End-to-End Kubernetes Guide

Deployments, Services, Ingress, and Real-time Examples

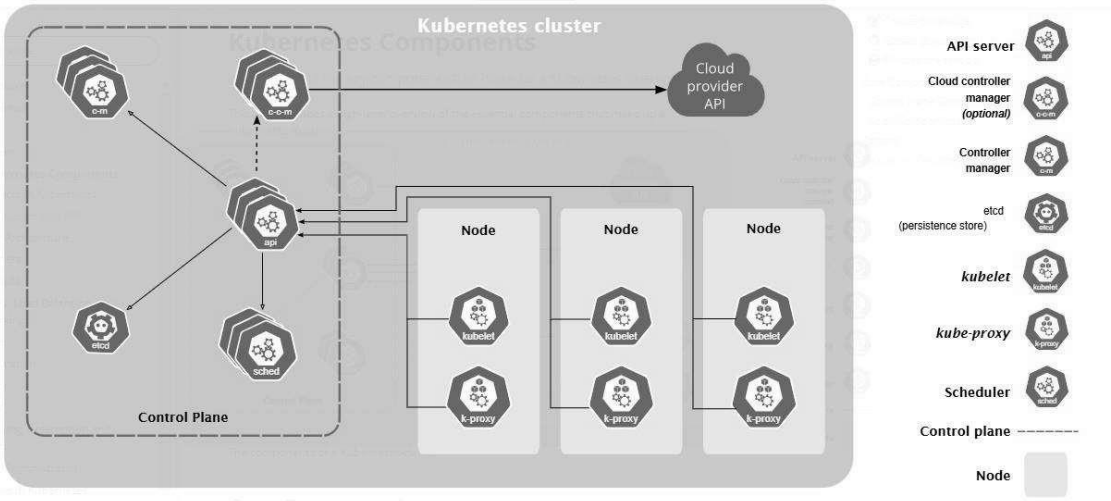
Version 1.0 | September 2025

Index

Why Kubernetes –

- Auto start of container
- Health check
- Autoscaling
- Load balancing
- Network
- Auto-healing
- Auto-node allocating
- Updates/new release/deployment
- Secrete / config (master node)

Kubernetes architecture –



Kubernetes Objects –

- Pod
- Service
- Volume
- Network
- Configmap
- Secrete
- Ingress

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
    containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

1. Pod

- **Definition**: The smallest and simplest Kubernetes object, representing a single instance of a running process in your cluster.
- **Purpose**: Groups one or more containers (e.g., Docker containers) that share the same network namespace and storage volumes.
- **Key Features**:
    - Containers in a pod share the same IP address and port space.
    - Pods are ephemeral; when a pod is deleted, a new pod may be created with a different IP.
- Example –

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: my-container
    image: nginx
```

2. Service

- **Definition**: A stable, permanent endpoint to access one or more pods.
- **Purpose**: Provides load balancing and service discovery for pods.
- **Key Features**:
  - Types: ClusterIP (default), NodePort, LoadBalancer, ExternalName.
  - Abstracts away the dynamic nature of pod IPs.
- **Example**:

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
  type: ClusterIP
```

3. Volume

- **Definition**: A way to provide persistent storage to containers running inside a pod.
- **Purpose**: Persist data beyond the lifetime of a pod and share data between containers in a pod.
- **Key Features**:
  - Types include emptyDir, hostPath, NFS, ConfigMap, Secret, PersistentVolume, PersistentVolumeClaim.
  - Decouples storage from pod lifecycle.
- **Example**

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: my-container
    image: nginx
    volumeMounts:
    - mountPath: "/data"
      name: my-volume
  volumes:
  - name: my-volume
    emptyDir: {}
```

4. Network

- **Definition**: Refers to Kubernetes networking objects and policies that manage communication within and outside the cluster.
- **Purpose**: Allow communication between pods, services, and external resources.
- **Key Features**:
  - Kubernetes assigns each pod a unique IP address.
  - Supports network policies for controlling traffic flow.
  - Uses CNI (Container Network Interface) plugins like Calico, Flannel, or WeaveNet.
- **Example (Network Policy:**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-traffic
spec:
```

```
  podSelector:
    matchLabels:
      app: my-app
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: allowed-app
```

5. ConfigMap

- **Definition**: Stores configuration data as key-value pairs.
- **Purpose**: Allows separation of configuration from application code.
- **Key Features**:
  - Inject configuration data into containers as environment variables, command-line arguments, or mounted files.
  - Non-sensitive data storage
- **Example**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  key1: value1
  key2: value2
```

7. Ingress

- **Definition**: Manages external HTTP/S access to services within the cluster.
- **Purpose:** Acts as a reverse proxy and load balancer, enabling custom URLs and SSL/TLS termination.
- **Key Features:**
    - Provides routing rules based on hostnames and paths.
    - Requires an ingress controller (e.g., NGINX, Traefik) to function.
- **Example :**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - host: example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: my-service
            port:
              number: 80
```

These objects together provide the building blocks for deploying, scaling, and managing applications in Kubernetes.

Kubernetes Installation –

Launching two ubuntu server named masternode (t2. small) and workernode(t2. micro)

On masternode and workernode - ->

sudo apt-get update

sudo apt-get install docker.io

sudo apt-get install -y apt-transport-https ca-certificates curl gnupg

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

sudo chmod 644 /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo chmod 644 /etc/apt/sources.list.d/kubernetes.list

sudo apt-get install -y kubectl kubeadm kubelet

On masternode - - >

sudo kubeadm init --ignore-preflight-errors=all

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

kubeadm token create --print-join-command
[add port 6443 in both master and worker node]

On Worker - >

Use kubeadm join command with token (copy the command from master and paste on worker)

sudo (command)

On Master - ->

kubectl get nodes

Pod creation –

kubectl get nodes

```
ubuntu@ip-172-31-21-116:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-21-116   Ready    control-plane   30h   v1.30.7
ip-172-31-94-46    Ready    <none>          30h   v1.30.7
```

kubectl get pods -n kube-system

```
ubuntu@ip-172-31-21-116:~$ kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS        AGE
calico-kube-controllers-6cdb97b867-zk6tr   1/1     Running   2 (3m8s ago)    30h
calico-node-8k8bg                          0/1     Running   9 (2m5s ago)    30h
calico-node-j5mqz                          0/1     Running   4 (30h ago)     30h
coredns-55cb58b774-84665                   1/1     Running   1 (3m8s ago)    30h
coredns-55cb58b774-zgx4j                   1/1     Running   1 (3m6s ago)    30h
etcd-ip-172-31-21-116                      1/1     Running   1 (3m8s ago)    30h
kube-apiserver-ip-172-31-21-116            1/1     Running   1 (3m8s ago)    30h
kube-controller-manager-ip-172-31-21-116   1/1     Running   1 (3m8s ago)    30h
kube-proxy-96fds                           1/1     Running   9 (69s ago)     30h
kube-proxy-d572p                           1/1     Running   7 (63s ago)     30h
kube-scheduler-ip-172-31-21-116            1/1     Running   1 (3m8s ago)    30h
```

mkdir nginxpod

cd nginxpod

nano nginxpod.yml

```
 GNU nano 7.2                                  nginxpod.yml
apiVersion: v1
kind: Pod
metadata:
 name: nginxpod
 labels:
  app: myapp
spec:
 containers:
  - name: nginxcontainer
    image: nginx
    ports:
     - containerPort: 80
```

kubectl get pods

```
ubuntu@ip-172-31-21-116:~/nginxpod$ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
nginxpod   1/1     Running   0          6m17s
```

kubectl describe pod podname

kubectl logs pod_name

kubectl exec -it nginxpod - - /bin/bash

```
ubuntu@ip-172-31-21-116:~/nginxpod$ kubectl exec -it nginxpod -- /bin/bash
root@nginxpod:/# ls
bin  boot  dev  docker-entrypoint.d  docker-entrypoint.sh  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@nginxpod:/# cd /usr/share/nginx/html
root@nginxpod:/usr/share/nginx/html# ls
50x.html  index.html
root@nginxpod:/usr/share/nginx/html# nano index.html
bash: nano: command not found
root@nginxpod:/usr/share/nginx/html# nano index.html
bash: nano: command not found
root@nginxpod:/usr/share/nginx/html# sudo nano index.html
bash: sudo: command not found
root@nginxpod:/usr/share/nginx/html# cat index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
```

```
ubuntu@ip-172-31-21-116:~/nginxpod$ kubectl describe pod nginxpod
Name:             nginxpod
Namespace:        default
Priority:         0
Service Account:  default
Node:             ip-172-31-94-46/172.31.94.46
Start Time:       Wed, 11 Dec 2024 10:34:24 +0000
Labels:           app=myapp
Annotations:      cni.projectcalico.org/containerID: 7dd18d0e3adf9c0446912651ade6401146e9f0bff9f329ec058974bf4779ae3e
                  cni.projectcalico.org/podIP: 192.168.205.131/32
                  cni.projectcalico.org/podIPs: 192.168.205.131/32
Status:           Running
IP:               192.168.205.131
IPs:
  IP:  192.168.205.131
Containers:
  nginxcontainer:
    Container ID:   containerd://af1bab342d114b9eb2b4a65d75d0dc4f9dee591d1b1c6610cc1bba78e8996884
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:fb197595ebe76b9c0c14ab68159fd3c08bd067ec62300583543f0ebda353b5be
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 11 Dec 2024 10:34:29 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-jb7db (ro)
```

```
ubuntu@ip-172-31-21-116:~/nginxpod$ kubectl logs nginxpod
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/12/11 10:34:29 [notice] 1#1: using the "epoll" event method
2024/12/11 10:34:29 [notice] 1#1: nginx/1.27.3
2024/12/11 10:34:29 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/12/11 10:34:29 [notice] 1#1: OS: Linux 6.8.0-1018-aws
2024/12/11 10:34:29 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/12/11 10:34:29 [notice] 1#1: start worker processes
2024/12/11 10:34:29 [notice] 1#1: start worker process 29
```

Kubernetes Services –

1)NodePort –

cd nodepod

ls

nano myservicepod.yml

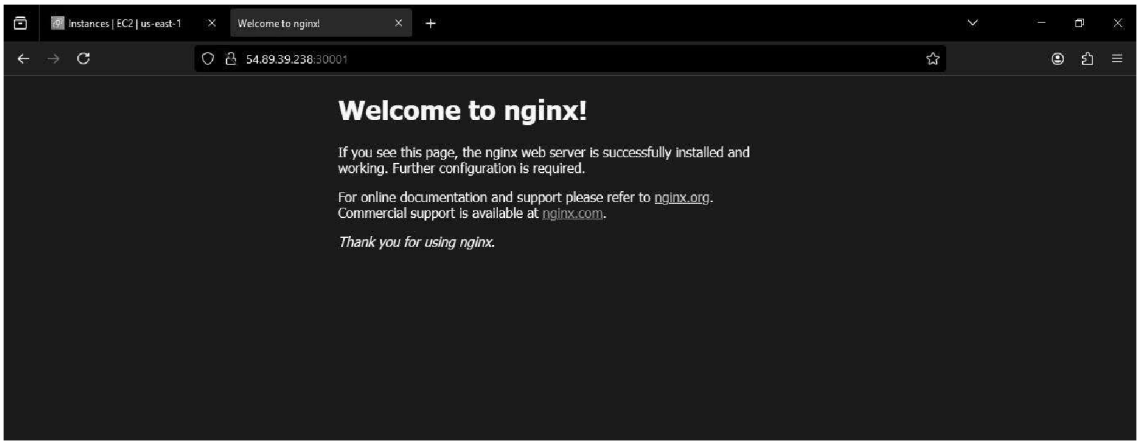
```
apiVersion: v1
kind: Service
metadata:
   name: mynodeportservice
spec:
 type: NodePort
 selector:
     app: myapp
 ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30001
```

kubectl apply -f myservice.yml

kubectl get pods

kubectl get services

Add 30000-32000 in worker node security inbound rule.



