Kubernetes for Developers: Deploying Your Code

KUBERNETES DEPLOYMENTS OVERVIEW

Dan Wahlin
WAHLIN CONSULTING

@danwahlin   www.codewithdan.com

Course Overview

Kubernetes Deployments Overview

Performing Rolling Update Deployments

Performing Canary Deployments

Performing Blue-Green Deployments

Running Jobs and CronJobs

Performing Monitoring and Troubleshooting Tasks

Putting It All Together

Target Audience

Developers looking to learn different techniques for deploying code to Kubernetes

Course Pre-Reqs

Comfortable using command-line tools and virtual machines

General understanding of Docker containers and how they work

Understand Kubernetes core concepts

It's recommended that you watch the **Kubernetes for Developers: Core Concepts** course first

Required Software

Docker Desktop

https://www.docker.com/products/docker-desktop

Minikube

https://github.com/kubernetes/minikube

kind

https://kind.sigs.k8s.io

kubeadm

https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm

Code Samples

https://github.com/DanWahlin/DockerAndKubernetesCourseCode

Look in the "samples" folder

Introduction

Module Overview

Kubernetes Deployments Overview

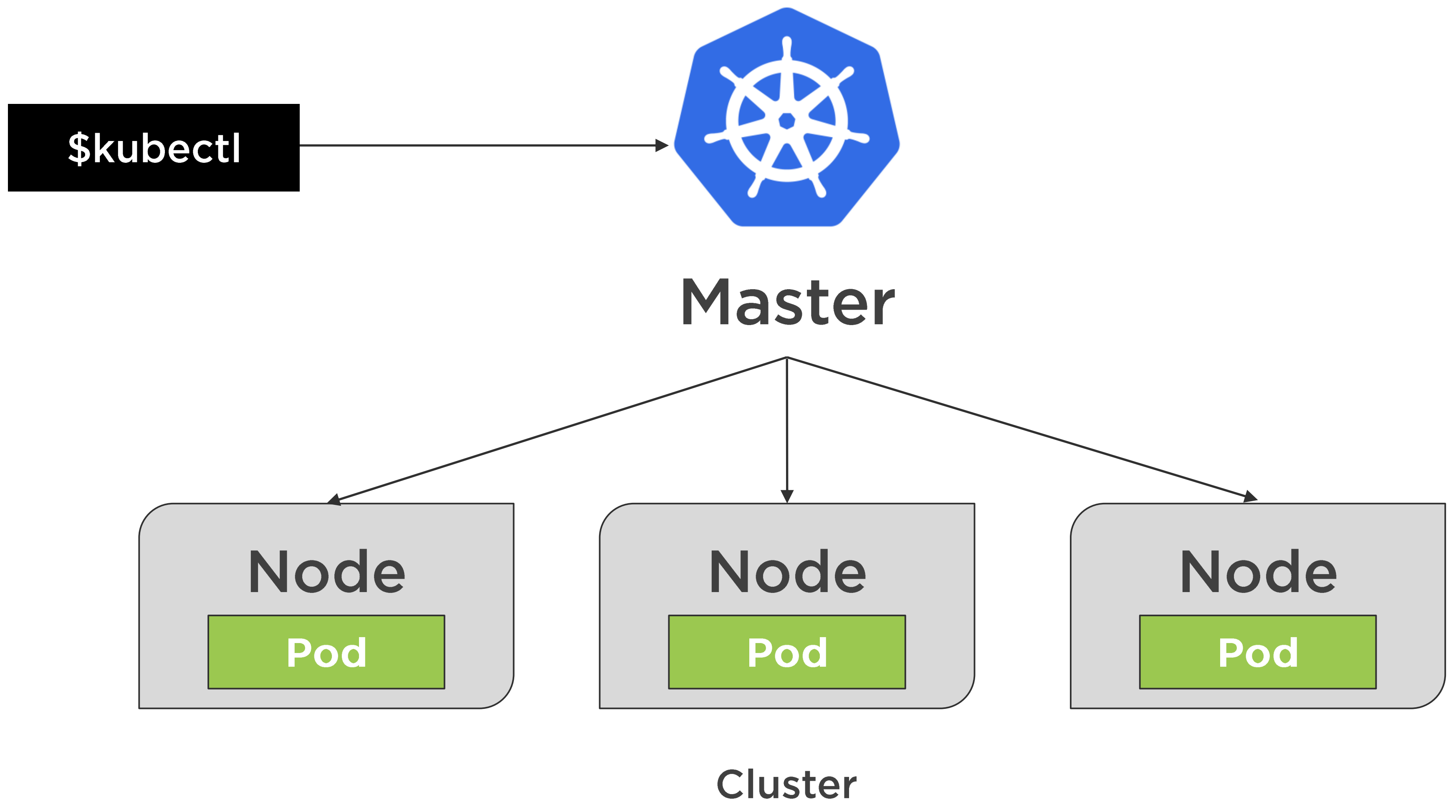
Creating an Initial Deployment

Kubernetes Deployments in Action

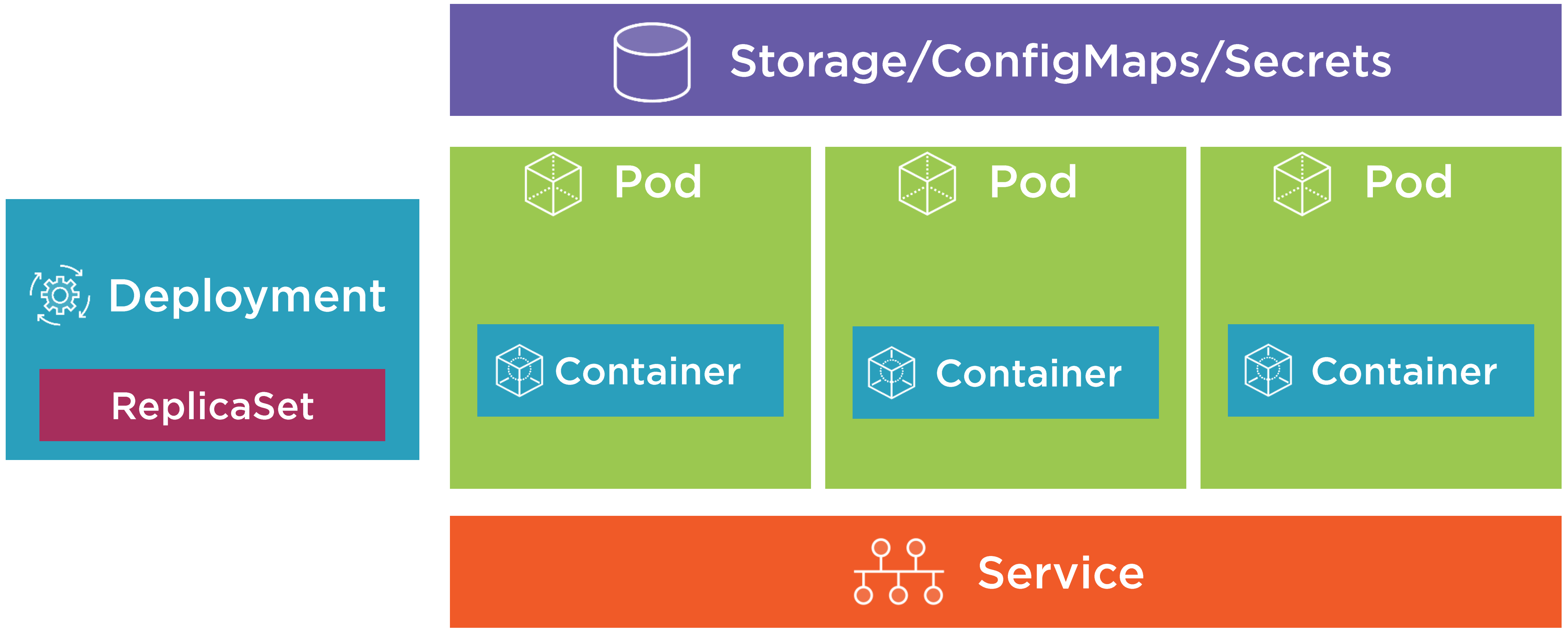
Kubernetes Deployment Options

Kubernetes Deployments Overview

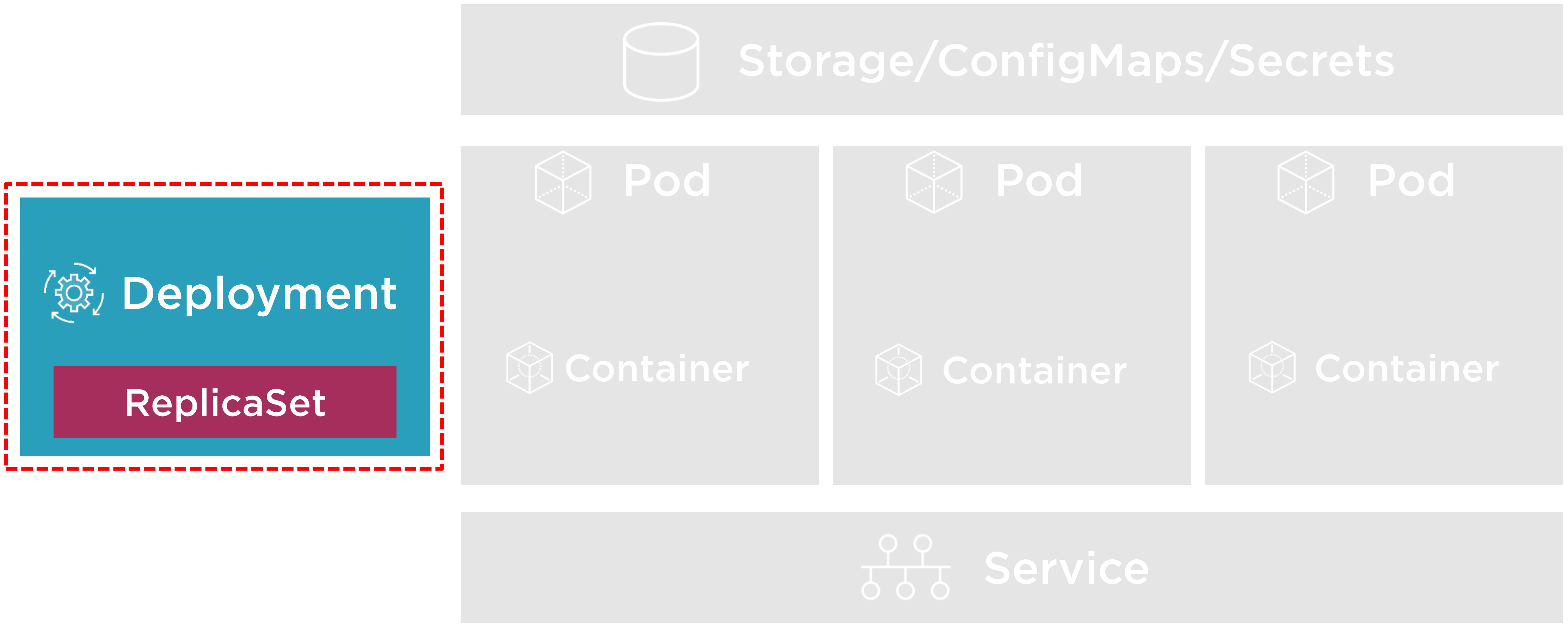
The Big Picture



Kubernetes Resources



Kubernetes Resources

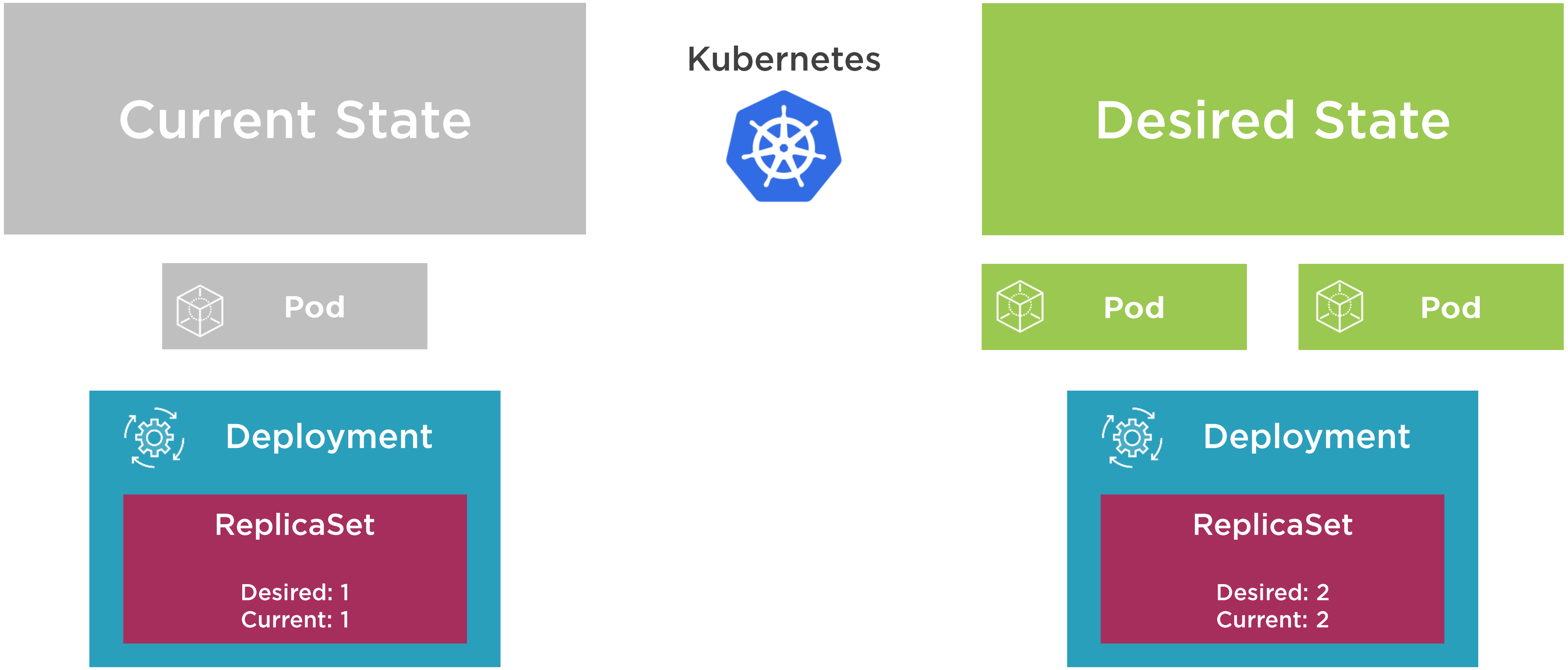


A ReplicaSet is a declarative way to manage Pods.

A Deployment is a declarative way to manage Pods using a ReplicaSet.

Pod
Deployment/ReplicaSet
Pod
Pod

Moving to a Desired State

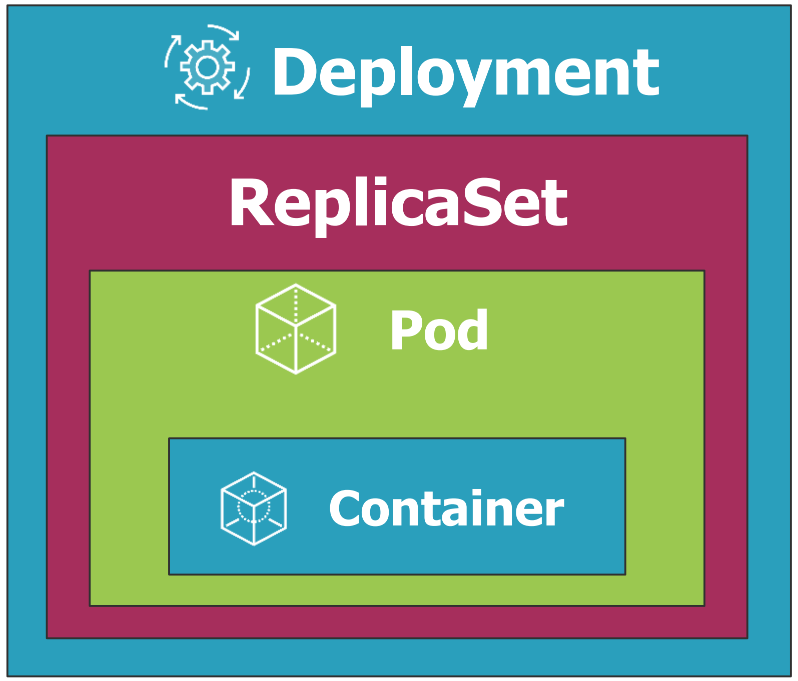


Creating an Initial Deployment

Creating a Deployment

Deployment

+ kubectl =



Defining a Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: my-nginx
    tier: frontend
spec:
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: my-nginx
        image: nginx:alpine
```

◀ Kubernetes API version and resource type (Deployment)

◀ Metadata about the Deployment

◀ The selector is used to "select" the template to use (based on labels)

◀ Template to use to create the Pod/Containers (note that the selector matches the label)

Store current properties in resource's annotations

