Kubernetes Ingress

Ingress is used to expose HTTP and HTTPS routes to external traffic from outside the Kubernetes cluster. It supports the following key features:

- **Path-based routing: Routes traffic to specific services based on the URL path (e.g., example.com/app1).**
- **Host-based routing: Routes traffic based on the hostname or subdomain (e.g., app.example.com).**
- **Load balancing: Distributes traffic among multiple pods of a service.**
- **SSL termination: Provides secure communication by terminating SSL/TLS.**

Ingress redirects incoming requests to the appropriate services within the cluster based on the web URL or path. It also provides encryption features and helps balance application load effectively.

Why Use Ingress Instead of a Load Balancer?

While a load balancer can manage external traffic to services, Ingress offers more advanced features:

1. **Fine-grained access control: Ingress supports host-based and path-based routing, allowing you to route traffic based on the requested URL or hostname.**
2. **SSL termination: Ingress can handle secure HTTPS communication, which is not inherently managed by a standard load balancer.**
3. **Single entry point: Ingress consolidates multiple services under a single IP address, whereas a load balancer typically requires one IP address per service.**

A load balancer is useful for managing traffic at a higher level but lacks the fine-grained routing capabilities of Ingress.

Example Scenario

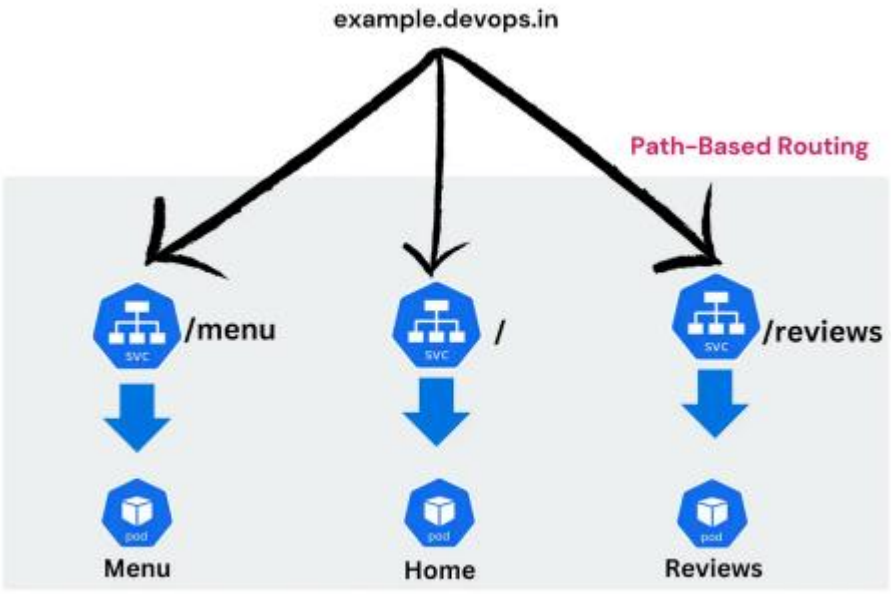
Imagine you have multiple services running in your Kubernetes cluster, with each serving a different application:

- **example.com/app1 for one application**
- **example.com/app2 for another**

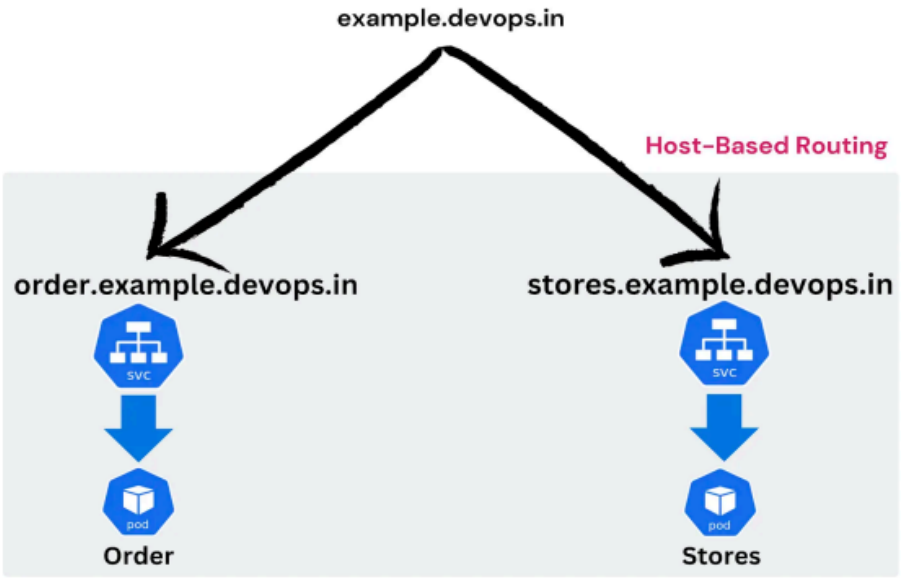
Using Ingress, you can configure routing rules to direct traffic to the correct service based on the requested path or hostname. In contrast, a load balancer can only route traffic based on ports and cannot handle URL-based routing.

