BATCH: 149

26/10/2023

DT/NT : DT

LESSON : DevOps

SUBJECT: Kubernetes-2

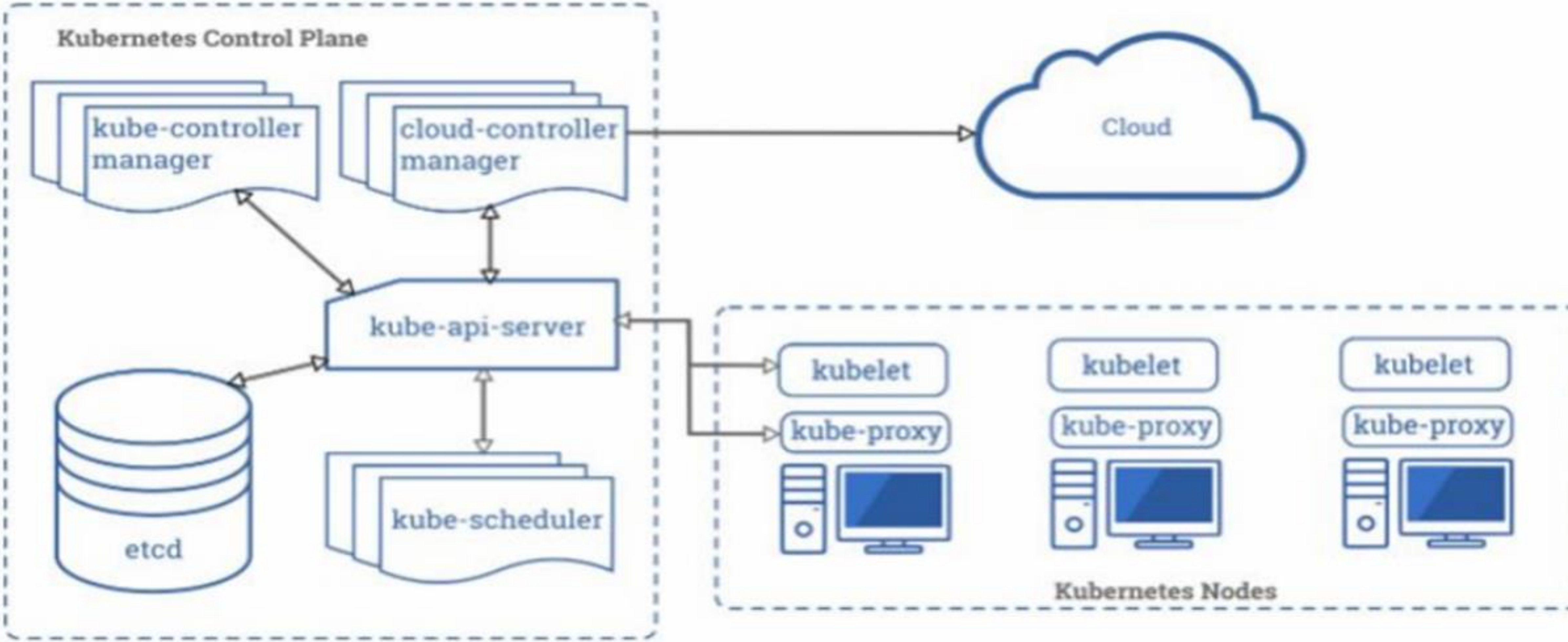
POd

ReplicaSets

Deployment

TECHPRO EDUCATION

Control Plane Components



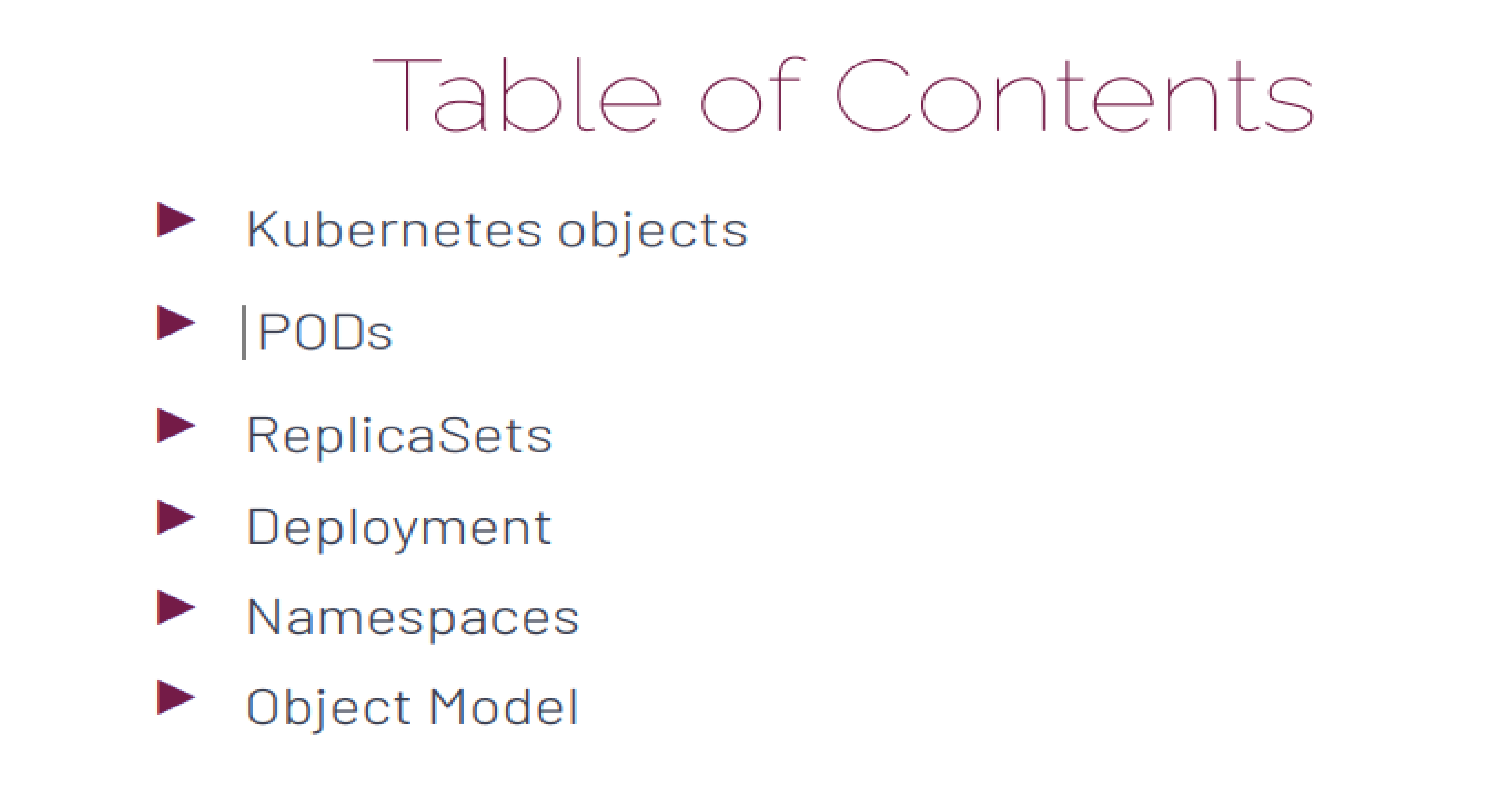
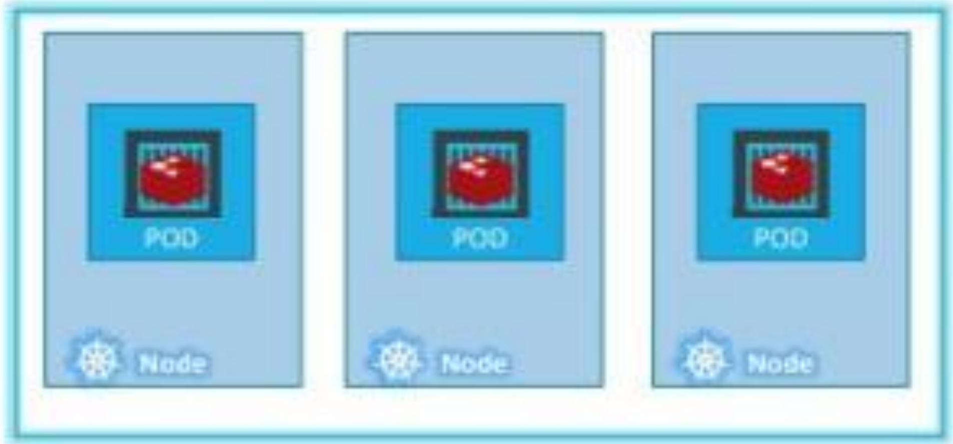


