Exercise 11.1: Service Mesh

> If you have a large number of services to expose outside of the cluster, or to expose a low-number port on the host node you can deploy an ingress controller. While nginx and GCE have controllers mentioned a lot in Kubernetes.io, there are many to chose from. Even more functionality and metrics come from the use of a service mesh, such as Istio, Linkerd, Contour, Aspen, or several others.

1. We will install linkerd using their own scripts. There is quite a bit of output. Instead of showing all of it the output has been omitted. Look through the output and ensure that everything gets a green check mark. Some steps may take a few minutes to complete. Each command is listed here to make install easier. As well these steps are in the `setupLinkerd.txt` file.

   ```
   student@cp:~$ curl -sL run.linkerd.io/install | sh
   ```

   ```
   student@cp:~$ export PATH=$PATH:/home/student/.linkerd2/bin
   ```

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

   ```
   student@cp:~$ linkerd check --pre
   ```

   ```
   student@cp:~$ linkerd install | kubectl apply -f -
   ```

   ```
   student@cp:~$ linkerd check
   ```

   ```
   student@cp:~$ linkerd viz install | kubectl apply -f -
   ```

   ```
   student@cp:~$ linkerd viz check
   ```

   ```
   student@cp:~$ linkerd viz dashboard &
   ```

2. By default the GUI is on available on the localhost. We will need to edit the service and the deployment to allow outside access, in case you are using a cloud provider for the nodes. Edit to remove all characters after equal sign for `-enforced-host`, which is around line 59.

   ```
   student@cp:~/app2$ kubectl -n linkerd-viz edit deploy web
   ```

   ```yaml
   1  spec:
   2      containers:
   3      - args:
   4        - -linkerd-controller-api-addr=linkerd-controller-api.linkerd.svc.cluster.local:8085
   5        - -linkerd-metrics-api-addr=metrics-api.linkerd-viz.svc.cluster.local:8085
   6        - -cluster-domain=cluster.local
   7        - -grafana-addr=grafana.linkerd-viz.svc.cluster.local:3000
   8        - -controller-namespace=linkerd
   9        - -viz-namespace=linkerd-viz
   10       - -log-level=info
   11       - -enforced-host=                            #<-- Remove everything after equal sign
   12       image: cr.l5d.io/linkerd/web:stable-2.10.2
   13       imagePullPolicy: IfNotPresent
   ```

3. Now edit the http nodePort and type to be a NodePort.

   ```
   student@cp:~/app2$ kubectl edit svc web -n linkerd-viz
   ```

```
1   ....
2   ports:
3     - name: http
4       nodePort: 31500                                  #<-- Add line with an easy to remember port
5       port: 8084
6   ....
7     sessionAffinity: None
8     type: NodePort                                      #<-- Edit type to be NodePort
9   status:
10    loadBalancer: {}
11  ....
```

4. Test access using a local browser to your public IP. Your IP will be different than the one shown below.

    `student@cp:~/app2$ curl ifconfig.io`

```
1   104.197.159.20
```

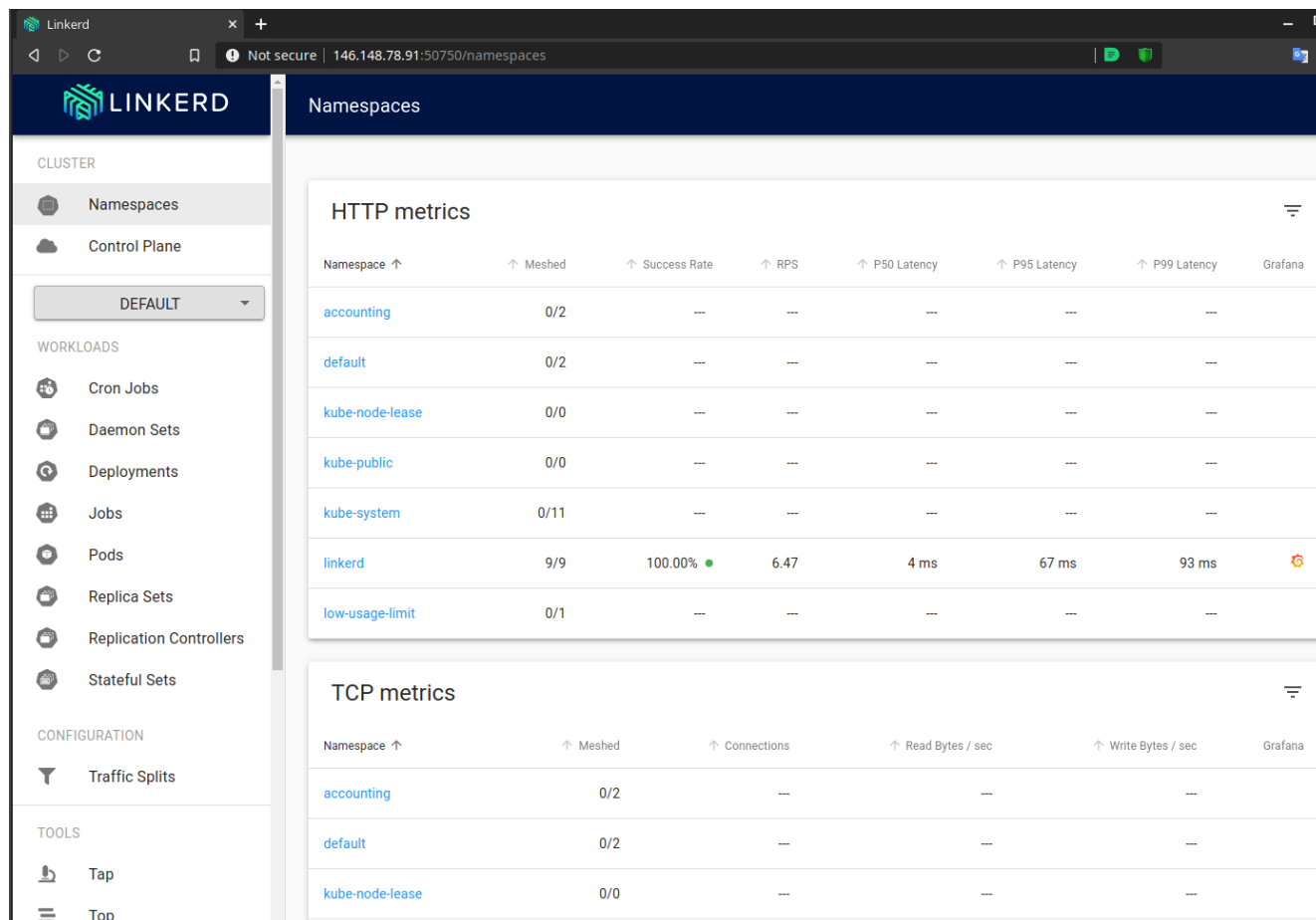5. From you local system open a browser and go to the public IP and the high-number nodePort.

