Guide: Elastic Cassandra Setup

Daniel Moldovan
E-mail: d.moldovan@dsg.tuwien.ac.at

Contents

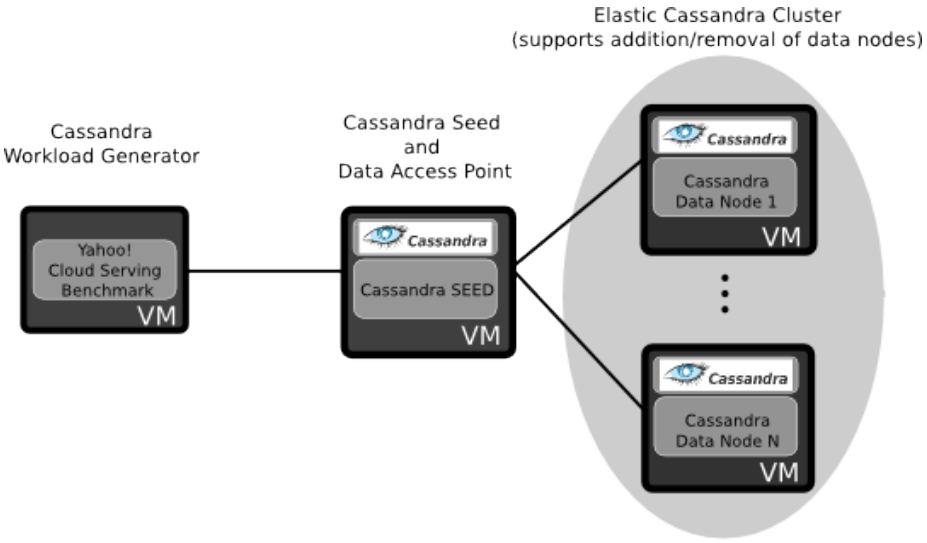

Figure 1: Elastic Cassandra topology overview

1 Introduction

This guide explains how to setup Cassandra[1] NoSQL distributed data store cluster using a set of scripts in order to behave elastically in a virtualized environment, by allowing dynamic addition and removal of Data Processing Nodes.

The general idea of our setup is to maintain a single Cassandra Node acting as Seed, representing the node to which any other Cassandra machines connect to and get a load token to join the Cassandra cluster, and support automated addition and removal of Data Nodes (Figure 1). Having this node alive throughout the lifetime of the Cassandra Cluster (from deployment up to when the cluster is decommissioned and all nodes destroyed), this node can also be considered as the data access point. While in this guide we assume there is only one Seed node, in practice one can setup using our scripts multiple Seed nodes, thus avoiding situations in which one data access machine can become a bottleneck. The supplied scripts can be used to configure Virtual Machine (VM) images as Cassandra Seeds, and other machines as Cassandra Data Nodes. When scaling out the system, the added Data Node machines automatically connect to the supplied Seed IP and join the Cassandra cluster, while when scaling in the cluster, we supply a mechanism for gracefully removing a Data Node from the cluster from the Seed machine.

One weak point could be that we use the default Cassandra token allocation mechanism, an advantage being that we obtain a shorter scaling out time, but possibly resulting in an unbalanced data distribution, since we do not rebalance the load across all data nodes.

[1] `http://cassandra.apache.org/`

To provide realistic workload for the data end, we provide scripts to be used in configuring a separate machine as Cassandra workload generator, running the Yahoo! Cloud Serving Benchmark (YCSB)[2], benchmark built to measure the performance (such as latency or throughput) of distributed data engines.

2 Setup Elastic Cassandra

We provide a series of scripts which configure a Cassandra setup to be elastic, supporting addition and removal of data processing nodes. The sequence of necessary operations for configuring Cassandra to be elastic is depicted in Figure 2:

1. A new virtual machine is created having Ubuntu as operating system. Actually, any Ubuntu-like system should work, as long as it maintains the same service definition mechanisms (init.d).

2. One of the archives we provide is copied and extracted in the new virtual machine. We provide archives containing only our scripts, containing a pre-packed Cassandra, and containing a pre-packed Cassandra and JREs (x64 and X86).

