ConfigMap与Secret

前言

- 本章节介绍了ConfigMap与Secret两种对象的概念和使用。ConfigMap用于将一些配置文件传递给Pod，而Secret用于将一些敏感信息通过一些安全手段传递给Pod。

目标

- 学完本课程后，您将能够：
  - 描述ConfigMap的功能
  - 使用ConfigMap
  - 区分Secret和ConfigMap的区别
  - 使用Secret

目录

开发中的困难

- 当开发人员开发完成一个应用程序，比如一个Web程序，在正式上线之前需要在各种环境中运行，例如开发时的开发环境，测试环节的测试环境，直到最终的线上环境。Web程序在各种不同的环境中都需要对接不同的数据库、中间件等服务，在传统方式中我们通过配置文件来定义这些配置，而kubernetes中如果需要进入一个个Pod来配置，那将会是一个巨大的麻烦。

- 容器在使用volume时不需要关心后端存储是什么系统，对它来说，所有类型的volume都只是一个目录。

ConfigMap的功能

- ConfigMap用于容器的配置文件管理。它作为多个properties文件的应用，类似一个专门存储配置文件的目录，里面存放着各种配置文件。
- ConfigMap实现了image和应用程序的配置文件、命令行参数和环境变量等信息解耦。
- ConfigMap和Secrets类似，但ConfigMap用于处理不含敏感信息的配置文件。

- 相同的Pod模板加不同的ConfigMap在不同环境中可以顺利运行。

创建ConfigMap - 从目录创建

- 创建两个配置文件如右图，放置在/runfile/configmap文件夹下。
- 使用kubectl命令可以创建一个ConfigMap。--from-file目录下的所有文件都会被用在ConfigMap中创建一个键值对，键是文件名，值是文件的内容。

```
[root@k8s-master configmap]# cat 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
```

```
[root@k8s-master configmap]# cat ui.properties
color.good=purple
color.bad=yellow
allow.textmode=true
how.nice.to.look=fairlyNice
```

```
# kubectl create configmap game-config --from-file=/runfile/configmap
configmap/game-config created
```


- 本案例中configMap中的内容其实没有实际意义。

查看ConfigMap

- 创建完成后依然可以通过get或describe命令查看ConfigMap。

```
# kubectl get configmap
NAME          DATA   AGE
game-config   2      28s
```

```
[root@k8s-master configmap]# kubectl
describe configmaps
Name:         game-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
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

ui.properties:
----
color.good=purple
color.bad=yellow
allow.textmode=true
how.nice.to.look=fairlyNice
```

创建ConfigMap - 从文件创建

- 从目录创建时指定了/runfile/configmap目录，创建时使用了目录下的所有文件。如果指定的不是目录而是单独文件，即从文件创建ConfigMap。

```
#kubectl create configmap game-config --from-file=/runfile/configmap
```

- --from-file参数可以多次使用，用于从多个文件创建ConfigMap的场景：

```
# kubectl create configmap game-config-3 --from-file=/runfile/configmap/game.properties
--from-file=/runfile/configmap/ui.properties
configmap/game-config-3 created
```


- 可以通过describe命令对比看到，ConfigMap game-config-3其实内容与之前通过目录创建的game-config一致。

创建ConfigMap - 从literal值创建

- 之前的方式都需要新建配置文件，然后从文件创建ConfigMap。Kubernetes还提供使用实际配置值创建的方式，通过--from-literal参数实现。

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

```
# kubectl describe configmaps special-config
Name:         special-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
special.how:
----
very
special.type:
----
charm
Events:  <none>
```


- 命令保存了两个数据，第一个的key是special.how，值是very，第二个的key是special.type，值是charm

创建ConfigMap - 从Yaml文件创建

- 与deployment，pod等资源对象相同，ConfigMap也可以使用Yaml文件进行创建。
- Data字段中，key1的定义方式类似使用--from-iterial，pro.property的定义方式类似使用--from-file.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: specialconfig-2
data:
  key1: value1
  pro.property: |
    key2: value2
    key3: value3
```

```
kubectl describe configmaps
Name:         specialconfig-2
Namespace:    default
Labels:       <none>
Annotations:
kubectl.kubernetes.io/last-applied-
configuration:

{"apiVersion":"v1","data":{"key1":"valu
e1","pro.property":"key2: value2\nkey3:
value3\n"},"kind":"ConfigMap","metadata
":{"annotations":{},...

Data
====
key1:
----
value1
pro.property:
----
key2: value2
key3: value3
```


HUAWEI

- 本实验保存了几个数据？Key分别是什么？

使用ConfigMap - 通过环境变量

- 可以直接将ConfigMap里的参数直接做为容器的环境变量供其中程序调用。
- env可以创建环境变量的键，并且引用configmap里特定参数做为值。envFrom则自动将configmap的所有的键值对自动变成环境变量。
- 创建完成后容器中额外的环境变量如下：

```
special-env=value1
key1=value1
pro.property=key2: value2
key3: value3
```

```
apiVersion: v1
kind: Pod
metadata:
  name: cm-test-pod
spec:
  containers:
    - name: cm-container
      image: busybox
      args: ["/bin/sh", "-c", "env"]
      env:
        - name: special-env
          valueFrom:
            configMapKeyRef:
              name: specialconfig-2
              key: key1
      envFrom:
        - configMapRef:
            name: specialconfig-2
  restartPolicy: Never
```


- 这些环境变量分别是env和envFrom中哪个字段创建的?

使用ConfigMap - 通过Volume

- 使用方式和存储章节中使用emptyDir 和 hostPath 的volume类似，configMap挂载后变成了容器内的一个目录。
- 在容器的/etc/db目录中，我们可以看到两个文件，分别是key1，和pro.property，对应configmap中的两个DATA。键就是文件名，值就是文件内容。

