AWS DevOps Project Report

Project Title:

Multi-Region 3- tier application EKS Deployment Using CloudFormation and Terraform with CI/CD Pipeline

KARANAM YESWANTH TEJA – 289224

1. Project Overview

This project demonstrates the implementation of a complete DevOps pipeline on AWS by leveraging infrastructure-as-code tools (CloudFormation and Terraform) to deploy a 3-tier application across two AWS regions using Amazon EKS. It also incorporates CI/CD practices, EKS clusters, and Ingress NGINX for seamless access and routing. It includes containerized deployments on EKS, CI/CD pipelines, code quality analysis with SonarQube, DNS management with Route 53, and monitoring via AWS CloudWatch, Grafana, EventBridge.

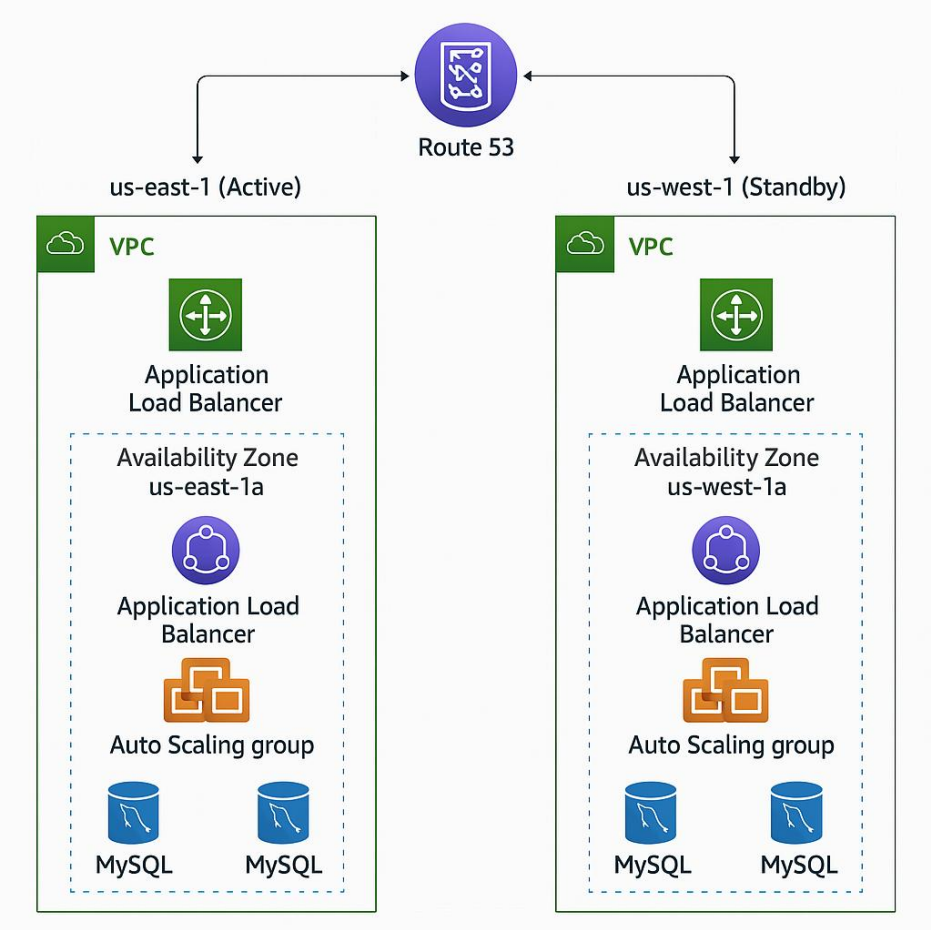
2. Objective

- Deploy a full-stack 3-tier application on AWS EKS in two different regions.
- Use **CloudFormation** and **Terraform** as IaC tools.
- Set up CI/CD pipelines for both frontend and backend.
- Implement scalable and secure infrastructure.
- Store and retrieve user data using a cloud-based database connected to the backend.
- Maintain high code quality using SonarQube.
- Ensure observability using CloudWatch,Grafana, EventBridge with Lambda function.
- Provide a user-friendly DNS with Route 53.

