

Kubernetes from Basic to Advanced



kubernetes

Session #03

Deployments



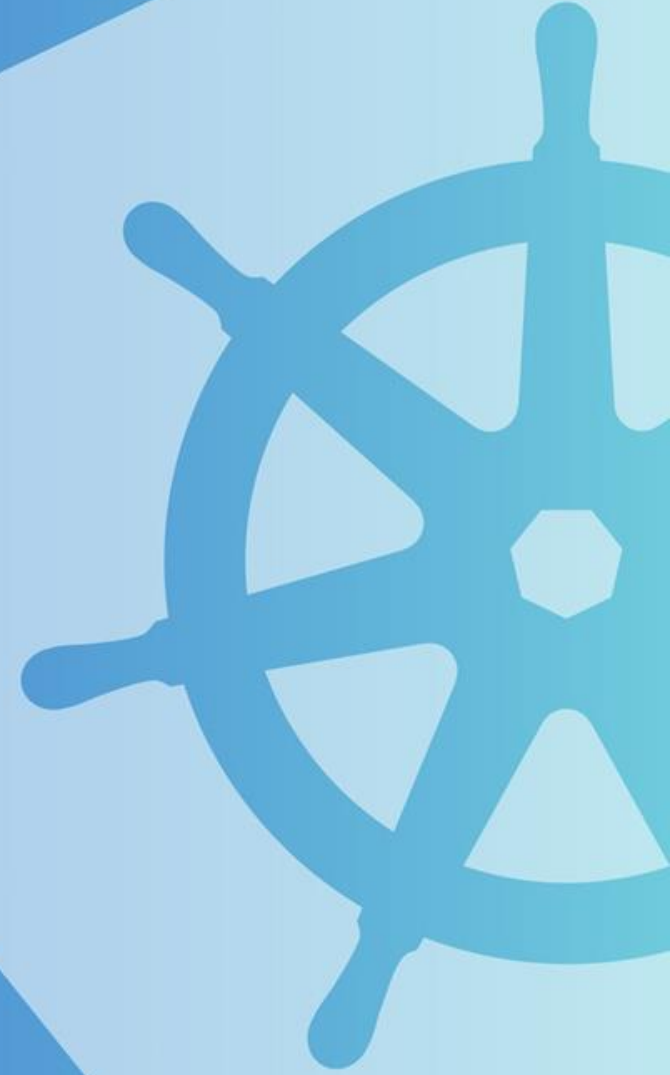
kubernetes

Session Contents



- Labels & Selectors
- ReplicaSets
- Deployments

Labels & Selectors



Motivation



- Number of pods start to grow inside clusters
- Need to make some grouping to better handle similar pods
- At same time, this grouping need to agility and openness
- The answer is using labels like in many other systems and solutions

Labels



- Declarative way to identify (categorize, group) Kubernetes objects
- Enable mapping organizational structures onto system objects in a loosely coupled fashion, without requiring clients to store these mappings
- Can be set on any Kubernetes object (Nodes, pods, ...)
- A Kubernetes object can have zero to many labels

Labels: Properties



- Simple key-value pair
- Cannot be a list
- Needs to be unique per object
- Labels are case sensitive

Labels: Name format



prefix/name

- **prefix**
 - Is optional
 - Must follow a DNS subdomain format
 - No longer than 63 chars
 - Example: k8s.io, contoso.com, acme.net
- **name**
 - Is mandatory
 - Must start alphanumeric character
 - Can include dash (-), underscore (_) or dot (.)
 - No longer than 253 characters

Labels: Selectors



- Label selectors allow to identify a group of objects
- A selector uses labels to make a match and create the group
- This matching is dynamic meaning if new objects are created, they are included on the group
- Two options: equality-based and set-based
- Equality: `=`, `==`, `!=`
- Set: `in`, `notin`