Table of Contents

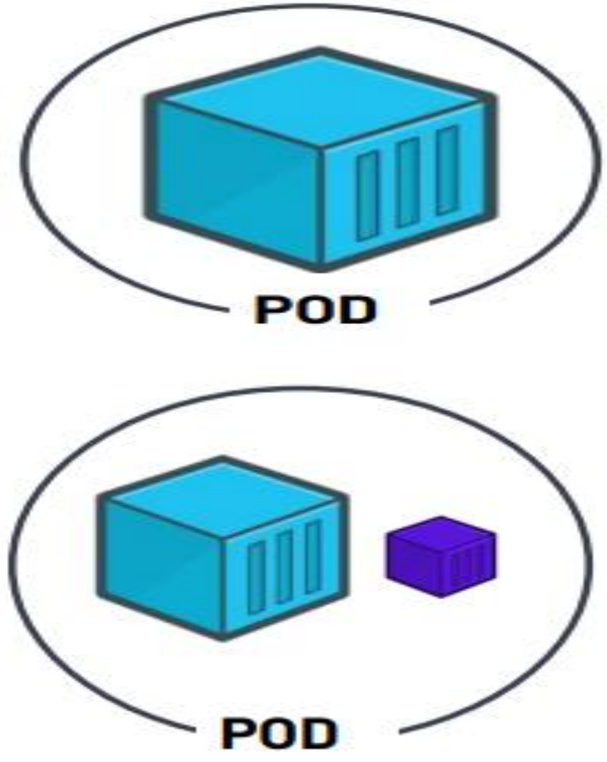
- Kubernetes objects
- PODs
- ReplicaSets
- Deployment
- Namespaces
- Object Model

