Lab 5.1 - Install a Linkerd Service Mesh

Overview

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

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

2. To start the cluster that will contain 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. Download the **`linkerd`** command line interface (CLI):

```
yourname@ubuntu-vm:~$ curl -sL https://run.linkerd.io/install |
sh

Downloading linkerd2-cli-stable-2.8.1-linux...
 % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
```

```
                                 Dload  Upload   Total   Spent
Left  Speed
100   644  100   644    0     0   4380      0 --:--:-- --:--:--
--:--:--  4380
100 37.0M  100 37.0M    0     0  17.5M      0  0:00:02  0:00:02
--:--:-- 22.8M
Download complete!

Validating checksum...
Checksum valid.

Linkerd stable-2.8.1 was successfully installed 🎉

Add the linkerd CLI to your path with:
 export PATH=$PATH:/home/yourname/.linkerd2/bin

Now run:
 linkerd check --pre                     # validate that Linkerd
can be installed
 linkerd install | kubectl apply -f -    # install the control
plane into the 'linkerd' namespace
 linkerd check                           # validate everything
worked!
 linkerd dashboard                       # launch the dashboard

Looking for more? Visit https://linkerd.io/2/next-steps
```

5. Add the linkerd CLI to your current **PATH**:

```
yourname@ubuntu-vm:~$ export PATH=$PATH:$HOME/.linkerd2/bin

[no output if the command succeeds]
```

6. You should also add the linkerd CLI to the **$PATH** variable for your shell with the command below (assuming you are using **bash** for your shell). This will permanently add it to your **PATH**. Without doing this once, if you log out and log back in, your **PATH** will be replaced and your **linkerd** commands will not work correctly until you manually update your **PATH** as you did in the previous step.

```
echo "export PATH=$PATH:$HOME/.linkerd2/bin" >> ~/.bashrc

[no output if the command succeeds]
```

7. Verify the download and that your cluster is ready to install Linkerd with the command below. If any of the status checks fail, investigate and address the issue, then run this command again to confirm the status checks pass before continuing to the next step.

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

kubernetes-api
--------------
√ can initialize the client
√ can query the Kubernetes API

kubernetes-version
------------------
√ is running the minimum Kubernetes API version
√ is running the minimum kubectl version

pre-kubernetes-setup
--------------------
√ control plane namespace does not already exist
√ can create non-namespaced resources
√ can create ServiceAccounts
√ can create Services
√ can create Deployments
√ can create CronJobs
√ can create ConfigMaps
√ can create Secrets
√ can read Secrets
√ can read extension-apiserver-authentication configmap
√ no clock skew detected

pre-kubernetes-capability
-------------------------
√ has NET_ADMIN capability
√ has NET_RAW capability

linkerd-version
---------------
√ can determine the latest version
√ cli is up-to-date

Status check results are √
```

8. Install Linkerd. Make sure that the single quotes you use are plain quotes and not curly quotes.

```
yourname@ubuntu-vm:~$ linkerd install \
  | sed 's|-enforced-host=.*|-enforced-host=|' \
  | kubectl apply -f -

