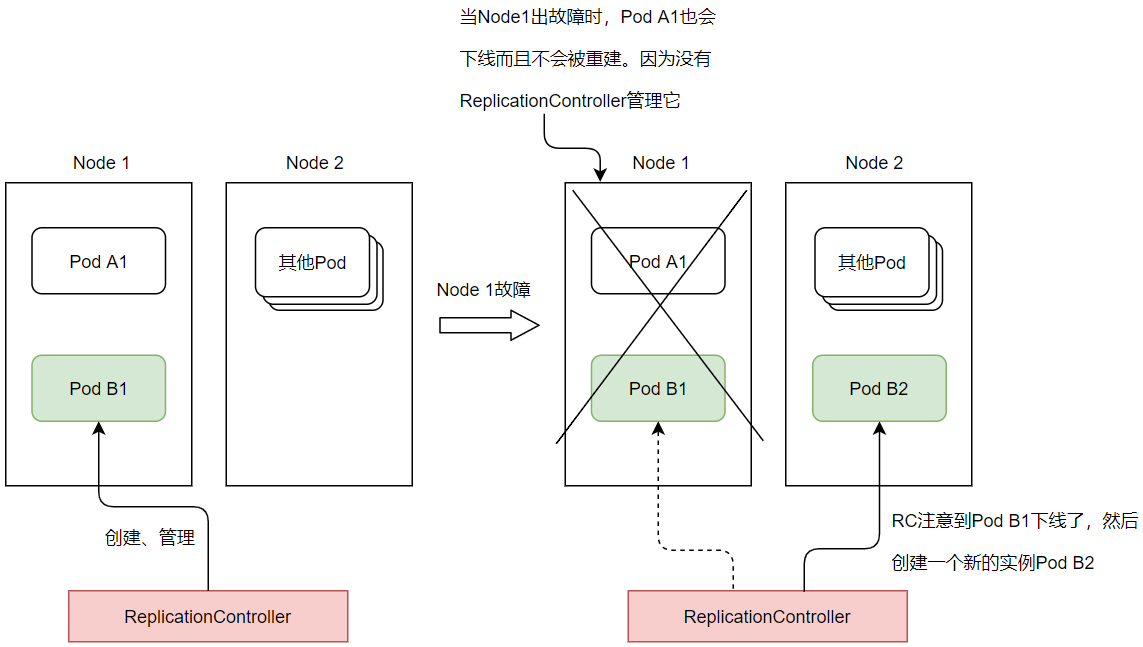
20.ReplicationController

1.ReplicationController简介

ReplicationController（简称RC）是一种Kubernetes资源，能够确保它所对应的Pod总是保持运行状态。如果Pod由于某种原因消失掉了，比如节点从集群消失了或者Pod从节点中被剔除了，那么RC会注意到丢失了Pod，然后就会创建一个新的Pod。

下图演示了当一个包含两个Pod的节点下线了会发生什么。Pod A是用户直接创建的，因此是一个非托管的Pod；Pod B是由ReplicationController管理的。当节点下线后，ReplicationController会创建一个新的Pod替换消失的Pod B，而Pod A就完全消失了，因为没有什么能恢复它。
图中的RC只管理了一个Pod，但是通常来说，RC旨在创建和管理一个Pod的多个副本。这就是ReplicationController名字的由来。



ReplicationController会不断地监控正在运行的Pod列表，并保证相应“类型”的Pod数量与期望的相符。如果正在运行的Pod数比期望的少，它就根据Pod模板创建一个新的副本；如果比期望的多，它就会移除多余的副本。

