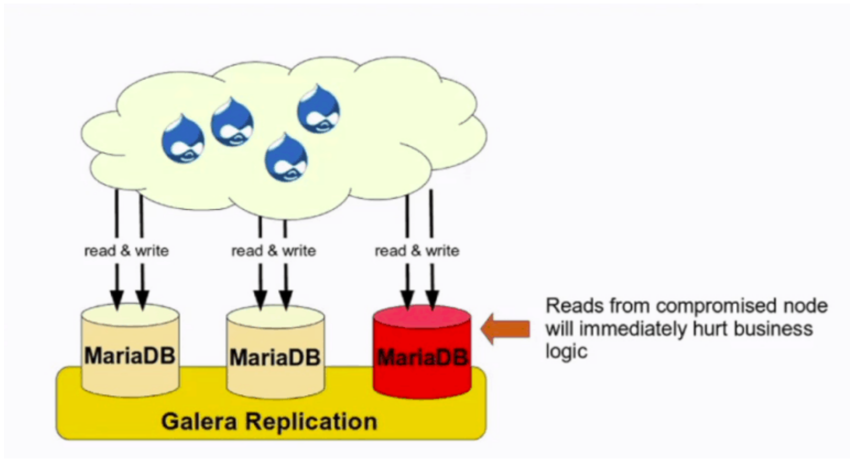
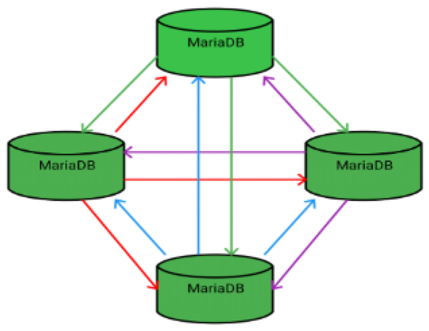
What Is MariaDB Galera Cluster?

Galera Cluster is a contemporary multi-master database cluster that is designed keeping in mind the contemporary replication, MySQL (or MariaDB), and InnoDB. It is used to direct read and write databases to any node. An individual node can be lost if the operation is uninterrupted and when the complex failover procedures are not being used.
How Does MariaDB Galera Cluster Work?

Galera Cluster is only available on Linux and supports the InnoDB storage engine. It consists of the database server that uses the Galera Replication Plugins to direct replication efficiently. MariaDB replication plugin API is extended to deliver all the information and secure necessary for true multi-master, synchronous replication. This extended API is known as Write-Set Replication API (or wsrep).

This API enables MariaDB Galera Cluster to deliver certification-based replication. It consists of database rows to replicate and the information related to the entire locks held by the database during the transaction. Then, in the applier queue, each node certifies the replicated write-set against another write-set. This write-set is applied to the conflicting locks that are not required. Now, the transaction is considered committed. Later, each of these nodes continues to apply it to the tablespace





Features Of MariaDB Galera Cluster

- It is virtually synchronous replication.
- It can read and write to any cluster node.
- It consists of active multi-primary topology.
- It controls the membership automatically and drops the failed nodes from the cluster.
- It joins the node automatically.
- It connects to the client directly by providing the look and feel of native MariaDB.
- It delivers true parallel replication on row level.
- It provides better performance for all the workload.
- Users need not require VIP or master-slave operation if they use MariaDB.
- No downtime is available in terms of failure or intentionally taking down of nodes due to less availability of failover.
- Users can avoid database backup manually.

Benefits

The above features yield several benefits for a DBMS clustering solution, including:

- No replica lag
- No lost transactions
- Read scalability
- Smaller client latencies

MariaDB Clustering Setup:

…………………………………
node1 :
  pub : 13.235.244.29
  prv : 172.31.36.160
…………………………………
node2 :
  pub : 13.233.223.33
  prv : 172.31.42.241
…………………………………
node3 :
  pub : 43.205.136.47
  prv : 172.31.46.119
…………………………………

Installing MariaDB on All Nodes

Prior to MariaDB 10.1, the Galera cluster feature is bundled into MariaDB. So you will need to install the MariaDB server package on all three nodes. You can install it using the following command:

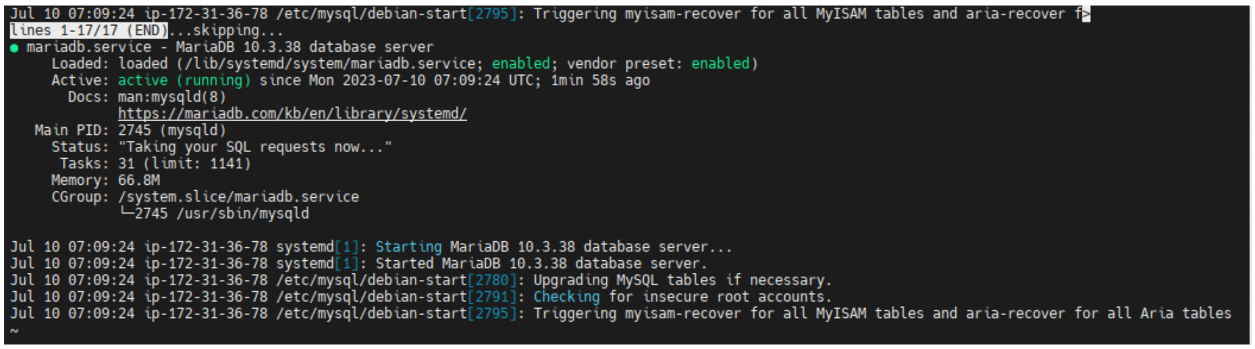
```
apt-get install mariadb-server -y
```

Once the MariaDB has been installed, start the MariaDB service on all nodes:

```
systemctl start mariadb
```

You can now check the status of the MariaDB with the following command:

```
systemctl status mariadb
```



```
Jul 10 07:09:24 ip-172-31-36-78 /etc/mysql/debian-start[2795]: Triggering myisam-recover for all MyISAM tables and aria-recover f>
lines 1-17/17 (END)...skipping...
● mariadb.service - MariaDB 10.3.38 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-07-10 07:09:24 UTC; 1min 58s ago
       Docs: man:mysqld(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 2745 (mysqld)
     Status: "Taking your SQL requests now..."
      Tasks: 31 (limit: 1141)
     Memory: 66.8M
     CGroup: /system.slice/mariadb.service
             └─2745 /usr/sbin/mysqld

Jul 10 07:09:24 ip-172-31-36-78 systemd[1]: Starting MariaDB 10.3.38 database server...
Jul 10 07:09:24 ip-172-31-36-78 systemd[1]: Started MariaDB 10.3.38 database server.
Jul 10 07:09:24 ip-172-31-36-78 /etc/mysql/debian-start[2780]: Upgrading MySQL tables if necessary.
Jul 10 07:09:24 ip-172-31-36-78 /etc/mysql/debian-start[2791]: Checking for insecure root accounts.
Jul 10 07:09:24 ip-172-31-36-78 /etc/mysql/debian-start[2795]: Triggering myisam-recover for all MyISAM tables and aria-recover for all Aria tables
~
```

You should see the above output on all nodes.