PODs

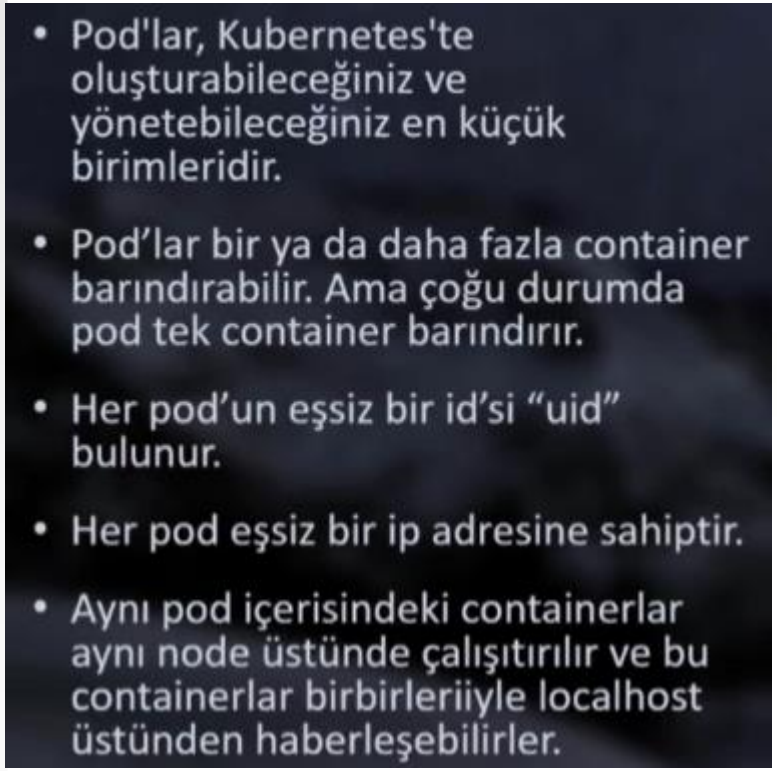


PODs

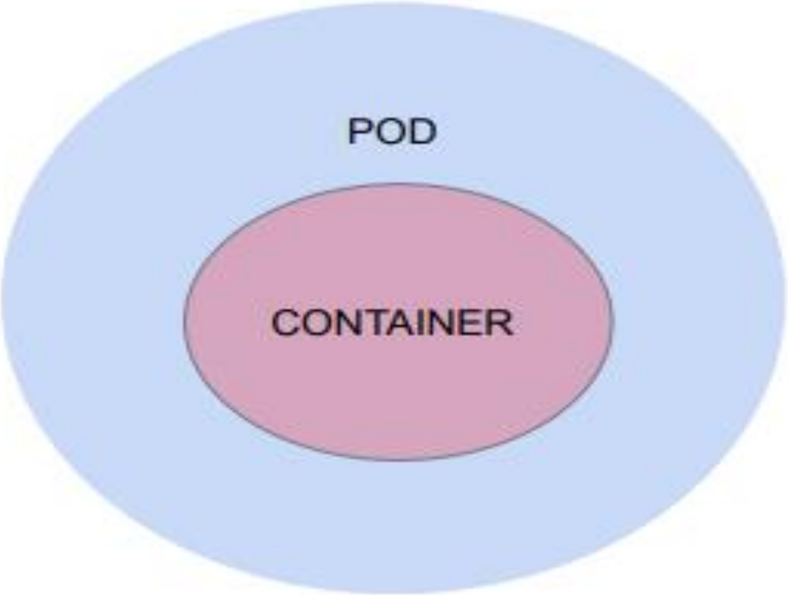
- Kubernetes doesn't deal with containers directly.
- PODs are Kubernetes objects that encapsulate the containers.
- Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.



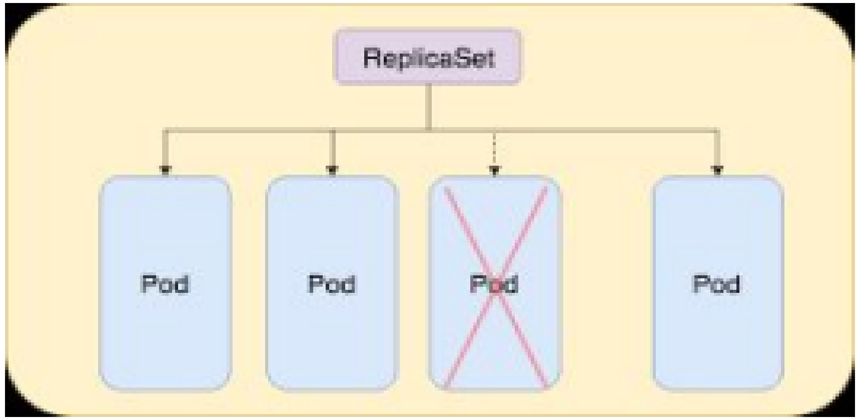
PODs



- Pod'lar, Kubernetes'te oluşturabileceğiniz ve yönetebileceğiniz en küçük birimleridir.
- Pod’lar bir ya da daha fazla container barındırabilir. Ama çoğu durumda pod tek container barındırır.
- Her pod’un eşsiz bir id’si “uid” bulunur.
- Her pod eşsiz bir ip adresine sahiptir.
- Aynı pod içerisindeki containerlar aynı node üstünde çalışıtırılır ve bu containerlar birbirleriiyle localhost üstünden haberleşebilirler.



