MY-PLAYERSLIST

CAPSTONE PROJECT

GITHUB-URL: https://github.com/SanjidakramN/capstone-project

TOPICS:

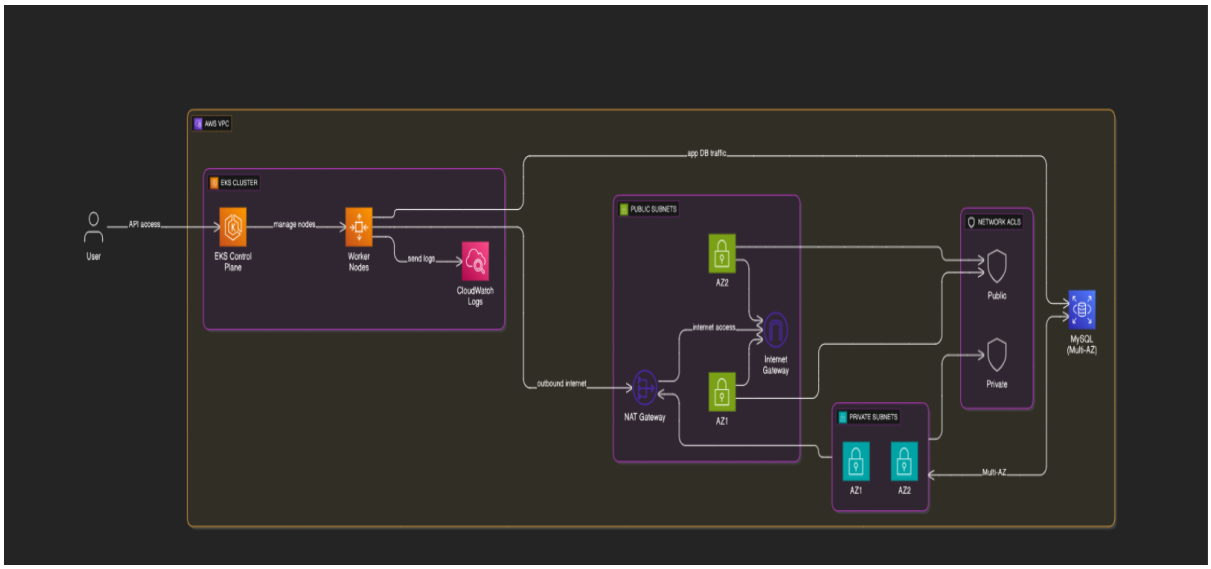
- **CLOUDFORMATION - US-EAST-1**
- **TERRAFORM - US-WEST-2**
- **ROUTE 53**
- **CLOUD WATCH**
- **ALARM & SNS NOTIFICATION**
- **CodeCommit**
- **SONARQUBE**

OVERVIEW:

This project involves deploying a Three-Tier Web Application (ReactJS frontend, NodeJS backend, and MongoDB database) on Amazon EKS. The application is containerized using Docker and images are stored in Amazon ECR.

SonarQube is integrated for code quality checks during the CI/CD process. Amazon CloudWatch monitors application health, with alarms triggering SNS notifications for issues like pod crashes. Amazon Route 53 is used to route traffic to the frontend via a custom domain, enabling external access.

ARCHITECTURE OF APPLICATION



1. User Access:

   Users interact with the application via API requests sent to the EKS Control Plane.

2. EKS Cluster:

   . EKS Control Plane manages the Kubernetes Worker Nodes.

   . Worker nodes run the application workloads (frontend/backend service)

3. CloudWatch Logs:

   . Worker nodes send logs to **Amazon CloudWatch** for monitoring and observability.

4. VPC & Subnets:

   . All resources are within a single **AWS VPC**.

   . Split into **Public Subnets** (AZ1, AZ2) and **Private Subnets** (AZ1, AZ2) for high availability.

5. Internet Connectivity:

   . **Internet Gateway** allows public subnets internet access.\

   . **NAT Gateway** enables private subnets to reach the internet securely for updates.

6. Network ACLs:

   . Network ACLs control traffic at the subnet level for both public and private traffic filtering.

7. Database Layer:

- Uses **Amazon documentDB for MySQL** with **Multi-AZ** deployment for high availability and reliability.

.The application communicates with the database through private networking.

1 CloudFormation via CodePipeline

.us-east-1

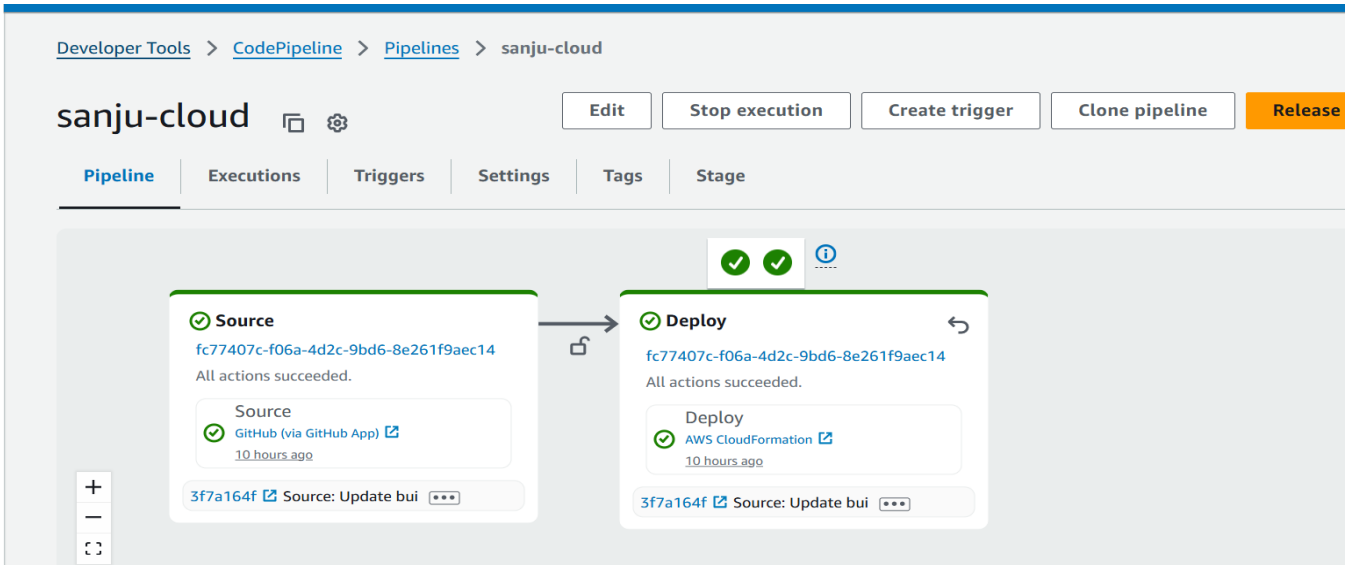
.git-url: https://github.com/SanjidakramN/capstone-project/blob/main/cloudFormation.yaml

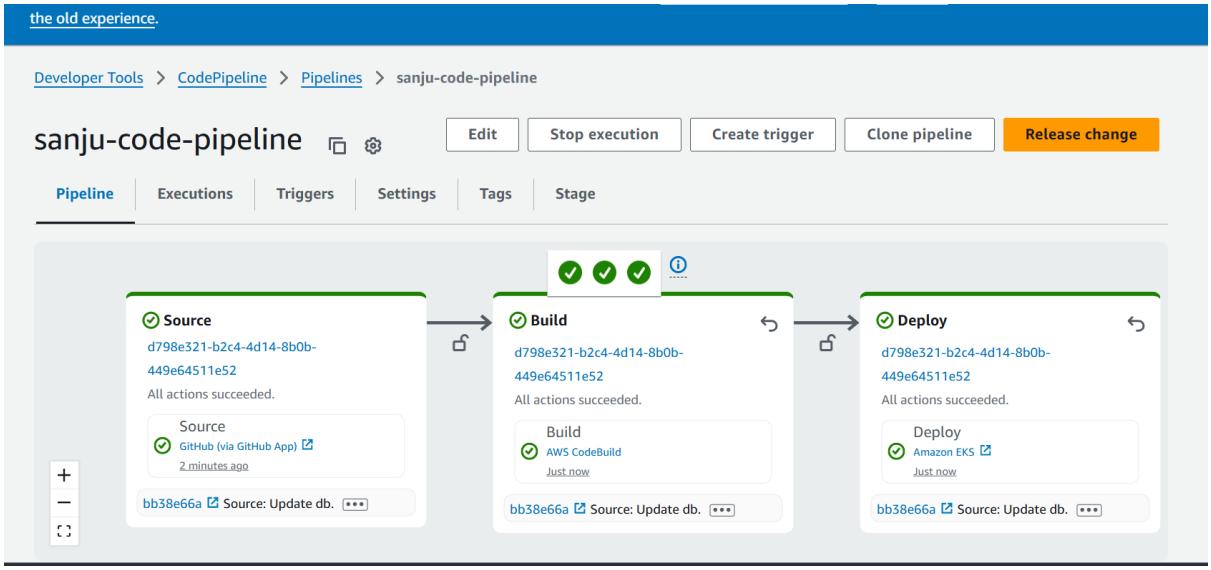
. Uses **CloudFormation templates** to provision infrastructure (VPC, EKS, DB, etc.).

. Whenever code is pushed, **CodePipeline** triggers:

1. **Source stage** fetches code (e.g., from GitHub or CodeCommit).
2. **Build stage** runs validations (e.g., CodeBuild).
3. **Deploy stage** uses **CloudFormation** to create/update AWS resources.
4. Ensures **repeatable**, **version-controlled**, and **automated** infrastructure setup.

Screenshots of running codePipelines





.aws eks update-kubeconfig --region us-east-1 --name SanjuEKSCluster

.export AWS_REGION=us-east-1

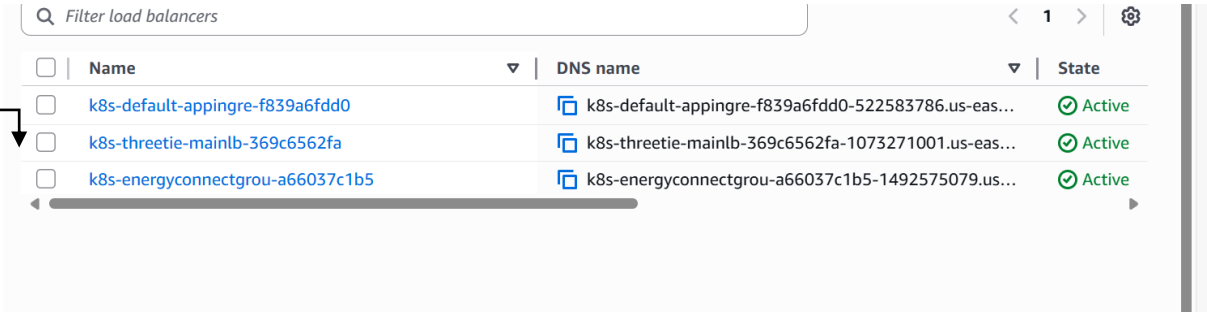
kubectl get all pod –n three-tier

```
root@ip-172-31-23-245:/home/ubuntu# kubectl get all -n three-tier
NAME                                 READY   STATUS    RESTARTS   AGE
pod/api-5d966c6fdc-j7dsj             1/1     Running   0          27h
pod/api-5d966c6fdc-wqjvh             1/1     Running   0          27h
pod/frontend-55bbdc94fd-kspmk        1/1     Running   0          27h
pod/mongodb-799f9bc559-gp4zx         1/1     Running   0          28h
pod/restart-test-56bc999b9-4sl9s     1/1     Running   0          101m

NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
service/api           ClusterIP   172.20.60.111    <none>        3500/TCP    28h
service/frontend      ClusterIP   172.20.250.206   <none>        3000/TCP    28h
service/mongodb-svc   ClusterIP   172.20.101.147   <none>        27017/TCP   28h

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/api                  2/2     2            2           28h
deployment.apps/frontend             1/1     1            1           28h
deployment.apps/mongodb              1/1     1            1           28h
deployment.apps/restart-test         1/1     1            1           103m

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/api-5d966c6fdc                 2         2         2       27h
```

