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

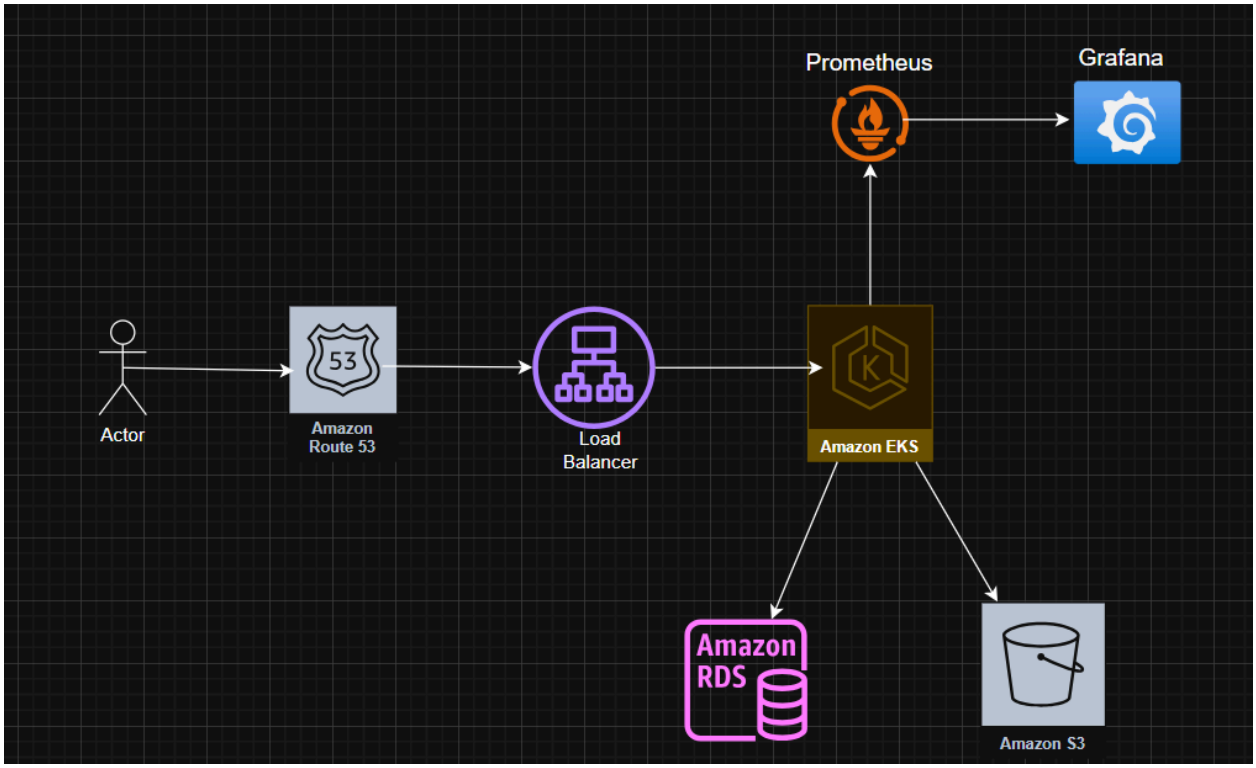
1. Introduction

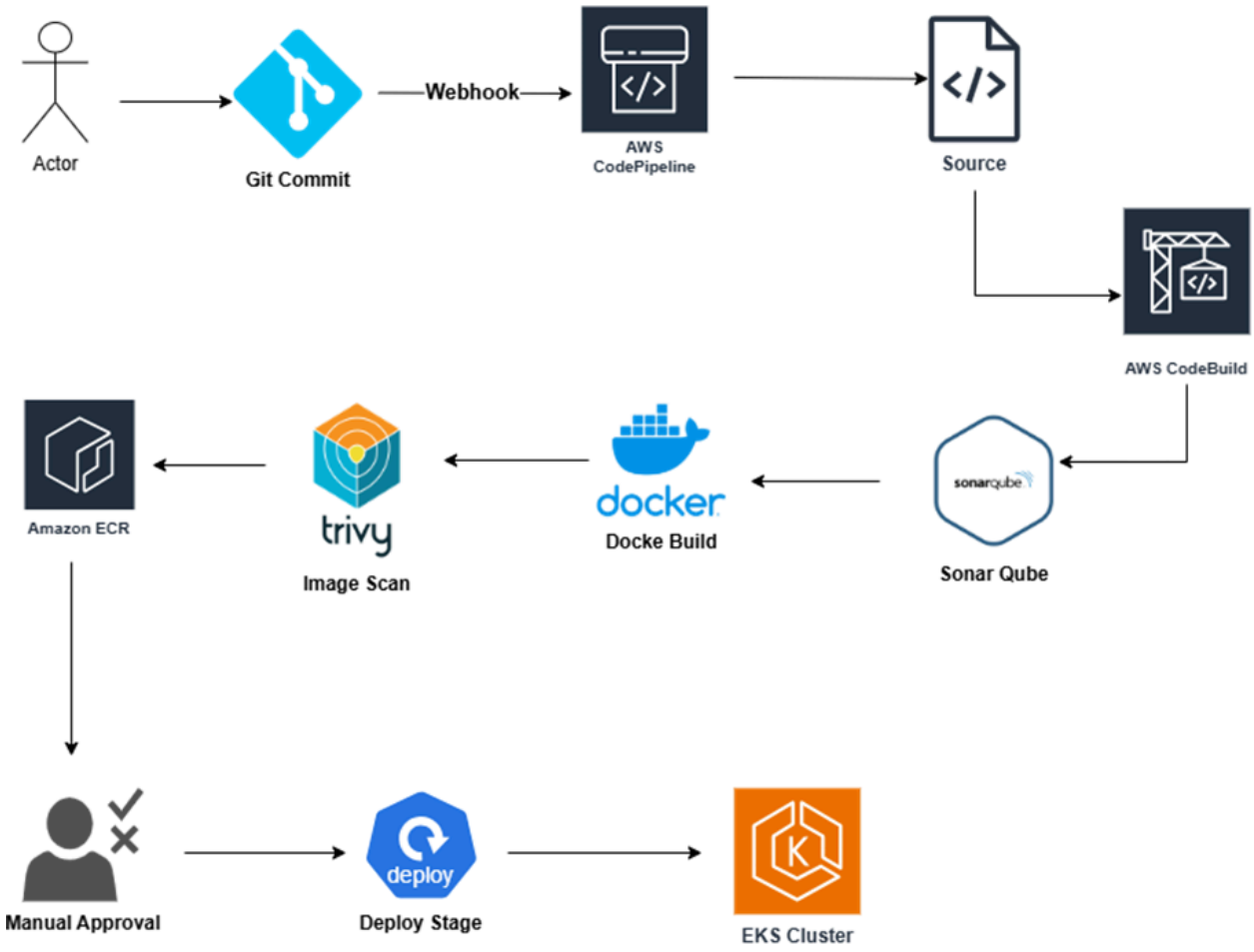
This project delivers a fully automated, highly available, multi-region Student-Teacher Portal deployed on AWS using EKS, CloudFormation, Terraform, and CodePipeline. It features CI/CD, infrastructure as code, container security, and real-time monitoring via Prometheus and Grafana, replacing traditional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




