Oracle® Communications

DB Tier User Guide

Release 1.7.0

F38027-01

January 2021

Oracle Communications DB Tier User Guide, Release 1.7.0

F38027-01

Contents

My Oracle Support

My Oracle Support (https://support.oracle.com) is your initial point of contact for all product support and training needs. A representative at Customer Access Support can assist you with My Oracle Support registration.

Call the Customer Access Support main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html. When calling, make the selections in the sequence shown below on the Support telephone menu:

1. Select **2** for New Service Request.
2. Select **3** for Hardware, Networking and Solaris Operating System Support.
3. Select one of the following options:
    - For Technical issues such as creating a new Service Request (SR), select **1**.
    - For Non-technical issues such as registration or assistance with My Oracle Support, select **2**.

You are connected to a live agent who can assist you with My Oracle Support registration and opening a support ticket.

My Oracle Support is available 24 hours a day, 7 days a week, 365 days a year.

What's New in This Guide

This section introduces the documentation updates for Release 1.7.x in Oracle Communications DB Tier User Guide.

Metric Type is added for all the metrics under DB Tier Metrics chapter.

1 Introduction

The database (DB) Tier is the geo-diverse database layer provided as part of Oracle Communications Cloud Native Environment (OCCNE). This document provides the metrics and alerts of DB Tier and the manual configuration procedure of DB Tier Node for TLS support.

2
DB Tier Metrics and Alerts

DB Tier generates metrics that can be used by the end users. The metrics, and the alerts generated by DB Tier can be seen in the Prometheus dashboard which could be used by the end users to take necessary actions. Prometheus gets installed as part of common services during the vCNE installation. Following are the available DB Tier metrics and alerts for the end users:

DB Tier Metrics

Following sections list the metrics in DB Tier.

DB Tier Node Status Metrics

Table 2-1 DB Tier Node Status Metrics

Metric Name	Parameters	Values	Metric Type	Description
db_tier_node_status	Node ID, Node Type, Node version	Node ID - `node_id` of the DB node Node Type - `node_type` of the DB node (Data Node, Management Node or SQL Node) Node version: `node_version` DB Tier Cluster software version	Gauge	DB Tier node status of the node with Node ID as "node_id", Node Type as "node_type" and Node version as "node_version". The value of this metrics is "0" if the node is DOWN and "1" if the node is UP.

DB Tier Table Read Write Metrics

Table 2-2 DB Tier Table Read Write Metrics

Metric Name	Parameters	Values	Metric Type	Description
db_tier_local_operations	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total number of local operations in DB Tier for the node.
db_tier_transactions	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total number of transactions in DB Tier for the node.
db_tier_commits	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total number of commits in DB Tier for the node.
db_tier_reads	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total number of reads in DB Tier for the node.
db_tier_local_reads	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total number of local reads in DB Tier for the node.
db_tier_writes	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total number of writes in DB Tier cluster for the node.

Table 2-2 (Cont.) DB Tier Table Read Write Metrics

Metric Name	Parameters	Values	Metric Type	Description
db_tier_local_writes	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total number of local writes in DB Tier for the node.
db_tier_aborts	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total number of aborted transactions in DB Tier for the node.
db_tier_table_scans	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total no of table scan in DB Tier for the node.
db_tier_range_scans	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Total number of range scans in DB Tier for the node.
db_tier_transporter_overload	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Transporter overload in DB Tier for the node.
db_tier_scan_slowdown	Node ID, Block Name, Block Instance	Node ID: `node_id` Block Name: `block_name` Block Instance: `block_instance`	Gauge	Scan slowdown in DB Tier for the node.

DB Tier CPU Usage Metrics

Table 2-3 DB Tier CPU Usage Metrics

Metric Name	Parameters	Values	Metric Type	Description
db_tier_cpu_os_user	Node ID, Thread	Node ID: `node_id` Thread: `thread ID`	Gauge	DB Tier User CPU usage for the node.
db_tier_cpu_os_system	Node ID, Thread	Node ID: `node_id` Thread: `thread ID`	Gauge	DB Tier System CPU usage for the node.
db_tier_cpu_os_idle	Node ID, Thread	Node ID: `node_id` Thread: `thread ID`	Gauge	Idle CPU statistics for the node.

DB Tier Memory Usage Metrics

Table 2-4 DB Tier Memory Usage Metrics

Metric Name	Parameters	Values	Metric Type	Description
db_tier_memory_used_bytes	Node ID Memory Type	Node ID: `node_id` Memory Type: `memory_type`	Gauge	Memory used for the node.
db_tier_memory_total_bytes	Node ID Memory Type	Node ID: `node_id` Memory Type: `memory_type`	Gauge	Total memory for the node.

DB Tier Bin Log Usage Metrics

Table 2-5 DB Tier Bin Log Usage Metrics

Metric Name	Parameters	Values	Metric Type	Description
db_tier_binlog_used_bytes_percentage	Node ID	Node ID: `node_id`	Gauge	Percentage of total memory used by bin log in the SQL node.

