

# Instalación y configuración

---

## Apache Cassandra - Bases de datos II

Alberto Díaz Álvarez (<alberto.díaz@upm.es>)

Departamento de Sistemas Informáticos

Escuela Técnica superior de Ingeniería de Sistemas Informáticos

License CC BY-NC-SA 4.0

# Introducción

# Instalación

---

Cassandra funciona en muchos sistemas operativos diferentes

- GNU/Linux, Windows, Mac OS X, FreeBSD, Solaris, etc.
- Nos centraremos en GNU/Linux, concretamente en Ubuntu GNU/Linux 22.04
- Bueno, y en Docker, porque es muy cómodo

Prerequisitos:

- La última versión de Java 8 o Java 11 (`java -version`)
  - Valen tanto la *OpenJDK* como la *Oracle Java Standard Edition*
- Python 3.6 o superior (`python --version`)
  - Esto es por si queremos ejecutar `cqlsh`, la consola de Cassandra
- Docker (si queremos instalar mediante Docker, claro)

# Despliegue mediante Docker

# Instalación y despliegue con la imagen de **docker**

---

Este es el medio más sencillo de instalación:

- Descargar (**pull**) la imagen de Docker de Cassandra:

```
docker pull cassandra:latest
```

- Iniciar un contenedor a partir de la imagen descargada:

```
docker container run -d cassandra --name cassandra_n1
```

- Conectarse al contenedor para interactuar con Cassandra mediante **cqlsh**

```
docker exec -it cassandra_n1 cqlsh
```

Y ya estaría

# Creando un clúster de tres nodos (I)

---

Vamos a comenzar creando una red para los contenedores de Cassandra:

```
docker network create cassandra
```

- No es necesario, pero nos permite acceder a los puertos del contenedor sin exponerlos en el host

Ahora, crearemos el primer nodo de nuestro clúster de la manera habitual:

```
docker container run -d --name cassandra_n1 --network cassandra cassandra
```

Y para este nodo obtendremos su dirección IP:

```
NODE1=$(docker inspect --format="{{ .NetworkSettings.Networks.cassandra.IPAddress }}" cassandra_n1)
```

# Creando un clúster de tres nodos (II)

Con esta IP, crearemos el segundo nodo y con ambas IPs el tercero:

```
docker container run -d --name cassandra_n2 --network cassandra -e CASSANDRA_SEEDS="$NODE1" cassandra
NODE2=$(docker inspect --format="{{ .NetworkSettings.Networks.cassandra.IPAddress }}" cassandra_n2)
docker container run -d --name cassandra_n3 --network cassandra -e CASSANDRA_SEEDS="$NODE1,$NODE2" cassandra
```

Ahora, vamos a comprobar el estado de los tres nodos (hay que esperar un poco)

```
docker exec -it cassandra_n1 nodetool status
```

Tras un rato, deberíamos ver algo tan bonito como esto:

```
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address            Load           Tokens   Owns (effective)  Host ID                                           Rack
UN  172.18.0.4          70.24 KiB      16       76.0%             1c79f250-68e0-41d4-b492-dfe77b78907b          rack1
UN  172.18.0.3          70.24 KiB      16       59.3%             c15431bb-cb4f-4efb-806a-78b004e91a4c          rack1
UN  172.18.0.2          109.39 KiB     16       64.7%             241f0aa1-d4c0-46d9-bd6b-747247eecda9          rack1
```

# Creando un clúster de tres nodos (y III)

Ahora podemos conectarnos a cualquiera de los nodos para ejecutar `cqlsh`:

```
docker exec -it cassandra_n1 cqlsh
```

Y ya podemos empezar a trabajar con el clúster igual que con un sólo nodo:

```
cqlsh> CREATE KEYSPACE test WITH replication = {  
    ...     'class': 'NetworkTopologyStrategy',  
    ...     'replication_factor': 3  
    ... };  
cqlsh> DESCRIBE KEYSPACES;  
  
system          system_distributed  system_traces  system_virtual_schema  
system_auth     system_schema      system_views   test  
  
cqlsh>
```



