

# Formation administration et mise en place d'un cluster Cassandra

## Travaux pratiques

## Prérequis :

- Avoir Docker et Docker-Compose installés sur votre machine.
- Des connaissances basiques sur le fonctionnement des containers.

## Table des matières

Exercice 1 : Déployer un cluster Cassandra avec 1 seul node.....	3
Exercice 2 : Scaler horizontalement un cluster Cassandra.....	4
Exercice 3 : Création d'un nouveau role et un premier keyspace dans Cassandra.....	5
Exercice 4 : Préparation d'un profile du test avec cassandra-stress.....	6
Exercice 5 : Gestion des TTL .....	7
Exercice 6 : Gestion des Tombstones.....	8
Exercice 7 : HintedHandoff et rebuild d'un node.....	9
Exercice 8 : L'outil reaper .....	12
Exercice 9 : Replace d'un node.....	16
Exercice 10 : Mise en place du socle de graphage .....	18
Exercice 11 : Backup des données et restaure avec nodetool refresh .....	22
Exercice 12 : Configurer la sauvergarde incrémentale.....	24

## Exercice 1 : Déployer un cluster Cassandra avec 1 seul node

### Objectif :

L'objectif de cet exercice est d'installer et de configurer une instance de Cassandra en utilisant Docker Compose. À la fin de cet exercice, vous aurez une instance de Cassandra fonctionnelle accessible via Docker.

### Etapes à suivre

#### 1- Créer l'espace de travail :

- Ouvrez votre terminal et créez le répertoire cassandra-docker.
- Créez ensuite le fichier docker-compose.yml
- Editez le fichier docker-compose.yml pour instancier votre premier cluster Cassandra avec un seul node nommé cassandra1
  - Nom du container : cassandra1
  - *Nom du cluster : cluster1*
  - *Nom du host: cassandra1*
  - *Datacenter : dc1*
  - *Rack : rack1*
  - *Snitch : GossipingPropertyFileSnitch*

*Se référer à la documentation de l'image officielle pour la configuration du container :*

[https://hub.docker.com/\\_/cassandra](https://hub.docker.com/_/cassandra)

#### 2- Démarrer le docker-compose

- Exécutez la commande suivante pour démarrer votre premier cluster Cassandra

```
$ sudo docker-compose -f docker-compose.yml up -d
```

- Utilisez la commande suivante pour vérifier les logs dans Cassandra :

```
$ sudo docker-compose logs cassandra1 | grep 'Starting listening for CQL clients'
```

- Vérifier le démarrage du cluster :

```
$ sudo docker exec -it cassandra1 nodetool status
```

- Se connecter au cluster avec cqlsh

```
$ sudo docker exec -it cassandra1 cqlsh
```

## Exercice 2 : Scaler horizontalement un cluster Cassandra

### Objectif :

L'objectif de cet exercice est de rajouter 2 nouveaux nodes Cassandra au cluster déjà créé dans l'exercice 1.

### Etapes à suivre

- 1- Modification du docker-compose existant :
  - Ouvrez votre terminal et modifier le docker-compose.yml et ajoutez les 2 nouveaux nodes à votre cluster existant :
    - Node 2
      - Nom du node : cassandra2
      - Datacenter : dc2
      - Rack : rack1
    - Node 3
      - Nom du node : cassandra3
      - Datacenter : dc3
      - Rack : rack1

*Pensez à bien renseigner le **seed** pour les nouveaux nodes.*

*Vous pouvez vous baser sur le fichier docker-compose-three-nodes.yml fourni avec le support.*

- 1- Exécuter le docker-compose :

```
$ sudo docker-compose -f docker-compose.yml up -d
```

- Vérifier le démarrage des nouveaux nodes :

```
$ sudo docker-compose logs cassandra2 | grep 'Starting listening for CQL clients'
```

```
$ sudo docker exec -it cassandra1 nodetool status
```

## Exercice 3 : Création d'un nouveau role et un premier keyspace dans Cassandra

### Objectif :

L'objectif de cet exercice est d'interagir avec un cluster Cassandra existant en utilisant le CQL pour la création d'un nouveau role (user) et une base de données (keyspace) pour la suite des travaux pratiques.

### Etapes à suivre

- 1- Modifiez tout d'abord la conf de cassandra pour activer l'authentification :

```
$ sudo docker exec -it cassandra1 sed -i 's/authenticator: AllowAllAuthenticator/authenticator: PasswordAuthenticator/g' /etc/cassandra/cassandra.yaml
$ sudo docker exec -it cassandra1 sed -i 's/authorizer: AllowAllAuthorizer/authorizer: CassandraAuthorizer/g' /etc/cassandra/cassandra.yaml

$ sudo docker exec -it cassandra2 sed -i 's/authenticator: AllowAllAuthenticator/authenticator: PasswordAuthenticator/g' /etc/cassandra/cassandra.yaml
$ sudo docker exec -it cassandra2 sed -i 's/authorizer: AllowAllAuthorizer/authorizer: CassandraAuthorizer/g' /etc/cassandra/cassandra.yaml

$ sudo docker exec -it cassandra3 sed -i 's/authenticator: AllowAllAuthenticator/authenticator: PasswordAuthenticator/g' /etc/cassandra/cassandra.yaml
$ sudo docker exec -it cassandra3 sed -i 's/authorizer: AllowAllAuthorizer/authorizer: CassandraAuthorizer/g' /etc/cassandra/cassandra.yaml

$ sudo docker-compose -f docker-compose.yml restart
```

- Vérifier la configuration du keyspace en utilisant l'une des commandes suivantes :

```
$ sudo docker exec -it cassandra1 cqlsh
```

Vous arrivez à vous vous connecter avec cqlsh ?

```
select * from system_schema.keyspaces ;
```

```
desc keyspace system_auth ;
```

- 2- Modifiez la réplication du keyspace **system\_auth** pour que l'authentification dans Cassandra continue à fonctionner même si un node Cassandra est down.

