EKS Harbor – Project Documentation

Architecture Overview

This project provisions an AWS EKS cluster using Terraform and deploys the Harbor container registry using Helm. The application is exposed publicly via an AWS Application Load Balancer (ALB) with HTTPS enabled. Monitoring is provided by Prometheus and Grafana.

Logical Architecture

```
Internet
   |
   | HTTPS (443)
   v
AWS Application Load Balancer (ALB)
   |
   | AWS Load Balancer Controller (Ingress)
   v
Amazon EKS Cluster
 ■■ Spot Node Group (2 × t3a.small)
 ■    ■■ Harbor Pods
 ■
 ■■ Prometheus
 ■■ Grafana
```

Terraform Structure

The infrastructure is organized using reusable Terraform modules. Environment-specific configuration is stored under environments/dev. Each module has a single responsibility, following Terraform best practices.

Installation Steps

1. Clone the repository. 2. Navigate to environments/dev. 3. Run 'terraform init'. 4. Run 'terraform apply'. Terraform will provision all AWS and Kubernetes resources automatically.

Application Deployment

Harbor is deployed using the Terraform helm_release resource. No manual Helm or kubectl commands are required. Once deployed, Harbor is accessible via a public HTTPS domain.

Load Testing

Load testing can be performed using k6. Traffic flows from the load test tool through the ALB and Ingress to the Harbor pods, validating ingress and application performance.

Dashboards and Alerts

Prometheus collects metrics from the EKS cluster and applications. Grafana provides dashboards for CPU, memory, pod, and node metrics. Alertmanager triggers alerts for unhealthy pods or nodes.

Cleanup Instructions

To remove all infrastructure resources, run 'terraform destroy' from the environments/dev directory. This will delete the EKS cluster, load balancer, node groups, application, and monitoring stack.