Kubernetes Deployment 控制器

前言：

课程名称：Kubernetes Deployment 控制器

实验环境：

本章节 Kubernetes 集群环境如下：

角色	IP	主机名	组件	硬件
控制节点	192.168.128.11	k8s-master01	apiserver controller-manager scheduler etcd containerd	CPU：4vCPU 硬盘：100G 内存：4GB 开启虚拟化
工作节点	192.168.128.21	k8s-node01	kubelet kube-proxy containerd calico coredns	CPU：6vCPU 硬盘：100G 内存：8GB 开启虚拟化
工作节点	192.168.128.22	k8s-node02	kubelet kube-proxy containerd calico coredns	CPU：6vCPU 硬盘：100G 内存：8GB 开启虚拟化

张岩峰老师微信，加我微信，邀请你加入 VIP 交流答疑群：

微信号：ZhangYanFeng0429

二维码：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

1、初识 Deployment 控制器

1.1、什么是 Deployment？

Deployment 是 Kubernetes 中控制 Pod 的一种资源对象，它可以提供应用的声明式更新和回滚策略，使得应用的部署变得更加简单和自动化。

Deployment 控制器的工作方式如下：

1、定义 Deployment

Deployment 通过使用 Pod 模板来定义应用的运行方式，同时也可以定义多个 Pod 副本的数量、容器的镜像版本、Pod 的标签、容器的环境变量等。Deployment 还为这些 Pod 副本配置了许多其他的控制选项，例如 Pod 终止的创建策略、滚动升级和回滚策略等。

2、创建和更新 Pod

Deployment 控制器使用模板定义来启动 Pod 副本，认为这些副本组成了应用的一个部署。当副本数量不足时，控制器会启动新的 Pod 实例并与已有实例形成一个集合。同样，当需要升级应用时，Deployment 控制器会创建新的 Pod 副本集，替换原先的 Pod 集合，从而让新版本的实例上线并控制新旧版本 Pod 的滚动升级和回滚策略。

3、维护应用状态

Deployment 控制器会检查创建的 Pod 的运行状态和可用性，并根据应用监控功能检查这些 Pod 是否工作正常。如果 Pod 出现故障，控制器会自动更新运行

Pod 的集合。

4、提供更新和回滚策略

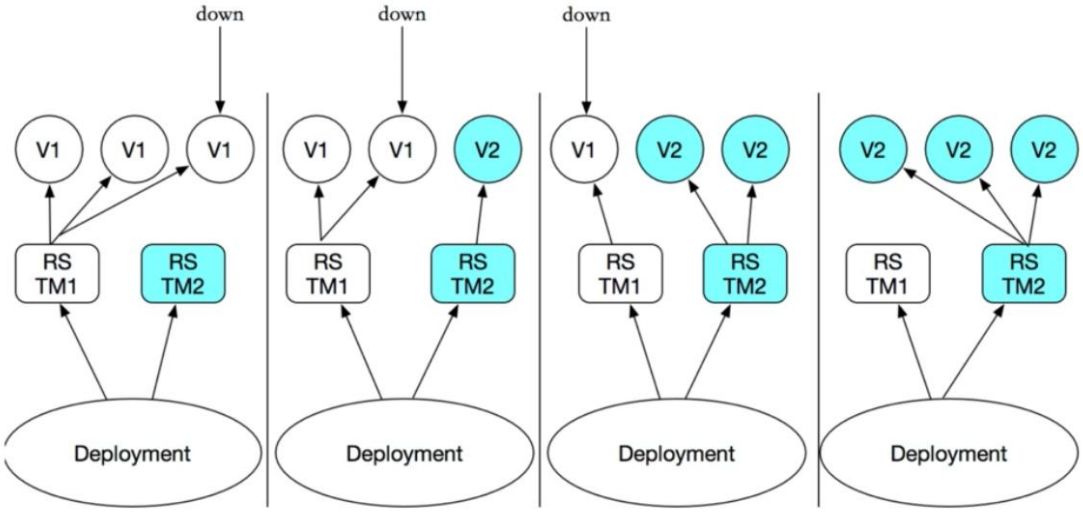
Deployment 控制器提供了自动化的更新和回滚策略，能够更好地满足应用开发和运维的需求。可以使用更新策略将 Pod 副本逐步升级到新版本，并能够取消升级、回滚到之前的版本或暂停升级，从而降低整个应用因升级操作而造成的可用性损失。

