Cassandra Installation and Configuration Guide

Orchestration Server 8.1.4

12/29/2021

Table of Contents

Cassandra 2.2.5 / 3.9 Installation/ Configuration Guide

Apache Cassandra is an open source, eventually consistent, distributed database system used by Orchestration Server to store customer session information. It is designed to be highly scalable and decentralized (such that there is no single point of failure).

- This guide describes working with a Cassandra 2.2.5 installation (supported as of ORS Release 8.1.400.40 on 03/31/16), and a Cassandra 3.9 installation (supported as of ORS Release 8.1.400.51 on 01/12/17).
- For information on working with Cassandra versions prior to 2.2.5, see the *Cassandra 1.1.x Installation/ Configuration Guide* on the Orchestration Server documentation website.
- For more detailed and up-to-date information on the supported operating environments for ORS, see the Supported Operating Environment: Orchestration Server page.

> **Important**
>
> Cassandra connectivity is recommended for Orchestration Server deployments if session recovery is desired. This functionality will not be available without connectivity to at least one configured Cassandra node. Orchestration Server will execute without Cassandra.

> **Important**
>
> ORS supports only Apache's implementation of Cassandra. All references to Cassandra in ORS documentation refer to Apache Cassandra. The DataStax DSE edition of Cassandra is not supported as of now. ORS uses `gcassandralib`, which is a C++ Thrift based library to interface with Cassandra.

Overview

Apache Cassandra is an open source, eventually consistent, distributed database system used by Orchestration Server to store customer session information. It is designed to be highly scalable and decentralized (such that there is no single point of failure).

Data Model

The Cassandra data model is divided into four basic elements. They are:

- Columns
- Column Families
- Rows
- Keyspaces

Columns

A column is the smallest element of the Cassandra data structure. It is a tuple which consists of a name, value, and timestamp.

Column Families

A column family is to Cassandra what a table is to SQL (RDBMS). Column families store rows, and every row is referenced by a key.

Rows

A row groups related columns together. Every row is given a key and the key determines which Cassandra node the data will be stored to.

Keyspaces

A keyspace is roughly the equivalent of a schema or a database in SQL (RDBMS). Keyspaces contain collections of column families. The name of the keyspace is a first dimension of a Cassandra hash.

Example:

```
{
  "Genesys" : { // Keyspace
    "Services" : { // Column Family
      "ORS" : { // Row
        "Name" : "Orchestration", // Column
        "Version" : "8.1.2" }, // Column
      "GVP" : {
        "Name" : "Genesys Voice Platform",
        "Version" : "8.1.1" },
```

```
      "URS" : {
        "Name" : "Universal Routing Server",
        "Version" : "8.1.2" }
    }
  }
}
```

Prerequisites

Cassandra will run on both 32-bit and 64-bit operating systems, with 32-bit or 64-bit Java. The 64-bit versions are preferred.

- Download the latest stable 2.2.5 or 3.9 release of Cassandra from http://cassandra.apache.org/.
- Download Commons Daemon from http://www.apache.org/dist/commons/daemon/binaries/. This tool allows Cassandra to be run as a service/daemon.

> **Tip**
>
> If running Windows, make sure you download the Windows archive! The archive should contain 32-bit `prunsrv.exe`. The 64-bit version will be found in the appropriate subdirectory, either `amd64` or `ia64`. Also, currently, versions 1.0.14 and older work correctly, but version 1.0.15 fails to properly stop the service.

- Install JDK/JRE 8 - Preferably the most stable version available.
- Install Python version 2.7.10 – which is required by CQLSH.

Installation

Note: These instructions may vary for 64-bit versions. The following examples use 32-bit versions of Java and `prunsrv`.

Step 1: Downloading and Setting Environmental Variables

Extract the contents of the Cassandra (and Commons Daemon) archive(s) on each node. On Windows, use WinZip "Extract here..." or equivalent. On Linux, use `gunzip` or `tar -xvf`. If following the example below, the directories should be placed as such:

```
Windows: C:\Cassandra\apache-cassandra-2.2.x
Linux: /cassandra/apache-cassandra-2.2.x
```

The contents of the Commons Daemon archive are placed in the above installation directories in the `bin` folder, in a subdirectory named `daemon`, as below:

```
Windows: C:\Cassandra\apache-cassandra-2.2.x\bin\daemon
Linux: /cassandra/apache-cassandra-2.2.x/bin/daemon
```

Set the `JAVA_HOME` environment variable to the Java JRE/JDK root, for example:

```
set JAVA_HOME= C:\Program Files\Java\jdk1.8.0_73
or
export JAVA_HOME=/usr/java/jdk1.x.x_x
```

For Linux-like installs, edit `JAVA_HOME` in `%CASSANDRA_HOME%\bin\cassandra-in.sh`.

Step 2: Edit configuration files

The Cassandra distribution comes with a number of configuration files that should be edited (located in `%CASSANDRA_HOME%\conf` directory).

Step 2.0

The Cassandra distribution comes with a number of configuration files that should be edited (located in `%CASSANDRA_HOME%\conf` directory).