kubectl exec -it nginxpod - - /bin/bash

cd /usr/share/nginx/html

ls

touch ravi.html

echo “this is ravi.html page under nginxpod” <ravi.html

t

```
root@nginxpod:/usr/share/nginx/html# ls
50x.html  index.html  ravi.html
root@nginxpod:/usr/share/nginx/html#
```

For load balancing –

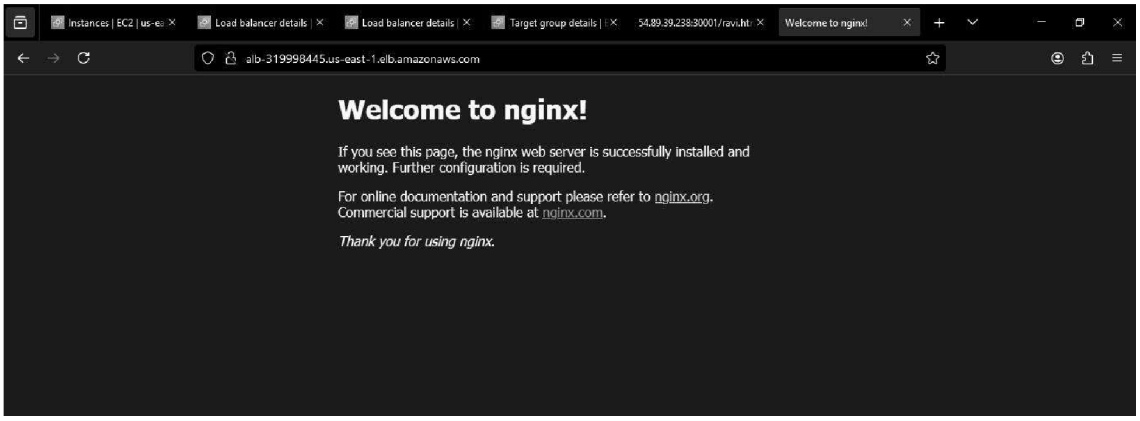
cd nginxpod

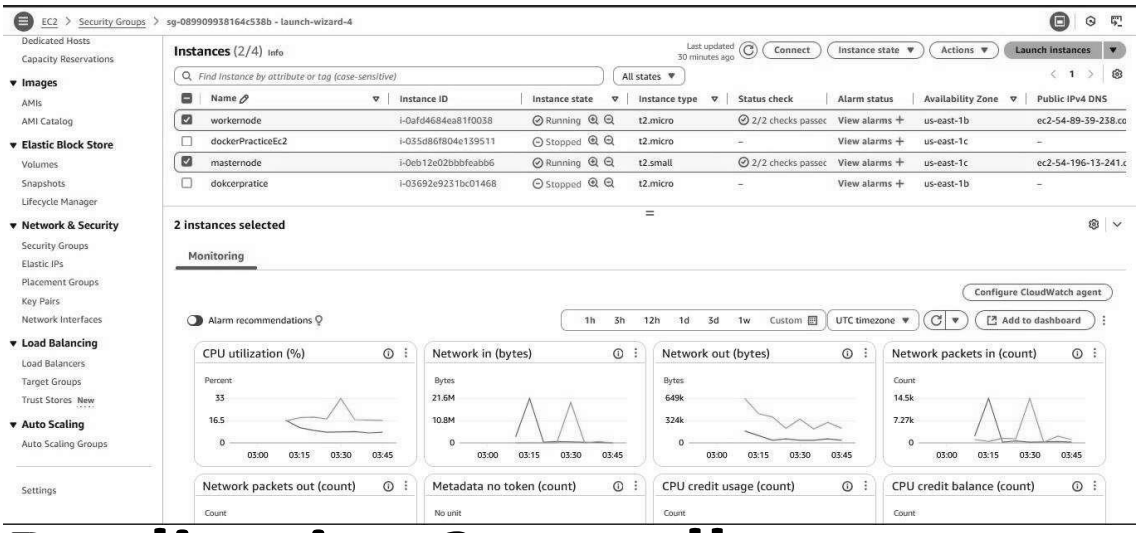
cp myservice.yml loadwalaservice.yml

(chages in loadwalaservice.yml)

Create load balancer in AWS and at target group (add load balancer service port)

Copy DNS of load balancer.





ReplicationController –

ReplicationController –

- Pod recreation
- Scaling (manual)
- Replicas

Pratical -

kubectl get nodes

kubectl delete all –all –force

mkdir replicationController

cd replicationController

nano mynginx.yml

```
ubuntu@ip-172-31-21-116:~$ mkdir replicationcontrollerwala
ubuntu@ip-172-31-21-116:~$ cd replicationcontrollerwala
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ ls
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ nano mynginx.yml
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ ls
mynginx.yml
```

kubectl apply -f mynginx.yml

kubectl get rc

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl get rc
NAME    DESIRED   CURRENT   READY   AGE
myrc    3         3         3       110s
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl get pods

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
myrc-grcq6   1/1     Running   0          2m18s
myrc-snzk8   1/1     Running   0          2m18s
myrc-wn2zt   1/1     Running   0          2m18s
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl delete pod anypod_name –force

(to checking auto creation of pod of not)

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl delete pod myrc-grcq6 --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may continue to run on the cl
uster indefinitely.
pod "myrc-grcq6" force deleted
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl get pods -o wide

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl get pod -o wide
NAME         READY   STATUS    RESTARTS   AGE     IP              NODE              NOMINATED NODE   READINESS GATES
myrc-php62   1/1     Running   0          105s    192.168.205.140   ip-172-31-94-46   <none>           <none>
myrc-snzk8   1/1     Running   0          4m50s   192.168.205.137   ip-172-31-94-46   <none>           <none>
myrc-wn2zt   1/1     Running   0          4m50s   192.168.205.138   ip-172-31-94-46   <none>           <none>
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl scale rc myrc –replicas=4

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl scale rc myrc --replicas=4
replicationcontroller/myrc scaled
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl get rc

kubectl get pod -o wide

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl get rc
NAME   DESIRED   CURRENT   READY   AGE
myrc   4         4         4       6m45s
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
myrc-php62   1/1     Running   0          3m47s
myrc-snzk8   1/1     Running   0          6m52s
myrc-vjfj2   1/1     Running   0          64s
myrc-wn2zt   1/1     Running   0          6m52s
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl delete all - -all - -force

```
^Cubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl delete all --all --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may continue to run on the cl
uster indefinitely.
pod "myrc-php62" force deleted
pod "myrc-snzk8" force deleted
pod "myrc-vjfj2" force deleted
pod "myrc-wn2zt" force deleted
replicationcontroller "myrc" force deleted
service "kubernetes" force deleted
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

cp mynginx.yml yournginx.yml

cp mynginx.yml ournginx.yml

ls

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ cp mynginx.yml yournginx.yml
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ cp mynginx.yml ournginx.yml
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ ls
mynginx.yml  ournginx.yml  yournginx.yml
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl apply -f yournginx.yml

kubectl apply -f ournginx.yml

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl apply -f yournginx.yml
replicationcontroller/myrc1 created
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl apply -f ournginx.yml
replicationcontroller/myrc2 created
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl get all

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl get all
NAME              READY   STATUS    RESTARTS   AGE
pod/myrc-8t2vg    1/1     Running   0          4m2s
pod/myrc-9bncs    1/1     Running   0          4m2s
pod/myrc-td4km    1/1     Running   0          4m2s
pod/myrc1-d48x2   1/1     Running   0          53s
pod/myrc1-jdp98   1/1     Running   0          53s
pod/myrc1-pjvfk   1/1     Running   0          53s
pod/myrc2-49cr6   1/1     Running   0          36s
pod/myrc2-6v7p6   1/1     Running   0          36s
pod/myrc2-72cg7   1/1     Running   0          36s

NAME                            DESIRED   CURRENT   READY   AGE
replicationcontroller/myrc      3         3         3       4m2s
replicationcontroller/myrc1     3         3         3       53s
replicationcontroller/myrc2     3         3         3       36s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   5m3s
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl get rc -o wide

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl get rc -o wide
NAME    DESIRED   CURRENT   READY   AGE     CONTAINERS       IMAGES   SELECTOR
myrc    3         3         3       4m51s   nginxconatiner   nginx    env=dev
myrc1   3         3         3       102s    nginxconatiner   nginx    env=prod
myrc2   3         3         3       85s     nginxconatiner   nginx    env=testing
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl get pod -l env

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl get pod -l env
NAME          READY   STATUS    RESTARTS   AGE
myrc-8t2vg    1/1     Running   0          5m34s
myrc-9bncs    1/1     Running   0          5m34s
myrc-td4km    1/1     Running   0          5m34s
myrc1-d48x2   1/1     Running   0          2m25s
myrc1-jdp98   1/1     Running   0          2m25s
myrc1-pjvfk   1/1     Running   0          2m25s
myrc2-49cr6   1/1     Running   0          2m8s
myrc2-6v7p6   1/1     Running   0          2m8s
myrc2-72cg7   1/1     Running   0          2m8s
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

kubectl get pod -l env=dev

```
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$ kubectl get pod -l env=dev
NAME          READY   STATUS    RESTARTS   AGE
myrc-8t2vg    1/1     Running   0          6m2s
myrc-9bncs    1/1     Running   0          6m2s
myrc-td4km    1/1     Running   0          6m2s
ubuntu@ip-172-31-21-116:~/replicationcontrollerwala$
```

ReplicaSet –

cp -r replicationcontroller replicasetwala

cd replicasetwala

ls

nano mynginx.yml

nano yourngінx.yml

nano ourngінx.yml

```
ubuntu@ip-172-31-21-116:~/replicasetwala$ ls
mynginx.yml  ourginx.yml  yourginx.yml
ubuntu@ip-172-31-21-116:~/replicasetwala$
```

```
 GNU nano 7.2                                  mynginx.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myrc
spec:
  replicas: 3
  selector:
    env: dev
  template:
    metadata:
     labels:
      env: dev
    spec:
     containers:
      - name: nginxconatiner
        image: nginx
        ports:
         - containerPort: 80
```

```
 GNU nano 7.2                                  yourginx.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myrc1
spec:
  replicas: 3
  selector:
    matchLabels:
      env: prod
  template:
    metadata:
      labels:
        env: prod
    spec:
      containers:
        - name: nginxcontainer
          image: nginx
          ports:
            - containerPort: 80
```

```
GNU nano 7.2                                    ournginx.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myrc2
spec:
  replicas: 3
  selector:
    matchLabels:
      env: testing
  template:
    metadata:
      labels:
        env: testing
    spec:
      containers:
        - name: nginxcontainer
          image: nginx
          ports:
            - containerPort: 80
```

kubectl apply -f mynginx.yml

kubectl apply -f yournginx.yml

kunectl apply -f ournginx.yml

```
ubuntu@ip-172-31-21-116:~/replicasetwala$ kubectl apply -f mynginx.yml
replicaset.apps/myrc created
ubuntu@ip-172-31-21-116:~/replicasetwala$ kubectl apply -f yournginx.yml
replicaset.apps/myrc1 created
ubuntu@ip-172-31-21-116:~/replicasetwala$ kubectl apply -f ournginx.yml
replicaset.apps/myrc2 created
ubuntu@ip-172-31-21-116:~/replicasetwala$
```

kubectl get all

```
ubuntu@ip-172-31-21-116:~/replicasetwala$ kubectl get all
NAME              READY   STATUS    RESTARTS   AGE
pod/myrc-8n9vh    1/1     Running   0          56s
pod/myrc-m97vm    1/1     Running   0          56s
pod/myrc-s4hvb    1/1     Running   0          56s
pod/myrc1-5hkhj   1/1     Running   0          44s
pod/myrc1-gqc29   1/1     Running   0          44s
pod/myrc1-mhjld   1/1     Running   0          44s
pod/myrc2-5cjlg   1/1     Running   0          35s
pod/myrc2-qczlp   1/1     Running   0          35s
pod/myrc2-xrwh9   1/1     Running   0          35s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   68s

NAME                    DESIRED   CURRENT   READY   AGE
replicaset.apps/myrc    3         3         3       56s
replicaset.apps/myrc1   3         3         3       44s
replicaset.apps/myrc2   3         3         3       35s
ubuntu@ip-172-31-21-116:~/replicasetwala$
```

kubectl get pods - - selector ‘env in (dev, testing)’