Next, you will need to secure the MariaDB installation and set a MariaDB root password on all nodes. You can do it by running the following script:

```
mysql_secure_installation
```

Answer all the questions as shown below:

```
Enter current password for root (enter for none):
Switch to unix_socket authentication [Y/n] n
Change the root password? [Y/n] Y
New password:
Re-enter new password:
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

Configuring Galera Cluster:

Next, you will need to create a MariaDB Galera configuration file on each node. You can create a file in the **/etc/mysql/conf.d** directory.

Configuring the First Node

Login on each node and create a MariaDB Galera configuration file:

```
nano /etc/mysql/conf.d/galera.cnf
```

Add the following lines :

Note: Remember to Replace ip and `wsrep_node_name`

Sample lines , you can refer this one , so that you will have the idea,

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://ip-node-1,ip-node-2,ip-node-3"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="ip-node-1 or ip-node-2 or ip-node-3" #replace with node ip
wsrep_node_name="mariadb-node-name"
```

For node 1:

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://172.31.36.160,172.31.42.241,172.31.46.119"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="172.31.36.160"
wsrep_node_name="mariadb01"
```

For node 2:

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://172.31.36.160,172.31.42.241,172.31.46.119"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="172.31.42.241"
wsrep_node_name="mariadb02"
```

For node 3:

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://172.31.36.160,172.31.42.241,172.31.46.119"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="172.31.46.119"
wsrep_node_name="mariadb03"
```

Save and close the file when you are finished.

Starting the Galera Cluster

At this point, all nodes are configured. You are now ready to bring up the cluster.

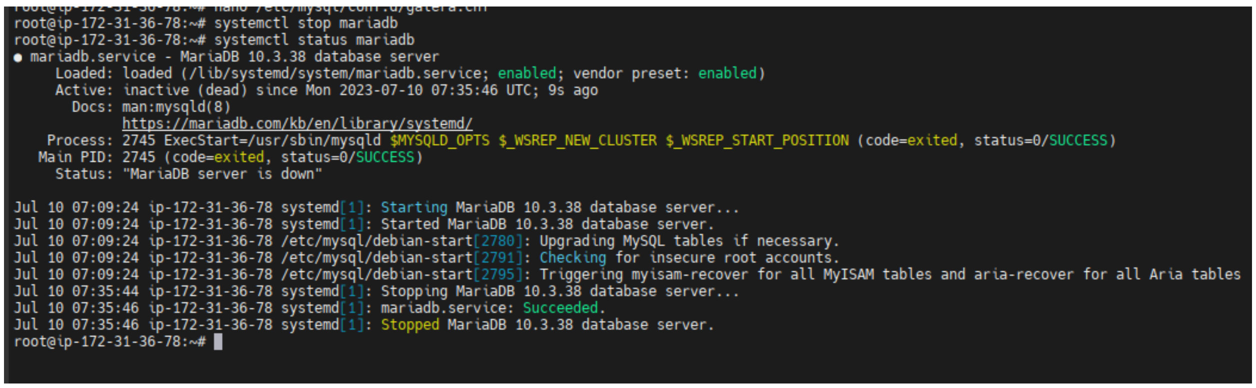
Stop MariaDB on All Nodes

First, run the following command on all nodes to stop the MariaDB service:

```
systemctl stop mariadb
```

Next, make sure MariaDB service is stopped:

```
systemctl status mariadb
```



```
root@ip-172-31-36-78:~# systemctl stop mariadb
root@ip-172-31-36-78:~# systemctl status mariadb
● mariadb.service - MariaDB 10.3.38 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
     Active: inactive (dead) since Mon 2023-07-10 07:35:46 UTC; 9s ago
       Docs: man:mysqld(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 2745 ExecStart=/usr/sbin/mysqld $MYSQLD_OPTS $_WSREP_NEW_CLUSTER $_WSREP_START_POSITION (code=exited, status=0/SUCCESS)
   Main PID: 2745 (code=exited, status=0/SUCCESS)
     Status: "MariaDB server is down"

