Cassandra Implementation

Overall Implementation Summary
1. Setup Java in the home directory and replicate to all the nodes.
2. Setup Cassandra and replicate changes to all the other nodes.

Java Setup

You can download and set up the Java Development Kit (JDK) version 11.0.2 using Command Prompt.
Follow these steps:

1. Download JDK 11.0.2:
```
curl -O https://download.java.net/java/GA/jdk11/9/GPL/openjdk-11.0.2_windows-x64_bin.zip
```
you will get a folder like : openjdk-11.0.2_linux-x64_bin.tar.gz.

2. Unzip the downloaded file:
```
tar -xvf openjdk-11.0.2_windows-x64_bin.zip
```

3. Set up a new environment variable for Java:
```
export JAVA_HOME="/path/to/your/jdk-11.0.2"
```
in our case it was : /home/stuproj/cs4224b/jdk-11.0.2/

update in bash rc using the following command : `nano ~/.bashrc`
once you are in the file,
```
export JAVA_HOME=/home/stuproj/cs4224b/jdk-11.0.2/
export PATH=$JAVA_HOME/bin:$PATH
```

4. Append the Java bin directory to the system's PATH variable:
```
export PATH="$PATH:/path/to/your/jdk-11.0.2/bin"
```
You can verify if java is properly installed by running the command in home directory : `java -version`

This was the result we got:
```
cs4224b@xcnc44:~$ java -version
openjdk version "11.0.2" 2019-01-15
OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)
```

Cassandra Setup

We had setup Cassandra in shared drive and copied it into temp/cs4224b/apache-cassandra-4.1.3, so that we could copy the Cassandra folder entirely from temp to other nodes. (within temp folder, we created the folder cs4224b and copied the Cassandra folder after all initial configuration and setup).

These steps are to be done initially in shared drive for and after downloading.

To download and unzip Apache Cassandra 4.1.13 through the command prompt, follow these steps:

1. Download the Apache Cassandra 4.1.13 tar.gz file:
```
wget http://mirror.metrocast.net/apache/cassandra/4.0.1/apache-cassandra-4.0.1-bin.tar.gz
```

2. Unzip the downloaded tar.gz file:
```
tar -xvzf apache-cassandra-4.1.13-bin.tar.gz
```

3. Set up a new environment variable for Cassandra:

```
export CASSANDRA_HOME=/home/yourusername/apache-cassandra-4.0.1/
```

we have unzipped and installed Cassandra in /temp/cs4224b/apache-cassandra-4.1.3

update in bash rc using the following command : `nano ~/.bashrc`

once you are in the file,

```
export CASSANDRA_HOME=/temp/cs4224b/apache-cassandra-4.1.3
export PATH=$CASSANDRA_HOME/bin:$PATH
```

Configure Cassandra:

Initially we have to do the below steps when we install in the shared drive.

Within CONF folder of Cassandra

As we would later move the files, this setting up will be common for all, apart from a few changes depending upon the node like (rpcaddress,listenaddress).

1. Navigate to the extracted Cassandra directory and configure the necessary settings.

Setting up the Cassandra environment file

Now navigate to **cassandra-env.sh** (environment file):

cd /temp/cs4224b/apache-cassandra-4.1.3/conf/ cassandra-env.sh

now once you are in the file, scroll down to the bottom and paste the below texts

```
JVM_OPTS="$JVM_OPTS $JVM_EXTRA_OPTS"
JVM_OPTS="$JVM_OPTS -Dcassandra.ignore_dc=true"
JVM_OPTS="$JVM_OPTS -Dcassandra.ignore_rack=true"
```

The two JVM options provided in your configuration scripts for Apache Cassandra, `-Dcassandra.ignore_dc=true` and `-Dcassandra.ignore_rack=true`, help to avoid potential errors in the following scenarios:

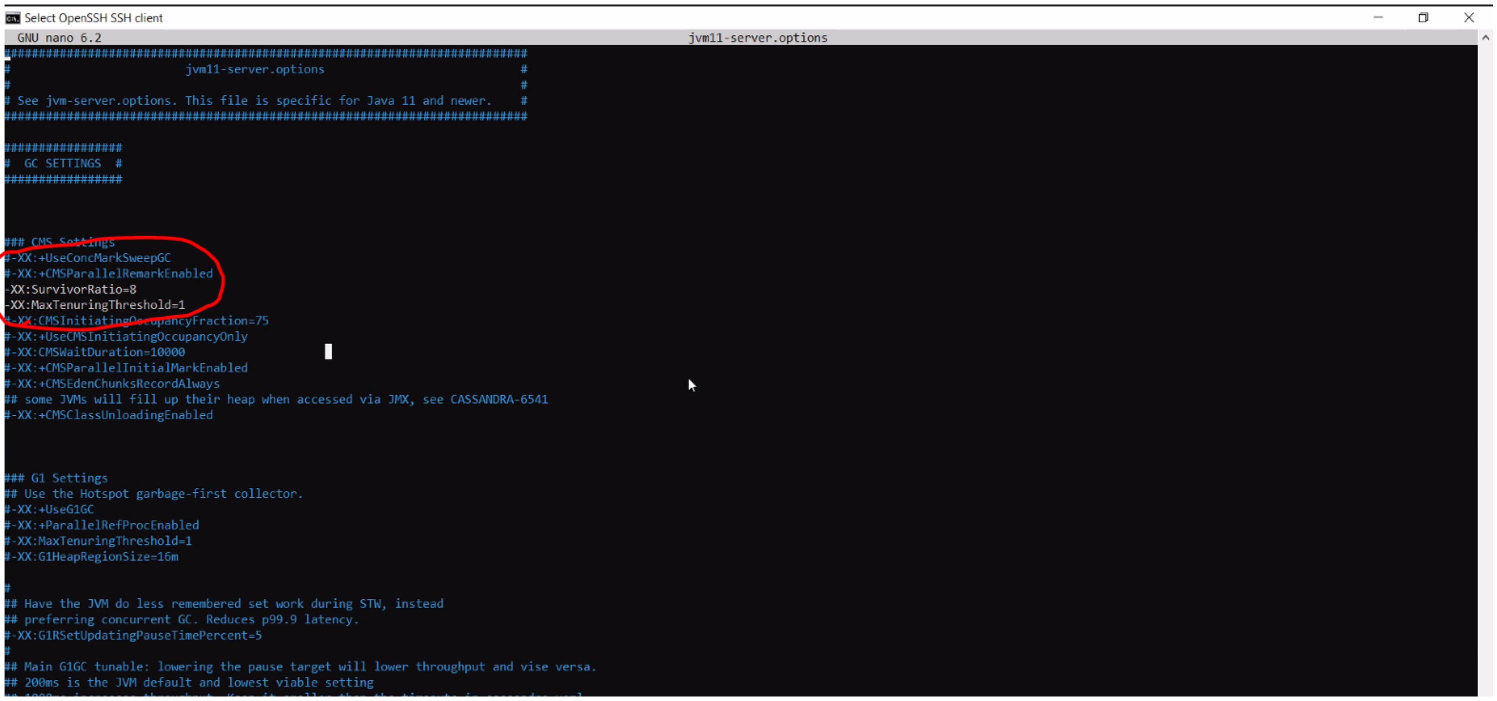
1. **`-Dcassandra.ignore_dc=true`:**
   - Avoids errors related to data center (DC) configuration during startup.
   - Useful when setting up Cassandra in non-production environments where data center information is not relevant or needs to be bypassed.
2. **`-Dcassandra.ignore_rack=true`:**
   - Prevents errors associated with rack configuration during startup.
   - Helpful in development or testing phases where the rack information may not be necessary or needs to be bypassed.

By using these options, you can streamline the setup process and avoid unnecessary configuration-related errors, particularly in non-production environments where a simplified configuration may be more practical.

2. <u>we have to open **jvm11-server.options** now.</u>

Uncomment the following line and comment the rest :

```
-XX:SurvivorRatio=8
-XX:MaxTenuringThreshold=1
```