Step 2.1: Edit cassandra.yaml

The included `cassandra.yaml` contains default configurations for the Cassandra cluster.

When Cassandra versions 2.2.5 virtual nodes have been implemented, the initial_token should be left as is, with the exception of nodes that are being migrated from older 1.x.x versions. If this is the case, refer to the documentation specified in the yaml.

Ensure that the following options are pointing to the desired paths. Cassandra will create the directories on startup. Paths specified below are examples:

```
data_file_directories:
     - C:\Cassandra\apache-cassandra-2.2.x\data
 commitlog_directory: C:\Cassandra\apache-cassandra-2.2.x\commitlog
 saved_caches_directory: C:\Cassandra\apache-cassandra-2.2.x\saved_caches
```

Also in `cassandra.yaml`, configure the `cluster_name`, `key_cache_size_in_mb`, `counter_cache_size_in_mb`, `seeds`, `listen_address`, `start_rpc`, and `rpc_address`.

Follow the instructions in the yam regarding the settings for the following items, which all relate to memory allocation and number of processors that the installation platform has available.

```
concurrent_reads
concurrent_writes
concurrent_counter_writes
file_cache_size_in_mb
memtable_heap_space_in_mb
memtable_offheap_space_in_mb
commitlog_total_space_in_mb
```

- The `cluster_name` must be identical for all nodes within a Cassandra cluster.
- `key_cache_size_in_mb` should be set to `0` to disable key caching.
- `counter_cache_size_in_mb` should be set to `0` to disable counters.
- The seeds must be provided as a comma-delimited list of IP addresses to which new nodes will be able to contact for information about the Cassandra cluster. It is recommended that all nodes have the same list of seeds specified, and that all nodes be specified as seed nodes.
- `listen_address` is the IP address that other Cassandra nodes use to connect to this node.
- The `rpc_address` is the listen address for remote procedure calls (and clients, such as the `cassandra-cli`).

Make sure that `start_rpc` is set to `true`. If authentication is required, leave the `start_native_transport` at `true`. This port will be required to set the user name and passwords in Cassandra. If authentication is not required set `start_native_transport` to false.

> **Tip**
>
> If authentication is required – set the authenticator to `PasswordAuthenticator` and set the authorizer to `CassandraAuthorizer`. Refer to Step 5 for further information on setting the username and password.

The addresses are defaulted to localhost; it is recommended to set these to the IP address.

Verify that `storage_port` and `rpc_port` do not conflict with other configured services. The

`storage_port`, which defaults to `7000`, is the port used by Cassandra nodes for inter-node communication. The `rpc_port`, which defaults to `9160`, is used for remote procedure calls (such as `cassandra-cli`) and the Thrift service. This is the port to use when building clients for the Cassandra API.

If Cassandra is being deployed in a multiple data center configuration, the `endpoint_snitch` should be modified from the default of `SimpleSnitch` to `PropertyFileSnitch`, where the snitch then employs the `cassandra-topology.properties` to determine the nodes in the cluster. There are other snitch types available; please refer to the `cassandra.yaml` for the descriptions of these types. If a multiple data center deployment is chosen, the schema will require the correct replication information to be provided in the strategy option pairs that represent the cluster. An example for manually loading with the `PropertyFileSnitch` is described below. For Orchestration loading, the pairs in the strategy option should be the same in the persistence configuration.

Step 2.2: Edit cassandra-env and Cassandra Startup Script

If deployed on a Linux platform proceed to Step 2.2.3.

If deployed on a Windows platform, determine if Windows PowerShell is enabled and the current execution policies with the following command.

From a command prompt execute:

```
C:\>powershell Get-ExecutionPolicy -List
```

If PowerShell is available, the result should be similar to the following:

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

If PowerShell is not available, the above command will fail, proceed to Step 2.2.1.

> **Important**
>
> The default cassandra.bat attempts to invoke the PowerShell startup scripts, and if not restricted will do so.

At this point, since PowerShell is available, decide if Cassandra startup should be performed with the PowerShell or legacy methods. If the choice is legacy, then restrict the PowerShell execution policy at the level of your choice. The following command will restrict the policy at the CurrentUser scope:

```
C:\> powershell Set-ExecutionPolicy -ExecutionPolicy Restricted -Scope CurrentUser
```

After successful execution of the above, re-check the policy with the powershell `Get-ExecutionPolicy -List` command. If set to your choice proceed to Step 2.2.1.

If your choice is to allow PowerShell execution of Cassandra startup, proceed to Step 2.2.2.

Step 2.2.1: Windows without Windows PowerShell or PowerShell Restricted

To configure Cassandra's JVM (Java Virtual Machine) and the JMX (Java Management Extensions) interface, edit `bin/cassandra.bat`.

The JMX port is used for management connections (such as nodetool). If necessary, edit the following line and ensure that there are no port conflicts with existing services.

To enable remote JMX access see this topic on the Apache website.

Note that remote access via the JMX port is not recommended due to the possibility of unintended access to that port, which could disrupt Cassandra operation.

