

# **Capstone Final Project Report**

## **Project Title: Event Dash Board**

**Cloud-Native Deployment and Monitoring of a Microservices Application using AWS EKS, CI/CD, Terraform, and Route 53**

## **GitHub Repository:**

🔗 <https://github.com/sujeetsrahate/Capstone-Final-z>

---

## **Table of Contents**

1. Introduction
  2. Objective
  3. Tools and Technologies Used
  4. Architecture Overview
  5. Phase 1: EKS Cluster Deployment
  6. Phase 2: CI/CD Pipeline Deployment
  7. Phase 3: Multi-Region Infra Setup using CloudFormation & Terraform
  8. Terraform Infra Structure
  9. Phase 4: Route 53 Restructuring (Disaster Recovery)
  10. Phase 5: Monitoring with CloudWatch
  11. SonarQube Integration
  12. Challenges Faced
  13. Future Scope
  14. Conclusion
  15. Appendix (YAML & Terraform Snippets)
- 

### **1. Introduction**

The goal of this capstone project was to design and deploy a full-stack cloud-native microservices application using AWS managed services and Infrastructure as Code (IaC) tools. This project integrates EKS, Terraform, AWS CodePipeline, Route 53, and CloudWatch to provide a scalable, highly available, and observable solution.

---

## 2. Objective

To implement a complete DevOps workflow with multi-region disaster recovery, monitoring, and CI/CD capabilities by:

- Deploying Java Spring Boot & ReactJS applications.
  - Automating infrastructure using CloudFormation and Terraform.
  - Implementing automated pipelines using CodePipeline.
  - Configuring Route 53 for disaster recovery.
  - Setting up CloudWatch for monitoring and logging.
- 

## 3. Tools and Technologies Used

- **AWS Services:** EKS, EC2, RDS, CloudWatch, Route 53, CodePipeline, S3, IAM
  - **IaC:** Terraform, AWS CloudFormation
  - **CI/CD:** AWS CodePipeline, CodeBuild
  - **Containerization:** Docker
  - **Monitoring:** CloudWatch
  - **Languages:** Java (Spring Boot), ReactJS, YAML, HCL
- 

## 4. Architecture Overview

The project implements a **three-tier microservice architecture**:

- **Frontend:** ReactJS
  - **Backend:** Spring Boot + MySQL (RDS)
  - **Infrastructure:** Provisioned across regions using CloudFormation and Terraform
  - **Pipeline:** Source -> Build -> Deploy stages in AWS CodePipeline
- 

## 5. Phase 1: Deploy High-Level Application on EKS

The initial phase involves deploying the application on Amazon EKS. Components:

- EKS Cluster creation (via AWS CLI)

- Kubernetes manifests for:
  - Spring Boot deployment & service
  - MySQL deployment & persistent volume
  - ReactJS frontend deployment & service
  - Ingress controller

Manifests used:

- `springboot-deploy.yaml`
  - `mysql-deploy.yaml`
  - `react-event-deploy.yaml`
  - `ingress.yaml`
- 

## 6. Phase 2: CI/CD Pipeline with CodePipeline

This phase integrates CI/CD via AWS CodePipeline.

### Key Files:

- `buildspec.yaml`  
Specifies how CodeBuild builds the project and pushes Docker images to ECR.

### Pipeline Flow:

1. Code push to GitHub
  2. CodeBuild builds and pushes to ECR
  3. Deploys to EKS using `kubectl`
- 

## 7. Phase 3: Multi-Region Infra via CloudFormation & Terraform

This phase establishes high availability and disaster recovery.

- CloudFormation stacks create VPC, subnets, security groups, etc.
- Terraform modules provision additional resources and EKS nodes.

### Repositories:

- [CloudFormation.yaml](#)
- [Infra-terraform.tf](#)

---

## **8. Terraform Infra Structure**

Additional infrastructure is provisioned using Terraform:

- VPC and subnets
- EKS cluster and worker node groups
- Route tables and gateways
- S3 and IAM roles

**GitHub Repo:** [sujeetsrahate/terraform\\_infra](https://github.com/sujeetsrahate/terraform_infra)

---

## **9. Phase 4: Route 53 Restructuring (Disaster Recovery)**

Route 53 is configured with:

- **Primary hosted zone** for active region
- **Secondary failover policy** for standby region
- Health checks on application endpoints
- Weighted routing policies for zero-downtime switch

---

## **10. Phase 5: Monitoring with CloudWatch**

CloudWatch is configured to:

- Collect logs from EKS using FluentBit
- Visualize metrics like CPU/Memory usage
- Trigger alarms for unhealthy pods or services
- Store logs from Spring Boot backend and application frontend

### **Importance:**

- Ensures observability
- Enables root cause analysis
- Supports performance tuning

---

## **11. SonarQube Integration**

SonarQube was added as a quality gate in the CI/CD pipeline.

