Genesys Mobile Services Deployment Guide

Configure an External Cassandra

7/1/2021

Configure an External Cassandra

Modified in: 8.5.207.05, 8.5.206.04, 8.5.203.02

Version	Description
8.5.207+	• New deployment for GMS schemas in the Cassandra cluster. Create and deploy GMS schemas in the Cassandra cluster before starting GMS nodes. This change improves the management of the Cassandra authorization for a specific user; as a result, the user no longer needs to own super-user authorizations to create keyspaces. If, later, new tables might be created during an update or a migration, configure create-tables = `true` to allow GMS to create the missing tables in GMS keyspaces. • Genesys simplifies GMS deployments with Cassandra by deprecating old options and introducing new ones. See the cassandra Section section of the *Genesys Mobile Engagement Configuration Options* guide. • The minimum supported version of Cassandra is now 2.1. Refer to the Prerequisites section.
8.5.206+	• GMS no longer supports installing embedded Cassandra (in lab or production). • Change of Cassandra driver. The new driver used by GMS to connect to Cassandra supports setting multiple Cassandra instances. Other Cassandra instances are automatically detected. If one of them dies, the driver connects to all the detected Cassandra instances, including those that are not defined in the GMS options.
8.5.203+	GMS only supports deployments with an external Cassandra. An external Cassandra might be used in the following scenarios: • You already have a Cassandra/DataStax deployment. • You are securing your data in segregated networks, cages, and racks. • You want multiple redundancy features, such as distinct data centers, rack, and chassis awareness. • You are installing GMS on a production server or on a Windows host.

	• Starting with 8.5.206, GMS no longer supports installing embedded Cassandra (in lab or production). • Genesys does not recommend installing Cassandra on Windows.

Configuring GMS for an external Cassandra is a multi-step process to enable connection, TLS connection, authentication, and authorization. The steps include setting configuration options in Configuration Manager (or Genesys Administrator), changing configuration settings in the `cassandra.yaml` file, and executing Cassandra Query Language (CQL) commands.

Important

- All external Cassandra nodes must be of the same version.
- Cassandra is not required if you deploy Genesys Mobile Environment for Chat API V2, Email API V2, and Open Media API V2.
- When you are deploying Cassandra and GMS instances, ensure perfect time synchronization between hosts. NTP or another daemon should be installed on each host to ensure only a few milliseconds difference between hosts at maximum. Over time, data in a Cassandra replica can become inconsistent with other replicas due to the distributed nature of the database. In this scenario, you need to repair them as described in the Official Cassandra documentation.

Install External Cassandra Nodes

If you have not already installed Cassandra, then follow these guidelines.

Prerequisites

- GMS supports Cassandra 2.1 and higher (≤ 3.11):
  - The latest tested version is 3.11
  - For Cassandra 2.1, use CQL 3.1.
  - For Cassandra 3.0 and 3.1, use CQL 3.3.
- For the Network Topology, use one or several clusters.

Install Cassandra on All Nodes

Download Apache Cassandra. Use the right Unix user to install the application on all of your nodes (or host):

```
$  wget http://www-us.apache.org/dist/cassandra/3.11.2/apache-cassandra-3.11.2-bin.tar.gz
```

Extract

```
$ tar xf  apache-cassandra-3.11.2-bin.tar.gz
```

Configure Cassandra for Network Topology

1. Edit the `conf/cassandra.yaml` file for all of the nodes that you installed:
   - Modify the `cluster_name` value:

     ```
     cluster_name: '<the same name for all the cluster/datacenter>'
     ```

   - Modify the `seeds` value to specify a comma-separated list of host IP addresses.

