Création et utilisation d'un Secret

Exercice

Dans cet exercice nous allons voir l'utilisation d'un Secret pour se connecter à une base de données externe.

1. Le context

L'image *lucj/messages:1.0* contient une application simple qui permet, via des requêtes HTTP, de créer des messages ou de lister les messages existant.

Ces messages sont sauvegardés dans une base de données *MongoDB* dont l'URL de connexion doit être fournie à l'application de façon à ce que celle-ci puisse s'y connecter. On peut lui fournir via une variable d'environnement MONGODB_URL ou via un fichier texte accessible depuis */run/secrets/MONGODB_URL*.

2. La base de données

Pour cet exercice la base de données suivante sera utilisée:

- host: *db.techwhale.io*
- port: *27017* (port par défaut de Mongo)
- nom de la base: *message*
- tls activé
- utilisateur: *k8sExercice* / *k8sExercice*

L'URL de connection est la suivante:

```
mongodb://k8sExercice:k8sExercice@db.techwhale.io:27017/message?
ssl=true&authSource=admin
```

3. Création du Secret

Créez un Secret, nommé *mongo*, dont le champ *data* contient la clé *mongo_url* dont la valeur est la chaine de connection spécifiée ci-dessus.

Vous avez plusieurs options pour cela:

- l'utilisation de la commande `kubectl create secret generic` avec l'option `--from-file`
- l'utilisation de la commande `kubectl create secret generic` avec l'option `--from-literal`
- l'utilisation d'un fichier de spécification

4. Utilisation du Secret dans une variable d'environnement

Définissez un Pod nommé *api-env* dont l'unique container a la spécification suivante:

- nom: *api*
- image: *lucj/messages:1.0*
- une variable d'environnement *MONGODB_URL* ayant la valeur liée à la clé *mongo_url* du Secret *mongo* créé précédemment

Créez le Pod et vérifier que vous pouvez créer un message avec la commande suivante (vous pourrez utiliser `kubectl port-forward` pour exposer l'application du Pod)

```
curl -H 'Content-Type: application/json' -XPOST -d '{"from":"me",
"msg":"hello"}' http://IP:PORT/messages
```

5. Utilisation du Secret dans un volume

Définissez un Pod nommé *api-vol* ayant la spécification suivante:

- un volume nommé *mongo-creds* basé sur le Secret *mongo* dont la clé *mongo_url* est renommée *MONGODB_URL* (utilisation du couple (key,path) sous la clé secret/items)
- un container ayant la spécification suivante:
  - nom: *api*
  - image: *lucj/messages:1.0*
  - une instructions *volumeMounts* permettant de monter le volume *mongo-creds* sur le path */run/secrets*

Créez le Pod et vérifier que vous pouvez créer un message.

Correction

3. Création du Secret

1. Avec la commande `kubectl create secret generic` avec l'option `--from-file`

On commence par créer un fichier *mongo_url* contenant la chaine de connexion à la base de données.

```
echo -n "mongodb://k8sExercice:k8sExercice@db.techwhale.io:27017/message?
ssl=true&authSource=admin" > mongo_url
```

On créé ensuite le Secret à partir de ce fichier:

```
$ kubectl create secret generic mongo --from-file=mongo_url
```

2. Avec la commande `kubectl create secret generic` avec l'option `--from-literal`

La commande suivante permet de créer le Secret à partir de valeurs littérales

```
$ kubectl create secret generic mongo \
--from-
literal=mongo_url='mongodb://k8sExercice:k8sExercice@db.techwhale.io:27017/message?
ssl=true&authSource=admin'
```

3. En utilisant un fichier de spécification

La première étape est d'encrypter en base64 la chaine de connexion

```
$ echo -n 'mongodb://k8sExercice:k8sExercice@db.techwhale.io:27017/message?
ssl=true&authSource=admin' | base64

bW9uZ29kYjovL2s4c...yY2U9YWRtaW4=
```

Ensuite on peut définir le fichier de spécification mongo-secret.yaml:

```
apiVersion: v1
kind: Secret
metadata:
```

```
  name: mongo
data:
  mongo_url: bW9uZ29kYjovL2s4c...yY2U9YWRtaW4=
```

La dernière étape consiste à créer le Secret à partir de ce fichier

```
$ kubectl create -f mongo-secret.yaml
secret "mongo" created
```

4. Utilisation du Secret dans une variable d'environnement

Nous définissons la spécification suivante dans le fichier *pod_messages_env.yaml*

```
apiVersion: v1
kind: Pod
metadata:
  name: api-env
spec:
  containers:
  - name: api
    image: lucj/messages:1.0
    env:
    - name: MONGODB_URL
      valueFrom:
        secretKeyRef:
          name: mongo
          key: mongo_url
```

On peut alors créer le Pod:

```
$ kubectl create -f pod_messages_env.yaml
pod "api-env" created
```

La commande suivante permet d'exposer en localhost l'API tournant dans le container du Pod:

```
$ kubectl port-forward api-env 8888:80
Forwarding from 127.0.0.1:8888 -> 80
...
```

Depuis la machine locale, on peut alors envoyer une rquête POST sur l'API:

```
curl -H 'Content-Type: application/json' -XPOST -d '{"from":"me", "msg":"hey"}'
http://localhost:8888/messages
{"from":"me","msg":"hey","at":"2018-04-
03T12:45:07.688Z","_id":"5ac37753dfe0ee000f9b65e0"}
```

5. Utilisation du Secret dans un volume

Nous définissons la spécification suivante dans le fichier *pod_messages_vol.yaml*

```
apiVersion: v1
kind: Pod
metadata:
  name: api-vol
spec:
  containers:
  - name: api
    image: lucj/messages:1.0
    volumeMounts:
    - name: mongo-creds
      mountPath: "/run/secrets"
      readOnly: true
  volumes:
  - name: mongo-creds
    secret:
      secretName: mongo
      items:
      - key: mongo_url
        path: MONGODB_URL
```

On peut alors créer le Pod:

```
$ kubectl create -f pod_messages_vol.yaml
pod "api-vol" created
```

La commande suivante permet d'exposer en localhost l'API tournant dans le container du Pod:

```
$ kubectl port-forward api-vol 8889:80
Forwarding from 127.0.0.1:8889 -> 80
...
```

Depuis la machine locale, on peut alors envoyer une rquête POST sur l'API:

```
$ curl -H 'Content-Type: application/json' -XPOST -d '{"from":"me",
"msg":"hola"}' http://localhost:8889/messages
{"from":"me","msg":"hola","at":"2018-04-
03T13:05:11.408Z","_id":"5ac37c07bace38000f9b09e2"}
```