- Scans for bugs, vulnerabilities, and code smells
  - Improves maintainability and security
  - Integrated in CodeBuild using Dockerized scanner
- 

## **12. Challenges Faced**

- Managing VPC peering across regions
  - IAM role policies during automated provisioning
  - Synchronizing Terraform and CloudFormation resources
  - Kubernetes networking in multi-region setup
  - Log ingestion and visibility in CloudWatch
- 

## **13. Future Scope**

- Use AWS Backup to automate RDS backups
  - Integrate Prometheus & Grafana for advanced monitoring
  - Add more microservices for horizontal scaling
  - Integrate with GitHub Actions or Jenkins for hybrid CI/CD
- 

## **14. Conclusion**

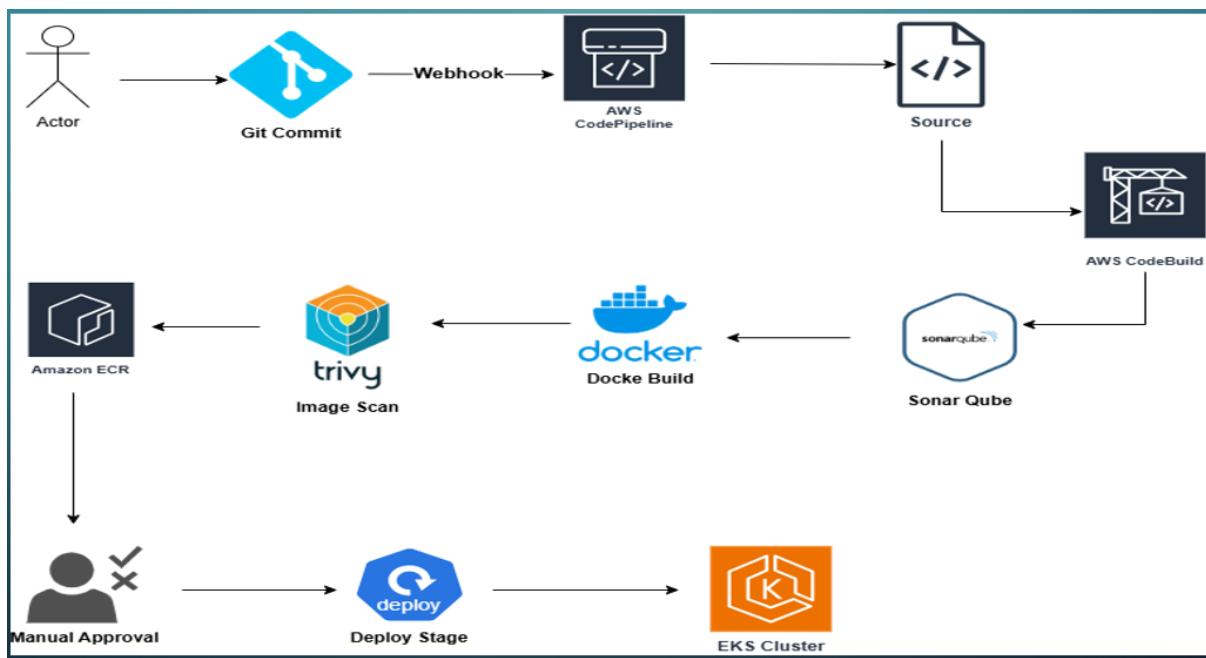
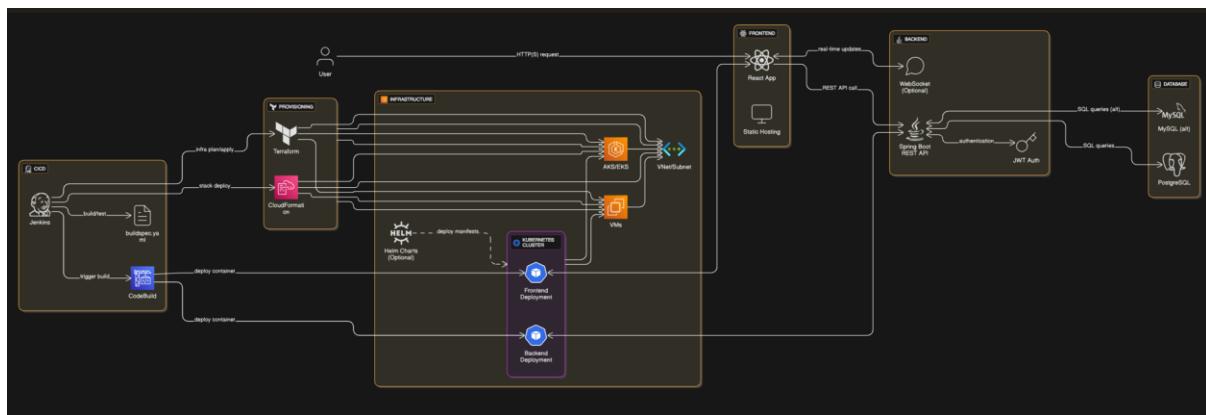
This capstone project successfully delivered a multi-region, production-grade cloud-native application using industry-standard DevOps tools and AWS services. All core objectives including CI/CD, container orchestration, disaster recovery, and monitoring were achieved.

