**CAPSTONE PROJECT:**

AWS Multi-Region 3-Tier Project Documentation

**Project Title**: Multi-Region 3-Tier AWS Infrastructure with CI/CD (EXPESNE TRACKER)

**Overview**

This project demonstrates the end-to-end setup of a production-grade multi-region architecture in AWS. The infrastructure includes Amazon EKS clusters, RDS databases, CI/CD pipelines, monitoring, alerting, and domain routing via Route 53. It uses CloudFormation in Region 1 and Terraform in Region 2 to provision resources. Application deployment is fully automated using AWS CodePipeline, and code quality/security is enforced using SonarQube and Trivy.

**1. Architecture Diagram**

**Description:**

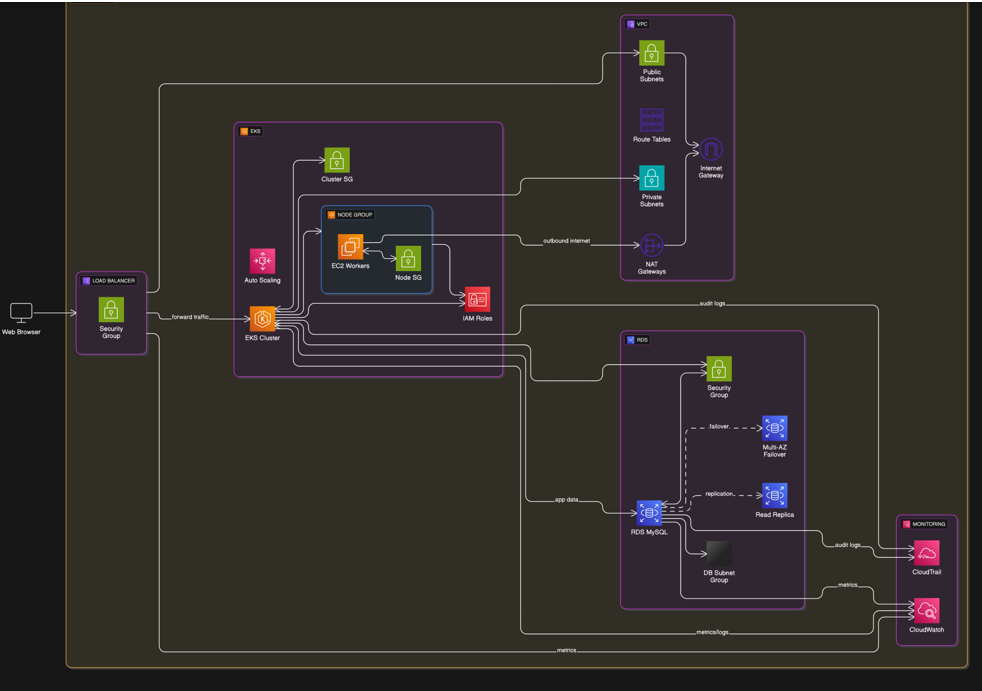
The architecture is designed for high availability and includes the following layers:

• Web Tier: Contains Application Load Balancers (ALBs) in public subnets for receiving external traffic.

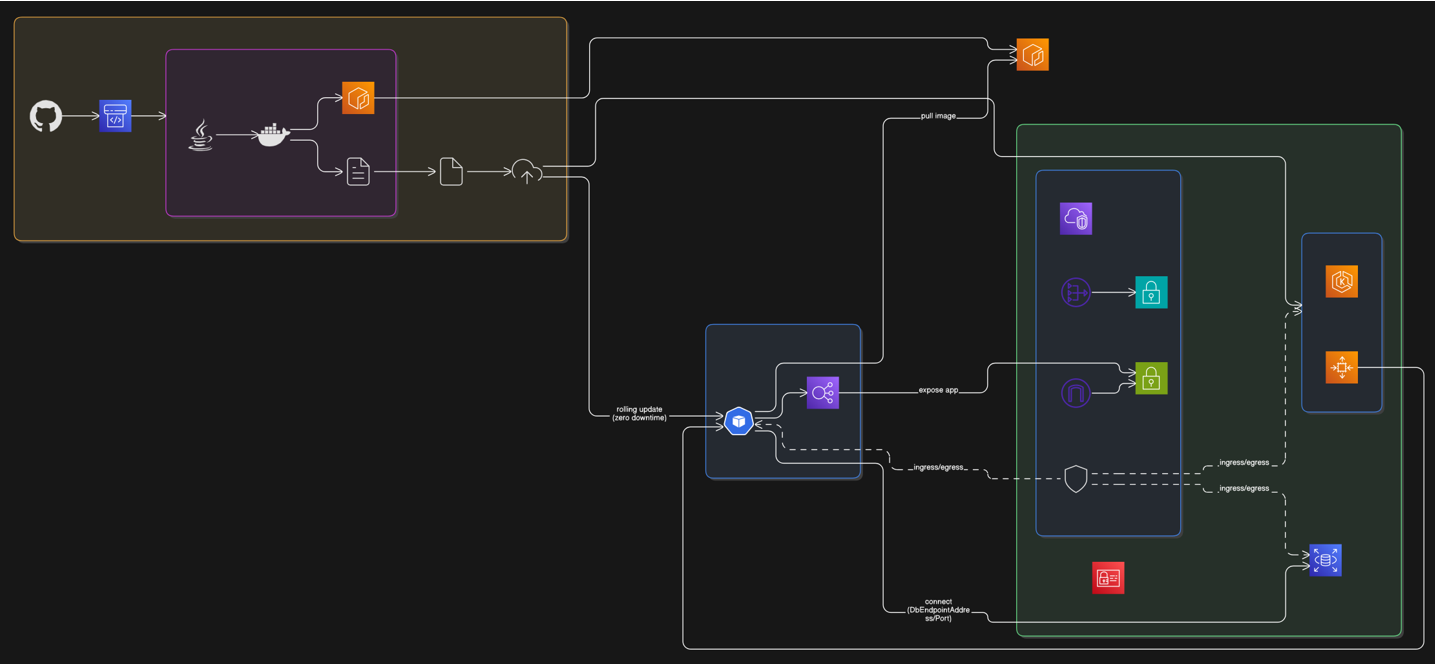
• App Tier: EKS worker nodes are deployed in private subnets to securely run containerized applications.

• DB Tier: Amazon RDS (MySQL) instances are provisioned in isolated private subnets to prevent direct internet access.

• Two separate environments are created in different AWS regions to ensure regional redundancy.



**2. CodePipeline Architecture**



**Description:**

AWS CodePipeline automates the entire lifecycle from code commit to deployment. The stages include:

• **Source Stage**: Triggers the pipeline when new code is pushed to GitHub.

• **Build Stage**: Uses CodeBuild to compile the code, run tests, analyze with SonarQube, and scan with Trivy.

• **Provision Stage**: Executes CloudFormation (Region 1) and Terraform (Region 2) to provision infrastructure.

• **Deploy Stage**: Pushes Docker image to ECR and deploys to EKS using kubectl.

• **Notification Stage**: Integrates with EventBridge to send alerts on failures.

