Project Report: Kubernetes-Based Canary Deployment with K3s and Istio

Objective

The goal of this project is to simulate a modern canary deployment strategy using K3s (a lightweight Kubernetes distribution) and Istio (a service mesh platform). This involves deploying two versions of an application (v1 and v2), configuring traffic routing using Istio to split traffic (80% to v1 and 20% to v2), monitoring performance, and managing promotion or rollback based on real-time metrics.

Architecture Overview

Client

|

Ingress Gateway (Istio)

|

VirtualService (Traffic Split: 80% v1, 20% v2)

|

Service (my-app)

|

Deployments

v1 (Stable)

v2 (Canary)

Tools & Technologies Used

- K3s: Lightweight Kubernetes distribution
- Istio: Service mesh for advanced traffic management
- Docker: Containerization
- Node.js / Python: Sample application base
- kubectl: Kubernetes CLI
- Prometheus + Grafana (optional): For performance monitoring
- Helm (optional): For packaging YAML files

Project Report: Kubernetes-Based Canary Deployment with K3s and Istio

Files Created

deployment-v1.yaml - Deploys version 1 (stable) of the app

deployment-v2.yaml - Deploys version 2 (canary) of the app

service.yaml - Kubernetes service to expose the app

destination-rule.yaml - Defines subsets for v1 and v2 in Istio routing

gateway.yaml - Sets up Istio Gateway to expose app externally

virtual-service.yaml - Splits traffic between v1 (80%) and v2 (20%)

README.md - Complete setup guide and strategy document

Setup Instructions

1. Apply deployments for v1 and v2:

kubectl apply -f deployment-v1.yaml

kubectl apply -f deployment-v2.yaml

2. Apply the service:

kubectl apply -f service.yaml

3. Apply Istio configurations:

kubectl apply -f destination-rule.yaml

kubectl apply -f gateway.yaml

kubectl apply -f virtual-service.yaml

4. Get the external IP of the Istio ingress gateway:

kubectl get svc -n istio-system

Monitoring & Testing

Option 1: Manual Curl Logs

- Run curl requests to simulate traffic:

Project Report: Kubernetes-Based Canary Deployment with K3s and Istio

for i in {1..20}; do curl http://<EXTERNAL-IP>/; done

- Check logs for both versions:

kubectl logs -l version=v1

kubectl logs -l version=v2

Option 2: Prometheus + Grafana

- Monitor metrics:
  - istio_requests_total
  - istio_request_duration_seconds
  - Error rates and latency

Canary Deployment Strategy

- Deploy both v1 (stable) and v2 (canary)
- Use Istio VirtualService to route 80% to v1 and 20% to v2
- Monitor for issues
- Promote or rollback based on metrics

Promotion Plan

- If v2 performs well, promote gradually: Change weight to 50/50, then 100% to v2

Rollback Plan

- If v2 has issues, route 100% traffic back to v1

Deliverables

- YAML files for deployment and Istio config
- README.md with full setup guide
- Monitoring steps and strategy document
- This project report

Project Report: Kubernetes-Based Canary Deployment with K3s and Istio

Conclusion

This project showcases how to implement a safe, controlled canary deployment strategy using open-source tools like K3s and Istio. It emphasizes traffic management, observability, and rollback mechanisms essential practices in modern DevOps pipelines.