---

Capstone Project:

GitHub Repository: [sujeetsrahate/Capstone-Final-z](https://github.com/sujeetsrahate/Capstone-Final-z)

Phase 1: Deploy high level application on EKS Cluster



**Task Tracker**

Manage tasks and track their status for different events.

Add Task

Task Name	Event Name	Status	Actions
project	test	Completed	<button>Edit</button> <button>Delete</button>

**Event Management**

Add Event

Event_Id	Name	Description	Location	Date	Actions
1	djfnierebgfi	sdfnwj	nfskdjfniwb	2025-06-19	<button>Edit</button> <button>Delete</button>
2	test	test	test	2025-06-18	<button>Edit</button> <button>Delete</button>
3	kaushal	cka	bombay	2025-07-01	<button>Edit</button> <button>Delete</button>
5	ani	python	wb	2025-07-02	<button>Edit</button> <button>Delete</button>
6	yogesh	boss	trivandrum	2025-06-24	<button>Edit</button> <button>Delete</button>

## Phase 2: deploy Application Through pipeline

Git Repo(buildspec.yaml): [Capstone-Final-z/buildspec.yaml at main · sujeetsrahate/Capstone-Final-z](#)

The screenshot shows the AWS CodeBuild console. On the left, a sidebar menu for 'CodeBuild' includes options like 'Source', 'Artifacts', 'Build', 'Getting started', 'Build projects', 'Build history', 'Report groups', 'Report history', 'Compute fleets', 'Account metrics', and 'Related integrations'. The main area is titled 'Build projects' with a 'Create project' button. It lists two projects:

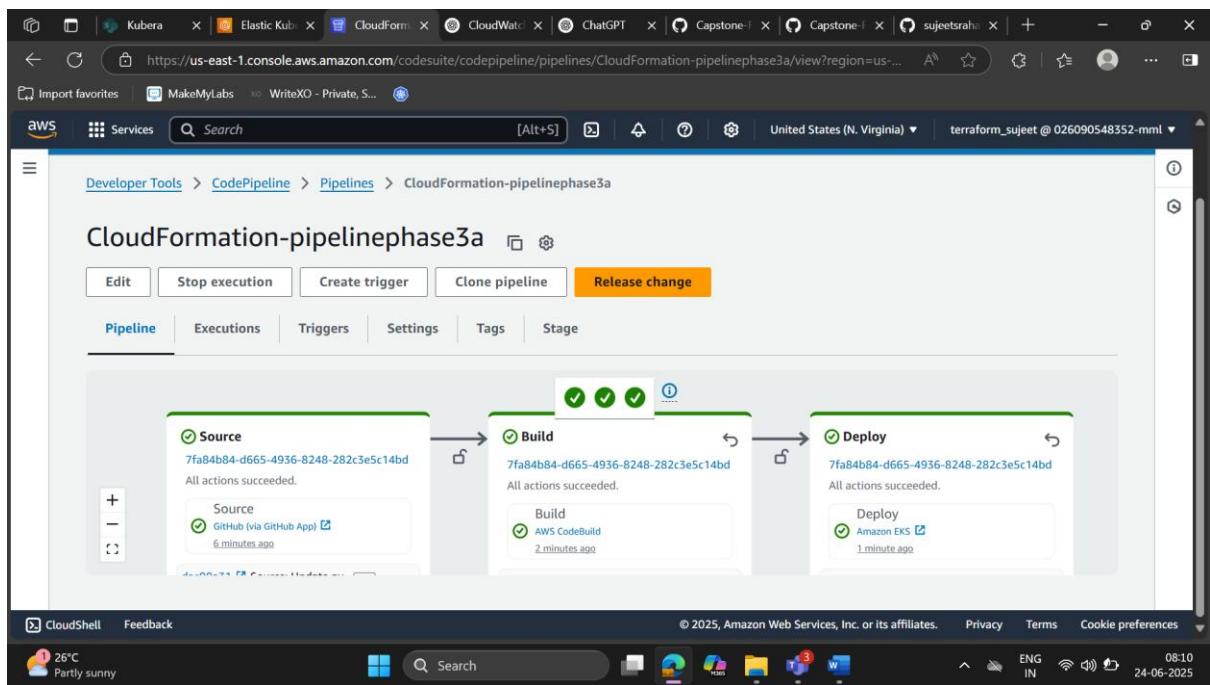
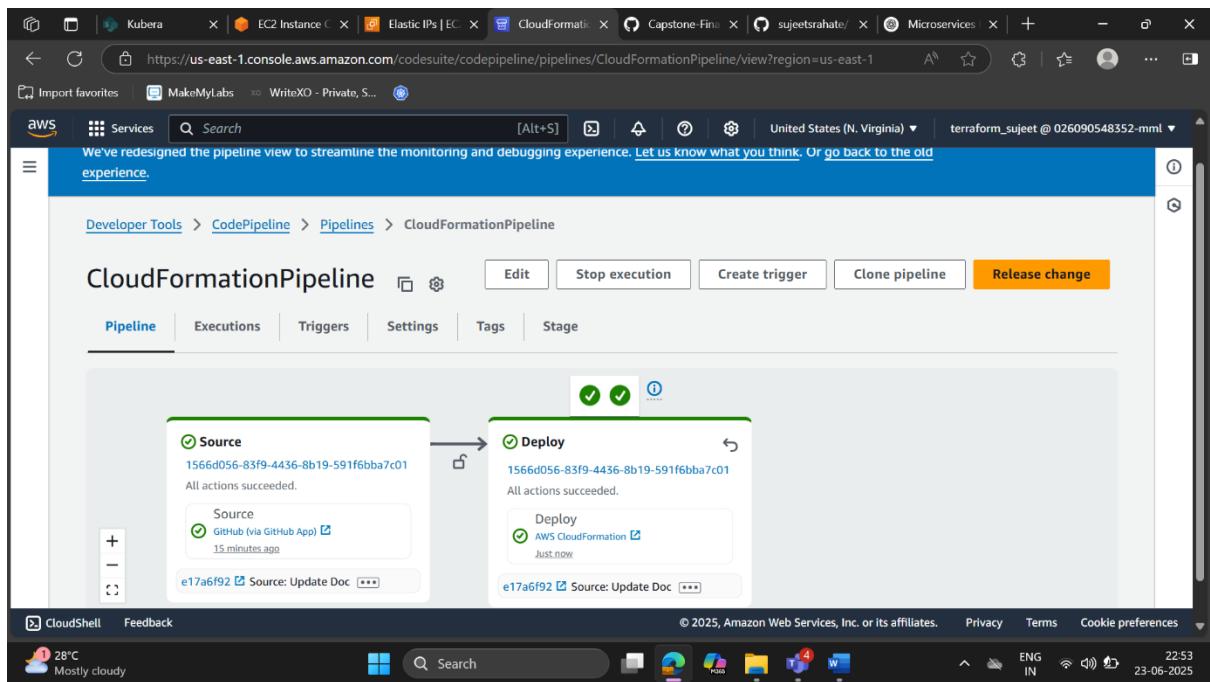
Name	Source provider	Repository	Latest build status	Description	Last Modified
final-z	GitHub	sujeetsrahate/Capstone-Final-z	Succeeded	-	6 hours ago
cloud-build	GitHub	sujeetsrahate/Project-high-level-deployment	Succeeded	-	4 days ago

The screenshot shows the AWS CodePipeline console. The pipeline is named 'project-pipeline'. It consists of three stages: 'Source', 'Build', and 'Deploy'. The 'Source' stage is triggered by GitHub and has succeeded. The 'Build' stage uses AWS CodeBuild and has succeeded. The 'Deploy' stage uses Amazon EKS and has succeeded. All actions in the pipeline have succeeded.

Phase 3: Use AWS Auto pipeline Multi Region of infrastructure with CFT and Terraform.

Git Hub Repo: [Capstone-Final-z/CloudFormation.yaml at main · sujeetsrahate/Capstone-Final-z](#)

Terraform GitHubRepo: [Capstone-Final-z/Infra-terraform.tf at main · sujeetsrahate/Capstone-Final-z](#)