```
ubuntu@ip-172-31-21-116:~/replicasetwala$ kubectl get pods --selector 'env in (dev,testing)'
NAME          READY   STATUS    RESTARTS   AGE
myrc-8n9vh    1/1     Running   0          115s
myrc-m97vm    1/1     Running   0          115s
myrc-s4hvb    1/1     Running   0          115s
myrc2-5cjlg   1/1     Running   0          94s
myrc2-qczlp   1/1     Running   0          94s
myrc2-xrwh9   1/1     Running   0          94s
ubuntu@ip-172-31-21-116:~/replicasetwala$
```

Deployment –

Deployment is internally replica set, internally replication controller, internally pod.

Deployment solved problem of -

a) Rolling update
b) Give batch wise update
c) Rollback functionality

mkdir deployment

cd deployment

```
ubuntu@ip-172-31-21-116:~$ ls
nginxpod  replicasetwala  replicationcontrollerwala
ubuntu@ip-172-31-21-116:~$ mkdir deployment
ubuntu@ip-172-31-21-116:~$ ls
deployment  nginxpod  replicasetwala  replicationcontrollerwala
ubuntu@ip-172-31-21-116:~$ cd deployment
ubuntu@ip-172-31-21-116:~/deployment$
```

nano mynginx.yml

```
 GNU nano 7.2                                  mynginx.yml
apiVersion: apps/v1
kind: Deployment
metadata:
   name: mydeploy
spec:
  replicas: 2
  selector:
   matchLabels:
      env: dev
  template:
       metadata:
          labels:
              env: dev
       spec:
            containers:
                - name: nginxcontainer
                  image: nginx
                  ports:
                    - containerPort: 80
```

kubectl apply -f mynginix.yml

```
ubuntu@ip-172-31-21-116:~/deployment$ nano mynginx.yml
ubuntu@ip-172-31-21-116:~/deployment$ kubectl apply -f mynginx.yml
deployment.apps/mydeploy created
ubuntu@ip-172-31-21-116:~/deployment$
```

kubectl get deploy

kubectl get all

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl get deploy
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
mydeploy   2/2     2            2           28s
ubuntu@ip-172-31-21-116:~/deployment$ kubectl get all
NAME                            READY   STATUS    RESTARTS   AGE
pod/mydeploy-7769958795-96d8l   1/1     Running   0          60s
pod/mydeploy-7769958795-p5hll   1/1     Running   0          60s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   117s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeploy   2/2     2            2           60s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeploy-7769958795   2         2         2       60s
ubuntu@ip-172-31-21-116:~/deployment$
```

For update –

kubectl set image deployment mydeploy nginxcontainer=httpd

kubectl rollout status deployment mydeploy

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl set image deployment mydeploy nginxcontainer=httpd
deployment.apps/mydeploy image updated
ubuntu@ip-172-31-21-116:~/deployment$ kubectl rollout status deployment mydeploy
deployment "mydeploy" successfully rolled out
ubuntu@ip-172-31-21-116:~/deployment$
```

kubectl rollout history deployment mydeploy

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl rollout history deploy mydeploy
deployment.apps/mydeploy
REVISION  CHANGE-CAUSE
1         <none>
2         <none>

ubuntu@ip-172-31-21-116:~/deployment$
```

kubectl get deployment mydeploy -o wide

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl get deployment mydeploy -o wide
NAME       READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS       IMAGES   SELECTOR
mydeploy   2/2     2            2           4m12s   nginxcontainer   httpd    env=dev
ubuntu@ip-172-31-21-116:~/deployment$
```

kubectl get deployment mydeploy -o yaml

```
      env: dev
    spec:
      containers:
      - image: httpd
        imagePullPolicy: Always
        name: nginxcontainer
        ports:
        - containerPort: 80
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: "2024-12-17T03:40:07Z"
    lastUpdateTime: "2024-12-17T03:40:07Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2024-12-17T03:40:05Z"
    lastUpdateTime: "2024-12-17T03:47:43Z"
    message: ReplicaSet "mydeploy-68cc955b75" has successfully progressed.
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 5
  readyReplicas: 2
  replicas: 2
  updatedReplicas: 2
ubuntu@ip-172-31-21-116:~/deployment$
```

kubectl apply -f mynginx.yml

kubectl rollout history deployment mydeploy

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl apply -f mynginx.yml
deployment.apps/mydeploy configured
ubuntu@ip-172-31-21-116:~/deployment$ kubectl get deploy mydeploy -o wide
NAME       READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS       IMAGES   SELECTOR
mydeploy   2/2     2            2           5m55s   nginxcontainer   nginx    env=dev
ubuntu@ip-172-31-21-116:~/deployment$ kubectl rollout history deploy mydeploy
deployment.apps/mydeploy
REVISION  CHANGE-CAUSE
2         <none>
3         <none>

ubuntu@ip-172-31-21-116:~/deployment$
```

kubectl rollout undo deploy mydeploy - -to-revision=2

kubectl rollout history deploy mydeploy

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl rollout undo deployment mydeploy --to-revision=2
deployment.apps/mydeploy rolled back
ubuntu@ip-172-31-21-116:~/deployment$ kubectl rollout history deployment mydeploy
deployment.apps/mydeploy
REVISION  CHANGE-CAUSE
3         <none>
4         <none>

ubuntu@ip-172-31-21-116:~/deployment$
```

kubectl annotate deployments.apps mydeploy kubernetes.io/change-cause=”version4”

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl annotate deployment.apps mydeploy kubernetes.io/change-cause="version4"
deployment.apps/mydeploy annotated
ubuntu@ip-172-31-21-116:~/deployment$ kubectl rollout history deployment mydeploy
deployment.apps/mydeploy
REVISION  CHANGE-CAUSE
3         <none>
4         version4

ubuntu@ip-172-31-21-116:~/deployment$
```

nano updatefile.yml

kubectl apply -f updatefile.yml

kubectl rollout history deployment mydeployment

kubectl get deploy - -watch

kubectl get deploy/mydeploy - -watch

For configure (healthProbe) -

nano configurefile.yml #livenessProbe

kubectl apply -f configurefile.yml

```
ubuntu@ip-172-31-21-116:~/deployment$ nano configurefile.yml
ubuntu@ip-172-31-21-116:~/deployment$ kubectl apply -f configurefile.yml
deployment.apps/mydeploy created
ubuntu@ip-172-31-21-116:~/deployment$
```

kubectl get pod

kubectl get all

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl get all
NAME                            READY   STATUS    RESTARTS   AGE
pod/mydeploy-86cb6f57d6-9k9j5   1/1     Running   0          23s
pod/mydeploy-86cb6f57d6-m95nj   1/1     Running   0          23s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   7m52s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeploy   2/2     2            2           23s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeploy-86cb6f57d6   2         2         2       23s
ubuntu@ip-172-31-21-116:~/deployment$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mydeploy-86cb6f57d6-9k9j5   1/1     Running   0          35s
mydeploy-86cb6f57d6-m95nj   1/1     Running   0          35s
ubuntu@ip-172-31-21-116:~/deployment$
```

kubectl delete pod_name - - force

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl get pod
NAME                        READY   STATUS    RESTARTS   AGE
mydeploy-86cb6f57d6-9k9j5   1/1     Running   0          6m38s
mydeploy-86cb6f57d6-m95nj   1/1     Running   0          6m38s
ubuntu@ip-172-31-21-116:~/deployment$ kubectl delete pod mydeploy-86cb6f57d6-9k9j5 --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may continue to run on the cluster indefinitely.
pod "mydeploy-86cb6f57d6-9k9j5" force deleted
```

kubectl get pods

```
ubuntu@ip-172-31-21-116:~/deployment$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mydeploy-86cb6f57d6-m5hbm   1/1     Running   0          16s
mydeploy-86cb6f57d6-m95nj   1/1     Running   0          7m14s
ubuntu@ip-172-31-21-116:~/deployment$
```

Check another pod is created or not intent of crashed Pod.

Health Probe –

Kubernetes has various types of probes:

Liveness probe

Liveness probes determine when to restart a container. For example, liveness probes could catch a deadlock when an application is running but unable to make progress.

If a container fails its liveness probe repeatedly, the kubelet restarts the container.

Liveness probes do not wait for readiness probes to succeed. If you want to wait before executing a liveness probe, you can either define initialDelaySeconds or use startup probe.

```
ubuntu@ip-172-31-21-116:~$ mkdir healthprobe
mkdir: cannot create directory ‘healthprobe’: File exists
ubuntu@ip-172-31-21-116:~$ cd healthprobe/
ubuntu@ip-172-31-21-116:~/healthprobe$ ls
livenessprobe.yml  readinessprobe.yml  startupprobe.yml
ubuntu@ip-172-31-21-116:~/healthprobe$ rm livenessprobe.yml
ubuntu@ip-172-31-21-116:~/healthprobe$ ls
readinessprobe.yml  startupprobe.yml
ubuntu@ip-172-31-21-116:~/healthprobe$ nano livenessfile.yml
```

```
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl apply -f livenessfile.yml
replicationcontroller/mypod created
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
mypod-24t2r   1/1     Running   0          5s
mypod-6kp68   1/1     Running   0          5s
mypod-fdbbp   1/1     Running   0          5s
mypod-k4grs   1/1     Running   0          5s
mypod-tk4sq   1/1     Running   0          5s
ubuntu@ip-172-31-21-116:~/healthprobe$
```

```
ubuntu@ip-172-31-21-116:~/healthprobe$ nano livenesscommandlinefile.yml
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl apply -f livenesscommandlinefile.yml
replicationcontroller/mypod created
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
mypod-4mpph   1/1     Running   0          6s
mypod-dhblr   1/1     Running   0          6s
mypod-jw8s8   1/1     Running   0          6s
mypod-sq8vs   1/1     Running   0          6s
mypod-xq87h   1/1     Running   0          6s
ubuntu@ip-172-31-21-116:~/healthprobe$
```

```
 GNU nano 7.2                                livenesscommandlinefile.yml
apiVersion: v1
kind: ReplicationController
metadata:
  name: mypod
spec:
    replicas: 5
    template:
      metadata:
         labels:
          app: test
      spec:
        containers:
        - image: nginx
          name: mycont
          ports:
            - containerPort: 80
          livenessProbe:
             exec:
              command:
                - ls /usr/share/nginx/html
```

Readiness probe

Readiness probes determine when a container is ready to start accepting traffic. This is useful when waiting for an application to perform time-consuming initial tasks, such as establishing network connections, loading files, and warming caches.

If the readiness probe returns a failed state, Kubernetes removes the pod from all matching service endpoints.