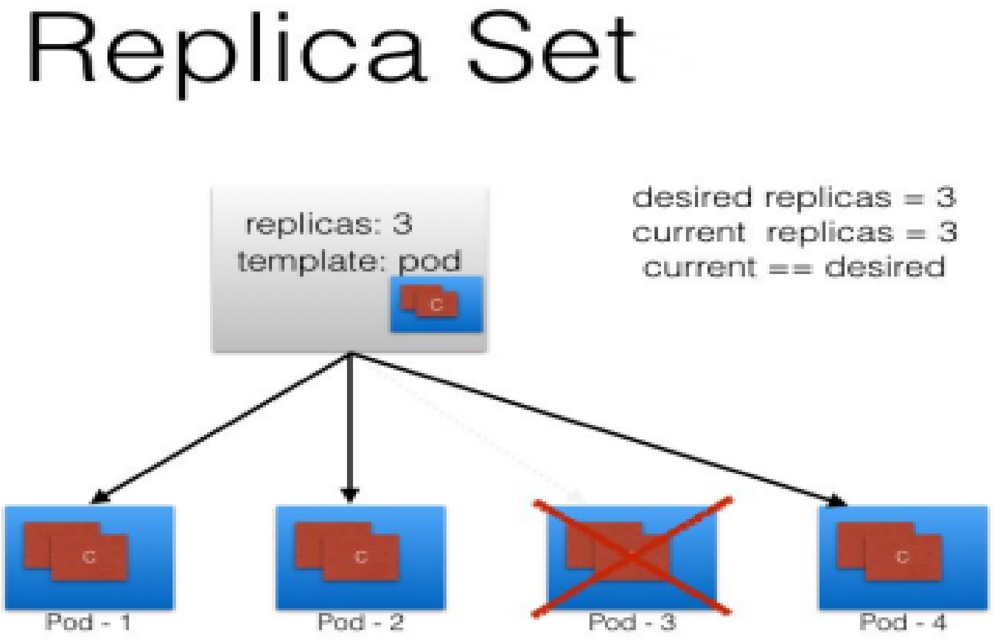
ReplicaSets



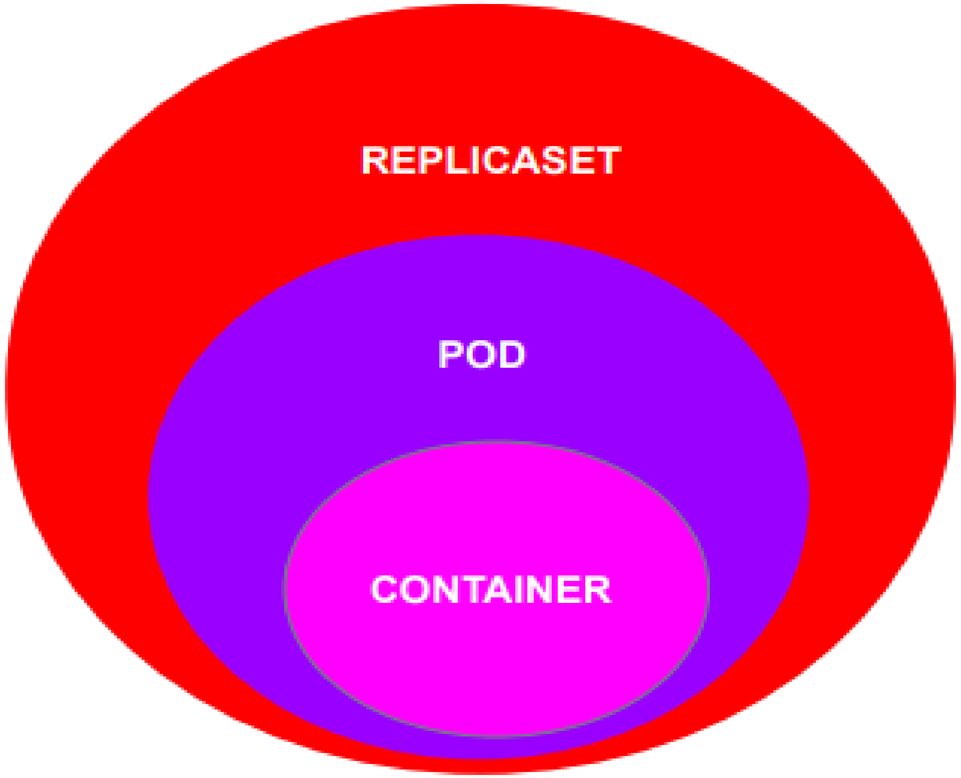
ReplicaSets

- A **ReplicaSet's** purpose is to maintain a stable set of replica Pods running at any given time.

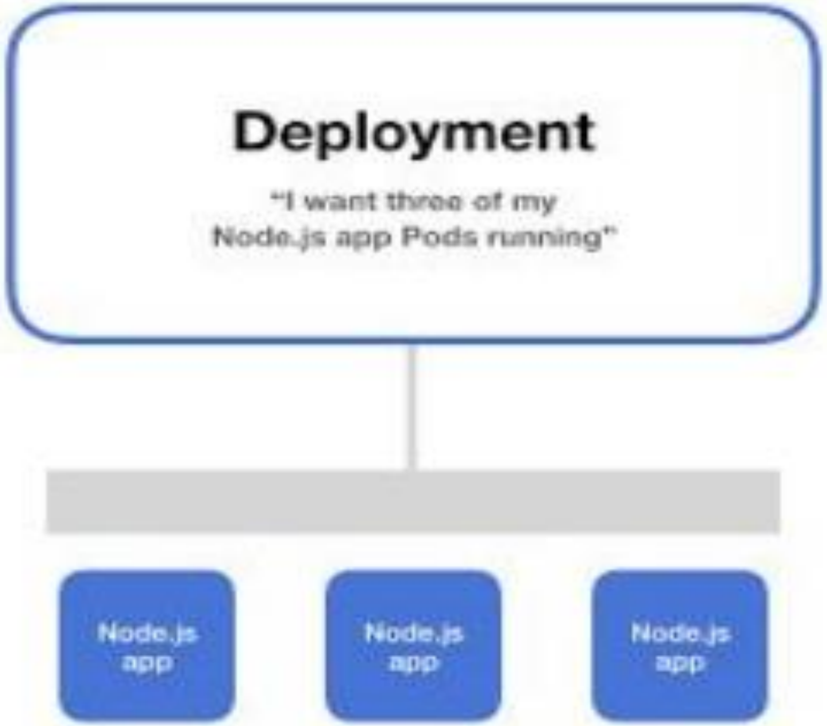
- Even if you have a single POD, the ReplicaSet will bring up a new POD when the existing one fails.