CloudFormation working proper and make the vpc and everything

### 3B: creating Infra Structure through Terraform

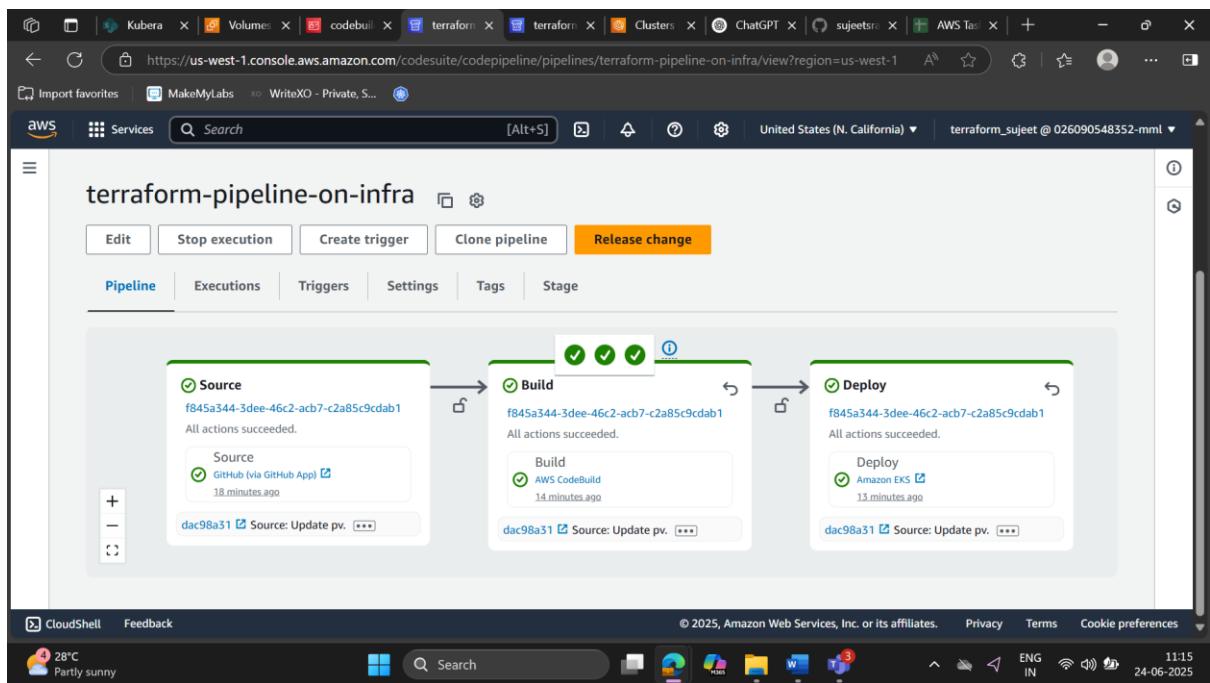
Git Hub Repo : [sujeetsrahate/terraform\\_infra](https://github.com/sujeetsrahate/terraform_infra)

The screenshot shows a log stream from AWS CloudWatch. The log entries are as follows:

```

26725 [Container] 2025/06/24 05:01:29.622500 Running command cp $K8S_MANIFEST_DIR/mysql-deploy.yaml k8s-out/mysql-deploy.yaml
26726 [Container] 2025/06/24 05:01:29.630529 Running command cp $K8S_MANIFEST_DIR/mysql-service.yaml k8s-out/mysql-service.yaml
26727
26728 [Container] 2025/06/24 05:01:29.630529 Running command cp $K8S_MANIFEST_DIR/pvc.yaml k8s-out/pvc.yaml
26729 [Container] 2025/06/24 05:01:29.638609 Running command cp $K8S_MANIFEST_DIR/pv.yaml k8s-out/pv.yaml
26730 [Container] 2025/06/24 05:01:29.646542 Running command cp $K8S_MANIFEST_DIR/pvc.yaml k8s-out/pvc.yaml
26731
26732 [Container] 2025/06/24 05:01:29.654765 Phase complete: POST_BUILD State: SUCCEEDED
26733 [Container] 2025/06/24 05:01:29.654779 Phase context status code: Message:
26734 [Container] 2025/06/24 05:01:29.892638 Set report auto-discover timeout to 5 seconds
26735 [Container] 2025/06/24 05:01:29.895951 Expanding file paths for base directory .
26736 [Container] 2025/06/24 05:01:29.896041 Assembling file list
26737 [Container] 2025/06/24 05:01:29.896350 Expanding base directory path: .
26738 [Container] 2025/06/24 05:01:29.896366 Expanding .
26739 [Container] 2025/06/24 05:01:29.896366 Expanding .
26740 [Container] 2025/06/24 05:01:29.899591 Expanding file paths for base directory .
26741 [Container] 2025/06/24 05:01:29.899604 Assembling file list
26742 [Container] 2025/06/24 05:01:29.899607 Expanding */
26743 [Container] 2025/06/24 05:01:29.903678 Found 1 file(s)
26744 [Container] 2025/06/24 05:01:29.903702 Report auto-discover file discovery took 0.011064 seconds
26745 [Container] 2025/06/24 05:01:29.904183 Phase complete: UPLOAD_ARTIFACTS State: SUCCEEDED
26746 [Container] 2025/06/24 05:01:29.904194 Phase context status code: Message:
26747

```



## Phase 4: Make Route 53 Restructure. (make disaster strategy)

https://us-east-1.console.aws.amazon.com/route53/v2/hostedzones?region=us-west-1#CreateRecordSet/Z0980777JFO0DP...

