Kubernetes Architecture

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

---

1 Control Plane (Master Node)

The **Control Plane** is responsible for cluster management, scheduling, and monitoring.

Component	Description
API Server (kube-apiserver)	Acts as the entry point for all Kubernetes commands (kubectl, API requests).
Scheduler (kube-scheduler)	Assigns workloads (Pods) to worker nodes based on resource availability.
Controller Manager (kube-controller-manager)	Manages cluster controllers like Node Controller, Replication Controller, etc.
etcd	Key-value store that maintains cluster state, configuration, and metadata.
Cloud Controller Manager	Manages cloud provider-specific integrations (like AWS, GCP, Azure).

---

2 Worker Nodes

Worker nodes run application workloads. Each node contains:

Component	Description
Kubelet	Ensures containers are running in a Pod, communicates with the API server.
Container Runtime	Runs containers (Docker, containerd, CRI-O, etc.).
Kube Proxy	Manages networking between Pods and enables communication inside/outside the cluster.
Pods	The smallest deployable unit in Kubernetes, containing one or more containers.

---

3 Kubernetes Networking

- **ClusterIP**: Internal communication within the cluster.
- **NodePort**: Exposes services externally via a static port.
- **LoadBalancer**: Routes traffic through a cloud-based load balancer.
- **Ingress**: Manages external HTTP/S access to services.

4 Kubernetes Objects

Object	Description
Pods	The smallest unit in Kubernetes, running containerized applications.
Services	Exposes Pods inside/outside the cluster.
Deployments	Manages Pod scaling and rolling updates.
ConfigMaps & Secrets	Stores configuration data and sensitive information.
Namespaces	Isolates resources within a cluster for better management.

5 Flow of Kubernetes Deployment

1. **kubectl apply -f deployment.yml** → Sends request to **API Server**
2. **API Server** validates and stores the request in **etcd**
3. **Scheduler** assigns Pods to **Worker Nodes**
4. **Kubelet** on Worker Node pulls the container image and runs the Pod
5. **Kube Proxy** manages networking, making the application accessible

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.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
```

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
```

Pod-ns.yml:

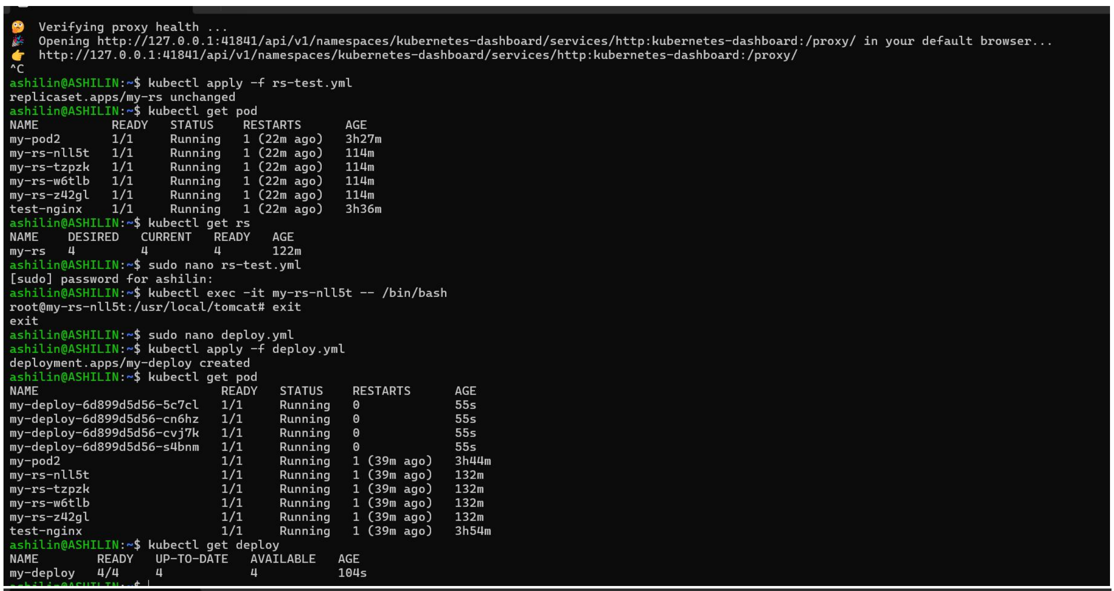
```
apiVersion: v1
kind: Pod
metadata:
  name: my-deploy
  namespace: mydeploy
spec:
  containers:
  - name: my-container
    image: nginx:latest
```

Ns-test.yml:

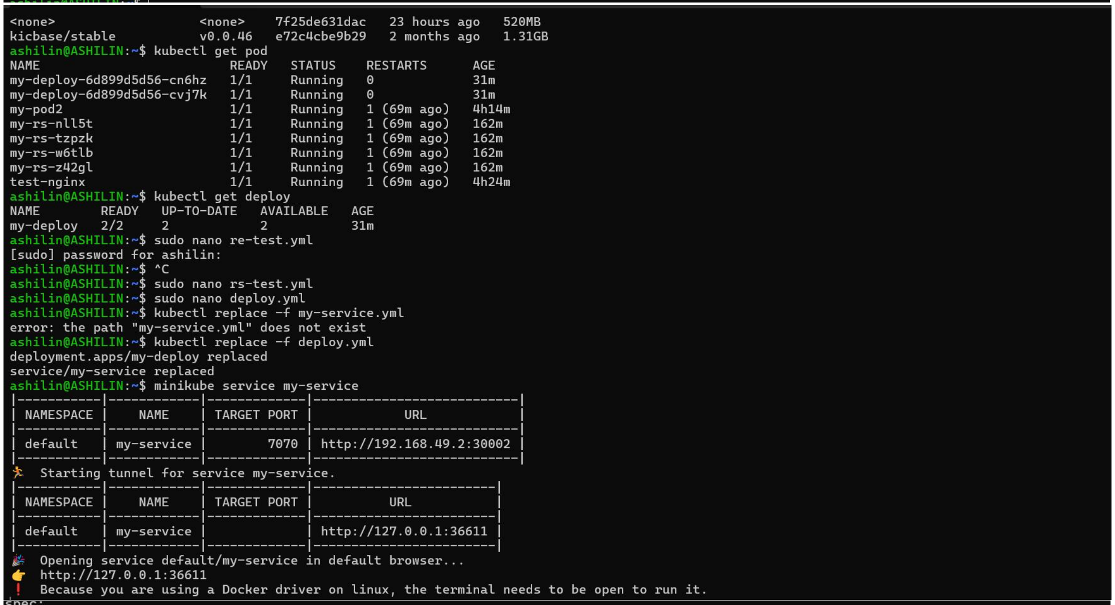
```
apiVersion: v1
kind: Namespace
metadata:
  name: my-demo-ns
```

Rs-test.yml:

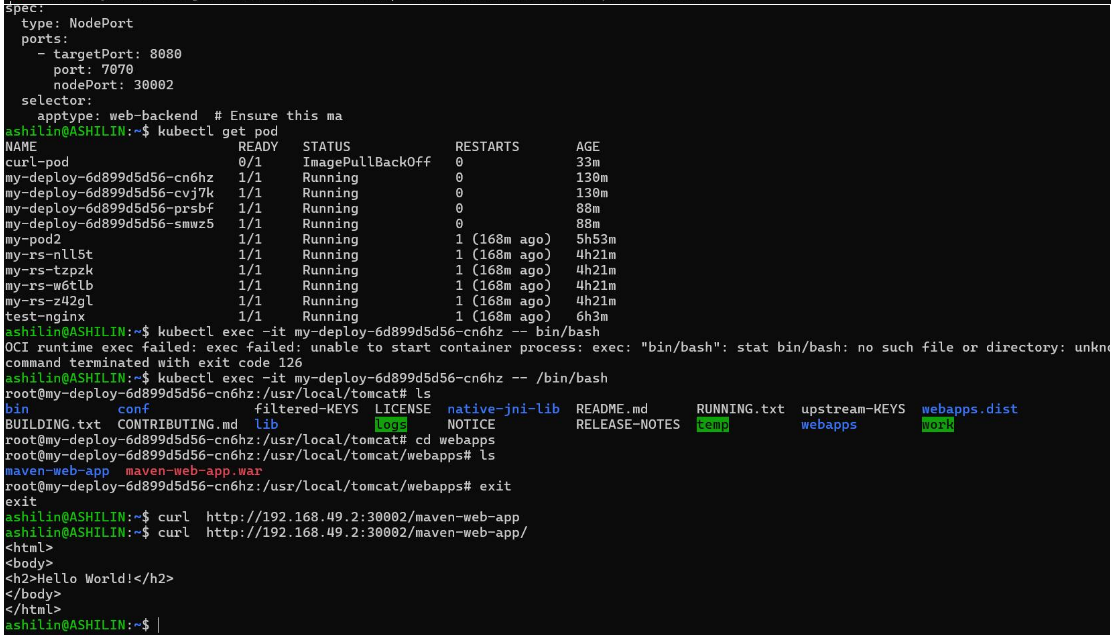
```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-rs
  labels:
    name: my-rs
spec:
  replicas: 4
  selector:
    matchLabels:
      apptype: web-backend
  template:
    metadata:
      labels:
        apptype: web-backend
    spec:
      containers:
      - name: my-app
        image: ashilin20/app:latest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



```
😳  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
```



```
<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: unkn
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:~$ |
```