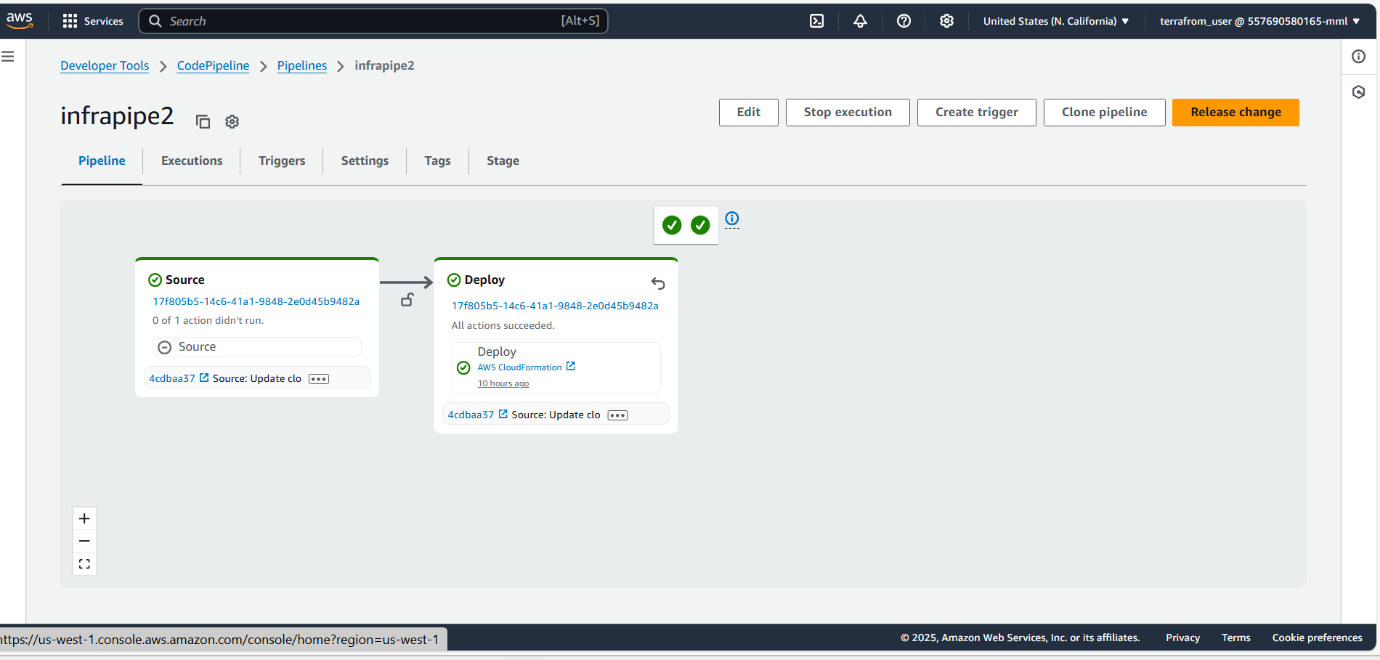
**3. Infrastructure Provisioning**

**Region 1 - Using CloudFormation**

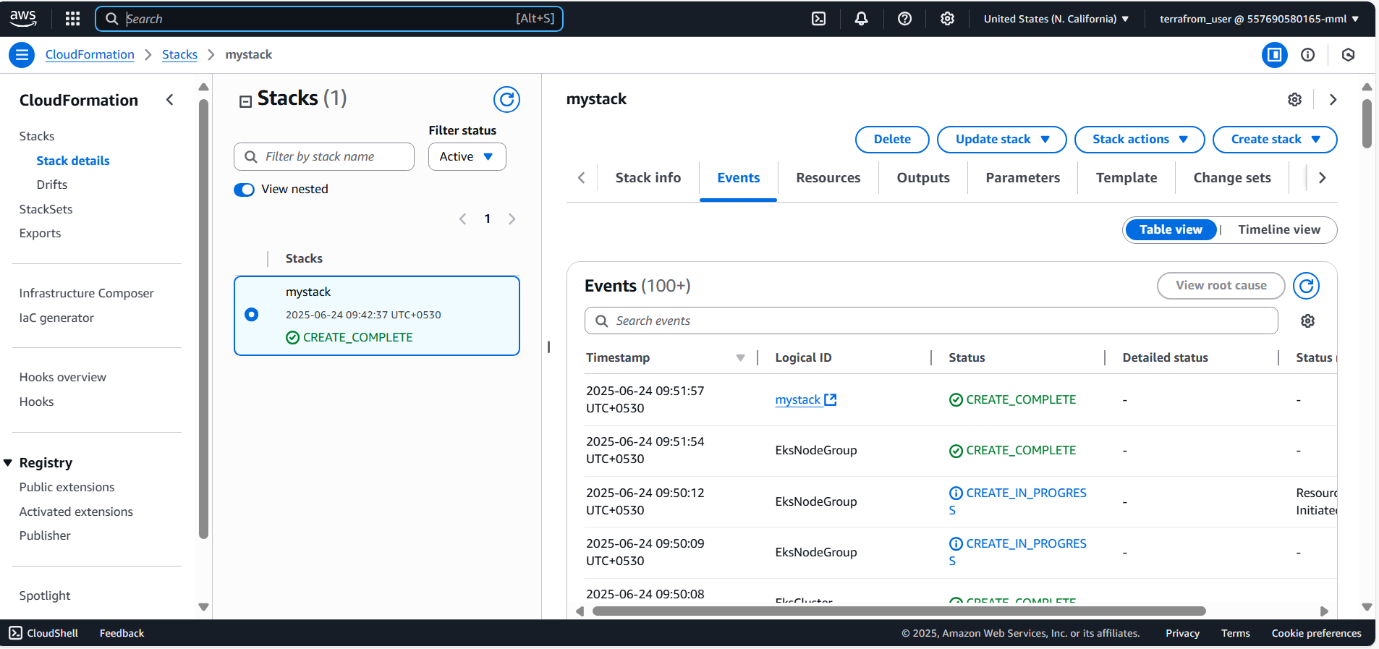
• CloudFormation template provisions VPC, subnets, Internet Gateway, NAT Gateway, route tables, security groups, Amazon EKS cluster, node group, and RDS MySQL database.

• EKS is created in private subnets for better security.

• RDS is configured with a DB subnet group to span across multiple AZs.

****

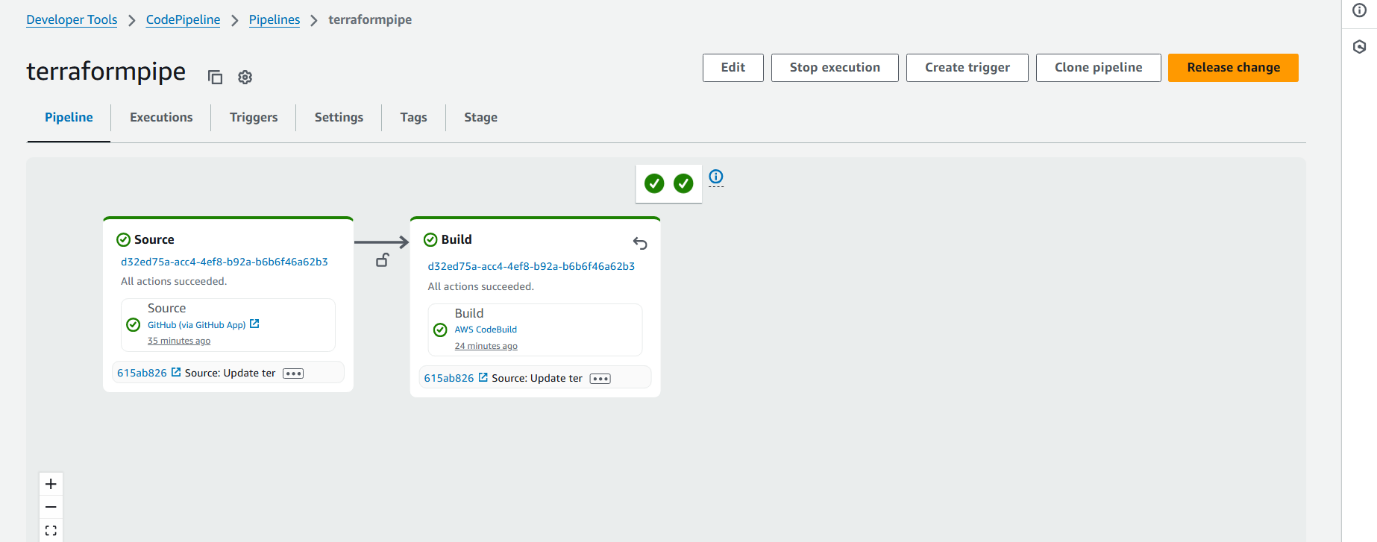
**Stack created automatically:**



**Region 2 - Using Terraform**

• Terraform scripts create the same setup as CloudFormation: VPC, subnets, route tables, EKS, and RDS.

• Infrastructure is reproducible and modular, supporting scalability and automation.



**4. Application Deployment**

Description:

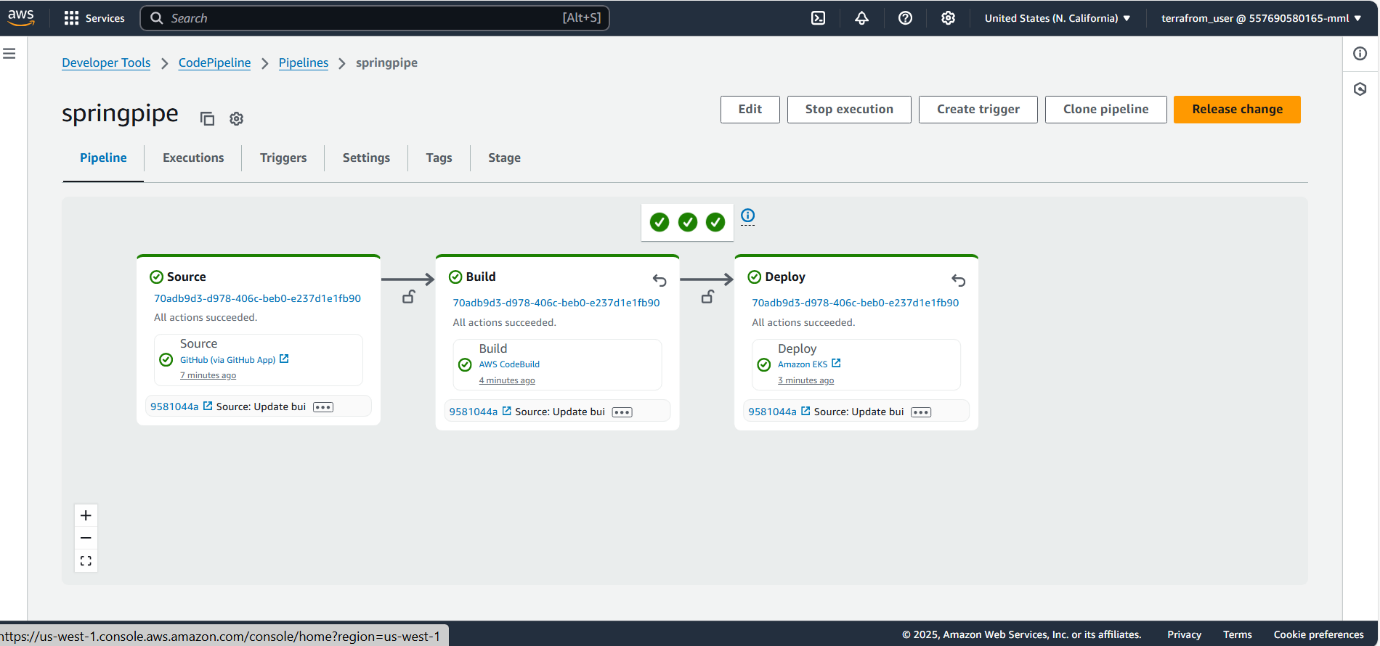
• Application is containerized using Docker.

• Deployment manifests (deployment.yaml and service.yaml) are applied to EKS using CodeBuild steps.

• The application is exposed via a Kubernetes service of type LoadBalancer, which maps to an ALB.

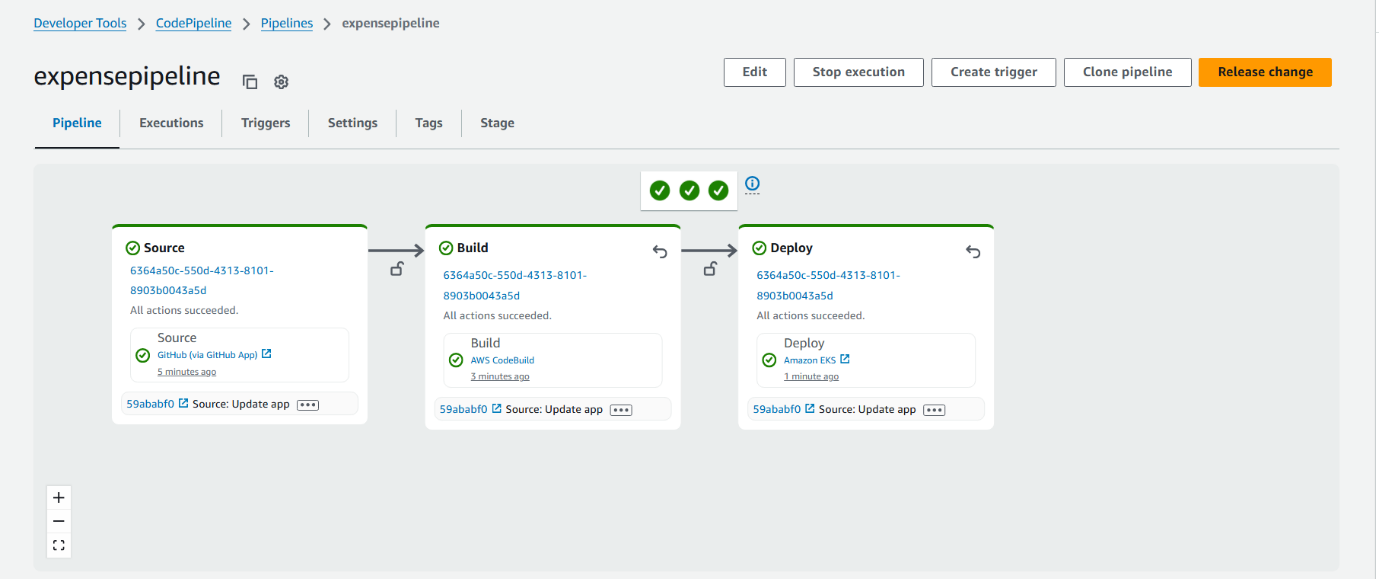
Region 1:

• Pipeline deploys the app to the EKS cluster created using CloudFormation.