综上所述，Deployment 控制器是 Kubernetes 中重要的控制器之一，能够提供应用的自动化部署、升级和回滚功能。通过 Deployment 的使用，您可以管理和维护应用程序的运行状态，增强应用的稳定性、可用性以及可维护性。

1.2、Deployment 是如何管理 ReplicaSet 的？

当你创建一个 Deployment 资源对象时，它基于其定义的 Pod 模板创建一个对应的 ReplicaSet 控制器，因此可以把 Deployment 视为 ReplicaSet 的一个高级别封装，提供更高级别的 Pod 缩放、滚动更新等控制。

Deployment 通过使用 ReplicaSet 控制器来维护一组目标 Pod 副本的数量。当你更新 Deployment 资源对象中的 Pod 模板定义时，Deployment 控制器会使用新的 Pod 模板创建一个新的 ReplicaSet，然后逐渐将原先的 ReplicaSet 中的 Pod 副本替换为新 ReplicaSet 中的 Pod 副本，以实现滚动升级或回滚功能，并在过度期间自动确保目标 Pod 副本数量不过多或过少。如下图：



rs TM1 控制三个 pod，删除一个 pod，在 rs TM2 上重新建立一个 pod，依次类推，直到全部都是由 rs TM2 控制，如果 rs TM2 有问题，还可以回滚，Deployment 是建构在 rs 之上的，多个 rs 组成一个 Deployment，但是只有一个 rs 处于活跃状态。

通过这种方式，Deployment 能够实现自动化、流程化的部署、升级和回滚，

提供简单而灵活的管理方式，同时也保证了 Pod 的高可用性和稳定性。除此之外，Deployment 还能够应对突发事件将如故障恢复，能够自动扩展和缩减 Pod 副本数量，从而在更大范围应用场景下提供给应用足够的处理能力。

1.3、Deployment 更新节奏和更新逻辑

什么叫做更新节奏和更新逻辑呢？

比如说 Deployment 控制 5 个 pod 副本，pod 的期望值是 5 个，但是升级的时候需要额外多几个 pod，那我们控制器可以控制在 5 个 pod 副本之外还能再增加几个 pod 副本。比方说能多一个，但是不能少，那么升级的时候就是先增加一个，再删除一个，增加一个删除一个，始终保持 pod 副本数是 5 个。

还有一种情况，最多允许多一个，最少允许少一个，也就是最多 6 个，最少 4 个，第一次加一个，删除两个，第二次加两个，删除两个，依次类推，可以自己控制更新方式，这种滚动更新需要加 readinessProbe 和 livenessProbe 探测，确保 pod 中容器里的应用都正常启动了才删除之前的 pod。

2、Deployment 定义详解

Deployment 资源可以通过如下命令查看相关语法：

```
[root@k8s-master01 ~]# kubectl explain deployment
```

- Deployment 资源说明

属性名称	取值类型	取值说明
apiVersion	<string>	Api 版本
kind	<string>	资源类型
metadata	<Object>	元数据
metadata.name	String	控制器的名称
metadata.namespace	String	控制器所属的命名空间，默认值为 default
metadata.labels[]	List	自定义标签列表
metadata.annotation[]	List	自定义注解列表
spec	<Object>	规范 Deployment 所需行为的规范
spec.replicas	<integer>	Pod 的副本数
spec.revisionHistoryLimit	<integer>	保留的历史版本，默认是 10
spec.minReadySeconds	<integer>	当新的 pod 启动几秒种后，再 kill 掉旧的 pod
spec.strategy	<Object>	定义更新策略
spec.strategy.type	<string>	可选值：Recreate、RollingUpdate，默认值是 RollingUpdate。 Recreate 是重建式更新，删除一个更新一个。 RollingUpdate 滚动更新，定义滚动更新方式，也就是 pod 能多几个，少几个。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

