4CS372: Advance Database System Lab (ADSL)

Assignment NO:9

May 21, 2021

GAURAV ARORA(PRN:2018BTECS00032)
VIDHAN SHAH(PRN:2018BTECS00058) ,
GroupNo:2018BCGRP24

2018BCGRP24

Title: Cassandra Clustering

Objective/Aim:

1. Setup a multi-node Cassandra Cluster on single windows machine.
2. Install the DataStax OpsCenter community edition.
3. Demonstrate the cluster operations of above use case using OpsCenter.

Introduction & Theory:

Apache Cassandra is an open source NoSQL distributed database trusted by thousands of companies for scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data.

What is Cluster?

The cluster is a collection of nodes that represents a single system. A cluster in Cassandra is one of the shells in the whole Cassandra database. Many Cassandra Clusters combine together to form the database in Cassandra.A Cluster is basically the outermost shell or storage unit in a database. The Cassandra Cluster contains many different layers of storage units. Each layer contains the other.

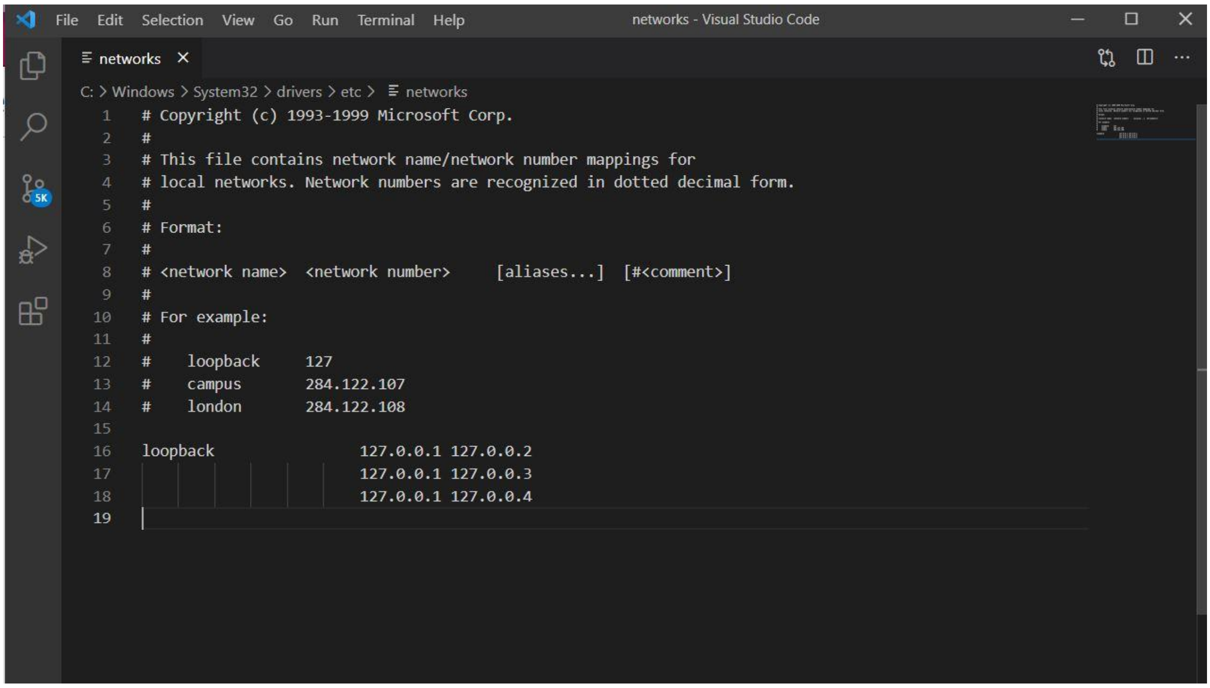
Implementation:

1)Setup a multi-node Cassandra Cluster on single windows machine. Give your group name (2018BCGRP**) to cluster. Follow the steps given in below link https://extendit.us/articles/steps-configure-multiple-nodescassandra-single-windows-machine

Step 1:
C:\Windows\System32\drivers\etc

Update the file with the following entries:

```
127.0.0.1 127.0.0.2
127.0.0.1 127.0.0.3
127.0.0.1 127.0.0.4
```

