Apache Cassandra: replication

v1.0

Table of Contents

Objectifs

1. Dans ce TP nous allons commencer par utiliser **Cassandra Cluster Manager** (https://github.com/pcmanus/ccm)(***ccm***) pour lancer un premier cluster Cassandra qui a déjà été crée pour nous.
2. Nous allons ensuite passer en revue les différentes options pour la configuration des noeuds Cassandra.
3. A la fin du TP, nous allons créer un nouveau cluster Cassandra à partir des besoins spécifiques de résilience et de cohérence et nous allons utiliser *cqlsh* pour tracer les échanges entre les noeuds lors de l'exécution des quelques requêtes basiques.

Configuration de l'environnement Cassandra

Pour ce TP vous allez utiliser une machine virtuelle qui contient un cluster Cassandra de 3 noeuds hébergés sur la meme VM. Vous allez vous connecter à cette machine via ***ssh***.

Pour ce TP vous avez besoin d'un poste (idéalement Linux) avec 4GB de RAM et 4GB d'espace disque disponible. ***VirtualBox*** et un client ***SSH*** doivent être déjà installés.

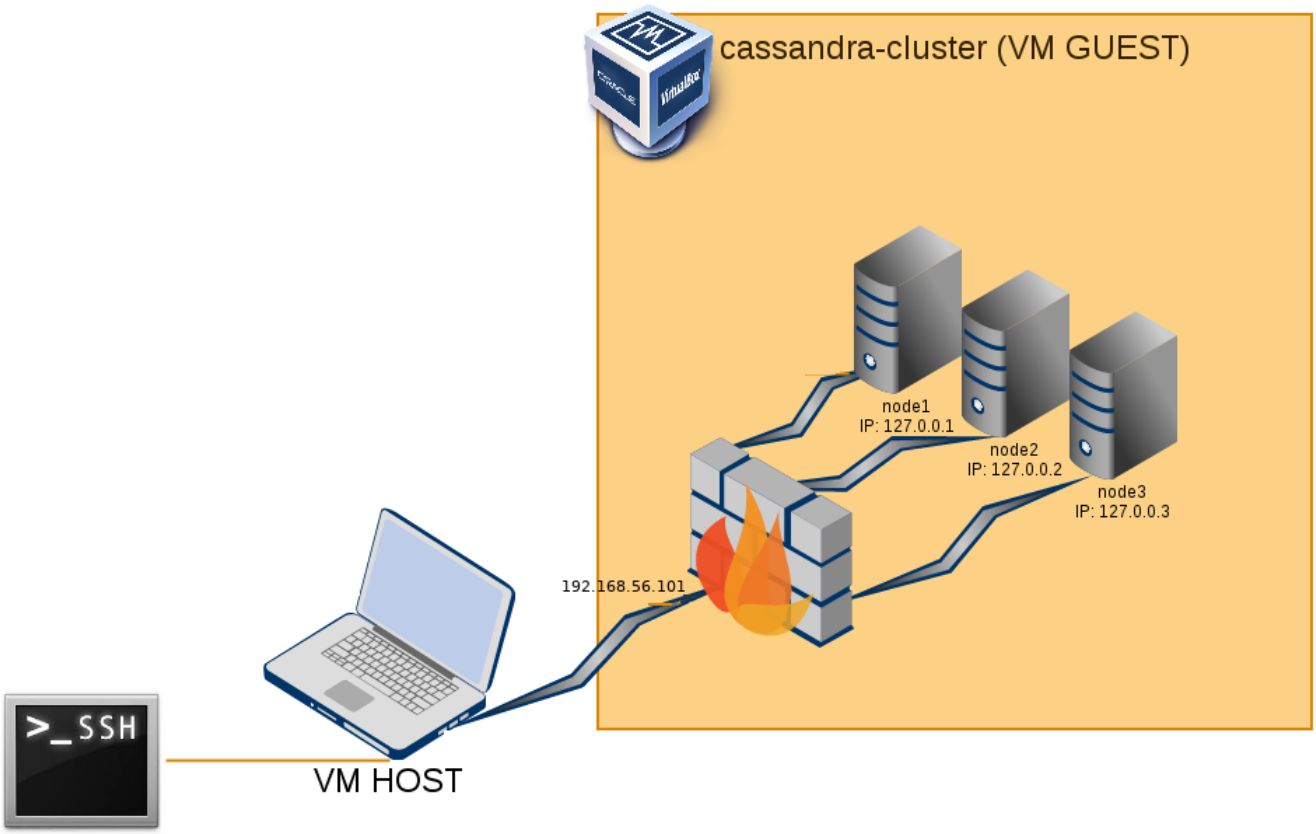

Figure 1. Environnement TP

Installation de la machine virtuelle

1. Telechargez la VM (https://drive.google.com/file/d/1jaxXAn1tLkEhSFUtFh3gfAwZ8Rp4Cylr/view?usp=sharing)
2. Importez cette VM dans votre VirtualBox
3. Lancez la VM
4. Notez l'adresse IP affichée lors du démarrage
5. Si vous avez eu une adresse IP, l'installation s'est bien passé. Félicitations :)

Dans nos exemples l'adresse IP de la VM est ***192.168.56.101***. ***N'oubliez pas de remplacer partout dans les exemples cette adresse par l'adresse affichée dans la console VirtualBox***:

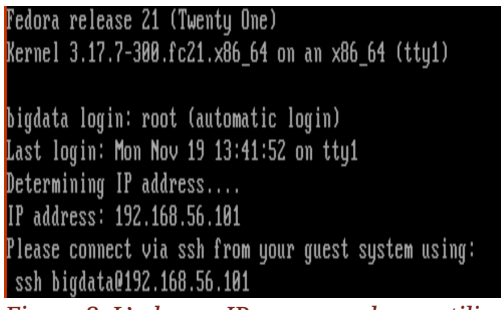

Figure 2. L'adresse IP que vous devez utiliser est affichée dans la fenêtre de la VM

Connexion en ssh sur la VM

Connectez vous en ssh sur la VM par exemple depuis votre terminal Linux:

```
[andrei@desktop ~]$ ssh bigdata@192.168.56.101
bigdata@192.168.56.101's password:
Last login: Sun Jan  4 14:53:32 2015 from pc12.home
[bigdata@bigdata ~]$
```

Identifiants de connexion:

- utilisateur: ***bigdata***
- password: ***bigdatafuret***

Si vous êtes sur Windows vous pouvez utiliser putty (http://www.putty.org/)

Gestion du cluster via CCM

Dans la VM on a pre-configuré un cluster ***test*** de 3 noeuds via CCM (https://github.com/pcmanus/ccm#usage). Ces noeuds sont configurés sur les adresses locales ***127.0.0.1***, ***127.0.0.2*** et ***127.0.0.3***. Les fichiers de configuration pour ce cluster sont dans le répertoire ***/home/bigdata/.ccm/test***

1) Lister les clusters installés sur votre machine. Notez le cluster actif qui est marqué par un *:

```
[bigdata@bigdata ~]$ ccm list
 *cassandra-2.1.16
  test
```

2) Activer le cluster *test*

```
[bigdata@bigdata ~]$ ccm list
 *cassandra-2.1.16
  test

[bigdata@bigdata ~]$ ccm switch test
[bigdata@bigdata ~]$ ccm list
  cassandra-2.1.16
 *test
```

3) Vérifier l'état du cluster active (***test***):

```
[bigdata@bigdata ~]$ ccm status

Cluster: 'test'

node1: DOWN
node3: DOWN
node2: DOWN
```

4) Démarrez le cluster

```
[bigdata@bigdata ~]$ ccm start
```

5) Vérifiez que le cluster a bien démarré et que les noeuds sont UP

BASH

```
[bigdata@bigdata ~]$ ccm status
Cluster: 'test'
---------------
node1: UP
node3: UP
node2: UP
```

6) Via ccm, affichez les paramètres (port, initial_token) du noeud 1

BASH

```
[bigdata@bigdata ~]$ ccm node1 show
node1: UP
       cluster=test
       auto_bootstrap=False
       thrift=('127.0.0.1', 9160)
       binary=('127.0.0.1', 9042)
       storage=('127.0.0.1', 7000)
       jmx_port=7100
       remote_debug_port=0
       initial_token=-9223372036854775808
       pid=1588
```

7) Exécutez la commande nodetool status (https://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsStatus.html) sur un noeud. Observez l'état des noeuds, la charge, la distribution des clés et les intervalles des tokens