Jul 10 07:09:24 ip-172-31-36-78 systemd[1]: Starting MariaDB 10.3.38 database server...
Jul 10 07:09:24 ip-172-31-36-78 systemd[1]: Started MariaDB 10.3.38 database server.
Jul 10 07:09:24 ip-172-31-36-78 /etc/mysql/debian-start[2780]: Upgrading MySQL tables if necessary.
Jul 10 07:09:24 ip-172-31-36-78 /etc/mysql/debian-start[2791]: Checking for insecure root accounts.
Jul 10 07:09:24 ip-172-31-36-78 /etc/mysql/debian-start[2795]: Triggering myisam-recover for all MyISAM tables and aria-recover for all Aria tables
Jul 10 07:35:44 ip-172-31-36-78 systemd[1]: Stopping MariaDB 10.3.38 database server...
Jul 10 07:35:46 ip-172-31-36-78 systemd[1]: mariadb.service: Succeeded.
Jul 10 07:35:46 ip-172-31-36-78 systemd[1]: Stopped MariaDB 10.3.38 database server.
root@ip-172-31-36-78:~#
```

Start the First Node

Now, go to the first node and use the galera_new_cluster script to start the first node.

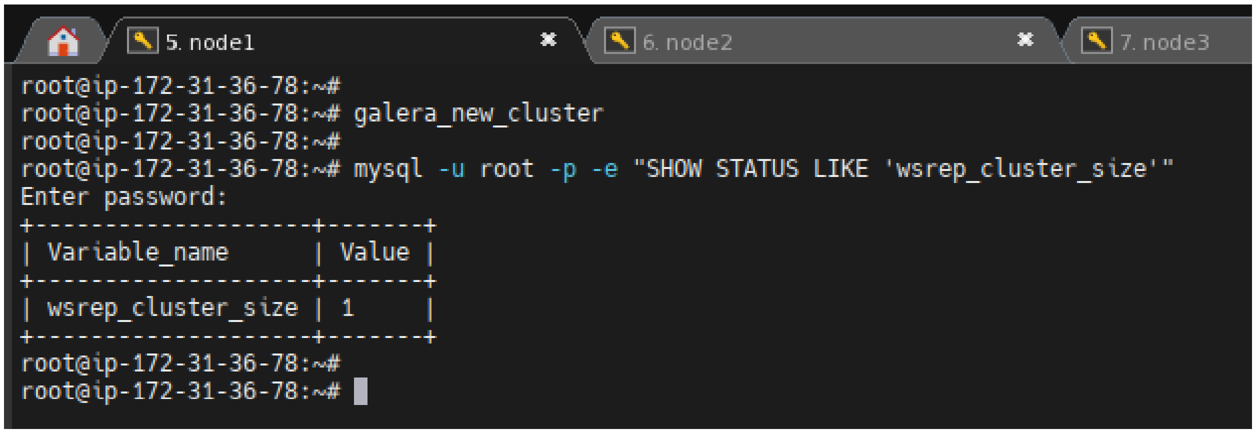
```
galera_new_cluster
```

You should not get any output if the script is executed successfully.

Next, verify the cluster status using the following command:

```
mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
```

You should get the following output:



```
root@ip-172-31-36-78:~#
root@ip-172-31-36-78:~# galera_new_cluster
root@ip-172-31-36-78:~#
root@ip-172-31-36-78:~# mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
Enter password:
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 1     |
+--------------------+-------+
root@ip-172-31-36-78:~#
root@ip-172-31-36-78:~#
```

Start the Second Node & 3 rd node

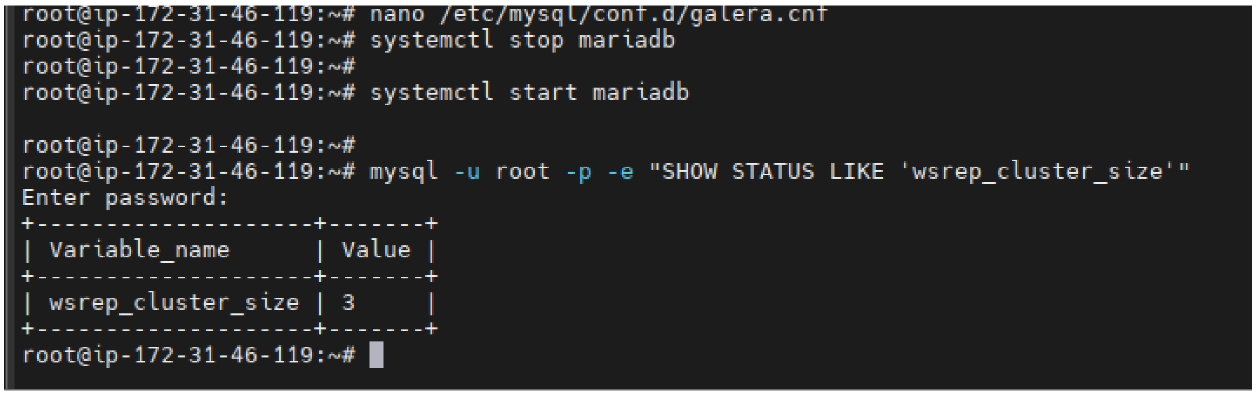
Now, go to the second node and bring up the MariaDB service with the following command:

```
systemctl start mariadb
```

Next, verify the cluster status using the following command:

```
mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
```

The following output indicates that the second node has joined the cluster :



```
root@ip-172-31-46-119:~# nano /etc/mysql/conf.d/galera.cnf
root@ip-172-31-46-119:~# systemctl stop mariadb
root@ip-172-31-46-119:~#
root@ip-172-31-46-119:~# systemctl start mariadb

root@ip-172-31-46-119:~#
root@ip-172-31-46-119:~# mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
Enter password:
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
root@ip-172-31-46-119:~#
```

Verify Replication :

At this point, your MariaDB cluster is up and running. Now, you will need to test whether the replication is working or not.

Create a Database and Table on the First Node

First, go to the first node and connect to the MariaDB using the following command:

```
mysql -u root -p
```

Once you are connected, create a database named **classdb**:

```
MariaDB [(none)]> CREATE DATABASE classdb;
```

Next, switch the database to **classdb** and create a table named **students**:

```
MariaDB [(none)]> USE classdb;
MariaDB [classdb]> CREATE TABLE students (id int, name varchar(20), surname varchar(20));
```

Next, insert some data into **students** table:

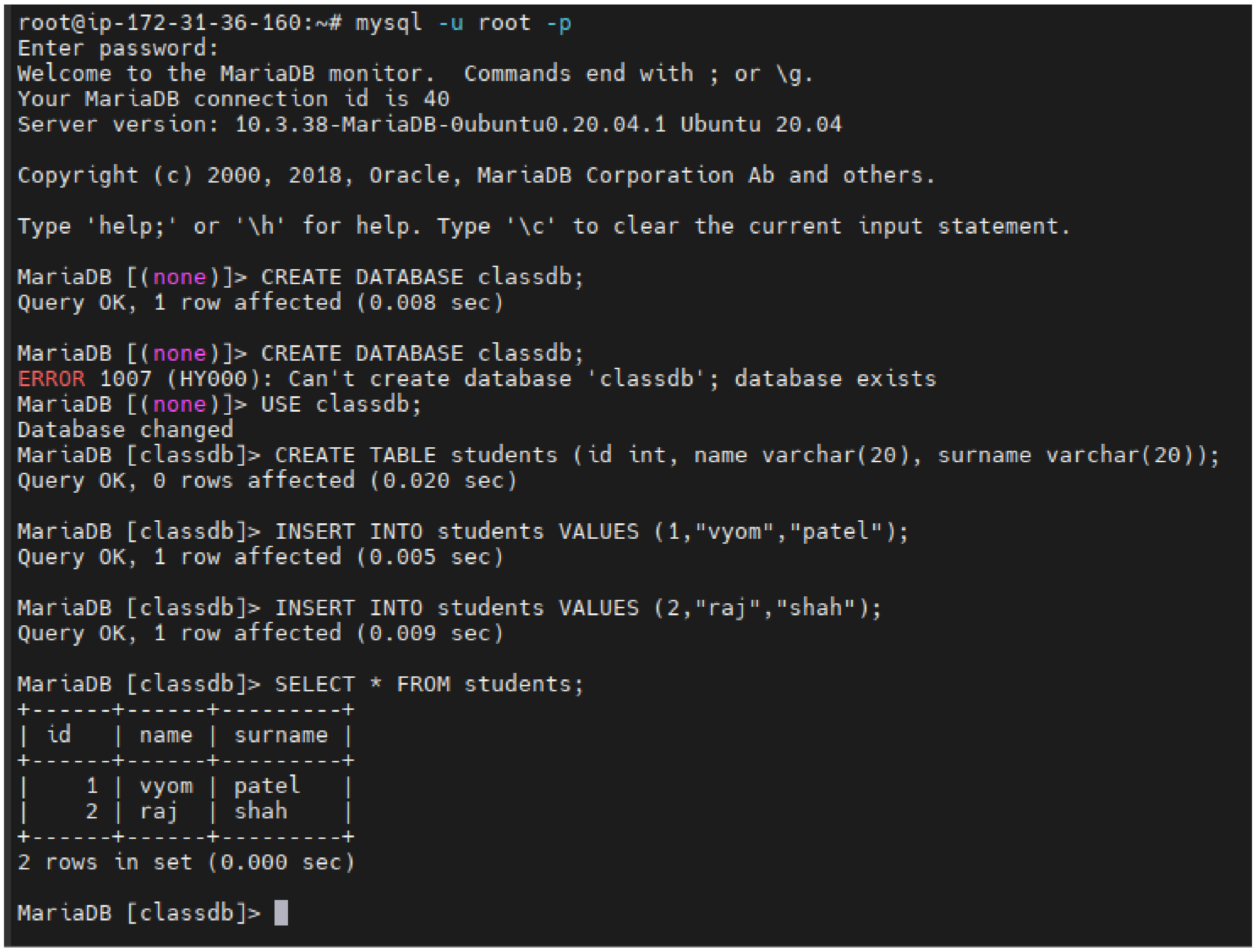
```
MariaDB [classdb]> INSERT INTO students VALUES (1,"vyom","patel");
MariaDB [classdb]> INSERT INTO students VALUES (2,"raj","shah");
```

Now, verify the inserted data with the following command:

```
MariaDB [schooldb]> SELECT * FROM students;
```

You should get the following output:

```
+------+------+---------+
| id   | name | surname |
+------+------+---------+
|    1 | vyom | patel   |
|    2 | raj  | shah    |
+------+------+---------+
```



```
root@ip-172-31-36-160:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 40
Server version: 10.3.38-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE classdb;
Query OK, 1 row affected (0.008 sec)

MariaDB [(none)]> CREATE DATABASE classdb;
ERROR 1007 (HY000): Can't create database 'classdb'; database exists
MariaDB [(none)]> USE classdb;
Database changed
MariaDB [classdb]> CREATE TABLE students (id int, name varchar(20), surname varchar(20));
Query OK, 0 rows affected (0.020 sec)

MariaDB [classdb]> INSERT INTO students VALUES (1,"vyom","patel");
Query OK, 1 row affected (0.005 sec)

MariaDB [classdb]> INSERT INTO students VALUES (2,"raj","shah");
Query OK, 1 row affected (0.009 sec)

MariaDB [classdb]> SELECT * FROM students;
+------+------+---------+
| id   | name | surname |
+------+------+---------+
|    1 | vyom | patel   |
|    2 | raj  | shah    |
+------+------+---------+
2 rows in set (0.000 sec)

MariaDB [classdb]>
```

Verify Replication on the Second and Third Node

Now, it's time to verify whether the **classdb** database is replicated or not.
First, go to the second node and connect to the MariaDB with the following command:

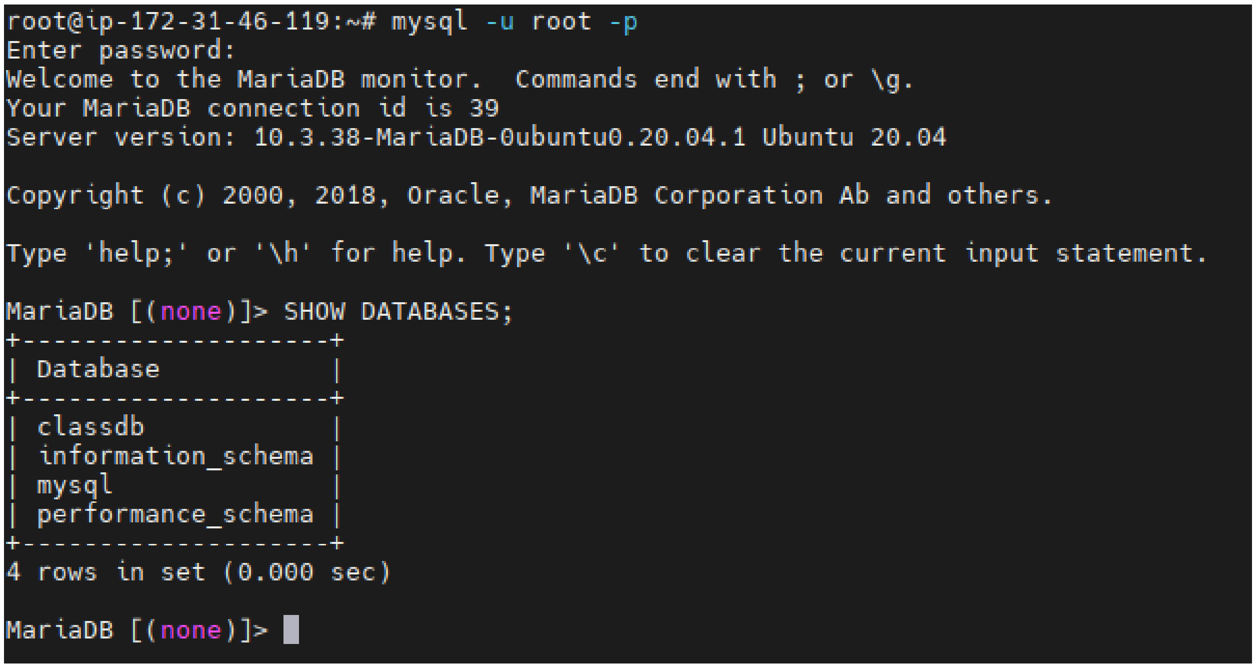
```
mysql -u root -p
```

Once you are connected, verify all the databases:

```
MariaDB [(none)]> SHOW DATABASES;
```

You should see that your classdb database is replicated to the second node:

```
+--------------------+
| Database           |
+--------------------+
| classdb            |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
```

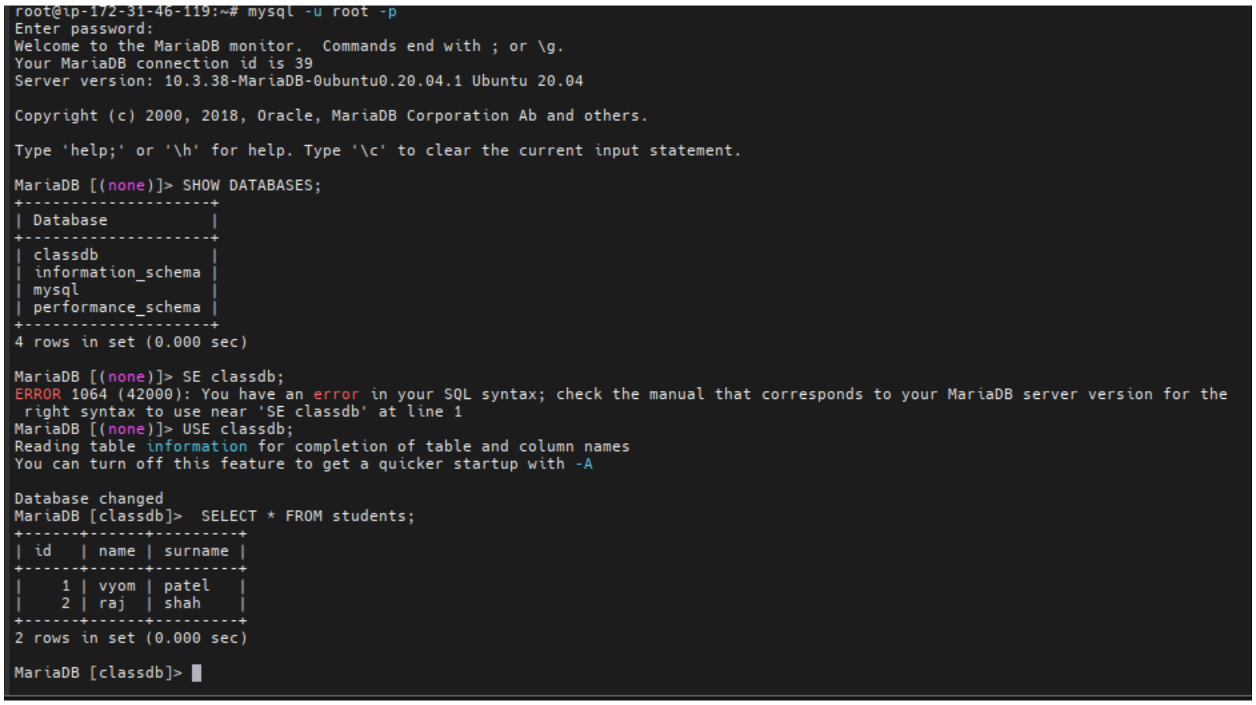


```
root@ip-172-31-46-119:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
Server version: 10.3.38-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| classdb            |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.000 sec)

MariaDB [(none)]>
```

Now, switch the database to **classdb** and verify your **students** table:

```
MariaDB [(none)]> USE classdb;
MariaDB [classdb]> SELECT * FROM students;
```

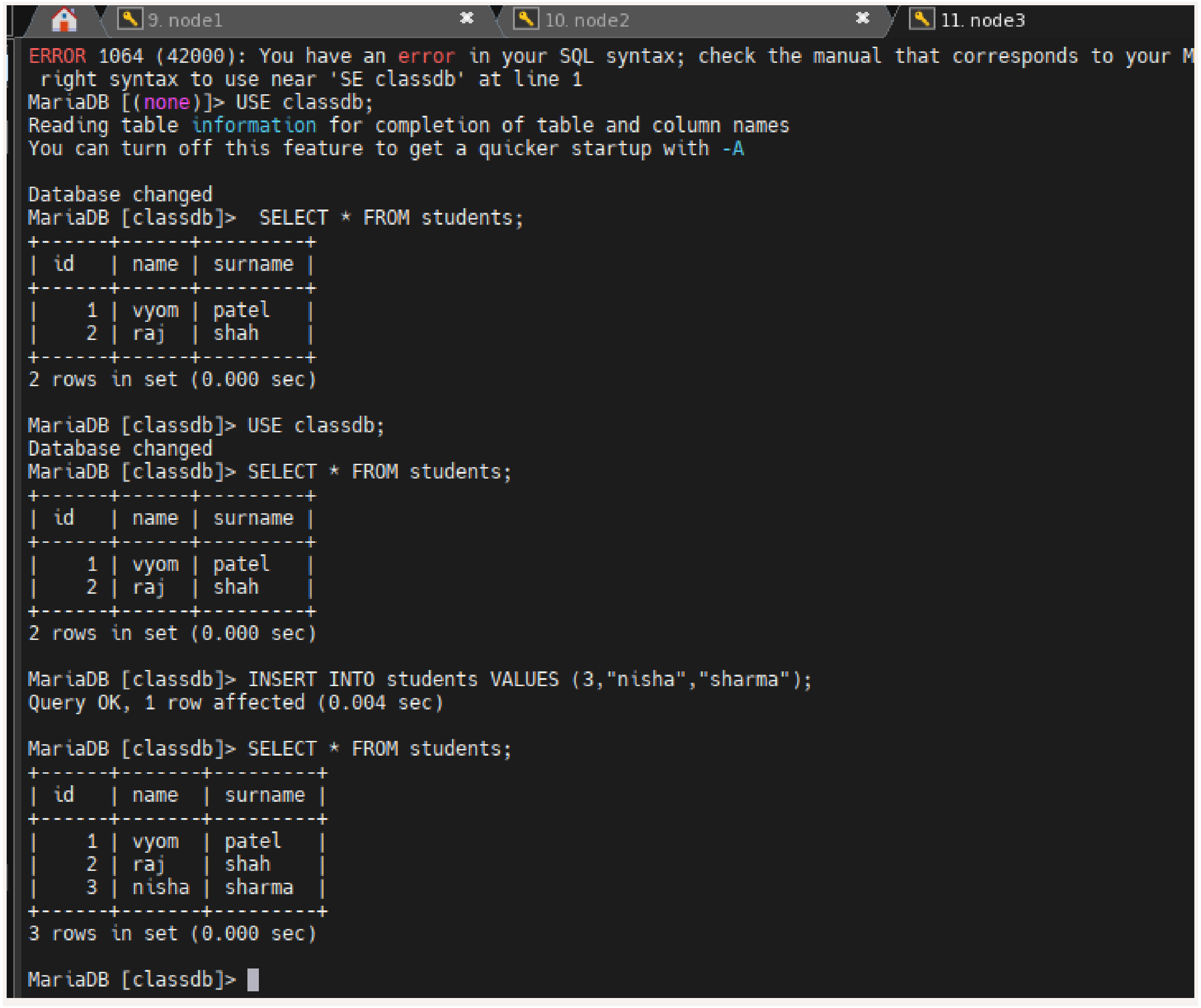
You should see the following output:



```
root@ip-172-31-46-119:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
Server version: 10.3.38-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| classdb            |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.000 sec)

MariaDB [(none)]> SE classdb;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the
 right syntax to use near 'SE classdb' at line 1
MariaDB [(none)]> USE classdb;
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
MariaDB [classdb]>  SELECT * FROM students;
+------+------+---------+
| id   | name | surname |
+------+------+---------+
|    1 | vyom | patel   |
|    2 | raj  | shah    |
+------+------+---------+
2 rows in set (0.000 sec)

MariaDB [classdb]>
```

Now, add some information in the third row:

```
MariaDB [classdb]> INSERT INTO students VALUES (3,"nisha","sharma");
```



9. node1 10. node2 11. node3

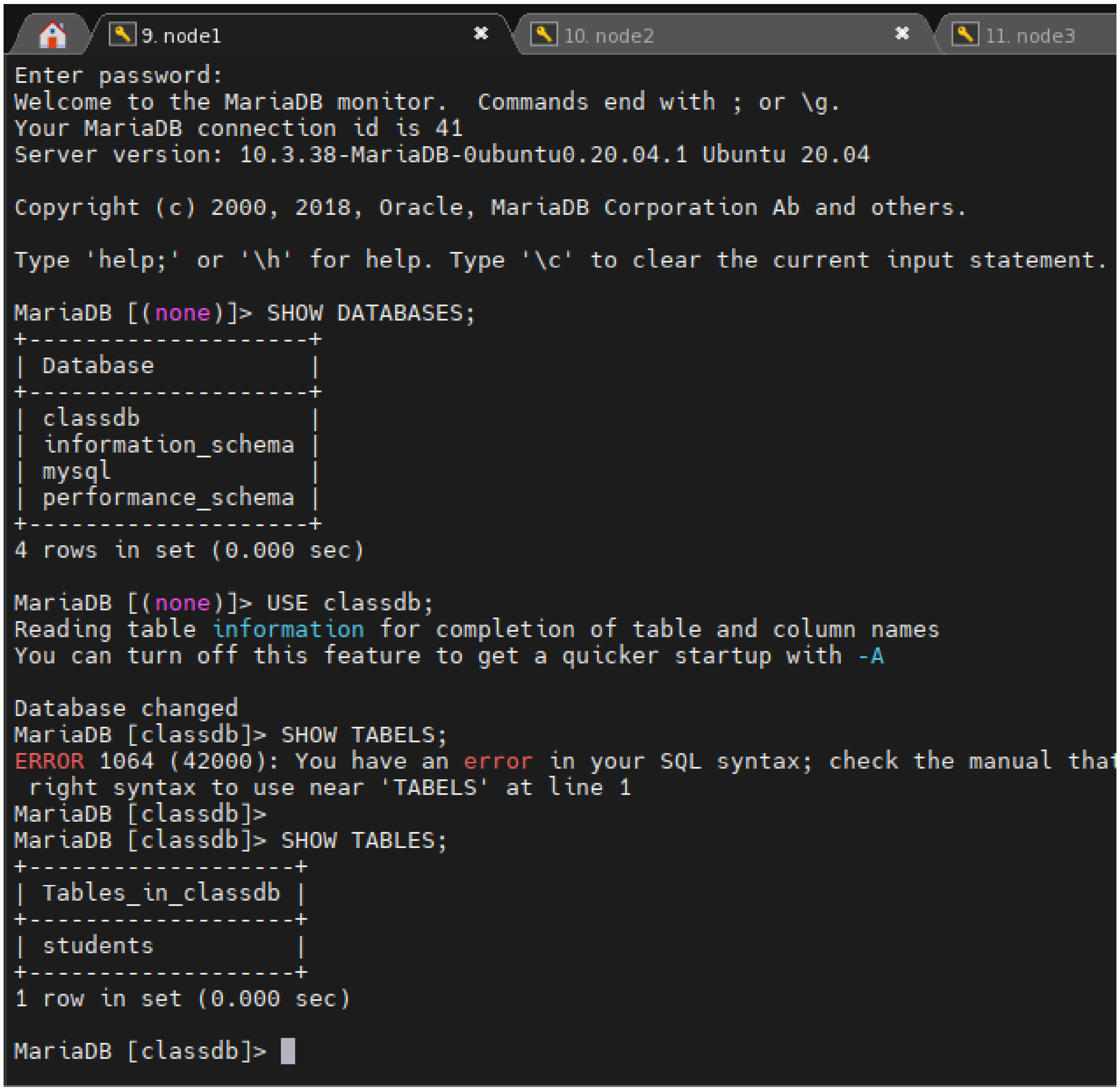
```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your M
 right syntax to use near 'SE classdb' at line 1
MariaDB [(none)]> USE classdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [classdb]>  SELECT * FROM students;
+------+------+---------+
| id   | name | surname |
+------+------+---------+
|    1 | vyom | patel   |
|    2 | raj  | shah    |
+------+------+---------+
2 rows in set (0.000 sec)

