WEBINAR SERIES

Part 2 – 23 July 2025

Achieving High Availability (HA) and Security of Modern applications using MariaDB Enterprise

Kanthimathi Kailasanathan (Kanthi)
Senior Solutions Engineer
MariaDB Corporation

About Speaker

Kanthi

Senior Solutions Engineer

✉ kanthimathi.kailasanathan@mariadb.com

in https://www.linkedin.com/in/kanthimathi-kailasanathan/

Celebrating 15 Years of MariaDB !!

The first version of MariaDB, 5.1.38, was released on 29th of October 2009 !

We have come a long way since then!

More information at

https://monty-says.blogspot.com/2024/10/celebrating-15-years-of-mariadb.html

MariaDB

About MariaDB

Created by the original developers of MySQL, MariaDB provides a powerful, open-source core database for enterprises. Now the default in the majority of Linux distributions, it gives businesses the strategic freedom to break from proprietary databases and build modern, scalable applications for the future.

Market Leadership

75%
Of Fortune 500 companies use MariaDB

1B+
Docker Hub downloads

2.5B+
Reach via Linux distros

200K+
Open source contributions

700+ Customers Globally

Amdocs
Deutsche Bank
Development Bank of Singapore (DBS)
Nokia
Samsung
SelectQuote
ServiceNow
Virgin Media O2

200+ Employees

Proven leadership team

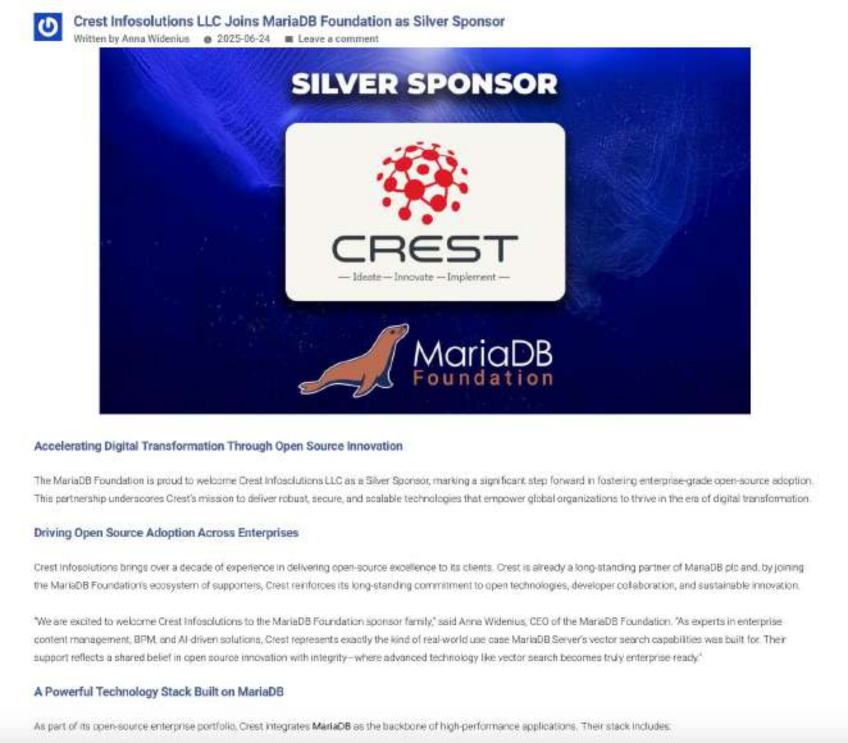
World class relational database engineering team, including the original core MySQL team

Dual headquartered

- Europe: Dublin, Ireland
- USA: Silicon Valley, California

Introducing Crest Infosolutions

- ✔ Founded in 2012 in Singapore
- ✔ Serving customers globally with presence in Singapore, Malaysia, Indonesia, USA and Netherlands
- ✔ **MariaDB distributor** and partner since 2015.
- ✔ Strong MariaDB consulting team with experience in setting-up and securing MariaDB at scale.
- ✔ **Migration team to support** customers in their database migration journey from Oracle, MS SQL, MySQL or PostgreSQL to MariaDB.



Crest Infosolutions LLC Joins MariaDB Foundation as Silver Sponsor

Written by Anna Widenius 2025-06-24 Leave a comment

Accelerating Digital Transformation Through Open Source Innovation

The MariaDB Foundation is proud to welcome Crest Infosolutions LLC as a Silver Sponsor, marking a significant step forward in fostering enterprise-grade open-source adoption. This partnership underscores Crest's mission to deliver robust, secure, and scalable technologies that empower global organizations to thrive in the era of digital transformation.

Driving Open Source Adoption Across Enterprises

Crest Infosolutions brings over a decade of experience in delivering open-source excellence to its clients. Crest is already a long-standing partner of MariaDB plc and, by joining the MariaDB Foundation's ecosystem of supporters, Crest reinforces its long-standing commitment to open technologies, developer collaboration, and sustainable innovation.

"We are excited to welcome Crest Infosolutions to the MariaDB Foundation sponsor family," said Anna Widenius, CEO of the MariaDB Foundation. "As experts in enterprise content management, BPM, and AI-driven solutions, Crest represents exactly the kind of real-world use case MariaDB Server's vector search capabilities was built for. Their support reflects a shared belief in open source innovation with integrity—where advanced technology like vector search becomes truly enterprise-ready."

A Powerful Technology Stack Built on MariaDB

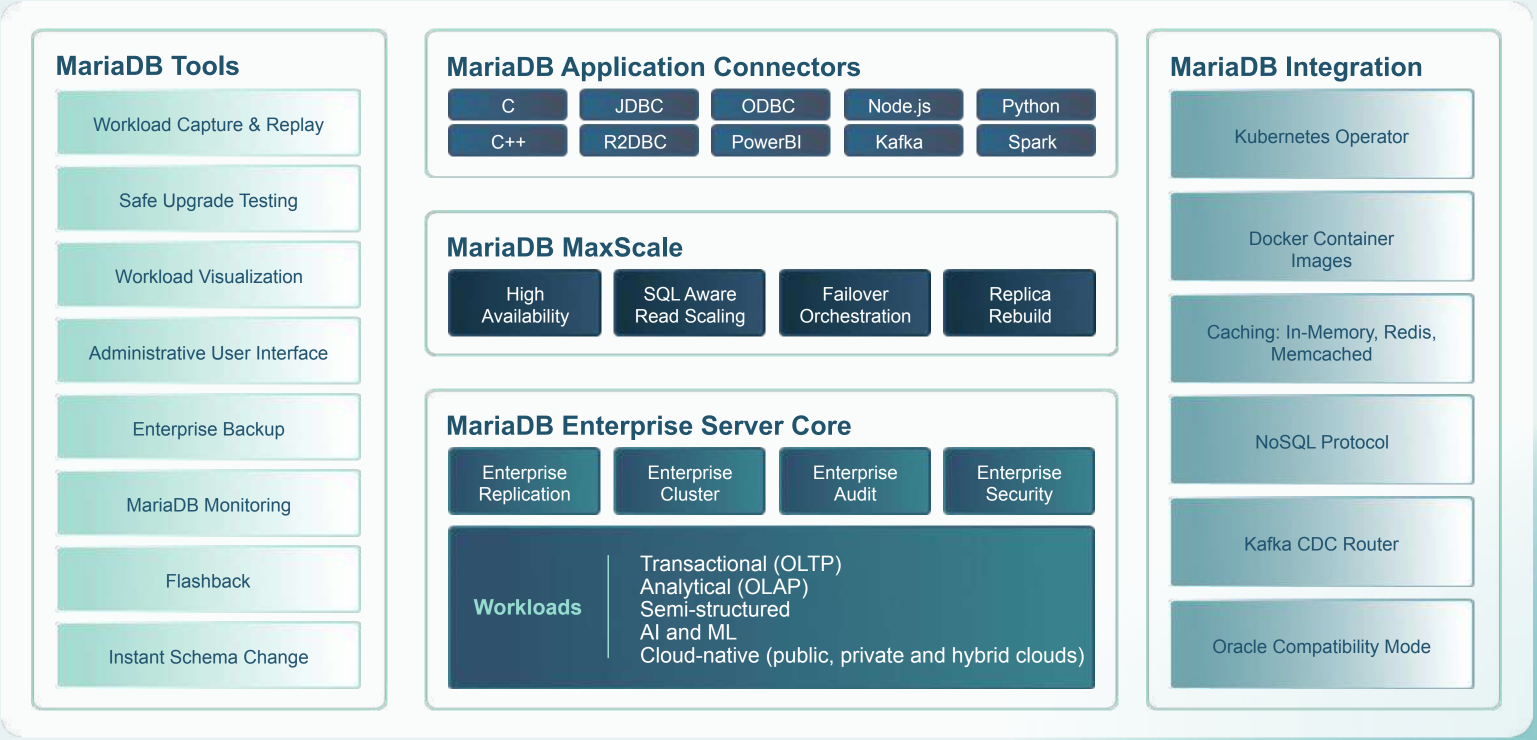
As part of its open-source enterprise portfolio, Crest integrates **MariaDB** as the backbone of high-performance applications. Their stack includes:

MariaDB

Enterprise Platform

The Complete Enterprise-Grade Platform

Delivering Performance, Scalability, Availability, and Security



MariaDB Tools

- Workload Capture & Replay
- Safe Upgrade Testing
- Workload Visualization
- Administrative User Interface
- Enterprise Backup
- MariaDB Monitoring
- Flashback
- Instant Schema Change

MariaDB Application Connectors

- C
- JDBC
- ODBC
- Node.js
- Python
- C++
- R2DBC
- PowerBI
- Kafka
- Spark

MariaDB MaxScale

- High Availability
- SQL Aware Read Scaling
- Failover Orchestration
- Replica Rebuild

MariaDB Enterprise Server Core

- Enterprise Replication
- Enterprise Cluster
- Enterprise Audit
- Enterprise Security

