Table of Contents

Minikube

minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.

Launch EC2 Instance

- Use the following configurations for the Instance:
  - AMI : **Ubuntu Server 22.04 LTS**
  - Instance Type: `t2.medium` as Minikube requires a minimum of 2 vCPUs.
    - If we use `t2.micro`, there will be an error like **Requested cpu count 1 is less than the minimum allowed of 2**

> Since t2.medium does not come into AWS Free Tier, make sure to stop or terminate the instance when not in use to avoid AWS Billing.

Installation of kubectl, docker and Minikube

- Navigate to downloadkubernetes

kubectl

```
# Replace the below link from above website

### For ARM 64-bit CPU ###
curl -LO https://dl.k8s.io/v1.26.3/bin/linux/arm64/kubectl

### For Intel 64-bit CPU ###
# curl -LO https://dl.k8s.io/v1.26.3/bin/linux/amd64/kubectl
# curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s
https://storage.googleapis.com/kubernetes-
release/release/stable.txt`/bin/linux/amd64/kubectl

chmod +x ./kubectl && sudo mv ./kubectl /usr/local/bin/kubectl
```

```
kubectl version
```

Docker

- Install Docker

```
sudo apt-get update && sudo apt-get install docker.io net-tools -y
sudo docker images
sudo docker ps
```

Minikube

- Install Minikube

```
# If Host Instance Architecture is ARM64
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
arm64
sudo install minikube-linux-arm64 minikube
sudo chmod +x minikube && sudo mv minikube /usr/local/bin/

# If Host Instance Architecture is AMD64
# curl -Lo minikube
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 &&
chmod +x minikube && sudo mv minikube /usr/local/bin/

minikube version
```

Pre-Requisites for Minikube

- Running Minikube on Ubuntu EC2

```
# sudo minikube start --vm-driver=none

# X Exiting due to GUEST_MISSING_CONNTRACK: Sorry, Kubernetes 1.20.2 requires
conntrack to be installed in root's path
# ❌  Exiting due to GUEST_MISSING_CONNTRACK: Sorry, Kubernetes 1.26.3 requires
crictl to be installed in root's path
# The none driver with Kubernetes v1.24+ and the docker container-runtime requires
cri-dockerd.

sudo apt install conntrack -y
sudo apt-get install containernetworking-plugins

# To check the latest version: https://github.com/kubernetes-sigs/cri-
tools/releases
```

```
### For Intel 64-bit CPU ###
# VERSION="v1.26.1"
# wget https://github.com/kubernetes-sigs/cri-
tools/releases/download/$VERSION/crictl-$VERSION-linux-amd64.tar.gz
# sudo tar zxvf crictl-$VERSION-linux-amd64.tar.gz -C /usr/local/bin
# rm -f crictl-$VERSION-linux-amd64.tar.gz

### For ARM 64-bit CPU ###
VERSION="v1.26.1"
wget https://github.com/kubernetes-sigs/cri-
tools/releases/download/$VERSION/crictl-$VERSION-linux-arm64.tar.gz
sudo tar zxvf crictl-$VERSION-linux-arm64.tar.gz -C /usr/local/bin
rm -f crictl-$VERSION-linux-arm64.tar.gz

# Validate if binary is installed.
sudo /usr/local/bin/crictl version
sudo /usr/local/bin/crictl info

# Install cri-dockerd
VER=$(curl -s https://api.github.com/repos/Mirantis/cri-
dockerd/releases/latest|grep tag_name | cut -d '"' -f 4|sed 's/v//g')
echo $VER

### For Intel 64-bit CPU ###
# wget https://github.com/Mirantis/cri-dockerd/releases/download/v${VER}/cri-
dockerd-${VER}.amd64.tgz
# tar xvf cri-dockerd-${VER}.amd64.tgz

### For ARM 64-bit CPU ###

wget https://github.com/Mirantis/cri-dockerd/releases/download/v${VER}/cri-
dockerd-${VER}.arm64.tgz
tar xvf cri-dockerd-${VER}.arm64.tgz

# Move cri-dockerd binary package to /usr/local/bin directory
sudo mv cri-dockerd/cri-dockerd /usr/local/bin/

# Validate the installation
cri-dockerd --version

wget https://raw.githubusercontent.com/Mirantis/cri-
dockerd/master/packaging/systemd/cri-docker.service
wget https://raw.githubusercontent.com/Mirantis/cri-
dockerd/master/packaging/systemd/cri-docker.socket
sudo mv cri-docker.socket cri-docker.service /etc/systemd/system/
sudo sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,'
/etc/systemd/system/cri-docker.service

sudo systemctl daemon-reload
sudo systemctl enable cri-docker.service
sudo systemctl enable --now cri-docker.socket

# Since this Host contains Memory as 1851mb
# sudo minikube config set memory 1800
```

```
sudo minikube start --driver=none
sudo minikube start --driver=none --network-plugin=cni --cni=calico
ubuntu@ip-172-31-42-35:~$ sudo minikube start --driver=none --network-plugin=cni -
-cni=calico
😄  minikube v1.30.1 on Ubuntu 22.04 (arm64)
✨  Using the none driver based on user configuration
👍  Starting control plane node minikube in cluster minikube
🤹  Running on localhost (CPUs=2, Memory=1851MB, Disk=7764MB) ...
ℹ  OS release is Ubuntu 22.04.2 LTS
🐳  Preparing Kubernetes v1.26.3 on Docker 20.10.21 ...
    ▪ kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🤹  Configuring local host environment ...

❗  The 'none' driver is designed for experts who need to integrate with an
existing VM
💡  Most users should use the newer 'docker' driver instead, which does not
require root!
📘  For more information, see:
https://minikube.sigs.k8s.io/docs/reference/drivers/none/

❗  kubectl and minikube configuration will be stored in /root
❗  To use kubectl or minikube commands as your own user, you may need to
relocate them. For example, to overwrite your own settings, run:

    ▪ sudo mv /root/.kube /root/.minikube $HOME
    ▪ sudo chown -R $USER $HOME/.kube $HOME/.minikube

💡  This can also be done automatically by setting the env var
CHANGE_MINIKUBE_NONE_USER=true
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🔎  Verifying Kubernetes components...
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default"
namespace by default
##

minikube status

# sudo kubectl get node ip-172-31-42-35 -o jsonpath='{.status.capacity}'
# {"cpu":"2","ephemeral-storage":"7950536Ki","hugepages-1Gi":"0","hugepages-
2Mi":"0","hugepages-32Mi":"0","hugepages-
64Ki":"0","memory":"1896032Ki","pods":"110"}
```

