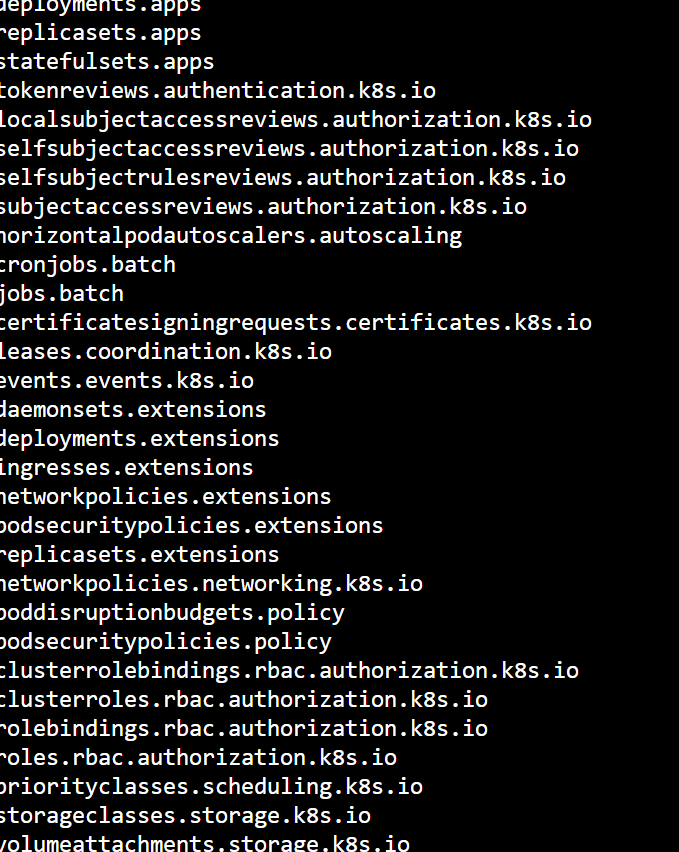
Kubernetes API Primitives :

Kubernetes objects:

1. Pod
2. Node
3. Service
4. Service Account.

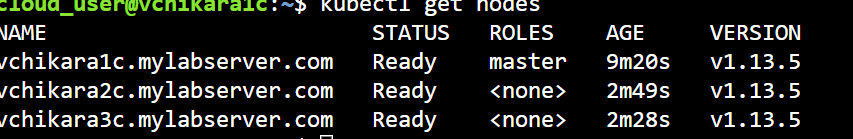
kubectl api-resources -o name



kubectl get pods -n kube-system



kubectl get nodes



kubectl get nodes $node\_name

kubectl get nodes $node\_name -o yaml

kubectl describe node $node\_name

Every Object has a spec and status :

Spec: You provide the spec. This defines the desire state of object.

Status: This is provided by the Kubernetes cluster and contains information about the current state of object.

Kubernetes are often represented in the yaml.

**Pod:**

*Pods are the basic building blocks of any application running in Kubernetes.*

A pod consists of one or more containers and shared by those resources. All container managed by Kubernetes cluster are part of pod.

My-pod.yml

apiVersion: v1

kind: Pod

metadata:

name: my-pod

labels:

app: myapp

spec:

containers:

- name: myapp-container

image: busybox

command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']

kubectl create -f my-pod.yml

Edit the pod

kubectl apply -f my-pod.yml

kubectl edit pod my-pod

Namespace:

kubectl get namespaces

kubectl create ns my-ns

To assign an object to a custom namespace, simply specify its metadata.namespace attribute.

apiVersion: v1

kind: Pod

metadata:

name: my-ns-pod

namespace: my-ns

labels:

app: myapp

spec:

containers:

- name: myapp-container

image: busybox

command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']

**Basic Container Configuration:**

Command and arguments, containerPort

apiVersion: v1

kind: Pod

metadata:

name: my-command-pod

labels:

app: myapp

spec:

containers:

- name: myapp-container

image: busybox

command: ['echo']

restartPolicy: Never

custom arguments :

apiVersion: v1

kind: Pod

metadata:

name: my-args-pod

labels:

app: myapp

spec:

containers:

- name: myapp-container

image: busybox

command: ['echo']

args: ['This is my custom argument']

restartPolicy: Never

containerPort:

apiVersion: v1

kind: Pod

metadata:

name: my-containerport-pod

labels:

app: myapp

spec:

containers:

- name: myapp-container

image: nginx

ports:

- containerPort: 80

Configuration:

Configmaps:

A configmap is a Kubernetes Object that store configuration data in key-value format. This configuration data can be used to configure software running in container, by referencing the Configmap in Podspec.

apiVersion: v1

kind: ConfigMap

metadata:

name: my-config-map

data:

myKey: myValue

anotherKey: anotherValue

Passing ConfigMap data to a container as an environment variable looks like this:

apiVersion: v1

kind: Pod

metadata:

name: my-configmap-pod

spec:

containers:

- name: myapp-container

image: busybox

command: ['sh', '-c', "echo $(MY\_VAR) && sleep 3600"]

env:

- name: MY\_VAR

valueFrom:

configMapKeyRef:

name: my-config-map

key: myKey

It's also possible to pass ConfigMap data to containers, in the form of file using a mounted volume, like so:

apiVersion: v1

kind: Pod

metadata:

name: my-configmap-volume-pod

spec:

containers:

- name: myapp-container

image: busybox

command: ['sh', '-c', "echo $(cat /etc/config/myKey) && sleep 3600"]

volumeMounts:

- name: config-volume

mountPath: /etc/config

volumes:

- name: config-volume

configMap:

name: my-config-map

**Security Context:**

**Create the user and group on the worker node.**

sudo useradd -u 2000 container-user-0

sudo groupadd -g 3000 container-group-0

sudo useradd -u 2001 container-user-1

sudo groupadd -g 3001 container-group-1

sudo mkdir -p /etc/message/

echo "Hello, World!" | sudo tee -a /etc/message/message.txt

sudo chown 2000:3000 /etc/message/message.txt

sudo chmod 640 /etc/message/message.txt

vi my-securitycontext-pod.yml

apiVersion: v1

kind: Pod

metadata:

name: my-securitycontext-pod

spec:

securityContext:

runAsUser: 2000

fsGroup: 3000

containers:

- name: myapp-container

image: busybox

command: ['sh', '-c', "cat /message/message.txt && sleep 3600"]

volumeMounts:

- name: message-volume

mountPath: /message

volumes:

- name: message-volume

hostPath:

path: /etc/message

**check the log you should able to see the message from the file**

kubectl logs my-securitycontext-pod

**Resources:**

apiVersion: v1

kind: Pod

metadata:

name: my-resource-pod

spec:

containers:

- name: myapp-container

image: busybox

command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']

resources:

requests:

memory: "64Mi"

cpu: "250m"

limits:

memory: "128Mi"

cpu: "500m"

**secrets:**

are piece of secret information stored in the Kubernetes cluster such as password, token, keys.

apiVersion: v1

kind: Secret

metadata:

name: my-secret

stringData:

myKey: myPassword

apiVersion: v1

kind: Pod

metadata:

name: my-secret-pod

spec:

containers:

- name: myapp-container

image: busybox

command: ['sh', '-c', "echo Hello, Kubernetes! && sleep 3600"]

env:

- name: MY\_PASSWORD

valueFrom:

secretKeyRef:

name: my-secret

key: myKey

ServiceAcccounts:

kubectl create serviceaccount my-serviceaccount

apiVersion: v1

kind: Pod

metadata:

name: my-serviceaccount-pod

spec:

serviceAccountName: my-serviceaccount

containers:

- name: myapp-container

image: busybox

command: ['sh', '-c', "echo Hello, Kubernetes! && sleep 3600"]