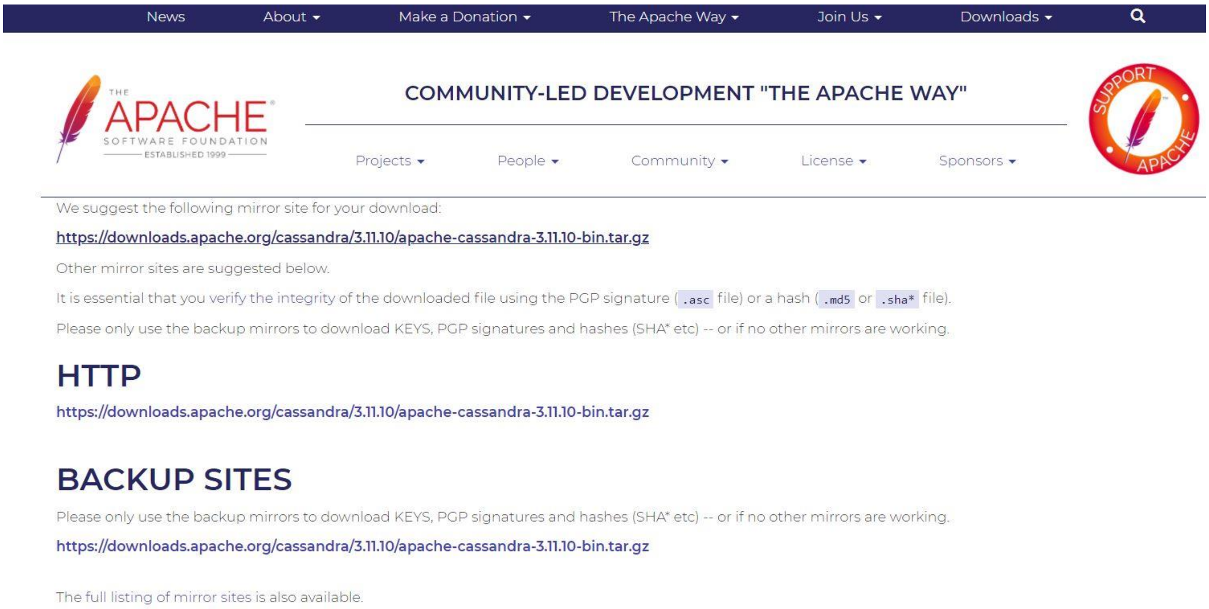

File
Edit
Selection
View
Go
Run
Terminal
Help
networks - Visual Studio Code
networks
C: > Windows > System32 > drivers > etc > networks
5K

```
# Copyright (c) 1993-1999 Microsoft Corp.
#
# This file contains network name/network number mappings for
# local networks. Network numbers are recognized in dotted decimal form.
#
# Format:
#
# <network name>  <network number>     [aliases...]  [#<comment>]
#
# For example:
#
#    loopback     127
#    campus       284.122.107
#    london       284.122.108

loopback                127.0.0.1 127.0.0.2
                        127.0.0.1 127.0.0.3
                        127.0.0.1 127.0.0.4

```

Step 2:

Next, create the folder structure for your Cassandra installation.

Installation: Install the DataStax OpsCenter community edition



We suggest the following mirror site for your download:

https://downloads.apache.org/cassandra/3.11.10/apache-cassandra-3.11.10-bin.tar.gz

Other mirror sites are suggested below.

It is essential that you verify the integrity of the downloaded file using the PGP signature (`.asc` file) or a hash (`.md5` or `.sha*` file).

Please only use the backup mirrors to download KEYS, PGP signatures and hashes (SHA* etc) -- or if no other mirrors are working.

HTTP

https://downloads.apache.org/cassandra/3.11.10/apache-cassandra-3.11.10-bin.tar.gz

BACKUP SITES

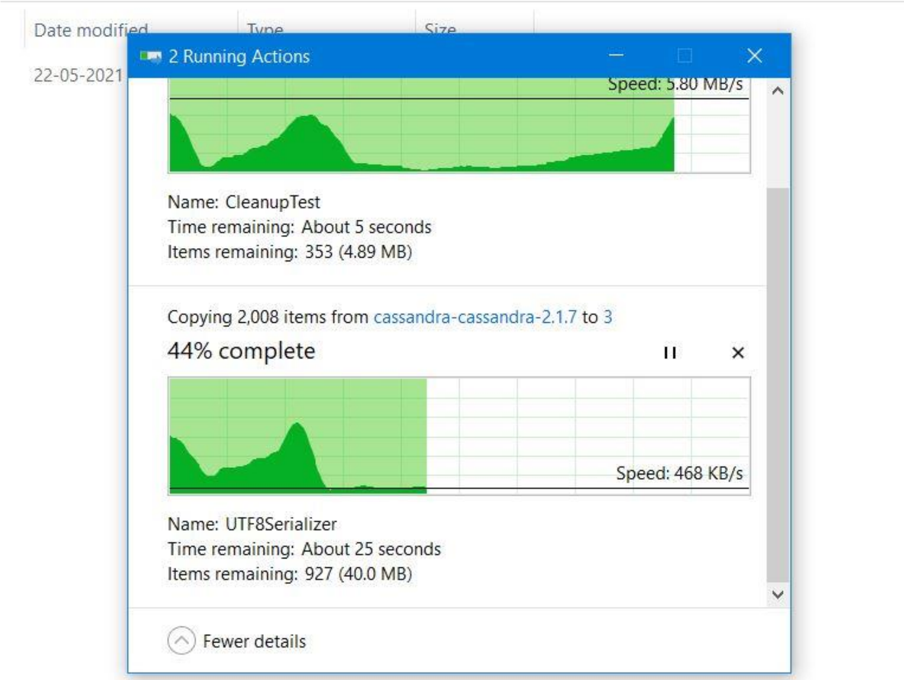
Please only use the backup mirrors to download KEYS, PGP signatures and hashes (SHA* etc) -- or if no other mirrors are working.

https://downloads.apache.org/cassandra/3.11.10/apache-cassandra-3.11.10-bin.tar.gz

The full listing of mirror sites is also available.

Step 3:

Extract the Cassandra distribution into folder 1,2 and 3:



Date modified   Type   Size

22-05-2021

2 Running Actions

Speed: 5.80 MB/s

Name: CleanupTest
Time remaining: About 5 seconds
Items remaining: 353 (4.89 MB)

Copying 2,008 items from cassandra-cassandra-2.1.7 to 3

44% complete

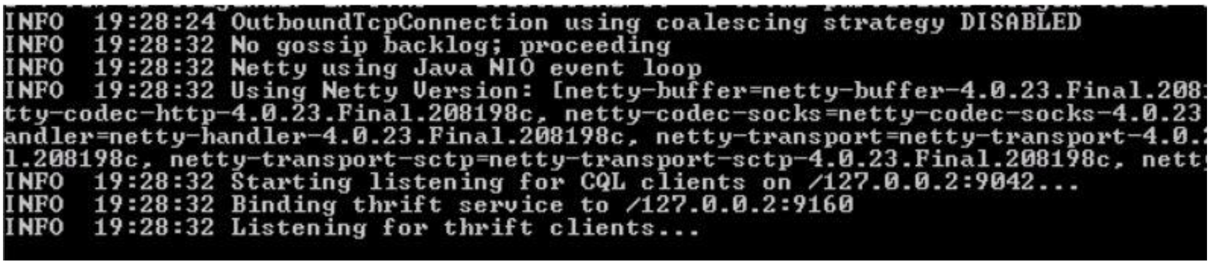
Speed: 468 KB/s

Name: UTF8Serializer
Time remaining: About 25 seconds
Items remaining: 927 (40.0 MB)