Actor
Git Commit
Webhook
AWS
CodePipeline
Source
AWS CodeBuild
Sonar Qube
Docke Build
Image Scan
Amazon ECR
Manual Approval
deploy
Deploy Stage
EKS Cluster

3. Tools and Technologies Used

Category	Tools/Technologies
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

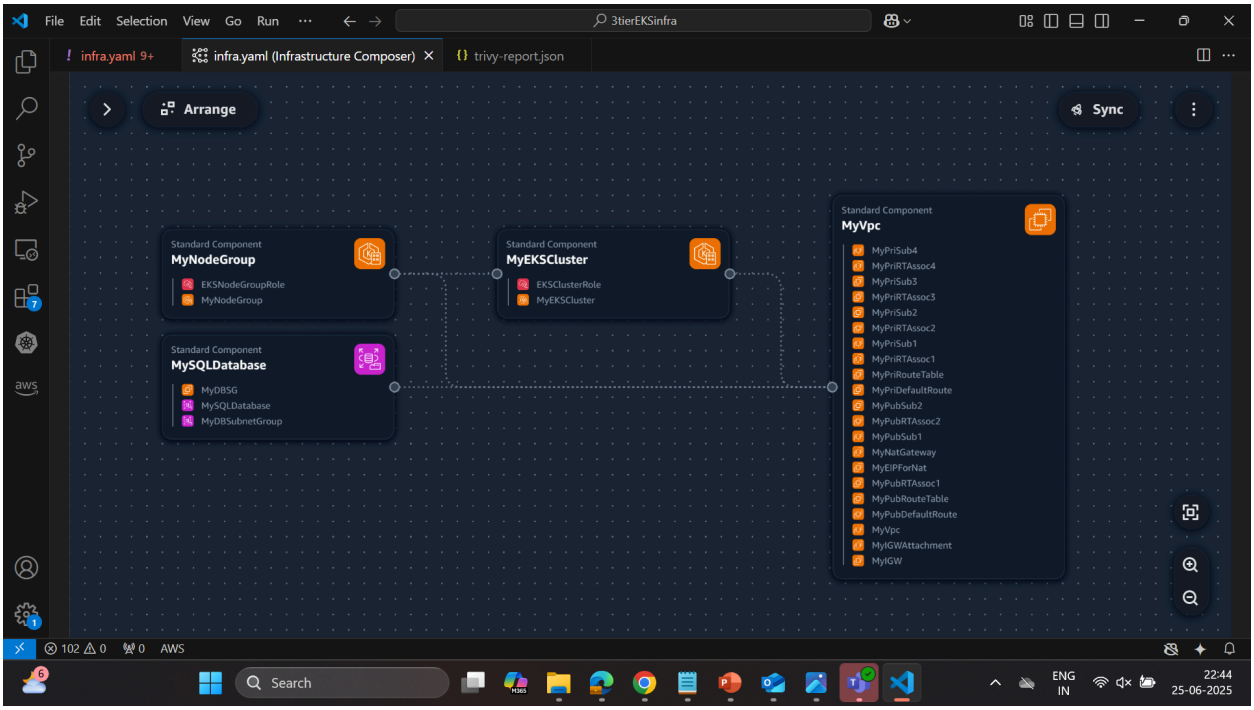
5. Phase 1: EKS Cluster Deployment

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

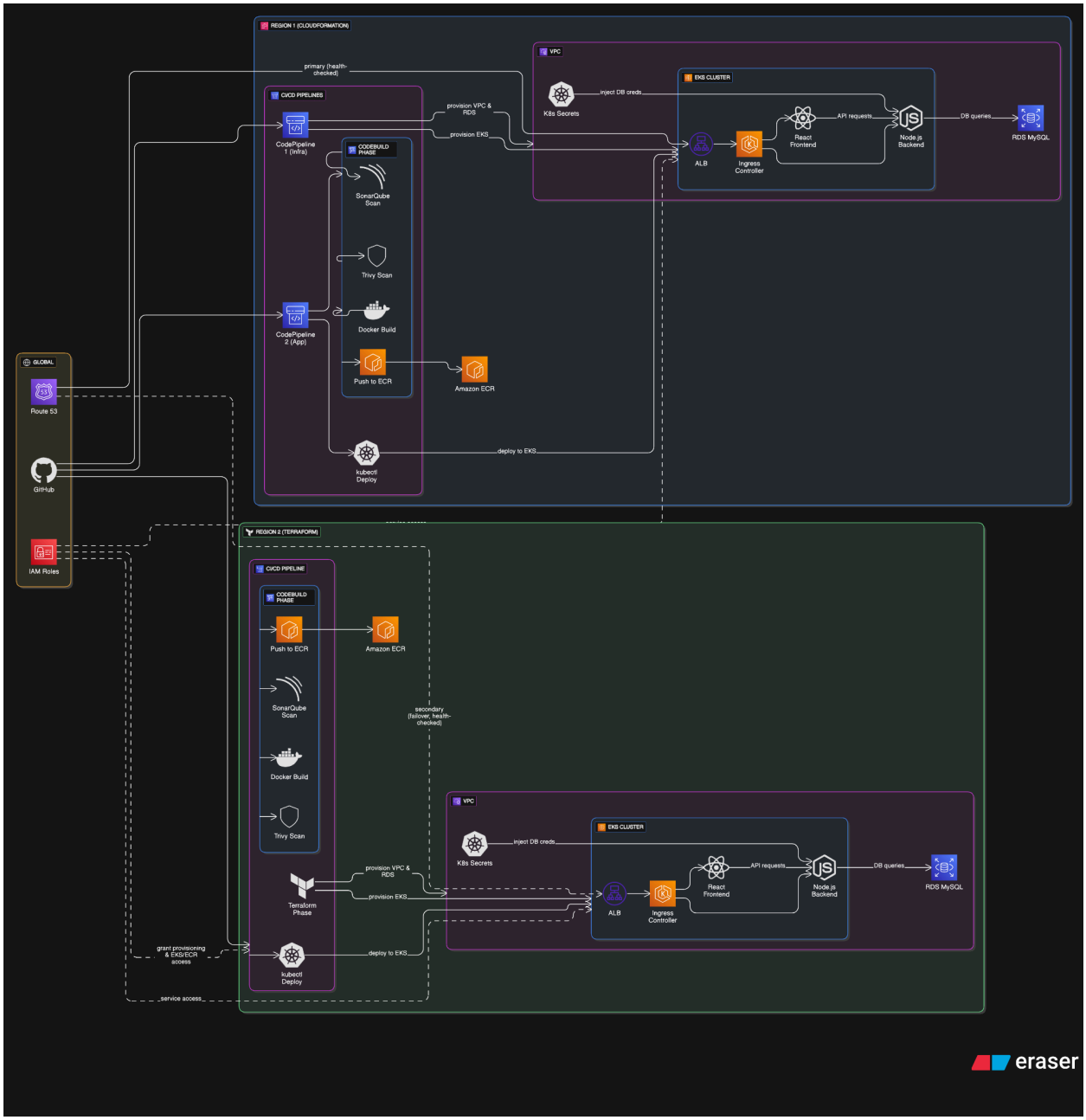
Implemented via AWS CodePipeline in both regions. GitHub Repo:

🔗 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

