## ****🔥 Summary of Steps****

1. **Installed and Configured** kind
   * Faced an issue with kind installation (exec format error on Mac). (brew install kind)
   * Resolved by ensuring kind was properly installed for the correct architecture.
2. **Deployed Ingress Controller**
   * Initially, the Ingress controller pod was stuck in Pending due to **node selector issues**.
   * Fixed it by ensuring the node selector matches the Kind cluster configuration.
3. **Created and Applied Kubernetes Resources**
   * **Deployments & Services for** app1 **and** app2
   * **Ingress for path-based routing** (/app1 → app1-service, /app2 → app2-service).
   * Everything worked fine after applying kubectl apply -f on YAML files.

## ****✅ Graceful Shutdown Steps****

### ****1️⃣ Stop Running Applications & Ingress****

kubectl delete ingress path-based-ingress

kubectl delete service app1-service app2-service

kubectl delete deployment app1-deployment app2-deployment

### ****2️⃣ Stop Kind Cluster****

sh

kind delete cluster

This will delete the entire Kind cluster along with everything running inside it.

## ****🚀 Restarting Everything****

### ****1️⃣ Start Kind Cluster****

sh

-----

kind create cluster

### ****2️⃣ Deploy**** app1 ****and**** app2

Apply the deployments and services:

sh

-----

kubectl apply -f deployments.yaml

### ****3️⃣ Deploy Ingress****

sh

-----

kubectl apply -f ingress.yaml

### ****4️⃣ Verify Everything is Running****

sh

-----

kubectl get pods

kubectl get svc

kubectl get ingress

### ****5️⃣ Test with Port Forwarding (if needed)****

sh

-----

kubectl port-forward --namespace ingress-nginx service/ingress-nginx-controller 8080:80

Now open:

* **http://localhost:8080/app1** (should show Nginx page)
* **http://localhost:8080/app2** (should show Apache page)

### ****🎯 Quick Commands for Reference****

| Action | Command |
| --- | --- |
| Stop all apps & services | kubectl delete -f deployments.yaml |
| Stop Kind cluster | kind delete cluster |
| Start Kind cluster | kind create cluster |
| Restart everything | kubectl apply -f deployments.yaml && kubectl apply -f ingress.yaml |
| Check resources | kubectl get pods,svc,ingress |

Now you have a fully documented **start → stop → restart** process! 🚀 Let me know if you need any refinements.

Detailed

# ****Kubernetes Ingress Setup with Kind on Mac****

This guide covers everything from **installing Kind** to **deploying a demo** with **Ingress-based routing**.

## ****🔹 1. Install Pre-requisites****

Ensure you have the following installed:

### ****📌 Install Homebrew (if not installed)****

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

### ****📌 Install Kind****

sh

brew install kind

### ****📌 Install kubectl****

sh

brew install kubectl

### ****📌 Verify Installation****

sh

kind version

kubectl version --client

✅ If installed correctly, you will see version details.

## ****🔹 2. Create a Kind Cluster****

Run the following command to create a Kind cluster with Ingress enabled:

kind create cluster --config=kind-config.yaml

-----

cat <<EOF | kind create cluster --config=-

kind: Cluster

apiVersion: kind.x-k8s.io/v1alpha4

nodes:

- role: control-plane

extraPortMappings:

- containerPort: 80

hostPort: 80

- containerPort: 443

hostPort: 443

EOF

### ****📌 Verify Cluster****

sh

-----

kubectl cluster-info

kubectl get nodes

✅ The cluster should be **Running**.

## ****🔹 3. Install Ingress Controller****

Run:

sh

-----

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml

correct the node label:

kubectl label nodes $(kubectl get nodes -o=jsonpath='{.items[0].metadata.name}') ingress-ready=true

### ****📌 Wait for Ingress to be Ready****

sh

-----

kubectl wait --namespace ingress-nginx \

--for=condition=ready pod \

--selector=app.kubernetes.io/component=controller \

--timeout=90s

### ****📌 Verify****

kubectl get pods -n ingress-nginx

✅ The Ingress controller should be **Running**.