Region 2:

• Pipeline deploys the app to the EKS cluster provisioned via Terraform.



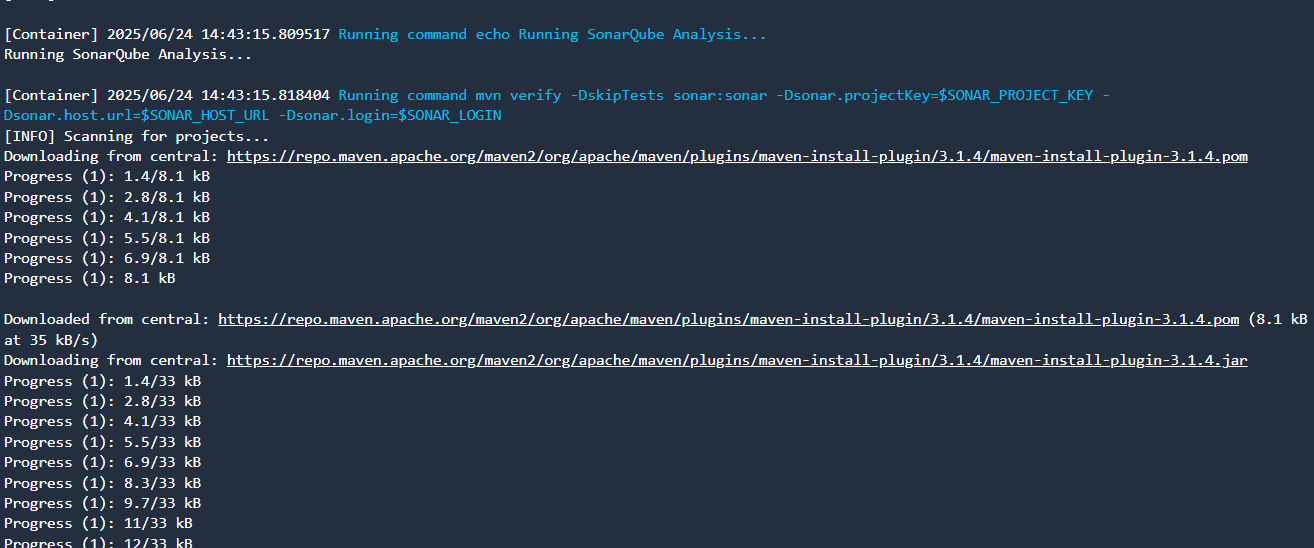
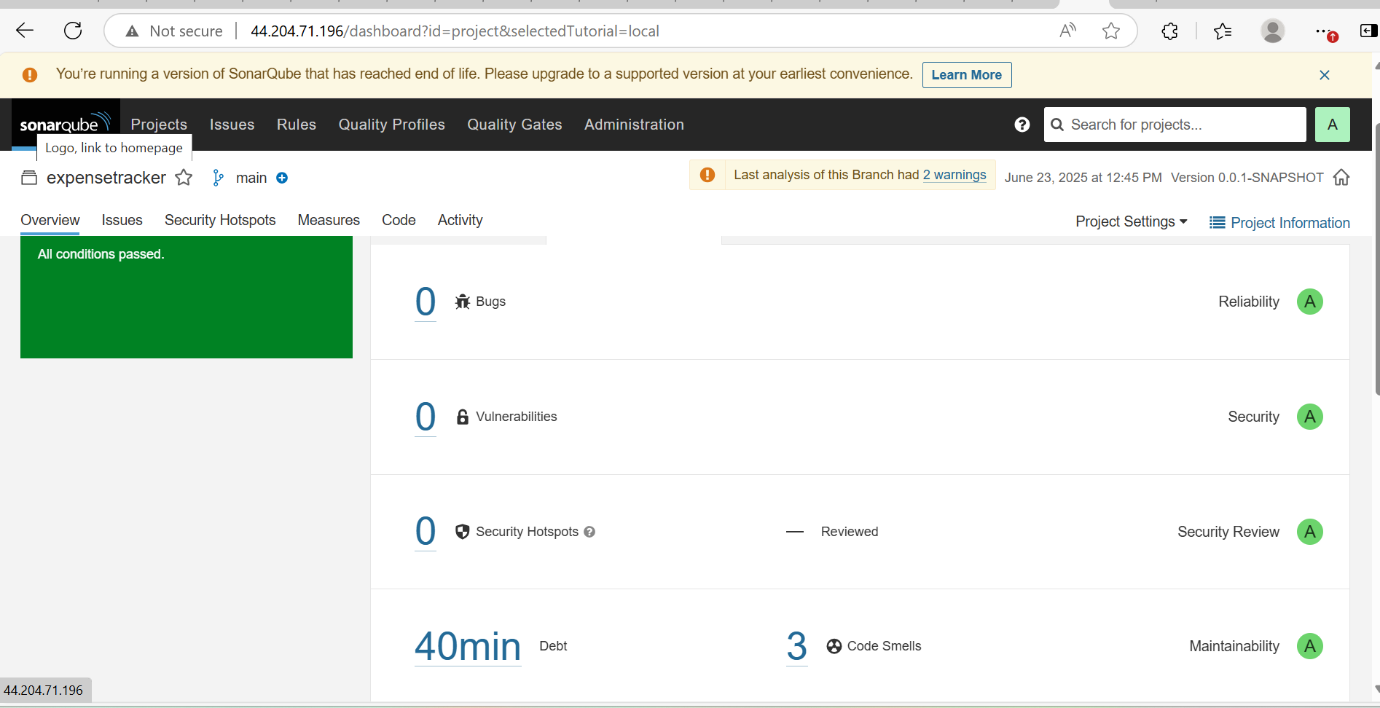
**5. Code Quality & Security Scanning**

SonarQube:

• mvn clean verify sonar:sonar is run during the build.

• SonarQube scans for code smells, bugs, and technical debt.

• The report is accessible through the SonarQube dashboard.

