OSPO CON

CYOK

Choose Your Own Kubernetes

For Local Development

#ospocon @gkarthiks

THE LINUX FOUNDATION

About me

Karthikeyan Govindaraj
OpenSource Enthusiast | Writer | CNCF Speaker

Developer Evangelist @ BlackRock

https://bio.link/gkarthiks

https://gkarthiks.medium.com

THE LINUX FOUNDATION

@gkarthics

Concepts

Single Node Cluster: A complete Kubernetes services viz., kube-apiserver, etcd, kube-scheduler, kube-controller-manager running in a single instance of a machine

Multi Node Cluster(HA): A Kubernetes cluster which has multiple worker nodes for workload scheduling, providing HA for the deployed service

HA Control-Plane Cluster: A Kubernetes cluster with multiple worker nodes for deployed workload HA and multiple control-plane node for the Kubernetes cluster HA itself

microK8s

Creates a single/multi node cluster

Built-in container registry as add-on

Start Kubernetes

```
microk8s install
microk8s status --wait-ready
microk8s enable dashboard dns registry
microk8s kubectl get all -A
alias kubectl="microk8s kubectl"
```

```
ubuntu@microk8s-vm-01:~$ microk8s status --wait-ready
microk8s is running
high-availability: no
  datastore master nodes: 127.0.0.1:19001
  datastore standby nodes: none
addons:
  enabled:
    dashboard            # The Kubernetes dashboard
    dns                  # CoreDNS
    ha-cluster           # Configure high availability on the current node
    metrics-server       # K8s Metrics Server for API access to service metrics
    registry             # Private image registry exposed on localhost:32000
    storage              # Storage class; allocates storage from host directory
  disabled:
    ambassador           # Ambassador API Gateway and Ingress
    cilium               # SDN, fast with full network policy
    fluentd              # Elasticsearch-Fluentd-Kibana logging and monitoring
    gpu                  # Automatic enablement of Nvidia CUDA
    helm                 # Helm 2 - the package manager for Kubernetes
    helm3                # Helm 3 - Kubernetes package manager
    host-access          # Allow Pods connecting to Host services smoothly
    ingress              # Ingress controller for external access
    istio                # Core Istio service mesh services
    jaeger               # Kubernetes Jaeger operator with its simple config
    keda                 # Kubernetes-based Event Driven Autoscaling
    knative              # The Knative framework on Kubernetes.
    kubeflow             # Kubeflow for easy ML deployments
    linkerd              # Linkerd is a service mesh for Kubernetes and other frameworks
    metallb              # Loadbalancer for your Kubernetes cluster
    multus               # Multus CNI enables attaching multiple network interfaces to pods
    openebs              # OpenEBS is the open-source storage solution for Kubernetes
    openfaas             # openfaas serverless framework
    portainer            # Portainer UI for your Kubernetes cluster
    prometheus           # Prometheus operator for monitoring and logging
    rbac                 # Role-Based Access Control for authorisation
    traefik              # traefik Ingress controller for external access
```

microK8s ....High Availability

Easy multi-node setup

multipass can be used for managing VMs

```
multipass launch --name microk8s-vm-01 --mem 4G --disk 40G
multipass launch --name microk8s-02 --mem 4G --disk 40G
multipass launch --name microk8s-03 --mem 4G --disk 40G
```

```
Tue Sep 07@03:46:42 ~/Desktop * > multipass ls
Name                    State             IPv4             Image
microk8s-02             Running           10.1.63.128      Ubuntu 20.04 LTS
microk8s-03             Running           10.1.62.192      Ubuntu 20.04 LTS
microk8s-vm-01          Running           10.1.17.0        Ubuntu 20.04 LTS
```

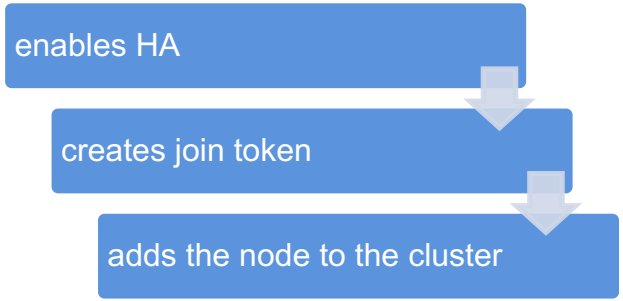
microK8s ....High Availability ….contd

- Exec into VM via `multipass shell <control-plane vm-name>`
- snap install microk8s in all VMs
- Create join token - `microk8s add-node`

```
ubuntu@microk8s-vm-01:~$ microk8s add-node
From the node you wish to join to this cluster, run the following:
microk8s join 10.0.2.15:25000/d6ee805bde2477d4792fee86490e531c/4d8393ca856b

If the node you are adding is not reachable through the default interface you can use one of the following:
 microk8s join 10.0.2.15:25000/d6ee805bde2477d4792fee86490e531c/4d8393ca856b
```