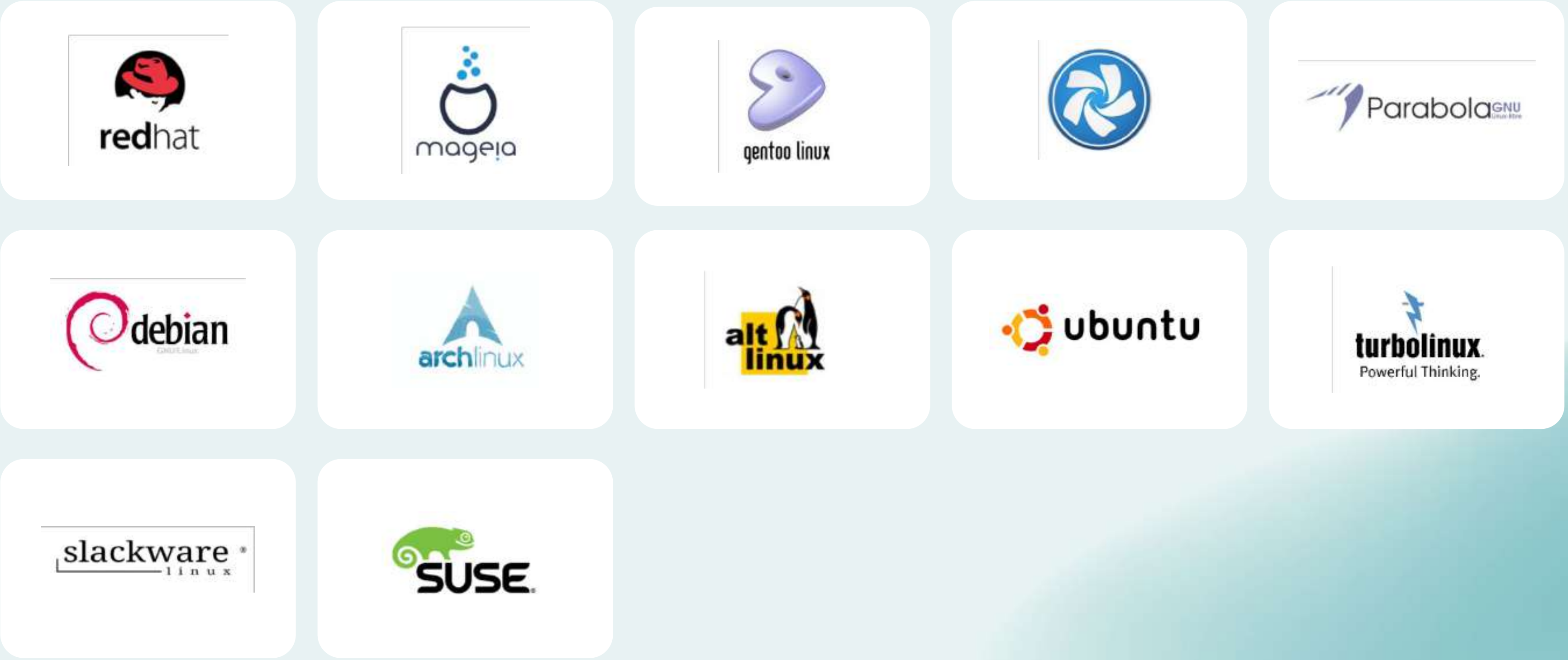
Workloads

- Transactional (OLTP)
- Analytical (OLAP)
- Semi-structured
- AI and ML
- Cloud-native (public, private and hybrid clouds)

MariaDB Integration

- Kubernetes Operator
- Docker Container Images
- Caching: In-Memory, Redis, Memcached
- NoSQL Protocol
- Kafka CDC Router
- Oracle Compatibility Mode

MariaDB is the default over MySQL on Linux distributions



75% of the Fortune 500 uses MariaDB

amdocs

InfoArmor
an Allstate company

CommunityChoice
FINANCIAL

pixid

servicenow

SELECTQUOTE

DBS

digicert

bandwidth

Deutsche Bank

auto europe

Virgin media O2

SAMSUNG

IHME

cardstream

OPPENHEIMER

tock

ingenico

NOKIA

Copart

MariaDB Enterprise Platform is your complete database solution

Enterprise-Grade

End-To-End

Open-Source

Enterprise Features

Enhanced Data Security

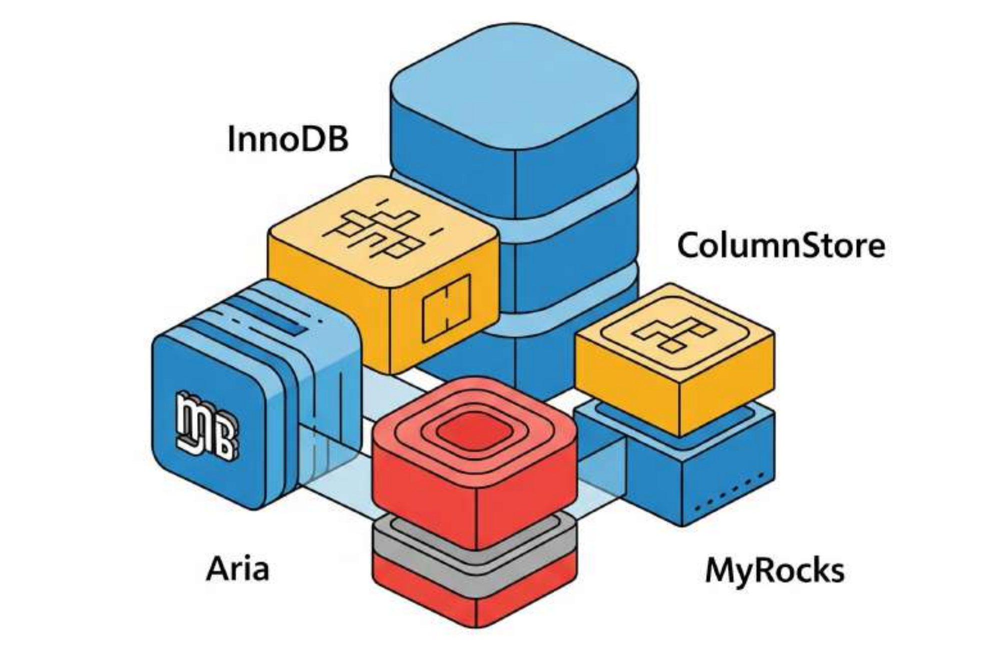
End to end Encryption

High Availability

Improved Auditing

Pluggable Storage Engines

- MariaDB supports multiple storage engines for diverse needs
- InnoDB provides ACID compliance and data integrity features
- ColumnStore offers columnar storage for analytical workloads
- MyRocks engine is designed for write-intensive applications
- Aria engine is ideal for temporary tables and internal processes



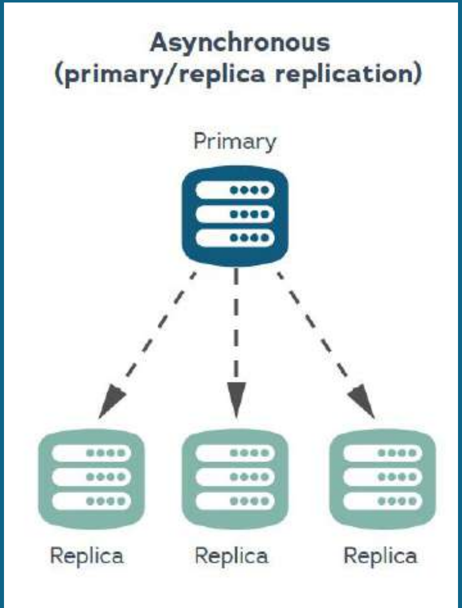
MaxScale

Intelligent Load Balancing

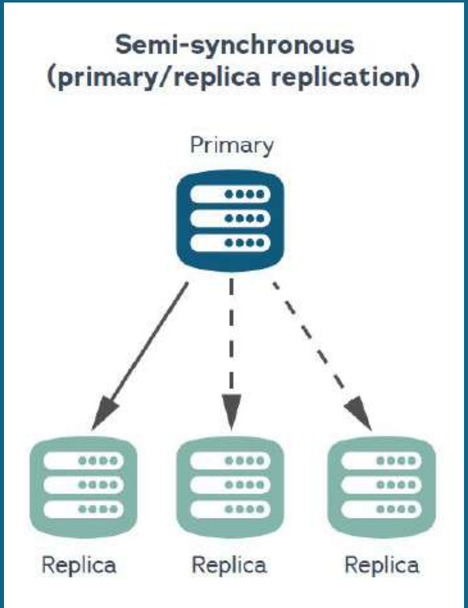
Comprehensive Compliance Needs

Database Ease of Use

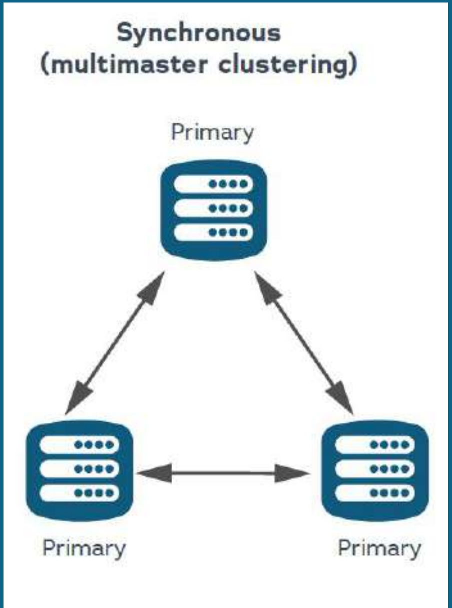
High Availability and Data Replication Options



Asynchronous (primary/replica replication)

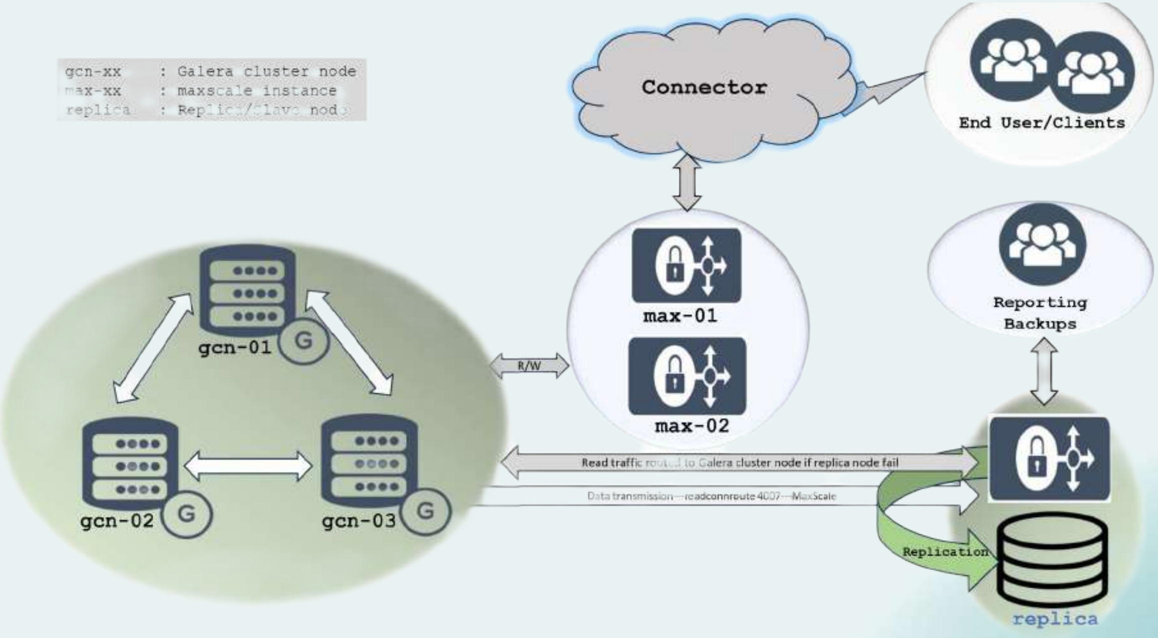


Semi-synchronous (primary/replica replication)