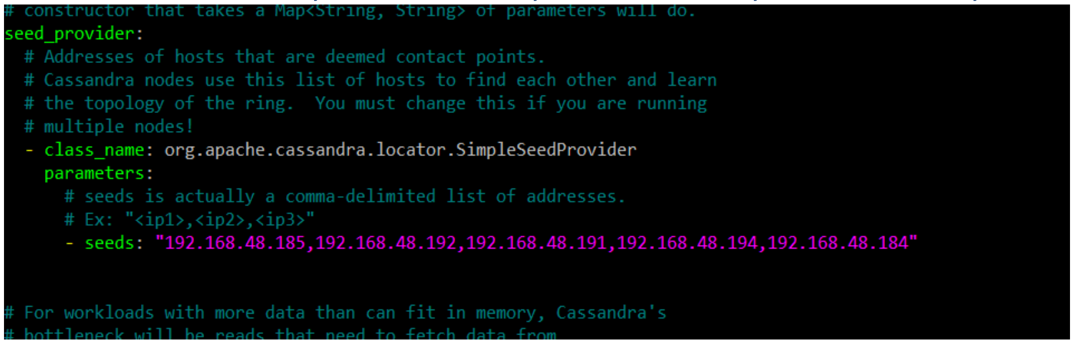


```
Select OpenSSH SSH client
  GNU nano 6.2                                                                                    jvm11-server.options
###################################################################################
#                           jvm11-server.options                                  #
#                                                                                 #
# See jvm-server.options. This file is specific for Java 11 and newer.           #
###################################################################################

#################
#  GC SETTINGS  #
#################



### CMS Settings
#-XX:+UseConcMarkSweepGC
#-XX:+CMSParallelRemarkEnabled
-XX:SurvivorRatio=8
-XX:MaxTenuringThreshold=1
#-XX:CMSInitiatingOccupancyFraction=75
#-XX:+UseCMSInitiatingOccupancyOnly
#-XX:CMSWaitDuration=10000
#-XX:+CMSParallelInitialMarkEnabled
#-XX:+CMSEdenChunksRecordAlways
## some JVMs will fill up their heap when accessed via JMX, see CASSANDRA-6541
#-XX:+CMSClassUnloadingEnabled



### G1 Settings
## Use the Hotspot garbage-first collector.
#-XX:+UseG1GC
#-XX:+ParallelRefProcEnabled
#-XX:MaxTenuringThreshold=1
#-XX:G1HeapRegionSize=16m

#
## Have the JVM do less remembered set work during STW, instead
## preferring concurrent GC. Reduces p99.9 latency.
#-XX:G1RSetUpdatingPauseTimePercent=5
#
## Main G1GC tunable: lowering the pause target will lower throughput and vise versa.
## 200ms is the JVM default and lowest viable setting
```

3. <u>Next Navigate to **cassandra.yaml** file</u>

1.) Keep scrolling until you arrive at **seed_provider** section, and here you would find the variable 'seeds', were we will have to declare all the nodes we want to connect .

Enter the IP-ADDRESS OF ALL 5 NODES and the 'seeds' values should remain constant across all the seed_provider section in all files.
For us the seeds value looked like this: -
seeds: "192.168.48.185, 192.168.48.192, 192.168.48.191, 192.168.48.194, 192.168.48.184"



```
# constructor that takes a Map<String, String> of parameters will do.
seed_provider:
  # Addresses of hosts that are deemed contact points.
  # Cassandra nodes use this list of hosts to find each other and learn
  # the topology of the ring.  You must change this if you are running
  # multiple nodes!
  - class_name: org.apache.cassandra.locator.SimpleSeedProvider
    parameters:
      # seeds is actually a comma-delimited list of addresses.
      # Ex: "<ip1>,<ip2>,<ip3>"
      - seeds: "192.168.48.185,192.168.48.192,192.168.48.191,192.168.48.194,192.168.48.184"


# For workloads with more data than can fit in memory, Cassandra's
# bottleneck will be reads that need to fetch data from
```

2.) Now in the same file '**cassandra.yaml',** don't forget to change the values of these variables to IP address of that node.

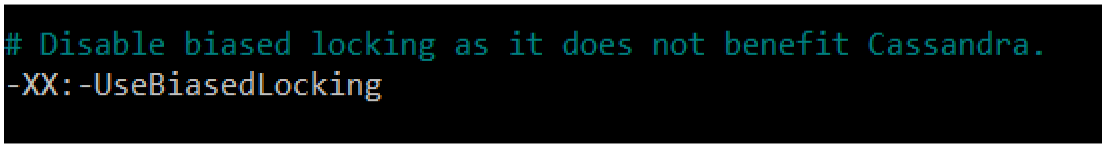
1. **listen_address**
2. **rpc_address**

3.) then change, the variable value of **endpoint_snitch** to RackInferringSnitch.

```
endpoint_snitch: RackInferringSnitch
```

4. <u>now move to **jvm-server.options** file.</u>
   Uncomment the line below:

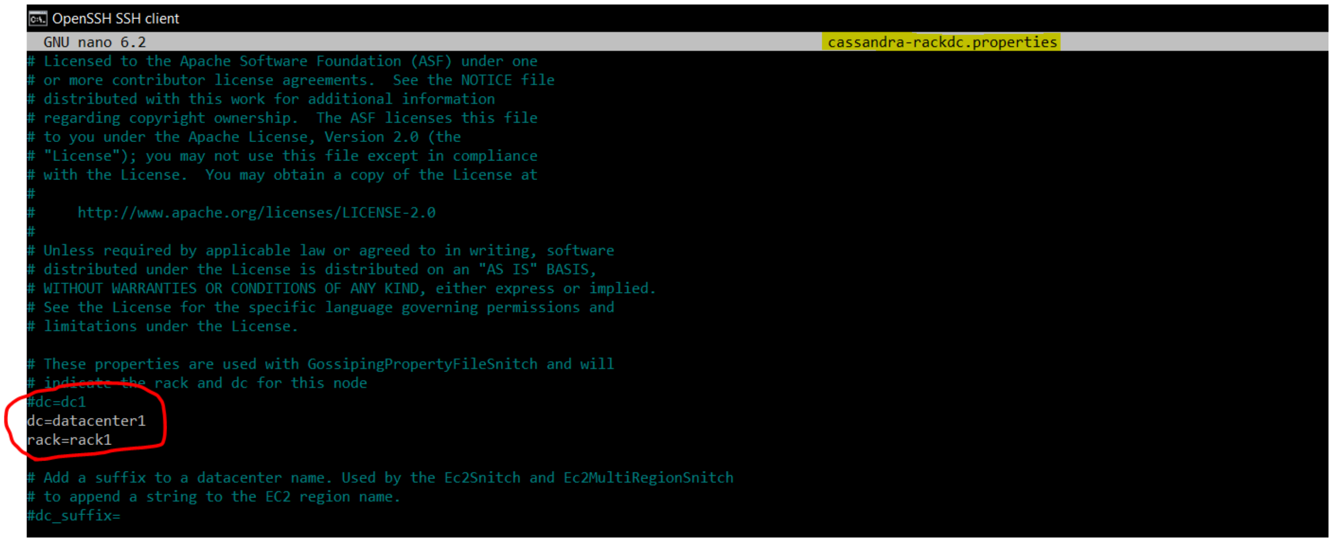
```
-XX:-UseBiasedLocking
```



```
# Disable biased locking as it does not benefit Cassandra.
-XX:-UseBiasedLocking
```

5. Open cassandra-rackdc.properties.
   Give values for dc and rack.

```
dc=datacenter1
rack=rack1
```



OpenSSH SSH client

```
  GNU nano 6.2                                                                      cassandra-rackdc.properties
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# These properties are used with GossipingPropertyFileSnitch and will
# indicate the rack and dc for this node
#dc=dc1
dc=datacenter1
rack=rack1

# Add a suffix to a datacenter name. Used by the Ec2Snitch and Ec2MultiRegionSnitch
# to append a string to the EC2 region name.
#dc_suffix=
```

Within BIN folder of Cassandra

1. We have to open **cassandra.in.sh** now.
   We should manually give the java path here:

```
JAVA_HOME=/home/stuproj/cs4224b/jdk-11.0.2/
```

Multinode Cassandra setup

Once the Basic setup is completed is completed in the shared drive, copy the Cassandra from from shared drive to temp folder and then to all other nodes individually.

We used the following line to perform the above function.

```
cp -/temp/cs4224b/apache-cassandra-4.1.3 .
```

Once Cassandra is copied, you would have to open the cassandra.yaml in conf folder of Cassandra in each node separately and make the changes to variable (seeds within seed_provider, listen_address,rpc_address).

After all this is done, you can start Cassandra in the nodes individually but around the same time.

Now within each node, navigate to the folder where Cassandra is installed and execute the below commands.

```
nohup ./cassandra -f &
```

if you want to see the logs while starting Cassandra, use this : `tail -f nohup.out`
then you can run : `nodetool status`, to see if Cassandra is interconnected in all nodes.

Creating Keyspaces:

```
CREATE KEYSPACE CS4224b WITH replication = {'class': 'SimpleStrategy','replication_factor' :3};
```

Steps to create the tables and insert data

The create_conn.py file contains the code to establish a connection and create a session with the Cassandra database. The create_tables.py can be executed to create all the tables. `python create_tables.py` The following scripts perform the task of inserting the data into the respective tables.

1. insert_data_item.py
2. insert_data_order_customer.py
3. insert_data_order_customer_no_carrier.py
4. insert_data_order_line.py
5. insert_data_related_customer_order_line_new.py
6. insert_data_stock.py
7. transform_input_data.py

These files can be executed using the following commands `python <file_name>`

Steps to run the transactions

The following scripts contain the commands to run the transaction files concurrently.

1. parallel_0.sh
2. parallel_1.sh
3. parallel_2.sh
4. parallel_3.sh
5. parallel_4.sh

These scripts can be executed using the following command ./<filename>