MARIADB CLUSTER

MARIADB **TRAINING**

MARIADB PROFILE

mariadb.com

CUSTOMERS ACROSS ALL INDUSTRIES

- Barclays
- Brussels Airport
- DBS
- CVS Health
- National Inst. of Health
- Royal Bank of Canada
- Samsung
- T-Systems
- Verizon
- Walgreens

OUR INVESTORS

- Alibaba Cloud
- Intel Capital
- California Technology Ventures
- ServiceNow
- Open Ocean
- Runa Capital

200+ EMPLOYEES

Proven leadership team

26 countries
9 offices worldwide

DISTINCTIONS & AWARDS

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

191k+ Open Source contributions, the highest number in the industry

Voted database of the year 2013 – 2020, LinuxQuestions.org

New Cloud Offering MariaDB SkySQL:

- 20 Coolest Cloud Software Companies Of The 2021 Cloud 100, CRN
- 2021 Technology of the Year Award winner, InfoWorld

CLASS INTRODUCTIONS

Instructor

Name and background

Class Participants

Name and company

MariaDB experience

How you currently use MariaDB

What you expect to get out of this course

MARIADB PRODUCTS

Products

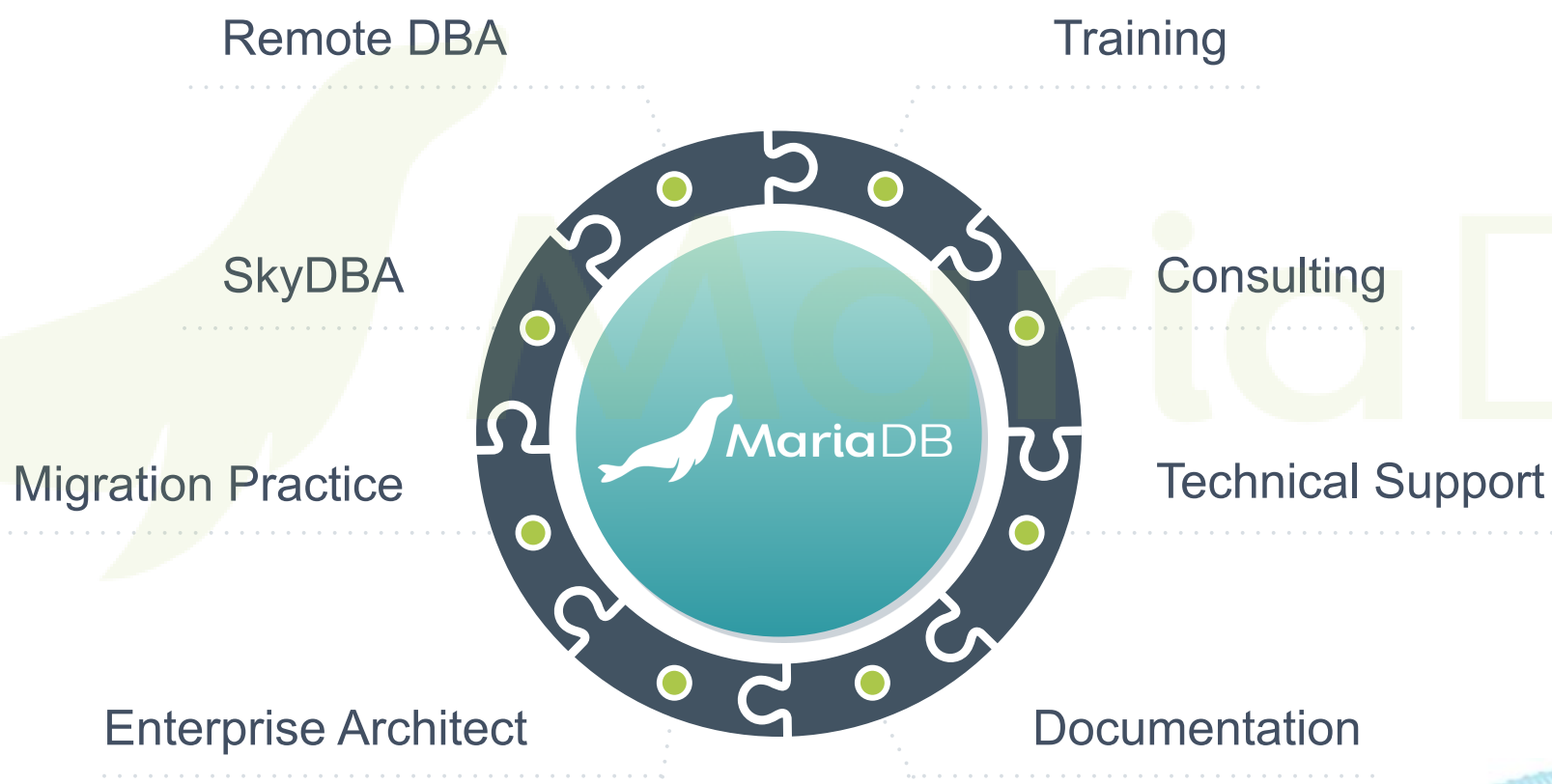
Solutions

- MariaDB Enterprise
- MariaDB SkySQL

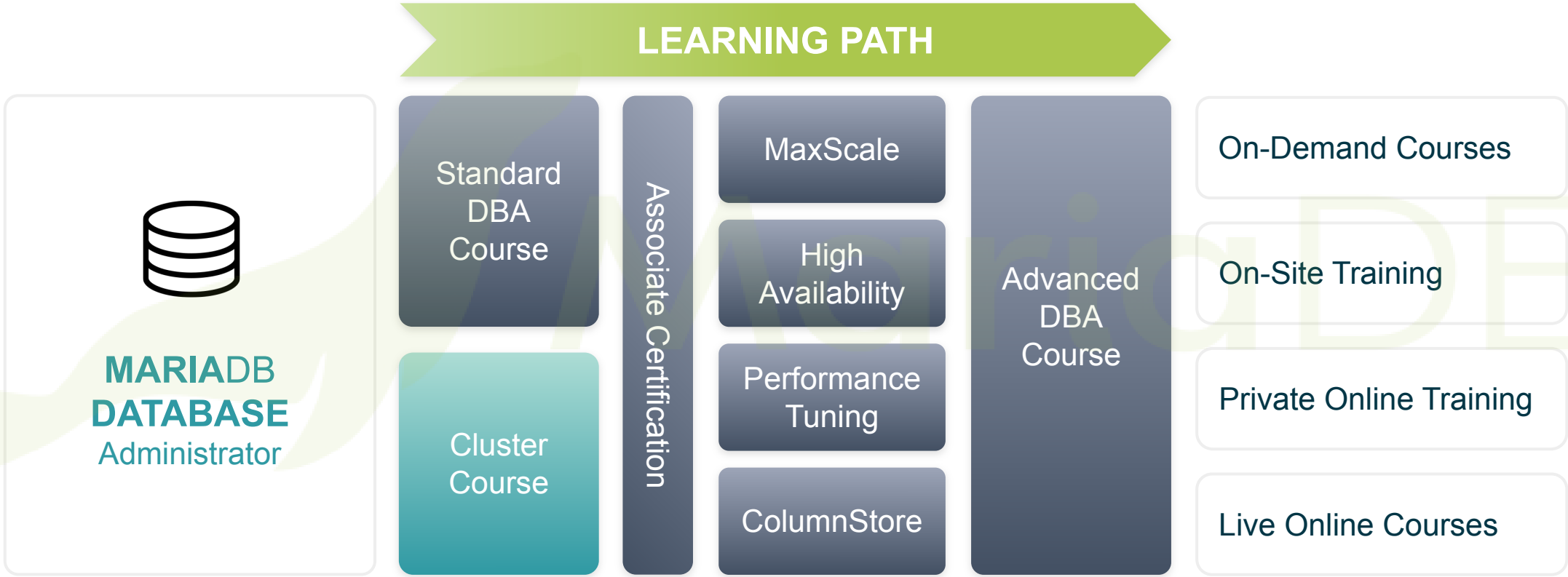
Technology

- Database: Enterprise Server
- Distributed SQL: Xpand
- Columnar Storage: ColumnStore
- Database Proxy: MaxScale

MARIADB SERVICES AND SUPPORT



MARIADB CERTIFICATION & TRAINING



COURSE OBJECTIVES

- Understand concepts such as cluster replication associated with MariaDB Cluster
- Explain the architecture of MariaDB Cluster
- Install and configure a basic setup of MariaDB Cluster
- Plan and perform an upgrade of MariaDB Cluster
- Understand how to migrate standalone or replication topologies to MariaDB Cluster
- Perform backup and restore operations using MariaDB Backup
- Secure MariaDB Cluster
- Monitor and improve MariaDB Cluster performance based on key performance indicators
- Understand and troubleshoot common MariaDB Cluster problems
- Describe common cluster topologies
