****

Cassandra :: Multi-Node Cluster

****

Apache Cassandra is a highly scalable open source database system, achieving great performance on multi-node setups.

Previously, we went over how to run a single-node Cassandra cluster. In this tutorial, you'll learn how to install and use Cassandra to run a multi-node cluster on Ubuntu 14.04.

Prerequisites

1) ssh user without password authentication
2) install cassendra on all nodes with single-node installation

in my case, i used 2 nodes.

1) 192.168.1.253 ubuntu-server.com server
2) 192.168.1.254 ubuntu-client.com client
ssh-user without password authentication ubuntu

****

Setup SSH for Auto Login without a Password

****

on both server :

 # adduser ubuntu

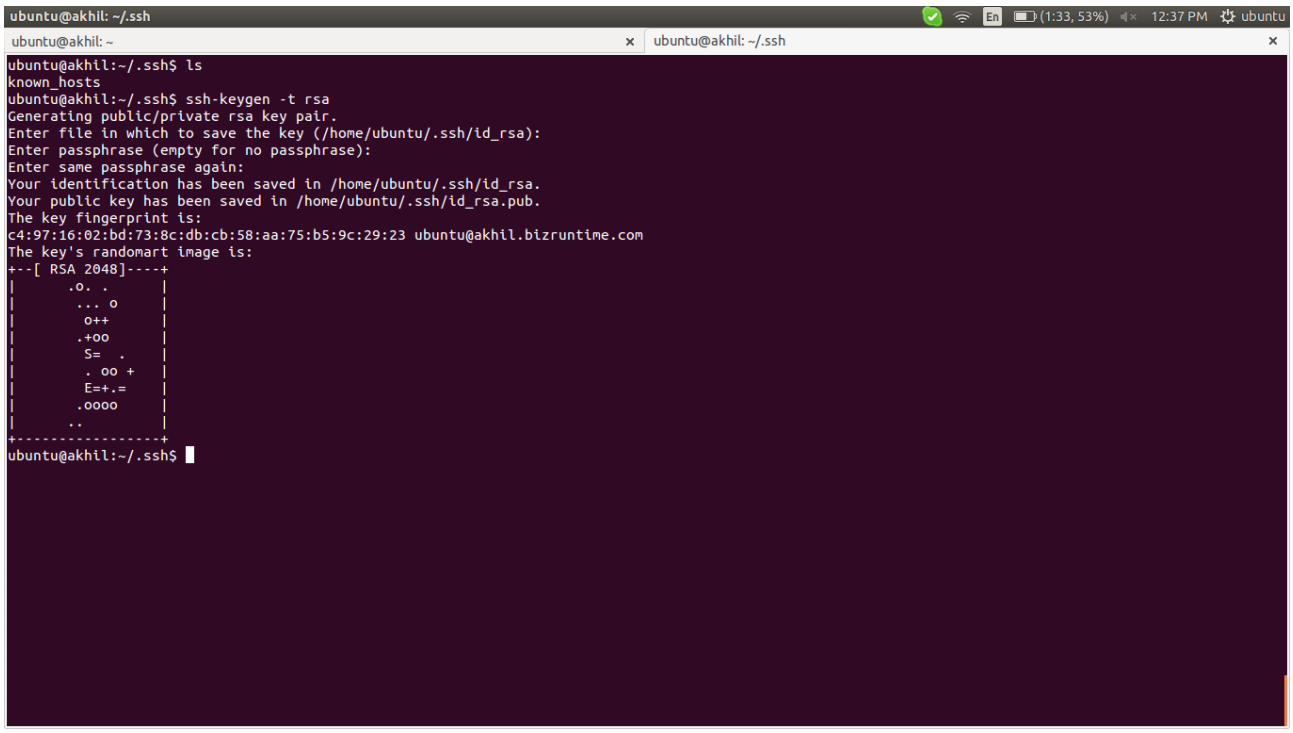
give all details for this user in both machine

 # usermod -aG sudo ubuntu
 # visudo

add this line at the bottom of this file

ubuntu ALL=(ALL) NOPASSWD: ALL

 # /etc/init.d/ssh restart
 # su ubuntu
 # cd /home/ubuntu/.ssh/
 # ssh-keygen -t rsa



on server-side:

 # sudo ssh-copy-id 192.168.1.254

on client-side:

 # cd /home/ubuntu/.ssh/
 # cat id_rsa.pub >> authorized_keys

copy this output

on server-side:

 # cd /home/ubuntu/.ssh/
 # nano authorized_keys

paste that in this file.

On both server :

 # sudo /etc/init.d/ssh restart

****

links: http://www.rebol.com/docs/ssh-auto-login.html

****

****

Cassandra Configuration

****

On both machine do this,

 # su ubuntu

Cassandra requires that the Oracle Java SE Runtime Environment (JRE) be installed. So, in this step, you'll install and verify that it's the default JRE.

