Day3 Introduction to Kubernetes (K8S)

Agenda

* What is K8S
* Why K8S
* How to implement/Use K8S
* Types of Deployment
* Master/Slave Architecture (Control Plane/ Nodes/Pods)
* Terminologies used in K8S

Revisit

Day1 DevOps Introduction (Development + Operations)

1. Git (SCM Tool – Source Code Mgmt/ Software Config Mgmt.) (Repository)
2. Jenkins, Github Actions, Circle CI, Argo CD, Puppet (CI/CD)
3. Containerization (Docker) – Docker Desktop, Podman Desktop.
4. Local vs Cloud Deployment (On-Prem & Remote deployment)
5. Container Orchestration (K8S, NoMad)
6. Helm Chart (It’s a package manager for K8S)

Source Code + Libraries + Runtime Env + Platform (Hardware + OS)

1. Dual Booting (Only one OS can run at a time)
2. Install Alongside (VM ware)
3. Docker

Docker Client (CLI/GUI)

Docker Server/Docker Deamon/Docker Host

Docker Hub (Registry for Images)

Docker Image – Class

Docker Container – Object

Interacting with Docke

1. CLI Mode (Difficult mode) – docker version, docker build, docker run, docker login, docker push, docker pull,

<https://kubernetes.io/>

docker pull syskantechnosoft/hello-world-app:latest

<https://hub.docker.com/r/syskantechnosoft/hello-world-app/tags>

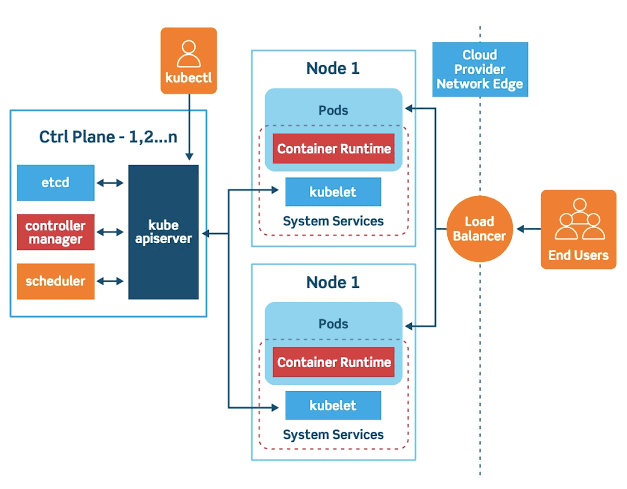
Kubernetes (K8S)

