```
======================
Kubernetes Ingress
======================

-> Deploy two application Into K8S using Cluster IP Service


-------------------------------- java-web-app-deploy.yml-------------------------------------
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: javawebappdeployment
spec:
 replicas: 1
 strategy:
    type: Recreate
 selector:
   matchLabels:
     app: javawebapp
 template:
  metadata:
   name: javawebapppod
   labels:
     app: javawebapp
  spec:
    containers:
    - name: javawebappcontainer
      image: ashokit/javawebapp
      ports:
      - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
 name: javawebappsvc
spec:
  type: ClusterIP
  selector:
   app: javawebapp
  ports:
   - port: 80
     targetPort: 8080
...

---------------------------------------------------------maven-web-app-deploy.yml--------------------
---------------------------------------
apiVersion: apps/v1
kind: Deployment
metadata:
 name: mavenwebappdeployment
spec:
 replicas: 2
 selector:
   matchLabels:
     app: mavenwebapp
 template:
  metadata:
   name: mavenwebapppod
   labels:
     app: mavenwebapp
  spec:
    containers:
    - name: mavenwebappcontainer
```

```
      image: ashokit/mavenwebapp
      ports:
      - containerPort: 8080
      resources:
        requests:
         cpu: 200m
         memory: 1Gi
        limits:
         cpu: 500m
         memory: 2Gi
---
apiVersion: v1
kind: Service
metadata:
 name: mavenwebappsvc
spec:
  type: ClusterIP
  selector:
   app: mavenwebapp
  ports:
   - port: 80
     targetPort: 8080
----------------------------------------------------------------------------------------------------
----------------------------------------------

$ kubectl apply -f javawebapp.yml
$ kubectl apply -f mavenwebapp.yml
```

-> Now we have 2 services running in K8S cluster with LBR service.

-> We will use Ingress to provide routing for these two services from external traffic

-> K8S ingress is a resource to add rules for routing traffic from external sources to the services in the k8s cluster

-> K8S ingress is a native k8s resource where you can have rules to route traffic from an external source to service endpoints residing inside the cluster.

-> It requires an ingress controller for routing the rules specified in the ingress object

-> Ingress controller is typically a proxy service deployed in the cluster. It is nothing but a Kubernetes deployment exposed to a service.

############
Ingress Setup
############

```
# git clone k8s-ingress
$ git clone https://github.com/ashokitschool/kubernetes_ingress.git

$ cd kubernetes_ingress

# Create namespace and service-account
$ kubectl apply -f common/ns-and-sa.yaml

# create RBAC and configMap
$ kubectl apply -f common/

# Deploy Ingress controller
```

-> We have 2 options to deploy ingress controller

```
1) Deployment
2) DaemonSet

$ kubectl apply -f daemon-set/nginx-ingress.yaml

# Get ingress pods using namespace
$ kubectl get all -n nginx-ingress

# create LBR service

$ kubectl apply -f service/loadbalancer-aws-elb.yaml

Note: It will generate LBR DNS

*******************   Map LBR dns to route 53 domain  *************************

-> Create Ingress kind with rules

============================
Path Based Routing
============================

$ vi ingress-rules-routes.yml

---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-resource
spec:
  ingressClassName: nginx
  rules:
  - host: ashokit.org
    http:
      paths:
      - pathType: Prefix
        path: "/java-web-app"
        backend:
         service:
          name: javawebappsvc
          port:
           number: 80
      - pathType: Prefix
        path: "/maven-web-app"
        backend:
         service:
          name: mavenwebappsvc
          port:
           number: 80
...

#####################################
Access the application using below URL
####################################

URL-1 :  www.ashokit.org/java-web-app

URL-2 : www.ashokit.org/maven-web-app
```