# Más allá del despliegue local

---

Un clúster de Cassandra puede ser un despliegue en un mismo centro de datos

- En una máquina o en varias máquinas repartidas a lo largo de la misma red

Sin embargo, soporta el despliegue en múltiples centros de datos

- A efectos del sistema cliente, siempre se ve una única instancia de Cassandra
- Internamente, Cassandra se encarga de la replicación de datos entre centros de datos
- Y de la comunicación entre nodos, independientemente de su localización

# Despliegue en Ubuntu GNU/Linux

# Instalación de prerequisites

---

Comenzaremos con la actualización de los repositorios y paquetes de ubuntu:

```
$ sudo apt update && sudo apt upgrade -y
```

Tras ello, instalaremos la última versión de Java 11:

```
$ sudo apt install openjdk-11-jdk -y
```

No hará falta instalar Python, ya que viene instalado por defecto en Ubuntu GNU/Linux 22.04

```
$ python3 --version  
Python 3.10.6
```

# Instalación del paquete de Cassandra

---

Primero, añadiremos el repositorio de Cassandra a la lista de repositorios de apt:

```
$ echo "deb http://www.apache.org/dist/cassandra/debian 40x main" | \
sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list
```

Y ahora, añadiremos la clave pública de Cassandra:

```
$ wget -q -O - https://www.apache.org/dist/cassandra/KEYS | \
sudo tee /etc/apt/trusted.gpg.d/cassandra.asc
```

Por último, tras actualizar los índices de nuevo, instalaremos Cassandra

```
sudo apt update && sudo apt install cassandra -y
```

# Comprobación de que el nodo está funcionando

---

No debería ser necesario reiniciar el sistema, el demonio de cassandra se iniciará automáticamente

Para comprobar que está corriendo, basta con comprobar el estado del servicio:

```
$ sudo systemctl status cassandra
```

Deberíamos obtener algo como esto:

```
• cassandra.service - LSB: distributed storage system for structured data  
  Loaded: loaded (/etc/init.d/cassandra; generated)  
  Active: active (running) since Tue 2023-03-16 03:25:52 UTC; 1min 32s ago  
  ...
```

# Conenctándonos al nodo

---

Podemos comprobar el estado del nodo (o del clúster si hay varios nodos conectados) con `nodetool`:

```
$ sudo nodetool status
```

Para conectarnos, al clúster basta con hacer uso de `cqlsh`:

```
$ cqlsh
```

Esto nos conectará al nodo por defecto en la máquina que se ejecuta el comando

# Sobre el funcionamiento

# Configuración

---

La configuración de Apache Cassandra se encuentra en el archivo `cassandra.yaml`

- Los ajustes comunes incluyen el tamaño del heap, la dirección IP de escucha y la compresión de datos
- Algunos cambios en la configuración requerirán un reinicio del nodo



# Ajustes de memoria

---

La memoria heap es la usada para almacenar datos en caché y procesar consultas

- Es importante ajustarla para equilibrar rendimiento y estabilidad del nodo:
  - Demasiada: Latencia adicional y mayor probabilidad de fallos
  - Muy poca: Funcionamiento incorrecto y bajo rendimiento

# La herramienta `nodetool`

---

Herramienta de línea de comandos para administrar un clúster de Cassandra

- Permite realizar tareas de diagnóstico y mantenimiento en un nodo individual o en todo el clúster

Ejemplos de uso de `nodetool`

- `nodetool status`: Estado de un nodo y los datos de replicación del clúster
- `nodetool repair`: Repara datos inconsistentes en un nodo o todo el clúster
- `nodetool snapshot`: Captura una instantánea de un nodo o todo el clúster
- `nodetool cfstats`: Estadísticas de rendimiento de una tabla en particular

**Gracias**