spec.strategy.rollingUpdate	<Object>	滚动更新
spec.strategy.rollingUpdate.maxSurge	<string>	它有两种取值方式，第一种直接给定数量，第二种根据百分比。 假设原本是 5 个，最多可以超出 20%，那就允许多一个，最多可以超过 40%，那就允许多两个
spec.strategy.rollingUpdate.maxUnavailable	<string>	最多允许几个不可用。 假设有 5 个副本，最多一个不可用，就表示最少有 4 个可用。
spec.selector	<Object>	标签选择器
spec.selector.matchLabels	<map[string]string>	匹配 pod 标签，匹配 spec.template.metadata.labels 所定义的标签，必须一模一样。
spec.template	<Object>	Pod 模板
spec.template.metadata	<Object>	Pod 的元数据
spec.template.metadata.labels	<map[string]string>	定义 Pod 的标签
spec.template.spec	<Object>	等同于 Pod 的 spec

3、Pod 中添加域名解析和 DNS 配置

3.1、Pod 中添加域名解析

hostAliases 字段用于将一些主机别名映射到 Pod 内的 IP 地址，以便 Pod 内的容器可以通过这些别名来访问主机上的服务。

目标：给 pod 容器添加域名解析,实际操作容器的/etc/hosts

（1）在 master 节点创建 yaml 文件

```
[root@k8s-master01 ~]# vi nginx-hosts.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-hosts
  labels:
    app: web
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web
  template:
```

```
      metadata:
        labels:
          app: web
      spec:
        containers:
        - name: nginx-hosts
          image: nginx:latest
          imagePullPolicy: IfNotPresent
        hostAliases:
        - ip: "192.168.128.11"
          hostnames:
          - "zyf.qq.com"
          - "zyf.test.com"
```

（2）更新清单文件

```
[root@k8s-master01 ~]# kubectl apply -f nginx-hosts.yaml
deployment.apps/nginx-hosts created
```

（3）查看创建的 Pod：

```
[root@k8s-master01 ~]# kubectl get pods
[root@k8s-master01 ~]# kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
nginx-hosts-c49499bfd-jvxq7    1/1     Running   0          58s
nginx-hosts-c49499bfd-m74hq    1/1     Running   0          58s
[root@k8s-master01 ~]#
```

（4）进入 pod 查看域名解析是否添加成功

```
[root@k8s-master01 ~]# kubectl exec -it nginx-hosts-c49499bfd-jvxq7 -- cat /etc/hosts
[root@k8s-master01 ~]# kubectl exec -it nginx-hosts-c49499bfd-jvxq7 -- cat /etc/hosts
# Kubernetes-managed hosts file.
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
10.244.85.197   nginx-hosts-c49499bfd-jvxq7

# Entries added by HostAliases.
192.168.128.11  zyf.qq.com      zyf.test.com
[root@k8s-master01 ~]#
```

3.2、Pod 中 DNS 配置

目标：在 Pod 容器中添加 dns 解析，实际操作容器的/etc/resolv.conf 文件

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

（1）创建 yaml 文件

```
[root@k8s-master01 ~]# vi nginx-dns.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-dns
  labels:
    app: web
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: nginx-dns
        image: nginx:latest
        imagePullPolicy: IfNotPresent
      hostAliases:
      - ip: "192.168.128.11"
        hostnames:
        - "zyf.qq.com"
        - "zyf.test.com"
      dnsPolicy: None
      dnsConfig:
        nameservers:
        - 192.168.128.254
        - 192.168.128.1
        searches:
        - xiaozhang.svc.cluster.local
        - my.dns.search.xiaozhang
```

（2）更新清单文件

```
[root@k8s-master01 ~]# kubectl apply -f nginx-dns.yaml
deployment.apps/nginx-dns created
```

（3）查看创建的 Pod

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 ~]# kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
nginx-dns-55c976dcf6-k4zxq   1/1     Running   0          55s
nginx-dns-55c976dcf6-r5q4w   1/1     Running   0          55s
[root@k8s-master01 ~]#
```

（4）进入 pod 查看 DNS 配置是否添加成功

