Kubernetes Architecture

Kubernetes follows a **Master-Worker Node** architecture, where the **Control Plane (Master Node)** manages the **Worker Nodes** to run containerized applications.

Control Plane /Master Node
The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).
Control plane components can be run on any machine in the cluster. Do not run user containers on this machine.
Node Components / Worker Nodes
Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

1. Master Node: The master node is responsible for managing the cluster and coordinating the overall state of the system. It includes the following components:

a. API Server: The API server is the central control point for all interactions with the cluster. It exposes the Kubernetes API and handles requests from users and other components.

b. Scheduler: The scheduler is responsible for assigning workloads (pods) to individual worker nodes based on resource requirements, constraints, and other policies.

c. Controller Manager: The controller manager runs various controllers that monitor the cluster state and drive it towards the desired state. Examples include the replication controller, node controller, and service controller.

d. etcd: etcd is a distributed key-value store used by Kubernetes to store cluster state and configuration data.

KUBERNETES COMMANDS:

Create a pod using run command

```
$ kubectl run <pod-name> --image=<image-name> --port=<container-port>
$ kubectl run my-pod --image=nginx --port=80
```

2. View all the pods

(In default namespace)

```
$ kubectl get pods
```

(In All namespace)

```
$ kubectl get pods -A
# For a specific namespace
$ kubectl get pods -n kube-system

# For a specific type
$ kubectl get pods <pod-name>
$ kubectl get pods <pod-name> -o wide
$ kubectl get pods <pod-name> -o yaml
$ kubectl get pods <pod-name> -o json
```

3. Describe a pod (View Pod details)

```
$ kubectl describe pod <pod-name>
$ kubectl describe pod my-pod
```

4. View Logs of a pod

```
$ kubectl logs <pod-name>
$ kubectl logs my-pod
$ kubectl exec <pod-name> -- <command>
```

POD:

The basic building block of Kubernetes. A pod represents a single instance of a running process within the cluster. It can encapsulate one or more containers that share the same network and storage resources.

Pod.YML:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  labels:
      app: my-web-app

spec:
  containers:
    - name: nginx-container
      image: ashilin20/app:latest
      ports:
        - containerPort: 80
```

Services (short name = svc):

Service is an abstraction that defines a logical set of pods and a policy to access them. Services enable network connectivity and load balancing to the pods that are part of the service, allowing other components within or outside the cluster to interact with the application.

Service Types: Kubernetes supports different types of services:

1. NodePort: Exposes the service on a static port on each selected node's IP. This type makes the service accessible from outside the cluster by the <NodeIP>:<NodePort> combination.

2. ClusterIP: Exposes the service on a cluster-internal IP. This type makes the service only reachable within the cluster.

3. LoadBalancer: Creates an external load balancer in cloud environments, which routes traffic to the service.

DEPLOY COMMANDS:

2. Create Deployment by executing above YAML file

```
$ kubectl create -f web-deploy.yml
# Do necessary modifications if exist, else create new
$ kubectl create -f web-deploy.yml
# Completely Modify Pod Template
$ kubectl replace –f web-deploy.yml
```

3. View Deployments

```
$ kubectl get deployments
$ kubectl get deploy
$ kubectl get deploy -o wide
$ kubectl get deploy <deployment-name> -o json
$ kubectl get deploy <deployment-name> -o yaml
```

4. View Deployment Description

```
$ kubectl describe deploy <deployment-name>
```

5. We can modify generated/updated YAML file

```
$ kubectl edit deploy <deployment-name>
## change replicas: count to any other value then (ESC):wq

# We can modify our YAML file and then execute apply command
$ kubectl apply -f web-deploy.yml

## We can Even scale using command also
$ kubectl  scale  deploy  <deployment-name>   --replicas=<desired-replica-count>
```

6. Delete Deployment

```
$ kubectl delete deploy <deployment-name>
$ kubectl delete -f web-deploy.yml
```

Deploy.yml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deploy
  labels:
    name: my-deploy
spec:
  replicas: 4
  selector:
    matchLabels:
      apptype: web-backend
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        apptype: web-backend
    spec:
      containers:
      - name: my-app
        image: ashilin20/app:latest
        ports:
            - containerPort: 7070
```

