# Deployment strategy in kubernetes

1. Pods
2. Replication Controller
3. ReplicaSet
4. Deployment
5. DaemonSet

## 1. Pods – Let's perform some tasks using Pods strategy.

Alright! Now, we will create a **Pod** using the **file method** in Kubernetes instead of using a Deployment. This approach is useful when you want full control over individual Pods without using a ReplicaSet or Deployment.

## Steps to Create a Pod Using a YAML File

1. **Create a YAML file** defining the Pod.
2. **Apply the YAML file** using kubectl.
3. **Verify the Pod status**.

**1. Create a YAML file** defining the Pod.

```
vim pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
        name: mypod
spec:
        containers:
              - name: test
                image: nginx
```

kubectl create -f pod.yml

```
root@master-node:~# vim pod.yml
root@master-node:~# kubectl create -f pod.yml
pod/mypod created
root@master-node:~#
```

```
root@master-node:~# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
mypod   1/1     Running   0          56s
root@master-node:~#
```

kubectl describe pod mypod

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  100s  default-scheduler  Successfully assigned default/mypod to worker1-node.
com
  Normal  Pulling    99s   kubelet            Pulling image "nginx"
  Normal  Pulled     99s   kubelet            Successfully pulled image "nginx" in 259ms (259ms in
cluding waiting). Image size: 72188133 bytes.
  Normal  Created    98s   kubelet            Created container test
  Normal  Started    98s   kubelet            Started container test
root@master-node:~#
```

kubectl delete pod mypod –force

```
root@master-node:~# kubectl delete pod mypod --force
Warning: Immediate deletion does not wait for confirmation that the running resource has been term
inated. The resource may continue to run on the cluster indefinitely.
pod "mypod" force deleted
root@master-node:~#
```

```
root@master:~# kubectl delete -f pod.yaml
pod "rupesh" deleted
root@master:~#
root@master:~# kubectl get pods
```

**Next – How we can run a command inside the container**

kubectl exec mypod -- hostname

```
root@master-node:~# kubectl exec mypod -- hostname
mypod
root@master-node:~# kubectl exec mypod -- pwd
/
root@master-node:~# kubectl exec mypod -- ls
bin
boot
dev
docker-entrypoint.d
docker-entrypoint.sh
etc
```

**Login into the pod.**

```
kubectl exec mypod -it -- bash
```

```
root@master-node:~#
root@master-node:~# kubectl exec mypod -it -- bash
root@mypod:/# pwd
/
root@mypod:/#
```

```
root@master-node:~# kubectl logs mypod
/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
2025/02/19 13:11:13 [notice] 1#1: using the "epoll" event method
2025/02/19 13:11:13 [notice] 1#1: nginx/1.27.4
2025/02/19 13:11:13 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/02/19 13:11:13 [notice] 1#1: OS: Linux 6.8.0-1021-aws
2025/02/19 13:11:13 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/02/19 13:11:13 [notice] 1#1: start worker processes
2025/02/19 13:11:13 [notice] 1#1: start worker process 29
root@master-node:~#
```

**Copy content inside the pod**

```
kubectl cp abc mypod:/
```

```
root@master-node:~# kubectl cp abc mypod:/
root@master-node:~#
root@master-node:~# kubectl exec mypod -- ls
abc
bin
boot
```

**How we can assign labels to pods**

```
vim pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
        name: mypod
        labels:
                app: wordpress
                env: prod
spec:
        containers:
              - nam
                image: nginx
                ports:
                    - containerPort: 80
```

```
kubectl create -f pod.yml

kubectl describe pod mypod
```

```
root@master-node:~# kubectl create -f pod.yml
pod/mypod created
root@master-node:~# kubectl describe pod mypod
Name:             mypod
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker1-node.com/172.31.17.146
Start Time:       Wed, 19 Feb 2025 14:00:20 +0000
Labels:           app=wordpress
                  env=prod
Annotations:      <none>
Status:           Running
IP:               10.46.0.1
IPs:
  IP:  10.46.0.1
```

kubectl get pods --show-labels