Synchronous (multimaster clustering)

Synchronous and Asynchronous Replication using Galera

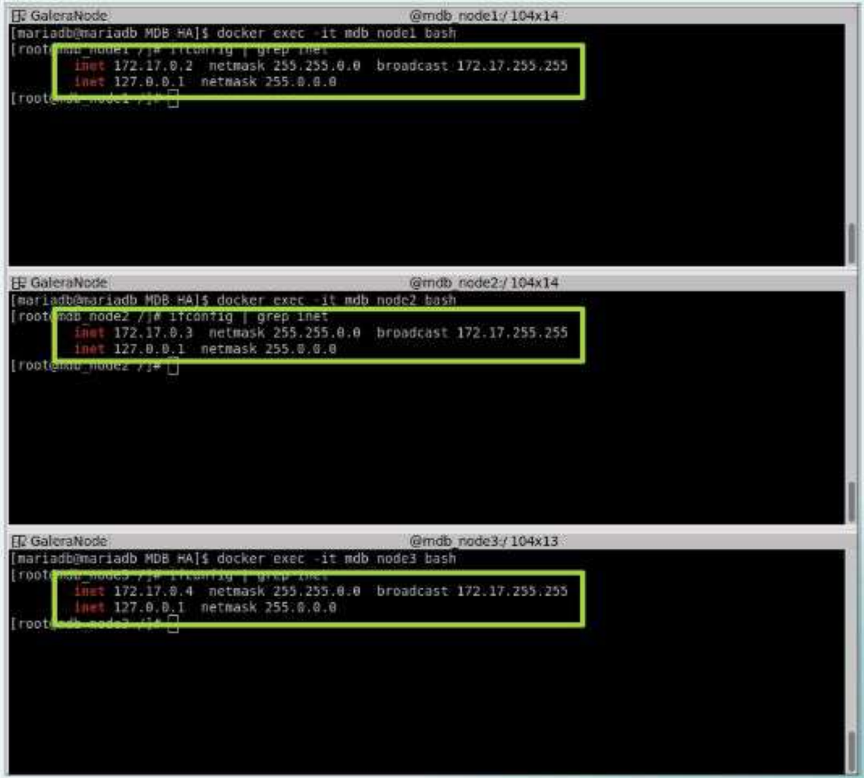

gcn-xx : Galera cluster node
max-xx : maxscale instance
replica : Replica/Slave node
Connector
End User/Clients
gcn-01
gcn-02
gcn-03
G
R/W
max-01
max-02
Reporting
Backups
Read traffic routed to Galera cluster node if replica node fail
Data transmission---readconnroute 4007---MaxScale
Replication
replica

Creating a MariaDB Galera Cluster

Identify the IP for Each Container

The IP will be used to configure the MariaDB Cluster

$ ifconfig | grep inet



```
[mariadb@mariadb MDB_HA]$ docker exec -it mdb_node1 bash
[root@mdb_node1 /]# ifconfig | grep inet
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        inet 127.0.0.1  netmask 255.0.0.0
[root@mdb_node1 /]#
```

```
[mariadb@mariadb MDB_HA]$ docker exec -it mdb_node2 bash
[root@mdb_node2 /]# ifconfig | grep inet
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
        inet 127.0.0.1  netmask 255.0.0.0
[root@mdb_node2 /]#
```

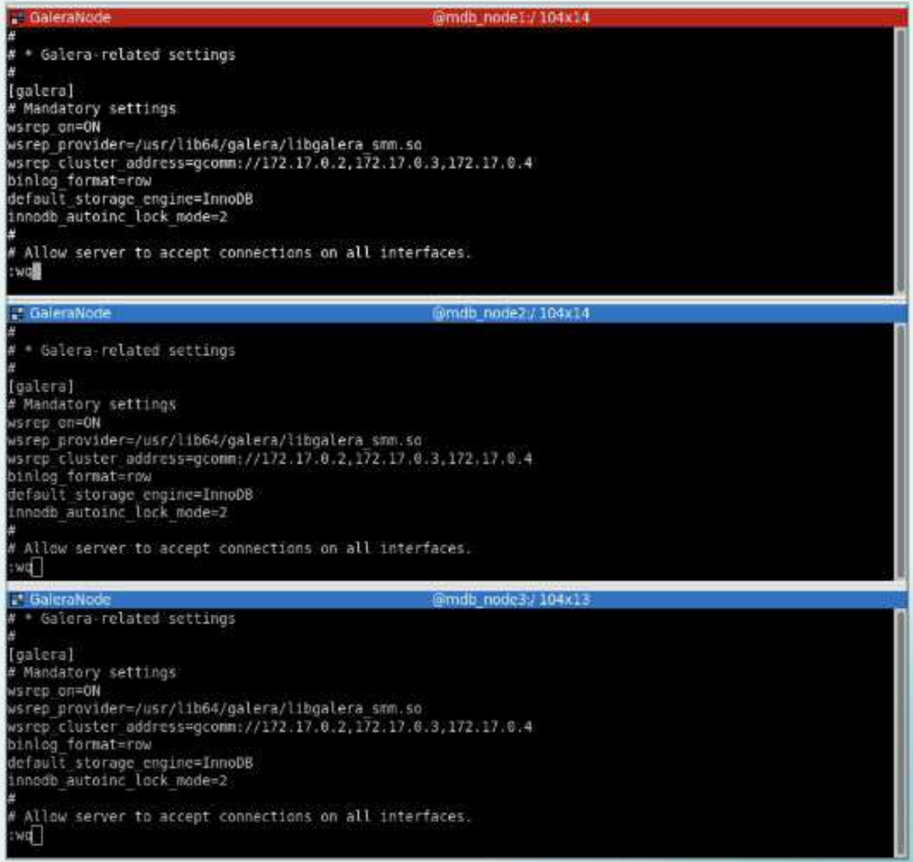
```
[mariadb@mariadb MDB_HA]$ docker exec -it mdb_node3 bash
[root@mdb_node3 /]# ifconfig | grep inet
        inet 172.17.0.4  netmask 255.255.0.0  broadcast 172.17.255.255
        inet 127.0.0.1  netmask 255.0.0.0
[root@mdb_node3 /]#
```

Configure the MariaDB Cluster

Using an editor, make the following changes to the /etc/my.cnf.d/server.cnf file:

wsrep_on
wsrep_provider
wsrep_cluster_address
binlog_format
default_storage_engine
innodb_autoinc_lock_mode

Note: use the IP addresses obtained in the previous step to set the wsrep_cluster_address parameter.



```
#
# * Galera-related settings
#
[galera]
# Mandatory settings
wsrep_on=ON
wsrep_provider=/usr/lib64/galera/libgalera_smm.so
wsrep_cluster_address=gcomm://172.17.0.2,172.17.0.3,172.17.0.4
binlog_format=row
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
#
# Allow server to accept connections on all interfaces.
:wq
```

```
#
# * Galera-related settings
#
[galera]
# Mandatory settings
wsrep_on=ON
wsrep_provider=/usr/lib64/galera/libgalera_smm.so
wsrep_cluster_address=gcomm://172.17.0.2,172.17.0.3,172.17.0.4
binlog_format=row
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
#
# Allow server to accept connections on all interfaces.
:wq
```

```
# * Galera-related settings
#
[galera]
# Mandatory settings
wsrep_on=ON
wsrep_provider=/usr/lib64/galera/libgalera_smm.so
wsrep_cluster_address=gcomm://172.17.0.2,172.17.0.3,172.17.0.4
binlog_format=row
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
#
# Allow server to accept connections on all interfaces.
:wq
```

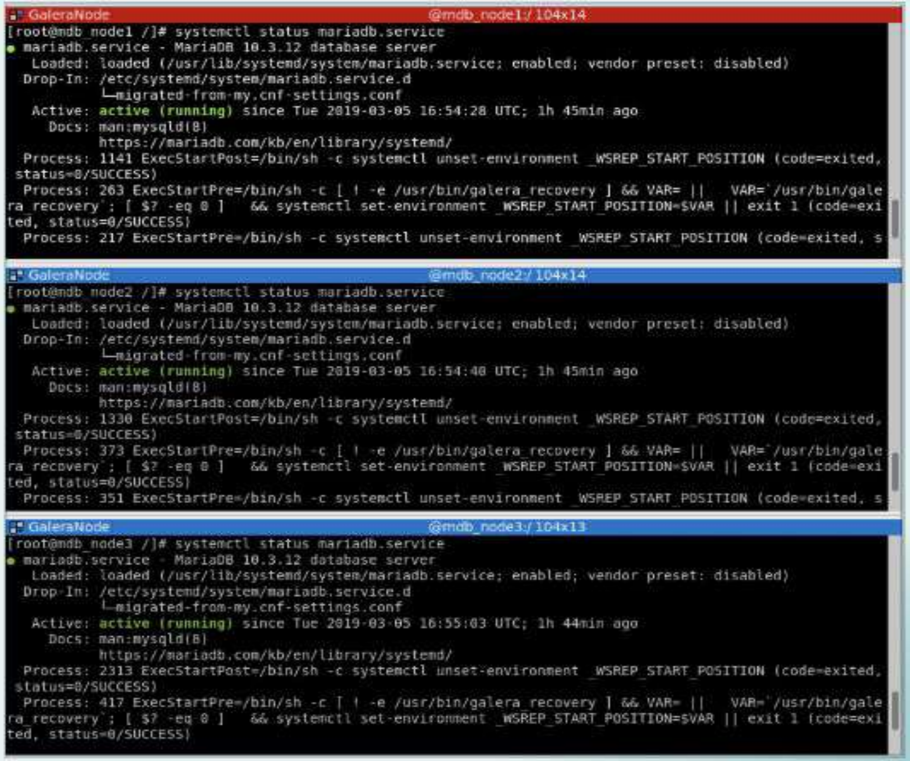
Before Starting the Cluster

For each node in the cluster, check if the MariaDB server is running:

$ systemctrl status mariadb.service.