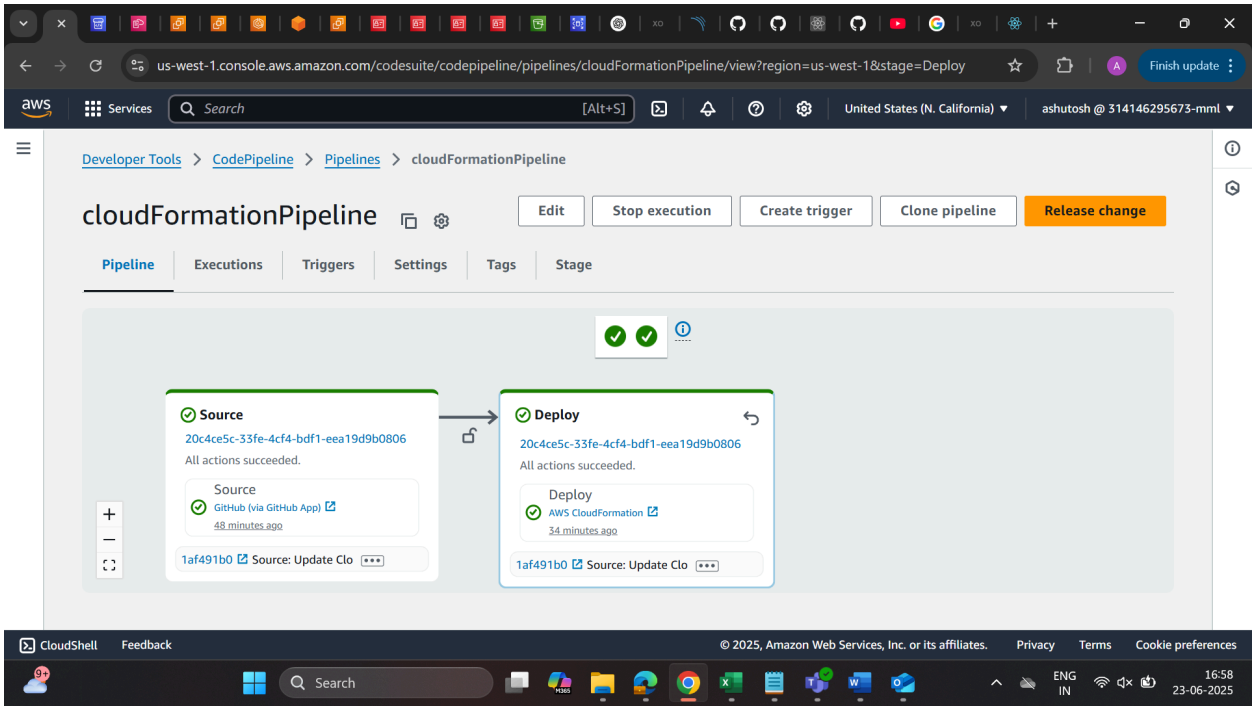

REGION 1 (CLOUDFORMATION)
VPC
primary (health-checked)
CI/CD PIPELINES
provision VPC & RDS
provision EKS
K8s Secrets
inject DB creds
EKS CLUSTER
CodePipeline 1 (Infra)
CODEBUILD PHASE
SonarQube Scan
Trivy Scan
Docker Build
Push to ECR
Amazon ECR
ALB
Ingress Controller
React Frontend
API requests
Node.js Backend
DB queries
RDS MySQL
CodePipeline 2 (App)
kubectl Deploy
deploy to EKS
GLOBAL
Route 53
GitHub
IAM Roles
REGION 2 (TERRAFORM)
CI/CD PIPELINE
secondary (failover, health-checked)
Terraform Phase
grant provisioning & EKS/ECR access
service access
eraser

