Ingress

**What Is Ingress in Kubernetes?**

**Ingress** is a Kubernetes API object that manages **external HTTP/HTTPS access** to services inside your cluster. Think of it as a smart router or reverse proxy that:

* Routes traffic based on **hostnames** or **URL paths**
* Handles **SSL/TLS termination**
* Provides **centralized access control**
* Reduces the need for multiple LoadBalancer or NodePort services

But here’s the catch: an Ingress resource **does nothing on its own** — it needs an **Ingress Controller** (like NGINX, Traefik, or HAProxy) to actually enforce the routing rules.

**How It Fits in Your Project**

You currently expose your frontend, backend, and mysql services — probably using NodePort or LoadBalancer. With Ingress, you can:

| **Route** | **Service** |
| --- | --- |
| yourdomain.com/ → frontend-service |  |
| yourdomain.com/api → backend-service |  |

This gives you a **single external IP** and clean URLs — no more port juggling.

**How to Implement Ingress (Step-by-Step)**

**1. Install an Ingress Controller (NGINX example)**

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.9.4/deploy/static/provider/cloud/deploy.yaml

Wait for the controller pod to be ready:

kubectl get pods -n ingress-nginx

**2. Create an Ingress Resource**

*apiVersion: networking.k8s.io/v1*

*kind: Ingress*

*metadata:*

*name: app-ingress*

*annotations:*

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

*spec:*

*ingressClassName: nginx*

*rules:*

*- http:*

*paths:*

*- path: /*

*pathType: Prefix*

*backend:*

*service:*

*name: frontend-service*

*port:*

*number: 80*

*- path: /api*

*pathType: Prefix*

*backend:*

*service:*

*name: springboot-service*

*port:*

*number: 80*

*Apply it:*

*kubectl apply -f ingress.yml*

**3. Access Your App**

Get the external IP:

kubectl get svc -n ingress-nginx

Then visit:

* http://<external-ip>/ → frontend
* http://<external-ip>/api → backend

If the *kubectl apply -f ingress.yml* doesn’t work and get the below error

*ubuntu@Master:~/k8s$ kubectl apply -f ingress.yml*

*Error from server (InternalError): error when creating "ingress.yml": Internal error occurred: failed calling webhook "validate.nginx.ingress.kubernetes.io": failed to call webhook: Post "https://ingress-nginx-controller-admission.ingress-nginx.svc:443/networking/v1/ingresses?timeout=10s": context deadline exceeded*

It is probably due to Nginx Webhook Validation

To bypass the webhook validation

*kubectl delete ValidatingWebhookConfiguration ingress-nginx-admission*

*kubectl apply -f ingress.yml*

If the external port mapping is in pending state as below

*ubuntu@Master:~/k8s$ kubectl get svc -n ingress-nginx*

*NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE*

*ingress-nginx-controller LoadBalancer 10.107.173.190 <pending> 80:31364/TCP,443:32732/TCP 40m*

*ingress-nginx-controller-admission ClusterIP 10.106.133.201 <none> 443/TCP 40m*

And then access it via

A NodePort Instead (Works Immediately)

Patch the service to make it reachable via node IP and port:

*kubectl patch svc ingress-nginx-controller -n ingress-nginx \ -p '{"spec": {"type": "NodePort"}}'*

***Ingress in Kubernetes***

***What Is Ingress?***

*Ingress is a Kubernetes API object that manages* ***external HTTP/HTTPS access*** *to services inside your cluster. It functions as a reverse proxy and layer 7 load balancer. Key features:*

* *Routes traffic based on* ***hostnames or URL paths***
* *Handles* ***SSL/TLS termination***
* *Enables* ***centralized access control***
* *Reduces the need for multiple LoadBalancer or NodePort services*

*An Ingress object alone does nothing—it requires an* ***Ingress Controller*** *(e.g., NGINX, Traefik, HAProxy) to process and enforce its rules.*

***How It Fits in Your Project***

*Your services: | Route | Service | |--------------------|----------------------| | / | frontend-service | | /api | springboot-service |*

*Using Ingress, you expose both services through* ***one external IP*** *and route traffic via clean, path-based URLs. No separate NodePorts needed.*

***Step-by-Step Implementation***

***1. Install NGINX Ingress Controller***

*kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.9.4/deploy/static/provider/cloud/deploy.yaml*

*Verify the controller is ready:*

*kubectl get pods -n ingress-nginx*

***2. Create an Ingress Resource***

*Save the following YAML as ingress.yml:*

*apiVersion: networking.k8s.io/v1*

*kind: Ingress*

*metadata:*

*name: app-ingress*

*annotations:*

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

*spec:*

*ingressClassName: nginx*

*rules:*

*- http:*

*paths:*

*- path: /*

*pathType: Prefix*

*backend:*

*service:*

*name: frontend-service*

*port:*

*number: 80*

*- path: /api*

*pathType: Prefix*

*backend:*

*service:*

*name: springboot-service*

*port:*

*number: 80*

*Apply the resource:*

*kubectl apply -f ingress.yml*

***Error and Resolution: Webhook Timeout***

***Error Encountered***

*Error from server (InternalError): error when creating "ingress.yml":*

*Internal error occurred: failed calling webhook "validate.nginx.ingress.kubernetes.io":*

*failed to call webhook: Post "https://ingress-nginx-controller-admission.ingress-nginx.svc:443/networking/v1/ingresses?timeout=10s": context deadline exceeded*

***Cause***

*This error occurs when the* ***Ingress webhook validation service is unreachable****—often due to:*

* *The secret ingress-nginx-admission missing or improperly mounted*
* *TLS configuration issues*
* *Cold start delay during controller deployment*

***Resolution Steps***

1. *Check if the admission secret exists:*

*kubectl describe secret ingress-nginx-admission -n ingress-nginx*

1. *If the secret exists and you still see the error, bypass webhook validation:*

*kubectl delete ValidatingWebhookConfiguration ingress-nginx-admission*

*kubectl apply -f ingress.yml*

*This disables admission webhook validation temporarily for Ingress resources. Use with caution.*

***External IP Pending: Fix With NodePort***

***Issue***

*Service remains in pending state:*

*kubectl get svc -n ingress-nginx*

*Output:*

*NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE*

*ingress-nginx-controller LoadBalancer 10.107.173.190 <pending> 80:31364/TCP,443:32732/TCP*

*This means your cloud provider hasn't provisioned an external IP (common with manual or on-prem clusters).*

***Solution***

*Change the service to NodePort so you can access via EC2 public IP:*

*kubectl patch svc ingress-nginx-controller -n ingress-nginx \*

*-p '{"spec": {"type": "NodePort"}}'*

*Then retrieve the NodePort:*

*kubectl get svc ingress-nginx-controller -n ingress-nginx*

*Access your app via:*

*http://<ec2-public-ip>:<node-port>/*

*http://<ec2-public-ip>:<node-port>/api*

*Open these NodePorts in the EC2 security group if needed.*

*Let me know if you'd like to expand this guide to include TLS with cert-manager, domain mapping (e.g. app.quantumsoft.io), or health checks. This setup is now ready for scaling and public access.*