Kubernetes Objects

Table of Contents

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

1

Kubernetes Objects

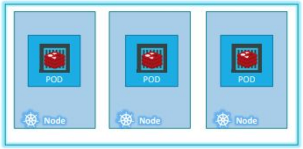
Kubernetes Objects

Kubernetes objects are persistent entities in the Kubernetes system. Kubernetes uses these entities to represent the state of your cluster. Specifically, they can describe:

- What containerized applications are running (and on which nodes)
- The resources available to those applications
- The policies around how those applications behave, such as restart policies, upgrades, and fault-tolerance

2

PODs

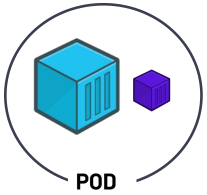


PODs

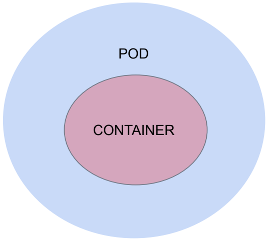
- Kubernetes doesn't deal with containers directly.
- PODs are Kubernetes objects that encapsulate the containers.
- A POD is a single instance of an application.
- Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.

PODs

- A POD can have multiple containers.
- Sometimes you might need a helper container for a primary application, such as logging, monitoring, etc.
- These helper containers should coexist with your application container.
- In that scenario, you CAN put both of these containers part of the same POD, so that when a new application container is created, the helper is created as well, and when the application container dies, the helper dies as well.

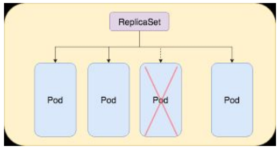


PODs



3

ReplicaSets



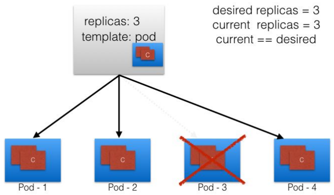
ReplicaSets

- Let's assume that, we have a single POD running our application that serves a set of users. What if for some reason, our application crashes and the POD fails? Our application will no longer be available to users.
- Sometimes we might need a lot of pods running at the same time to prevent users from losing access to our application.
- How can we keep a stable set of Pods running at any given time?

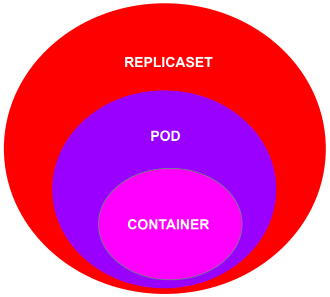
ReplicaSets

- A **ReplicaSet's** purpose is to maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical Pods.
- Even if you have a single POD, the ReplicaSet will bring up a new POD when the existing one fails.

Replica Set

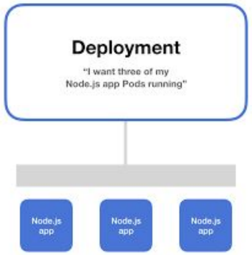


Replication Sets



4

Deployment

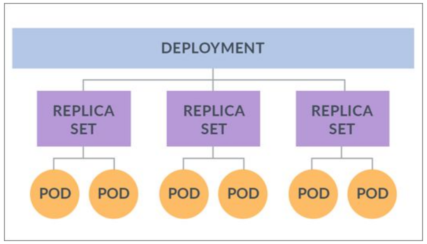


Deployment

- So far, we have a web application that serves a set of users. For this, we create a ReplicaSet and we have three pods inside the ReplicaSet. This time, the newer version of the application is built, and we want to update our application. How can we update our application on Kubernetes?

- Suppose that, we update our application, and there is an unexpected error, and we are asked to undo the recent update. So we need to roll back the changes that were recently implemented. How can we roll back the previous versions of our application?

Deployment



DEPLOYMENT

REPLICA SET

REPLICA SET

REPLICA SET

POD

POD

POD

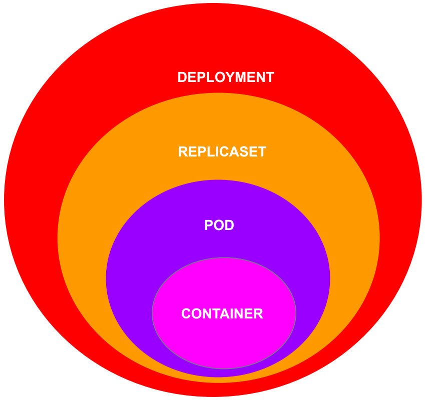
POD

POD

POD

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

Deployment



DEPLOYMENT

REPLICASET

POD

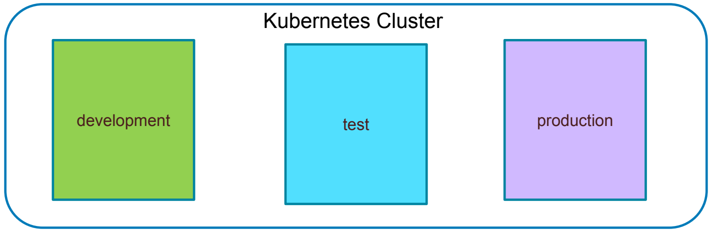
CONTAINER

5

Namespaces

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.