Demo | Labels & Selectors



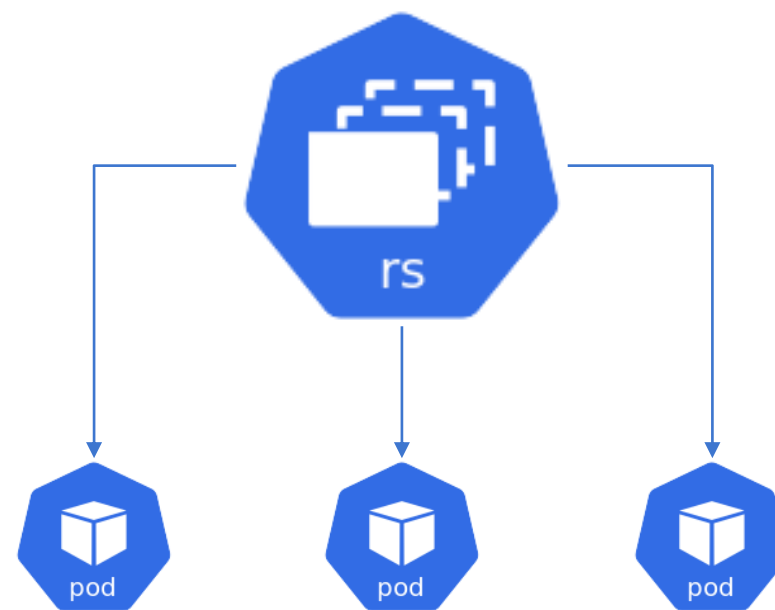
ReplicaSet



ReplicaSets



- ReplicaSet purpose is to maintain a stable set of replica Pods running at any given time
- Used to guarantee the availability of a specified number of identical Pods



How it works?



- A ReplicaSet includes a selector to identify Pods to be managed by
- Defines a number of replicas indicating how many Pods it should be maintaining
- A pod template specifying the data of new Pods it should create to meet the number of replicas criteria
- A ReplicaSet then fulfills its purpose by creating and deleting Pods as needed to reach the desired number
- When a ReplicaSet needs to create new Pods, it uses its Pod template