 # sudo add-apt-repository ppa:webupd8team/java

 # sudo apt-get update

Then install the Oracle JRE. Installing this particular package not only installs it but also makes it the default JRE. When prompted, accept the license agreement:

 # sudo apt-get install oracle-java8-set-default
 # java -version

You should see output similar to the following:

Output

java version "1.8.0_60"
Java(TM) SE Runtime Environment (build 1.8.0_60-b27)
Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)

We'll install Cassandra using packages from the official Apache Software Foundation repositories, so start by adding the repo so that the packages are available to your system. Note that Cassandra 2.2.2 is the latest version at the time of this publication. Change the 22x to match the latest version. For example, use 39x if Cassandra 3.9 is the latest version:

 # echo "deb http://www.apache.org/dist/cassandra/debian 39x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list

 # echo "deb-src http://www.apache.org/dist/cassandra/debian 39x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list

****

links ::

latest version ::
http://cassandra.apache.org/download/

****

 # sudo apt-get update

Finally, install Cassandra:

 # sudo apt-get install cassandra

Ordinarily, Cassandra should have been started automatically at this point. However, because of a bug, it does not. To confirm that it's not running, type:

 # sudo service cassandra status
 # sudo service cassandra restart

wait for few minutes and check,

 # sudo tailf /var/log/cassandra/system.log

 # sudo service cassandra status

If it is not running, the following output will be displayed:

Output

**** could not access pidfile for Cassandra***

and check the log file also which mentioned above.

Then, resolve this issue by,

 # sudo nano +60 /etc/init.d/cassandra

That line should read:

 # sudo nano /etc/init.d/cassandra

CMD_PATT="cassandra.+CassandraDaemon"

Change it to:

CMD_PATT="cassandra"

Close and save the file, then reboot the server:

 # sudo reboot

After logging back in, Cassandra should now be running. Verify:

 # sudo service cassandra status

If you are successful, you will see:

Output

**** Cassandra is running***

If you were able to successfully start Cassandra, check the status of the cluster:

 # sudo nodetool status

In the output, **UN** means it's **U**p and **N**ormal:

Output

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens       Owns    Host ID                               Rack
UN  127.0.0.1  142.02 KB  256          ?       2053956d-7461-41e6-8dd2-0af59436f736  rack1
```

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless

Then connect to it using its interactive command line interface cqlsh.

 # cqlsh

You will see it connect:
Output

```
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 2.2.2 | CQL spec 3.3.1 | Native protocol v4]
Use HELP for help.
cqlsh>
```

Type exit to quit:

> exit

****
links ::

https://www.digitalocean.com/community/tutorials/how-to-install-cassandra-and-run-a-single-node-cluster-on-ubuntu-14-04
****

The first command you'll run on each node will stop the Cassandra daemon.

 # sudo service cassandra stop

When that's completed, delete the default dataset.

 # sudo rm -rf /var/lib/cassandra/data/*
 # sudo mkdir /var/lib/cassandra/data/system
 # sudo chown -R cassandra:cassandra /var/lib/cassandra/data

Configuring the Cluster

Cassandra's configuration file is located in the /etc/cassandra directory. That configuration file, cassandra.yaml, contains many directives and is very well commented. In this step, we'll modify that file to set up the cluster.
Only the following directives need to be modified to set up a multi-node Cassandra cluster:

- cluster_name: This is the name of your cluster.
- -seeds: This is a comma-delimited list of the IP address of each node in the cluster.
- listen_address: This is IP address that other nodes in the cluster will use to connect to this one. It defaults to **localhost** and needs changed to the IP address of the node.
- rpc_address: This is the IP address for remote procedure calls. It defaults to **localhost**. If the server's hostname is properly configured, leave this as is. Otherwise, change to server's IP address or the loopback address (127.0.0.1).
- endpoint_snitch: Name of the snitch, which is what tells Cassandra about what its network looks like. This defaults to **SimpleSnitch**, which is used for networks in one datacenter. In our case, we'll change it to **GossipingPropertyFileSnitch**, which is preferred for production setups.
- auto_bootstrap: This directive is not in the configuration file, so it has to be added and set to **false**. This makes new nodes automatically use the right data. It is optional if you're adding nodes to an existing cluster, but required when you're initializing a fresh cluster, that is, one with no data.

Open the configuration file for editing using nano or your favorite text editor.

 # sudo nano /etc/cassandra/cassandra.yaml

Search the file for the following directives and modify them as below to match your cluster. Replace your_server_ip with the IP address of the server you're currently working on. The - seeds: list should be the same on every server, and will contain each server's IP address separated by commas.

. . .

cluster_name: 'CassandraDOCluster'

. . .