- Understand how to load balance MariaDB Cluster using MariaDB MaxScale
- Explain and understand how to manage some known cluster specifics of such as Split Brain, consistent reads, controlling the AUTO_INCREMENT attribute, detecting slow nodes and parallel replication
- Use best practices to perform schema upgrades
- Explain how State Snapshot Transfers (SSTs) and Incremental State Transfers (ISTs) work in MariaDB Cluster
- Describe the advantages and disadvantages of Geo-Replication
- Set up advanced features such as MariaDB Cluster Notifications and Arbitrator
- Explain how and when to use other advanced features such as multicasting, multiple clusters, and WAN replication

COURSE OVERVIEW

ARCHITECTURE
Cluster Replication
System Tables
Global Transaction ID (GTID)

GETTING STARTED
Installation
Configuration
Node Initialization
Cluster Initialization
Migrating to MariaDB Cluster
Upgrading MariaDB Cluster

STATE TRANSFERS
State Snapshot Transfer (SST)
Incremental State Transfer (IST)

CLUSTER SPECIFICS
Split Brain
Controlling AUTO_INCREMENT
AUTO_INCREMENT Behavior
Causal Reads
Flow Control
True Parallel Replication

BENCHMARKING
Cluster Performance
Sysbench Installation
Benchmarking with sysbench
Sysbench Output - Prepare
Sysbench Output - Run

SECURITY
Firewall Settings
SELinux Settings
Data-In-Transit Encryption
Data At Rest Encryption

MONITORING AND TROUBLESHOOTING
Database Logs
Status Variables
Replication Performance
Streaming Replication
Conflict Resolution
Node Failure and Recovery

MAINTENANCE
Schema Upgrades
Backing Up MariaDB Cluster
Cluster Recovery

HIGH AVAILABILITY
Cluster Topologies
Load Balancing
Geo-Distributed Database Clusters

ADVANCED FEATURES
Weighted Quorum
Arbitrator
Notifications
Multicasting
WAN Replication

APPENDICES
A - Lab Exercises Workbook

ARCHITECTURE

MariaDB Training

LEARNING **OBJECTIVES**

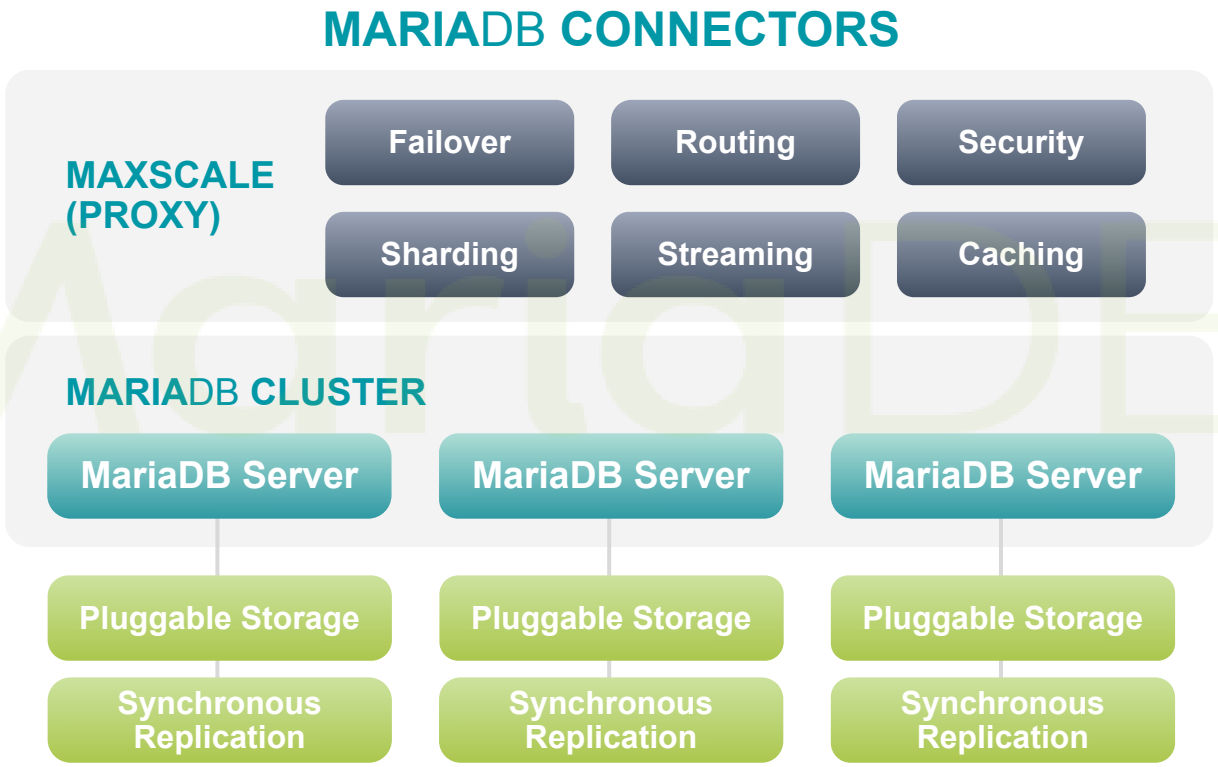
Describe the features of MariaDB Cluster