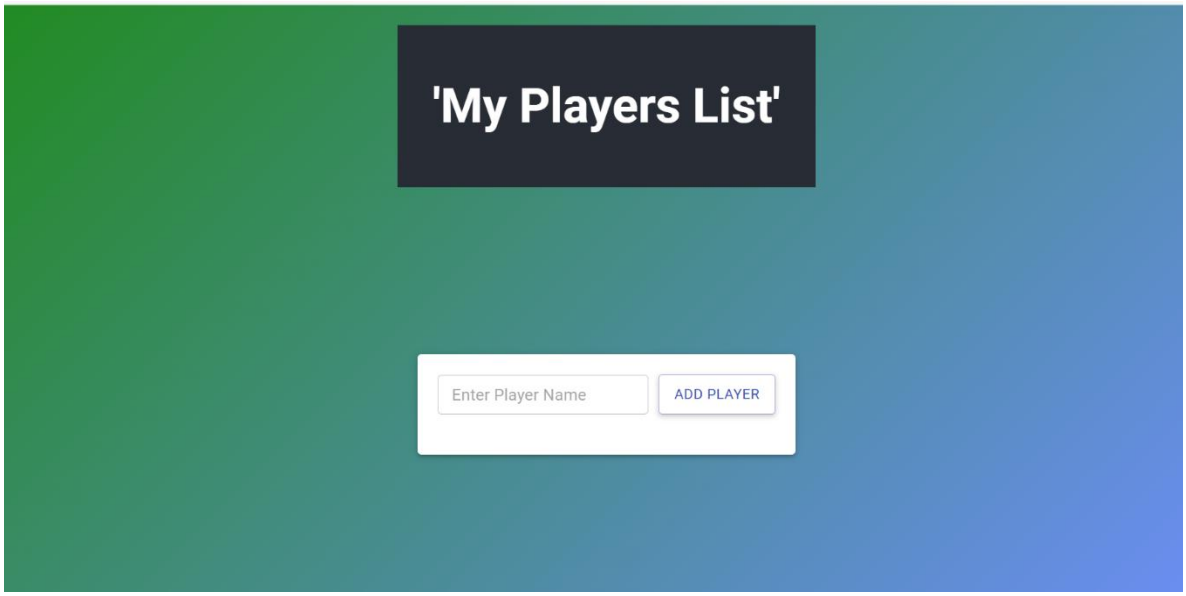
Loadbalancer:

. us-east-1

. k8s-threetie-mainlb-369c6562fa-1073271001.us-east-1.elb.amazonaws.com



. Getting output through loadbalancer



MongoDB with Amazon DocumentDB:

.Amazon DocumentDB is a fully managed document database service.

.Compatible with MongoDB APIs – supports MongoDB drivers/tools.

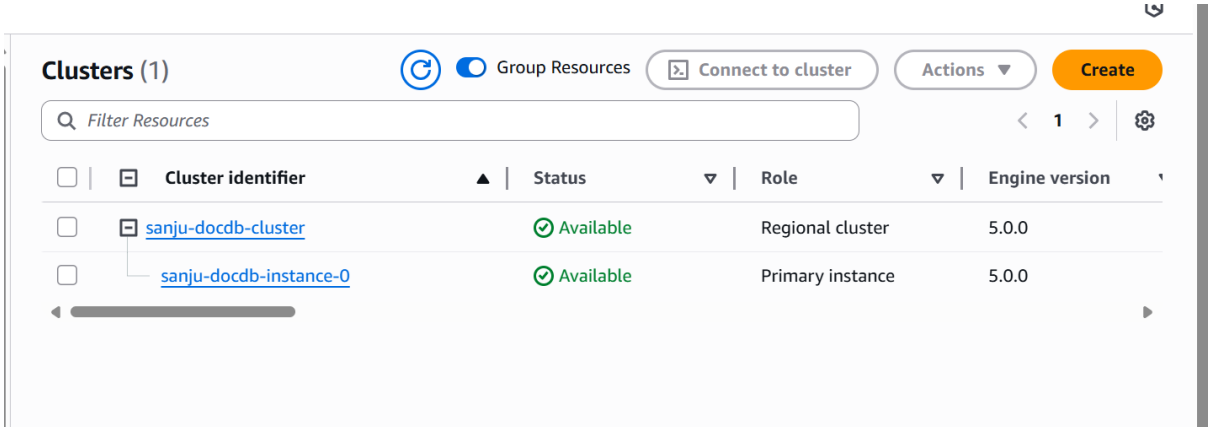
.Used to store JSON-like (BSON) documents.

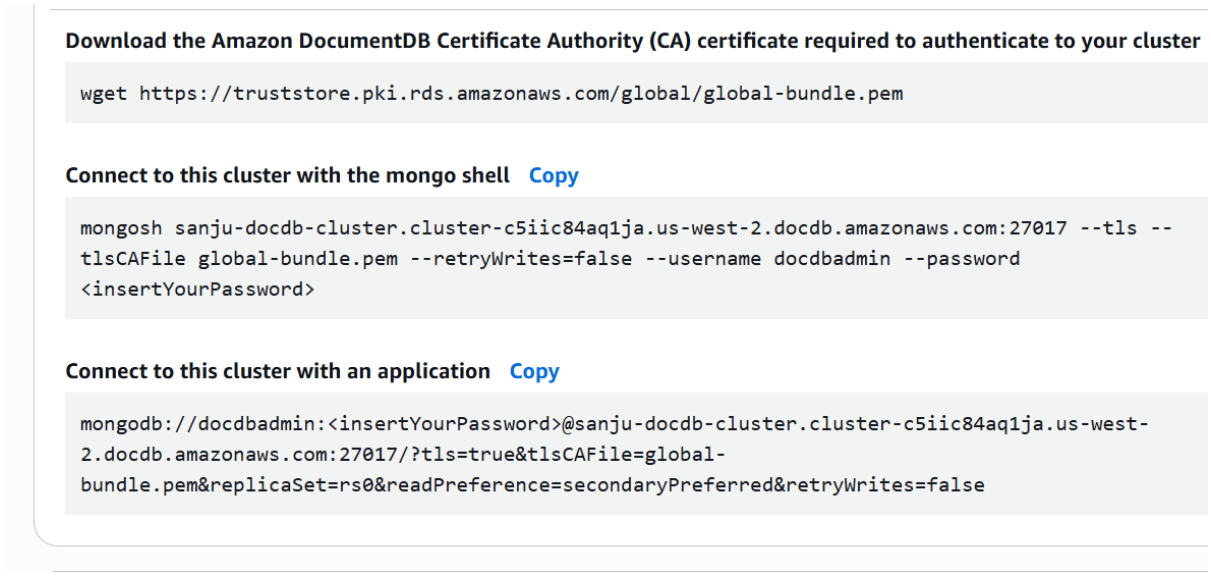
.Provides scalability, high availability, and backup support.

.Ideal for microservices needing schema flexibility.

.curl -o global-bundle.pem https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem

.mongosh "mongodb://Sanjidakram:Rabiyasanju@sanjudocdb-cluster.ch2c82wwifaa.us-east-1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-bundle.pem&retryWrites=false“





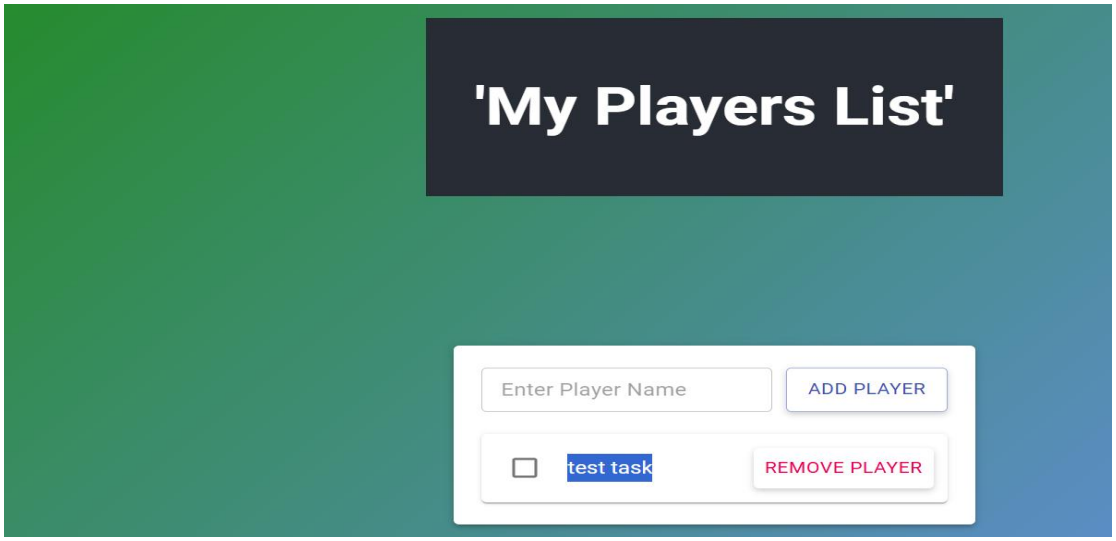
. show dbs

. use todo

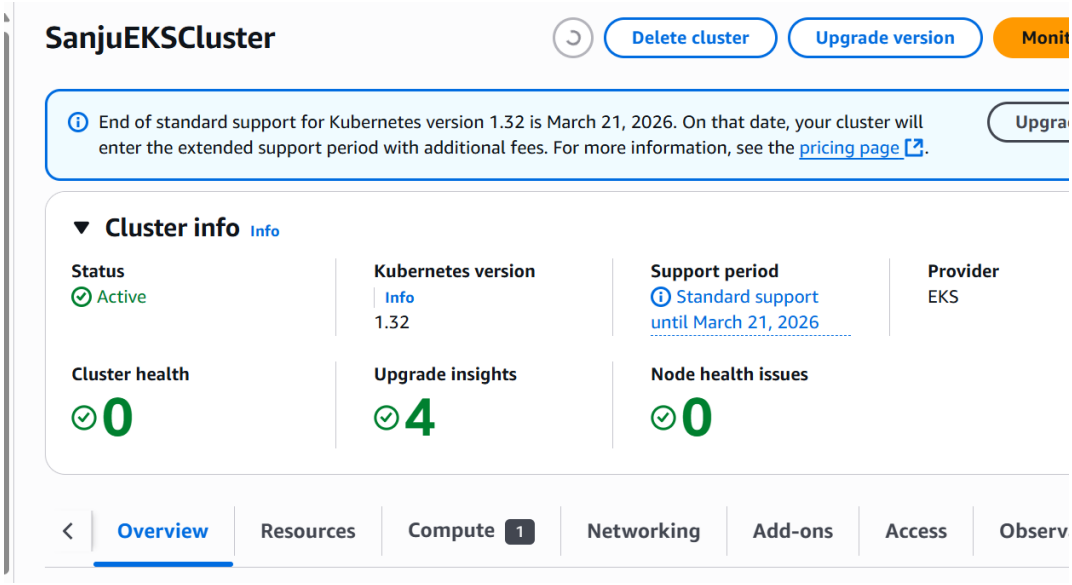
.show collections

.db.tasks.find().pretty()