MariaDB [classdb]> USE classdb;
Database changed
MariaDB [classdb]> SELECT * FROM students;
+------+------+---------+
| id   | name | surname |
+------+------+---------+
|    1 | vyom | patel   |
|    2 | raj  | shah    |
+------+------+---------+
2 rows in set (0.000 sec)

MariaDB [classdb]> INSERT INTO students VALUES (3,"nisha","sharma");
Query OK, 1 row affected (0.004 sec)

MariaDB [classdb]> SELECT * FROM students;
+------+-------+---------+
| id   | name  | surname |
+------+-------+---------+
|    1 | vyom  | patel   |
|    2 | raj   | shah    |
|    3 | nisha | sharma  |
+------+-------+---------+
3 rows in set (0.000 sec)

MariaDB [classdb]>
```

Next, go to the first node and connect to the MariaDB with the following command:

```
mysql -u root -p
```

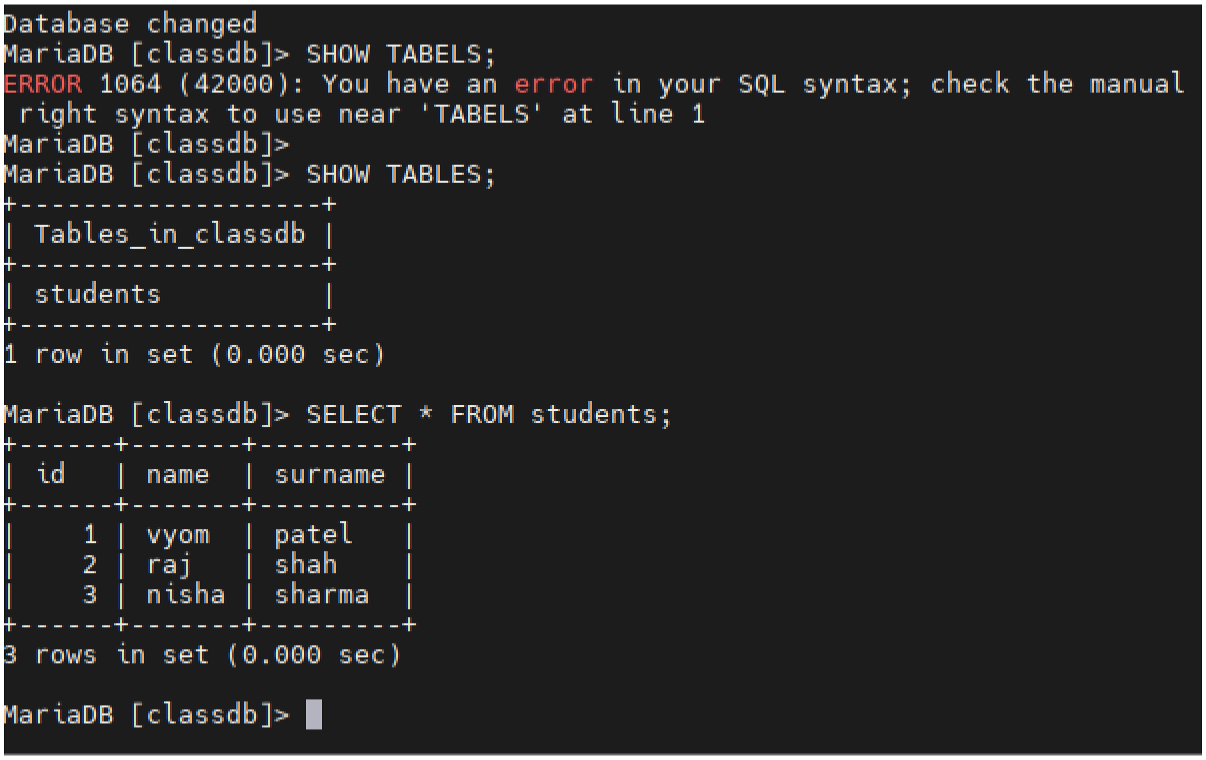
Next, switch the database to **classdb** database and verify your tables:

```
MariaDB [(none)]> USE classdb;
MariaDB [classdb]> SHOW TABLES;
```



9. node1 10. node2 11. node3

```
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 41
Server version: 10.3.38-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| classdb            |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.000 sec)

MariaDB [(none)]> USE classdb;
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
MariaDB [classdb]> SHOW TABELS;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual tha
 right syntax to use near 'TABELS' at line 1
MariaDB [classdb]>
MariaDB [classdb]> SHOW TABLES;
+-------------------+
| Tables_in_classdb |
+-------------------+
| students          |
+-------------------+
1 row in set (0.000 sec)

MariaDB [classdb]>
```



```
Database changed
MariaDB [classdb]> SHOW TABELS;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
 right syntax to use near 'TABELS' at line 1
MariaDB [classdb]>
MariaDB [classdb]> SHOW TABLES;
+-------------------+
| Tables_in_classdb |
+-------------------+
| students          |
+-------------------+
1 row in set (0.000 sec)

MariaDB [classdb]> SELECT * FROM students;
+------+-------+---------+
| id   | name  | surname |
+------+-------+---------+
|    1 | vyom  | patel   |
|    2 | raj   | shah    |
|    3 | nisha | sharma  |
+------+-------+---------+
3 rows in set (0.000 sec)