```
# Create a Deployment
kubectl create -f file.deployment.yml --save-config
```

Creating a Deployment

Use the kubectl create command along with the --filename or -f switch

Creating or Applying Changes

Use the kubectl apply command along with the --filename or -f switch

```
# Alternate way to create or apply changes to a
# Deployment from YAML
kubectl apply -f file.deployment.yml
```

```
# Scale the Deployment Pods to 5 (imperative)
kubectl scale deployment [deployment-name] --replicas=5

# Scale by refencing the YAML file (imperative)
kubectl scale -f file.deployment.yml --replicas=5
```

Scaling Pods Horizontally

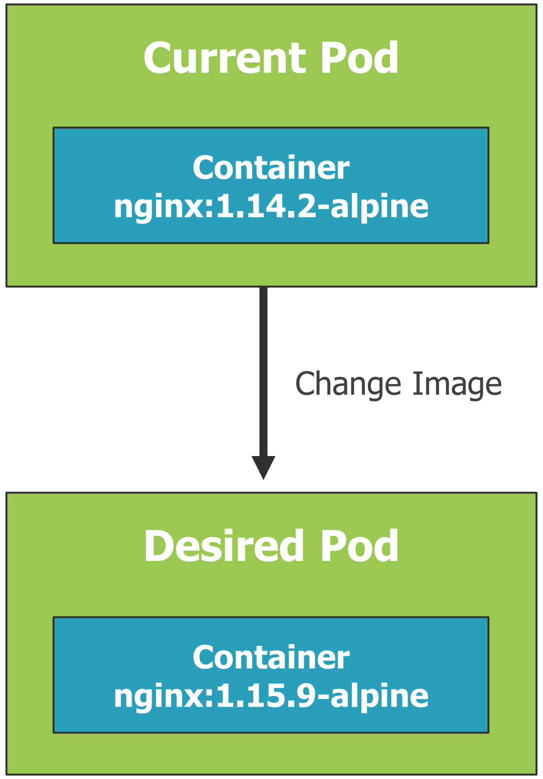
Update the YAML file (declarative) or use the kubectl scale command