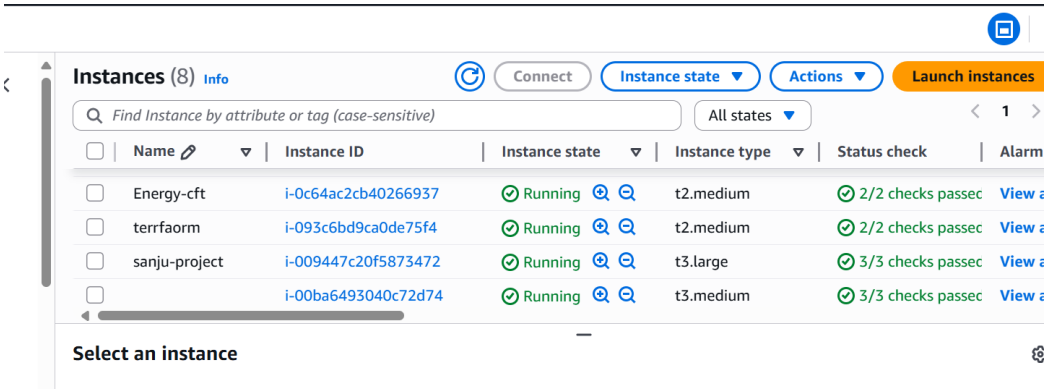
```
 3 |

s0 [direct: primary] todo> db.tasks.find().pretty()

 {
   _id: ObjectId('6858e91addea27a704baa8b9'),
   task: 'Test Task',
   completed: false
 }

s0 [direct: primary] todo>
```



Running EKSCluster



Running Instances



2. Terraform with CodePipeline for Infrastructure:

.us-west-2

. Infrastructure as Code (IaC): Terraform is used to define and version infrastructure in code format.

. **CodePipeline Integration:** Automates the deployment of Terraform templates using AWS CodePipeline.

Stages:

1. **Source Stage:** Fetches Terraform code from a repo (e.g., GitHub/CodeCommit).
2. **Build Stage:** Runs terraform init, plan, and apply via CodeBuild.

Provisioning: Automatically creates resources like **VPC, Subnets, EKS, IAM roles, RDS,DocumentDB** etc.

Benefits: Ensures **consistent, repeatable**, and **automated infrastructure deployment**.

.aws eks update-kubeconfig --region us-west-2 --name SanjuEKSCluster-terraform

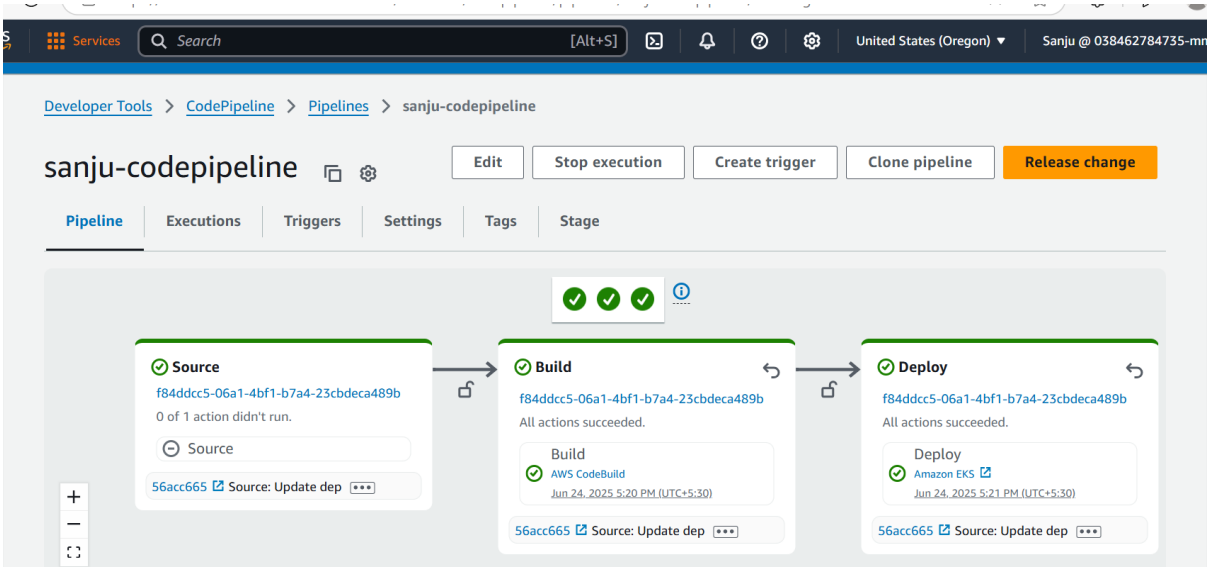
. export AWS_REGION=us-west-2

. kubectl get all –n three-tier

```
NAME                                READY   STATUS    RESTARTS   AGE
pod/api-5d966c6fdc-kf85p            1/1     Running   0          2d10h
pod/api-5d966c6fdc-pdhxt            1/1     Running   0          2d10h
pod/frontend-55bbdc94fd-67r8l       1/1     Running   0          2d10h
pod/mongodb-799f9bc559-n2xn5        1/1     Running   0          2d10h

NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
service/api             ClusterIP   172.20.50.213    <none>        3500/TCP    3d4h
service/frontend        ClusterIP   172.20.212.32    <none>        3000/TCP    3d4h
service/mongodb-svc     ClusterIP   172.20.60.98     <none>        27017/TCP   3d4h

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/api            2/2     2            2           2d10h
deployment.apps/frontend       1/1     1            1           2d10h
deployment.apps/mongodb        1/1     1            1           2d10h

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/api-5d966c6fdc            2         2         2       2d10h
replicaset.apps/frontend-55bbdc94fd       1         1         1       2d10h
replicaset.apps/mongodb-799f9bc559        1         1         1       2d10h
root@ip-172-31-23-245:/home/ubuntu#
```

Codebuild through terraform

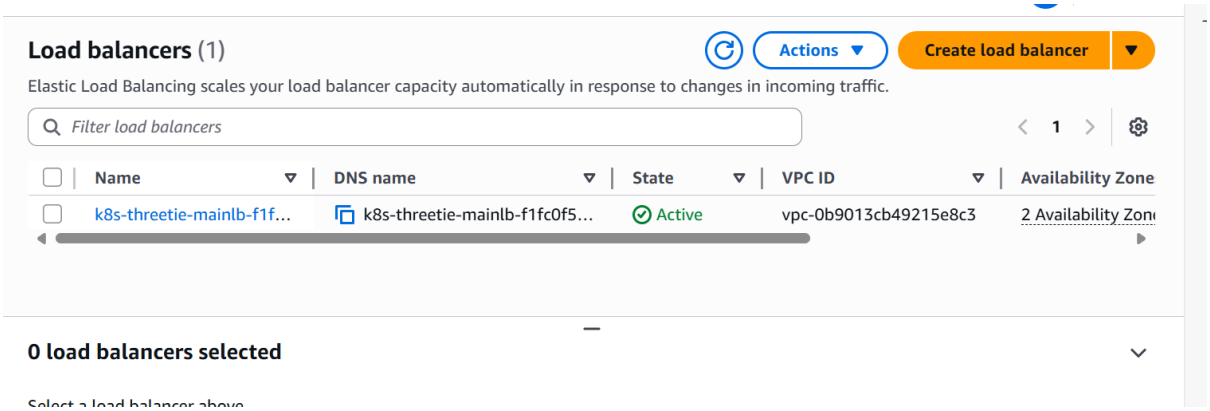
Codepipeline

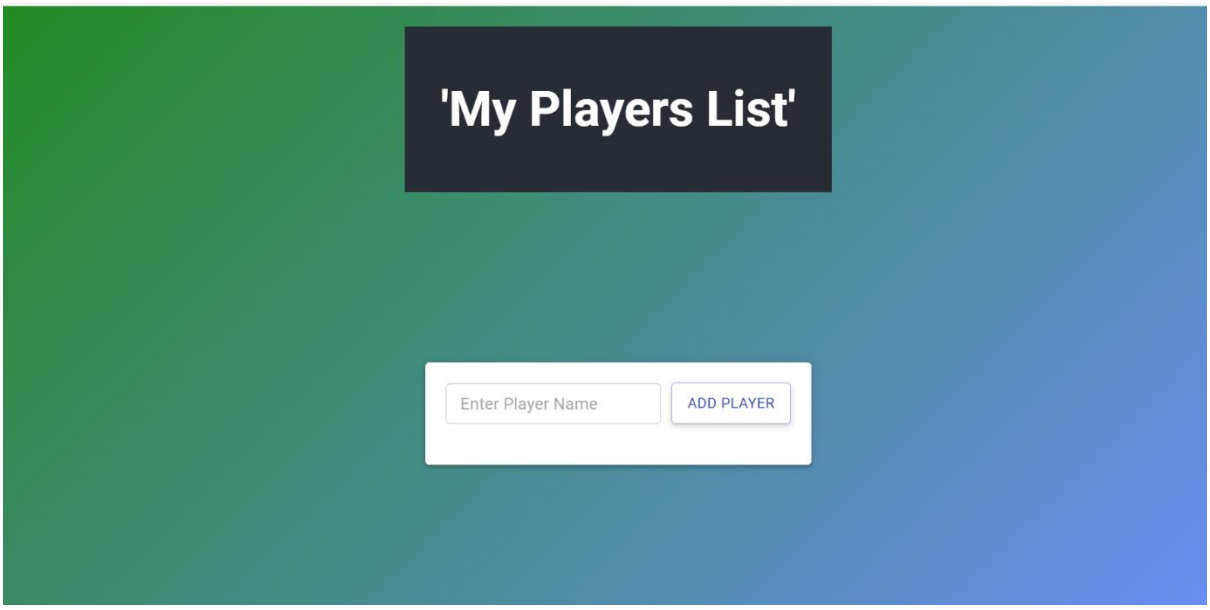


LoadBalancer – us-west-2:

Distributes Traffic: Automatically distributes incoming application traffic across multiple EC2 instances.