```
[root@k8s-master01 ~]# kubectl exec -it nginx-dns-55c976dcf6-k4zxq
-- cat /etc/resolv.conf
search xiaozhang.svc.cluster.local my.dns.search.xiaozhang
nameserver 192.168.128.254
nameserver 192.168.128.1
```

4、Deployment 控制器创建一个 web 站点

deployment 是一个三级结构，deployment 管理 replicaset，replicaset 管理 pod。

（1）在 master 节点创建 yaml 文件：

```
[root@k8s-master01 ~]# vi deploy-demo.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 2
  selector:
   matchLabels:
    app: myapp
    version: v1
  template:
   metadata:
    labels:
     app: myapp
     version: v1
   spec:
    containers:
    - name: myapp
      image: janakiramm/myapp:v1
      imagePullPolicy: IfNotPresent
      ports:
      - containerPort: 80
```

对上面的 yaml 文件说明：

这段代码是一个 Kubernetes 的 Deployment 对象，用于创建和管理 Pod 的副本集。

其中，apiVersion 字段定义使用的 Kubernetes API 版本；kind 字段定义对象的类型为 Deployment；metadata 字段中的 name 字段值定义了 Deployment 的名称。

在 spec 字段中，replicas 字段表示创建的 Pod 的副本数量为 2；selector 字段是一个标签选择器，它用于选择管理的 Pod 的标签；template 字段定义了要创建的 Pod 的模板，其中 metadata 字段定义了 Pod 的元数据（如标签），spec 字段定义了 Pod 的规范，包括它要包含的容器、启动策略等。

在这个模板中，我们定义了一个名为 myapp 的容器，image 字段指定了使用的镜像，并且通过指定 imagePullPolicy 字段，告诉 Kubernetes 在镜像不存在时是否尝试拉取镜像；ports 字段指定了容器要监听的端口。

整个 Deployment 的作用是创建两个包含 myapp 容器的 Pod，这些 Pod 具有 app=myapp 和 version=v1 的标签，并且在容器中运行的镜像为 janakiramm/myapp:v1。

（2）更新清单文件

```
[root@k8s-master01 ~]# kubectl apply -f deploy-demo.yaml
deployment.apps/myapp created
```

（3）查看 deploy 状态：

```
[root@k8s-master01 ~]# kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   2/2     2            2           4m9s
```

对上面的输出结果说明：

NAME：列出名称空间中 deployment 的名称。
READY：显示 deployment 有多少副本数。它遵循 ready/desired 的模式。
UP-TO-DATE：显示已更新到所需状态的副本数。
AVAILABLE：显示你的可以使用多少个应用程序副本。
AGE：显示应用程序已运行的时间。

（4）查看 rs

```
[root@k8s-master01 ~]# kubectl get rs
NAME               DESIRED   CURRENT   READY   AGE
myapp-7b55fffbb5   2         2         2       5m4s
```

创建 deploy 的时候也会创建一个 rs（replicaset），7b55fffbb5 这个随机数字是我们引用 pod 的模板 template 的名字的 hash 值。

NAME：列出名称空间中 ReplicaSet 资源
DESIRED：显示应用程序的所需副本数，这些副本数是在创建时定义的。这是所需的状态。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    CURRENT：显示当前正在运行多少个副本。
    READY：显示你的用户可以使用多少个应用程序副本。
    AGE：显示应用程序已运行的时间。

请注意：
    ReplicaSet 的名称始终设置为[DEPLOYMENT-NAME]-[RANDOM-STRING]。
RANDOM-STRING 是随机生成的。
```

（5）查看创建的 Pod：

```
    [root@k8s-master01 ~]# kubectl get pods
    NAME                     READY   STATUS    RESTARTS   AGE
    myapp-7b55fffbb5-jcncs   1/1     Running   0          6m3s
    myapp-7b55fffbb5-zlfq6   1/1     Running   0          6m3s
```