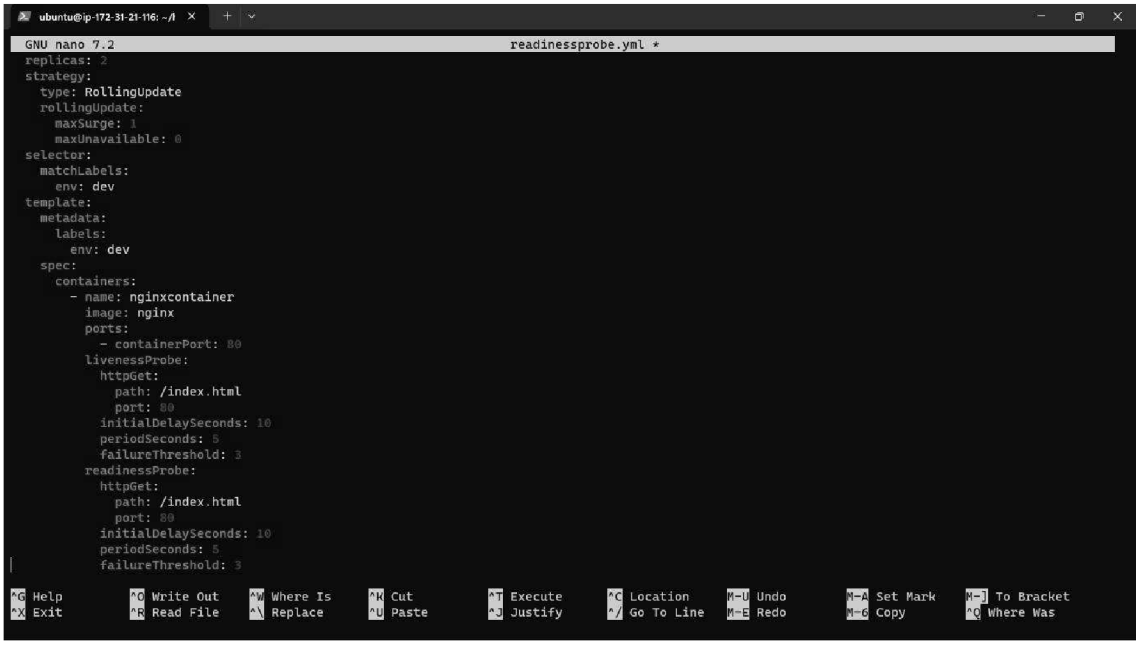
Readiness probes run on the container during its whole lifecycle.

mkdir healthprobe

cd healthprobe

nano readinessprobe.yml

```
ubuntu@ip-172-31-21-116:~$ mkdir healthprobe
ubuntu@ip-172-31-21-116:~$ ls
deployment  healthprobe  nginxpod  replicasetwala  replicationcontrollerwala
ubuntu@ip-172-31-21-116:~$ cd healthprobe
ubuntu@ip-172-31-21-116:~/healthprobe$ ls
ubuntu@ip-172-31-21-116:~/healthprobe$ nano livenessprobe.yml
ubuntu@ip-172-31-21-116:~/healthprobe$ nano readinessprobe.yml
ubuntu@ip-172-31-21-116:~/healthprobe$
```



kubectl apply -f readinessprobe.yml

```
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl apply -f readinessprobe.yml
deployment.apps/mydeploy created
ubuntu@ip-172-31-21-116:~/healthprobe$
```

kubectl get all

```
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl get all
NAME                             READY   STATUS    RESTARTS   AGE
pod/mydeploy-57f99b6f89-qgsvs    1/1     Running   0          106s
pod/mydeploy-57f99b6f89-v89ms    1/1     Running   0          106s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   23h

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeploy    2/2     2            2           106s

NAME                                   DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeploy-57f99b6f89    2         2         2       106s
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
mydeploy-57f99b6f89-qgsvs    1/1     Running   0          114s
mydeploy-57f99b6f89-v89ms    1/1     Running   0          114s
ubuntu@ip-172-31-21-116:~/healthprobe$
```

kubectl exec -it pod_name - - /bin/bash

cd /usr/share/nginx/html

ls

rm index.html

exit

```
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl exec -it mydeploy-57f99b6f89-qgsvs -- /bin/bash
root@mydeploy-57f99b6f89-qgsvs:/# cd /usr/share/nginx/html
root@mydeploy-57f99b6f89-qgsvs:/usr/share/nginx/html# ls
50x.html  index.html
root@mydeploy-57f99b6f89-qgsvs:/usr/share/nginx/html# rm index.html
root@mydeploy-57f99b6f89-qgsvs:/usr/share/nginx/html# ls
50x.html
root@mydeploy-57f99b6f89-qgsvs:/usr/share/nginx/html# exit
exit
ubuntu@ip-172-31-21-116:~/healthprobe$
```

kubectl get pods - - watch

```
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl get pod --watch
NAME                         READY   STATUS    RESTARTS   AGE
mydeploy-57f99b6f89-qgsvs    0/1     Running   0          3m49s
mydeploy-57f99b6f89-v89ms    1/1     Running   0          3m49s
```

kubectl exec -it pod_name - - /bin/bash

cd /usr/share/nginx/html

touch index.html

echo “hi there” >index.html

exit

```
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl exec -it mydeploy-57f99b6f89-qgsvs -- /bin/bash
root@mydeploy-57f99b6f89-qgsvs:/# cd /usr/share/nginx/html
root@mydeploy-57f99b6f89-qgsvs:/usr/share/nginx/html# ls
50x.html
root@mydeploy-57f99b6f89-qgsvs:/usr/share/nginx/html# touch index.html
root@mydeploy-57f99b6f89-qgsvs:/usr/share/nginx/html# echo "hi there" >index.html
root@mydeploy-57f99b6f89-qgsvs:/usr/share/nginx/html# ls
50x.html  index.html
root@mydeploy-57f99b6f89-qgsvs:/usr/share/nginx/html#
```

kubectl get pods

```
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
mydeploy-57f99b6f89-qgsvs    1/1     Running   0          6m13s
mydeploy-57f99b6f89-v89ms    1/1     Running   0          6m13s
ubuntu@ip-172-31-21-116:~/healthprobe$
```

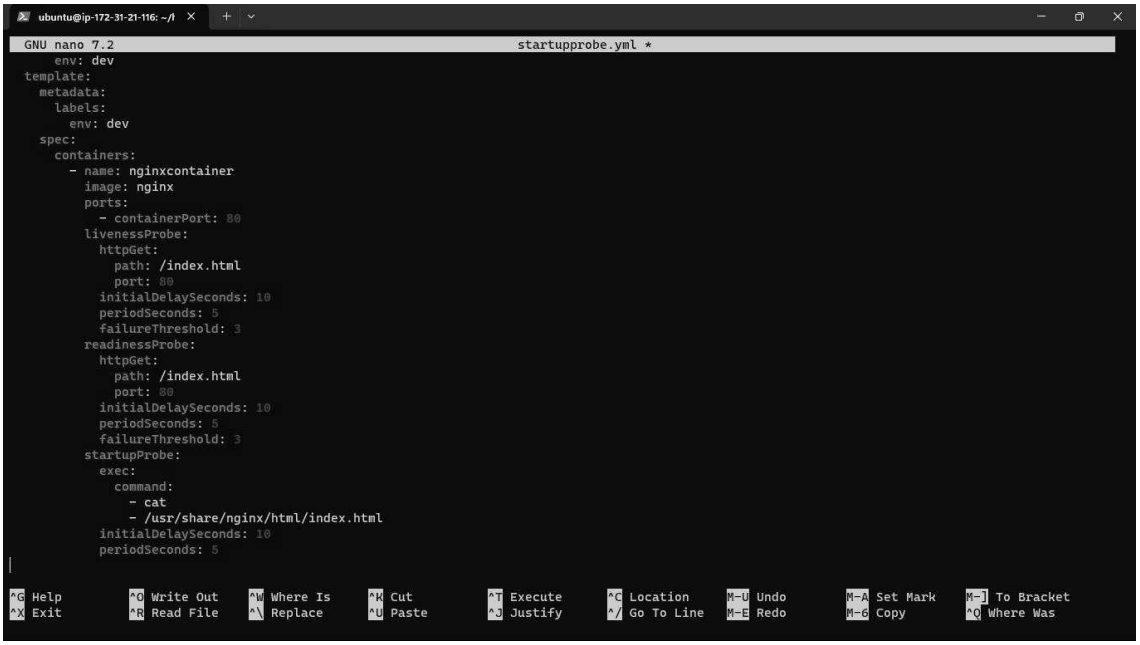
Startup probe

A startup probe verifies whether the application within a container is started. This can be used to adopt liveness checks on slow starting containers, avoiding them getting killed by the kubelet before they are up and running.

If such a probe is configured, it disables liveness and readiness checks until it succeeds.

This type of probe is only executed at startup, unlike liveness and readiness probes, which are run periodically.

nano startupprobe.yml



kubectl apply -f startupprobe.yml

kubectl describe pod pod_name

```
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       False
  ContainersReady             False
  PodScheduled                True
Volumes:
  kube-api-access-hcvdf:
    Type:                     Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:   3607
    ConfigMapName:            kube-root-ca.crt
    ConfigMapOptional:        <nil>
    DownwardAPI:              true
QoS Class:                    BestEffort
Node-Selectors:               <none>
Tolerations:                  node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type     Reason     Age                    From               Message
  ----     ------     ----                   ----               -------
  Normal   Scheduled  4m28s                  default-scheduler  Successfully assigned default/mydeploy-5fd95555b6-4g7tc to ip-172-31-94-46
  Normal   Pulling    4m28s                  kubelet            Pulling image "nginx"
  Normal   Pulled     4m27s                  kubelet            Successfully pulled image "nginx" in 145ms (237ms including waiting). Image size: 7
2099501 bytes.
  Normal   Created    4m27s                  kubelet            Created container nginxcontainer
  Normal   Started    4m27s                  kubelet            Started container nginxcontainer
  Warning  Unhealthy  4m2s                   kubelet            Liveness probe failed: Get "http://192.168.205.139:80/index.html": dial tcp 192.168
.205.139:80: i/o timeout (Client.Timeout exceeded while awaiting headers)
  Warning  Unhealthy  3m57s (x2 over 4m7s)   kubelet            Liveness probe failed: Get "http://192.168.205.139:80/index.html": context deadline
 exceeded (Client.Timeout exceeded while awaiting headers)
  Normal   Killing    3m57s                  kubelet            Container nginxcontainer failed liveness probe, will be restarted
  Warning  Unhealthy  3m56s                  kubelet            Readiness probe failed: Get "http://192.168.205.139:80/index.html": dial tcp 192.16
8.205.139:80: i/o timeout (Client.Timeout exceeded while awaiting headers)
  Normal   Killing    3m15s (x2 over 3m57s)  kubelet            Stopping container nginxcontainer
  Warning  Unhealthy  3m12s (x14 over 4m12s) kubelet            Readiness probe failed: Get "http://192.168.205.139:80/index.html": context deadlin
e exceeded (Client.Timeout exceeded while awaiting headers)
ubuntu@ip-172-31-21-116:~/healthprobe$
```

kubectl get pods

```
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
mydeploy-5fd95555b6-4g7tc   0/1     Running   0          3m55s
mydeploy-5fd95555b6-jdbn2   0/1     Running   0          3m55s
```

kubectl get pods #After startup

```
ubuntu@ip-172-31-21-116:~/healthprobe$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
mydeploy-5fd95555b6-4g7tc   1/1     Running   0          9m58s
mydeploy-5fd95555b6-jdbn2   1/1     Running   0          9m58s
ubuntu@ip-172-31-21-116:~/healthprobe$
```

Namespace -

In Kubernetes, *namespaces* provide a mechanism for isolating groups of resources within a single cluster. Names of resources need to be unique within a namespace, but not across namespaces. Namespace-based scoping is applicable only for namespaced object *(e.g. Deployments, Services, etc.)* and not for cluster-wide objects *(e.g. StorageClass, Nodes, PersistentVolumes, etc.)*.

