Ingress

Ingress is a Kubernetes object that manages external access (usually HTTP/HTTPS) to services in your cluster.

It routes traffic from outside the cluster (like the internet) to services inside the cluster, based on:

- Hostnames (like app.example.com)
- Paths (like /api or /login)
- Or both

Why do we need Ingress

Let's say you have multiple services:

- frontend-service
- backend-service
- admin-service

If you want users to access these from the internet, you have two options:

Option 1: Expose each service using LoadBalancer or NodePort

- Every service needs its own public IP or port.
- Not scalable, expensive, messy.

Option 2: Use Ingress

- Create one LoadBalancer (Ingress controller)
- Route traffic based on domain/path
- Clean, cheap, scalable

How ingress works

Component	Description
Ingress Controller	The actual software that handles routing (e.g., `nginx-ingress`, `traefik`, etc.)
Ingress Resource	The YAML configuration that defines the routing rules
Service	Your backend services (e.g., frontend, backend, etc.) that the Ingress routes to

```
Internet
   |
   ▼
[ Ingress Controller ]   ← Acts like a smart router (usually NGINX)
   |
   ├── / → frontend-service
   ├── /api → backend-service
   └── /admin → admin-service
```

Ingress example

Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
    spec:
      containers:
      - name: hello
        image: hashicorp/http-echo
        args:
        - "-text=Hello from Ingress"
        ports:
        - containerPort: 5678
```

Service

```
apiVersion: v1
kind: Service
metadata:
  name: hello-service
spec:
  selector:
    app: hello
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5678
```

Ingress Resource

```
# Define the kind of Kubernetes object: Ingress
apiVersion: networking.k8s.io/v1
kind: Ingress

# Metadata for naming and adding annotations
metadata:
```

```
    name: hello-ingress # Name of the Ingress
    annotations:
      nginx.ingress.kubernetes.io/rewrite-target: /  #Rewrite incoming path to `/`

  # Ingress specification: defines routing rules
  spec:
    ingressClassName: nginx
    rules:
      - host: hello.example.com  # Domain to listen on
        http:
          paths:
          - path: /  # Incoming path
            pathType: Prefix  # Match all paths starting with `/`
            backend:
              service:
                name: hello-service  # Name of the Kubernetes service to route to
                port:
                  number: 80  # Port that the service is exposing
```

- You deploy your app and expose it using a Service
- You create an Ingress Resource to define rules:
  - Host: hello.example.com
  - Path: / → go to hello-service
- The Ingress Controller picks this up and routes external traffic accordingly.

Think of Ingress like the reception desk at an office:

- One public entry point (Ingress)
- You tell the receptionist (host/path), and you're directed to the right department (Service)

NOTE:-

- Ingress itself does not expose anything until an Ingress Controller is installed
- NGINX is the most commonly used Ingress Controller
- You need to update your DNS or use /etc/hosts for testing

Feature	Ingress
Purpose	Route external HTTP/HTTPS traffic to services
Acts Like	A smart router (based on domain/path)
Benefits	One entry point, DNS-based routing, TLS support
Needs	Ingress Controller (e.g., NGINX) installed
Cost	One LoadBalancer for many services

commands related to Ingress

```
# List all ingresses in the current namespace
kubectl get ingress
```

```
# List ingresses in a specific namespace
kubectl get ingress -n <namespace>

# Describe a specific ingress to see rules and events
kubectl describe ingress <ingress-name>

# Get full YAML of an ingress
kubectl get ingress <ingress-name> -o yaml

# Apply an Ingress manifest from a file
kubectl apply -f ingress.yaml

# Create an Ingress directly from command (basic example)
kubectl create ingress my-ingress \
  --rule="myapp.example.com/*=myapp-service:80"

# Delete an ingress
kubectl delete ingress <ingress-name>

# Check if the NGINX ingress controller is running (usually in kube-system or
ingress-nginx namespace)
kubectl get pods -n ingress-nginx

# Get external IP of the ingress controller (to test access)
kubectl get service ingress-nginx-controller -n ingress-nginx

# Curl the Ingress endpoint (after DNS/host setup)
curl http://myapp.example.com
```