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

Region 1 – CloudFormation-based Deployment

Infrastructure Pipeline:

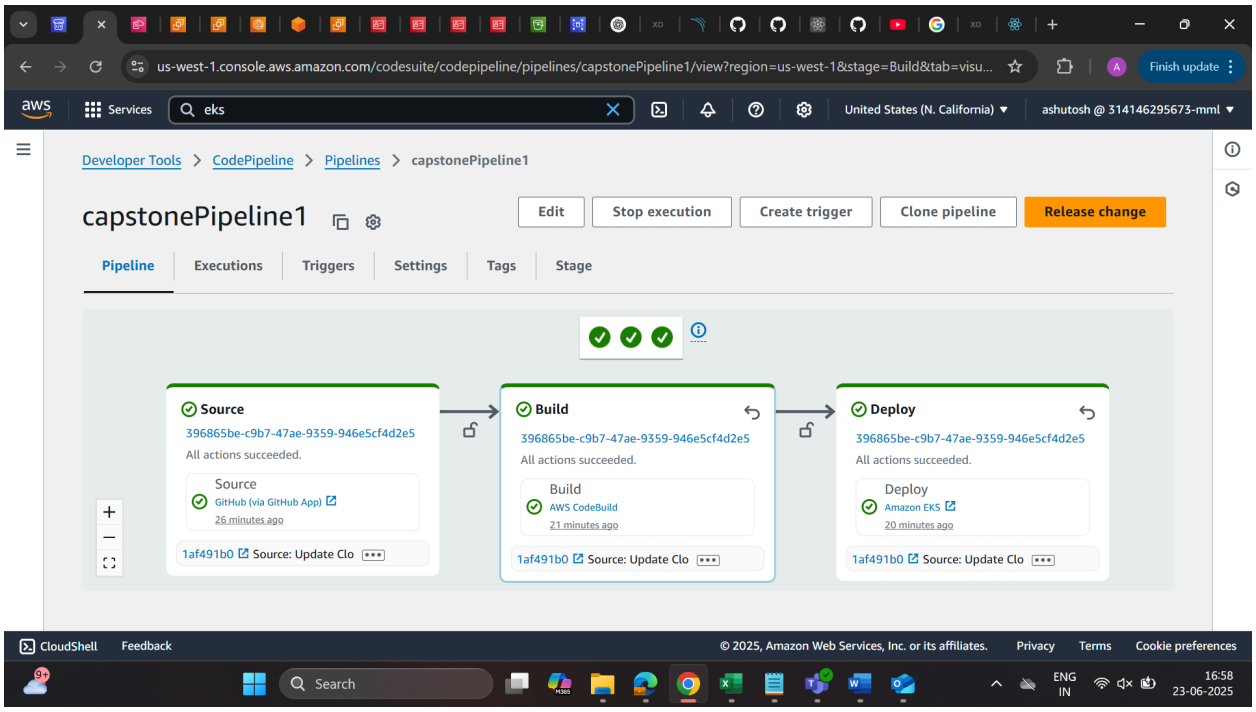
- Source: GitHub (CloudFormation.yaml)
- Build: Deploys stack using aws cloudformation deploy
- Resources: VPC, NAT Gateways, EKS cluster, RDS



Application Pipeline:

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

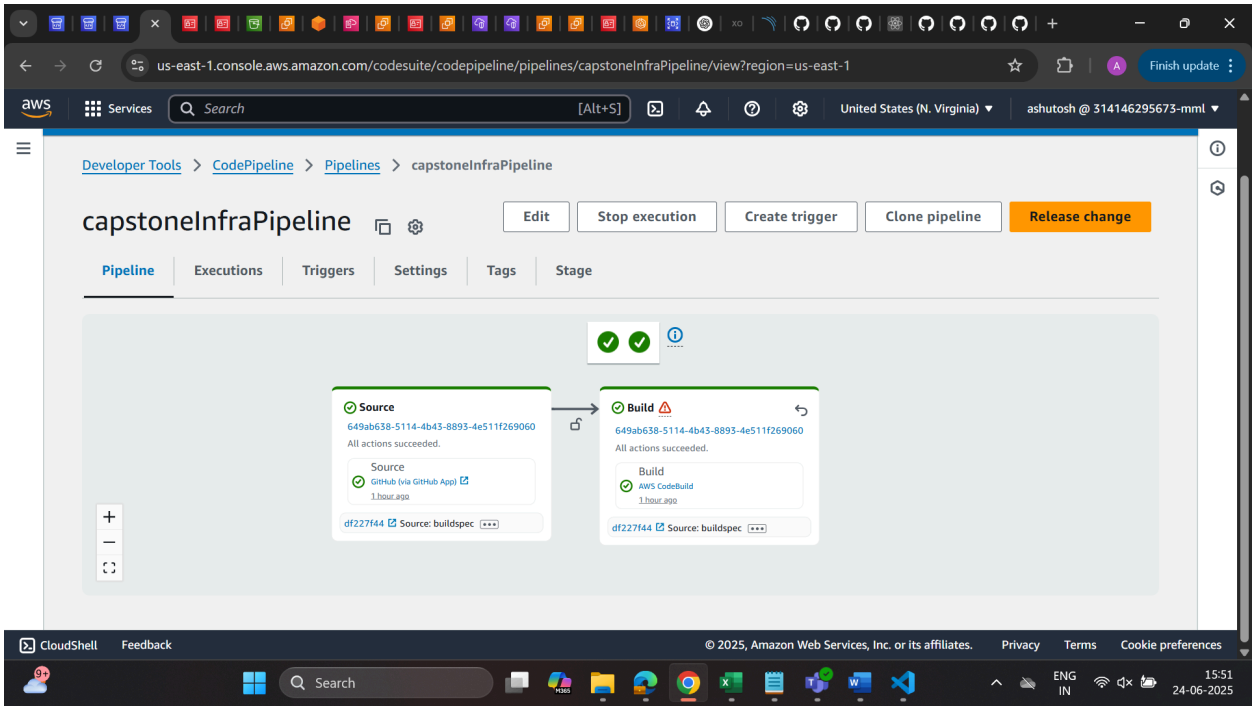
    - Docker build for frontend/backend
    - Trivy scan for both images
- Deploy:
    - Push to ECR
    - Apply Kubernetes manifests to EKS