with the `add-node` command



Run the command in other VMs - `microk8s join` to join the cluster

Highly Available Cluster is ready

microK8s ....Usage

enable dashboard:
"*microk8s enable dashboard*"

setup kubeconfig:
"*multipass exec microk8s-vm-01 -- sudo /snap/bin/microk8s.config > kubeconfig*"

run from local:
kubectl --kubeconfig=kubeconfig get all --all-namespaces

proxy local:
kubectl --kubeconfig=kubeconfig proxy

microK8s ....Observation

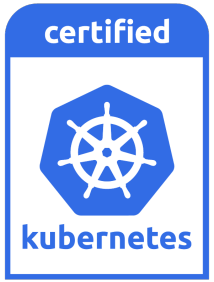
No need of docker daemon to run locally

Can run in all OS viz., Win, Mac, Linux

Built-in container registry for Images

Well suited for CI environment

Supports Windows containers



kind

Creates a single/multi node cluster

Follows the concept of **K**ubernetes **I**n **D**ocker

Owned by k8s SIG

Working with cluster

kind create cluster

```
Tue Sep 07@05:26:53  ~/Desktop * > kind create cluster
Creating cluster "kind" ...
 ✓ Ensuring node image (kindest/node:v1.21.1)
 ✓ Preparing nodes
 ✓ Writing configuration
 ✓ Starting control-plane
 ✓ Installing CNI
 ✓ Installing StorageClass
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Have a question, bug, or feature request? Let us know! https://kind.sigs.k8s.io/#community
```

kind …features

Create a custom
build of K8s:
“kind build node-image”

Create & Run
custom built k8s cluster
“kind create cluster --image <img>”

supports Linux,
macOS and Windows
(with docker daemon)

kind …features

Images can be loaded from local build

Allows locally built image to directly run inside K8s cluster

```
1  kind: Cluster
2  apiVersion: kind.x-k8s.io/v1alpha4
3  nodes:
4  - role: control-plane
5  - role: control-plane
6  - role: control-plane
7  - role: worker
8  - role: worker
9  - role: worker
```

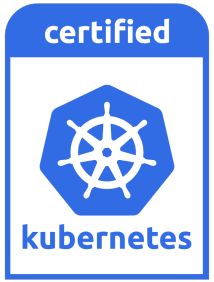
Supports kind configuration file

Creates multi-node and control-plane HA cluster

```
Tue Sep 07@10:12:36 ~/Desktop * > kubectl get no
NAME                  STATUS   ROLES                  AGE   VERSION
cp-ha-control-plane   Ready    control-plane,master   4h1m  v1.21.1
cp-ha-control-plane2  Ready    control-plane,master   4h1m  v1.21.1
cp-ha-control-plane3  Ready    control-plane,master   4h    v1.21.1
cp-ha-worker          Ready    <none>                 4h    v1.21.1
cp-ha-worker2         Ready    <none>                 4h    v1.21.1
cp-ha-worker3         Ready    <none>                 4h    v1.21.1
```