REPLICA COMMANDS:

```
Create ReplicaSet by executing above YAML file
$ kubectl create -f rs-test.yml
# Do necessary modifications if exist, else create new
$ kubectl apply -f rs-test.yml
# Completely Modify Pod Template
$ kubectl replace –f rs-test.yml
```

3. View ReplicaSets

```
$ kubectl get replicasets
$ kubectl get rs
$ kubectl get rs –o wide
$ kubectl get rs <replica-set-name> –o json
$ kubectl get rs <replica-set-name> –o yaml
```

4. View ReplicaSet Description

```
$ kubectl describe rs <replica-set-name>
```

5. We can modify generated/updated YAML file

```
$ kubectl edit rs <replica-set-name>
## change replicas: count to any other value then (ESC):wq

# We can modify our YAML file and then execute apply command
$ kubectl apply -f rs-test.yml

## We can Even scale using command also
$ kubectl scale replicaset <replicaset-name> --replicas=<desired-replica-count>
```

Rs-test.yml:

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-rs
  labels:
    name: my-rs
spec:
  replicas: 4
  selector:
    matchLabels:
      apptype: web-backend
  template:
    metadata:
      labels:
        apptype: web-backend
    spec:
      containers:
      - name: my-app
        image: ashilin20/app:latest
        ports:
          - containerPort: 8081
```

NAMESPACE:

Namespace (short name = ns):
namespace is a virtual cluster or logical partition within a cluster that provides a way to organize and isolate resources. It allows multiple teams or projects to share the same physical cluster while maintaining resource separation and access control.

Pod-ns.yml:

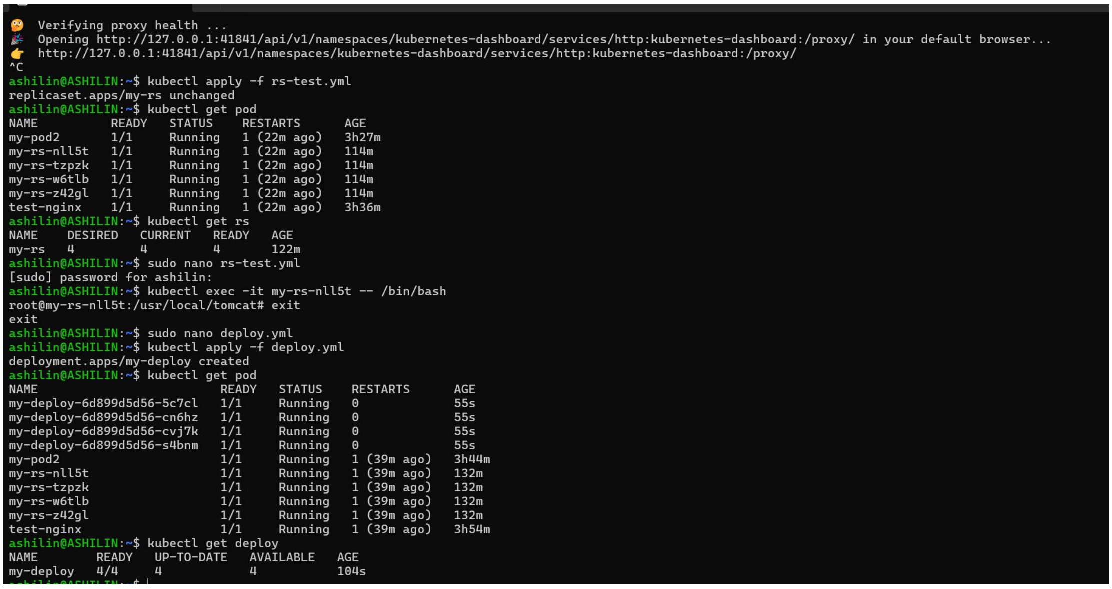
```
apiVersion: v1
kind: Pod
metadata:
  name: my-deploy
  namespace: mydeploy
spec:
  containers:
  - name: my-container
    image: nginx:latest
```

Ns-test.yml:

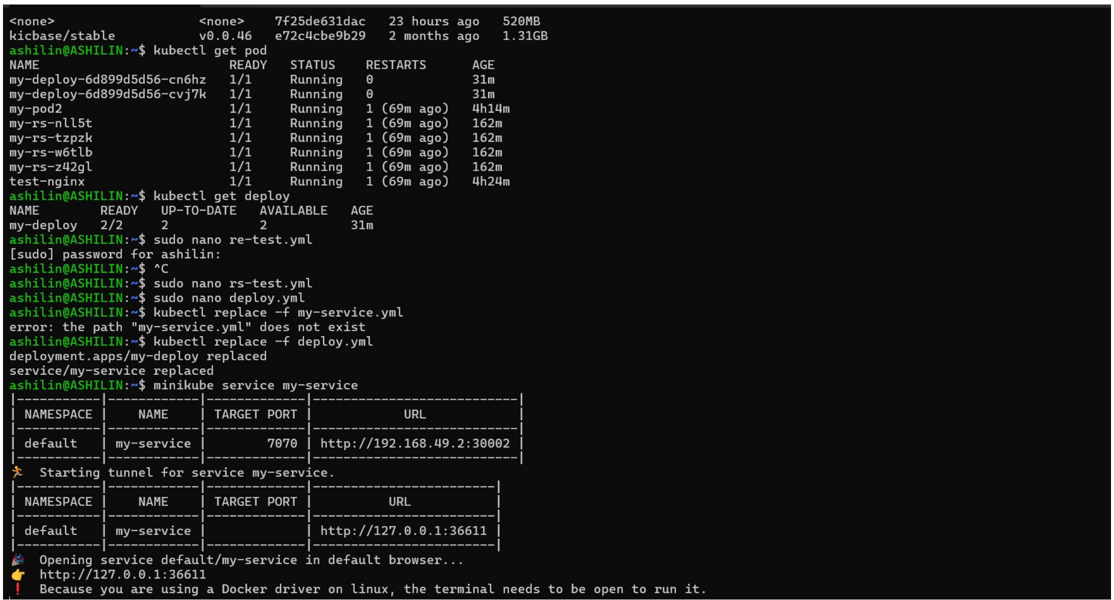
```
apiVersion: v1
kind: Namespace
metadata:
  name: my-demo-ns
```

Namespace commands:

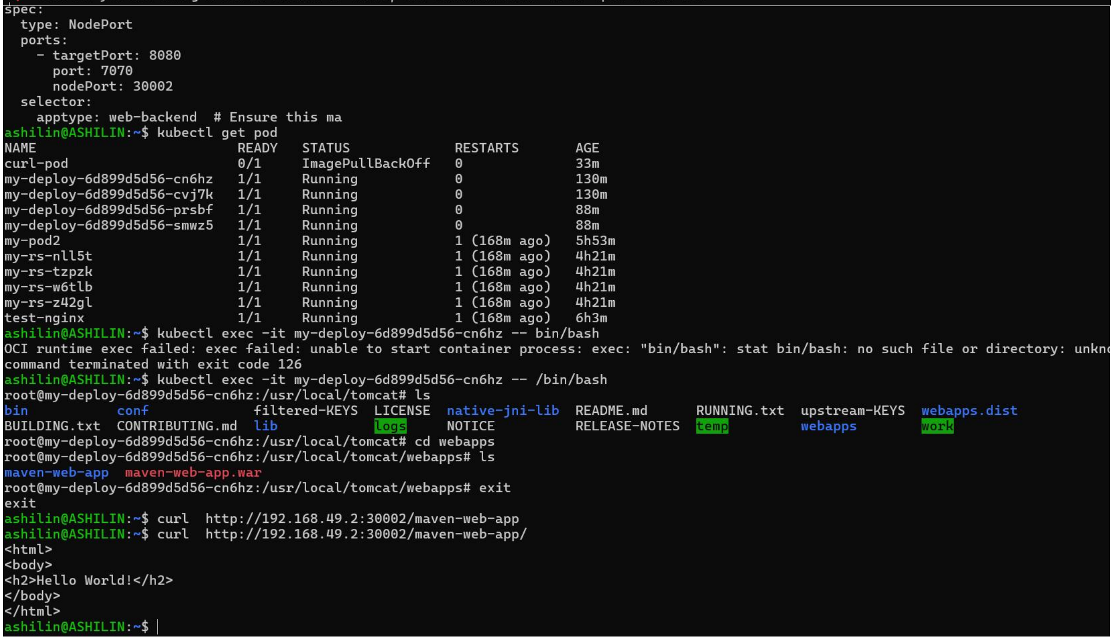
```
 To create a namespace:
$ kubectl create namespace <namespace-name>
$ kubectl create ns my-bank
# To switch to a specific namespace: (make this as default type)
$ kubectl config set-context --current --namespace=<namespace-name>
# To list all namespaces:
$ kubectl get namespaces
# To get resources within a specific namespace:
$ kubectl get <resource-type> -n <namespace-name>
$ kubectl get deploy -n my-bank
$ kubectl get deploy --namespace my-bank
$ kubectl get all --namespace my-bank
# To delete a namespace and all associated resources:
$ kubectl delete namespace <namespace-name>
$ kubectl delete ns my-bank
```