Region 2 – Terraform-based Deployment

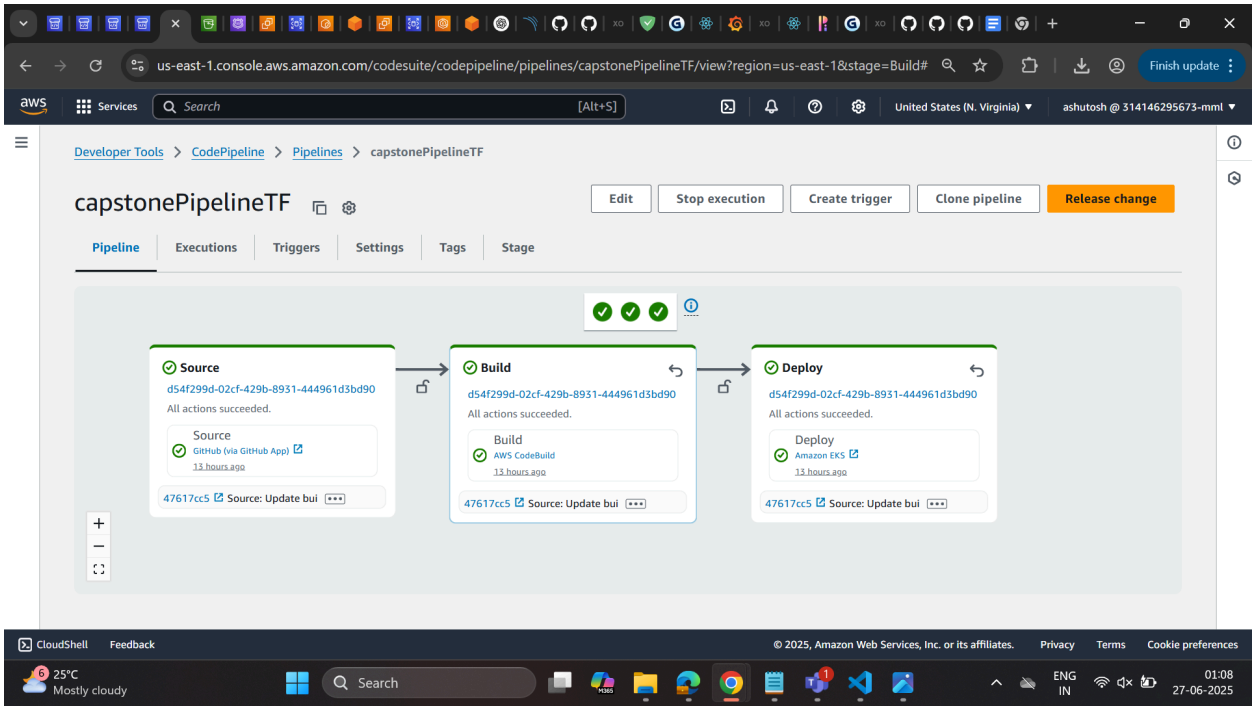
Infrastructure Pipeline:

- 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:

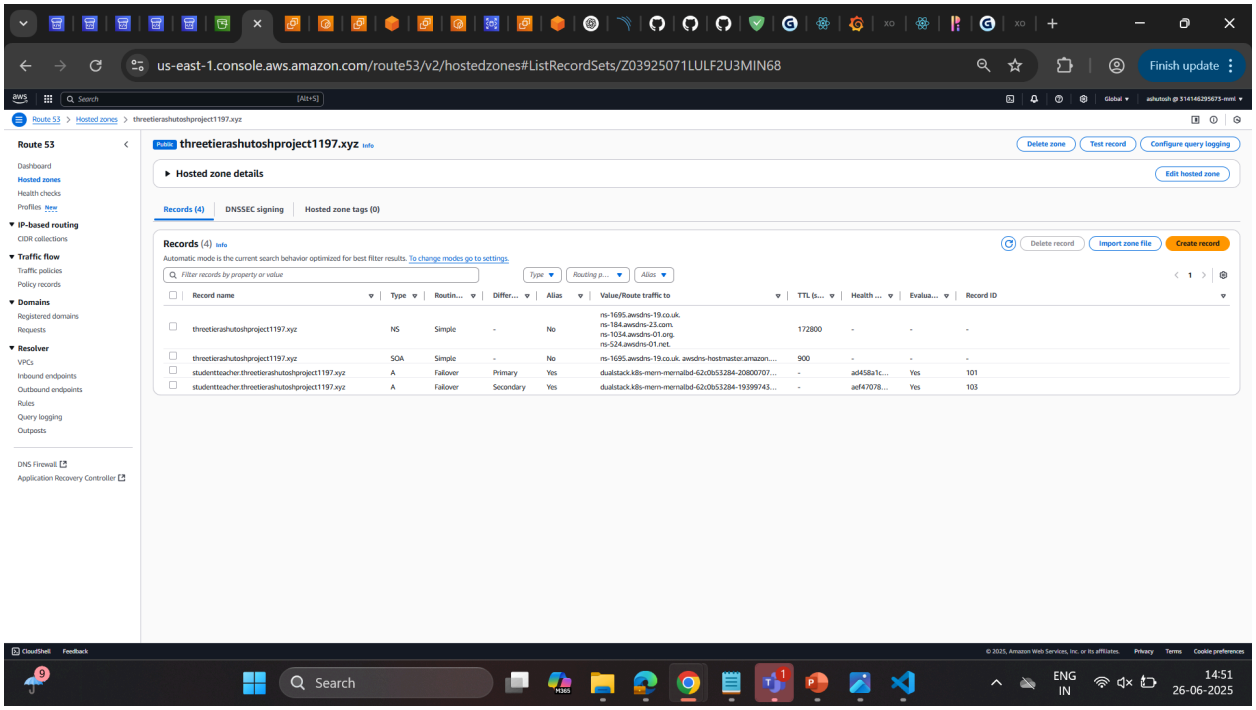
- Reuses the same `buildspec.yaml` logic as Region 1
- Scans code → builds images → Trivy scans → deploys to Terraform-provisioned EKS