Namespace by commands.

kubectl get namespace

```
ubuntu@ip-172-31-21-116:~$ mkdir namespacewala
ubuntu@ip-172-31-21-116:~$ ls
deployment  healthprobe  namespacewala  nginxpod  replicasetwala  replicationcontrollerwala
ubuntu@ip-172-31-21-116:~$ cd namespacewala/
ubuntu@ip-172-31-21-116:~/namespacewala$ ls
ubuntu@ip-172-31-21-116:~/namespacewala$ nano namespaceexample.yml
ubuntu@ip-172-31-21-116:~/namespacewala$
```

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl get namespace
NAME              STATUS   AGE
default           Active   9d
kube-node-lease   Active   9d
kube-public       Active   9d
kube-system       Active   9d
ubuntu@ip-172-31-21-116:~/namespacewala$
```

kubectl get pods -n kube-system

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl get pods -n kube-system
NAME                                       READY   STATUS             RESTARTS        AGE
calico-kube-controllers-6cdb97b867-zk6tr   1/1     Running            9 (4m3s ago)    9d
calico-node-8k8bg                          0/1     CrashLoopBackOff   66 (16s ago)    9d
calico-node-j5mqz                          0/1     Running            67 (114s ago)   9d
coredns-55cb58b774-84665                   1/1     Running            8 (4m3s ago)    9d
coredns-55cb58b774-zgx4j                   1/1     Running            8 (4m5s ago)    9d
etcd-ip-172-31-21-116                      1/1     Running            8 (4m3s ago)    9d
kube-apiserver-ip-172-31-21-116            1/1     Running            8 (4m3s ago)    9d
kube-controller-manager-ip-172-31-21-116   1/1     Running            8 (4m3s ago)    9d
kube-proxy-96fds                           0/1     CrashLoopBackOff   72 (23s ago)    9d
kube-proxy-d572p                           1/1     Running            71 (64s ago)    9d
kube-scheduler-ip-172-31-21-116            1/1     Running            8 (4m3s ago)    9d
ubuntu@ip-172-31-21-116:~/namespacewala$
```

kubectl create namespace mynamespace

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl create namespace mynamespace
namespace/mynamespace created
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl get namespace
NAME              STATUS   AGE
default           Active   9d
kube-node-lease   Active   9d
kube-public       Active   9d
kube-system       Active   9d
mynamespace       Active   5s
ubuntu@ip-172-31-21-116:~/namespacewala$
```

kubectl get namespace

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl create namespace mynamespace
namespace/mynamespace created
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl get namespace
NAME              STATUS   AGE
default           Active   9d
kube-node-lease   Active   9d
kube-public       Active   9d
kube-system       Active   9d
mynamespace       Active   5s
ubuntu@ip-172-31-21-116:~/namespacewala$
```

kubectl apply -f namespaceexample.yml -n mynamespace

```
ubuntu@ip-172-31-21-116:~/namespacewala$ ls
namespaceexample.yml
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl apply -f namespaceexample.yml  -n mynamespace
pod/nginxpod created
ubuntu@ip-172-31-21-116:~/namespacewala$
```

kubectl get pods

kubectl get pods -n mynamespace

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl get pods
No resources found in default namespace.
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl get pods -n mynamespace
NAME       READY   STATUS    RESTARTS   AGE
nginxpod   1/1     Running   0          35s
ubuntu@ip-172-31-21-116:~/namespacewala$
```

kubectl config set-context - -current - - namespace mynamespace

#To creating custom namespace as default namespace

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl config set-context --current --namespace mynamespace
Context "kubernetes-admin@kubernetes" modified.
ubuntu@ip-172-31-21-116:~/namespacewala$
```

kubectl get pods

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl config set-context --current --namespace mynamespace
Context "kubernetes-admin@kubernetes" modified.
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
nginxpod   1/1     Running   0          112s
ubuntu@ip-172-31-21-116:~/namespacewala$
```

kubectl config set-context - -current - -namespace default

kubectl get pods

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl config set-context --current --namespace default
Context "kubernetes-admin@kubernetes" modified.
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl get pods
No resources found in default namespace.
ubuntu@ip-172-31-21-116:~/namespacewala$
```

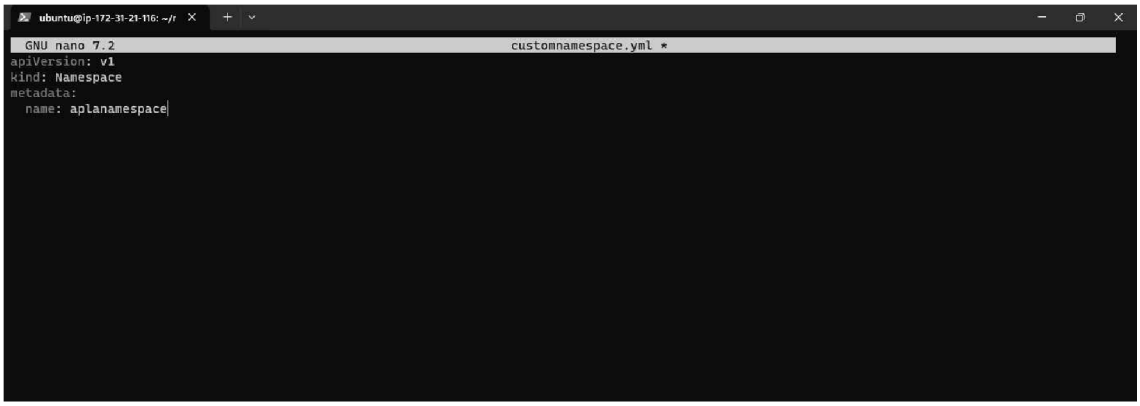
kubectl delete namespace mynamespace

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl delete pod nginxpod
Error from server (NotFound): pods "nginxpod" not found
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl delete pod nginxpod -n mynamespace
pod "nginxpod" deleted
```

```
ubuntu@ip-172-31-21-116:~/namespacewala$ kubectl delete namespace mynamespace --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been terminated. The resource may continue to run on the cl
uster indefinitely.
namespace "mynamespace" force deleted
```

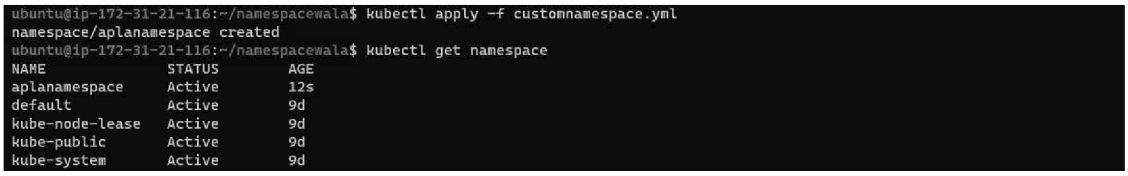
Namespace by .yml file

nano customnamespace.yml



kubectl apply -f customnamespace.yml

kubectl get namespace



Volume -

In Kubernetes, Volumes are used to provide persistent or ephemeral storage to containers running within Pods. Unlike container storage, which is ephemeral and tied to the lifecycle of a container, volumes allow data to persist beyond the container's lifecycle.

- emptyDir
- hostPath
- nfs
- persistent volume and persistent volume chain
- configmap
- secrete

emptyDir:

emptyDir is a type of Kubernetes volume that is created when a Pod is assigned to a Node. As the name suggests, it starts out empty and is used as a temporary storage location for data. The data in an emptyDir volume is deleted when the Pod is terminated or removed. It's typically used for tasks like caching, temporary file storage, or storing data that doesn't need to persist across Pod restarts.

hostPath:

hostPath is a Kubernetes volume that mounts a file or directory from the host node’s filesystem into the Pod. This allows the Pod to access data or configuration files from the host machine. It can be useful in scenarios where you need to access specific files from the host or interact with hardware devices on the host machine. However, it can have security implications, as it provides the Pod access to the host’s filesystem.

nfs:

nfs (Network File System) is a volume type that allows Pods to access shared storage over a network. NFS volumes are used when you need to share data across multiple Pods or Nodes. This is helpful in situations where the data needs to be accessible by multiple Pods concurrently or when you want to store data externally.

Persistent Volume (PV) and Persistent Volume Claim (PVC):

- **Persistent Volume (PV)**: A PV is a piece of storage in the cluster that has been provisioned by an administrator. It can be backed by various storage backends like NFS, cloud storage, or even local disks. The PV has a lifecycle independent of the Pods that use it.
- **Persistent Volume Claim (PVC)**: A PVC is a request for storage by a user. It is similar to a Pod in the sense that it is a resource request for storage. A PVC binds to a PV that matches the requested size and access modes. Once a PVC is bound to a PV, the Pod can use the storage defined in the PV.

ConfigMap:

A ConfigMap is a Kubernetes object that allows you to store configuration data as key-value pairs. ConfigMaps provide a way to inject configuration into Pods without altering the container image. They are often used for environment variables, command-line arguments, or configuration files that Pods can reference at runtime. ConfigMaps help to decouple configuration from the application code, making it easier to manage and change settings without rebuilding or redeploying the application.

Secret:

A Secret is similar to a ConfigMap, but it is used to store sensitive information, such as passwords, API keys, or certificates. Kubernetes secrets are encoded in base64, and while they are more secure than plain text, you should still be careful with access control and encryption to protect sensitive data.

```
ubuntu@ip-172-31-21-116:~$ ls
autoscaling  cronjob  daemonsetwala  healthprobe  job  jobwala  statefulset
ubuntu@ip-172-31-21-116:~$ mkdir emptydir
ubuntu@ip-172-31-21-116:~$ cd emptydir/
ubuntu@ip-172-31-21-116:~/emptydir$ nano emptydirfile.yml
```

```
 GNU nano 7.2                                                emptydirfile.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: mypod2
spec:
  replicas: 5
  selector:
    matchLabels:
      app: dev
  template:
    metadata:
      labels:
        app: dev
    spec:
      containers:
        - image: nginx
          name: mycont
          ports:
            - containerPort: 80
          volumeMounts:
            - name: myvolume
              mountPath: /usr/share/nginx/html/
        - image: nginx
          name: mycont1
          ports:
            - containerPort: 90
          volumeMounts:
            - name: myvolume
              mountPath: /usr/share/nginx/html/
      volumes:
        - name: myvolume
          emptyDir: {}
```

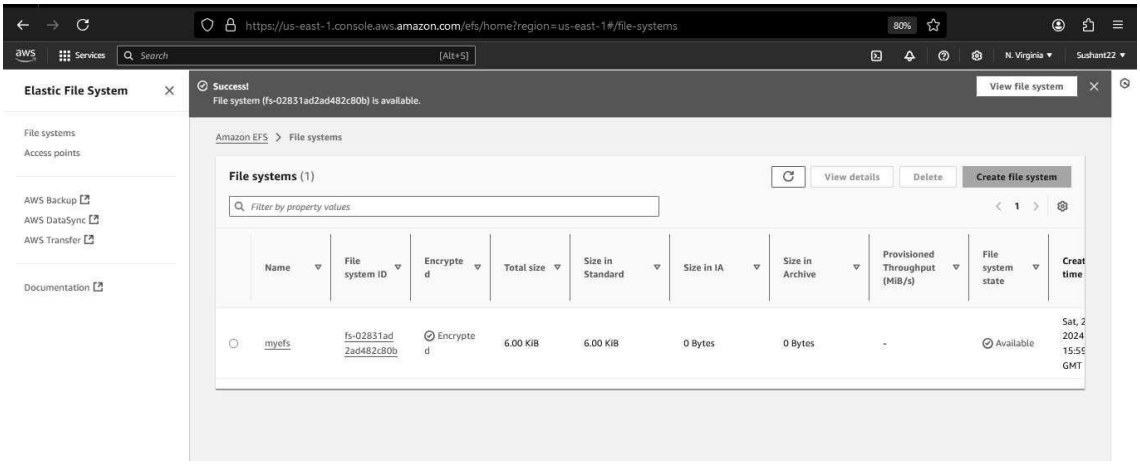
```
ubuntu@ip-172-31-21-116:~/emptydir$ kubectl apply -f emptydirfile.yml
replicaset.apps/mypod2 created
ubuntu@ip-172-31-21-116:~/emptydir$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
mypod2-bkb5r   2/2     Running   0          5s
mypod2-dhxrm   2/2     Running   0          5s
mypod2-dpkfm   2/2     Running   0          5s
mypod2-t524f   2/2     Running   0          5s
mypod2-zwgxp   2/2     Running   0          5s
```