- Quelle stratégie de réplication utiliser : *SimpleStrategy* ou *NetworkTopologyStrategy* ?
- Pour un RF de combien PAR DATACENTER ? 1, 2 ou 3 ?

```
ALTER KEYSPACE system_auth WITH REPLICATION={ 'class': '.....', 'first_dc': 'rf', 'second_dc': 'rf'..... } ;
```

3- Créez un nouveau role dans Cassandra :

- Nom du role : tp2024
- Mot de passe : tp2024
- Permissions : CREATE/MODIFY/SELECT ON ALL KEYSPACES

```
CREATE ROLE role_name WITH PASSWORD = 'password' AND LOGIN = true;  
  
GRANT permission_name ON resource TO role_name ;
```

4- Connectez-vous avec le nouveau user et créez le keyspace 'weather' avec un RF de 1 par datacenter :

```
$ sudo docker exec -it cassandra1 cqlsh -u tp2024 -p tp2024
```

```
CREATE keyspace weather WITH REPLICATION={ 'class': 'NetworkTopologyStrategy', 'dc1' : '1', 'dc2' : '1', 'dc3' : '1' } ;
```

## Exercice 4 : Préparation d'un profile du test avec cassandra-stress

### Objectif :

L'objectif de cet exercice est de créer un modèle de données dans Cassandra et l'alimenter avec des données générées avec l'outil cassandra-stress.

### Etapes à suivre :

- 1- Ouvrez le fichier tp-stress.yaml fourni avec le support et affichez son contenu.
- 2- Copiez le profile cassandra-stress dans votre répertoire du travail puis téléchargez cassandra-stress et lancez le stress en utilisant les commandes suivantes :

```
wget https://dlcdn.apache.org/cassandra/4.0.13/apache-cassandra-4.0.13-bin.tar.gz  
  
tar zxvf apache-cassandra-4.0.13-bin.tar.gz  
  
apache-cassandra-4.0.13/tools/bin/cassandra-stress  
  
--Récupérer l'adresse ip du node cassandra1 avec nodetool status et modifier la commande suivante avec le bon ip  
  
apache-cassandra-4.0.13/tools/bin/cassandra-stress user profile=tp-stress.yaml no-warmup cl=QUORUM ops\{(insert=1\  
n=100 -rate threads=1 -node x.x.x.x -mode native cql3 user=tp2024 password=tp2024
```

## Exercice 5 : Gestion des TTL

### Objectif :

L'objectif de cet exercice est de manipuler les TTL dans Cassandra, être capable d'insérer une donnée en précisant une durée de vie et de constater que la donnée disparaîtra de la base.

### Etapes à suivre

- 1- Insérer deux exemples de données avec un TTL d'une minute pour la première et un TTL de 2 minutes pour la seconde dans la table *weather\_data* précédemment créée par cassandra-stress.

Exemple d'insertion avec TTL :

```
# Insertion des données avec un TTL de 60 secondes dans une table exemple

$ sudo docker exec -it cassandra1 cqlsh -u tp2024 -p tp2024 -e " INSERT INTO
weather.weather_data (weather_station_id,year, day_of_year, time, sensor_id,
temperature) VALUES (123, 2024,1,toTimeStamp(now()),1,32) USING TTL 60"
```

- 2- Vérification des données

Pour vérifier les données avec TTL, vous pouvez utiliser des requêtes simples pour récupérer avant et expiration du TTL :

```
# Récupérer les données avant expiration

$ sudo docker exec -it cassandra1 cqlsh -u tp2024 -p tp2024 -e "SELECT * FROM weather.weather_data where
weather_station_id = 123 and year = 2024 "

$ sudo docker exec -it cassandra1 cqlsh -u tp2024 -p tp2024 -e "SELECT TTL(temperature) FROM weather.weather_data
where weather_station_id = 123 and year = 2024 "

#Attendre la durée du TTL et ensuite vérifier avec la même requête du SELECT
```

## Exercice 6 : Gestion des Tombstones

Objectif : Créer et gérer les tombstones dans Cassandra.

### Etapes à suivre

- 1- Importez des données dans la table *weather\_data* avec la commande *copy*. Pour cela, utilisez le fichier *weather\_data.csv* fourni avec le support.

```
$ sudo docker cp weather_data.csv cassandra1:/
```

```
$ sudo docker exec -it cassandra1 cqlsh -u tp2024 -p tp2024 -e "COPY weather.weather_data (weather_station_id, year, day_of_year, time, sensor_id, temperature, humidity) FROM 'weather_data.csv' "
```

- 2- Vérifiez le nombre de lignes importées :

```
$ sudo docker exec -it cassandra1 cqlsh -u cassandra -p cassandra -e "SELECT COUNT(*) FROM weather.weather_data where weather_station_id=9999 and year=2023"
```

- 3- Supprimez des données :

```
time for i in {400..1500}; do $ sudo docker exec -it cassandra1 cqlsh -u tp2024 -p tp2024 -e "DELETE FROM weather.weather_data where weather_station_id=9999 AND year=2023 AND day_of_year =$i"; done
```

- 4- Vérifiez du nouveau le nombre de lignes pour la partition :

```
$ sudo docker exec -it cassandra1 cqlsh -u tp2024 -p tp2024 -e "SELECT COUNT(*) FROM weather.weather_data where weather_station_id=9999 and year=2023"
```

- 5- Vérifiez les tombstones générés par Cassandra :

```
$ sudo docker-compose logs cassandra1 | grep 'tombstone cells'
```

```
cassandra1 | WARN [ReadStage-2] 2024-06-16 09:32:24,874 ReadCommand.java:536 - Read 100 live rows and 2202 tombstone cells for query SELECT * FROM weather.weather_data WHERE weather_station_id, year = 9999, 2023 LIMIT 100; token -8579015433728958466 (see tombstone_warn_threshold)
```



- 6- Supprimez définitivement les tombstones de la base
- Commencez par modifier la valeur du `gc_grace_seconds`

```
#Noter la valeur actuelle du gc_grace_seconds

$ sudo docker exec -it cassandra1 cqlsh -u tp2024 -p tp2024 -e "DESC TABLE weather.weather_data"

# Alter du gc_grace_seconds

$ sudo docker exec -it cassandra1 cqlsh -u tp2024 -p tp2024 -e "ALTER TABLE weather.weather_data WITH
gc_grace_seconds=1"
```

- Lancez une compaction forcée

```
$ sudo docker exec -it cassandra1 nodetool compact weather weather_data
```

- Relancez la commande `cql` et vérifiez que les tombstones ont été supprimés définitivement :

```
$ sudo docker exec -it cassandra1 cqlsh -u cassandra -p cassandra -e "SELECT COUNT(*) FROM weather.weather_data
where weather_station_id=9999 and year=2023"
```

## Exercice 7 : HintedHandoff et rebuild d'un node

Objectif : Observer les HH générés par les nodes Cassandra et reconstruction d'un node avec l'aide de la commande `rebuild`.

