# Cassandra Distributed – was implemented using cqlsh and Docker

## 1) Сконфігурувати кластер з 3-х нод

Для запуску і налаштування кластеру з 3-з нод використав Docker. Команди можна знайти в файлі DockerCommands.txt, що лежить в папці ./Task7.

## 2) Перевірити правильність конфігурації за допомогою *nodetool status*

**Command:**

docker exec cassandra-3 nodetool status

**Result:**

```
root@OleksiiLP:~/gitrepo/ditributed_final# docker exec cassandra-3 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.17.0.4  75.21 KiB  16      76.0%             4c23028b-b9e4-4432-a179-0a0ce594039d  rack1
UN  172.17.0.3  75.19 KiB  16      59.3%             08217b34-a327-4969-ab5e-91cb8a13cc83  rack1
UN  172.17.0.2  109.4 KiB  16      64.7%             7b5aa66b-ab9e-4057-8350-393c9049e300  rack1
```

## 3) Використовуючи *cqlsh*, створити три *Keyspace* з replication factor 1, 2, 3

**Command:**

CREATE KEYSPACE cas_rep3 WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'datacenter1' : 3};

CREATE KEYSPACE cas_rep2 WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'datacenter1' : 2};

CREATE KEYSPACE cas_rep1 WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'datacenter1' : 1};

DESCRIBE keyspaces;

**Result:**

```
cqlsh> CREATE KEYSPACE cas_rep3 WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'datacenter1' : 3};
cqlsh> CREATE KEYSPACE cas_rep2 WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'datacenter1' : 2};
cqlsh> CREATE KEYSPACE cas_rep1 WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'datacenter1' : 1};
cqlsh> DESCRIBE keyspaces;

cas_rep1  cas_rep3  system_auth         system_schema  system_views
cas_rep2  system    system_distributed  system_traces  system_virtual_schema
```

4) **В кожному з кейспейсів створити таблиці**

**Command:**

CREATE TABLE cas_rep3.items (category text, price int, model text, producer text, PRIMARY KEY (category, price) ) WITH CLUSTERING ORDER BY (price ASC);

CREATE TABLE cas_rep2.items (category text, price int, model text, producer text, PRIMARY KEY (category, price) ) WITH CLUSTERING ORDER BY (price ASC);

CREATE TABLE cas_rep1.items (category text, price int, model text, producer text, PRIMARY KEY (category, price) ) WITH CLUSTERING ORDER BY (price ASC);

5) **Спробуйте писати і читати на / та з різних нод.**

**Command:**

docker exec -it cassandra-1 cqlsh

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 600, 'Iphone 6', 'Apple');

select * from cas_rep3.items;

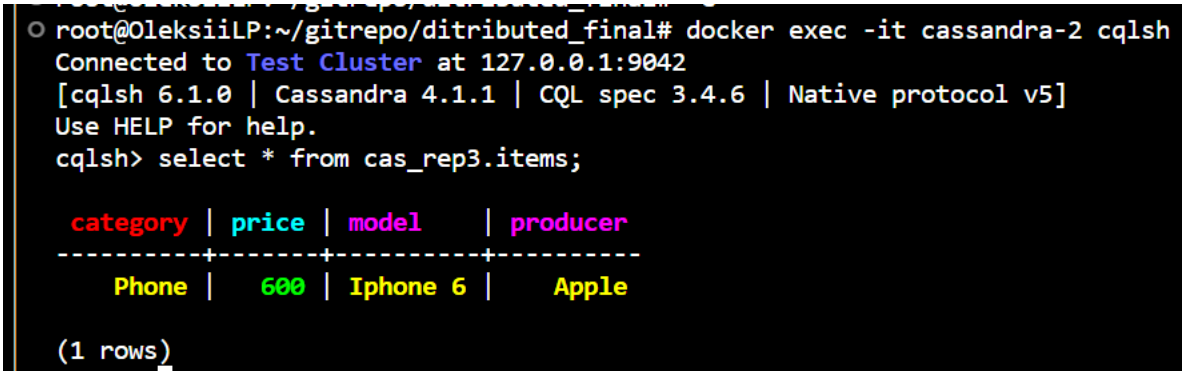
**Result:**

```
cqlsh> INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 600, 'Iphone 6', 'Apple');
cqlsh> select * from cas_rep3.items;

 category | price | model    | producer
----------+-------+----------+----------
    Phone |   600 | Iphone 6 |    Apple

(1 rows)
```

