**What is Kubernetes Ingress?**

Kubernetes **Ingress** is an API object that manages external access to services within a cluster. It provides **routing** based on hostnames and paths and is commonly used with an **Ingress Controller** (such as Nginx, Traefik, or HAProxy).

**Prerequisites for Ingress**

1. **Ensure an Ingress Controller is Installed**
   * If you are using Minikube, enable the Nginx Ingress Controller:

minikube addons enable ingress

* + For a Kubernetes cluster (cloud or on-prem), deploy Nginx:

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress->nginx/main/deploy/static/provider/cloud/deploy.yaml

1. **Verify the Ingress Controller is Running**

kubectl get pods -n kube-system

*(Look for ingress-nginx-controller in the output.)*

**1. Simple Ingress Example (Single Host)**

**Use Case: Route traffic to a single backend service.**

**Deployment (webapp-deployment.yaml)**

apiVersion: apps/v1

kind: Deployment

metadata:

name: webapp-deployment

spec:

replicas: 2

selector:

matchLabels:

app: webapp

template:

metadata:

labels:

app: webapp

spec:

containers:

- name: webapp

image: nginx

ports:

- containerPort: 80

**Service (webapp-service.yaml)**

apiVersion: v1

kind: Service

metadata:

name: webapp-service

spec:

selector:

app: webapp

ports:

- protocol: TCP

port: 80

targetPort: 80

type: ClusterIP

**Ingress (webapp-ingress.yaml)**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: webapp-ingress

spec:

rules:

- host: webapp.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: webapp-service

port:

number: 80

**Deploy and Test**

kubectl apply -f webapp-deployment.yaml

kubectl apply -f webapp-service.yaml

kubectl apply -f webapp-ingress.yaml

* Add this entry to /etc/hosts (Linux/macOS) or C:\Windows\System32\drivers\etc\hosts (Windows):

<IngressController-IP> webapp.example.com

*(Find <IngressController-IP> using kubectl get svc -n kube-system | grep ingress-nginx.)*

*Use localhost port 127.0.0.1*

*Start tunnel: minikube tunnel*

* Access in browser: **http://webapp.example.com**

**2. Ingress with Multiple Hosts**

**Use Case: Route different hostnames to different services.**

**Additional Deployment (admin-deployment.yaml)**

apiVersion: apps/v1

kind: Deployment

metadata:

name: admin-deployment

spec:

replicas: 2

selector:

matchLabels:

app: admin

template:

metadata:

labels:

app: admin

spec:

containers:

- name: admin

image: nginx

ports:

- containerPort: 80

**Additional Service (admin-service.yaml)**

apiVersion: v1

kind: Service

metadata:

name: admin-service

spec:

selector:

app: admin

ports:

- protocol: TCP

port: 80

targetPort: 80

type: ClusterIP

**Ingress for Multiple Hosts (multi-host-ingress.yaml)**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: multi-host-ingress

spec:

rules:

- host: webapp.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: webapp-service

port:

number: 80

- host: admin.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: admin-service

port:

number: 80

**Deploy and Test**

kubectl apply -f admin-deployment.yaml

kubectl apply -f admin-service.yaml

Note: delete the ingress created in the previous example

kubectl apply -f multi-host-ingress.yaml

* Add these entries to /etc/hosts or C:\Windows\System32\drivers\etc\hosts:

<IngressController-IP> webapp.example.com

<IngressController-IP> admin.example.com

*Use localhost port 127.0.0.1*

*Start tunnel: minikube tunnel*

* Access in browser:
  + **http://webapp.example.com** → Routes to webapp-service
  + **http://admin.example.com** → Routes to admin-service

**3. Ingress with Path-Based Routing**

**Use Case: Route different paths to different services.**

**Ingress with Path Routing (path-routing-ingress.yaml)**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: path-routing-ingress

spec:

rules:

- host: myapp.example.com

http:

paths:

- path: /webapp

pathType: Prefix

backend:

service:

name: webapp-service

port:

number: 80

- path: /admin

pathType: Prefix

backend:

service:

name: admin-service

port:

number: 80

**Deploy and Test**

kubectl apply -f path-routing-ingress.yaml

* Add to /etc/hosts or C:\Windows\System32\drivers\etc\hosts:

<IngressController-IP> myapp.example.com

* Access in browser:
  + **http://myapp.example.com/webapp** → Routes to webapp-service
  + **http://myapp.example.com/admin** → Routes to admin-service

**4. Ingress with TLS (HTTPS)**

**Use Case: Secure ingress with HTTPS.**

**TLS Secret**

kubectl create secret tls my-tls-secret --cert=cert.pem --key=key.pem

**Ingress with TLS (tls-ingress.yaml)**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: tls-ingress

spec:

tls:

- hosts:

- secure.example.com

secretName: my-tls-secret

rules:

- host: secure.example.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: webapp-service

port:

number: 80

**Deploy and Test**

kubectl apply -f tls-ingress.yaml

* Add to /etc/hosts:

<IngressController-IP> secure.example.com

* Access: **https://secure.example.com**

**Summary**

| **Scenario** | **Ingress Type** | **Example URL** |
| --- | --- | --- |
| **Basic Ingress** | Single Host | http://webapp.example.com |
| **Multiple Hosts** | Multi-Domain | http://admin.example.com |
| **Path-Based Routing** | Same Host, Different Paths | http://myapp.example.com/webapp |
| **TLS Ingress** | HTTPS with SSL | https://secure.example.com |