Kubernetes Self notes from the Official Documents of K8s

# 1. Node:

* For Creating Nodes. The syntax of the yaml file should be as follows.

Code:

{

**"kind"**: "Node",

**"apiVersion"**: "v1",

**"metadata"**: {

**"name"**: "Testing Node",

**"labels"**: {

**"name"**: "my-first-k8s-node"

}

}

}

* Two Nodes cannot have the same name at the same time. Kubernetes also assumes that a resource with the same name is the same object.
* Node monitor grace period = 40s
* Pod eviction timeout = 5min
* Node Monitor period = 5s

# 2. ETCD Database:

* ETCD is a key value database used in keeping the data of a Kubernetes cluster.

# 3. Replica sets:

* A **ReplicaSet** in Kubernetes ensures a specified number of identical pod replicas are running at all times, maintaining high availability and fault tolerance.
* **apiVersion**: apps/v1
* **kind**: ReplicaSet
* **metadata**:
* **name**: frontend
* **labels**:
* **app**: guestbook
* **tier**: frontend
* **spec**:
* *# modify replicas according to your case*
* **replicas**: 3
* **selector**:
* **matchLabels**:
* **tier**: frontend
* **template**:
* **metadata**:
* **labels**:
* **tier**: frontend
* **spec**:
* **containers**:
* - **name**: php-redis
* **image**: us-docker.pkg.dev/google-samples/containers/gke/gb-frontend:v5
* Adding ‘ Selectors ’ section to the definition file, makes ReplicaSets different from Replication Controller
* Rest for listing ReplicaSet created you can use “ Kubectl get ReplicaSet <ReplicaSet-name>”
* For scaling replicas from single command use -> “ Kubectl scale –-replicas=6” -f replicaset-defination.yml
* If stuck with any syntax or parameters in Kubernetes, Then just pass on “ Kubectl explain Replicaset"

# Networking Concept – Kubernetes:

* Imp Commands : SWITCH Routing

1. Ip Link -> Is to list and modify interfaces on the host  
2. Ip addr -> Is to see Ip address assigned to those interfaces

3. Ip addr add < IP/24 > dev eth0 -> Is used to set the Ip address on the interfaces  
4. Ip route -> Is used to view the Route Table  
5. Ip route add < IP Destination where you want to reach > via < IP gateway through which the packets should be sent > -> command is used to add a route to the routing table in Linux

6. cat /proc/sys/net/ipv4/ip\_forward -> The command is used to check the current status of **IP forwarding** in the Linux kernel.  
< 0 >: IP forwarding is disabled. The system won't forward packets between interfaces.

< 1 >: IP forwarding is enabled. The system will forward packets between interfaces.

* Imp Commands : DNS  
    
  1. Ip netns add < Namespace-name > 🡪 To create a new Network Namespace on a Linux host

2. Ip netns 🡪 To list the namespaces

3. Ip Link 🡪 To list the interfaces on the host

4. Ip netns exec < Namespace name > ip link 🡪 is used to list network interfaces within a specific network namespace in Linux.

5. Ip link add veth-red type veth peer name veth-blue 🡪 This command creates a pair of **virtual Ethernet (veth) interfaces** that are interconnected. A veth pair is like a virtual cable: whatever is sent into one interface (e.g., veth-red) is received by its peer interface (e.g., veth-blue).

6. ip link set <Virtual ethernet Pair> netns <Namespace> 🡪 This command moves the virtual Ethernet interface veth-red into the network namespace named red

7. Ip -n <Namespace> addr add < IP > dev <Virtual ethernet Pair> 🡪 For defining or adding an Ip address to the newly created interface.

8. Ip link add < Name > type bridge 🡪 For creating an internal bridge we have to add a new interface to the host

9. Ip link -n red link del veth-red 🡪 To delete the cable which was set up at point 5

10. Now similarly as we had connected the different hosts with help of the cable , now we have to connect same hosts to bridge interface using same commands from 5, 6, 7 and establish a cable and link it to the bridge.