. k8s-threetie-mainlb-f1fc0f5c18-1847115127.us-west-2.elb.amazonaws.com



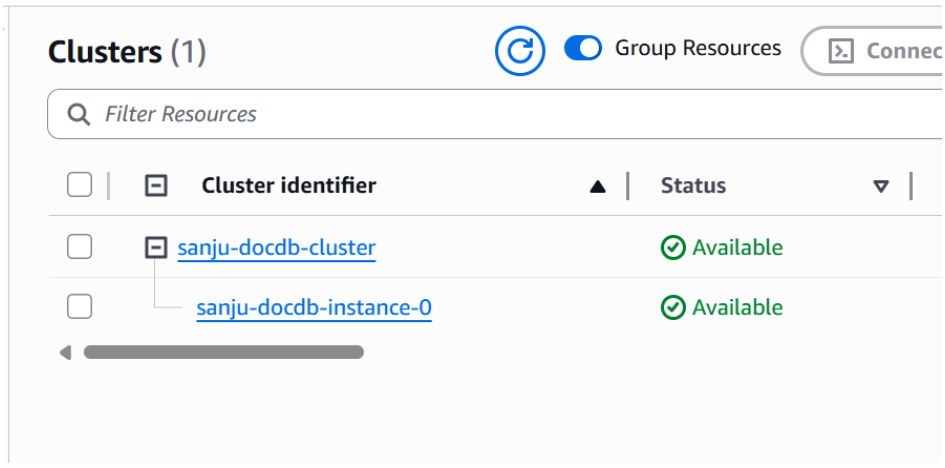


MongoDB with Amazon DocumentDB:

US-WEST-2

curl -o global-bundle.pem https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem

mongosh "mongodb://Sanjidakram:Rabiyasanju@sanjudocdb-cluster.ch2c82wwifaa.us-east-1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-bundle.pem&retryWrites=false“



Download the Amazon DocumentDB Certificate Authority (CA) certificate required to authenticate to your cluster Copy

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Connect to this cluster with the mongo shell Copy

```
mongosh sanju-docdb-cluster.cluster-c5iic84aq1ja.us-west-2.docdb.amazonaws.com:27017 --tls --
tlsCAFile global-bundle.pem --retryWrites=false --username docdbadmin --password
<insertYourPassword>
```

Connect to this cluster with an application Copy

```
mongodb://docdbadmin:<insertYourPassword>@sanju-docdb-cluster.cluster-c5iic84aq1ja.us-west-
2.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false
```

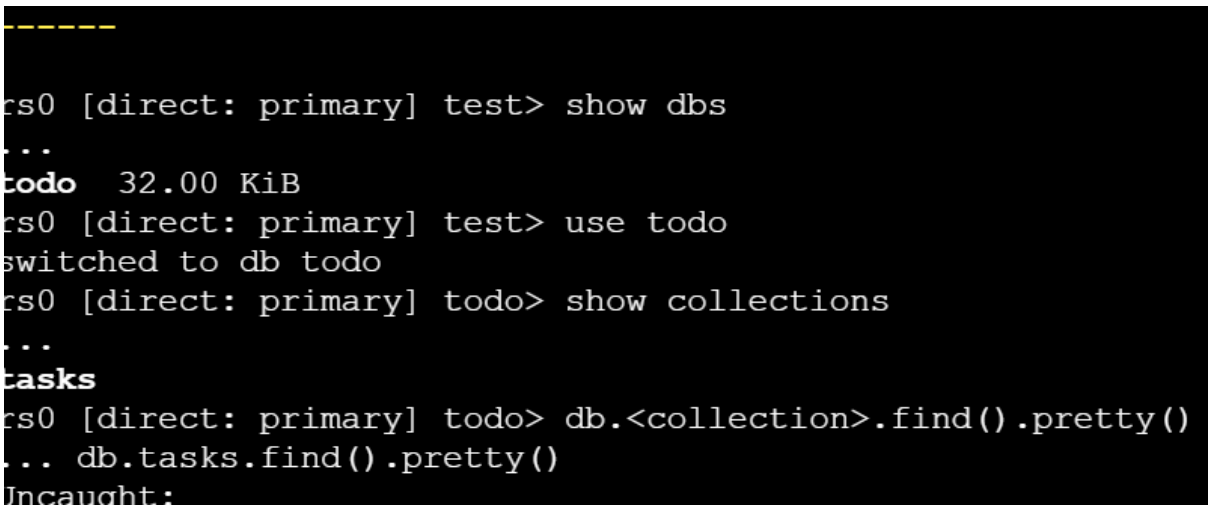
Security Groups (1)

. Show dbs

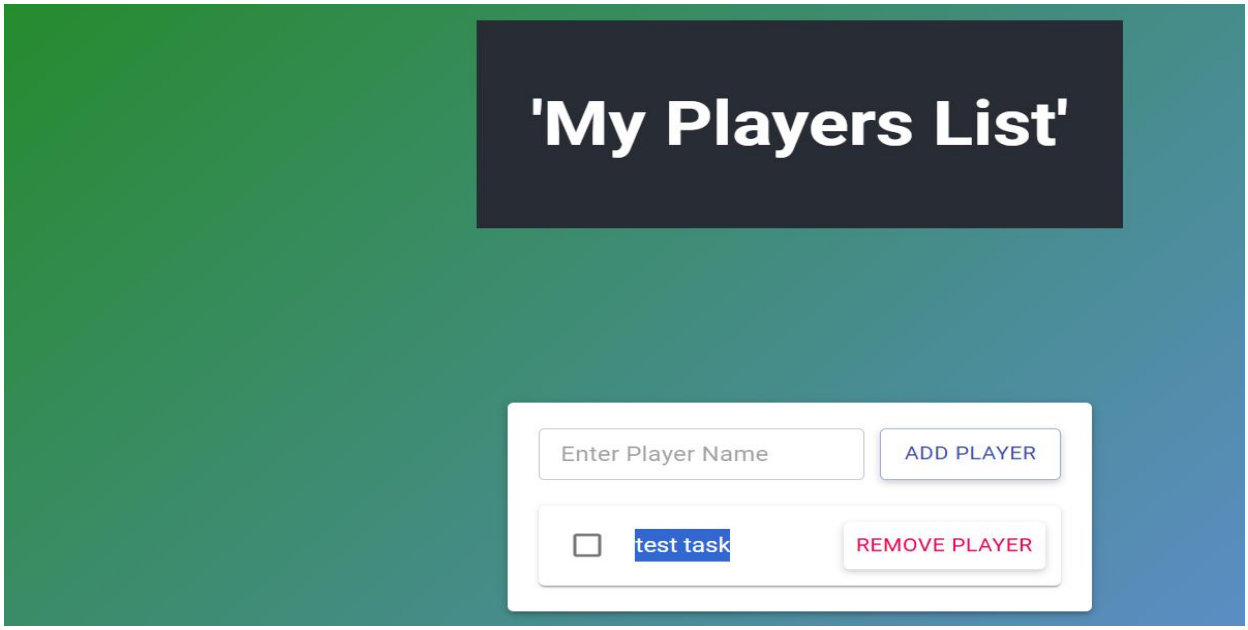
. use todo

. show collections

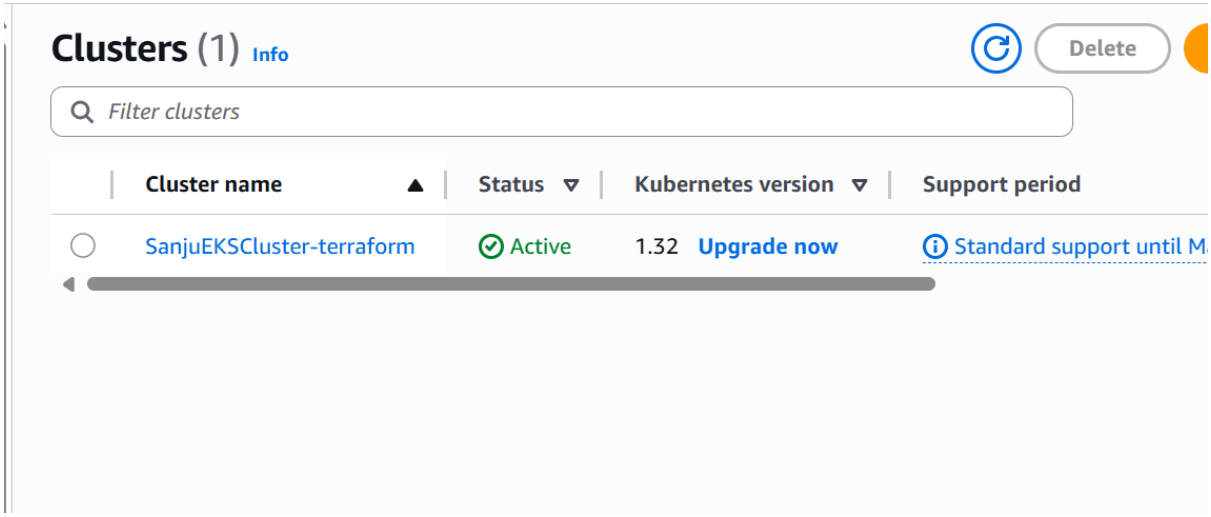
.db.tasks.find().pretty()



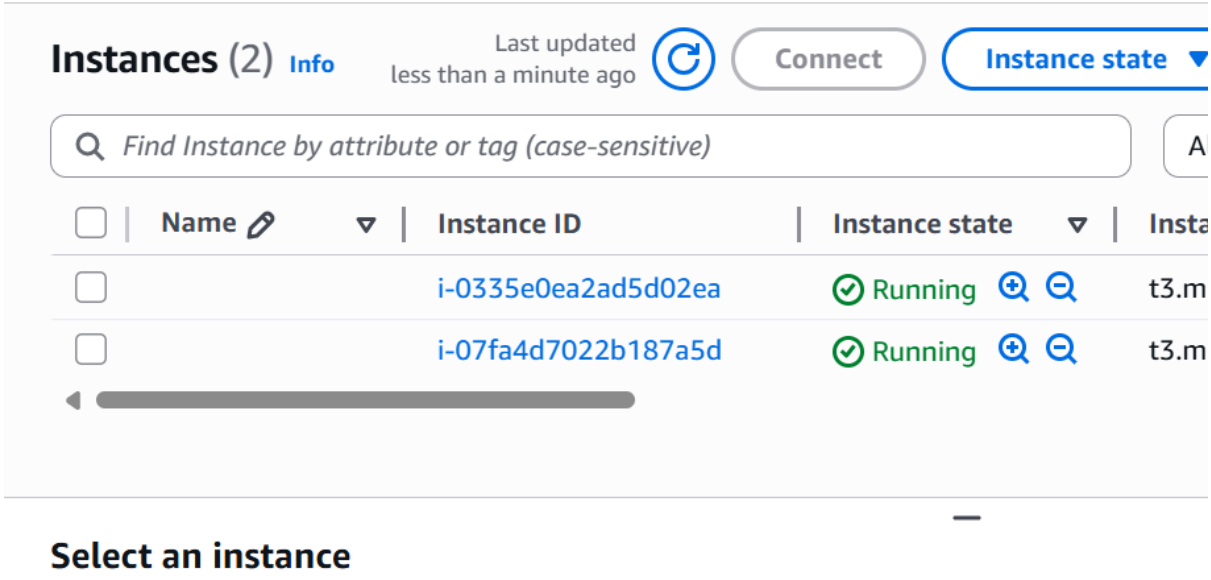
```
3
s0 [direct: primary] todo> db.tasks.find().pretty()
{
  _id: ObjectId('6858e91addea27a704baa8b9'),
  task: 'Test Task',
  completed: false
}
s0 [direct: primary] todo>
```