```
🤔  Verifying proxy health ...
🎉  Opening http://127.0.0.1:41841/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...
👉  http://127.0.0.1:41841/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/
^C
ashilin@ASHILIN:~$ kubectl apply -f rs-test.yml
replicaset.apps/my-rs unchanged
ashilin@ASHILIN:~$ kubectl get pod
NAME           READY   STATUS    RESTARTS      AGE
my-pod2        1/1     Running   1 (22m ago)   3h27m
my-rs-nll5t    1/1     Running   1 (22m ago)   114m
my-rs-tzpzk    1/1     Running   1 (22m ago)   114m
my-rs-w6tlb    1/1     Running   1 (22m ago)   114m
my-rs-z42gl    1/1     Running   1 (22m ago)   114m
test-nginx     1/1     Running   1 (22m ago)   3h36m
ashilin@ASHILIN:~$ kubectl get rs
NAME    DESIRED   CURRENT   READY   AGE
my-rs   4         4         4       122m
ashilin@ASHILIN:~$ sudo nano rs-test.yml
[sudo] password for ashilin:
ashilin@ASHILIN:~$ kubectl exec -it my-rs-nll5t -- /bin/bash
root@my-rs-nll5t:/usr/local/tomcat# exit
exit
ashilin@ASHILIN:~$ sudo nano deploy.yml
ashilin@ASHILIN:~$ kubectl apply -f deploy.yml
deployment.apps/my-deploy created
ashilin@ASHILIN:~$ kubectl get pod
NAME                         READY   STATUS    RESTARTS      AGE
my-deploy-6d899d5d56-5c7cl   1/1     Running   0             55s
my-deploy-6d899d5d56-cn6hz   1/1     Running   0             55s
my-deploy-6d899d5d56-cvj7k   1/1     Running   0             55s
my-deploy-6d899d5d56-s4bnm   1/1     Running   0             55s
my-pod2                      1/1     Running   1 (39m ago)   3h44m
my-rs-nll5t                  1/1     Running   1 (39m ago)   132m
my-rs-tzpzk                  1/1     Running   1 (39m ago)   132m
my-rs-w6tlb                  1/1     Running   1 (39m ago)   132m
my-rs-z42gl                  1/1     Running   1 (39m ago)   132m
test-nginx                   1/1     Running   1 (39m ago)   3h54m
ashilin@ASHILIN:~$ kubectl get deploy
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
my-deploy   4/4     4            4           104s
ashilin@ASHILIN:~$
```