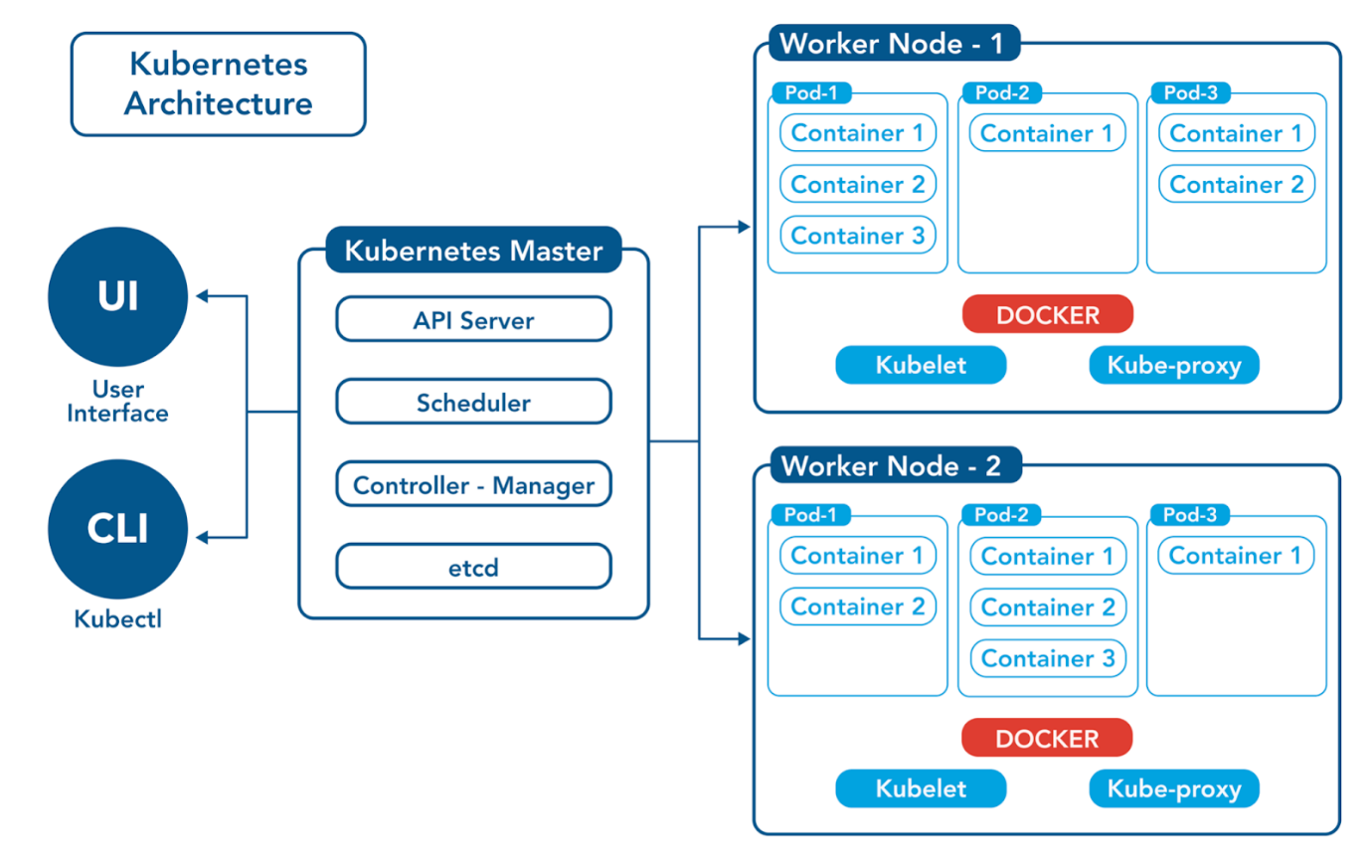
Container Orchestration Tool – Production Grade Container Orchestration.

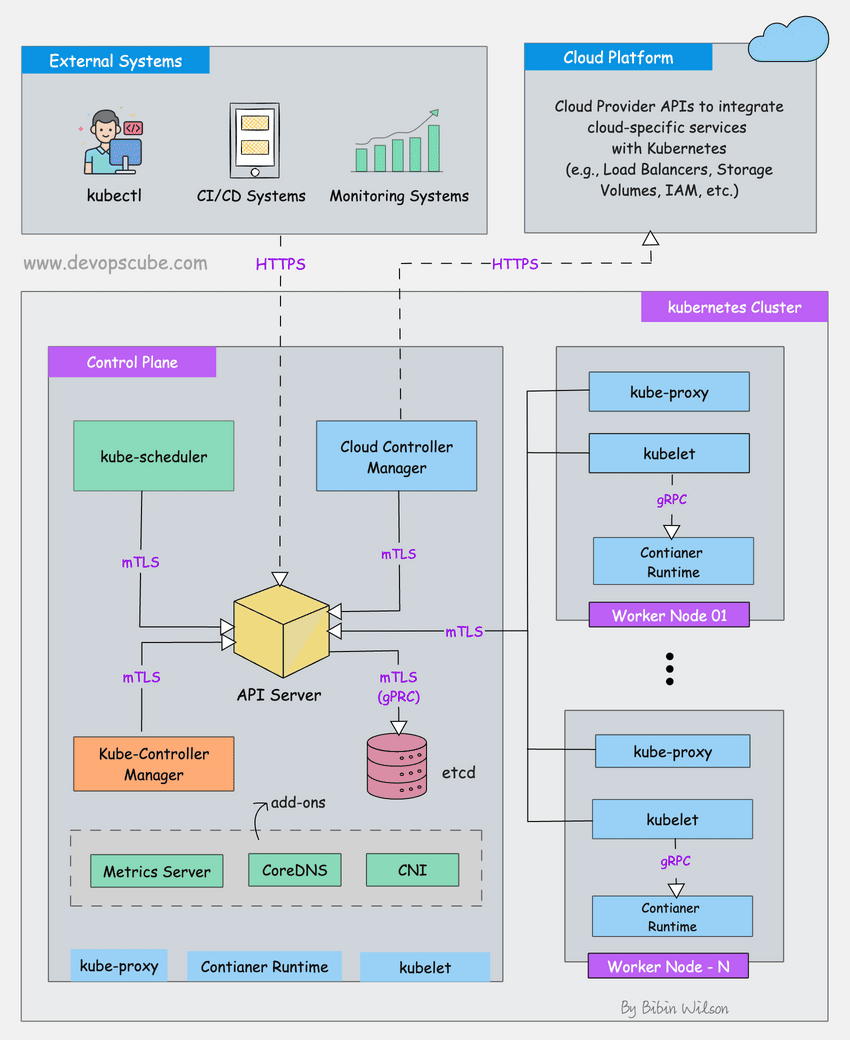
[Kubernetes](https://kubernetes.io/docs/concepts/overview/), also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications.