If the default JMX port, 7199, needs to be modified, change the following line to the desired port.

```
cassandra.jmx.local.port=7199>
```

If Cassandra will be installed as a service with a service name other than ‘cassandra’, modify the `SERVICE_JVM` as shown below to the desired name.

```
:doInstallOperation
set SERVICE_JVM="cassandra_gre_01"
rem location of Prunsrv
set PATH_PRUNSRV=%CASSANDRA_HOME%\bin\daemon\

# For x64 installations, (OS and JAVA) set
PATH_PRUNSRV=%CASSANDRA_HOME%\bin\daemon\amd64
set PR_LOGPATH=%CASSANDRA_HOME%\logs
```

Proceed to Step 2.3.

Step 2.2.2: Windows with Windows PowerShell Execution Policy Unrestricted

To configure Cassandra's JVM (Java Virtual Machine) options and the JMX (Java Management Extensions) interface, edit `conf/ cassandra-env.ps1`.

The JMX port is used for management connections (such as nodetool). If necessary, edit the following line and ensure that there are no port conflicts with existing services.

To enable remote JMX access see: https://wiki.apache.org/cassandra/JmxSecurity.

Note that remote access via the JMX port is not recommended due to the possibility of unintended access to that port, which could disrupt Cassandra operation.

If the default JMX port, 7199, needs to be modified, change the `JMX_PORT` to the desired port.

```
#   Specifies the default port over which Cassandra will be available for
#   JMX connections.
$JMX_PORT="7199"
```

The PowerShell function, `CalculateHeapSizes`, set these as follows:

```
# set max heap size based on the following
```

```
# max(min(1/2 ram, 1024MB), min(1/4 ram, 8GB))
# calculate 1/2 ram and cap to 1024MB
# calculate 1/4 ram and cap to 8192MB
# pick the max
```

If the heap sizes need to be defined rather than allowing the script to determine the values, modify the following to set the values:

```
    #$env:MAX_HEAP_SIZE="4096M"
    #$env:HEAP_NEWSIZE="800M"
    CalculateHeapSizes
```

If Cassandra will be installed as a service with a service name other than ‘cassandra’, edit `bin/cassandra.ps1`, and set the `SERVICE_JVM` to the desired name. Function HandleInstallation

```
{
                    $SERVICE_JVM = """CassandraForCluster1"""
                    $PATH_PRUNSRV = "$env:CASSANDRA_HOME\bin\daemon"
        …
```

Proceed to Step 2.3.

Step 2.2.3: Linux

To configure Cassandra's JVM (Java Virtual Machine) and the JMX (Java Management Extensions) interface, edit `conf/cassandra-env.sh`.

The JMX port is used for management connections (such as nodetool). If necessary, edit the following line and ensure that there are no port conflicts with existing services.

To enable remote JMX access see: https://wiki.apache.org/cassandra/JmxSecurity.

Note that remote access via the JMX port is not recommended due to the possibility of unintended access to that port, which could disrupt Cassandra operation.

If the default JMX port, 7199, needs to be modified, change the following line to the desired port.

```
#Specifies the default port over which Cassandra will be available for
# local JMX connections.
# JMX_PORT="7199"
```

Step 2.3: Edit logback.xml

Logging options can be found in `conf/logback.xml`. The default directory for logging is `%CASSANDRA_HOME%\logs`, with log file name `system.log.n`, where `n` is the wrap number. Cassandra versions 2.2.x and later, by default, enables debug level logging to separate file names. In order to disable `debug.log`, comment-out the `ASYNCDEBUGLOG` appender reference in the root level section.

Step 2B: Set up the Cassandra service (Windows)

Install the Cassandra service if desired.

To install: 'bin\cassandra.bat -INSTALL'

To uninstall: 'bin\cassandra.bat -UNINSTALL'

Once installed, you will be able to find and start up the Cassandra service from the Windows Services GUI. The name of the service depends on the value of `SERVICE_JVM`.

Step 3: Start up Cassandra

Linux:

Start up Cassandra by invoking `bin/cassandra -f`. It will start up in the foreground and will log to std-out. If you don't see any *error* or *fatal* log messages or Java stack traces, then chances are you've succeeded.

Press "Control-C" to stop Cassandra.

If you start up Cassandra without "-f" option, it will run in background, so you need to kill the process to stop.

Windows:

To start the service from Windows, there are two options:

1. Use the Windows Services GUI
2. From commandline, in the `daemon` dir (as set above):

- start: prunsrv.exe start <Cassandra Service Name>
- stop: prunsrv.exe stop <Cassandra Service Name>;

NOTE: There is currently a bug in prunsrv version 1.0.15.0 which prevents the service from being stopped. Use version 1.0.14.0 to prevent this, available from http://archive.apache.org/dist/commons/daemon/binaries/windows/

Step 4: Using nodetool

Once all Cassandra nodes have been started, we can check the status of the Cassandra cluster using nodetool.

On one of the Cassandra nodes, from `%CASSANDRA_HOME%`, run

```
/bin/nodetool -h <listen_address> -p <jmx_port> status
```

The output of this command should be similar to the example found in the Useful Tools section. There should be as many addresses in the list as the number of Cassandra nodes configured. If there are fewer nodes than expected, make sure that all nodes have unique initial tokens.