     ```
      - seeds: "<cassandra_seed_host_ip_address_cluster_datacenter1>,<cassandra_seed_host_ip_address_clus
     ```

     For example DC1 and DC2 (see below):

     ```
      - seeds: "192.168.1.172,10.100.198.143"
     ```

   - Modify the `listen_address` value:

     ```
     listen_address: <externaly reachable cassandra host ip address>
     ```

   - Modify the `native_transport_port` value to specify the port matching the value of the GMS Configuration option native-port :

     ```
      native_transport_port: 9042 # default port

     native_transport_port_ssl: 9142 # default TLS port if you need native_transport_port for non-TLS con
     ```

     Note: If you enable TLS, use **native_transport_port** for TLS connections. If you need to keep **native_transport_port** for non-TLS connections, use **native_transport_port_ssl** to enable client encryption on the secured TLS port. See Client-to-node encryption in the *Official Apache Cassandra* documentation.

   - Modify the `rpc_address` value:

     ```
     rpc_address: <externaly reachable cassandra host ip address>
     ```

   - Modify the `endpoint_snitch` value:

     ```
     endpoint_snitch: PropertyFileSnitch
     ```

     > **Important**
     >
     > Depending on your topology, you may use other values such as `GossipingPropertyFileSnitch` for the snitch endpoint.

2. Edit the `conf/cassandra-topology.properties` file for all of the nodes:

```
# Cassandra Node IP=Data Center:Rack
# DC1
192.168.1.172=DC1:RAC1 # seed node DC1
192.168.1.215=DC1:RAC1
```

```
192.168.1.234=DC1:RAC1

 # DC2
10.100.198.143=DC2:RAC1        # seed node DC2
10.100.198.94=DC2:RAC1
10.100.198.136=DC2:RAC1
# default for unknown nodes
default=DC1:RAC1
```

Start all of the cassandra nodes. Now, your cluster of cassandra nodes is functional.

Synchronize Cassandra Hosts

To avoid synchronization issues, synchronize the clock of all GMS nodes. On Windows, use following the command:

```
net time \\<ComputerName> /set /yes
```

Where `\\<ComputerName>` specifies the name of a server you want to check or with which you want to synchronize.

Deploy GMS schemas in Cassandra

Starting in 8.5.207, it is mandatory to create and deploy GMS schemas in the Cassandra cluster before starting GMS nodes. This change improves the management of the Cassandra authorization for a specific user; as a result, the Cassandra user no longer needs to own super-user authorizations to create keyspaces.

Create Schemas for your First Deployment

In a new fresh deployment, you need to create GMS keyspaces and tables in **one** of your nodes (not in all of them). Before you run the CQL scripts that perform these operations, edit the following files:

- `<GMS Home>/scripts/cassandra_gsg_schema.cql`
- `<GMS Home>/scripts/cassandra_gsg_dd_schema.cql`

Replace the text placeholders `[ToBeChanged:<keyspace_name>]` with your keyspace names and set the replication factor to `NetworkTopologyStrategy`.

For example, if you edit the `cassandra_gsg_schema.cql` file, here are the first lines that you will see:

```
CREATE KEYSPACE [ToBeChanged:<keyspace_name>] WITH replication =
{'class': 'SimpleStrategy', 'replication_factor': '2'}
AND durable_writes = true;

CREATE TABLE [ToBeChanged:<keyspace_name>].gsg_principal_roles (
    key blob,
    column1 blob,
    value blob,
```

```
    PRIMARY KEY (key, column1)
)
```

If your GMS keyspace name is `gsg`, you should edit these lines as follows:

```
CREATE KEYSPACE gsg WITH replication =
{ 'class' : 'NetworkTopologyStrategy', 'DC1' : 2, 'DC2' : 2 }
AND durable_writes = true;

CREATE TABLE gsg.gsg_principal_roles (
    key blob,
    column1 blob,
    value blob,
    PRIMARY KEY (key, column1)
)
```

After you have edited the CQL scripts, launch them to create the GMS schemas in the Cassandra node:

```
$ <CASSANDRA_HOME>/bin/cqlsh <cassandra_host> <cassandra_port>  -f <GMS_HOME>/scripts/cassandra_gsg_schema.cql
$ <CASSANDRA_HOME>/bin/cqlsh <cassandra_host> <cassandra_port>  -f <GMS_HOME>/scripts/cassandra_gsg_dd_schema.cq
```

> **Important**
>
> Create these schemas in one node only.

