阿里云 × CLOUD NATIVE COMPUTING FOUNDATION

云原生技术公开课

第 06 讲

应用编排与管理：Deployment

酒祝 阿里巴巴高级开发工程师

关注“阿里巴巴云原生”公众号
获取第一手技术资料

1 需求来源

2 用例解读

3 操作演示

4 架构设计

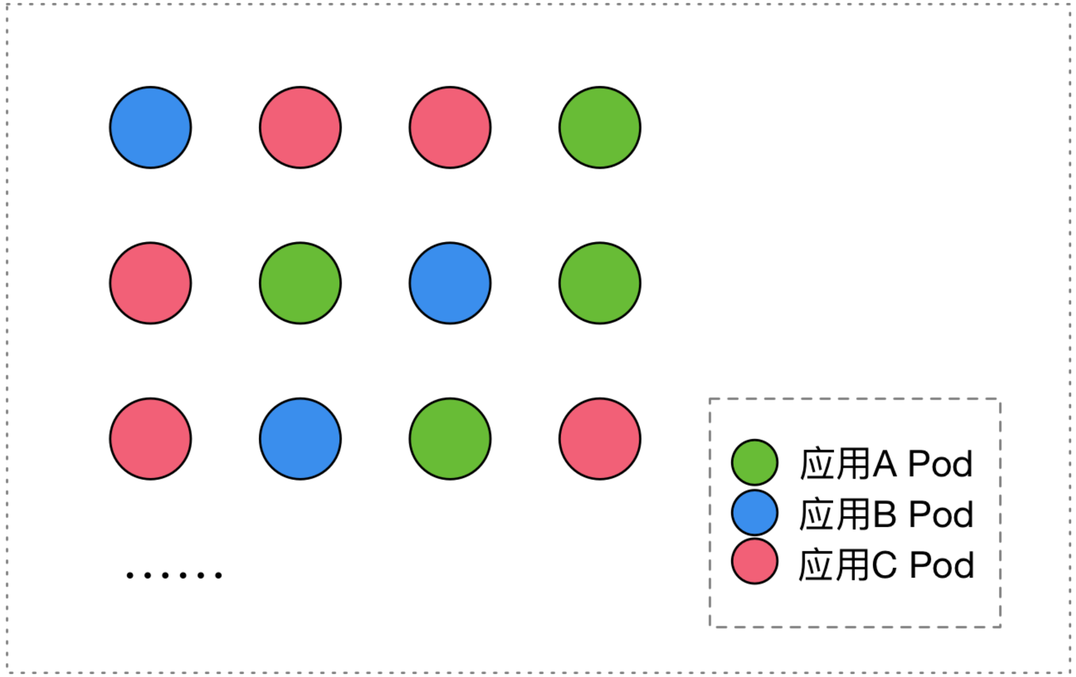
5 课后思考实践

背景问题

我们可以直接管理集群中所有的Pod吗？

如果这样做，以下的问题有什么方式来解决？

1. 如何保证集群内可用Pod的数量
2. 如何为所有Pod更新镜像版本
3. 更新的过程中，如何保证服务可用性
4. 更新的过程中，发现问题如何快速回滚



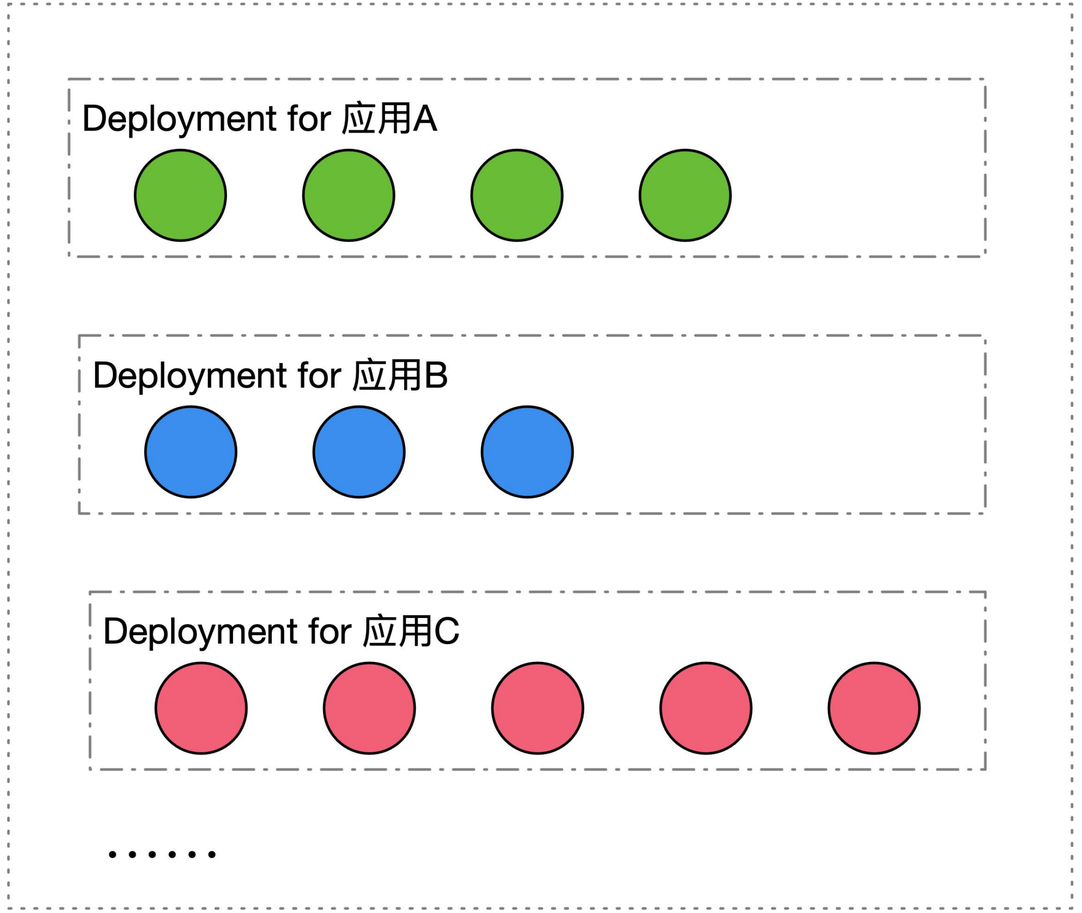
Deployment: 管理部署发布的控制器

Deployment能帮助我们做什么事情？

1. 定义一组Pod的期望数量， controller会维持Pod数量与期望数量一致

2. 配置Pod发布方式，controller会按照给定策略更新Pod，保证更新过程中不可用的pod数量在限定范围内

3. 如果发布有问题，支持“一键”回滚



1 需求来源

2 用例解读

3 操作演示

4 架构设计

5 课后思考实践

Deployment语法

新知识点：
replicas：终态数量
template：pod模板

往期回顾：
labels：标签
selector：选择器
pod image：镜像版本

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Deployment元信息

期望Pod数量

Pod的选择器

Pod模板

查看Deployment状态

```
$ kubectl create -f nginx-deployment.yaml
```

```
$ kubectl get deployment
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3         3         3            3           80m
```

