

# **Capstone Final Project Report**

## **Project Title: Student-Teacher-Portal**

Multi-Region Cloud-Native Deployment with Disaster Recovery, CI/CD, Container Security and Observability using AWS, Terraform, CloudFormation, EKS, Route 53, Prometheus, and Grafana

### **GitHub Repositories**

- CloudFormation Infrastructure & CI/CD (Region 1):  
 <https://github.com/ashutoshsanghi3/Student-Teacher-Portal>  
(Contains: CloudFormation.yaml, buildspec.yaml)
- Terraform Infrastructure (Region 2):  
 <https://github.com/ashutoshsanghi3/TF-EKS-RDS>  
(Contains: All .tf files for VPC, EKS, RDS, etc.)
- CI/CD Pipeline for Terraform Phase (Region 2)::  
 <https://github.com/ashutoshsanghi3/Student-Teacher-Portal>  
(Contains: buildspec.yaml shared with CloudFormation pipeline)

### **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
8. Phase 4: Route 53 Failover Configuration
9. Phase 5: Monitoring with Prometheus and Grafana
10. SonarQube Integration
11. Trivy Security Scanning
12. Setup Instructions
13. Challenges Faced
14. Future Scope
15. Conclusion

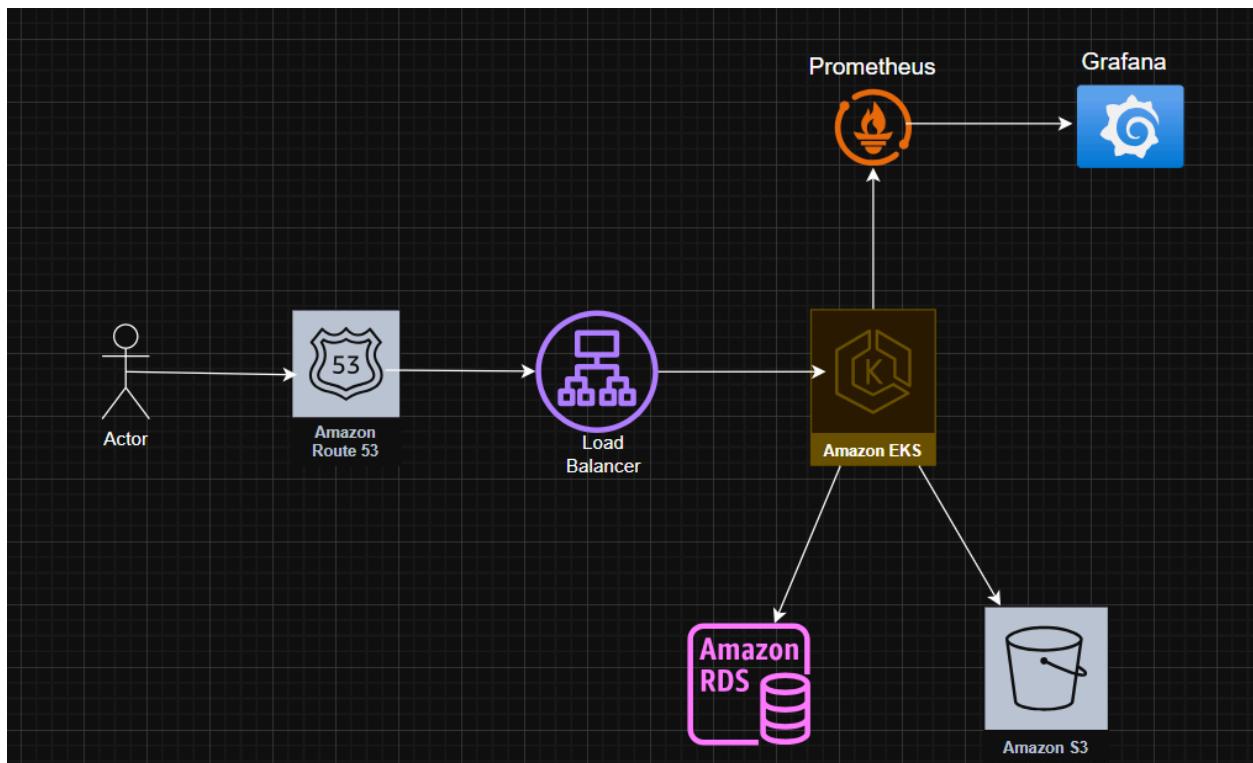
## 1. Introduction

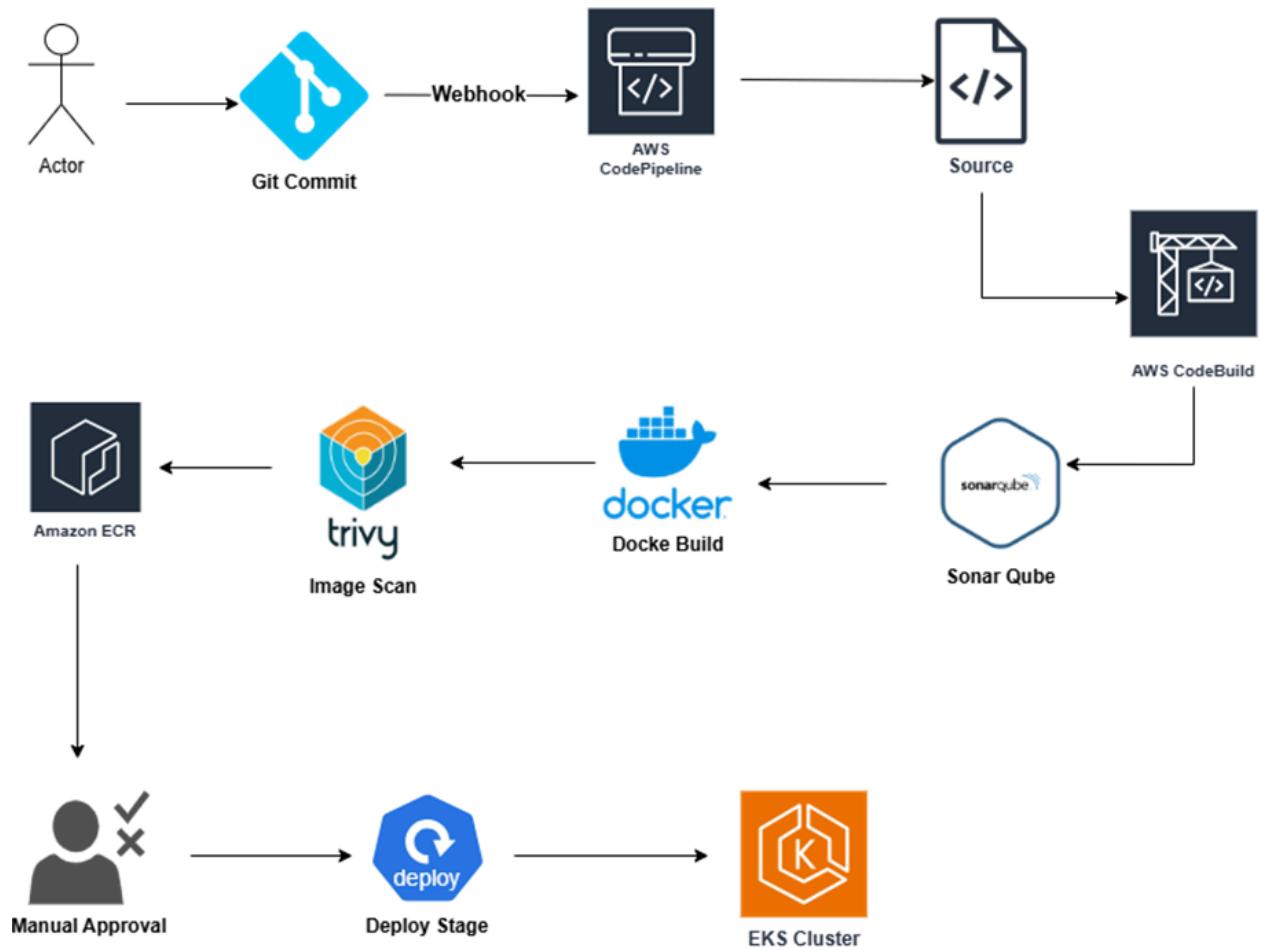