Step 5: Setting Username and Password for Authentication (Optional)

Once all Cassandra nodes have been started, set the user name and password that will be used for client connection login authorization. This is performed with the `cqlsh.bat` procedure for Windows-based deployments, and with `cqlsh` procedure for Linux-based deployments. These are located in the Cassandra installation bin directory.

Step 5.1: Login using CQLSH with Default Superuser

Start `cqlsh` with the Cassandra default superuser, user name `cassandra` and password `cassandra`. Note that the host and port that CQL connects to by default are 'localhost' and 9042. The host should be that which was specified in the `listen_address`, and the port should be what was defined in `native_transport_port`. Set the environment variable `CQLSH_HOST` to that specified in the listen_address. If the `native_transport_port` was changed from the default, the port can be set in the `CQLSH_PORT` environment variable. The following example for a Windows deployment had `CQLSH_HOST=dwswin7`. The port was not changed.

```
C:\Cassandra\apache-cassandra-2.2.5>.\bin\cqlsh.bat -u cassandra -p cassandra
Connected to Orchestration Windows PerfTest Cluster 2.1.12 at dwswin7:9042.
[cqlsh 5.0.1 | Cassandra 2.2.5 | CQL spec 3.3.1 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing.  Install to enable tab completion.
cassandra@cqlsh>
```

Step 5.2: Update the system_auth Replication Factor

Once connected, display the properties of the `system_auth` keyspace. This keyspace holds the authentication information that will be defined as described below. The replication factor of this keyspace should be increased, if the cluster has a small number of nodes, i.e., less than 10. In this case, set the replication factor to the number of nodes in the Cassandra cluster, or in each datacenter for multiple datacenter deployments. The replication factor will determine the number of instances that can fail and a client will still be able to login. The following describes the method to accomplish this for each datacenter with two Cassandra instances.

```
cassandra@cqlsh> ALTER KEYSPACE system_auth WITH
 replication = {'class': 'SimpleStrategy', 'replication_factor': '2'};
cassandra@cqlsh> DESCRIBE KEYSPACE system_auth;
CREATE KEYSPACE system_auth WITH replication = {'class': 'SimpleStrategy',
'replication_factor': '2'}
 AND durable_writes = true;
..
```

The DESCRIBE KEYSPACE shows the CQL that would be used to create the keyspace; it also displays the tables (column families) within the keyspace. For multiple datacenters, the `ALTER KEYSPACE` must be performed in each datacenter.

Step 5.3: Create the Desired User and Password

Now that the replication factor has been set on the `system_auth` keyspace, create a new user and password, as below. Note that this must be performed in each datacenter, also that the help requires

an underscore, while the command does not. For mixed or special characters, the single quotes are required:

```
cassandra@cqlsh> help CREATE_ROLE
cassandra@cqlsh> CREATE ROLE genesys WITH PASSWORD = 'g3n3sys!' AND LOGIN = true AND
SUPERUSER = true;
```

In this case a superuser genesys with password g3n3sys! is created. Once a new superuser is created, it is recommended that the default superuser cassandra be dropped. The following process describes how this can be accomplished.

First, perform a LIST USERS to make sure the new superuser is there:

```
cassandra@cqlsh> LIST USERS;
name      | super
-----------+-------
cassandra | True
genesys   | True
```

Now the original, default, superuser can be dropped, which must be done for each datacenter.

> **Tip**
>
> Make sure the created superuser and password are remembered; if not the Cassandra instances will have to be re-installed to allow the defaults to be used.

To drop the default user, `cqlsh` must be restarted with the new superuser:

```
cassandra@cqlsh> quit;
C:\Cassandra\apache-cassandra-2.2.5>.\bin\cqlsh.bat -u genesys -p g3n3sys!
```

Now drop the cassandra user:

```
genesys@cqlsh> DROP USER cassandra;
genesys@cqlsh> LIST USERS;
name   | super
-----------+-------
genesys| True
```

At this point, the new superuser can be specified in the Orchestration `persistence` section, options `username` and `password`, or another, non-superuser, may be created for that purpose, reserving the superuser for administrative purposes. An example follows.

```
genesys@cqlsh> CREATE ROLE orspersistence WITH PASSWORD = 'orsG3n3sys!' AND LOGIN = true;
genesys@cqlsh> LIST USERS;

cassandra@cqlsh> LIST USERS;

name           | super
----------------+-------
genesys        | True
orspersistence | False
```

Next, determine if the Orchestration KEYSPACE exists in Cassandra with the following:

```
genesys@cqlsh> DESCRIBE KEYSPACES;
```

The result will be something like:

```
system_traces system_auth system system_distributed
```

or

```
system_traces system_auth system “Orchestration” system_distributed
```

In the first case, proceed to step 5.3.1. In the second case, proceed to step 5.3.2.

Step 5.3.1 Orchestration KEYSPACE Does Not Yet Exist in Cassandra

The non superuser must be granted permission to create KEYSPACEs, the following will set the permission:

```
genesys@cqlsh>GRANT CREATE ON ALL KEYSPACES TO orspersistence;
```

Next, set the username and password in the Orchestration options to the non superuser name and password, then start Orchestration. This will create the Orchestration KEYSPACE. Check that the KEYSPACE exists with the DESCRIBE KEYSPACES command. If it exists, stop Orchestration, then proceed to step 5.3.2.