3. Tools & Technologies Used

Category	Tools/Services
Cloud Provider	AWS (Amazon Web Services)
IaC Tools	AWS CloudFormation, Terraform
Compute	Amazon EKS (Elastic Kubernetes Service)
Networking	VPC, Subnets, Route Tables, IGW/NAT
CI/CD	AWS CodePipeline, CodeBuild
Ingress	NGINX Ingress Controller
Database	RDS (MySQL)
Monitoring	CloudWatch, Prometheus, Grafana, EventBridge with lambda function
Frontend	Angular
Backend	Spring Boot
DNS	AWS Route 53
Code Quality	SonarQube

4. Architecture Diagram



5. Deployment Strategy

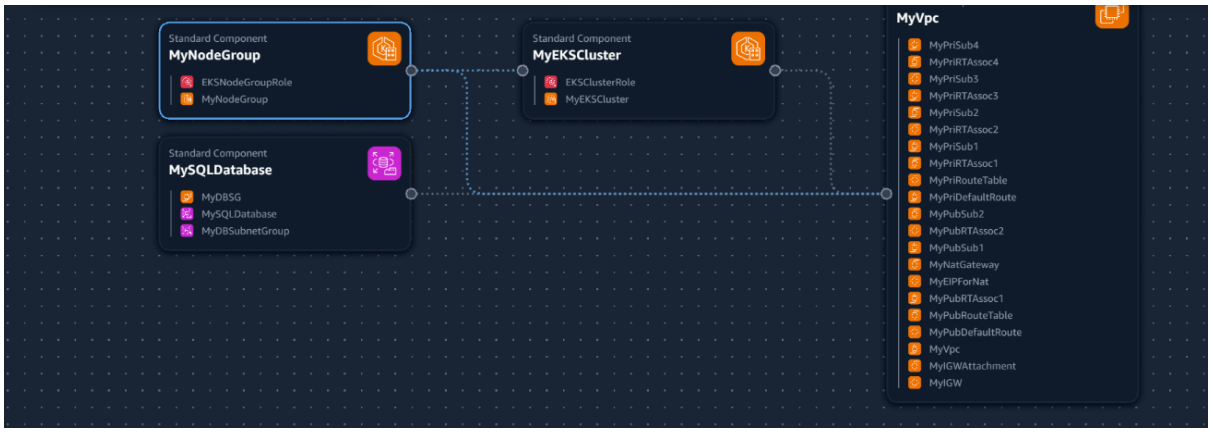
Region 1 (us-east-1) - Using CloudFormation

- Created infrastructure using a CloudFormation template:
    - VPC
    - Public and private subnets
    - Route tables
    - Internet and NAT gateways
    - EKS Cluster
    - RDS Database
- Deployed:
    - Backend application connected to the RDS DB

    - Frontend application connected to backend
- CI/CD Pipelines:
    - **Pipeline 1**: CloudFormation Backend Repo (Build + Deploy on EKS)
    - **Pipeline 2**: CloudFormation Frontend Repo (Build + Deploy on EKS)
- DNS: Route 53 for custom domain access
- Monitoring: CloudWatch for EKS and DB metrics

Region 2 (us-west-1) - Using Terraform

- Created similar infrastructure using Terraform:
    - Modular setup for VPC, subnets, routing, EKS, and RDS
- Deployed:
    - Backend application using kubectl with Terraform
- CI/CD Pipeline:
    - **Pipeline 3**: Terraform Backend Repo (Deploy)
    - **Pipeline 4**: Terraform Frontend Repo
- Monitoring: CloudWatch integrated for logs and metrics



6. Application Functionality

- **Frontend**: Accessible via Ingress NGINX with a Load Balancer.
    - Collects user data (e.g., name, email)
    - Sends POST/GET requests to backend service
- **Backend**: API server hosted on EKS
    - Handles CRUD operations and business logic
    - Connected securely to the RDS instance for data persistence
- **Ingress**: Configured for path-based routing, SSL (optional), and access control

7. CI/CD Pipeline Flow

Pipeline Stages (per repo):