Minikube Kubernetes Operations

kubectl commands

get

- Prints a table of the most important information about the specified resources.

```
# List all pods in ps output format
sudo kubectl get pods

sudo kubectl get nodes

sudo kubectl describe node NODENAME
# node.kubernetes.io/not-ready:NoSchedule

# kubectl taint node ip-172-31-29-172 node.kubernetes.io/not-ready:NoSchedule-


# kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

- To execute the containers using `kubectl`:

```


# Run a test container image that includes a webserver
sudo kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-
images/agnhost:2.39 -- /agnhost netexec --http-port=8080
sudo kubectl get deployments
sudo kubectl get pods

# sudo kubectl run hello-minikube --image=gcr.io/google_containers/echoserver:1.4
--port=8080

sudo kubectl get svc
# NAME          TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
# kubernetes    ClusterIP   10.96.0.1     <none>        443/TCP   3m23s

sudo kubectl get all
# NAME                  READY   STATUS    RESTARTS   AGE
# pod/hello-minikube    1/1     Running   0          2m5s

# NAME                  TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
# service/kubernetes    ClusterIP   10.96.0.1    <none>        443/TCP   3m43s
```

> `ClusterIP` (default) - Exposes the Service on an internal IP in the cluster. This type makes the Service only reachable from within the cluster. `NodePort` - Exposes the Service on the same port of each selected Node in the cluster using NAT. Makes a Service accessible from outside the cluster using : Superset of ClusterIP.

- Expose the container port to access it and find as to where the port 8080 in the container exposed is on the EC2 Instance port.

```
sudo kubectl expose pod hello-minikube --type=NodePort
# service/hello-minikube exposed

# Display the list of services and their exposed ports.
sudo kubectl get svc
# NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
# hello-minikube    NodePort    10.101.217.142   <none>        8080:30821/TCP
4m25s
# kubernetes        ClusterIP   10.96.0.1        <none>        443/TCP          14m
```

- Create a deployment object using a manifest file

```
sudo kubectl apply -f https://k8s.io/examples/controllers/nginx-deployment.yaml
sudo kubectl get deployments
sudo kubectl get rs
sudo kubectl get pods
```

- To check for particular process running on port in linux:

```
# As netstat command is not available in ubuntu, install using below
sudo apt install net-tools -y
netstat -nltp
```

> `Note`: Port `30821` is the EC2 Instance Port where the Port 8080 of the container is exposed. The value of EC2 Instance Port changes each time you expose a port, so this may be different for your instance.

- Access the container response via the EC2 Instance Port on a web browser by `http://<EC2_PUBLIC_IP>:30821`