3.5 - Tutorial Cassandra

En este trabajo se nos propone realizar una documentación siguiendo el tutorial de la página web facilitada en la rúbrica del ejercicio. Para ello lo organizaremos siguiendo la estructura de la página web, que está fragmentada en diversos apartados. Para cada uno de ellos iremos siguiendo los pasos establecidos y aportando como documentación capturas de pantalla de los pasos realizados, así como una breve descripción aclaratoria cuando sea necesario. Por último, para encontrar el fichero .yml utilizado, facilitare el mismo en mi cuenta de Github.

Pasos previos a la creación del contenedor

Para evitar posibles problemas en el futuro a la hora de levantar el contenedor, realizaremos una limpieza de los puertos de red que no estamos utilizando por ninguno de nuestros contenedores. Para esto nos apoyaremos del comando prune.

```
[~]$ sudo docker network prune
WARNING! This will remove all custom networks not used by at least one container.
Are you sure you want to continue? [y/N] y
```

Tras esto, crearemos el fichero .yml que después utilizaremos con la herramiento docker compose para crear nuestro contenedor. Para ello crearemos un directorio donde guardaremos todos archivos relacionados con Cassandra, así como el editor nano para la creación del fichero. En este pdf no aparece toda la configuración del fichero, para encontrarlo entero, puedes encontrarlo en mi Github.

```
  GNU nano 6.2                          docker-compose.yml *
version: '3'
services:
# The first node and config in the first datacenter.
    node1:
        image: datastax/dse-server:6.8.16-ubi7
        container_name: DSE-6_node1
        hostname: node1
        # use static ip address
        networks:
            dc1ring:
                ipv4_address: 172.30.0.2
        # Maps cassandra exercises to a local folder.
        # This preserves data across container restarts.
        volumes:
            - ./musicdb:/opt/dse/musicdb
        # Docker container environment variable. We are using the
        # CASSANDRA_CLUSTER_NAME to name the cluster. This needs to be the same
        # across clusters. We are also declaring that node1 is a seed node etc.
        environment:
            - DS_LICENSE=accept
```

Levantando el contenedor

Ahora que tenemos nuestro archivo de configuración listo, usaremos el siguiente comando para levantarlo.

```
[~/cassandra]$ sudo  docker-compose -f docker-compose.yml up
Creating network "cassandra_dc1ring" with the default driver
Pulling node1 (datastax/dse-server:6.8.16-ubi7)...
6.8.16-ubi7: Pulling from datastax/dse-server
```

```
DSE-6_node1 | Applying changes to /opt/dse/resources/cassandra/conf/cassandra.yaml ...
DSE-6_node1 | done.
DSE-6_node1 | Applying changes to /opt/dse/resources/cassandra/conf/cassandra-rackdc.properties ...
DSE-6_node1 | done.
DSE-6_node1 | Running dse cassandra -f -R
DSE-6_node2 | Applying changes to /opt/dse/resources/cassandra/conf/cassandra.yaml ...
DSE-6_node3 | Applying changes to /opt/dse/resources/cassandra/conf/cassandra.yaml ...
DSE-6_node2 | done.
DSE-6_node2 | Applying changes to /opt/dse/resources/cassandra/conf/cassandra-rackdc.properties ...
DSE-6_node2 | done.
DSE-6_node2 | Running dse cassandra -f -R
DSE-6_node3 | done.
DSE-6_node3 | Applying changes to /opt/dse/resources/cassandra/conf/cassandra-rackdc.properties ...
DSE-6_node3 | done.
DSE-6_node3 | Running dse cassandra -f -R
DSE-6_node3 exited with code 1
```

Comprobación de los contenedores

```
[~]$ sudo docker ps
[sudo] contraseña para pablovillarte:
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS          PORTS

6ddfcedb2b74   datastax/dse-server:6.8.16-ubi7   "/entrypoint.sh dse …"   18 minutes ago   Up 18 minutes   4040/tcp
90/tcp, 8182/tcp, 8609/tcp, 8983-8984/tcp, 9103/tcp, 9142/tcp, 9160/tcp, 9999-10000/tcp, 18080/tcp, 0.0.0.0:9043->9
cf71cc5cb8f1   datastax/dse-server:6.8.16-ubi7   "/entrypoint.sh dse …"   18 minutes ago   Up 18 minutes   4040/tcp
90/tcp, 8182/tcp, 8609/tcp, 8983-8984/tcp, 9103/tcp, 9142/tcp, 9160/tcp, 9999-10000/tcp, 18080/tcp, 0.0.0.0:9042->9
```

Comprobamos los contenedores activos, y todo ha salido según lo esperado. Por tanto no necestaremos de la herramienta nodetool para activarlos, así que omitiré esa parte de la guía.

Creación del keyspace

```
[~]$ sudo docker exec -it DSE-6_node1 bash
dse@node1:~$ cqlsh
Connected to dse51_cluster at 127.0.0.1:9042.
[cqlsh 6.8.0 | DSE 6.8.16 | CQL spec 3.4.5 | DSE protocol v2]
Use HELP for help.
cqlsh> DESC keyspaces;

system_virtual_schema  system_schema  dse_leases          system_traces
dse_system_local       system_auth    system_backups      dse_perf
dse_security           system_views   dse_insights        dse_insights_local
solr_admin             system         system_distributed  dse_system
```

Con el comando DESC vemos la información del primer cluster (al cual nos hemos conectado)

Creamos el keyspace musicdb y nos conectamos.

```
cqlsh> CREATE KEYSPACE musicDb WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : '3'};
cqlsh> USE musicDb;
```

Utilizando cassandra

```
cqlsh:musicdb> CREATE TABLE musics_by_genre (
           ...   genre VARCHAR,
           ...   performer VARCHAR,
           ...   year INT,
           ...   title VARCHAR,
           ...   PRIMARY KEY ((genre), performer, year, title)
           ... ) WITH CLUSTERING ORDER BY (performer ASC, year DESC, title ASC);
cqlsh:musicdb> DESC TABLE musics_by_genre;

CREATE TABLE musicdb.musics_by_genre (
    genre text,
    performer text,
    year int,
    title text,
    PRIMARY KEY (genre, performer, year, title)
) WITH CLUSTERING ORDER BY (performer ASC, year DESC, title ASC)
    AND additional_write_policy = '99PERCENTILE'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND nodesync = {'enabled': 'true', 'incremental': 'true'}
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';
```

```
[~]$ sudo docker stop DSE-6_node2
DSE-6_node2
[~]$ sudo docker stop DSE-6_node3
DSE-6_node3
```

```
dse@node1:~$ cqlsh
Connected to dse51_cluster at 127.0.0.1:9042.
[cqlsh 6.8.0 | DSE 6.8.16 | CQL spec 3.4.5 | DSE protocol v2]
Use HELP for help.
cqlsh> USE musicDb;
cqlsh:musicdb> CONSISTENCY ALL;
Consistency level set to ALL.
cqlsh:musicdb> SELECT * FROM musics_by_genre WHERE genre='Rock';
NoHostAvailable:
cqlsh:musicdb> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:musicdb> SELECT * FROM musics_by_genre WHERE genre='Rock';

 genre | performer | year | title
-------+-----------+------+-------------------------
  Rock |   Nirvana | 1991 | Smells Like Teen Spirit

(1 rows)
cqlsh:musicdb>
```