Entrega Individual - Blob City DB

Angela Gonzalez Garcia

Listado de las tareas realizadas

- Introducción
- Cliente de la aplicación asignada
- Repositorio en DockerHub
- Entorno Docker Compose
- Volúmenes de Docker

¿Qué es BlobCity DB?

BlobCity DB es una base de datos diseñada para el procesamiento de datos a escala organizativa. Utiliza un sistema *NoSQL*, lo que quiere decir que no se basa en el lenguaje *SQL* para realizar sus consultas. Esta base de datos utiliza un procesamiento transaccional/analítico híbrido, esto significa que genera análisis de los datos a la vez que son introducidos en la base de datos de producción en el día a día.

La estructura de datos de está base de datos está formada por un *Data Lake*, esta estructura tiene una arquitectura plana, lo que significa que no está estructurada en archivos y carpetas. Cada elemento de datos de un lago de datos recibe un identificador único y se etiqueta con un conjunto de información de metadatos. Las bases de datos estructuradas de esta manera están orientadas al análisis y procesamiento de datos ya que su estructura permite que los análisis se puedan centrar más en los patrones de significado de los datos sin tener que preocuparse de cómo están organizados.

¿Por qué elegir BlobCity DB frente a otras bases de datos?

BlobCity DB le da mucha importancia a soportar todos los posibles tipos de datos que los usuarios quieran gestionar, siendo compatible con *JSON, XML, CSV, SQL, Log data, Plaintext, RTF, PDF, Word, Excel, Powerpoint, GIS, Syslog*, imagen, video y audio.

Además, esta plataforma ofrece una combinación de servicios que pretende sustituir la necesidad de combinar diferentes tecnologías que puedan ofrecer otras bases de datos unificando todos en una sola. Como no es posible en la práctica tener una base de datos de documentos *(MongoDB)*, luego otra para lago de datos *(Hadoop)*, luego otra de almacenamiento en columnas *(Cassandra)*, luego otra base de datos gráfica *(Neo4J)*, almacenamiento *XML (MarkLogic)* y otra para procesamiento en memoria *(MemSQL)*, si una organización necesitará una tecnología para atender todas estas funciones, todas ellas estarían disponibles en BlobCity DB.

Cliente para BlobCity DB

Esta aplicación cliente es un script de comandos bash que ejecuta una serie de comandos a través del *CLI* (Interfaz de Línea de Comandos) de la base de datos mediante *NetCat*.

```
sleep 10
nc blobcity-db-container 10113 < comandos-blobcitydb.txt
echo "Creada nueva base de datos: REGISTRO"
echo "Creada nueva coleccion: VISITAS"
```

El comando **sleep** le da un margen de 10 segundos al contenedor de la base de datos para iniciarse y a continuación abre una conexión con el puerto 10113 de este contenedor mediante **nc** *(NetCat)*. En el archivo *comandos-blobcitydb.txt* podemos encontrar los comandos que se van a pasar a través de esta conexión a la base de datos: los dos primeros indican el usuario y la contraseña respectivamente y los siguientes se encargar de crear la base de datos y colección y cerrar la conexión.

```
root
root
create-ds registro
create-collection registro.visitas
exit
```

Dockerfile

Esta imagen tiene como base una imagen de ubuntu. Los comandos **COPY** incluyen los dos archivos anteriormente mencionados a la carpeta *datosCliente* creada mediante **WORKDIR**. Mediante el comando *chmod* le damos permisos de ejecución al script de bash y con *apt* instalamos *NetCat* para poder utilizarlo en el interior del contenedor. Finalmente utilizamos el comando **CMD** para que se ejecute el script cada vez que se inicie el contenedor.

```
FROM ubuntu:22.04

#Crear un directorio
WORKDIR /datosCliente

#Añadir el script de al directorio
COPY ./comandos-blobcitydb.txt /datosCliente
COPY ./cliente-blobcitydb.sh /datosCliente

#Dar los permisos necesarios
RUN chmod +x cliente-blobcitydb.sh

#Instalar netcat
RUN apt update
```

```
RUN apt -y install netcat

#Ejecutar el script
CMD ./cliente-blobcitydb.sh
```