Explain Cluster Replication and its implementation in MariaDB Cluster

Understand the Global Transaction Identifier (GTID) in MariaDB Cluster

OVERVIEW OF MARIADB CLUSTER

- Typically an odd number of MariaDB servers
- Databases must be InnoDB
- Based on open wsrep API specification
- Synchronous replication hooks into the `COMMIT` process
- Uses row-based binlog information



MARIADB CLUSTER FEATURES

- Parallel slave applying
- Practically no replica lag
- Instant, trivial failover
- Automatic node provisioning
- Works well in WAN

CLUSTER REPLICATION

MariaDB Training

CERTIFICATION -BASED REPLICATION

Certification-Based Replication requires:

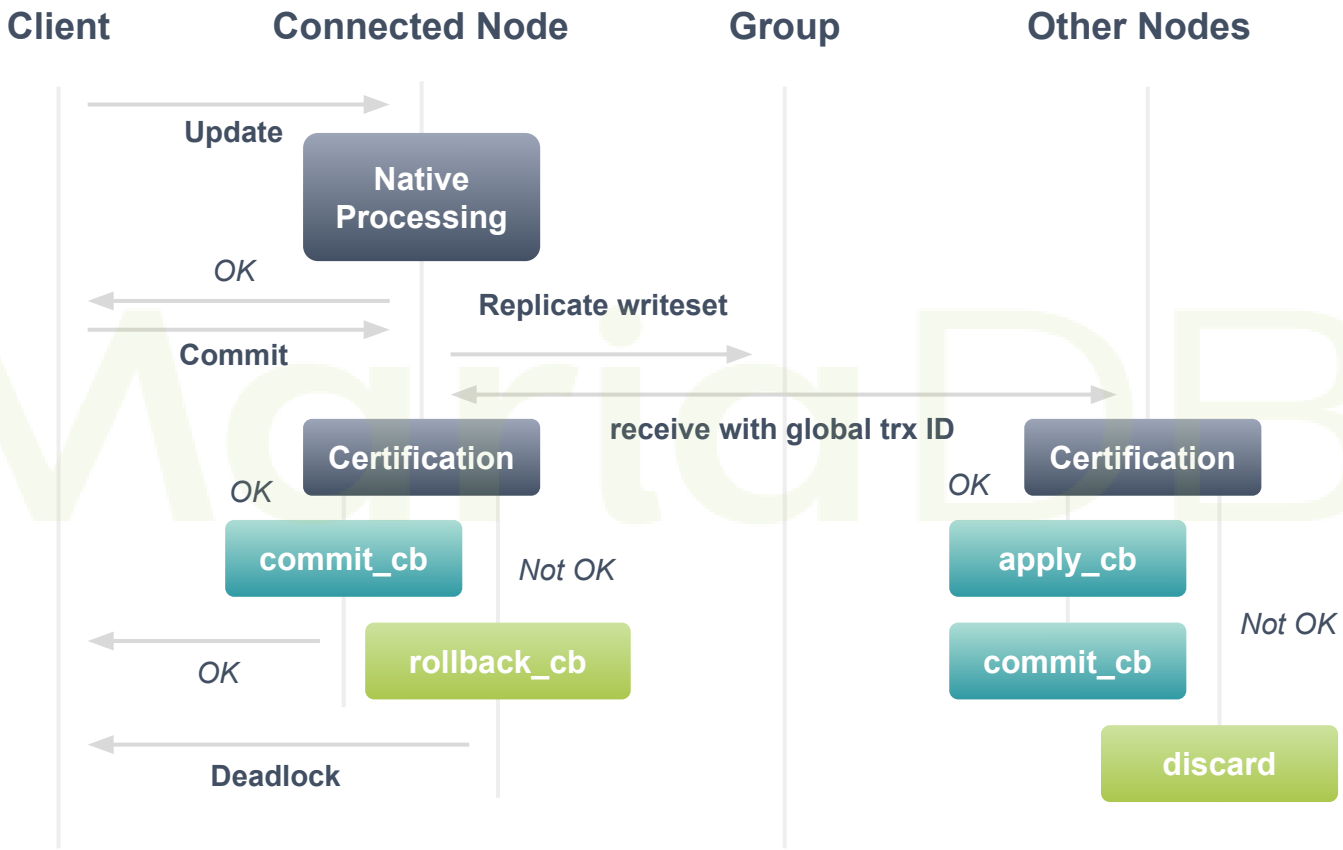
Transactional Database

Atomic Changes

Global Ordering

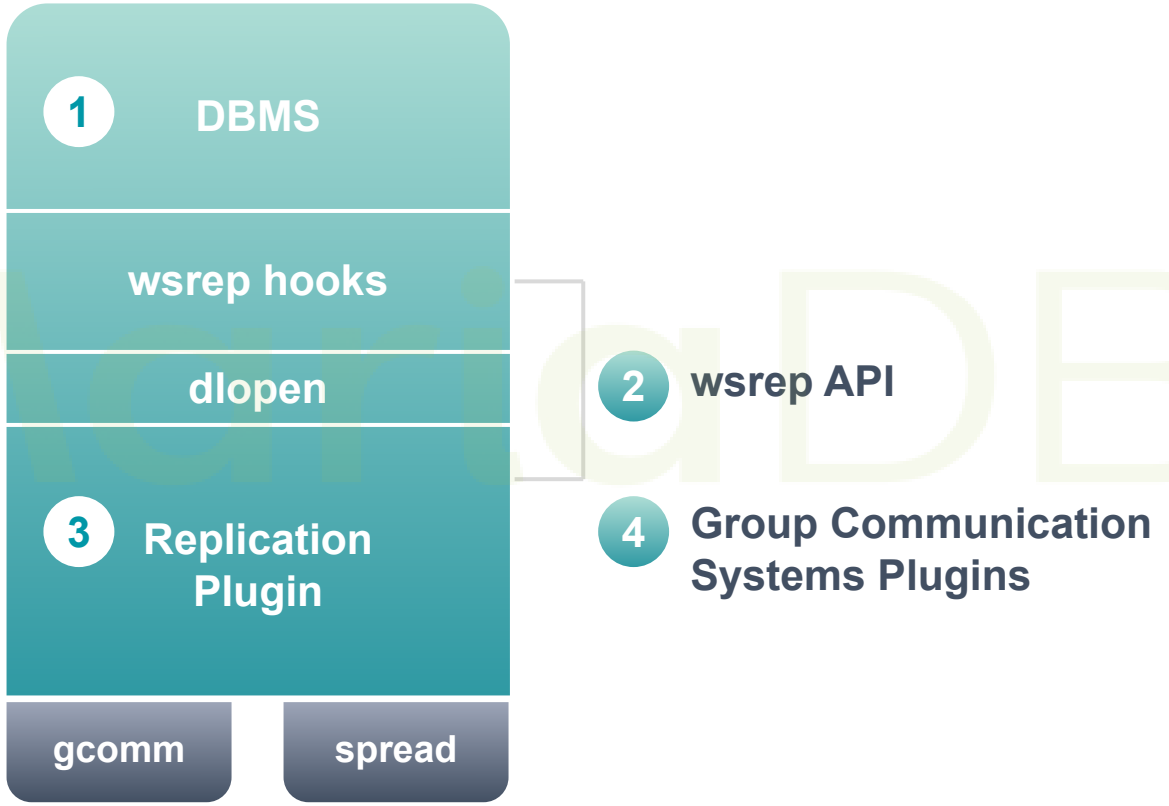
MariaDB Cluster's implementation of certification-based replication depends on the global ordering of transactions

OPTIMISTIC EXECUTION



REPLICATION API

The internal architecture of MariaDB Cluster revolves around four components



SYNCHRONOUS PENALTIES