```
ubuntu@ip-172-31-21-116:~/volume/emptydir$ cd ..
ubuntu@ip-172-31-21-116:~/volume$ mkdir hostpath
ubuntu@ip-172-31-21-116:~/volume$ cd hostpath
ubuntu@ip-172-31-21-116:~/volume/hostpath$ ls
ubuntu@ip-172-31-21-116:~/volume/hostpath$
```

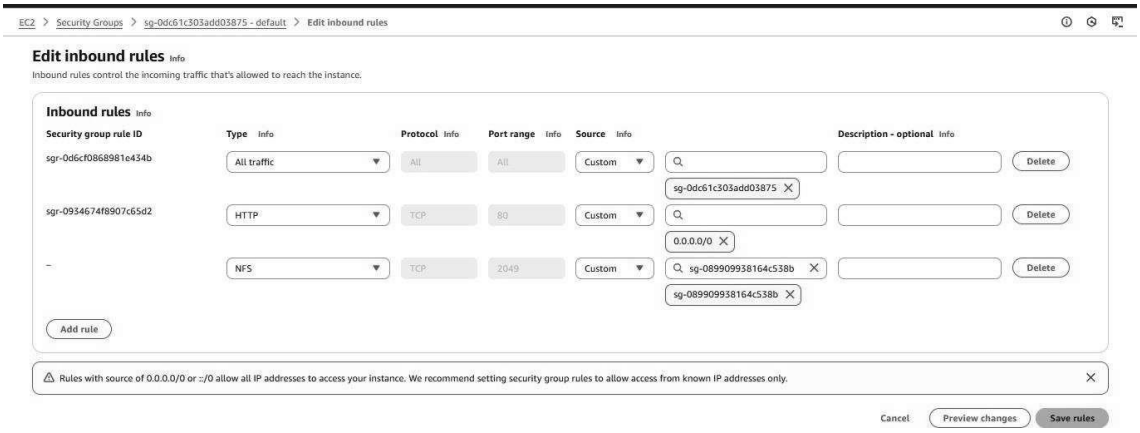
```
ubuntu@ip-172-31-21-116:~$ cd volume
ubuntu@ip-172-31-21-116:~/volume$ ls
emptydir  hostpath
ubuntu@ip-172-31-21-116:~/volume$ cd hostpath
ubuntu@ip-172-31-21-116:~/volume/hostpath$ ls
ubuntu@ip-172-31-21-116:~/volume/hostpath$ nano hostpathfile.yml
ubuntu@ip-172-31-21-116:~/volume/hostpath$ kubectl get all
```

```
 GNU nano 7.2                                                hostpathfile.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeploy
spec:
  replicas: 2
  selector:
    matchLabels:
      env: dev
  template:
    metadata:
      labels:
        env: dev
    spec:
      containers:
        - name: nginxcontainer
          image: nginx
          ports:
            - containerPort: 80
          volumeMounts:
            - mountPath: "/usr/share/nginx/html"
              name: myvolume
      volumes:
        - name: myvolume
          hostPath:
            path: /data
            type: DirectoryOrCreate
```

```
ubuntu@ip-172-31-21-116:~/volume/hostpath$ kubectl apply -f hostpathfile.yml
deployment.apps/mydeploy created
ubuntu@ip-172-31-21-116:~/volume/hostpath$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mydeploy-7f4d64bc6b-j8mcw   1/1     Running   0          8s
mydeploy-7f4d64bc6b-k46h9   1/1     Running   0          8s
```




```
ubuntu@ip-172-31-94-46:~$ mkdir myvolume
ubuntu@ip-172-31-94-46:~$ ls
myvolume
ubuntu@ip-172-31-94-46:~$
```



```
ubuntu@ip-172-31-94-46:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [761 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [173 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [965 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [238 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [309 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [572 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [110 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.7 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:20 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [572 kB]
Get:21 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [111 kB]
Get:22 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [7232 B]
Get:23 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [795 kB]
Get:24 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [169 kB]
Get:25 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.0 kB]
Get:26 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [560 kB]
Get:27 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [108 kB]
Get:28 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:29 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Fetched 6046 kB in 2s (3272 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-94-46:~$
```

```
ubuntu@ip-172-31-94-46:~$ sudo apt-get install nfs-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  keyutils libnfsidmap1 rpcbind
Suggested packages:
  watchdog
The following NEW packages will be installed:
  keyutils libnfsidmap1 nfs-common rpcbind
0 upgraded, 4 newly installed, 0 to remove and 44 not upgraded.
Need to get 400 kB of archives.
After this operation, 1416 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libnfsidmap1 amd64 1:2.6.4-3ubuntu5 [48.2 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 rpcbind amd64 1.2.6-7ubuntu2 [46.5 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 keyutils amd64 1.6.3-3build1 [56.8 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 nfs-common amd64 1:2.6.4-3ubuntu5 [248 kB]
Fetched 400 kB in 0s (15.0 MB/s)
Selecting previously unselected package libnfsidmap1:amd64.
(Reading database ... 101626 files and directories currently installed.)
Preparing to unpack .../libnfsidmap1_1%3a2.6.4-3ubuntu5_amd64.deb ...
Unpacking libnfsidmap1:amd64 (1:2.6.4-3ubuntu5) ...
Selecting previously unselected package rpcbind.
Preparing to unpack .../rpcbind_1.2.6-7ubuntu2_amd64.deb ...
Unpacking rpcbind (1.2.6-7ubuntu2) ...
Selecting previously unselected package keyutils.
Preparing to unpack .../keyutils_1.6.3-3build1_amd64.deb ...
Unpacking keyutils (1.6.3-3build1) ...
Selecting previously unselected package nfs-common.
Preparing to unpack .../nfs-common_1%3a2.6.4-3ubuntu5_amd64.deb ...
Unpacking nfs-common (1:2.6.4-3ubuntu5) ...
Setting up libnfsidmap1:amd64 (1:2.6.4-3ubuntu5) ...
Setting up rpcbind (1.2.6-7ubuntu2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/rpcbind.service → /usr/lib/systemd/system/rpcbind.service.
Created symlink /etc/systemd/system/sockets.target.wants/rpcbind.socket → /usr/lib/systemd/system/rpcbind.socket.
```

```
ubuntu@ip-172-31-94-46:~$ sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-02831ad2ad482c80b.efs
.us-east-1.amazonaws.com:/ ~/myvolume
ubuntu@ip-172-31-94-46:~$
```

```
ubuntu@ip-172-31-94-46: ~/r
 GNU nano 7.2                                    index.html *
<h1>This is nfs storage file in index.html page</h1>
```

```
ubuntu@ip-172-31-94-46:~/myvolume$ sudo rm index.html
ubuntu@ip-172-31-94-46:~/myvolume$ ls
ubuntu@ip-172-31-94-46:~/myvolume$ sudo mkdir test
ubuntu@ip-172-31-94-46:~/myvolume$ cd test
ubuntu@ip-172-31-94-46:~/myvolume/test$ sudo nano index.html
ubuntu@ip-172-31-94-46:~/myvolume/test$
```

```
 GNU nano 7.2                                                    nfsfile.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: mypod2
spec:
  replicas: 3
  selector:
    matchLabels:
      app: dev
  template:
    metadata:
      labels:
        app: dev
    spec:
      containers:
        - image: nginx
          name: mycont
          ports:
            - containerPort: 80
          volumeMounts:
            - name: myvolume
              mountPath: /usr/share/nginx/html/
      volumes:
        - name: myvolume
          nfs:
            server: fs-02831ad2ad482c80b.efs.us-east-1.amazonaws.com
            path: /test
```

```
ubuntu@ip-172-31-21-116:~$ kubectl apply -f nfsfile.yml
replicaset.apps/mypod2 created
ubuntu@ip-172-31-21-116:~$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
mypod2-q4c6g   1/1     Running   0          7s
mypod2-sxxzd   1/1     Running   0          7s
mypod2-x4bpj   1/1     Running   0          7s
ubuntu@ip-172-31-21-116:~$
```

```
    Environment:    <none>
    Mounts:
      /usr/share/nginx/html/ from myvolume (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-4tcjn (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  myvolume:
    Type:      NFS (an NFS mount that lasts the lifetime of a pod)
    Server:    fs-02831ad2ad482c80b.efs.us-east-1.amazonaws.com
    Path:      /test
    ReadOnly:  false
  kube-api-access-4tcjn:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  40s   default-scheduler  Successfully assigned default/mypod2-q4c6g to ip-172-31-94-46
  Normal  Pulling    39s   kubelet            Pulling image "nginx"
  Normal  Pulled     39s   kubelet            Successfully pulled image "nginx" in 113ms (113ms including waiting). Image size: 72099410 bytes.
  Normal  Created    39s   kubelet            Created container mycont
  Normal  Started    39s   kubelet            Started container mycont
  Normal  Killing    38s   kubelet            Stopping container mycont
ubuntu@ip-172-31-21-116:~$
```

```
ubuntu@ip-172-31-21-116:~$ kubectl exec -it mypod2-q4c6g -- /bin/bash
root@mypod2-q4c6g:/# cd /usr/share/nginx/html
root@mypod2-q4c6g:/usr/share/nginx/html# ls
index.html
root@mypod2-q4c6g:/usr/share/nginx/html# cat index.html
<h1>This is nfs storage file in index.html page</h1>
root@mypod2-q4c6g:/usr/share/nginx/html#
```

```
ubuntu@ip-172-31-21-116:~$ kubectl get all
NAME               READY   STATUS    RESTARTS   AGE
pod/mypod2-q4c6g   1/1     Running   0          2m18s
pod/mypod2-sxxzd   1/1     Running   0          2m18s
pod/mypod2-x4bpj   1/1     Running   0          2m18s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   7m58s

NAME                     DESIRED   CURRENT   READY   AGE
replicaset.apps/mypod2   3         3         3       2m18s
ubuntu@ip-172-31-21-116:~$
```

```
ubuntu@ip-172-31-21-116:~$ cd volume
ubuntu@ip-172-31-21-116:~/volume$ ls
emptydir  hostpath  nfswala
ubuntu@ip-172-31-21-116:~/volume$ mkdir configmap
ubuntu@ip-172-31-21-116:~/volume$ ls
configmap  emptydir  hostpath  nfswala
ubuntu@ip-172-31-21-116:~/volume$ cd configmap/
ubuntu@ip-172-31-21-116:~/volume/configmap$
```

```
 GNU nano 7.2                                                    mysqlwala.yml *
apiVersion: v1
kind: Pod
metadata:
 name: mysqlwala
spec:
 containers:
  - name: mysqlcontainer
    image: mysql:8.0
    env:
     - name: MYSQL_ROOT_PASSWORD
       valueFrom:
        configMapKeyRef:
         name: mysql-config
         key: mysql-password
```

```
ubuntu@ip-172-31-21-116:~/volume/configmap$ kubectl create configmap mysql-config --from-literal mysql-password=Pass@123
configmap/mysql-config created
```

```
ubuntu@ip-172-31-21-116:~/volume/configmap$ kubectl get configmap
NAME               DATA   AGE
kube-root-ca.crt   1      13d
mysql-config       1      27s
```

```
ubuntu@ip-172-31-21-116:~/volume/configmap$ kubectl apply -f mysqlwala.yml
pod/mysqlwala created
```

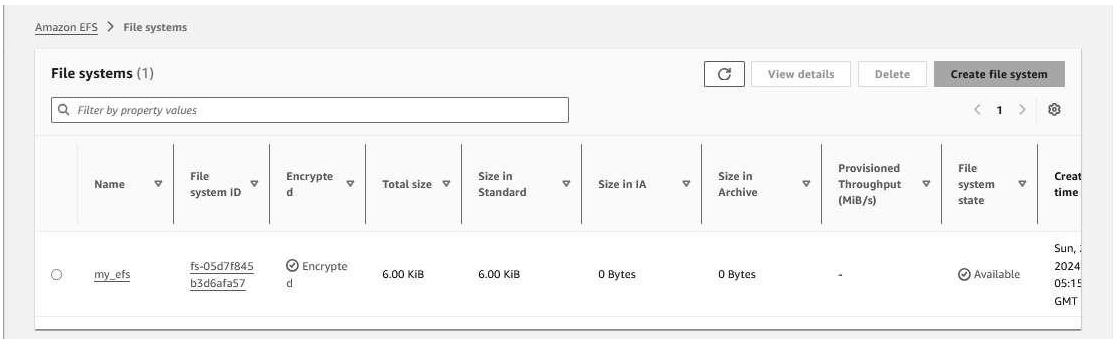
```
ubuntu@ip-172-31-21-116:~/volume/configmap$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
mysqlwala   1/1     Running   0          21s
```

```
 GNU nano 7.2                                                  genericsecreate.yml
apiVersion: v1
kind: Pod
metadata:
  name: secretwala
spec:
  containers:
    - name: mysqlwalacontainer
      image: mysql:8.0
      env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mygenericsecret
              key: mysql-password
```

```
ubuntu@ip-172-31-21-116:~/volume/secrete$ kubectl create secret generic mygenericsecret --from-literal=mysql-password=Pass@123
secret/mygenericsecret created
```