11. A bridge network is like an interface to the host but a Switch to the interfaces(Namespaces or Containers) within the host.

* Container Network Interface : CNI  
    
  - All the network plugins are installed in the directory /opt/cni/bin , so that’s where container Runtime find the plugin  
    
  - But which plugin to use and how to use it is configured in the directory /etc/cni/net.d .

- This is the format defined by the CNI standard for plugin configuration file.   
  


1. netstat –help 🡪 The netstat command is a network troubleshooting tool used to display network connections, routing tables, interface statistics, and other network-related information

2. Pod Networking in K8 🡪 Networking Model as per K8 standard are as following :   
- Every POD should have an IP address

- Every POD should be able to communicate with every pod on the same node

- Similarly , Every POD should be able to connect to other POD on other nodes without NAT

3.

# Storage Section:

* When you install docker in the machine, it installs folder structure at directory /var/lib/docker. This is where docker saves all of its data.
* For Creating Persistent Volume in K8, the definition file is as follows:  
  
* In Kubernetes, **Persistent Volumes (PVs)** and **Persistent Volume Claims (PVCs)** are used to manage and abstract storage resources for applications running in the cluster. They decouple the storage configuration from the pods, enabling better management and flexibility.
* A **Persistent Volume** is a piece of storage in the cluster that has been provisioned by an administrator or dynamically created using storage classes. A PV is a cluster-wide resource that can be used by any pod within the cluster.
* A **Persistent Volume Claim** is a request for storage by a user or an application. It specifies the desired size, access mode, and, optionally, a storage class. A PVC binds to a suitable PV that meets its requirements.
* The definition file for creating a PV claim is as follows:   
  
* Also, Once you create a PVC use it in a POD definition file by specifying the PVC Claim name under persistentVolumeClaim section in the volumes section like this:

apiVersion: v1

kind: Pod

metadata:

name: mypod

spec:

containers:

- name: myfrontend

image: nginx

volumeMounts:

- mountPath: "/var/www/html"

name: mypd

volumes:

- name: mypd

persistentVolumeClaim:

claimName: myclaim

* The same is true for ReplicaSets or Deployments. Add this to the pod template section of a Deployment on ReplicaSet.

# Designing A Kubernetes Cluster.

* Questions to Ask before creating a Cluster:  
  \* Purpose?
* Education
* Development & Testing
* Hosting Production Applications

• Cloud or OnPrem?

• Workloads

* How many?
* What kind?

- Web

- Big Data/Analytics

• Application Resource Requirements?

- CPU Intensive

- Memory Intensive

• Traffic?