DESIRED：期望pod数量（replicas）
CURRENT：当前实际的pod数量
UP-TO-DATE：到达期望版本的pod数量
AVAILABLE：运行中并可用的pod数量
AGE：deployment创建的时长

查看Pod

```
$ kubectl get pod
NAME
nginx-deployment-5c689d88bb-ck974
nginx-deployment-5c689d88bb-f88bm
nginx-deployment-5c689d88bb-xjqd9
```

Pod名字格式：

${deployment-name}-${template-hash}-${random-suffix}

```
$ kubectl get pod/nginx-deployment-5c689d88bb-ck974 -o yaml
apiVersion: v1
kind: Pod
metadata:
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicaSet
    name: nginx-deployment-5c689d88bb
    uid: 07980df9-51fe-11e9-b06e-264e0b6d8534
# ...
```

Pod owner：

ReplicaSet，而非Deployment

更新镜像

```
$ kubectl set image deployment.v1.apps/nginx-deployment nginx=nginx:1.9.1
```

设置镜像

资源类型
固定写法，也可写为
deployment或者
deployment.apps

要更新的
Deployment名字

要更新的
容器名字

新的镜像

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
    # . . .
    spec:
      containers:
      - name: nginx
        image: nginx:1.9.1
        ports:
        - containerPort: 80
```

快速回滚