Step 5.3.2 Orchestration KEYSPACE Exists in Cassandra

The non-superuser must be granted access to the Orchestration keyspace. This can be done with the following CQL command. (note that if mixed case is present the double quotes are required)

> **Tip**
> if mixed case is present the double quotes are required.

```
genesys@cqlsh>GRANT ALL PERMISSIONS ON KEYSPACE “Orchestration” TO orspersistence;
```

If not already done, set the non-superuser name and password in the Orchestration options and start Orchestration.

Step 5.3.3 Troubleshooting Authorization

If the Orchestration connection to Cassandra cannot be authorized, the following entry will appear in the Orchestration log, and Orchestration will continue to run without persistence enabled:

```
Std 23002 ORS Cassandra schema version ORS8130000 No Cassandra hosts available
<Cassandra node [172.21.83.74] login failed - Transport exception>. Persistence is
unavailable.
```

Check the steps taken in 5.3.

If the Orchestration KEYSPACE does not exist and the non-superuser username has not been granted

CREATE permission on all KEYSPACES, the following entry will appear in the Orchestration log, Orchestration will continue to run without persistence enabled:

```
Std 23002 ORS Cassandra schema version ORS8130000 No Cassandra hosts available
<User orspersistence has no CREATE permission on <all keyspaces> or any of its parents>.
Persistence is unavailable.
```

Check the steps taken in 5.3.1.

If the Orchestration non-superuser username has not been granted permissions to the Orchestration keyspace, the following entry will appear in the Orchestration log, and Orchestration will be terminated:

```
Std 23001 ORS Cassandra schema version ORS8130000 Schema validation failed
<Insert into schema version failed>. Orchestration is terminating.
Std 23011 Orchestration Server::Stop() entered - stopping components
```

Check the steps taken in 5.3.2.

Configuring Cassandra

Cassandra can be configured prior to installation by editing the files located in `%CASSANDRA_HOME%\conf`. Within the `conf` directories are `cassandra.yaml`, `logback.xml`, and other files which may be edited to tune Cassandra's performance, to customize the Cassandra cluster settings or even change logging settings.

Basic Configuration

Prior to creating a Cassandra cluster, it is important to first modify a few core settings in `cassandra.yaml`:

cluster_name:

> Name of the Cassandra cluster. Must be identical for all nodes in cluster.

num_tokens
Leave the default value – unless a cluster is being migrated from a version 1.1.x cluster and the data needs to be maintained. Refer to the reference in the yaml for more information.

initial_token:
Leave the default value.

data_file_directories:
commitlog_directory:
saved_caches_directory:
Ensure that the above are all pointing to valid directories.

seeds: (default: "127.0.0.1")
Specifies a comma-delimited list of IP addresses. New nodes will contact the seed nodes to determine the ring topology and to obtain gossip information about other nodes in the cluster. Every node should have the same list of seeds.

start_native_transport: `false` (default is `true`)

start_rpc: `true` (default is `false`)

listen_address: (default: `localhost`)
The IP address that other Cassandra nodes will use to connect to this node. If left blank, uses the hostname configuration of the node.

rpc_address: (default: `localhost`)
The listen address for remote procedure calls. To listen on all interfaces, set to `0.0.0.0`. If left blank,

uses the hostname configuration of the node.

rpc_port: (default: 9160)
The port for remote procedure calls and the Thrift service.

NOTE: For Orchestration, the Thrift interface is required. Assure that the `start_native_transport` is set to `false`, and that the start_rpc is set to `true`.

storage_port: (default: 7000)
The port for inter-node communication.

endpoint_snitch: (default: `SimpleSnitch`)
This option determines how Cassandra views the cluster, `SimpleSnitch` for a single cluster and `PropertyFileSnitch`, or other snitch chosen in the yaml, for a multiple data center cluster.

Note that the `PropertyFileSnitch` requires the `cassandra-topology.properties` file to describe the multiple data center cluster, for example within that file the following will need to be provided:

```
# Cassandra Node IP=Data Center:Rack
135.225.58.81=DC1:RAC1

135.225.58.82=DC1:RAC2

135.225.58.83=DC2:RAC1

135.225.58.90=DC2:RAC2
```

Next, modify `%CASSANDRA_HOME%\bin\cassandra.bat` (if Windows), or `%CASSANDRA_HOME%/conf/cassandra-env.sh` (if Unix-based) to configure the JVM. It is important to verify that the JMX port does not conflict with other configured services:

```
-Dcassandra.jmx.local.port=7199 (in cassandra.bat)
JMX_PORT="7199" (in cassandra-env.sh)
```

Note that remote access via the JMX port is not recommended due to the possibility of unintended access to that port, which could disrupt Cassandra operation.

