

Lab13 – Defining and using an Ingress

In this exercise, you will create an Ingress with rules that routes traffic to 2 Services.

Ingress Controller setup

The Kubernetes documentation (<https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/>) lists a wide range of Ingress Controller implementations.

1. Use command bellow to implement an ingress controller.

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.10.0/deploy/static/provider/cloud/deploy.yaml

```
brahim@Training:~/lab13-ingress$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.10.0/deploy/static/provider/cloud/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
brahim@Training:~/lab13-ingress$
```

2. The Ingress controller will run as a Pod in the `ingress-nginx` namespace. Make sure that the Pod `ingress-nginx-controller-...` transitions into the "Running" status.

```
brahim@Training:~/lab13-ingress$ kubectl get pod -n ingress-nginx
NAME                                READY   STATUS    RESTARTS   AGE
ingress-nginx-admission-create-cx8cv 0/1     Completed   0          28m
ingress-nginx-admission-patch-8smgg 0/1     Completed   2          28m
ingress-nginx-controller-594555f486-vcn2k 1/1     Running    0          28m
brahim@Training:~/lab13-ingress$
brahim@Training:~/lab13-ingress$
```

Using Ingress

3. Create two pods that output a slightly different response :

- *apple-app* : execute image *hashicorp/http-echo*, print apple
- *banana-app* : execute image *hashicorp/http-echo*, print banana

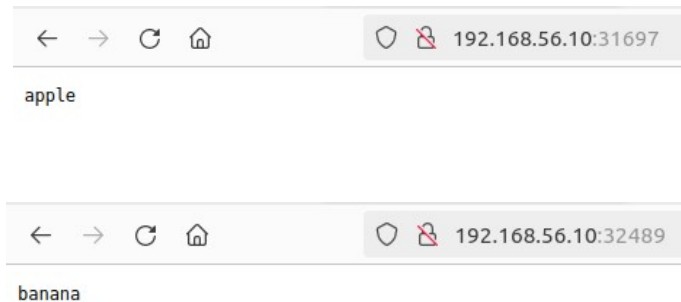
```
brahim@Training:~/lab13-ingress$ vim apple-app.yaml
brahim@Training:~/lab13-ingress$ cat apple-app.yaml
kind: Pod
apiVersion: v1
metadata:
  name: apple-app
  labels:
    app: apple
spec:
  containers:
  - name: apple-app
    image: hashicorp/http-echo
    args:
      - "-text=apple"
brahim@Training:~/lab13-ingress$
brahim@Training:~/lab13-ingress$ kubectl apply -f apple-app.yaml
pod/apple-app created
brahim@Training:~/lab13-ingress$
brahim@Training:~/lab13-ingress$ vim banana-app.yaml
brahim@Training:~/lab13-ingress$ cat banana-app.yaml
kind: Pod
apiVersion: v1
metadata:
  name: banana-app
  labels:
    app: banana
spec:
  containers:
  - name: banana-app
    image: hashicorp/http-echo
    args:
      - "-text=banana"
brahim@Training:~/lab13-ingress$ kubectl apply -f banana-app.yaml
pod/banana-app created
brahim@Training:~/lab13-ingress$ □
```

4. Expose the pods with a Services named `apple` and `banana` of type `NodePort`.

The Services routes traffic, respectively, to the Pods `apple-app` and `banana-app`.

```
brahim@Training:~/lab13-ingress$ kubectl expose pod apple-app --port=5678 --type=NodePort --selector app=apple
service/apple-app exposed
brahim@Training:~/lab13-ingress$
brahim@Training:~/lab13-ingress$ kubectl expose pod banana-app --port=5678 --type=NodePort --selector app=banana
service/banana-app exposed
brahim@Training:~/lab13-ingress$
brahim@Training:~/lab13-ingress$ kubectl get svc
NAME                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
apple-app           NodePort            10.108.188.115  <none>           5678:31697/TCP   25s
banana-app          NodePort            10.109.222.212  <none>           5678:32489/TCP   5s
kubernetes           ClusterIP           10.96.0.1       <none>           443/TCP          8m25s
brahim@Training:~/lab13-ingress$ □
```

- Make a request to the endpoints of the applications to test services.



- Create an Ingress that exposes the paths `/apple` and `/banana` for the host `fruit.exposed.com`. The traffic should be routed to the Services created earlier.

```
brahim@Training:~/lab13-ingress$ vim ingress.yaml
brahim@Training:~/lab13-ingress$ cat ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: fruit-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: fruit.exposed.com
    http:
      paths:
      - path: "/apple"
        pathType: Exact
        backend:
          service:
            name: apple-app
            port:
              number: 5678
      - path: "/banana"
        pathType: Prefix
        backend:
          service:
            name: banana-app
            port:
              number: 5678
brahim@Training:~/lab13-ingress$ kubectl apply -f ingress.yaml
ingress.networking.k8s.io/fruit-ingress created
brahim@Training:~/lab13-ingress$
```

- List the Ingress object. The value for the IP address will populate after waiting for a little while. You may have to run the command multiple times.

```
brahim@Training:~/lab13-ingress$ kubectl get ing
NAME          CLASS  HOSTS          ADDRESS          PORTS  AGE
fruit-ingress  nginx  fruit.exposed.com    
brahim@Training:~/lab13-ingress$
```

8. Add an entry in `/etc/hosts` that maps the virtual node IP address to the host `fruit.exposed.com`.

```
brahim@Training:~/lab13-ingress$ sudo vim /etc/hosts
brahim@Training:~/lab13-ingress$ cat /etc/hosts
127.0.0.1    localhost
127.0.1.1    Training
# 127.0.0.1    gitlab.orsys.fr
127.0.0.1    fruit.exposed.com

# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
brahim@Training:~/lab13-ingress$
```

9. Now, forward a local port to the ingress controller with this command :

kubectl port-forward --namespace=ingress-nginx service/ingress-nginx-controller 8080:80

```
brahim@Training:~/lab13-ingress$ kubectl port-forward --namespace=ingress-nginx service/ingress-nginx-controller 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
□
```

10. Test access to app using curl or your browser

```
brahim@Training:~/lab13-ingress$ curl fruit.exposed.com:8080/apple
apple
brahim@Training:~/lab13-ingress$ curl fruit.exposed.com:8080/banana
banana
brahim@Training:~/lab13-ingress$ _
```

apple

banana