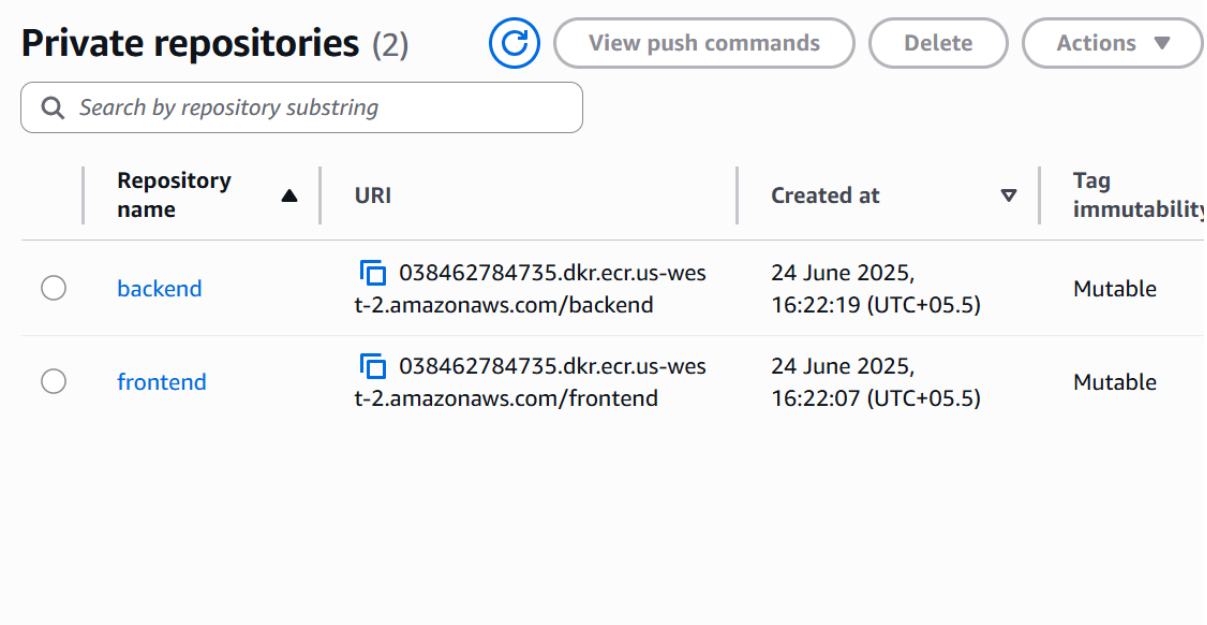
Cluster us-west-2



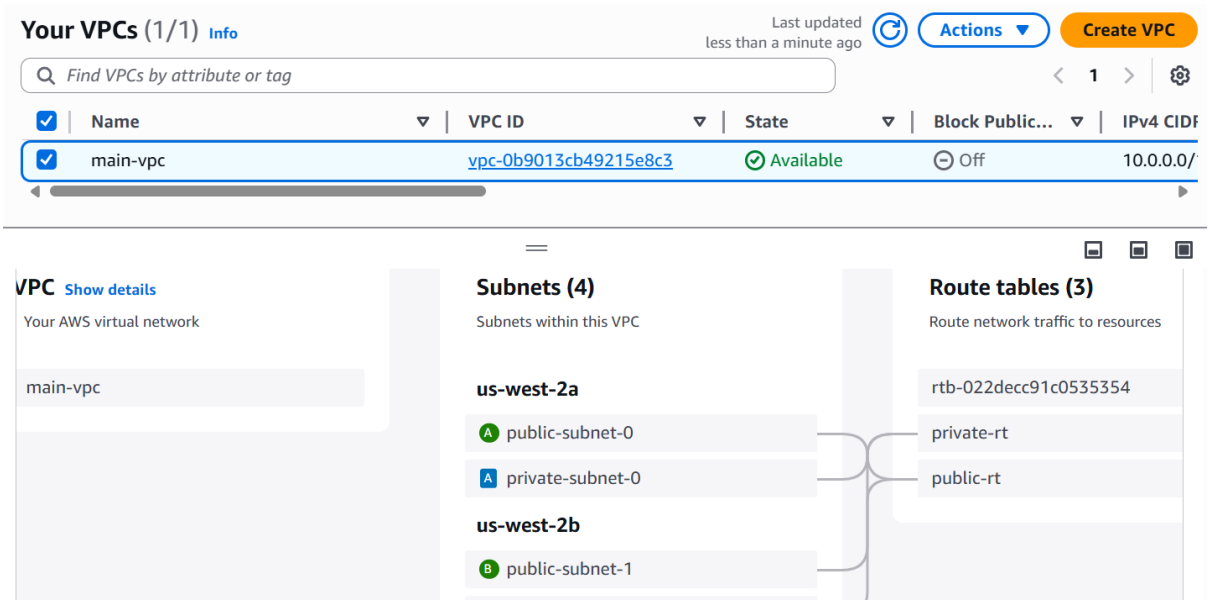
Instances:



ECR repositories



Vpc



Amazon Route 53 and Failover:

Route 53 is a DNS web service that routes user requests to AWS resources (like EC2, Load Balancer).

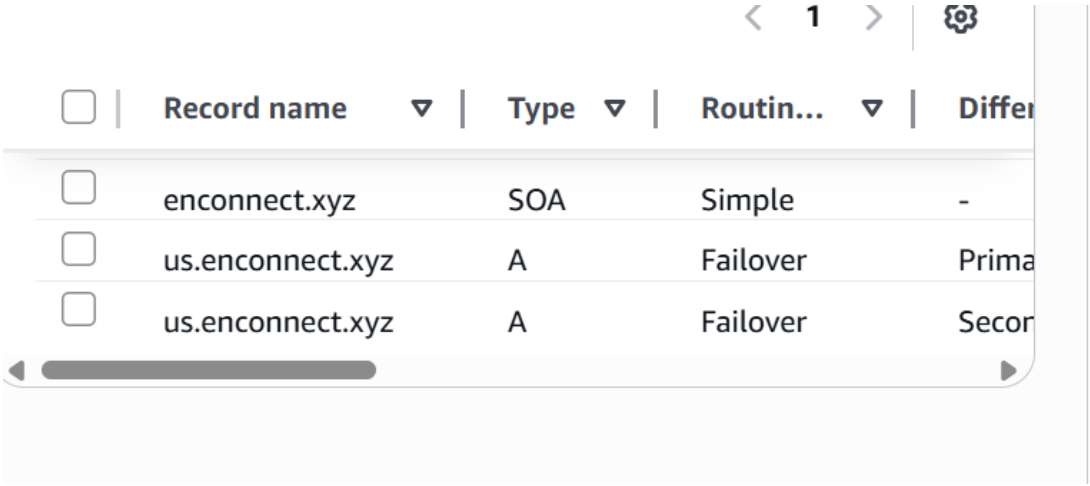
- **Use Case:** Ensures **high availability** for critical applications by reducing downtime.
- Failover Routing Policy:

- Used to automatically redirect traffic from a **primary** resource to a **secondary (backup)** resource if the primary fails.
- Works by configuring **health checks** to monitor endpoint health.

. enconnect.xyz

us.enconnect.xyz

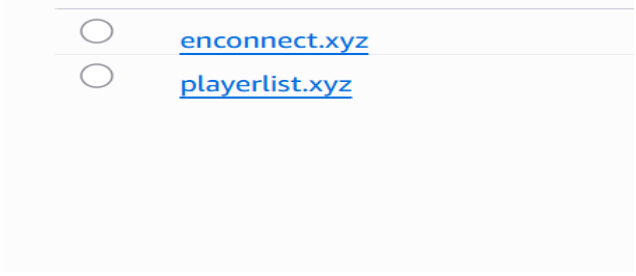
us.enconnect.xyz

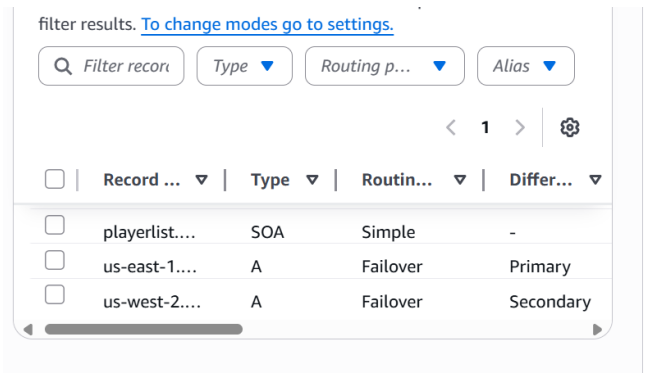


. playerlist.xyz

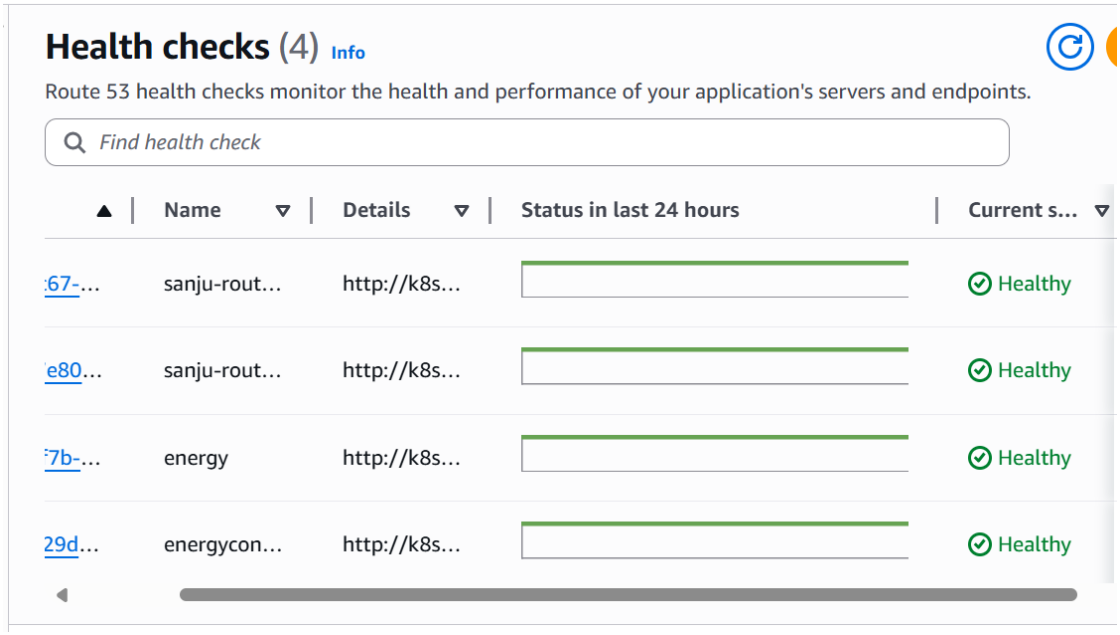
play.playerlist.xyz

play.playerlist.xyz





Healthcheck:



Amazon CloudWatch:

. Monitoring Tool: Tracks metrics, logs, and events for AWS resources and applications.