```
root@master-node:~# kubectl get pods --show-labels
NAME    READY   STATUS    RESTARTS   AGE     LABELS
mypod   1/1     Running   0          4m45s   app=wordpress,env=prod
root@master-node:~#
```

## Why Do We Assign Labels to Pods in Kubernetes?

Labels in Kubernetes are **key-value pairs** assigned to objects (like Pods) to organize, identify, and manage them efficiently. They help in **grouping, selecting, and filtering** resources.

## Key Reasons for Using Labels in Pods

### ☑ *1. Logical Grouping of Pods*

- Helps categorize Pods based on environment, application, or role.
- Example:

```
labels:
  app: webserver
  environment: production
```

- Now, all Pods running the **webserver** in the **production** environment can be easily managed together.

### ☑ *2. Service Discovery & Load Balancing*

- Services use labels to select the right Pods for traffic routing.
- Example:

```
selector:
  app: webserver
```

- This ensures that the **Service** only routes traffic to Pods with the app: webserver label.

### ☑ *3. Managing Deployments & Scaling*

- Labels help **ReplicationControllers, ReplicaSets, and Deployments** manage specific sets of Pods.
- Example:

```
selector:
  matchLabels:
    app: database
```

- A **ReplicaSet** will ensure the required number of database Pods always runs.

### ☑ *4. Easy Filtering & Querying*

- Helps in searching for specific Pods quickly.
- Example:

```
kubectl get pods -l app=webserver
```

- Lists only the Pods labeled as webserver.

### ☑ *5. Rollouts & Canary Deployments*

- Helps in **progressive deployment** and testing new versions before full rollout.
- Example:

```
labels:
  app: webserver
  version: v1
```

- Later, a v2 version can be tested by selecting only version: v2 Pods.

## 2. ReplicationController – Let's perform some tasks using replication controller

```
# vim rc.yaml
apiVersion: v1
kind: ReplicationController
metadata:
 name: myapp
spec:
 replicas: 10
 template:
  metadata:
   name: xyz
   labels:
    app: webserver
```

```
spec:
  containers:
    - name: sdfdf
      image: httpd
```

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
        name: myapp
spec:
        replicas: 10
        template:
            metadata:
              name: xyz
              labels:
                   app: webserver
            spec:
              containers:
                 - name: sdfdf
                   image: httpd
```

```
kubectl create -f rc.yaml

kubectl get rc

kubectl get pods
```

```
root@master-node:~# kubectl create -f rc.yaml
replicationcontroller/myapp created
root@master-node:~#
root@master-node:~# kubectl get rc
NAME    DESIRED   CURRENT   READY   AGE
myapp   10        10        10      19s
root@master-node:~#
root@master-node:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
myapp-5f9mv   1/1     Running   0          30s
myapp-bzxfh   1/1     Running   0          30s
myapp-czfbd   1/1     Running   0          30s
myapp-kgwxv   1/1     Running   0          30s
myapp-kxxcw   1/1     Running   0          30s
myapp-m697l   1/1     Running   0          30s
myapp-p475k   1/1     Running   0          30s
myapp-r8pqp   1/1     Running   0          30s
myapp-s6htg   1/1     Running   0          30s
myapp-tv885   1/1     Running   0          30s
mypod         1/1     Running   0          19m
root@master-node:~#
```

Yes! Since you're using a **ReplicationController**, it ensures that the desired number of replicas (Pods) is always running. If you manually delete a Pod, the ReplicationController will **automatically create a new Pod** to maintain the specified count.

```
root@master-node:~# kubectl delete pod myapp-5f9mv
pod "myapp-5f9mv" deleted
root@master-node:~#
root@master-node:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
myapp-bzxfh   1/1     Running   0          6m11s
myapp-czfbd   1/1     Running   0          6m11s
myapp-kgwxv   1/1     Running   0          6m11s
myapp-kxxcw   1/1     Running   0          6m11s
myapp-m697l   1/1     Running   0          6m11s
myapp-n2nps   1/1     Running   0          7s
myapp-p475k   1/1     Running   0          6m11s
myapp-r8pqp   1/1     Running   0          6m11s
myapp-s6htg   1/1     Running   0          6m11s
myapp-tv885   1/1     Running   0          6m11s
mypod         1/1     Running   0          24m
root@master-node:~#
```