```
ubuntu@ip-172-31-21-116:~/volume/secrete$ kubectl apply -f genericsecreate.yml
pod/secretwala created
ubuntu@ip-172-31-21-116:~/volume/secrete$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
secretwala   1/1     Running   0          6s
ubuntu@ip-172-31-21-116:~/volume/secrete$
```

```
ubuntu@ip-172-31-21-116:~/pv$ nano pvfile.yml
ubuntu@ip-172-31-21-116:~/pv$ kubectl apply -f pvfile.yml
persistentvolume/mypv created
ubuntu@ip-172-31-21-116:~/pv$
```


Amazon EFS > File systems
File systems (1)
View details
Delete
Create file system
Filter by property values
1
Name
File system ID
Encrypted
Total size
Size in Standard
Size in IA
Size in Archive
Provisioned Throughput (MiB/s)
File system state
Creation time
my_efs
fs-05d7f845b3d6afa57
Encrypted
6.00 KiB
6.00 KiB
0 Bytes
0 Bytes
-
Available
Sun, 2024 05:15 GMT


Edit inbound rules Info
Inbound rules control the incoming traffic that's allowed to reach the instance.
Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
sgr-0d6cf0868981e434b
All traffic
All
All
Custom
sg-0dc61c303add03875
Delete
sgr-0934674f8907c65d2
HTTP
TCP
80
Custom
0.0.0.0/0
Delete
sgr-0ae618ac11b62dd86
NFS
TCP
2049
Custom
sg-089909938164c538b
sg-089909938164c538b
Delete
Add rule
Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.
Cancel
Preview changes
Save rules

```
ubuntu@ip-172-31-94-46:~$ #workernode
ubuntu@ip-172-31-94-46:~$ ls
myvolume
ubuntu@ip-172-31-94-46:~$ mkdir mydir
ubuntu@ip-172-31-94-46:~$ ls
mydir  myvolume
ubuntu@ip-172-31-94-46:~$ sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-05d7f845b3d6afa57.efs
.us-east-1.amazonaws.com:/ ~/mydir/
ubuntu@ip-172-31-94-46:~$
```

```
ubuntu@ip-172-31-94-46:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [309 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.7 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:14 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [7224 B]
Get:15 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.0 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 912 kB in 1s (815 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-94-46:~$
```

```
ubuntu@ip-172-31-94-46:~$ sudo apt-get install nfs-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nfs-common is already the newest version (1:2.6.4-3ubuntu5).
0 upgraded, 0 newly installed, 0 to remove and 44 not upgraded.
ubuntu@ip-172-31-94-46:~$
```

```
 GNU nano 7.2                                                  index.html *
<h1>This is pv&pvc efs storage efs index.html file</h1>
```

```
ubuntu@ip-172-31-94-46:~$ cd mydir
ubuntu@ip-172-31-94-46:~/mydir$ sudo mkdir test
ubuntu@ip-172-31-94-46:~/mydir$ cd test
ubuntu@ip-172-31-94-46:~/mydir/test$ sudo nano index.html
ubuntu@ip-172-31-94-46:~/mydir/test$
```

```
 GNU nano 7.2                                                  pvfile.yml *
apiVersion: v1
kind: PersistentVolume
metadata:
   name: mypv
   labels:
     app: dev
spec:
  capacity:
     storage: 20Gi
  accessModes:
     - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  nfs:
   server: fs-05d7f845b3d6afa57.efs.us-east-1.amazonaws.com
   path: /test
```

```
ubuntu@ip-172-31-21-116:~/pv$ nano pvfile.yml
ubuntu@ip-172-31-21-116:~/pv$ kubectl apply -f pvfile.yml
persistentvolume/mypv created
ubuntu@ip-172-31-21-116:~/pv$
```

```
 GNU nano 7.2                                                  pvcfile.yml *
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
   name: mypvc
   labels:
     app: dev
spec:
  accessModes:
     - ReadWriteOnce
  resources:
   requests:
    storage: 10Gi
  selector:
    matchLabels:
      app: dev
```

```
ubuntu@ip-172-31-21-116:~/pv$ nano pvcfile.yml
ubuntu@ip-172-31-21-116:~/pv$ kubectl apply -f pvcfile.yml
persistentvolumeclaim/mypvc created
ubuntu@ip-172-31-21-116:~/pv$
```

```
ubuntu@ip-172-31-21-116:~/pv$ kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM           STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
mypv   20Gi       RWO            Retain           Bound    default/mypvc                  <unset>                          83s
ubuntu@ip-172-31-21-116:~/pv$ kubectl get pvc
NAME    STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mypvc   Bound    mypv     20Gi       RWO                           <unset>                 27s
ubuntu@ip-172-31-21-116:~/pv$
```

```
 GNU nano 7.2                                    podfile.yml *
apiVersion: v1
kind: Pod
metadata:
  name: mypod
  labels:
    app: dev
spec:
   containers:
     - image: nginx
       name: mycont
       ports:
        - containerPort: 80
       volumeMounts:
         - name: my-volume
           mountPath: /usr/share/nginx/html
   volumes:
      - name: my-volume
        persistentVolumeClaim:
           claimName: mypvc
```

```
ubuntu@ip-172-31-21-116:~/pv$ nano podfile.yml
ubuntu@ip-172-31-21-116:~/pv$ kubectl apply -f podfile.yml
pod/mypod created
ubuntu@ip-172-31-21-116:~/pv$
```

```
ubuntu@ip-172-31-21-116:~/pv$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
mypod   1/1     Running   0          24s
ubuntu@ip-172-31-21-116:~/pv$
```

```
ubuntu@ip-172-31-21-116:~/pv$ kubectl exec -it mypod -- /bin/bash
root@mypod:/# cd /usr/share/nginx/html
root@mypod:/usr/share/nginx/html# ls
index.html
root@mypod:/usr/share/nginx/html#
```

DaemonSet –

A daemonSet in Kubernetes ensures that a copy of a specific Pod is running on all (or some) nodes in a cluster. It's commonly used to deploy system-level workloads like logging agents, monitoring daemons, or other infrastructure-related components.

Key Features of DaemonSet :

- **Node-Level Deployment**: Ensures that one Pod is running on each eligible node.
- **Dynamic Scaling**: Automatically deploys Pods to new nodes when they are added to the cluster.
- **Selective Deployment**: Can be restricted to specific nodes using node selectors, affinity rules, or taints and tolerations.
- **Self-Healing**: Automatically redeploys Pods if they fail or if the node, they are on becomes unavailable.

```
ubuntu@ip-172-31-21-116:~$ cd daemonsetwala/
ubuntu@ip-172-31-21-116:~/daemonsetwala$ ls
ubuntu@ip-172-31-21-116:~/daemonsetwala$ nano daemonsetfiel.yml
```

```
 GNU nano 7.2                                  daemonsetfiel.yml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: daemonwala
  labels:
    app: promi
spec:
  selector:
    matchLabels:
      app: promi
  template:
    metadata:
      labels:
        app: promi
    spec:
      containers:
        - name: daemonwala
          image: influxdb
          ports:
            - containerPort: 80
```

```
ubuntu@ip-172-31-21-116:~/daemonsetwala$ kubectl apply -f daemonsetfiel.yml
daemonset.apps/daemonwala created
```

```
ubuntu@ip-172-31-21-116:~/daemonsetwala$ kubectl get ds
NAME        DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonwala   1         1         1       1            1           <none>          2m20s
ubuntu@ip-172-31-21-116:~/daemonsetwala$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
daemonwala-c2ng4   1/1     Running   0          2m25s
ubuntu@ip-172-31-21-116:~/daemonsetwala$
```

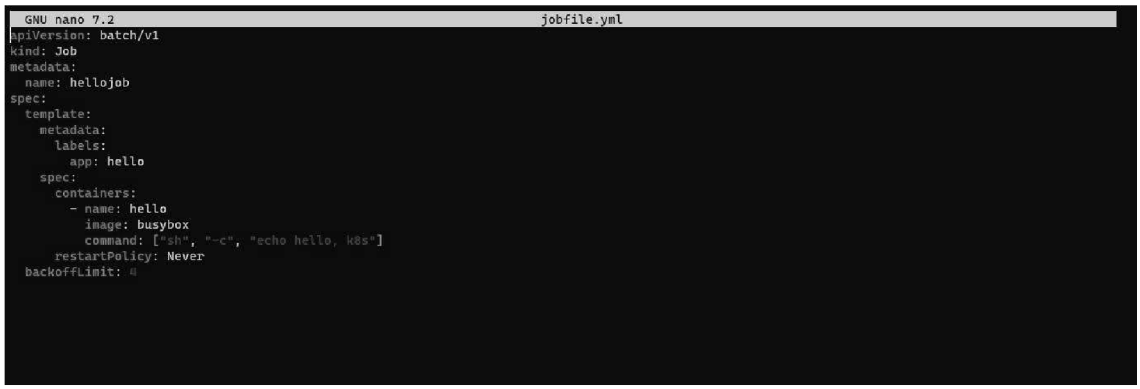
Jobs –

A Job creates one or more Pods and will continue to retry execution of the Pods until a specified number of them successfully terminate. As pods successfully complete, the Job tracks the successful completions. When a specified number of successful completions is reached, the task (ie, Job) is complete. Deleting a Job will clean up the Pods it created. Suspending a Job will delete its active Pods until the Job is resumed again.

Definition: A Kubernetes Job is a higher-level abstraction that manages one or more Pods to ensure a task runs to completion successfully.

Purpose: Ideal for batch or one-time tasks like data processing, backups, or running scripts.

Lifecycle: The Job ensures that the specified number of Pods complete successfully.

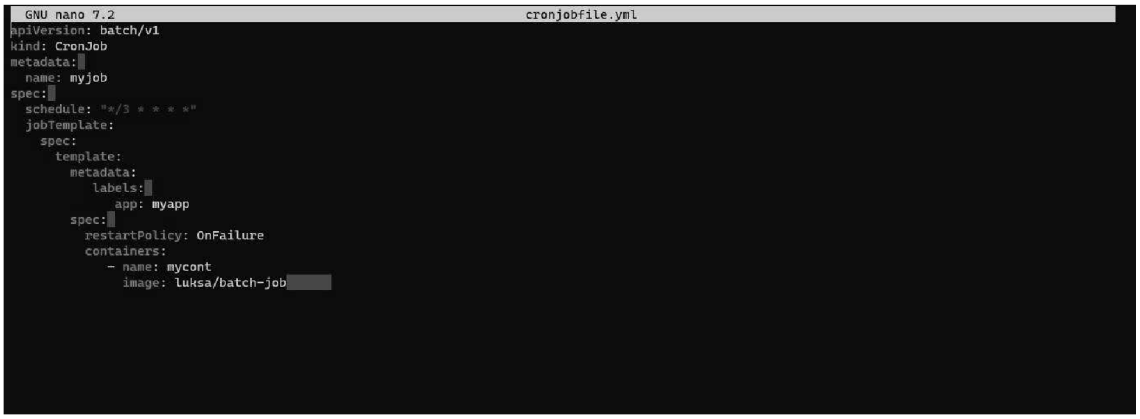


```
 GNU nano 7.2                                                    jobfile1.yml
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob
spec:
  completions: 3
  parallelism: 2
  template:
    metadata:
      labels:
        app: myapp
    spec:
      restartPolicy: OnFailure
      containers:
        - name: mycont
          image: luksa/batch-job
```

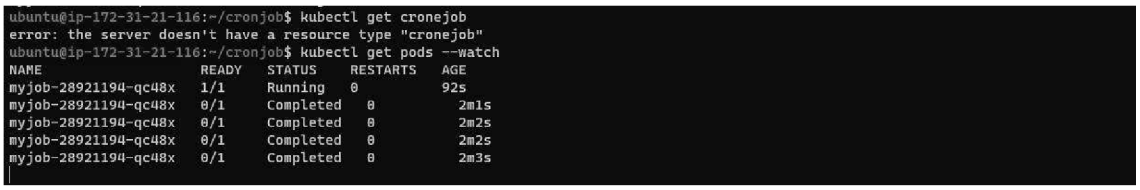
```
ubuntu@ip-172-31-21-116:~/job$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
example-statefulset-0    0/1     Pending   0          15h
myjob-bkjct              1/1     Running   0          37s
myjob-lrjtz              1/1     Running   0          37s
ubuntu@ip-172-31-21-116:~/job$
```

Cronjob –

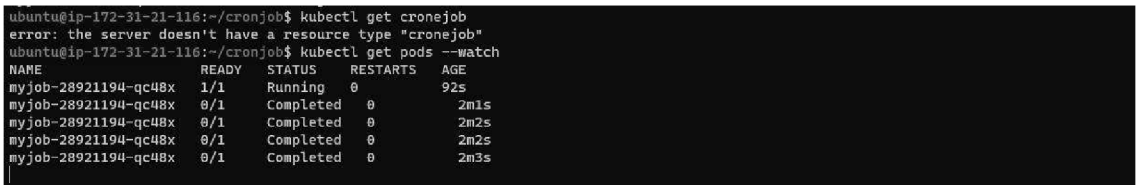
A corn job is a scheduled task in Unix-like operating systems that allows you to automate repetitive tasks at specified times or intervals. These tasks are executed by the corn daemon.



