**Installing Cassandra on Red Hat Enterprise Linux**

Before we start with installation, let's discuss few concepts first which will help you to understand installation process.

**Bootstrapping -**Bootstrapping is the process in which a newly-joining node gets the required data from the neighbors in the ring, so it can join the ring with the required data. Typically, a bootstrapping node joins the ring without any state or token and understands the ring structure after starting the gossip with the seed nodes; the second step is to choose a token to bootstrap. During the bootstrap, the bootstrapping node will receive writes for the range that it will be responsible for after the bootstrap is completed. This additional write is done to ensure that the node doesn’t miss any new data during the bootstrap from the point when we requested the streaming to the point at which the node comes online.

**Seed Nodes -**The seed node designation has no purpose other than bootstrapping the gossip process for new nodes joining the cluster. Seed nodes are not a single point of failure, nor do they have any other special purpose in cluster operations beyond the bootstrapping of nodes.

In cluster formation, nodes see each other and “join”. They do not join just any node which respects the protocol, however. This would be risky: old partitioned replicas, different clusters, even malicious nodes, so on. So a cluster is defined by some initial nodes which are available at clear addresses and they become a reference for that cluster for any new nodes to join in trustable way. The seed nodes can go away after some time, the cluster will keep on.

**Prerequisites**

You will need to set up static IP Address on all 4 nodes and make sure all 4 nodes are reachable to each other via hostname/IP.

Since we are working in a lab environment, following credentials will be used through out this guide and we will set up Cassandra nodes in /etc/hosts file as below. You can update /etc/hosts file by executing the following command on each node.  
vi /etc/hosts  
  
192.168.10.70 cassdb01 #SEED-node  
192.168.10.71 cassdb02 #Worker-node1  
192.168.10.72 cassdb03 #Worker-node2  
192.168.10.73 cassdb04 #Worker-node3  
  
Save and exit

**Open Firewall Ports**  
You need to allow access on 7000 and 9160 ports of Cassandra if you are using iptables with the following command on each node.  
  
iptables -A INPUT -p tcp –dport 7000 -j ACCEPT  
iptables -A INPUT -p tcp –dport 9160 -j ACCEPT  
  
**Create Cassandra user with sudo permissions.**  
You can download and use following script to create a user on server with sudo permissions.  
  
The following steps will be performed on**cassdb01** node.  
  
wget https://raw.githubusercontent.com/zubayr/create\_user\_script/master/create\_user\_script.sh  
chmod 777 create\_user\_script.sh  
sh create\_user\_script.sh -s cassandra  
  
When you are done with above commands, make sure cassandra user/group is created on server with the following command.  
  
cat /etc/passwd | grep cassandra  
  
Output  
cassandra:x:501:501::/home/cassandra:/bin/bash  
  
cat /etc/group | grep cassandra  
  
Output  
cassandra:x:501:

**Installing Java**  
We need to install oracle java (jdk or jre) version 7 or greater and defined *JAVA\_HOME*accordingly. You can install java with rpm based installer or using tar file.  
  
Cassandra 3.0 and later require Java 8u40 or later. I'll be installing rpm package.  
  
rpm -Uvh jdk-8u111-linux-x64.rpm  
  
**Note: If you have openjdk installed on your system then please remove it before installing oracle java.**  
  
Verify that JAVA\_HOME is set correctly and you are getting an output similar to below and an output for java -version command  
  
cat .bash\_profile | grep JAVA\_HOME  
  
Output  
JAVA\_HOME=/usr/java/jdk1.8.0\_111  
PATH=$PATH:$HOME/bin/:$JAVA\_HOME/bin:$CASSANDRA\_HOME/bin  
export PATH JAVA\_HOME CASSANDRA\_HOME  
  
java -version  
  
Output  
java version “1.8.0\_111”  
Java(TM) SE Runtime Environment (build 1.8.0\_111-b14)  
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)  
  
**Install and Configure Cassandra**  
Latest Cassandra version can be found from cassandra [Home Page](http://cassandra.apache.org/download/). Download and extract apache-cassandra tar.gz file in a directory of your choice. I used /opt as destination directory.  
  