DB Tier Replication Metrics

Table 2-6 DB Tier Replication Metrics

Metric Name	Parameters	Values	Metric Type	Description
db_tier_replication_status	Node ID Source ID	Node ID: `node_id` Source ID: `source_uuid`	Gauge	This Metrics value is: 1. "0" : Replication Channel Status of local site is ON 2. "1" : Replication Channel Status of local site is OFF 3. "2" : Replication Channel Status of local site is CONNECTING
db_tier_replication_slave_delay	Channel ID Master Node IP Slave Node IP	Channel ID: `channel_id` Master Node IP: `master_node_ip` Slave Node IP: `slave_node_ip`	Gauge	Number of seconds that the last record read by the slave is behind the latest record written by the master

DB Tier Alerts

Table 2-7 DB Tier Alerts

Alert Name	Summary	Severity	Expression	For	SNMP Trap ID	Service Affecting?	Notes
NODE_DOWN	MySQL {{ $labels.node_type }} node having node id {{ $labels.node_id }} is down	major	db_tier_data_node_status == 0	N/A	2001	Y	db_tier_data_node_status value "0" indicates that a node is DOWN and value "1" indicates that the node is UP.
HIGH_CPU	Node ID {{ $labels.node_id }} CPU utilization at {{ value }} percent.	warning	(100 - (avg(avg_over_time(db_tier_cpu_os_idle[10m]))BY (node_id)))>= 85	1m	2002	N	HIGH_CPU alert is fired when CPU usage of any node >=85%

Table 2-7 (Cont.) DB Tier Alerts

Alert Name	Summary	Severity	Expression	For	SNMP Trap ID	Service Affecting?	Notes
LOW_MEMORY	Node ID {{ $labels.node_id }} memory utilization at {{ value }} percent.	major	(avg_over_time(db_tier_memory_used_bytes[1m])BY (node_id,memory_type) / avg_over_time(db_tier_memory_total_bytes[1m])BY (node_id, memory_type)) * 100>= 80	1m	2003	N	LOW_MEMORY alert is fired when RAM usage of any node >=80%
OUT_OF_MEMORY	Node ID {{ $labels.node_id }} out of memory.	critical	(avg_over_time(db_tier_memory_used_bytes[1m])BY (node_id,memory_type) / avg_over_time(db_tier_memory_total_bytes[1m])BY (node_id, memory_type)) * 100>= 90	N/A	2004	Y	OUT_OF_MEMORY alert is fired when RAM usage of any node >= 90%
BINLOG_STORAGE_LOW	Disk storage on SQL node with node ID {{ $labels.node_id }} at {{ $value }} percent	minor	(avg_over_time(db_tier_binlog_used_bytes_percentage[5m]) >= 70) and (avg_over_time(db_tier_binlog_used_bytes_percentage[5m]) < 80)	5m	2007	N	BINLOG_STORAGE_LOW alert is fired with Minor Severity when the total BinLog size of the SQL node is >=70% and <80% of Total SQL node Disk size.

Table 2-7 (Cont.) DB Tier Alerts

Alert Name	Summary	Severity	Expression	For	SNMP Trap ID	Service Affecting?	Notes
BINLOG_STORAGE_LOW	Disk storage on SQL node with node ID {{ $labels.node_id }} at {{ $value }} percent	major	(avg_over_time(db_tier_binlog_used_bytes_percentage[5m]) >= 80) and (avg_over_time(db_tier_binlog_used_bytes_percentage[5m])< 95)	5m	2007	N	BINLOG_STORAGE_LOW alert is fired with Major Severity when the total BinLog size of the SQL node is >=80% and <95% of Total SQL node Disk size.
BINLOG_STORAGE_FULL	Disk storage on SQL node with node ID {{ $labels.node_id }} is full	critical	avg_over_time(db_tier_binlog_used_bytes_percentage[5m]) >= 95		2008	Y	BINLOG_STORAGE_LOW alert is fired with Critical Severity when the total BinLog size of the SQL node is >=95% of Total SQL node Disk size.
SLAVE_REPLICATION_DELAY_HIGH	Slave replication on SQL node at {{ $labels.slave_ip }} is {{ $value }} seconds behind the master	major	avg(avg_over_time(db_tier_replication_slave_delay[5m])) by (master_node_ip, slave_node_ip) >= 5*60 and avg(avg_over_time(db_tier_replication_slave_delay[5m])) by (master_node_ip, slave_node_ip) < 48*3600		2009	N	The last record read by the slave is more than 5 minutes behind the latest record written by the master.

Table 2-7 (Cont.) DB Tier Alerts

Alert Name	Summary	Severity	Expression	For	SNMP Trap ID	Service Affecting?	Notes
SLAVE_REPLICATION_FAILED	Slave replication has fallen more than 48 hours behind the master. Manual restore from backup may be required.	critical	avg(avg_over_time(db_tier_replication_slave_delay[5m])) by (master_node_ip, slave_node_ip) >= 48*3600		2010	Y	The last record read by the slave is more than 48 hours behind the latest record written by the master.

3

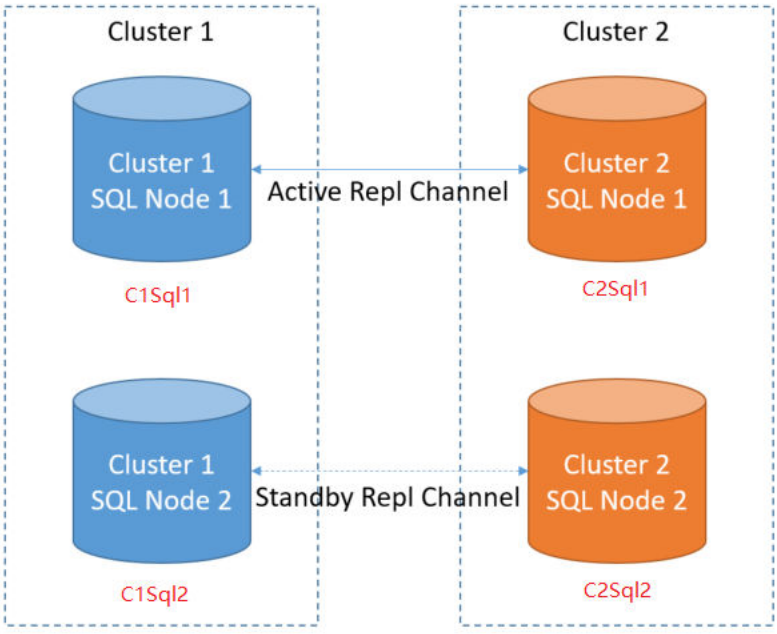
Configuring DB Tier SQL Nodes to Support TLS for Geo Replication

This chapter provides manual procedure to configure DB Tier SQL nodes to support TLS for geo replication channels between two mate sites for OCCNE customer.