```
 GNU nano 7.2                                                cronjobfile.yml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: myjob
spec:
  schedule: "*/3 * * * *"
  jobTemplate:
    spec:
      template:
        metadata:
          labels:
            app: myapp
        spec:
          restartPolicy: OnFailure
          containers:
            - name: mycont
              image: luksa/batch-job
```



```
ubuntu@ip-172-31-21-116:~/cronjob$ kubectl get cronejob
error: the server doesn't have a resource type "cronejob"
ubuntu@ip-172-31-21-116:~/cronjob$ kubectl get pods --watch
NAME                    READY   STATUS    RESTARTS   AGE
myjob-28921194-qc48x    1/1     Running   0          92s
myjob-28921194-qc48x    0/1     Completed   0          2m1s
myjob-28921194-qc48x    0/1     Completed   0          2m2s
myjob-28921194-qc48x    0/1     Completed   0          2m2s
myjob-28921194-qc48x    0/1     Completed   0          2m3s
```



```
ubuntu@ip-172-31-21-116:~/cronjob$ kubectl get cronejob
error: the server doesn't have a resource type "cronejob"
ubuntu@ip-172-31-21-116:~/cronjob$ kubectl get pods --watch
NAME                    READY   STATUS    RESTARTS   AGE
myjob-28921194-qc48x    1/1     Running   0          92s
myjob-28921194-qc48x    0/1     Completed   0          2m1s
myjob-28921194-qc48x    0/1     Completed   0          2m2s
myjob-28921194-qc48x    0/1     Completed   0          2m2s
myjob-28921194-qc48x    0/1     Completed   0          2m3s
```

StatefulSet -

A **S**tatefulSet is a Kubernetes API object used to manage and deploy applications that require stable and unique network identifiers, persistent storage, and ordered or graceful deployment and scaling. It's ideal for stateful applications like databases, messaging systems, and distributed systems.

Key Features of StatefulSet:

1. Stable Pod Identity:
    - Each pod in a StatefulSet gets a unique, stable hostname and identity (e.g., pod-name-0, pod-name-1).
    - This identity persists across pod restarts.
2. Stable Storage:
    - Each pod can be associated with its own PersistentVolume (PV), ensuring data persists even if the pod is rescheduled or restarted.
3. Ordered Deployment and Scaling:
    - Pods are deployed, updated, or deleted sequentially.
    - Ensures that dependent systems can rely on the order of operations.
4. Graceful Rollout and Termination:
    - Pods are scaled up or down gracefully to ensure consistent application behavior.
5. DNS Management:
    - StatefulSets automatically assign DNS names to pods, simplifying service discovery

```
 GNU nano 7.2                                    statefulset.yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mypod3
spec:
  replicas: 3
  serviceName: mypod-service
  selector:
    matchLabels:
      app: dev
  template:
    metadata:
      labels:
        app: dev
    spec:
      containers:
      - image: nginx
        name: mycont
        ports:
        - containerPort: 80
```

Horizontal Pod Autoscaling –

In Kubernetes, a *HorizontalPodAutoscaler* automatically updates a workload resource (such as deployment or statefullet), with the aim of automatically scaling the workload to match demand.

Horizontal scaling means that the response to increased load is to deploy more pods. This is different from *vertical* scaling, which for Kubernetes would mean assigning more resources (for example: memory or CPU) to the Pods that are already running for the workload.

If the load decreases, and the number of Pods is above the configured minimum, the HorizontalPodAutoscaler instructs the workload resource (the Deployment, StatefulSet, or other similar resource) to scale back down.

Horizontal pod autoscaling does not apply to objects that can't be scaled (for example: a daemonset.)

The HorizontalPodAutoscaler is implemented as a Kubernetes API resource and a controller. The resource determines the behavior of the controller. The horizontal pod autoscaling controller, running within the Kubernetes control plane, periodically adjusts the desired scale of its target (for example, a Deployment) to match observed metrics such as average CPU utilization, average memory utilization, or any other custom metric you specify.

```
ubuntu@ip-172-31-21-116:~$ mkdir autoscaling
ubuntu@ip-172-31-21-116:~$ cd autoscaling/
ubuntu@ip-172-31-21-116:~/autoscaling$ ls
ubuntu@ip-172-31-21-116:~/autoscaling$ nano components.yml
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl apply -f components.yml
```

```
 GNU nano 7.2                                components.yml
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: metrics-server
  name: metrics-server
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    k8s-app: metrics-server
    rbac.authorization.k8s.io/aggregate-to-admin: "true"
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
    rbac.authorization.k8s.io/aggregate-to-view: "true"
  name: system:aggregated-metrics-reader
rules:
- apiGroups:
  - metrics.k8s.io
  resources:
  - pods
  - nodes
  verbs:
  - get
  - list
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    k8s-app: metrics-server
                          [ Read 203 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute   ^C Location  M-U Undo    M-A Set Mark  M-] To Bracket
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo    M-6 Copy      ^Q Where Was
```

```
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl get pods  -n kube-system
NAME                                       READY   STATUS             RESTARTS          AGE
calico-kube-controllers-6cdb97b867-zk6tr   1/1     Running            16 (5m6s ago)     17d
calico-node-8k8bg                          0/1     Running            120 (51s ago)     17d
calico-node-j5mqz                          0/1     Running            125 (2m58s ago)   17d
coredns-55cb58b774-84665                   1/1     Running            16 (5m6s ago)     17d
coredns-55cb58b774-vhm2b                   1/1     Running            3 (5m6s ago)      3d22h
etcd-ip-172-31-21-116                      1/1     Running            15 (5m6s ago)     17d
kube-apiserver-ip-172-31-21-116            1/1     Running            15 (5m6s ago)     17d
kube-controller-manager-ip-172-31-21-116   1/1     Running            15 (5m6s ago)     17d
kube-proxy-96fds                           1/1     Running            134 (64s ago)     17d
kube-proxy-d572p                           0/1     CrashLoopBackOff   134 (54s ago)     17d
kube-scheduler-ip-172-31-21-116            1/1     Running            15 (5m6s ago)     17d
metrics-server-554cf459c5-9njq9            1/1     Running            0                 2m23s
ubuntu@ip-172-31-21-116:~/autoscaling$
```

```
ubuntu@ip-172-31-21-116:~/autoscaling$ nano mynginx.yml
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl apply -f mynginx.yml
deployment.apps/mydeployment created
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
mydeployment-545fc4bd99-727ds   1/1     Running   0          5s
ubuntu@ip-172-31-21-116:~/autoscaling$
```

```
 GNU nano 7.2                                                    mynginx.yml
apiVersion: apps/v1
kind: Deployment
metadata:
   name: mydeployment
spec:
  replicas: 1
  strategy:
     type: RollingUpdate
     rollingUpdate:
         maxSurge: 2
         maxUnavailable: 0
  selector:
   matchLabels:
      env: dev
  template:
       metadata:
         labels:
             env: dev
       spec:
            containers:
                - name: nginxcontaine
                  image: nginx
                  ports:
                    - containerPort: 80
                  resources:
                        requests:
                          cpu: "100m"
                        limits:
                          cpu: "500m"
```

```
ubuntu@ip-172-31-21-116:~/autoscaling$ nano hpafile.yml
ubuntu@ip-172-31-21-116:~/autoscaling$ kuubectl apply -f hpafile.yml
Command 'kuubectl' not found, did you mean:
  command 'kubectl' from snap kubectl (1.31.4)
See 'snap info <snapname>' for additional versions.
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl apply -f hpafile.yml
horizontalpodautoscaler.autoscaling/nginxhpa created
ubuntu@ip-172-31-21-116:~/autoscaling$
```

```
 GNU nano 7.2                                                    hpafile.yml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
       name: nginxhpa
spec:
  scaleTargetRef:
     apiVersion: apps/v1
     kind: Deployment
     name: mydeployment
  minReplicas: 2
  maxReplicas: 6
  metrics:
  - type: Resource
    resource:
       name: cpu
       target:
         type: Utilization
         averageUtilization: 10
```

```
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl get hpa
NAME       REFERENCE                  TARGETS              MINPODS   MAXPODS   REPLICAS   AGE
nginxhpa   Deployment/mydeployment   cpu: <unknown>/10%   2         6         2          69s
ubuntu@ip-172-31-21-116:~/autoscaling$
```

```
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
mydeployment-545fc4bd99-727ds   1/1     Running   0          3m55s
mydeployment-545fc4bd99-kpfbh   1/1     Running   0          86s
ubuntu@ip-172-31-21-116:~/autoscaling$
```

```
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl run trafficpod --image=busybox -- /bin/sh -c "while true; do wget -q -O - http://192.168.205.179; don
e"
pod/trafficpod created
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
mydeployment-545fc4bd99-727ds   1/1     Running   0          9m20s
mydeployment-545fc4bd99-kpfbh   1/1     Running   0          6m51s
trafficpod                      1/1     Running   0          7s
ubuntu@ip-172-31-21-116:~/autoscaling$
```

```
ubuntu@ip-172-31-21-116:~/autoscaling$ kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
cpu-stress                      1/1     Running   0          3m
mydeployment-545fc4bd99-4v8s8   1/1     Running   0          4m59s
mydeployment-545fc4bd99-kpfbh   1/1     Running   0          16m
trafficpod                      1/1     Running   0          9m29s
trafficpod2                     1/1     Running   0          5m32s
ubuntu@ip-172-31-21-116:~/autoscaling$
```

Ingress –

Ingress in the context of Kubernetes refers to a set of rules that govern how external access to services within a Kubernetes cluster is managed. It acts as an entry point to the cluster, routing external HTTP/HTTPS traffic to the appropriate services.

Benefits of Using Ingress:

1. **Single Entry Point**: Centralized management of external access.
2. **Load Balancing**: Balances traffic across services.
3. **TLS/SSL Termination**: Secure communication with HTTPS.
4. **Path-Based Routing**: Route traffic based on URL paths.
5. **Name-Based Virtual Hosting**: Use multiple domain names to access different services.

```
ubuntu@ip-172-31-21-116:~$ mkdir ingress
ubuntu@ip-172-31-21-116:~$ cd ingress
ubuntu@ip-172-31-21-116:~/ingress$ ls
ubuntu@ip-172-31-21-116:~/ingress$ nano ingressfile.yml
ubuntu@ip-172-31-21-116:~/ingress$ kubectl apply -f ingressfile.yml
ingress.networking.k8s.io/myingress created
```

```
 GNU nano 7.2                                    ingressfile.yml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: myingress
spec:
  rules:
  - host: tujanena.com
    http:
     paths:
      - path: /
        pathType: Prefix
        backend:
         service:
           name: myservice1
           port:
             number: 80
  - host: example.com
    http:
     paths:
      - path: /
        pathType: Prefix
        backend:
         service:
           name: myservice2
           port:
             number: 80
```

```
ubuntu@ip-172-31-21-116:~/ingress$ kubectl get ingress
NAME        CLASS    HOSTS                   ADDRESS   PORTS   AGE
myingress   <none>   tujanena.com,example.com           80      3m1s
```