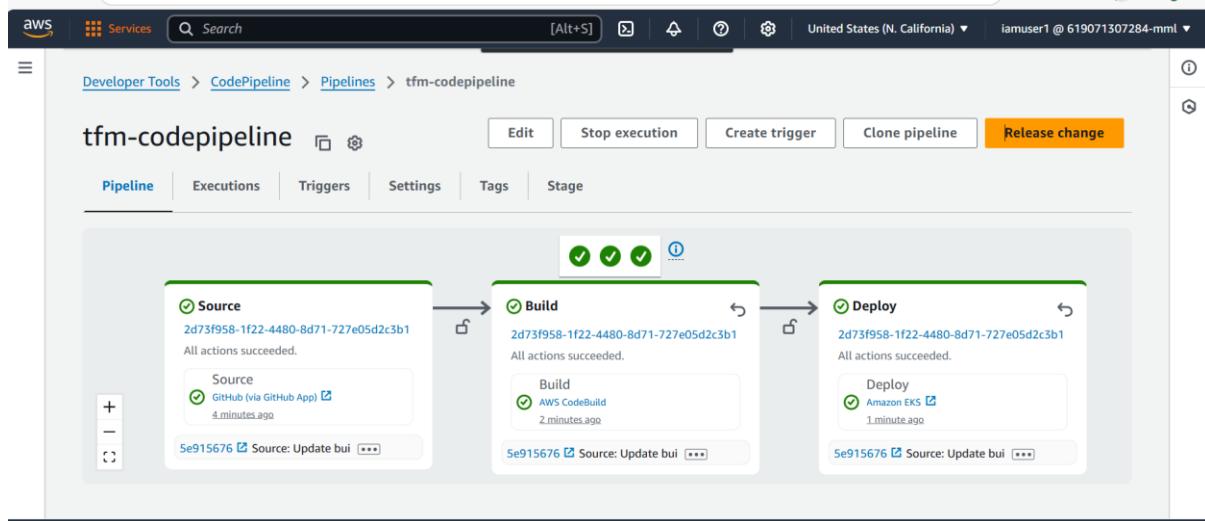
Core architecture is deployed using terraform

```
20250624095611546200000004]
aws iam role policy attachment.eks_node_role_attachments["AmazonEC2ContainerRegistryReadOnly"] Refreshing state... [id=EKSNodeGroupRoleTerraform-20250624095611389300000001]
aws iam role policy attachment.eks_cluster_role_attachments["AmazonEKSClusterPolicy"] Refreshing state... [id=EKSClusterRoleTerraform-20250624095611390300000002]
aws iam role policy attachment.eks_cluster_role_attachments["AmazonEKSVPCResourceController"] Refreshing state... [id=EKSClusterRoleTerraform-20250624095611611000000006]
aws iam role policy attachment.eks_cluster_role_attachments["AdministratorAccess"] Refreshing state... [id=EKSClusterRoleTerraform-20250624095611635200000007]
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-00fcdd8619e45b1067]
aws_route_table_association.public[0]: Refreshing state... [id=rtbassoc-04099c52cddcb8b52]
aws_route_table.private: Refreshing state... [id=rtb-0540dba4cc57de429]
aws_route_table_association.private[0]: Refreshing state... [id=rtbassoc-0e385b9771b4358c7]
aws_db_instance.mysql: Refreshing state... [id=db-OLDKPH7TPNWB5ARDAMMUIJ5EOQ]
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

The screenshot shows a GitHub repository page for "Capstone-Final-tf". The "buildspec.yaml" file is open. A specific command in the "post_build" section is highlighted in blue:

```
34     - docker tag $REPO_NAME:$IMAGE_TAG $IMAGE_URI
35     - cd ..
36
37   post_build:
38     commands:
39       - echo Pushing image to Amazon ECR...
40       - docker push $IMAGE_URI
41       - echo Scanning image with Trivy...
42       - trivy image $IMAGE_URI --severity CRITICAL,HIGH --format json --output trivy-report.json || true
43       - echo Preparing Kubernetes manifests...
44       - mkdir -p k8s-out
45       - sed "s|IMAGE_PLACEHOLDER|$IMAGE_URI|g" k8s/deployment.yaml > k8s-out/deployment.yaml
46       - cp k8s/service.yaml k8s-out/service.yaml
47       - mv trivy-report.json k8s-out/
48       - echo Done preparing artifacts.
49
50   artifacts:
51     base-directory: k8s-out
52     files:
53       - deployment.yaml
54       - service.yaml
55       - trivy-report.json
```

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

Argo CD set up independently for GitOps-style deployment:

The screenshot shows the Argo CD interface for the 'budgetledger app'. The top navigation bar includes 'APPS', 'Synced', 'Sync Status', 'History and Rollback', 'Delete', and 'Refresh'. On the left, there's a sidebar with icons for Home, Applications, Workspaces, and Logs. The main area has tabs for 'APP HEALTH' (Healthy), 'SYNC STATUS' (Synced to HEAD (29b6424)), and 'LAST SYNC' (Sync OK to 29b6424). Below these, a deployment graph shows the flow from the 'budgetledger app' pod through a 'budgetledger service' to a 'budgetledger' deployment, which then connects to 8 Replicasets. The graph includes timestamps like '13 hours' and '13 hours, 40m'.

For manual check:

```
aws CloudShell Search [Alt+S] United States (N. California) iamuser1 @ 619071307284-mm1 ▾
NAME           READY   STATUS    RESTARTS   AGE
pod/todo-app-6d47b5d6bc-9jth9   1/1     Running   0          41h
pod/todo-app-6d47b5d6bc-ncnr6   1/1     Running   0          41h

NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
(s)   AGE
service/kubernetes   ClusterIP   172.20.0.1    <none>        443/
TCP    2d
service/todo-app-service   LoadBalancer  172.20.142.108  a6c7dfcc782e8489ca09c14a2b34ebf5-1630131080.us-west-1.elb.amazonaws.com  80:3
1551/TCP  2d

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/todo-app   2/2     2           2           2d

NAME           DESIRED   CURRENT   READY   AGE
replicaset.apps/todo-app-5ffd46b7c  0         0         0       46h
replicaset.apps/todo-app-6d47b5d6bc  2         2         2       41h
replicaset.apps/todo-app-6fcf77bf7d  0         0         0       47h
replicaset.apps/todo-app-76b7746c5   0         0         0       2d
replicaset.apps/todo-app-78d4c6f9d5  0         0         0       42h
root@ip-10-0-0-178:/home/ubuntu#
```

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

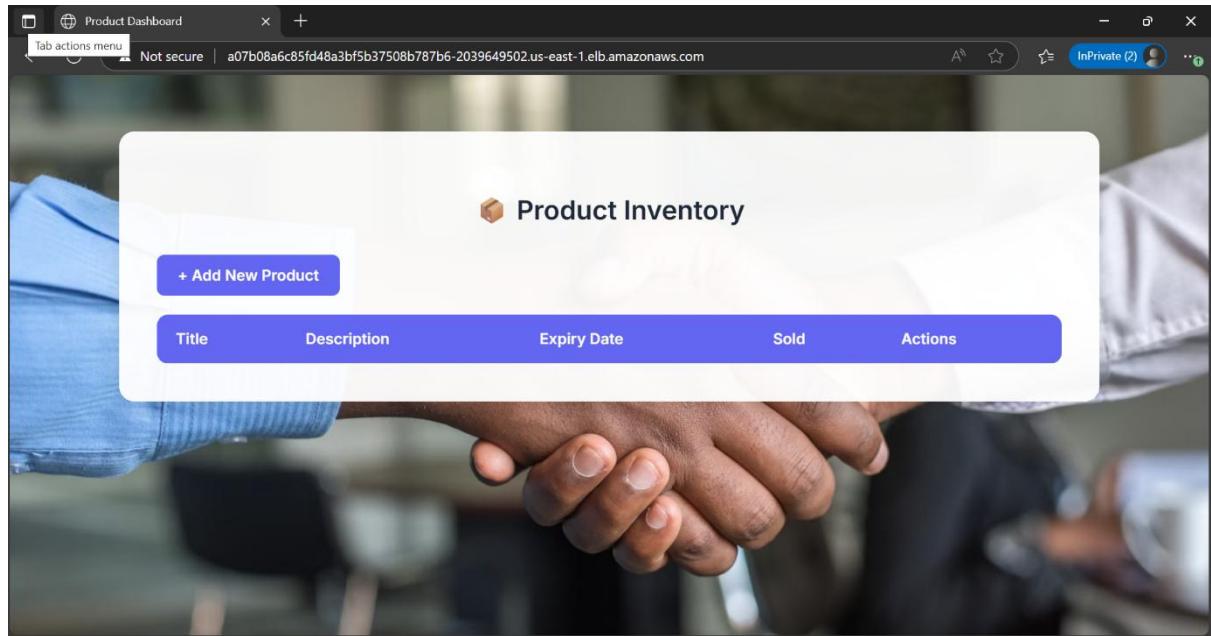
```
aws CloudShell Search [Alt+S] United States (N. Virginia) iamuser1 @ 619071307284-mm1 ▾
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| todo |
+-----+
rows in set (0.00 sec)

mysql> use todo;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from to_do;
+----+-----+-----+-----+
| id | completed | description | due_date | title |
+----+-----+-----+-----+
| 7 | 0x01 | new list | 2025-06-26 | test1 |
+----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