Prerequisites

1. Make sure that both the Geo redundant mate sites are setup and running.
2. Ensure that the active replication has been setup between Cluster 1 SQL Node 1 (C1Sql1) and Cluster 2 SQL Node 1 (C2Sql1).
3. Make sure that the standby replication channel has been configured between Cluster 1 SQL Node 2 (C1Sql2) and Cluster 2 SQL Node 2 (C2Sql2) and ready to be promoted to active replication channel when active replication is stopped/failed.

Figure 3-1 Geo-Replication setup



4. Following SSL certificates and private keys for each of the DB Tier SQL nodes with the identical names are required for setting up the encrypted connection between geo redundant sites:

Table 3-1 SSL Certificates

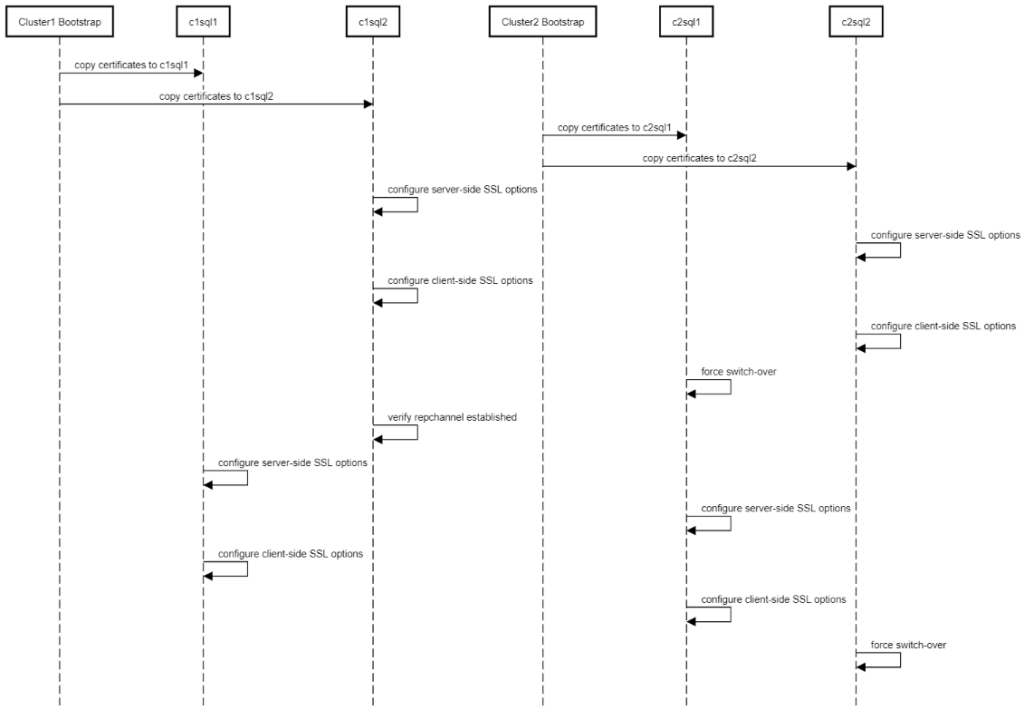
File Name	Description
ca.pem	A certificate authority (CA) certification for self-signing server-side and client-side certificates and private keys.
ca-key.pem	The matching private key for the CA certification file.
server-cert.pem	server-side self-signed certificate for enabling encrypted connection on master.
server-key.pem	The matching private key for the server side certificate file.
client-cert.pem	Slave certificate for connecting to master via encrypted connection.
client-key.pem	The matching private key for slave certificate.

You should use same set of certificates/private keys for all the DB Tier SQL nodes in both the sites or different set of certificates/private keys for each of the sites or each of the DB Tier SQL nodes.

Configuring DB Tier SQL Node

The following diagram shows the geo replication setup between two sites (Cluster1 and Cluster2):

Figure 3-2 Call Flow of TLS Support between Geo Replication Channels



Configuring DB Tier SQL Nodes

The following procedure describes how to configure DB Tier nodes to support TLS for Geo Replication:

1. Ensure that the certificates and private key files mentioned in the prerequisites are available in `/var/occnedb/mysqlcerts` directory on every SQL node of both the sites.

