Exercise 7.2: Service Mesh and Ingress Controller

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.

   The most recent versions of linkerd may have some issues. As a result we will use a stable version of Linkerd, such as 2.12.2, or 2.11.4. Feel free to experiment with other versions as time and interest allows. Investigate **linkerd uninstall** and **linkerd uninject** as necessary.

   Some commands will take a while to spin up various pods.

```
student@cp:~/app2$ curl -sL run.linkerd.io/install > setup.sh
```

```
student@cp:~/app2$ vim setup.sh
```

```
#!/bin/sh

set -eu

LINKERD2_VERSION=${LINKERD2_VERSION:-stable-2.14.9}  #<-- Edit to earlier stable version
INSTALLROOT=${INSTALLROOT:-"${HOME}/.linkerd2"}

happyexit() {
....
```

```
student@cp:~/app2$ sh setup.sh

student@cp:~/app2$ export PATH=$PATH:/home/student/.linkerd2/bin

student@cp:~/app2$ linkerd check --pre

student@cp:~/app2$ linkerd install --crds | kubectl apply -f -

student@cp:~/app2$ linkerd install | kubectl apply -f -

student@cp:~/app2$ linkerd check

student@cp:~/app2$ linkerd viz install | kubectl apply -f -

student@cp:~/app2$ linkerd viz check

student@cp:~/app2$ linkerd viz dashboard &
```

2. By default the GUI is only available on the localhost, setup in the final dashboard command. 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 61.

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

```yaml
spec:
      containers:
      - args:
        - -linkerd-controller-api-addr=linkerd-controller-api.linkerd.svc.cluster.local:8085
        - -linkerd-metrics-api-addr=metrics-api.linkerd-viz.svc.cluster.local:8085
        - -cluster-domain=cluster.local
        - -grafana-addr=grafana.linkerd-viz.svc.cluster.local:3000
        - -controller-namespace=linkerd
        - -viz-namespace=linkerd-viz
        - -log-level=info
        - -enforced-host=                                #<-- Remove everything after equal sign
        image: cr.l5d.io/linkerd/web:stable-2.12.2
        imagePullPolicy: IfNotPresent
```

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

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

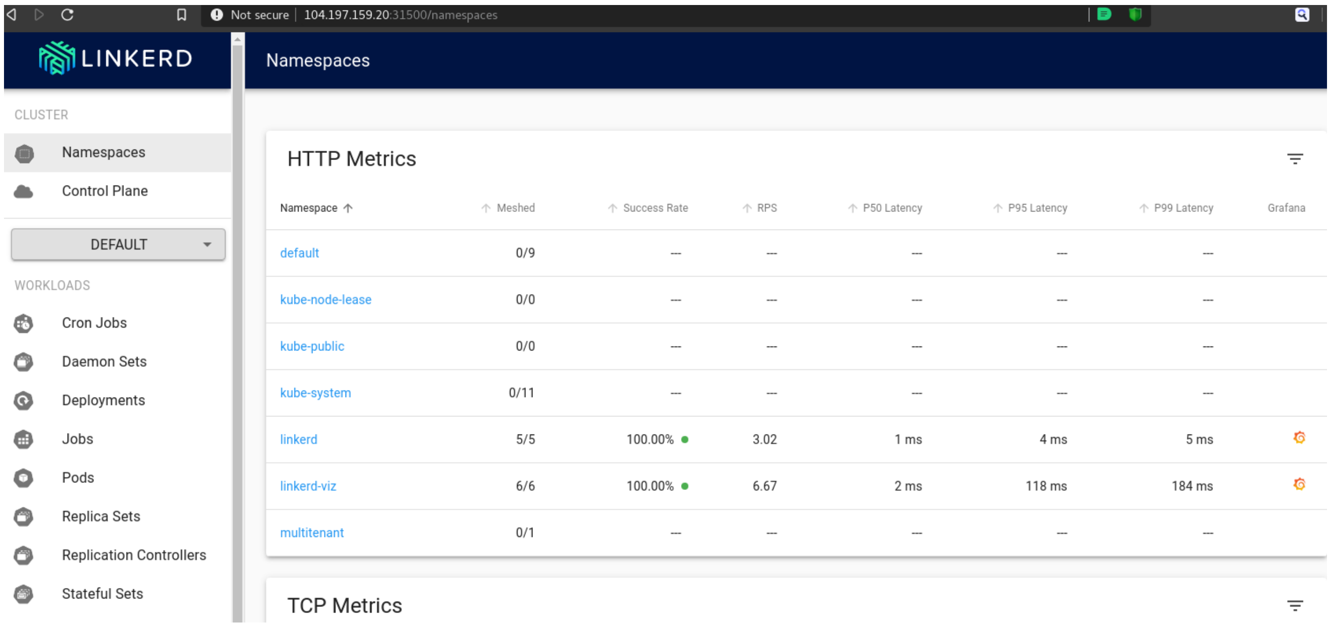
```yaml
....
ports:
  - name: http
    nodePort: 31500                             #<-- Add line with an easy to remember port
    port: 8084
....
  sessionAffinity: None
  type: NodePort                                #<-- Edit type to be NodePort
status:
  loadBalancer: {}
....
```