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



9. Phase 5: Monitoring with Prometheus & Grafana

Prometheus Setup:

Deployed via Helm

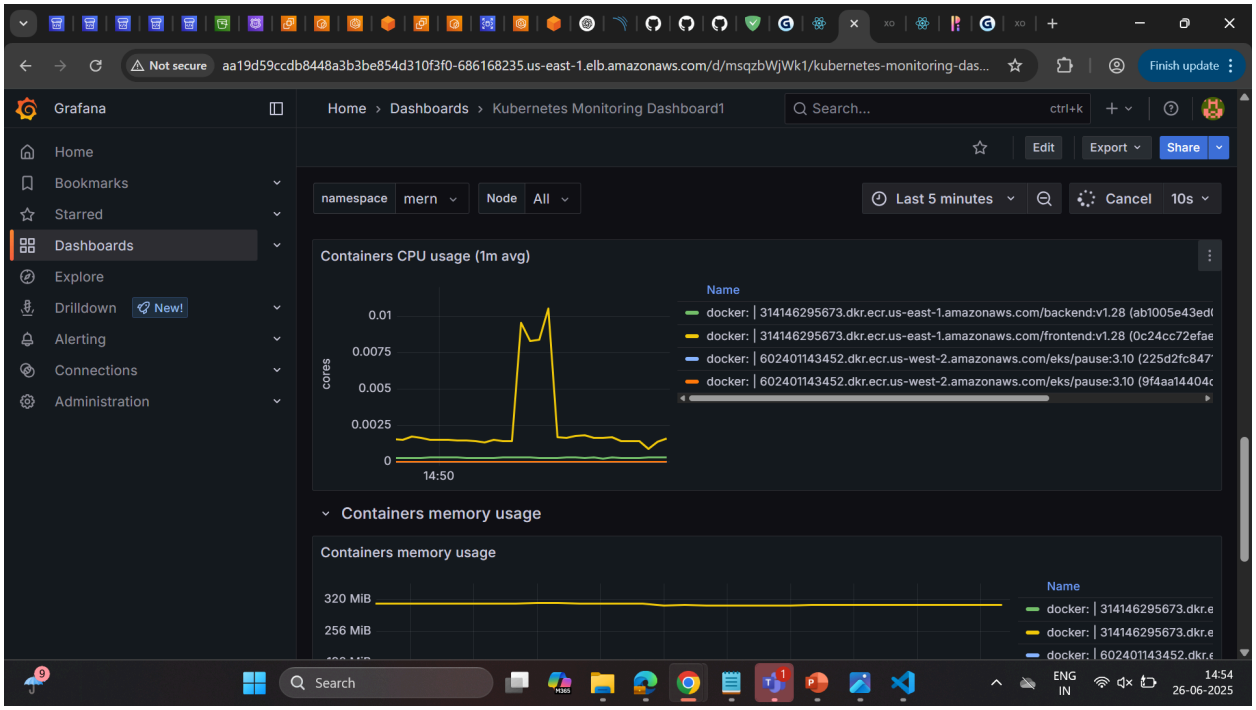
Scrapes metrics from:

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

Grafana Setup:

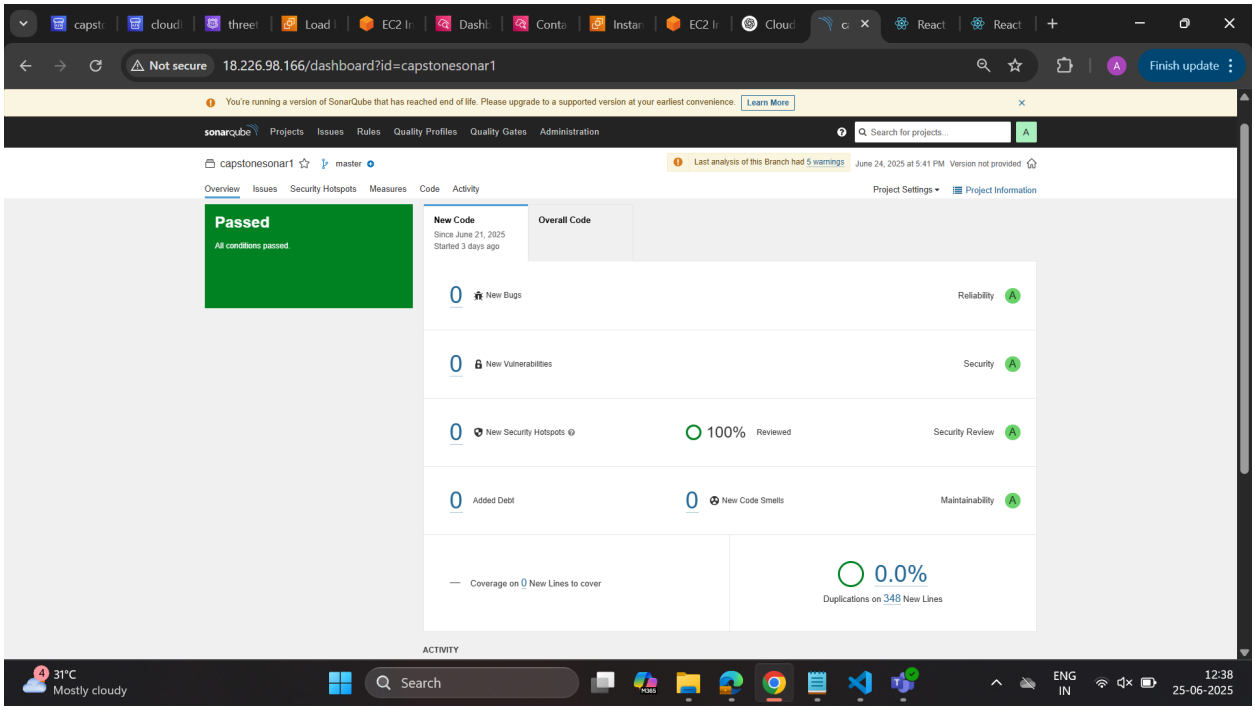
Preconfigured dashboards for:

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



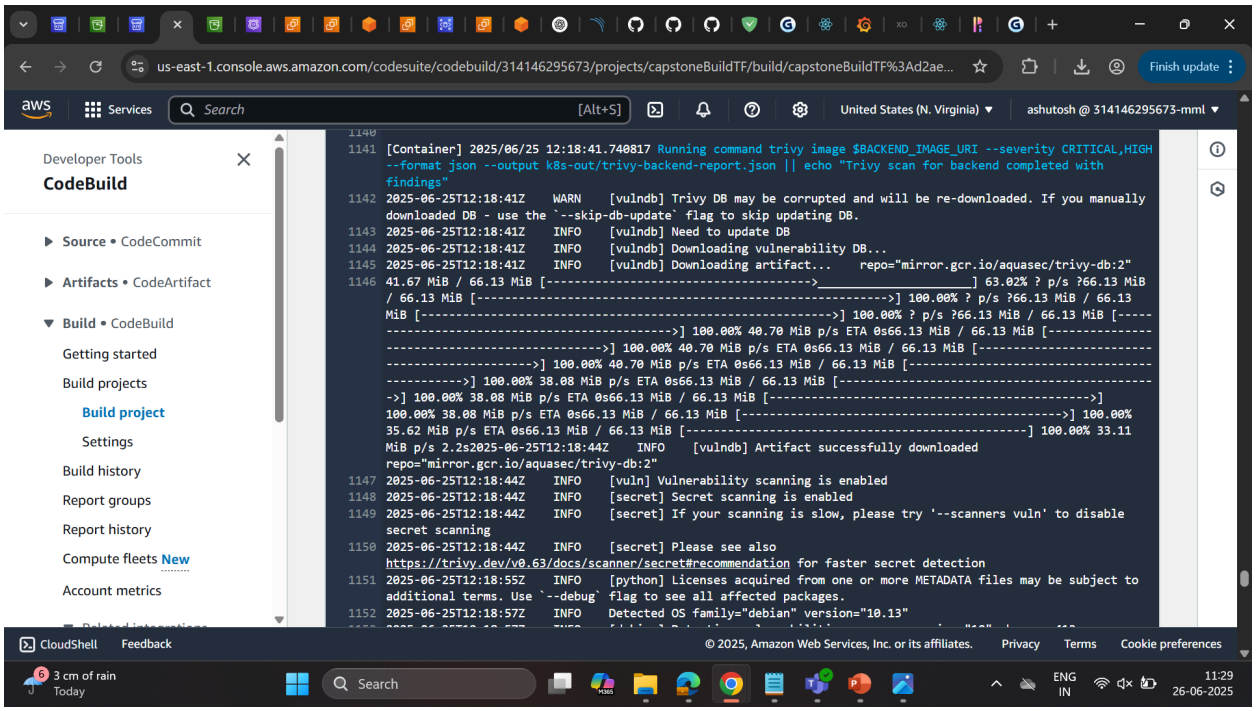
10. SonarQube Integration

- 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



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

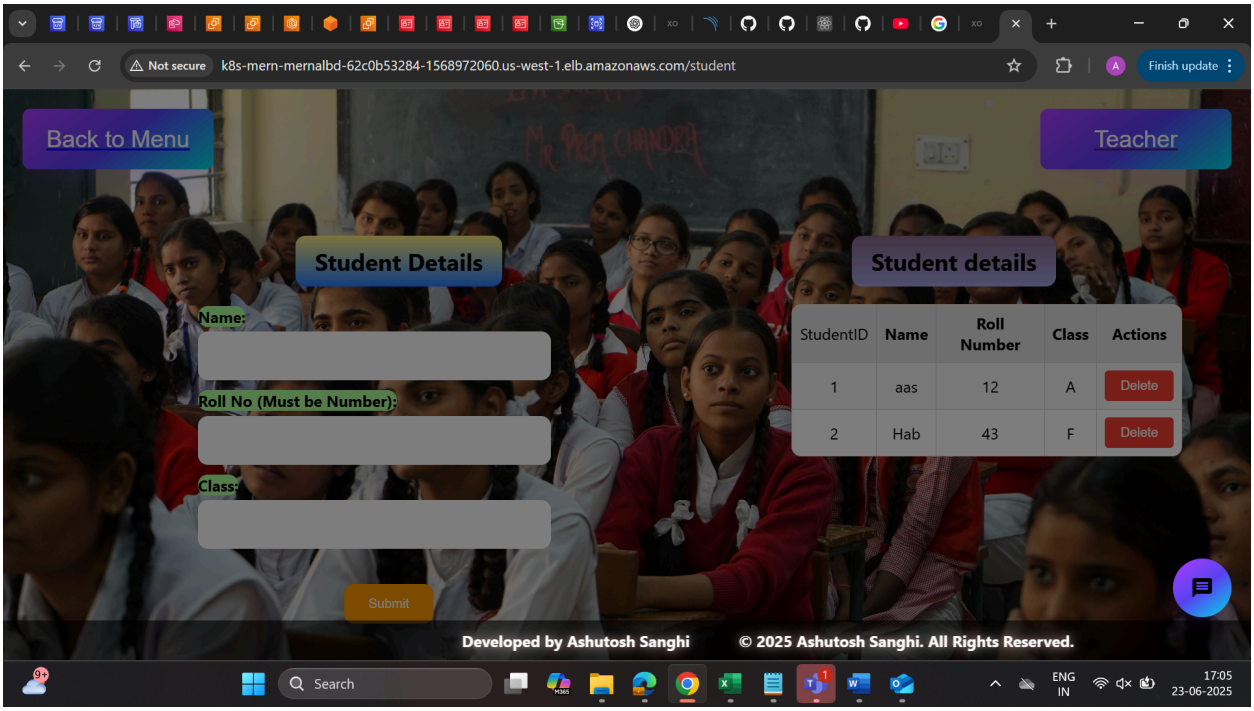
- Managing IAM roles and RBAC for EKS and exporters
- Coordinating Route 53 failover validation
- Merging CloudFormation and Terraform environments
- Integrating open-source monitoring stack securely into EKS

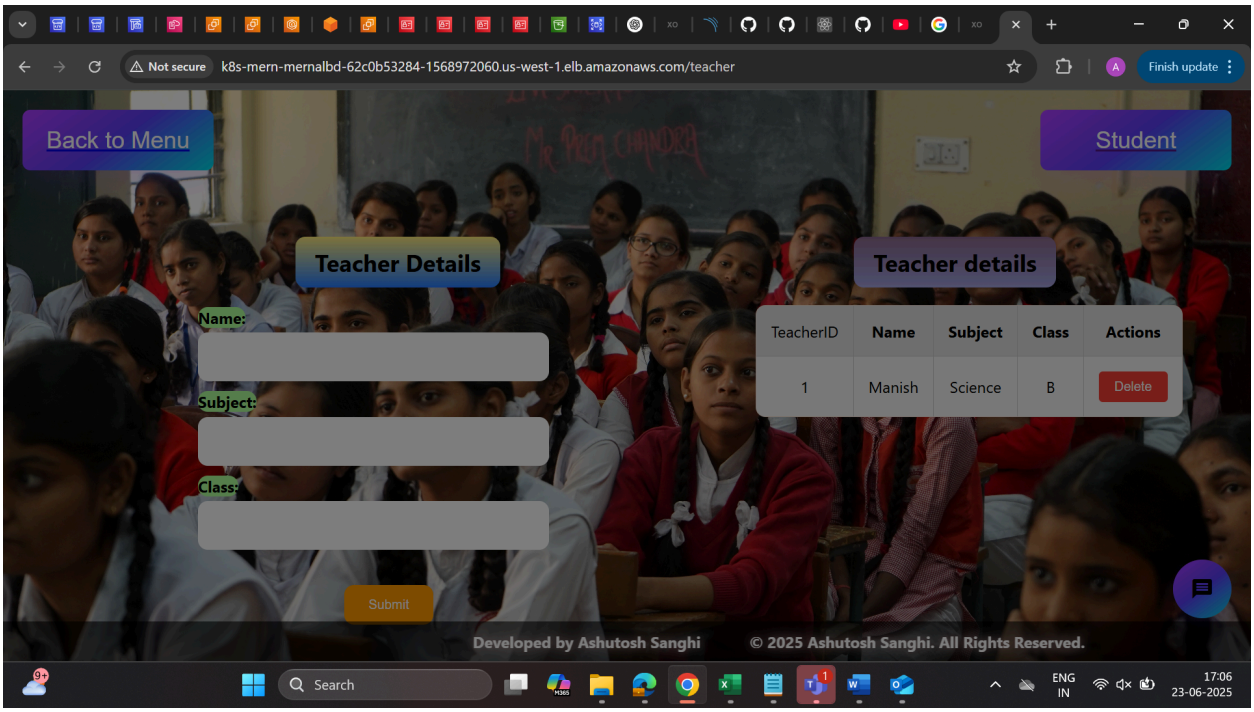
14. Future Scope

- 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.


Not secure
k8s-mern-mernalbd-62c0b53284-1568972060.us-west-1.elb.amazonaws.com/student
Finish update
Back to Menu
Teacher
Student Details
Name:
Roll No (Must be Number):
Class:
Submit
Student details
StudentID
Name
Roll Number
Class
Actions
1
aas
12
A
Delete
2
Hab
43
F
Delete
Developed by Ashutosh Sanghi
© 2025 Ashutosh Sanghi. All Rights Reserved.
Search
ENG
IN
17:05
23-06-2025


Not secure
k8s-mern-mernalbd-62c0b53284-1568972060.us-west-1.elb.amazonaws.com/teacher
Finish update
Back to Menu
Student
Teacher Details
Name:
Subject:
Class:
Submit
Teacher details
TeacherID
Name
Subject
Class
Actions
1
Manish
Science
B
Delete
Developed by Ashutosh Sanghi
© 2025 Ashutosh Sanghi. All Rights Reserved.
Search
ENG
IN
17:06
23-06-2025