This project delivers a fully automated, highly available, multi-region Student-Teacher Portal deployed on AWS using EKS, CloudFormation, Terraform, and CodePipeline. It features CI/CD, infrastructure as code, container security, and real-time monitoring via Prometheus and Grafana, replacing traditional CloudWatch monitoring.

## 2. Objective

To build a production-ready, fault-tolerant cloud-native system that:

- Deploys a React + Node.js + MySQL 3-tier app
- Uses both CloudFormation and Terraform for IaC
- Enforces CI/CD best practices with SonarQube and Trivy
- Implements regional failover using Route 53
- Uses Prometheus & Grafana for advanced observability





### **3. Tools and Technologies Used**

<b>Category</b>	<b>Tools/Technologies</b>
IaC	Terraform, CloudFormation
Containerization	Docker
Orchestration	Amazon EKS
CI/CD	AWS CodePipeline, CodeBuild
Monitoring	Prometheus, Grafana
Security	Trivy (container scanning)
Code Quality	SonarQube
Networking	Route 53, ALB, Ingress, NACLs, Security Groups
Languages	Node.js, React, YAML, SQL

## 4. Architecture Overview

- Frontend: React SPA running on EKS
- Backend: Node.js/Express app in EKS
- Database: Amazon RDS (MySQL)
- Regions:
  - Region 1 (CloudFormation)
  - Region 2 (Terraform)
- Failover: Route 53 failover routing
- Monitoring: Prometheus for metrics collection and Grafana for dashboards
- CI/CD: CodePipeline pipelines in both regions