Etapes à suivre :

- Générez de l'activité dans le cluster Cassandra en utilisant `cassandra-stress`

```
apache-cassandra-4.0.13/tools/bin/cassandra-stress user profile=tp-
stress.yaml duration=30m no-warmup cl=QUORUM ops\ (insert=1\ ) -rate
threads=1 -node x.x.x.x -mode native cql3 user=tp2024 password=tp2024
```

- Depuis une 2ème session, arrêtez le node `cassandra3` :

```
$ sudo docker ps
--noter le container_id du node cassandra3
$ sudo docker stop cassandra3_container_id
```

- 3- Confirmez que le node a été bien arrêté et constatez l'apparition des HH depuis les autres nodes du cluster :

```
$ sudo docker exec -it cassandra1 nodetool status

$ sudo docker exec -it cassandra1 ls /opt/cassandra/data/hints

$ sudo docker exec -it cassandra2 ls /opt/cassandra/data/hints
```

- 4- Démarrez du nouveau le node arrêté, confirmez qu'il est du nouveau dans cluster et vérifiez le dépliement des HH :

```
$ sudo docker-compose -f docker-compose.yml up -d

$ sudo docker exec -it cassandra1 nodetool status

$ sudo docker-compose logs cassandra1 | grep 'Finished hinted handoff'
$ sudo docker-compose logs cassandra2 | grep 'Finished hinted handoff'

cassandra1 | INFO [HintsDispatcher:3] 2024-06-16 09:38:56,843
HintsDispatchExecutor.java:288 - Finished hinted handoff of file db3f7974-
4976-436f-b8d5-0dbc80b187ba-1718530611822-2.hints to endpoint
/172.24.0.4:7000: db3f7974-4976-436f-b8d5-0dbc80b187ba

$ sudo docker exec -it cassandra1 ls /opt/cassandra/data/hints

$ sudo docker exec -it cassandra2 ls /opt/cassandra/data/hints
```

- 5- Rebuild d'un node

Arrêtez du nouveau le node cassandra3 – Ici on va simuler l'arrêt avec un disablegossip :

```
$ sudo docker exec -it cassandra3 nodetool disablegossip

Datacenter: dc3
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens   Owns (effective)  Host ID
Rack
DN 172.24.0.4   605.17 KiB    128      100.0%             db3f7974-4976-436f-
b8d5-0dbc80b187ba rack1
```

Ici, on va supposer que cassandra3 est resté 'down ' pendant une période supérieure à la valeur du paramètre :

```
# this defines the maximum amount of time a dead host will have hints
# generated. After it has been dead this long, new hints for it will not be
# created until it has been seen alive and gone down again.

# Min unit: ms

max_hint_window: 3h
```

Donc, le node a besoin d'être resynchronisé du nouveau lors de l'intégration dans le cluster. Une des options serait d'utiliser la commande nodetool rebuild.

Cassandra3 étant toujours down, supprimer l'ensemble des données dans le répertoire data de cassandra :

```
$ sudo docker exec -it cassandra3 sh

# cd /opt/cassandra/data
# ls
commitlog  data  hints  saved_caches
# cd /opt/cassandra/data/data
# ls
system  system_auth  system_distributed  system_schema  system_traces
weather
--supprimer le dossier du keyspace weather
# rm -rf weather

# ls
system  system_auth  system_distributed  system_schema  system_traces
```

Démarrez du nouveau cassandra3

```
$ $ sudo docker exec -it cassandra3 nodetool enablegossip

$ $ sudo docker stop container_id

$ $ sudo docker start container_id

Datacenter: dc3
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens   Owns (effective)  Host ID
Rack
UN  172.24.0.4   605.17 KiB    128      100.0%            db3f7974-4976-436f-
b8d5-0dbc80b187ba  rack1

--Checker le dossier data du ks weather_data
--Penser à modifier l'uuid de de la table si besoin

$ $ sudo docker exec -it cassandra3 ls
/opt/cassandra/data/data/weather/weather_data-
d19e7c602bbe11efb66581f23dc2900a

backups
```

Lancez le rebuild

```
$ sudo docker exec -it cassandra3 nodetool rebuild -dc dc1 -ks weather
```

```
sudo docker exec -it cassandra3 ls /opt/cassandra/data/data/weather/weather_data-d19e7c602bbe11efb66581f23dc2900a
nb-1-big-Digest.crc32  nb-1-big-Statistics.db  nb-2-big-Data.db  nb-2-big-Index.db
nb-1-big-Filter.db    nb-1-big-Summary.db    nb-2-big-Digest.crc32  nb-2-big-Statistics.db
nb-1-big-Index.db     nb-2-big-CompressionInfo.db  nb-2-big-Filter.db  nb-2-big-Summary.db
```

## Exercice 8 : L'outil reaper

Objectif : Installer et configurer l'outil cassandra-reaper pour la programmation des opérations de repair dans un cluster Cassandra.

Etapes à suivre :

- 1- Modifiez le docker-compose.yml pour ajouter le service cassandra-reaper (Se référer au fichier *docker-compose-three-nodes-with-reaper.yml* fourni avec le support):

*reaper-in-memory:*

*image:* thelastpickle/cassandra-reaper:latest

*container\_name:* reaper-in-memory

*environment:*

- REAPER\_STORAGE\_TYPE=memory

*ports:*

- "8080:8080"

- "8081:8081"

*volumes:*

- ./data/reaper:/var/log/cassandra-reaper

Puis, démarrer le nouveau container :

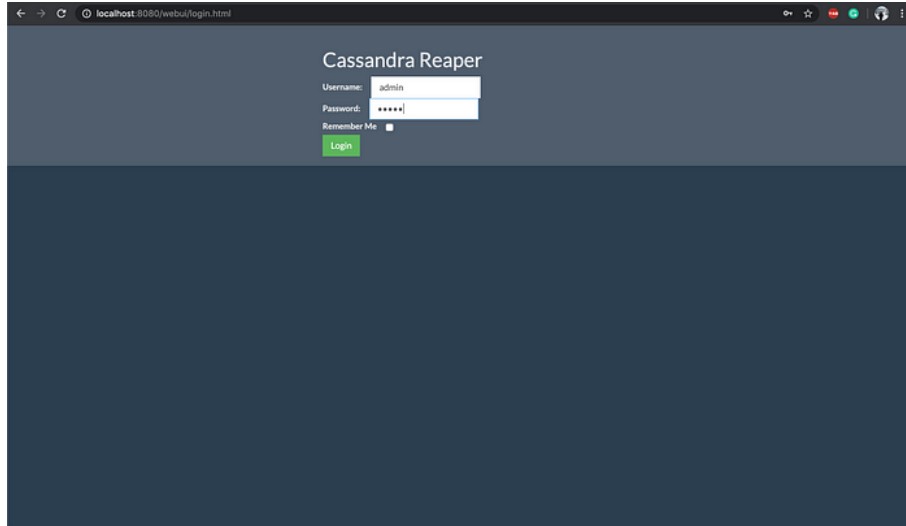
```
$ sudo docker-compose -f docker-compose.yml up -d reaper-in-memory
```