   > **Note:**
   >
   > On bare metal installation, login as 'admusr'; on vCNE installation, login as 'cloud-user'.

   a. Check if the certificates and private key files are available in `/var/occnedb/mysqlcerts` directory:

   ```
   [cloud-user@occne1-cgbu-cne-dbtier-db-data-sql-nf-1 mysqlcerts]$
   pwd
   /var/occnedb/mysqlcerts
   [cloud-user@occne1-cgbu-cne-dbtier-db-data-sql-nf-1 mysqlcerts]$
   ll
   total 32
   -rw-------. 1 cloud-user cloud-user 1675 Jul  6 07:29 ca-key.pem
   -rw-------. 1 cloud-user cloud-user 1131 Jul  6 07:29 ca.pem
   -rw-------. 1 cloud-user cloud-user 1135 Jul  6 07:29 client-
   cert.pem
   -rw-------. 1 cloud-user cloud-user 1679 Jul  6 07:29 client-
   key.pem
   -rw-------. 1 cloud-user cloud-user 1675 Jul  6 07:29
   private_key.pem
   -rw-------. 1 cloud-user cloud-user  451 Jul  6 07:29
   public_key.pem
   -rw-------. 1 cloud-user cloud-user 1135 Jul  6 07:29 server-
   cert.pem
   -rw-------. 1 cloud-user cloud-user 1675 Jul  6 07:29 server-
   key.pem
   ```

   b. Change file access permission for certificates and private key files (for bare metal setup, change 'cloud-user' to 'admusr').

   ```
   [cloud-user@occne1-cgbu-cne-dbtier-db-data-sql-nf-1 mysqlcerts]$
   chmod -R 600 /var/occnedb/mysqlcerts/*;
   [cloud-user@occne1-cgbu-cne-dbtier-db-data-sql-nf-1 mysqlcerts]$
   chown -R cloud-user:cloud-user /var/occnedb/mysqlcerts;
   ```

   c. In case you are using different set of certificates, copy client CA, certificate and private key from its master SQL node to the slave node, for example, copy from ***C2Sql1*** to ***C1Sql1***. On ***C1Sql1***, copy ca.pem, client-cert.pem and client-key.pem to /var/occnedb/mysqlcerts/c2sql1 directory after creating c2sql1 directory:

   ```
   [cloud-user@occne1-cgbu-cne-dbtier-bootstrap mysqlcerts]$ ssh
   cloud-user@"10.75.226.51" 'mkdir /var/occnedb/mysqlcerts/c2sql1'
   [cloud-user@occne1-cgbu-cne-dbtier-bootstrap
   mysqlcerts]$ scp /var/terraform/mysqlcerts/c2sql1/* cloud-
   user@"10.75.226.51":/var/occnedb/mysqlcerts/c2sql1/
   ```

```
ca.pem                  100% 1131     1.1MB/s   00:00
client-cert.pem         100% 1135     1.3MB/s   00:00
client-key.pem          100% 1675     1.9MB/s   00:00
```

d. On ***C1Sql2***, copy ca.pem, client-cert.pem and client-key.pem to /var/occnedb/mysqlcerts/c2sql2 directory after creating c2sql2 directory:

```
[cloud-user@occne1-cgbu-cne-dbtier-bootstrap mysqlcerts]$ ssh
cloud-user@"10.75.225.195" 'mkdir /var/occnedb/mysqlcerts/c2sql2'
[cloud-user@occne1-cgbu-cne-dbtier-bootstrap
mysqlcerts]$ scp /var/terraform/mysqlcerts/c2sql2/* cloud-
user@"10.75.225.195":/var/occnedb/mysqlcerts/c2sql2/
ca.pem                  100% 1131     1.1MB/s   00:00
client-cert.pem         100% 1135     1.3MB/s   00:00
client-key.pem          100% 1675     1.9MB/s   00:00
```

e. On ***C2Sql1***, copy ca.pem, client-cert.pem and client-key.pem to /var/occnedb/mysqlcerts/c1sql1 directory after creating c1sql1 directory:

```
[cloud-user@occne2-cgbu-cne-dbtier-bootstrap mysqlcerts]$ ssh
cloud-user@"10.75.153.106" 'mkdir /var/occnedb/mysqlcerts/c1sql1'
[cloud-user@occne2-cgbu-cne-dbtier-bootstrap
mysqlcerts]$ scp /var/terraform/mysqlcerts/c1sql1/* cloud-
user@"10.75.153.106":/var/occnedb/mysqlcerts/c1sql1/
ca.pem                  100% 1131     1.1MB/s   00:00
client-cert.pem         100% 1135     1.3MB/s   00:00
client-key.pem          100% 1675     1.9MB/s   00:00
```

f. On ***C2Sql2***, copy ca.pem, client-cert.pem and client-key.pem to /var/occnedb/mysqlcerts/c2sql1 directory after creating c2sql1 directory:

```
[cloud-user@occne2-cgbu-cne-dbtier-bootstrap mysqlcerts]$ ssh
cloud-user@"10.75.153.196" 'mkdir /var/occnedb/mysqlcerts/c2sql1'
[cloud-user@occne2-cgbu-cne-dbtier-bootstrap
mysqlcerts]$ scp /var/terraform/mysqlcerts/c2sql1/* cloud-
user@"10.75.153.196":/var/occnedb/mysqlcerts/c2sql1/
ca.pem                  100% 1131     1.1MB/s   00:00
client-cert.pem         100% 1135     1.3MB/s   00:00
client-key.pem          100% 1675     1.9MB/s   00:00
```

2. On ***C1Sql2***, login to the MySQL Cluster Management (MCM) client, stop MySQL process on this SQL node (by specifying its process ID, i.e. 57) and configure server-side SSL options by executing the following commands:

   a. Stop the process (ID: 57):

```
mcm> stop process 57 occnendbclustera;
+-----------------------------+
| Command result              |
+-----------------------------+
| Process stopped successfully |
+-----------------------------+
1 row in set (6.65 sec)
```

b. Set the SSL Certificate Authority (CA) for the process (ID: 57):

```
mcm> set ssl-ca:mysqld:57=/var/occnedb/mysqlcerts/ca.pem
occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.38 sec)
```

c. Set the server-side SSL certificate to the process (ID: 57):

```
mcm> set ssl-cert:mysqld:57=/var/occnedb/mysqlcerts/server-
cert.pem occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

d. Set the server-side secret key to the process (ID: 57):