```
seed_provider:
  - class_name: org.apache.cassandra.locator.SimpleSeedProvider
    parameters:
        - seeds: "your_server_ip,your_server_ip_2,...your_server_ip_n"
```

. . .

listen_address: your_server_ip

. . .

rpc_address: your_server_ip

. . .

endpoint_snitch: GossipingPropertyFileSnitch

. . .

At the bottom of the file, add in the auto_bootstrap directive by pasting in this line:

auto_bootstrap: false

example :

on server-side:

. . .

cluster_name: 'bizruntime'

. . .

seed_provider:
- class_name: org.apache.cassandra.locator.SimpleSeedProvider
parameters:
- seeds: "192.168.1.253"

. . .

listen_address: 192.168.1.253

. . .

rpc_address: 192.168.1.253

. . .

endpoint_snitch: GossipingPropertyFileSnitch

. . .
At the bottom of the file, add in the auto_bootstrap directive by pasting in this line:

auto_bootstrap: false

on client -side:

. . .

cluster_name: 'bizruntime'

. . .

```
seed_provider:
  - class_name: org.apache.cassandra.locator.SimpleSeedProvider
    parameters:
        - seeds: "192.168.1.253"
```

. . .

listen_address: 192.168.1.254

. . .

rpc_address: 192.168.1.254

. . .

endpoint_snitch: GossipingPropertyFileSnitch

. . .

At the bottom of the file, add in the auto_bootstrap directive by pasting in this line:
auto_bootstrap: false

 # sudo service cassandra start

If you check the status of the cluster, you'll find that only the local node is listed, because it's not yet able to communicate with the other nodes.

 # sudo nodetool status

```
Datacenter: dc1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address       Load       Tokens  Owns    Host ID                               Rack
UN  192.168.1.253  147.48 KB  256     ?       f50799ee-8589-4eb8-a0c8-241cd254e424  rack1
UN  192.168.1.254  139.04 KB  256     ?       54b16af1-ad0a-4288-b34e-cacab39caeec  rack1
```

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless

You can also check if you can connect to the cluster using cqlsh, the Cassandra command line client. Note that you can specify the IP address of any node in the cluster for this command.

 # cqlsh your_server_ip 9042

example :

 # cqlsh 192.168.1.253 9042

Connected to bizruntime at 192.168.1.253:9042.
[cqlsh 5.0.1 | Cassandra 3.9 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
cqlsh>

****
links ::
https://www.digitalocean.com/community/tutorials/how-to-run-a-multi-node-cluster-database-with-cassandra-on-ubuntu-14-04
****

****
testing ::
****

login

 # cqlsh 192.168.1.253 9042

create keyspace with replication
> CREATE KEYSPACE biz WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 2 };

check on both server
> SELECT * FROM system_schema.keyspaces;

to use that keyspace (database)
> use biz;

create tables
> use biz;
> create table emp (empid int primary key,
... emp_first varchar, emp_last varchar, emp_dept varchar);

add another values to that table
> insert into emp (empid, emp_first, emp_last, emp_dept)
... values (1,'fred','smith','eng');

check from both server
> select * from emp;

```
 empid | emp_dept | emp_first | emp_last
-------+----------+-----------+----------
     1 |      eng |      fred |    smith