Update Schema after Upgrading

Starting in 8.5.102, Cassandra schemas are compatible with GMS 8.5.105+ and do **not** require any upgrade. But if you upgrade from GMS versions older than 8.5.102, you will need to manually update the Cassandra schemas in **one** of your nodes (not all).

Starting in 8.5.207, configure create-tables = `true` to allow GMS to create the missing tables in GMS keyspaces.

Before running the CQL scripts that perform these operations, edit the following files:

- `<GMS Home>/scripts/update_cassandra_gsg_schema.cql`
- `<GMS Home>/scripts/update_cassandra_gsg_dd_schema.cql`

Replace the text placeholders `[ToBeChanged:<keyspace_name>]` with your keyspace names and set the replication factor to `NetworkTopologyStrategy`.

For example, if you edit the `update_cassandra_gsg_schema.cql` file, here are the first lines that you will see:

```
CREATE KEYSPACE IF NOT EXISTS [ToBeChanged:<keyspace_name>] WITH replication = {'class': 'SimpleStrategy', 'repl
AND durable_writes = true;

CREATE TABLE IF NOT EXISTS [ToBeChanged:<keyspace_name>].gsg_principal_roles (
    key blob,
    column1 blob,
    value blob,
    PRIMARY KEY (key, column1)
)
```

If your GMS keyspace name is `gsg`, you should edit these lines as follows:

```
CREATE KEYSPACE IF NOT EXISTS gsg WITH replication = {'class' : 'NetworkTopologyStrategy', 'DC1' : 2, 'DC2' : 2}
AND durable_writes = true;

CREATE TABLE IF NOT EXISTS gsg.gsg_principal_roles (
    key blob,
    column1 blob,
    value blob,
    PRIMARY KEY (key, column1)
)
```

After you edited the CQL scripts, launch them to update the schemas:

```
$ <CASSANDRA_HOME>/bin/cqlsh <cassandra_host> <cassandra_port> -f <GMS_HOME>/scripts/update_cassandra_gsg_schema
$ <CASSANDRA_HOME>/bin/cqlsh <cassandra_host> <cassandra_port> -f <GMS_HOME>/scripts/update_cassandra_gsg_dd_sch
```

> **Important**
>
> Update the schemas in one node only.

Configuration Options

The cassandra and cassandra-authentication-security sections list the configuration options applicable to an external Cassandra deployment. Changes take effect after restart.

Connection to an External Cassandra

The following steps are required to enable GMS to connect to an external Cassandra.

1. In Configuration Manager, locate and open your GMS Application object.
2. On the Options tab, **[cassandra]** section, configure the following options:

   nodes = <your Cassandra hosts or IP addresses for the local datacenter (comma-separated list)>

   native-port = <your Cassandra port: default is 9042>

   (Deprecated in 8.5.207) **strategy-class** = `NetworkTopologyStrategy`

   (Deprecated in 8.5.207) **strategy-option** = `DC1:2;DC2:2`

   keyspace-prefix = <if you need to select a prefix other than the default one (`gsg`) for your schemas>
3. Restart GMS.

> **Important**
>
> (Deprecated in 8.5.207) The **strategy-class** and **strategy-option** options must match the replication factor that you set when creating or updating your Cassandra schemas.

Configuring TLS

Before starting your Cassandra nodes, you must configure the TLS options.

Cassandra Side

Create Cassandra node certificate

You can either create one certificate per Cassandra node or create only one certificate for all of your Cassandra nodes.