```
$ kubectl rollout undo deployment/nginx-deployment
```

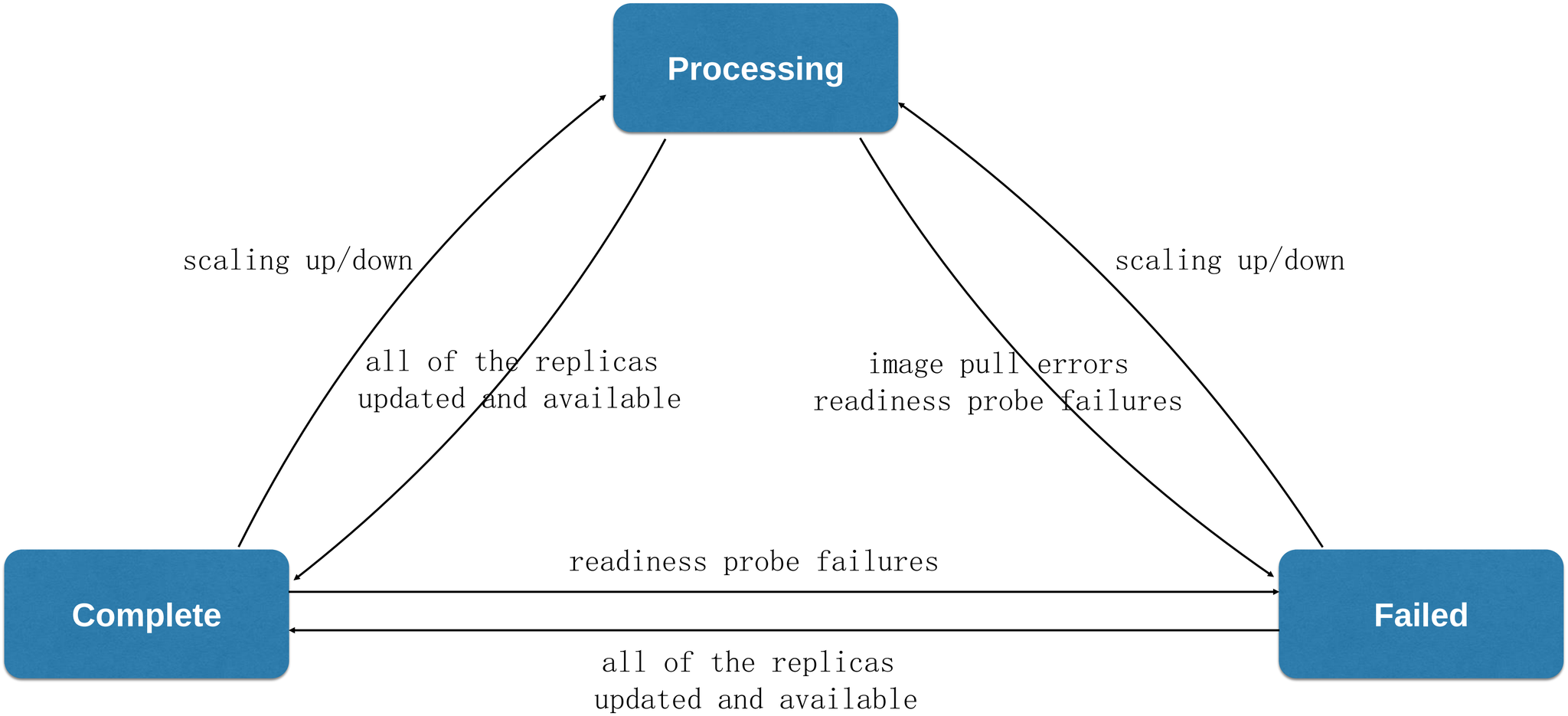
回滚到Deployment上一个版本

```
$ kubectl rollout undo deployment.v1.apps/nginx-deployment --to-revision=2
```

回滚到Deployment到某一个版本，需要先查询版本列表：