```
mcm> set ssl-key:mysqld:57=/var/occnedb/mysqlcerts/server-
key.pem occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

e. Set the TLS certificate version to the process (ID: 57):

```
mcm> set tls_version:mysqld:57=TLSv1.2 occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

f. Set the cipher type to the process (ID: 57):

```
mcm> set ssl-cipher:mysqld:57=DHE-RSA-AES128-GCM-SHA256
occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.28 sec)
```

g. Start the process (ID: 57):

```
mcm> start process 57 occnendbclustera;
+-----------------------------+
| Command result              |
+-----------------------------+
| Process started successfully |
+-----------------------------+
1 row in set (2.57 sec)
```

h. Exit MCM client and login to MySQL client with root privilege, perform the following steps:

i. Alter 'occnerepluser' account to "X509" by executing the following command:

```
mysql> ALTER USER 'occnerepluser'@'%' REQUIRE X509;
Query OK, 0 rows affected (0.01 sec)
```

ii. Flush the privileges to reload the grant tables:

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

3. On ***C2Sql2***, login to the MCM (MySQL Cluster Management) client, stop MySQL process on this SQL node (by specifying its process ID, i.e. 57) and configure server-side SSL options by executing the following commands:

a. Stop the process (ID: 57):

```
mcm> stop process 57 occnendbclustera;
+-----------------------------+
| Command result              |
+-----------------------------+
| Process stopped successfully |
+-----------------------------+
1 row in set (6.65 sec)
```

b. Set SSL CA to the process (ID: 57):

```
mcm> set ssl-ca:mysqld:57=/var/occnedb/mysqlcerts/ca.pem
occnendbclustera;
+---------------------------------+
| Command result                  |
+---------------------------------+
| Cluster reconfigured successfully |
+---------------------------------+
1 row in set (0.38 sec)
```

c. Set the sever-side SSL certificate to the process (ID: 57)

```
mcm> set ssl-cert:mysqld:57=/var/occnedb/mysqlcerts/server-
cert.pem occnendbclustera;
+---------------------------------+
| Command result                  |
```

```
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

d. Set the server-side secret key to the process (ID: 57):

```
mcm> set ssl-key:mysqld:57=/var/occnedb/mysqlcerts/server-
key.pem occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

e. Set the TLS certificate to the process (ID: 57):

```
mcm> set tls_version:mysqld:57=TLSv1.2 occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

f. Set the cipher type to the process (ID: 57):

```
mcm> set ssl-cipher:mysqld:57=DHE-RSA-AES128-GCM-SHA256
occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.28 sec)
```

g. Start the process (ID: 57):

```
mcm> start process 57 occnendbclustera;
+------------------------------+
| Command result               |
+------------------------------+
| Process started successfully |
+------------------------------+
1 row in set (2.57 sec)
```

h. Exit MCM client and login to MySQL client with root privilege and perform the following steps:

i. Alter 'occnerepluser' account to "X509" by executing the following command:

```
mysql> ALTER USER 'occnerepluser'@'%' REQUIRE X509;
Query OK, 0 rows affected (0.01 sec)
```

ii. Flush the privileges to reload the grant tables:

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

4. Login back to ***C1Sql2*** MySQL client, verify that server-side SSL configuration is applied correctly and perform client-side SSL configuration.

   a. Verify that 'have_openssl' and 'have_ssl' fields are set to "YES", and that 'ssl_ca', 'ssl_cert' and 'ssl_key' fields are set to the correct path of your certificates:

```
mysql> show variables like '%ssl%';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| have_openssl  |        YES                              |
| have_ssl      |        YES                              |
| ssl_ca        | /var/occnedb/mysqlcerts/ca.pem          |
| ssl_capath    |                                         |
| ssl_cert      | /var/occnedb/mysqlcerts/server-cert.pem |
| ssl_cipher    | DHE-RSA-AES128-GCM-SHA256               |
| ssl_crl       |                                         |
| ssl_crlpath   |                                         |
| ssl_key       | /var/occnedb/mysqlcerts/server-key.pem  |
+---------------+-----------------------------------------+
9 rows in set (0.01 sec)
```

   b. If you using same set of certificates as in server-side, run the following commands:

```
mysql> CHANGE MASTER TO MASTER_SSL=1, MASTER_SSL_CA
= '/var/occnedb/mysqlcerts/ca.pem', MASTER_SSL_CERT = '/var/
occnedb/mysqlcerts/client-cert.pem', MASTER_SSL_KEY = '/var/
occnedb/mysqlcerts/client-key.pem', Master_SSL_Cipher='DHE-RSA-
AES128-GCM-SHA256', Master_TLS_Version='TLSv1.2';
Query OK, 0 rows affected (0.03 sec)
```

   c. If using different set of certificates, run the following commands:

```
mysql> CHANGE MASTER TO MASTER_SSL=1,
MASTER_SSL_CA = '/var/occnedb/mysqlcerts/c2sql2/ca.pem',
MASTER_SSL_CERT = '/var/occnedb/mysqlcerts/c2sql2/client-
cert.pem', MASTER_SSL_KEY = '/var/occnedb/mysqlcerts/c2sql2/
client-key.pem', Master_SSL_Cipher='DHE-RSA-AES128-GCM-SHA256',
Master_TLS_Version='TLSv1.2';
Query OK, 0 rows affected (0.03 sec)
```

