08: Ingress

- Access to multiple web apps through a single LoadBalancer Service.
- Ingress it's a resource in the Kubernetes API
- LoadBalancer is a Kubernetes Service object of type=LoadBalancer

Benefits

- **NodePorts**:
  - only works on high port numbers
  - require knowledge of node name or IPs
- **LoadBalancer**:
  - require 1-to-1 mapping between an internal Service and a cloud load-balancer
  - a cluster with 25 internet-facing apps will need 25 cloud load-balancers
- **Ingress** Fixes:
  - uses a single cloud load-balancer
  - ports 80 to 443
  - host-based and path-based routing to send traffic to the backend service

Architecture

- Stable resource in the Kubernetes API
- v1 object
- spec: defines rules to govern traffic rounting and the controller implements them
- Once created **Ingress Controller** you deploy **Ingress Objects** with rules to hwo route requests.
- Ingress operates at layer 7 of the OSI model (App):
  - It has awareness of HTTP headers
  - can inspect them and forward traffic based on hostnames and paths

host-based	path-based	K8s Backend Svc
shield.mcu.com	mcu.com/shield	svc-shield
hydra.mcu.com	mcu.com/hydra	svc-hydra

INGINX Ingress Controller

- Installed from a YAML file hosted in Kubernetes GitHub repo.
- Namespace, ServiceAccounts, ConfigMap, Roles, etc.
- See `https://github.com/kubernetes/ingress-nginx/releases`

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-
nginx/

controller-v1.1.0/deploy/static/provider/cloud/deploy.yaml
```

```
kubectl get pods -n ingress-nginx \ -l app.kubernetes.io/name=ingress-nginx
```

Ingress Classes for Clustrers with multiple Ingress Controllers

1. Assign each Ingress controller to an Ingress class
2. When create Ingress objects assign them to an Ingress class

```
kubectl get ingressclass
```

Config host-based and path-based rounting

This section deploys two apps and a single Ingress object. Ingress routes both apps via a single load balancer

1. Deploy an app called `shield` and from it with a ClusterIP Service called svc-shield
2. Deploy an app called `hydra` and front it with a ClusterIP Service called svc-hydra
3. Deployu an Ingress object to route the following hostnames and paths
    - Host-based: shield.mcu.com >> svc-shield
    - Host-based: hydra.mcu.com >> svc-hydra
    - Path-based: mcu.com/shield >> svc-shield
    - Path-based: mcu.com/hydra >> svc-hydra
4. A cloud load balancer will be created and the ingress controller will monitor it for traffic
5. Configure DNS name resolution to point shield.mcu.com, hydra.mcu.com and mcu.com to the cloud load-balancer
6. A client will send traffic to shield.mcu.com DNS name resolution will send the traffic to the load-balancer's public endpoint
7. Ingress will read HTTP headers for the hostname resolution
8. Ingress rule will trigger and the traffic will be routed to the svc shield clusterIP backend
9. the clusterip service will ensure the traffic reaches the shield pod

```
kubectl delete ingress mcu-all
kubectl delete namespace ingress-nginx
kubectl delete clusterrole ingress-nginx
kubectl delete clusterroblebinding ingress-nginx

sudo vim /etc/hosts
```