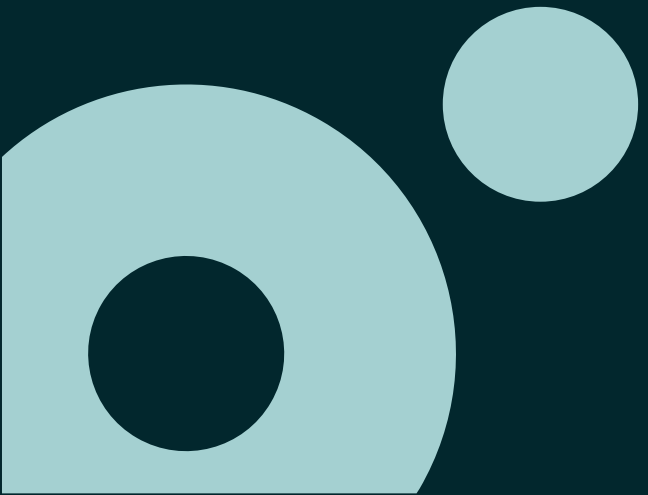
Containers & Kubernetes

Session #06

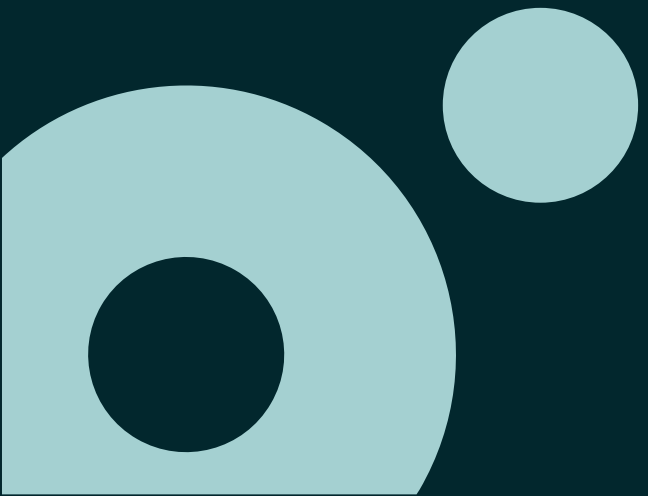


Labels & Selectors

Deployments & ReplicaSets

Lab

Labels & Selectors



Motivation

Labels & Selectors

- 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 the usage of the concept of labels as is used in many other systems and solutions

Labels

Labels & Selectors

- Declarative way to identify (categorize, group) Kubernetes objects
- Enable users to map their own organizational structures onto system objects in a loosely coupled fashion, without requiring clients to store these mappings
- Simple key-value pair
  - Cannot be a list
  - Needs to be unique per object
- Can be set on any Kubernetes object (Nodes, pods, …)
- A Kubernetes object can have zero to many labels
- Labels are case sensitive