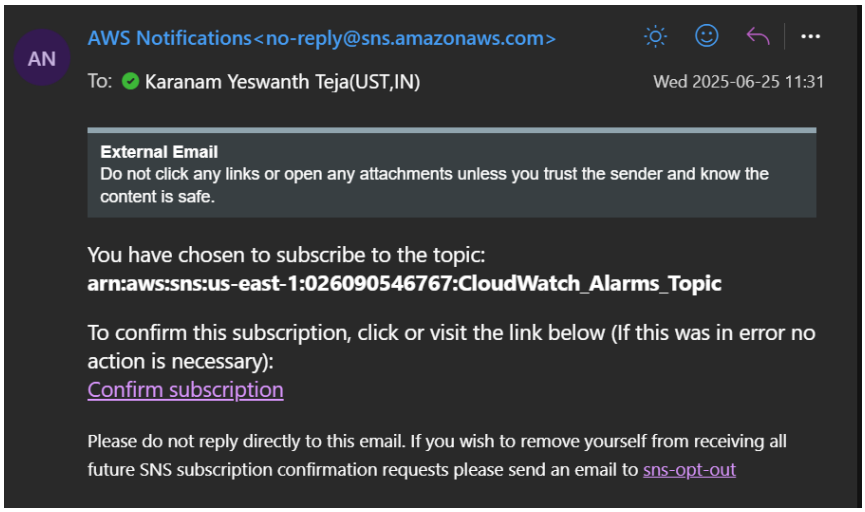
- **Source**: GitHub
- **Code Quality Stage**: SonarQube Scanner integration during CodeBuild
- **Build**: Using AWS CodeBuild to build images or prepare manifests
- **Deploy**:
    - Use kubectl and aws eks update-kubeconfig
    - Apply manifests to EKS cluster
    - Roll out deployments
- **Artifacts**: Docker images (optional), Kubernetes YAMLs
- **Monitoring Alerts**: CloudWatch alarms for pod failures, CPU, memory thresholds

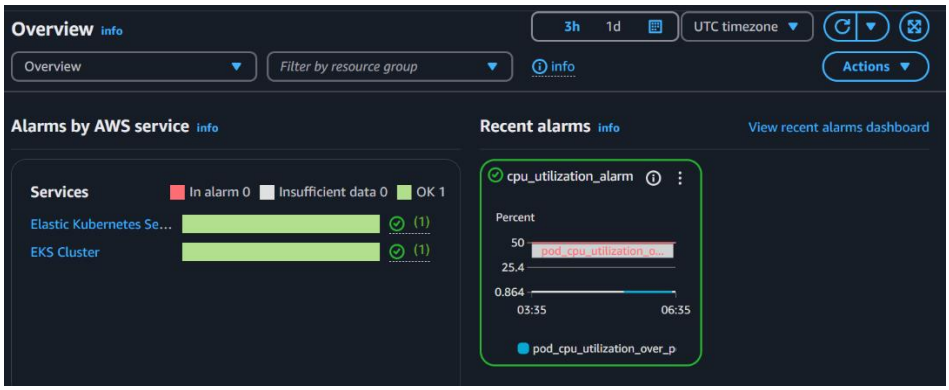
8. Monitoring and Observability

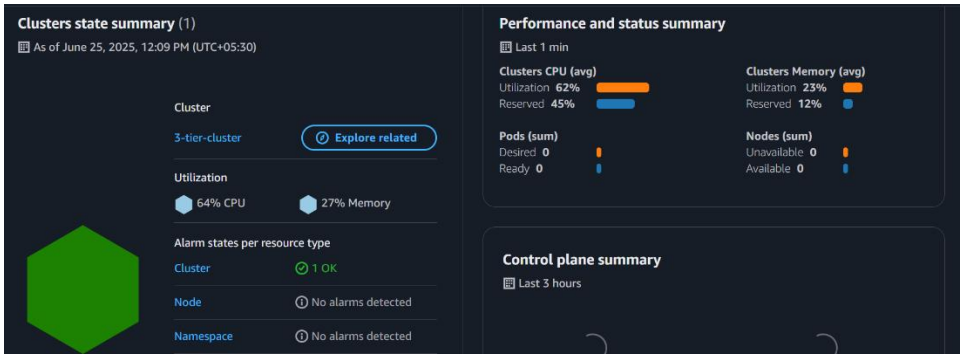
8.1 Amazon CloudWatch (Region: us-east-1)

- **Monitors:**
    - **EKS cluster nodes and pods**
    - **Amazon RDS (MySQL) instances**
    - **Backend application logs**

- **Log aggregation:**
    - **Collects logs from backend services using CloudWatch Agent or Fluent Bit.**
- **Alarms configured for:**
    - **High CPU / Memory usage on EKS pods**
    - **Database connectivity failures**
    - **CrashLoopBackOff or app restarts**