Vérifiez le bon démarrage du service :

```
$ sudo docker-compose logs -f reaper-in-memory
```

- 2- Une fois le service est démarré, accédez au reaper Web UI via l'url <http://localhost:8080/webui>

L'authentification est activée par défaut, le user/password par défaut => admin/admin



- 3- Ajoutez un nouveau cluster Cassandra depuis le menu Clusters

\*seed node => L'adresse d'un node Cassandra pour la découverte du cluster.

\*port JMX => 7199 pour la valeur par défaut

Pour des raisons de sécurité, le JMX est par défaut accessible uniquement depuis le *localhost*

```
# Cassandra ships with JMX accessible *only* from localhost.
# To enable remote JMX connections, uncomment lines below
# with authentication and/or ssl enabled. See https://wiki.apache.org/cassandra
#
if [ "x$LOCAL_JMX" = "x" ]; then
    LOCAL_JMX=yes
fi
```

Reaper qui communique avec Cassandra à travers le JMX a besoin que ce port soit exposé de l'extérieur aussi.

*Pensez à configurer la sécurité pour le port JMX lorsque vous modifiez cette variable en production 😊*

Ici, nous rajoutons dans le fichier de configuration `cassandra-env.sh` la ligne :

`LOCAL_JMX=no`

```
$ sudo docker exec -it cassandra1 sed -i '/JMX_PORT="7199"/i \LOCAL_JMX=no' /etc/cassandra/cassandra-env.sh
```

```
$ sudo docker exec -it cassandra1 sed -i '/JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.authenticate=true"/s/^/#/' /etc/cassandra/cassandra-env.sh
```

```
$ sudo docker exec -it cassandra2 sed -i '/JMX_PORT="7199"/i \LOCAL_JMX=no' /etc/cassandra/cassandra-env.sh
```

```
$ sudo docker exec -it cassandra2 sed -i '/JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.authenticate=true"/s/^/#/' /etc/cassandra/cassandra-env.sh
```

```
$ sudo docker exec -it cassandra3 sed -i '/JMX_PORT="7199"/i \LOCAL_JMX=no' /etc/cassandra/cassandra-env.sh
```

```
$ sudo docker exec -it cassandra3 sed -i '/JVM_OPTS="$JVM_OPTS -Dcom.sun.management.jmxremote.authenticate=true"/s/^/#/' /etc/cassandra/cassandra-env.sh
```

```
$ sudo docker-compose restart
```

Add Cluster ▾

Seed node\*

172.21.0.4

JMX port

7199

JMX username

JMX username of cluster (optional)

JMX password

JMX password of cluster (optional)

Add Cluster

Filter:

Start typing to filter

Start typing to filter

cluster1

Nodes: 3

Total load: 680.4 kB

Running repairs: 0

Forget cluster Info

dc1 223.8 kB	dc2 238.1 kB	dc3 218.5 kB
rack1 223.8 kB	rack1 238.1 kB	rack1 218.5 kB

4- Une fois le cluster est rajouté, créer un nouveau repair :

Enfin, activez le repair qui vient d'être crée :

Start	ETA	State	Cluster	Keyspace	Tables	Repaired	
		NOT_STARTED	cluster1	weather	1 table	0 / 248	<a href="#">View segments</a> <a href="#">Activate</a> <a href="#">Delete</a>

Le cycle du repair peut être monitoré depuis l'interface Repair :

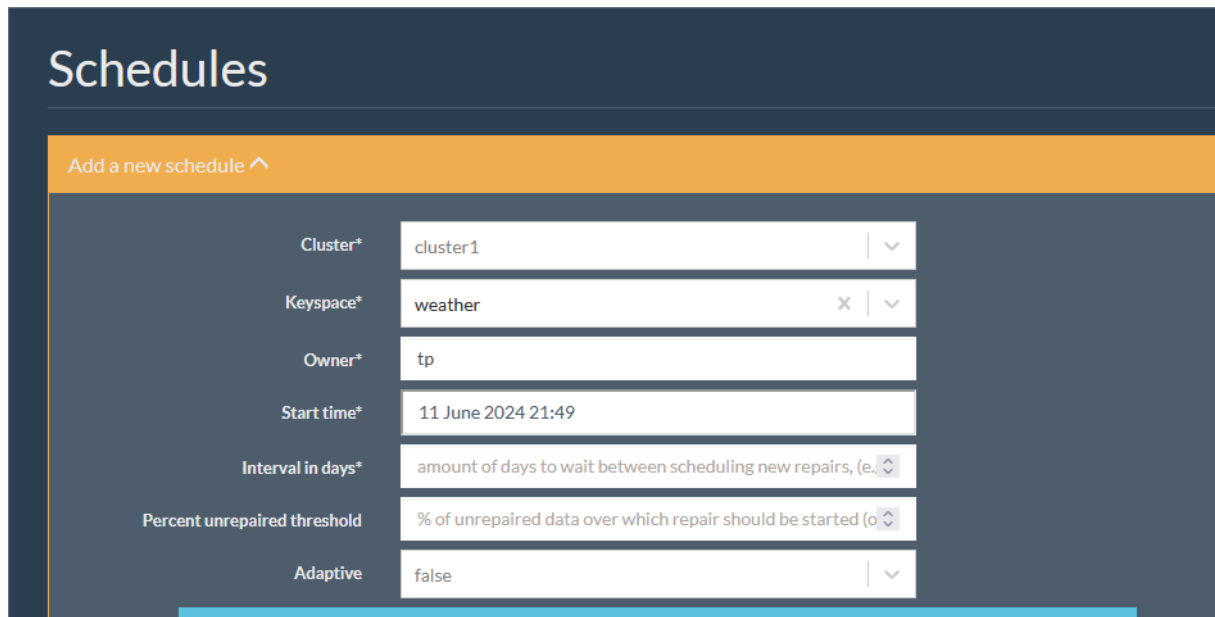
Start	ETA	State	Cluster	Keyspace	Tables	Repaired	
11 juin 2024 21:48	TBD	RUNNING	cluster1	weather	1 table	3 / 248	<a href="#">View segments</a> <a href="#">Stop</a>

