Var. Entorno, ConfigMap, Secrets

Antonio Espín Herranz

Contenidos

- Despliegues parametrizados:
  - Variables de entorno
  - ConfigMap
  - Secrets

Var Entorno

- Para añadir alguna configuración específica a la hora de lanzar un contenedor, se usan variables de entorno del contenedor cuyos valores se especifican al crear el contenedor para realizar una configuración concreta del mismo.

- Por ejemplo, si estudiamos la documentación de la imagen mariadb en Docker Hub, podemos comprobar que podemos definir un conjunto de variables de entorno como **MYSQL_ROOT_PASSWORD, MYSQL_DATABASE, MYSQL_USER, MYSQL_PASSWORD,** etc., que nos permitirán configurar de alguna forma determinada nuestro servidor de base de datos (indicando la contraseña del usuario root, creando una determinada base de datos o creando un usuario con una contraseña, por ejemplo.

Var. Entorno

- De la misma manera, al especificar los contenedores que contendrán los Pods que se van a crear desde un Deployment, también se podrán inicializar las variables de entorno necesarias.

Var. Entorno

Configuración de aplicaciones usando variables de entorno

- Vamos a hacer un despliegue de un servidor de base de datos mariadb. Si volvemos a estudiar la documentación de esta imagen en Docker Hub comprobamos que obligatoriamente tenemos que indicar la contraseña del usuario root inicializando la variable de entorno **MYSQL_ROOT_PASSWORD**. El fichero de despliegue que vamos a usar es mariadb-deployment-env.yaml, y vemos el fragmento del fichero donde se define el contenedor:

Var Entorno

```
spec:
    containers:
     - name: contenedor-mariadb
       image: mariadb
       ports:
        - containerPort: 3306
          name: db-port
       env:
        - name: MYSQL_ROOT_PASSWORD
•            value: my-password
```

Var Entorno

- En el apartado containers hemos incluido la sección env donde vamos indicando, como una lista, el nombre de la variable (name) y su valor (value). En este caso hemos indicado la contraseña my-password.

Vamos a comprobar si realmente se ha creado el servidor de base de datos con esa contraseña del root:

kubectl apply -f mariadb-deploymen-env.yaml

kubectl get all

kubectl exec -it deployment.apps/mariadb-deployment -- mysql -u root -p

ConfigMap

La solución de las variables de entorno puede tener alguna limitación:

- Los valores de las variables de entorno están escritos directamente en el fichero yaml. Estos ficheros yaml suelen estar en repositorios git y lógicamente no es el sitio más adecuado para ubicarlos.
- Por otro lado, escribiendo los valores de las variables de entorno directamente en los ficheros, hacemos que estos ficheros no sean reutilizables en otros despliegues y que el procedimiento de cambiar las variables sea tedioso y propenso a errores, porque hay que hacerlo en varios sitios.
- Para solucionar estas limitaciones, podemos usar un nuevo recurso de Kubernetes llamado ConfigMap.

ConfigMap

Configuración de aplicaciones usando ConfigMaps

ConfigMap permite definir un diccionario (clave,valor) para guardar información que se puede utilizar para configurar una aplicación.

Aunque hay distintas formas de indicar el conjunto de claves-valor de nuestro ConfigMap, en este caso vamos a usar literales, por ejemplo:

```
kubectl create cm mariadb --from-literal=root_password=my-password \
        --from-literal=mysql_usuario=usuario    \
        --from-literal=mysql_password=password-user \
        --from-literal=basededatos=test
```

ConfigMap

- Para ver los ConfigMap que tenemos creados, podemos utilizar:

kubectl get cm

Y para ver los detalles del mismo:

kubectl describe cm mariadb

- Una vez que creado el ConfigMap se puede crear un despliegue donde las variables de entorno se inicializan con los valores guardados en el ConfigMap. Por ejemplo, un despliegue de una base de datos lo podemos encontrar en el fichero mariadb-deployment-configmap.yaml y el fragmento donde definimos las variables de entorno quedaría:

Ejemplo

```
spec:
    containers:
     - name: contenedor-mariadb
       image: mariadb
       ports:
        - containerPort: 3306
          name: db-port
       env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
           configMapKeyRef:
            name: mariadb
            key: root_password
        - name: MYSQL_USER
          valueFrom:
            configMapKeyRef:
              name: mariadb
              key: mysql_usuario
        - name: MYSQL_PASSWORD
          valueFrom:
           configMapKeyRef:
            name: mariadb
            key: mysql_password
        - name: MYSQL_DATABASE
          valueFrom:
           configMapKeyRef:
            name: mariadb
        - key: basededatos
```

ConfigMap

- Observamos como al indicar las variables de entorno (sección env) seguimos indicado el nombre (name) pero el valor se indica con una clave de un ConfigMap (valueFrom: -configMapKeyRef:), para ello se indica el nombre del ConfigMap (name) y el valor que tiene una determinada clave (key).

- De esta manera, no guardamos en los ficheros yaml los valores específicos de las variables de entorno, y además, estos valores se pueden reutilizar para otros despliegues, por ejemplo, al desplegar un CMS indicar los mismos valores para las credenciales de acceso a la base de datos.

Secrets

- Cuando en un variable de entorno indicamos una información sensible, como por ejemplo, una contraseña o una clave ssh, es mejor utilizar un nuevo recurso de Kubernetes llamado Secret.
- Los Secrets permiten guardar información sensible que será **codificada** o **cifrada**.
- Hay distintos tipos de Secret, en este curso vamos a usar los genéricos y los vamos a crear a partir de un literal. Por ejemplo para guardar la contraseña del usuario root de una base de datos, crearíamos un Secret de la siguiente manera:
- **kubectl create secret generic mariadb --from-literal=password=my-password**

Secrets

Podemos obtener información de los Secret que hemos creado con las instrucciones:

- **kubectl get secret**
- **kubectl describe secret mariadb**
- El despliegue que usa el valor de un **Secret** para inicializar una variable de entorno. Vamos a usar el fichero mariadb-deployment-secret.yaml y el fragmento donde definimos las variables de entorno quedaría:

Secrets

```
spec:
  containers:
   - name: mariadb
     image: mariadb
     ports:
      - containerPort: 3306
        name: db-port
     env:
      - name: MYSQL_ROOT_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mariadb
            key: password
```

Secrets

- Observamos como al indicar las variables de entorno (sección env) seguimos indicado el nombre (name) pero el valor se indica con un valor de un Secret (valueFrom: - secretKeyRef:), indicando el nombre del Secret (name) y la clave correspondiente. (key).
- **Nota:** Por defecto, k8s no cifra el valor de los Secrets, simplemente los codifica en base64. El cifrado de los Secrets requiere configuraciones adicionales que se detallan en Encrypting Secret Data at Rest