```
$ kubectl rollout history deployment.v1.apps/nginx-deployment
```

DeploymentStatus



1 需求来源

2 用例解读

3 操作演示

4 架构设计

5 课后思考实践

1 需求来源

2 用例解读

3 操作演示

4 架构设计

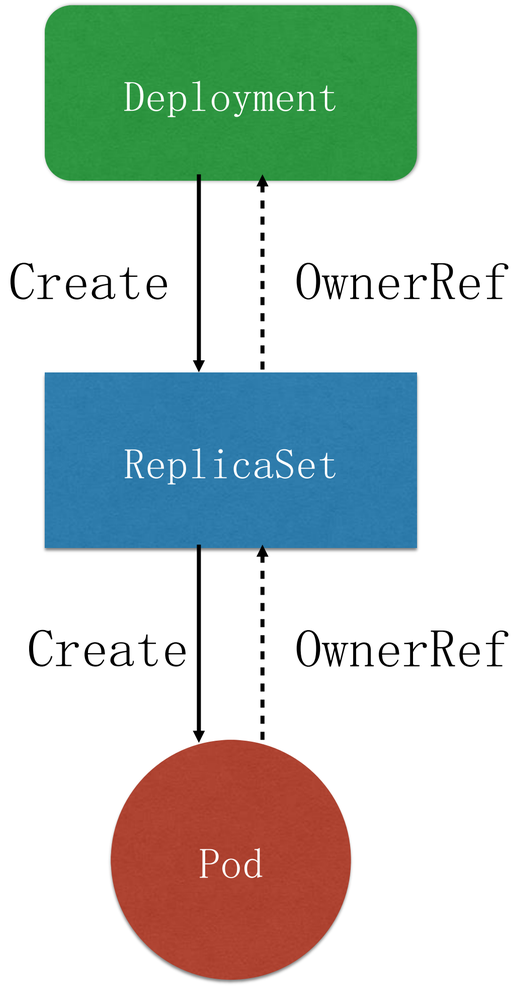
5 课后思考实践

管理模式

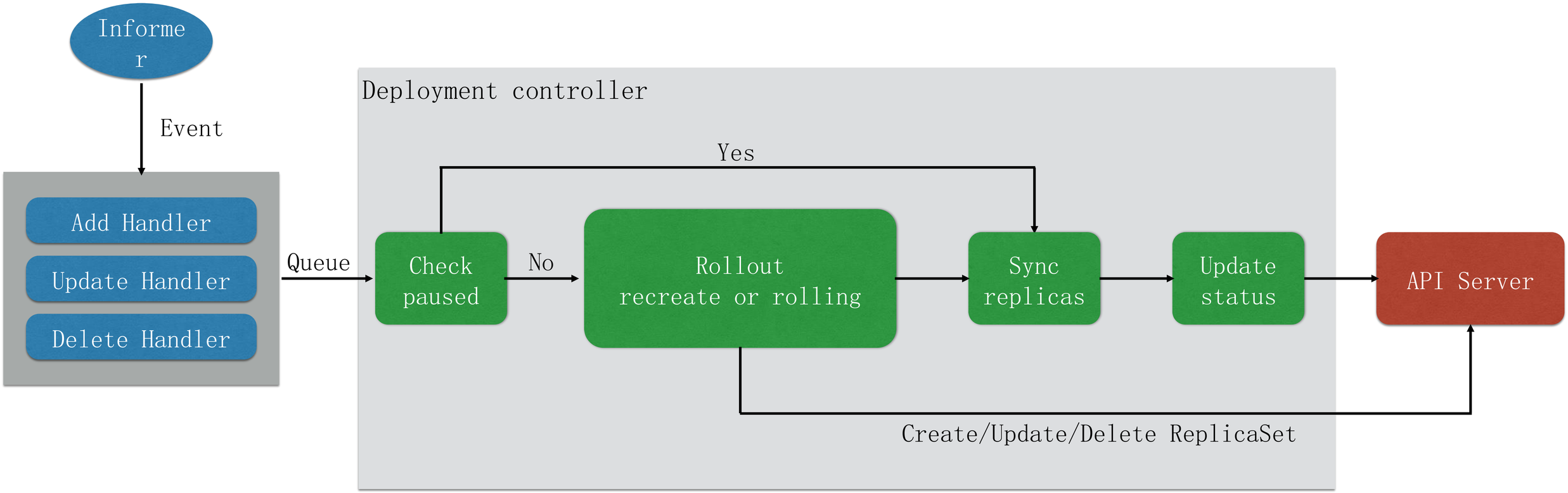
Deployment只负责管理不同版本的ReplicaSet，
由ReplicaSet管理Pod副本数