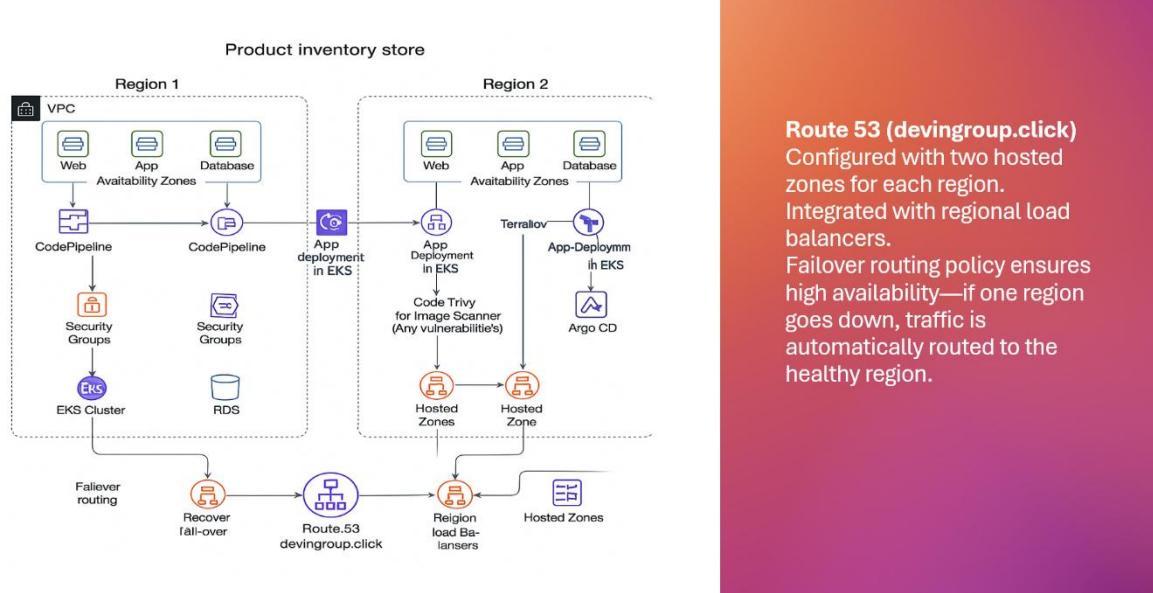
Accessing into externe browser:



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.



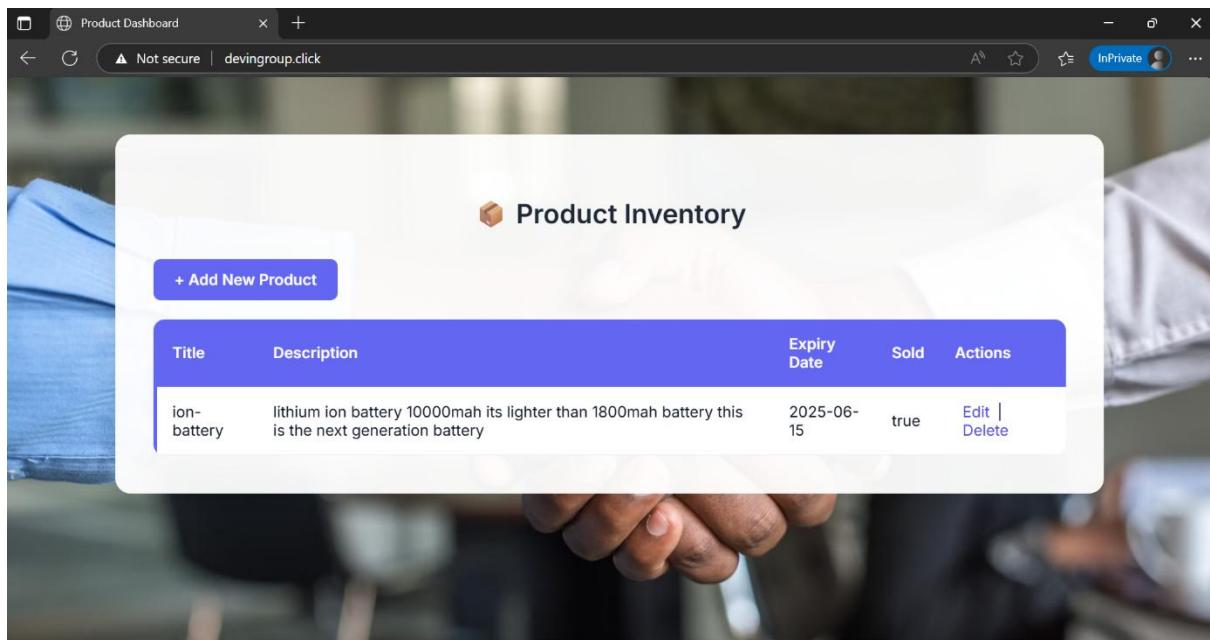
I have created a hosted zone (devingroup.click)

Created health check for 2 region load balancers

And created a records for this 2 load balancer end points

ID	Name	Details	Status in last 24 hours	Current status	Alarms	Actions
096287a1...	domaintfm_...	http://a6c7...	<div style="width: 100%; background-color: green;"></div>	Healthy	Non	⋮
4599855e-2...	2 hc	http://54.1...	<div style="width: 0%; background-color: red;"></div>	Unhealthy	Non	⋮
7b5cde20-a...	primary_hc	http://34.2...	<div style="width: 0%; background-color: red;"></div>	Unhealthy	Non	⋮
e517f166-1...	domain_he...	http://a07b...	<div style="width: 100%; background-color: green;"></div>	Healthy	Non	⋮

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.