If there are instances running, stop them before starting the cluster:

$ systemctl stop mariadb.service



```
[root@mdb node1 /]# systemctl status mariadb.service
● mariadb.service - MariaDB 10.3.12 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/mariadb.service.d
           └─migrated-from-my.cnf-settings.conf
   Active: active (running) since Tue 2019-03-05 16:54:28 UTC; 1h 45min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 1141 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited,
 status=0/SUCCESS)
  Process: 263 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= ||   VAR=`/usr/bin/gale
ra_recovery`; [ $? -eq 0 ]   && systemctl set-environment _WSREP_START_POSITION=$VAR || exit 1 (code=exi
ted, status=0/SUCCESS)
  Process: 217 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, s
```

```
[root@mdb node2 /]# systemctl status mariadb.service
● mariadb.service - MariaDB 10.3.12 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/mariadb.service.d
           └─migrated-from-my.cnf-settings.conf
   Active: active (running) since Tue 2019-03-05 16:54:40 UTC; 1h 45min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 1330 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited,
 status=0/SUCCESS)
  Process: 373 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= ||   VAR=`/usr/bin/gale
ra_recovery`; [ $? -eq 0 ]   && systemctl set-environment _WSREP_START_POSITION=$VAR || exit 1 (code=exi
ted, status=0/SUCCESS)
  Process: 351 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, s
```

```
[root@mdb node3 /]# systemctl status mariadb.service
● mariadb.service - MariaDB 10.3.12 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/mariadb.service.d
           └─migrated-from-my.cnf-settings.conf
   Active: active (running) since Tue 2019-03-05 16:55:03 UTC; 1h 44min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 2313 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited,
 status=0/SUCCESS)
  Process: 417 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= ||   VAR=`/usr/bin/gale
ra_recovery`; [ $? -eq 0 ]   && systemctl set-environment _WSREP_START_POSITION=$VAR || exit 1 (code=exi
ted, status=0/SUCCESS)
```

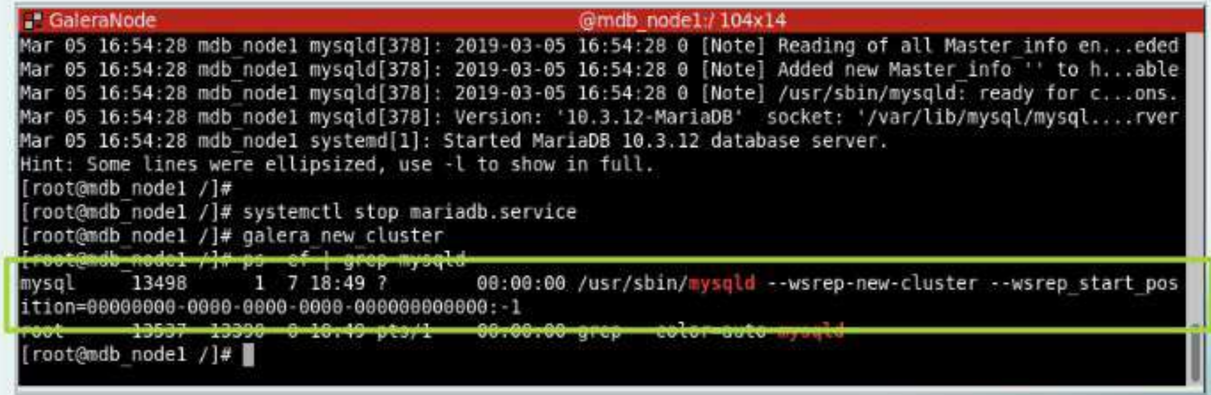
Start the First Node ONLY

On node1 start the initial node of the cluster using the following command:

$ galera_new_cluster

Validate the first node is running

$ ps -ef | grep mysqld



```
Mar 05 16:54:28 mdb_node1 mysqld[378]: 2019-03-05 16:54:28 0 [Note] Reading of all Master_info en...eded
Mar 05 16:54:28 mdb_node1 mysqld[378]: 2019-03-05 16:54:28 0 [Note] Added new Master_info '' to h...able
Mar 05 16:54:28 mdb_node1 mysqld[378]: 2019-03-05 16:54:28 0 [Note] /usr/sbin/mysqld: ready for c...ons.
Mar 05 16:54:28 mdb_node1 mysqld[378]: Version: '10.3.12-MariaDB'  socket: '/var/lib/mysql/mysql....rver
Mar 05 16:54:28 mdb_node1 systemd[1]: Started MariaDB 10.3.12 database server.
Hint: Some lines were ellipsized, use -l to show in full.
[root@mdb_node1 /]#
[root@mdb_node1 /]# systemctl stop mariadb.service
[root@mdb_node1 /]# galera_new_cluster
[root@mdb_node1 /]# ps -ef | grep mysqld
mysql     13498      1  7 18:49 ?        00:00:00 /usr/sbin/mysqld --wsrep-new-cluster --wsrep_start_pos
ition=00000000-0000-0000-0000-000000000000:-1
root      13537  13398  0 18:49 pts/1    00:00:00 grep --color=auto mysqld
[root@mdb_node1 /]#
```

Check Cluster Size

Use the mysql client from node1:

$ mariadb -u root

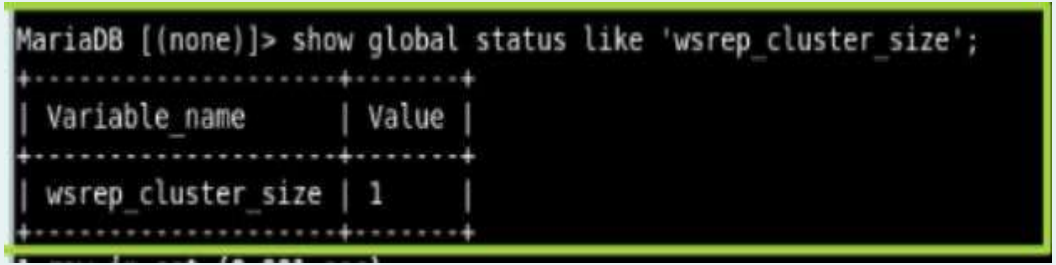
To check the cluster size execute the following command:

> show global status like 'wsrep_cluster_size’;

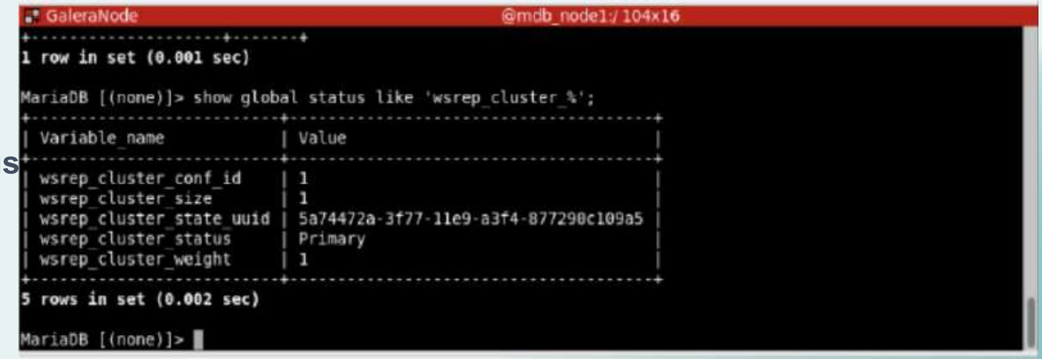
The cluster size should be 1

OPTIONALLY to get further details about the clus
execute:

> show global status like 'wsrep_cluster_%';



```
MariaDB [(none)]> show global status like 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 1     |
+--------------------+-------+
```



```
+--------------------+-------+
1 row in set (0.001 sec)

MariaDB [(none)]> show global status like 'wsrep_cluster_%';
+--------------------------+--------------------------------------+
| Variable_name            | Value                                |
+--------------------------+--------------------------------------+
| wsrep_cluster_conf_id    | 1                                    |
| wsrep_cluster_size       | 1                                    |
| wsrep_cluster_state_uuid | 5a74472a-3f77-11e9-a3f4-877290c109a5 |
| wsrep_cluster_status     | Primary                              |
| wsrep_cluster_weight     | 1                                    |
+--------------------------+--------------------------------------+
5 rows in set (0.002 sec)

MariaDB [(none)]> 
```

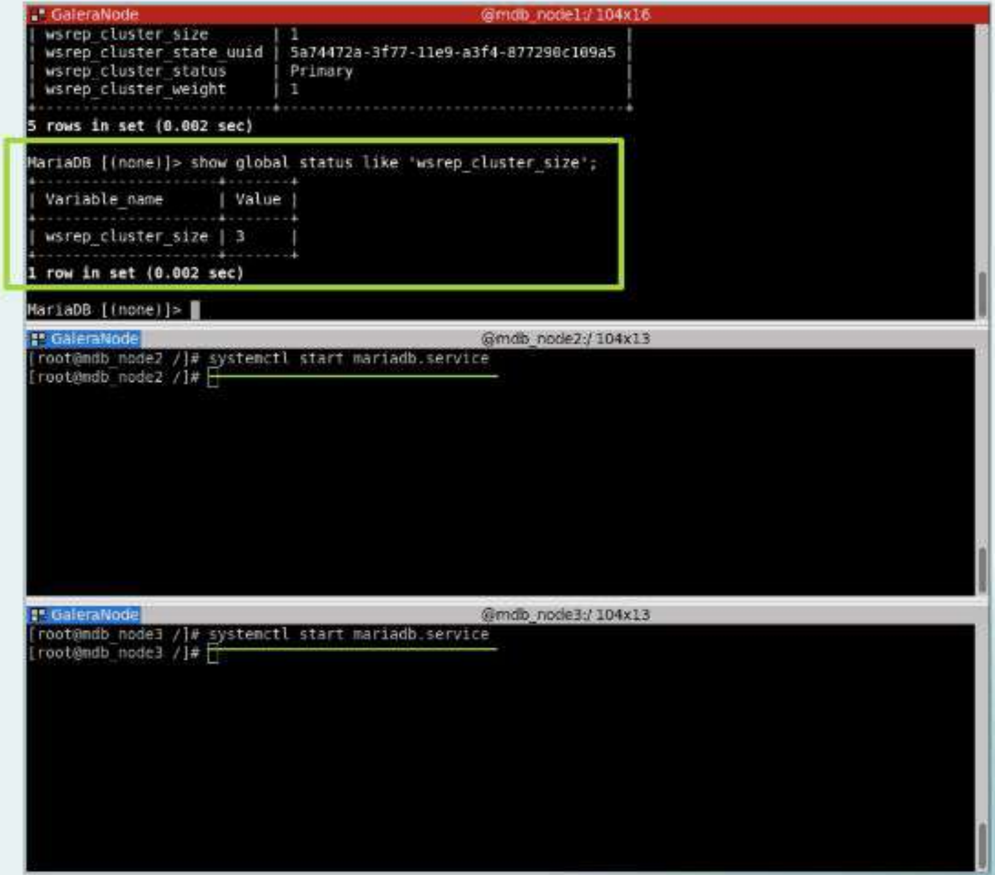
Start Additional Nodes

For each additional node start the MariaDB service.

