# Performing Rolling Update Deployments

**Dan Wahlin**
WAHLIN CONSULTING

@danwahlin www.codewithdan.com

## Module Overview

Understanding Rolling Update Deployments

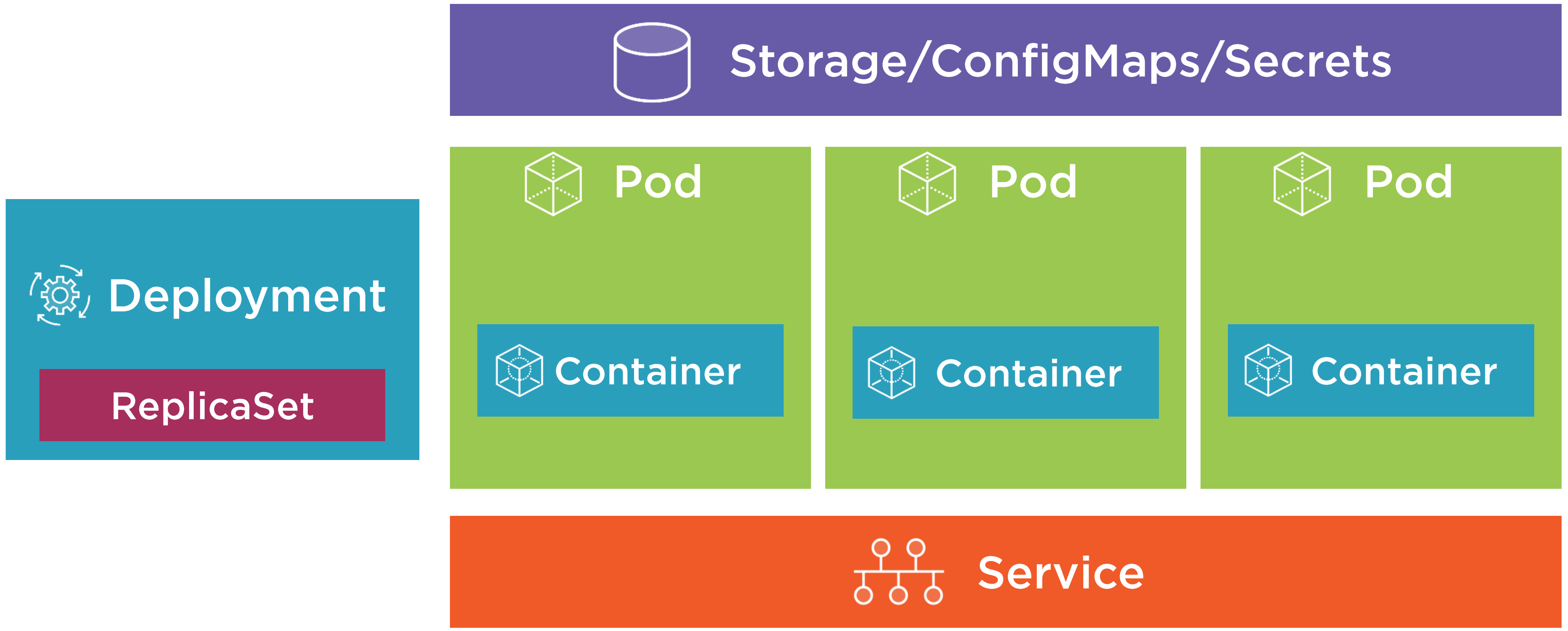
Creating a Rolling Update Deployment

Rolling Update Deployment in Action

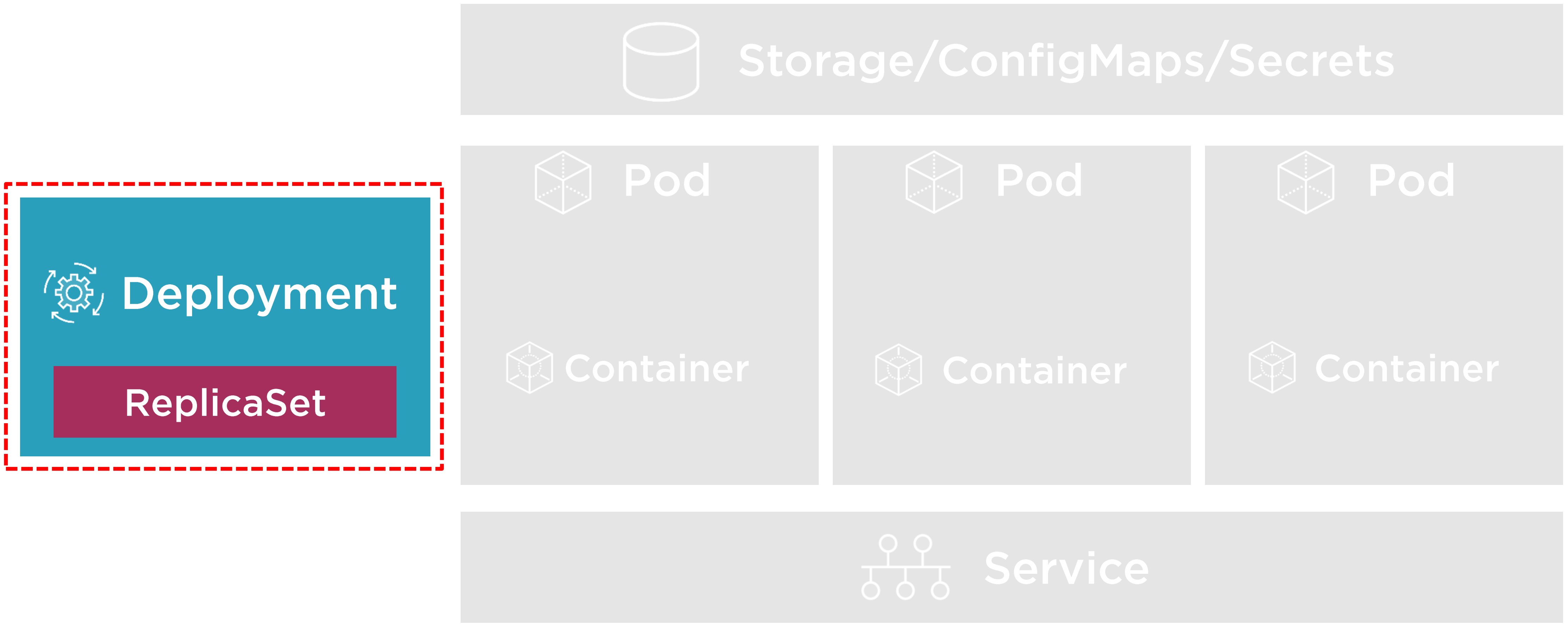