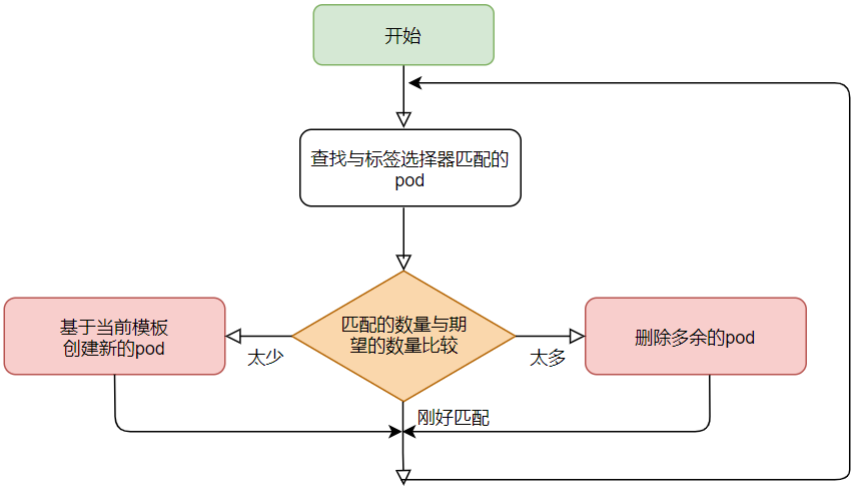
你可能会奇怪为什么正在运行的Pod数会比期望的多。这可能有几个原因：

- 有人手动创建了相同类型的Pod
- 有人更改了现有Pod的“类型”
- 有人降低了Pod的期望数量等等

我们刚刚多次使用了术语“类型（type）”，但实际上不存在这个东西。ReplicationController不会对Pod类型进行操作，而是对一组满足某个标签选择器的Pod进行操作。

容器的Reconciliation Loop机制

Reconciliation Loop就是不断使系统的当前状态向用户期望的状态移动，从而形成了一个循环的结构。ReplicationController的工作就是要确保Pod的实际数量始终与其标签选择器筛选出的数量匹配。如果不匹配，ReplicationController就会采取适当的操作来协调（reconcile）Pod的数量。如下图：



ReplicationController的三个部件

一个RC有三个关键部件：

- label selector（标签选择器），用于确定RC作用域下有哪些Pod
- replica count（副本数量），用于指定期望运行的pod数量
- pod template（pod模板），用于创建新的pod副本

ReplicationController的这三个部件都可以随时修改，但只有replica count的变更才会影响现有的Pod。

更改标签选择器或者Pod模板的影响

更改标签选择器或者pod模板对现有的pod没有影响。更改标签选择器会使现有的pod从ReplicationController的范围内脱离，因此ReplicationController不再关注它们。ReplicationController创建了pod后也不会关注pod的实际内容（容器镜像、环境变量等等）。因此pod模板只会影响ReplicationController创建的新pod。

ReplicationController的优点

ReplicationController是一个简单的概念，但是却提供了如下这些强大的功能：

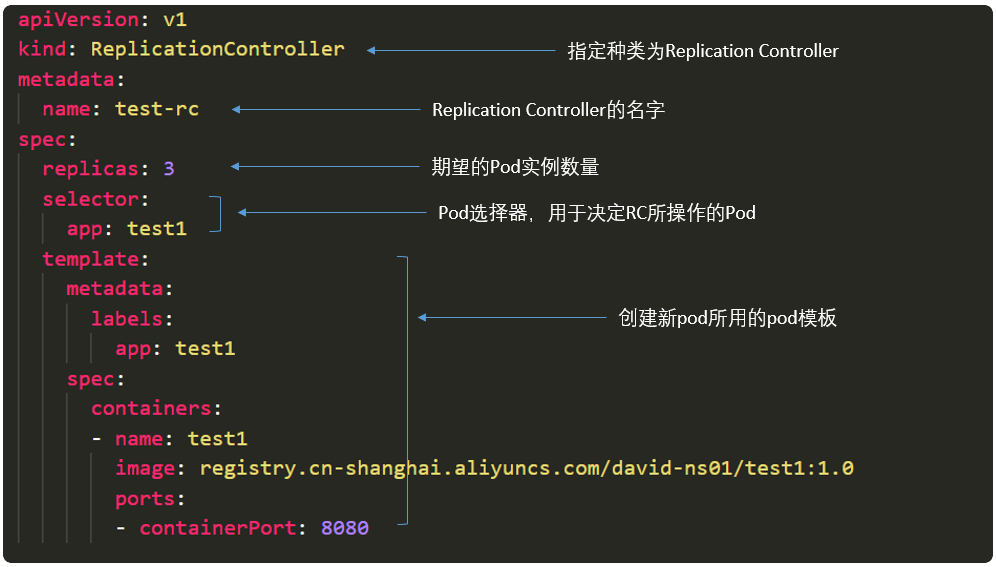
- 当现有的pod下线时，通过启动一个新pod来确保有一个pod（或者多个pod副本）一直保持运行状态
- 当集群节点出故障时，它会为该故障节点上曾经运行的pod（由ReplicationController控制的pod）创建替代副本
- 它能轻松实现pod的手动和自动水平伸缩