tar -zxvf apache-cassandra-3.9-bin.tar.gz  
ln -s /opt/apache-cassandra-3.9 /opt/apache-cassandra  
chown cassandra:cassandra -R /opt/apache-cassandra  
chown cassandra:cassandra -R /opt/apache-cassandra-3.9  
  
Now, create necessary directories (for cassandra to store data)  and assign permissions on those directories.  
  
mkdir /var/lib/cassandra/data  
mkdir /var/log/cassandra  
mkdir /var/lib/cassandra/commitlog  
chown -R cassandra:cassandra /var/lib/cassandra/data  
chown -R cassandra:cassandra /var/log/cassandra/  
chown -R cassandra:cassandra /var/lib/cassandra/commitlog  
  
Start the Cassandra service by executing the following command  
  
$CASSANDRA\_HOME/bin/cassandra -f -R  
  
You will see below messages on command prompt which shows that cassandra have been started without any issues.  
  
Output  
INFO 11:31:15 Starting listening for CQL clients on localhost/127.0.0.1:9042 (unencrypted)…  
INFO 11:31:15 Not starting RPC server as requested. Use JMX (StorageService->startRPCServer()) or nodetool (enablethrift) to start it  
INFO 11:31:24 Scheduling approximate time-check task with a precision of 10 milliseconds  
INFO 11:31:25 Created default superuser role ‘cassandra’  
  
If you want to start cassandra as a service, you can use this script from [github](https://gist.github.com/sgomezvillamor/5458309" \t "_blank). Change value of following variable as per your environment.  
  
CASS\_HOME=/opt/apache-cassandra  
CASS\_BIN=$CASS\_HOME/bin/cassandra  
CASS\_LOG=/var/log/cassandra/system.log  
CASS\_USER="root"  
CASS\_PID=/var/run/cassandra.pid  
  
Save the file in /etc/init.d directory.  
  
Now execute the following commands to add cassandra as a service.  
  
chmod +x /etc/init.d/cassandra  
chkconfig –add cassandra  
chkconfig cassandra on  
  
Start cassandra service and verify its status by checking the system.log file  
  
service cassandra status  
  
Output  
Cassandra is running.  
  
System.log file contains the following info on my system and it means all well.  
INFO  12:45:50 Node localhost/127.0.0.1 state jump to NORMAL

You can also verify using Nodetool and its output says node as UP and Normal

$CASSANDRA\_HOME/bin/nodetool status  
  
Output  
Datacenter: datacenter1  
=======================  
Status=Up/Down  
|/ State=Normal/Leaving/Joining/Moving  
— Address Load Tokens Owns (effective) Host ID Rack  
  
UN 127.0.0.1 168.62 KiB 256 100.0% 14ba62c6-59e4-404b-a6a6-30c9503ef3a4 rack1

**Adding Node To Cassandra Cluster**

Before we start installing apache cassandra on remaining nodes, We need to perform following configuration changes in cassandra.yaml file.

Navigate to cassandra.yaml file which is located under ***cassandra\_install\_dir/conf*** folder. Open the file in editor of your choice and look for following options:

**Listen\_address**: Address where gossip will be listening to. This address can’t be localhost or 0.0.0.0, because the rest of nodes will try to connect to this address.  
**RPC\_address**: This is the address where thrift will be listening. We must put a existing IP address (it may be localhost, if we want to), or 0.0.0.0 if we want to listen through all of them. This is the address to which client applications interact with cassandra DB.  
**Seeds**: Seed nodes are the nodes which will provide cluster info to the new nodes which are bootstrapped and are ready to join the cluster. Seed nodes become a reference for any new nodes to join cluster in trustable way.

The above settings need to be configured in ***cassandra.yaml*** file on each node which we want to put into the cluster.  
  