**Now, let's read from other node:**

docker exec -it cassandra-2 cqlsh

select * from cas_rep3.items;




6) **Вставте дані в створені таблиці і подивіться на їх розподіл по вузлах кластера (для кожного з кейспесов - *nodetool status*)**

**Command:**

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 700, 'Iphone 7', 'Apple');

INSERT INTO cas_rep2.items (category, price, model, producer) VALUES ('Phone', 700, 'Iphone 7', 'Apple');

INSERT INTO cas_rep1.items (category, price, model, producer) VALUES ('Phone', 700, 'Iphone 7', 'Apple');

INSERT INTO cas_rep2.items (category, price, model, producer) VALUES ('Phone', 600, 'Iphone 6', 'Apple');

INSERT INTO cas_rep1.items (category, price, model, producer) VALUES ('Phone', 600, 'Iphone 6', 'Apple');

docker exec cassandra-3 nodetool status

**Result:**

```
root@OleksiiLP:~/gitrepo/ditributed_final# docker exec cassandra-3 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
UN  172.17.0.4  138.84 KiB  16      ?     4c23028b-b9e4-4432-a179-0a0ce594039d  rack1
UN  172.17.0.3  138.79 KiB  16      ?     08217b34-a327-4969-ab5e-91cb8a13cc83  rack1
UN  172.17.0.2  167.63 KiB  16      ?     7b5aa66b-ab9e-4057-8350-393c9049e300  rack1
```

7) **Для якогось запису з кожного з кейспейсу виведіть ноди на яких зберігаються дані**

**Command:**

docker exec -it cassandra-1 sh

nodetool getendpoints cas_rep1 items "Phone"

nodetool getendpoints cas_rep2 items "Phone"

**Result:**

```
# nodetool getendpoints cas_rep1 items "Phone"
172.17.0.3
# nodetool getendpoints cas_rep2 items "Phone"
172.17.0.3
172.17.0.2
#
```

8) **Відключиіть одну з нод. Для кожного з кейспейсів визначить з якими рівнями *consistency* можемо читати та писати, і які з них забезпечують *strong consistency***

**Command:**

CONSISTENCY QUORUM;

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 800, 'Iphone 8', 'Apple');

INSERT INTO cas_rep2.items (category, price, model, producer) VALUES ('Phone', 800, 'Iphone 8', 'Apple');

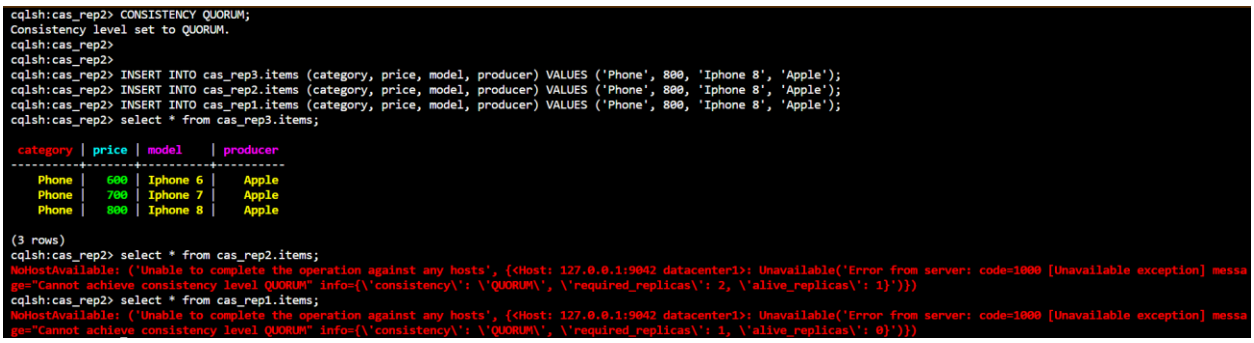
INSERT INTO cas_rep1.items (category, price, model, producer) VALUES ('Phone', 800, 'Iphone 8', 'Apple');

select * from cas_rep3.items;

select * from cas_rep2.items;

select * from cas_rep1.items;