Types of Routing in Ingress

1. **Path-based routing**:
   Directs traffic to different services based on the URL path.



2. **Host-based routing**:
   Directs traffic to different services based on the hostname or subdomain.



Steps to Install and Use Ingress in Kubernetes :

1. Install the NGINX Ingress Controller

To install the NGINX Ingress Controller, use the following command:

kubectl create -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.2.1/deploy/static/provider/cloud/deploy.yaml

2. Deploy the Applications

You need to deploy two applications for testing Ingress. The application manifests can be found in the following GitHub repository:
GitHub URL:
https://github.com/mustafaprojectsindevops/kubernetes/tree/master/ingress

3. Apply the Manifests

Execute all the files from the repository to deploy the necessary resources. Once the applications are deployed, use the following command to verify the Ingress resource:

kubectl get ing

4. Retrieve the Load Balancer DNS

After approximately 30 seconds, a load balancer DNS will be provided by the ingress controller. Use this DNS to access the deployed applications.

5. Access the Applications

Access the applications using the DNS provided by the ingress resource, appending the appropriate path for each application:

- **DNS/nginx**: Access the NGINX application.
- **DNS/httpd**: Access the HTTPD application.

The traffic will be routed to the respective applications based on the routing rules defined in the Ingress resource.

Example YAML File (ONE.YML)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: one
spec:
  replicas: 2
  selector:
    matchLabels:
      app: swiggy
  template:
    metadata:
      labels:
        app: swiggy
    spec:
      containers:
      - name: cont-1
        image: nginx
        ports:
        - containerPort: 80
        env:
        - name: TITLE
          value: "NGINX APP1"
```

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  type: NodePort
  ports:
  - port: 80
  selector:
    app: swiggy
```

```
[root@ip-172-31-89-240 ~]# git clone https://github.com/mustafaprojectsindevops/kubernetes.git
Cloning into 'kubernetes'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 22 (delta 8), reused 3 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (22/22), 6.27 KiB | 6.27 MiB/s, done.
Resolving deltas: 100% (8/8), done.
[root@ip-172-31-89-240 ~]# ll
total 65940
drwxr-xr-x 3 root root       78 Dec 13 19:24 aws
-rw-r--r-- 1 root root 67517785 Dec 16 05:08 awscliv2.zip
drwxr-xr-x 4 root root       33 Dec 16 07:02 kubernetes
-rw-r--r-- 1 root root      401 Dec 16 06:27 pod.yml
[root@ip-172-31-89-240 ~]# cd kubernetes
[root@ip-172-31-89-240 kubernetes]# ll
total 0
drwxr-xr-x 2 root root 55 Dec 16 07:02 ingress
[root@ip-172-31-89-240 kubernetes]# cd ingress
[root@ip-172-31-89-240 ingress]# vim one.yml
[root@ip-172-31-89-240 ingress]# vim two.yml
[root@ip-172-31-89-240 ingress]# vim ingress.yml
[root@ip-172-31-89-240 ingress]# kubectl create -f ingress.yml
ingress.networking.k8s.io/k8s-ingress created
[root@ip-172-31-89-240 ingress]# kubectl create -f one.yml
```

Example YAML File (Two.YML)