每个ReplicaSet对应了Deployment template的一个版本

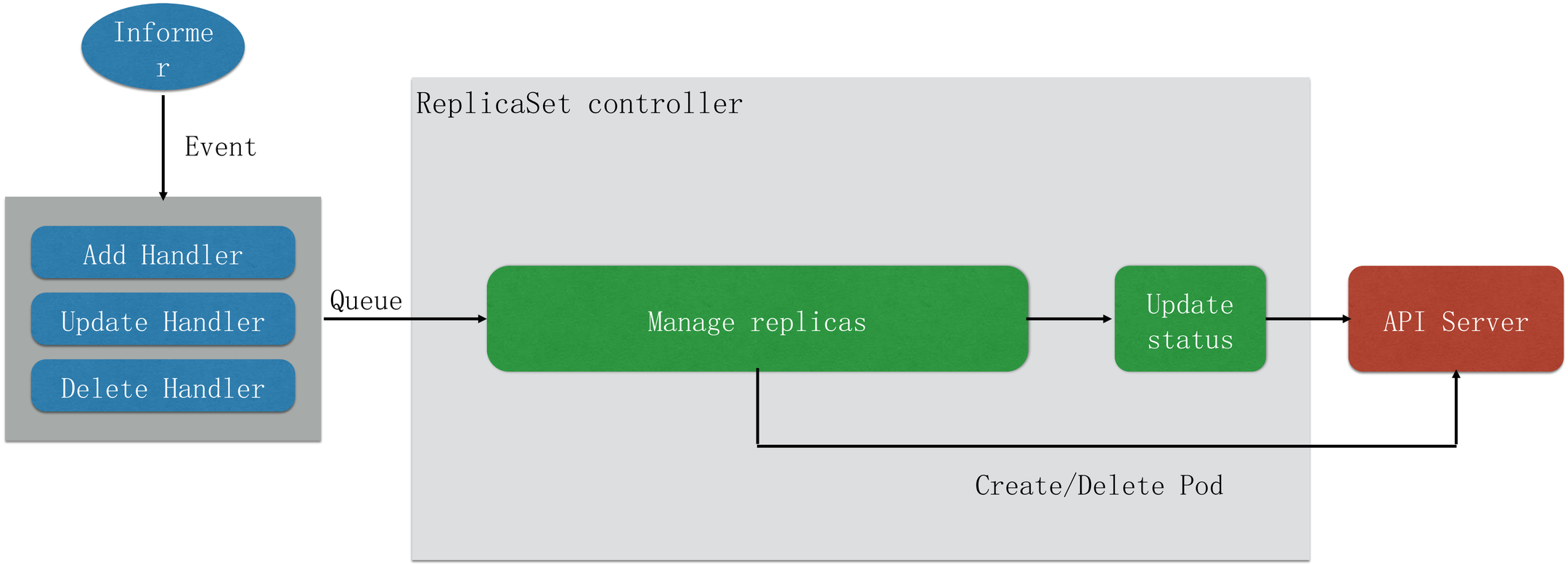
一个ReplicaSet下的Pod都是相同的版本



Deployment控制器



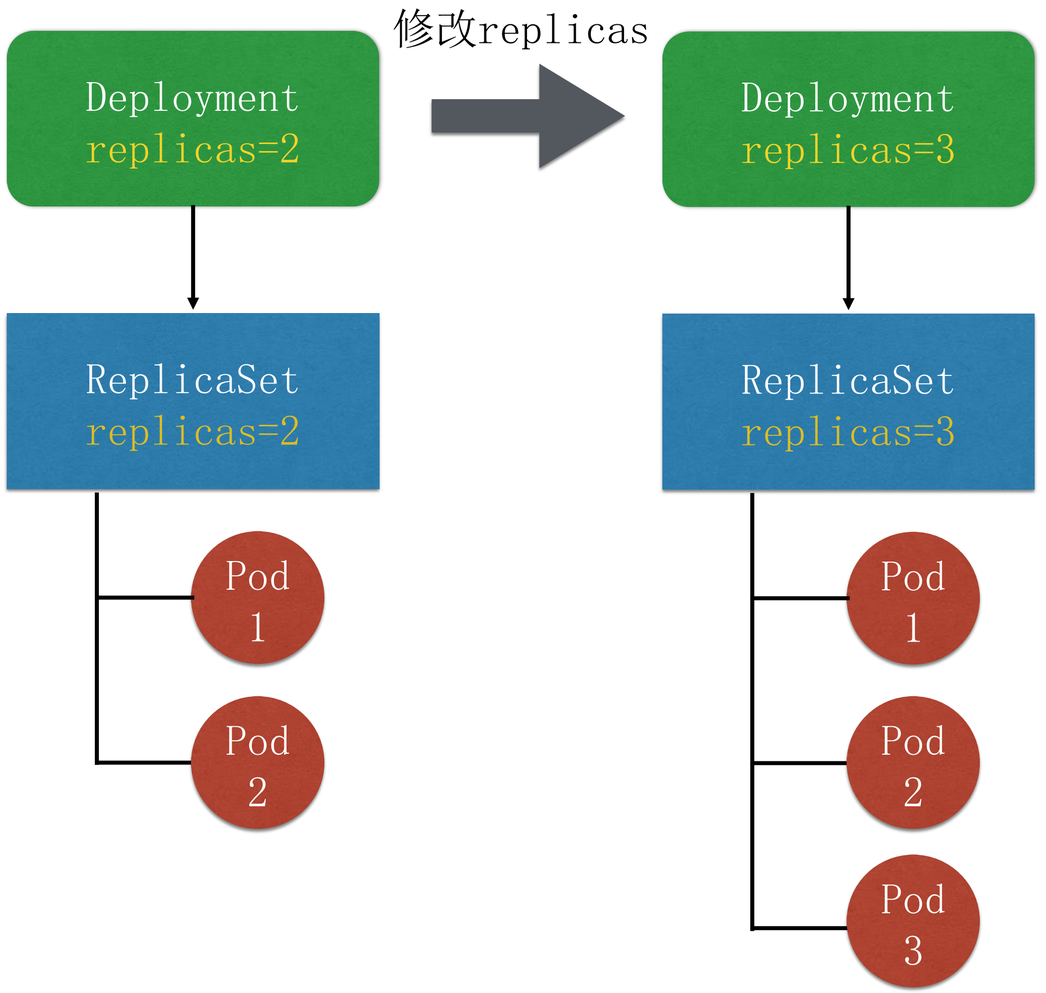
ReplicaSet控制器



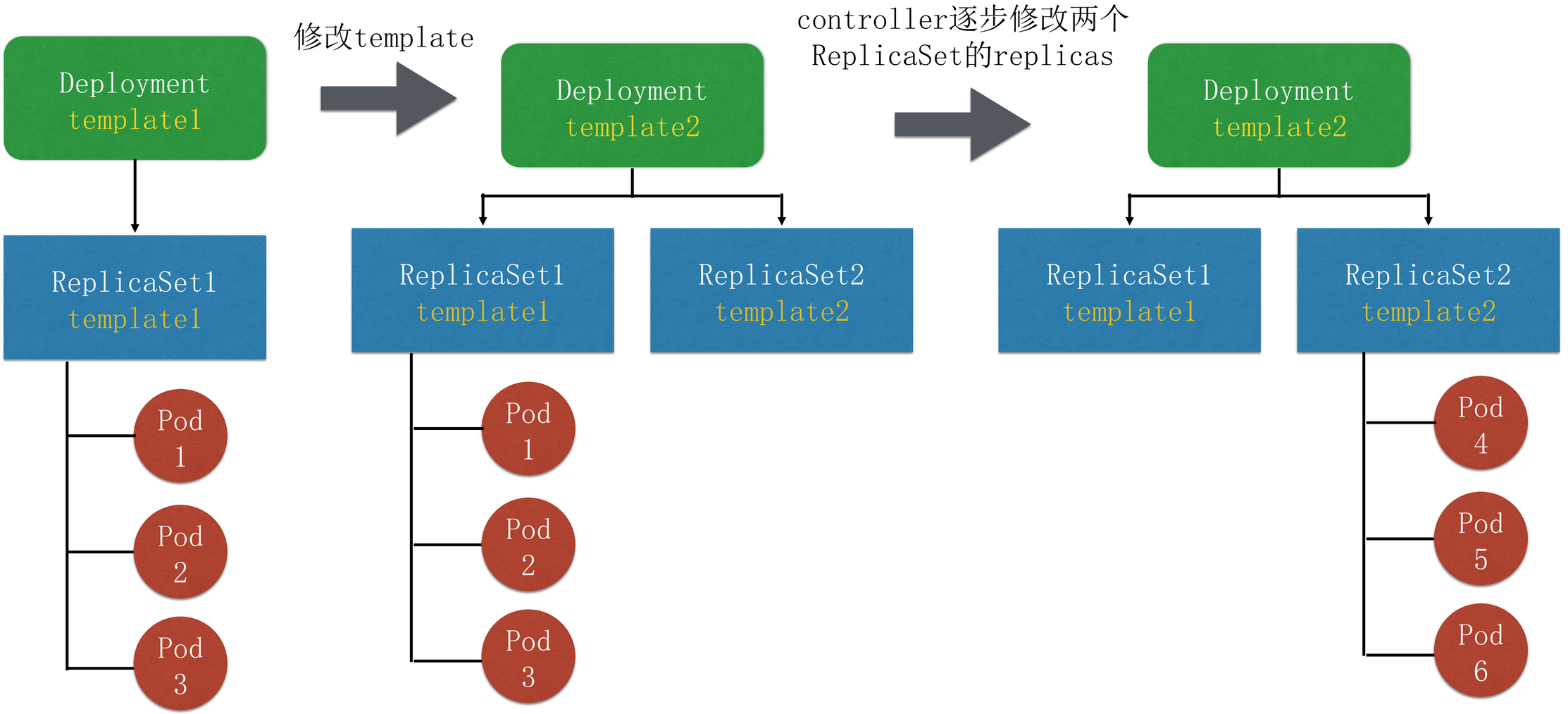
扩容模拟

Deployment的副本数由ReplicaSet管理

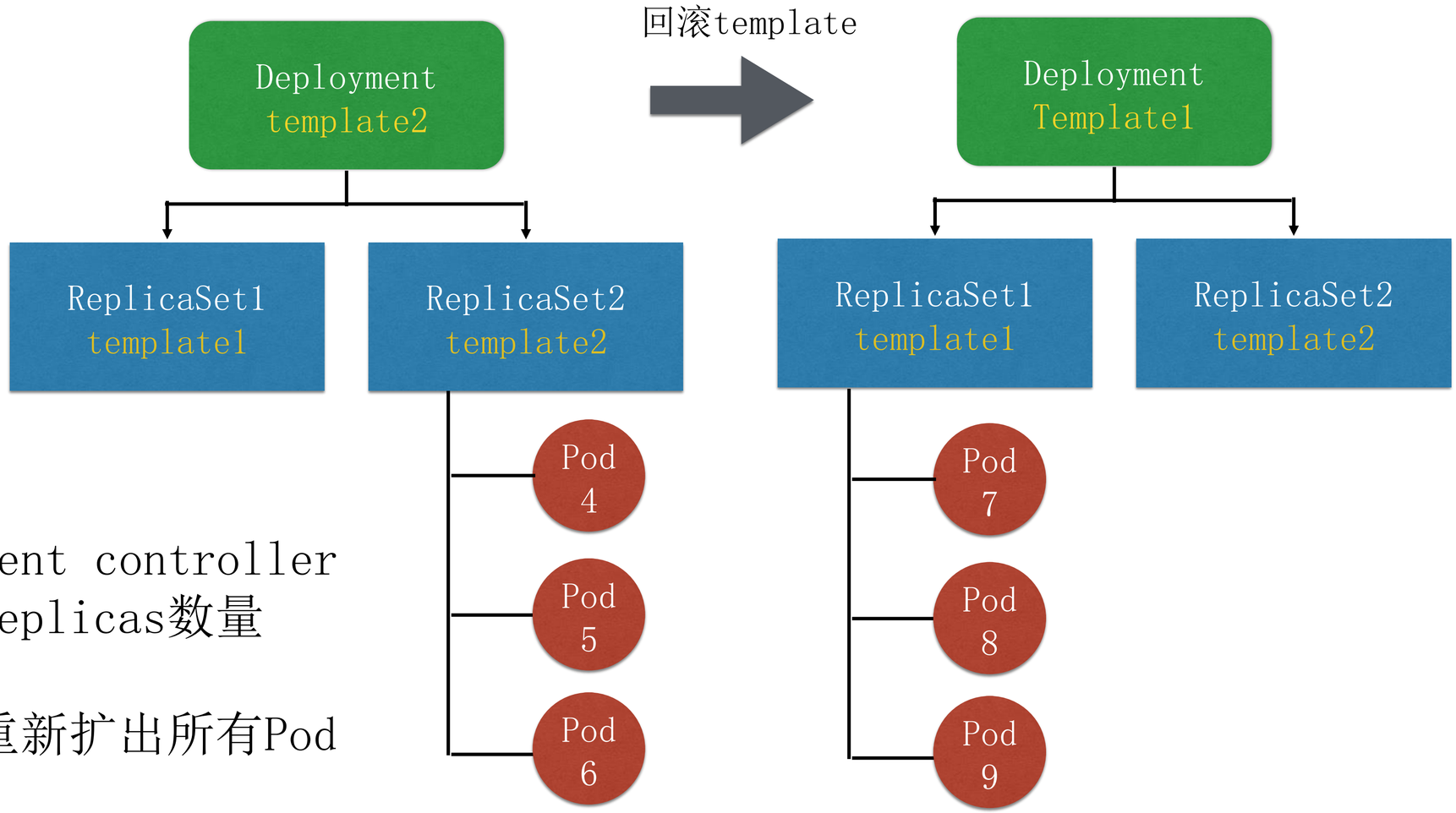
修改Deployment replicas之后，controller会把replicas同步到当前版本的ReplicaSet中，由ReplicaSet执行扩容/缩容



发布模拟


Deployment
template1
ReplicaSet1
template1
Pod
1
Pod
2
Pod
3
修改template
Deployment
template2
ReplicaSet1
template1
ReplicaSet2
template2
Pod
1
Pod
2
Pod
3
controller逐步修改两个
ReplicaSet的replicas
Deployment
template2
ReplicaSet1
template1
ReplicaSet2
template2
Pod
4
Pod
5
Pod
6

回滚模拟



回滚的过程，其实是Deployment controller重新调整下属ReplicaSet的replicas数量

最终使旧版本的ReplicaSet重新扩出所有Pod

spec字段解析

MinReadySeconds:
判断Pod available的最小ready时间

revisionHistoryLimit:
保留历史revision(ReplicaSet)的数量，默认值为10

paused:
标识Deployment只做数量维持、不做新的发布

progressDeadlineSeconds:
判断Deployment status condition为failed的最大时间