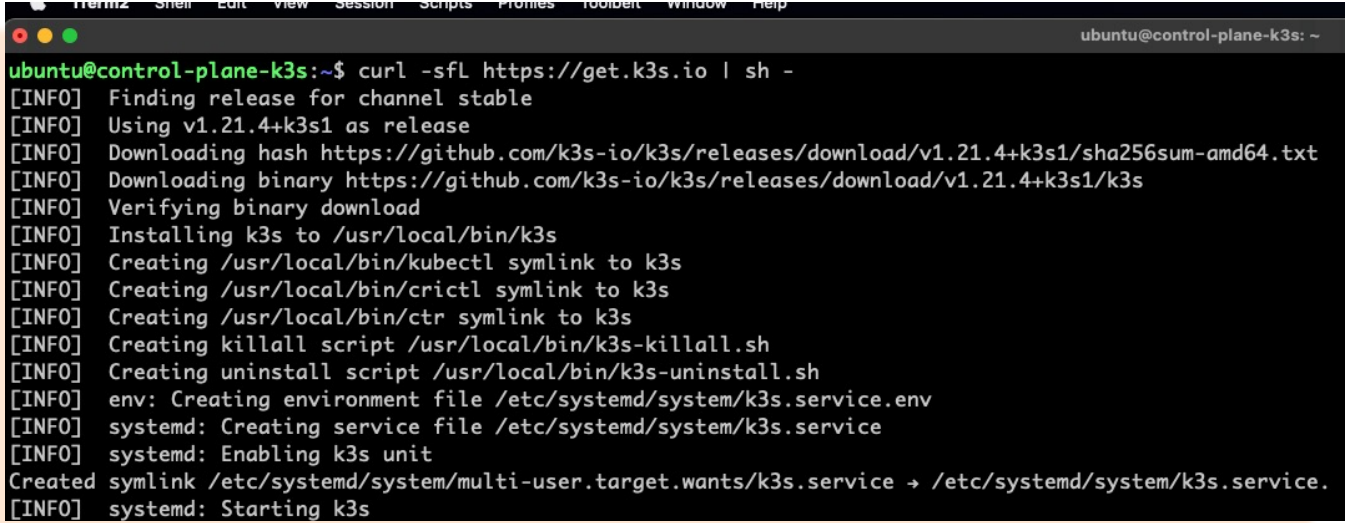
kind ….Observation

- Needs docker/podman on the host
- Runs in all OS viz., Win/Mac/Linux provided docker daemon is available.
- Additional deployment for lib/registry to run container registry for Images
- Well suited for CI environment
- Doesn't support Windows containers



k3s

- Can be installed directly only on Linux
- Needs either a VM in local machine or *k3d*
- multipass can be used for VMs



K3s …..multipass

```
ubuntu@control-plane-k3s:~$ k get  po -A
NAMESPACE     NAME                                     READY   STATUS      RESTARTS   AGE
kube-system   coredns-7448499f4d-qkrj7                 1/1     Running     0          12m
kube-system   metrics-server-86cbb8457f-8s8m8           1/1     Running     0          12m
kube-system   local-path-provisioner-5ff76fc89d-9gv9f  1/1     Running     0          12m
kube-system   helm-install-traefik-crd-mzz5l           0/1     Completed   0          12m
kube-system   helm-install-traefik-zhxl5               0/1     Completed   1          12m
kube-system   svclb-traefik-n5wzt                      2/2     Running     0          10m
kube-system   traefik-97b44b794-hcwvj                  1/1     Running     0          10m
```

Installs traefik by default

Simply export the config to access from local host

```
K3S_IP=$(multipass info control-plane-k3s | grep IPv4 | awk '{print $2}')
multipass exec control-plane-k3s sudo cat /etc/rancher/k3s/k3s.yaml > k3s.yaml
sed -i '' "s/127.0.0.1/${K3S_IP}/" k3s.yaml

kubectl --kubeconfig=k3s.yaml cluster-info
```

K3s multipass ….High Availability

K3s can also be run in HA mode

Follow same process to create multiple VMs

Get the control-plane token from "*/var/lib/rancher/k3s/server/node-token*"
& control-plane IP address

Set the token to the env variable "K3S_TOKEN"

Install k3s via script with token(automatically picked) and IP arguments as
curl -sfL https://get.k3s.io | k3S_URL=https://<control-plane-IP> sh -

k3d - Alternative to VM installations

k3d lightweight wrapper to run k3s in docker

```
Tue Sep 07@13:07:08 ~/Desktop * > k3d cluster create mycluster
INFO[0000] Prep: Network
INFO[0000] Created network 'k3d-mycluster' (feb31d765dc2c19faee37dd706d8bc587e1b94fe82abee3bf494493a8f918789)
INFO[0000] Created volume 'k3d-mycluster-images'
INFO[0001] Creating node 'k3d-mycluster-server-0'
INFO[0006] Pulling image 'docker.io/rancher/k3s:v1.21.3-k3s1'
INFO[0048] Creating LoadBalancer 'k3d-mycluster-serverlb'
INFO[0054] Pulling image 'docker.io/rancher/k3d-proxy:4.4.8'
INFO[0075] Starting cluster 'mycluster'
INFO[0075] Starting servers...
INFO[0075] Starting Node 'k3d-mycluster-server-0'
INFO[0081] Starting agents...
INFO[0081] Starting helpers...
INFO[0081] Starting Node 'k3d-mycluster-serverlb'
INFO[0081] (Optional) Trying to get IP of the docker host and inject it into the cluster as 'host.k3d.internal' for
WARN[0084] Failed to patch CoreDNS ConfigMap to include entry '192.168.65.2 host.k3d.internal': Exec process in node
INFO[0084] Successfully added host record to /etc/hosts in 2/2 nodes
INFO[0084] Cluster 'mycluster' created successfully!
INFO[0084] --kubeconfig-update-default=false --> sets --kubeconfig-switch-context=false
INFO[0084] You can now use it like this:
kubectl config use-context k3d-mycluster
kubectl cluster-info
```