MariaDB [classdb]>
```

Conclusion

In the above guide, we explained how to set up three nodes MariaDB cluster on Ubuntu 20.04. Now, data modifications on any node are replicated to all other nodes. You can now add more MariaDB nodes to increase the size of the cluster.

Q. How many servers, you've to configure Mariadb Galera Cluster?

The minimum number of servers required to configure a Galera Cluster is three.

This is because Galera Cluster uses a quorum-based voting system, where a majority of nodes must be available in order for the cluster to be considered healthy.
If you have only two nodes, and one of them goes down, the cluster will become unavailable.

However, it is generally recommended to have at least five nodes in a Galera Cluster. This provides you with more redundancy and allows you to handle more traffic. If you have a large database or a high volume of transactions, you may need even more nodes.

Here are some of the benefits of having more nodes in a Galera Cluster:

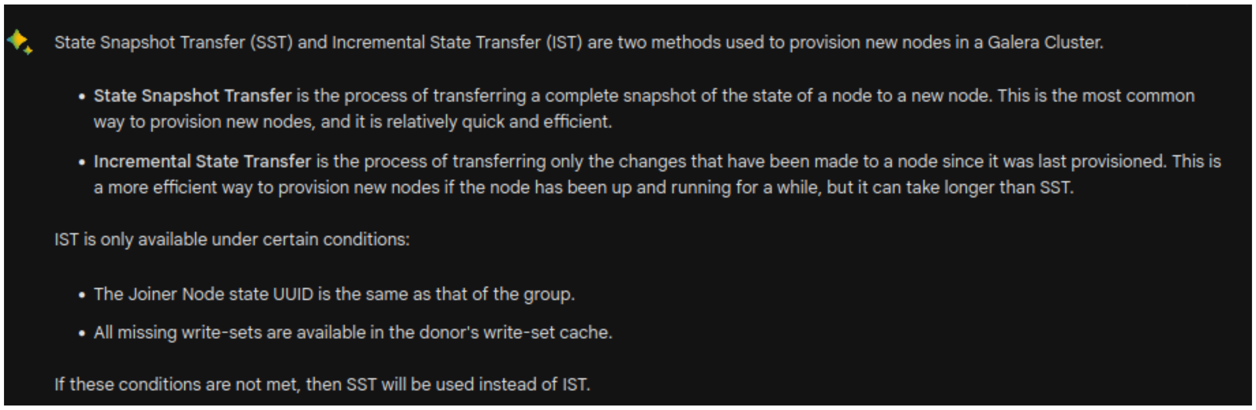
- Increased redundancy: If one node goes down, the cluster will still be able to operate with the remaining nodes.
- Increased scalability: You can add more nodes to the cluster as your traffic increases.
- Improved performance: The more nodes you have, the more traffic the cluster can handle.
- Better availability: The more nodes you have, the less likely it is that the cluster will become unavailable.

Of course, there are also some drawbacks to having more nodes in a Galera Cluster. For example, it will cost more to purchase and maintain more servers. Additionally, it can be more complex to manage a larger cluster.

Ultimately, the number of nodes you need for your Galera Cluster will depend on your specific requirements. If you are concerned about availability and scalability, then you may want to consider having more than three nodes. However, if you are on a tight budget or you do not need a lot of redundancy, then three nodes may be sufficient.

A quorum-based voting system is a method of voting where a decision is only considered valid if a majority of votes are cast in favor of it.

Q. Mariadb Ports and uses

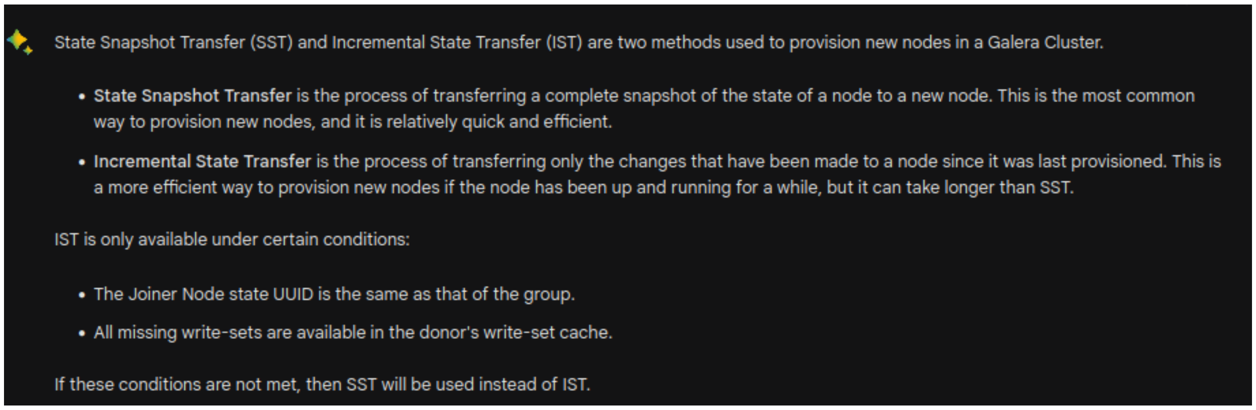

State Snapshot Transfer (SST) and Incremental State Transfer (IST) are two methods used to provision new nodes in a Galera Cluster.

- **State Snapshot Transfer** is the process of transferring a complete snapshot of the state of a node to a new node. This is the most common way to provision new nodes, and it is relatively quick and efficient.
- **Incremental State Transfer** is the process of transferring only the changes that have been made to a node since it was last provisioned. This is a more efficient way to provision new nodes if the node has been up and running for a while, but it can take longer than SST.

IST is only available under certain conditions:

- The Joiner Node state UUID is the same as that of the group.
- All missing write-sets are available in the donor's write-set cache.

If these conditions are not met, then SST will be used instead of IST.


State Snapshot Transfer (SST) and Incremental State Transfer (IST) are two methods used to provision new nodes in a Galera Cluster.

- **State Snapshot Transfer** is the process of transferring a complete snapshot of the state of a node to a new node. This is the most common way to provision new nodes, and it is relatively quick and efficient.
- **Incremental State Transfer** is the process of transferring only the changes that have been made to a node since it was last provisioned. This is a more efficient way to provision new nodes if the node has been up and running for a while, but it can take longer than SST.

IST is only available under certain conditions:

- The Joiner Node state UUID is the same as that of the group.
- All missing write-sets are available in the donor's write-set cache.

If these conditions are not met, then SST will be used instead of IST.