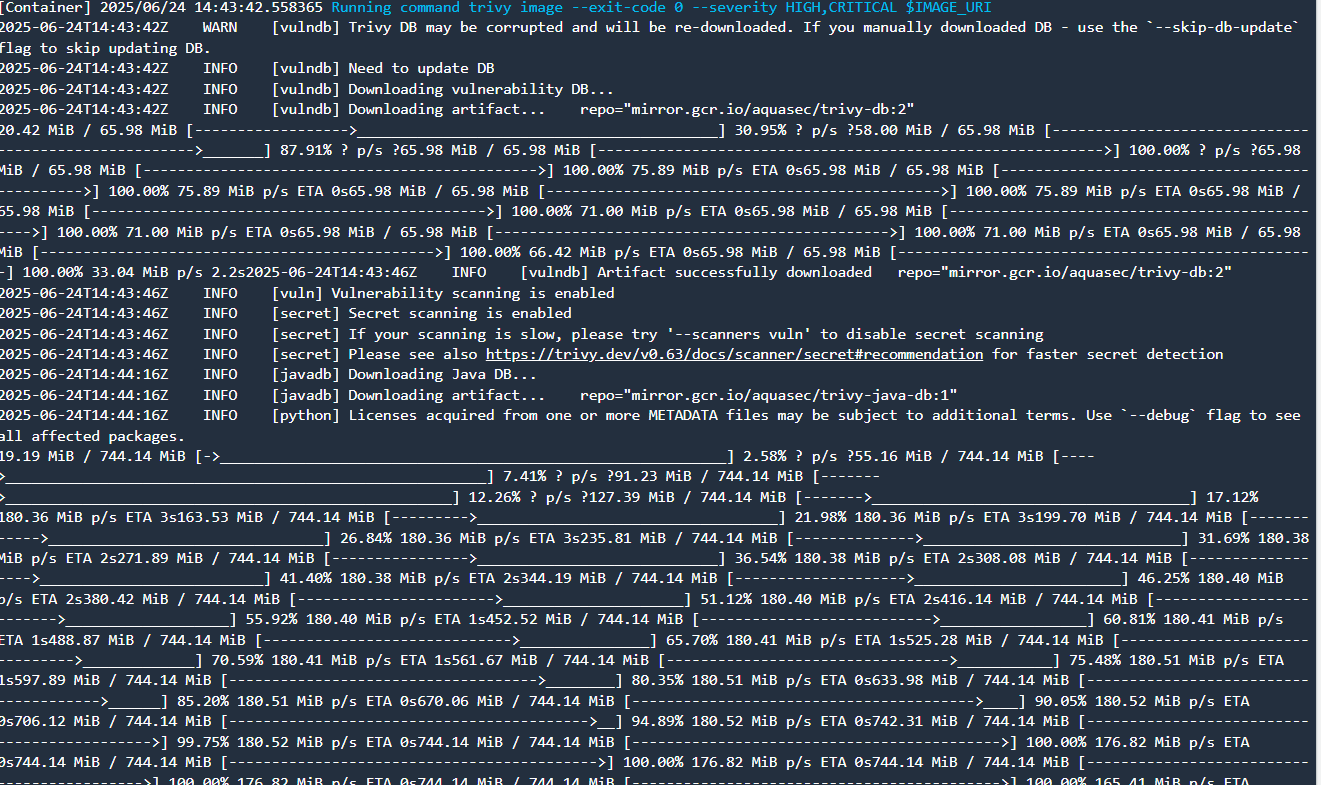
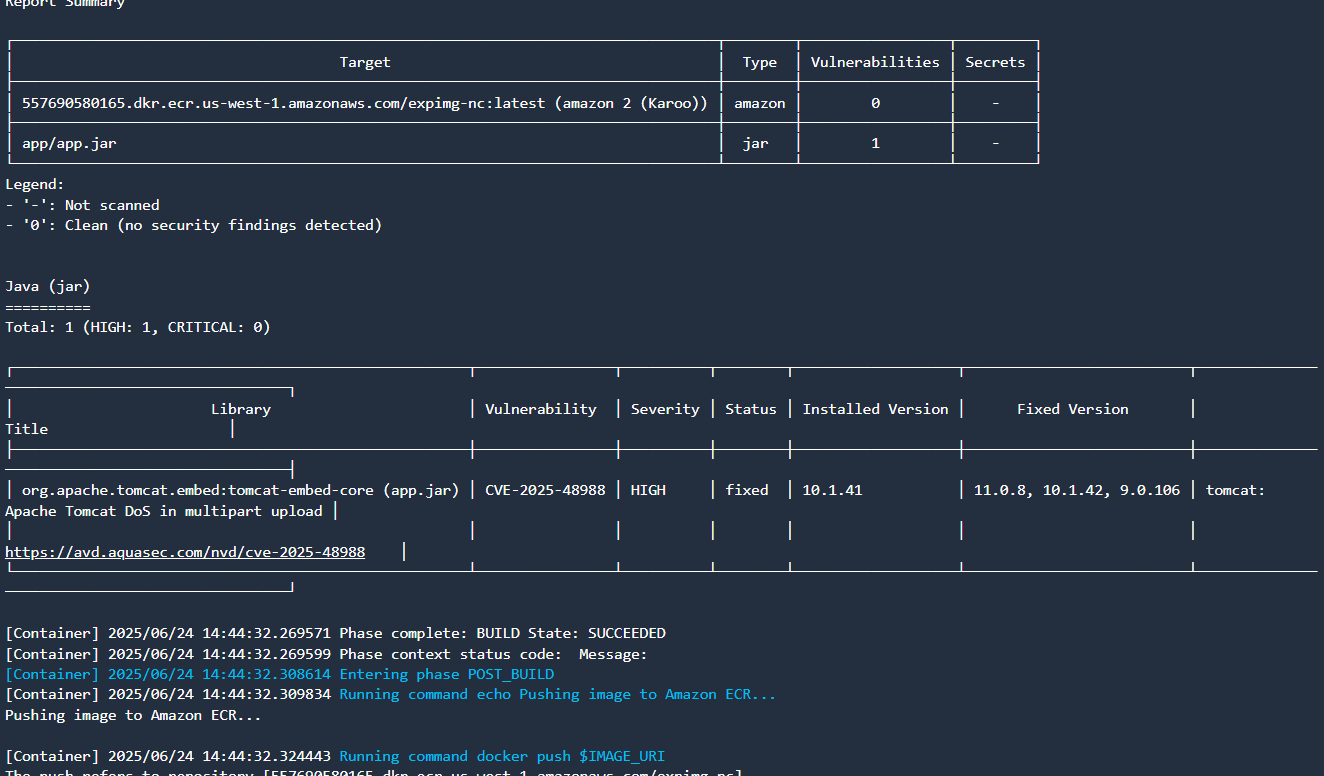
 

**Trivy:**

• Trivy scans the built Docker image for vulnerabilities.

• Scan results are printed in the build logs.

• Ensures container security before deployment.

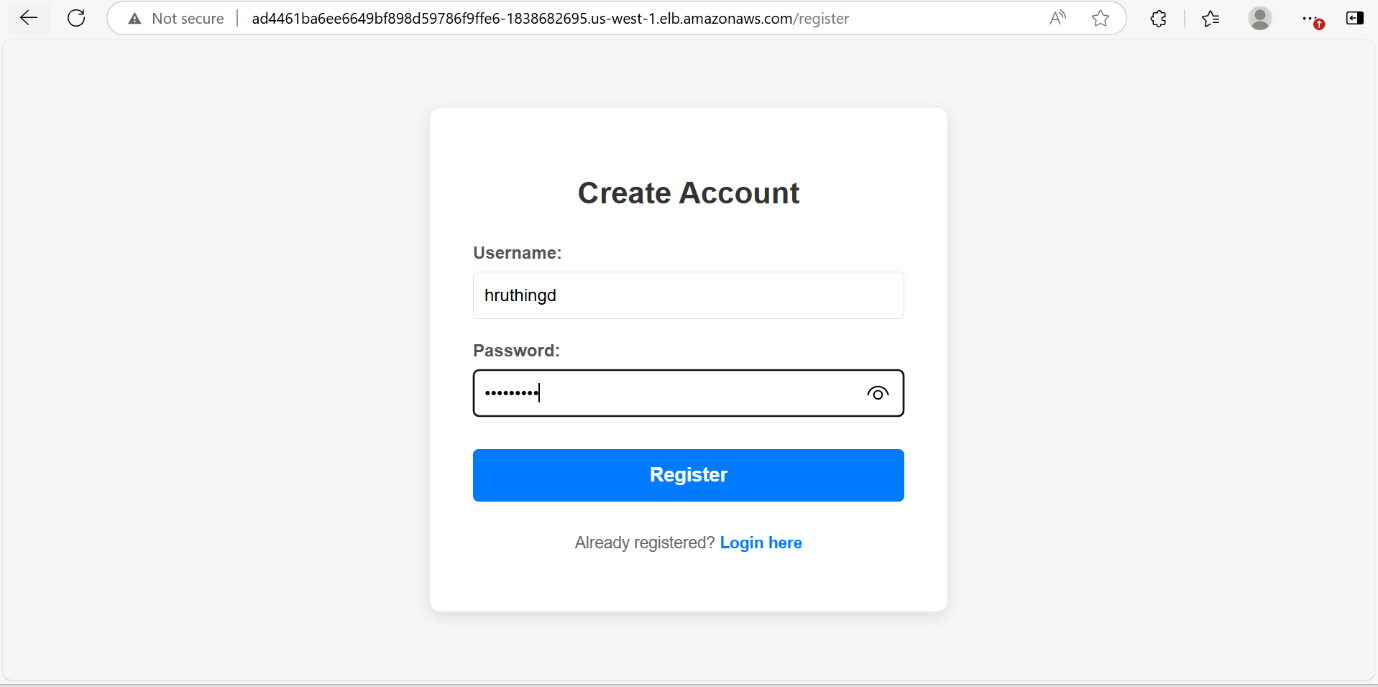
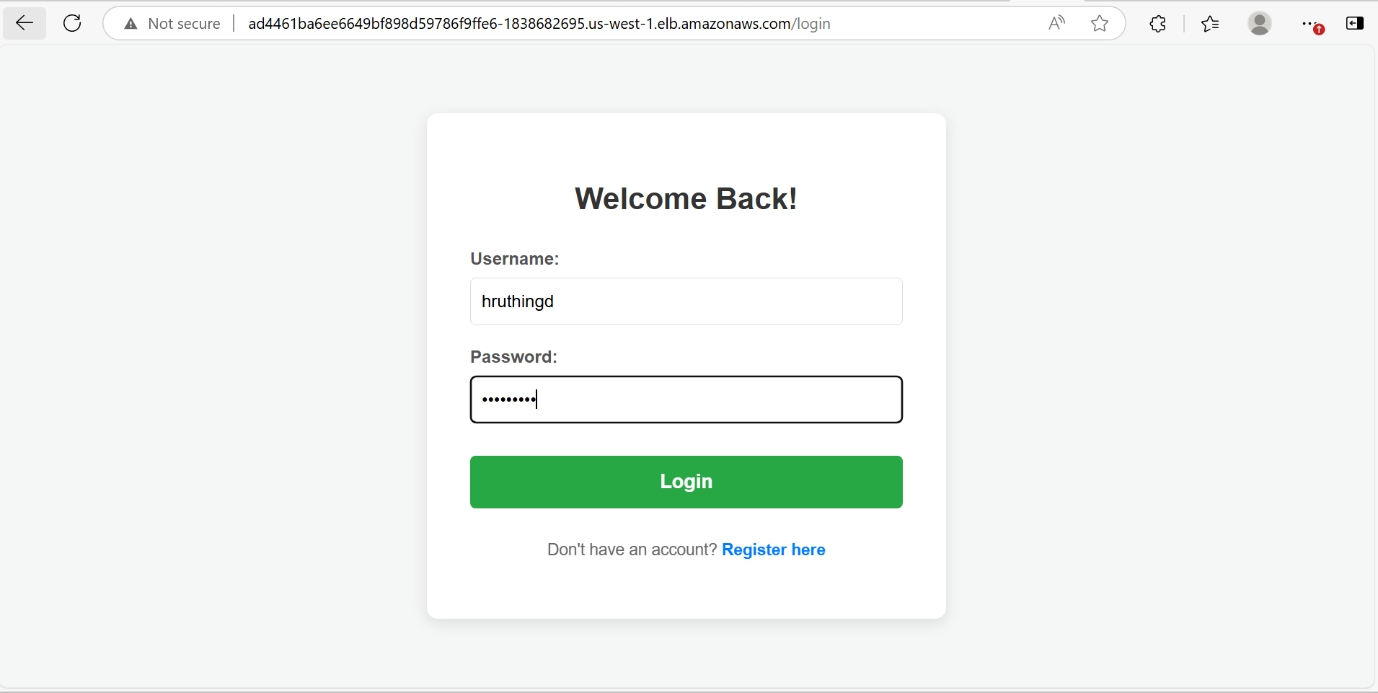
**6. Load Balancer and Application Access**

