configMap 描述信息

ConfigMap 功能在 Kubernetes1.2 版本中引入，许多应用程序会从配置文件、命令行参数或环境变量中读取配 置信息。ConfigMap API 给我们提供了向容器中注入配置信息的机制，ConfigMap 可以被用来保存单个属性，也 可以用来保存整个配置文件或者 JSON 二进制大对象

ConfigMap 的创建

Ⅰ、使用目录创建

$ ls docs/user-guide/configmap/kubectl/

game.properties

ui.properties

$ cat docs/user-guide/configmap/kubectl/game.properties

enemies=aliens

lives=3

enemies.cheat=true

enemies.cheat.level=noGoodRotten

secret.code.passphrase=UUDDLRLRBABAS

secret.code.allowed=true

secret.code.lives=30

$ cat docs/user-guide/configmap/kubectl/ui.properties

color.good=purple

color.bad=yellow

allow.textmode=true

how.nice.to.look=fairlyNice

$ kubectl create configmap game-config --from-file=docs/user-guide/configmap/kubectl

—from-file 指定在目录下的所有文件都会被用在 ConfigMap 里面创建一个键值对，键的名字就是文件名，值就 是文件的内容

Ⅱ、使用文件创建

只要指定为一个文件就可以从单个文件中创建 ConfigMap

$ kubectl create configmap game-config-2 --from-file=docs/user

guide/configmap/kubectl/game.properties

$ kubectl get configmaps game-config-2 -o yaml

—from-file 这个参数可以使用多次，你可以使用两次分别指定上个实例中的那两个配置文件，效果就跟指定整个 目录是一样的

Ⅲ、使用字面值创建

使用文字值创建，利用 —from-literal 参数传递配置信息，该参数可以使用多次，格式如下

$ kubectl create configmap special-config --from-literal=special.how=very --from literal=special.type=charm

$ kubectl get configmaps special-config -o yaml

Pod 中使用 ConfigMap

Ⅰ、使用 ConfigMap 来替代环境变量

apiVersion: v1

kind: ConfigMap

metadata:

name: special-config

namespace: default

data:

special.how: very

special.type: charm

apiVersion: v1

kind: ConfigMap

metadata:

name: env-config

namespace: default

data:

log\_level: INFO

apiVersion: v1

kind: Pod

metadata:

name: dapi-test-pod

spec:

containers:

- name: test-container

image: hub.atguigu.com/library/myapp:v1

command: [ "/bin/sh", "-c", "env" ]

env:

- name: SPECIAL\_LEVEL\_KEY

valueFrom:

configMapKeyRef:

name: special-config

key: special.how

- name: SPECIAL\_TYPE\_KEY

valueFrom:

configMapKeyRef:

name: special-config

key: special.type

envFrom:

- configMapRef:

name: env-config

restartPolicy: Never

Ⅱ、用 ConfigMap 设置命令行参数

apiVersion: v1

kind: ConfigMap

metadata:

name: special-config

namespace: default

data:

special.how: very

special.type: charm

apiVersion: v1

kind: Pod

metadata:

name: dapi-test-pod

spec:

containers:

- name: test-container

image: hub.atguigu.com/library/myapp:v1

command: [ "/bin/sh", "-c", "echo $(SPECIAL\_LEVEL\_KEY) $(SPECIAL\_TYPE\_KEY)" ] env:

- name: SPECIAL\_LEVEL\_KEY

valueFrom:

configMapKeyRef:

name: special-config

key: special.how

- name: SPECIAL\_TYPE\_KEY

valueFrom:

configMapKeyRef:

name: special-config

key: special.type

restartPolicy: Never

Ⅲ、通过数据卷插件使用ConfigMap

apiVersion: v1

kind: ConfigMap

metadata:

name: special-config

namespace: default

data:

special.how: very

special.type: charm

在数据卷里面使用这个 ConfigMap，有不同的选项。最基本的就是将文件填入数据卷，在这个文件中，键就是文 件名，键值就是文件内容

apiVersion: v1

kind: Pod

metadata:

name: dapi-test-pod

spec:

containers:

- name: test-container

image: hub.atguigu.com/library/myapp:v1

command: [ "/bin/sh", "-c", "cat /etc/config/special.how" ]

volumeMounts:

- name: config-volume

mountPath: /etc/config

volumes:

- name: config-volume

configMap:

name: special-config

restartPolicy: Never

ConfigMap 的热更新

apiVersion: v1

kind: ConfigMap

metadata:

name: log-config

namespace: default

data:

log\_level: INFO

---

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: my-nginx

spec:

replicas: 1

template:

metadata:

labels:

run: my-nginx

spec:

containers:

- name: my-nginx

image: hub.atguigu.com/library/myapp:v1

ports:

- containerPort: 80

volumeMounts:

- name: config-volume

mountPath: /etc/config

volumes:

- name: config-volume

configMap:

name: log-config

$ kubectl exec `kubectl get pods -l run=my-nginx -o=name|cut -d "/" -f2` cat /etc/config/log\_level

INFO

修改 ConfigMap

$ kubectl edit configmap log-config

修改 log\_level 的值为 DEBUG 等待大概 10 秒钟时间，再次查看环境变量的值

$ kubectl exec `kubectl get pods -l run=my-nginx -o=name|cut -d "/" -f2` cat /tmp/log\_level DEBUG

ConfigMap 更新后滚动更新 Pod

更新 ConfigMap 目前并不会触发相关 Pod 的滚动更新，可以通过修改 pod annotations 的方式强制触发滚动更新

$ kubectl patch deployment my-nginx --patch '{"spec": {"template": {"metadata": {"annotations": {"version/config": "20190411" }}}}}'

这个例子里我们在 .spec.template.metadata.annotations 中添加 version/config ，每次通过修改 version/config 来触发滚动更新

！！！ 更新 ConfigMap 后：

使用该 ConfigMap 挂载的 Env 不会同步更新

使用该 ConfigMap 挂载的 Volume 中的数据需要一段时间（实测大概10秒）才能同步更新