8.2 Grafana (Region: us-west-1)

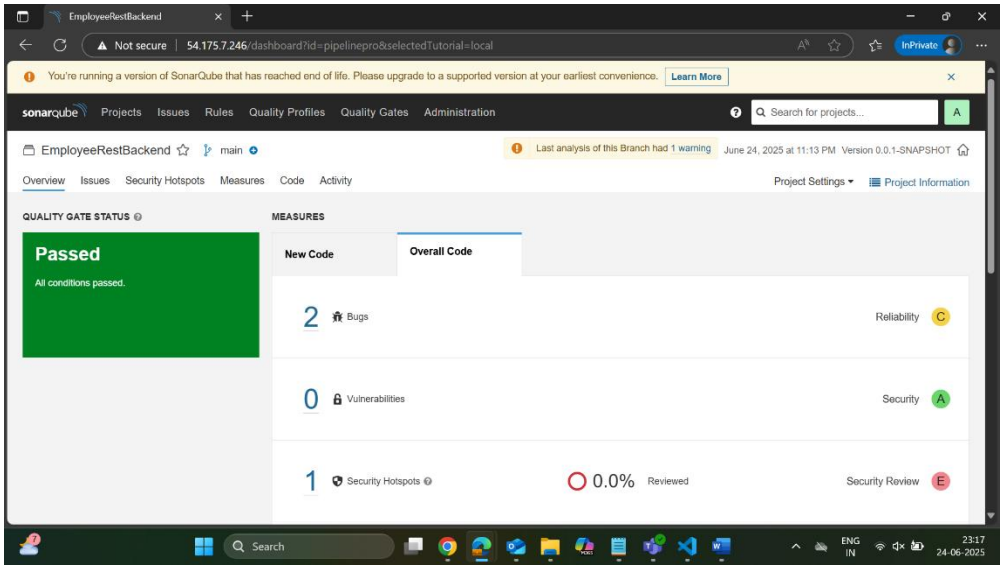
- **Deployed in the secondary region for custom metrics visualization**
- **Connected to Prometheus or CloudWatch data sources for:**
    - **Real-time dashboards of pod status, app latency, resource usage**
    - **Historical performance tracking**

8.3 Lambda + SNS

- **Monitors AWS CodePipeline events**
- **Triggers a Lambda function on pipeline failure**
- **Lambda publishes failure notification to an SNS topic**
- **Sends email alerts to stakeholders/developers for immediate response**