(1 rows)
```

****
link ::
to create tables >>
http://datascale.io/cassandra-partitioning-and-clustering-keys-explained/
****

****
multiple datacenters
****

This topic contains information for deploying a Cassandra cluster with multiple data centers. In Cassandra, the term datacenter is a grouping of nodes. datacenter is synonymous with replication group, that is, a grouping of nodes configured together for replication purposes.

Prerequisites

Each node must be correctly configured before starting the cluster. You must determine or perform the following before starting the cluster:

- A good understanding of how Cassandra works. Be sure to read at least Understanding the architecture, Data replication, and Cassandra's rack feature.
- Install Cassandra on each node.
- Choose a name for the cluster.
- Get the IP address of each node.
- Determine which nodes will be seed nodes. **Do not make all nodes seed nodes.** Please read Internode communications (gossip).
- Determine the snitch and replication strategy. The GossipingPropertyFileSnitch and NetworkTopologyStrategy are recommended for production environments.
- If using multiple datacenters, determine a naming convention for each data center and rack, for example: DC1, DC2 or 100, 200 and RAC1, RAC2 or R101, R102. Choose the name carefully; renaming a datacenter is not possible.
- Other possible configuration settings are described in cassandra.yaml configuration file and property files such as cassandra-rackdc.properties.

Procedure

1. Suppose you install Cassandra on these nodes:
   node0 10.168.66.41 (seed1)
   node1 10.176.43.66
   node2 10.168.247.41
   node3 10.176.170.59 (seed2)
   node4 10.169.61.170
   node5 10.169.30.138
   Note: It is a best practice to have more than one seed node per datacenter.
2. If you have a firewall running in your cluster, you must open certain ports for communication between the nodes. See Configuring firewall port access.
3. If Cassandra is running, you must stop the server and clear the data:
   Doing this removes the default cluster_name (Test Cluster) from the system table. All nodes must use the same cluster name.
   Package installations:
   a. Stop Cassandra:
      # sudo service cassandra stop
   b. Clear the data:
      # sudo rm -rf /var/lib/cassandra/data/system/*

4. Set the properties in the cassandra.yaml file for each node:
   Note: After making any changes in the cassandra.yaml file, you must restart the node for the changes to take effect.
   Properties to set:
   a. num_tokens: *recommended value: 256*
   b. -seeds: *internal IP address of each seed node*
      Seed nodes do not bootstrap, which is the process of a new node joining an existing cluster. For new clusters, the bootstrap process on seed nodes is skipped.
   c. listen_address:
      If not set, Cassandra asks the system for the local address, the one associated with its hostname. In some cases Cassandra doesn't produce the correct address and you must specify the listen_address.
   d. endpoint_snitch: *name of snitch* (See endpoint_snitch.) If you are changing snitches, see Switching snitches.
   e. auto_bootstrap: false (Add this setting **only** when initializing a fresh cluster with no data.)
   Note: If the nodes in the cluster are identical in terms of disk layout, shared libraries, and so on, you can use the same copy of the cassandra.yaml file on all of them.
   Example:
   cluster_name: 'MyCassandraCluster'
   num_tokens: 256
   seed_provider:
   - class_name: org.apache.cassandra.locator.SimpleSeedProvider
   parameters:
   - seeds: "10.168.66.41,10.176.170.59"
   listen_address:
   endpoint_snitch: GossipingPropertyFileSnitch
   Note: Include at least one node from *each* datacenter.
5. In the cassandra-rackdc.properties file, assign the data center and rack names you determined in the Prerequisites. For example:
   Nodes 0 to 2

indicate the rack and dc for this node
dc=DC1
rack=RAC1
Nodes 3 to 5
indicate the rack and dc for this node
dc=DC2
rack=RAC1

