Lab 5.4 - Install a Consul Service Mesh

Overview

In this lab, you'll be installing a service mesh using Consul.

Consul provides a lot of the same functionality as other service meshes, but instead of being designed to run entirely on Kubernetes, Consul was built to connect services throughout your datacenter regardless of where they are running.

In the previous labs, you created an Ingress Controller with annotations. One of those annotations was for Consul. Because the configuration is already set, you don't have to do anything other than installing Consul itself because the annotation already exists inside of the Ingress Controller. Below is the Nginx Ingress Controller manifest with the Consul annotation highlighted in orange to remind you of what it looks like. **Please note** that you do **not** have to re-run the configuration as you already did in the previous labs.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: emojivoto
  name: ingress-emojivoto
  annotations:
    ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/service-upstream: "true"
    linkerd.io/inject: enabled
    kubernetes.io/ingress.class: istio
    consul.hashicorp.com/connect-inject: "true"
spec:
  ingressClassName: nginx
  rules:
    - http:
```



```
    paths:
    - pathType: Prefix
      path: "/"
      backend:
        service:
          name: web-svc
          port:
            number: 80
```

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

2. To start the cluster that will contain the Consul service mesh, issue this command:

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

2e64e6e909e1
```

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

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

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

4. To install Consul with Helm, issue this command:

```
yourname@ubuntu-vm:~$ helm repo add hashicorp
https://helm.releases.hashicorp.com

"hashicorp" has been added to your repositories
```

5. Enter the following **two** commands to configure Consul to run on your single node cluster and automatically inject the sidecar to pods in the **emojivoto** namespace. This may take a while to complete.

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

namespace/consul created

yourname@ubuntu-vm:~$ helm upgrade --install -n consul consul
hashicorp/consul --wait -f - <<EOF
```



```
global:
  name: consul
server:
  replicas: 1
  bootstrapExpect: 1
connectInject:
  enabled: true
EOF

NAME: consul
LAST DEPLOYED: Thu Jul  9 20:05:10 2020
NAMESPACE: consul
STATUS: deployed
REVISION: 1
NOTES:
Thank you for installing HashiCorp Consul!
Now that you have deployed Consul, you should look over the docs
on using
Consul with Kubernetes available here:
https://www.consul.io/docs/platform/k8s/index.html
Your release is named consul.
To learn more about the release if you are using Helm 2, run:
 $ helm status consul
 $ helm get consul
To learn more about the release if you are using Helm 3, run:
 $ helm status consul
 $ helm get all consul
```

6. Make sure Consul has started before proceeding with the next step by issuing the following command. You should see three names starting with **-consul**, and each one of them should have a status of **Running** and a Ready value of **1/1**. If you don't see those for each of the three names, wait a few minutes and run the command again.

```
yourname@ubuntu-vm:~$ kubectl get pods -n consul

consul-connect-injector-webhook-deployment-dcd56544f-kpq9t   1/1
Running   0          70s
consul-p2rv7                                                 1/1
Running   0          70s
consul-server-0                                              1/1
Running   0          68s
```


7. Let's explore what you just installed! Run the command below, which will output all of the configuration you installed in the cluster. This includes a Consul server backend, an agent that acts as the control plane for the sidecar proxies, and a service in charge of adding your services to the mesh. Running through this output will give you a better sense of Consul's architecture.

```
helm get all consul -n consul

NAME: consul
LAST DEPLOYED: Sat Jul 25 17:59:43 2020
NAMESPACE: consul
STATUS: deployed
REVISION: 1
USER-SUPPLIED VALUES:
connectInject:
 enabled: true
global:
 name: consul
server:
 bootstrapExpect: 1
 replicas: 1

[remaining content deleted for brevity]
```

8. After installing Consul, you need to add the application you installed to the mesh. Consul decides to add a pod to the mesh by looking for the **consul.hashicorp.com/connect-inject: "true"** annotation. Since the `consul` CLI does not have a command for automatically do this, you can manually accomplish this with the following command:

```
yourname@ubuntu-vm:~$ curl -sL
https://run.linkerd.io/emojivoto.yml \
  | sed 's|    metadata:|    metadata:\n      annotations:\n
consul.hashicorp.com/connect-inject: "true"|' \
  | kubectl apply -f -

namespace/emojivoto unchanged
serviceaccount/emoji unchanged
serviceaccount/voting unchanged
serviceaccount/web unchanged
service/emoji-svc unchanged
service/voting-svc unchanged
```



```
service/web-svc unchanged
deployment.apps/emoji configured
deployment.apps/vote-bot configured
deployment.apps/voting configured
deployment.apps/web configured
```

9. Verify that the Emoji Vote pods are running in the cluster and ready by issuing this command. You should see a status of "**Running**" and a Ready value of "**3/3**" for each of the four pods. These extra pods are the **consul-connect-sidecar-proxy** and the **consul-connect-lifecycle-sidecar**, which are responsible for routing traffic and interacting with the control plane, respectively. If you don't see the expected status and Ready values, wait a few minutes and try the command again.