Round Trip Time (RTT) is Length of Time for a Signal to be sent and Receipt of Acknowledgement

MariaDB Cluster copies data buffer to all cluster members on `COMMIT` from client (~1 RTT added latency)

Connection throughput equals 1/RTT trx/sec

Total throughput equals 1/RTT trx/sec × #connections

A given row can't be modified more than 1/RTT times a second

With parallel writes on all nodes the synchronous replication can suffer from network bandwidth

```
100Mbit/s from 3 nodes equals 300Mbit/s traffic
```

SYSTEM TABLES

MariaDB Training

SYSTEM TABLES – CLUSTER REPLICATION

Galera 4 added three new tables to the `mysql` system database.

`wsrep_cluster`: stores the UUID of the cluster and some other identification information, as well as the cluster's capabilities.

`wsrep_cluster_members`: stores current membership of the cluster; it will contain a row for each node in the cluster.

`wsrep_streaming_log`: meta data and row events for ongoing streaming transactions, write set fragment per row.

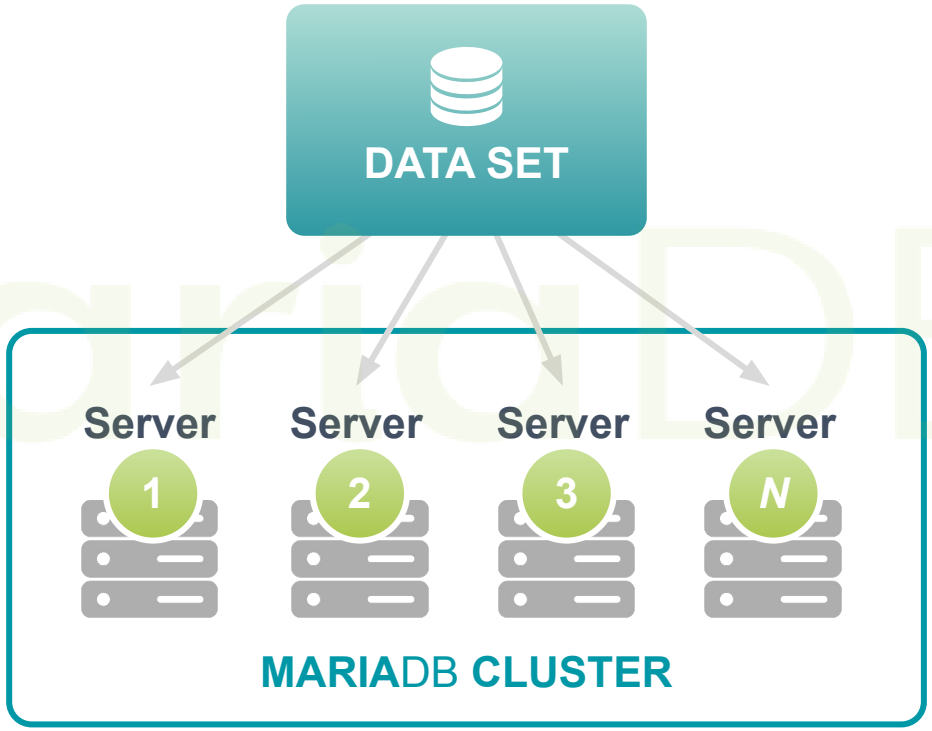
GLOBAL TRANSACTION ID (GTID)