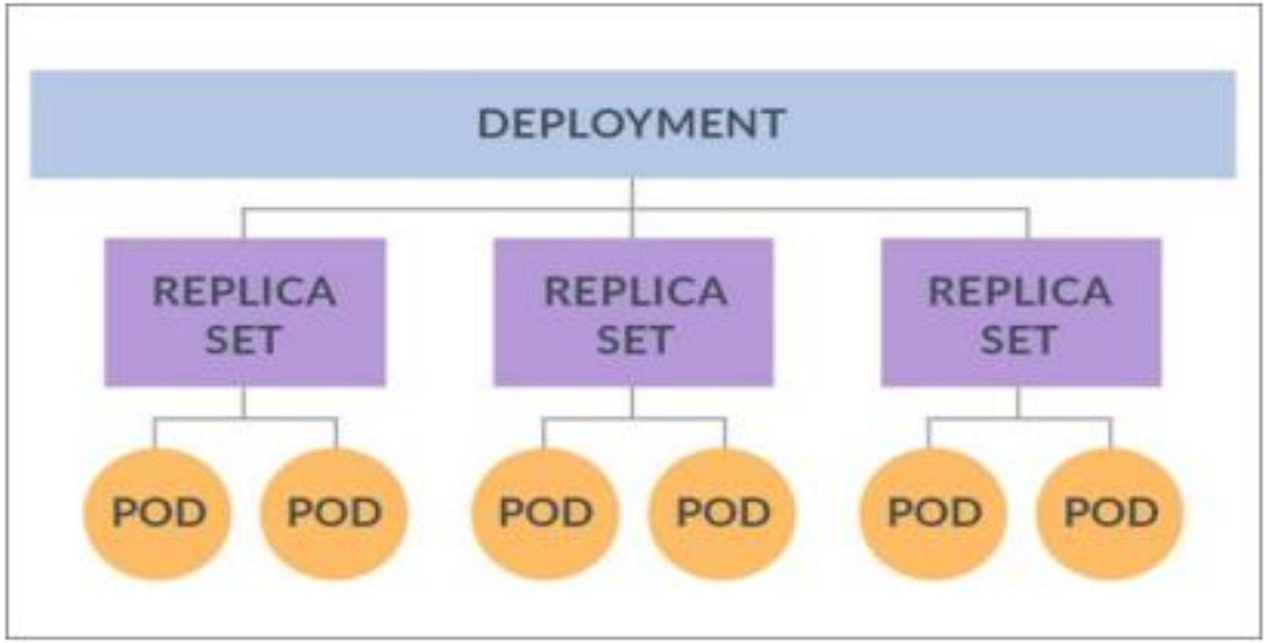
ReplicaSets



Deployment

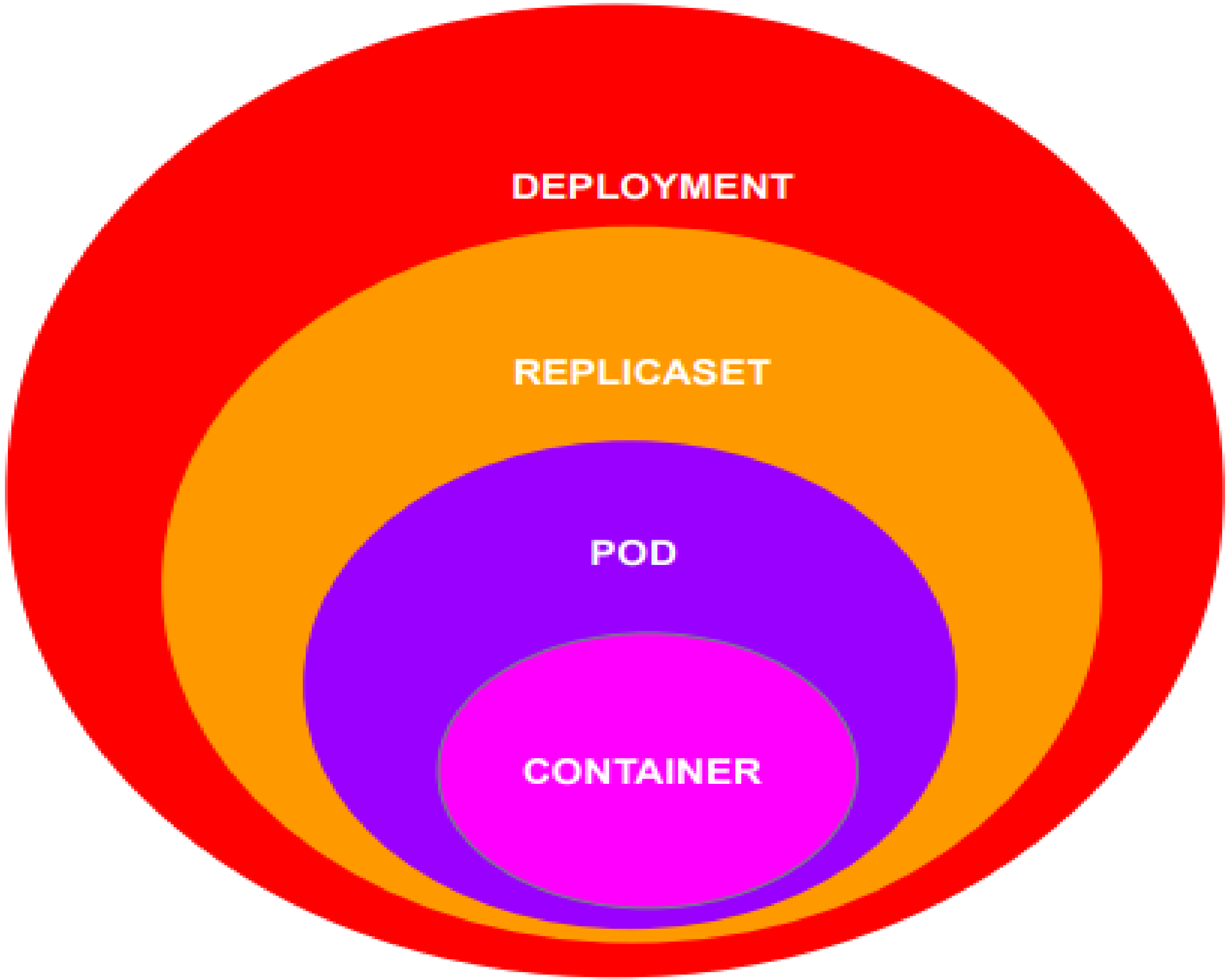

Deployment
"I want three of my
Node.js app Pods running"
Node.js
app
Node.js
app
Node.js
app

Deployment



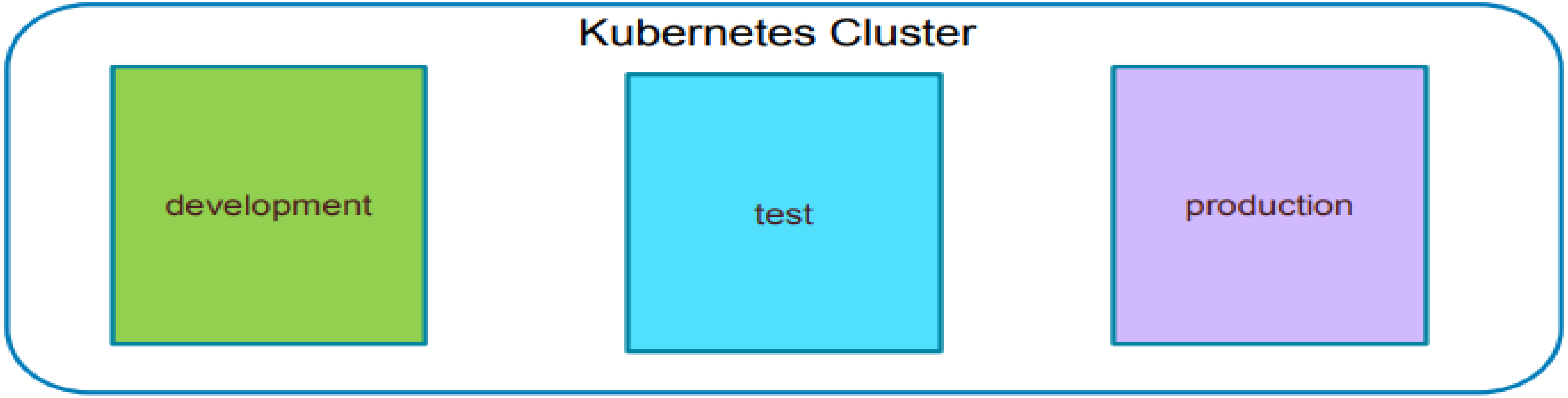
- One step higher in the hierarchy, deployments provides declarative updates for Pods and ReplicaSets.