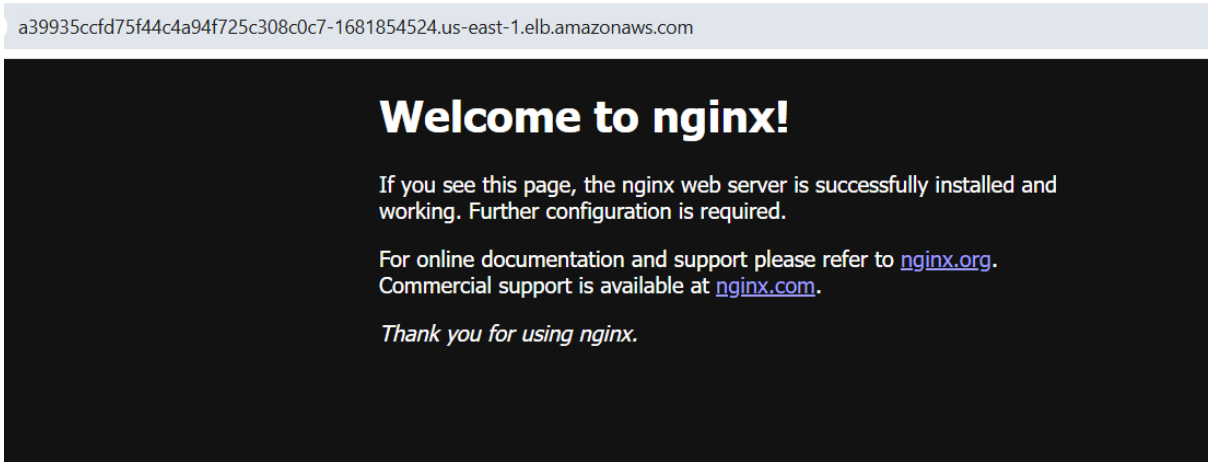
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: two
spec:
  replicas: 2
  selector:
    matchLabels:
      app: zomato
  template:
    metadata:
      labels:
        app: zomato
    spec:
      containers:
      - name: cont-2
        image: httpd
        ports:
        - containerPort: 80
        env:
        - name: TITLE
          value: "APACHE APP2"
```

```yaml
---
apiVersion: v1
kind: Service
metadata:
   name: httpd
spec:
   type: NodePort
   ports:
   - port: 80
   selector:
      app: zomato
```

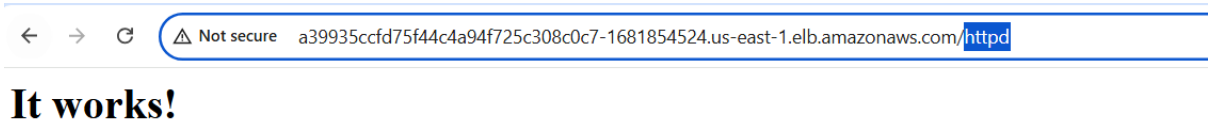
Creating one.yml, two.yml and ingress.yml

```
[root@ip-172-31-89-240 ingress]# vim one.yml
[root@ip-172-31-89-240 ingress]# vim two.yml
[root@ip-172-31-89-240 ingress]# vim ingress.yml
[root@ip-172-31-89-240 ingress]# kubectl create -f ingress.yml
ingress.networking.k8s.io/k8s-ingress created
[root@ip-172-31-89-240 ingress]# kubectl create -f one.yml
deployment.apps/one created
service/nginx created
[root@ip-172-31-89-240 ingress]# kubectl create -f two.yml
deployment.apps/two created
service/httpd created
[root@ip-172-31-89-240 ingress]# kubectl get ingress
NAME          CLASS   HOSTS   ADDRESS                                                                   PORTS   AGE
k8s-ingress   nginx   *       a39935ccfd75f44c4a94f725c308c0c7-1681854524.us-east-1.elb.amazonaws.com   80      84s
[root@ip-172-31-89-240 ingress]#
```

Here, we are accessing the NGINX server interface through the LoadBalancer's DNS.



Here, we are accessing the applications using the LoadBalancer, but the key difference from the previous setup is that now we explicitly add the path /httpd to the end of the LoadBalancer URL. This ensures it directly opens the HTTPD page. Previously, when we used only the LoadBalancer URL without a specific path, it would default to opening the NGINX default page.



It works!

Example of an Ingress YAML file (ingress.yml):

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: k8s-ingress
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/use-regex: "true"
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - path: /nginx(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: nginx
            port:
              number: 80
      - path: /httpd(/|$)(.*)
        pathType: Prefix
        backend:
          service:
```