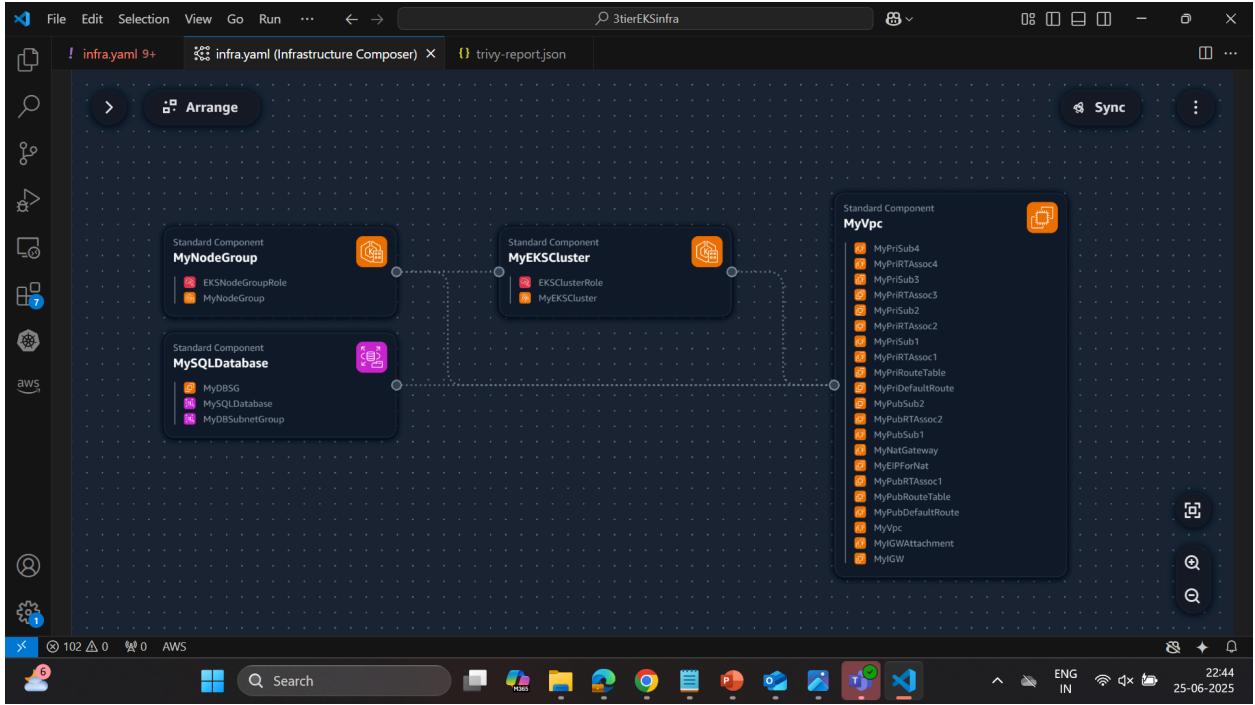
## 5. Phase 1: EKS Cluster Deployment

Each region provisions its own EKS cluster using:

- CloudFormation (Region 1)
- Terraform (Region 2)

Provisioned components include:

- Custom VPCs
- Private/public subnets
- NAT Gateways
- Route Tables
- Security Groups
- RDS MySQL instance
- EKS node groups
- Kubernetes manifests (React, Node.js, MySQL, Ingress)



## 6. Phase 2: CI/CD Pipeline Deployment

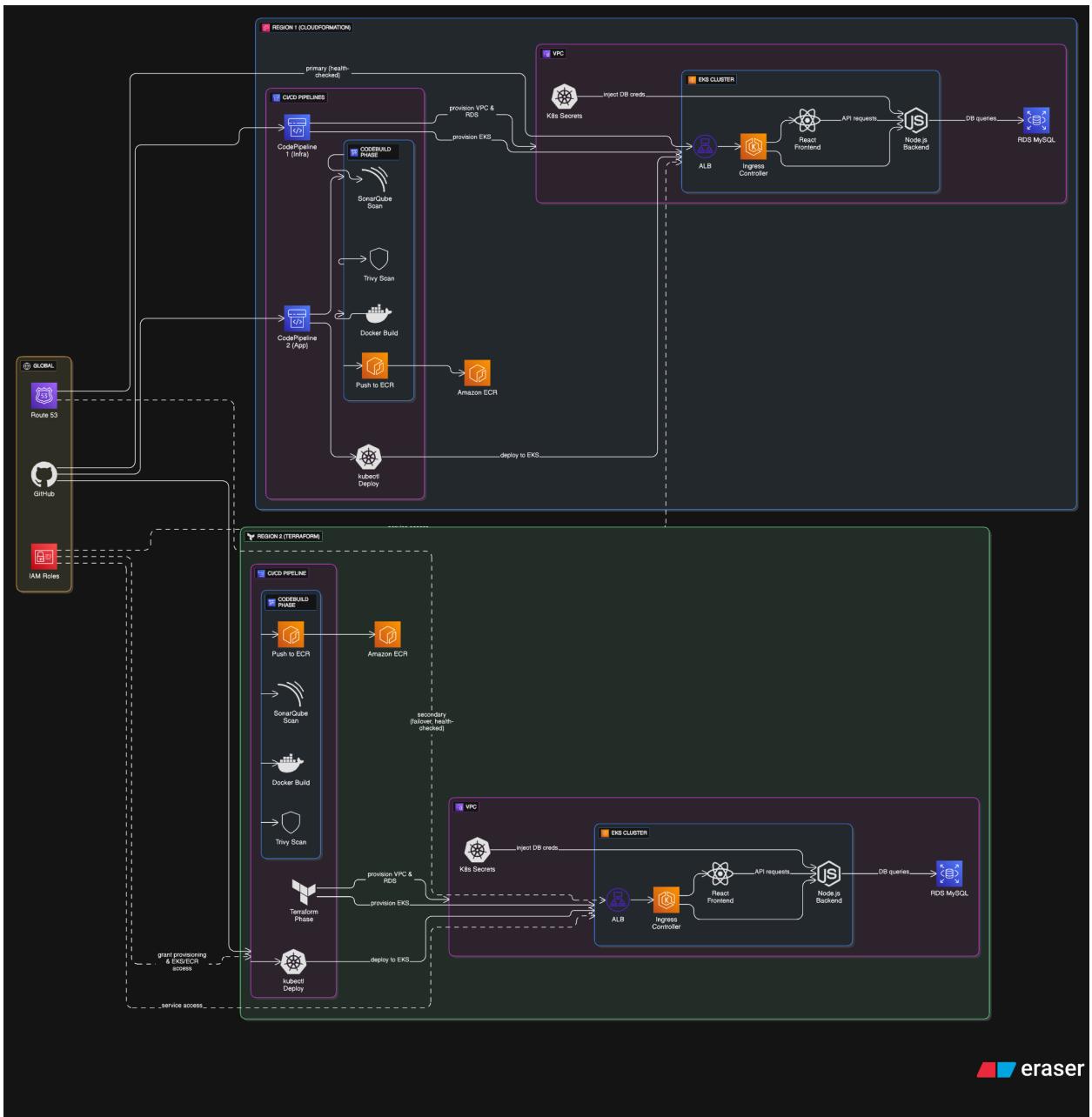
Implemented via AWS CodePipeline in both regions. GitHub Repo:

<https://github.com/ashutoshsanghi3/Student-Teacher-Portal>

<https://github.com/ashutoshsanghi3/TF-EKS-RDS>

(Shared buildspec.yaml used for both CloudFormation and Terraform deployments) Pipeline Workflow:

1. GitHub source trigger
2. SonarQube scan
3. Docker image build
4. Trivy vulnerability scan
5. Image push to ECR
6. Kubernetes deploy to EKS

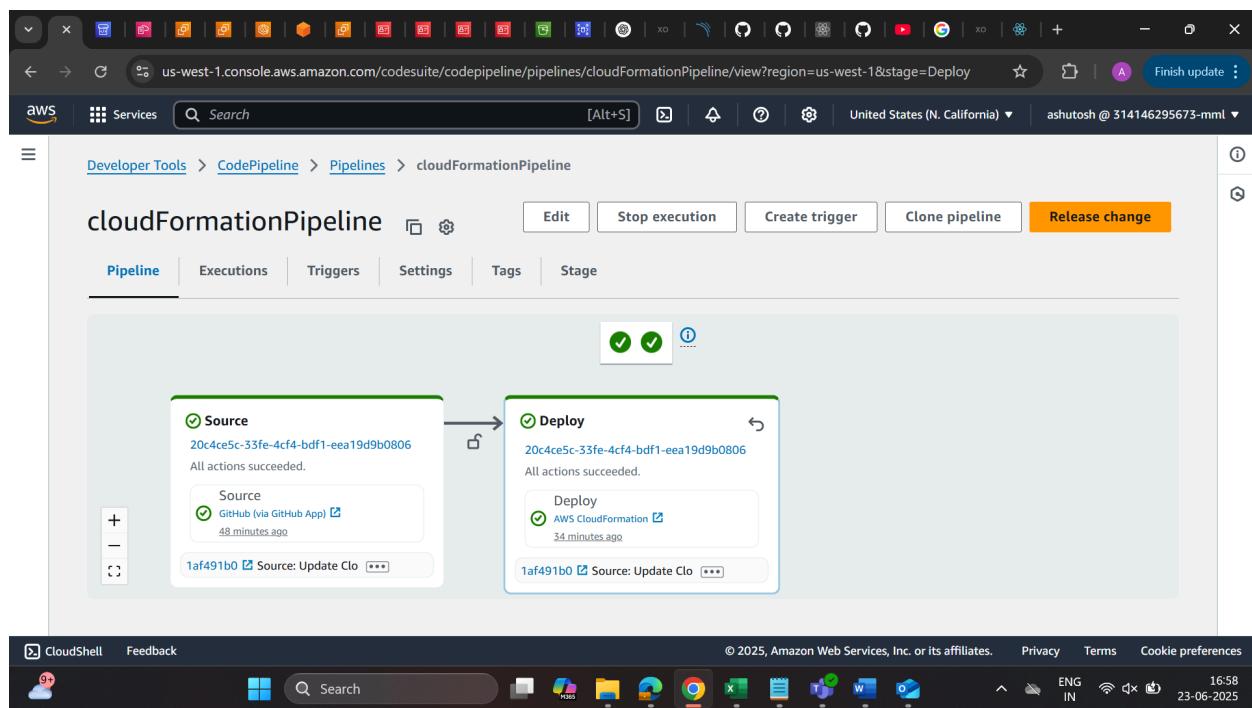


## 7. Phase 3: Multi-Region Infrastructure & CI/CD Pipelines

Region 1 – CloudFormation-based Deployment

Infrastructure Pipeline:

