1. What is Kind?

Kind 는 로컬 컴퓨터 환경에 쿠버네티스 클러스터를 손쉽고 빠르게 설치 하기 위해 만들어진 도구 입니다.

Kind는 Go 언어를 기반으로 만들어 졌으며, Docker 이미지를 기반으로 kubeadm을 이용하여 클러스터를 배포 합니다.

kind 공식 홈페이지 : kind.sigs.k8s.io

2. Kind 설치 하기

설치 가이드 원본 URL : https://kind.sigs.k8s.io/docs/user/quick-start/#installation

MacOS

```
brew install kind
```

Windows

```
choco install kind
```

Linux

```
curl.exe -Lo kind-windows-amd64.exe https://kind.sigs.k8s.io/dl/v0.15.0/kind-windows-amd64
Move-Item .\kind-windows-amd64.exe c:\some-dir-in-your-PATH\kind.exe
```

Kind 로 클러스터 생성 (초간단)

클러스터 생성

```
kind create cluster # Default cluster context 이름은 'kind' 로 생성
kind create cluster --name dangtong # cluster context 이름을 'dangtong' 으로 지정
```

클러스터 생성 확인

```
kind get clusters
kubectl cluster-info --context dangtong
```

클러스터 삭제

```
kind delete cluster
```

설정 파일을 이용한 Kind 클러스터 생성

클러스터 생성

설정 파일을 이용한 클러스터 생성

설정파일을 이용해서 kind 클러스터를 생성할 수 있습니다.

```
kind create cluster --config kind-example-config.yaml
```

3개 노드 클러스터 생성

3개 노드(1 controller, 2worker) 클러스터 설정

```
# three node (two workers) cluster config
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
- role: worker
- role: worker
```

6개 노드 클러스터 생성

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
- role: control-plane
- role: control-plane
- role: worker
- role: worker
- role: worker
```

쿠버네티스 버전 설정

쿠버네티스 버전에 따른 이미지는 링크에서 확인 가능 : https://github.com/kubernetes-sigs/kind/releases

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  image:
kindest/node:v1.16.4@sha256:b91a2c2317a000f3a783489dfb755064177dbc3a0b2f4147d50f04825d0
16f55
- role: worker
  image:
kindest/node:v1.16.4@sha256:b91a2c2317a000f3a783489dfb755064177dbc3a0b2f4147d50f04825d0
16f55
```

네트워크 설정

- Pod Subnet 설정

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
networking:
  podSubnet: "10.244.0.0/16"
```

- Service Subnet 설정

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
networking:
  serviceSubnet: "10.96.0.0/12"
```

- Default CNI 설정

Caliaco 완 같은 3rd party CNI 사용을 위해서는 default CNI 설치를 하지 말아야 합니다.

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
networking:
  # default CNI가 설치 되지 않습니다.
  disableDefaultCNI: true
```

- kube-proxy 모드 설정

iptables 또는 IPVS 중에 선택해서 사용 가능. default 는 iptables

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
networking:
  kubeProxyMode: "ipvs"
```

Ingress 및 LoadBalancer 설정

Ingress 및 Loadbalancer 를 설정하기 위해서는 KIND 를 이용한 클러스터 생성시 extraPortMapping 설정을 하고, kubeadm툴을 통해 custom node label 을 노드에 설정해야 합니다.

Ingress 가능한 클러스터 생성

kind 클러스터를 extraPortMappings 및 node-lables 설정과 함께 생성 합니다.

- ExtreaPortMappings : 로컬 호스트가 80 및 443 포트를 통해 Ingress Controller로 요청이 가능하게 설정합니다.
- node--labels : Ingress Controller 가 특정 라벨을 가진 노드에서만 수행 되도록 합니다.

```
cat <<EOF | kind create cluster --config=-
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  kubeadmConfigPatches:
  - |
    kind: InitConfiguration
    nodeRegistration:
      kubeletExtraArgs:
        node-labels: "ingress-ready=true"
  extraPortMappings:
  - containerPort: 80
    hostPort: 80
    protocol: TCP
  - containerPort: 443
    hostPort: 443
    protocol: TCP
EOF
```

Countour Ingress 생성

- Contour 설치

```
kubectl apply -f https://projectcontour.io/quickstart/contour.yaml
```

- Contour 설정 업데이트

```
kubectl patch daemonsets -n projectcontour envoy -p '{"spec":{"template":{"spec":
{"nodeSelector":{"ingress-ready":"true"},"tolerations":[{"key":"node-
role.kubernetes.io/control-plane","operator":"Equal","effect":"NoSchedule"},
{"key":"node-role.kubernetes.io/master","operator":"Equal","effect":"NoSchedule"}]}}}}'
```

Kong Ingress 생성

- Kong 설치

```
kubectl apply -f https://raw.githubusercontent.com/Kong/kubernetes-ingress-
controller/master/deploy/single/all-in-one-dbless.yaml
```

- Kong 설정 업데이트

```
kubectl patch deployment -n kong ingress-kong -p '{"spec":{"template":{"spec":
{"containers":[{"name":"proxy","ports":
[{"containerPort":8000,"hostPort":80,"name":"proxy","protocol":"TCP"},
{"containerPort":8443,"hostPort":443,"name":"proxy-
ssl","protocol":"TCP"}]}],"nodeSelector":{"ingress-ready":"true"},"tolerations":
[{"key":"node-role.kubernetes.io/control-
plane","operator":"Equal","effect":"NoSchedule"},{"key":"node-
role.kubernetes.io/master","operator":"Equal","effect":"NoSchedule"}]}}}}'
```

Nginx Ingress 생성

- Nginx 설치

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

- Nginx 설정 업데이트

```
kubectl wait --namespace ingress-nginx \
  --for=condition=ready pod \
  --selector=app.kubernetes.io/component=controller \
  --timeout=90s
```

Ingress 사용 예제

```
kind: Pod
apiVersion: v1
metadata:
  name: foo-app
  labels:
    app: foo
spec:
```

```
  containers:
  - name: foo-app
    image: hashicorp/http-echo:0.2.3
    args:
    - "-text=foo"
---
kind: Service
apiVersion: v1
metadata:
  name: foo-service
spec:
  selector:
    app: foo
  ports:
  # Default port used by the image
  - port: 5678
---
kind: Pod
apiVersion: v1
metadata:
  name: bar-app
  labels:
    app: bar
spec:
  containers:
  - name: bar-app
    image: hashicorp/http-echo:0.2.3
    args:
    - "-text=bar"
---
kind: Service
apiVersion: v1
metadata:
  name: bar-service
spec:
  selector:
    app: bar
  ports:
  # Default port used by the image
  - port: 5678
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
spec:
  rules:
  - http:
      paths:
```

```
      - pathType: Prefix
        path: "/foo"
        backend:
          service:
            name: foo-service
            port:
              number: 5678
      - pathType: Prefix
        path: "/bar"
        backend:
          service:
            name: bar-service
            port:
              number: 5678
---
```