MariaDB Training

DATA CENTRIC APPROACH

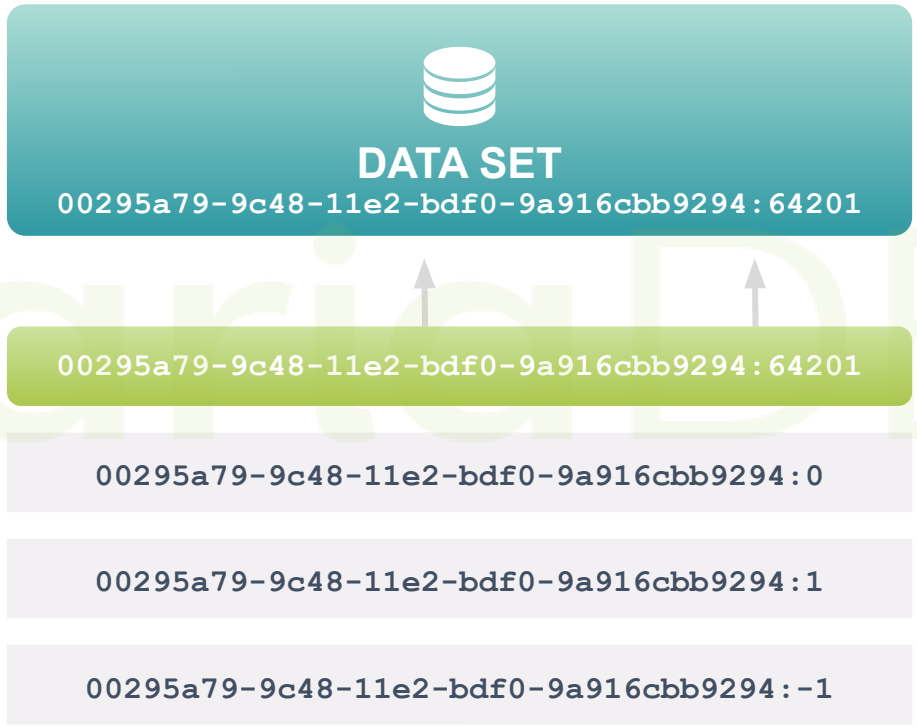
- Data doesn't belong to a single node
- Nodes belong to the data
- Data is synchronized among two or more MariaDB servers
- MariaDB Cluster nodes are anonymous and all nodes are equal
- MariaDB Cluster is one large distributed primary
- A data set needs an identifier
  - Data set identifier is a Cluster identifier

```
00295a79-9c48-11e2-bdf0-9a916cbb9294
```



GLOBAL **TRANSACTION IDENTIFIER (GTID)**

The Global Transaction Identifier (GTID) in MariaDB Cluster is composed of the data set plus a sequence of atomic changes.



DIFFERING GTID IMPLEMENTATIONS

MariaDB 10 GTID

```
0-10-12345
(domain - server identifier - data change in asynchronous cluster)
```

MariaDB Cluster GTID

```
00295a79-9c48-11e2-bdf0-9a916cbb9294:64201
(data & cluster Identifier : data change in cluster)
```

LESSON **SUMMARY**

Describe the features of MariaDB Cluster

Explain Cluster Replication and its implementation in MariaDB Cluster

Understand the Global Transaction Identifier (GTID) in MariaDB Cluster

GETTING STARTED

MariaDB Training

LEARNING **OBJECTIVES**

Install MariaDB Cluster

Configure MariaDB Cluster server options

Initialize a new cluster and add new nodes

Plan for and perform an upgrade to MariaDB Cluster

Migrate to MariaDB Cluster

INSTALLATION

MariaDB Training

INSTALLING MARIADB CLUSTER

MariaDB's repository configuration tool is available for all major distributions.

https://downloads.mariadb.org/mariadb/repositories

On each database node, install:

MariaDB Server

MariaDB Backup

Extra packages for Linux

- epel-release
- socat
- pigz

If the system's package manager has automatically started the MariaDB server, shut it down.

```
# systemctl stop mariadb
```

CONFIGURATION

MariaDB Training

LINUX SERVER CONFIGURATION

OS Configuration

SELinux Runtime Configuration

```
# setenforce 0
# getenforce
Permissive
```

SELinux Persistent Configuration

```
# vi /etc/selinux/config
SELINUX=permissive
```

Set Start Timeout for systemd (CentOS 7+)

```
# vi
/etc/systemd/system/mariadb.service.d/timeout.conf
[Service]
TimeoutStartSec=2h
TimeoutStopSec=2h
# systemctl daemon-reload
```

OS Firewall Configuration

Ports

- 3306
  - Client connections to nodes
  - mariadb-dump connections between nodes
- 4567 – Replication protocol
- 4568 – Incremental State Transfer (IST)
- 4444 – State Snapshot Transfer (SST)

```
# systemctl stop firewalld
# systemctl disable firewalld
```

IST & SST methods may have additional connectivity requirements

MARIADB CLUSTER CONFIGURATION

On all nodes configure the MariaDB Cluster by editing the my.cnf file located in either /etc/my.cnf.d/server.cnf or /etc/my.cnf

The provider path is normally

/usr/lib64/galera-4
/usr/lib64/galera-enterprise

```
[mariadb]
datadir=<path-to-your-data-dir>
default_storage_engine=InnoDB
max_connections=1000
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
binlog_format=row

[galera]
wsrep_on=ON
wsrep_provider=<path-to-libgalera_smm.so>
wsrep_cluster_address="gcomm://<node-or-ip-1[:port]
>,<node-or-ip-1[:port]>,..."
# wsrep_auto_increment_control = ON
wsrep_cluster_name="<cluster-name>"
wsrep_sst_method=rsync
# wsrep_sst_auth=<backupuser>:<password>
# wsrep_node_name=<node-name>
wsrep_node_address=<ip address[:port]>
```

MARIADB CLUSTER CONFIGURATION

Additional configuration options can be added to the configuration file for optimization.

Optional Settings for Speed

For applying of write sets:

```
wsrep_slave_threads= <twice the no of CPUs>
```

For `COMMIT` operation:

- Reduce `I/O` on `COMMIT` to log_files
- Data is already persisted in Galera cache

```
innodb_flush_log_at_trx_commit=2
```

For SST:

- Use parallel compression and streamin

```
[sst]
streamfmt=xbstream
transferfmt=socat
compressor=pigz
decompressor="pigz -dc"
inno-apply-opts="--use-memory=1G"
```

NODE INITIALIZATION

MariaDB Training

START FIRST NODE

Starting the first node bootstraps the cluster and registers the cluster name and id.

On systemd operating systems use the **galera_new_cluster** wrapper script.

```
# galera_new_cluster
```

Use **SHOW STATUS** and **SHOW VARIABLES** to list Galera options.

```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_cluster_%';
+--------------------------+--------------------------------------+
| Variable_name            | Value                                |
+--------------------------+--------------------------------------+
| wsrep_cluster_conf_id    | 1                                    |
| wsrep_cluster_size       | 1                                    |
| wsrep_cluster_state_uuid | c414d03a-3d83-11e8-acaa-cf4bfafd35a4 |
| wsrep_cluster_status     | Primary                              |
+--------------------------+--------------------------------------+
4 rows in set (0.00 sec)
```

Create a MariaDB Backup user.

```
MariaDB [(none)]> CREATE USER 'backupuser'@'localhost' IDENTIFIED BY
'<password>';

MariaDB [(none)]> GRANT RELOAD, LOCK TABLES, REPLICATION CLIENT,
PROCESS ON *.* TO 'backupuser'@'localhost';
```

START OTHER NODES

1. Start the next node.

```
# systemctl start mariadb
```

2. Check the cluster status on the node.

```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_cluster_%';
+--------------------------+--------------------------------------+
| Variable_name            | Value                                |
+--------------------------+--------------------------------------+
| wsrep_cluster_conf_id    | 2                                    |
| wsrep_cluster_size       | 2                                    |
| wsrep_cluster_state_uuid | c414d03a-3d83-11e8-acaa-cf4bfafd35a4 |
| wsrep_cluster_status     | Primary                              |
+--------------------------+--------------------------------------+
4 rows in set (0.00 sec)
```

3. Repeat steps 1 & 2 for the other nodes.

CLUSTER INITIALIZATION

MariaDB Training

INITIAL CLUSTER STARTUP

All nodes should now be running and consistent.