5- Scheduler un repair

Pour créer un nouveau schedule pour le repair des données dans cluster, aller dans l'interface Schedules :

## Schedules

Et créer un nouveau schedule :



The screenshot shows a web interface for configuring schedules. The title 'Schedules' is at the top. Below it is a button 'Add a new schedule' with an upward arrow. The form contains several fields:

Field	Value
Cluster*	cluster1
Keyspace*	weather
Owner*	tp
Start time*	11 June 2024 21:49
Interval in days*	amount of days to wait between scheduling new repairs, (e.g. 1)
Percent unrepared threshold	% of unrepared data over which repair should be started (0-100)
Adaptive	false

## Exercice 9 : Replace d'un node

Objectif : Remplacer cassandra3 après l'avoir stopper par le nouveau node cassandra4 portant la même @ip.

Etapes à suivre :

- 1- Commencez par noter l'@ ip du node cassandra3 puis l'arrêter :

```
$ sudo docker-compose stop cassandra3  
$ sudo docker-compose rm cassandra3
```

Avec un nodetool status, confirmez que cassandra3 est down et notez son Host ID



```

Datacenter: dc1
=====
Status=Up/Down
[/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens     Owns (effective)  Host ID                               Rack
UN 172.21.0.2    265.69 KiB    128        100.0%           e3504bae-d15e-455d-af94-43ca0c4833ac rack1

Datacenter: dc2
=====
Status=Up/Down
[/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens     Owns (effective)  Host ID                               Rack
UN 172.21.0.3    280.03 KiB    128        100.0%           726bdaa5-2a41-4480-bc25-2608b4bf9dc6 rack1

Datacenter: dc3
=====
Status=Up/Down
[/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens     Owns (effective)  Host ID                               Rack
DN 172.21.0.5    271.27 KiB    128        100.0%           83ef38e2-a44d-47e3-8eec-18739d2dc809 rack1

```

## 2- Mettre à jour le fichier docker-compose.yml

Mettez à jour votre fichier docker-compose.yml et remplacez cassandra3 avec le nouveau node cassandra4 avec l'option de configuration personnalisée cassandra-env.sh :

```
JVM_OPTS="$JVM_OPTS -Dcassandra.replace_address=address_of_dead_node
```

Voir Fichiers fournis avec le support :

- docker-compose-replace-node.yml
- cassandra-env.sh

## 3- Démarrez le nouveau node

```
$ sudo docker-compose up -d cassandra4
```

Avec un nodetool status, confirmez le bon démarrage de cassandra4 et comparez son Host ID avec celui de l'ancien node cassandra3 :

```

Datacenter: dc1
=====
Status=Up/Down
[/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens     Owns (effective)  Host ID                               Rack
UN 172.21.0.2    294.34 KiB    128        100.0%           e3504bae-d15e-455d-af94-43ca0c4833ac rack1

Datacenter: dc2
=====
Status=Up/Down
[/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens     Owns (effective)  Host ID                               Rack
UN 172.21.0.3    308.68 KiB    128        100.0%           726bdaa5-2a41-4480-bc25-2608b4bf9dc6 rack1

Datacenter: dc3
=====
Status=Up/Down
[/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens     Owns (effective)  Host ID                               Rack
UN 172.21.0.5    260.44 KiB    128        100.0%           83ef38e2-a44d-47e3-8eec-18739d2dc809 rack1

```

## Exercice 10 : Mise en place du socle de graphage

Objectif : Configurer un environnement où les métriques de Cassandra sont collectées via *jmx\_exporter*, puis scrapées par *Prometheus*, et enfin visualisées dans *Grafana*.

### Étapes à suivre :

- 1- Préparer les fichiers de configuration

*docker-compose-monitoring.yaml*

Ce fichier définira les services pour Cassandra, jmx\_exporter, Prometheus et Grafana.

*jmx\_exporter/config.yml*

Ce fichier configure jmx\_exporter pour collecter les métriques de Cassandra.

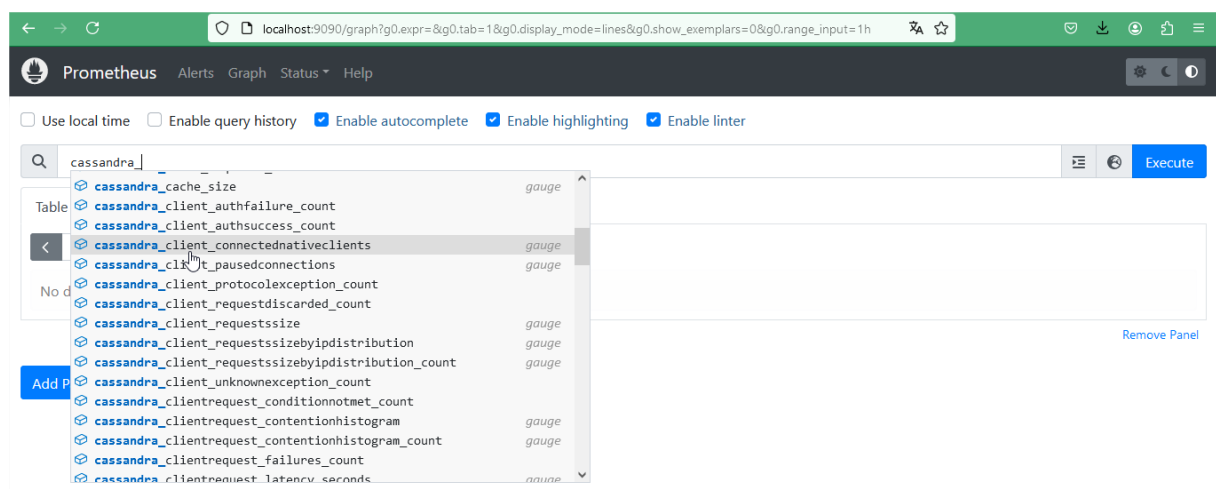
- 2- Démarrer les services avec Docker Compose

Dans le répertoire contenant les fichiers de configuration, exécutez la commande suivante (Assurez-vous que les autres nodes cassandra ont été arrêtés avant de lancer la commande suivante)

```
$ sudo docker-compose -f docker-compose-monitoring.yaml up -d
```