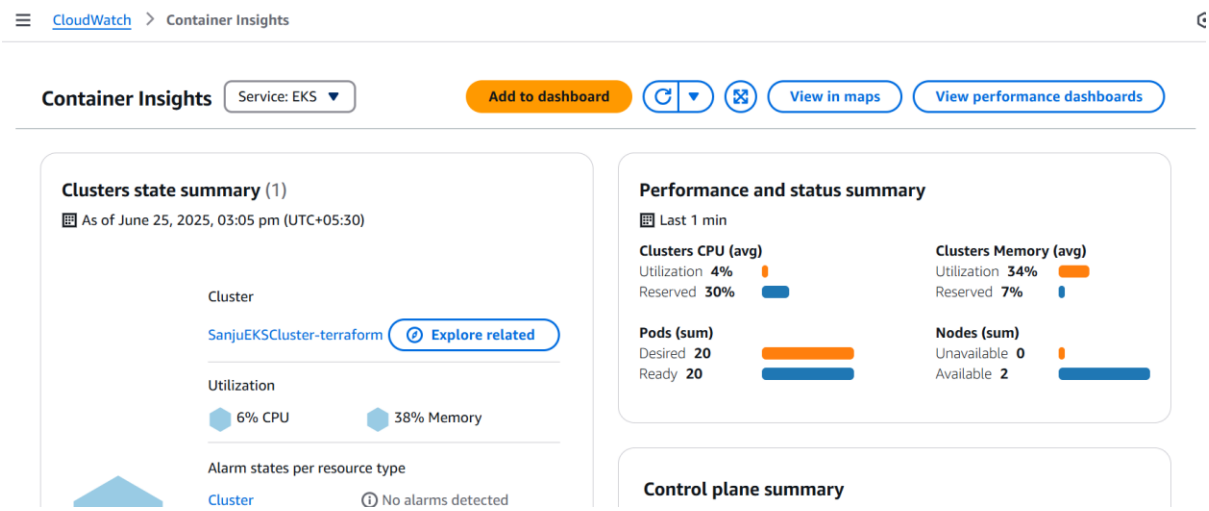
. **Real-Time Metrics:** Monitors CPU, memory, disk, and network usage of EC2, EKS, RDS, etc.

. **Log Collection:** Collects and stores logs from applications and containers

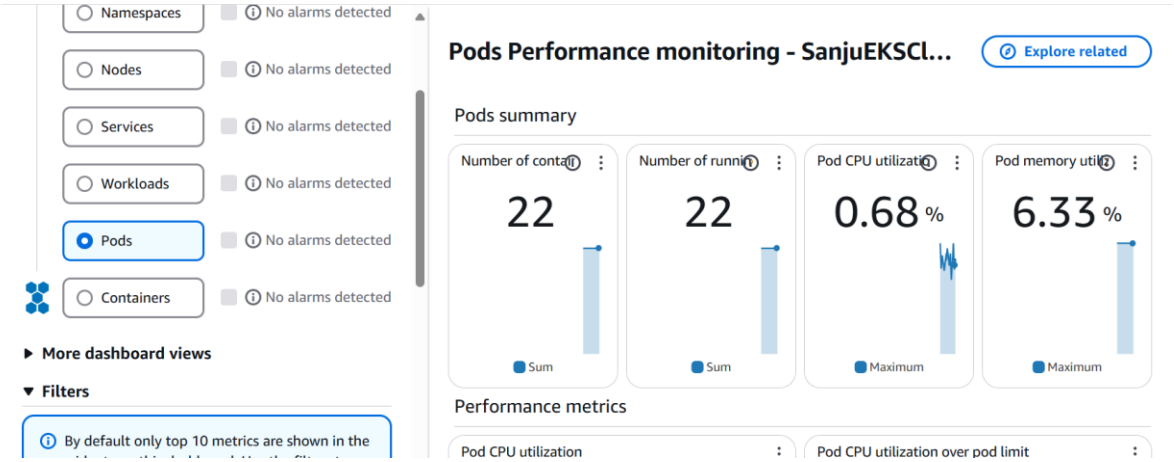
. **Alarms & Notifications:** Triggers alarms based on thresholds and integrates with **SNS** for alerts.

. **Dashboard Support:** Visualizes metrics using custom dashboards for better insights.

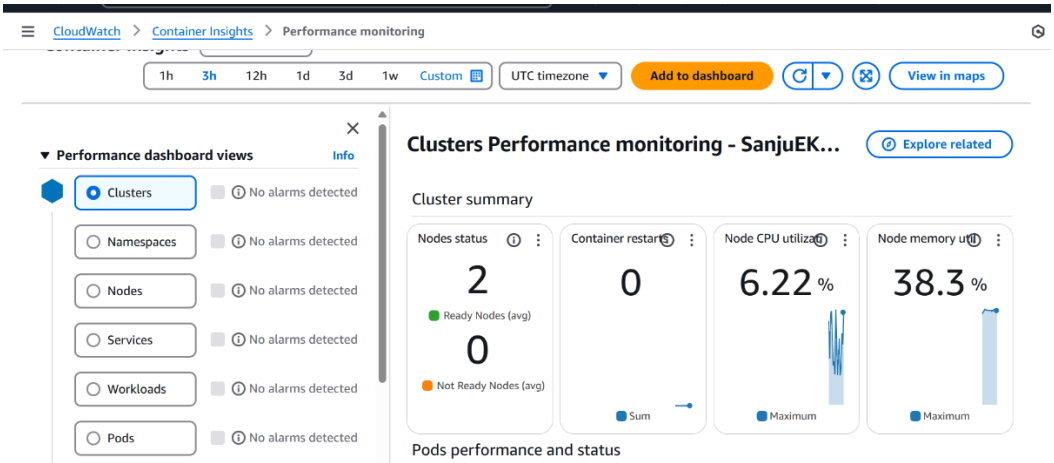
containerInsights:

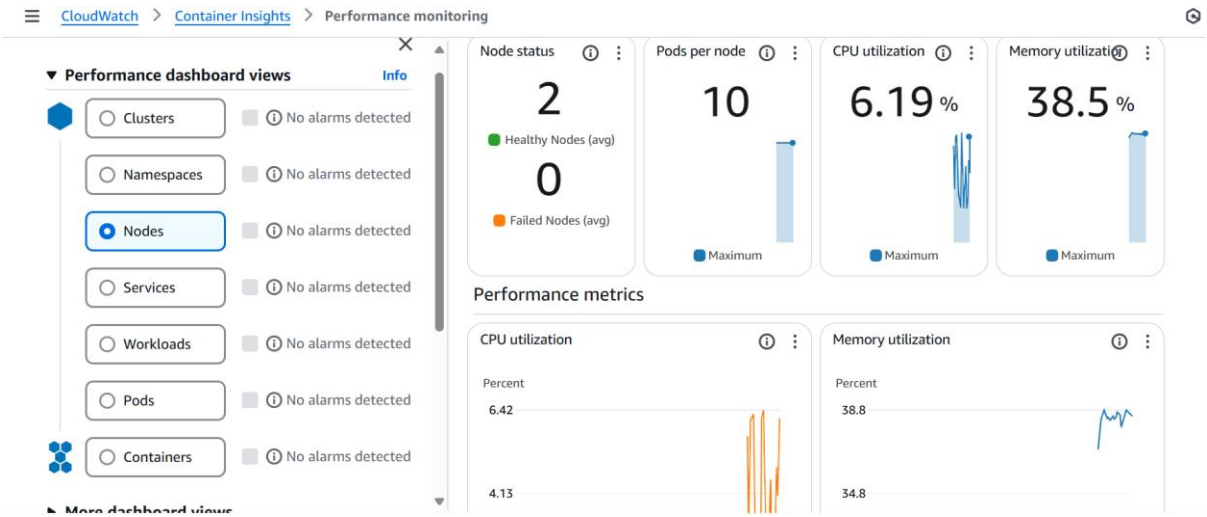


Pods performance monitoring

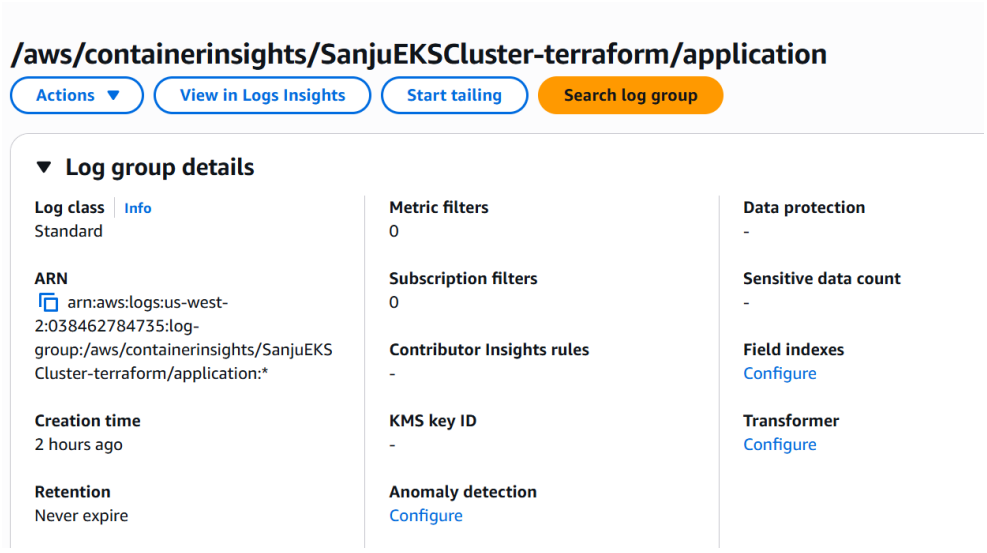


Cluster performance monitoring:

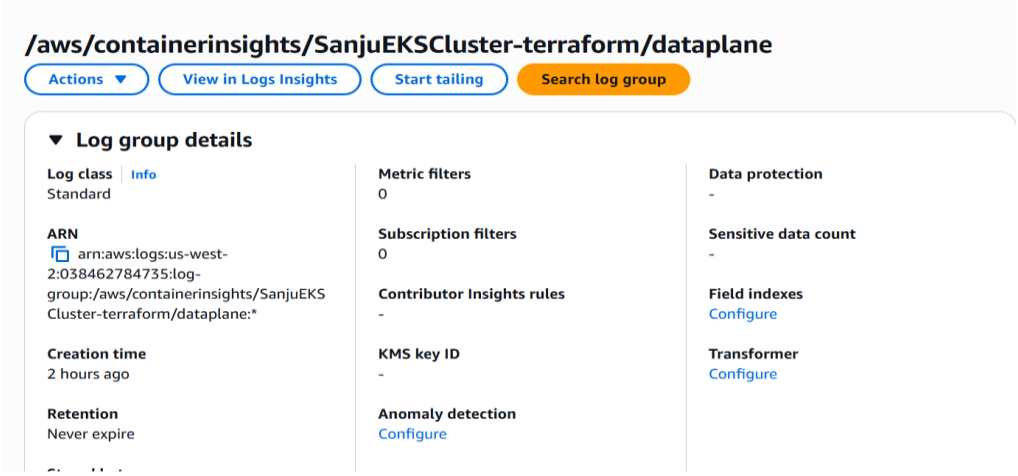




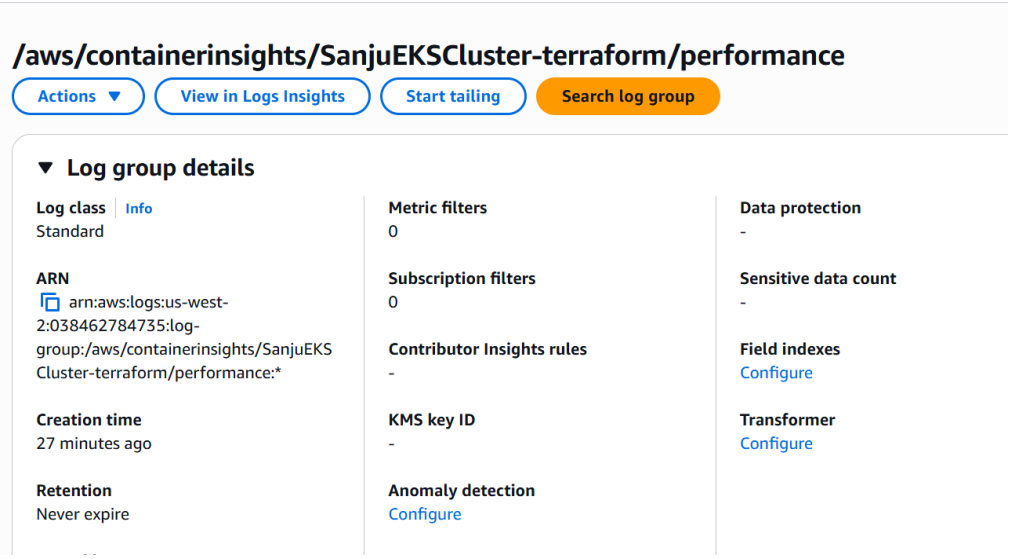
Log group – application



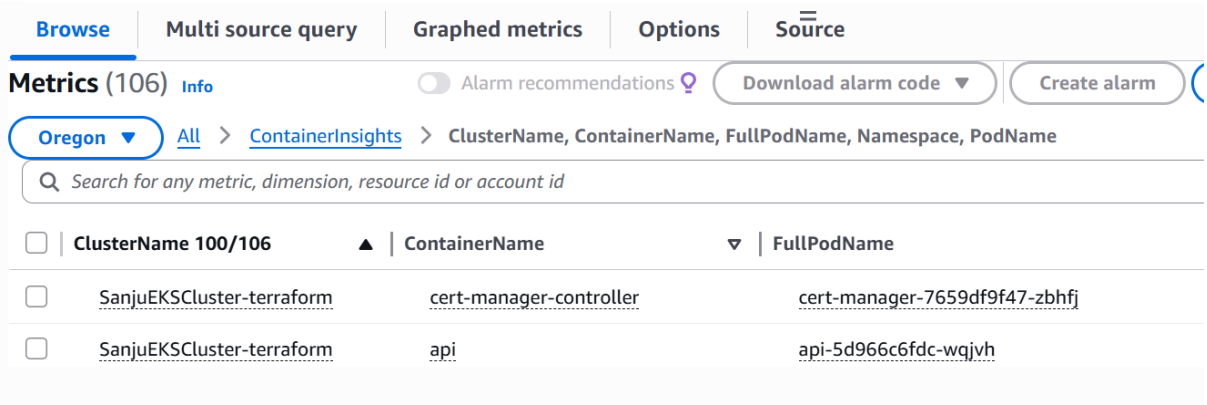
Log group – dataplane



Log group – performance



Metrics:



Alarms & SNS Notifications:

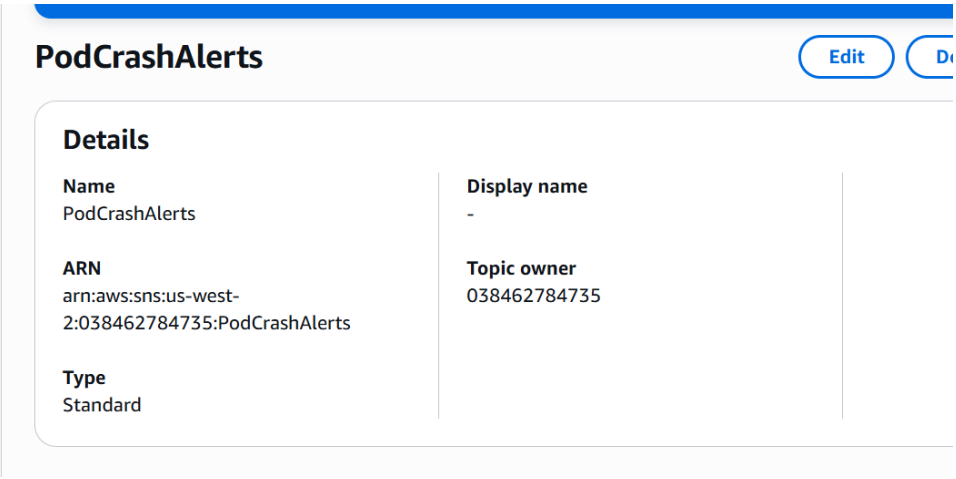
CloudWatch Alarms: Monitor specific metrics (e.g., CPU > 80%) and detect threshold breaches**.**

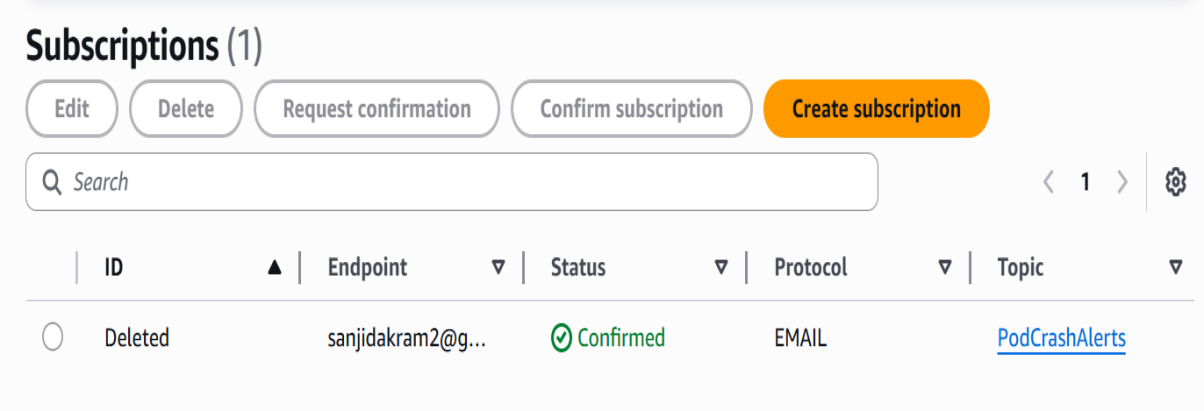
Automatic Alerts: Trigger actions like notifications, Auto Scaling, or Lambda functions.

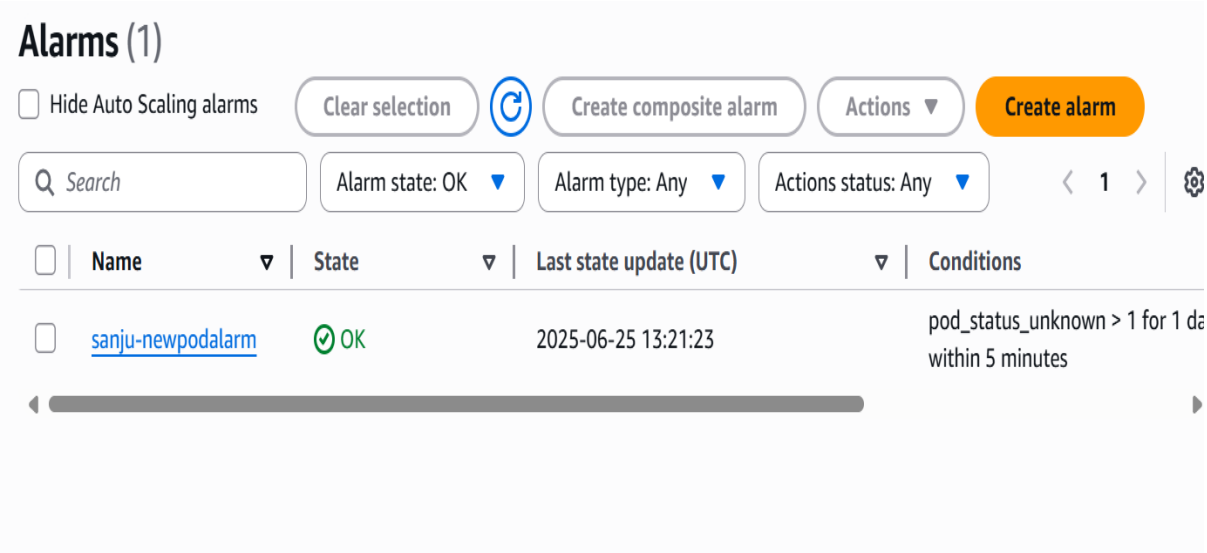
SNS (Simple Notification Service): Sends alerts via **email, SMS**, or **HTTP endpoints**.

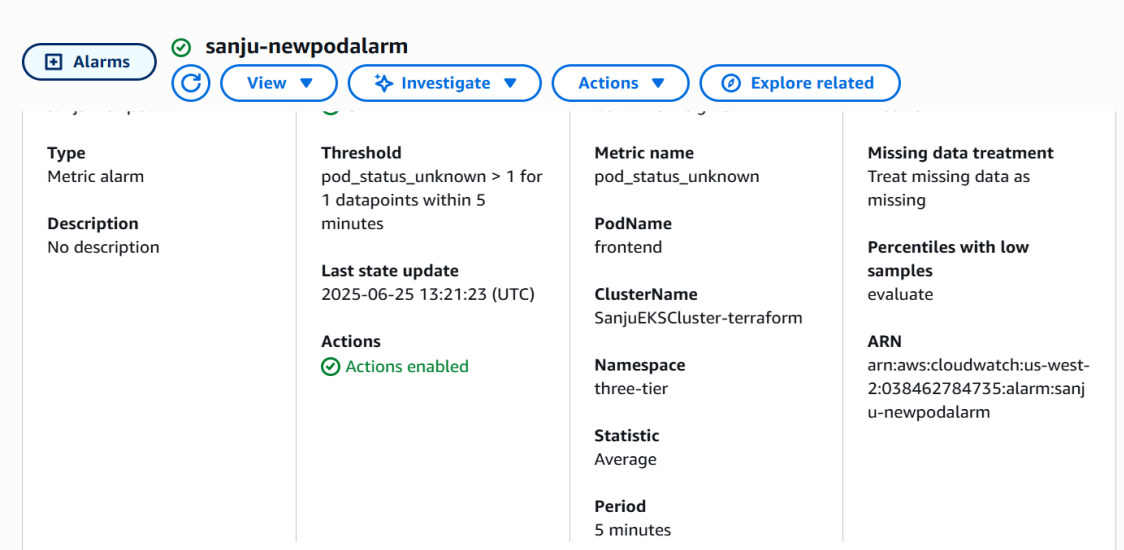
Integration: Alarms are linked to SNS topics to **notify teams instantly** of issues.

Use Case: Helps ensure **quick response** to failures or performance drops.