5、Deployment 管理 pod

5.1、Deployment 实现 pod 的动态扩容

下面介绍两种方法来实现 pod 的动态扩容：

方法一：编辑 yaml 文件实现扩容（推荐）

（1）修改 yaml 文件

```
    [root@k8s-master01 ~]# vi deploy-demo.yaml
    apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: myapp
    spec:
      replicas: 3
      selector:
       matchLabels:
        app: myapp
        version: v1
      template:
       metadata:
        labels:
         app: myapp
         version: v1
       spec:
        containers:
        - name: myapp
          image: janakiramm/myapp:v1
```

```
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
```

（2）更新 Pod：

```
[root@k8s-master01 ~]# kubectl apply -f deploy-demo.yaml
deployment.apps/myapp configured
```

注意：apply 不同于 create，apply 可以执行多次。create 执行一次，再执行就会报错复。

（3）查看创建的 Pod

```
[root@k8s-master01 ~]# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-7b55fffbb5-97cc9   1/1     Running   0          18s
myapp-7b55fffbb5-jcncs   1/1     Running   0          22m
myapp-7b55fffbb5-zlfq6   1/1     Running   0          22m
```

上面可以看到 pod 副本数变成了 3 个

方法二：编辑控制器实现扩容（不推荐）

（1）查看 deploy 状态：

```
[root@k8s-master01 ~]# kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   3/3     3            3           23m
```

（2）编辑 deploy 控制器的 myapp

```
[root@k8s-master01 ~]# kubectl edit deploy myapp
```

```
spec:
  progressDeadlineSeconds: 600
  replicas: 4
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: myapp
      version: v1
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
```

编辑好之后，wq 保存会立即生效。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

（3）查看 deploy 状态

```
[root@k8s-master01 ~]# kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   4/4     4            4           24m
```

（4）查看创建的 Pod

```
[root@k8s-master01 ~]# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-7b55fffbb5-97cc9   1/1     Running   0          2m3s
myapp-7b55fffbb5-h8ql7   1/1     Running   0          14s
myapp-7b55fffbb5-jcncs   1/1     Running   0          24m
myapp-7b55fffbb5-zlfq6   1/1     Running   0          24m
```

5.2、Deployment 实现 pod 的动态缩容

下面介绍两种方法来实现 pod 的动态缩容：

方法一：编辑 yaml 文件实现缩容（推荐）

（1）修改 yaml 文件

```
[root@k8s-master01 ~]# vi deploy-demo.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 3
  selector:
   matchLabels:
    app: myapp
    version: v1
  template:
   metadata:
    labels:
     app: myapp
     version: v1
   spec:
    containers:
    - name: myapp
      image: janakiramm/myapp:v1
      imagePullPolicy: IfNotPresent
      ports:
```

```
        - containerPort: 80
```

（2）更新 Pod：

```
[root@k8s-master01 ~]# kubectl apply -f deploy-demo.yaml
deployment.apps/myapp configured
```

注意：apply 不同于 create，apply 可以执行多次。create 执行一次，再执行就会报错复。

（3）查看创建的 Pod

```
[root@k8s-master01 ~]# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-7b55fffbb5-97cc9   1/1     Running   0          18s
myapp-7b55fffbb5-jcncs   1/1     Running   0          22m
myapp-7b55fffbb5-zlfq6   1/1     Running   0          22m
```

上面可以看到 pod 副本数变成了 3 个

方法二：编辑控制器实现缩容（不推荐）

（1）查看 deploy 状态：

```
[root@k8s-master01 ~]# kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myapp   3/3     3            3           26m
```

（2）编辑 deploy 控制器的 myapp

```
[root@k8s-master01 ~]# kubectl edit deploy myapp
spec:
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: myapp
      version: v1
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: myapp
        version: v1
```

编辑好之后，wq 保存会立即生效。

（3）查看 deploy 状态

```
[root@k8s-master01 yaml]# kubectl get deploy
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
myapp-v1   2/2     2            2           46m
```

（4）查看创建的 Pod

```
[root@k8s-master01 ~]# kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
myapp-7b55fffbb5-jcncs      1/1     Running   0          27m
myapp-7b55fffbb5-zlfq6      1/1     Running   0          27m
```