> 一个pod实例绝不会重新放置到另外的节点，而是由ReplicationController创建一个与被替代的Pod毫不相干的全新的pod实例

2.创建ReplicationController

同创建Pod一样，我们也可以基于JSON和YAML格式文件来创建ReplicationController。
创建一个名为test-rc.yaml的文件，内容如下：

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: test-rc
spec:
  replicas: 3
  selector:
    app: test1
  template:
    metadata:
      labels:
        app: test1
    spec:
      containers:
      - name: test1
        image: registry.cn-shanghai.aliyuncs.com/david-ns01/test1:1.0
        ports:
        - containerPort: 8080
```



当我们将这个文件发送给API Server时，Kubernetes会创建一个新的名为test-rc的ReplicationController，它会确保始终有三个pod实例与标签选择器app=test1匹配。当没有pod数量不够时，就会基于指定的pod模板创建新的pod。从上面的YAML文件可以看到，pod模板的内容与我们之前创建的pod定义几乎一样。

模板中的pod标签显然必须与ReplicationController中标签选择器匹配，否作RC就会无休止地创建新pod。为了防止这种情况发生，API Server会验证RC定义信息，如果配置有问题就不会创建。
为了避免犯错以及保持YAML文件简洁，我们可以不指定标签选择器，而是通过pod模板中的标签让其自动配置。
现在我们来创建一个ReplicationController：
kubectl create –f test–rc.yaml

```
[david@dhr-demo ~]$ kubectl get pods
No resources found in default namespace.
[david@dhr-demo ~]$ kubectl create -f test-rc.yaml
replicationcontroller/test-rc created
[david@dhr-demo ~]$ kubectl get pods
NAME            READY   STATUS              RESTARTS   AGE
test-rc-cddsz   0/1     ContainerCreating   0          6s
test-rc-ph4lb   0/1     ContainerCreating   0          6s
test-rc-v4xsq   0/1     ContainerCreating   0          6s
[david@dhr-demo ~]$ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
test-rc-cddsz   1/1     Running   0          23s
test-rc-ph4lb   1/1     Running   0          23s
test-rc-v4xsq   1/1     Running   0          23s
[david@dhr-demo ~]$
```

ReplicationController一创建好就会开始执行工作，所在在上图中我们看到ReplicationController自动为我们创建了三个pod。
现在删除一个pod试试：
kubectl delete pod test–rc–cddsz

```
[david@dhr-demo ~]$ kubectl delete pod test-rc-cddsz
pod "test-rc-cddsz" deleted
[david@dhr-demo ~]$ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
test-rc-98gw2   1/1     Running   0          42s
test-rc-ph4lb   1/1     Running   0          29m
test-rc-v4xsq   1/1     Running   0          29m
[david@dhr-demo ~]$
```

可以看到ReplicationController为我们又创建了一个新的pod。

查看ReplicationController的信息

执行如下命令：
kubectl get rc

```
[david@dhr-demo ~]$ kubectl get rc
NAME      DESIRED   CURRENT   READY   AGE
test-rc   3         3         3       52m
[david@dhr-demo ~]$
```

- DESIRED：表示期望的pod数
- CURRENT：表示当前实际的pod数
- READY：表示就绪的pod数

通过kubectl describe命令，我们还可以了解到RC的其他信息：
kubectl describe rc test–rc