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
```

```
104.197.159.20
```

5. From you local system open a browser and go to the public IP and the high-number nodePort, such as 31500. It will not be the localhost port seen in output.



HTTP Metrics

Namespace	Meshed	Success Rate	RPS	P50 Latency	P95 Latency	P99 Latency	Grafana
default	0/9	—	—	—	—	—	
kube-node-lease	0/0	—	—	—	—	—	
kube-public	0/0	—	—	—	—	—	
kube-system	0/11	—	—	—	—	—	
linkerd	5/5	100.00% ●	3.02	1 ms	4 ms	5 ms	
linkerd-viz	6/6	100.00% ●	6.67	2 ms	118 ms	184 ms	
multitenant	0/1	—	—	—	—	—	

TCP Metrics

Figure 7.5: **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 multitenant get deploy mainapp -o yaml | \
     linkerd inject - | kubectl apply -f -
```

```
<output_omitted>
```

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

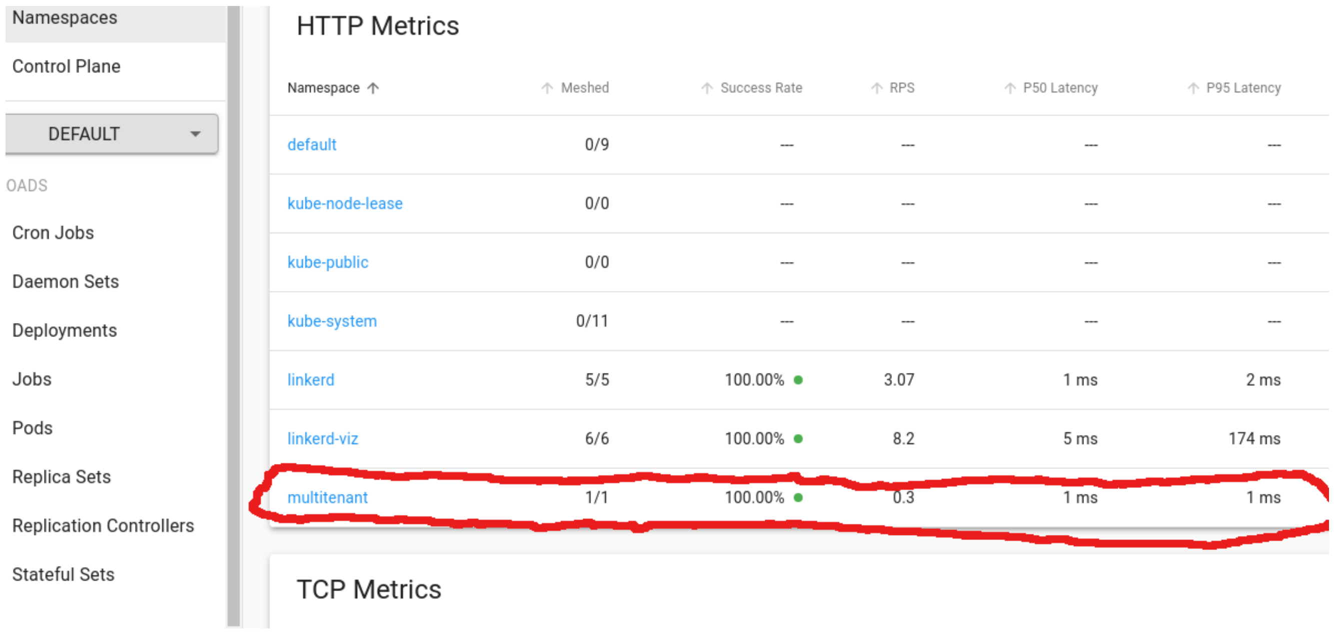
8. Generate some traffic to the pods, and watch the traffic via the GUI.

```
student@cp:~$ kubectl -n multitenant get svc
```

```
NAME        TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
shopping    NodePort    10.102.8.205   <none>        80:32518/TCP   4h54m
```

```
student@cp:~$ curl 10.102.8.205
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<output_omitted>
```



