 # Despliegues parametrizados: ConfigMaps

En el apartado anterior hemos estudiado como podemos definir las variables de entorno de los contenedores que vamos a desplegar. Sin embargo, la solución que presentamos 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.

Configuración de aplicaciones usando ConfigMaps

[ConfigMap](https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-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
```

En el ejemplo anterior, hemos creado un ConfigMap llamado `mariadb` con cuatro pares clave-valor. 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`](files/mariadb-deployment-configmap.yaml)
y el fragmento donde definimos las variables de entorno quedaría:

```yaml
...
  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
```

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.