Big Data Analysis

Practical 9

Objective :

Setup Cassandra environment in your system and apply Create, Update, Read and Delete operations.

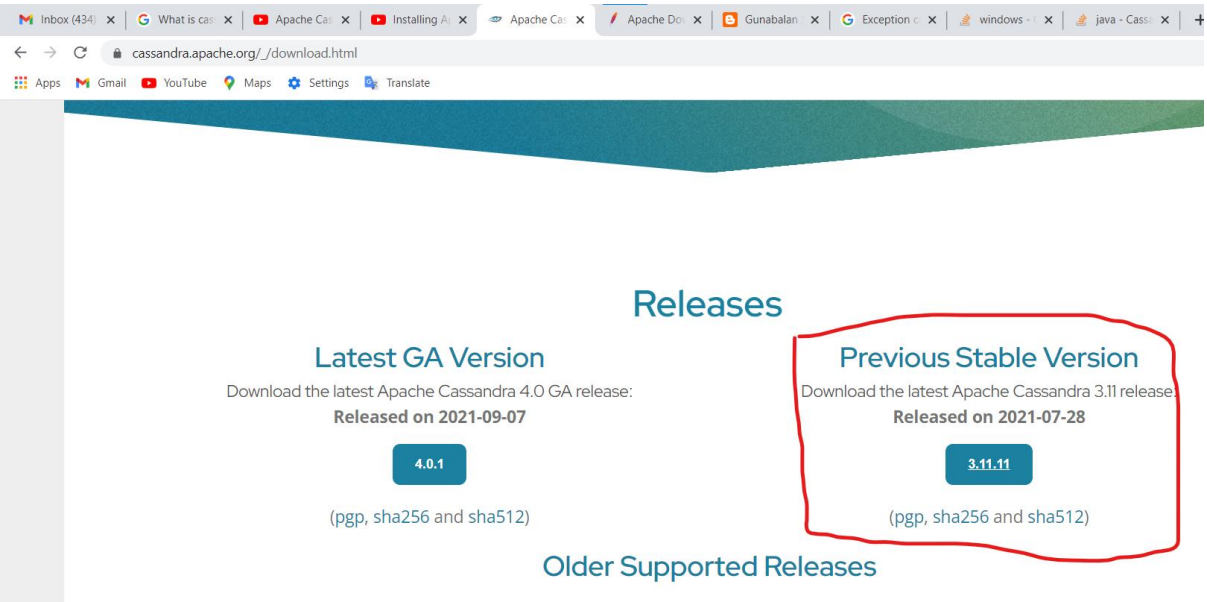
Roll No. & Name : 18bce183 & Prince Prajapati

Submitted to: Prof. Jaiprakash Verma

What is Apache Cassandra?

Apache Cassandra is a highly scalable, high-performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. It is a type of NoSQL database

Download Cassandra:



/ ESTABLISHED 1999 Projects People Community License Sp

We suggest the following mirror site for your download:

https://dlcdn.apache.org/cassandra/3.11.11/apache-cassandra-3.11.11-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://dlcdn.apache.org/cassandra/3.11.11/apache-cassandra-3.11.11-bin.tar.gz

Set PATH:

CASSANDRA_HOME	C:\Program Files\apache-cassandra-3.11.11
JAVA_BIN	C:\Progra~1\Java\jdk1.8.0_221\bin
JAVA_HOME	C:\Progra~1\Java\jdk1.8.0_221

SET ExecutionPolicy:

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> Get-ExecutionPolicy -List

        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process       Undefined
  CurrentUser       Undefined
 LocalMachine       Undefined


PS C:\WINDOWS\system32> Set-ExecutionPolicy Unrestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy m
you to the security risks described in the about_Execution_Policies help topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): y
PS C:\WINDOWS\system32> Get-ExecutionPolicy -List

        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process       Undefined
  CurrentUser       Undefined
 LocalMachine    Unrestricted


PS C:\WINDOWS\system32>
```

Now Check Cassandra running :

```
C:\Program Files\apache-cassandra-3.11.11\bin>cassandra
Detected powershell execution permissions.  Running with enhanced startup scripts.
*-----------------------------------------------------------------------*
*-----------------------------------------------------------------------*

    WARNING!  Automatic page file configuration detected.
    It is recommended that you disable swap when running Cassandra
    for performance and stability reasons.