```
spec:
  replicas: 5
  selector:
    tier: bizrules
```

Kubernetes Deployments in Action

Kubernetes Deployment Options

How Do You Update Existing Pods?



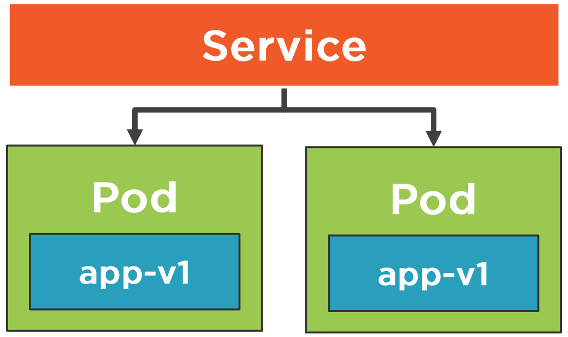
Delete all existing Pods and replace with new Pods? Leads to a short down-time.

Start new Pods and then delete old Pods? Need to be able to run two versions simultaneously.

Replace existing Pods one by one without impacting traffic to Pods?

Delete Existing Pods – Replace with New Pods

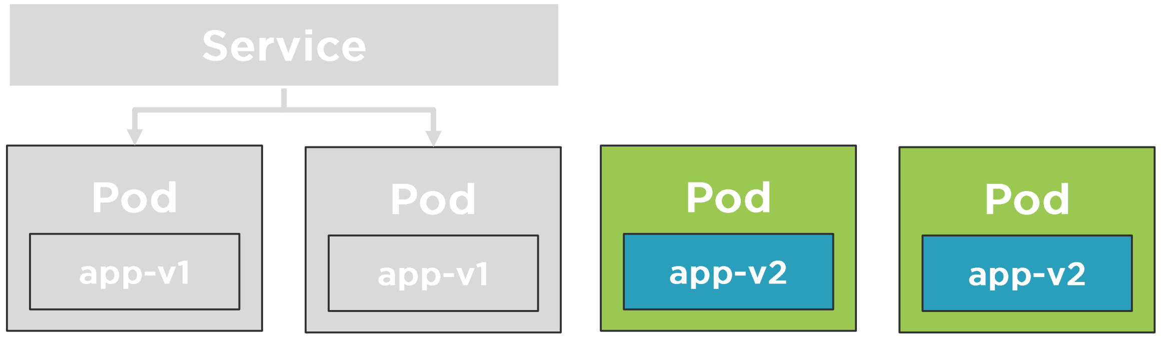
Initial Pod State



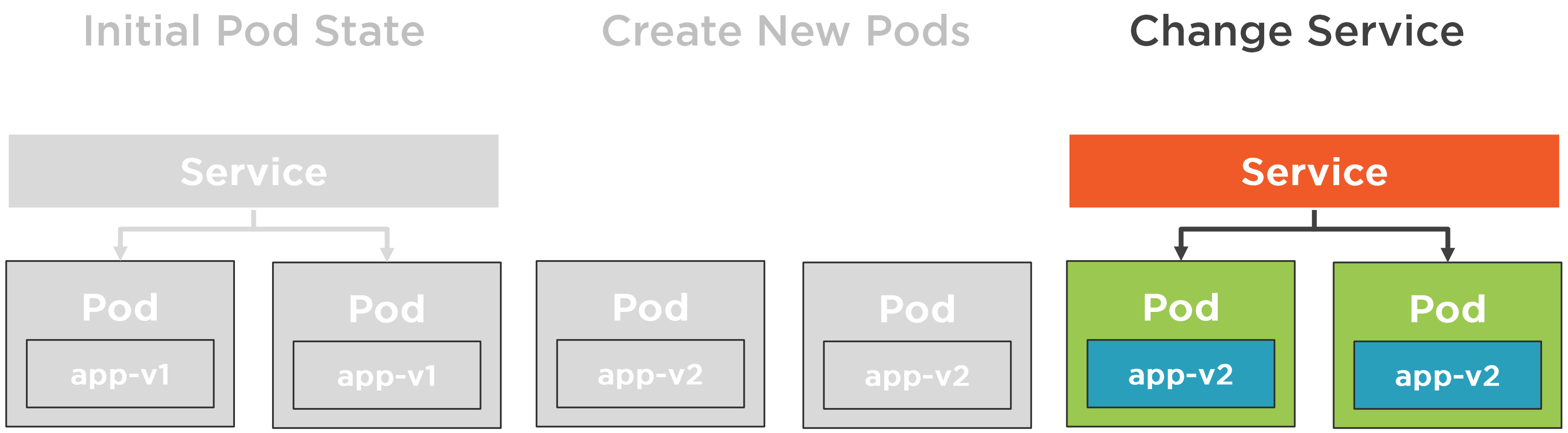
Delete Existing Pods – Replace with New Pods

Initial Pod State

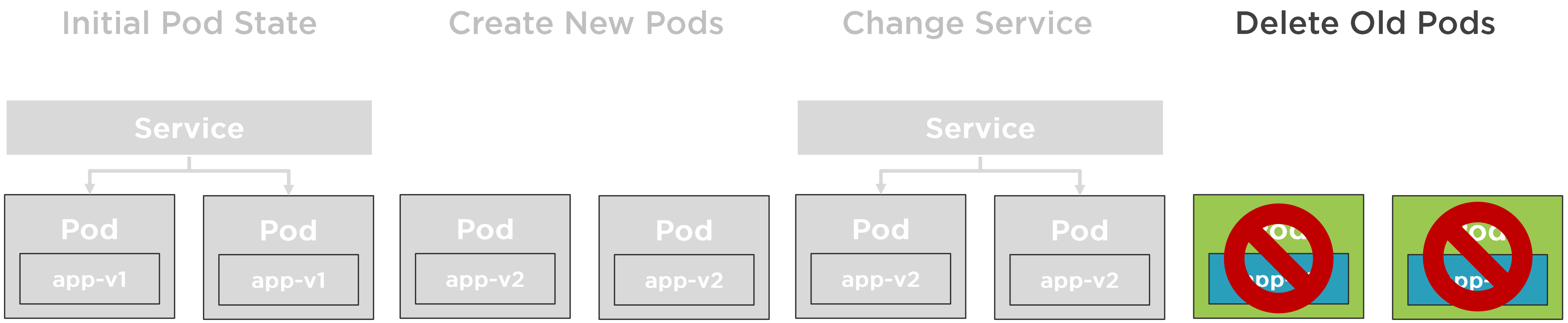
Create New Pods



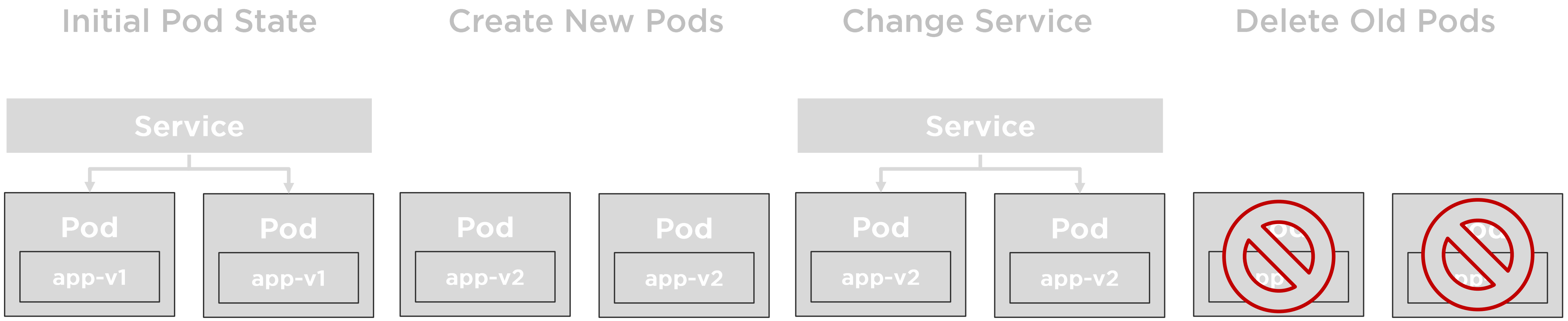
Delete Existing Pods – Replace with New Pods