Deployment



Namespaces

- Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called **namespaces**.
- Namespaces are intended for use in environments with many users spread across multiple teams, or projects.



Kubernetes Cluster

development

test

production

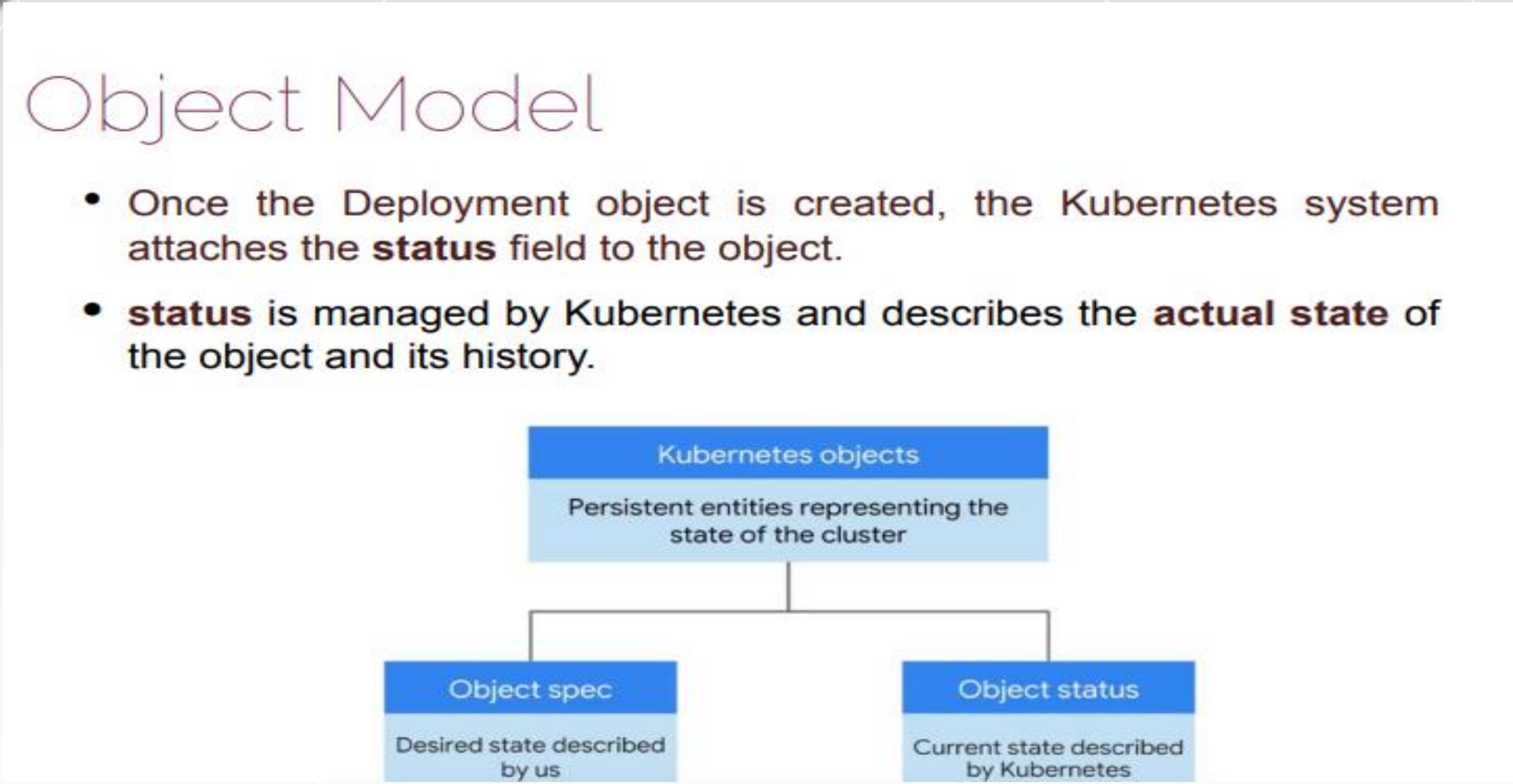
Object Model

Object Model

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

All objects must have **apiVersion, kind, metadata** and **spec** fields.

- **apiVersion:** Which version of the Kubernetes API you're using to create this object
- **kind:** What kind of object you want to create
- **metadata:** Data that helps uniquely identify the object, including a **name** string, **labels**, and optional **namespace**
- **spec:** What state you desire for the object



Object Model

- Once the Deployment object is created, the Kubernetes system attaches the **status** field to the object.
- **status** is managed by Kubernetes and describes the **actual state** of the object and its history.