5.3、Deployment 实现 pod 的滚动升级

（1）查看正在运行的 pod 页面信息

```
[root@k8s-master01 ~]# kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS   AGE   IP              NODE        NOMINATED NODE   READINESS GATES
myapp-7b55fffbb5-jcncs      1/1     Running   0          30m   10.244.85.200   k8s-node01  <none>           <none>
myapp-7b55fffbb5-zlfq6      1/1     Running   0          30m   10.244.58.202   k8s-node02  <none>           <none>
```

```
[root@k8s-master01 ~]# curl 10.244.85.200
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Sample Deployment</title>
  <style>
    body {
      color: #ffffff;
      background-color: blue;
      font-family: Arial, sans-serif;
      font-size: 14px;
    }
```

（2）多一个终端窗口执行如下：

```
[root@k8s-master01 ~]# kubectl get pods -l app=myapp -w
NAME                        READY   STATUS    RESTARTS   AGE
myapp-v1-5948b89958-tw195   1/1     Running   0          38m
myapp-v1-5948b89958-v24jt   1/1     Running   0          38m
```

（3）更改镜像版本，按如下操作：

```
[root@k8s-master01 ~]# vi deploy-demo.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 2
```

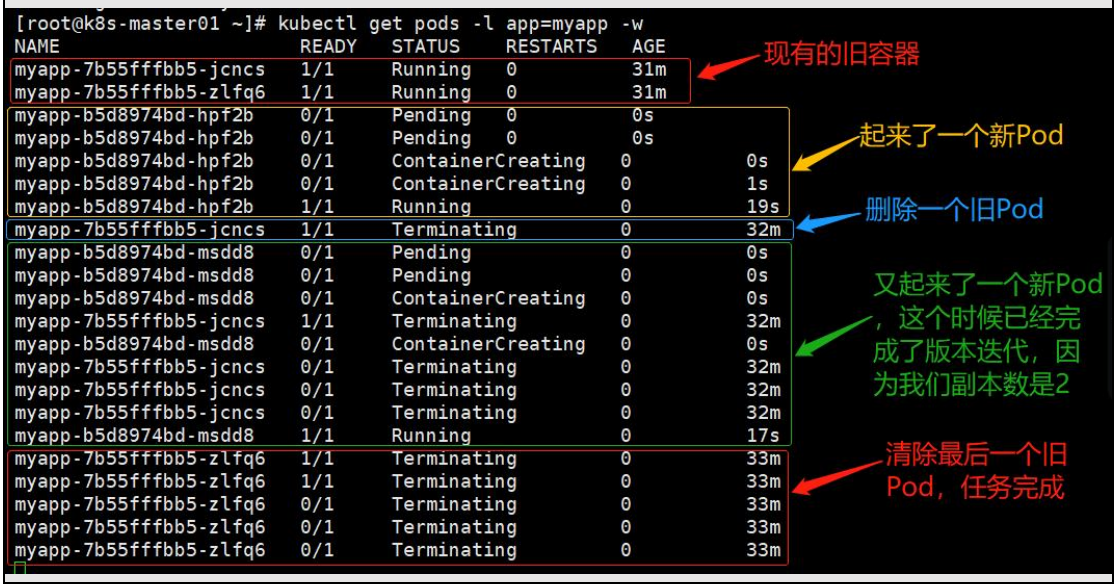
```
    selector:
     matchLabels:
      app: myapp
      version: v1
    template:
     metadata:
      labels:
       app: myapp
       version: v1
     spec:
      containers:
      - name: myapp
        image: janakiramm/myapp:v2
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
```

（4）更新迭代版本

```
[root@k8s-master01 ~]# kubectl apply -f deploy-demo.yaml
deployment.apps/myapp configured
```

（5）查看更新过程

```
[root@k8s-master01 ~]# kubectl get pods -l app=myapp -w
```



pending 表示正在进行调度，ContainerCreating 表示正在创建一个 pod，running 表示运行一个 pod，running 起来一个 pod 之后再 Terminating（停掉）一个 pod，以此类推，直到所有 pod 完成滚动升级。