```
[david@dhr-demo ~]$ kubectl describe rc test-rc
Name:         test-rc
Namespace:    default
Selector:     app=test1
Labels:       app=test1
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=test1
  Containers:
   test1:
    Image:        registry.cn-shanghai.aliyuncs.com/david-ns01/test1:1.0
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age   From                    Message
  ----    ------            ----  ----                    -------
  Normal  SuccessfulCreate  32m   replication-controller  Created pod: test-rc-98gw2
[david@dhr-demo ~]$
```

可以看到当前的副本（replica）数与期望的副本数是匹配的。底部的Events区域显示的是ReplicationController所执行的操作。

虽然当一个pod被删除后，控制器立即就收到了通知，但这并不是引起控制器创建替代pod的原因。通知会触发控制器检查pod的实际数量并采取适当的操作。

pod标签的修改对ReplicationController作用域的影响

由ReplicationController创建的pod与ReplicationController本身不是绑定的。在任何时候，ReplicationController只负责管理与其标签选择器匹配的pod。修改pod的标签会使其退出或者加入ReplicationController的作用域，甚至可以从一个ReplicationController移动到另一个。

虽然pod不与某个ReplicationController绑定，但是却在metadata.ownerReference字段中引用了它，从而我们可以轻易地找到pod所属的ReplicationController。

如果我们修改了某个pod的标签以至于它不再与某个ReplicationController的标签选择器匹配，则这个pod就会变得跟其他手动创建的pod一样，不再受任何东西的管控了。如果运行这个pod的节点故障了，该pod很显然就不会被重新调度。不过当ReplicationController注意到pod丢失时，就会启动一个新的pod来替代它。

由于当前的ReplicationController管理的是带有app=test1的标签的pod，所以我们可以通过删除这个标签或者更改标签值来将这个pod从ReplicationController的作用域中移除。给这个pod添加其他标签并不会影响该ReplicationController的作用域，因为ReplicationController并不会关心pod是否还有其他标签，它只关心这个pod是否具有标签选择器中指定的所有标签。

现在让我们来证实一下给pod添加标签是否会对ReplicationController的作用域造成影响：

kubectl label pod test-rc-98gw2 type=special

然后显示所有的pod：

kubectl get po --show-labels

```
[david@dhr-demo root]$ kubectl get po --show-labels
NAME            READY   STATUS    RESTARTS   AGE   LABELS
test-rc-98gw2   1/1     Running   0          20h   app=test1,type=special
test-rc-ph4lb   1/1     Running   0          20h   app=test1
test-rc-v4xsq   1/1     Running   0          20h   app=test1
```

可以看到，除了有一个pod多了一个type=special标签以外，列表中的三个pod跟之前完全一样，说明添加标签并不会影响ReplicationController的作用域。

我们再来修改pod的标签，看对ReplicationController是否有影响。将app=test1标签修改成其他值后，pod就不再与ReplicationController中的标签选择器匹配。因此ReplicationController会启动一个新的pod使pod的总数等于3：

kubectl label po test-rc-98gw2 app=other --overwrite