HTTP Metrics

Namespace	Meshed	Success Rate	RPS	P50 Latency	P95 Latency
default	0/9	—	—	—	—
kube-node-lease	0/0	—	—	—	—
kube-public	0/0	—	—	—	—
kube-system	0/11	—	—	—	—
linkerd	5/5	100.00% ●	3.07	1 ms	2 ms
linkerd-viz	6/6	100.00% ●	8.2	5 ms	174 ms
multitenant	1/1	100.00% ●	0.3	1 ms	1 ms

Figure 7.6: **Now shows meshed**

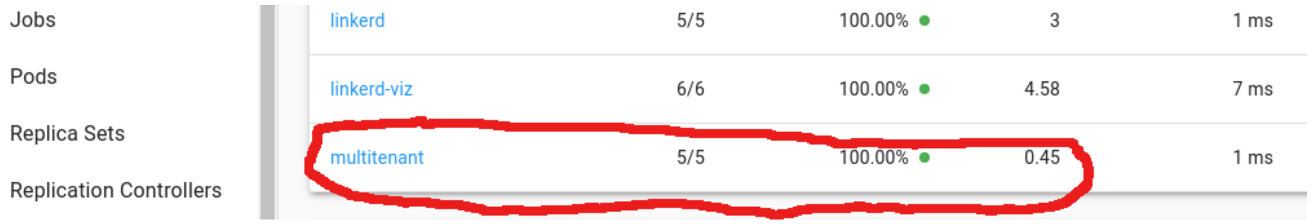
9. Scale up the mainapp deployment. Generate traffic to get metrics for all the pods.

```
student@cp:~$ kubectl -n multitenant scale deploy mainapp --replicas=5
```

```
deployment.apps/mainapp scaled
```

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

10. Explore some of the other information provided by the GUI.



linkerd	5/5	100.00% ●	3	1 ms
linkerd-viz	6/6	100.00% ●	4.58	7 ms
multitenant	5/5	100.00% ●	0.45	1 ms

Figure 7.7: **Five meshed pods**

11. Linkerd does not come with an ingress controller, so we will add one to help manage traffic. We will leverage a **Helm** chart to install an ingress controller. Search the hub to find that there are many available.

```
student@cp:~$ helm search hub ingress
```

```
URL                                                     CHART VERSION
APP VERSION                DESCRIPTION
https://artifacthub.io/packages/helm/k8s-as-hel...       1.0.2
v1.0.0                     Helm Chart representing a single Ingress Kubern...
https://artifacthub.io/packages/helm/openstack-...       0.2.1
v0.32.0                    OpenStack-Helm Ingress Controller
<output_omitted>
https://artifacthub.io/packages/helm/api/ingres...       3.29.1
0.45.0                     Ingress controller for Kubernetes using NGINX a...
https://artifacthub.io/packages/helm/wener/ingr...       3.31.0
0.46.0                     Ingress controller for Kubernetes using NGINX a...
https://artifacthub.io/packages/helm/nginx/ngin...       0.9.2
```

```
1.11.2                    NGINX Ingress Controller
<output_omitted>
```