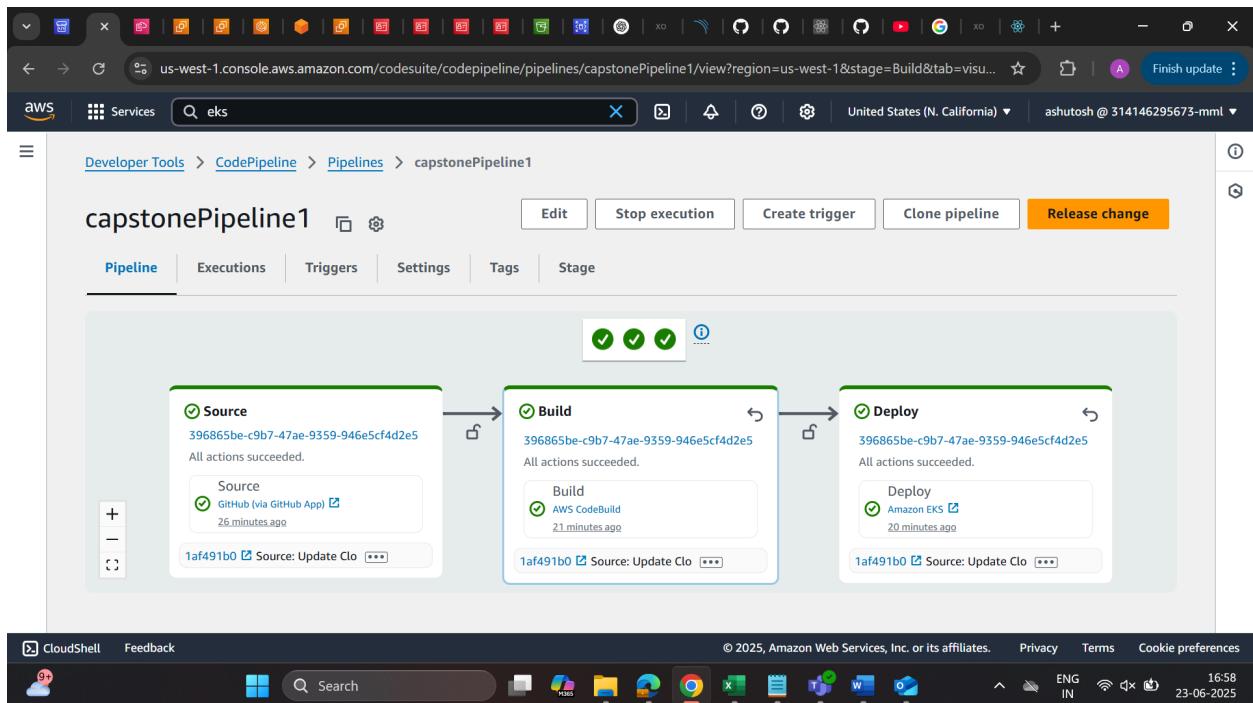
- Source: GitHub (CloudFormation.yaml)
- Build: Deploys stack using aws cloudformation deploy
- Resources: VPC, NAT Gateways, EKS cluster, RDS



Application Pipeline:

- Source: GitHub app repo
- Build:
  - SonarQube scan

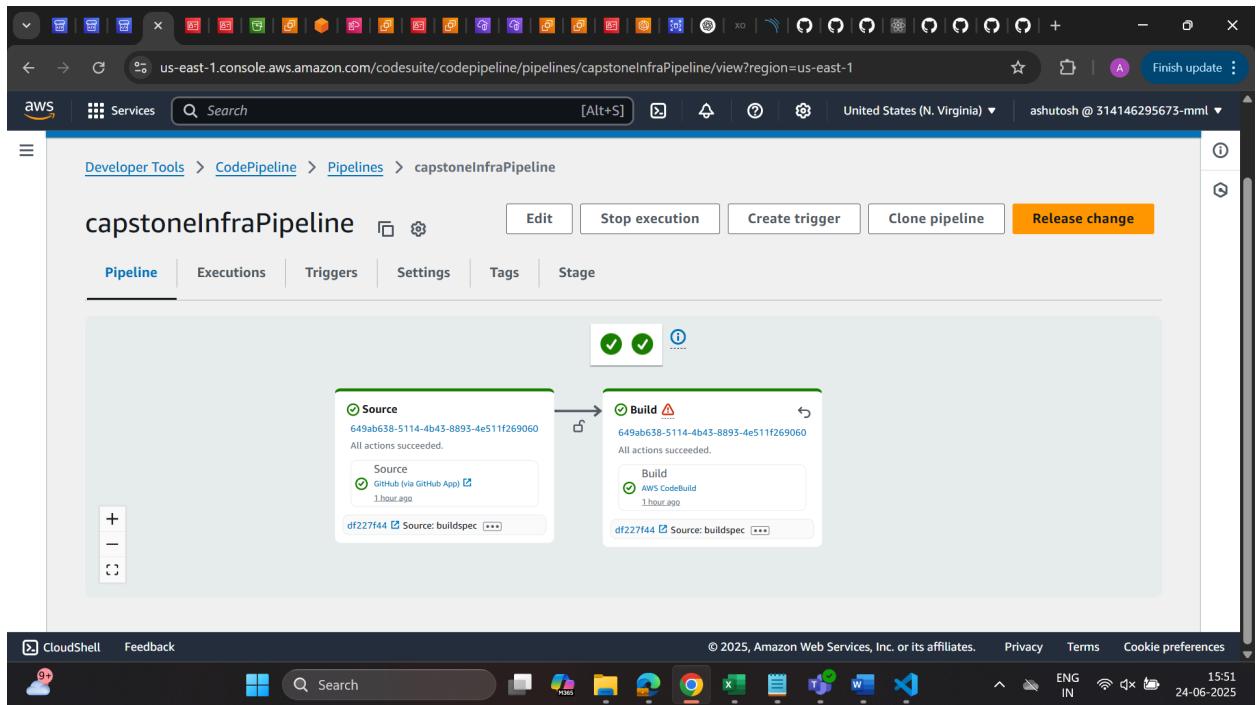
- Docker build for frontend/backend
- Trivy scan for both images
- Deploy:
  - Push to ECR
  - Apply Kubernetes manifests to EKS



