Lab 6.1 - Configure SMI Traffic Splitting with Linkerd

Overview

In this lab, you'll be implementing traffic splitting by using the SMI specification with Linkerd to perform a canary release. The canary release will transition users to a new version of the demo app that fixes the bug with the donut emoji.

1. Connect to the Ubuntu VM that hosts your Kubernetes clusters.

2. To start the cluster that contains the Linkerd service mesh, issue this command:

```
yourname@ubuntu-vm:~$ docker start $(docker ps -a -f
name=linkerd-control-plane -q)

8af4c08524df
```

3. To switch the **kind** context to the Linkerd cluster, use this command:

```
yourname@ubuntu-vm:~$ kubectl config use-context kind-linkerd

Switched to context "kind-linkerd".
```

4. Install the Linkerd SMI

```
    yourname@ubuntu-vm:~$ curl --proto '=https' --tlsv1.2 -sSfL
https://linkerd.github.io/linkerd-smi/install | sh

    yourname@ubuntu-vm:~$ linkerd smi install | kubectl apply -f -
```

```
namespace/linkerd-smi created
deployment.apps/smi-adaptor created
clusterrole.rbac.authorization.k8s.io/smi-adaptor created
clusterrolebinding.rbac.authorization.k8s.io/smi-adaptor created
serviceaccount/smi-adaptor created
customresourcedefinition.apiextensions.k8s.io/trafficsplits.split.smi-sp
ec.io configured
```

5. Verify that the installation was successful.

```
yourname@ubuntu-vm:~$ linkerd smi check

linkerd-smi
-----------
√ linkerd-smi extension Namespace exists
√ SMI extension service account exists
√ SMI extension pods are injected
√ SMI extension pods are running
√ SMI extension proxies are healthy

Status check results are √
```

6. Create a new namespace and install the SMI sample app:

```
    yourname@ubuntu-vm:~$ kubectl create namespace trafficsplit-sample

    yourname@ubuntu-vm:~$ linkerd inject
https://raw.githubusercontent.com/linkerd/linkerd2/main/test/integration/viz/t
rafficsplit/testdata/application.yaml | kubectl -n trafficsplit-sample apply
-f -
```

7. Confirm that the installation of the app was successful.

```
    yourname@ubuntu-vm:~$ kubectl get deployments -n
trafficsplit-sample

      NAME          READY   UP-TO-DATE   AVAILABLE   AGE
backend       1/1     1            1            70s
failing       1/1     1            1            70s
slow-cooker   1/1     1            1            69s
```

8. Next, configure a traffic split to split traffic on the **`backend-svc`** to distribute load between it and the **`failing-svc`**.

```
yourname@ubuntu-vm:~$ cat <<EOF | kubectl apply -f -
apiVersion: split.smi-spec.io/v1alpha2
kind: TrafficSplit
metadata:
  name: backend-split
  namespace: trafficsplit-sample
spec:
  service: backend-svc
  backends:
  - service: backend-svc
    weight: 500
  - service: failing-svc
    weight: 500
EOF
```

9. Verify that the traffic splitting is working as expected by running the following command.

```
yourname@ubuntu-vm:~$ linkerd viz edges deploy -n trafficsplit-sample
```

```
      SRC           DST           SRC_NS                 DST_NS
SECURED
prometheus    backend       linkerd-viz            trafficsplit-sample   √
prometheus    failing       linkerd-viz            trafficsplit-sample   √
prometheus    slow-cooker   linkerd-viz            trafficsplit-sample   √
slow-cooker   backend       trafficsplit-sample    trafficsplit-sample   √
slow-cooker   failing       trafficsplit-sample    trafficsplit-sample   √
```

10. Clean up the environment with the sample app that you just deployed for this lab with the following command.

```
yourname@ubuntu-vm:~$ kubectl delete namespace/trafficsplit-sample
```

11. Make sure to stop the Linkerd cluster by running this command:

```
yourname@ubuntu-vm:~$ docker stop $(docker ps -a -f
name=linkerd-control-plane -q)

8af4c08524df
```