Rolling Back Deployments

Rolling Back Deployments in Action

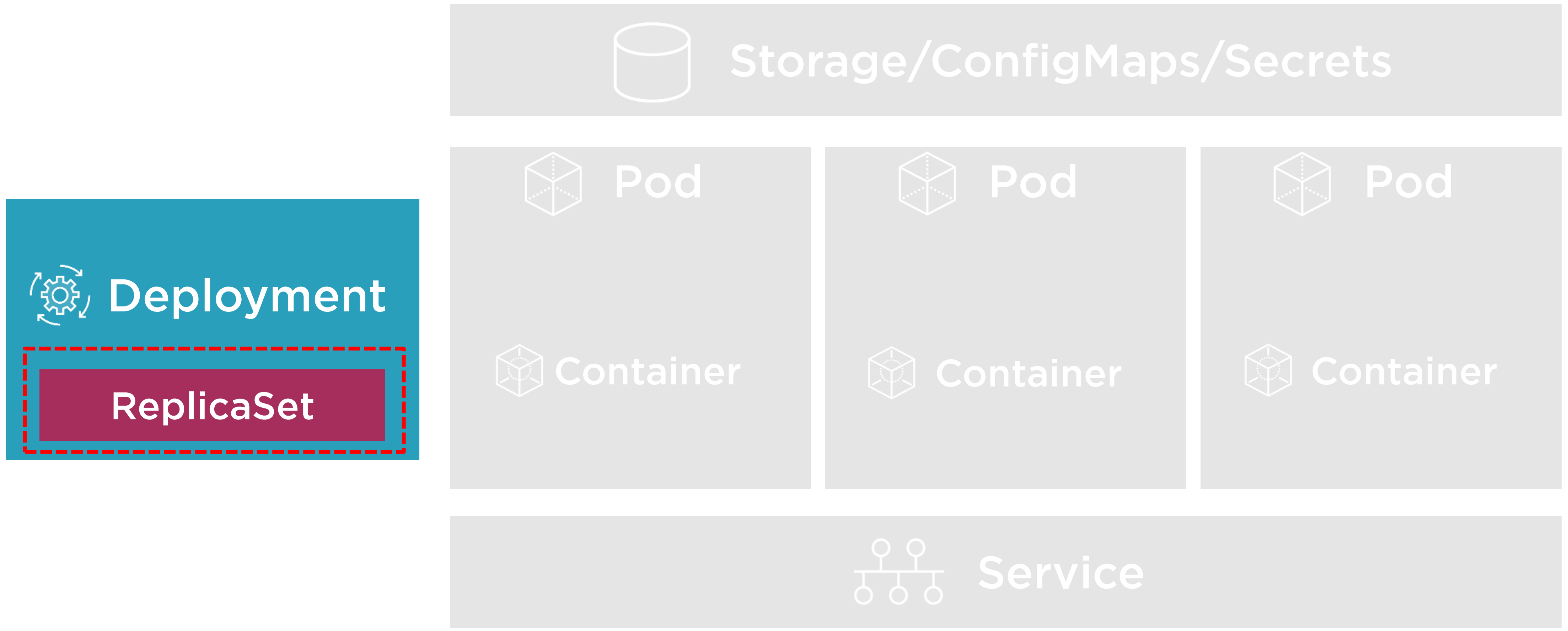
# Kubernetes Resources



# Kubernetes Resources



# Kubernetes Resources

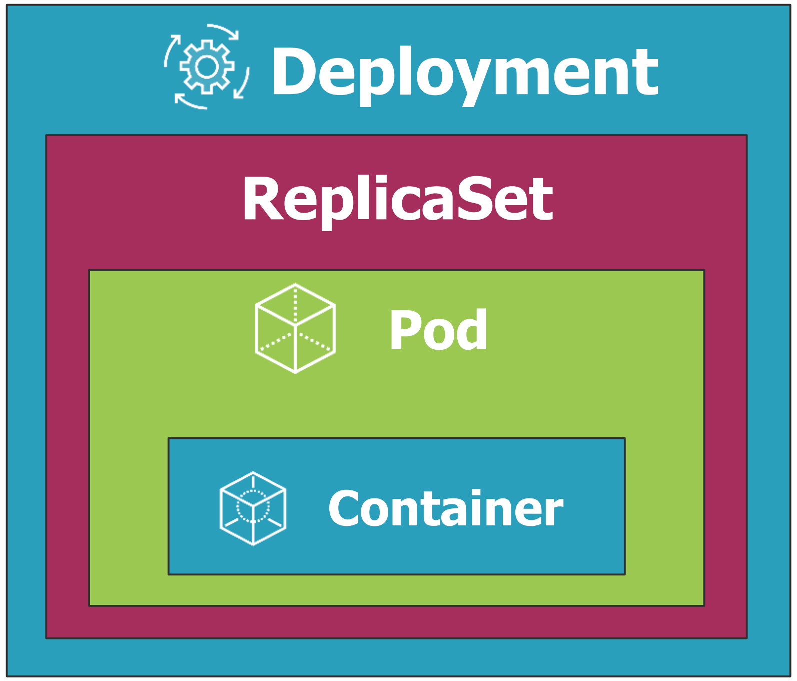


## Understanding Rolling Update Deployments

"Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones."

**Kubernetes Documentation**

## Rolling Update Deployments



**ReplicaSets increase new Pods while decreasing old Pods**

**Service handles load balancing traffic to available Pods**

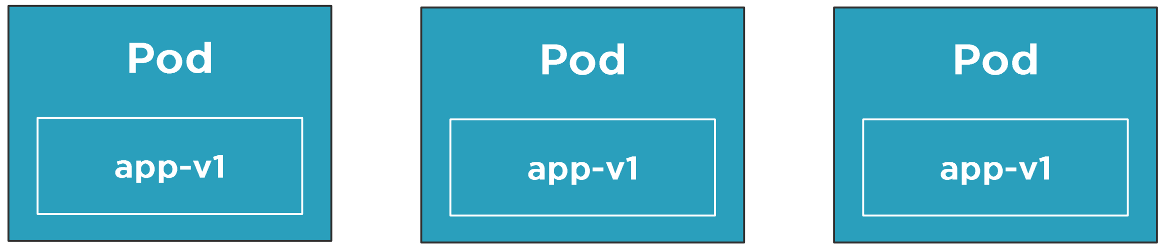
**New Pods only scheduled on available Nodes**