```
[david@dhr-demo root]$ kubectl label po test-rc-98gw2 app=other --overwrite
pod/test-rc-98gw2 labeled
[david@dhr-demo root]$ kubectl get po --show-labels
NAME            READY   STATUS    RESTARTS   AGE   LABELS
test-rc-98gw2   1/1     Running   0          20h   app=other,type=special
test-rc-m9nx8   1/1     Running   0          4s    app=test1
test-rc-ph4lb   1/1     Running   0          21h   app=test1
test-rc-v4xsq   1/1     Running   0          21h   app=test1
```

可以看到，当前有4个pod正在运行。从AGE列可以看出test-rc-m9nx8的年龄最小，它是ReplicationController为我们自动创建的一个新的pod。test-rc-98gw2标签成功更新为app=other，并且从ReplicationController的作用域中脱离，成为了一个不受ReplicationController管控的独立的pod。当我们想对某个特定的pod做一些操作的时候，这种方式就比较有用。比如，有一个bug导致pod在运行一段时间之后出现异常，我们就可以将其脱离ReplicationController的作用域，让控制器创建一个新的替代pod，然后对这个出问题的pod进行调试或者任何操作。

修改ReplicationController的标签选择器

如果我们不修改pod的标签，而是修改ReplicationController的标签选择器，那么原先受这个ReplicationController管控的所有pod都会脱离它的作用域，从而导致ReplicationController创建三个新的pod。

Kubernetes确实允许我们修改ReplicationController的标签选择器，但这不适用于其他管控pod的资源（后面章节中我们会讲到）。我们绝不应该更改ReplicationController的标签选择器，通常的做法是更改pod模板。

3.更改pod模板

我们可以随时修改ReplicationController的pod模板。更改pod模板只会影响后面创建的pod，不会影响已经存在的pod。

如果要修改旧的pod，就需要删除它们，并让ReplicationController基于新的模板创建新的pod来替换旧有的pod。

现在我们来编辑之前创建的ReplicationController的pod模板，为其添加一个标签：

kubectl edit rc test-rc

执行上面的命令后，我们就可以编辑ReplicationController的YAML定义文件。找到pod的template区段，在metadata下面添加一个额外的标签location=A1:

```
apiVersion: v1
kind: ReplicationController
metadata:
  creationTimestamp: "2020-11-23T14:47:14Z"
  generation: 1
  labels:
    app: test1
  name: test-rc
  namespace: default
  resourceVersion: "1037801"
  selfLink: /api/v1/namespaces/default/replicationcontrollers/test-rc
  uid: 5b8cc80b-3496-44a0-ad7e-08dcba474052
spec:
  replicas: 3
  selector:
    app: test1
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: test1
        location: A1
    spec:
      containers:
      - image: registry.cn-shanghai.aliyuncs.com/david-ns01/test1:1.0
        imagePullPolicy: IfNotPresent
        name: test1
        ports:
        - containerPort: 8080
```

保存后会显示：

```
[david@dhr-demo root]$ kubectl edit rc test-rc
replicationcontroller/test-rc edited
[david@dhr-demo root]$
```

kubectl get pod --show-labels

```
[david@dhr-demo root]$ kubectl get pod --show-labels
NAME            READY   STATUS    RESTARTS   AGE    LABELS
test-rc-m9nx8   1/1     Running   0          171m   app=test1
test-rc-ph4lb   1/1     Running   0          24h    app=test1
test-rc-v4xsq   1/1     Running   0          24h    app=test1
```

可以看到pod的标签还未变化。但是如果我们删除一个pod，ReplicationController就会自动创建一个新的pod，并且多了一个locaton=A1标签：

kubectl delete pod test-rc-m9nx8

```
[david@dhr-demo root]$ kubectl get pod --show-labels
NAME            READY   STATUS    RESTARTS   AGE   LABELS
test-rc-8w759   1/1     Running   0          43s   app=test1,location=A1
test-rc-ph4lb   1/1     Running   0          24h   app=test1
test-rc-v4xsq   1/1     Running   0          24h   app=test1
[david@dhr-demo root]$
```

我们也可以通过类似上面的方式修改pod模板中的容器镜像，然后删除已有的pod，基于修改后的pod模板来升级pod。不过还有更好的方式，后面的学习中再介绍。

4.水平缩放pod

我们已经知道了ReplicationController是如何确保特定数量的pod实例始终保持运行状态。原理非常简单，只需要更改副本的期望数即可。同样，放大和缩小pod的数量规模就和在ReplicationController资源中修改replicas字段的值一样简单。更改之后，如果ReplicationController发现已经存在太多的pod，就会进行缩容操作，删除一部分pod；如果发现pod数不够，就会进行扩容操作，新增一些pod。

当前，我们的ReplicationController保证有三个pod实例在运行。现在将其扩容到5个pod，可以通过如下方式来实现：kubectl scale rc test-rc --replicas=5

