Maintaining Applications with Deployments

Anthony E. Nocentino
ENTERPRISE ARCHITECT @ CENTINO SYSTEMS

@nocentino www.centinosystems.com

Course Overview

Using Controllers to Deploy Applications and Deployment Basics

Maintaining Applications with Deployments

Deploying and Maintaining Applications with DaemonSets and Jobs

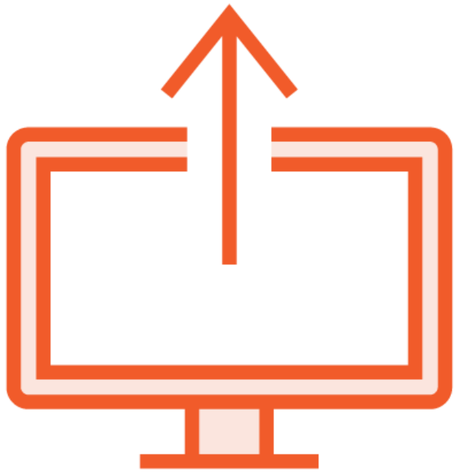
Overview

Configuring and Managing Application State with `Deployments`

- Updating `Deployments`
- Controlling Rollouts
- Scaling Applications

Managing Application State with Deployments

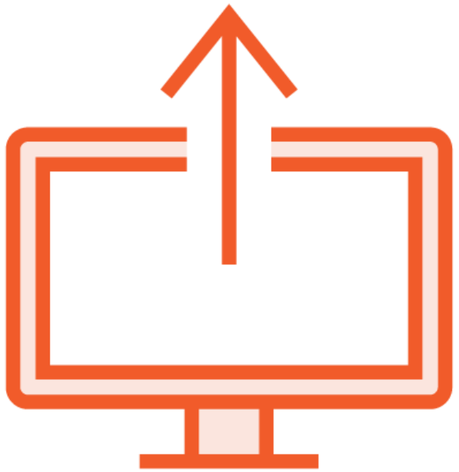
Creating



Updating

Scaling

Updating a Deployment

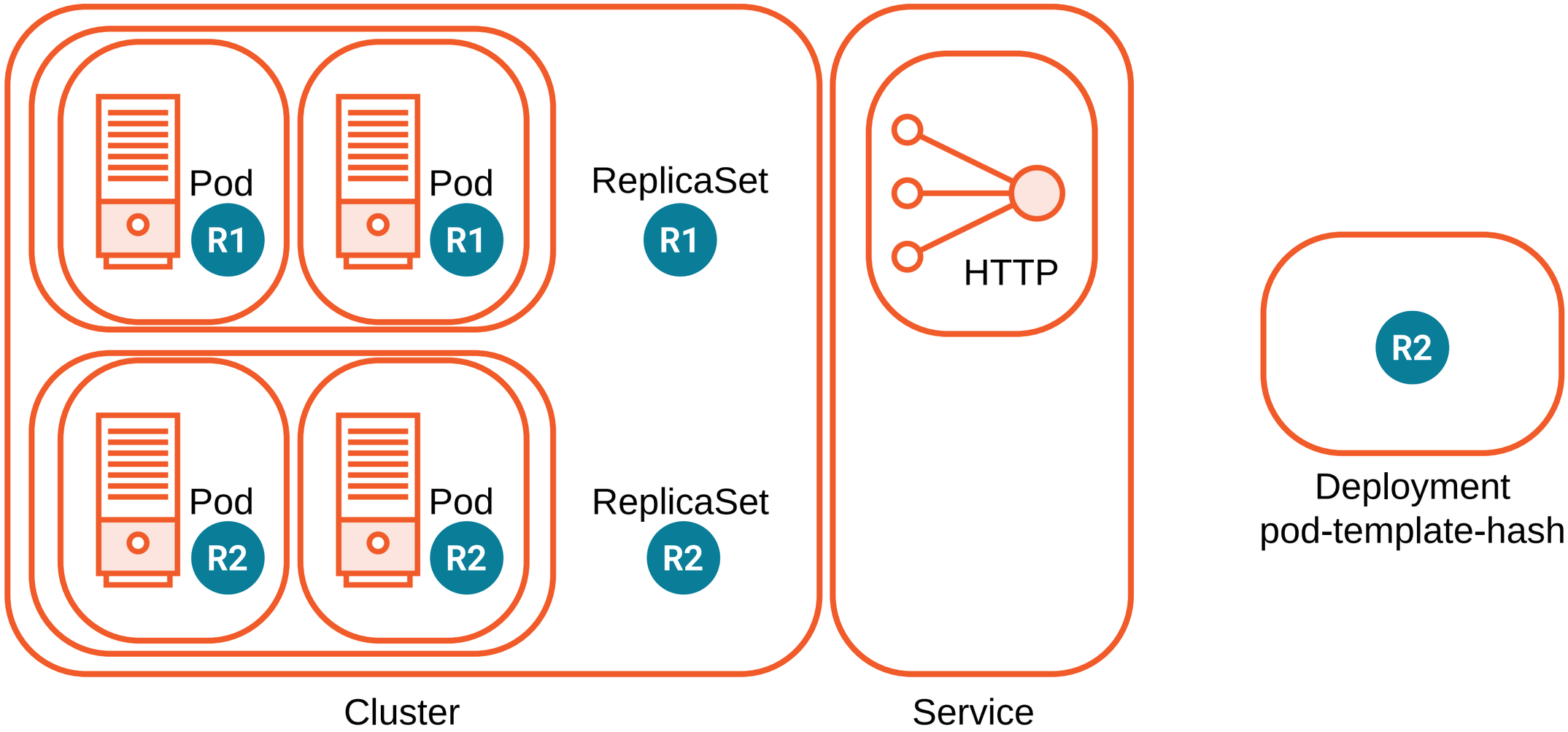


Rolling out a new container image

Triggered by changing the Pod Template

Other fields can be changed without triggering an update

Controller Operations - Deployment Updates



Updating a Deployment Object

```
kubectl set image deployment hello-world hello-world=hello-app:2.0
```

```
kubectl set image deployment hello-world hello-world=hello-app:2.0 --record
```

```
kubectl edit deployment hello-world
```

```
kubectl apply -f hello-world-deployment.yaml --record
```

Checking Deployment Status

```
kubectl rollout status deployment [name]

