

Installing and using keyspace, column families and sets with Cassandra

1. Download Apache Cassandra

Previous Stable Version

Download the latest Apache Cassandra 3.11 release:

Released on 2022-06-17
Maintained until 4.2.0 release (May-July 2023)

3.11.13

2. Install Apache Cassandra

```
C:\Windows\system32\cmd.exe
Microsoft Windows [version 10.0.19044.2006]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Compte Principal>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     223,21 KiB 256        100,0%            2c2fc7e7-adb4-419d-b9c7-0c5f3e797b2c rack1

C:\Users\Compte Principal>
```

```
C:\Windows\system32\cmd.exe
85843694, 3210216969826594577, 3244186556916093159, 3339719070596585446, 3345865900069885806, 3447426761
730303158, 3454626143423981176, 3516942112737594550, 3573016408191526938, 3649435088755652611, 367147121
6727460701, 3671478481781087582, 3711300227458516751, 3762791667034088282, 3822066897852120272, 38646946
57878699372, 3882966750843012611, 3911041710478511511, 3927512031848288854, 4035996019163317938, 4089884
420361928693, 4229952065381684454, 4248692775355516925, 4293866962417447692, 4455259212058741960, 490306
4578220487134, 4909242789662464304, 4931167894533608102, 5029309775135884372, 5097993866580590973, 52138
33903031513885, 5242650438214844717, 5278927175245897465, 5340066091959012310, 5462470530696760099, 5490
947496206967553, 5512026509549526142, 5528017842384188080, 5542932723145811970, 5565796598690600898, 562
1970705595127762, 5735909114489105546, 5777356988339309902, 5873284241845928421, 5914141093389494372, 59
32116200151864151, 5934784188025790645, 5995316151306005818, 6030962558129348026, 6070255094352788852, 6
075800244219097940, 6088650477486518311, 6293833649640130466, 6372285423505378232, 6378950376759431199,
6451953887914801185, 6460159632615801754, 6463265530553131908, 6486948010811907012, 6544424987723748431,
6549863400977163287, 6690760718515827950, 6706372217681872322, 6864308473750214786, 6894598847814477186
, 6908459173236553858, 70423972712653493, 7167509159760540327, 7246218750812421384, 7284121910277119772,
7345356883465074642, 7427440150120387190, 7470187562658660279, 7545710283198441214, 7669520968609482642,
7725423037640327459, 7814540089292102930, 7865075375661400528, 7883196595016578818, 801288830218089765
2, 8147192892326550425, 8208781942185663806, 8431901662884363694, 8529289125532453646, 85835608602636971
80, 8584703692131440568, 8701455348630542575, 8744592601737073471, 8815217874918331929, 8846482348144117
629, 909198276965228899, 9135030819009940573, 9196209262836530051]
INFO [main] 2022-10-08 22:50:23,928 StorageService.java:1442 - JOINING: Finish joining ring
INFO [main] 2022-10-08 22:50:24,043 StorageService.java:2268 - Node localhost/127.0.0.1 state jump to N
ORMAL
```

3. Connect to Cassandra with CQL

```
username = louhou
pwsd = test
=====
C:\Users\Compte Principal>cqlsh -u louhou -p test localhost
Connected to Test Cluster at localhost:9042.
[cqlsh 5.0.1 | Cassandra 3.11.1 | CQL spec 3.4.4 | Native
protocol v4]
Use HELP for help.
louhou@cqlsh>
=====
```

4. Create a Keyspace called test with a replication factor 1 and simple strategy

```
=====
louhou@cqlsh> CREATE KEYSPACE test with REPLICATION =
{'class' : 'SimpleStrategy',
... 'replication_factor': 1};
=====
```

5. Go to the keyspace test

```
=====
louhou@cqlsh> USE test;
louhou@cqlsh:test>
=====
```

6.- Create a column family (table) called person with id text, email text, name text, surname text and is as primary key PRIMARY KEY (id);

```
=====
louhou@cqlsh:test> DESCRIBE table person

CREATE TABLE test.person (
    id text PRIMARY KEY,
    email text,
    name text,
    surname text
)
=====
```

7.- check the schema of the column family person

```
=====
louhou@cqlsh:test> DESCRIBE person ;

CREATE TABLE test.person (
    id text PRIMARY KEY,
```

```

        email text,
        name text,
        surname text
    )
=====

```

8. Populate person column family

```

=====
louhou@cqlsh:test> INSERT INTO person (id, name, surname,
email) VALUES ('001', 'Shalabh', 'Aggarwal',
'contact@shalabhaggarwal.com');
louhou@cqlsh:test> INSERT INTO person (id, name, surname,
email) VALUES ('002', 'John', 'Doe',
... 'john@example.com');
louhou@cqlsh:test> INSERT INTO person (id, name, surname,
email) VALUES ('003', 'Harry',
... 'Potter', 'harry@example.com');
=====

```

9. Get all information from person

```

=====
louhou@cqlsh:test> SELECT * FROM person ;

id | email | name | surname
---+-----+-----+-----
002 | john@example.com | John | Doe
001 | contact@shalabhaggarwal.com | Shalabu | Aggarwal
003 | harry@example.com | Harry | Potter
=====

```

get all information from person where id = 001

```

=====
louhou@cqlsh:test> SELECT * FROM person WHERE id='001';

id | email | name | surname
---+-----+-----+-----
001 | contact@shalabhaggarwal.com | Shalabh | Aggarwal

(1 rows)
=====

```

10. Using sets

```

=====
louhou@cqlsh:test> CREATE COLUMNFAMILY users (
... key varchar PRIMARY KEY,
... full_name varchar,
... birth_date int,
... state varchar,

```

```
... emails set<text>);
=====
```

11.- create an index on users with column birth_date and other index on state

```
=====
louhou@cqlsh:test> CREATE INDEX ON users (birth_date);
louhou@cqlsh:test> CREATE INDEX ON users (state);
=====
```

12. populate users columnfamily with the following data:

```
=====
louhou@cqlsh:test> SELECT * FROM users ;

key          | birth_date | emails                                     | full_name      | state
-----+-----+-----+-----+-----
htayler      | 1968      | {'hty@Hotmail.com,otheremail@Gmail.com'} | Howard Tayler  | UT
asmith       | 1973      | {'asmith@Gmail.com'}                     | Alice Smith    | WI
pangeles     | 1975      | {'plang@gmail.com,mpa@hotmail.com'}       | Pilar Angeles  | UT

(3 rows)
=====
```

13. Get full name and emails for Pilar Angeles

```
=====
louhou@cqlsh:test> Select full_name, emails from users where
key= 'pangeles';

full_name      | emails
-----+-----
Pilar Angeles | {'plang@gmail.com,mpa@hotmail.com'}

(1 rows)
=====
```

14. Get key and state from users

```
=====
louhou@cqlsh:test> SELECT key, state from users;

key          | state
-----+-----
htayler      | UT
asmith       | WI
pangeles     | UT

(3 rows)
=====
```

15. Get all users that live in UT and were born after 1970

```
=====
louhou@cqlsh:test> SELECT * FROM users WHERE state='UT' AND birth_date > 1970 ALLOW
```

FILTERING;

key	birth_date	emails	full_name	state
pangeles	1975	{'plang@gmail.com,mpa@hotmail.com'}	Pilar Angeles	UT

(1 rows)

=====

Questions:

How are the relationships one to many implemented in Cassandra? => By creating and managing one table

How are the relationships many to many implemented in Cassandra? => By creating and managing many tables