6. After you have installed and configured Cassandra on all nodes, start the seed nodes one at a time, and then start the rest of the nodes.
   Note: If the node has restarted because of automatic restart, you must first stop the node and clear the data directories, as described above.
   Package installations:
   # sudo service cassandra start
   Tarball installations:
   # cd install_location
   # bin/cassandra
7. To check that the ring is up and running, run:
   Package installations:
   # nodetool status

to create keyspace with particular replication in each datacenter
use “NetworkTopologyStrategy” instead of “SimpleStrategy” for create keyspace

> CREATE KEYSPACE "Excalibur" WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 2};

****
link ::
https://docs.datastax.com/en/cassandra/2.1/cassandra/initialize/initializeMultipleDS.html
https://www.packtpub.com/books/content/apache-cassandra-working-multiple-datacenter-environments

commands for create keyspace in multi-datacenter ::
https://docs.datastax.com/en/cql/3.1/cql/cql_reference/create_keyspace_r.html
****

****

Cassandra Data partitioning

****

Cassandra is a distributed database that runs on multiple nodes. When you write data to the cluster, **partitioning scheme determines which node in the cluster stores that data**. For example, suppose you are inserting some data (Column-Value pair identified by a Row Key). Data partitioning protocol will dictate which node in the cluster is responsible for storing this data. Similarly, when you request data, the partitioning protocol will examine the Row Key and find the node in the cluster responsible for the row key and retrieve data from it.

Difference between Partitioning and Replication?

Data partitioning is concerned with picking a node in the cluster to store the first copy of data on. Replication determines number of additional nodes that will store the same data (for performance and fault tolerance reasons). Replication is discussed in the next section.

Partitioning => Picking out one node to store first copy of data on

Replication => Picking out additional nodes to store more copies of data

When you deploy a Cassandra cluster, you must assign a partitioner and assign each node an *initial_token* value so each node is responsible for roughly an equal amount of data (load balancing). DataStax strongly recommends using the RandomPartitioner (default) for all cluster deployments.

To calculate the tokens for nodes in a single data center cluster, you divide the range by the total number of nodes in the cluster. In multiple data center deployments, you calculate the tokens such that each data center is individually load balanced. See *Generating Tokens* for the different approaches to generating tokens for nodes in single and multiple data center clusters.

Unlike almost every other configuration choice in Cassandra, the partitioner may not be changed without reloading all of your data. Therefore, it is important to choose and configure the correct partitioner before initializing your cluster. You set the partitioner in the *cassandra.yaml* file.

1) Murmur3Partitioner

****

link ::

http://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architecturePartitionerM3P_c.html

****

2) RandomPartitioner

The RandomPartitioner uses tokens to help assign equal portions of data to each node and evenly distribute data from all the tables throughout the ring or other grouping, such as a keyspace. This is true even if the tables use different row keys, such as usernames or timestamps. Moreover, the read and write requests to the cluster are also evenly distributed and load balancing is simplified because each part of the hash range receives an equal number of rows on average. The RandomPartitioner (org.apache.cassandra.dht.RandomPartitioner) is the default partitioning strategy for a Cassandra cluster, and in almost all cases is the right choice. The RandomPartition distributes data evenly across the nodes using an MD5 hash value of the row key. The possible range of hash values is from 0 to 2^{127} -1.