The /etc/cassandra/conf/cassandra.yaml will look similar to like below:

cluster\_name: '[Cluster Name]'  
listen\_address: [192.168.10.70]  
rpc\_address: [192.168.10.70]  
num\_tokens: 256  
seed\_provider:  
  - class\_name: org.apache.cassandra.locator.SimpleSeedProvider  
parameters:  
  - seeds: "[192.168.10.70],[192.168.10.71],[192.168.10.72],[192.168.10.73]"  
endpoint\_snitch: GossipingPropertyFileSnitch  
auto\_bootstrap: false  
 **Note: You must install the same version of Cassandra on the remaining nodes in the cluster.**

**Adding new nodes in Cassandra cluster**  
Install Cassandra on the new nodes, but do not start Cassandra service.  
Set the following properties in the ***cassandra.yaml***file and, depending on the snitch, the cassandra-topology.properties or cassandra-rackdc.properties configuration files:

**auto\_bootstrap** – This property is not listed in the default cassandra.yaml configuration file, but it might have been added and set to false by other operations. If it is not defined in cassandra.yaml, Cassandra uses true as a default value. For this operation, search for this property in the cassandra.yaml file. If it is present, set it to true or delete it..  
**cluster\_name** – The name of the cluster the new node is joining. Ensure that cluster name is same for all nodes which will be part of cluster.  
**listen\_address** – Can usually be left blank. Otherwise, use IP address or hostname that other Cassandra nodes use to connect to the new node.  
**endpoint\_snitch** – The snitch Cassandra uses for locating nodes and routing requests. In my lab I am using simple snitch which is present as default in cassandra.yaml file and so I did not change or edit this.  
**num\_tokens** – The number of vnodes to assign to the node. If the hardware capabilities vary among the nodes in your cluster, you can assign a proportional number of vnodes to the larger machines.  
**seeds** – Determines which nodes the new node contacts to learn about the cluster and establish the gossip process. Make sure that the -seeds list includes the address of at least one node in the existing cluster.

Installation and configuration changes steps will be remain same as we have already performed in above on **cassdb01**. Once you are done installing cassandra and making the configuration changes as mentioned above on your second node, you can start cassandra service on second node and look into nodetool status on cassandra node 1 and you will observe the new node joining the cluster.

**Nodetool Status Output**  
You can see in below output that our second node has been added to cluster and its up and running.

Every 2.0s: /opt/apache-cassandra/bin/nodetool status Thru 20 23:31:44 2016  
Datacenter: datacenter1  
=======================  
Status=Up/Down  
|/ State=Normal/Leaving/Joining/Moving  
— Address               Load           Tokens Owns (effective) Host ID Rack  
UN 192.168.10.70 214.99 KiB  256 100.0%    14ba62c6-59e4-404b-a6a6-30c9503ef3a4   rack1  
UN 192.168.10.71 103.47 KiB  256 100.0%    3b19bc83-f483-4a60-82e4-109c90c49a14   rack1  
  
You need to repeat the same steps for each node which you want to add in Cassandra cluster.

**How Nodetool Works?**  
We will explain nodetool and see how we can manage a cassandra cluster using nodetool utility.  
  
The nodetool utility is a command line interface for managing a cluster. It provides a simple command line interface to expose operations and attributes available with cassandra. There are hundreds of options available with nodetool utility but we will cover only those which are being used more often.  
  
**Nodetool version**: This provides the version of Cassandra running on the specified node.

[root@cassdb01 ~]# nodetool version  
ReleaseVersion: 3.9

**Nodetool status**: This is one of the most common command which you will be using in a cassandra cluster. It provide information about the cluster, such as the state, load, and IDs. It will aslo tell you the name of datacenter where your nodes are lying and what is their state.  
  
State ‘UN’ referes to up and normal. When a new node is added to cluster, you might see the state of node as ‘UJ’ which means node is up and now in process of joining the cluster.  
  
Nodetool status will give you IP address of all of your nodes and also how much percentage load each node is owning. It is not neccsary that each node will own exactly the same percentage of load. For e.g in a 4 node cluster, it is not neccessary that each node owns exactly 25% of total load on cluster. One node might be owning 30% and other may be at 22% or so. But there should not be much difference in % of load being owned by each node.