ReplicaSet Manifest

ReplicaSet properties



replica set the number of pods



selector finds the template



template defines PodSpec



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

Demo | ReplicaSets

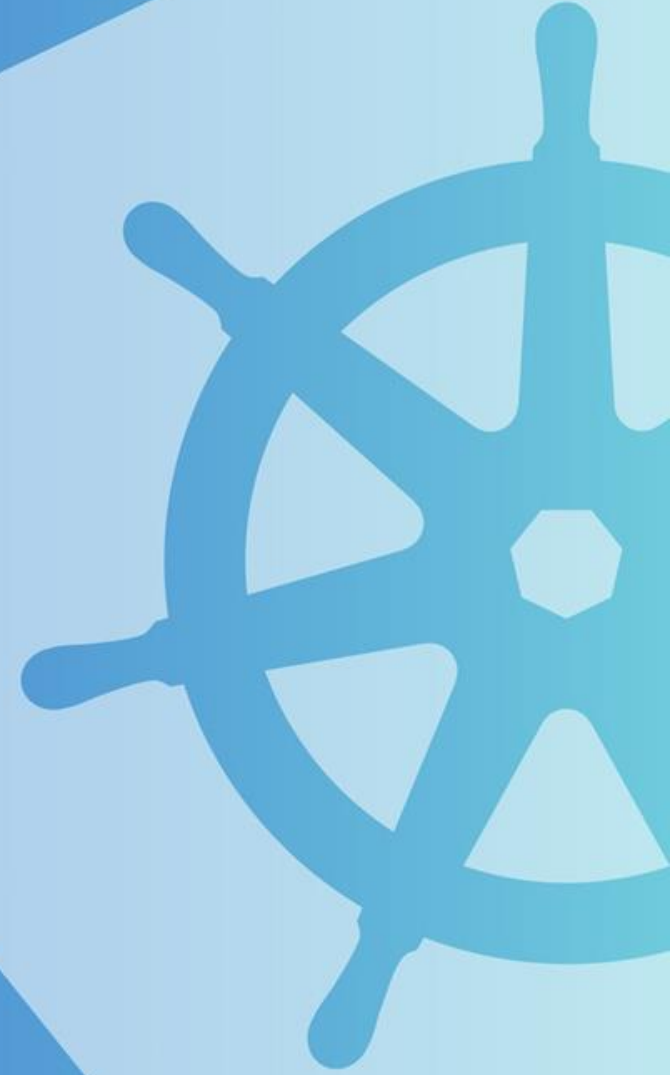


Some issues



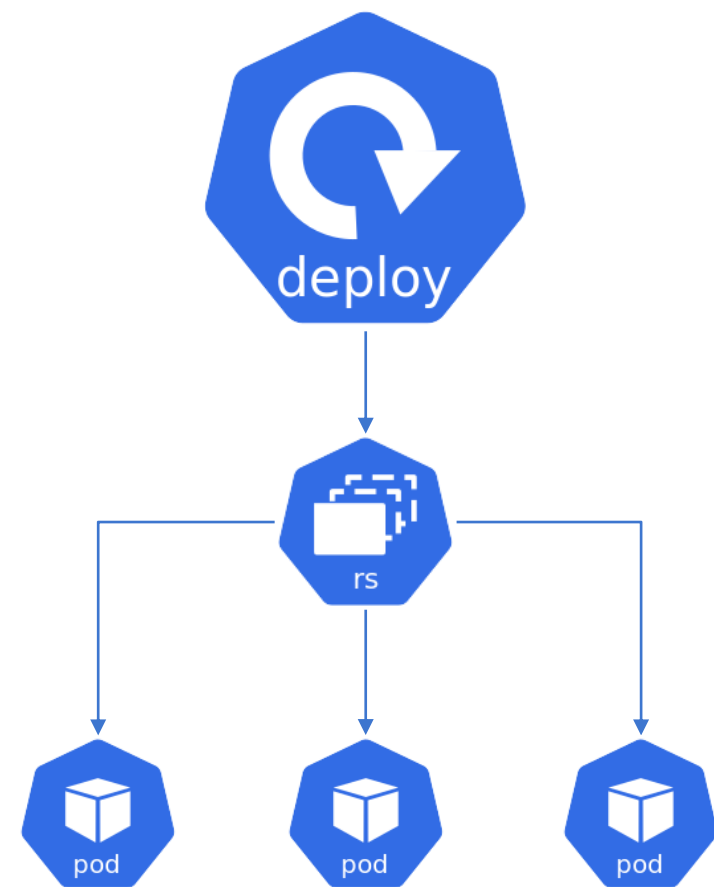
- ReplicaSet identifies new Pods to acquire by using its selector
- If there is a Pod that matches a ReplicaSet's selector, it will be immediately acquired
- This means that you may never need to manipulate ReplicaSet objects directly

Deployments



Deployments

- A Deployment is a higher-level controller that manages ReplicaSets
- Provides declarative updates to Pods along with a lot of other useful features.



Deployment Manifest

Deployment properties →

replica set the number of pods →

selector finds the template →

template defines PodSpec → {

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: workload-1-dep
spec:
  replicas: 6
  selector:
    matchLabels:
      app: workload-1
  template:
    metadata:
      labels:
        app: workload-1
        color: lime
    spec:
      containers:
        - name: workload
          image: nginx:1.18
          ports:
            - containerPort: 80
```

Deployment



- Describe a [desired state](#) in a Deployment
- Deployment Controller changes the actual state to the desired state at a controlled rate
- Deployments provide fine-grained control over how and when a new pod version is rolled out as well as rolled back to a previous state

How it works?



1. We are running a deployment with 2 replicas of a pod with an application with yellow background
2. Developer makes the change to the deployment image tag and to sets the background to green
3. The Deployment creates a ReplicaSet v2, which will create 2 new Pods with the updated image.
4. Once the new Pods are up and running, the Pods attached to ReplicaSet v1 are deleted.
5. ReplicaSet v1 is not deleted



Deployment Strategy



- A deployment strategy is a way to change or upgrade an application
- **Rolling Update** - Consists of slowly rolling out a new version of an application by replacing instances one after the old version until all the instances are rolled out. This is the default strategy for Kubernetes Deployments when none is specified.
- **Recreate** - Consists of shutting down all instances of the current version, then deploying the new version. This technique implies downtime of the service that depends on both shutdown and boot duration of the application.

Demo | Deployments



Questions?



kubernetes

Lab #02: Deployment lifecycle



kubernetes