3. The user must untar the Cassandra and JRE archives, and change directory to "./ElasticCassandraScripts".

4. The environment variables contained in the Config file must me configured. The variables are as follows:

   - HOME: The directory in which the scripts are located.
   - CASSANDRA_HOME: Cassandra home directory.
   - CASSANDRA_BIN : Location of the Cassandra "bin" directory or binary files (cassandra, nodetool, etc).
   - CASSANDRA_CONFIG: Location of the Cassandra configuration files.
   - JAVA_HOME: Root directory containing a standalone Java deployment.
   - CASSANDRA_SEED_IP_SOURCE: A script/bash command that can retrieve the SEED IP at system boot time (for example using contextualization mechanisms) or directly a comma separated values list. This source is used to configure Cassandra Nodes in order to connect directly to the SEED IPs when booting for joining the Cassandra cluster.

5. If the pre-packed Cassandra is used, the user must run the "setupCassandra.sh" script, which uses the supplied JAVA_HOME to deploy a standalone Cassandra.

[2] `http://research.yahoo.com/Web_Information_Management/YCSB`

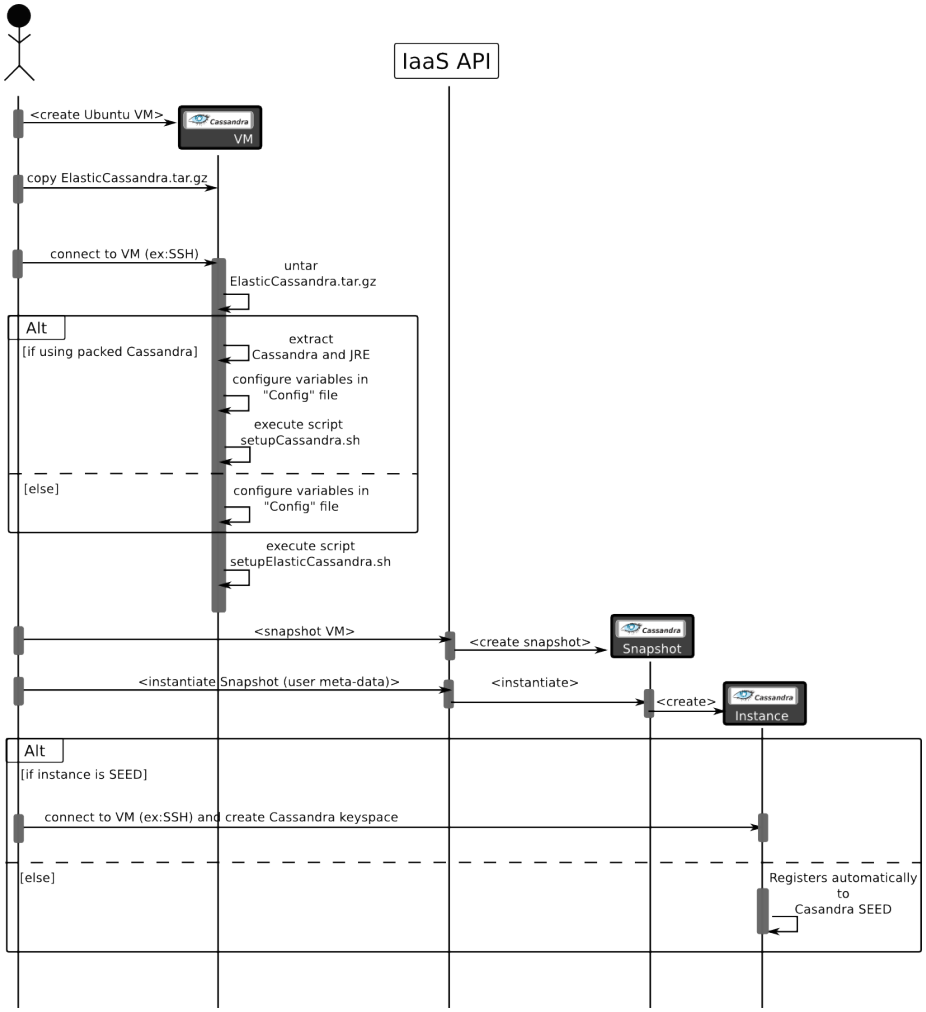

Figure 2: Operations sequence for configuring Elastic Cassandra

6. The second "setupElasticCassandra.sh" script configures depending on the node type (Seed or Node) the Cassandra installation, as to enable addition and removal of data nodes at run-time. The script prompts the user to select if he wants to configure the respective machine as "Seed (Main Node)" or "Data Node (Slave)".

7. The resulting Cassandra Seed or Data Node VM must be snapshotted BEFORE the Cassandra service is started, as to avoid the creation of load tokens. If a token is created, then no-mater how many VM's are instantiated using a snapshot, they will all have exact the same database load allocated to them, rendering the scale out process useless.