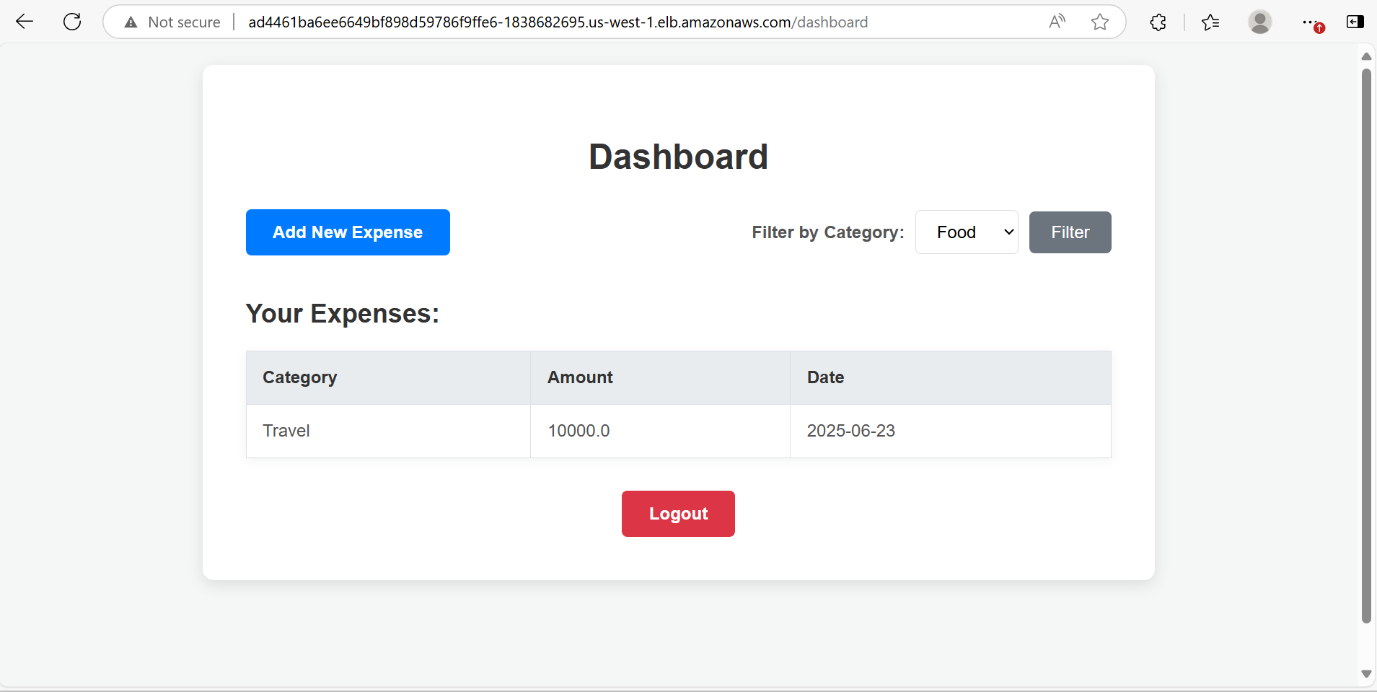
• Each region creates its own Application Load Balancer.

• Application can be accessed directly via ALB DNS or through a domain name configured in Route 53.

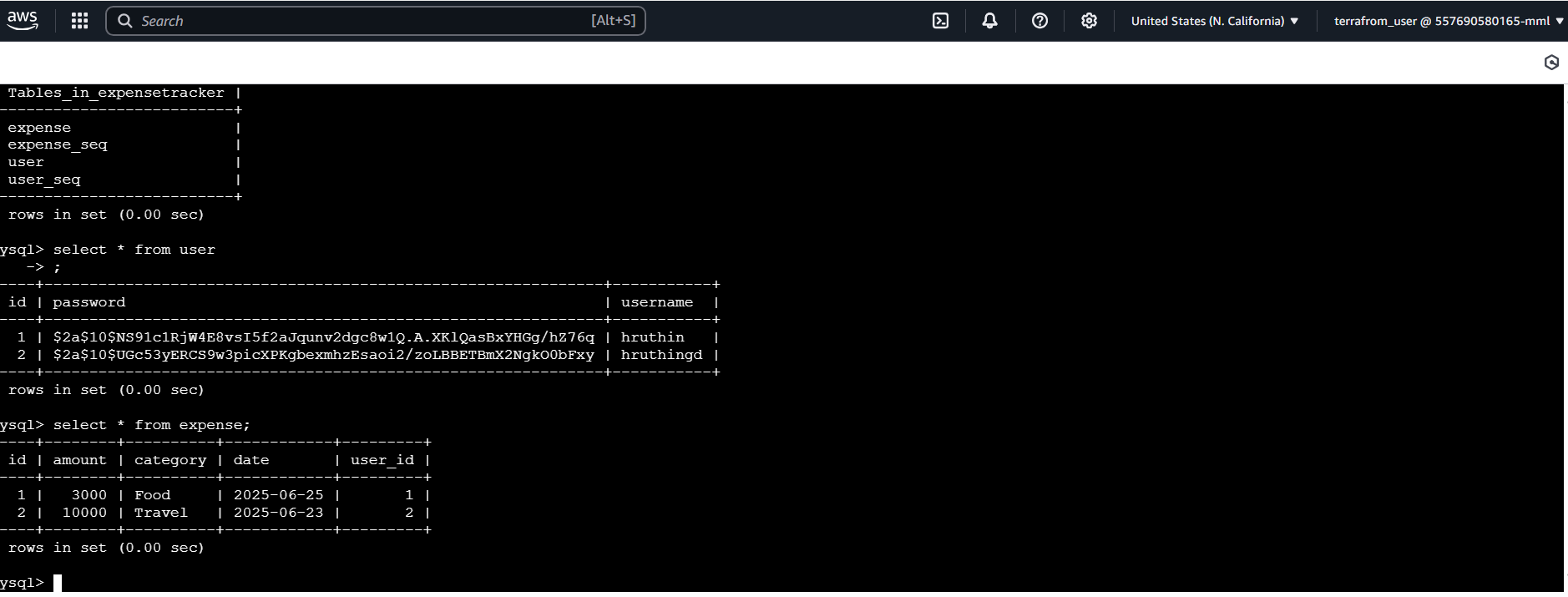
• Screenshots showing app running via both ALB endpoints.