```
yourname@ubuntu-vm:~$ kubectl get pod -n emojivoto -w

NAME                        READY   STATUS    RESTARTS   AGE
emoji-587bf97dbb-6jmsx      3/3     Running   0          91s
vote-bot-5f987ddd8f-7fsxk   3/3     Running   0          91s
voting-85477587c9-8qmfk     3/3     Running   0          91s
web-bd44c459c-dzczb         3/3     Running   0          91s
```

10. Once you see a **Running** status and a Ready value of **3/3** for all four names, hit Control-C to stop the status updates from being displayed. This may take a few minutes.

11. Consul does not automatically hijack the pods' networking as other meshes do. This means that you can still connect directly to the container running on the pod without going through the service proxy. This has the benefits of making migrations to mesh easier and giving options to opt out of the mesh, but you would need to update the `Service` definitions so requests go to the service proxy instead of the application container. Combined with the previous command above to inject the service proxies, the following command will inject and connect the application to the mesh.

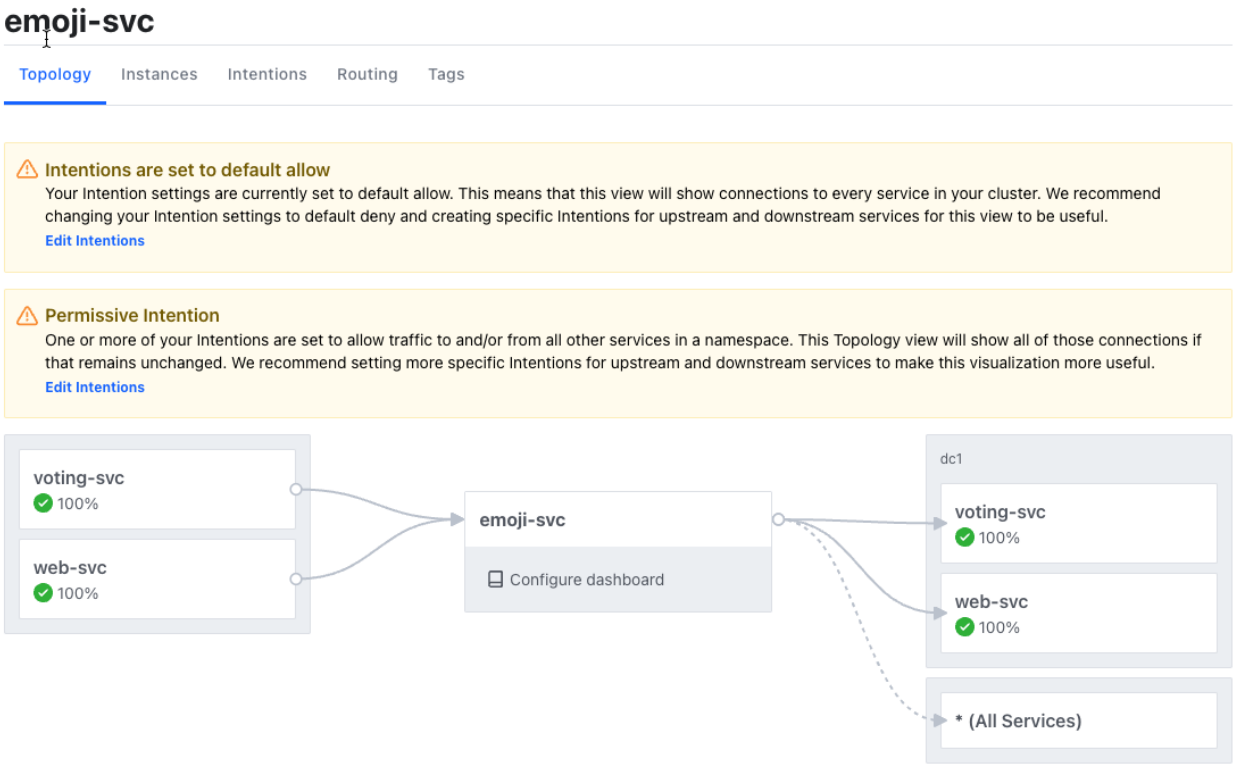
```
yourname@ubuntu-vm:~$ curl -sL
https://run.linkerd.io/emojivoto.yml \
  | sed 's|    metadata:|    metadata:\n      annotations:\n
consul.hashicorp.com/connect-inject: "true"|' \
  | sed 's|targetPort: 8080|targetPort: 20000|' \
  | kubectl apply -f -

namespace/emojivoto unchanged
serviceaccount/emoji unchanged
serviceaccount/voting unchanged
```



```
serviceaccount/web unchanged
service/emoji-svc configured
service/voting-svc configured
service/web-svc configured
deployment.apps/emoji unchanged
deployment.apps/vote-bot unchanged
deployment.apps/voting unchanged
deployment.apps/web unchanged
```

12. The demo application is now a part of the Consul Connect service mesh and ready to receive traffic from Ngins Ingress. Take some time to explore Consul a little more.



13. You've reached the end of this lab. If you're not starting the next lab right away, you can stop the Consul cluster by running this command:

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

2e64e6e909e1
```