[root@cassdb04 ~]# nodetool status  
  
Datacenter: datacenter1  
=======================  
Status=Up/Down  
|/ State=Normal/Leaving/Joining/Moving  
— Address Load Tokens Owns (effective) Host ID Rack  
UN 192.168.10.72 175.92 KiB 256 48.8% 32da4275-0b20-4805-ab3e-2067f3b2b32b rack1  
UN 192.168.10.73 124.63 KiB 256 50.0% c04ac5dd-02db-420c-9933-181b99848c4f rack1  
UN 192.168.10.70 298.79 KiB 256 50.8% 14ba62c6-59e4-404b-a6a6-30c9503ef3a4 rack1  
UN 192.168.10.71 240.57 KiB 256 50.4% 3b19bc83-f483-4a60-82e4-109c90c49a14 rack1

**Nodetool info**: This returns information about specific node. In output of above command you can see gossip state (whether its active or not), load on node,  rack and datacenter where node is placed.  
  
[root@cassdb04 ~]# nodetool info  
ID : c04ac5dd-02db-420c-9933-181b99848c4f  
Gossip active : true  
Thrift active : false  
Native Transport active: true  
Load : 124.63 KiB  
Generation No : 1484323575  
Uptime (seconds) : 1285  
Heap Memory (MB) : 65.46 / 1976.00  
Off Heap Memory (MB) : 0.00  
Data Center : datacenter1  
Rack : rack1  
Exceptions : 0  
Key Cache : entries 11, size 888 bytes, capacity 98 MiB, 323 hits, 370 requests, 0.873 recent hit rate, 14400 save period in seconds  
Row Cache : entries 0, size 0 bytes, capacity 0 bytes, 0 hits, 0 requests, NaN recent hit rate, 0 save period in seconds  
Counter Cache : entries 0, size 0 bytes, capacity 49 MiB, 0 hits, 0 requests, NaN recent hit rate, 7200 save period in seconds  
Chunk Cache : entries 16, size 1 MiB, capacity 462 MiB, 60 misses, 533 requests, 0.887 recent hit rate, 70.149 microseconds miss latency  
Percent Repaired : 100.0%  
Token : (invoke with -T/–tokens to see all 256 tokens)  
  
Note: To query about information of remote node, you can use -h and -p switch with nodetool info coomand. -h needs ip/fqdn of remote node and -p is the jmx port.  
  
[root@cassdb04 ~]# nodetool -h 192.168.10.70 -p 7199 info  
ID : 14ba62c6-59e4-404b-a6a6-30c9503ef3a4  
Gossip active : true  
Thrift active : false  
Native Transport active: true  
Load : 198.57 KiB  
Generation No : 1484589468  
Uptime (seconds) : 165  
Heap Memory (MB) : 91.97 / 1986.00  
Off Heap Memory (MB) : 0.00  
Data Center : datacenter1  
Rack : rack1  
Exceptions : 0  
Key Cache : entries 17, size 1.37 KiB, capacity 99 MiB, 71 hits, 102 requests, 0.696 recent hit rate, 14400 save period in seconds  
Row Cache : entries 0, size 0 bytes, capacity 0 bytes, 0 hits, 0 requests, NaN recent hit rate, 0 save period in seconds  
Counter Cache : entries 0, size 0 bytes, capacity 49 MiB, 0 hits, 0 requests, NaN recent hit rate, 7200 save period in seconds  
Chunk Cache : entries 12, size 768 KiB, capacity 464 MiB, 78 misses, 230 requests, 0.661 recent hit rate, 412.649 microseconds miss latency  
Percent Repaired : 100.0%  
Token : (invoke with -T/–tokens to see all 256 tokens)  
  
Nodetool describecluster: This command will give you name of the cassandra cluster, default partitioner which is used in cluster, type of snitch being used etc.

[root@cassdb01 ~]# nodetool describecluster

Cluster Information:

Name: Test Cluster

Snitch: org.apache.cassandra.locator.DynamicEndpointSnitch

Partitioner: org.apache.cassandra.dht.Murmur3Partitioner

Schema versions:

86afa796-d883-3932-aa73-6b017cef0d19: [192.168.10.72, 192.168.10.73, 192.168.10.70, 192.168.10.71]

**Nodetool ring**: This command will tell you which node is responsible for handling which range of tokens. If you are using virtual node concept, each node will be responsible for 256 token ranges. This command will give you a very lengthy output as it will display each and every token associated with each node.  
  