## Region 2 – Terraform-based Deployment

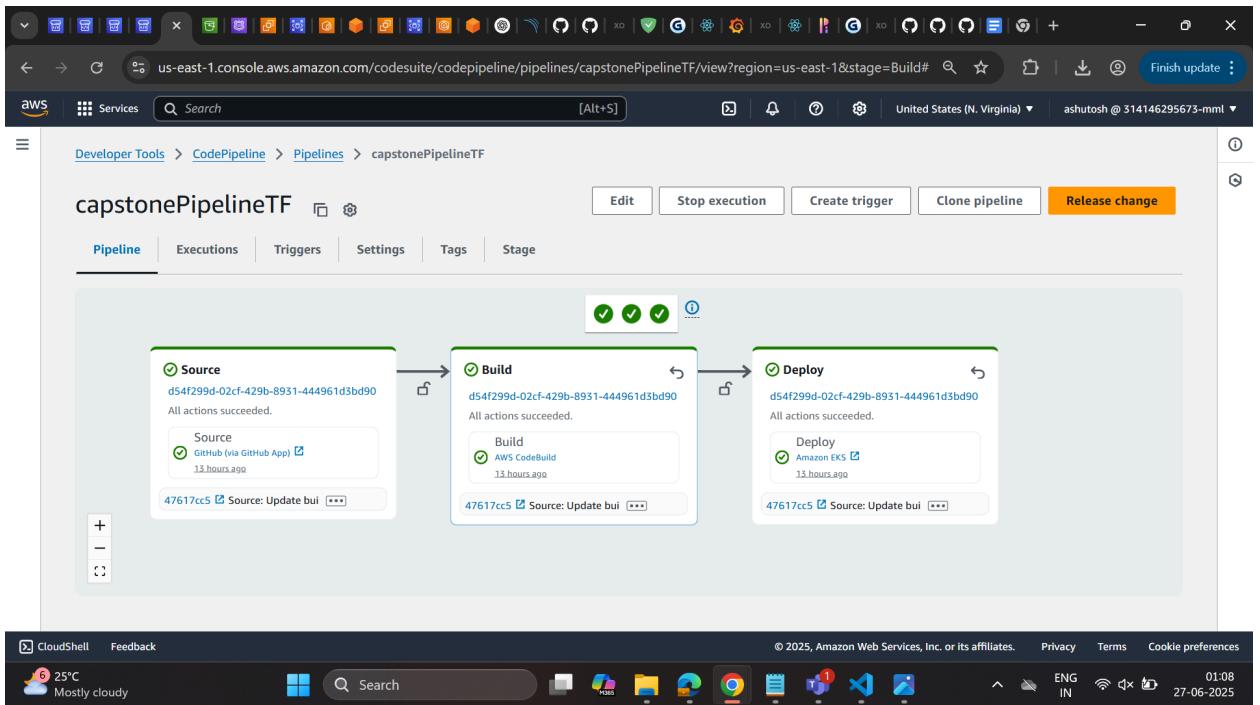
### Infrastructure Pipeline:

- Source: GitHub (`.tf` files)
- Build:
  - Initialize Terraform
  - Plan & apply to create infra (VPC, RDS, EKS)
- State: Managed via S3 backend
- Trigger: On merge to `infra` branch



## Application Pipeline:

- Reuses the same `buildspec.yaml` logic as Region 1
- Scans code → builds images → Trivy scans → deploys to Terraform-provisioned EKS



## 8. Phase 4: Route 53 Failover Configuration

Domain: studentteacher.threetierashutoshproject1197.xyz

- Primary Region: CloudFormation-based Load Balancer
- Secondary Region: Terraform-based Load Balancer
- Health Checks: On /health endpoint of frontend/backend
- Routing Policy: Failover
- Result: Fully automated disaster recovery via DNS redirection

The screenshot shows the AWS Route 53 Hosted Zone Details page for the zone `threetierashutoshproject1197.xyz`. The left sidebar contains navigation links for Route 53, Hosted zones, Health checks, Profiles, IP-based routing, Traffic flow, Domains, Resolver, and DNS Firewall. The main content area displays the 'Hosted zone details' for the selected zone. It includes sections for 'Records (4)', 'DNSSEC signing', and 'Hosted zone tags (0)'. A table lists four records:

Record name	Type	Routing	Alias	Value/Route traffic	TTL	Health	Evaluate	Record ID
<code>threetierashutoshproject1197.xyz</code>	NS	Simple	-	No	rs-1696.awsdns-19.co.uk rs-184.awsdns-23.co.uk rs-1034.awsdns-01.org rs-524.awsdns-07.net	172800	-	-
<code>threetierashutoshproject1197.xyz</code>	SOA	Simple	-	No	rs-1695.awsdns-19.co.uk. awsdns-hostmaster.amazon...	900	-	-
<code>studentteacher.threetierashutoshproject1197.xyz</code>	A	Failover	Primary	Yes	dualstack.k8s-mem-memalbd-620053284-208307...	-	ad458a1c...	101
<code>studentteacher.threetierashutoshproject1197.xyz</code>	A	Failover	Secondary	Yes	dualstack.k8s-mem-memalbd-620053284-1939974...	-	ad47078...	103