```
<none>                   <none>    7f25de631dac   23 hours ago   520MB
kicbase/stable           v0.0.46   e72c4cbe9b29   2 months ago   1.31GB
ashilin@ASHILIN:~$ kubectl get pod
NAME                         READY   STATUS    RESTARTS      AGE
my-deploy-6d899d5d56-cn6hz   1/1     Running   0             31m
my-deploy-6d899d5d56-cvj7k   1/1     Running   0             31m
my-pod2                      1/1     Running   1 (69m ago)   4h14m
my-rs-nll5t                  1/1     Running   1 (69m ago)   162m
my-rs-tzpzk                  1/1     Running   1 (69m ago)   162m
my-rs-w6tlb                  1/1     Running   1 (69m ago)   162m
my-rs-z42gl                  1/1     Running   1 (69m ago)   162m
test-nginx                   1/1     Running   1 (69m ago)   4h24m
ashilin@ASHILIN:~$ kubectl get deploy
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
my-deploy   2/2     2            2           31m
ashilin@ASHILIN:~$ sudo nano re-test.yml
[sudo] password for ashilin:
ashilin@ASHILIN:~$ ^C
ashilin@ASHILIN:~$ sudo nano rs-test.yml
ashilin@ASHILIN:~$ sudo nano deploy.yml
ashilin@ASHILIN:~$ kubectl replace -f my-service.yml
error: the path "my-service.yml" does not exist
ashilin@ASHILIN:~$ kubectl replace -f deploy.yml
deployment.apps/my-deploy replaced
service/my-service replaced
ashilin@ASHILIN:~$ minikube service my-service
|-----------|------------|-------------|---------------------------|
| NAMESPACE |    NAME    | TARGET PORT |            URL            |
|-----------|------------|-------------|---------------------------|
| default   | my-service |        7070 | http://192.168.49.2:30002 |
|-----------|------------|-------------|---------------------------|
🏃  Starting tunnel for service my-service.
|-----------|------------|-------------|------------------------|
| NAMESPACE |    NAME    | TARGET PORT |          URL           |
|-----------|------------|-------------|------------------------|
| default   | my-service |             | http://127.0.0.1:36611 |
|-----------|------------|-------------|------------------------|
🎉  Opening service default/my-service in default browser...
👉  http://127.0.0.1:36611
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```



```
spec:
  type: NodePort
  ports:
    - targetPort: 8080
      port: 7070
      nodePort: 30002
  selector:
    apptype: web-backend  # Ensure this ma
ashilin@ASHILIN:~$ kubectl get pod
NAME                         READY   STATUS             RESTARTS       AGE
curl-pod                     0/1     ImagePullBackOff   0              33m
my-deploy-6d899d5d56-cn6hz   1/1     Running            0              130m
my-deploy-6d899d5d56-cvj7k   1/1     Running            0              130m
my-deploy-6d899d5d56-prsbf   1/1     Running            0              88m
my-deploy-6d899d5d56-smwz5   1/1     Running            0              88m
my-pod2                      1/1     Running            1 (168m ago)   5h53m
my-rs-nll5t                  1/1     Running            1 (168m ago)   4h21m
my-rs-tzpzk                  1/1     Running            1 (168m ago)   4h21m
my-rs-w6tlb                  1/1     Running            1 (168m ago)   4h21m
my-rs-z42gl                  1/1     Running            1 (168m ago)   4h21m
test-nginx                   1/1     Running            1 (168m ago)   6h3m
ashilin@ASHILIN:~$ kubectl exec -it my-deploy-6d899d5d56-cn6hz -- bin/bash
OCI runtime exec failed: exec failed: unable to start container process: exec: "bin/bash": stat bin/bash: no such file or directory: unkno
command terminated with exit code 126
ashilin@ASHILIN:~$ kubectl exec -it my-deploy-6d899d5d56-cn6hz -- /bin/bash
root@my-deploy-6d899d5d56-cn6hz:/usr/local/tomcat# ls
bin           conf             filtered-KEYS  LICENSE  native-jni-lib  README.md      RUNNING.txt  upstream-KEYS  webapps.dist
BUILDING.txt  CONTRIBUTING.md  lib            logs     NOTICE          RELEASE-NOTES  temp         webapps        work
root@my-deploy-6d899d5d56-cn6hz:/usr/local/tomcat# cd webapps
root@my-deploy-6d899d5d56-cn6hz:/usr/local/tomcat/webapps# ls
maven-web-app  maven-web-app.war
root@my-deploy-6d899d5d56-cn6hz:/usr/local/tomcat/webapps# exit
exit
ashilin@ASHILIN:~$ curl  http://192.168.49.2:30002/maven-web-app
ashilin@ASHILIN:~$ curl  http://192.168.49.2:30002/maven-web-app/
<html>
<body>
<h2>Hello World!</h2>
</body>
</html>
ashilin@ASHILIN:~$
```