****

link ::

http://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architecturePartitionerRandom_c.html

http://docs.datastax.com/en/archived/cassandra/1.1/docs/cluster_architecture/partitioning.html#data-distribution-in-the-ring

****

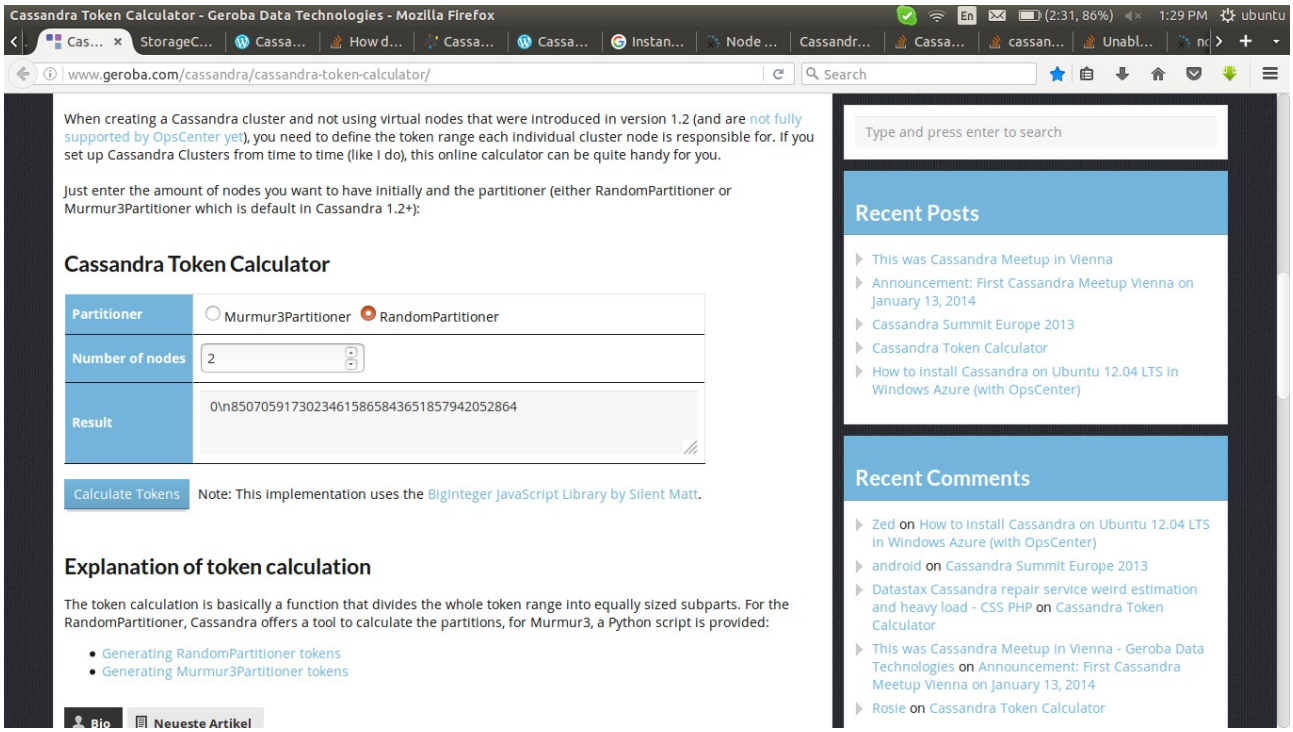
Cassandra Token Calculator

****

link ::

http://www.geroba.com/cassandra/cassandra-token-calculator/

****



here i created token for 2 node,
0\n85070591730234615865843651857942052864

for server-node : 0
for client-node : 85070591730234615865843651857942052864

configuration

The selection of partitioner need to be take before start the replication
all other configuration parameters in this file same as i done earlier. Make changes only below lines in that configuration file.

Step 1 - Calculate the token

****
links ::
http://www.geroba.com/cassandra/cassandra-token-calculator/
****

Step 2 - Reconfigure the token properties in cassandra.yaml:

- comment out num_tokens
- set the token assignment initial_token
- leave auto_bootstrap: true

example ::

 # sudo service cassandra stop

 # sudo nano /etc/cassandra/cassandra.yaml

num_tokens:
...

initial_token 85070591730234615865843651857942052864
....
auto_bootstrap: true

remove all data form server

 # sudo rm -rf /var/lib/cassandra/data/*
 # sudo mkdir /var/lib/cassandra/data/system
 # sudo chown -R cassandra:cassandra /var/lib/cassandra/data

start the server and check the error log.

 # service cassandra restart
 # tailf /var/log/cassandra/system.log
 # service cassandra status

if all working well, then :
(here i done with RandomPartitioner)

 # sudo nodetool status

Datacenter: dc1
==========
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address Load Tokens Owns (effective) Host ID Rack
UN 192.168.1.253 219.24 KiB 1 50.0% 21828972-91c2-403d-a228-1537b1630c99 rack1
UN 192.168.1.254 232.89 KiB 1 50.0% f8fc523a-2786-4e74-8a82-75b0ffd67586 rack1

here the owns shows 50 %

****
link ::
detailed output info ::
https://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsStatus.html
****

if there is any error while checking the log,
error :: Unable to start Cassandra: “node already exists”

 # find / -name cassandra-env.sh

 # nano /path/to/cassandra-env.sh

add this at bottom of file

JVM_OPTS="$JVM_OPTS -Dcassandra.replace_address=192.168.1.253"

 # service cassandra restart
 # tailf /var/log/cassandra/system.log
 # service cassandra status

Don't forget to remove it once your done.

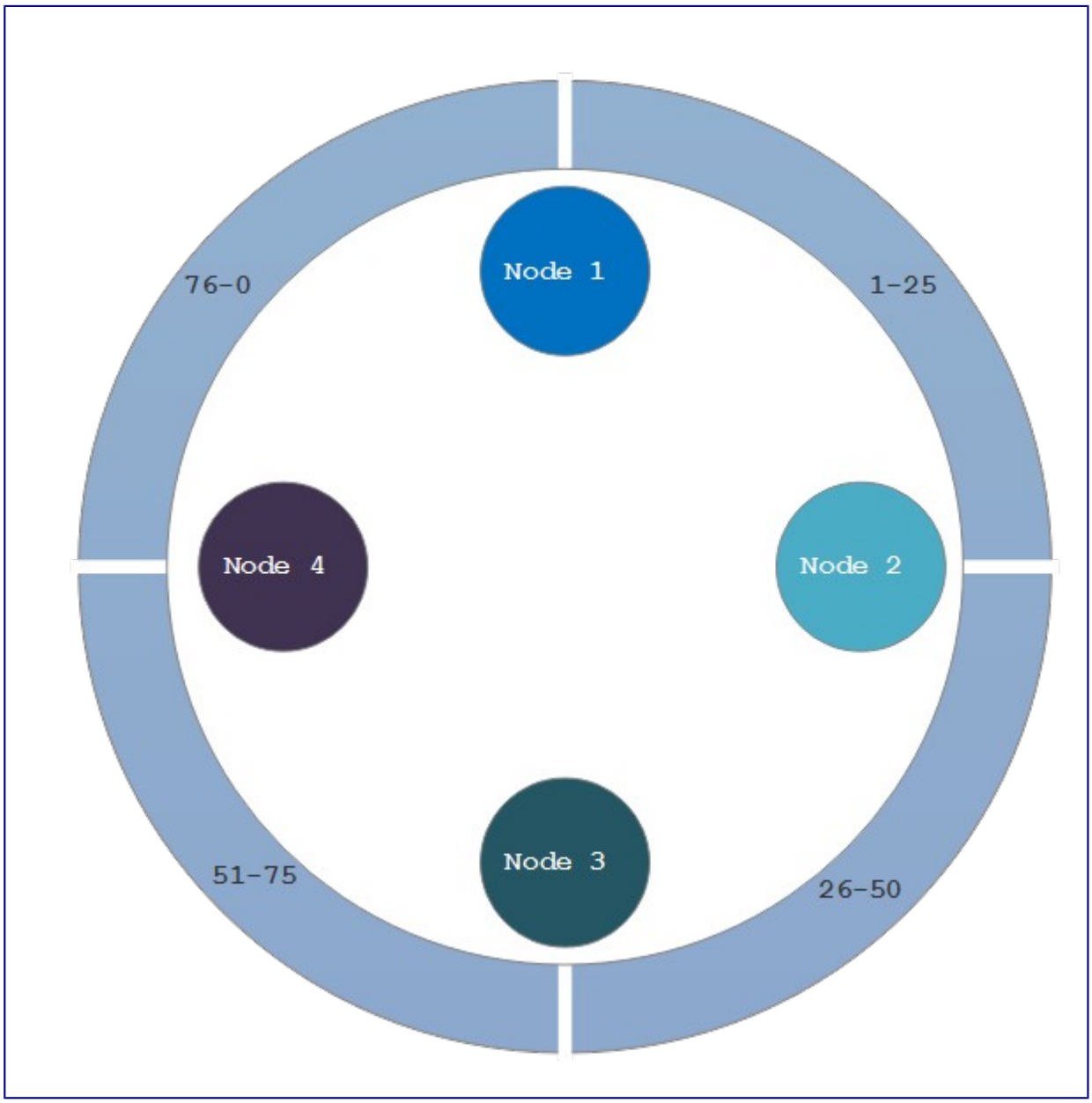
****
Link ::
http://stackoverflow.com/questions/29323709/unable-to-start-cassandra-node-already-exists
****

Cassandra Partitioning & Clustering Keys Explained

Primary Keys are defined when you create your table. The most basic primary key is a single column. A single column is great for when you know the value that you will be searching for

Partition Key

The Partition Key is responsible for the distribution of data amongst the nodes. Let's look back to an earlier post on Cassandra Data Model Basics, in which I described a four node cluster, as shown below. For simplicities' sake, let's assume hash values are between 0-100. When we insert the first row into the crossfit_gyms table, the value of gym_name will be hashed. Let's also assume that the first record will have a hash of 34. That will fall into the values that Node 2's partition is assigned. So the value of the Partition Key, 34, indicates the partition, 26-50, in the cluster/ring that the piece of data will be stored. Makes sense, huh?



Compound Keys
But wait, there's more! Primary keys can also be more than one column. A multi-column primary key is called a Compound Key.

****

testing ::

****

login to server
 # cqlsh 192.168.1.253 9042

create a keyspace
> CREATE KEYSPACE biz WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };

check whether the keyspace created or not
> SELECT * FROM system_schema.keyspaces;

get into that keyspace
> use biz;

create a table with partion key for partitioning the data