## 9. Phase 5: Monitoring with Prometheus & Grafana

### Prometheus Setup:

Deployed via Helm

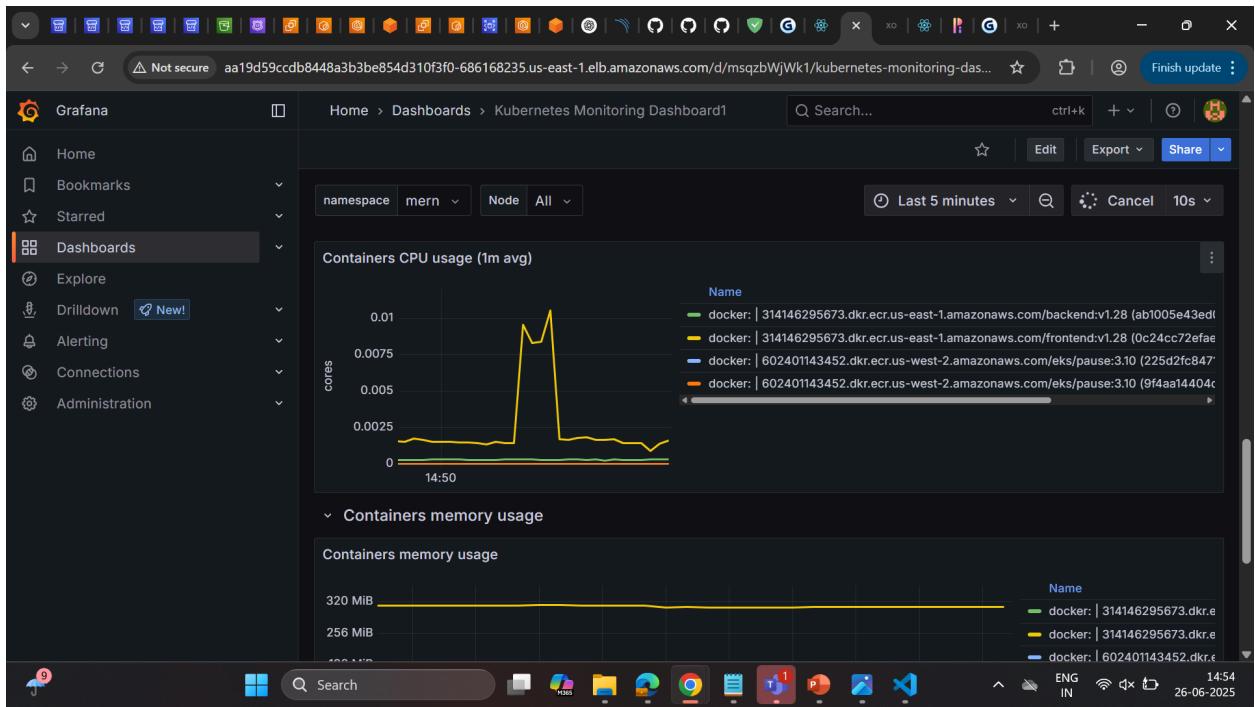
Scrapes metrics from:

- Kubernetes components
- Node.js backend
- React frontend
- kube-state-metrics and cAdvisor

### Grafana Setup:

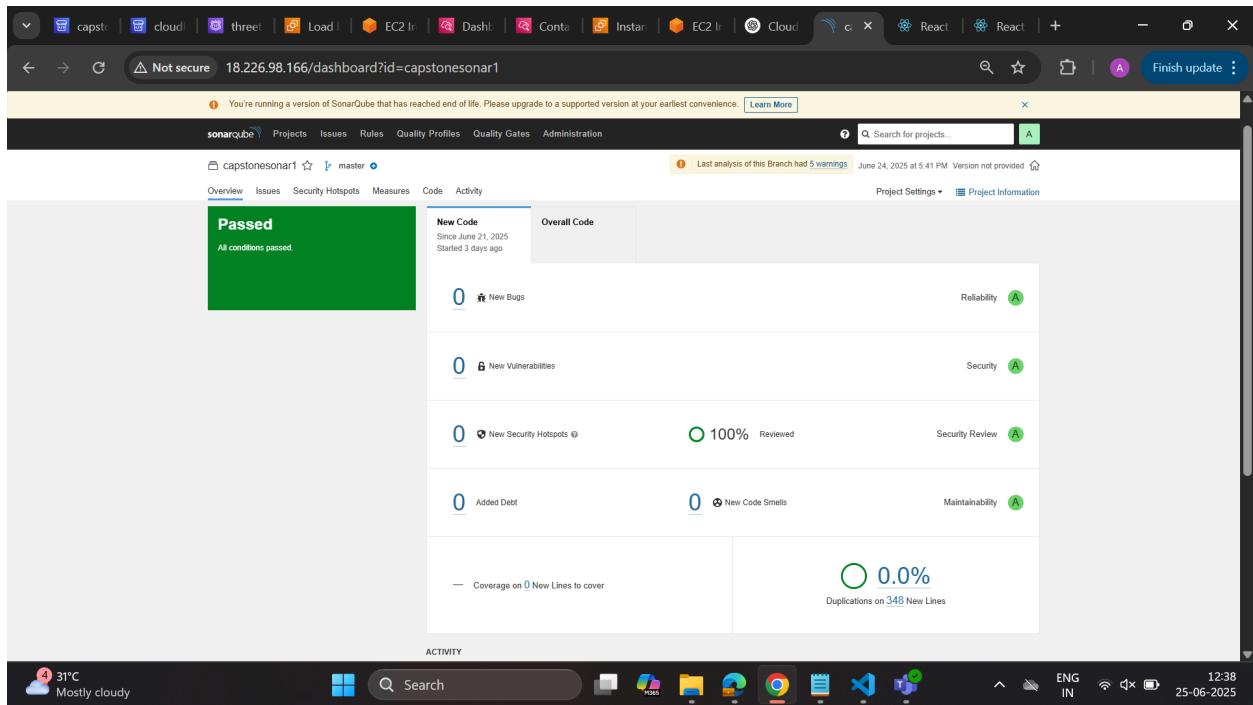
Preconfigured dashboards for:

- Pod metrics (CPU, memory)
- Request rate and errors
- MySQL database stats
- EKS node usage



## 10. SonarQube Integration

- Code quality scan during the build stage
- Integrated into CodeBuild pipeline
- Blocks deployment if quality gate fails



