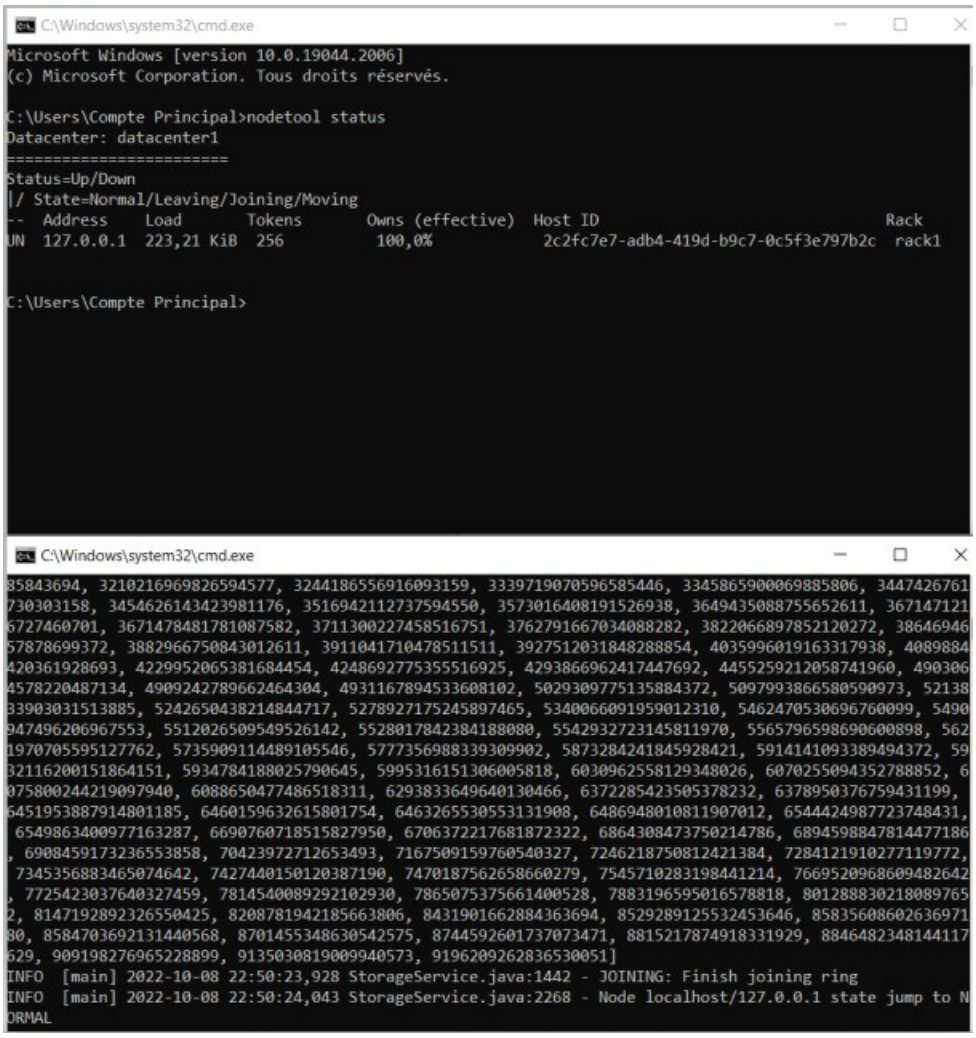
Installing and using keyspace, column families and sets with Cassandra

1. Download Apache Cassandra

2. Install Apache Cassandra



3. Connect to Cassandra with CQL

```
usrname = 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 ang managing one table

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