**Deployments support two strategy options:**

- Rolling Update (default and our focus here)
- Recreate (can result in down-time)

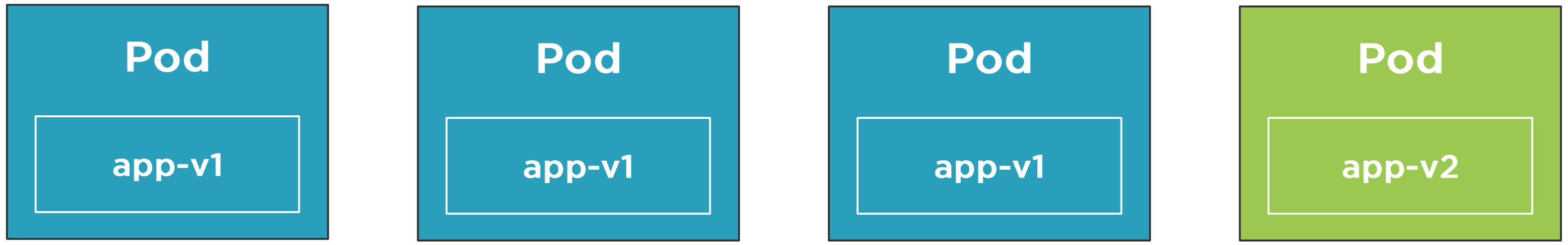
## Rolling Update Deployments

### Initial Pod State



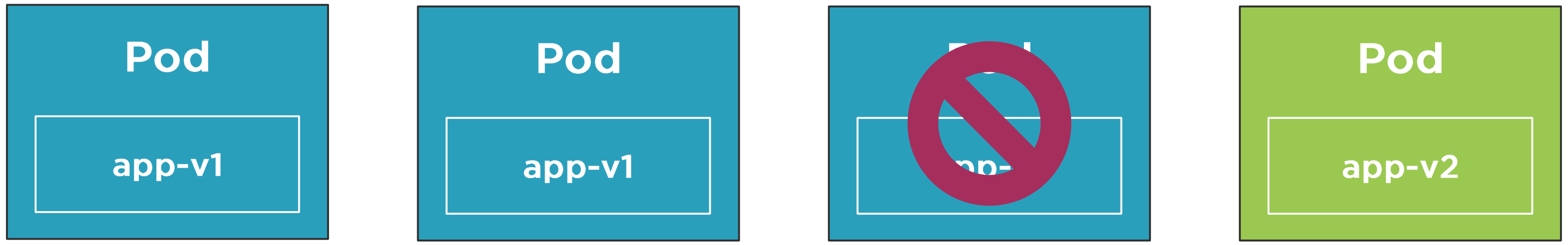
## Rolling Update Deployments

### Rollout New Pod



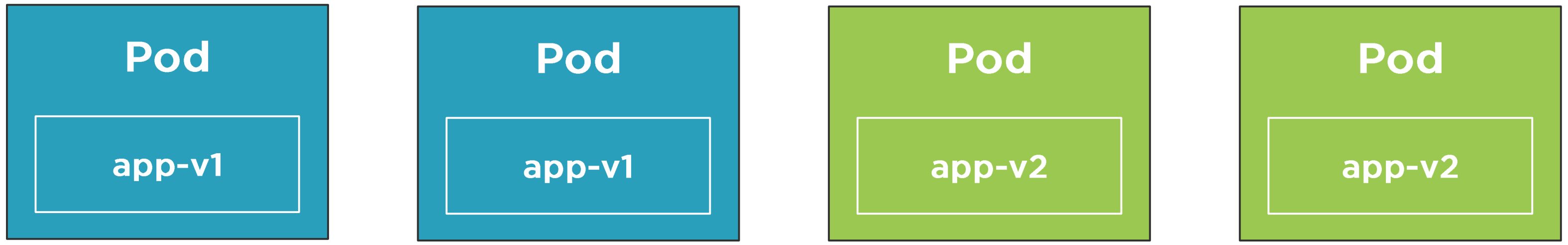