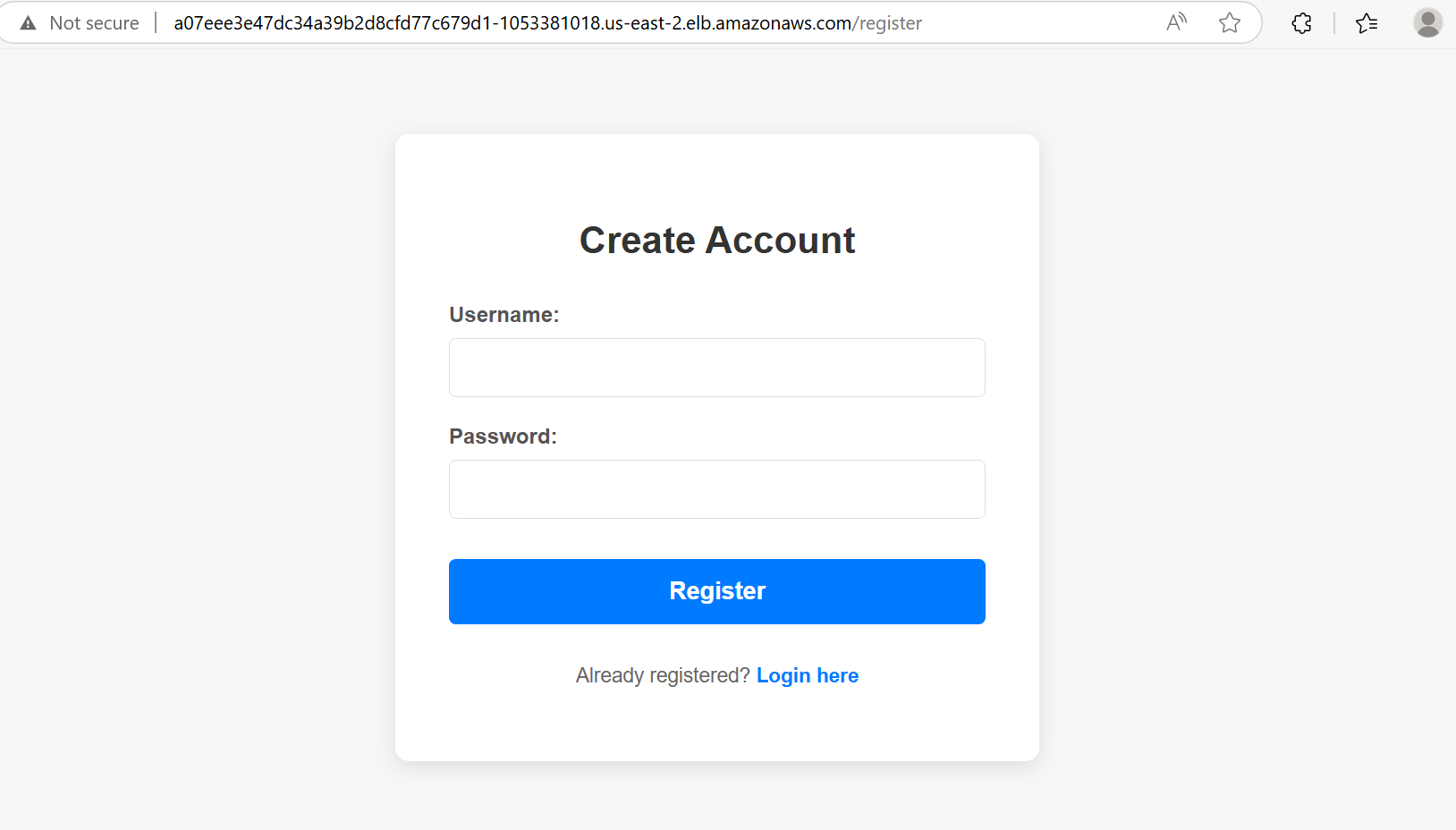
**(Region 1 load balancer)**



**Data added to database:**



**(Region 2 load balancer)** 

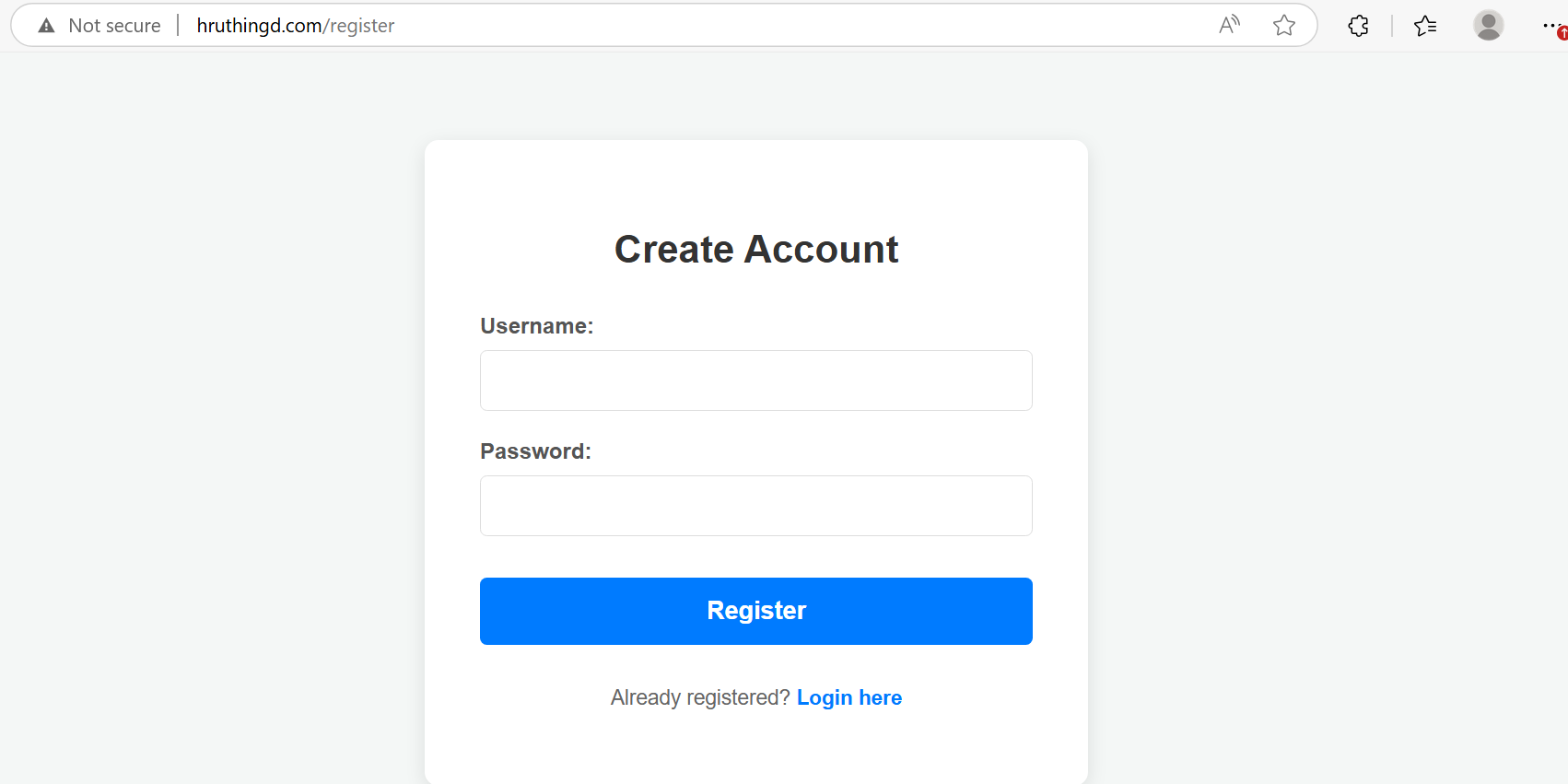
**7. Route 53 DNS Configuration**

Description:

• A hosted zone is created.

• Two A-records are added for the domain pointing to the ALBs in both regions.

• Routing policy can be set to Latency-based or Failover based on availability.



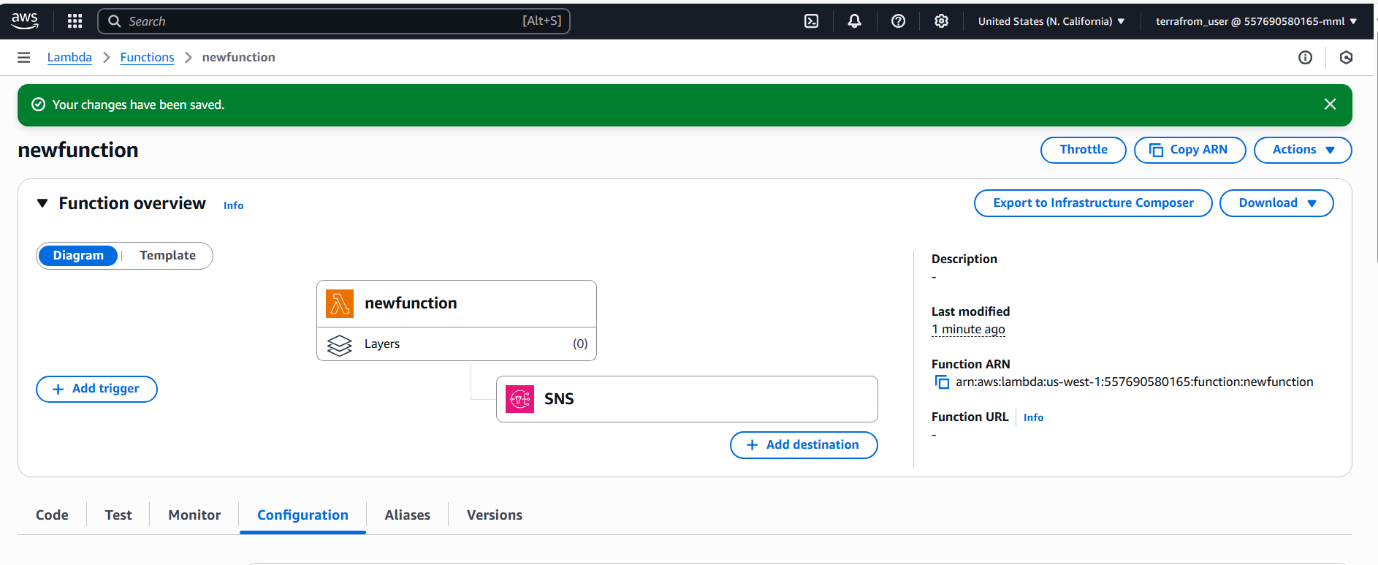
**8. EventBridge Notification**

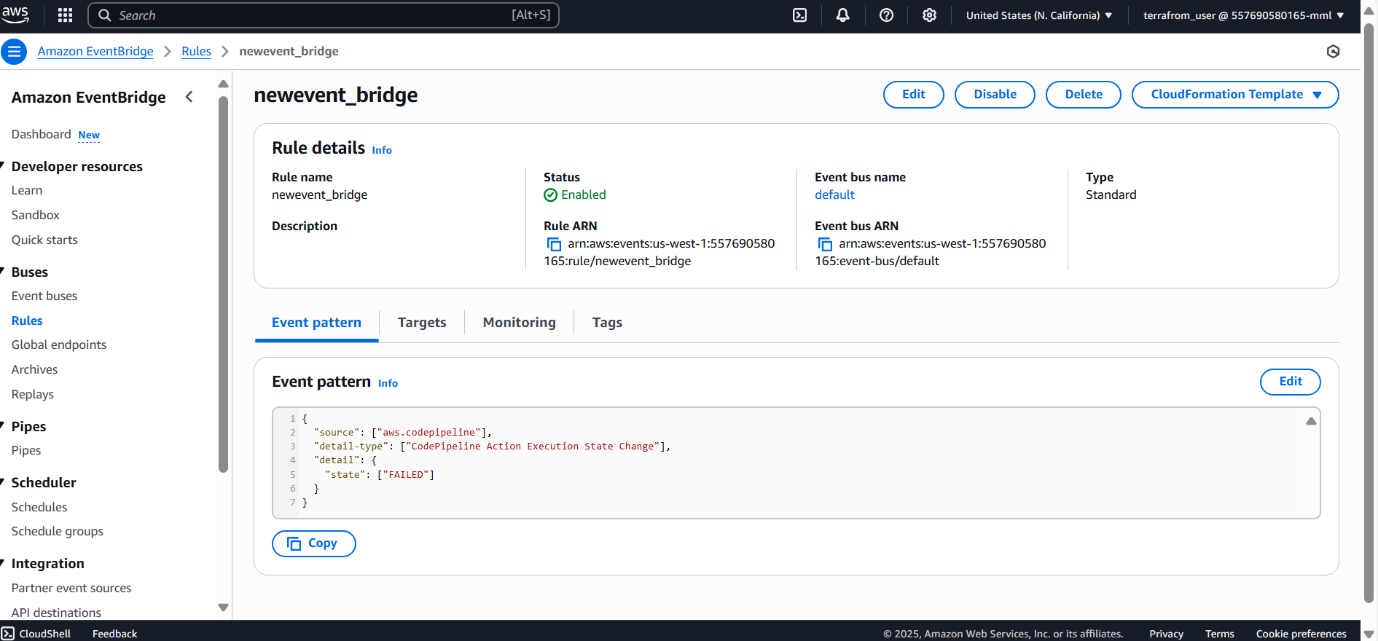
Description:

• EventBridge rule listens for CodePipeline Execution Failed events.

• On failure, it triggers a Lambda function or SNS topic that sends an email alert.

• Ensures timely awareness of pipeline issues.