The **ReplicationController** continuously monitors the state of the cluster. If the number of running Pods is **less than the desired replicas (10 in your case)**, it will automatically create a new Pod to match the desired state.

**Delete RC**

```
kubectl delete rc myapp
```

```
root@master-node:~# kubectl delete rc myapp
replicationcontroller "myapp" deleted
root@master-node:~#
root@master-node:~# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
mypod   1/1     Running   0          40m
root@master-node:~#
```

## 3. ReplicaSet – Let's perform some tasks using ReplicatSet

### Difference Between ReplicationController and ReplicaSet in Kubernetes

Both **ReplicationController** and **ReplicaSet** are used to ensure that a specified number of Pod replicas are running in Kubernetes. However, **ReplicaSet is the modern replacement for ReplicationController** with more advanced features.

# 1. ReplicationController (Old)

☑ **Ensures a fixed number of Pod replicas**
☑ **Restarts missing Pods** automatically
❌ **Uses exact match selectors** (less flexible)
❌ **Deprecated in favor of ReplicaSet**

### Example: ReplicationController YAML

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: myapp
spec:
```

```
  replicas: 3
  selector:
    app: webserver
  template:
    metadata:
      labels:
        app: webserver
    spec:
      containers:
        - name: nginx
          image: nginx
```

## 2. ReplicaSet (New & Improved)

☑ **Improved version of ReplicationController**
☑ **Supports "Set-Based" selectors** (more flexible)
☑ **Works with Deployments** for better rollouts
❌ **Still not commonly used directly (Deployments are preferred)**

### Example: ReplicaSet YAML

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webserver
  template:
    metadata:
      labels:
        app: webserver
    spec:
      containers:
```

```
        - name: nginx
          image: nginx
```

**Let's create a ReplicaSet**

```
vim  rs.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: mywebsite
spec:
 selector:
  matchLabels:
   app: webserver
 replicas: 20
 template:
  metadata:
   name: xyz
   labels:
    app: webserver
  spec:
   containers:
    - name: sdfdf
      image: httpd
```

```
root@master-node:~# vim rs.yaml
root@master-node:~# kubectl create -f rs.yaml
replicaset.apps/mywebsite created
root@master-node:~# kubectl get rs
NAME        DESIRED   CURRENT   READY   AGE
mywebsite   20        20        20      6s
root@master-node:~# kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
mypod             1/1     Running   0          3h4m
mywebsite-24w5c   1/1     Running   0          17s
mywebsite-2bbvs   1/1     Running   0          17s
```

Okay, now let's change the desired state of the ReplicaSet. Here, we can **scale up and scale down** as needed. For example, if I currently have **20 pods** and I set the replicas to **15**, it will delete **5 pods**, leaving a total of **15 pods** running.

kubectl scale rs --replicas 15 mywebsite

```
root@master-node:~# kubectl scale rs --replicas 15 mywebsite
replicaset.apps/mywebsite scaled
root@master-node:~#
root@master-node:~# kubectl get rs
NAME        DESIRED   CURRENT   READY   AGE
mywebsite   15        15        15      4m57s
root@master-node:~#
```

We can scale up and scale down by modifying the file as well. **Let's do it.**

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: mywebsite
spec:
  selector:
    matchLabels:
      app: webserver
  replicas: 10
  template:
    metadata:
      name: xyz
      labels:
        app: webserver
    spec:
      containers:
        - name: sdfdf
          image: httpd
```

kubectl replace -f rs.yaml

```
root@master-node:~# kubectl replace -f rs.yaml
replicaset.apps/mywebsite replaced
root@master-node:~# kubectl get rs
NAME        DESIRED   CURRENT   READY   AGE
mywebsite   10        10        10      8m32s
root@master-node:~# ~^C
root@master-node:~#
```

## 3. Deployment

Let's create a deployment.

