UNIDAD 3
DESPLIEGUE DE APLICACIONES EN KUBERNETES (III)

Variables de Entorno

Podemos configurar las variables de entorno de los contenedores mediante variables de entorno:

Ejemplo 1: Configuración de contraseña en MariaDB

Nota: Ya veremos que no es la forma más adecuada de almacenar una contraseña en k8s

ConfigMap

ConfigMap es un objeto de kubernetes(ConfigMap) que permite definir un diccionario clave, valor para guardar datos no confidenciales

Ejemplo2: Configuración de la BBDD MariaDB

Nota: ConfigMap tampoco es la forma adecuada de guardar la contraseña

Secrets

Los Secrets nos permiten guardar información sensible que será codificada. Por ejemplo,nos permite guarda contraseñas, claves ssh, …

- Se pueden utilizar de tres formas:
  - Como ficheros en un volumen montado en uno o más pods
  - Como una variable de entorno del contenedor
  - Obtenido por kubelet al obtener la imagen para el pod
- Tipos más habituales
  - generic: Se pueden indicar desde un directorio, un fichero o un literal. Para una contraseña, una clave ssh, etc.
  - docker-registry: Para las credenciales de acceso a un registro de docker
  - tls: Para un par de claves privada/pública TLS

EJEMPLO 3: Uso de secrets

EJEMPLO 4: Desplegando WordPress con MariaDB

Los pods son efímeros

Cuando se elimina un pod su información se pierde. Por lo tanto nos podemos encontrar con algunas circunstancias:

1. ¿Qué pasa si eliminamos el despliegue de mariadb?, o, ¿se elimina el pod de mariadb y se crea uno nuevo?.
2. ¿Qué pasa si escalamos el despliegue de la base de datos y tenemos dos pods ofreciendo la base de datos?.
3. Si escribimos un post en wordpress y subimos una imagen, ¿qué pasa con esta información en el pod?
4. En el caso que tengamos un pod con contenido estático (por ejemplo imágenes), ¿qué pasa si escalamos el despliegue de wordpress a dos pods?

StatefulSet

El objeto **StatefulSet** controla el despliegue de pods con identidades únicas y persistentes, y nombres de host estables. Veamos algunos ejemplos en los que podemos usarlo:

- Un despliegue de redis master-slave: necesita que **el master esté corriendo antes de que podamos configurar las réplicas**.
- Un cluster mongodb: Los diferentes nodos deben **tener una identidad de red persistente** (ya que el DNS es estático), para que se produzca la sincronización después de reinicios o fallos.
- Zookeeper: cada nodo necesita **almacenamiento único y estable**, ya que el identificador de cada nodo se guarda en un fichero.

Por lo tanto el objeto StatefulSet nos ofrece las siguientes **características**:

- Estable y único identificador de red (Ejemplo mongodb)
- Almacenamiento estable (Ejemplo Zookeeper)
- Despliegues y escalado ordenado (Ejemplo redis)
- Eliminación y actualizaciones ordenadas

StatefulSet

Por lo tanto cada pod es distinto (tiene una identidad única), y este hecho tiene algunas consecuencias:

- El nombre de cada pod tendrá un número (1,2,...) que lo identifica y que nos proporciona la posibilidad de que la creación actualización y eliminación sea ordenada.
- Si un nuevo pod es recreado, obtendrá el mismo nombre (hostname), los mismos nombres DNS (aunque la IP pueda cambiar) y el mismo volumen que tenía asociado.
- Necesitamos crear un servicio especial, llamado **Headless Service**, que nos permite acceder a los pods de forma independiente, pero que no balancea la carga entre ellos, por lo tanto este servicio no tendrá una ClusterIP.

StatefulSet vs Deployment

- A diferencia de un Deployment, un StatefulSet mantiene una identidad fija para cada uno de sus Pods.
- Eliminar y / o escalar un StatefulSet no eliminará los volúmenes asociados con StatefulSet.
- StatefulSets actualmente requiere que un Headless Service sea responsable de la identidad de red de los Pods.
- Cuando use StatefulSets, cada Pod recibirá un PersistentVolume independiente.
- StatefulSet actualmente no admite el escalado automático

Componentes StatefulSet: headless service

```yaml
apiVersion: v1
kind: Service
metadata:
 name: nginx
 labels:
   app: nginx
spec:
 ports:
 - port: 80
   name: web
 clusterIP: None
 selector:
   app: nginx
```

- El headless service nos proporciona acceso a los pods creados.
- No va a tener una IP (`clusterIP: None`)
- Será referencia por el objeto StatefulSet (`name: nginx`)

Headless Service

Componentes StatefulSet: StatefulSet

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: web
spec:
 serviceName: "nginx"
 replicas: 2
 selector:
   matchLabels:
     app: nginx
 template:
   metadata:
     labels:
       app: nginx
   spec:
     containers:
     - name: nginx
       image: k8s.gcr.io/nginx-slim:0.8
       ports:
       - containerPort: 80
         name: web
       volumeMounts:
       - name: www
         mountPath: /usr/share/nginx/html
   ...
```

- Se indica los pods que vamos a controlar que vamos a utilizar:

```yaml
selector:
   matchLabels:
     app: nginx
```

- Vamos a montar un volumen persistente

```yaml
volumeMounts:
    - name: www
        mountPath: /usr/share/nginx/html
```

Componentes StatefulSet: volumenClainTemplate

```
...
volumeClaimTemplates:
 - metadata:
     name: www
   spec:
     accessModes: [ "ReadWriteOnce" ]
     resources:
       requests:
         storage: 1Gi
```

- Nos ofrece almacenamiento estable usando PersistentVolumes
- La definición es similar a un `PersistantVolumenClaim`.

Ejemplo5: StatefulSet

DaemonSet

El objeto **DaemonSet (DS)** nos asegura que en todos (o en algunos) nodos de nuestro cluster vamos a tener un pod ejecutándose. Si añadimos nuevos nodos al cluster se crearán nuevo pods. Para que podemos necesitar esta característica:

- Monitorización del cluster (Prometheus)
- Recolección y gestión de logs (fluentd)
- Cluster de almacenamiento (glusterd o ceph)

Ejemplo 6

Otros recursos: Jobs, cronJobs,...

Jobs

- Deseamos ejecutar una acción y asegurarse que se finaliza correctamente (Rellenar una base de datos, descargar datos,...)
- Un Job crea uno o más pods y se asegura que un número determinado de ellos ha terminado de forma adecuada.

Si necesita que un Job se repita periódicamente usamos un cronJob:

- Por ejemplo si quieres hacer backup de base de datos
- Se puede especificar una momento determinado, o indicar una repetición periódica.

Horizontal Pod AutoScaler

El HPA de Kubernetes nos permite variar el número de pods desplegados mediante un *deployment* en función de diferentes métricas: de forma estable usando el porcentaje de **CPU** utilizado y de forma experimental de utilización de la **memoria.**

Necesitamos tener instalado en nuestro cluster un software que permita monitorizar el uso de recursos: **metrics-server.**
En minikube es muy fácil instalarlo: minikube addons enable metrics-server

Ejemplo 13