```
apiVersion: v1
kind: Pod
metadata:
  name: cmpod2
spec:
  containers:
  - name: cmpod2
    image: busybox
    args: [ "/bin/sh", "-c", "sleep 3000" ]
    volumeMounts:
    - name: db
      mountPath: "/etc/db"
      readOnly: true
  volumes:
  - name: db
    configMap:
      name: specialconfig-2
```


- DATA数量由configmap创建时决定

- Pod创建时会检查ConfigMap

目录

HUAWEI

Secret概述

- Secret 是一种包含少量敏感信息例如密码、token 或 key 的对象。这样的信息可能会被放在 Pod spec 中或者镜像中；将其放在一个 secret 对象中可以更好地控制它的用途，并降低意外暴露的风险。
- ConfigMap主要解决配置文件的存储问题，而Secret主要用来解决密码、token、密钥等敏感数据。
  - 在创建、查看和编辑Pod的流程中Secret暴露风险较小。
  - 系统会对Secret对象采取额外的预防措施，例如避免将其写入磁盘中可能的位置。
  - 只有Pod请求的Secret在其容器中才是可见的，一个Pod不能访问另一个Pod的Secret。

- Secret较ConfigMap安全，但不是绝对安全的，数据仍然有被发现的风险。

创建Secret - 使用kubectl命令创建

- 创建一个两个文件，写入用户名和密码。

```
[root@k8s-master secret]# echo -n "admin" > username.txt
[root@k8s-master secret]# echo -n "Huawei@123" > password.txt
```

- 使用kubectl命令创建。Secret的名称为db-user-pass。

```
[root@k8s-master secret]#  kubectl create secret generic db-user-
pass --from-file=./username.txt --from-file=./password.txt
```

- 其中generic参数代表从本地的文件、目录或实际值（literal value）。

- 注意，上下两个命令创建的secret是不同的。

查看Secret

- 使用get命令可以查看到Secret信息，其中Opaque类型表示base64编码格式的Secret。

```
[root@k8s-master secret]# kubectl get secret
NAME                  TYPE                                  DATA   AGE
db-user-pass          Opaque                                2      5m30s
```

- 通过describe命令可以发现Secret与ConfigMap不同，不会直接显示内容。

```
Type:  Opaque

Data
====
password.txt:  10 bytes
username.txt:  5 bytes
```


- Base64编码可以被解码，因此不是一种绝对安全的方式。

创建Secret - 使用Yaml文件创建

- 为避免Yaml文件中的值被查看到，因此需要先用base64编码后，将值写入Yaml。

```
[root@k8s-master secret]# echo -n "admin" | base64
YWRtaW4=
[root@k8s-master secret]# echo -n "Huawei@123" |
base64
SHVhd2VpQDEyMw==
```

- 创建Yaml文件。

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: YWRtaW4=
  password: SHVhd2VpQDEyMw==
```


- 如果写入yaml时没有经过base64编码，在pod使用secret时会由于解码获得不正确的值。

使用Secret - volume方式

- 使用volume方式挂载Secret给Pod和使用ConfigMap类似。
- 一个Secret在容器内体现为一个目录，一对键值对是一个文件，其中键是文件名称，值是文件内容。
- Secret内容在挂载给pod时进行解码，因此在Pod内部是明文呈现。

```
apiVersion: v1
kind: Pod
metadata:
  name: spod
spec:
  containers:
  - image: busybox
    name: spod
    args: ["/bin/sh","-c","sleep 3000"]
    volumeMounts:
    - name: secrets
      mountPath: "/etc/secret"
      readOnly: true
  volumes:
  - name: secrets
    secret:
      secretName: mysecret
```


- Pod会自动解码secret内容

使用Secret - 挂载指定值

- 在挂载Secret的时候可以指定items，只将Secret中的某些参数传递到Pod中。如右图所示：
  - Secret中password传递到了pod中，而username没有。
  - 在Pod中/etc/secret/my-group有一个my-passwd文件，内容为Huawei@123。

```
apiVersion: v1
kind: Pod
metadata:
  name: spod2
spec:
  containers:
  - image: busybox
    name: spod2
    args: ["/bin/sh","-c","sleep 3000"]
    volumeMounts:
    - name: secrets
      mountPath: "/etc/secret"
      readOnly: true
  volumes:
  - name: secrets
    secret:
      secretName: mysecret
      items:
      - key: password
        path: my-group/my-passwd
```


- 本方法没有传递完整的secret内容。

实验&实训任务

- 实验任务
  - 请按照实验手册的2.11章节完成ConfigMap和Secret实验，包括：
    - 创建和使用ConfigMap
    - 创建和使用Secret
- 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，按照实验手册的2.11.5章节完成ConfigMap和Secret的实训任务。

本章总结

- 本章节介绍了两种将个性化数据传递个Pod的方式：
  - ConfigMap，用于传递非敏感信息
  - Secret，用于传递敏感信息


谢 谢
www.huawei.com