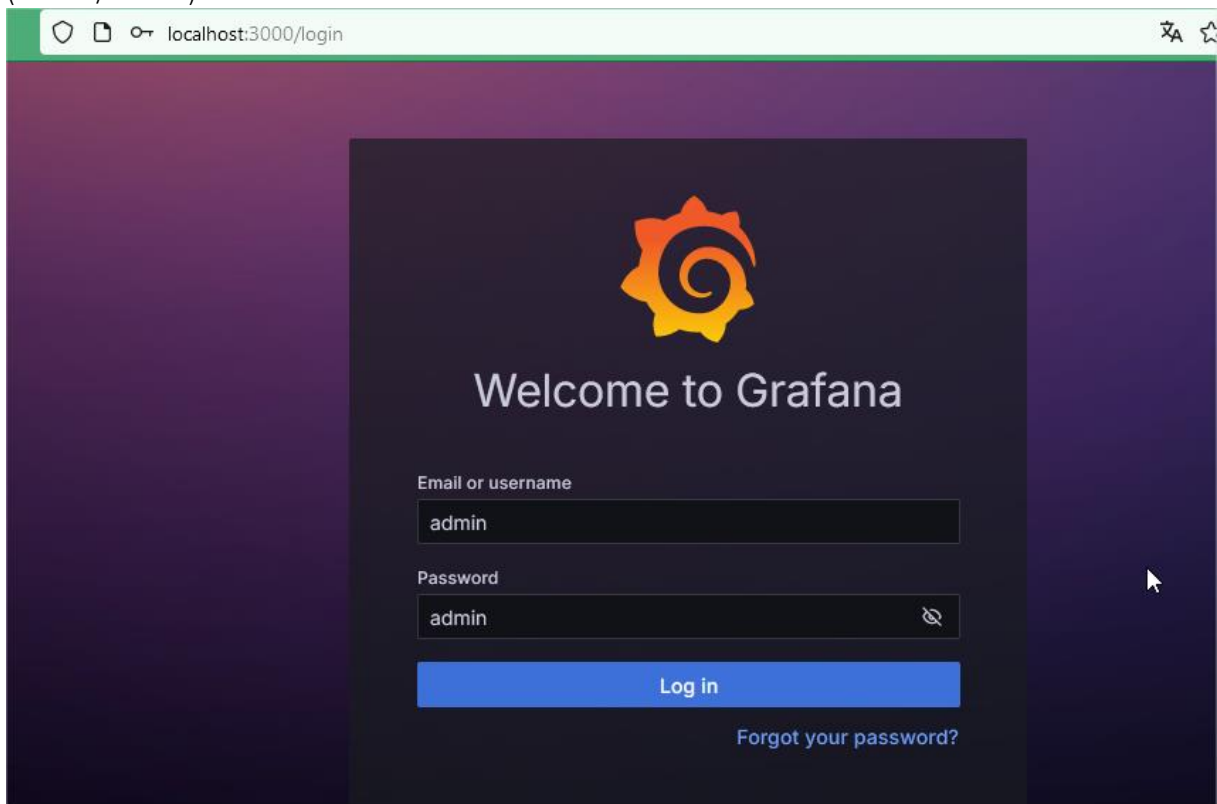
- 3- Vérifier les différents services lancés

Ouvrez l'interface du Prometheus depuis l'url <http://localhost:9090> et confirmer la présence des métriques Cassandra :



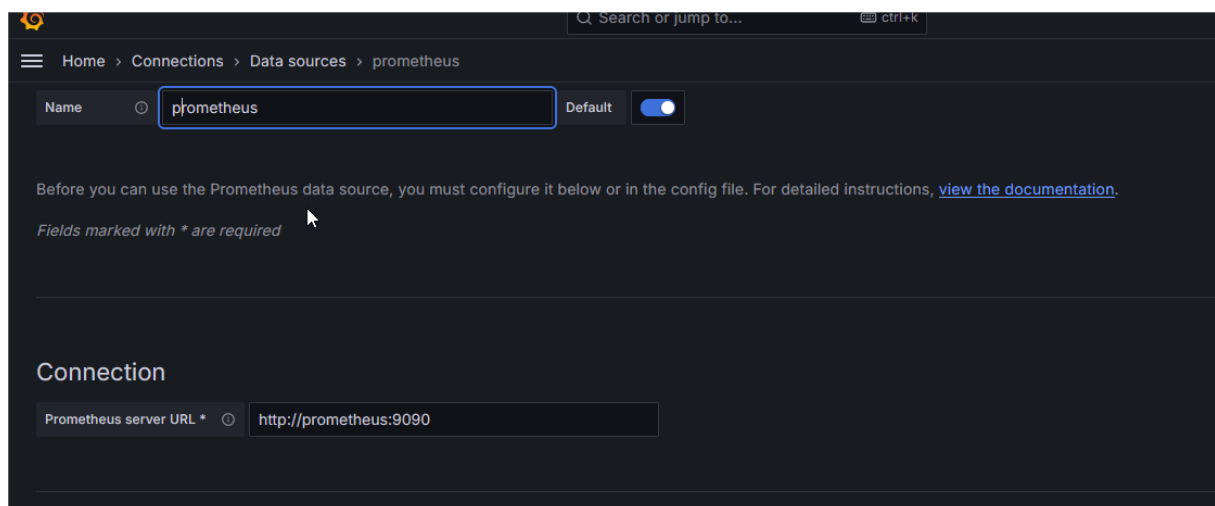
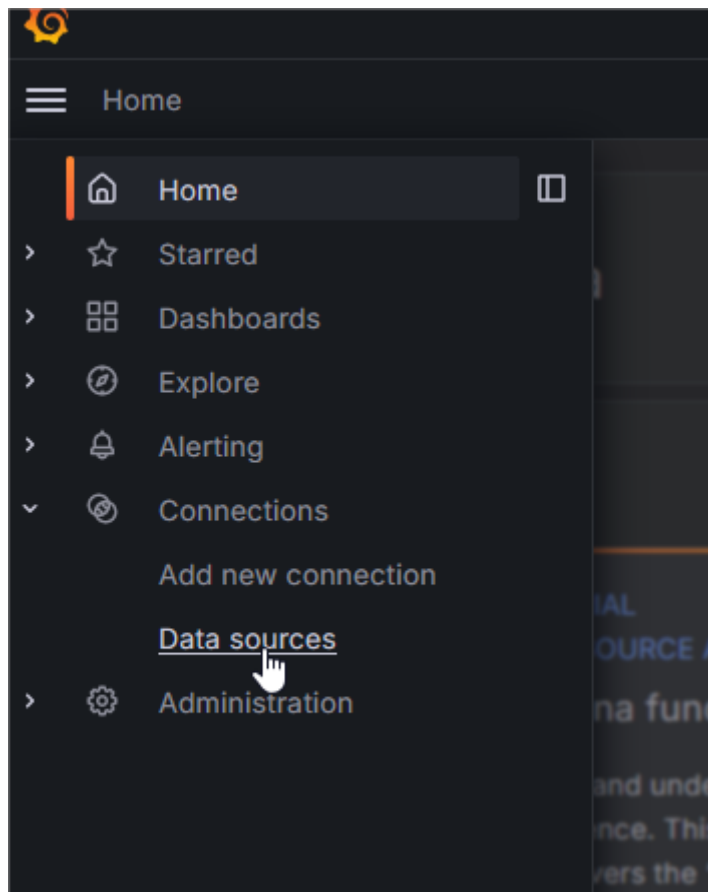
#### 4- Configurer Grafana