5. Login back to ***C2Sql2*** and repeat same operations as mentioned in Step 4:

a. Check the variables using the following command:

```
mysql> show variables like '%ssl%';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| have_openssl  | YES                                     |
| have_ssl      | YES                                     |
| ssl_ca        | /var/occnedb/mysqlcerts/ca.pem          |
| ssl_capath    |                                         |
| ssl_cert      | /var/occnedb/mysqlcerts/server-cert.pem |
| ssl_cipher    | DHE-RSA-AES128-GCM-SHA256               |
| ssl_crl       |                                         |
| ssl_crlpath   |                                         |
| ssl_key       | /var/occnedb/mysqlcerts/server-key.pem  |
+---------------+-----------------------------------------+
9 rows in set (0.01 sec)
```

b. If you are using same set of certificates as in C1Sql2, run the following commands:

```
mysql> CHANGE MASTER TO MASTER_SSL=1, MASTER_SSL_CA
= '/var/occnedb/mysqlcerts/ca.pem', MASTER_SSL_CERT = '/var/
occnedb/mysqlcerts/client-cert.pem', MASTER_SSL_KEY = '/var/
occnedb/mysqlcerts/client-key.pem', Master_SSL_Cipher='DHE-RSA-
AES128-GCM-SHA256', Master_TLS_Version='TLSv1.2';
Query OK, 0 rows affected (0.03 sec)
```

c. If using different set of certificates, run the following commands:

```
mysql> CHANGE MASTER TO MASTER_SSL=1,
MASTER_SSL_CA = '/var/occnedb/mysqlcerts/c1sql2/ca.pem',
MASTER_SSL_CERT = '/var/occnedb/mysqlcerts/c1sql2/client-
cert.pem', MASTER_SSL_KEY = '/var/occnedb/mysqlcerts/c1sql2/
client-key.pem', Master_SSL_Cipher='DHE-RSA-AES128-GCM-SHA256',
Master_TLS_Version='TLSv1.2';
Query OK, 0 rows affected (0.03 sec)
```

6. Login to ***C2Sql1*** MySQL client, check slave status and then stop the slave to force a switch over:

```
mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 10.75.226.51
                  Master_User: occnerepluser
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000003
          Read_Master_Log_Pos: 9365
               Relay_Log_File: mysql-relay-bin.000002
                Relay_Log_Pos: 4437
        Relay_Master_Log_File: mysql-bin.000003
             Slave_IO_Running: Yes
```

```
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 9365
              Relay_Log_Space: 4644
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 3000
                  Master_UUID: ab4bab28-c59b-11ea-846a-fa163f91fe89
             Master_Info_File: /var/occnedb/mysql/master.info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log;
waiting for more updates
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp:
                Master_SSL_Crl:
            Master_SSL_Crlpath:
            Retrieved_Gtid_Set:
             Executed_Gtid_Set:
                 Auto_Position: 0
          Replicate_Rewrite_DB:
                  Channel_Name:
           Master_TLS_Version:
1 row in set (0.00 sec)


mysql> stop slave;
Query OK, 0 rows affected (0.00 sec)
```

7. Login to ***C1Sql2*** MySQL client, verify that slave is started with an active channel. Verify that 'Master_User' field is "occnerepluser", 'Slave_IO_Running' field is set to "Yes" and "Slave_SQL_Running" field is set to "Yes":

```
mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 10.75.153.106
                  Master_User: occnerepluser
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000005
          Read_Master_Log_Pos: 5022
               Relay_Log_File: mysql-relay-bin.000002
                Relay_Log_Pos: 4382
        Relay_Master_Log_File: mysql-bin.000005
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 5022
              Relay_Log_Space: 4589
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: Yes
           Master_SSL_CA_File: /var/occnedb/mysqlcerts/ca.pem
           Master_SSL_CA_Path:
              Master_SSL_Cert: /var/occnedb/mysqlcerts/client-
cert.pem
            Master_SSL_Cipher: DHE-RSA-AES128-GCM-SHA256
               Master_SSL_Key: /var/occnedb/mysqlcerts/client-
key.pem
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1000
                  Master_UUID: c4f6833c-c59d-11ea-972f-fa163f3da281
             Master_Info_File: /var/occnedb/mysql/master.info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log;
waiting for more updates
           Master_Retry_Count: 86400
```

```
                Master_Bind:
    Last_IO_Error_Timestamp:
   Last_SQL_Error_Timestamp:
             Master_SSL_Crl:
         Master_SSL_Crlpath:
         Retrieved_Gtid_Set:
          Executed_Gtid_Set:
              Auto_Position: 0
      Replicate_Rewrite_DB:
               Channel_Name:
        Master_TLS_Version: TLSv1.2
1 row in set (0.00 sec)
```

8. Login to ***C1Sql1*** MCM client, stop MySQL process on this SQL node (by specifying its process ID, i.e. 56) and configure server-side SSL options by executing the following commands:

   a. Stop the process (ID: 56):

   ```
   mcm> stop process 56 occnendbclustera;
   +-----------------------------+
   | Command result              |
   +-----------------------------+
   | Process stopped successfully |
   +-----------------------------+
   1 row in set (6.65 sec)
   ```

   b. Set the SSL CA to the process (ID: 56):