```
> CREATE TABLE crossfit_gyms_by_location (
        country_code text,
        state_province text,
        city text,
        gym_name text,
        PRIMARY KEY (country_code, state_province, city, gym_name)
) WITH CLUSTERING ORDER BY (state_province DESC, city ASC, gym_name ASC);
```

add the values to the table for testing

```
> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('USA', 'CA', 'San Francisco', 'San Francisco CrossFit');

> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('USA', 'CA', 'San Francisco', 'LaLanne Fitness CrossFit');

> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('USA', 'NY', 'New York', 'CrossFit NYC');

> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('USA', 'NY', 'New York', 'CrossFit Metropolis');

> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('USA', 'NV', 'Las Vegas', 'CrossFit Las Vegas');

> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('USA', 'NV', 'Las Vegas', 'Kaizen CrossFit');
```

```
> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('CAN', 'ON', 'Toronto', 'CrossFit Toronto');

> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('CAN', 'ON', 'Toronto', 'CrossFit Leslieville');

> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('CAN', 'BC', 'Vancouver', 'CrossFit Vancouver');

> INSERT INTO crossfit_gyms_by_location (country_code, state_province, city, gym_name) VALUES ('CAN', 'BC', 'Vancouver', 'CrossFit BC');
```

check the data replicated on all node

```
> select * from crossfit_gyms_by_location;

 country_code | state_province | city          | gym_name
--------------+----------------+---------------+--------------------------
          USA |             NY |      New York |      CrossFit Metropolis
          USA |             NY |      New York |           CrossFit NYC
          USA |             NV |     Las Vegas |       CrossFit Las Vegas
          USA |             NV |     Las Vegas |         Kaizen CrossFit
          USA |             CA | San Francisco | LaLanne Fitness CrossFit
          USA |             CA | San Francisco |    San Francisco CrossFit
          CAN |             ON |       Toronto |     CrossFit Leslieville
          CAN |             ON |       Toronto |        CrossFit Toronto
          CAN |             BC |     Vancouver |             CrossFit BC
          CAN |             BC |     Vancouver |       CrossFit Vancouver

(10 rows)
```

here i use " country_code" as the partion key
to see the harshed values of " country_code"

```
>  SELECT token(country_code),country_code FROM crossfit_gyms_by_location;

 system.token(country_code)           | country_code
------------------------------------------+--------------
  1147721109762333499202743075838748414 0 |          USA
  1147721109762333499202743075838748414 0 |          USA
  1147721109762333499202743075838748414 0 |          USA
  1147721109762333499202743075838748414 0 |          USA
  1147721109762333499202743075838748414 0 |          USA
  1147721109762333499202743075838748414 0 |          USA
 1453028775038189367033357764616591361 35 |          CAN
 1453028775038189367033357764616591361 35 |          CAN
 1453028775038189367033357764616591361 35 |          CAN
 1453028775038189367033357764616591361 35 |          CAN