（6）测试访问 Pod

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 ~]# kubectl get pods -o wide
```

```
[root@k8s-master01 ~]# kubectl get pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE     IP              NODE         NOMINATED NODE   READINESS GATES
myapp-b5d8974bd-hpf2b   1/1     Running   0          7m2s    10.244.85.202   k8s-node01   <none>           <none>
myapp-b5d8974bd-msdd8   1/1     Running   0          6m43s   10.244.58.204   k8s-node02   <none>           <none>
[root@k8s-master01 ~]#
```

```
[root@k8s-master01 ~]# curl 10.244.85.202
```

```
[root@k8s-node01 ~]# curl 172.16.85.247
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Sample Deployment</title>
  <style>
    body {
      color: #ffffff;
      background-color: green;
      font-family: Arial, sans-serif;
      font-size: 14px;
    }
```

（7）查看 rs

```
[root@k8s-master01 ~]# kubectl get rs
NAME               DESIRED   CURRENT   READY   AGE
myapp-7b55fffbb5   0         0         0       40m
myapp-b5d8974bd    2         2         2       8m
```

上面可以看到 rs 有两个，上面那个是升级之前的，已经被停掉，但是可以随时回滚。

（8）查看 myapp 这个控制器的历史版本

```
[root@k8s-master01 ~]# kubectl rollout history deployment myapp
deployment.apps/myapp
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

默认保留 10 个版本，数字从小到大，数字小的表示旧版本。

5.4、Deployment 实现 pod 的回滚

目标：研发部突然发消息说刚才上的新版本有问题，让马上回滚。现在我们对 deployment myapp 进行版本回退。

（1）查看 Deployment myapp 对象的历史版本

```
[root@k8s-master01 ~]# kubectl rollout history deployment myapp
```

```
deployment.apps/myapp
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

（2）回滚 myapp 的历史版本为 1

```
[root@k8s-master01 ~]# kubectl rollout undo deployment myapp
--to-revision=1
deployment.apps/myapp rolled back
```

（3）查看 Deployment myapp 对象的历史版本

```
[root@k8s-master01 ~]# kubectl rollout history deployment myapp
deployment.apps/myapp
REVISION  CHANGE-CAUSE
2         <none>    #<==这里是 v2 版本
3         <none>    #<==这里是 v1 版本，也是我们当前版本
```

（4）查看 rs 控制器

```
[root@k8s-master01 ~]# kubectl get rs
NAME               DESIRED   CURRENT   READY   AGE
myapp-7b55fffbb5   2         2         2       45m
myapp-b5d8974bd    0         0         0       12m
```

发现原来 myapp-b5d8974bd 管理的 pod 都到 myapp-7b55fffbb5 上面了。

（5）最后测试访问，是否版本回退

```
[root@k8s-master01 ~]# kubectl get pod -o wide
```

```
[root@k8s-master01 ~]# kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP              NODE        NOMINATED NODE   READINESS GATES
myapp-7b55fffbb5-cjbwt   1/1     Running   0          94s   10.244.58.205   k8s-node02   <none>           <none>
myapp-7b55fffbb5-hh9nz   1/1     Running   0          95s   10.244.85.203   k8s-node01   <none>           <none>
[root@k8s-master01 ~]#
```

```
[root@k8s-master01 ~]# curl 10.244.58.205
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 ~]# curl 10.244.58.205
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Sample Deployment</title>
  <style>
    body {
      color: #ffffff;
      background-color: blue;
      font-family: Arial, sans-serif;
      font-size: 14px;
    }