Use **keytool**, which is part of the JDK toolset, to create your certificate:

Example

```
$ keytool -genkey -keyalg RSA -alias cassandranode -keystore keystore.node -storepass cassandra -keypass cassand
# will create keystore.node file
$ keytool -export -alias cassandranode -file cassandranode.cer -keystore keystore.node                # passwor
# will create cassandranode.cer file
$ keytool -import -v -trustcacerts -alias cassandranode -file cassandranode.cer -keystore truststore.node
# new password: cassandra
# will create create truststore.node file
```

Prepare keystore file for cassandra configuration file (cassandra.yaml)

Copy and secure the keystore file, **keystore.node**, in an appropriate path in your system

Example

```
# copy keystore.node file in <cassandra home path>/conf/cassandra.yaml
$ cp keystore.node .keystore
# for Cassandra server side (client_encryption config)
# restrict access to this file
$ chmod 600 .keystore
```

Set up Cassandra client encryption in cassandra.yaml file

```
# enable or disable client/server encryption.
client_encryption_options:
    enabled: true     ## to be changed
    # If enabled and optional is set to true encrypted and unencrypted connections are handled.
    optional: false
    keystore: conf/.keystore                ## to be changed
    keystore_password: cassandra     ## to be changed
    # require_client_auth: false
```

```
    # Set trustore and truststore_password if require_client_auth is true
    # truststore: conf/.truststore
    # truststore_password: cassandra
    # More advanced defaults below:
    # protocol: TLS
    # algorithm: SunX509
    # store_type: JKS
    # cipher_suites: [TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_DHE_RSA_WITH_AES_128_CBC_SHA
```

You can now start all of your Cassandra nodes and use CQLSH to provision them.

CQLSH TLS configuration

If you set up both an unsecured and a secured Cassandra port, you can use CQLSH on the unsecured Cassandra port as is. However, if you need to use CQLSH on a secured Cassandra port, you must export the cassandra nodes certificate to another format:

```
$ keytool -importkeystore -srckeystore keystore.node -destkeystore cassandranode.p12 -deststoretype PKCS12 -srcs
# will create cassandranode.p12 file
$ openssl pkcs12 -in cassandranode.p12 -nokeys -out cassandranode.cer.pem -passin pass:cassandra
# will create cassandranode.cer.pem file
$ openssl pkcs12 -in cassandranode.p12 -nodes -nocerts -out cassandranode.key.pem -passin pass:cassandra
# will create cassandranode.key.pem file
```

CQLSH TLS setup

Using a cqlshrc file

Create a cqlshrc file in **/home/*user home*/.cassandra/cqlshrc**:

```
[ssl]
certfile = <path to certificate>/cassandranode.cer.pem
validate = false ;; Optional, true by default.
```

Note: If **validate** is enabled, the host in the certificate is compared to the host of the machine to which it is connected, to verify that the certificate is trusted.

Using an environment variable

You must set the **SSL_CERTFILE** and **SSL_VALIDATE** environment variables:

```
$ export SSL_CERTFILE=<path to certificate>/cassandranode.cer.pem
$ export SSL_VALIDATE=true
```

CQLSH TLS connection

To start CQLSH, you must log in with user **cassandra** (default password: *cassandra*):

```
<apache-cassandra-3.11.2 home>$ bin/cqlsh --ssl -u cassandra -p cassandra <cassandra node IP address> <native po
```

GMS side

GMS TLS configuration

Import the Cassandra node certificate

You can import the **cassandranode.cer** file created above in one of two ways:

- Import the Cassandra node certificate into a specific GMS file. **Note:** If you use this method, you won't be able to use some features of the notification API that needs to read other certificates.
- Import the certificate into the Java JDK/JRE security file that resides in the JDK you use to start GMS: ***Java JDK home*/jre/lib/security/cacerts**

Using a specific GMS Certificate file

To import the Cassandra certificate to a specific Java client truststore file (for GMS), use **keytool**, which is part of the JDK toolset:

```
<GMS client side>$ keytool -import -v -trustcacerts -alias cassandranode -file cassandranode.cer -keystore clien
# password: cassandra
```