Step 4:

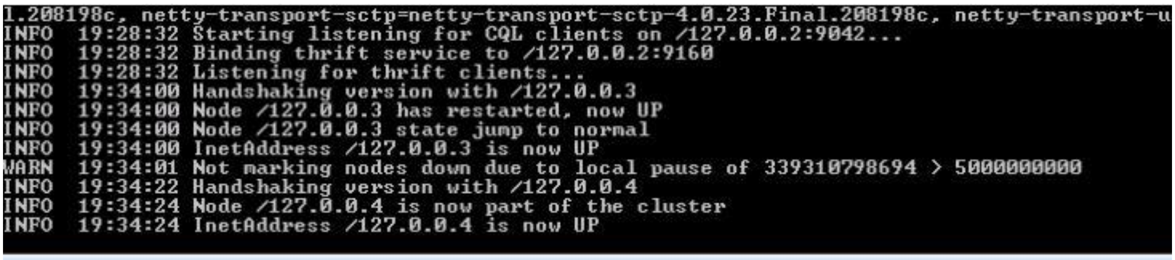
Now edit the code in each section in the file conf/cassadra.yaml and configure each of the Cassandra's nodes JMX port to point to different port numbers.
Open up cassandra.bat file for each of your nodes under the bin directory and look for the value:

1st file:



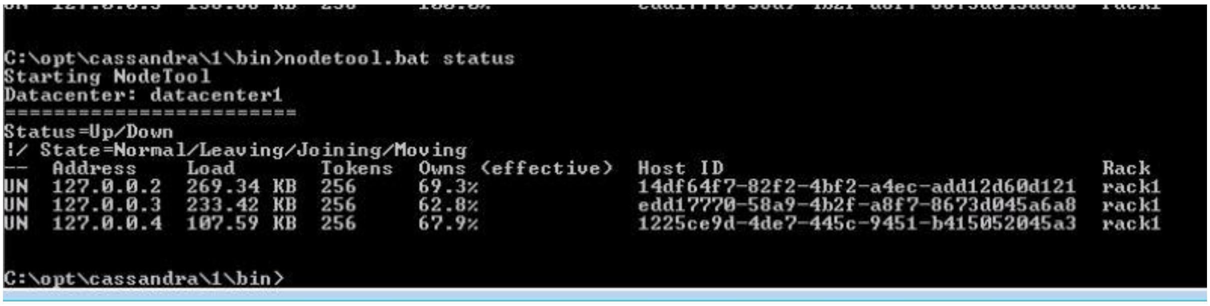
```
INFO  19:28:24 OutboundTcpConnection using coalescing strategy DISABLED
INFO  19:28:32 No gossip backlog; proceeding
INFO  19:28:32 Netty using Java NIO event loop
INFO  19:28:32 Using Netty Version: [netty-buffer=netty-buffer-4.0.23.Final.208
tty-codec-http-4.0.23.Final.208198c, netty-codec-socks=netty-codec-socks-4.0.23
andler=netty-handler-4.0.23.Final.208198c, netty-transport=netty-transport-4.0.
l.208198c, netty-transport-sctp=netty-transport-sctp-4.0.23.Final.208198c, nett
INFO  19:28:32 Starting listening for CQL clients on /127.0.0.2:9042...
INFO  19:28:32 Binding thrift service to /127.0.0.2:9160
INFO  19:28:32 Listening for thrift clients...
```

2nd file:



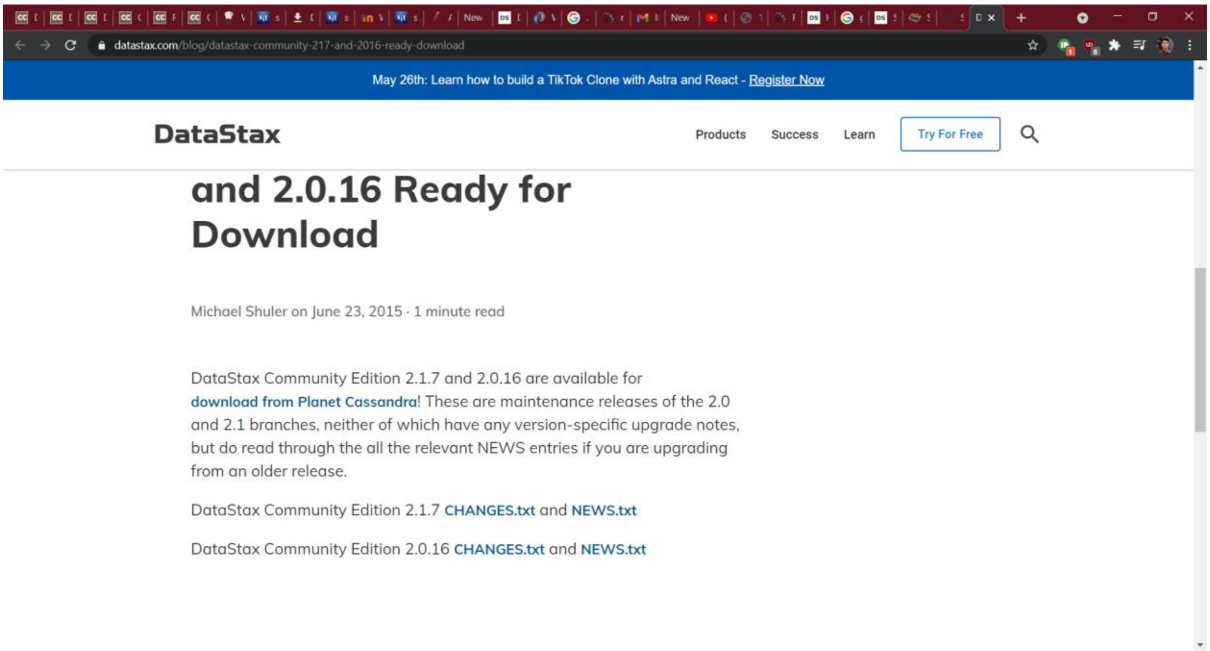
```
l.208198c, netty-transport-sctp=netty-transport-sctp-4.0.23.Final.208198c, netty-transport-u
INFO  19:28:32 Starting listening for CQL clients on /127.0.0.2:9042...
INFO  19:28:32 Binding thrift service to /127.0.0.2:9160
INFO  19:28:32 Listening for thrift clients...
INFO  19:34:00 Handshaking version with /127.0.0.3
INFO  19:34:00 Node /127.0.0.3 has restarted, now UP
INFO  19:34:00 Node /127.0.0.3 state jump to normal
INFO  19:34:00 InetAddress /127.0.0.3 is now UP
WARN  19:34:01 Not marking nodes down due to local pause of 339310798694 > 5000000000
INFO  19:34:22 Handshaking version with /127.0.0.4
INFO  19:34:24 Node /127.0.0.4 is now part of the cluster
INFO  19:34:24 InetAddress /127.0.0.4 is now UP
```

3rd file:



```
C:\opt\cassandra\1\bin>nodetool.bat status
Starting NodeTool
Datacenter: datacenter1
========================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens    Owns (effective)  Host ID                               Rack
UN  127.0.0.2  269.34 KB  256       69.3%             14df64f7-82f2-4bf2-a4ec-add12d60d121  rack1
UN  127.0.0.3  233.42 KB  256       62.8%             edd17770-58a9-4b2f-a8f7-8673d045a6a8  rack1
UN  127.0.0.4  107.59 KB  256       67.9%             1225ce9d-4de7-445c-9451-b415052045a3  rack1

C:\opt\cassandra\1\bin>
```

2. Install the DataStax OpsCenter community edition (https://www.datastax.com/blog/datastax-community-217-and-2016-ready-download) and configure it for above cluster formed.



datastax.com/blog/datastax-community-217-and-2016-ready-download
May 26th: Learn how to build a TikTok Clone with Astra and React - Register Now
DataStax
Products
Success
Learn
Try For Free

and 2.0.16 Ready for Download

Michael Shuler on June 23, 2015 · 1 minute read

DataStax Community Edition 2.1.7 and 2.0.16 are available for download from Planet Cassandra! These are maintenance releases of the 2.0 and 2.1 branches, neither of which have any version-specific upgrade notes, but do read through the all the relevant NEWS entries if you are upgrading from an older release.

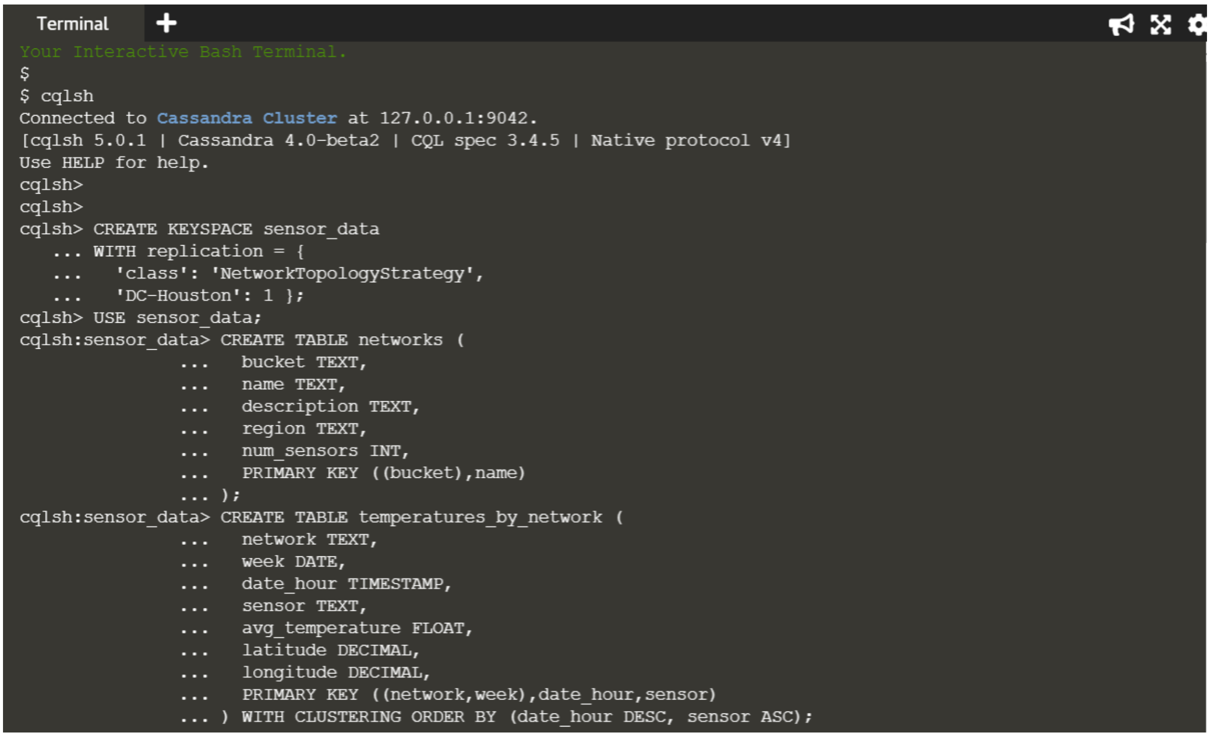
DataStax Community Edition 2.1.7 CHANGES.txt and NEWS.txt