> exit
```

 # nodetool getendpoints keyspace table key-value

ubuntu@ubuntu-client:/root$ nodetool getendpoints biz crossfit_gyms_by_location USA
192.168.1.254

ubuntu@ubuntu-client:/root$ nodetool getendpoints biz crossfit_gyms_by_location CAN
192.168.1.253

or search with the harshed value of key

ubuntu@ubuntu-client:/root$ nodetool getendpoints biz crossfit_gyms_by_location 114772110976233349920274307583874841 40
192.168.1.254

ubuntu@ubuntu-client:/root$ nodetool getendpoints biz crossfit_gyms_by_location 145302877503818936703335776461659136135
192.168.1.253

****

link ::

to add table ::
http://datascale.io/cassandra-partitioning-and-clustering-keys-explained/

data partitioning ::
http://docs.datastax.com/en/archived/cassandra/1.1/docs/cluster_architecture/partitioning.html#data-distribution-in-the-ring

http://docs.datastax.com/en/archived/cassandra/1.1/docs/configuration/node_configuration.html#initial-token

partitioning keys ::
https://jagadeeshs.wordpress.com/2015/07/15/cassandra/
https://dzone.com/articles/cassandra-data-modeling-primary-clustering-partiti

type of partitioner ::
https://10kloc.wordpress.com/2012/12/27/cassandra-chapter-4-data-partitioning/

token ::
http://docs.datastax.com/en/archived/cassandra/1.1/docs/initialize/token_generation.html#token-gen-cassandra

http://docs.datastax.com/en/archived/cassandra/1.1/docs/initialize/token_generation.html#token-gen-cassandra

partitioning configuration ::
https://support.datastax.com/hc/en-us/articles/205199715-Node-startup-fails-with-error-ConfigurationException-Cannot-change-the-number-of-tokens-

visual testing ::
http://psanford.github.io/cassandra-visual-ring/

testing ::
http://stackoverflow.com/questions/30514237/what-node-does-cassandra-store-data-on

****

****

All links ::

tutorial ::
https://www.tutorialspoint.com/cassandra/cassandra_quick_guide.htm

latest version check ::
http://cassandra.apache.org/download/

installation and configuration ::
https://www.digitalocean.com/community/tutorials/how-to-run-a-multi-node-cluster-database-with-cassandra-on-ubuntu-14-04

https://www.digitalocean.com/community/tutorials/how-to-install-cassandra-and-run-a-single-node-cluster-on-ubuntu-14-04

configuration directives ::
http://docs.datastax.com/en/archived/cassandra/1.1/docs/configuration/node_configuration.html#initial-token

keyspace creation ::
https://docs.datastax.com/en/cql/3.1/cql/cql_reference/create_keyspace_r.html
http://datascale.io/cassandra-partitioning-and-clustering-keys-explained/

multi-datacenter ::
https://docs.datastax.com/en/cassandra/2.1/cassandra/initialize/initializeMultipleDS.html

https://www.packtpub.com/books/content/apache-cassandra-working-multiple-datacenter-environments

commands for create keyspace in multi-datacenter ::
https://docs.datastax.com/en/cql/3.1/cql/cql_reference/create_keyspace_r.html

create keyspace for multi-data center ::
https://books.google.co.in/books?id=aIro2eZEz4YC&pg=PT26&lpg=PT26&dq=Random+Partitioner+configuration+in+cassandra&source=bl&ots=qWhK0lXgOZ&sig=BZf3o_gM-laTIO0ayfd2z5Dsnvg&hl=en&sa=X&ved=0ahUKEwjkhMOO5sPQAhXDqY8KHRUSCh0Q6AEITzAH#v=onepage&q&f=false

partitioning ::
https://10kloc.wordpress.com/2012/12/27/cassandra-chapter-4-data-partitioning/

https://support.datastax.com/hc/en-us/articles/205199715-Node-startup-fails-with-error-ConfigurationException-Cannot-change-the-number-of-tokens-

http://distributeddatastore.blogspot.in/2015/07/cassandra-data-partitioning-using.html

http://docs.datastax.com/en/archived/cassandra/1.1/docs/cluster_architecture/partitioning.html#data-distribution-in-the-ring

token generation ::
http://www.geroba.com/cassandra/cassandra-token-calculator/

http://docs.datastax.com/en/cassandra/2.0/cassandra/configuration/configGenTokens_c.html

http://docs.datastax.com/en/archived/cassandra/1.1/docs/initialize/token_generation.html#token-gen-cassandra

partion key and clustering key ::
https://jagadeeshs.wordpress.com/2015/07/15/cassandra/

https://dzone.com/articles/cassandra-data-modeling-primary-clustering-partiti

error solution for already existing data if initial_token changed ::
http://stackoverflow.com/questions/29323709/unable-to-start-cassandra-node-already-exists

detailed infoemation in partitioning ::
https://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsStatus.html

partitioning testing ::
http://stackoverflow.com/questions/30514237/what-node-does-cassandra-store-data-on

visual testing of nodes ::
http://psanford.github.io/cassandra-visual-ring/

****