```
// Minimum number of seconds for which a newly created pod should be ready
// without any of its container crashing, for it to be considered available.
// Defaults to 0 (pod will be considered available as soon as it is ready)
// +optional
MinReadySeconds int32 `json:"minReadySeconds,omitempty" protobuf:"varint,5,opt,name=minReadySecon

// The number of old ReplicaSets to retain to allow rollback.
// This is a pointer to distinguish between explicit zero and not specified.
// Defaults to 10.
// +optional
RevisionHistoryLimit *int32 `json:"revisionHistoryLimit,omitempty" protobuf:"varint,6,opt,name=re

// Indicates that the deployment is paused.
// +optional
Paused bool `json:"paused,omitempty" protobuf:"varint,7,opt,name=paused"`

// The maximum time in seconds for a deployment to make progress before it
// is considered to be failed. The deployment controller will continue to
// process failed deployments and a condition with a ProgressDeadlineExceeded
// reason will be surfaced in the deployment status. Note that progress will
// not be estimated during the time a deployment is paused. Defaults to 600s.
ProgressDeadlineSeconds *int32 `json:"progressDeadlineSeconds,omitempty" protobuf:"varint,9,opt,n
```

升级策略字段解析

MaxUnavailable:
滚动过程中最多有
多少个Pod不可用

MaxSurge:
滚动过程中最多存在
多少个Pod超过期望
replicas数量

```go
// Spec to control the desired behavior of rolling update.
type RollingUpdateDeployment struct {
    // The maximum number of pods that can be unavailable during the update.
    // Value can be an absolute number (ex: 5) or a percentage of desired pods (ex: 10%).
    // Absolute number is calculated from percentage by rounding down.
    // This can not be 0 if MaxSurge is 0.
    // Defaults to 25%.
    // Example: when this is set to 30%, the old ReplicaSet can be scaled down to 70% of desired pods
    // immediately when the rolling update starts. Once new pods are ready, old ReplicaSet
    // can be scaled down further, followed by scaling up the new ReplicaSet, ensuring
    // that the total number of pods available at all times during the update is at
    // least 70% of desired pods.
    // +optional
    MaxUnavailable *intstr.IntOrString `json:"maxUnavailable,omitempty" protobuf:"bytes,1,opt,name=maxUnavailable"`

    // The maximum number of pods that can be scheduled above the desired number of
    // pods.
    // Value can be an absolute number (ex: 5) or a percentage of desired pods (ex: 10%).
    // This can not be 0 if MaxUnavailable is 0.
    // Absolute number is calculated from percentage by rounding up.
    // Defaults to 25%.
    // Example: when this is set to 30%, the new ReplicaSet can be scaled up immediately when
    // the rolling update starts, such that the total number of old and new pods do not exceed
    // 130% of desired pods. Once old pods have been killed,
    // new ReplicaSet can be scaled up further, ensuring that total number of pods running
    // at any time during the update is at most 130% of desired pods.
    // +optional
    MaxSurge *intstr.IntOrString `json:"maxSurge,omitempty" protobuf:"bytes,2,opt,name=maxSurge"`
}
```

谢谢观看

THANK YOU

关注“阿里巴巴云原生”公众号
获取第一手技术资料