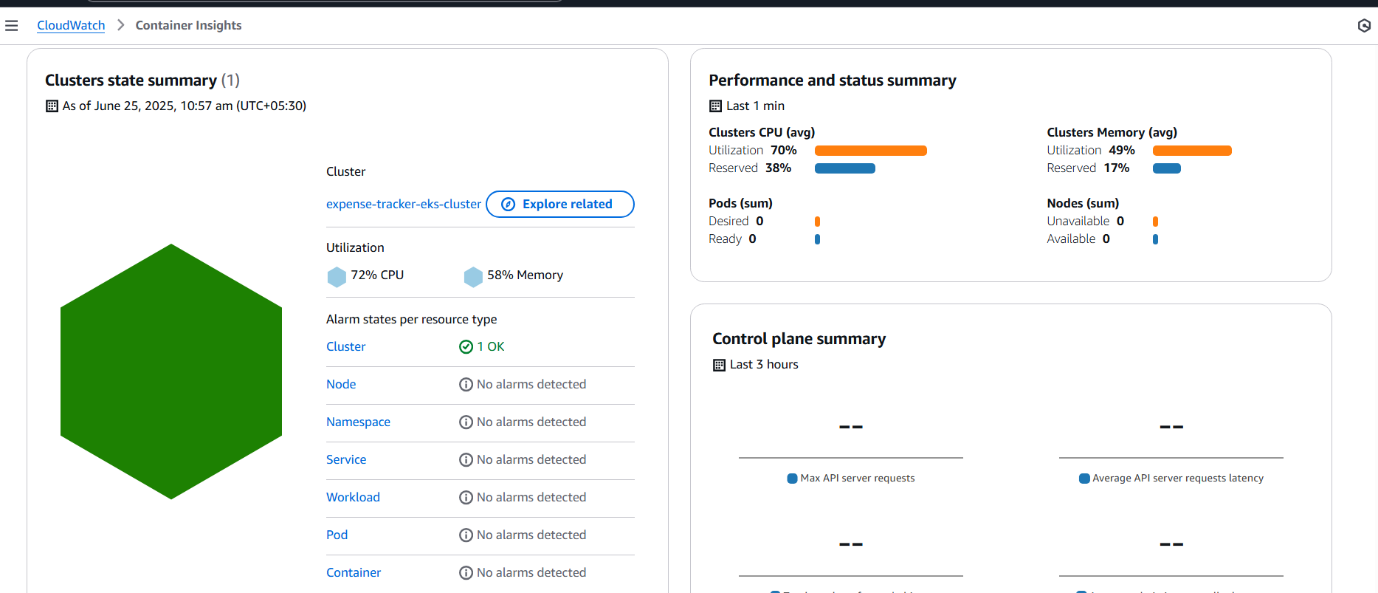
**9. CloudWatch Monitoring**

Description:

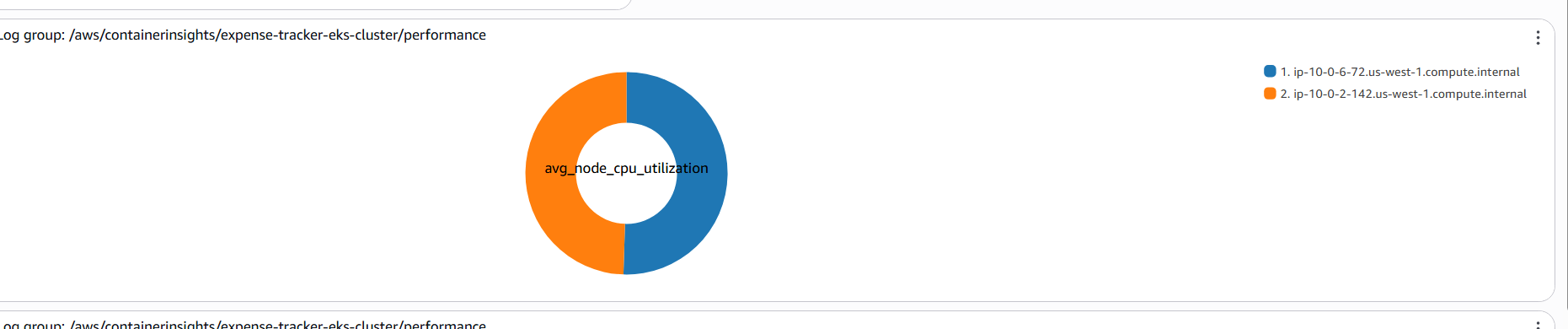
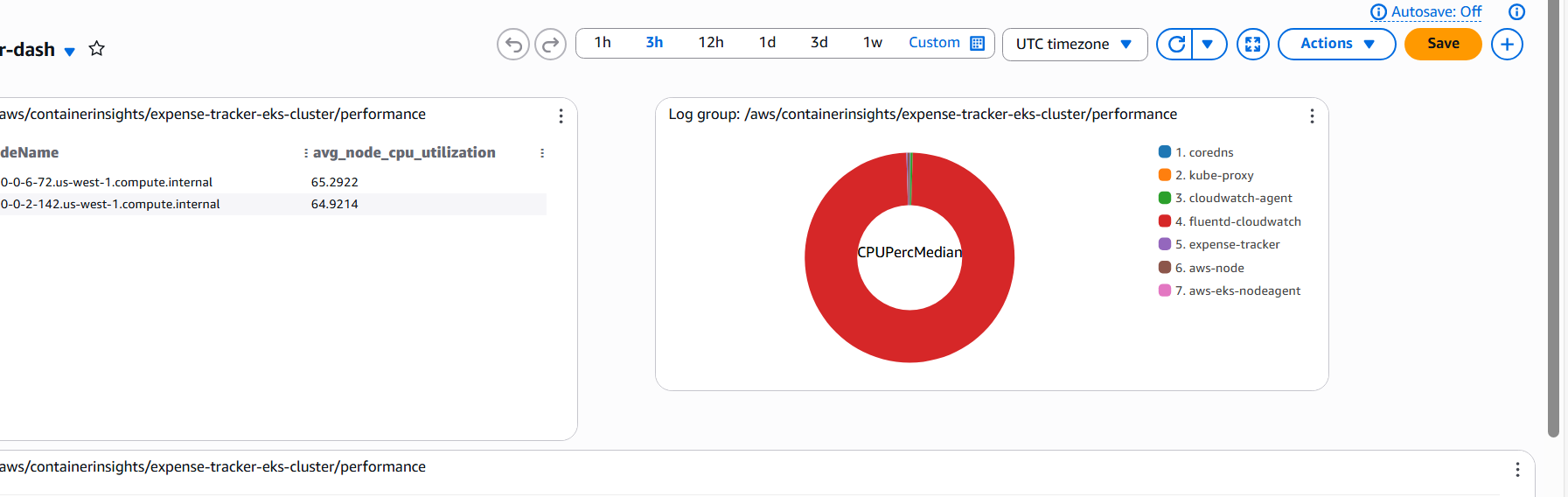
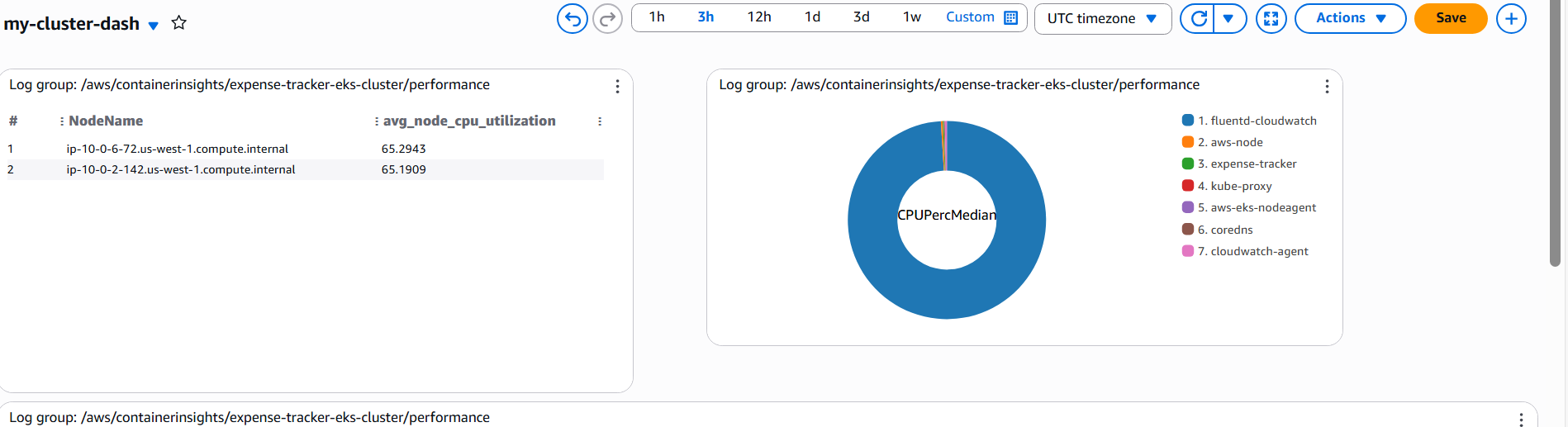
• CloudWatch is integrated with EKS, CodeBuild, and RDS for real-time logging and metrics.

• Logs can be viewed by service or resource.

• Helps in debugging and performance monitoring.