12. We will use a popular ingress controller provided by **NGINX**.

```
student@cp:~$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

```
"ingress-nginx" has been added to your repositories
```

```
student@cp:~$ helm repo update
```

```
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
Update Complete. -Happy Helming!-
```

13. Download and edit the `values.yaml` file and change it to use a `DaemonSet` instead of a `Deployment`. This way there will be a pod on every node to handle traffic if using an external load balancer.

```
student@cp:~$ helm fetch ingress-nginx/ingress-nginx --untar
```

```
student@cp:~$ cd ingress-nginx
```

```
student@cp:~/ingress-nginx$ ls
```

```
CHANGELOG.md  Chart.yaml  OWNERS  README.md  ci  templates  values.yaml
```

```
student@cp:~/ingress-nginx$ vim values.yaml
```

values.yaml

```
....
  ## DaemonSet or Deployment
  ##
  kind: DaemonSet                      #<-- Change to DaemonSet, around line 150

  ## Annotations to be added to the controller Deployment or DaemonSet
....
```

14. Now install the controller using the chart. Note the use of the dot (.) to look in the current directory.

```
student@cp:~/ingress-nginx$ helm install myingress .
```

```
NAME: myingress
LAST DEPLOYED: Wed May 19 22:24:27 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The ingress-nginx controller has been installed.
It may take a few minutes for the LoadBalancer IP to be available.
You can watch the status by running
'kubectl --namespace default get services -o wide -w myingress-ingress-nginx-controller'

An example Ingress that makes use of the controller:
<output_omitted>
```

15. We now have an ingress controller running, but no rules yet. View the resources that exist. Use the **-w** option to watch the ingress controller service show up. After it is available use **ctrl-c** to quit and move to the next command.

```
student@cp:˜$ kubectl get ingress --all-namespaces
```

```
No resources found
```

```
student@cp:˜$ kubectl --namespace default get services -o wide  myingress-ingress-nginx-controller
```

```
NAME                                 TYPE           CLUSTER-IP      EXTERNAL-IP
   PORT(S)                      AGE   SELECTOR
myingress-ingress-nginx-controller   LoadBalancer   10.104.227.79   <pending>
   80:32558/TCP,443:30219/TCP   47s   app.kubernetes.io/component=controller,
   app.kubernetes.io/instance=myingress,app.kubernetes.io/name=ingress-nginx
```

```
student@cp:˜$ kubectl get pod --all-namespaces |grep nginx
```

```
default      myingress-ingress-nginx-controller-mrqt5    1/1     Running   0      20s
default      myingress-ingress-nginx-controller-pkdxm    1/1     Running   0      62s
default      nginx-b68dd9f75-h6ww7                       1/1     Running   0      21h
```

16. Now we can add rules which match HTTP headers to services. Remember to check the course files from the tarball.

```
student@cp:˜$ vim ingress.yaml
```

ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-test
  annotations:
    nginx.ingress.kubernetes.io/service-upstream: "true"
  namespace: default
spec:
  ingressClassName: nginx
  rules:
  - host: www.example.com
    http:
      paths:
      - backend:
          service:
            name: secondapp
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
```

