Label与Label Selector

前言

- 本章节主要讲述标签和标签选择器相关知识，包括定义、如何使用标签和标签选择器，不同类型的标签选择器等。

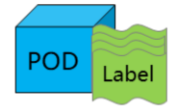
目标

- 学完本课程后，您将能够：
  - 使用标签标注Kubernetes对象
  - 使用标签选择器筛选Kubernetes对象
  - 区别不同类型的标签选择器

目录

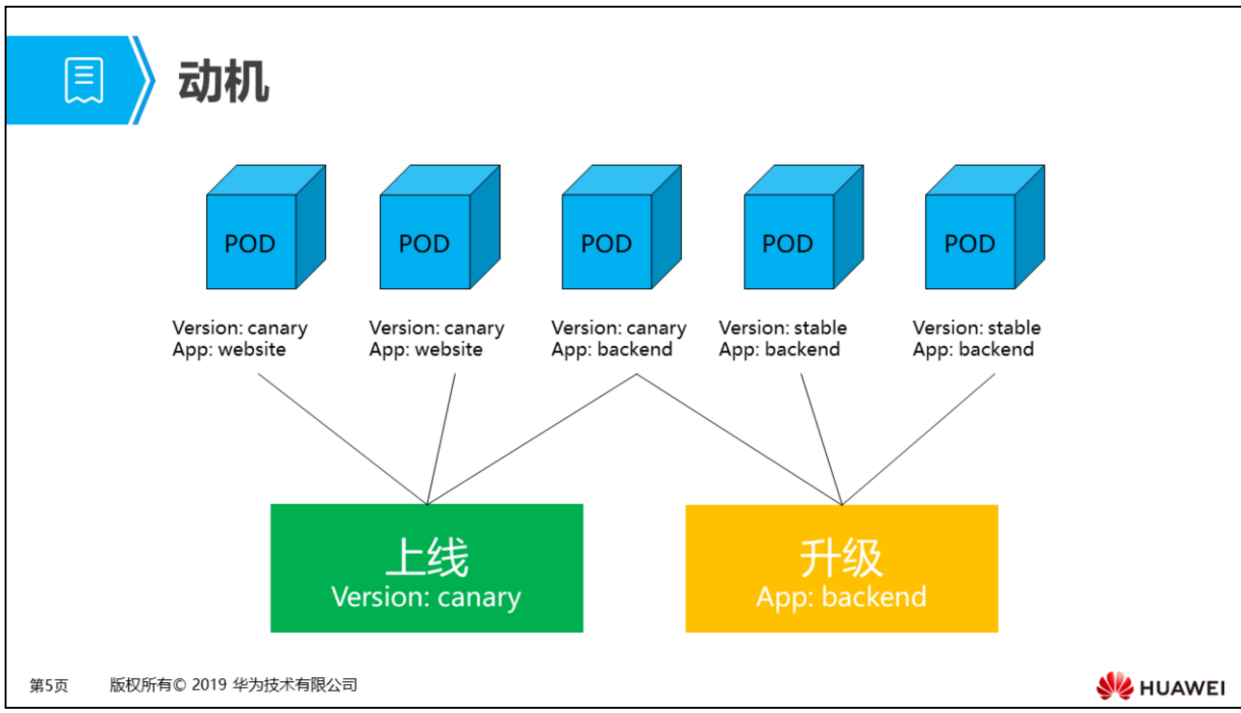
Label

- 标签（Label）是附在Kubernetes对象（如pod，deployment等）上的键值对（key-value），可以在创建时指定，也可以在创建后指定。
- Label的值本身不具备具体含义，但可以通过label来筛选对象特定的子集，便于管理。
- 每一个对象可以有多个标签。




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


- Label能够将组织架构映射到系统架构上（就像是康威定律），这样能够更便于微服务的管理。



- 上线操作时指定Version：canary可以只操作前三个pod；升级操作时指定app：backend可以只升级后三个pod

标签的语法

- 标签由一组键值对构成。
- Label key的组成：
  - Key值必须是唯一的
  - 不得超过63个字符
  - 可以使用前缀，使用/分隔。前缀必须是DNS子域，不得超过253个字符。系统中的自动化组件创建的label必须指定前缀，kubernetes.io/由kubernetes保留
  - 起始必须是字母（大小写都可以）或数字，中间可以有连字符、下划线和点
- Label value的组成：
  - 不得超过63个字符
  - 起始必须是字母（大小写都可以）或数字，中间可以有连字符、下划线和点

- 不要在label中使用大型、非标识的结构化数据，记录这样的数据应该用annotation。

创建时指定标签

- 使用labelpod.yaml文件创建一个pod。
- 在创建时指定两个label
  - app：busybox
  - version：new

```
kind: Pod
apiVersion: v1
metadata:
  name: labelpod
  labels:
    app: busybox
    version: new
spec:
  containers:
    - name: labelpod
      image: busybox
      args:
      - /bin/sh
      - -c
      - sleep 30000
```