Imagen en DockerHub

Esta aplicación está encapsulada en *Docker* y se puede encontrar en el siguiente repositorio público de *DockerHub*:

https://hub.docker.com/r/soyangela/blobcity-db-client

Docker Compose

Para gestionar la ejecución tanto del contenedor de la instancia de blobcityDB como la del cliente utilizaremos *Docker Compose*. Definiremos dos servicios, uno para la base de datos y otro para el cliente, en el primero utilizamos la imagen oficial de blobcity que podemos encontrar en la página de los creadores en *DockerHub* (https://hub.docker.com/r/blobcity/db), nombramos el contenedor y exponemos los puertos 10111 y 10113 que darán acceso a la base de datos a través de los servicios web *REST* y el *CLI* respectivamente. En esta primera parte también definimos el volumen *blob-vol* que le dará apoyo a la persistencia de la base de datos en el momento de recuperar los datos tras reiniciar el contenedor, esta sección esta mejor explicada mas adelante en el apartado de Persistencia de la base de datos . En lo que respecta al servicio del cliente utilizamos la imagen anteriormente creada directamente desde *DockerHub* y nombramos el contenedor.

```
services:
  blobcity:
    image: blobcity/db
    container_name: blobcity-db-container
    ports:
      - 10111:10111
      - 10113:10113
    volumes:
      - blob-vol:/data
  cliente:
    image: soyangela/blobcity-db-client
    container_name: blob_client_container
volumes:
  blob-vol:
```

Persistencia de la base de datos

A pesar de que los requisitos mínimos de la entrega no requerían un guardado de datos de forma persistente, la base de datos trae por defecto una contraseña que se almacena en el archivo */data/root-pass.txt* de la instancia del contenedor *blobcity-db-container*. Es por esto que para que el cliente funcione de forma correcta es necesario un almacenamiento que permita cambiar la contraseña de forma permanente, ya que de lo contrario haría falta extraer la contraseña cada vez que se cree un contenedor.

Ejecución

Para poner en funcionamiento el cliente será necesario ejecutar los siguientes comandos en la carpeta contenedora del archivo *docker-compose.yml*. Todos los archivos necesarios se pueden encontrar en el repositorio público de GitHub:

https://github.com/soyAngela/blobcity-db-as

Como mencionado anteriormente en el apartado de Persistencia de la base de datos la primera vez que se crea el contenedor de BlobCityDB la contraseña autogenerada se almacena en este por lo que será necesario ejecutar un script que cambie la contraseña antes de ejecutar el cliente. Este script cambia la contraseña del usuario *root* a *root*.

Esto lo haremos ejecutando el contenedor y a continuación el script con los siguientes comandos:

```
sudo docker compose up
chmod +x password-changer-blobcitydb.sh
./password-changer-blobcitydb.sh
sudo docker compose down
```

Finalmente, ejecutar de nuevo el contenedor*:*

```
docker compose up
```

Resultado

Al finalizar los pasos anteriores deberíamos poder ver el mensaje que indica que se ha creado la base de datos y la colección. Podemos comprobar que esto es cierto abriendo una conexión de la siguiente manera con las credenciales username *root* y password *root:*

```
nc localhost 10113
```

- *list-ds* → Para listar todas las bases de datos
- *list-collections registro* → Para mostrar las colecciones de la base de datos “registro”
- *drop-ds registro* → Para borrar la base de datos “registro”

Es importante ejecutar este último comando antes de volver a ejecutar el contenedor ya que no se podrá volver a crear la base de datos y colección si ya existen en la instancia de BlobCityDB.

Problemas y dificultades

Inicialmente diseñé un cliente con python que se comunicaba con la base de datos a traves de una API REST, este archivo está incluido en la carpeta de cliente-python del repositorio. A pesar de conseguir una respuesta 200 OK de parte del contenedor, no se llegaban a cumplir las peticiones a la base de datos por lo que tuve que recurrir a introducir los comandos a través del CLI del propio servicio.