    h1 {
```

版本回退完毕。

5.5、自定义滚动更新策略

- 自定义更新策略说明

在 Kubernetes 中，可以通过设置 Deployment 的 spec.strategy 字段来自定义滚动更新策略。

通常，Kubernetes 默认的滚动更新策略是 RollingUpdate，它在更新新版 Pod 的同时逐步关闭旧版 Pod，这样可以确保服务在整个更新期间保持可用性。

RollingUpdate 有两个可选值：maxSurge 和 maxUnavailable 用来控制滚动更新的更新策略。

maxSurge：新增实例数，值可以是整数或百分比等。比如 10 个副本，5%的话==0.5 个，但计算按照 0 个。

maxUnavailable：可用实例最大数，值可以是整数或百分比等。比如 10 个副本，5%的话==0.5 个，但计算按照 1 个。

注意：maxSurge 和 maxUnavailable 两者不能同时为 0。

例如：

当我有 10 个副本，maxSurge 为 1，maxUnavailable 为 0。

当 maxUnavailable 为 0 时，实例最大数为 10。那么现在只有 10 个副本，无法删除。maxSurge 为 1 表示新增实例数，那么会先创建 1 个 pod，现在就有 11 个 Pod。maxUnavailable 保证有 10 个实例，就会删除一个。然后反复直到迭代完成。

建议配置：

```
maxSurge == 1
maxUnavailable == 0
```

这是我们生产环境提供给用户的默认配置。即“一上一下，先上后下”最平滑原则：

1 个新版本 pod ready（结合 readiness）后，才销毁旧版本 pod。此配置适用场景是平滑更新、保证服务平稳，但也有缺点，就是“太慢”了。

● 自定义更新策略实战

实例 1：

（1）创建 deployment

```
[root@k8s-master01 yaml]# vi deploy-demo.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deploy
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: janakiramm/myapp:v1
        ports:
        - containerPort: 80
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 50%
      maxUnavailable: 0

[root@k8s-master01 ~]# kubectl apply -f deploy-demo.yaml
deployment.apps/my-deploy created
```

当副本数为 2 时，“maxSurge: 50%”就等于 1。“maxUnavailable: 0”，那么就表示创建 1 个 pod，删除 1 个 pod，直到更新完成。

（2）更新 pod：

在更新 Pod 之前，可以在开一个窗口，使用“kubectl get pods -w”命令，动态观察 Pod 状态。

```
# 更新 pod
[root@k8s-master01 ~]# kubectl set image deployment/my-deploy
```

```
my-container=janakiramm/myapp:v2
    deployment.apps/my-deploy image updated
```

```
[root@k8s-master01 ~]# kubectl get pods -w
NAME                         READY   STATUS    RESTARTS   AGE
my-deploy-8d77b6878-6xpq9    1/1     Running   0          10s
my-deploy-8d77b6878-96bpm    1/1     Running   0          10s
my-deploy-77cffdc547-kkkt7   0/1     Pending   0          0s
my-deploy-77cffdc547-kkkt7   0/1     Pending   0          0s
my-deploy-77cffdc547-kkkt7   0/1     ContainerCreating   0          0s
my-deploy-77cffdc547-kkkt7   0/1     ContainerCreating   0          1s
my-deploy-77cffdc547-kkkt7   1/1     Running             0          1s
my-deploy-8d77b6878-6xpq9    1/1     Terminating         0          16s
my-deploy-77cffdc547-dcxpb   0/1     Pending             0          0s
my-deploy-77cffdc547-dcxpb   0/1     Pending             0          0s
my-deploy-77cffdc547-dcxpb   0/1     ContainerCreating   0          0s
my-deploy-8d77b6878-6xpq9    1/1     Terminating         0          16s
my-deploy-77cffdc547-dcxpb   0/1     ContainerCreating   0          1s
my-deploy-77cffdc547-dcxpb   1/1     Running             0          1s
my-deploy-8d77b6878-96bpm    1/1     Terminating         0          17s
my-deploy-8d77b6878-6xpq9    0/1     Terminating         0          17s
my-deploy-8d77b6878-6xpq9    0/1     Terminating         0          17s
my-deploy-8d77b6878-6xpq9    0/1     Terminating         0          17s
my-deploy-8d77b6878-96bpm    1/1     Terminating         0          17s
my-deploy-8d77b6878-96bpm    0/1     Terminating         0          18s
my-deploy-8d77b6878-96bpm    0/1     Terminating         0          18s
my-deploy-8d77b6878-96bpm    0/1     Terminating         0          18s
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。