Import favorites MakeMyLabs WriteXO - Private, S... [Alt+S]

Route 53 > Hosted zones > sujeetrahat.shop > Create record

### Quick create record

**Record 1**

Record name	.sugeetrahat.shop	Record type	A – Routes traffic to an IPv4 address and some AWS resources
Keep blank to create a record for the root domain.		Switch to wizard	
<input checked="" type="checkbox"/> Alias			
Route traffic to	Alias to Application and Classic Load Balancer		
US West (N. California)			
Q. dualstack.af9421b408e103441da74fc5b3bcfe5e3-699696660.us-west-1.elb.amazonaws.com			
Alias hosted zone ID: Z368ELLRRE2KJ0			
Routing policy	Failover		
Failover record type Choose Primary to route traffic to the specified resource by default or Secondary to route traffic to the specified resource when the primary resource is unavailable. You can create only one failover record of each type.			
<input checked="" type="checkbox"/> Secondary			

https://us-east-1.console.aws.amazon.com/route53/v2/hostedzones?region=us-west-1#ListRecordSets/Z0980777JFO0DP...

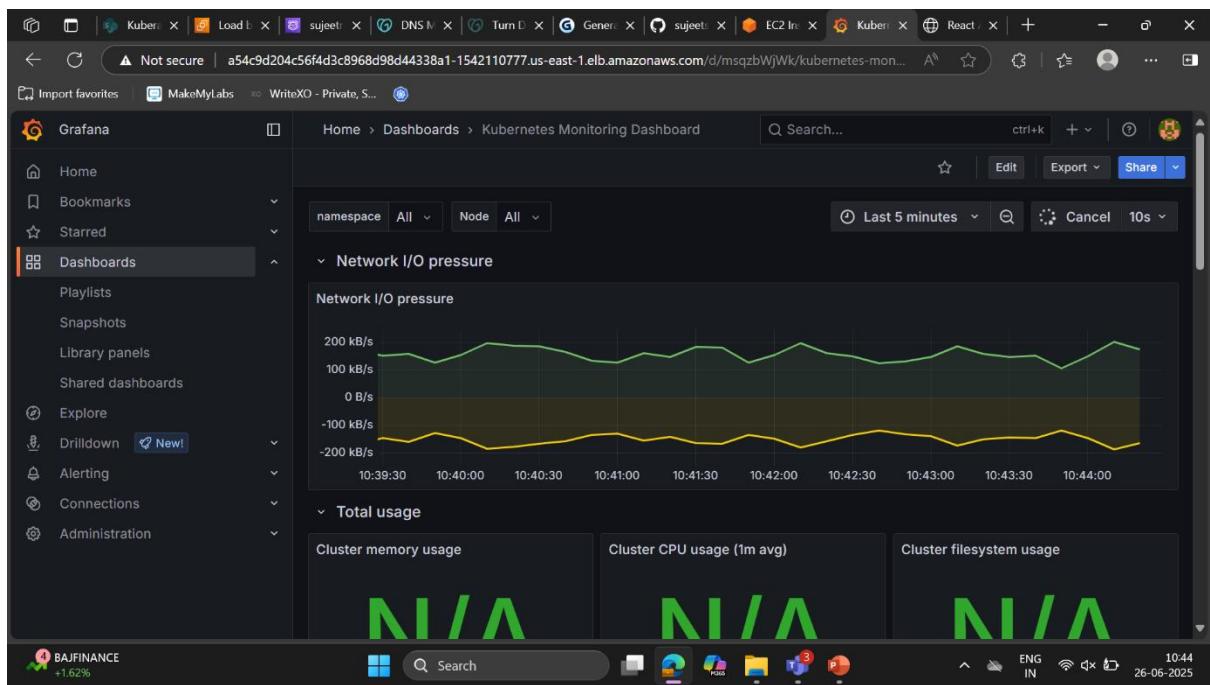
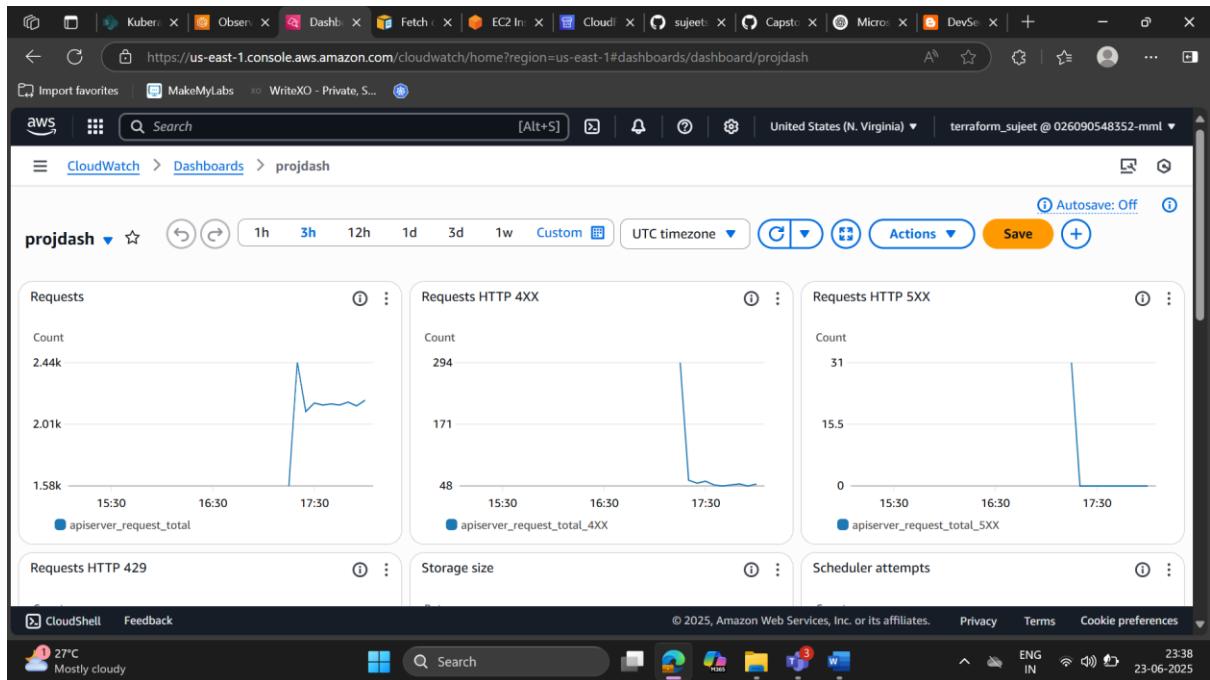
Import favorites MakeMyLabs WriteXO - Private, S... [Alt+S]