**Result:**




9) **Зробить так щоб три ноди працювали, але не бачили одна одну по мережі (відключити зв'язок між ними)**

**Command:**

**Для кейспейсу з *replication factor* 3 задайте рівень consistency рівним 1. Виконайте запис одного й того самого значення, з однаковим primary key, але різними іншими значенням на кожну з нод (тобто створіть конфлікт)**

CONSISTENCY ONE;

For cassandra_1:

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 1000, 'Iphone 10', 'Apple');

For cassandra_2:

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 1000, 'Iphone 11', 'Apple');

For cassandra_3:

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 1000, 'Iphone 12', 'Apple');

**Result:**

Об’єднайте ноди в кластер і визначте яке значення було прийнято кластером та за яким принципом.

Запускаємо всі ноди, перевіряємо чи ноди об’єднались в кластер.

```
root@OleksiiLP:~/gitrepo/ditributed_final# docker exec cassandra-3 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
UN  172.17.0.4  255.89 KiB  16      ?     4c23028b-b9e4-4432-a179-0a0ce594039d  rack1
UN  172.17.0.3  268.22 KiB  16      ?     08217b34-a327-4969-ab5e-91cb8a13cc83  rack1
UN  172.17.0.2  221.53 KiB  16      ?     7b5aa66b-ab9e-4057-8350-393c9049e300  rack1
```

Під’єднуємось до cassandra_1 і перевіряємо яке значення залишилось.

docker exec -it cassandra-1 cqlsh

select * from cas_rep3.items;

```
cqlsh> select * from cas_rep3.items;

 category | price | model     | producer
----------+-------+-----------+----------
    Phone |   600 |  Iphone 6 |    Apple
    Phone |   700 |  Iphone 7 |    Apple
    Phone |   800 |  Iphone 8 |    Apple
    Phone |  1000 | Iphone 12 |    Apple
```

Як бачимо, залишилось значення, яке записували останнє. Відповідно. При неконсистентності вибирається запис з останнім записом по timestamp.

**12) Перевірте поведінку *lightweight transactions* для попередніх пунктів у розділеному та не розділеному на три частини кластері**

Роз’єднані кластери

**Command:**

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 1000, 'Galaxy S21', 'Samsung') IF NOT EXISTS;

Команду з lightweight transaction неможливо виконати у випадку коли кластери роз’єднані. Адже при виконанні lightweight transaction примусово ставиться SERIAL consistency level.

**Result:**


Об’єднані кластери

**Command:**

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 1100, 'Galaxy S21', 'Samsung') IF NOT EXISTS;

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 1100, 'Galaxy S22', 'Samsung') IF NOT EXISTS;

INSERT INTO cas_rep3.items (category, price, model, producer) VALUES ('Phone', 1100, 'Galaxy S23', 'Samsung') IF NOT EXISTS;

select * from cas_rep3.items;

**Result:**

```
cqlsh> select * from cas_rep3.items;

 category | price | model      | producer
----------+-------+------------+----------
    Phone |   600 |   Iphone 6 |    Apple
    Phone |   700 |   Iphone 7 |    Apple
    Phone |   800 |   Iphone 8 |    Apple
    Phone |  1000 |  Iphone 12 |    Apple
    Phone |  1100 | Galaxy S21 |  Samsung

(5 rows)
```

Зберігається перше значення, яке було вставлене. Всі інші ігноруються.