Kubernetes objects

Persistent entities representing the state of the cluster

Object spec

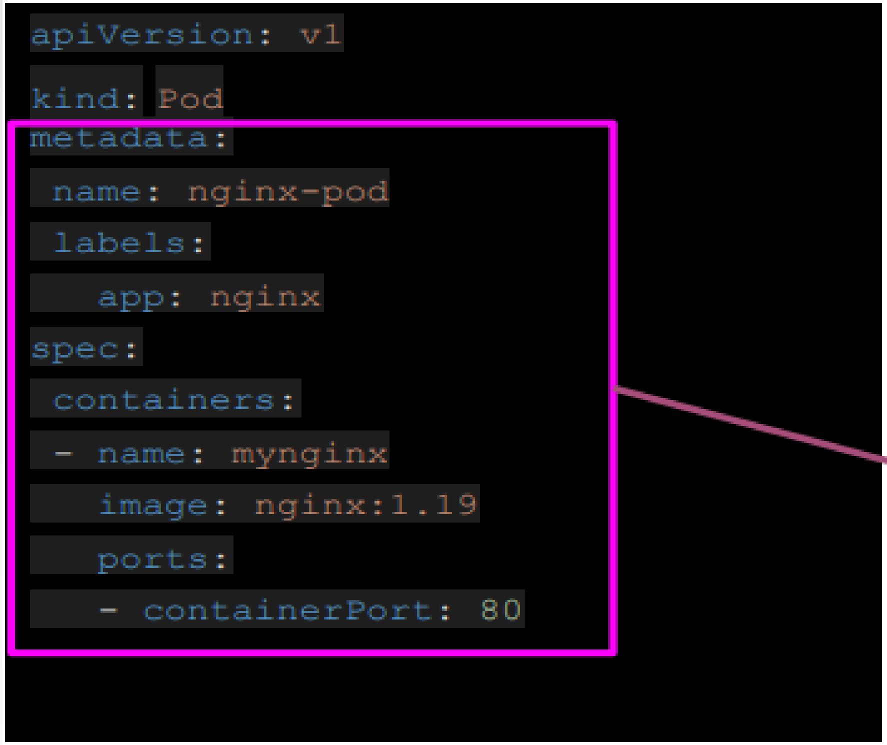
Desired state described by us

Object status

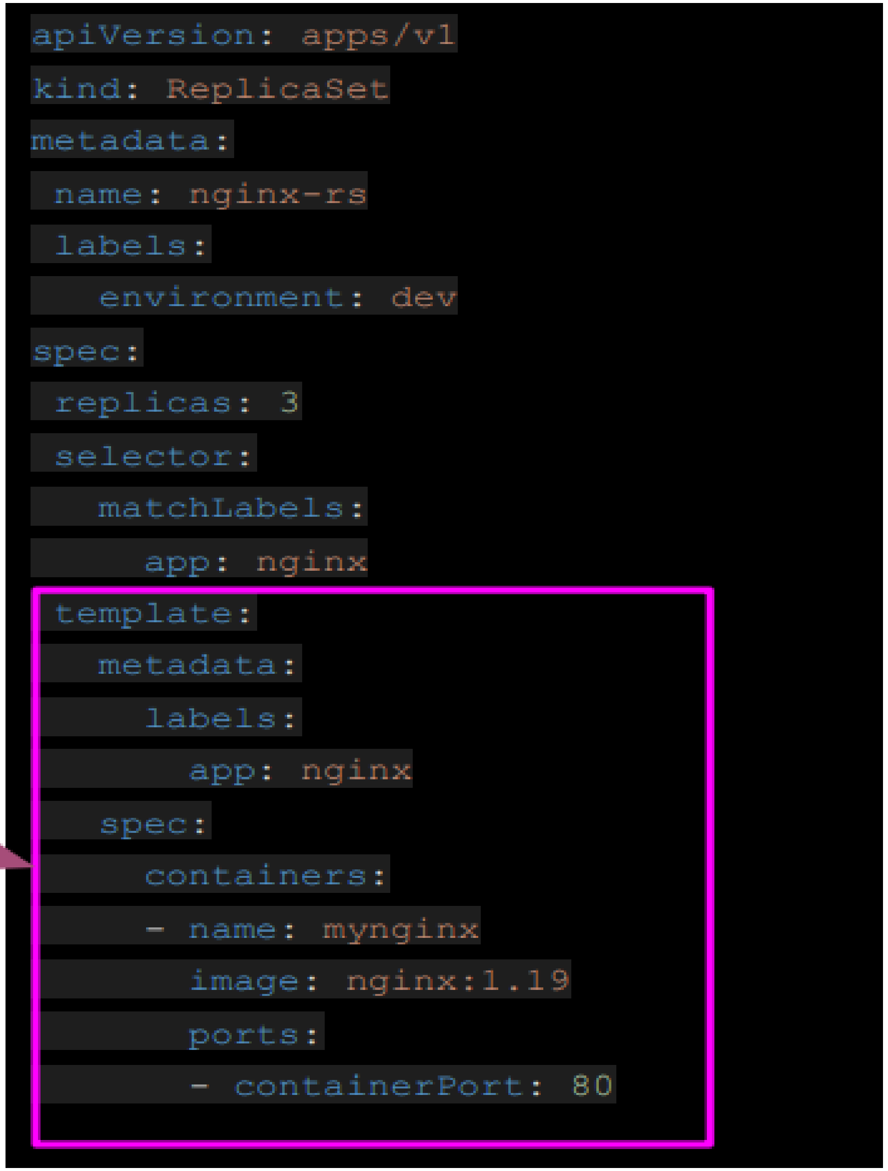
Current state described by Kubernetes

Object Model

Pod to ReplicaSet

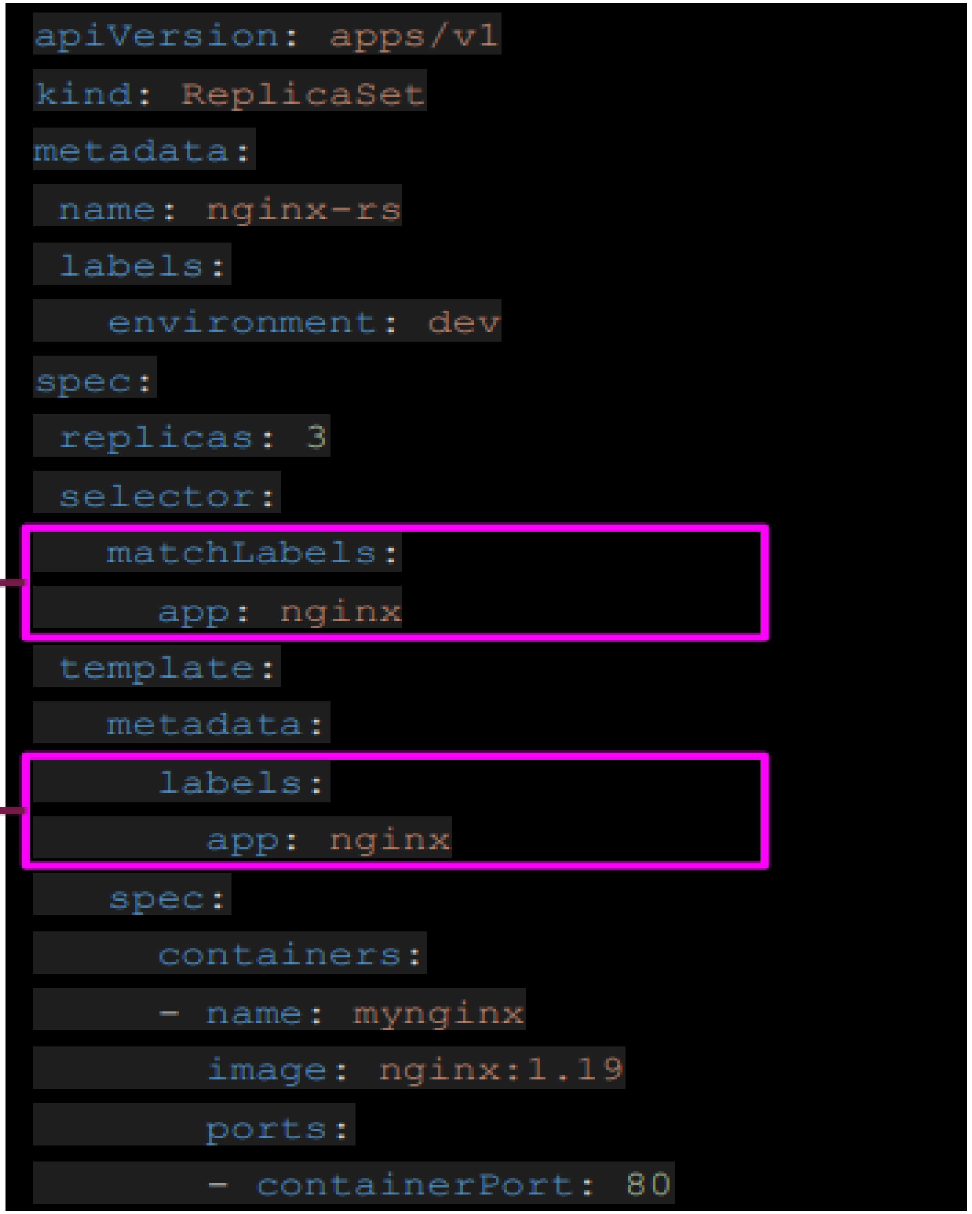


```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx-pod
 labels:
   app: nginx
spec:
 containers:
 - name: mynginx
   image: nginx:1.19
   ports:
   - containerPort: 80
```

