High Availability Kubernetes Platform Design Document

1. Overview

This document describes the design of a high availability (HA) platform using Kubernetes (k8s) with Hot/Standby regional redundancy. It enables developers to deploy TCP-based REST API services with automated failover, security, and observability.

2. Objectives

- Design an HA architecture using two k8s clusters in separate AWS regions.
- Enable seamless application deployment through CI/CD pipelines.
- Automate infrastructure setup using Ansible.
- Ensure failover from hot to standby region.
- Implement secure networking and observability.

3. Assumptions

- Two Kubernetes clusters are pre-deployed: Region 1 and Region 2 (Standby)
- Each cluster has Nginx Ingress Controller deployed.
- Services are containerized and exposed via TCP/REST APIs.

4. Architecture Components

Component	Description
Route 53	DNS-based traffic routing to primary/secondary load balancer.
Load Balancers (ELB)	Routes traffic to Nginx Ingress controller.
Ingress Controller	Nginx controller routes traffic inside k8s cluster.
Kubernetes Clusters	Two regional clusters.
Bastion Hosts	Access point to private nodes, secured via public subnet.
VPC/Subnets	Isolated networking environment. Public and private subnets in each AZ.

5. High Availability & Failover

- Active-Standby Design: Region 1 serves traffic actively; Region 2 is standby.
- Failover Mechanism:
  a. Route 53 health checks monitor Region 1's load balancer.
  b. On failure, Route 53 shifts traffic to region 2 (secondary ELB).
  c. Cluster-local readiness/liveness probes ensure service health.

6. Networking and Security

- Ingress and Load Balancer Security Groups: Only allow required ports (e.g., 80, 443, 22).
- Private Subnets: Only nodes reside inside private subnets.
- IAM Roles: Fine-grained access control using the least privilege.
- TLS/HTTPS: Certificates via AWS ACM or Cert-Manager with Let's Encrypt.

7. Monitoring & Logging

- Utilized Prometheus and Grafana for Logging, Metrics and Alerting.
- Kube-state-metrics and Node Exporter.

8. Scaling and Production Readiness

- Horizontal Pod Autoscaler (HPA) based on CPU/memory metrics.
- Cluster Autoscaler for dynamic node scaling.
- Canary and Blue-Green deployments via CI/CD.
- Regular backups using Velero or other tools.