Modify launcher.xml

Add the following parameters to **launcher.xml** (note that the **debugtls** parameter, which is used for debugging, is optional):

```
<parameter name="cassandranodes_truststore" displayName="cassandratrustore" mandatory="true" hidden="true" readO
     <description><![CDATA[Certificates trustStore for Cassandra nodes]]></description>
     <valid-description><![CDATA[]]></valid-description>
     <effective-description/>
     <format type="string" default="-Djavax.net.ssl.trustStore=client.truststore" />
     <validation></validation>
</parameter>
<parameter name="cassandranodes_trustStorePassword" displayName="cassandratrustStorePassword" mandatory="true" h
     <description><![CDATA[Certificates trustStore password for Cassandra nodes]]></description>
     <valid-description><![CDATA[]]></valid-description>
     <effective-description/>
     <format type="string" default="-Djavax.net.ssl.trustStorePassword=cassandra" />
     <validation></validation>
</parameter>
<parameter name="debugtls" displayName="Debug TLS" mandatory="true" hidden="true" readOnly="true">
     <description><![CDATA[Add debug mode for SSL]]></description>
     <valid-description><![CDATA[]]></valid-description>
     <effective-description/>
     <format type="string" default="-Djavax.net.debug=ssl" />
     <validation></validation>
</parameter>
```

Using a global Certificate file

For Windows: Add **Certificates trustStore for Cassandra nodes** to the main JRE **lib/security/cacerts** file:

```
$ sudo keytool -importcert -file cassandranodes.cer -alias cassandranodes -keystore $JAVA_HOME/jre/lib/security/
```

You do not need to change **launcher.xml** when using a global Certificate file.

Set the TLS-related GMS options

Set the GMS native-port and secured application options to the values shown here:

```
[cassandra]
nodes = <your Cassandra hosts or IP addresses for the local datacenter (comma-separated list)>
native-port = 9142
secured = true
keyspace-prefix = <if you need to select a prefix other than the default one ('''gsg''') for your schemas>
```

For versions prior to GMS 8.5.207, you also need to configure the following options:

```
strategy-class = NetworkTopologyStrategy
strategy-option = DC1:2;DC2:2
```

Authentication on External Cassandra

> **Important**
> Supports Cassandra version 2.1.x and higher (≤ 3.11).

The following steps are prerequisites prior to enabling authentication.

Configure cassandra.yaml File

1. Stop the Cassandra nodes.
2. Edit the `conf/cassandra.yaml` file for all nodes.
3. Ensure that `cluster_name` is identical for all nodes.
4. Locate the seed nodes. This is the field for all Cassandra nodes; change it accordingly:
   - For the seed node, this will be its own port.
   - For the non-seed nodes, this will be the IP address of the seed node.
5. Ensure that `listen_address` is changed from `127.0.0.1` to the current IP address.
6. Ensure that `rpc_address` is changed from `127.0.0.1` to the current IP address.
7. Locate the `authenticator` field.
8. Change the value from `AllowAllAuthenticator` to `PasswordAuthenticator`.

   ```
   authenticator: PasswordAuthenticator
   ```

   Note: The full classname is `org.apache.cassandra.auth.PasswordAuthenticator`.
9. Save the file.
10. Repeat these steps on each external Cassandra instance.

11. Start all the cassandra nodes.

Execute CQL Commands

1. On the external Cassandra, using the cqlsh utility (included with Cassandra), create your username and password for **one** of the nodes (not all). The following example shows the creation of a `genesys` user with `genesys` password.

```
$ cqlsh -u cassandra -p cassandra cassandra_host cassandra_port
> CREATE USER genesys WITH PASSWORD 'genesys';
> LIST USERS;
    name | super
    -----------+-------
     genesys | False
     cassandra | True
```

> **Important**
>
> The default superuser is `cassandra` with password `cassandra`. This step is required to be completed on only one external Cassandra instance. It will then be replicated to the other nodes.