namespace/linkerd created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-identity
created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-iden
tity created
[additional output omitted]
serviceaccount/linkerd-grafana created
configmap/linkerd-grafana-config created
service/linkerd-grafana created
deployment.apps/linkerd-grafana created
```

9. Wait for Linkerd to start before continuing. This command will check to see if everything is ready, and if not will keep you updates on the status.

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

kubernetes-api
--------------
√ can initialize the client
√ can query the Kubernetes API

kubernetes-version
------------------
√ is running the minimum Kubernetes API version
√ is running the minimum kubectl version

linkerd-existence
-----------------
√ 'linkerd-config' config map exists
√ heartbeat ServiceAccount exist
√ control plane replica sets are ready
√ no unschedulable pods
√ controller pod is running
√ can initialize the client
√ can query the control plane API

linkerd-config
--------------
√ control plane Namespace exists
√ control plane ClusterRoles exist
√ control plane ClusterRoleBindings exist
√ control plane ServiceAccounts exist
√ control plane CustomResourceDefinitions exist
√ control plane MutatingWebhookConfigurations exist
√ control plane ValidatingWebhookConfigurations exist
√ control plane PodSecurityPolicies exist

linkerd-identity
----------------
√ certificate config is valid
√ trust anchors are using supported crypto algorithm
√ trust anchors are within their validity period
√ trust anchors are valid for at least 60 days
√ issuer cert is using supported crypto algorithm
√ issuer cert is within its validity period
√ issuer cert is valid for at least 60 days
√ issuer cert is issued by the trust anchor

linkerd-api
-----------
√ control plane pods are ready
√ control plane self-check
√ [kubernetes] control plane can talk to Kubernetes
√ [prometheus] control plane can talk to Prometheus
√ tap api service is running

linkerd-version
---------------
√ can determine the latest version
√ cli is up-to-date

control-plane-version
---------------------
√ control plane is up-to-date
√ control plane and cli versions match
```

```
linkerd-addons
--------------
√ 'linkerd-config-addons' config map exists

linkerd-grafana
---------------
√ grafana add-on service account exists
√ grafana add-on config map exists
√ grafana pod is running

Status check results are √
```

10. Install the Linkerd dashboard so you can get a visual of everything that's happening in your environment:

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

11. Check the Linkerd viz environment to confirm it deployed as expected.

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

√ linkerd-viz Namespace exists
√ linkerd-viz ClusterRoles exist
√ linkerd-viz ClusterRoleBindings exist
√ tap API server has valid cert
√ tap API server cert is valid for at least 60 days
√ tap API service is running
√ linkerd-viz pods are injected
√ viz extension pods are running
√ viz extension proxies are healthy
√ viz extension proxies are up-to-date
√ viz extension proxies and cli versions match
√ prometheus is installed and configured correctly
√ can initialize the client
√ viz extension self-check

Status check results are √
```

12. With Linkerd, the proxy is lightweight and transparently intercepts traffic, so unlike the Envoy-based service mesh implementations, you will actually inject Linkerd into Nginx Ingress via sidecar proxy to the mesh by updating the Ingress manifest to include the

Istio injector annotation. Below is the manifest to run and you'll see the update to the annotation in orange:

```
yourname@ubuntu-vm:~$

kubectl apply -f - <<EOF
---
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
EOF
```

13. After installing Linkerd and updating the Nginx Ingress controller, you need to add the demo application you installed to the mesh by using **`linkerd`** and **`kubectl`**:

```
yourname@ubuntu-vm:~$ curl -sL
https://run.linkerd.io/emojivoto.yml \
  | linkerd inject - \
  | kubectl apply -f -

namespace "emojivoto" injected
serviceaccount "emoji" skipped
serviceaccount "voting" skipped
serviceaccount "web" skipped
service "emoji-svc" skipped
service "voting-svc" skipped
service "web-svc" skipped
deployment "emoji" injected
deployment "vote-bot" injected
deployment "voting" injected
deployment "web" injected

