Cassandra Installation and Configuration Guide

Configuring Cassandra

12/16/2025

Contents

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`.