14.什么是Label

> 上一节我们学习了如何创建一个Pod，这节我们将学习如何使用Label对Pod进行分类

从之前章节的学习到现在，我们一共创建了两个Pod，在实际应用中，我们可能会有很多运行的Pod。因此将这些Pod分类到子集很有必要。
现在微服务架构越来越流行，一个应用由多达几十个微服务组成的情况很常见。特别是有些微服务组件一般还会部署多套，从而导致系统中可能就有上百个Pod。因此我们需要一个很好的方式，能够基于任意条件对这些Pod进行分类。而且有时候我们可能更希望对一组Pod进行统一的操作，而不是单独对每个Pod单进行重复的操作。
基于上面提到的需求，Kubernetes为我们提供了标签（Label）功能。

什么是标签(Label)

标签是一个很简单但是非常强大的功能，可以对所有Kubernetes资源对象进行组合分类。它是一个附加到资源上的任意的键值对。Kubernetes在利用标签选择器（label selectors）过滤资源的时候会通过标签对资源进行甄别，具体原理就是判断资源是否具有标签选择器中指定的标签来对资源进行过滤。
一个资源可以附加多个关键字（Key）唯一的标签。通常是在创建资源的时候为其加上标签，但是我们也可以为已经存在的资源添加额外的标签，甚至可以修改已经存在的标签，而且不需要重建资源。
对于一个拥有大量Pod的系统，为每个Pod添加标签将使我们的系统看起来更加有组织秩序。

创建Pod时指定标签

现在让我们来创建一个新的yaml文件：
vim test1-manual-with-labels.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test1-manual-v2
  labels:
    creation_method: manual
    env: prod
spec:
  containers:
  - image: registry.cn-shanghai.aliyuncs.com/david-ns01/test1:1.0
    name: test1
    ports:
    - containerPort: 8080
      protocol: TCP
~
~
```

apiVersion: v1

kind: Pod

metadata:

  name: test1-manual-v2

  labels:

    creation_method: manual

    env: prod

spec:

  containers:

  - image: registry.cn-shanghai.aliyuncs.com/david-ns01/test1:1.0

    name: test1

    ports:

    - containerPort: 8080

      protocol: TCP

然后创建Pod：

kubectl create -f test1-manual-with-labels.yaml

```
[david@dhr-demo ~]$ kubectl create -f test1-manual-with-labels.yaml
pod/test1-manual-v2 created
[david@dhr-demo ~]$
```

kubectl get pods 命令默认是不会显示标签的：

```
[david@dhr-demo root]$ kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
test1            1/1     Running   0          9d
test1-manual     1/1     Running   0          2d11h
test1-manual-v2  1/1     Running   0          11h
```

我们可以通过指定--show-labels选项来显示标签：

```
[david@dhr-demo root]$ kubectl get pods --show-labels
NAME              READY   STATUS    RESTARTS   AGE     LABELS
test1             1/1     Running   0          9d      run=test1
test1-manual      1/1     Running   0          2d11h   <none>
test1-manual-v2   1/1     Running   0          11h     creation_method=manual,env=prod
[david@dhr-demo root]$
```

如果我们只想关注某些标签，可以使用-L开关，让该选项指定的标签都以一列单独展示：

kubectl get pod -L creation_method,env

```
[david@dhr-demo root]$ kubectl get pod -L creation_method,env
NAME              READY   STATUS    RESTARTS   AGE     CREATION_METHOD   ENV
test1             1/1     Running   0          9d
test1-manual      1/1     Running   0          2d11h
test1-manual-v2   1/1     Running   0          11h     manual            prod
```

修改标签

就拿之前手动创建的名为test1-manual的Pod来举例，我们可以给它添加一个标签

creation_method=manual

kubectl label po test1-manual creation_method=manual

```
[david@dhr-demo root]$ kubectl get pod -L creation_method,env
NAME              READY   STATUS    RESTARTS   AGE     CREATION_METHOD   ENV
test1             1/1     Running   0          9d
test1-manual      1/1     Running   0          2d11h   manual
test1-manual-v2   1/1     Running   0          11h     manual            prod
```

还可以对已经存在的标签进行修改：

kubectl label po test1-manual-v2 env=test --overwrite

```
[david@dhr-demo root]$ kubectl get pods --show-labels
NAME              READY   STATUS    RESTARTS   AGE     LABELS
test1             1/1     Running   0          9d      run=test1
test1-manual      1/1     Running   0          2d11h   creation_method=manual
test1-manual-v2   1/1     Running   0          11h     creation_method=manual,env=test
[david@dhr-demo root]$
```

上面我们将Pod test1-manual-v2的env=prod标签更改为env=test。

需要注意的是修改已经存在的标签需要指定--overwrite选项，否则就报下面的错：

```
[david@dhr-demo root]$ kubectl label po test1-manual-v2 env=dev
error: 'env' already has a value (test), and --overwrite is false
[david@dhr-demo root]$
```