DataStax Community Edition 2.0.16 CHANGES.txt and NEWS.txt

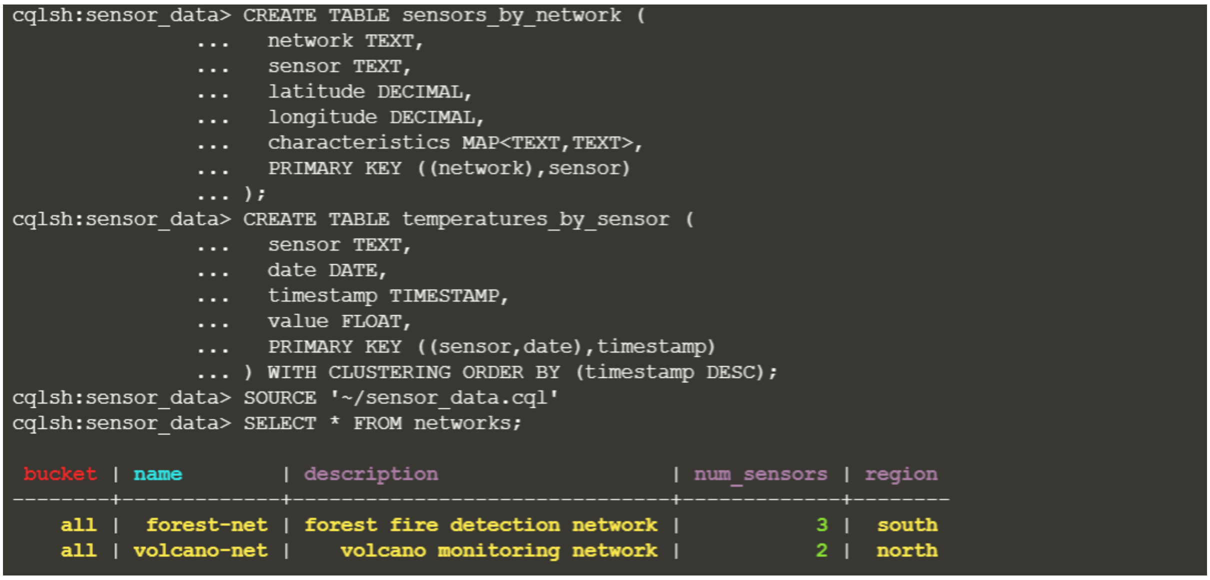
3. Use Case - Weather Station IoT Temperature Sensor Data : There are set of weather stations at different remote location with “weatherStationID”. Each station record the

temperature after every 5 minutes and push the data to nearest node in above cluster. Design the cluster database to hold these weather data. User should be able to retrieve the data in any dimensions

1) Creating table sensor_data

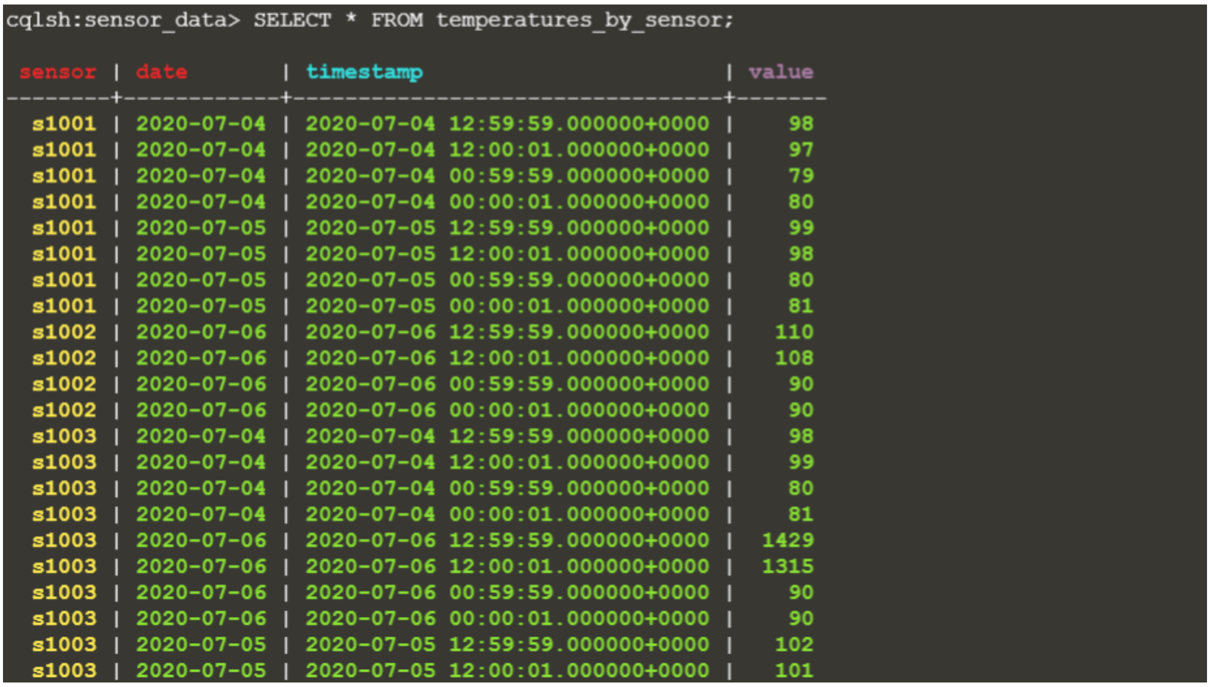


```
Your Interactive Bash Terminal.
$
$ cqlsh
Connected to Cassandra Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 4.0-beta2 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
cqlsh>
cqlsh>
cqlsh> CREATE KEYSPACE sensor_data
   ... WITH replication = {
   ...   'class': 'NetworkTopologyStrategy',
   ...   'DC-Houston': 1 };
cqlsh> USE sensor_data;
cqlsh:sensor_data> CREATE TABLE networks (
               ...   bucket TEXT,
               ...   name TEXT,
               ...   description TEXT,
               ...   region TEXT,
               ...   num_sensors INT,
               ...   PRIMARY KEY ((bucket),name)
               ... );
cqlsh:sensor_data> CREATE TABLE temperatures_by_network (
               ...   network TEXT,
               ...   week DATE,
               ...   date_hour TIMESTAMP,
               ...   sensor TEXT,
               ...   avg_temperature FLOAT,
               ...   latitude DECIMAL,
               ...   longitude DECIMAL,
               ...   PRIMARY KEY ((network,week),date_hour,sensor)
               ... ) WITH CLUSTERING ORDER BY (date_hour DESC, sensor ASC);
```