AWS CodeCommit:

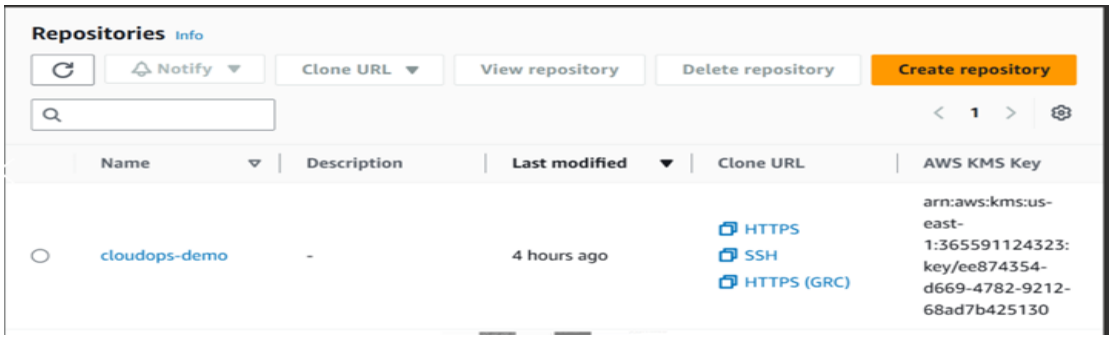
. us-east-1

. **Fully Managed Git Repository:** Securely stores your source code in the cloud.

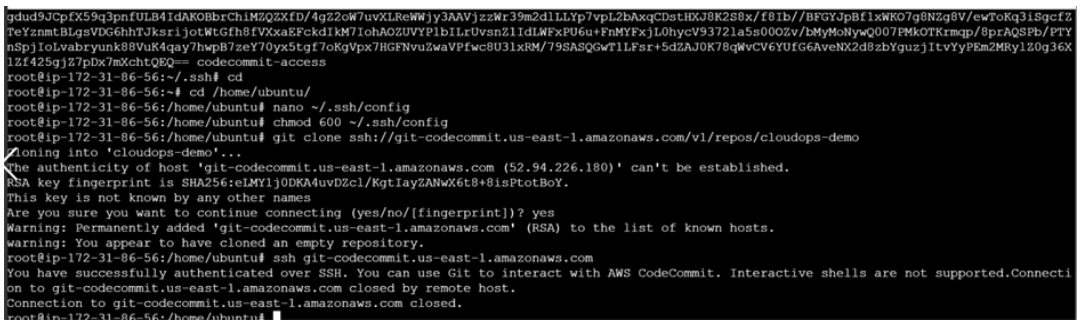
. **Private & Secure:** Integrated with **IAM** for access control and encryption.

. **Integration Ready:** Easily connects with **CodePipeline, CodeBuild**, and **CodeDeploy** for CI/CD.

CodeCommit repository



Running of CodeCommit.



SonarQube for Application

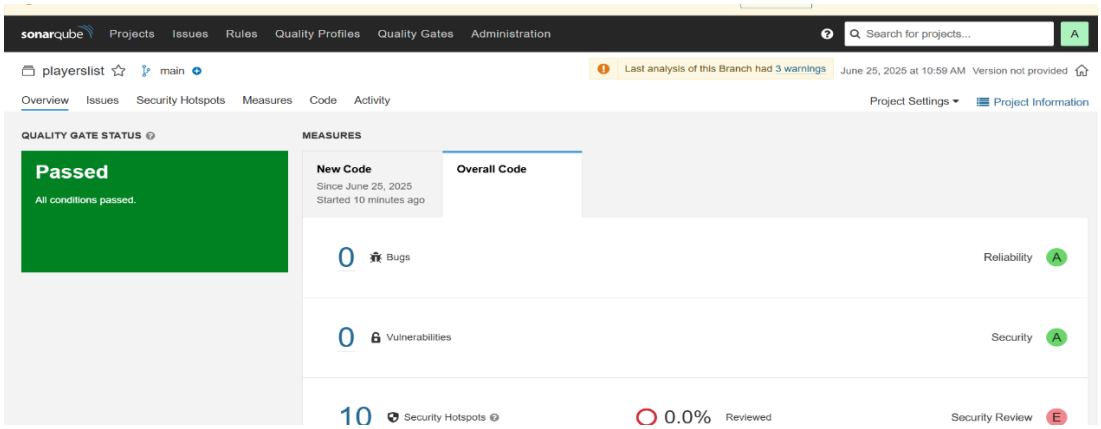
.Code Quality Tool: Analyzes source code for bugs, vulnerabilities, and code smells.

CI/CD Integration: Integrated with **CodeBuild** or **GitHub Actions** to scan code during each deployment.

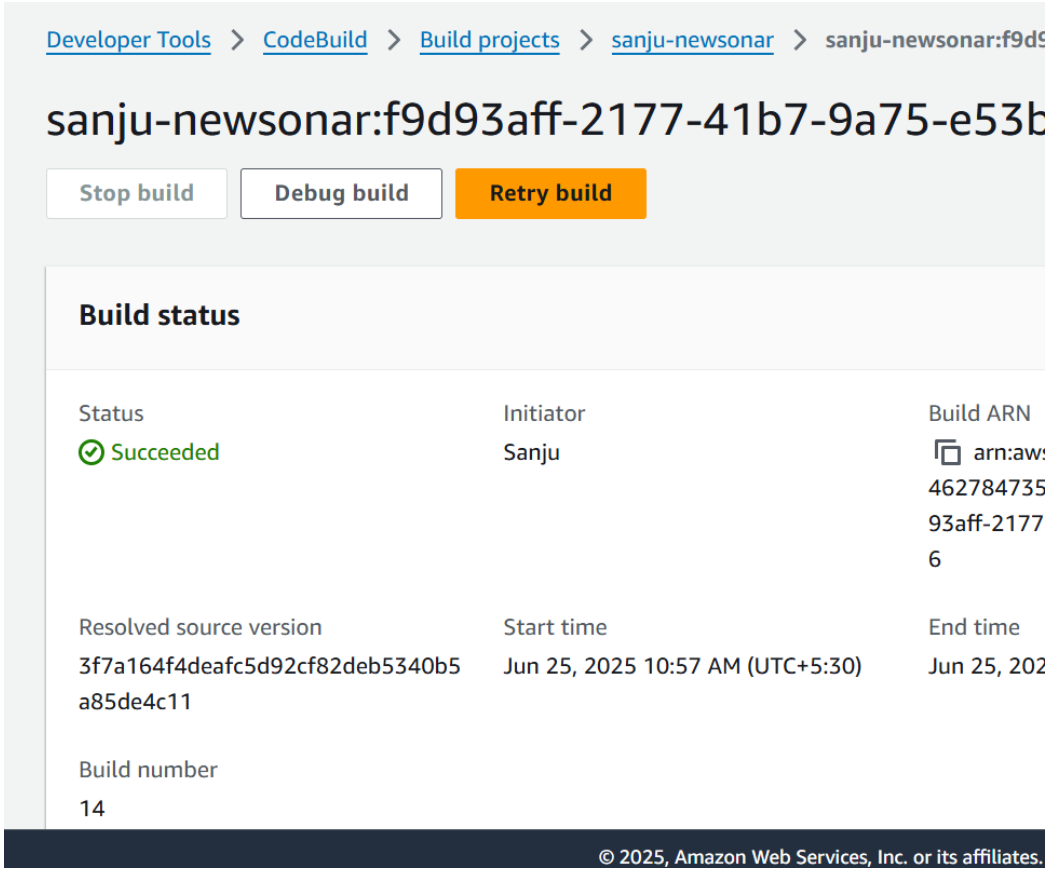
Improves Reliability: Helps maintain **clean, secure, and efficient codebase** in your application.

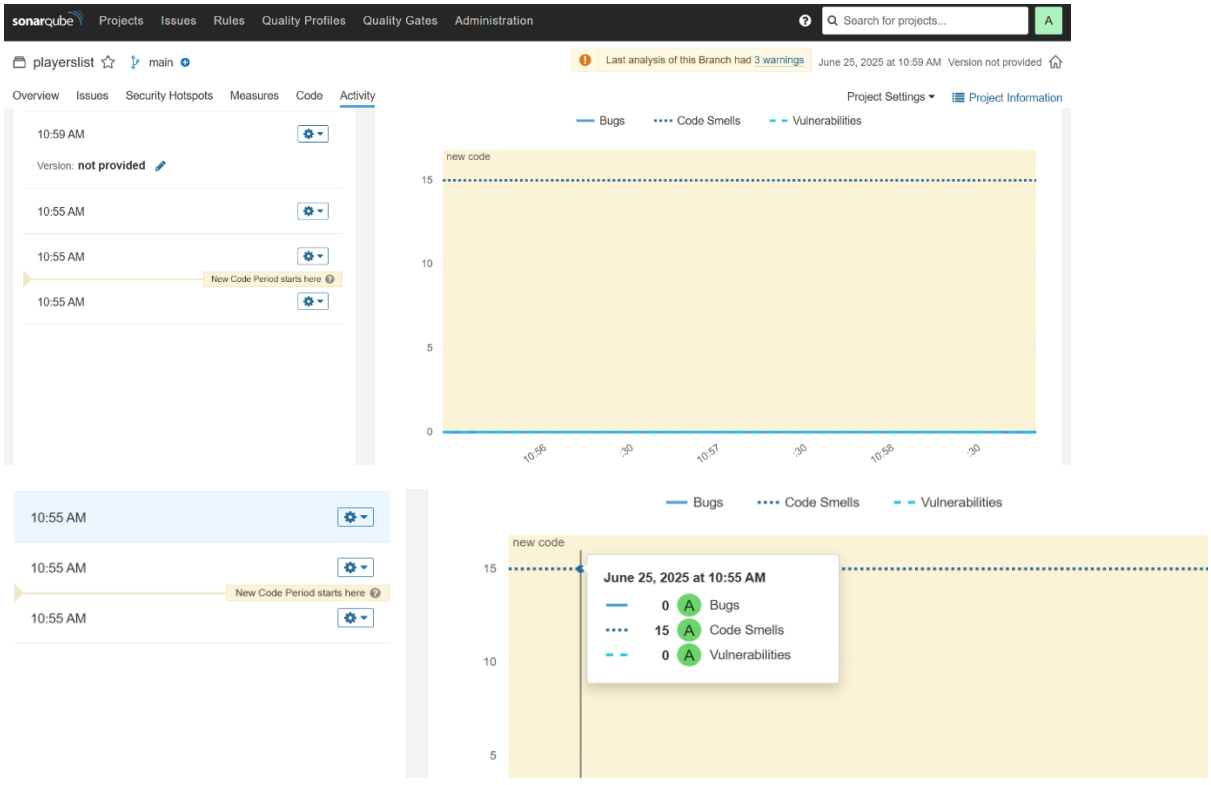
http://18.212.217.245/dashboard?id=playerslist

```
sonar-scanner \
  -Dsonar.projectKey=playerslist \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://18.212.217.245 \
  -Dsonar.login=sqp_ba1336c4d48188123a5cb0c4b98461522b71d5f4
```



SonarQube – codepipeline





Final Output – MyPlayerList Application

Purpose: A full-stack To-Do List style app for managing a list of players.

Features:

. Add, update, and delete player details.

. User-friendly frontend built with **ReactJS**.

. Backend powered by **Node.js** with a **MongoDB** (Amazon DocumentDB) database.

Deployed on: Amazon EKS using Docker containers and in **DocumentDB service**

CI/CD Enabled: Fully automated deployment via **CodePipeline** in **CloudFormation** and **Terraform**.

Monitoring & Security:

CloudWatch for logs and metrics.

SonarQube ensures code quality.

Route 53 & Load Balancer handle traffic routing and failover.