Accédez à Grafana à l'adresse <http://localhost:3000> et Connectez-vous avec les identifiants par défaut (admin / admin).



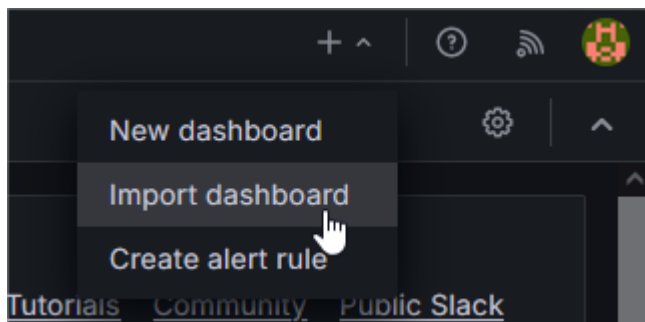
Une fois connecté, allez dans Data Source et Ajoutez Prometheus comme source de données :

- Nom : Prometheus
- URL : <http://prometheus:9090>



5- Ajouter un nouveau dashboard

Depuis le menu en haut à droite, appuyez sur 'import dashboard'



Puis, entrez l'id du dashboard Cassandra

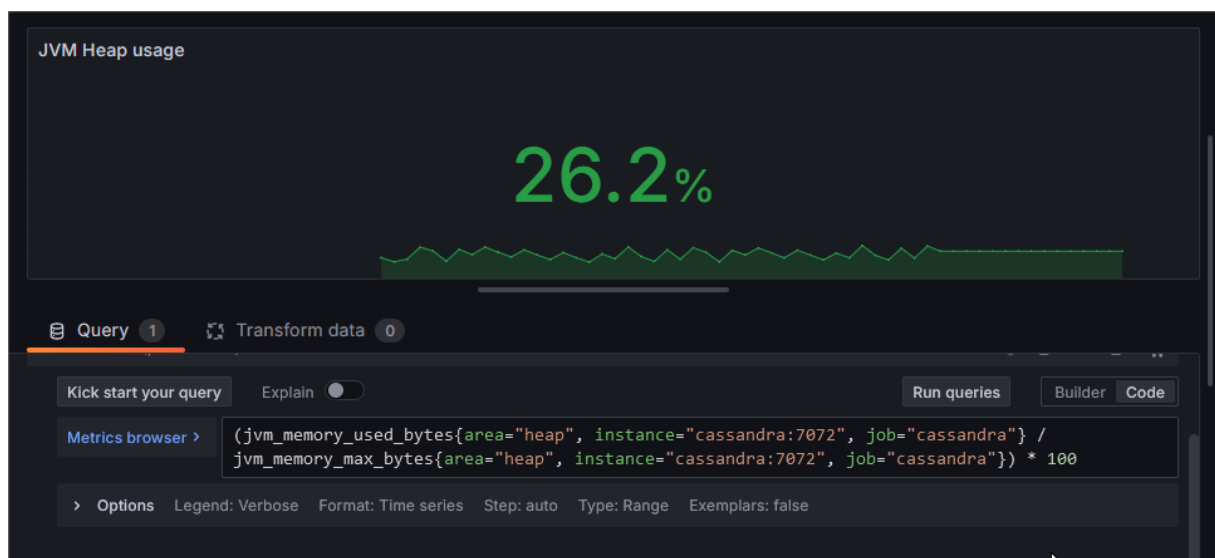
A screenshot of the 'Import dashboard' form in Grafana. It has a dark background. At the top, it says 'Find and import dashboards for common applications at [grafana.com/dashboards](https://grafana.com/dashboards)'. Below this is a text input field labeled 'Grafana.com dashboard URL or ID' and a blue 'Load' button.

Liste des dashboard Cassandra accessible ici =>  
<https://grafana.com/grafana/dashboards/?search=cassandra>

Dans notre exemple, on va upload le dashboard d'id 5408

*Note : Parcourez et modifiez avec le formateur le dashboard afin d'adapter et/ou corriger les erreurs dans le dashboard uploadé.*

Exemple d'un graphe à corriger avec le formateur :



## Exercice 11 : Backup des données et restaure avec nodetool refresh

### Etapes à suivre :

- 1- Sauvegarder les données de Cassandra

Démarrez un nouveau container Cassandra :

```
$ sudo docker run --name my-cassandra -d cassandra:4.0.13
```

Attendez que Cassandra soit passé en UP et générer des données dans le cluster en exécutant les commandes suivantes :

```
--Attendre que le node soit démarrée
$ sudo docker exec -it my-cassandra nodetool status

--Créer le keyspace
$ sudo docker exec -it my-cassandra cqlsh -e "CREATE KEYSPACE my_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};"

--Créer la table
$ sudo docker exec -it my-cassandra cqlsh -e "CREATE TABLE my_keyspace.users ( user_id UUID PRIMARY KEY, name text, email text );"

--Insérer des données

$ sudo docker exec -it my-cassandra cqlsh -e "INSERT INTO my_keyspace.users (user_id, name, email) VALUES (uuid(), 'John Doe', 'john.doe@example.com');"
$ sudo docker exec -it my-cassandra cqlsh -e "INSERT INTO my_keyspace.users (user_id, name, email) VALUES (uuid(), 'Jane Smith', 'jane.smith@example.com');"
```