Now test replication within the Cluster.

On Node 1:

```
MariaDB [(none)]> CREATE TABLE test.table1
(col1 INT UNSIGNED KEY);
INSERT INTO test.table1
VALUES (1),(2),(3);
```

On Node 2:

```
MariaDB [(none)]> SELECT * FROM test.table1;
INSERT INTO table1
VALUES (4);
```

On Node 3:

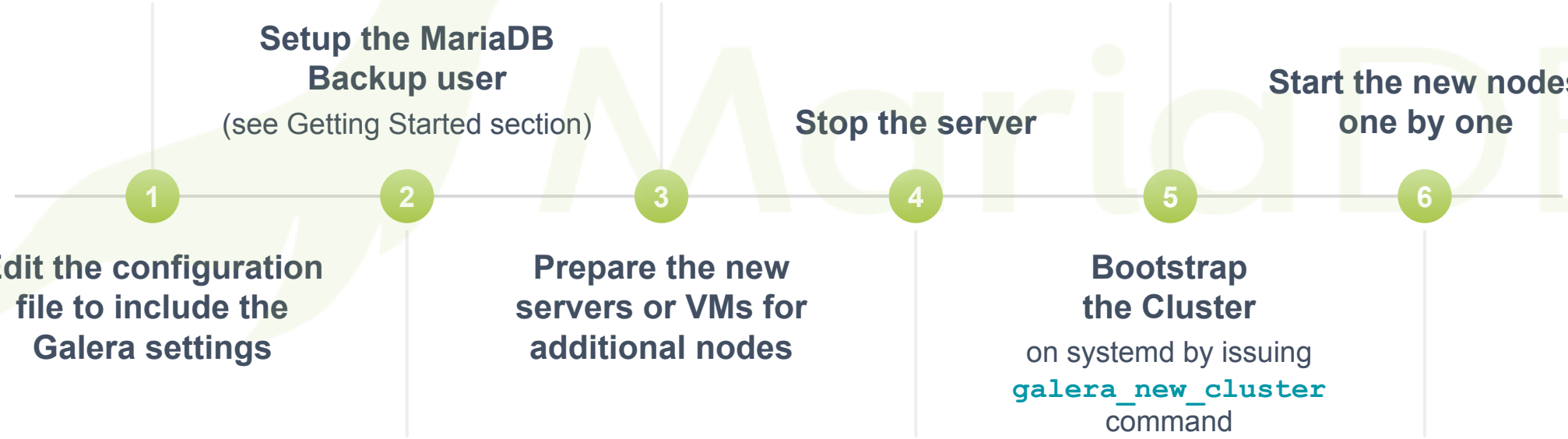
```
MariaDB [(none)]> SELECT * FROM test.table1;
```

MIGRATING TO MARIADB CLUSTER

MariaDB Training

STANDALONE TO CLUSTER

Using a standalone server to bootstrap a Cluster



PRIMARY – **REPLICA TO CLUSTER**

Converting a Replication Topology into a Cluster

If you have an existing Primary – Replica topology you can reuse the replicas as Galera nodes

Stop the replicas

Convert the primary to a bootstrap node like in the previous slide

Add the former replicas as additional nodes

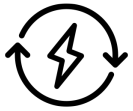
- They will perform a State Snapshot Transfer although they already have data but they miss the Galera GTIDs

UPGRADING MARIADB CLUSTER

MariaDB Training

PREPARING **FOR CLUSTER UPGRADE**

Steps to perform before upgrading the cluster



Check that your Galera Cache is large enough to cover the downtime needed per node.

A script to determine Galera Cache depth is shown in the Monitoring & Troubleshooting section

As always take a backup before upgrading and check that the backup is valid.

UPGRADING MARIADB CLUSTER

Performing the upgrade with a rolling restart

1. Stop the first node
2. Make sure the node has been stopped without errors
   - The file grastate.dat in the data directory must have a valid seqno, the last transaction seen before shutdown
3. Upgrade the node as you would for a standalone node
4. Start the node again and wait until it is synced
5. Run the `mysql_upgrade` script
6. Repeat steps 1-5 with the other nodes

If you use MaxScale set node to maintenance mode before starting the upgrade process and reset from maintenance only after mysql_upgrade was successful

LESSON **SUMMARY**

Install MariaDB Cluster

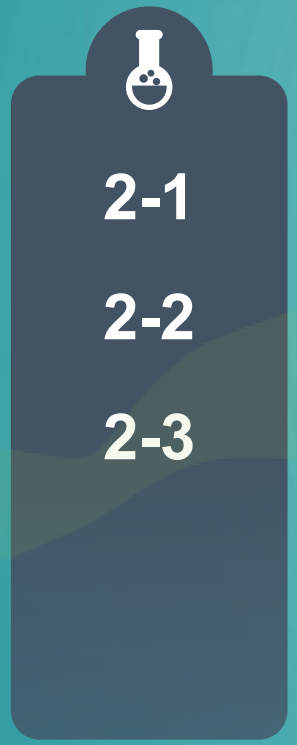
Configure MariaDB Cluster server options