Route 53 > Hosted zones > sujeetrahat.shop

Route 53		Records (4)		DNSSEC signing		Hosted zone tags (0)		Record details	
Records (1/4) Info									
Delete record Import zone file Create record									
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.									
Filter records by property or value Type Routing p... Alias									
1									
Record name		Type	Routin...	Differ...	Alias	Valu			
sugeetrahat.shop	NS	Simple	-	No	ns-1 ns-8 ns-6 ns-1				
sugeetrahat.shop	SOA	Simple	-	No	ns-1				
<input checked="" type="checkbox"/> first.sugeetrahat.shop	A	Failover	Primary	Yes	dual				
second.sugeetrahat.shop	A	Failover	Secondary	Yes	dual				

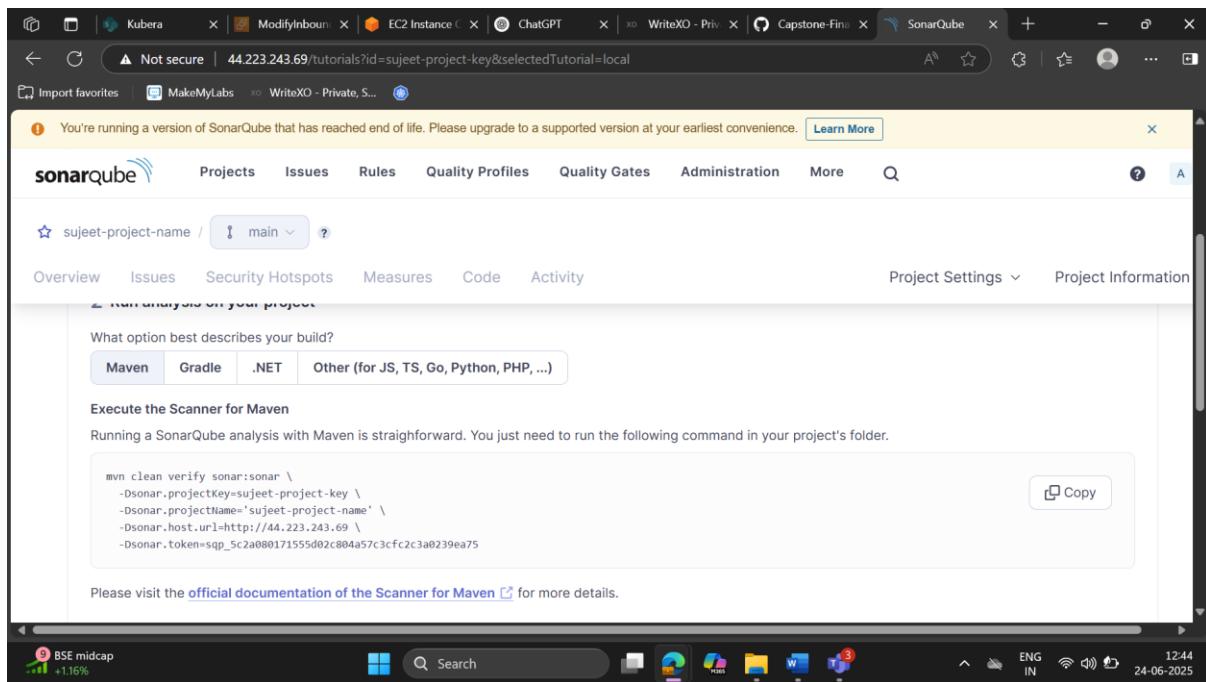
© 2025, Amazon Web Services, Inc. or its affiliates. ENG IN 21:03 25-06-2025