```
[david@dhr-demo root]$ kubectl scale rc test-rc --replicas=5
replicationcontroller/test-rc scaled
[david@dhr-demo root]$ kubectl get rc
NAME      DESIRED   CURRENT   READY   AGE
test-rc   5         5         3       25h
[david@dhr-demo root]$ kubectl get rc
NAME      DESIRED   CURRENT   READY   AGE
test-rc   5         5         5       25h
[david@dhr-demo root]$ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
test-rc-8fwp6   1/1     Running   0          18s
test-rc-8w759   1/1     Running   0          65m
test-rc-mnwxt   1/1     Running   0          18s
test-rc-ph4lb   1/1     Running   0          25h
test-rc-v4xsq   1/1     Running   0          25h
[david@dhr-demo root]$
```

还可以编辑ReplicationController的定义：

kubectl edit rc test-rc

```
[david@dhr-demo root]$ kubectl edit rc test-rc

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this fi
# reopened with the relevant failures.
#
apiVersion: v1
kind: ReplicationController
metadata:
  creationTimestamp: "2020-11-23T14:47:14Z"
  generation: 3
  labels:
    app: test1
  name: test-rc
  namespace: default
  resourceVersion: "1047917"
  selfLink: /api/v1/namespaces/default/replicationcontrollers/test-rc
  uid: 5b8cc80b-3496-44a0-ad7e-08dcba474052
spec:
  replicas: 5
  selector:
    app: test1
  template:
```

缩容到3个pod：

kubectl scale rc test-rc --replicas=3

```
[david@dhr-demo root]$ kubectl scale rc test-rc --replicas=3
replicationcontroller/test-rc scaled
[david@dhr-demo root]$ kubectl get rc
NAME      DESIRED   CURRENT   READY   AGE
test-rc   3         3         3       25h
[david@dhr-demo root]$ kubectl get pods
NAME            READY   STATUS        RESTARTS   AGE
test-rc-8fwp6   1/1     Terminating   0          6m27s
test-rc-8w759   1/1     Running       0          71m
test-rc-mnwxt   1/1     Terminating   0          6m27s
test-rc-ph4lb   1/1     Running       0          25h
test-rc-v4xsq   1/1     Running       0          25h
[david@dhr-demo root]$ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
test-rc-8w759   1/1     Running   0          72m
test-rc-ph4lb   1/1     Running   0          25h
test-rc-v4xsq   1/1     Running   0          25h
[david@dhr-demo root]$
```

kubectl scacle命令实际上是修改ReplicationController定义信息中的spec.replicas字段，与通过kubectl edit的效果一样。

5.删除ReplicationController

当通过kubectl delete删除ReplicationController的时候，相应的pod也会被删除。但是由于RC创建的这些pod并不是ReplicationController的组成部分，只是受这个ReplicationController管理，所以我们可以只删除ReplicationController并保持pod运行。

如果我们最开始的时候有一组受ReplicationController管理的pod，后来决定用ReplicaSet来替代ReplicationController的话，这就很有用了。可以在不影响pod的情况下来执行这个操作，并且在替换ReplicationController的时候保持pod不中断地运行。

当使用kubectl delete命令来删除ReplicationController的时候，我们可以通过指定--cascade=false选项，表示不要级联删除ReplicationController管理的pod：

kubectl delete rc test-rc --cascade=false

```
[david@dhr-demo root]$ kubectl delete rc test-rc --cascade=false
replicationcontroller "test-rc" deleted
[david@dhr-demo root]$
```

可以看到ReplicationController管理的这三个pod仍在运行：

```
[david@dhr-demo root]$ kubectl get po
NAME            READY   STATUS    RESTARTS   AGE
test-rc-8w759   1/1     Running   0          11h
test-rc-ph4lb   1/1     Running   0          35h
test-rc-v4xsq   1/1     Running   0          35h
[david@dhr-demo root]$
```

这三个独立的pod不再受ReplicationController管理，但是我们可以创建一个新的ReplicationController，只要它具有合适的标签选择器，这些pod就能再次受ReplicationController的管理。