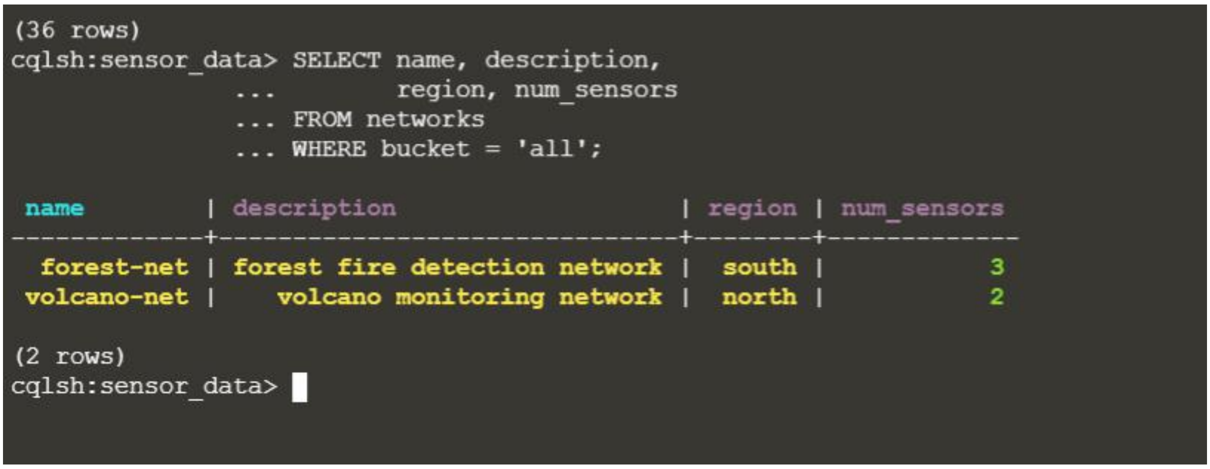


```
cqlsh:sensor_data> CREATE TABLE sensors_by_network (
               ...   network TEXT,
               ...   sensor TEXT,
               ...   latitude DECIMAL,
               ...   longitude DECIMAL,
               ...   characteristics MAP<TEXT,TEXT>,
               ...   PRIMARY KEY ((network),sensor)
               ... );
cqlsh:sensor_data> CREATE TABLE temperatures_by_sensor (
               ...   sensor TEXT,
               ...   date DATE,
               ...   timestamp TIMESTAMP,
               ...   value FLOAT,
               ...   PRIMARY KEY ((sensor,date),timestamp)
               ... ) WITH CLUSTERING ORDER BY (timestamp DESC);
cqlsh:sensor_data> SOURCE '~/sensor_data.cql'
cqlsh:sensor_data> SELECT * FROM networks;

 bucket | name        | description              | num_sensors | region
--------+-------------+--------------------------+-------------+--------
    all |  forest-net | forest fire detection network |        3 |  south
    all | volcano-net |   volcano monitoring network |        2 |  north
```

2) Getting temperature of different zones.

