kubernetes

Kubernetes: Canary Deployment

Canary Deployment

- **Canary Deployment** is to shift a controlled percentage of user traffic to a newer version of the service in the process of phasing out the older version. This technique is called a ***canary deployment***.
- Kubernetes cluster operators can **orchestrate canary deployments** natively using **labels** and Deployments.
- In this approach, **traffic distribution** and **replica counts** are coupled, which in practice means replica ratios must be controlled manually in order to limit traffic to the canary release.
- Deploying with an **Istio service mesh** can address this issue by enabling a clear separation between **replica counts** and **traffic management**.

KUBERNETES : MicroServices

- Istio mesh allows fine-grained traffic control that **decouples traffic distribution** and **management** from replica scaling. Instead of manually controlling **replica ratios**, you can define traffic percentages and targets, and Istio will manage the rest.
- Here, You will deploy two versions of a demo **Node.js** application, and use **Virtual Service** and **Destination Rule** resources to configure traffic routing to both the newer and older versions.

KUBERNETES : MicroServices

- As part of New Deployment we will modify the node-application manifest first.
- We will update the version in Deployment manifest.
- Update the Image In Manifest.
- **Will not touch the labels in manifest.**
- **Also will not modify the Service Manifest.**

KUBERNETES : MicroServices

- As part of New Deployment we will modify the node-istio manifest file to update the Gateway and VS rules.
- We will add a **Subset Rule** to define additional, version-based policies to the routing rules you are applying to your application Service.
- We would like to configure a routing rule that will send **80%** of traffic to our original application, and **20%** to the newer version.
- Next, add a **Destination Rule** that will apply rules to incoming traffic after that traffic has been routed to the appropriate Service.

- Application manifests are updated, but we still need to apply these changes to our **Kubernetes** cluster.
- We'll use the **kubectl apply command** to apply our changes without completely overwriting the existing configuration.

  ```
  kubectl apply -f node-app.yaml

  kubectl apply -f node-istio.yaml
  ```

Will see you in Next Lecture…

See you in next lecture …