16.什么是Annotation

添加、修改、删除注解

查看已有对象的注解

> 上一节我们学习了标签选择器以及如何通过标签选择器筛选Pod和Node。这一节我们来学习什么是注解以及如何为Pod添加注解。

除了前面讲到的标签以外，Pod和其他Kubernetes对象还可以附加注解（Annotation）。注解其实跟标签类似，也是键值对形式的结构，但是与标签筛选对象、组合对象的目的不同，注解的主要目的是给对象附加任意非标识的元数据，工具或者库之类的客户端可以检索该元数据。
注解中的元数据是可大可小、结构化或者非结构化的，可以包含标签中不允许的字符。
注解的结构也是key/value映射，如下：

```
"metadata": {
  "annotations": {
    "key1" : "value1",
    "key2" : "value2"
  }
}
```

在向Kubernetes引入新特性时，也经常会用到注释。通常情况下，新功能的alpha和beta版本不会向API对象引入任何新的字段，而是使用注解。一旦所需的API变更变得清晰并且得到Kubernetes开发人员的认可，就会引入新的字段，废弃相关的注解。
注释的一个重要的用途是为每个pod或其他API对象添加描述信息，这样每个使用集群的人都可以快速了解到每个对象的信息。比如在注解中指定对象创建人、创建时间等等。
以下是一些可以记录在注释中的信息示例：

- 由声明式配置层管理的字段。将这些字段作为注解可以将它们与客户端或服务器设置的默认值、自动生成的字段以及通过自动调整大小或自动缩放系统设置的字段区分开
- 构建、发布或者镜像信息。比如时间戳、发行版ID、git分支、PR号、镜像哈希以及注册表地址
- 指向日志、监控、分析或审计仓库的指针
- 可以用于调试目的的客户端库或者工具信息。比如姓名、版本号、以及构建信息
- 用户或工具/系统出处信息。例如来自其他生态系统组件的相关对象的URL
- 轻量级的发行工具元数据:例如，配置或检查点
- 负责人员的联系信息，或指定在何处可以获取这些信息的目录条目，如团队网站
- 从终端用户到实现的指令，以修改行为或使用非标准功能

添加、修改、删除注解

可以在创建对象的时候为其添加注解（通过在YAML文件中指定），也可以为已经存在的对象添加或者修改注解（通过kubectl annotate命令实现）。

例如，通过如下命令，我们可以为之前创建的test1-manual Pod添加一个注解：

kubectl annotate pod test1-manual test.com/annotation1="foo bar"

```
[david@dhr-demo root]$ kubectl annotate pod test1-manual test.com/annotation1="foo bar"
pod/test1-manual annotated
[david@dhr-demo root]$
```

如果要修改已有的注解，需要指定--overwrite选项：

kubectl annotate --overwrite pod test1-manual test.com/annotation1="mytest"

如果想删除已有的注解，可以在注解的key后面加上减号：

kubectl annotate pod test1-manual test.com/annotation1-

查看已有对象的注解

如果要查看pod上所有的注解，可以通过kubectl describe命令查看，例如：

kubectl describe pod test1-manual

```
[david@dhr-demo root]$ kubectl describe pod test1-manual
Name:         test1-manual
Namespace:    default
Priority:     0
Node:         minikube/172.17.0.3
Start Time:   Sat, 14 Nov 2020 12:04:45 +0800
Labels:       creation_method=manual
Annotations:  test.com/annotation1: mytest
Status:       Running
IP:           172.18.0.6
IPs:
  IP:  172.18.0.6
Containers:
```

当然我们也可以通过kubectl get命令来查看对象的注解：

kubectl get po test1-manual -o yaml

```
[david@dhr-demo root]$ kubectl get po test1-manual -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    test.com/annotation1: mytest
  creationTimestamp: "2020-11-14T04:04:45Z"
  labels:
    creation_method: manual
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:spec:
        f:containers:
          k:{"name":"test1"}:
            .: {}
```

更多关于kubectl annotate的用法，可以参考官方文档：

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#annotate