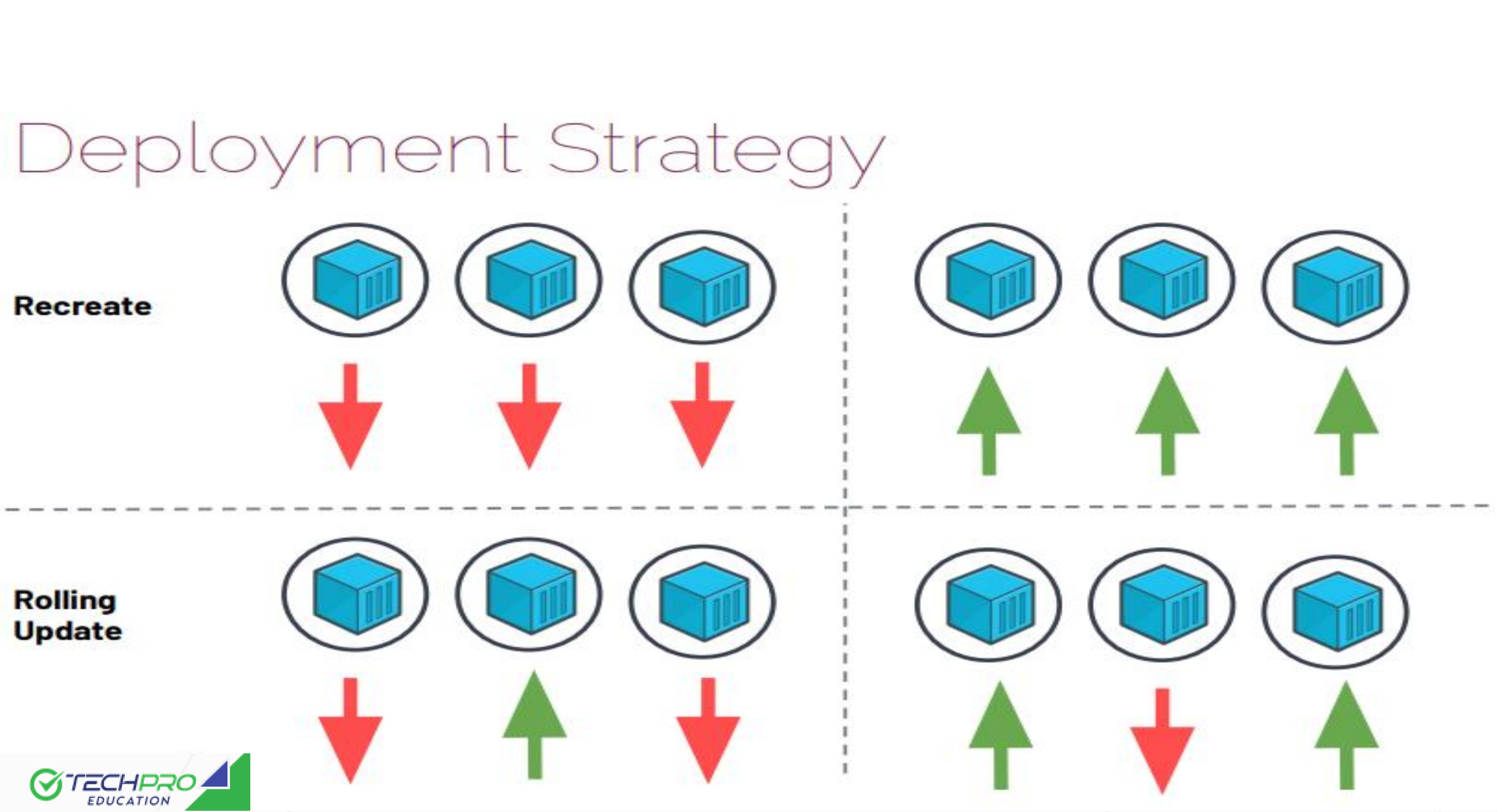


```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: nginx-rs
 labels:
   environment: dev
spec:
 replicas: 3
 selector:
   matchLabels:
     app: nginx
 template:
   metadata:
     labels:
       app: nginx
   spec:
     containers:
     - name: mynginx
       image: nginx:1.19
       ports:
       - containerPort: 80
```

Pod Selector



```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: nginx-rs
 labels:
   environment: dev
spec:
 replicas: 3
 selector:
   matchLabels:
     app: nginx
 template:
   metadata:
     labels:
       app: nginx
   spec:
     containers:
     - name: mynginx
       image: nginx:1.19
       ports:
       - containerPort: 80
```



Deployment Strategy