K8S Arch. (Master Slave Architecture)

1. Master Node (Control Plane) [Only one] {etcd – Database, controller, schedulers, api-server}
2. Worker Node (Slave Node) – Node [One or more] {pods, kubelet & kubeproxy }







K8S Tools -- <https://kubernetes.io/docs/tasks/tools/>

1. Kubectl
2. Minikube/kind – To create k8s Cluster
3. Kubeadm

<https://www.geeksforgeeks.org/devops/kubernetes-tutorial/>

<https://labs.play-with-k8s.com/>

Playground to learn the basic k8s operations.

1. Initializes Cluster Master Node

kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16

2. Initializa cluster networking

kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml

3. (optional) Create an nginx deployment

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml>

Podman Desktop Handson

<https://developers.redhat.com/articles/2024/05/07/podman-desktop-red-hat-developer-subscription#bootc_extension_for_podman_desktop>

<https://developers.redhat.com/articles/2023/06/09/deploy-and-test-kubernetes-containers-using-podman-desktop>

<https://developers.redhat.com/articles/2023/11/06/working-kubernetes-podman-desktop#running_an_application_in_kubernetes_from_podman_desktop>

Workflow (Github Actions)

ArgoCD or Jenkins or GithubActions

Front End

* (React or NextJS based app)
* Public
* Src
* Package.json
* Readme.md
* .gitignore
* Dockerfile
* Main.tf

Back End

* Eureka (discovery) Server [Consul/Zoo Keeper]
* API gateway service
* Config server
* Microservice1
  + Src
    - Src/main/java
    - Src/main/resource
    - Src/test/java
    - Src/test/resource (optional)
  + Target
    - App-0.0.1-Snapshot.jar
  + Pom.xml
  + Dockerfile
* Microservice2
* Microservice3
* Microservice4
* Dockercompose.yml
* Deployment.yaml

ReadMe.MD

Sample files

apiVersion: v1

kind: ConfigMap

metadata:

name: kube-router-cfg

namespace: kube-system

labels:

tier: node

k8s-app: kube-router

data:

cni-conf.json: |

{

"cniVersion":"0.3.0",

"name":"mynet",

"plugins":[

{

"name":"kubernetes",

"type":"bridge",

"bridge":"kube-bridge",

"isDefaultGateway":true,

"ipam":{

"type":"host-local"

}

}

]

}

---

apiVersion: apps/v1

kind: DaemonSet

metadata:

labels:

k8s-app: kube-router

tier: node

name: kube-router

namespace: kube-system

spec:

selector:

matchLabels:

k8s-app: kube-router

tier: node

template:

metadata:

labels:

k8s-app: kube-router

tier: node

spec:

priorityClassName: system-node-critical

serviceAccountName: kube-router

serviceAccount: kube-router

containers:

- name: kube-router

image: docker.io/cloudnativelabs/kube-router

imagePullPolicy: Always

args:

- --run-router=true

- --run-firewall=true

- --run-service-proxy=false

- --bgp-graceful-restart=true

env:

- name: NODE\_NAME

valueFrom:

fieldRef:

fieldPath: spec.nodeName

- name: POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: KUBE\_ROUTER\_CNI\_CONF\_FILE

value: /etc/cni/net.d/10-kuberouter.conflist

livenessProbe:

httpGet:

path: /healthz

port: 20244

initialDelaySeconds: 10

periodSeconds: 3

resources:

requests:

cpu: 250m

memory: 250Mi

securityContext:

privileged: true

volumeMounts:

- name: lib-modules

mountPath: /lib/modules

readOnly: true

- name: cni-conf-dir

mountPath: /etc/cni/net.d

- name: kubeconfig

mountPath: /var/lib/kube-router/kubeconfig

readOnly: true

- name: xtables-lock

mountPath: /run/xtables.lock

readOnly: false

initContainers:

- name: install-cni

image: docker.io/cloudnativelabs/kube-router

imagePullPolicy: Always

command:

- /bin/sh

- -c

- set -e -x;

if [ ! -f /etc/cni/net.d/10-kuberouter.conflist ]; then

if [ -f /etc/cni/net.d/\*.conf ]; then

rm -f /etc/cni/net.d/\*.conf;

fi;

TMP=/etc/cni/net.d/.tmp-kuberouter-cfg;

cp /etc/kube-router/cni-conf.json ${TMP};

mv ${TMP} /etc/cni/net.d/10-kuberouter.conflist;

fi;

if [ -x /usr/local/bin/cni-install ]; then

/usr/local/bin/cni-install;

fi;

volumeMounts:

- mountPath: /etc/cni/net.d

name: cni-conf-dir

- mountPath: /etc/kube-router

name: kube-router-cfg

- name: host-opt

mountPath: /opt

hostNetwork: true

hostPID: true

tolerations:

- effect: NoSchedule

operator: Exists

- key: CriticalAddonsOnly

operator: Exists

- effect: NoExecute

operator: Exists

volumes:

- name: lib-modules

hostPath:

path: /lib/modules

- name: cni-conf-dir

hostPath:

path: /etc/cni/net.d

- name: kube-router-cfg

configMap:

name: kube-router-cfg

- name: kubeconfig

hostPath:

path: /var/lib/kube-router/kubeconfig

- name: xtables-lock

hostPath:

path: /run/xtables.lock

type: FileOrCreate

- name: host-opt

hostPath:

path: /opt

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: kube-router

namespace: kube-system

---

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: kube-router

namespace: kube-system

rules:

- apiGroups:

- ""

resources:

- namespaces

- pods

- services

- nodes

- endpoints

verbs:

- list

- get

- watch

- apiGroups:

- "networking.k8s.io"

resources:

- networkpolicies

verbs:

- list

- get

- watch

- apiGroups:

- extensions

resources:

- networkpolicies

verbs:

- get

- list

- watch

- apiGroups:

- "coordination.k8s.io"

resources:

- leases

verbs:

- get

- create

- update

- apiGroups:

- ""

resources:

- services/status

verbs:

- update

- apiGroups:

- "discovery.k8s.io"

resources:

- endpointslices

verbs:

- get

- list

- watch

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: kube-router

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: kube-router

subjects:

- kind: ServiceAccount

name: kube-router

namespace: kube-system

<https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml>

<https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml>

apiVersion: v1

kind: Service

metadata:

name: my-nginx-svc

labels:

app: nginx

spec:

type: LoadBalancer

ports:

- port: 80

selector:

app: nginx

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-nginx

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

<https://labs.iximiuz.com/playgrounds/k3s>

<https://morningspace.github.io/lab-k8s-playground/>