Kubernetes (K8s) does not have a built-in Layer 4 or Layer 7 load balancer implementation. Instead, it provides service types like ClusterIP, NodePort, and LoadBalancer, which rely on cloud provider integrations for external traffic handling. To achieve Layer 7 (HTTP/S) load balancing, Ingress is used. Ingress provides advanced traffic routing, SSL termination, and host/path-based routing using an Ingress Controller.

**Ingress Networking in Kubernetes**

**1. Introduction**

Ingress networking in Kubernetes is a way to manage external access to services within a cluster. It allows HTTP and HTTPS traffic routing using an Ingress resource and an Ingress Controller. Unlike traditional LoadBalancers and NodePorts, Ingress provides more control over traffic routing, SSL termination, and host-based or path-based routing.

**2. Components of Ingress Networking**

**2.1 Ingress Resource**

An Ingress is a Kubernetes API object that defines how external HTTP/S traffic is routed to services inside the cluster. It includes:

* Host-based routing (e.g., app.example.com)
* Path-based routing (e.g., /api, /dashboard)
* TLS termination (for HTTPS)

**Example of an Ingress resource:**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: my-app-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: myapp.example.com

http:

paths:

- path: /app1

pathType: Prefix

backend:

service:

name: app1-service

port:

number: 80

- path: /app2

pathType: Prefix

backend:

service:

name: app2-service

port:

number: 80

tls:

- hosts:

- myapp.example.com

secretName: myapp-tls

**2.2 Ingress Controller**

An Ingress Controller is a pod or service that processes Ingress resources and manages traffic routing. Kubernetes does not provide an Ingress Controller by default. You must deploy one, such as:

* NGINX Ingress Controller
* Traefik
* HAProxy
* AWS ALB Ingress Controller (for AWS EKS)

**Example: Deploying NGINX Ingress Controller using Helm**

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

helm install my-ingress ingress-nginx/ingress-nginx

**3. Working of Ingress Networking**

1. **User Requests** → User sends an HTTP/S request to a domain.
2. **Ingress Controller** → The request is intercepted by the Ingress Controller.
3. **Ingress Rules** → The controller applies routing rules defined in the Ingress resource.
4. **Service and Pod Routing** → The request is forwarded to the correct Kubernetes service and underlying pod.

**4. Advantages of Using Ingress**

* **Consolidated Traffic Management**: Instead of exposing multiple services, a single entry point handles external requests.
* **Load Balancing**: Distributes incoming requests among backend pods.
* **TLS Termination**: Manages SSL certificates using Kubernetes secrets.
* **Path-Based and Host-Based Routing**: Routes traffic based on URL paths and hostnames.
* **Cost-Effective**: Reduces the need for multiple LoadBalancer services.

**5. Troubleshooting Common Ingress Issues**

**5.1 Verify Ingress Resource**

kubectl get ingress -n my-namespace

kubectl describe ingress my-app-ingress -n my-namespace

**5.2 Check Ingress Controller Logs**

kubectl logs -l app.kubernetes.io/name=ingress-nginx -n ingress-nginx

**5.3 Ensure DNS Resolution**

Check if the domain resolves to the Ingress Controller's external IP:

nslookup myapp.example.com

**5.4 Debugging with Curl**

curl -v https://myapp.example.com/app1

**6. Conclusion**

Ingress networking is an essential part of Kubernetes, providing efficient, cost-effective, and secure traffic management for services. Choosing the right Ingress Controller and configuring proper routing rules ensures smooth external access to Kubernetes applications.