- Label可以在yaml中指定，也可以直接使用命令为现有对象添加label。

查看Label

- 通过命令“--show-labels”可以查看指定对象的所有label

```
[root@k8s-master runfile]# kubectl get pods --show-labels
NAME       READY   STATUS    RESTARTS   AGE   LABELS
labelpod   1/1     Running   0          22m   app=busybox,version=new
```

- 同样可以使用label指令在已创建的对象上添加标签

```
root@k8s-master runfile]# kubectl label pods labelpod time=2019
pod/labelpod labeled
[root@k8s-master runfile]# kubectl get pods --show-labels
NAME       READY   STATUS    RESTARTS   AGE   LABELS
labelpod   1/1     Running   0          30m   app=busybox,time=2019,version=new
```

目录

标签选择器 (Labels Selectors)

- 标签不具备唯一性，在通常情况下，多个不同的对象有着相同的标签。
- 通过标签选择器，用户或客户端可以指定批量的对象进行操作。标签选择器也是Kubernetes的核心分组方式。
- 目前支持两种标签选择器，基于等值的（equality-based）和基于集合的（set-based）。

- 一个为空的标签选择器（即有0个必须条件的选择器）会选择集合中的每一个对象。
- 一个null型标签选择器（仅对于可选的选择器字段才可能）不会返回任何对象。

基于等值的标签选择器

- Equality-based标签选择器允许用标签的key和values过滤。有三种运算符可以使用，分别是“=”，“==”和“！=”。前两种运算符同义，代表相等；后一种代表不相等。
  - 查看有特定标签的对象

```
[root@k8s-master runfile]# kubectl get pods -l time=2019 --show-labels
NAME       READY   STATUS    RESTARTS   AGE   LABELS
labelpod   1/1     Running   0          51m   app=busybox,time=2019,version=new
```

  - 查看不包括特定标签的对象

```
[root@k8s-master runfile]# kubectl get pods -l time!=2019 --show-labels
NAME     READY   STATUS    RESTARTS   AGE   LABELS
mypod1   1/1     Running   0          19m   <none>
```


- time=2019和time==2019意义相同

基于集合的标签选择器

- Set-based的标签条件允许用一组value来过滤key。支持三种操作符：in ， notin和exists(仅针对于key符号) 。

```
environment in (production, qa)
tier notin (frontend, backend)
partition
!partition
```

- 两种标签选择器也可以混用，如：
  - partition in (customerA, customerB),environment!=qa

  HUAWEI

- 前半部分使用的是基于集合的标签选择器，后半部分是基于等式的。

标签选择器常用命令

- 查看对象的清单时，可以选择使用-L命令查看标签，或使用-l命令筛选对象。
- 查看pod，并显示app标签的值

```
[root@k8s-master runfile]# kubectl get pod -L app
NAME                              READY   STATUS    RESTARTS   AGE   APP
mypod                             2/2     Running   0          19s   nginx
```

- 筛选pod中标签为app=nginx的pod

```
[root@k8s-master runfile]# kubectl get pod -l app=nginx
NAME                              READY   STATUS    RESTARTS   AGE
mypod                             2/2     Running   0          3m1s
```


- Kubectl命令大小写敏感

将节点打上标签

- 将节点2打上标签

```
Kubectl label nodes k8s-node2 env=test
```

- 查看节点上的env标签

```
[root@k8s-master runfile]# kubectl get node -L env
NAME          STATUS   ROLES    AGE   VERSION   ENV
k8s-master    Ready    master   21d   v1.14.1
k8s-node1     Ready    <none>   21d   v1.14.1
k8s-node2     Ready    <none>   21d   v1.14.1   test
```

- 也可以使用--show-labels命令查看所有标签

```
Kubectl get nodes --show-labels
```


- 本页和下一页为使用标签和标签选择器的一个实际案例。

使用Nodeselector选择节点

- 编辑yaml文件如右，创建deployment
- 查看pod所在节点位置，可以看到创建出来的pod都运行在Node2上。
- 本实验nodeSelector使用的是哪一种选择器?

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
      nodeSelector:
        env: test
```

使用Node affinity

- 继续使用上一页的yaml文件，将nodeselector替换为下图所示内容。

```
affinity:
       nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
           nodeSelectorTerms:
           - matchExpressions:
             - key: env
               operator: In
               values:
               - test
```

- 查看pod运行节点，依然全部运行在node2上。
- 本次使用的选择器是基于集合的选择器。

- 本章节讲述的亲和性调度功能我们后续会详细介绍，此处仅作为演示基于集合的选择器示例。

实验&实训任务

- 实验任务
  - 请按照实验手册2.5章节完成标签选择器和标签实验，包括：
    - 创建标签
    - 使用标签选择器
    - 使用标签选择器实现调度
- 实训任务
  - 请灵活使用本章节课程及实验手册中学到的知识，按照实验手册2.5.4章节完成标签和标签选择器实训任务。

本章总结

- 本章节介绍了：
  - 标签的定义与使用
  - 标签选择器的定义与使用


谢 谢
www.huawei.com