Deployments in Kubernetes

A **Deployment** in Kubernetes is a higher-level controller that manages **ReplicaSets** and provides declarative updates to Pods.

It ensures:

- Desired number of replicas are running
- Rolling updates of applications
- Rollbacks to previous versions
- Self-healing capabilities

In production environments, you typically use a Deployment instead of creating ReplicaSets directly.

Why Use Deployment?

Without Deployment:

- You manually manage ReplicaSets
- No easy rollback
- No controlled rolling updates

With Deployment:

- Automated rollout
- Version history tracking
- Easy rollback
- Zero-downtime upgrades (if configured properly)

How Deployment Works

1. You define desired state in YAML.
2. Deployment creates a ReplicaSet.
3. ReplicaSet creates Pods.
4. If Pods crash → ReplicaSet recreates them.
5. If image version changes → Deployment creates new ReplicaSet.
6. Old ReplicaSet is scaled down gradually.

Example Deployment YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
```

```
      app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25
        ports:
        - containerPort: 80
```

Key Components of Deployment

replicas

Number of Pod copies to run.

selector

Matches Pods managed by this Deployment.

template

Defines Pod configuration.

strategy

Defines how updates occur.

Deployment Strategies

Rolling Update (Default)

- Gradually replaces old Pods with new ones.
- Ensures minimal downtime.

Options:

```
• maxUnavailable
• maxSurge
```

Recreate

- Terminates all old Pods first.
- Then creates new Pods.
- Causes downtime.

Scaling a Deployment

You can scale manually:

```
kubectl scale deployment nginx-deployment --replicas=5
```

Or update YAML replicas field.

Updating a Deployment

Example: Change image version

```
kubectl set image deployment/nginx-deployment nginx=nginx:1.26
```

Kubernetes:

- Creates new ReplicaSet
- Gradually shifts traffic
- Scales down old ReplicaSet

Rollback in Deployment

Check rollout history:

```
kubectl rollout history deployment/nginx-deployment
```

Rollback:

```
kubectl rollout undo deployment/nginx-deployment
```

Deployment keeps revision history for rollback.

Deployment Lifecycle

States include:

- Progressing
- Available
- Failed

Deployment vs ReplicaSet

Feature	**ReplicaSet**	**Deployment**
Maintains replicas	Yes	Yes
Rolling updates	No	Yes
Rollback support	No	Yes
Version history	No	Yes
Recommended for production	No	Yes

Deployment in Real-World Scenario

Imagine you run an e-commerce app:

1. Deployment runs 4 Pods of version 1.0.
2. You release version 1.1.
3. Deployment creates new ReplicaSet.
4. Gradually replaces old Pods.
5. If bug detected → rollback instantly.

No downtime for users.

Internal Flow of Deployment

Important kubectl Commands

Command	**Purpose**
kubectl get deployments	List deployments
kubectl describe deployment <name>	Detailed info
kubectl rollout status deployment/<name>	Check rollout
kubectl scale deployment <name> --replicas=3	Scale
kubectl rollout undo deployment/<name>	Rollback