   ```
   mcm> set ssl-ca:mysqld:56=/var/occnedb/mysqlcerts/ca.pem
   occnendbclustera;
   +----------------------------------+
   | Command result                   |
   +----------------------------------+
   | Cluster reconfigured successfully |
   +----------------------------------+
   1 row in set (0.38 sec)
   ```

   c. Set the server-side SSL certificate to the process (ID: 56):

   ```
   mcm> set ssl-cert:mysqld:56=/var/occnedb/mysqlcerts/server-
   cert.pem occnendbclustera;
   +----------------------------------+
   | Command result                   |
   +----------------------------------+
   | Cluster reconfigured successfully |
   +----------------------------------+
   1 row in set (0.30 sec)
   ```

   d. Set the server-side SSL key to the process (ID: 56):

   ```
   mcm> set ssl-key:mysqld:56=/var/occnedb/mysqlcerts/server-
   key.pem occnendbclustera;
   +----------------------------------+
   | Command result                   |
   ```

```
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

e. Set the TLS certificate to the process (ID: 56):

```
mcm> set tls_version:mysqld:56=TLSv1.2 occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

f. Set the cipher type to the process (ID: 56):

```
mcm> set ssl-cipher:mysqld:56=DHE-RSA-AES128-GCM-SHA256
occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.28 sec)
```

g. Start the process (ID: 56):

```
mcm> start process 56 occnendbclustera;
+------------------------------+
| Command result               |
+------------------------------+
| Process started successfully |
+------------------------------+
1 row in set (2.57 sec)
```

h. Login to MySQL client, alter 'occnerepluser' account to "X509" by executing the following command:

i. Alter 'occnerepluser' account to "X509" by executing the following command:

```
mysql> ALTER USER 'occnerepluser'@'%' REQUIRE X509;
Query OK, 0 rows affected (0.01 sec)
```

ii. Flush the privileges to reload the grant tables:

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

9. Login to ***C2Sql1*** MCM client, stop MySQL process on this SQL node (by specifying its process ID: 56) and configure server-side SSL options by executing the following commands:

a. Stop the MySQL process:

```
mcm> stop process 56 occnendbclustera;
+-----------------------------+
| Command result              |
+-----------------------------+
| Process stopped successfully |
+-----------------------------+
1 row in set (6.65 sec)
```

b. Set the SSL Certificate Authority to the process (ID: 56):

```
mcm> set ssl-ca:mysqld:56=/var/occnedb/mysqlcerts/ca.pem
occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.38 sec)
```

c. Set the server-side SSL certificate to the process (ID: 56):

```
mcm> set ssl-cert:mysqld:56=/var/occnedb/mysqlcerts/server-
cert.pem occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

d. Set the SSL key to the process (ID: 56):

```
mcm> set ssl-key:mysqld:56=/var/occnedb/mysqlcerts/server-
key.pem occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

e. Set the TLS version to the process (ID: 56):

```
mcm> set tls_version:mysqld:56=TLSv1.2 occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.30 sec)
```

f. Set the cipher suites to the process (ID: 56):

```
mcm> set ssl-cipher:mysqld:56=DHE-RSA-AES128-GCM-SHA256
occnendbclustera;
+----------------------------------+
| Command result                   |
+----------------------------------+
| Cluster reconfigured successfully |
+----------------------------------+
1 row in set (0.28 sec)
```

g. Start the process (ID: 56):

```
mcm> start process 56 occnendbclustera;
+-----------------------------+
| Command result              |
+-----------------------------+
| Process started successfully |
+-----------------------------+
1 row in set (2.57 sec)
```

h. Login to MySQL client, alter 'occnerepluser' account to require "X509" by executing the following command:

   i. Alter 'occnerepluser' account to "X509" by executing the following command:

```
mysql> ALTER USER 'occnerepluser'@'%' REQUIRE X509;
Query OK, 0 rows affected (0.01 sec)
```

   ii. Flush the privileges to reload the grant tables:

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

10. Login back to ***C1Sql1*** MySQL client, verify that server-side SSL configuration is applied correctly and perform client-side SSL configuration. Verify that 'have_openssl' and 'have_ssl' fields are set to "YES", verify that 'ssl_ca', 'ssl_cert' and 'ssl_key' fields are set the correct certificate/key file path:

    a. Check the values of MySQL variables using the following command:

```
mysql> show variables like '%ssl%';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| have_openssl  | YES                                     |
| have_ssl      | YES                                     |
| ssl_ca        | /var/occnedb/mysqlcerts/ca.pem          |
| ssl_capath    |                                         |
| ssl_cert      | /var/occnedb/mysqlcerts/server-cert.pem |
| ssl_cipher    | DHE-RSA-AES128-GCM-SHA256               |
| ssl_crl       |                                         |
| ssl_crlpath   |                                         |
| ssl_key       | /var/occnedb/mysqlcerts/server-key.pem  |
```

```
+--------------+----------------------------------------+
9 rows in set (0.01 sec)
```

b. If you are using the same set of certificates, run the following command:

```
mysql> CHANGE MASTER TO MASTER_SSL=1, MASTER_SSL_CA
= '/var/occnedb/mysqlcerts/ca.pem', MASTER_SSL_CERT = '/var/
occnedb/mysqlcerts/client-cert.pem', MASTER_SSL_KEY = '/var/
occnedb/mysqlcerts/client-key.pem', Master_SSL_Cipher='DHE-RSA-
AES128-GCM-SHA256', Master_TLS_Version='TLSv1.2';
Query OK, 0 rows affected (0.03 sec)
```

c. If using different set of certificates, run the following command:

```
mysql> CHANGE MASTER TO MASTER_SSL=1,
MASTER_SSL_CA = '/var/occnedb/mysqlcerts/c2sql1/ca.pem',
MASTER_SSL_CERT = '/var/occnedb/mysqlcerts/c2sql1/client-
cert.pem', MASTER_SSL_KEY = '/var/occnedb/mysqlcerts/c2sql1/
client-key.pem', Master_SSL_Cipher='DHE-RSA-AES128-GCM-SHA256',
Master_TLS_Version='TLSv1.2';
Query OK, 0 rows affected (0.03 sec)
```