BASH

```
[bigdata@bigdata ~]$ ccm node1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens       Owns (effective)  Host ID                               Rack
UN  127.0.0.1  165.81 KB  1            66.7%             ba6bce17-b815-426e-8535-bb30ebe52908  rack1
UN  127.0.0.2  153.53 KB  1            66.7%             a6fc393f-827a-4de3-88c5-9e718fde2b70  rack1
UN  127.0.0.3  158.85 KB  1            66.7%             0329e34d-e14b-4aa5-a736-d40304e8d942  rack1
```

Configuration du cluster Cassandra

8) Regarder dans la configuration des noeuds quelles sont les valeurs pour les principaux paramètres de configuration du cluster (conf/cassandra.yaml). Repérer les paramètres essentiels, et pour chaque paramètre expliquer à quoi il correspond et les valeurs usuelles (documentation (https://docs.datastax.com/en/cassandra/2.1/cassandra/configuration/configCassandra_yaml_r.html#configCassandra_yaml_r__commonProps))

- cluster_name
- listen_address
- seeds
- authenticator
- authorizer
- partitioner
- endpoint_snitch
- initial_token

BASH
```
[bigdata@bigdata ~]$ cat .ccm/test/node1/conf/cassandra.yaml | grep -E
'cluster_name|listen_address|seeds|authenticator|authorizer|partitioner|endpoint_snitch|initial_token'
authenticator: AllowAllAuthenticator
authorizer: AllowAllAuthorizer
cluster_name: test
endpoint_snitch: SimpleSnitch
initial_token: -9223372036854775808
listen_address: 127.0.0.1
partitioner: org.apache.cassandra.dht.Murmur3Partitioner
  - seeds: 127.0.0.1
```

Pour chaque noeud du cluster tous les fichiers (binaires/configuration) sont stockés dans le repertoire .ccm/{NOM_CLUSTEUR}/node{X}. Par exemple, pour afficher le contenu des parametres *param1* et *param2* du fichier de configuration du noeud1 du cluster *test* vous pouvez exécuter:

BASH
```
[bigdata@bigdata ~]$ cat .ccm/test/node1/conf/cassandra.yaml | grep -E 'param1|param2'
authenticator: AllowAllAuthenticator
authorizer: AllowAllAuthorizer
cluster_name: test
endpoint_snitch: SimpleSnitch
initial_token: -9223372036854775808
listen_address: 127.0.0.1
partitioner: org.apache.cassandra.dht.Murmur3Partitioner
  - seeds: 127.0.0.1
```

9) Quelles sont les intervalles de tokens pour chaque noeud du cluster ?

BASH
```
[bigdata@bigdata ~]$ ccm node1 ring

Datacenter: datacenter1
==========
Address    Rack        Status State   Load            Owns                Token
                                                                          3074457345618258602
127.0.0.1  rack1       Up     Normal  165.81 KB       66.67%              -9223372036854775808
127.0.0.2  rack1       Up     Normal  153.53 KB       66.67%              -3074457345618258603
127.0.0.3  rack1       Up     Normal  158.85 KB       66.67%              3074457345618258602


Node1 = [ -9223372036854775808, -3074457345618258603)
Node2 = [ -3074457345618258603,  3074457345618258602)
Node3 = [  3074457345618258602,  9223372036854775808)

Interval de tokens = -9223372036854775808, 9223372036854775807
Interval de tokens =                -2^63,              2^63-1
```

Cohérence dans Apache Cassandra

10) Arrêtez le cluster de test et vérifiez qu'il s'est bien arrêté. Nous voulons créer un nouveau cluster qui peut subir la perte de 2 noeuds et continuer à avoir une cohérence forte(immediate).