```
vim deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
        name: mywebsite
spec:
       selector:
          matchLabels:
             app: webserver
      replicas: 50
      template:
             metadata:
                name: xyz
                labels:
                    app: webserver
             spec:
                    containers:
                          - name: sdfdf
                            image: httpd
```

```
kubectl create -f deployment.yaml
```

```
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   50/50   50           50          2m45s
root@master-node:~#
root@master-node:~# kubectl get  rs
NAME                   DESIRED   CURRENT   READY   AGE
mywebsite-84ffd6ccdf   50        50        50      2m51s
root@master-node:~#
```

Here we can see here Deployment automatically create a replicaset.

Okay, now let's perform a **rollout**.

We will update the **nginx image version** in our deployment.yaml file, meaning we will deploy a different version.

In this process:

- The **old pods will terminate**.
- **New pods** with the updated image version will be created to match the desired state.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mywebsite
spec:
  selector:
    matchLabels:
      app: webserver
  replicas: 50
  template:
    metadata:
      name: xyz
      labels:
        app: webserver
    spec:
      containers:
        - name: sdfdf
          image: nginx:1.22
```

```
kubectl replace -f deployment.yaml
```

```
root@master-node:~# kubectl replace -f deployment.yaml
deployment.apps/mywebsite replaced
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   38/50   25           38          12m
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   38/50   50           38          12m
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   40/50   50           40          12m
root@master-node:~#
```

No again rollout with nginx latest version

vim deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mywebsite
spec:
  selector:
    matchLabels:
      app: webserver
  replicas: 50
  template:
    metadata:
      name: xyz
      labels:
        app: webserver
    spec:
      containers:
        - name: sdfdf
          image: nginx:latest
```

kubectl replace -f deployment.yaml

```
root@master-node:~# kubectl replace -f deployment.yaml
deployment.apps/mywebsite replaced
root@master-node:~#
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   38/50   25           38          15m
root@master-node:~#
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   38/50   41           38          15m
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   38/50   50           38          15m
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   50/50   50           50          15m
root@master-node:~# vim deployment.yaml
root@master-node:~#
```

```
root@master-node:~# kubectl rollout status deployment mywebsite
deployment "mywebsite" successfully rolled out
root@master-node:~#
root@master-node:~#
```

kubectl describe deployments.apps

```
root@master-node:~# kubectl describe deployments.apps
Name:                   mywebsite
Namespace:              default
CreationTimestamp:      Thu, 20 Feb 2025 17:39:48 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 3
Selector:               app=webserver
Replicas:               50 desired | 50 updated | 50 total | 50 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=webserver
  Containers:
   sdfdf:
    Image:         nginx:latest
    Port:          <none>
    Host Port:     <none>
    Environment:   <none>
```

kubectl get rs

```
root@master-node:~# kubectl get rs
NAME                   DESIRED   CURRENT   READY   AGE
mywebsite-55d8c74676   0         0         0       7m56s
mywebsite-5979986654   50        50        50      5m1s
mywebsite-84ffd6ccdf   0         0         0       20m
root@master-node:~#
```

kubectl rollout history deployment mywebsite

```
root@master-node:~# kubectl rollout history deployment mywebsite
deployment.apps/mywebsite
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
3         <none>

root@master-node:~#
```

kubectl set image deployment mywebsite nginx=nginx:latest –record

```
root@master-node:~# kubectl set image deployment mywebsite nginx=nginx:latest --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/mywebsite image updated
error: unable to find container named "nginx"
root@master-node:~# kubectl rollout history deployment mywebsite
deployment.apps/mywebsite
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
3         kubectl set image deployment mywebsite nginx=nginx:latest --record=true

root@master-node:~#
```

kubectl rollout history deployment mywebsite --revision 1