Storage Schema

Creation of the schema is performed by Orchestration on startup if not done so manually. Before starting Orchestration in this case, ensure that the Cassandra cluster is started first, then start one Orchestration instance. The schema will be created and propagated to all Cassandra instances. Manual schema creation can be done with the Cassandra CLI, note that the cassandra-cli is not available in Cassandra versions after 2.1.x, see Useful Tools section for more details. Seen below is a schema example for Orchestration on Cassandra 2.x (note that it conforms to the cassandra-cli syntax, again see Useful Tools section. Note that the replication factor is set to 1 in this sample and is the only allowed value for a single node deployment.

For a multiple node cassandra cluster this should be increased to increase availability. Refer to the following web site to determine the replication factor required for your deployment, noting that

Orchestration performs all operations at consistency level of ONE.

http://www.ecyrd.com/cassandracalculator/ http://www.ecyrd.com/cassandracalculator/

Note: The only exception from that rule is Column Family "SessionIDServerInfo" - For each SessionIDServerID entry ORS performs first attempt to write with consistency level QUORUM and all subsequent attempts with ONE, if first attempt failed.

For more discussion on this topic, please refer to

http://www.datastax.com/docs/1.1/dml/data_consistency http://www.datastax.com/docs/1.1/dml/data_consistency

for a discussion of consistency and the replication factor (RF).

Sample Orchestration Schema for Cassandra 2.2.5 and Orchestration version 8.1.4

This file contains an example of the Orchestration keyspace, which should be tailored to the deployed cassandra instance capabilities. This file should be copied to the cassandra install conf directory. The schema can be loaded using the `cassandra-cli` command line interface from the cassandra root install directory as follows:

```
        ./bin/cassandra-cli -host ip-address-of-cassandra-host --file conf/orchestration-schema.txt
```

where `ip-address-of-cassandra-host` is the IP form of the host - i,e., 199.166.88.61:9210

Note that the above assumes that the Thrift port is the default of 9160.

The `cassandra-cli` includes online help that explains the statements below. You can access the help without connecting to a running cassandra instance by starting the client and typing "help;" NOTE: Please assure that the replication_factor is set correctly. Use Cassandra version 2.2.5.

```
create keyspace Orchestration
    with strategy_options={replication_factor:1}
    and placement_strategy = 'org.apache.cassandra.locator.SimpleStrategy';

use Orchestration;

create column family Document
    with comparator = UTF8Type
    and column_type = Standard
    and memtable_throughput = 128
    and memtable_operations = 0.29
    and read_repair_chance = 1.0
    and max_compaction_threshold = 32
    and min_compaction_threshold = 4
    and gc_grace = 86400
    and comment = 'JSON form of the scxml document, keyed by md5 of document';

create column family Session
    with comparator = UTF8Type
    and column_type = Standard
    and memtable_throughput = 128
    and memtable_operations = 0.29
    and read_repair_chance = 1.0
    and max_compaction_threshold = 32
    and min_compaction_threshold = 4
    and gc_grace = 86400
```

```
    and comment = 'JSON form of the session, keyed by session GUUID';

create column family ScheduleByTimeInterval
    with comparator = UTF8Type
    and column_type = Standard
    and memtable_throughput = 128
    and memtable_operations = 0.29
    and read_repair_chance = 1.0
    and max_compaction_threshold = 32
    and min_compaction_threshold = 4
    and gc_grace = 86400
    and comment = 'Column names are the concatenation of scheduled ActionGUUID, action type,
and idealtime in msecs,
    column values are the action content. The keys are in form of time since the epoch in
msecs divided by some time increment,
    say 60000, for 1 minute intervals.';

create column family ScheduleBySessionID
    with comparator = UTF8Type
    and column_type = Standard
    and memtable_throughput = 128
    and memtable_operations = 0.29
    and read_repair_chance = 1.0
    and max_compaction_threshold = 32
    and min_compaction_threshold = 4
    and gc_grace = 86400
    and comment = 'Column names are the concatenation of scheduled ActionGUUID, action type,
and idealtime in msecs,
    column values are the idealtime in msecs, keyed by session id';

create column family SessionIDServerInfo
    with comparator = UTF8Type
    and column_type = Standard
    and memtable_throughput = 128
    and memtable_operations = 0.29
    and read_repair_chance = 1.0
    and max_compaction_threshold = 32
    and min_compaction_threshold = 4
    and gc_grace = 86400
    and comment = 'Session id and assigned node, keyed by session id';

create column family SessionIDServerInfoRIndex
    with comparator = UTF8Type
    and column_type = Standard
    and memtable_throughput = 128
    and memtable_operations = 0.29
    and read_repair_chance = 1.0
    and max_compaction_threshold = 32
    and min_compaction_threshold = 4
    and gc_grace = 86400
    and comment = 'Columns are session ids and the column values are also the session id,
keyed by the string
    form of the server node which owns the session.';

create column family RecoverSessionIDServerInfoRIndex
    with comparator = UTF8Type
    and column_type = Standard
    and memtable_throughput = 128
    and memtable_operations = 0.29
    and read_repair_chance = 1.0
    and max_compaction_threshold = 32
    and min_compaction_threshold = 4
    and gc_grace = 86400
```

```
    and comment = 'Columns are session ids and the column values are also the session id,
keyed by the string form
    of the server node which owns the session. Entries are only those sessions for which
recovery is enabled.';

create column family ORS8130000
    with comparator = UTF8Type
    and column_type = Standard
    and memtable_throughput = 128
    and memtable_operations = 0.29
    and read_repair_chance = 1.0
    and max_compaction_threshold = 32
    and min_compaction_threshold = 4
    and gc_grace = 86400
    and comment = 'Dummy column family to designate the Orchestration schema version.'; conf
directory.
```