Labels: Syntax

Labels & Selectors

- Labels 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

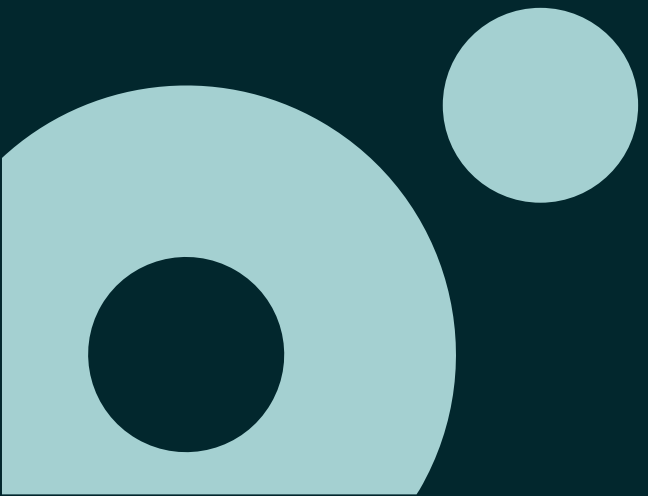
Selectors

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

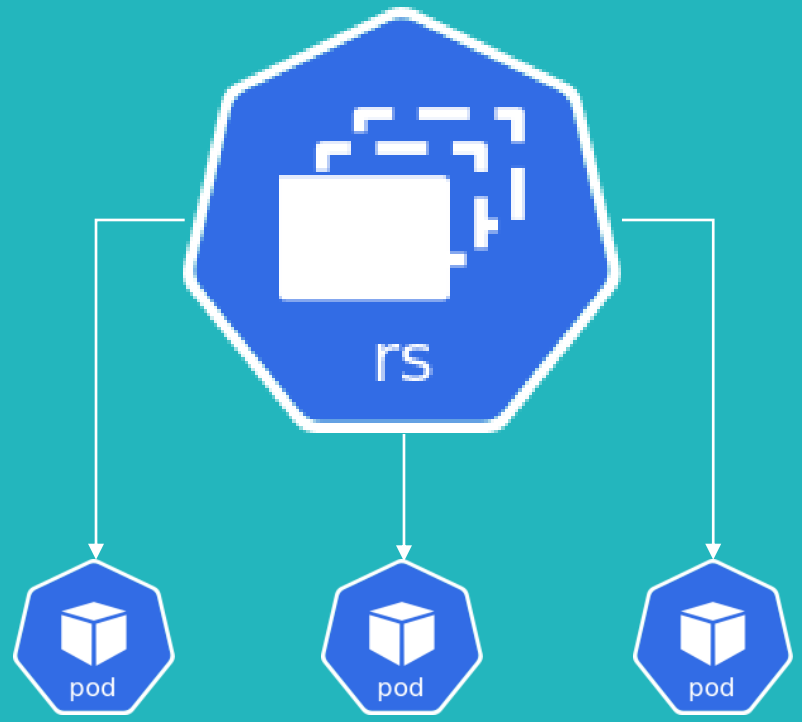
Deployments & ReplicaSets



ReplicaSets

Deployments & 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
- Uses selectors to create pods groups