# Rolling Update Deployments

**Delete Pod**



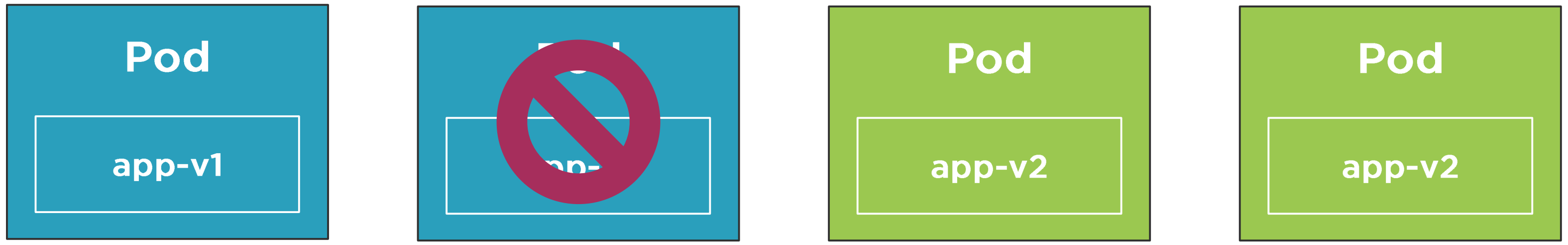
## Rolling Update Deployments

### Rollout New Pod



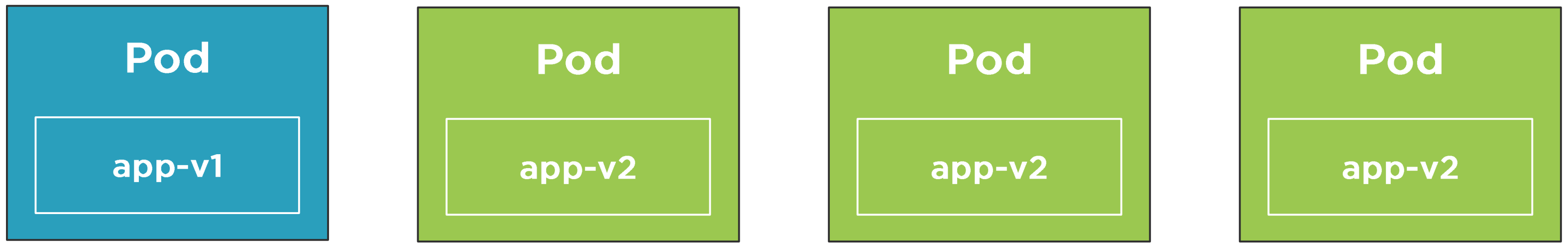
# Rolling Update Deployments

**Delete Pod**



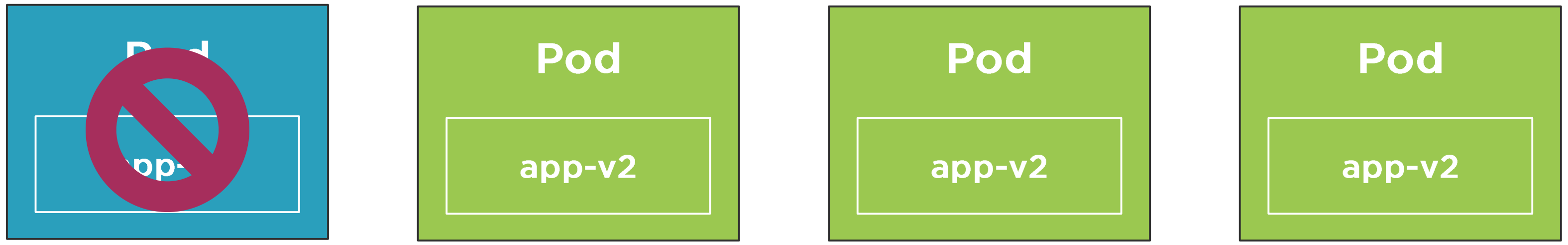
# Rolling Update Deployments

## Rollout New Pod



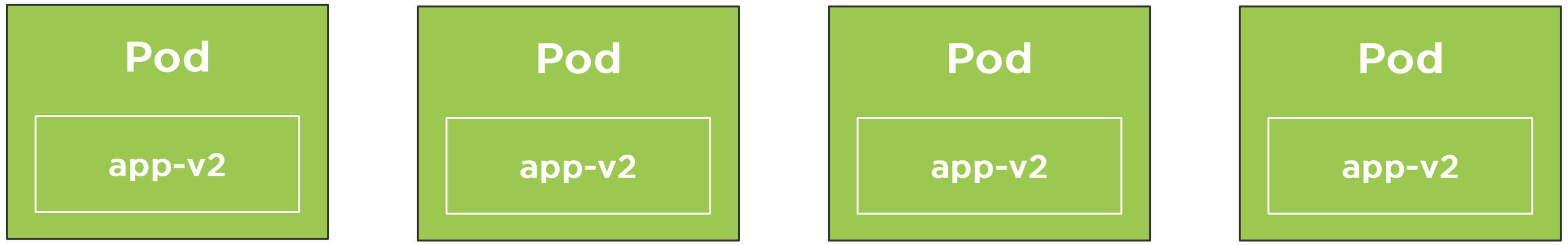
## Rolling Update Deployments

**Delete Pod**

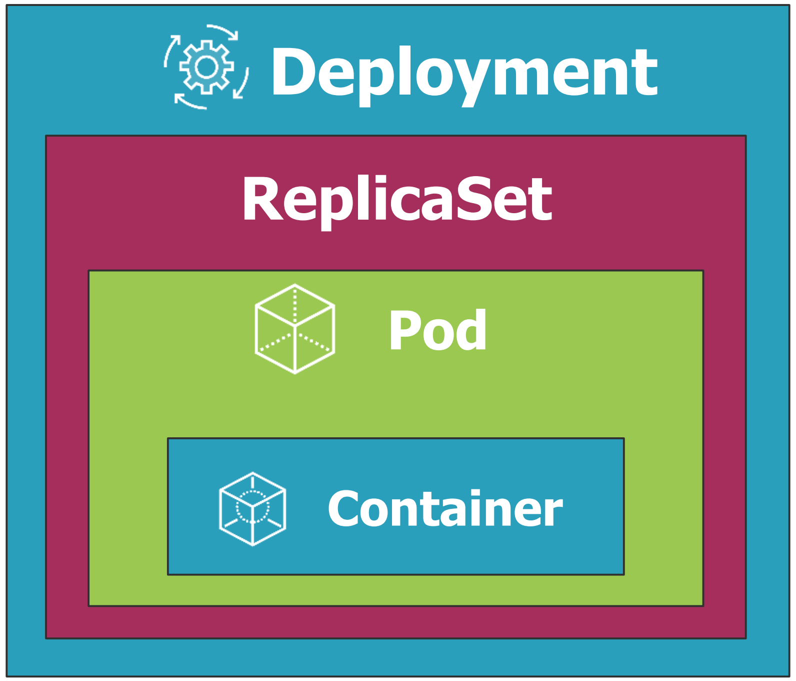