In the above examples, one may notice that during the creation of keyspaces and column families, it is possible to configure various attributes. These attributes are described in the table below:

Keyspace Attributes

Option	Default	Description
name	N/A (Required)	Name for the keyspace.
placement_strategy	org.apache.casandra.locator.SimpleStrategy	Determines how replicas will be distributed among nodes in a Cassandra cluster. Allowed values: • org.apache.cassandra.locator.SimpleStrategy • org.apache.cassandra.locator.NetworkTopolog A simple strategy simply distributes replicas to the next N-1 nodes in the ring for a replication_factor of N. A network topology strategy requires the Cassandra cluster to be location-aware (able to determine location of rack/datacentre). In this case, the replication_factor is set on a per-datacentre basis.
strategy_options	N/A	Specifies configuration options for the replication strategy. For SimpleStrategy, one must specify `replication_factor:`*`number_of_replicas`*. For NetworkTopologyStrategy, one must specify *`datacentre_name:number_of_replicas`*.

Column Family Attributes

Option	Default	Description
comparator	BytesType	Defines data type to use when validating or sorting column names. The comparator cannot be changed once a column family has been created.
column_type	Standard	Determines whether column family is a regular column family or a super column family. Use `Super` for super column families.
read_repair_chance	0.1	Specifies probability that read repairs should be invoked on non-quorum reads. Value must be between 0 and 1. Lower values improve read throughput but increases chances of stale values when not using a strong consistency level.
min_compaction_threshold	4	Sets the minimum number of SSTables to trigger a minor compaction when `compaction_strategy=sizeTieredCompactionStrategy`. Raising this value causes minor compactions to start less frequently and be more I/O-intensive. Setting this value to 0 disables minor compactions.
gc_grace_seconds	864000 (10 days)	Specifies the time to wait before garbage collecting tombstones (items marked for deletion). In a single node cluster, it can be safely set to zero.
comment	N/A	A human readable comment describing the column family.
column_metadata	N/A	Defines the attributes of a column. For each column, values for `name` and `validation_class` must be specified. It is also possible to create a secondary index for a column by setting `index_type` and `index_name`.

Performance Tuning

Besides configuring keyspaces and column families, it is possible to further tweak the performance of Cassandra by editing `cassandra.yaml` (Node and Cluster Configuration) or by editing `cassandra-env.sh` (JVM Configuration).

Descriptions of tunable properties can be found in both `cassandra.yaml` and `cassandra-env.sh`. A summary of these properties can be seen in the tables below:

Performance Tuning Properties (cassandra.yaml)

Option	Default	Description
column_index_size_in_kb	64	The size at which column indexes are added to a row. Value should be kept small if only a select few columns are consistently read from each row as a higher value implies that more row data must be deserialized for each read (until index is added).
commitlog_sync	periodic	Allowed values are `periodic` or `batch`. In `periodic` mode, the value of `commitlog_sync_period_in_ms` determines how frequently the commitlog is synchronized to disk. Writes are acknowledged at every periodic sync. If set to batch, writes are not acknowledged until fsynced to disk.
commitlog_sync_period_in_ms	10000 (10 seconds)	Determines how often (in milliseconds) to sync commitlog to disk when `commitlog_sync` is set to `periodic`.
commitlog_total_space_in_mb	4096	When commitlog reaches specified size, Cassandra flushes memtables to disk for oldest `commitlog` segments. Reduces amount of data to replay on startup.
compaction_throughput_mb_per_sec	16	Throttles compaction to the given total throughput across entire system. Value should be proportional to rate of write throughput (16 to 32 times). Setting to 0 disables compaction throttling.
concurrent_compactors	1 (per CPU core)	Max number of concurrent compaction processes allowed on a node.

Option	Default	Description
concurrent_reads	16	Recommended setting is 16 * number_of_drives. This allows enough operations to queue such that the OS and drives can reorder them and minimize disk fetches.
concurrent_writes	32	Number of concurrent writes should be proportional to number of CPU cores in system. Recommended setting is (8 * number_of_cpu_cores).
memtable_flush_writers	1 per data directory	Number of memtable flush writer threads. Influences flush performance and can be increased if you have a large Java heap size and many data directories.
memtable_heap_space_in_mb	1/4 of heap	Total memory used for all column family memtables on a node.
stream_throughput_outbound_megabits_per_sec	400	Max outbound throughput on a node for streaming file transfers.