Figure 11.3: **Main Linkerd Page**

6. In order for linkerd to pay attention to an object we need to add an annotation. The **linkerd inject** command will do this for us. Generate YAML and pipe it to **linkerd** then pipe again to **kubectl**. Expect an error about how the object was created, but the process will work. The command can run on one line if you omit the back-slash.

    `student@cp:~/app2$  kubectl -n accounting get deploy nginx-one -o yaml | \`
    `    linkerd inject - | kubectl apply -f -`

```
1  <output_omitted>
```

7. Check the GUI, you should see that the `accounting` namespaces and pods are now meshed, and the name is a link.

8. Generate some traffic to the pods, and watch the traffic via the GUI. Use the `service-lab` service.

```
student@cp:~$ kubectl -n accounting get svc
1  NAME           TYPE        CLUSTER-IP       EXTERNAL-IP    PORT(S)         AGE
2  nginx-one      ClusterIP   10.107.141.227   <none>         8080/TCP        5h15m
3  service-lab    NodePort    10.102.8.205     <none>         80:30759/TCP    5h14m
4
```

```
student@cp:~$ curl 10.102.8.205
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>Welcome to nginx!</title>
5  <output_omitted>
```
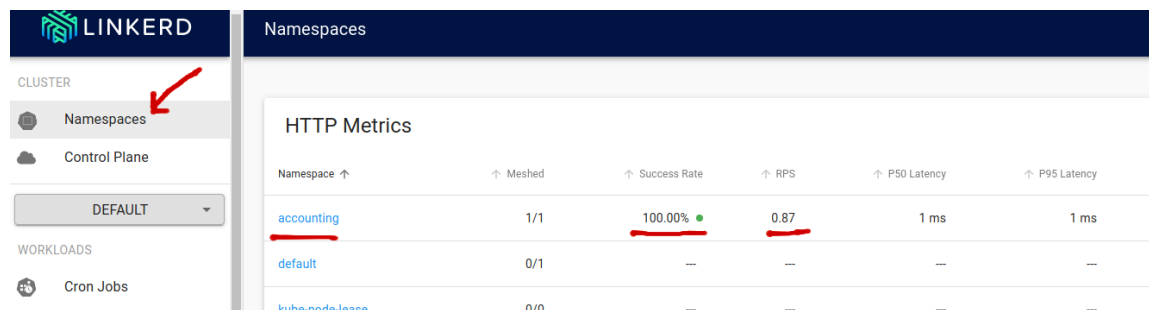

Figure 11.4: **Now shows meshed**

9. Scale up the `nginx-one` deployment. Generate traffic to get metrics for all the pods.

```
student@cp:~$ kubectl -n accounting scale deploy nginx-one --replicas=5
1  deployment.apps/nginx-one scaled
```

```
student@cp:~$ curl 10.102.8.205    #Several times
```

10. Explore some of the other information provided by the GUI. Note that the initial view is of the `default` namespaces. Change to `accounting` to see details of the nginx-one deployment.
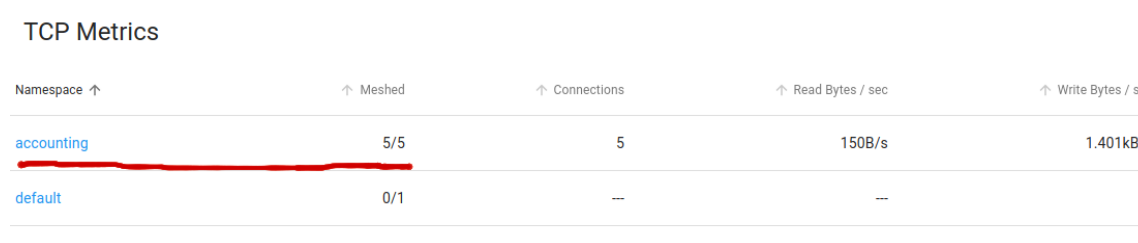

Figure 11.5: **Five meshed pods**