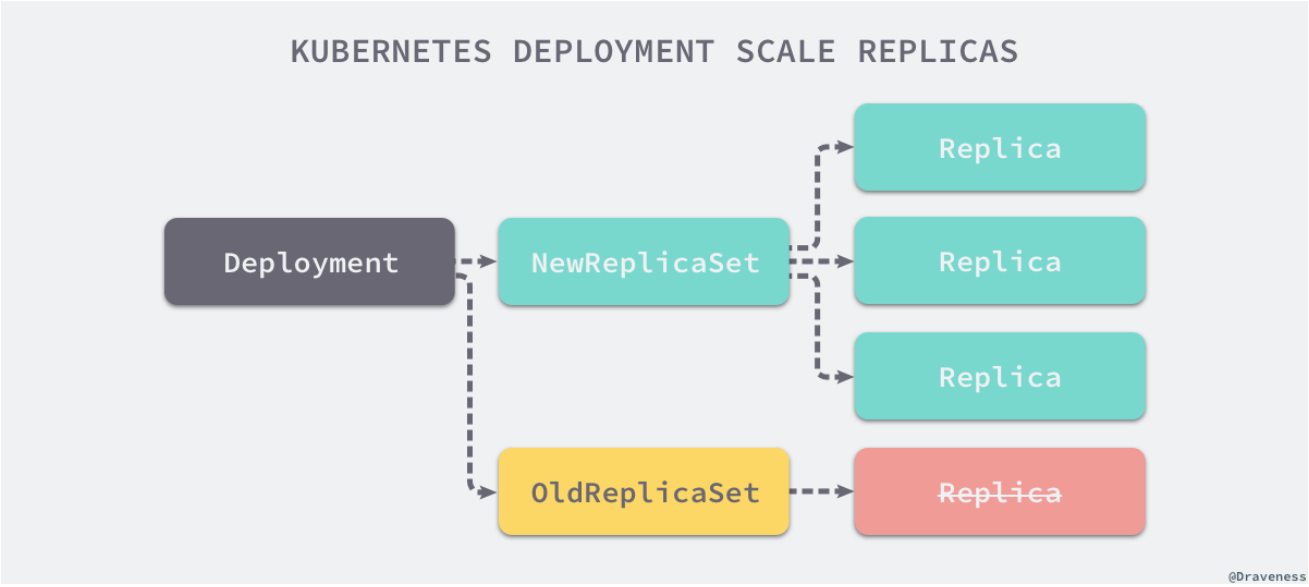
RS 与 RC 与 Deployment 关联

RC （ReplicationController ）主要的作用就是用来确保容器应用的副本数始终保持在用户定义的副本数 。即如果有容器异常退出，会自动创建新的Pod来替代；而如果异常多出来的容器也会自动回收

Kubernetes 官方建议使用 RS（ReplicaSet ） 替代 RC （ReplicationController ） 进行部署，RS 跟 RC 没有本质的不同，只是名字不一样，并且 RS 支持集合式的 selector

```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 80
```

RS 与 Deployment 的关联



Deployment

Deployment 为 Pod 和 ReplicaSet 提供了一个声明式定义(declarative)方法，用来替代以前的 ReplicationController 来方便的管理应用。典型的应用场景包括：

- **定义Deployment来创建Pod和ReplicaSet**
- **滚动升级和回滚应用**
- **扩容和缩容**
- **暂停和继续Deployment**

Ⅰ、部署一个简单的 Nginx 应用

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

```
kubectl create -f https://kubernetes.io/docs/user-guide/nginx-deployment.yaml --record
## --record参数可以记录命令，我们可以很方便的查看每次 revision 的变化
```

Ⅱ、扩容

```
kubectl scale deployment nginx-deployment --replicas 10
```

Ⅲ、如果集群支持 horizontal pod autoscaling 的话，还可以为Deployment设置自动扩展

```
kubectl autoscale deployment nginx-deployment --min=10 --max=15 --cpu-percent=80
```

Ⅳ、更新镜像也比较简单

```
kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
```

Ⅴ、回滚

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

更新 Deployment

假如我们现在想要让 nginx pod 使用`nginx:1.9.1`的镜像来代替原来的`nginx:1.7.9`的镜像

```
$ kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
deployment "nginx-deployment" image updated
```

可以使用`edit`命令来编辑 Deployment

```
$ kubectl edit deployment/nginx-deployment
deployment "nginx-deployment" edited
```

查看 rollout 的状态

```
$ kubectl rollout status deployment/nginx-deployment
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
deployment "nginx-deployment" successfully rolled out
```

查看历史 RS

```
$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-1564180365   3         3         0       6s
nginx-deployment-2035384211   0         0         0       36s
```

Deployment 更新策略

Deployment 可以保证在升级时只有一定数量的 Pod 是 down 的。默认的，它会确保至少有比期望的Pod数量少一个是up状态（最多一个不可用）

Deployment 同时也可以确保只创建出超过期望数量的一定数量的 Pod。默认的，它会确保最多比期望的Pod数量多一个的 Pod 是 up 的（最多1个 surge ）

未来的 Kuberentes 版本中，将从1-1变成25%-25%

```
$ kubectl describe deployments
```

Rollover（多个rollout并行）

假如您创建了一个有5个 `niginx:1.7.9` replica的 Deployment，但是当还只有3个 `nginx:1.7.9` 的 replica 创建出来的时候您就开始更新含有5个 `nginx:1.9.1` replica 的 Deployment。在这种情况下，Deployment 会立即杀掉已创建的3个 `nginx:1.7.9` 的 Pod，并开始创建 `nginx:1.9.1` 的 Pod。它不会等到所有的5个 `nginx:1.7.9` 的 Pod 都创建完成后才开始改变航道

回退 Deployment

```
kubectl set image deployment/nginx-deployment nginx=nginx:1.91
kubectl rollout status deployments nginx-deployment
kubectl get pods
kubectl rollout history deployment/nginx-deployment
kubectl rollout undo deployment/nginx-deployment
kubectl rollout undo deployment/nginx-deployment --to-revision=2   ## 可以使用 --revision参数指定某个历史版本
kubectl rollout pause deployment/nginx-deployment    ## 暂停 deployment 的更新
```

您可以用 `kubectl rollout status` 命令查看 Deployment 是否完成。如果 rollout 成功完成， `kubectl rollout status` 将返回一个0值的 Exit Code

```
$ kubectl rollout status deploy/nginx
Waiting for rollout to finish: 2 of 3 updated replicas are available...
deployment "nginx" successfully rolled out
$ echo $?
0
```

清理 Policy

您可以通过设置 `.spec.revisonHistoryLimit` 项来指定 deployment 最多保留多少 revision 历史记录。默认的会保留所有的 revision；如果将该项设置为0，Deployment 就不允许回退了