*-----------------------------------------------------------------------*
*-----------------------------------------------------------------------*
*-----------------------------------------------------------------------*
*-----------------------------------------------------------------------*

    WARNING! Detected a power profile other than High Performance.
    Performance of this node will suffer.
    Modify conf\cassandra.env.ps1 to suppress this warning.
```

CREATE KEYSPACE:

```
CREATE KEYSPACE PRAC9 WITH replication
= {'class': 'SimpleStrategy', 'replication_factor': 3};
```

1. Strategy: There are two types of strategy declaration in Cassandra syntax:

Simple Strategy: Simple strategy is used in the case of one data center. In this strategy, the first replica is placed on the selected node and the remaining nodes are placed in clockwise direction in the ring without considering rack or node location.

Network Topology Strategy: This strategy is used in the case of more than one data centers. In this strategy, you have to provide replication factor for each data center separately.

2. Replication Factor: Replication factor is the number of replicas of data placed on different nodes. More than two replication

factor are good to attain no single point of failure. So, 3 is good replication factor.

```
cqlsh> CREATE KEYSPACE demo WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> CREATE KAYSPACE PRAC9_183 WITH replication = {
   ... 'class':'SimpleStrategy','replication_factor':3};
SyntaxException: line 1:7 no viable alternative at input 'KAYSPACE' ([CREATE] KAYSPACE...)
cqlsh> CREATE KEYSPACE PRAC9_183 WITH replication = {
   ...  'class':'SimpleStrategy','replication_factor':3};
cqlsh> CREATE KEYSPACE PRAC8_183 WITH replication = {  'class':'SimpleStrategy','replication_factor':3};
cqlsh> DDESCRIBE KEYSPACES;
```

VIEW ALL KEYSPACES:

```
DESCRIBE KEYSPACES;
```

```
cqlsh> DESCRIBE KEYSPACES;

demo       prac9_183  system_auth         system_schema  system_views
prac8_183  system     system_distributed  system_traces  system_virtual_schema
```

ALTER KEYSPACE:

```
ALTER KEYSPACE PRAC8 WITH replication =
{'class':'NetworkTopologyStrategy', 'replication_factor' : 1};
```

DROP KEYSPACE:

```
DROP  KEYSPACE PRAC8 ;
```

```
cqlsh> ALTER KEYSPACE PRAC8_183 WITH REPLICATION ={'class':'NetworkTopologyStrategy','replication_factor':1};
cqlsh> DROP KEYSPACE PRAC8_183;
cqlsh> DESCRIBE KEYSPACE;
No keyspace specified and no current keyspace
cqlsh> DESCRIBE KEYSPACEs;

demo       system       system_distributed  system_traces  system_virtual_schema
prac9_183  system_auth  system_schema       system_views
```

CREATE TABLE:

```
CREATE TABLE PRAC9_183.users
(firstname text PRIMARY KEY,
 lastname text,
email text);
```

```
cqlsh> CREATE TABLE PRAC9_183.users(firstname text PRIMARY KEY, lastname text,email text);
cqlsh> CREATE TABLE demo.users (lastname text PRIMARY KEY, firstname text, email text);
```

INSERT VALUES INTO TABLE(CREATE):

```
USE PRAC9_183;

INSERT INTO users(lastname,firstname,email)
VALUES(‘Round’, ‘Creaig’, ‘craig@gmail.com’);
```

SELECT (READ):

WHERE clause can be used only on the columns that are a part of primary key or have a secondary index on them.

```
SELECT * from users;
SELECT * FROM users WHERE firstname = 'Cassi';
```

```
cqlsh:demo> USE PRAC9_183;
cqlsh:prac9_183> INSERT INTO users (lastname, firstname, email) VALUES ('Round', 'Craig', 'craig@example.com');
cqlsh:prac9_183> INSERT INTO users (lastname, firstname, email) VALUES ('Pratico', 'Cassi', 'cassi@example.com');
cqlsh:prac9_183> SELECT * FROM USERS;

 firstname | email            | lastname
-----------+------------------+----------
     Cassi | cassi@example.com |  Pratico
     Craig | craig@example.com |    Round

(2 rows)
```

```
cqlsh:prac9_183> SELECT * FROM users WHERE firstname = 'Cassi';

 firstname | email               | lastname
-----------+---------------------+----------
     Cassi | cassi@example.com |  Pratico
```

SET (UPDATE):

```
UPDATE users SET email = '18bce183@nirmauni.ac.in'
WHERE firstname = 'Cassi';
```

```
cqlsh:prac9_183> UPDATE users SET email = '18bce183@nirmauni.ac.in'  WHERE firstname = 'Cassi';
cqlsh:prac9_183> SELECT * FROM users WHERE firstname = 'Cassi';

 firstname | email                   | lastname
-----------+-------------------------+----------
     Cassi | 18bce183@nirmauni.ac.in |  Pratico

(1 rows)
```

DELETE :

```
DELETE FROM users WHERE firstname='Cassi';
```

```
cqlsh:prac9_183> DELETE FROM users WHERE firstname='Cassi';
cqlsh:prac9_183> SELECT * FROM users WHERE firstname = 'Cassi';

 firstname | email | lastname
-----------+-------+----------

(0 rows)
```

Conclusion:

After implementing this practical now I have complete understanding of how Cassandra works and how to use it operations.