2. On Windows OS / Cassandra versions 2.1 or higher, replace:

```
$ cqlsh -u cassandra -p cassandra cassandra_host cassandra_port
```

with:

```
{path_to_cassandra}\bin>{path_to_python}\python.exe cqlsh cassandra_host -u cassandra -p cassandra
```

3. Set the options of `cqlsh` before parameters or set your python 2.7 path in PATH environment variable like this:

```
PATH={path_to_python};%PATH%
```

Therefore, you can launch the `cqlsh` script using the `cqlsh.bat` command:

```
cqlsh.bat -u cassandra -p cassandra cassandra_host
```

Using the default cassandra port of `native_transport_port` (default is `9042`). Otherwise you will need to add the port parameter to the `cqlsh` script.

4. Change the consistency level of the `system_auth` table and apply the `CREATE` command above according to the Cassandra version:

```
$ cqlsh -u cassandra -p cassandra cassandra_host cassandra_port
> ALTER KEYSPACE system_auth WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'DC1' : 2, 'DC2' : 2
```

Change the Replication Factor or Strategy

If you use authentication with Cassandra, the `system_auth` Cassandra system table has a replication factor of 1. In this scenario, only one node is aware of authentication).

If you change the strategy class or the replication factor, you must also report this change for all nodes of the cluster, as described here:

- Start all Cassandra nodes and changing the replication factor to 3.

```
cqlsh -u cassandra -p cassandra
ALTER KEYSPACE "system_auth" WITH REPLICATION = {'class' : 'SimpleStrategy', 'replication_factor' : 3};</source>
```

- Start repair task on all nodes:

```
nodetool repair
```

- Select a node (other than the first one) for testing:

```
cqlsh -u cassandra -p cassandra
```

Set Configuration Options

1. In Configuration Manager, locate and open your GMS Application object.
2. On the Options tab, **[cassandra-authentication-security]** section, set the following options with the same username and password that you just created on the external Cassandra.
   - **username**, for example, `genesys`
   - **password**, for example, `genesys`
3. **Restart GMS**. The Pelops and Hector clients connect to the external Cassandra using the login and password.

Authorization on External Cassandra

Updated in 8.5.207

> **Important**
> Supports Cassandra version 2.1.x and higher (≤ 3.11).

GMS 8.5.207+

Starting in 8.5.207, you only need to grant permissions in **one** of the nodes (not all) for the **genesys** user.

```
$ cqlsh -u cassandra -p cassandra cassandra_host cassandra_port
> LIST USERS; // or LIST ROLES; on cassandra 3.11 versions
        name | super
        -----------+-------
        genesys | False
        cassandra | True
> LIST ALL PERMISSIONS OF genesys;
        (0 rows)

> CREATE KEYSPACE gsg WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'DC1' : 2, 'DC2' : 2 };
> CREATE KEYSPACE gsg_dd WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'DC1' : 2, 'DC2' : 2 };
> GRANT CREATE ON KEYSPACE gsg TO genesys;
> GRANT MODIFY ON KEYSPACE gsg TO genesys;
```

```
> GRANT SELECT ON KEYSPACE gsg TO genesys;
> LIST ALL PERMISSIONS OF genesys;
username | resource | permission
        ----------+----------------+------------
        genesys | <keyspace gsg> | CREATE
        genesys | <keyspace gsg> | SELECT
        genesys | <keyspace gsg> | MODIFY
        (3 rows)
> GRANT CREATE ON KEYSPACE gsg_dd TO genesys;
> GRANT MODIFY ON KEYSPACE gsg_dd TO genesys;
> GRANT SELECT ON KEYSPACE gsg_dd TO genesys;
> LIST ALL PERMISSIONS OF genesys;
        username | resource | permission
        ----------+------------------+------------
        genesys | <keyspace gsg> | CREATE
        genesys | <keyspace gsg> | SELECT
        genesys | <keyspace gsg> | MODIFY
        genesys | <keyspace gsg_dd> | CREATE
        genesys | <keyspace gsg_dd> | SELECT
        genesys | <keyspace gsg_dd> | MODIFY
        (6 rows)
```

Versions Prior to 8.5.207

After creating the authentication, you must enable authorization and create keyspaces.