11. Login back to ***C2Sql1*** MySQL client, verify that server-side SSL configuration is applied correctly and perform client-side SSL configuration. Verify that 'have_openssl' and 'have_ssl' fields are set to "YES", verify that 'ssl_ca', 'ssl_cert' and 'ssl_key' fields are set the correct certificate/key file path:

a. Check the variables using the following command:

```
mysql> show variables like '%ssl%';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| have_openssl  | YES                                     |
| have_ssl      | YES                                     |
| ssl_ca        | /var/occnedb/mysqlcerts/ca.pem          |
| ssl_capath    |                                         |
| ssl_cert      | /var/occnedb/mysqlcerts/server-cert.pem |
| ssl_cipher    | DHE-RSA-AES128-GCM-SHA256               |
| ssl_crl       |                                         |
| ssl_crlpath   |                                         |
| ssl_key       | /var/occnedb/mysqlcerts/server-key.pem  |
+---------------+-----------------------------------------+
```

b. If you are using same set of certificates, run the following command:

```
mysql> CHANGE MASTER TO MASTER_SSL=1, MASTER_SSL_CA
= '/var/occnedb/mysqlcerts/ca.pem', MASTER_SSL_CERT = '/var/
occnedb/mysqlcerts/client-cert.pem', MASTER_SSL_KEY = '/var/
occnedb/mysqlcerts/client-key.pem', Master_SSL_Cipher='DHE-RSA-
AES128-GCM-SHA256', Master_TLS_Version='TLSv1.2';
Query OK, 0 rows affected (0.03 sec)
```

c. If using different set of certificates, run the following command:

```
mysql> CHANGE MASTER TO MASTER_SSL=1,
MASTER_SSL_CA = '/var/occnedb/mysqlcerts/c1sql1/ca.pem',
MASTER_SSL_CERT = '/var/occnedb/mysqlcerts/c1sql1/client-
cert.pem', MASTER_SSL_KEY = '/var/occnedb/mysqlcerts/c1sql1/
client-key.pem', Master_SSL_Cipher='DHE-RSA-AES128-GCM-SHA256',
Master_TLS_Version='TLSv1.2';
Query OK, 0 rows affected (0.03 sec)
```

12. Login back to ***C2Sql2*** MySQL client, run the 'stop slave' command to force a switch-over:

```
mysql> stop slave;
Query OK, 0 rows affected (0.00 sec)
```

Verifying Geo Replication

Create a table in ***C2Sql2*** and insert a record into it, verify that the table is replicated by inspecting it on ***C1Sql1***; login to ***C1Sql2***, insert a record into this new table and check whether it is replicated on ***C2Sql2***.

1. On ***C2Sql2***, login to MySQL client, create database, create table and insert record into it:

   a. Create database using the following command:

   ```
   mysql> create database test1;
   Query OK, 1 row affected (0.06 sec)
   ```

   b. Create table in the database:

   ```
   mysql> use test1;
   Database changed
   mysql> create table table1 (id int, msg varchar(255));
   Query OK, 0 rows affected (0.46 sec)
   ```

   c. Insert records to the table:

   ```
   mysql> insert into table1 (id, msg) values (1, "hello");
   Query OK, 1 row affected (0.01 sec)
   ```

   d. View the table content using the following command:

   ```
   mysql> select * from table1;
   +------+-------+
   | id   | msg   |
   +------+-------+
   |    1 | hello |
   +------+-------+
   1 row in set (0.01 sec)
   ```

2. On ***C1Sql1***, login to MySQL client, verify that the new table is replicated:

a. View the database using the following command:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| ndbinfo            |
| performance_schema |
| replication_info   |
| sys                |
| test1              |
+--------------------+
7 rows in set (0.00 sec)

mysql> use test1;
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Database changed
```

b. Check the table using the following command:

```
mysql> show tables;
+-----------------+
| Tables_in_test1 |
+-----------------+
| table1          |
+-----------------+
1 row in set (0.00 sec)
```

c. View the table:

```
mysql> select * from table1;
+------+-------+
| id   | msg   |
+------+-------+
|    1 | hello |
+------+-------+
1 row in set (0.01 sec)
```

3. On ***C1Sql2***, perform the following steps:

a. Insert a new record into this table:

```
mysql> insert into table1 (id, msg) values (2, "hello back");
Query OK, 1 row affected (0.01 sec)
```

b. Check if the new record is added to the table:

```
mysql> select * from table1;
+------+------------+
| id   | msg        |
```

```
+------+------------+
|    2 | hello back |
|    1 | hello      |
+------+------------+
2 rows in set (0.01 sec)
```

4. On **C2Sql2**, verify that the newly inserted record is replicated:

```
mysql> select * from test1.table1;
+------+------------+
| id   | msg        |
+------+------------+
|    2 | hello back |
|    1 | hello      |
+------+------------+
2 rows in set (0.01 sec)
```

5. Verification is successful. Login to any of the SQL node, drop the test1 database:

```
mysql> drop database test1;
Query OK, 0 rows affected (0.07 sec)
```