$ systemctl start mariadb.service

Then check the new servers were added to the cluster:

> show global status like 'wsrep_cluster_size';



```
| wsrep_cluster_size        | 1                                    |
| wsrep_cluster_state_uuid | 5a74472a-3f77-11e9-a3f4-877298c109a5 |
| wsrep_cluster_status      | Primary                              |
| wsrep_cluster_weight      | 1                                    |
+--------------------------+--------------------------------------+
5 rows in set (0.002 sec)

MariaDB [(none)]> show global status like 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
1 row in set (0.002 sec)

MariaDB [(none)]> 
```

```
[root@mdb_node2 /]# systemctl start mariadb.service
[root@mdb_node2 /]# 
```

```
[root@mdb_node3 /]# systemctl start mariadb.service
[root@mdb_node3 /]# 
```

Synchronous Replication

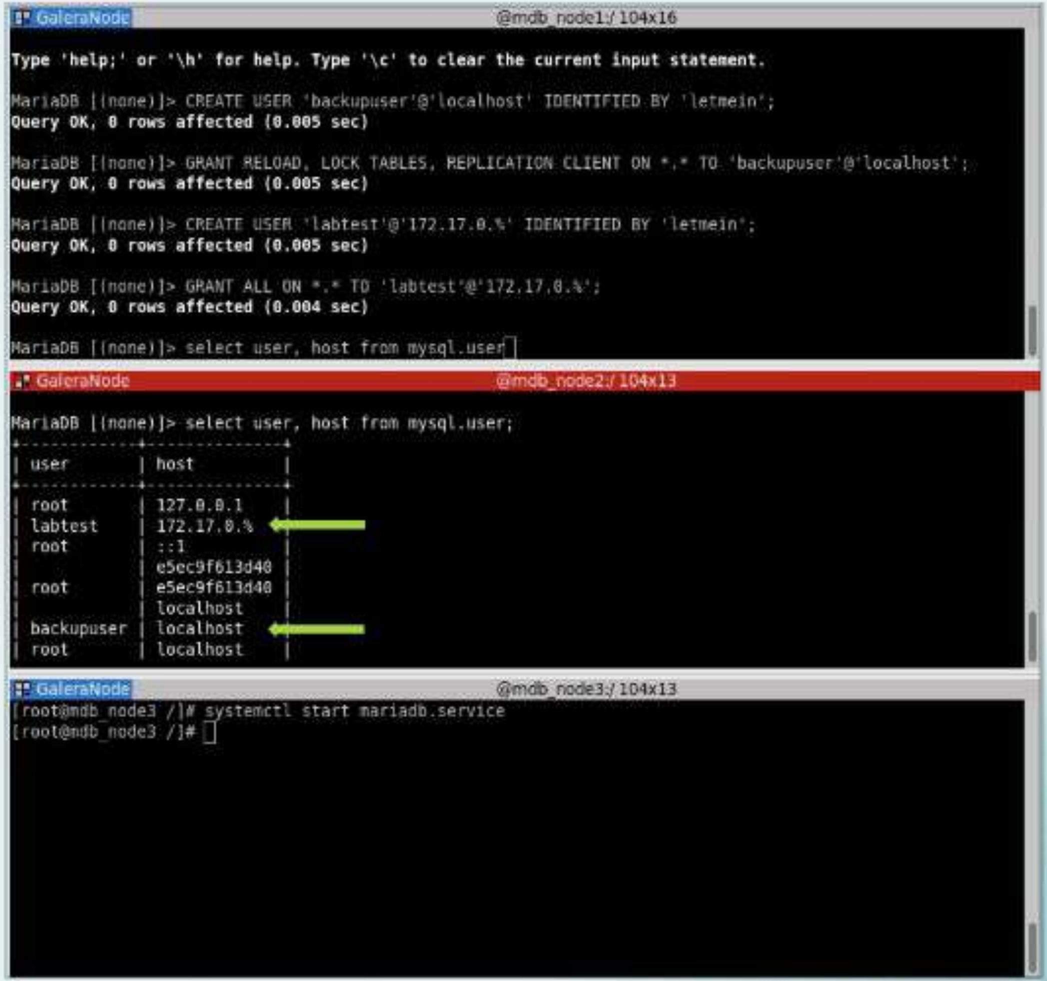
Create Users and Grant Privileges

On node 1 create a users and grant privileges. The DCL commands are replicated:

> CREATE USER 'backupuser'@'localhost' IDENTIFIED BY 'letmein';
> GRANT RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO 'backupuser'@'localhost';
> CREATE USER 'labtest'@'172.17.0.%' IDENTIFIED BY 'letmein';
> GRANT ALL ON *.* TO 'labtest'@'172.17.0.%';

On node 2 check the users have been replicated:

> select user, host from mysql.user;



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

MariaDB [(none)]> CREATE USER 'backupuser'@'localhost' IDENTIFIED BY 'letmein';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> GRANT RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO 'backupuser'@'localhost';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> CREATE USER 'labtest'@'172.17.0.%' IDENTIFIED BY 'letmein';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> GRANT ALL ON *.* TO 'labtest'@'172.17.0.%';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> select user, host from mysql.user
```

```
MariaDB [(none)]> select user, host from mysql.user;
+------------+--------------+
| user       | host         |
+------------+--------------+
| root       | 127.0.0.1    |
| labtest    | 172.17.0.%   |
| root       | ::1          |
|            | e5ec9f613d40 |
| root       | e5ec9f613d40 |
|            | localhost    |
| backupuser | localhost    |
| root       | localhost    |
```

```
[root@mdb_node3 /]# systemctl start mariadb.service
[root@mdb_node3 /]#
```

Replicate DDL and DML Statements

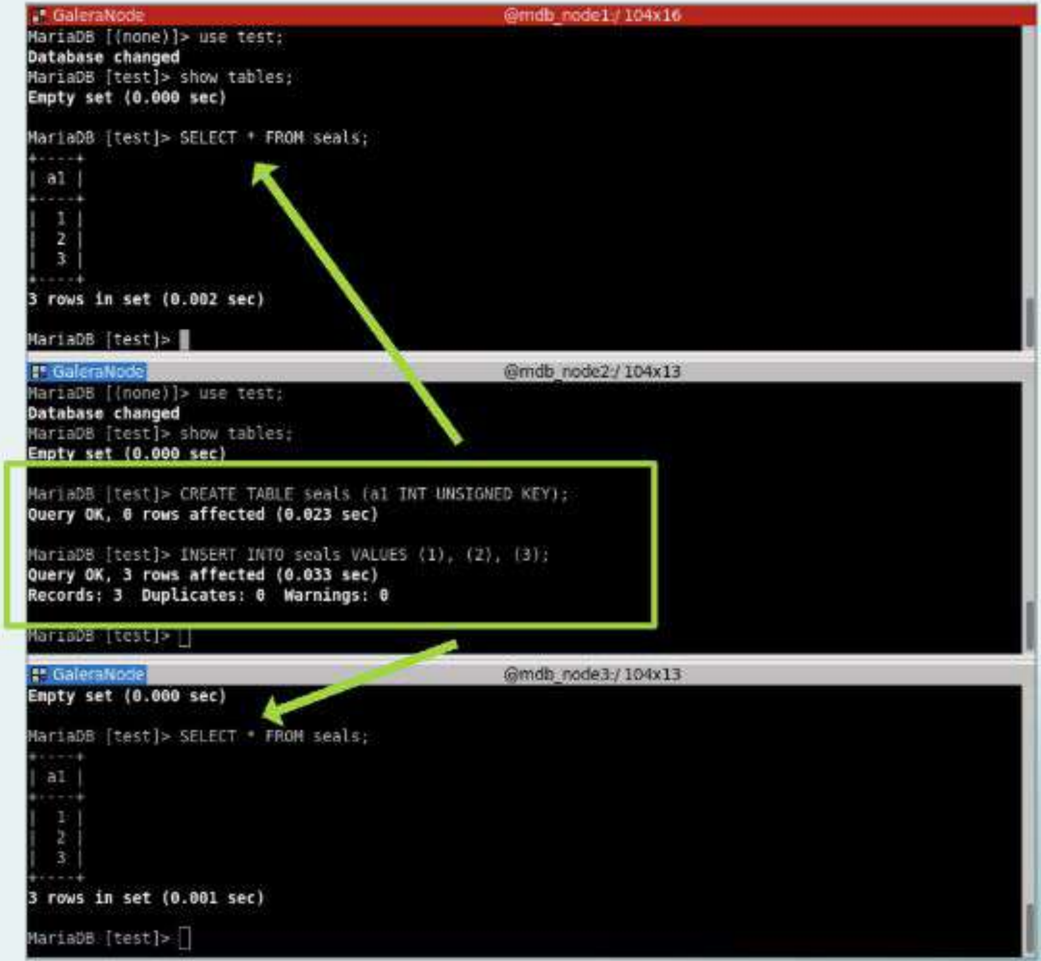
From any node in the cluster, create and insert values to a table (note there are no tables in the test database):

```
> CREATE TABLE seals (a1 INT UNSIGNED KEY);
> INSERT INTO seals VALUES (1), (2), (3);
```

Then select all rows from the table on different nodes:

```
> SELECT * FROM seals;
```

DML and DDL commands have been synchronously replicated



```
MariaDB [(none)]> use test;
Database changed
MariaDB [test]> show tables;
Empty set (0.000 sec)

MariaDB [test]> SELECT * FROM seals;
+----+
| a1 |
+----+
|  1 |
|  2 |
|  3 |
+----+
3 rows in set (0.002 sec)

MariaDB [test]>
```

```
MariaDB [(none)]> use test;
Database changed
MariaDB [test]> show tables;
Empty set (0.000 sec)

MariaDB [test]> CREATE TABLE seals (a1 INT UNSIGNED KEY);
Query OK, 0 rows affected (0.023 sec)

MariaDB [test]> INSERT INTO seals VALUES (1), (2), (3);
Query OK, 3 rows affected (0.033 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [test]>
```

```
Empty set (0.000 sec)

MariaDB [test]> SELECT * FROM seals;
+----+
| a1 |
+----+
|  1 |
|  2 |
|  3 |
+----+
3 rows in set (0.001 sec)