Configure the cassandra.yaml Files

1. Edit the `conf/cassandra.yaml` file for all nodes. Locate the `authorizer` field.
2. Change the value from `AllowAllAuthorizer` to `CassandraAuthorizer`.

   ```
   authorizer: CassandraAuthorizer
   ```

   Note: The full classname is `org.apache.cassandra.auth.CassandraAuthorizer`.
3. Save the file.
4. Repeat these steps on each external Cassandra instance.
5. Start all the cassandra nodes.

Execute CQL Commands

To authorize actions on the keyspace, you must first create the keyspace(s), then grant them permissions in **one** of the nodes (not all).

1. On the external Cassandra, using the `cqlsh` utility (included with Cassandra), create your keyspaces in one of the nodes. The following example shows the `gsg` and `gsg_dd` keyspaces.

> **Important**

> This step is required to be completed on only one external Cassandra instance. It will then be replicated to the other nodes.

2. Set permissions to GMS keyspaces in Cassandra using CQLSH (example for cassandra 2.1). Change the consistency level of GMS keyspaces and apply the CQL commands shown below according to the Cassandra version:

```
$ cqlsh -u cassandra -p cassandra cassandra_host cassandra_port
> LIST USERS;
        name | super
        -----------+-------
        genesys | False
        cassandra | True
> LIST ALL PERMISSIONS OF genesys;
        (0 rows)

> CREATE KEYSPACE gsg WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'DC1' : 2, 'DC2' : 2 };
> CREATE KEYSPACE gsg_dd WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'DC1' : 2, 'DC2' : 2 };
> GRANT ALTER ON KEYSPACE gsg TO genesys;
> GRANT CREATE ON KEYSPACE gsg TO genesys;
> GRANT DROP ON KEYSPACE gsg TO genesys;
> GRANT MODIFY ON KEYSPACE gsg TO genesys;
> GRANT SELECT ON KEYSPACE gsg TO genesys;
> LIST ALL PERMISSIONS OF genesys;
username | resource | permission
        ----------+---------------+------------
        genesys | <keyspace gsg> | CREATE
        genesys | <keyspace gsg> | ALTER
        genesys | <keyspace gsg> | DROP
        genesys | <keyspace gsg> | SELECT
        genesys | <keyspace gsg> | MODIFY
        (5 rows)
> GRANT ALTER ON KEYSPACE gsg_dd TO genesys;
> GRANT CREATE ON KEYSPACE gsg_dd TO genesys;
> GRANT DROP ON KEYSPACE gsg_dd TO genesys;
> GRANT MODIFY ON KEYSPACE gsg_dd TO genesys;
> GRANT SELECT ON KEYSPACE gsg_dd TO genesys;
> LIST ALL PERMISSIONS OF genesys;
        username | resource | permission
        ----------+------------------+------------
        genesys | <keyspace gsg> | CREATE
        genesys | <keyspace gsg> | ALTER
        genesys | <keyspace gsg> | DROP
        genesys | <keyspace gsg> | SELECT
        genesys | <keyspace gsg> | MODIFY
        genesys | <keyspace gsg_dd> | CREATE
        genesys | <keyspace gsg_dd> | ALTER
        genesys | <keyspace gsg_dd> | DROP
        genesys | <keyspace gsg_dd> | SELECT
        genesys | <keyspace gsg_dd> | MODIFY
        (10 rows)
```

3. Add the user to Cassandra by using the same CQLSH commands than for Authentication.
4. Restart GMS. The Pelops and Hector clients connect to the external Cassandra and are authorized to manage the GMS keyspaces (`gsg` and `gsg_dd`).

Final Steps

- In the GMS Application > Security section > Log On As SYSTEM Account.
- The time zone for all nodes must be identical. Make sure that you synchronize the time before testing.

Repairing Cassandra Nodes

> **Important**
> For detailed procedures about repairing nodes, refer to the Offical DataStax documentation.

