kind Kubernetes in Docker

CONCEPTS

Each kind node is a container. Your cluster lives inside Docker (or Podman), so it needs no VM and starts in under a minute.

Docker / Podman host runtime
- **Node container** kindest/node image
  - **Kubernetes** kubelet, pods

Loaded images with a :latest tag are pulled anyway. Use a versioned tag or imagePullPolicy: IfNotPresent.

Contexts are named kind-<name>, so the default cluster is kind-kind.

CLUSTERS

kind create cluster
Create a cluster named kind

kind create cluster --name *<name>*
Create a named cluster

kind get clusters
List clusters

kind delete cluster
Delete the default cluster

kind delete cluster --name *<name>*
Delete a named cluster

kind create cluster --image kindest/node:*<tag>*
Pick the Kubernetes version

kind create cluster --wait 5m
Block until the control plane is ready

kind get nodes --name *<name>*
List node containers

kind delete clusters --all
Delete every cluster

kind version
Show the kind version

CONFIG FILE

kind create cluster --config *<file>*
Create from a config file

CONFIG FILE (cont.)

nodes: [{role: control-plane}, {role: worker}]
Add worker nodes

extraPortMappings: [{containerPort, hostPort}]
Publish a node port on the host

image: kindest/node:*<tag>***@sha256:***<digest>*
Pin a node image

featureGates: {*<Gate>***: true}**
Enable a feature gate

Use three control-plane nodes in the config for an HA control plane.

IMAGES

kind load docker-image *<image>*
Copy a local image into the nodes

kind load docker-image *<image>* **--name** *<name>*
Load into a named cluster

kind load image-archive *<file.tar>*
Load an image from a tarball

docker exec -it *<name>***-control-plane crictl images**
List images inside a node

kind build node-image
Build a node image from Kubernetes source

KUBECONFIG

kubectl cluster-info --context kind-*<name>*
Check you can reach the cluster

kubectl config use-context kind-*<name>*
Switch kubectl to this cluster

kind get kubeconfig --name *<name>*
Print the kubeconfig

kind export kubeconfig --name *<name>*
Write the context into your kubeconfig

kind create cluster --kubeconfig *<file>*
Write to a separate file

DEBUGGING

kind export logs *<dir>*
Dump node and pod logs to a folder

docker exec -it *<name>***-control-plane bash**
Shell into a node

docker ps --filter name=*<name>*
See the node containers

KIND_EXPERIMENTAL_PROVIDER=podman kind create cluster
Use Podman instead of Docker

kind create cluster -v 5
Verbose output

Behind a proxy, set HTTP_PROXY, HTTPS_PROXY and NO_PROXY before creating the cluster.

RECIPES

Three nodes with ports 80 and 443

```
cat <<EOF > kind.yaml
kind: Cluster
apiVersion: kind.x-k8s.io/
v1alpha4
nodes:
- role: control-plane
  extraPortMappings:
  - {containerPort: 80, hostPort:
80}
  - {containerPort: 443,
hostPort: 443}
- role: worker
- role: worker
EOF
kind create cluster --name dev --
config kind.yaml
```

Test a local image

```
docker build -t myapp:dev .
kind load docker-image myapp:dev
--name dev
kubectl create deployment myapp
--image=myapp:dev
# set imagePullPolicy:
IfNotPresent so it is not pulled
```

Throwaway CI cluster

```
kind create cluster --name ci --
wait 5m
kubectl --context kind-ci get
nodes
# run tests ...
kind export logs ./logs --name ci
kind delete cluster --name ci
```