## Rolling Update Deployments

### Rollout New Pod



# Deployments and Replicasets



## Creating a Rolling Update Deployment

# Defining a Rolling Update Deployment

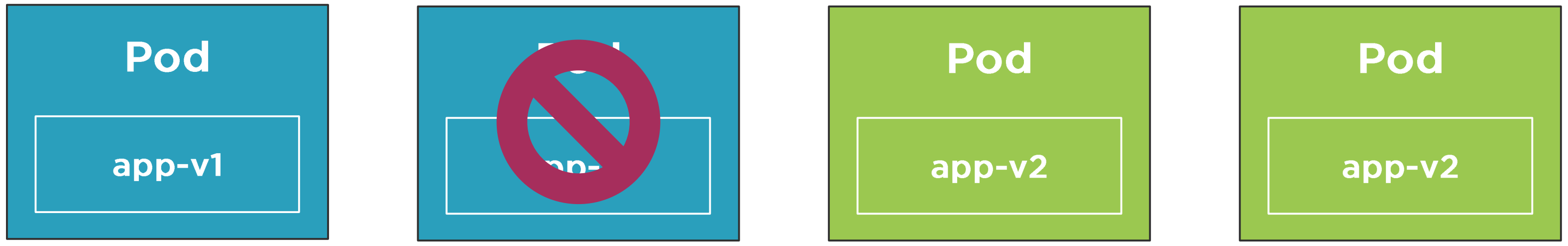
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 2
  minReadySeconds: 1
  progressDeadlineSeconds: 60
  revisionHistoryLimit: 5
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1