According to the DataStax documentation, repairing nodes is part of your housekeeping. Schedule this operation regularly, as detailed in When to run anti-entropy repair.

For detailed information about commands, see Manual repair: Anti-entropy repair.

Do not restart a C* instance down for a few hours

If a C* instance is down for a few hours, do not restart it to join the cluster. If you do so, you will create some inconsistencies. If you do not mind creating a few inconsistencies and if the number of involved instances is 1-2 C* max, a full cluster repair will resolve the inconsistencies. If you don't perform a full cluster repair, some issues will remain.

Do not repair C* instances that are down for too long

If the C* instances are down for too long, assuming that enough C* instances are part of the cluster, the cluster should survive the death of some C* instances. For example, if the replication factor is 4, written as RF=4, then having 5+ instances allows 1 instance to die or to be decommissioned safely without impacting the cluster dataset. This allows intermittent issues on a C* instance as long as the QUORUM is available, which is RF/2 + 1 = 3 in this example.

If the quorum is available, you should remove the dead nodes from the cluster in a clean way, before eventually rejoining.

To remove the dead nodes cleanly, you can try the following procedure:

```
# check if some removal in progress (removenode status)
$ nodetool removenode status
...

# check if there are dead nodes (describecluster)
$ nodetool describecluster
...

# check token id for dead nodes (status)
$ nodetool status
```

```
Datacenter: DC2
===============
Status=Up/Down |/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Tokens  Owns    Host ID                               Rack
UN  192.168.XXX.XXX 155.97 KB  256     ?       a5303996-e61d-4463-a5cd-bde36c8762f5  RAC2
UN  192.168.XXX.XXX  165.72 KB  256     ?       17c8c1d8-08bc-489b-85eb-92bc9bc0dedb  RAC1
Datacenter: DC1
===============
Status=Up/Down |/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Tokens  Owns    Host ID                               Rack
UN  192.168.XXX.XXX  116.01 KB  256     ?       2c75bf48-66ea-47fb-a4f4-0664c82060cb  RAC1
UN  192.168.XXX.XXX  101.8 KB   256     ?       3a61a518-b6e1-42f1-90f5-f8349f66fd3d  RAC2

# remove the dead token (removenode) using the *Host ID* field of the down node to be removed
$ nodetool removenode <token>
...
# If this command hangs for more than a minute cancel it (e.g. press Ctrl-C) and run:
$ nodetool removenode force
```

From this point, the removed C* instance is no longer part of C* cluster. You can now delete its data on disk. To add the C* instance back into the cluster like a new node, restart it.

> **Important**
>
> By default, the data folder contains all data related to the instance replica (data, commitlog, and more). Other folders might be found under the root `data` folder). You can delete all the content of the data folder.

Frequently Asked Questions

The 8.5.206 release introduces an important change of Cassandra driver.

What is the significance of adding all the nodes/IP addresses in the node section of the Cassandra option?

The driver used by GMS to connect to Cassandra supports setting multiple Cassandra instances.

- For versions older than 8.5.206, these instances are used as Cassandra coordinators. If one instance dies, the driver automatically switches to the next Cassandra instance that is used as coordinator.
- Starting in 8.5.206, other Cassandra instances are automatically detected. If one of them dies, the driver connects to all the detected Cassandra instances, including those that are not defined in the GMS options.

Can I add only one seed node in place of all the nodes/IP addresses in the node section of the Cassandra option?

Yes. However, adding only one seed node means that GMS will not be able to start if this instance is down.

- For versions older than 8.5.206, this also means that, after a correct GMS start, if your Cassandra node dies, GMS will die too because it can reach known Cassandra instances only.
- Starting in 8.5.206, other Cassandra instances are automatically detected.

My GMS Application is not starting up when one node/host server is in hung or downstate; what can be a permanent fix for this issue?

Genesys recommends that you upgrade to 8.5.206+. This scenario has been tested and it does not reproduce starting in 8.5.206.04. Note that this release also adds other features, like a secured connection from GMS to Cassandra.