Config Map and Secrets

A ConfigMap is a dictionary of key-value pairs that store configuration settings for your application.

Using configmaps we store configuration files in a ConfigMap and we can mount this configuration files into the container.

Configmaps and secrets both are similar, both work in the similar way.

Difference between secrets and configmaps is, we use secrets for some sensitive data and we use configmaps for non sensitive like configuration files and environment variables.

Secrets

secrets we can encrypt the sensitive data.

All the secretes are stored in ETCD database. By default secrets are encoded by base64.

# Creating Kubernetes Secrets Objects

## Create Secret Using Local Files with Kubectl tool

echo -n 'admin' > ./username.txt

echo -n '1f2d1e2e67df' > ./password.txt

kubectl create secret generic mysecret --from-file=./username.txt --from-file=./password.txt

## List the secrets:

kubectl get secrets

kubectl describe secrets mysecret

kubectl get secret mysecret -o yaml

## Decode

echo "YWRtaW4=" | base64 –decode

## Other ways to create

### Command line

kubectl create secret generic myliteralsecret --from-literal=username=admin --from-literal=password=1f2d1e2e67df

From Manifest file

kubectl create secret generic myliteralsecret --from-literal=username=admin --from-literal=password=1f2d1e2e67df

## Use Case

---

apiVersion: v1

kind: Secret

metadata:

name: db-creds

type: Opaque

stringData:

user: 'root'

password: 'mypassword'

host: 'mydb.example.com'

---

apiVersion: v1

kind: Pod

metadata:

name: mysqlclient

spec:

containers:

- name: mysql

image: mysql

command: ["/bin/sh"]

args: ["-c","mysql -u `cat /mnt/db-creds/user)` -p`cat /mnt/db-creds/password)` -h `cat /mnt/db-creds/host)`"]

volumeMounts:

- name: creds

mountPath: "/mnt/db-creds"

readOnly: **true**

volumes:

- name: creds

secret:

secretName: db-creds

---

apiVersion: v1

kind: Pod

metadata:

name: mysqlclient

spec:

containers:

- name: mysql

image: mysql

env:

- name: USER

valueFrom:

secretKeyRef:

name: db-creds

key: user

- name: PASSWORD

valueFrom:

secretKeyRef:

name: db-creds

key: password

- name: HOST

valueFrom:

secretKeyRef:

name: db-creds

key: host

command: ["/bin/sh"]

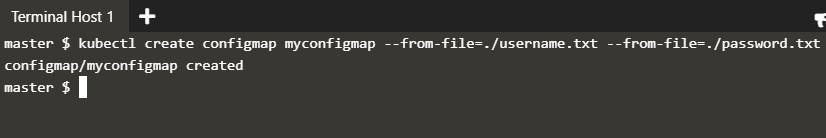
args: ["-c","mysql -u $USER -p$PASSWORD -h $HOST"]

ConfigMap

echo -n 'admin' > ./username.txt

echo -n '1f2d1e2e67df' > ./password.txt

kubectl create configmap myconfigmap --from-file=./username.txt --from-file=./password.txt



kubectl describe configmaps myconfigmap

kubectl get configmap myconfigmap -o yaml

## Create ConfigMaps from literal values

kubectl create configmap myconfig4 --from-literal=dev.admin=marcony --from-literal=db.admin=antony

## From Manifest

apiVersion: v1

kind: ConfigMap

metadata:

Name: game-demo

data:

*# property-like keys; each key maps to a simple value*

player\_initial\_lives: 3

ui\_properties\_file\_name: "user-interface.properties"

*#*

*# file-like keys*

game.properties: *|*

*enemy.types=aliens,monsters*

player.maximum-lives=5

user-interface.properties: *|*

*color.good=purple*

color.bad=yellow

allow.textmode=**true**

## Use Case

There are four different ways that you can use a ConfigMap to configure a container inside a Pod:

1. Command line arguments to the entrypoint of a container
2. Environment variables for a container
3. Add a file in read-only volume, for the application to read
4. Write code to run inside the Pod that uses the Kubernetes API to read a ConfigMap

## Example:

apiVersion: v1

kind: Pod

metadata:

name: configmap-demo-pod

spec:

containers:

- name: demo

image: game.example/demo-game

env:

*# Define the environment variable*

- name: PLAYER\_INITIAL\_LIVES *# Notice that the case is different here*

*# from the key name in the ConfigMap.*

valueFrom:

configMapKeyRef:

name: game-demo *# The ConfigMap this value comes from.*

key: player\_initial\_lives *# The key to fetch.*

- name: UI\_PROPERTIES\_FILE\_NAME

valueFrom:

configMapKeyRef:

name: game-demo

key: ui\_properties\_file\_name

volumeMounts:

- name: config

mountPath: "/config"

readOnly: **true**

volumes:

*# You set volumes at the Pod level, then mount them into containers inside that Pod*

- name: config

configMap:

*# Provide the name of the ConfigMap you want to mount.*

name: game-demo