namespace/emojivoto configured
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
```

14. The demo application is now meshed in the Linkerd mesh and ready to receive traffic from Nginx Ingress. Expose the Linkerd Dashboard by entering this command:

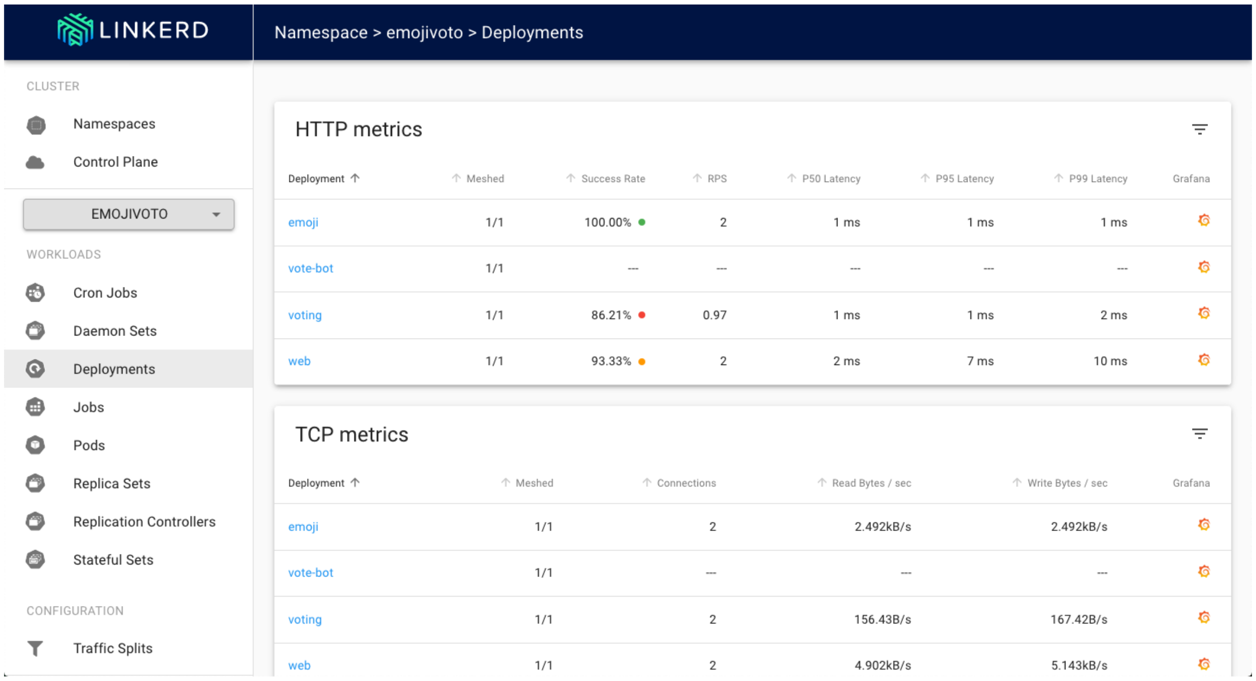
```
yourname@ubuntu-vm:~$ linkerd viz dashboard --port 8080 --address
0.0.0.0 --show url

http://0.0.0.0:8080
Grafana dashboard available at:
http://0.0.0.0:8080/grafana
```

15. Open and explore the Linkerd dashboard in your web browser by going to the IP address of your VM at port **`8080`**, which would look something like this:

    `http://127.0.0.1:8080`

    Here you can get a live look at how services in the mesh are performing. For example, if you take a look at the deployments running in the **`emojivoto`** namespace, you can see that there appears to be an issue with the **`voting`** service. You will learn how to address this in a future lab.



HTTP metrics

Deployment ↑	↑ Meshed	↑ Success Rate	↑ RPS	↑ P50 Latency	↑ P95 Latency	↑ P99 Latency	Grafana
emoji	1/1	100.00% ●	2	1 ms	1 ms	1 ms	
vote-bot	1/1	---	---	---	---	---	
voting	1/1	86.21% ●	0.97	1 ms	1 ms	2 ms	
web	1/1	93.33% ●	2	2 ms	7 ms	10 ms	

TCP metrics

Deployment ↑	↑ Meshed	↑ Connections	↑ Read Bytes / sec	↑ Write Bytes / sec	Grafana
emoji	1/1	2	2.492kB/s	2.492kB/s	
vote-bot	1/1	---	---	---	
voting	1/1	2	156.43B/s	167.42B/s	
web	1/1	2	4.902kB/s	5.143kB/s	

16. When you are done exploring the dashboard, hit Control-C in your terminal to stop the dashboard from running.

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

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

8af4c08524df
```