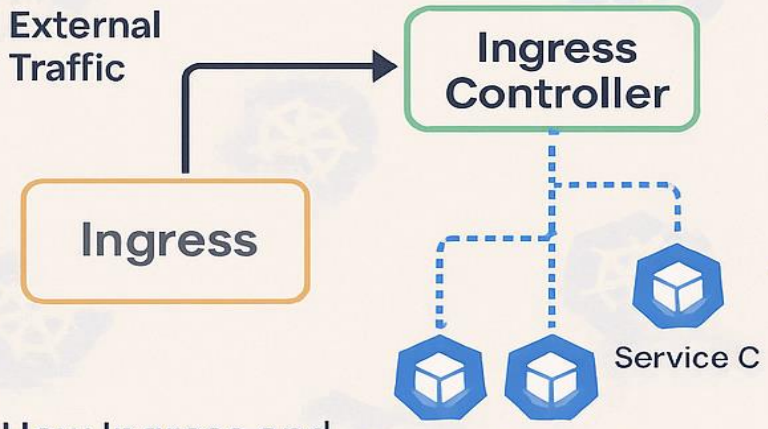
Kubernetes
INGRESS & INGRESS CONTROLLER

Ingress and Ingress Controller

Ingress is an API object that manages external access to services within a cluster, typically HTTPS traffic.

Ingress Controller is the component responsible for reading and processing Ingress resource information.

How Ingress and Ingress Controllers Work



How Ingress and Ingress Controllers Work

- Configures traffic to services based on defined rules
- The Ingress Controller watches for Ingress resource changes
- It configures the ingress based on these changes

Types of Ingress Controllers

- NGINX Ingress Controller
- HAProxy Ingress Controller
- Traefik Ingress Controller
- AWS Load Balancer Controller

Configurable Routing – Routes requests based on host, path, etc.

SSL/TLS Termination Offloads SSL/TLS encryption to the Ingress Controller

Virtual Hosts

✅ What is Kubernetes Ingress?

Kubernetes **Ingress** is an API object that **manages external access to services within a cluster**, typically HTTP/HTTPS.
It acts as an **entry point (Layer 7 – application layer)** for routing external traffic to internal services based on **rules like path, host, or protocol**.

Ingress = **smart router** that handles external traffic and forwards it to the appropriate service in the cluster.

✅ What is an Ingress Controller?

An **Ingress Controller** is the **actual implementation** of the Ingress. It listens for **Ingress resources** and configures the underlying load balancer (like NGINX, HAProxy, Traefik, etc.) to handle traffic based on Ingress rules.

Ingress is just a configuration; the Ingress Controller makes it **work**.

How Ingress and Ingress Controller Work

Step-by-Step Flow:

Let's break down the **Ingress flow**:

1. External Client Sends Request

A user sends a request like:
https://app.example.com/login

2. DNS Resolves the Domain

The domain app.example.com is mapped (via DNS) to the public IP of the **Ingress Controller** (e.g., LoadBalancer service or NodePort).

3. Ingress Controller Receives the Request

The Ingress Controller (like NGINX) listens on port 80/443 and intercepts the request.

4. Ingress Resource Rules Are Evaluated

The Ingress Controller checks the **Ingress resource** rules (YAML) to determine:

- Which service to forward the request to

- Based on the path or host (e.g., /login → auth-service)

5. Traffic is Routed to the Correct Kubernetes Service

The Controller forwards the request to the corresponding **Kubernetes Service**, which then sends it to a **Pod**.

6. Response is Returned

The application inside the Pod processes the request, and the response flows back via the same route.

✅ Why Use Ingress?

Without Ingress, exposing multiple services to the outside world means:

Creating a **NodePort** or **LoadBalancer** for each service (expensive & hard to manage)

No path-based or host-based routing

Ingress solves this by:

- Centralizing external access control

- Allowing multiple services behind a single IP or domain

- Supporting advanced routing, TLS termination, etc.

✅ When to Use Ingress?

Use Ingress when:

You need **host- or path-based routing** (e.g., /api, /admin)

You want to **consolidate external access** to multiple services

You want **TLS termination** at a single point (centralized HTTPS)

You need **custom routing rules, rate limiting, authentication, etc.**

Advantages of Ingress

Advantage	Description
Centralized Access	Route all traffic via a single point of entry
Path & Host Routing	Route /api to one service, /app to another
TLS Termination	Manage HTTPS certs in one place
Cost Efficient	Only 1 LoadBalancer needed instead of 1 per service
Scalable	Easily manage and scale multiple microservices
Declarative	Managed using YAML and GitOps principles

Disadvantages of Ingress

Disadvantage	Description
Setup Complexity	Requires configuring Ingress Controllers
Learning Curve	Complex for beginners (especially with TLS, custom rules)
Not Good for All Protocols	Designed for HTTP/HTTPS; not ideal for TCP/UDP

Types of Ingress Controllers

Ingress Controller	Description
NGINX Ingress Controller	Most widely used; stable, flexible
Traefik	Dynamic routing, metrics, simpler config
HAProxy	High-performance proxy; advanced features
Istio Gateway	Used in service mesh environments
AWS ALB Ingress Controller	AWS-native; integrates with ALB
GCE Ingress Controller	For GKE (Google Kubernetes Engine)
Contour	Lightweight and fast with Envoy proxy backend

Sample Ingress YAML

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: app.example.com
    http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          service:
            name: auth-service
            port:
              number: 80
      - path: /dashboard
        pathType: Prefix
        backend:
          service:
```

```
                name: dashboard-service
                port:
                  number: 80
```

Real-World Example Scenario

Problem (Without Ingress):

Dev team has:

- auth-service on /auth
- payment-service on /pay
- admin-service on /admin

Each service needs a LoadBalancer → Costly

SSL needs to be configured individually → Maintenance overhead

Solution (With Ingress):

Single **Ingress Controller** handles:

- Routing all paths/domains
- Central SSL termination

One external IP → /auth, /pay, /admin all routed internally