MariaDB [test]>
```

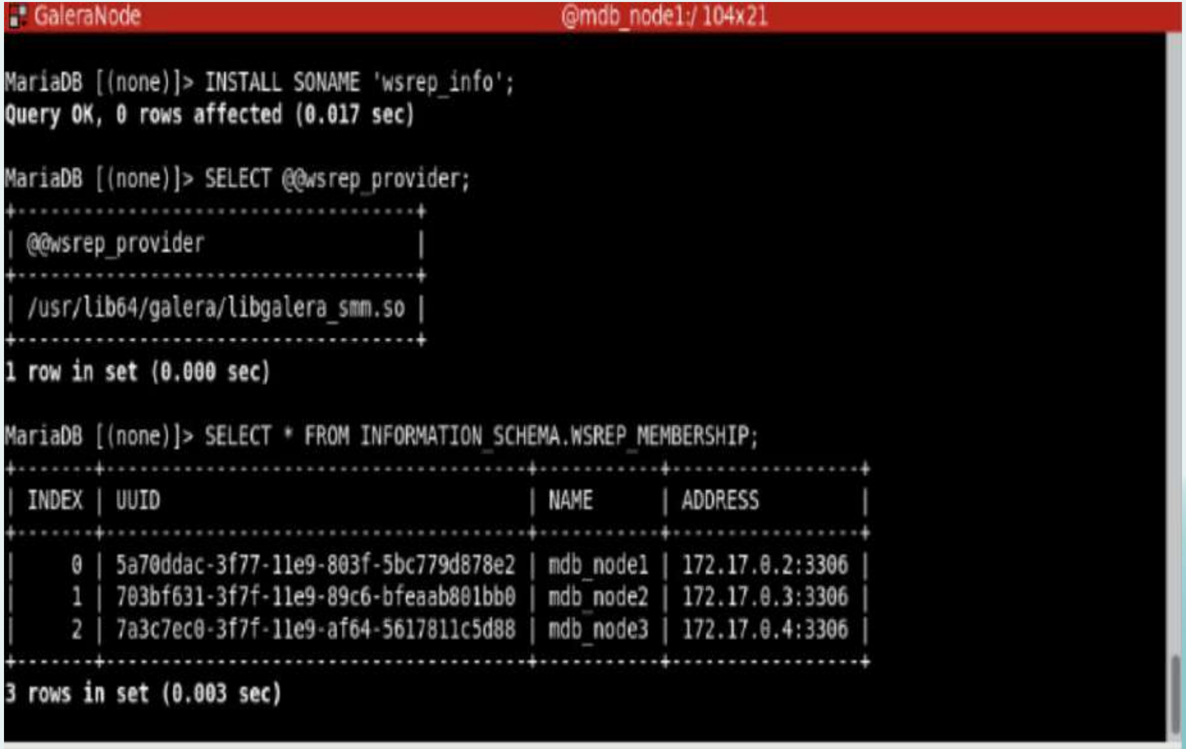
Information_Schema for Galeara

Check which replication provider is used:

> SELECT @@wsrep_provider;

Get details about replication members

> SELECT * FROM INFORMATION_SCHEMA.WSREP_MEMBERSHIP;



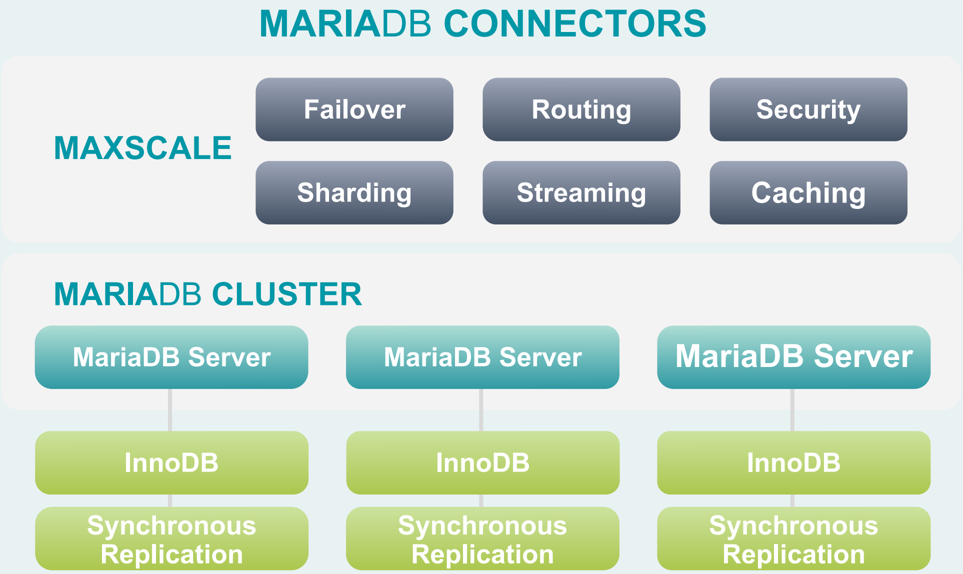
```
MariaDB [(none)]> INSTALL SONAME 'wsrep_info';
Query OK, 0 rows affected (0.017 sec)

MariaDB [(none)]> SELECT @@wsrep_provider;
+------------------------------------+
| @@wsrep_provider                   |
+------------------------------------+
| /usr/lib64/galera/libgalera_smm.so |
+------------------------------------+
1 row in set (0.000 sec)

MariaDB [(none)]> SELECT * FROM INFORMATION_SCHEMA.WSREP_MEMBERSHIP;
+-------+--------------------------------------+-----------+-----------------+
| INDEX | UUID                                 | NAME      | ADDRESS         |
+-------+--------------------------------------+-----------+-----------------+
|     0 | 5a70ddac-3f77-11e9-803f-5bc779d878e2 | mdb_node1 | 172.17.0.2:3306 |
|     1 | 703bf631-3f7f-11e9-89c6-bfeaab801bb0 | mdb_node2 | 172.17.0.3:3306 |
|     2 | 7a3c7ec0-3f7f-11e9-af64-5617811c5d88 | mdb_node3 | 172.17.0.4:3306 |
+-------+--------------------------------------+-----------+-----------------+
3 rows in set (0.003 sec)
```

MariaDB Enterprise Cluster, Powered By Galera

An active-active, multi-master synchronous replication solution

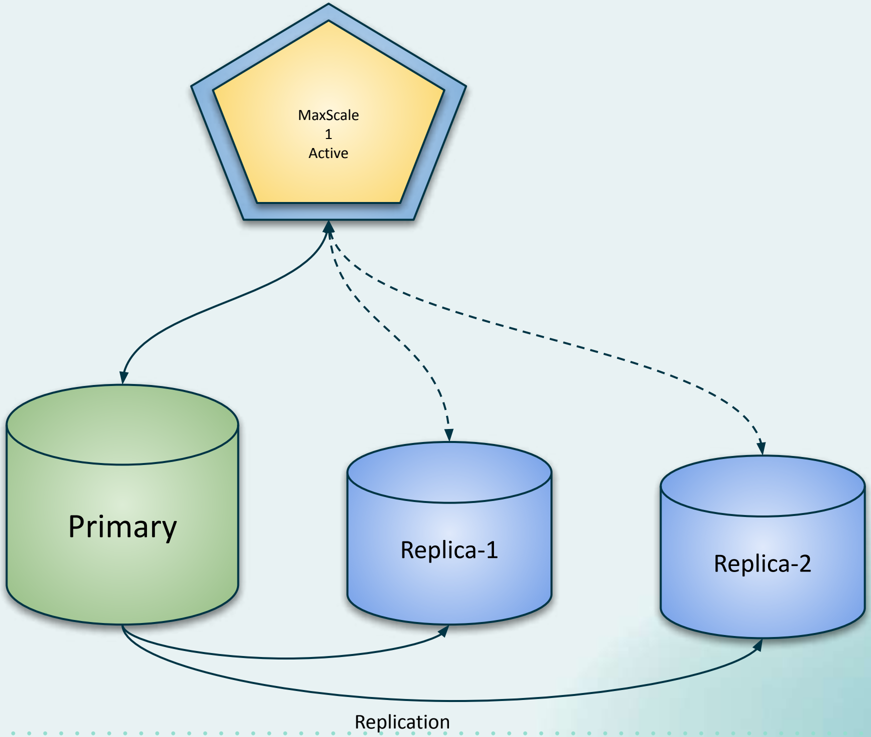


MariaDB Galera Cluster, an open source active-active, multimaster synchronous replication solution that provides parallel replication and data consistency across all nodes.

- High Availability for InnoDB (ACID compliant)
- All nodes are equal, read and writes to any node
- Asynchronous Replication between Clusters supported
- Typically an odd number of nodes to avoid split brain
- Automatically manages the identification and removal of failed nodes as well as rejoining new or repaired nodes

Asynchronous Replication

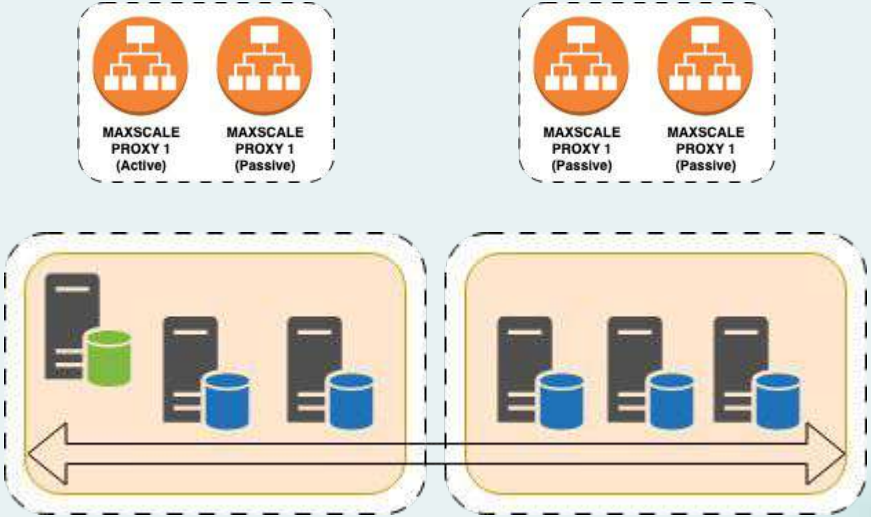
Basic HA Architecture



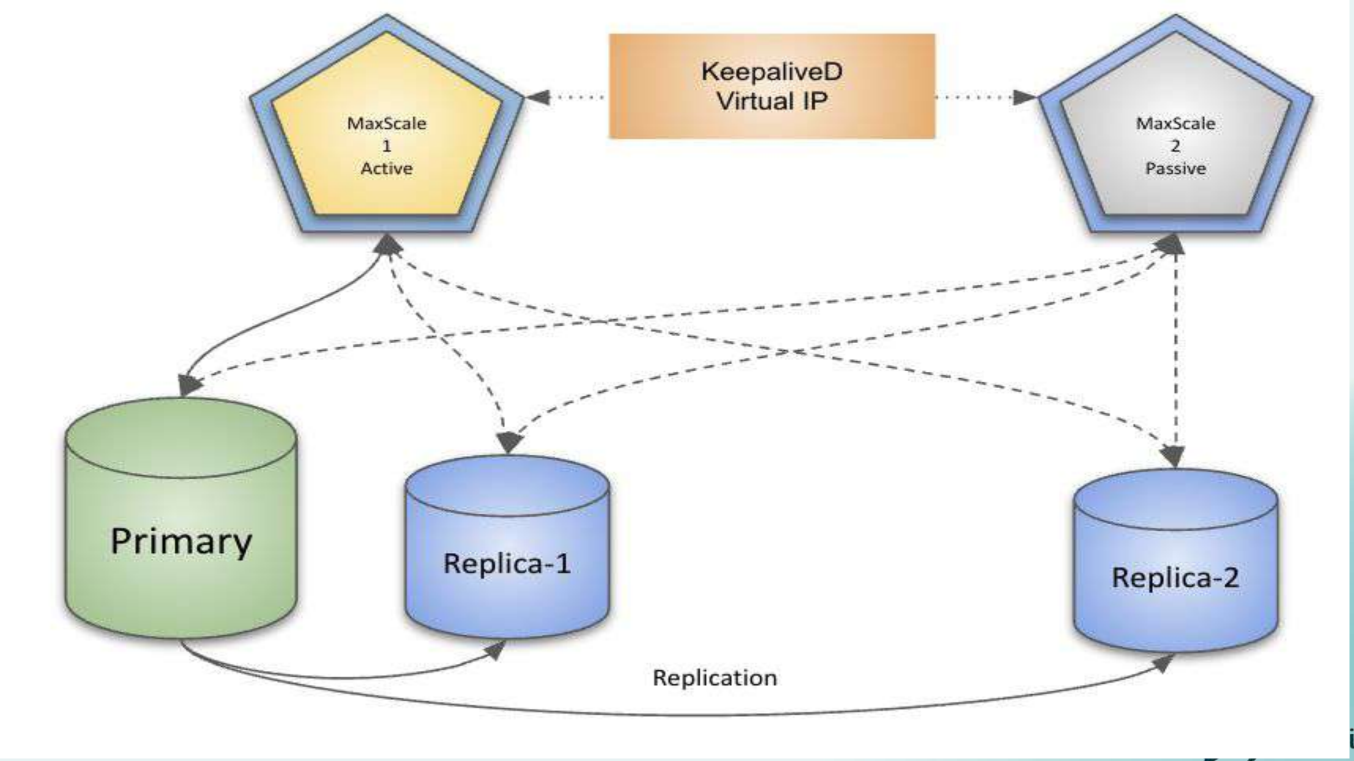
- Only one MaxScale
- Single point of failure
- MariaDB Backend HA

… and an Ideal data layer?