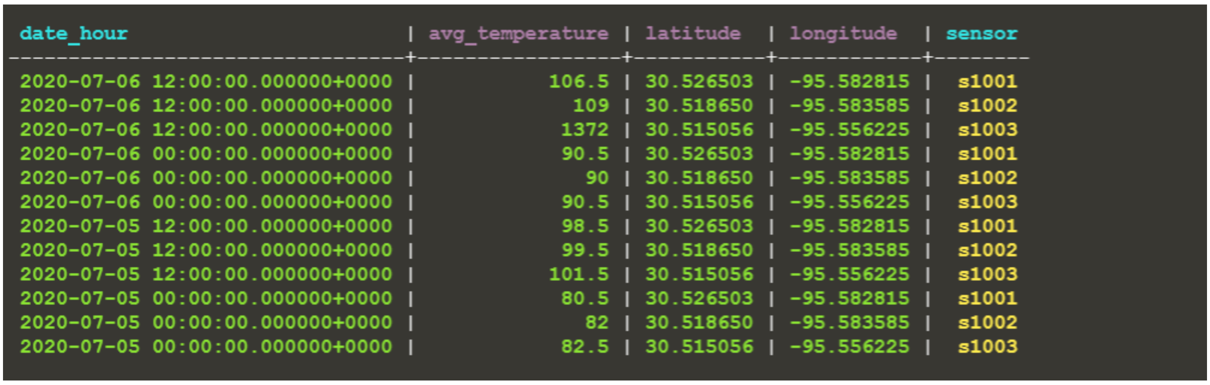


```
cqlsh:sensor_data> SELECT * FROM temperatures_by_sensor;

 sensor | date       | timestamp                       | value
--------+------------+---------------------------------+-------
  s1001 | 2020-07-04 | 2020-07-04 12:59:59.000000+0000 |    98
  s1001 | 2020-07-04 | 2020-07-04 12:00:01.000000+0000 |    97
  s1001 | 2020-07-04 | 2020-07-04 00:59:59.000000+0000 |    79
  s1001 | 2020-07-04 | 2020-07-04 00:00:01.000000+0000 |    80
  s1001 | 2020-07-05 | 2020-07-05 12:59:59.000000+0000 |    99
  s1001 | 2020-07-05 | 2020-07-05 12:00:01.000000+0000 |    98
  s1001 | 2020-07-05 | 2020-07-05 00:59:59.000000+0000 |    80
  s1001 | 2020-07-05 | 2020-07-05 00:00:01.000000+0000 |    81
  s1002 | 2020-07-06 | 2020-07-06 12:59:59.000000+0000 |   110
  s1002 | 2020-07-06 | 2020-07-06 12:00:01.000000+0000 |   108
  s1002 | 2020-07-06 | 2020-07-06 00:59:59.000000+0000 |    90
  s1002 | 2020-07-06 | 2020-07-06 00:00:01.000000+0000 |    90
  s1003 | 2020-07-04 | 2020-07-04 12:59:59.000000+0000 |    98
  s1003 | 2020-07-04 | 2020-07-04 12:00:01.000000+0000 |    99
  s1003 | 2020-07-04 | 2020-07-04 00:59:59.000000+0000 |    80
  s1003 | 2020-07-04 | 2020-07-04 00:00:01.000000+0000 |    81
  s1003 | 2020-07-06 | 2020-07-06 12:59:59.000000+0000 |  1429
  s1003 | 2020-07-06 | 2020-07-06 12:00:01.000000+0000 |  1315
  s1003 | 2020-07-06 | 2020-07-06 00:59:59.000000+0000 |    90
  s1003 | 2020-07-06 | 2020-07-06 00:00:01.000000+0000 |    90
  s1003 | 2020-07-05 | 2020-07-05 12:59:59.000000+0000 |   102
  s1003 | 2020-07-05 | 2020-07-05 12:00:01.000000+0000 |   101
```

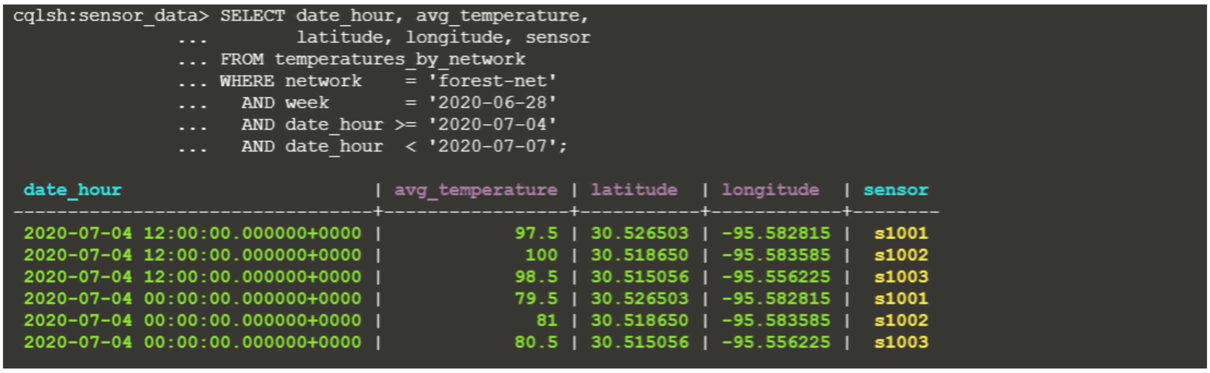


```
(36 rows)
cqlsh:sensor_data> SELECT name, description,
               ...        region, num_sensors
               ... FROM networks
               ... WHERE bucket = 'all';

 name        | description                    | region | num_sensors
-------------+--------------------------------+--------+-------------
  forest-net | forest fire detection network  |  south |           3
 volcano-net |     volcano monitoring network |  north |           2

(2 rows)
cqlsh:sensor_data> 
```