## Phase 5: Cloud Watch:



The screenshot shows the AWS CloudWatch Metrics interface. At the top, there are navigation links for 'CloudWatch' and 'Metrics'. Below that, a search bar and filter options for time (1h, 3h, 12h, 1d, 3d, 1w, Custom) and UTC timezone are present. A 'Actions' dropdown and 'Investigate' button are also visible. The main area displays an 'Untitled graph' with six metric groups: SSM-RunCommand (3), Scheduler (6), SecretsManager (1), States (69), Usage (580), and VPN (12). Each group has a link to 'View automatic dashboard'. The bottom of the screen shows standard browser controls and a status bar indicating 'CloudShell Feedback'.

The screenshot shows a web browser displaying a SonarQube login page. The URL in the address bar is [http://44.223.243.69/sessions/new?return\\_to=%2F](http://44.223.243.69/sessions/new?return_to=%2F). The page title is 'Log in to SonarQube'. It features the Sonar logo at the top. Below it is a form with two input fields: 'Login \*' containing 'admin' and 'Password \*' containing a masked password. There are 'Go back' and 'Log in' buttons at the bottom of the form. The browser's status bar shows 'BSE midcap +1.11%' and the date '24-06-2025'.



Here are the **important YAML files** from your project repo that are essential for deploying your application on EKS:

---

### springboot-deploy.yaml

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: springboot-app

spec:

replicas: 2

selector:

matchLabels:

app: springboot

template:

metadata:

labels:

```
app: springboot

spec:

containers:
  - name: springboot

image: <your-ecr-repo>/springboot-app:latest

ports:
  - containerPort: 8080

env:
  - name: SPRING_DATASOURCE_URL

    valueFrom:
      secretKeyRef:
        name: db-secret
        key: datasource_url
```

---

### **mysql-deploy.yaml**

```
yaml
CopyEdit
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
```

```
metadata:
  labels:
    app: mysql
  spec:
    containers:
      - image: mysql:5.7
        name: mysql
      env:
        - name: MYSQL_ROOT_PASSWORD
          value: rootpassword
        - name: MYSQL_DATABASE
          value: capstone_db
    ports:
      - containerPort: 3306
    volumeMounts:
      - name: mysql-persistent-storage
        mountPath: /var/lib/mysql
  volumes:
    - name: mysql-persistent-storage
      persistentVolumeClaim:
        claimName: mysql-pvc
```

---

### **react-event-deploy.yaml**

```
yaml
CopyEdit
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
name: react-frontend
spec:
replicas: 2
selector:
matchLabels:
  app: react
template:
metadata:
labels:
  app: react
spec:
containers:
- name: react
  image: <your-ecr-repo>/react-frontend:latest
ports:
- containerPort: 3000
```

---

### **ingress.yaml**

```
yaml
CopyEdit
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: capstone-ingress
annotations:
  nginx.ingress.kubernetes.io/rewrite-target: /
spec:
rules:
```

```
- http:  
  paths:  
    - path: /springboot  
      pathType: Prefix  
      backend:  
        service:  
          name: springboot-service  
        port:  
          number: 8080  
    - path: /react  
      pathType: Prefix  
      backend:  
        service:  
          name: react-service  
        port:  
          number: 3000
```

---

### **mysql-service.yaml**

```
yaml  
CopyEdit  
apiVersion: v1  
kind: Service  
metadata:  
  name: mysql  
spec:  
  ports:  
    - port: 3306  
  selector:
```

app: mysql

---

**java-springboot-service.yaml**

yaml  
CopyEdit  
apiVersion: v1  
kind: Service  
metadata:  
  name: springboot-service  
spec:  
  type: ClusterIP  
  selector:  
    app: springboot  
  ports:  
    - port: 8080  
      targetPort: 8080

---

**react-event-service.yaml**

yaml  
CopyEdit  
apiVersion: v1  
kind: Service  
metadata:  
  name: react-service  
spec:  
  type: ClusterIP  
  selector:  
    app: react

```
ports:  
  - port: 3000  
    targetPort: 3000
```

---

### **pv.yaml**

```
yaml  
CopyEdit  
apiVersion: v1  
kind: PersistentVolume  
metadata:  
  name: mysql-pv  
spec:  
  capacity:  
    storage: 1Gi  
  accessModes:  
    - ReadWriteOnce  
hostPath:  
  path: "/mnt/data"
```

---

### **pvc.yaml**

```
yaml  
CopyEdit  
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
  name: mysql-pvc  
spec:  
  accessModes:
```

- ReadWriteOnce

resources:

requests:

storage: 1Gi

=====Complete=====