8. If the configured machines do not need to be snapshotted, please reboot their OS (sudo reboot) as to start Cassandra properly, as our scripts are embedded in /etc/init.d and /etc/rc/.local and are configured to be executed during a normal boot sequence.

9. For connecting to Cassandra using "cassandra-cli" , the host argument "-h" must point to the machine IP, not "localhost" (CASSANDRA_PATH/bin/cassandra-cli -h IP)

10. After creating a snapshot to serve as base for Cassandra Seed, the user can configure another machine as Data Node, by selecting, when prompted, the second option "setupElasticCassandra.sh" script option ("Data Node (Slave)"), having the CASSANDRA_SEED_IP_SOURCE defined as a script which can retrieve data from a VM contextualization mechanism or as a static IP.

11. To start Cassandra properly on the Seed configured nodes, just reboot the OS. Mainly, /etc/rc.local is executed also at boot time, and calls a script which creates a Cassandra keyspace to be used by YCSB, and also configures some other Cassandra parameters.

12. To test if the Cassandra cluster is working, from any Cassandra VM (preferably but not mandatory Cassandra Seed), execute :

    $CASSANDRA\_PATH/bin/nodetool \ -h \ localhost \ ring$

    , which shows the IPs associated to a cluster, and their load distribution. Or use the supplied "checkCassandraStatus.sh" script, which execute the call described above. The output should show all your IPs associated to your cluster, but it might take some while (usually 30 seconds) after a Node boots until it starts joining the cluster.

Note If Cassandra operations should be run on a machine configured as Cassandra Seed as to create a needed keyspace, the needed operations can be inserted in the "createCassandraKeyspaceScript.txt", which is run when Cassandra Seed machine boots, thus running any user-defined Cassandra operations.

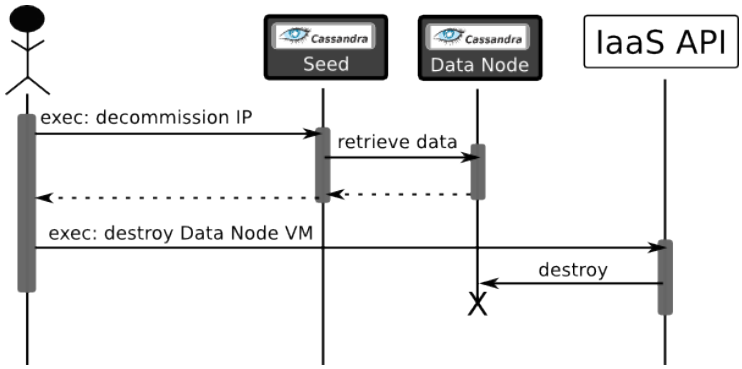

Figure 3: Operations sequence for Scaling In Elastic Cassandra Cluster

3 Scale In/Out Elastic Cassandra Cluster

After having a Cassandra cluster deployed, there are some simple steps to scale in and out the cluster.

Scaling In For scaling in the cluster (Figure 3), one must call in the Cassandra Seed machine (ex trough SSH) "decommission IP" (ex: "ssh HOST_IP decommission TARGET_IP"), which removes gracefully the node having the IP TARGET_IP. This step is important as Cassandra needs to move data to ensure consistency and redundancy. After the "decommission" command terminates, the virtual machine having the TARGET_IP can be destroyed using the used IaaS API.

Scaling Out For scaling out the cluster (Figure **??**), our scripts configure the Cassandra Node to automatically connect to a supplied Seed IP, in order to obtain a load allocation token. In order to be able to connect to a Seed IP, the CASSANDRA_SEED_IP_SOURCE entry in the "Config" file must be properly configured.

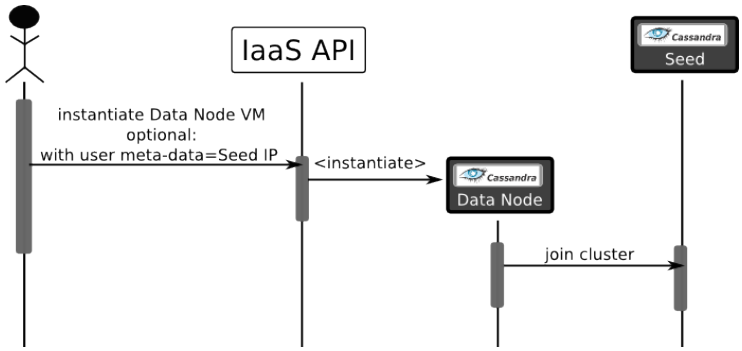

Figure 4: Operations sequence for Scaling Out Elastic Cassandra Cluster

Note