KIND ile Çalışma Adımları

1. KIND Cluster Oluşturma

kind-config.yaml

```
apiVersion: kind.x-k8s.io/v1alpha4
kind: Cluster
nodes:
- role: control-plane
  extraPortMappings:
  - containerPort: 30000
    hostPort: 30000
    listenAddress: "0.0.0.0" # Optional, defaults to "0.0.0.0"
    protocol: tcp # Optional, defaults to tcp
  - containerPort: 30001
    hostPort: 30001
  - containerPort: 30002
    hostPort: 30002
- role: worker
- role: worker
```

```
kind create cluster --name my-cluster --config kind-config.yaml
```

Bu komut, `my-cluster` adında bir Kubernetes cluster'ı oluşturur. Docker konteynerlerinde bir cluster kurar ve bu cluster ile etkileşimde bulunmanıza olanak sağlar.

2. KIND Cluster Bilgilerini Görüntüleme

```
kubectl cluster-info
```

Bu komut, Kubernetes cluster'ınız hakkında genel bilgileri gösterir. KIND tarafından oluşturulan cluster'ınızın adresini ve diğer bilgileri görüntüleyebilirsiniz.

3. Deployment Oluşturma

```
kubectl create deployment hello-kind --image=kicbase/echo-server:1.0
```

Bu komut, `hello-kind` adında bir deployment oluşturur ve `kicbase/echo-server:1.0` Docker imajını kullanır. Deployment, Kubernetes cluster'ınızda bir uygulamanın çalışmasını sağlar.

4. NodePort Servisi Oluşturma

hello-kind-svc.yaml

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: hello-kind
spec:
  type: NodePort
  selector:
    app: hello-kind
  ports:
    - port: 8080
      targetPort: 8080
      nodePort: 30001
```

```
kubectl apply -f hello-kind-svc.yaml
```

Bu komut, `hello-kind` deployment'ını bir NodePort türündeki servis olarak açar. Servis, 8080 portunda dinleyecek şekilde yapılandırılır ve cluster dışından erişilebilir hale gelir.

5. Servise Erişim Sağlama

```
kubectl get svc hello-kind
```

Bu komut, `hello-kind` servisinin IP adresini ve port numarasını gösterir. IP adresi ve port numarasını kullanarak servisinize erişebilirsiniz. KIND cluster'ı için IP genellikle localhost olarak ayarlanır.

```
curl http://localhost:30001
```

6. Pod'ların Durumunu Görüntüleme

```
kubectl get pods
```

7. Servislerin Durumunu Görüntüleme

```
kubectl get svc
```

8. Pod'a Bağlanma

```
kubectl exec -it <pod-name> -- /bin/bash
```

Bu komut, belirtilen pod'a etkileşimli bir terminal açar. `<pod-name>` yerine bağlanmak istediğiniz pod'un adını yazın.

9. NGINX Deployment Oluştur

```
kubectl create deployment nginx --image=nginx
```

nginx-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30002
```

```
kubectl apply -f nginx-service.yaml
kubectl get svc nginx
kubectl get pods
```

http://localhost:30002

🐳 KIND ile Docker İmajı Oluşturmak ve Yüklemek

Kind kullanmanın en güzel özelliklerinden biri, Docker Hub gibi bir imaj deposuna yüklemeden doğrudan local docker imajlarını Kind cluster'ına yükleyebilmemizdir.

Aşağıda, kendi Docker imajımızı oluşturup bu imajı Kind cluster'ına nasıl yükleyeceğimizi gösteriyoruz. Basit bir **NGINX tabanlı web sunucusu** kuracağız.

📁 Klasörleri Oluştur

```
mkdir sample-app
mkdir sample-app/files
```

📝 HTML ve NGINX Ayar Dosyalarını Oluştur

📄 `sample-app/files/index.html`

```
<html>
  <head>
    <title>Dockerfile</title>
  </head>
```

```
  <body>
    <div class="container">
      <h1>My App</h1>
      <h2>This is my first app for KIND</h2>
      <p>Hello everyone, This is running via Docker container</p>
    </div>
  </body>
</html>
```

📄 sample-app/files/default

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /usr/share/nginx/html;
    index index.html index.htm;

    server_name _;
    location / {
        try_files $uri $uri/ =404;
    }
}
```

🐋 Dockerfile Oluştur (sample-app/Dockerfile)

```
FROM ubuntu:24.04
RUN  apt -y update && apt -y install nginx
COPY files/default /etc/nginx/sites-available/default
COPY files/index.html /usr/share/nginx/html/index.html
EXPOSE 80
CMD ["/usr/sbin/nginx", "-g", "daemon off;"]
```

🔧 Docker İmajını Oluştur

```
cd sample-app
docker build -t sample-app:1.0 .
```

📦 Docker İmajını Kind Cluster'a Yükle

```
kind load docker-image sample-app:1.0 --name my-cluster
```

`my-cluster` senin cluster adın. Farklıysa kendi adını kullan.

🚀 Kubernetes Deployment ve Servis Dosyası Oluştur

📄 `my-deployment.yml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      containers:
      - name: sample-app
        image: sample-app:1.0
        imagePullPolicy: Never
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: sample-app-svc
spec:
  selector:
    app: sample-app
  type: NodePort
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30000
```

`imagePullPolicy: Never` → Çünkü imajı Docker Hub'dan değil, localden kullanıyoruz.
`nodePort: 30000` → `kind-config.yaml` dosyanda açtığın portla eşleşmeli.

✅ Deploy Et

```
kubectl apply -f my-deployment.yml
```

Pod çalışıyor mu kontrol et:

```
kubectl get pods
```

🌐 Uygulamayı Aç

Tarayıcıda aç:

http://localhost:30000

10. KIND Cluster'ını Silme

```
kind delete cluster --name my-cluster
```

Bu komut, `my-cluster` adındaki KIND cluster'ını siler ve tüm ilgili kaynakları kaldırır. Cluster tamamen kaldırılır ve geri yüklenemez.