Pods, Services, and ReplicaSets in Kubernetes

Pods in Kubernetes

A **Pod** is the **smallest deployable unit** in Kubernetes. It represents a single instance of a running process in your cluster.

A Pod can contain:

- One container (most common case)
- Multiple tightly coupled containers (sidecar pattern)

Pods share:

- Network (same IP address)
- Storage volumes
- Lifecycle

Key Characteristics of Pods

- Each Pod gets a **unique IP address**
- Containers inside a Pod communicate via **localhost**
- Pods are **dynamic** (they can be recreated at any time)
- Pods are usually managed by higher-level controllers like Deployments or ReplicaSets

---

Pod Architecture

Example Pod YAML

```
apiVersion: v1

kind: Pod

metadata:

  name: nginx-pod

spec:

  containers:

  - name: nginx-container

    image: nginx
```

Pod Lifecycle Phases

- Pending
- Running
- Succeeded
- Failed

When to Use Pods Directly?

You typically **do not create Pods directly in production**. Instead, you use:

- ReplicaSet
- Deployment

Pods are ideal for:

- Testing
- Debugging
- Temporary workloads

Services in Kubernetes

A **Service** is an abstraction that defines a logical set of Pods and provides stable network access to them.

Since Pods are ephemeral and can change IPs, Services provide:

- Stable IP
- DNS name
- Load balancing

Why Do We Need Services?

Pods:

- Can die and restart
- Get new IP addresses

Services:

- Provide stable endpoint
- Load balance traffic
- Enable service discovery

Example Service YAML

```
apiVersion: v1
kind: Service
metadata:
  name: my-web-service
spec:
  selector:
    app: lblnginx
```

```
ports:
- protocol: TCP
  port: 80
  nodePort: 30001
type: NodePort
```

How Services Work Internally

- Service selects Pods using labels
- kube-proxy configures networking rules
- Traffic is distributed among matching Pods

ReplicaSet in Kubernetes

A **ReplicaSet** ensures that a specified number of Pod replicas are running at any given time.

If a Pod crashes:

- ReplicaSet automatically creates a new one

Why Use ReplicaSet?

- High availability
- Scalability
- Self-healing
- Fault tolerance

Example ReplicaSet YAML

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```

How ReplicaSet Works

1. You define desired replica count.
2. ReplicaSet creates Pods.
3. If Pods die → it recreates them.
4. If replicas increased → creates more Pods.
5. If replicas decreased → deletes extra Pods.

Relationship Between Pod, ReplicaSet, and Service

Here's how they work together:

1. **ReplicaSet** creates and maintains multiple Pods.
2. **Service** exposes those Pods.
3. **Pod** runs the actual container.