17. Create then verify the ingress is working. If you don't pass a matching header you should get a 404 error.

```
student@cp:˜$ kubectl create -f ingress.yaml
```

```
ingress.networking.k8s.io/ingress-test created
```

```
student@cp:˜$ kubectl get ingress
```

```
NAME           CLASS    HOSTS             ADDRESS   PORTS    AGE
ingress-test   nginx    www.example.com             80       5s
```

```
student@cp:~$ kubectl get pod -o wide |grep myingress
```

```
myingress-ingress-nginx-controller-mrqt5   1/1     Running   0          8m9s    192.168.219.118
  cp   <none>           <none>
myingress-ingress-nginx-controller-pkdxm   1/1     Running   0          8m9s    192.168.219.119
  worker   <none>           <none>
```

```
student@cp:~/ingress-nginx$ curl 192.168.219.118
```

```
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

18. Check the ingress service and expect another 404 error, don't use the admission controller.

```
student@cp:~/ingress-nginx$ kubectl get svc |grep ingress
```

```
myingress-ingress-nginx-controller             LoadBalancer   10.104.227.79    <pending>
  80:32558/TCP,443:30219/TCP   10m
myingress-ingress-nginx-controller-admission   ClusterIP      10.97.132.127    <none>
  443/TCP                      10m
```

```
student@cp:~/ingress-nginx$ curl 10.104.227.79
```

```
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

19. Now pass a matching header and you should see the default web server page.

```
student@cp:~/ingress-nginx$ curl -H "Host: www.example.com" http://10.104.227.79
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
<output_omitted>
```

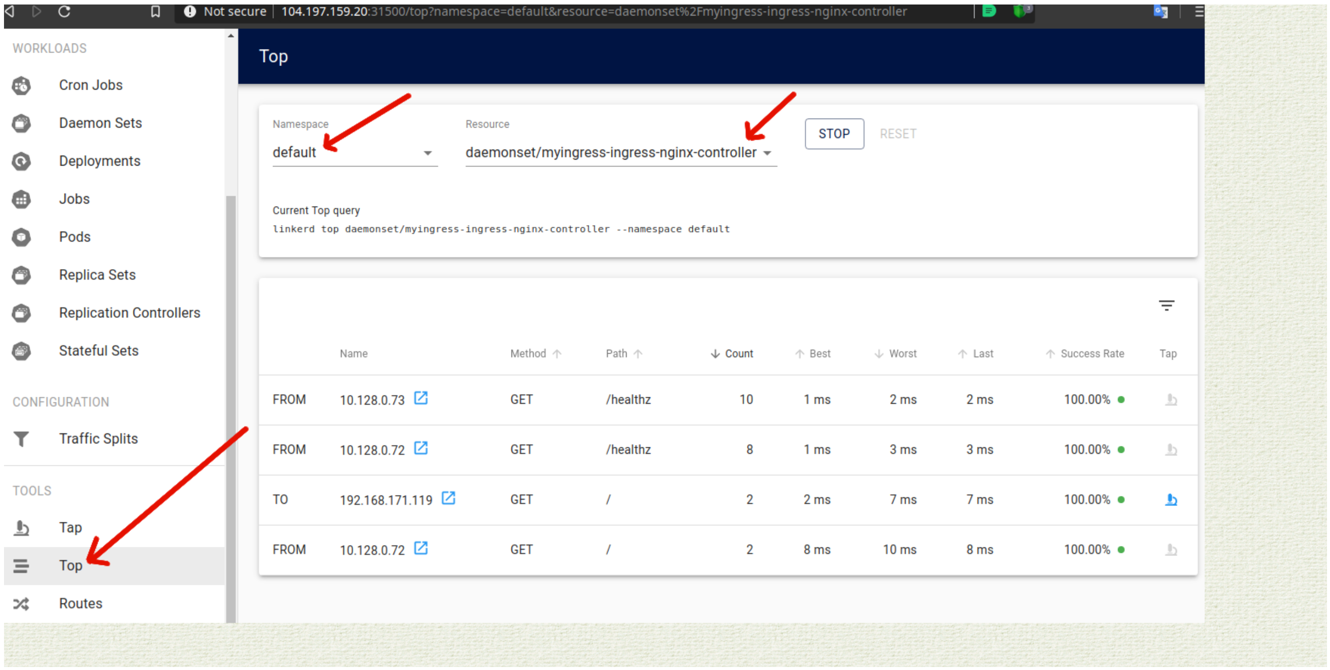
20. We can add an annotation to the ingress pods for Linkerd. You will get some warnings, but the command will work.

```
student@cp:~/ingress-nginx$ kubectl get ds myingress-ingress-nginx-controller -o yaml  | \
    linkerd inject --ingress - | kubectl apply -f -
```

```
daemonset "myingress-ingress-nginx-controller" injected

Warning: resource daemonsets/myingress-ingress-nginx-controller is missing the
kubectl.kubernetes.io/last-applied-configuration annotation which is required
by kubectl apply. kubectl apply should only be used on resources created
declaratively by either kubectl create --save-config or kubectl apply. The
missing annotation will be patched automatically.
daemonset.apps/myingress-ingress-nginx-controller configured
```