Delete Existing Pods – Replace with New Pods

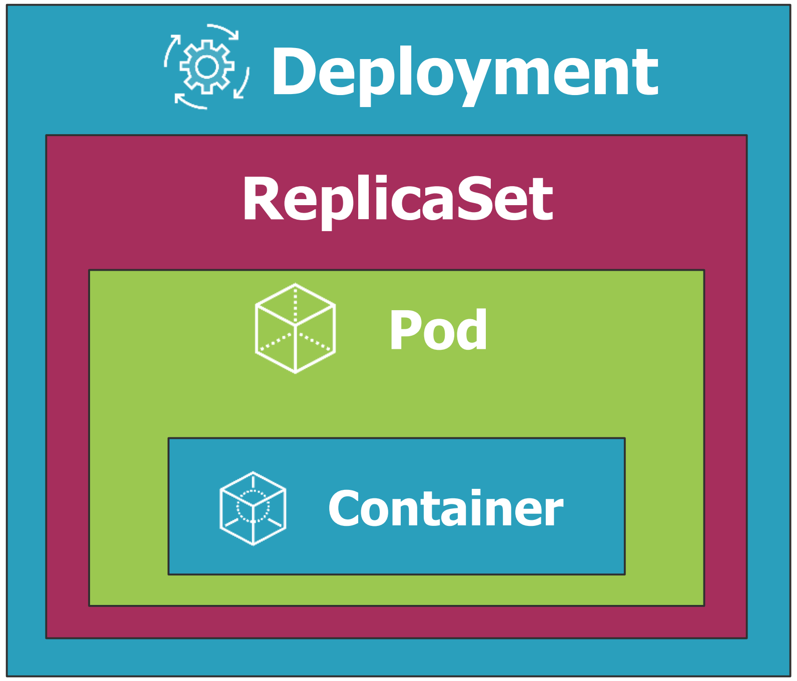


Delete Existing Pods – Replace with New Pods



Can your servers run the new and old Pods at the same time?

Deployment Options



One of the strengths of Kubernetes is "zero-downtime deployments"

Update an application's Pods without impacting end users

Several options are available:

- Rolling Updates
- Blue-Green Deployments
- Canary Deployments
- Rollbacks

Zero-downtime deployments allow software updates to be deployed to production without impacting end users.

Summary

Deployments are a key resource provided by Kubernetes

Deployments rely on ReplicaSets to schedule and manage Pods

Kubernetes supports Zero-downtime deployments out of the box

Several Deployment options exist:

- Zero-downtime
- Rolling Updates
- Canary
- Blue-Green