- Heavy traffic

- Burst Traffic

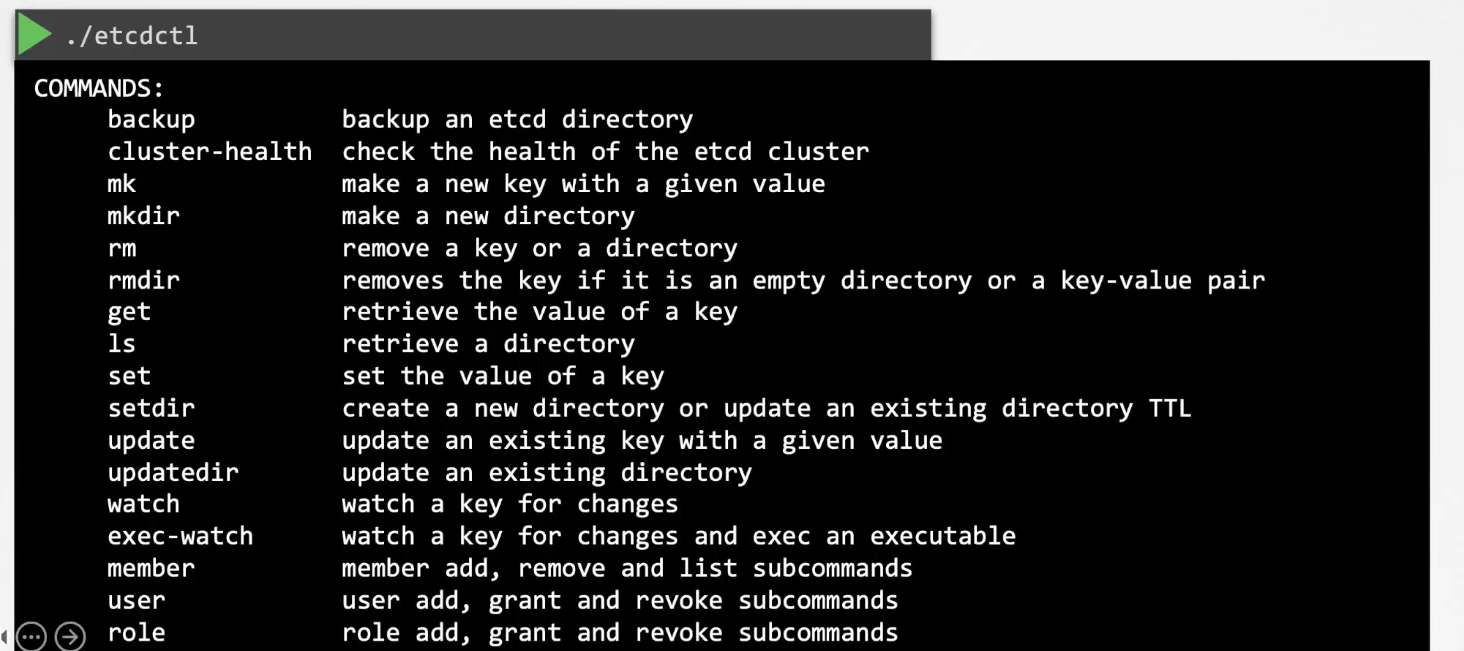
Commands

1. For creating Pod -> Kubectl create -f file.yml
2. To See list of pods -> Kubectl get pods
3. To see detailed information about the pod -> Kubectl describe pod <pod-name>
4. For deleting a pod -> Kubectl delete pod <pod-name>
5. To get a direct yaml file without creating it or writing it from scratch, this can be achieved by dry run process, for example,

“ Kubectl run <Pod-name> --image=<image-name> --dry-run=client -o yaml “

After using this you can get a direct copy of a yaml file which you can further edit using Vim or nano editor in machine.

1. You can direct copy the code from yaml file with dry run method to a file by doing following : “ Kubectl run Nginx –-image=nginx –dry-run=client -o yaml > MyPod-defination.yml
2. For updating the same file with changes in code use ->   
   “ Kubectl replace -f <file-name,yml> ”
3. For editing the existing yaml file -> Kubectl edit pod examplePod.yaml

* Commands used for ETCD in K8 are : 
* ETCD listens on port 2379
* If we setup our K8 cluster using Kubeadm, then Kubeadm deploys ETCD server as a pod in -kube-system namespace.   
  If we use following command “ Kubectl get pods -n -kube-system ”, Then we would be able to see all the pods deployed by Kubeadm while creating cluster.
* Whereas the commands are different in version 3
* etcdctl snapshot save
* etcdctl endpoint health
* etcdctl get
* etcdctl put
* To set the right version of API set the environment variable ETCDCTL\_API command

export ETCDCTL\_API=3

# 3. Kube-API server:

* Similar to ETCD, if we deploy cluster using Kubeadm, then Kube-api server is deployed as a pod in -kube-system namespace.
* You can see the other options of the kube-api server with help of below command “ cat /etc/Kubernetes/manifests/kube-apiserver.yaml “
* If the cluster is not deployed using KubeAdm and is deployed normally, then we can view the same options for Kub-api server by following command

“ cat /etc/system/system/kube-apiserver.service ”

* Can see the running process of Kub-api server by using   
  “ ps aux | grep kube-apiserver ”