21. Go to the `Top` page, change the namespace to default and the resource to `daemonset/myingress-ingress-nginx-controller`. Press start then pass more traffic to the ingress controller and view traffic metrics via the GUI. Let top run so we can see another page added in an upcoming step.



Current Top query

```
linkerd top daemonset/myingress-ingress-nginx-controller --namespace default
```

	Name	Method	Path	Count	Best	Worst	Last	Success Rate	Tap
FROM	10.128.0.73	GET	/healthz	10	1 ms	2 ms	2 ms	100.00% ●	
FROM	10.128.0.72	GET	/healthz	8	1 ms	3 ms	3 ms	100.00% ●	
TO	192.168.171.119	GET	/	2	2 ms	7 ms	7 ms	100.00% ●	
FROM	10.128.0.72	GET	/	2	8 ms	10 ms	8 ms	100.00% ●	

Figure 7.8: **Ingress Traffic**

22. At this point we would keep adding more and more web servers. We'll configure one more, which would then could be a process continued as many times as desired. Begin by deploying another **nginx** server. Give it a label and expose port 80.

```
student@cp:˜/app2$ kubectl create deployment thirdpage --image=nginx
```

```
deployment.apps "thirdpage" created
```

23. Assign a label for the ingress controller to match against. Your pod name is unique, you can use the **Tab** key to complete the name.

```
student@cp:˜/app2$ kubectl label pod thirdpage-<tab> example=third
```

24. Expose the new server as a NodePort.

```
student@cp:˜/app2$ kubectl expose deployment thirdpage --port=80 --type=NodePort
```

```
service/thirdpage exposed
```

25. Now we will customize the installation. Run a bash shell inside the new pod. Your pod name will end differently. Install **vim** or an editor inside the container then edit the `index.html` file of nginx so that the title of the web page will be `Third Page`. Much of the command output is not shown below.

```
student@cp:˜/app2$ kubectl exec -it thirdpage-<Tab> -- /bin/bash
```

On Container

```
root@thirdpage-:/# apt-get update

root@thirdpage-:/# apt-get install vim -y
```

```
root@thirdpage-:/# vim /usr/share/nginx/html/index.html
```

```
<!DOCTYPE html>
<html>
<head>
<title>Third Page</title>     #<-- Edit this line
<style>
<output_omitted>
```

```
root@thirdpage-:/$ exit
```

Edit the ingress rules to point the thirdpage service. It may be easiest to copy the existing `host` stanza and edit the `host` and `name`.

