Exercise 11.2: Ingress Controller

We will use the **Helm** tool we learned about earlier to install an ingress controller.

1. Create two deployments, `web-one` and `web-two`, both running nginx. Expose both as ClusterIP services. Use previous content to determine the steps if you are unfamiliar. Test that both ClusterIPs work before continuing to the next step.

2. 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>
```

3. 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!-
```

4. 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.

```
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
....
```

5. 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 2021
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>
```

6. 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 -w 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
```

7. Now we can add rules which match HTTP headers to services.

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

ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
```

```yaml
metadata:
  name: ingress-test
  namespace: default
spec:
  rules:
  - host: www.external.com
    http:
      paths:
      - backend:
          service:
            name: web-one
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
status:
  loadBalancer: {}
```

8. 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   <none>   www.external.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    192.168.219.118
  cp   <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>
```

9. 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>
```

10. Now pass a header which matches a URL to one of the services we exposed in an earlier step. You should see the default nginx web server page.

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

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

11. 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
```

12. 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.

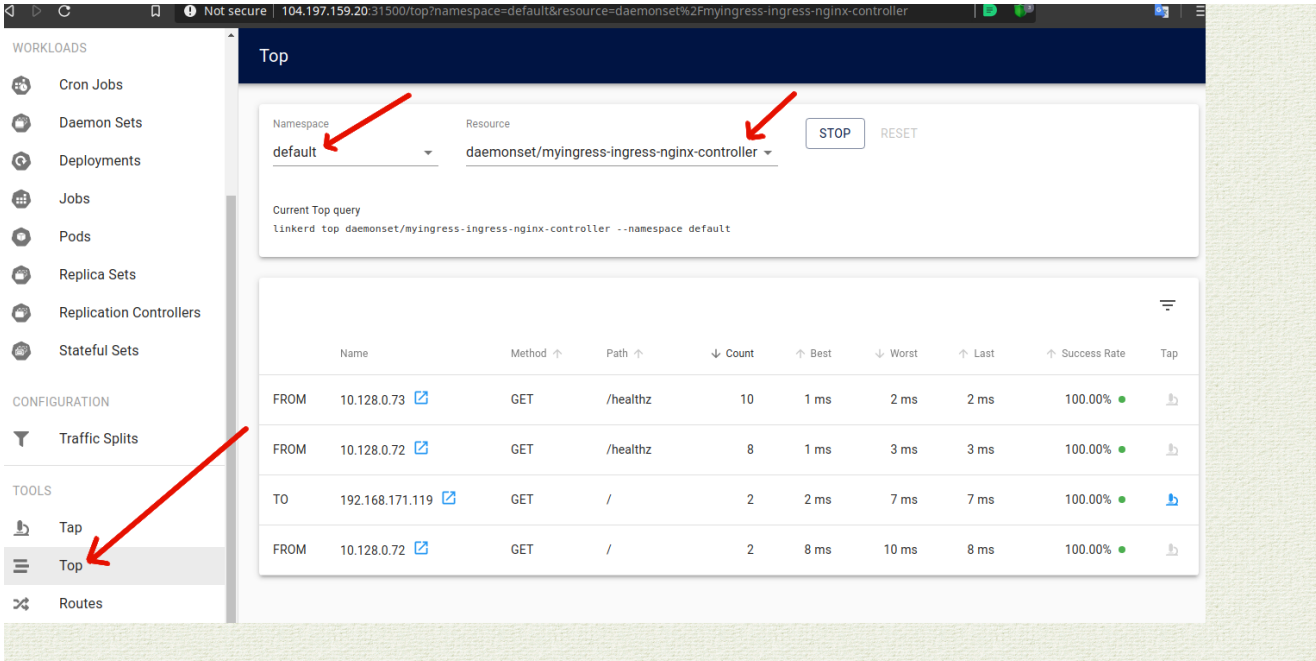

Figure 11.6: **Ingress Traffic**

13. At this point we would keep adding more and more servers. We'll configure one more, which would then could be a process continued as many times as desired.

    Customize the web-two welcome page. Run a bash shell inside the web-two 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 `Internal Welcome Page`. Much of the command output is not shown below.

```
student@cp:~$ kubectl exec -it web-two-<Tab> -- /bin/bash
```

On Container

```
root@web-two-...-:/# apt-get update

root@web-two-...-:/# apt-get install vim -y

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

```
<!DOCTYPE html>
<html>
<head>
<title>Internal Welcome 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`.

14. `student@cp:~/app2$ kubectl edit ingress ingress-test`

ingress-test

```yaml
....
spec:
  rules:
  - host: internal.org
    http:
      paths:
      - backend:
          service:
            name: web-two
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
  - host: www.external.com
    http:
      paths:
      - backend:
          service:
            name: web-one
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
status:
....
```

15. 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 internal.org.

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

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

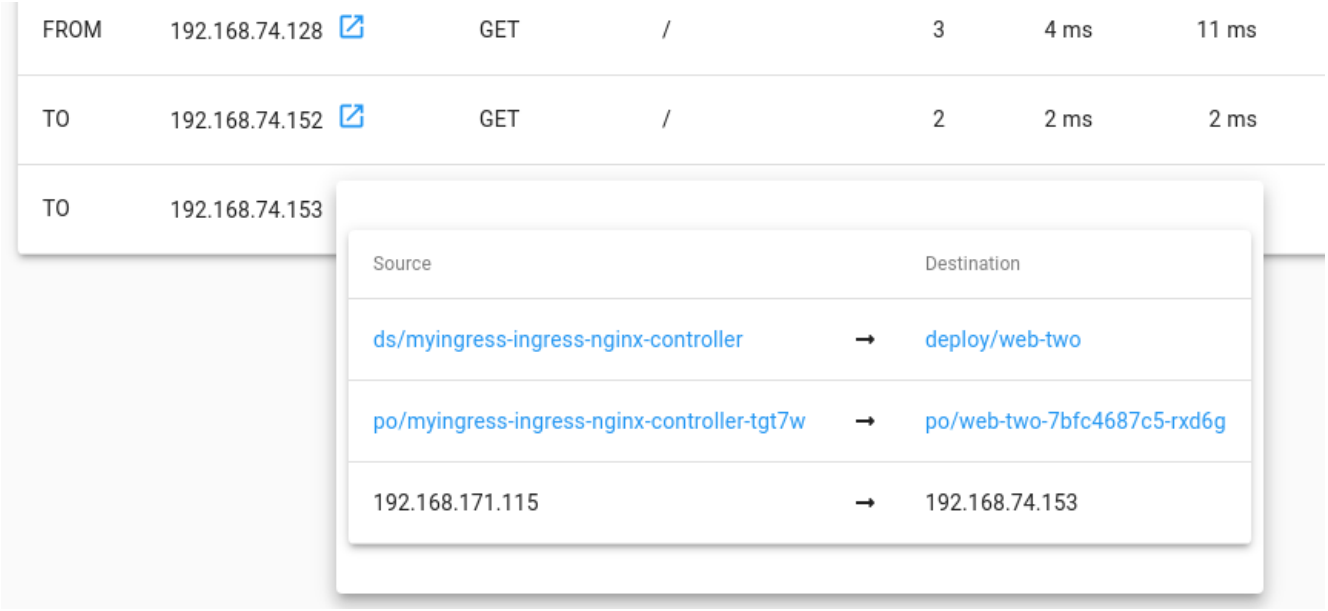

Figure 11.7: **Linkerd Top Metrics**