  ...
```

- ◀ Number of Pod replicas
- ◀ Seconds new Pod should be ready to be considered healthy (0)
- ◀ Seconds to wait before reporting stalled Deployment
- ◀ Number of ReplicaSets that can be rolled back (10)
- ◀ RollingUpdate (default) or Recreate strategy
- ◀ Max Pods that can exceed the replicas count (25%)
- ◀ Max Pods that are not operational (25%)

# Understanding maxSurge

**How many Pods can be added above the replicas count during the rolling update?**

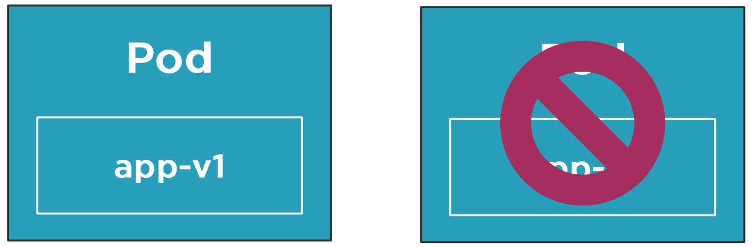


| | |
|---|---|
| replicas | = 2 |
| maxSurge | = 1 |
| current | = 3 |

1 above replicas count which is allowed

## Understanding maxUnavailable

**How many of the existing Pods can be made unavailable during a rolling update?**



maxUnavailable = 1

It's OK for 1 of the 2 replicas to be unavailable

## Creating the Deployment

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

Record the command in the Deployment revision history

```
# Create initial deployment
kubectl create -f file.deployment.yml --save-config --record
```

Save configuration in resource's annotations

## Creating or Modifying a Deployment

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

Record the command in the Deployment revision history

```
# Create initial deployment
kubectl apply -f file.deployment.yml --record
```

## Checking the Deployment Status

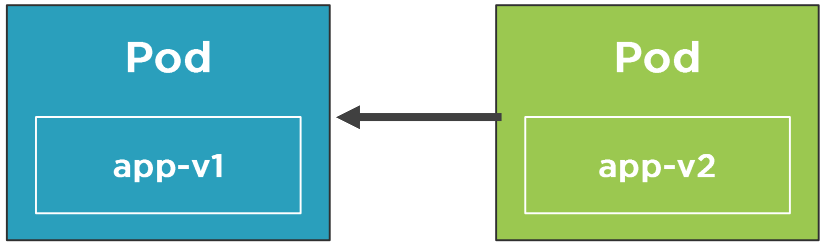
**The kubectl rollout status command can be used to get information about a specific Deployment**

```
# Get information about a Deployment
kubectl rollout status deployment [deployment-name]
```

## Rolling Update Deployment in Action

## Rolling Back Deployments

## Rolling Back Deployments



**Rolling update revisions can be tracked using --record**

**If a Deployment has issues, a new Deployment can be applied, or you can revert to a previous revision**

**Several kubectl commands can be used for rollbacks:**

- kubectl rollout status
- kubectl rollout history
- kubectl rollout undo

## Checking Deployment History

**The kubectl rollout history command can be used to view history of a Deployment**

```
# Get information about a Deployment
kubectl rollout history deployment [deployment-name]

# Get information about a Deployment
kubectl rollout history deployment [deployment-name] --revision=2
```

## Rolling Back a Deployment

**Use the kubectl rollout undo command to rollback to a specific Deployment revision**

```
# Check status
kubectl rollout status -f file.deployment.yml

# Rollback a Deployment
kubectl rollout undo -f file.deployment.yml

# Rollback to a specific revision
kubectl rollout undo deployment [deployment-name] --to-revision=2
```

## Rolling Back Deployments in Action

# Summary

Rolling updates are the default Deployment strategy used by Kubernetes

Ensures zero-downtime during a Deployment

Maximum and minimum Pods available during a Deployment can be defined

Deployments can be recorded and stored in history using --record

Deployments can be rolled back to a specific revision