```
jdbc:mariadb:sequential://10.0.6.1,10.0.6.2,10.0.6.3,10.0.6.4/database
```



If not, a typical recommended HA Architecture



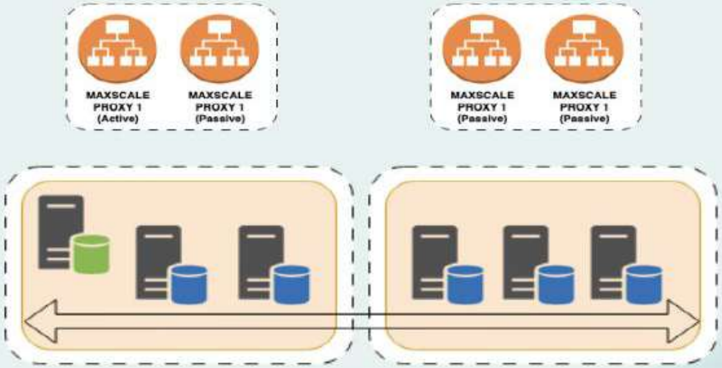
Demo time !

MaxScale Auto Failover

MaxScale Cooperative Monitoring

Why do we need Cooperative Monitoring?

When multiple MaxScale instances are used in a highly available deployment, MariaDB Monitor needs to ensure that only one MaxScale instance performs automatic failover operations at a given time. It does this by using cooperative locks on the back-end servers.


MAXSCALE PROXY 1 (Active)
MAXSCALE PROXY 1 (Passive)
MAXSCALE PROXY 1 (Passive)
MAXSCALE PROXY 1 (Passive)

Two important parameters to achieve this…

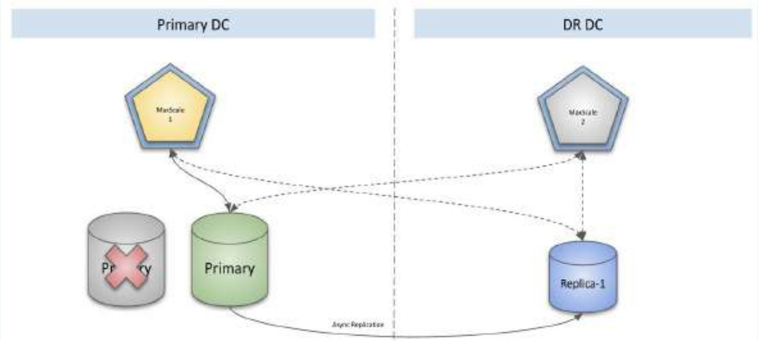
- ## Majority_of_running
  - MaxScale node that has the maximum number of locks will become the Primary
  - In this mode, the total number of “**Running**” MariaDB nodes are considered excluding the nodes that are down.

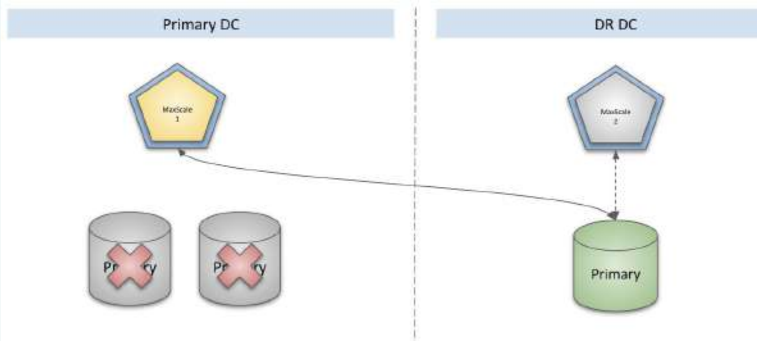
- ## Majority_of_all
  - In this mode, all the nodes are considered
  - MaxScale node that has the maximum number of locks will become the Primary

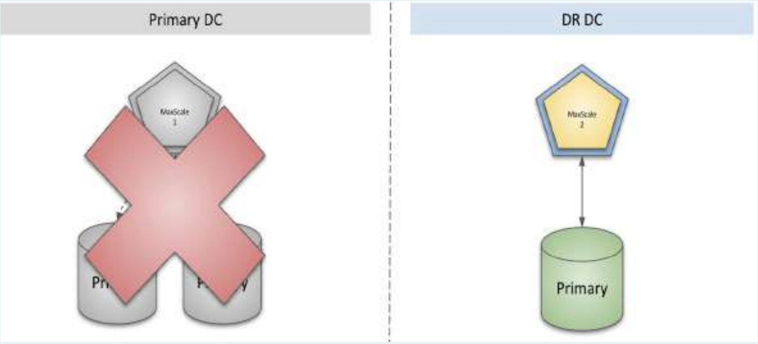
- ## How to calculate the locks?

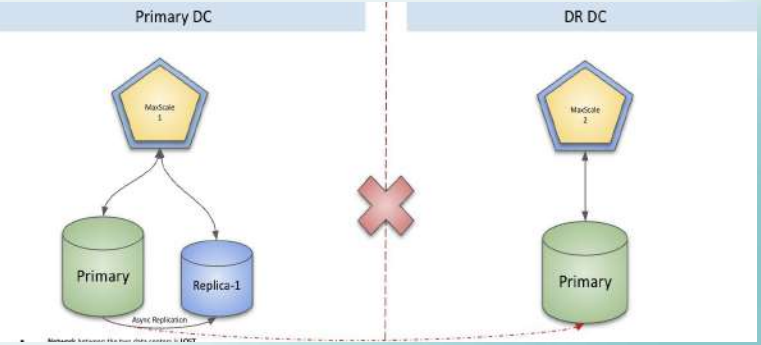
  - **`Round`** `the result down:` **`n_servers`**`/2 + 1`
  - **Majority_of_running - “`n_servers`”** is the total **number alive servers**
  - **Majority_of_all - “`n_servers`”** is the total **number of MariaDB servers**

Majority_of_running

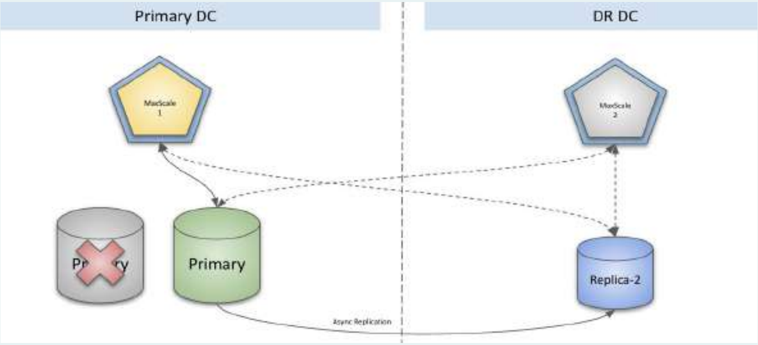


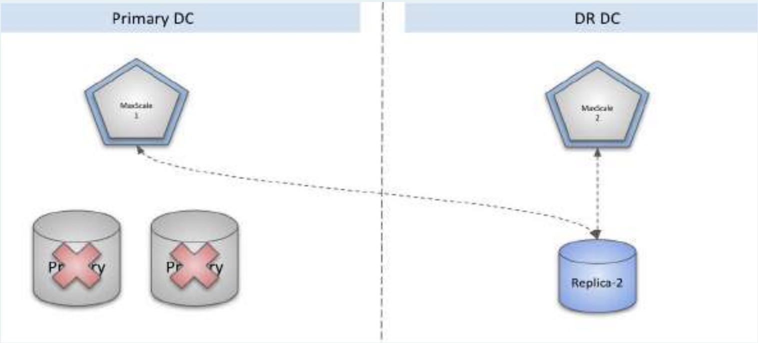


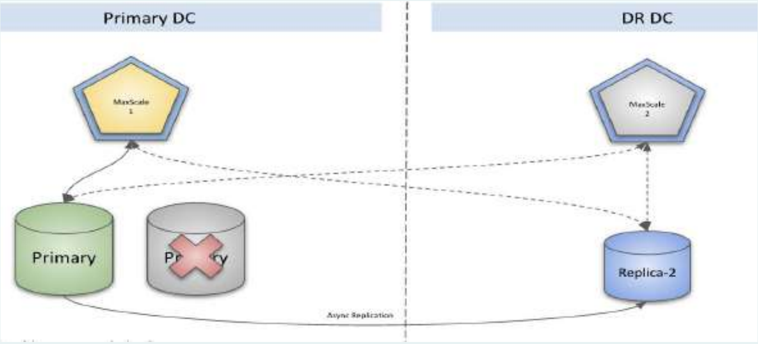


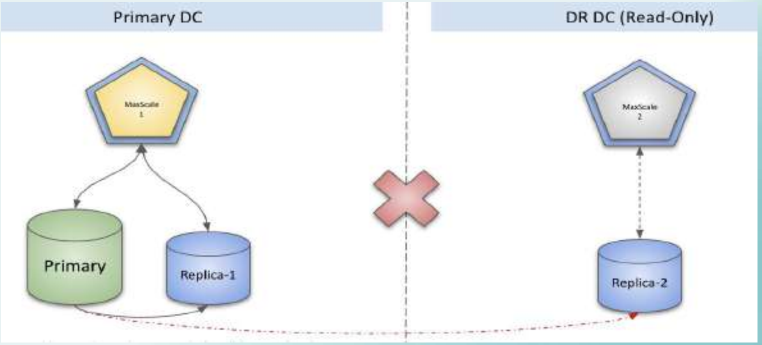


Majority_of_all

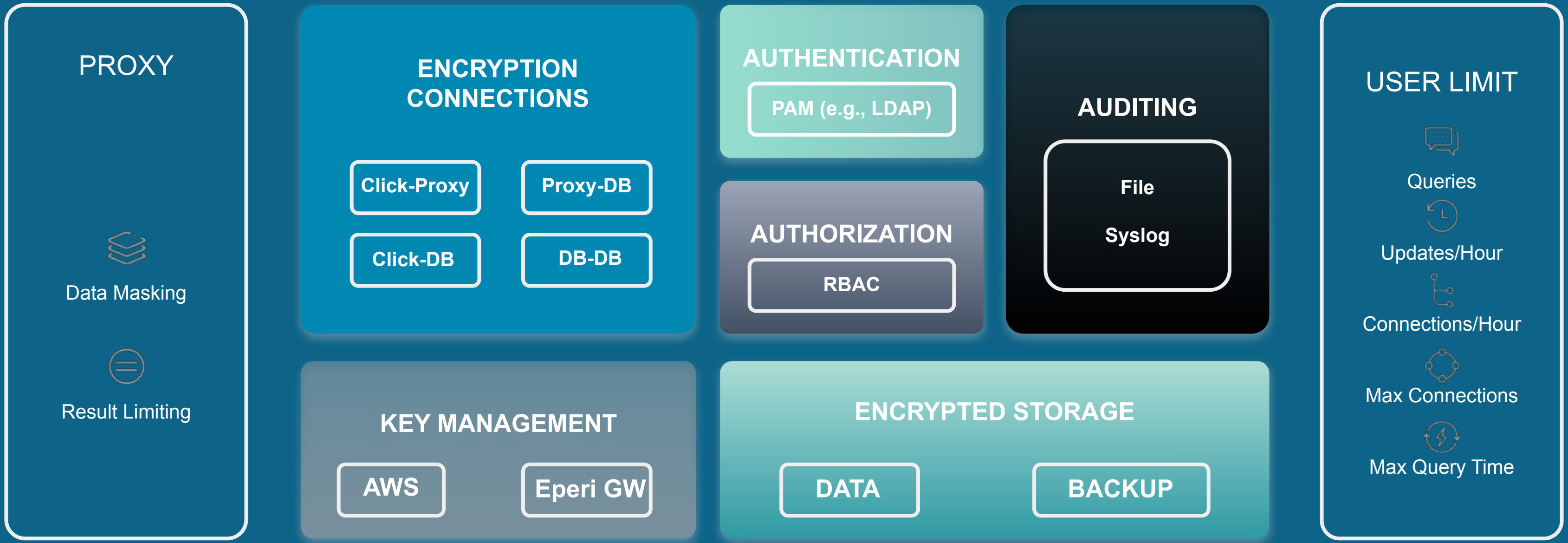








Comprehensive Security



PROXY

Data Masking

Result Limiting

ENCRYPTION CONNECTIONS

Click-Proxy

Proxy-DB

Click-DB

DB-DB

AUTHENTICATION

PAM (e.g., LDAP)

AUTHORIZATION

RBAC

AUDITING

File

Syslog

KEY MANAGEMENT

AWS

Eperi GW

ENCRYPTED STORAGE

DATA

BACKUP

USER LIMIT

Queries

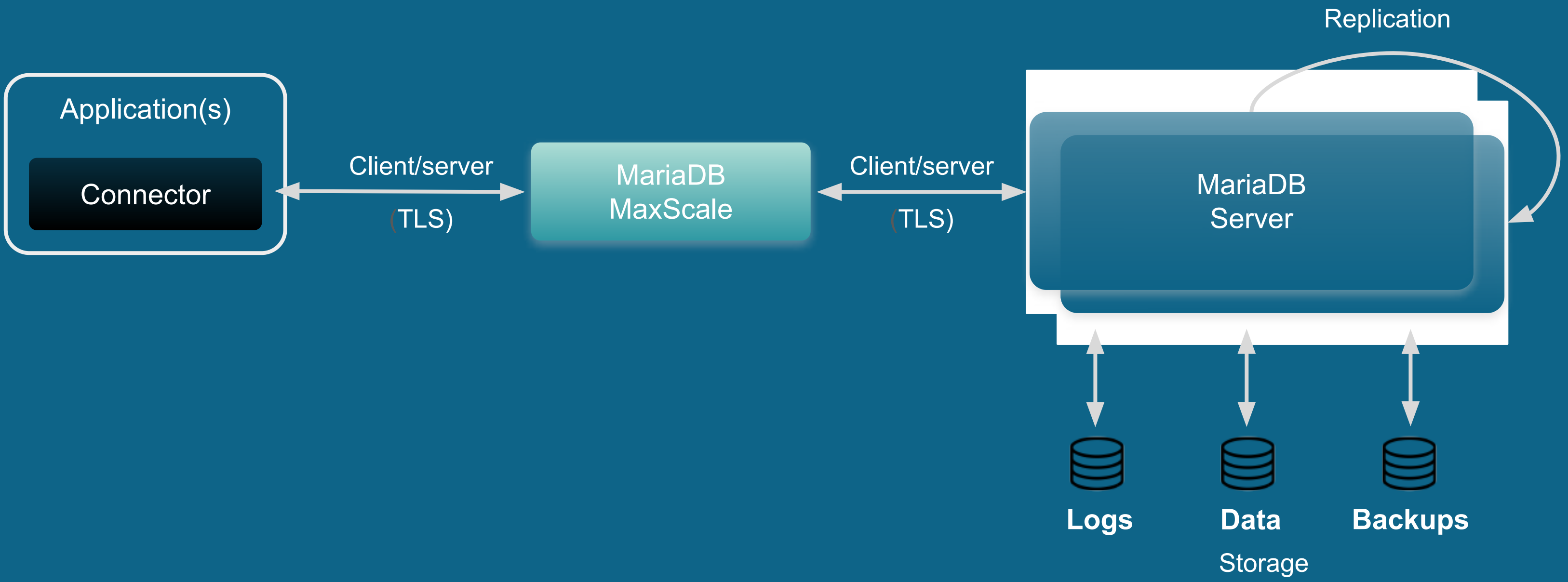
Updates/Hour

Connections/Hour

Max Connections

Max Query Time

END-TO-END ENCRYPTION



Demo time !

Transparent Data Encryption (TDE)

MariaDB Audit Plugins - Key Features

User Activity Logging