Supports multi node cluster as well

k3d cluster create multiserver --servers 3

Auto clustering in multi-server mode

```
Tue Sep 07@13:15:13 ~/Desktop * > k3d cluster create multiserver --servers 3
INFO[0000] Prep: Network
INFO[0000] Created network 'k3d-multiserver' (bd6c945ded60338c5154cdcb7276f8928b1444092c45a413937f408862d2e22c)
INFO[0000] Created volume 'k3d-multiserver-images'
INFO[0000] Creating initializing server node
INFO[0000] Creating node 'k3d-multiserver-server-0'
INFO[0001] Creating node 'k3d-multiserver-server-1'
INFO[0002] Creating node 'k3d-multiserver-server-2'
INFO[0002] Creating LoadBalancer 'k3d-multiserver-serverlb'
INFO[0002] Starting cluster 'multiserver'
INFO[0002] Starting the initializing server...
INFO[0002] Starting Node 'k3d-multiserver-server-0'
INFO[0003] Starting servers...
INFO[0003] Starting Node 'k3d-multiserver-server-1'
INFO[0025] Starting Node 'k3d-multiserver-server-2'
INFO[0042] Starting agents...
INFO[0042] Starting helpers...
INFO[0042] Starting Node 'k3d-multiserver-serverlb'
INFO[0046] (Optional) Trying to get IP of the docker host and inject it into the cluster as 'host.k3d.internal' for easy access
INFO[0050] Successfully added host record to /etc/hosts in 4/4 nodes and to the CoreDNS ConfigMap
INFO[0050] Cluster 'multiserver' created successfully!
INFO[0050] --kubeconfig-update-default=false --> sets --kubeconfig-switch-context=false
INFO[0050] You can now use it like this:
kubectl config use-context k3d-multiserver
kubectl cluster-info
```

K3s k3d …High Availability

```
Tue Sep 07@13:31:47  ~/Desktop * > kubectl get no
NAME                        STATUS   ROLES                       AGE   VERSION
k3d-multiserver-server-0    Ready    control-plane,etcd,master   15m   v1.21.3+k3s1
k3d-multiserver-server-1    Ready    control-plane,etcd,master   15m   v1.21.3+k3s1
k3d-multiserver-server-2    Ready    control-plane,etcd,master   14m   v1.21.3+k3s1
```

multiserver control-plane cluster

k3d cluster create mycluster --registry-create

K3s ….Observation

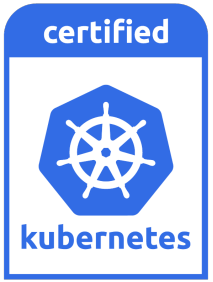
Compatible only with Linux (windows)

Needs multipass/vagrant in local (Linux VMs)

Also run with k3d

Needs docker on the host (for k3d mode)

Additional deployment for lib/registry to run container registry for Images (native k3s)



Concepts

Virtual Cluster: A complete Kubernetes Cluster spun up inside an existing Kubernetes Cluster

Host Cluster: An actual Kubernetes Cluster that hosts the virtual kubernetes clusters

Self Servicing K8s Cluster: A developer can login and create their own Kubernetes Cluster on demand

vCluster

Doesn’t create any cluster in local machine

Creates a cluster inside a running K8s cluster

Follows the *Kubernetes-IN-Kubernetes* concept

Developers works with remote cluster with thin cli

Developers can be (virtual)cluster admin

vCluster

CLI works with all OS viz., Mac/Linux/Windows

No container registry is available, as this will purely creates a Kubernetes cluster only

Namespaces and resources created within the virtual cluster will be encapsulated within the host namespace where vCluster runs

Creates an impartial cloud-native developer experience across all OS Developers

vCluster

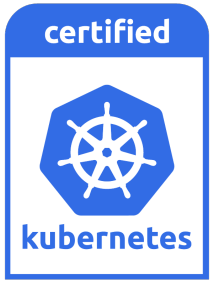
Creation of vCluster doesn't need admin perms

```
vcluster create my-vcluster -n host-namespace
vcluster connect my-vcluster -n host-namespace
export KUBECONFIG=./kubeconfig.yaml
kubectl get ns
kubectl get pods -n kube-system
```

From the host cluster, all the services are sync'd and pods are scheduled from ***syncer*** a scheduler handler

vCluster ….Observation

- A host cluster is mandatory
- Provides a remote development environment
- Need to deploy our own container registry, as this creates a pure K8s cluster only
- Impartial cloud-native developer experience for developers from all OS and machines viz., Mac/Win/Linux and VDIs




OSPO
CON