#Day -11

Complete Info About "Ingress"

Ingress:-

- It is a API object that manages external access to the service in the cluster typically HTTP & HTTPS.

It may provides

1) Load balancing
2) SSL Termination
3) Name based Virtual hosting

Q. When we have service to access the cluster/App. What is the need of this Ingress?

Solution:- Question is valid because when we have Load balancer service is there for external access, why we need this ingress? So here are the few reason why we need Ingress.

1) Services Lacking the commercial Enterprise level load balancers organizations such as F5 Load balancer NGINX .... These conventral load balancer are providing a lot of commercial features such as host based, path based, ratio based, WAF etc.

2) Lets assume we have 50 Sevices in an application so for each Sevice of type load balancer should have load balancer this means 50 Services needs 50 load balancer practically this will make more cost to client from csps.

Intial Version of K8's doesn't have the concept of "Ingress". The importance of this Ingress Came after people adopted K8's for production.

Openshift (based kubernetes) is developed by Redhat (IBM) is the first one to intreduce engress kind of feature i,e "Routes"

Later then K8's admitted the importance of the usage of commercial Load balancer infront of cluster.

At that point K8's intreduced Ingress to cluster, which is a resource of kubernetes cluster we can create.

Ingress controller :-

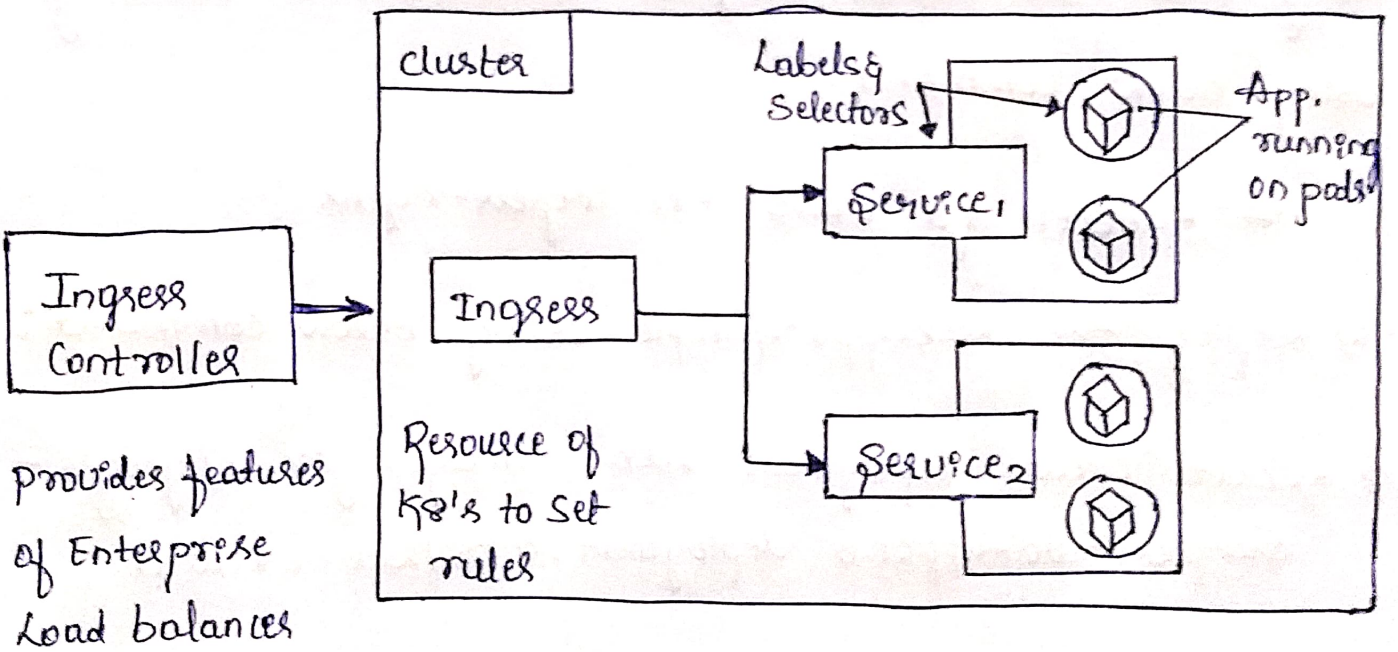
It is that wheree we can manage all the routing policies like a Commercial load balancer. These are developed by the respective organization to be used for kubernetes cluster.

NGINX Ingress Controller ⇒ Developed by NGINX

F5 Ingress Controller → Developed by F5.

Similarly remaining will also develop their Ingress controllers to be a part of kubernetes cluster.

For more understanding observe the flow.



→ Path based Routing Examples:

/Login → Service 1

/ Dashboard → Service 2

/ checkout → Service 3

/ Feed → Service 4

Now it's time to do handson this Ingress. I am using minikube - cluster. The process might be depends on type of platform your using.

In kubeendes documentation, Browse to "Setup Ingress on minikube with NGINX Ingress controller".

1) Enable Ingress - controller (NGINX here)

```
# minikube addon enable ingress
```

Verify that the NGINX Ingress controller is running using below command.

```
# kubectl get pods -n ingress-nginx
```

2) Deploy the webapp1-sample using below command:

```
$ kubectl create deployment web --image=gcr.io/google-
  samples/hello-app:1.0 deployment.apps/web created.

$ kubectl expose deployment web --type=Nodeport --port=8080
  service/web exposed

$ minikube service web --url
  http://192.168.59.109:32403
```

3) Create Ingress resource using below file except the /v2 → Lines (which will be added after creating web2 svc & deploy)

$ cat example-ingress.yml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations: nginx.ingress.kubernetes.io/rewrite-target:/$1
spec:
  rules:
    - host: test.domain
      http:
        paths:
          - path: /
            pathType: prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
          - path: /v2
            pathType: prefix
            backend:
              service:
                name: web2
                port:
                  number: 8080
```

4) Now verify the application endpoint using below command: (version 1.0.0)

$ curl --resolve "test.domain:80: $(minikube ip)" -i http://test.domain

5) Similarly create deployment & service for web2 as follows:-
(Here we will add /v2 lines in ingress file)

$ kubectl create deployment web2 --image=gcr.io/google-samples/hello-app:2.0 deployment.apps/web2 created

$ kubectl get deploy

$ kubectl expose deployment web2 --port=8080 --type=NodePort
service/web2 exposed

$ kubectl apply -f example-ingress.yml
ingress.networking.k8s.io/example-ingress configured

6) Now, Verify that web2 app is accessible at /v2 endpoint ending (version 2.0.0)

$ curl --resolve "test.domain:80:$(minikube ip)" -i
http://test.domain/v2

In a Nutshell:-