

KIND CLUSTER

(KUBERNETES IN DOCKER)

Kind Cluster

- Also known as (Kubernetes in Docker)
- Like Minikube
- It a tool that runs Kubernetes clusters entirely inside Docker containers.

When to Use Which?

- Minikube – Best for beginners, local development with persistent data, or simulating a "real" single-node cluster.
- Kind – Best for testing multi-node scenarios, CI/CD, contributing to Kubernetes, or rapid cluster experimentation.
-

Difference between Minikube and kind

Feature	Minikube	Kind
Purpose	Runs a single-node Kubernetes cluster locally	Runs multi-node Kubernetes clusters in Docker
Underlying Tech	Uses Virtual Machines (VMs) or containers	Uses Docker containers
Installation Complexity	Requires a hypervisor (if using VMs)	Requires only Docker
Cluster Type	Primarily single node	Supports multi-node clusters

Performance	Slightly heavier due to VM support	Lightweight since it runs fully in containers
Use Case	Best for local Kubernetes development and testing	Ideal for CI/CD and testing Kubernetes in containers
Networking	Creates its own VM or containerized network	Uses Docker's built-in networking
Resource Consumption	Higher (especially with VM-based setups)	Lower, as it runs entirely in containers
Preferred By	Developers testing full Kubernetes environments	CI/CD pipelines and quick Kubernetes testing

Prerequisites for Kind installation

Prerequisites	Details
Operating System	Linux, macOS, or Windows (with WSL2/Docker Desktop).
Docker	Installed and running (<code>docker --version</code>).
Kind CLI	Installed (<code>kind --version</code>).
kubectl	Installed (<code>kubectl version --client</code>).
System Resources	Minimum 2 CPU cores, 4 GB RAM, and 10 GB disk space.
Internet Connectivity	Required for downloading Kubernetes images and dependencies.

Kind installation in ubuntu (linux)

STEP 1 – Install Docker in ubuntu

1.1 Create a folder name docker

```
root@DELLG-15:/home/lili# mkdir docker
root@DELLG-15:/home/lili# ls
docker
root@DELLG-15:/home/lili# cd docker
root@DELLG-15:/home/lili/docker#
```

1.2 To check the current user

```
lili@DELLG-15:~/docker$ echo "Current user: $USER"
Current user: lili
lili@DELLG-15:~/docker$ ls
```

1.3 Then do

```
lili@DELLG-15:~$ sudo su
[sudo] password for lili:
root@DELLG-15:/home/lili#
```

1.4 Create a file name `install_docker.sh` in docker folder

```
root@DELLG-15:/home/lili/docker# vim install_docker.sh
root@DELLG-15:/home/lili/docker# cat install_docker.sh
#!/bin/bash

# Update package list
sudo apt-get update

# Install prerequisites
sudo apt-get install -y \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

# Add Docker's official GPG key
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
gpg --dearmor -o /etc/apt/keyrings/docker.gpg

# Set up the Docker repository
echo \
    "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null

# Update package index again
sudo apt-get update

# Install Docker Engine
```

```
sudo apt-get install -y \
    docker-ce \
    docker-ce-cli \
    containerd.io \
    docker-compose-plugin
```

```
# Verify Docker installation
sudo docker run hello-world
```

```
echo "Docker installed successfully! Please log out and back
in for group changes to take effect."
```

Change \$USER based on your user as for me its lili

1.5 Make it executable

```
root@DELLG-15:/home/lili/docker# chmod +x install_docker.sh
```

1.6 Run it:

```
root@DELLG-15:/home/lili/docker# ./install_docker.sh
```

1.7 Verify Installation

```
root@DELLG-15:/home/lili/docker# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

1.8 Now exit from root privileges then try "docker ps"

You will find this error

```
lili@DELLG-15:~$ docker ps permission denied while trying to
connect to the Docker daemon socket at
unix:///var/run/docker.sock: Get
"http://%2Fvar%2Frun%2Fdocker.sock/v1.47/containers/json":
dial unix /var/run/docker.sock: connect: permission denied
```

To resolve this

```
sudo usermod -aG docker $USER
```

then

```
newgrp docker
```

now try running

```
docker ps
```

without sudo

STEP 2 – Installing KIND and kubectl

2.1 Create a new folder name kind_kubectl

```
mkdir kind_kubectl
```

2.2 Create a file in folder kind_kubectl name [install_kind_kubernetes.sh](#)

```
vim install_kind_kubernetes.sh
```

2.3 Install KIND and kubectl using the provided script:

```
#!/bin/bash

[ $(uname -m) = x86_64 ] && curl -Lo ./kind
https://kind.sigs.k8s.io/dl/v0.20.0/kind-linux-amd64
chmod +x ./kind
sudo cp ./kind /usr/local/bin/kind

VERSION="v1.30.0"
URL="https://dl.k8s.io/release/${VERSION}/bin/linux/amd64/kube
ctl"
INSTALL_DIR="/usr/local/bin"

curl -LO "$URL"
chmod +x kubectl
sudo mv kubectl $INSTALL_DIR/
kubectl version --client

rm -f kubectl
rm -rf kind

echo "kind & kubectl installation complete."
```

VERSION – give the latest version

2.4 Make it executable

```
lili@DELLG-15:~/kind_kubectl$ chmod +x install_kind_kubernetes.sh
```

2.5 Run it:

```

lili@DELLG-15:~/kind_kubectrl$ ./install_kind_kubernetes.sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             %         Dload  Upload  Total  Spent    Left     Speed
100    97    100    97     0     0    116      0  --:--:-- --:--:-- --:--:--    116
0      0      0      0     0     0      0      0  --:--:-- 0:00:01 --:--:--      0
100 6304k  100 6304k     0     0   929k      0  0:00:06 0:00:06 --:--:--  1493k
[sudo] password for lili:
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             %         Dload  Upload  Total  Spent    Left     Speed
100   138    100   138     0     0    202      0  --:--:-- --:--:-- --:--:--    202
100 49.0M  100 49.0M     0     0  1986k      0  0:00:25 0:00:25 --:--:--  3034k
Client Version: v1.30.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
kind & kubectrl installation complete.

```

STEP 3 – Setting Up the KIND Cluster

3.1 Create a `kind-cluster-config.yaml` files in `kind_kubectrl`:

```

kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4

nodes:
- role: control-plane
  image: kindest/node:v1.32.1
- role: worker
  image: kindest/node:v1.32.1
- role: worker
  image: kindest/node:v1.32.1
  extraPortMappings:
  - containerPort: 80
    hostPort: 80
    protocol: TCP
  - containerPort: 443
    hostPort: 443
    protocol: TCP

```











`kindest/node:v1.32.1` – Go to DockerHub and search for latest image of `kindest/node`

3.2 Create the cluster using the configuration file:

```

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

```

```
Creating cluster "my-kind-cluster" ...  
✓ Ensuring node image (kindest/node:v1.32.1)   
✓ Preparing nodes      
✓ Writing configuration   
✓ Starting control-plane   
✓ Installing CNI   
✓ Installing StorageClass   
✓ Joining worker nodes   
Set kubectl context to "kind-my-kind-cluster"  
You can now use your cluster with:  
  
kubectl cluster-info --context kind-my-kind-cluster
```

in place of `my-kind-cluster` you can give any name as you want

3.3 Verify the cluster:

```
kubectl get nodes  
kubectl cluster-info
```