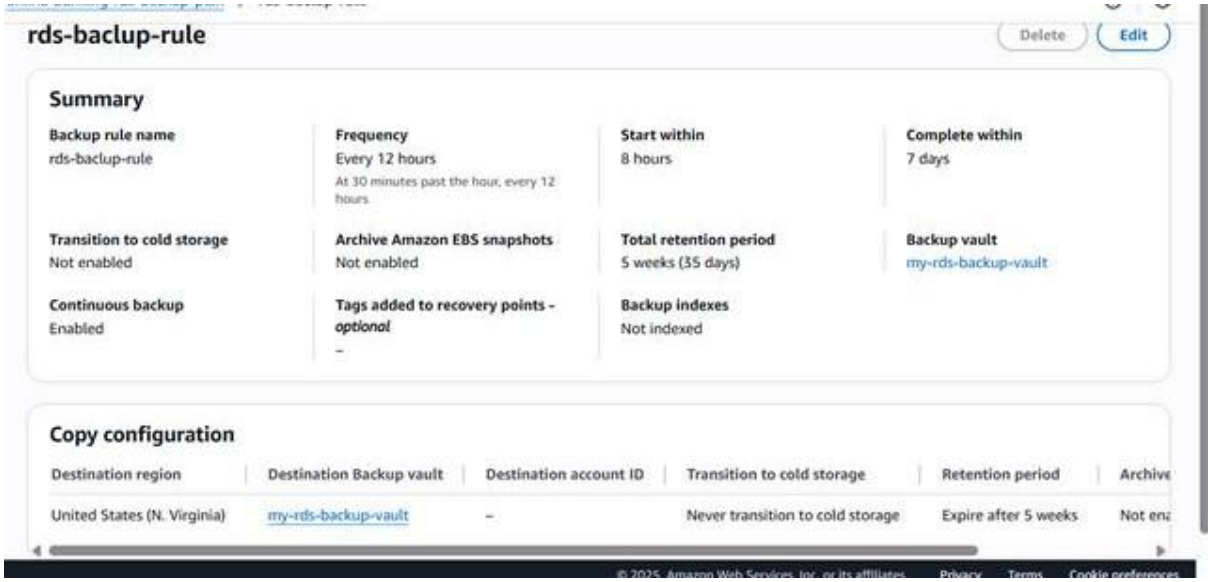
9. Code Quality and Security

- **SonarQube**:
    - Integrated into CodeBuild stage
    - Scans for code smells, bugs, vulnerabilities
    - Enforces code quality gates before deployment
    - Ensures best practices in JavaScript, Python, or backend language used



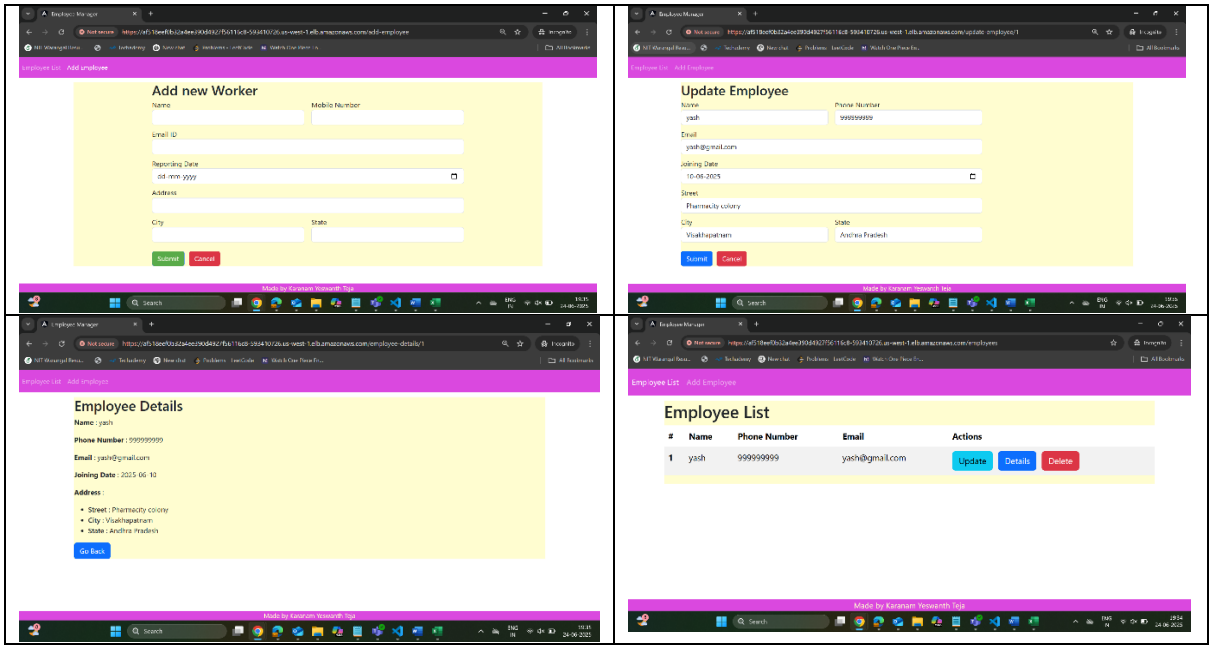
10. Amazon RDS Backup

- **Automated backups enabled** with a **30-day retention period**
- Supports **Point-in-Time Recovery (PITR)**
- **Manual snapshots** taken before major updates
- **Backups encrypted** and stored in Amazon S3
- **CloudWatch monitors** backup status and failures



11. Outcomes & Highlights

- Successfully deployed full-stack app in two AWS regions using different IaC tools
- Frontend interacts with backend over secure EKS communication
- CI/CD pipelines automate code delivery, infrastructure provisioning, and application deployment
- Application is scalable, fault-tolerant, and region-resilient
- Code is continuously monitored for quality and security
- Real-time monitoring and alerting via CloudWatch, Grafana, EventBridge



SOURCES

Cloud Formation

https://github.com/Yeswanthteja1010/captsone-employee-backend.git

https://github.com/Yeswanthteja1010/capstone-employee-frontend.git

Terraform

https://github.com/Yeswanthteja1010/capstone_backend_terraform.git

THANK YOU