Failover Mechanism

Objective

In the event of a failure, use automated detection and failover routing techniques to guarantee that traffic is seamlessly redirected from the primary (hot) Kubernetes cluster to the secondary (standby) cluster.

1. Failover Strategy Overview

- Primary Cluster: Actively handles production traffic.
- Secondary Cluster: Remains in sync but idle; activated only during primary failure.
- Routing Control: Managed by Route 53 with health checks and failover routing policy.
- Automation: Implemented via AWS Lambda to update Route 53 DNS records dynamically during failure.

2. Route 53 Failover Routing

- Health Checks are created for all the load balancers (e.g., /healthz endpoint from Ingress).
- If the primary health check fails, Route 53 automatically redirects to the secondary. This is the first layer of passive failover.

3. Lambda-Based Active Failover Enhancement

AWS Lambda can be used to enhance failover responsiveness with active DNS control based on custom health metrics or logs.

How It Works

1. Monitoring:
   Use CloudWatch Alarms to track application metrics (e.g., high latency, errors).
2. Trigger Lambda:
   If the alarm is triggered, a Lambda function is invoked to:
   a. Check status of both clusters.
   b. Change the Route 53 DNS record to point to the secondary cluster.
3. Lambda Logic:
   a. Uses boto3 to call Route 53 APIs.
   b. Validates standby health before switching.
   c. Can optionally notify teams via SNS/Slack.

4. Health Check Endpoint Best Practices

- Each cluster's NGINX ingress should expose a readinessProbe and livenessProbe.
- Use endpoints like /healthz that check: Ingress availability, Pod state and Service responsiveness