[root@cassdb04 ~]# nodetool ring  
  
Datacenter: datacenter1  
==========  
Address Rack Status State Load Owns Token  
9209474870556602003  
192.168.10.70 rack1 Up Normal 240.98 KiB 50.81% -9209386221367757374  
192.168.10.73 rack1 Up Normal 124.63 KiB 49.99% -9194836959115518616  
192.168.10.73 rack1 Up Normal 124.63 KiB 49.99% -9189566362031437022  
192.168.10.71 rack1 Up Normal 240.57 KiB 50.40% -9173836129733051192  
192.168.10.71 rack1 Up Normal 240.57 KiB 50.40% -9164925147537642235  
192.168.10.71 rack1 Up Normal 240.57 KiB 50.40% -9140745004897827128  
192.168.10.72 rack1 Up Normal 175.92 KiB 48.80% -9139635271358393037  
192.168.10.73 rack1 Up Normal 124.63 KiB 49.99% -9119385776093381962  
192.168.10.73 rack1 Up Normal 124.63 KiB 49.99% -9109674978522278948  
192.168.10.72 rack1 Up Normal 175.92 KiB 48.80% -9091325795617772970  
192.168.10.71 rack1 Up Normal 240.57 KiB 50.40% -9063930024148859956  
192.168.10.71 rack1 Up Normal 240.57 KiB 50.40% -9038394199082806631  
192.168.10.72 rack1 Up Normal 175.92 KiB 48.80% -9023437686068220058  
192.168.10.73 rack1 Up Normal 124.63 KiB 49.99% -9021385173053652727  
192.168.10.71 rack1 Up Normal 240.57 KiB 50.40% -9008429834541495946  
192.168.10.70 rack1 Up Normal 240.98 KiB 50.81% -9003901886367509605  
192.168.10.73 rack1 Up Normal 124.63 KiB 49.99% -8981251185746444704  
192.168.10.72 rack1 Up Normal 175.92 KiB 48.80% -8976243976974462778  
192.168.10.72 rack1 Up Normal 175.92 KiB 48.80% -8914749982949440380  
192.168.10.71 rack1 Up Normal 240.57 KiB 50.40% -8896728810258422258  
192.168.10.72 rack1 Up Normal 175.92 KiB 48.80% -8889132896797497885  
192.168.10.73 rack1 Up Normal 124.63 KiB 49.99% -8883470066211443416  
192.168.10.72 rack1 Up Normal 175.92 KiB 48.80% -8872886845775707512  
192.168.10.72 rack1 Up Normal 175.92 KiB 48.80% -8872853960586482247  
192.168.10.72 rack1 Up Normal 175.92 KiB 48.80% -8842804282688091715  
192.168.10.71 rack1 Up Normal 240.57 KiB 50.40% -8836328750414937464  
192.168.10.70 rack1 Up Normal 240.98 KiB 50.81% -8818194298147545683

**Nodetool cleanup**: Nodetool cleanup is used to remove that data from a node for which it is not responsible for.  
  
when a node auto bootstraps, it does not remove the data from the node that had previously been responsible for the data. This is so that,if  the new node were to go down shortly after coming online, the data would still exist.  
  
The command to do data cleanup is as below.  
  
[root@cassdb01 ~]# nodetool cleanup  
WARN 16:54:29 Small cdc volume detected at /opt/apache-cassandra/data/cdc\_raw; setting cdc\_total\_space\_in\_mb to 613. You can override this in cassandra.yaml  
  
WARN 16:54:29 Only 52.796GiB free across all data volumes. Consider adding more capacity to your cluster or removing obsolete snapshots  
  
Note: To remove data from a remote node, modify cleanup command as shown below  
  
[root@cassdb01 ~]# nodetool -h 192.168.10.72 cleanup  
  
To see what this command do, you can monitor on nodetool status and you will see load decreasing from that node where cleanup is ran.