JVM Configuration Settings

Linux: conf/cassandra-env.sh
Windows: bin\cassandra.bat

Option	Default	Description
MAX_HEAP_SIZE	Half of available physical memory	Maximum heap size for the JVM. Same value is used for minimum heap size, allowing heap to be locked in memory. Should be set in conjunction with HEAP_NEWSIZE.
HEAP_NEWSIZE	100 MB per physical CPU core	Size of young generation. Larger value leads to longer GC pause times while smaller value will typically lead to more expensive GC. Set in conjunction with MAX_HEAP_SIZE.
com.sun.management.jmxremote.port	7199	Port on which Cassandra listens for JMX connections.

Option	Default	Description
com.sun.management.jmxremote.ssl	false	Enable/disable SSL for JMX.
com.sun.management.jmxremote.authenticate	false	Enable/disable remote authentication for JMX.

Useful Tools

The Cassandra archive includes several helpful tools for viewing and configuring Cassandra. Most of which can be found in the `%CASSANDRA_HOME%\bin` directory.

cassandra-cli

NOTE: This tool is available only for Cassandra versions up to, and including, version 2.1.x. To use this tool, install a separate instance of Cassandra version 2.1.x. There is no need to configure or start this instance, this will be used only for `cassandra-cli` invocation. The defaults in the yaml, etc, can be left unchanged. For more information on the Cassandra-CLI utility, refer to the Cassandra-CLI utility documentation.

nodetool

The nodetool utility provides a couple of helpful features. The most important of which is the ability to view the status of a Cassandra cluster, for example:

```
C:\Cassandra\apache-cassandra-2.2.5\bin>nodetool.bat -h dwswin7 status
Starting NodeTool
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load   Tokens Owns Host ID Rack
UN 172.21.82.83 267.79 KB 256 ?    c3af2e2e-ed72-4f96-8bce-f6f6f9c9eb98 rack1
UN 172.21.83.74 320.89 KB 256 ?    788429a3-c7f9-4263-aabd-21be37cc3e30 rack1
```

Refer to the following DataStax website for more information:

https://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsTOC.html

JConsole

Available in the `jdk bin` directory. For example: C:\Program Files\Java\jdk1.8.0_73\bin\jconsole.exe.

After opening, select the remote process button and enter the `ip:jmx`port of the cassandra node. The jmx console will display heap memory usage, live threads, classes loaded, and CPU usage. Check out the MBeans tab for info regarding the Cassandra exposed information and tools. Multiple nodes may be monitored with one console - just select connection, new connection, and enter the Cassandra `ip:jmx` port for the new node.

Resetting Cassandra to Initial State

In some cases, due to errors in SXML scripts being exercised, there may be sessions which are started and never terminated. Depending on interaction volume, this may cause the Cassandra data to exceed the planned limits of the deployment. In this case, after removing issues with the SCXML scripts, which cause unterminated sessions:

- Stop all Orchestration nodes.
- Then stop all Cassandra nodes.
- Remove all entries in the yaml-defined paths for the data, saved_caches, and `commitlog` directories.
- Once this is complete for all Cassandra nodes in cluster, restart the Cassandra nodes, and reload the schema. Or allow Orchestration to load the schema (Orchestration versions 8.1.3 or later).

Routine Node Repair

On a production cluster, it is important to routinely run the Nodetool Repair command. Use this command to repair inconsistencies between nodes within a cluster. Stale data is updated by pulling the latest version from other nodes. Unless Cassandra applications perform no deletes at all, it is recommended to run scheduled repairs on all nodes at least once every `gc_grace_seconds`. If this procedure is not followed, deletes may be "forgotten" in the cluster following garbage collection.

> **Important**
>
> Nodetool Repair is a resource-intensive task and should be scheduled for low-usage hours. Avoid running nodetool repair on more than one node at a time. Only repair is required, compact is not recommended.

The following is an example for Windows.

```
nodetool.bat -h dwswin7 repair Orchestration
```

If Not Run within GCGraceSeconds

If Nodetool Repair is not run within `GCGraceSeconds` (default is 10 days), then you run the risk of forgotten deletes. This may lead to inconsistencies in the data returned by different nodes. Running Nodetool Repair will not correct the issue entirely. There are two recommended methods of dealing with this scenario:

1. Treat the node with inconsistent data as "failed" and replace it.
2. To minimize forgotten deletes, increase `GCGraceSeconds` for all Column Families via the CLI, perform a full repair on all nodes, and then change `GCGraceSeconds` back again.

Troubleshooting

Auto-generated Keyspaces

If you encounter issues with automatically generated Keyspaces through ORS startup when the Keyspace name does not exist, perform the following:

- On one Cassandra instance execute the following with `cqlsh`. Log in:

```
…>DROP KEYSPACE “Orchestration”;
```

This will take several minutes to propagate to all Cassandra instances. During this time ORS will lose connection with the Cassandra cluster. Once the Keyspace has been dropped by Cassandra, the instances will accept client (ORS) connections. ORS will then reconnect and auto generate the schema.

- To verify that ORS has reconnected and set the schema, check with `cqlsh`:

```
…>use “Orchestration”;
```

(if ORS has reconnected the above command will be successful).

```
…>describe schema;
```

(this will be the ORS auto-generated schema)

- Quit `cqlsh`.
- Run the following query to alter the auto-generated schema:

```
cqlsh -f cassandra_ors_schema_alter.cql
```