- Combien de noeuds nous sont nécessaires (on veut le minimum de noeuds) ?
- Quel sera le niveau de replication (RF) nécessaire ?
- Quel est le niveau de cohérence souhaité (ONE, ANY, QUORUM, ALL) pour les lectures et les écritures pour avoir une cohérence forte ?
- Quelles sont les proprietés de votre cluster en terme d'espace disque, latence et cohérence des données ?
- Créez et démarrez un tel cluster via CCM en utilisant la version 2.0.5 de Cassandra (pensez a arreter le cluster actif avant de creer un nouveau cluster !).

```
* Pour avoir une coherence forte il faut que apres la perte de 2 noeuds on arrive a avoir un quorum des replicas existants. Du coup 2 doit etre inferieur aux quorum => le quorum doit etre de minimum 3 noeuds => le nombre des replicas doit etre d'au moins 5 => le nombre de noeuds dans notre cluster doit etre au minimum 5

* RF = 5 (voir plus haut)

* Pour avoir la coherence forte on doit avoir W + R > RF => Si on prends un clusteur de 5 noeuds avec un RF de 5, le #replicas pour le QUORUM=3

On a plusieurs possibilitees :
W: One R: Quorum => 1 + 3 < 5 => on n'a pas de la coherence
W: Quorum R: Quorum => 3 + 3 > 5 => on a de la coherence
Toute autre variante avec W ou R en ALL satisfait aussi le besoin

* on a 5 noeuds avec un RF de 5 => chaque noeud contient la totalite des donnees. On peut configurer le niveau de coherence via le CONSISTENCY LEVEL => si on fais R=One et W=ALL on aura une latence reduite lors de lectures et une grande latence leurs des ecritures. Etc.

Si on a une partition qui isole 2 noeuds un client qui serait connecte a cet ilot ne pourrait plus avoir des reponse a ses requetes. Le clusteur se comporte donc comme un systeme CA.
[bigdata@bigdata ~]$ ccm stop  1
[bigdata@bigdata ~]$ ccm create test2 -v 2.0.5 -n 5 -s 1  2
Current cluster is now: test2
```

BASH

11) Traçage des requêtes dans *cqlsh*

- sur le cluster *test* lancez *cqlsh* (le shell de requêtage Cassandra) sur le *noeud1*
- executons quelques requêtes

```
  [bigdata@bigdata ~]$ ccm node1 cqlsh  1
Connected to test at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.0.15 | CQL spec 3.4.0 | Native protocol v4]
Use HELP for help.
cqlsh> show keyspaces
Improper show command.

cqlsh> CREATE KEYSPACE temperature WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 2};  2
cqlsh> USE temperature;
cqlsh:temperature> CREATE TABLE temperature1 (
                       ville text,
                       temperature int,
                       PRIMARY KEY (ville)
                   );
cqlsh:temperature>
cqlsh:temperature> INSERT INTO temperature1(ville, temperature ) VALUES ( 'Paris', 30);
cqlsh:temperature> INSERT INTO temperature1(ville, temperature ) VALUES ( 'Paris', 29);
cqlsh:temperature> INSERT INTO temperature1(ville, temperature ) VALUES ( 'Rennes', 30);
cqlsh:temperature>
cqlsh:temperature> SELECT * FROM temperature.temperature1;

 ville  | temperature
--------+-------------
  Paris |          29
 Rennes |          30

(2 rows)
```

BASH

- dans *cqlsh* activez le mode de traçage (TRACING) des requêtes, re-exécutez la requête precedente et observez les échanges entre les noeuds du cluster

BASH

```
cqlsh:temperature> TRACING ON
Now Tracing is enabled
cqlsh:temperature> select * from temperature.temperature1;

 ville  | temperature
--------+-------------
  Paris |          29
 Rennes |          30

(2 rows)

Tracing session: 02665450-ecd5-11e8-bfff-1f6784a5e176

 activity
| timestamp                  | source    | source_elapsed
-------------------------------------------------------------------------------------------------------------------
-------+----------------------------+-----------+----------------
                                                                                                        Execute CQL3
query | 2018-11-20 16:00:24.728000 | 127.0.0.1 |              0

                                                                        ...
```

- la commande CONSISTENCY de *cqlsh* permet de connaître le mode de cohérence active et de le changer pour la session en cours

BASH

```
cqlsh:temperature> CONSISTENCY
Current consistency level is ONE.
cqlsh:temperature> CONSISTENCY QUORUM
Consistency level set to QUORUM.
```

- utiliser CCM pour arrêter l'un des noeuds du cluster, rejouer quelques requêtes à un niveau de cohérence QUORUM, expliquer le résultat obtenu

☑ Afficher les reponses

BASH

```
Les reponses sont maintenant affichees !
```

Last updated 2018-11-20 16:51:30 CET