Initialize a new cluster and add new nodes

Plan for and perform an upgrade to MariaDB Cluster

Migrate to MariaDB Cluster

LAB **EXERCISES**



2-1 Installing and Configuring MariaDB Enterprise Cluster

2-2 Testing MariaDB Enterprise Cluster Replication

2-3 Upgrading MariaDB Enterprise Cluster

STATE TRANSFERS

MariaDB Training

LEARNING **OBJECTIVES**

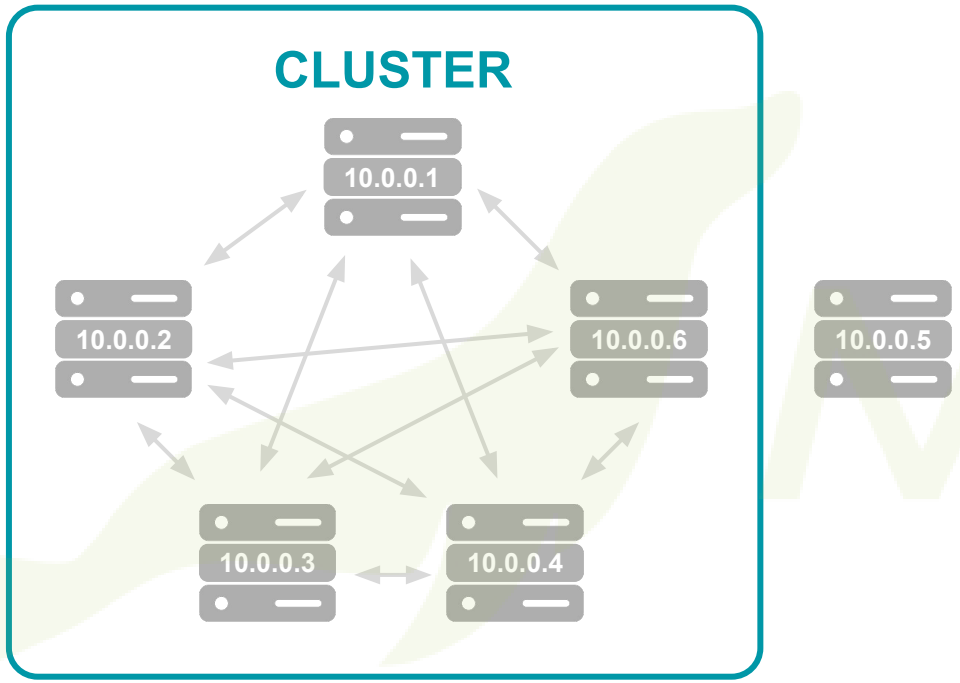
Gain an understanding of State Snapshot Transfers (SSTs)

How to force nodes to synchronize using SST

How to avoid using SST with new or desynchronized nodes

Gain an understanding of Incremental State Transfers (ISTs)

CLUSTER ADDRESS



wsrep_cluster_address - defines the IP addresses for the cluster in a comma separated list.

```
wsrep_cluster_address="gcomm://10.0.0.1,10.0.0.2,10.0.0.3,10.0.0.4,10.0.0.6"
```

WHEN A NEW NODE JOINS THE CLUSTER


```
wsrep_cluster_address="gcomm://10.0.0.1,10.0.0.2,10.0.0.3,10.0.0.4,10.0.0.6"
```

STATE SNAPSHOT TRANSFER (SST)

MariaDB Training

STATE SNAPSHOT TRANSFER (SST)

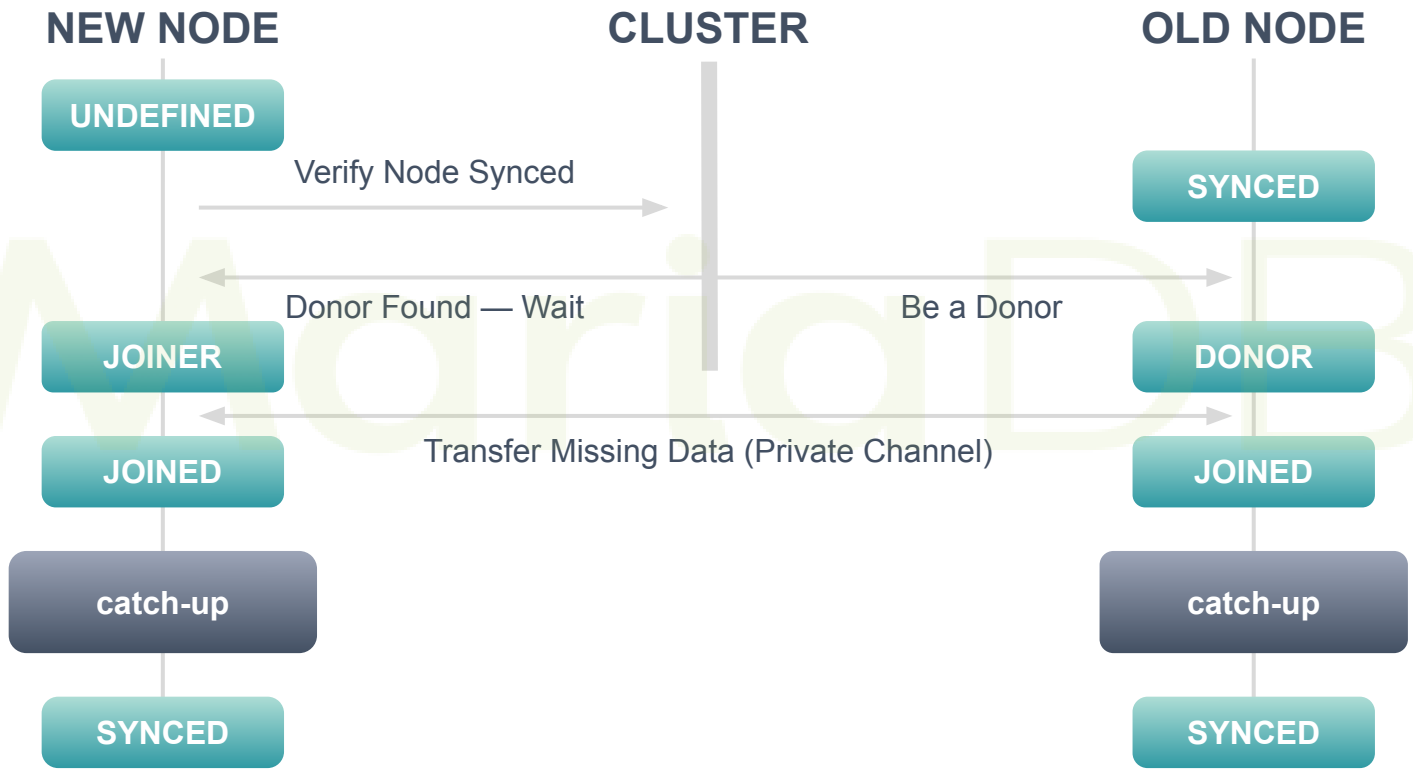
Used to send the full database state

New joining nodes use SST

An IST capable joining node falls back to SST if none of the active nodes still knows the latest GTID

Set `wsrep_sst_method` to configure the method to use

- `mariadb-dump`
- rsync (default)
- mariabackup (recommended)