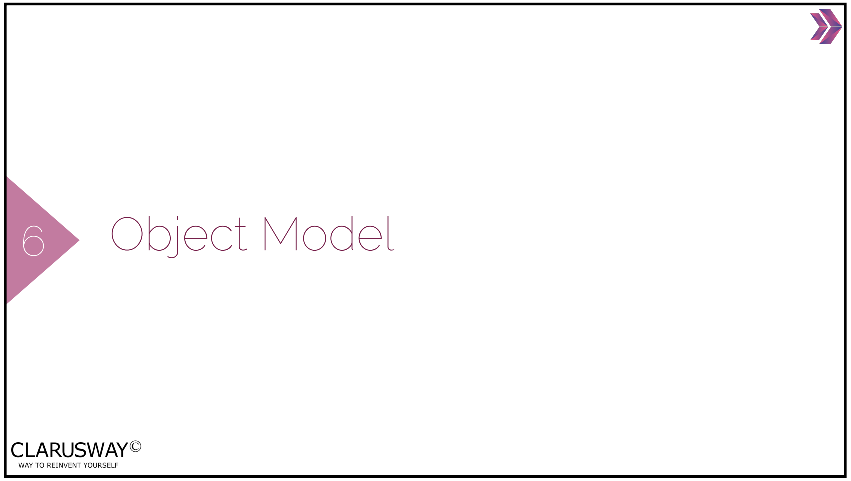


Kubernetes Cluster

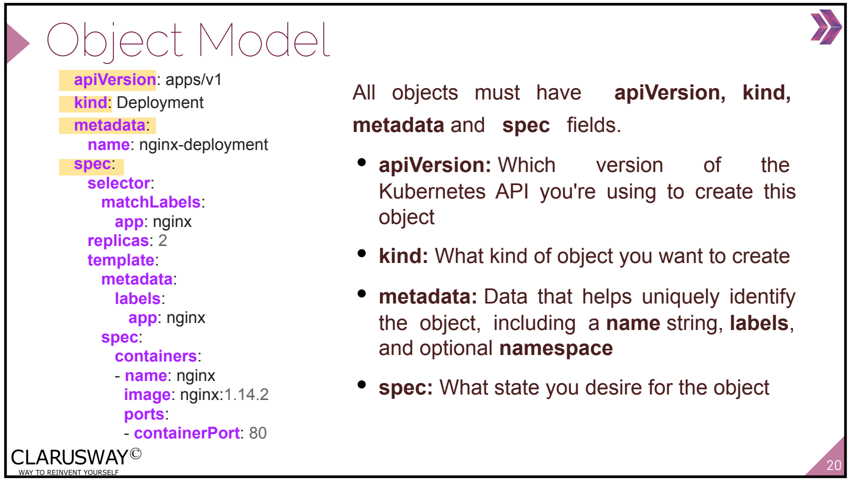
development

test

production



6 Object Model



Object Model

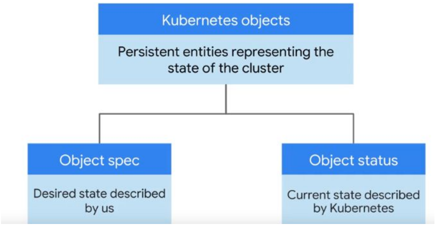
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        -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

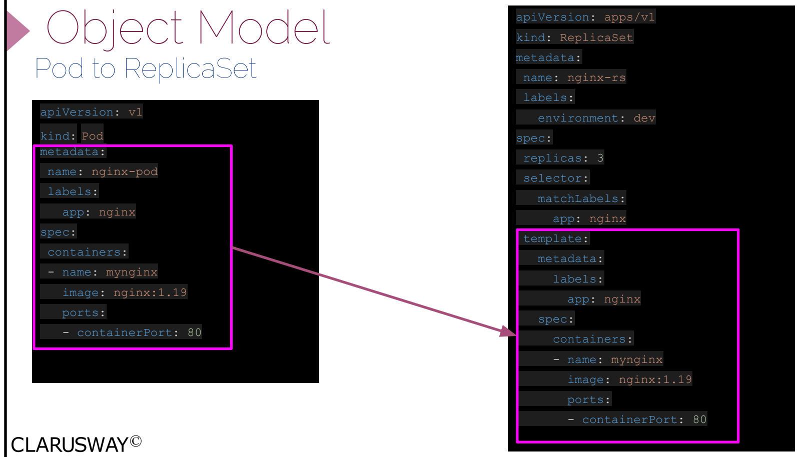
Desired state described by us

Object status

Current state described by Kubernetes

Object Model

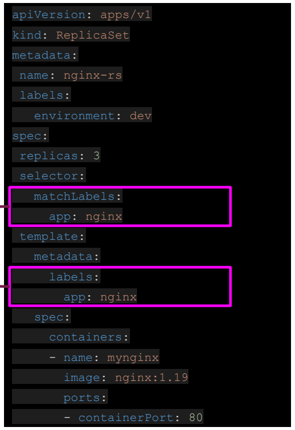
Pod to ReplicaSet



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

```
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
```

THANKS!

Any questions?

You can find me at:

- Joe@clarusway.com