```
root@master-node:~# kubectl rollout history deployment mywebsite --revision 1
deployment.apps/mywebsite with revision #1
Pod Template:
  Labels:       app=webserver
        pod-template-hash=84ffd6ccdf
  Containers:
   sdfdf:
    Image:      httpd
    Port:       <none>
    Host Port:  <none>
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
  Node-Selectors:       <none>
  Tolerations:  <none>

root@master-node:~#
```

kubectl rollout undo deployment mywebsite --to-revision 1

```
root@master-node:~# kubectl rollout undo deployment mywebsite --to-revision 1
deployment.apps/mywebsite rolled back
root@master-node:~#
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   38/50   48           38          31m
root@master-node:~# kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
mywebsite-84ffd6ccdf-2wfqp   1/1     Running   0          14s
mywebsite-84ffd6ccdf-4gcbd   1/1     Running   0          14s
mywebsite-84ffd6ccdf-4l7g4   1/1     Running   0          13s
```

kubectl rollout history deployment mywebsite

```
root@master-node:~# kubectl rollout history deployment mywebsite
deployment.apps/mywebsite
REVISION  CHANGE-CAUSE
2         <none>
3         kubectl set image deployment mywebsite nginx=nginx:latest --record=true
4         <none>

root@master-node:~#
```

kubectl scale deployment mywebsite --replicas 60

```
root@master-node:~# kubectl scale deployment mywebsite --replicas 60
deployment.apps/mywebsite scaled
root@master-node:~#
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   60/60   60           60          36m
root@master-node:~#
```

kubectl edit deployments.apps mywebsite

```
metadata:
  annotations:
    deployment.kubernetes.io/revision: "4"
  creationTimestamp: "2025-02-20T17:39:48Z"
  generation: 6
  name: mywebsite
  namespace: default
  resourceVersion: "75167"
  uid: c1aa0879-3698-4aa4-aaf1-4cace888ac5c
spec:
  progressDeadlineSeconds: 600
  replicas: 75
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: webserver
```

```
root@master-node:~# kubectl edit deployments.apps mywebsite
deployment.apps/mywebsite edited
root@master-node:~#
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   64/75   75           64          41m
root@master-node:~# kubectl get  deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mywebsite   68/75   75           68          41m
```

```
root@master-node:~# kubectl delete deployments.apps mywebsite
deployment.apps "mywebsite" deleted
root@master-node:~#
root@master-node:~# kubectl get deployments.apps
No resources found in default namespace.
root@master-node:~#
root@master-node:~# kubectl get rs
No resources found in default namespace.
root@master-node:~#
```

kubectl create deployment --image httpd --replicas 7 test

```
root@master-node:~# kubectl create deployment --image httpd --replicas 7 test
deployment.apps/test created
root@master-node:~#
root@master-node:~# kubectl get deployments
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
test   7/7     7            7           6s
root@master-node:~# kubectl get rs
NAME              DESIRED   CURRENT   READY   AGE
test-6866fd55cd   7         7         7       9s
root@master-node:~#
```

kubectl create deployment web-server --replicas 20 --image nginx --dry-run=client -o yaml

```
root@master-node:~# kubectl create deployment web-server --replicas 20 --image nginx --dry-run=client -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: web-server
  name: web-server
spec:
  replicas: 20
  selector:
    matchLabels:
      app: web-server
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: web-server
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
root@master-node:~#
```

kubectl create deployment web-server --replicas 20 --image nginx --dry-run=client -o yaml > nginx.yaml

kubectl run neel --image httpd --dry-run=client -o yaml

```
root@master-node:~# kubectl run neel --image httpd --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: neel
  name: neel
spec:
  containers:
  - image: httpd
    name: neel
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
root@master-node:~#
```

kubectl run neel --image httpd --dry-run=client -o yaml > web.yaml