Lancez la commande nodetool suivante pour créer une sauvegarde :

```
$ sudo docker exec -it my-cassandra nodetool snapshot -t backup_tp my_keyspace.users

--listsnapshot

$ sudo docker exec -it my-cassandra nodetool listsnapshots

--Vérifier le dossier data
--Penser à modifier l'uuid de la table users- ???????
$ sudo docker exec -it my-cassandra ls -l /var/lib/cassandra/data/my_keyspace/users-9efd30a029b411efbf6557d50806a80b/snapshots/backup_tp
```

## 2- Restaurer les données avec nodetool refresh

Suivre la procédure suivante pour restaurer les données dans la table users

```
--Truncate de la table
$ sudo docker exec -it my-cassandra cqlsh -e "TRUNCATE TABLE my_keyspace.users;"

$ sudo docker exec -it my-cassandra cqlsh -e "SELECT * FROM my_keyspace.users;"

user_id | email | name
-----+-----+-----

(0 rows)

--Copie des fichiers de sauvegarde
$ sudo docker exec -it my-cassandra sh

$ cp /var/lib/cassandra/data/my_keyspace/users-9efd30a029b411efbf6557d50806a80b/snapshots/backup_tp/*
/var/lib/cassandra/data/my_keyspace/users-9efd30a029b411efbf6557d50806a80b/

$ chown cassandra:cassandra /var/lib/cassandra/data/my_keyspace/users-
9efd30a029b411efbf6557d50806a80b/*

$ exit

--Lancer le nodetool refresh
$ sudo docker exec -it my-cassandra nodetool refresh my_keyspace users

$ sudo docker exec -it my-cassandra cqlsh -e "SELECT * FROM my_keyspace.users;"

user_id | email | name
-----+-----+-----
ba3ae279-3ae2-4472-82fb-7cfe98f871d4 | john.doe@example.com | John Doe
469b92bb-ed62-4398-9ddl-b43b2c1e66e1 | jane.smith@example.com | Jane Smith
```

## Exercice 12 : Configurer la sauvegarde incrémentale

La sauvegarde incrémentale de Cassandra vous permet de sauvegarder uniquement les nouvelles données ou les modifications apportées depuis la dernière sauvegarde complète.

### Étapes à suivre :

- 1- Activer les sauvegardes incrémentales

Editez le fichier 'cassandra.yaml' pour activer l'option *incremental\_backup*

```
$ sudo docker exec -it my-cassandra sed -i 's/incremental_backups: false/incremental_backups: true/g' /etc/cassandra/cassandra.yaml
```

Redémarrez le conteneur pour appliquer les modifications :

```
$ sudo docker restart my-cassandra
```

- 2- Prendre une sauvegarde incrémentale

Maintenant que les sauvegardes incrémentales sont activées, Cassandra créera automatiquement des fichiers de sauvegarde incrémentale chaque fois qu'une SSTable est modifiée. Pour créer une sauvegarde incrémentale manuellement, vous pouvez utiliser `nodetool flush` pour forcer l'écriture des données en mémoire sur disque.

--Insérer une nouvelle ligne dans la table

```
$ sudo docker exec -it my-cassandra cqlsh -e "INSERT INTO my_keyspace.users (user_id, name, email) VALUES (uuid(), 'Test Backup Incr', 'incr@example.com');"
```

--Noter l'uuid du dernier user ajouté :

```
$ sudo docker exec -it my-cassandra cqlsh -e "SELECT * FROM my_keyspace.users;"
```

user_id	email	name
91e57a70-0f20-4a35-8399-d50c6da9bc83	incr@example.com	Test Backup Incr

\$ exit

--flush des données :

```
$ sudo docker exec -it my-cassandra nodetool flush
```



### 3- Vérifiez les sauvegardes incrémentales dans le répertoire de données :

Les fichiers de sauvegarde incrémentale seront stockés dans le répertoire de données de Cassandra sous chaque keyspace et table, dans un sous-répertoire nommé backups.

```
$ sudo docker exec -it my-cassandra ls -l /var/lib/cassandra/data/my_keyspace/users-58f258e02c0911ef90616d2e6fdb8536/backups
total 36
-rw-r--r-- 2 cassandra cassandra 47 Jun 16 21:29 nb-3-big-CompressionInfo.db
-rw-r--r-- 2 cassandra cassandra 79 Jun 16 21:29 nb-3-big-Data.db
-rw-r--r-- 2 cassandra cassandra 9 Jun 16 21:29 nb-3-big-Digest.crc32
-rw-r--r-- 2 cassandra cassandra 16 Jun 16 21:29 nb-3-big-Filter.db
-rw-r--r-- 2 cassandra cassandra 20 Jun 16 21:29 nb-3-big-Index.db
-rw-r--r-- 2 cassandra cassandra 4759 Jun 16 21:29 nb-3-big-Statistics.db
-rw-r--r-- 2 cassandra cassandra 92 Jun 16 21:29 nb-3-big-Summary.db
-rw-r--r-- 2 cassandra cassandra 92 Jun 16 21:29 nb-3-big-TOC.txt
```

### 4- Restaurer la sauvegarde incrémentale

Suivez les étapes suivantes pour simuler et test le restaure des données backupées avec l'option incrémentale :

```
--Truncate de la table
$ sudo docker exec -it my-cassandra cqlsh -e "TRUNCATE TABLE my_keyspace.users;"

$ sudo docker exec -it my-cassandra cqlsh -e "SELECT * FROM my_keyspace.users;"

user_id | email | name
-----+-----+-----

(0 rows)

--Copie des fichiers de sauvegarde

$ sudo docker exec -it my-cassandra /bin/bash

$ cp /var/lib/cassandra/data/my_keyspace/users-58f258e02c0911ef90616d2e6fdb8536/backups/*
/var/lib/cassandra/data/my_keyspace/users-58f258e02c0911ef90616d2e6fdb8536/

$ chown cassandra:cassandra /var/lib/cassandra/data/my_keyspace/users-58f258e02c0911ef90616d2e6fdb8536/*

$ exit

--restart du container cassandra
$ sudo docker restart my-cassandra
--Attendre que cassandra soit du nouveau up
$ sudo docker exec -it my-cassandra cqlsh -e "SELECT * FROM my_keyspace.users;"

user_id | email | name
-----+-----+-----
91e57a70-0f20-4a35-8399-d50c6da9bc83 | incr@example.com | Test Backup Incr
```

*Notez qu'avec un restaure depuis un backup incrémental, nous ne disposons pas de l'intégrité des données, d'où le besoin de coupler à un restaure incrémental, un restaure de données complète depuis un snapshot full des données.*