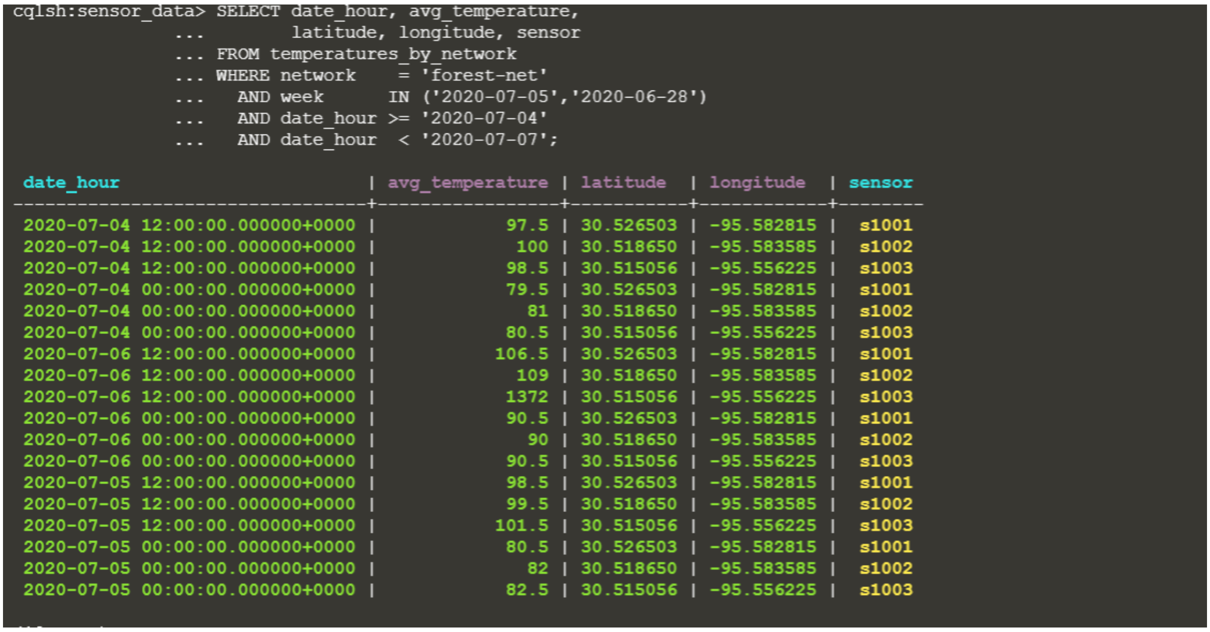
3) Getting average temperature at specific time



```
 date_hour                       | avg_temperature | latitude  | longitude  | sensor
---------------------------------+-----------------+-----------+------------+--------
 2020-07-06 12:00:00.000000+0000 |           106.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 12:00:00.000000+0000 |             109 | 30.518650 | -95.583585 |  s1002
 2020-07-06 12:00:00.000000+0000 |            1372 | 30.515056 | -95.556225 |  s1003
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 00:00:00.000000+0000 |              90 | 30.518650 | -95.583585 |  s1002
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 12:00:00.000000+0000 |            98.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 12:00:00.000000+0000 |            99.5 | 30.518650 | -95.583585 |  s1002
 2020-07-05 12:00:00.000000+0000 |           101.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 00:00:00.000000+0000 |            80.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 00:00:00.000000+0000 |              82 | 30.518650 | -95.583585 |  s1002
 2020-07-05 00:00:00.000000+0000 |            82.5 | 30.515056 | -95.556225 |  s1003
```



```
cqlsh:sensor_data> SELECT date_hour, avg_temperature,
               ...        latitude, longitude, sensor
               ... FROM temperatures_by_network
               ... WHERE network    = 'forest-net'
               ...   AND week       = '2020-06-28'
               ...   AND date_hour >= '2020-07-04'
               ...   AND date_hour  < '2020-07-07';

 date_hour                       | avg_temperature | latitude  | longitude  | sensor
---------------------------------+-----------------+-----------+------------+--------
 2020-07-04 12:00:00.000000+0000 |            97.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 12:00:00.000000+0000 |             100 | 30.518650 | -95.583585 |  s1002
 2020-07-04 12:00:00.000000+0000 |            98.5 | 30.515056 | -95.556225 |  s1003
 2020-07-04 00:00:00.000000+0000 |            79.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 00:00:00.000000+0000 |              81 | 30.518650 | -95.583585 |  s1002
 2020-07-04 00:00:00.000000+0000 |            80.5 | 30.515056 | -95.556225 |  s1003
```



```
cqlsh:sensor_data> SELECT date_hour, avg_temperature,
               ...        latitude, longitude, sensor
               ... FROM temperatures_by_network
               ... WHERE network    = 'forest-net'
               ...   AND week      IN ('2020-07-05','2020-06-28')
               ...   AND date_hour >= '2020-07-04'
               ...   AND date_hour  < '2020-07-07';

 date_hour                       | avg_temperature | latitude  | longitude  | sensor
---------------------------------+-----------------+-----------+------------+--------
 2020-07-04 12:00:00.000000+0000 |            97.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 12:00:00.000000+0000 |             100 | 30.518650 | -95.583585 |  s1002
 2020-07-04 12:00:00.000000+0000 |            98.5 | 30.515056 | -95.556225 |  s1003
 2020-07-04 00:00:00.000000+0000 |            79.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 00:00:00.000000+0000 |              81 | 30.518650 | -95.583585 |  s1002
 2020-07-04 00:00:00.000000+0000 |            80.5 | 30.515056 | -95.556225 |  s1003
 2020-07-06 12:00:00.000000+0000 |           106.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 12:00:00.000000+0000 |             109 | 30.518650 | -95.583585 |  s1002
 2020-07-06 12:00:00.000000+0000 |            1372 | 30.515056 | -95.556225 |  s1003
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 00:00:00.000000+0000 |              90 | 30.518650 | -95.583585 |  s1002
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 12:00:00.000000+0000 |            98.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 12:00:00.000000+0000 |            99.5 | 30.518650 | -95.583585 |  s1002
 2020-07-05 12:00:00.000000+0000 |           101.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 00:00:00.000000+0000 |            80.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 00:00:00.000000+0000 |              82 | 30.518650 | -95.583585 |  s1002
 2020-07-05 00:00:00.000000+0000 |            82.5 | 30.515056 | -95.556225 |  s1003
```

Conclusion:

Using DataStax we fetch the data and store it in Cassandra DB using clusters.

References:

- https://extendit.us/articles/steps-configure-multiple-nodes-cassandra-single-windows-machine
- https://www.datastax.com/blog/datastax-community-217-and-2016-ready-download.