kubectl describe deployment [name]
```

Deployment Status

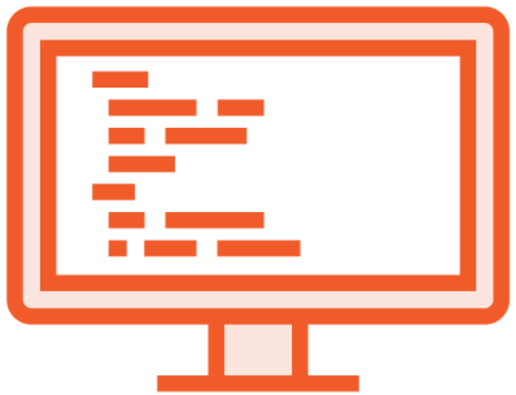
- Complete - all update work is finished
- Progressing - update in flight
- Failed - update could not complete

Demo

Updating a `Deployment`

Checking `Deployment` Rollout Status

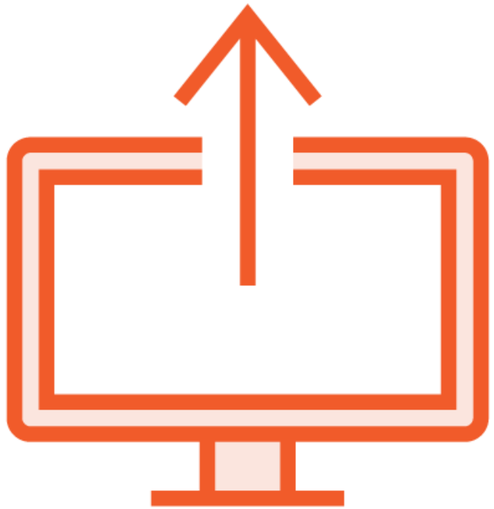
Using Deployments to Change State



Control rollouts of a new version of your application

- Update Strategy
- Pause to make corrections
- Rollback to an earlier version
- Restart a Deployment

Controlling Rollouts With Update Strategy



Controls Pods rollout

`RollingUpdate` (Default)

A new `ReplicaSet` starts scaling up and the old `ReplicaSet` starts scaling down

`Recreate`

Terminates all Pods in the current `ReplicaSet` set prior to scaling up the new `ReplicaSet`

Used when applications don’t support running different versions concurrently

Controlling the RollingUpdate Strategy

`maxUnavailable`	`maxSurge`
Ensures only a certain number of Pods are unavailable being updated	Ensure that only a certain number of Pods are created above the desired number of Pods

Successfully Controlling Deployment Rollouts

Update Strategy in a Deployment Spec

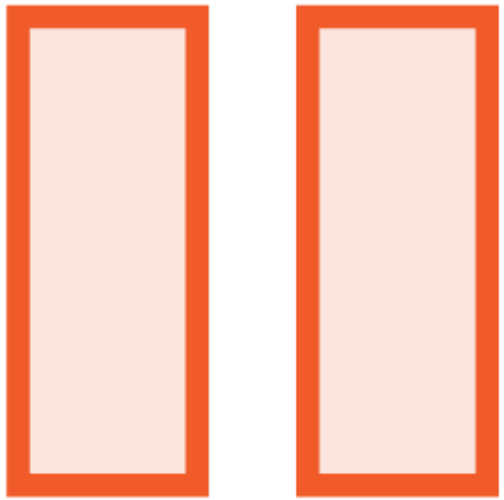
Readiness Probes in your Pod Template Spec