26. `student@cp:˜/app2$ kubectl edit ingress ingress-test`

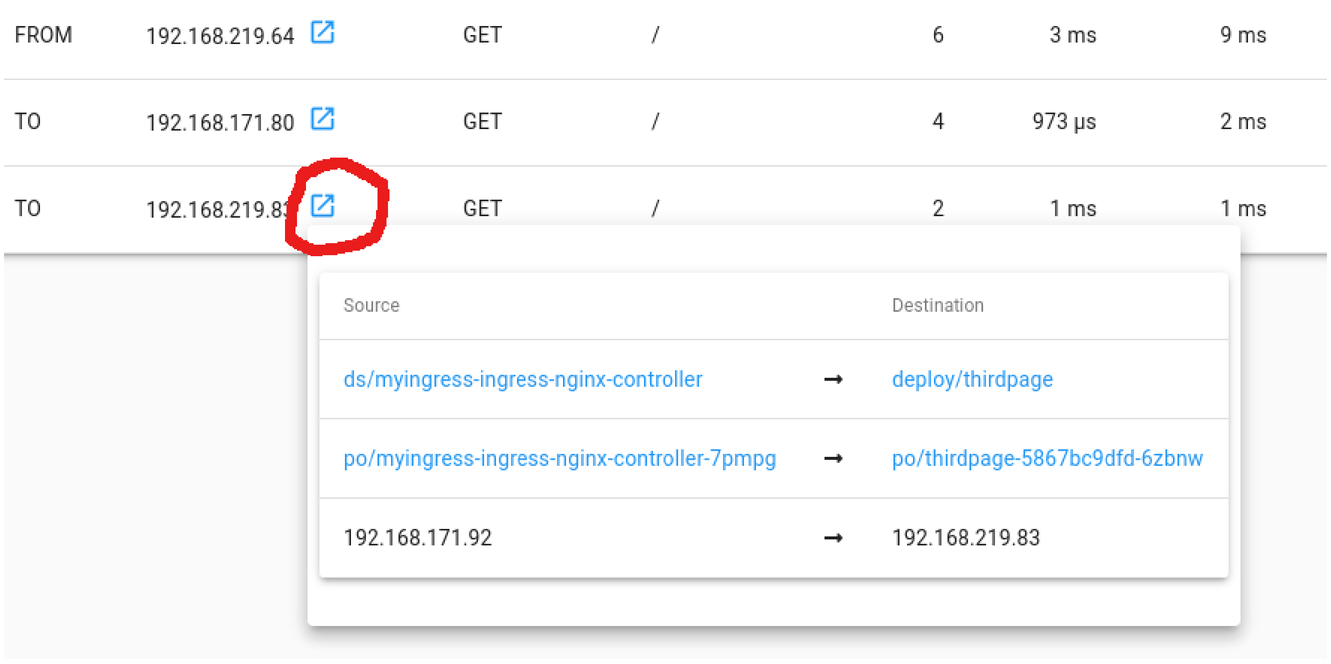
ingress-test

```yaml
....
spec:
  rules:
  - host: thirdpage.org
    http:
      paths:
      - backend:
          service:
            name: thirdpage
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
  - host: www.example.com
    http:
      paths:
      - backend:
          service:
            name: secondapp
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
status:
....
```

27. Test the second `Host:` setting using **curl** locally as well as from a remote system, be sure the `<title>` shows the non-default page. Use the main IP of either node. The Linkerd GUI should show a new `TO` line, if you select the small blue box with an arrow you will see the traffic is going to thirdpage.

```
student@cp:˜/app2$ curl -H "Host: thirdpage.org" http://10.128.0.7/
```

```
<!DOCTYPE html>
<html>
<head>
<title>Third Page</title>
<style>
<output_omitted>
```



FROM	192.168.219.64	GET	/	6	3 ms	9 ms
TO	192.168.171.80	GET	/	4	973 µs	2 ms
TO	192.168.219.83	GET	/	2	1 ms	1 ms

Source		Destination
ds/myingress-ingress-nginx-controller	→	deploy/thirdpage
po/myingress-ingress-nginx-controller-7pmpg	→	po/thirdpage-5867bc9dfd-6zbnw
192.168.171.92	→	192.168.219.83

Figure 7.9: **Linkerd Top Metrics**

28. Consider how you would edit the ingress rules to point at a different server, as we did when editing the service selector. Deploy the **httpd** server again and test changing traffic from `thirdpage` over to a new `httpd` sever using only ingress rule edits.