Logs user logins, logouts, and any SQL queries executed by users. This logging helps in tracking down unauthorized access and potential misuse.

Flexible Event Filtering

Customize the logging of specific event types, such as CONNECT, QUERY and TABLE (for table-specific actions). selecting only relevant event types and focusing on critical activities.

Compliance Support

GDPR, HIPAA, PCI DSS Compliant. The audit plugin provides a detailed and timestamped record of activities, which can be crucial during audits.

Json Log Output

The latest plugin version supports Json logging making it easier to parse and analyze logs with modern log management tools like Elastic Stack or Splunk.

Log Management System Integration

Administrators can review database activities alongside logs from other infrastructure, improving the ability to spot suspicious behavior.

Audit Log Destinations

Logging to a file or System Log

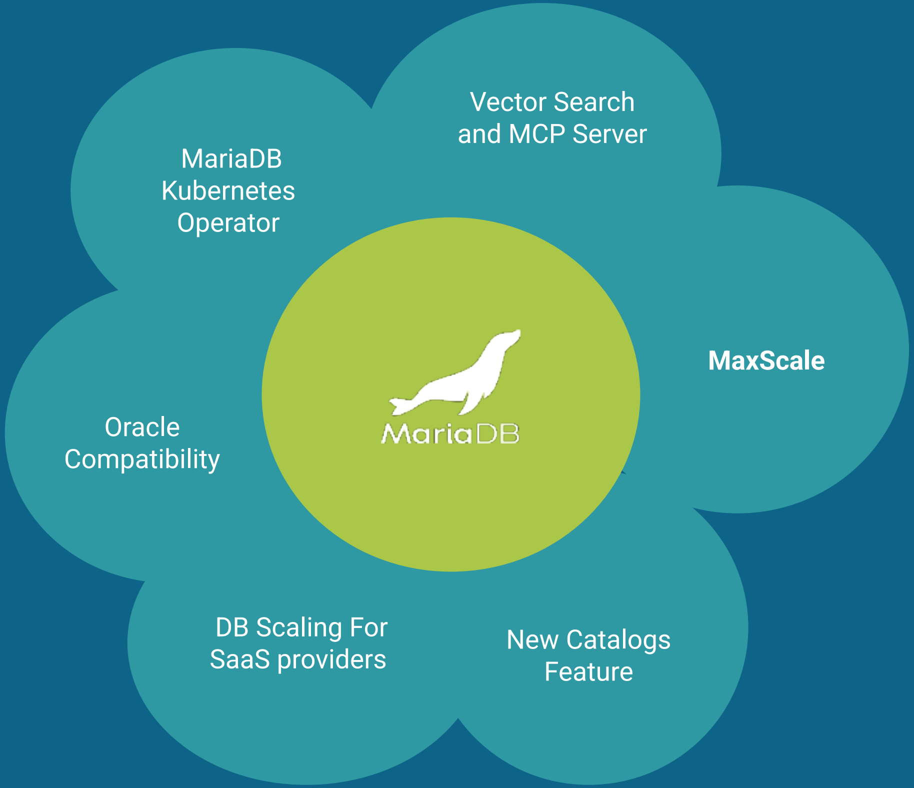
Log Rotation

MariaDB's audit plugin supports log rotation, allowing you to manage log file sizes and retain important audit data.

Authentication and Authorization in MariaDB

Authentication	PAM	Unix Socket	MySQL Native Password	GSSAPI, ED25591, PARSEC, SHA-256, Named Pipes
Authorization	RBAC	Principle of Least Privilege	Host Specific Access	Regular Audits
Benefits	Simplified Administration	Reduced Risk	Improved Compliance	Better scalability

Innovating in Motion



MariaDB Kubernetes Operator

Vector Search and MCP Server

MaxScale

New Catalogs Feature

DB Scaling For SaaS providers

Oracle Compatibility

Need Assessment of your MySQL / MariaDB environment, or

Looking for a PoC ?

NEXT – Part 3

Ensuring Business Continuity and Compliance with Backup, Restore and Disaster Recovery Strategies with MariaDB Enterprise

21 August 2025 02:00 pm MYT

Speaker

Kanthimathi Kailasanathan
Database Consultant,
Sales Engineering
at MariaDB

UPCOMING WEBINARS

25 Sep I 2pm
Avoiding Vendor lock-in and Lowering cost for Oracle dependent applications with MariaDB Enterprise

23 Oct I 2pm
Application Modernisation using GenAI capabilities with MariaDB Enterprise Vector Database

REGISTER NOW

Crest Infosolutions Sdn Bhd info@crestsolution.com www.crestsolution.com

Email: info@crestsolution.com

Web: www.crestsolution.com

Thank You

MariaDB

CREST

- Ideate - Innovate - Implement -