Update Strategy

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  replicas: 20
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 20%
      maxSurge: 5
...
```

```
template:
...
    spec:
      containers:
...
        readinessProbe:
          httpGet:
            path: /index.html
            port: 8080
          initialDelaySeconds: 10
          periodSeconds: 10
```

Pausing and Resuming a Deployment



Changes to the `Deployment` while paused are not rolled out

Batch changes together, then resume the rollout

The current state of the Deployment is maintained until it's resumed

Starts up a new `ReplicaSet` with the new changes

```
  kubectl rollout pause deployment \
my-deployment
```

```
  kubectl rollout resume deployment \
my-deployment
```

Rolling Back a Deployment

Rollout history

- `CHANGE-CAUSE` Annotation `Deployment`

Revision History

- `revisionHistoryLimit` defaults to 10
- Number of `ReplicaSets` retained in history
- Used for rolling back
- Can be set to 0 for immediate cleanup

Rolling Back a Deployment (con’t)

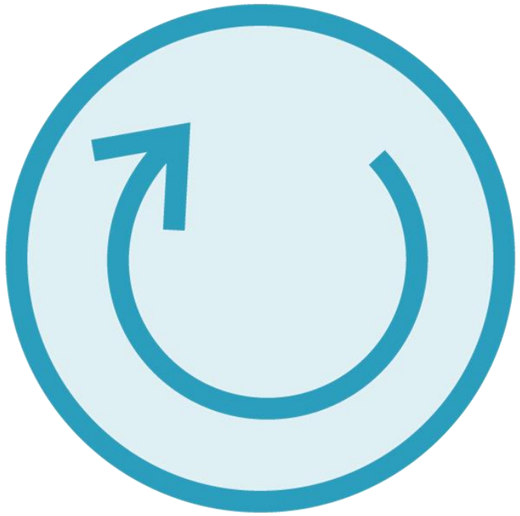
```
kubectl rollout history deployment \
hello-world

kubectl rollout history deployment \
hello-world --revision=1

kubectl rollout undo deployment
hello-world

kubectl rollout undo deployment \
hello-world --to-revision=1
```

Restarting a Deployment



Effectively restarts all the Pods

But no Pod is ever “recreated”

New ReplicaSet with the same Pod Spec

Uses Deployment’s Update Strategy

- `RollingUpdate`
- `Recreate`

```
kubectl rollout restart \
  deployment hello-world
```

Demo

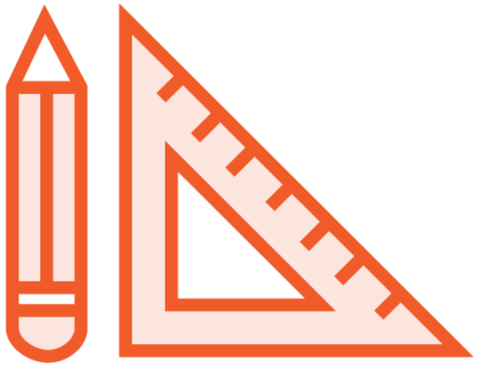
Rolling Back a `Deployment`

Controlling the rate of a Rollout

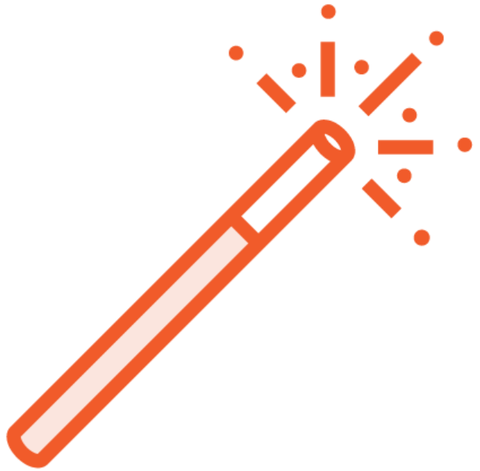
Using Readiness Probes to Control Rollout

Restarting a Deployment

Scaling Deployments



Manual



Horizontal Pod Autoscaler

```
kubectl scale deployment hello-world --replicas=10
kubectl apply -f deployment.yaml
```

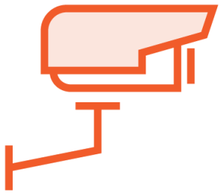
Demo

Scaling a `Deployment`

Deployment Tips

Control your rollouts with an Update Strategy appropriate for your application

Use Readiness Probes for your application



Use the `--record` option to leave a trail of your work for others

Review

Configuring and Managing Application State with `Deployments`

- Using `Deployments` to Change State
- Controlling Rollouts
- Scaling Applications

What’s Next!

Deploying and Maintaining Applications with DaemonSets and Jobs