ReplicaSets

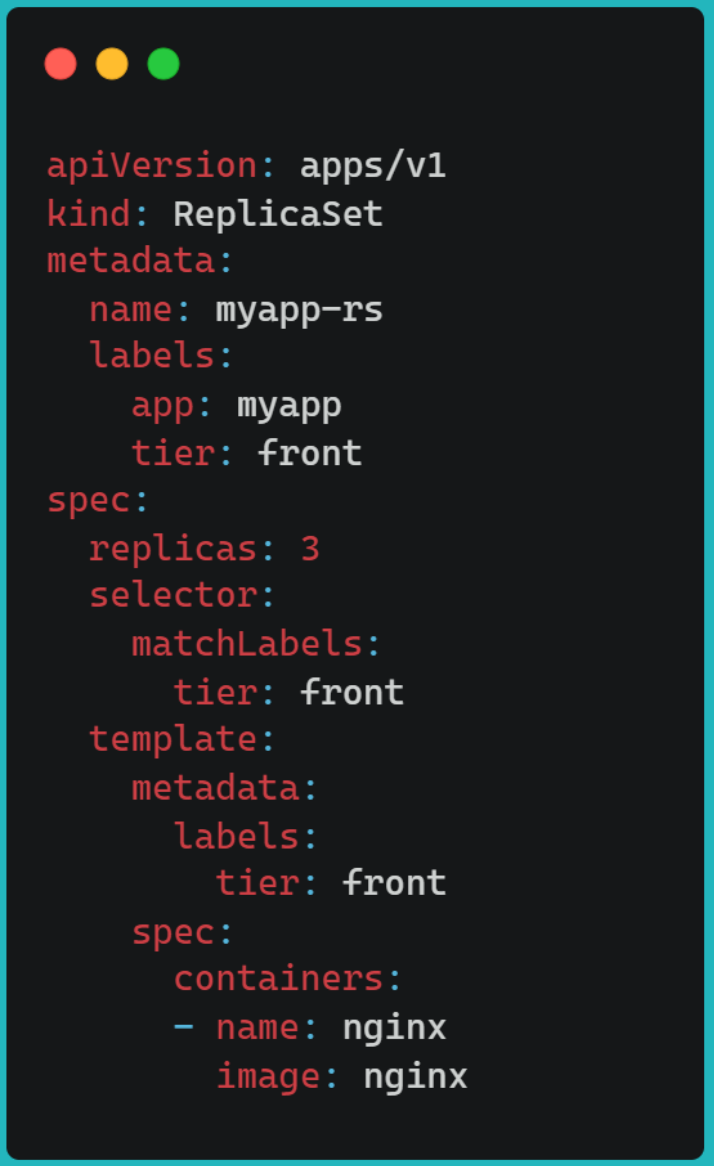
Deployments & ReplicaSets

ReplicaSet properties

replica set the number of pods

selector finds the template

template defines PodSpec



Demo: ReplicaSets

ReplicaSets

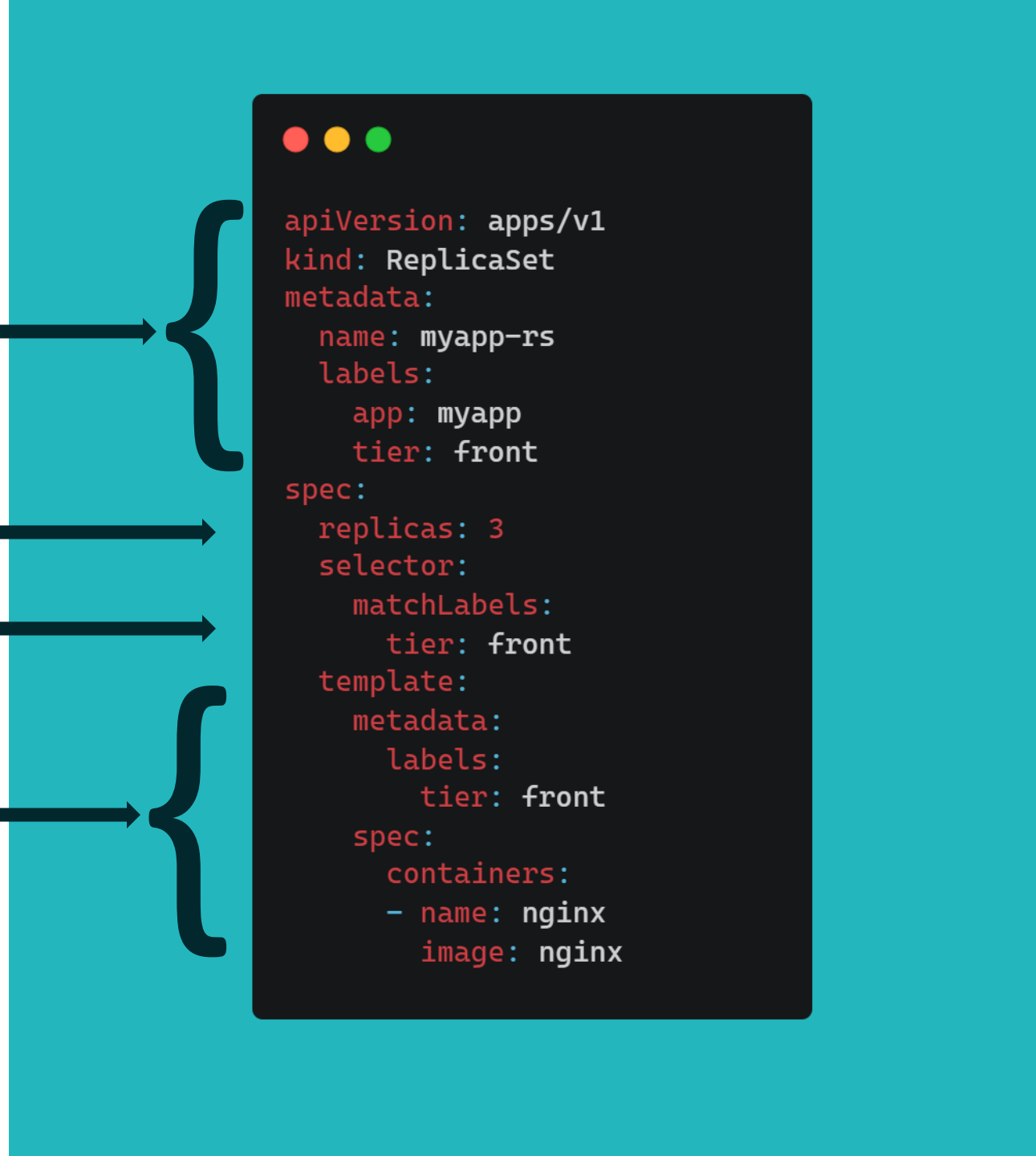
Deployments & ReplicaSets

ReplicaSet properties

replica set the number of pods

selector finds the template

template defines PodSpec

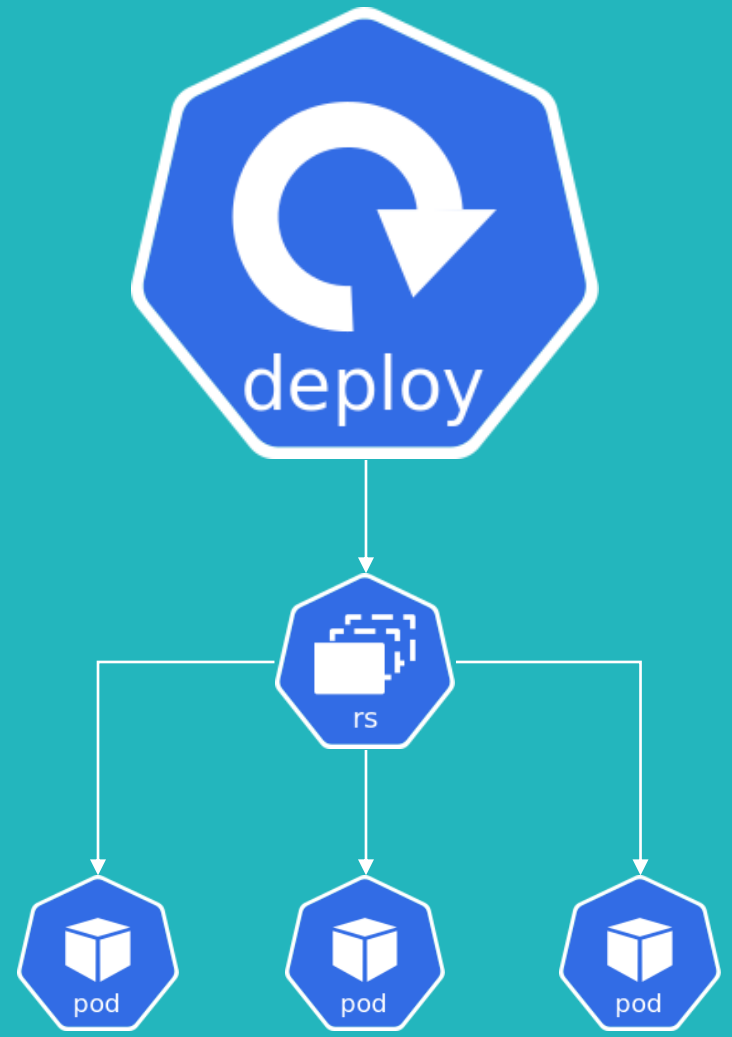


However, ReplicaSet is too low-level to handle lifecycle

Deployments

Deployments & ReplicaSets

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



Deployment

Deployments & ReplicaSets

replica set the number of pods

selector finds the template

template defines PodSpec

Deployment

Deployments & ReplicaSets

- 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

Deployments & ReplicaSets

1. We are running a deployment with 2 replicas of a pod 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 Strategies

Deployments & ReplicaSets

- 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.

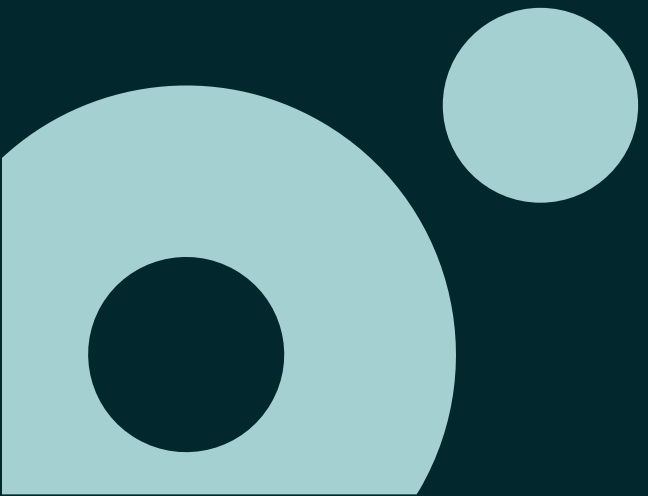
```
spec:
  replicas: 6
  strategy:
    type: RollingUpdate
```

- **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.

```
spec:
  replicas: 6
  strategy:
    type: Recreate
```

Demo: Deployments Lifecycle

DaemonSets



What is a DaemonSet

DaemonSets

- DaemonSets ensure all (or some) Nodes run an instance of a Pod.
- As nodes are added to the cluster, specified Pods are added to them.
- As nodes are removed from the cluster, those Pods are terminated.
- Deleting a DaemonSet will clean up the Pods it created.
- Some typical uses of a DaemonSet are:
  - Running a cluster storage daemon on every node
  - Running a logs collection daemon on every node
  - Running a node monitoring daemon on every node

Kube-proxy

DaemonSets

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  labels:
    k8s-app: kube-proxy
  name: kube-proxy
  namespace: kube-system
spec:
  selector:
    matchLabels:
      k8s-app: kube-proxy
  template:
    metadata:
      labels:
        k8s-app: kube-proxy
    spec:
      containers:
      - mage: k8s.gcr.io/kube-proxy:v1.22.5
        name: kube-proxy
```