Container insights

Node cpu\_utilization

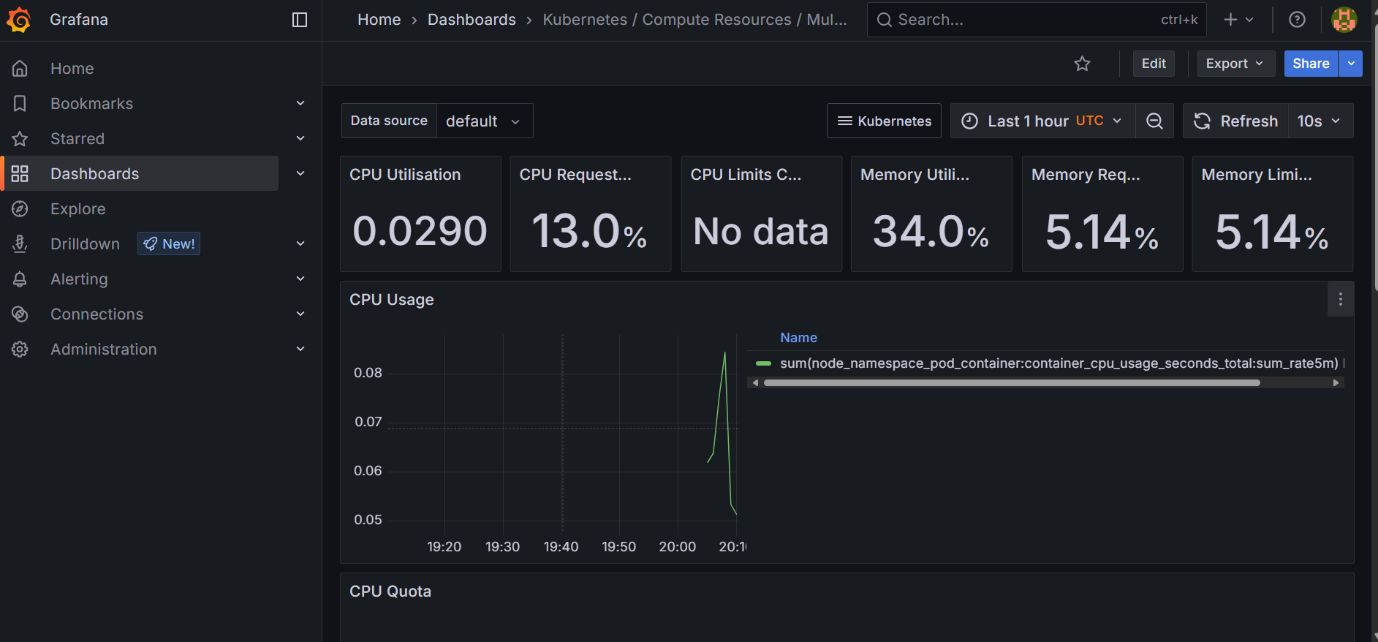
**10. Grafana Dashboard**

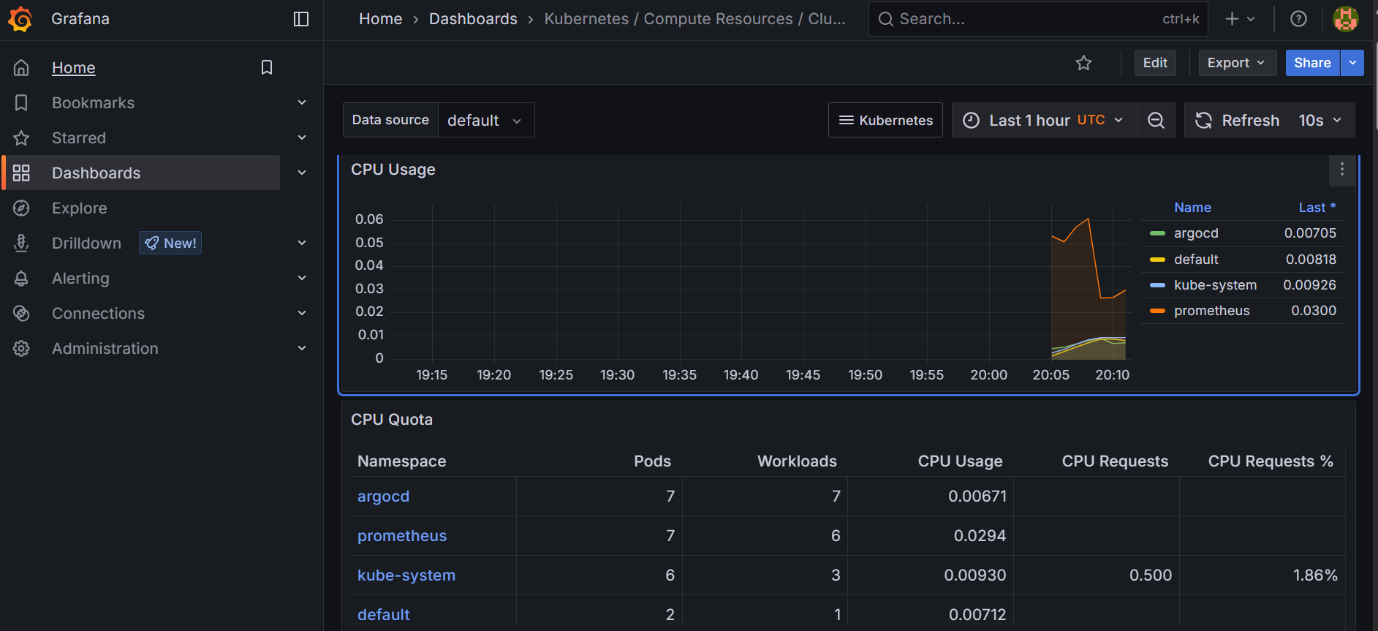
Description:

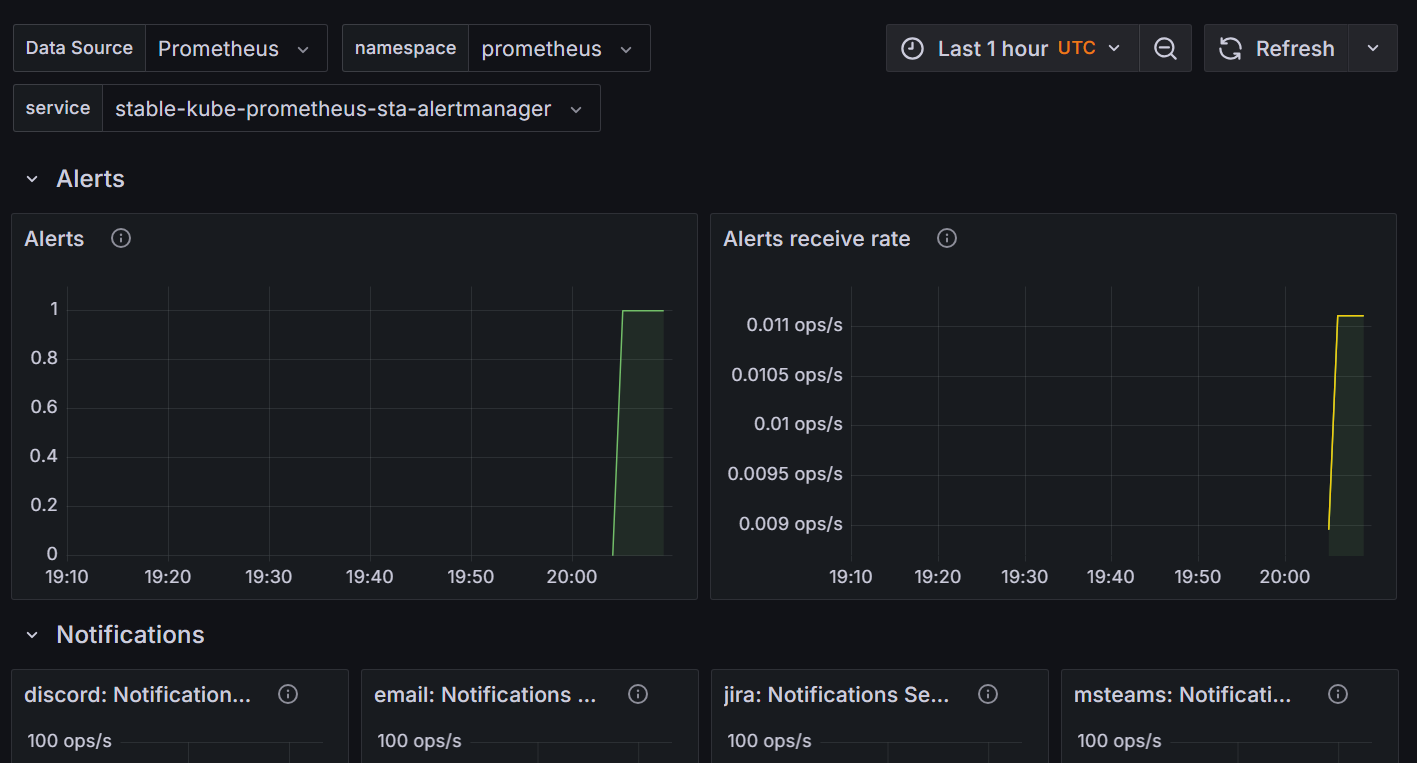
• Grafana is used to visualize metrics from CloudWatch.

• Dashboards show CPU/memory usage, pod health, and cluster performance.

• Custom panels can be created for specific service monitoring.







**11. Challenges Faced and Resolutions**

**Security Group Rules**

**Issue**: Application was not accessible, and pods couldn't connect to RDS.

**Cause**: Security groups were either too restrictive or missing appropriate ingress/egress rules.

**Solution**:

* Modified the security group to allow traffic from ALB to EKS nodes on the required ports (e.g., 80, 443).
* Allowed EKS nodes to access RDS port (3306) within VPC.
* Ensured least privilege access while maintaining functionality.

**IAM Roles and Permissions**

**Issue**: CodeBuild or EKS nodes couldn't perform certain actions (e.g., push to ECR, pull from ECR, deploy to cluster).

**Cause**: Missing IAM permissions for specific services.

**Solution**:

* Created IAM roles with required policies (e.g., AmazonEKSWorkerNodePolicy, AmazonEC2ContainerRegistryFullAccess, AmazonEKS\_CNI\_Policy).
* Attached roles correctly to CodeBuild, EKS node groups, and ServiceAccounts.
* Validated using IAM policy simulator.

**Build Stage Failures (SonarQube & Trivy)**

**Issue**: The build stage failed due to SonarQube being unable to connect to its database.

**Cause**:

* **SonarQube Issue**: The SonarQube server was configured to use a MySQL database deployed in a private subnet without internet access. Since CodeBuild runs in a public environment, it couldn't access the private subnet where the DB resides.

**Solution**:

* For SonarQube:

Used the Maven -DskipTests flag because SonarQube attempted to test the database connection during the analysis phase. Since the MySQL database was hosted in a private subnet and unreachable from the CodeBuild environment, the scan was failing. The -DskipTests flag was used as a workaround to bypass these checks and allow the scan to proceed.

This was a workaround to allow builds to pass until database access could be properly configured.

Ensured that this was used temporarily during development and not in production pipelines

**12. Final Notes and Recommendations**

**Best Practices Followed:**

* Used separate IaC tools in different regions to demonstrate tool versatility.
* Kept EKS and RDS in private subnets to improve security.
* Integrated DevSecOps tools like SonarQube and Trivy into the pipeline.
* Ensured monitoring and alerting using CloudWatch, EventBridge, and Grafana.
* Used Route 53 for domain resolution and high availability through latency-based routing.

**Recommendations for Improvement:**

* In production, configure a VPC endpoint or VPC peering to securely access private subnets for tools like SonarQube.
* Use Secrets Manager or SSM Parameter Store for secure credential management. I didn’t use because it takes more credits in the AWS account.

**13. Conclusion**

This project demonstrates a full-scale DevOps implementation on AWS using modern tools and best practices. It ensures scalability, security, and observability while supporting a multi-region deployment model.

**• GitHub Repository URL**

<https://github.com/hruthingali/expense-tracker.git>

**• Common CLI commands:**

* eksctl, kubectl, aws, terraform, mvn

•**Troubleshooting Notes:**

Build failures, network errors, IAM permissions