## 11. Trivy Security Scanning

- Pre-deployment container image scan
- Blocks builds on high/critical CVEs
- Enforced in buildspec.yaml for both frontend and backend

```

1140
1141 [Container] 2025/06/25 12:18:41.740817 Running command trivy image $BACKEND_IMAGE_URI --severity CRITICAL,HIGH
--format json --output k8s-out/trivy-backend-report.json || echo "Trivy scan for backend completed with
findings"
1142 2025-06-25T12:18:41Z  WARN  [vuulndb] Trivy DB may be corrupted and will be re-downloaded. If you manually
downloaded DB - use the `--skip-db-update` flag to skip updating DB.
1143 2025-06-25T12:18:41Z  INFO   [vuulndb] Need to update DB
1144 2025-06-25T12:18:41Z  INFO   [vuulndb] Downloading vulnerability DB...
1145 2025-06-25T12:18:41Z  INFO   [vuulndb] Downloading artifact...  repo="mirror.gcr.io/aquasec/trivy-db:2"
1146 41.67 MiB / 66.13 MiB [----->] 63.02% ? p/s ?66.13 MiB / 66.13 MiB [
1147 / 66.13 MiB [----->] 100.00% 40.70 MiB p/s ETA ?66.13 MiB / 66.13 MiB [
1148 ----->] 100.00% 40.70 MiB p/s ETA ?66.13 MiB / 66.13 MiB [
1149 ----->] 100.00% 38.08 MiB p/s ETA ?66.13 MiB / 66.13 MiB [
1150 ----->] 100.00% 38.08 MiB p/s ETA ?66.13 MiB / 66.13 MiB [
1151 ----->] 100.00% 35.62 MiB p/s ETA ?66.13 MiB / 66.13 MiB [
1152 ----->] 100.00% 33.11 MiB p/s 2.2s2025-06-25T12:18:44Z  INFO   [vuulndb] Artifact successfully downloaded
repo="mirror.gcr.io/aquasec/trivy-db:2"
1147 2025-06-25T12:18:44Z  INFO   [vuln] Vulnerability scanning is enabled
1148 2025-06-25T12:18:44Z  INFO   [secret] Secret scanning is enabled
1149 2025-06-25T12:18:44Z  INFO   [secret] If your scanning is slow, please try '--scanners vuln' to disable
secret scanning
1150 2025-06-25T12:18:44Z  INFO   [secret] Please see also
https://trivy.dev/v0.63/docs/scanner/secret#recommendation for faster secret detection
1151 2025-06-25T12:18:55Z  INFO   [python] Licenses acquired from one or more METADATA files may be subject to
additional terms. Use `--debug` flag to see all affected packages.
1152 2025-06-25T12:18:57Z  INFO   Detected OS family="debian" version="10.13"

```

## 12. Setup Instructions

### Prerequisites

- AWS CLI, Git, Docker
- IAM permissions for EKS, RDS, CodePipeline
- Access to SonarQube and Trivy

### Deploy Region 1 (CloudFormation)

1. Trigger CodePipeline from GitHub
2. CloudFormation provisions infrastructure and the app

### Deploy Region 2 (Terraform)

1. Deploy infra from TF-EKS-RDS repo
2. Run app deployment pipeline from Student-Teacher-Portal repo

## 13. Challenges Faced

- Managing IAM roles and RBAC for EKS and exporters
- Coordinating Route 53 failover validation
- Merging CloudFormation and Terraform environments
- Integrating open-source monitoring stack securely into EKS

## 14. Future Scope

- Add ArgoCD for GitOps delivery
- Introduce AWS Backup for managed RDS snapshots
- Extend functionality (quizzes, assignment uploads)
- Add Redis for performance optimization

## 15. Conclusion

This capstone demonstrates a modern, resilient, and secure AWS-based architecture for deploying a cloud-native Student-Teacher Portal. It showcases the best practices in IaC, CI/CD, observability, and regional failover — making it a production-grade DevOps project.

The screenshot shows a web application for managing student details. At the top right, there is a purple button labeled "Teacher". On the left, a blue button says "Back to Menu". In the center, a yellow button says "Student Details". A purple callout box labeled "Student details" points to a table listing student information. The table has columns for StudentID, Name, Roll Number, Class, and Actions (Delete). Two rows are shown: one for student ID 1, name aas, class A, and roll number 12; another for student ID 2, name Hab, class F, and roll number 43. Below the table is a form with fields for Name, Roll No (Must be Number), and Class, followed by a "Submit" button. The background features a photograph of a classroom full of students. The footer contains copyright information: "Developed by Ashutosh Sanghi" and "© 2025 Ashutosh Sanghi. All Rights Reserved." The browser status bar at the bottom indicates "Not secure" and the URL "k8s-mern-mernalbd-62c0b53284-1568972060.us-west-1.elb.amazonaws.com/student".

The screenshot shows a web application for managing teacher details. At the top right, there is a purple button labeled "Student". On the left, a blue button says "Back to Menu". In the center, a yellow button says "Teacher Details". A purple callout box labeled "Teacher details" points to a table listing teacher information. The table has columns for TeacherID, Name, Subject, Class, and Actions (Delete). One row is shown: for teacher ID 1, name Manish, subject Science, and class B. Below the table is a form with fields for Name, Subject, and Class, followed by a "Submit" button. The background features a photograph of a classroom full of students. The footer contains copyright information: "Developed by Ashutosh Sanghi" and "© 2025 Ashutosh Sanghi. All Rights Reserved." The browser status bar at the bottom indicates "Not secure" and the URL "k8s-mern-mernalbd-62c0b53284-1568972060.us-west-1.elb.amazonaws.com/teacher".