## ****🔹 4. Deploy App1 & App2****

### ****📌 Create a Deployment File (****deployments.yaml****)****

Create deployments.yaml and add the following:

yaml

-----

apiVersion: apps/v1

kind: Deployment

metadata:

name: app1

spec:

replicas: 1

selector:

matchLabels:

app: app1

template:

metadata:

labels:

app: app1

spec:

containers:

- name: app1

image: nginx

ports:

- containerPort: 80

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: app2

spec:

replicas: 1

selector:

matchLabels:

app: app2

template:

metadata:

labels:

app: app2

spec:

containers:

- name: app2

image: httpd

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: app1-service

spec:

selector:

app: app1

ports:

- protocol: TCP

port: 80

targetPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: app2-service

spec:

selector:

app: app2

ports:

- protocol: TCP

port: 80

targetPort: 80

### ****📌 Apply Deployments & Services****

sh

-----

kubectl apply -f deployments.yaml

### ****📌 Verify****

sh

-----

kubectl get pods

kubectl get svc

✅ Both **App1** and **App2** should be **Running**.

## ****🔹 5. Configure Ingress for Path-Based Routing****

### ****📌 Create an Ingress File (****ingress.yaml****)****

Create ingress.yaml and add:

yaml

-----

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: path-based-ingress

annotations:

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

spec:

ingressClassName: nginx

rules:

- host: localhost

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

### ****📌 Apply Ingress****

sh

-----

kubectl apply -f ingress.yaml

### ****📌 Verify****

sh

-----

kubectl get ingress

✅ The Ingress should be **configured**.

## ****🔹 6. Test the Setup****

### ****📌 Access Applications in Browser****

* **App1:** 👉 <http://localhost/app1>
* **App2:** 👉 http://localhost/app2

✅ You should see the **Nginx** page for /app1 and **Apache** page for /app2.

## ****🔹 7. Graceful Shutdown****

### ****1️⃣ Delete Ingress, Services, and Deployments****

sh

-----

kubectl delete -f ingress.yaml

kubectl delete -f deployments.yaml

### ****2️⃣ Stop & Delete Kind Cluster****

sh

-----

kind delete cluster

✅ This **cleans up everything**.

## ****🔹 8. Restart Everything****

### ****1️⃣ Start Kind Cluster****

sh

-----

kind create cluster --config=- <<EOF

kind: Cluster

apiVersion: kind.x-k8s.io/v1alpha4

nodes:

- role: control-plane

extraPortMappings:

- containerPort: 80

hostPort: 80

- containerPort: 443

hostPort: 443

EOF

### ****2️⃣ Re-Deploy Everything****

sh

-----

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml

kubectl apply -f deployments.yaml

kubectl apply -f ingress.yaml

### ****3️⃣ Verify****

sh

-----

kubectl get pods,svc,ingress

✅ Now test http://localhost/app1 and http://localhost/app2 again.

## ****🎯 Quick Commands Summary****

| ****Action**** | ****Command**** |
| --- | --- |
| **Create Cluster** | kind create cluster --config=- <config> |
| **Check Cluster** | kubectl cluster-info |
| **Install Ingress** | kubectl apply -f <ingress-url> |
| **Deploy Apps & Services** | kubectl apply -f deployments.yaml |
| **Deploy Ingress** | kubectl apply -f ingress.yaml |
| **Verify Resources** | kubectl get pods,svc,ingress |
| **Test Apps** | http://localhost/app1, http://localhost/app2 |
| **Stop Apps & Ingress** | kubectl delete -f deployments.yaml && kubectl delete -f ingress.yaml |
| **Delete Kind Cluster** | kind delete cluster |

## ****🚀 Conclusion****

This guide provides a **step-by-step** method to:

* **Install & Set up** Kind **on Mac**
* **Deploy & Expose** App1 **&** App2 **with Ingress**
* **Stop & Restart Everything Gracefully**