FORCED SST

Expediting full recovery by forcing a node to synchronize via SST by:

- Stopping the MariaDB server
- Deleting all the files in the data directory
- Starting the MariaDB server

Stop MariaDB on one of the nodes

```
# systemctl stop mariadb
```

Delete all the files in the data directory on the node where MariaDB is shut down

```
# rm -rf /var/lib/mysql/*
```

Synchronize the node by starting up MariaDB

```
# systemctl start mariadb
# tail -f /var/lib/mysql/*.err
```

AVOIDING **SST**

Manually set a new or failed node to IST capability

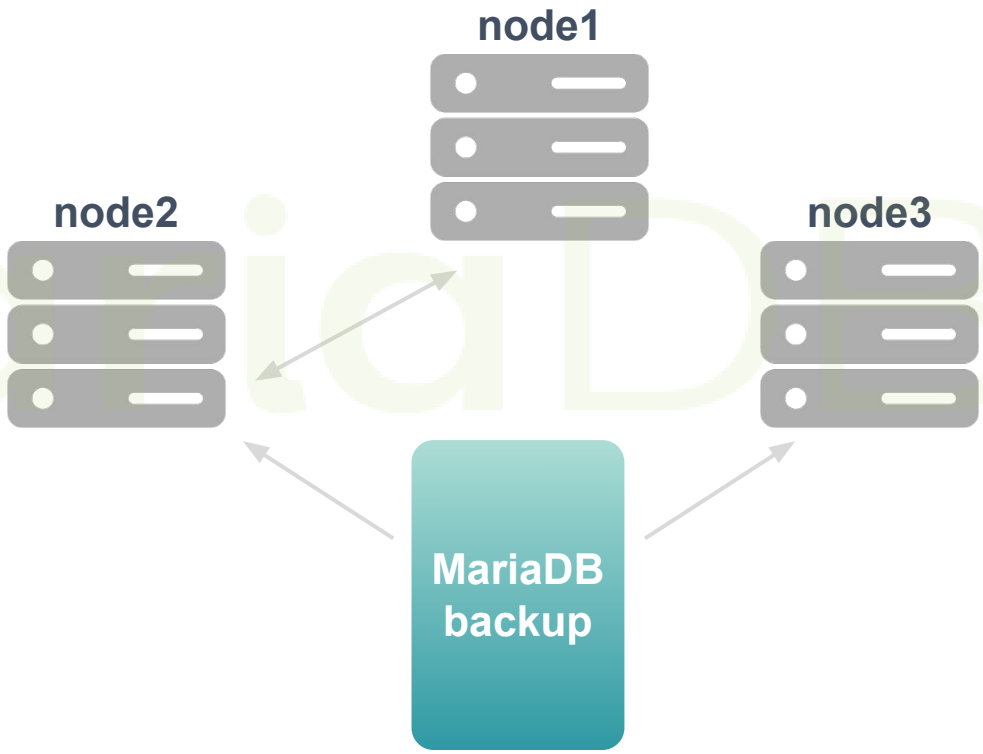
Check that the gcache.size is large enough to cover the time for:

- Taking a backup on one active node
- Copying the backup to a new or failed node
- Manually preparing a valid grastate.dat file on a node

```
# cat /var/lib/mysql/xtrabackup_galera_info
dc4bdb22-cc3b-11e6-9656-a65427fc7923:125879

# vi /var/lib/mysql/grastate.dat
version: 2.1
uuid:    dc4bdb22-cc3b-11e6-9656-a65427fc7923
seqno:   125879

# systemctl start mariadb
```



SST METHODS

MariaDB Cluster may perform SST in several ways

- **Logical SST:** a logical data dump and restore
- **Physical SST:** file-level backup and restore procedure

Logical SST uses mariadb-dump. This is the oldest and slowest method, which also blocks the donor for the duration of the dump.

Physical SST via mariabackup is non-blocking to the donor. A networking streaming helper is needed to automate the snapshot transfer (socat, netcat etc.)

Physical SST via rsync blocks the donor node only briefly; securing the rsync endpoint is a good practice.

INCREMENTAL STATE TRANSFER (IST)

MariaDB Training

INCREMENTAL STATE TRANSFER (IST)

Incremental State Transfer is very effective

`gcache.size` parameter defines size of write sets kept for IST

Gcache is mmap file

Available disk space is upper limit for size allocation

The JOINER node sends a broadcast with the GTID last seen. Active nodes check for that GTID in their GCache.

If GTID is still in Gcache, the Cluster decides which node assumes the DONOR role and sends all Write Sets with newer GTIDs to the JOINER

Pre-conditions

The `JOINER` must be in an IST capable state

- Database on `JOINER` must not be empty
- The status file `grastate.dat` in the data directory must exist, be readable and contain a valid group uuid and seqno → `GTID`

At least one node must have the last `GTID` seen by the `JOINER` in his GCache → `gcache.size`

GALERA SAVED STATE

`grastate.dat` is the replication state file

Resides in the data directory

Included in snapshots

```
# GALERA saved state
version: 2.1
uuid: c414d03a-3d83-11e8-acaa-cf4bfafd35a4
seqno:1234
safe_to_bootstrap: 0
```

`uuid` is the universal unique identifier

`seqno` is the sequence number of the last transaction

- A value greater than `0` indicates a proper shutdown
- A value of `-1` means there is no known sequence position and indicates that the server either crashed or is currently running

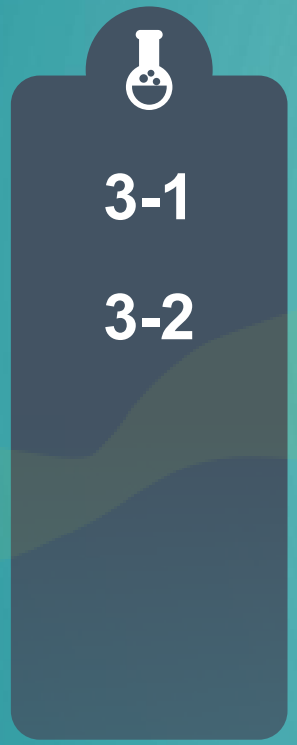
`safe_to_bootstrap` indicates whether the cluster thinks this node is ok to be bootstrapped

- A value of `0` means the node has crashed or was not the last running node at the time of shutdown

LESSON **SUMMARY**

- Gain an understanding of State Snapshot Transfers (SSTs)
- How to force nodes to synchronize using SST
- How to avoid using SST with new or desynchronized nodes
- Gain an understanding of Incremental State Transfers (ISTs)

LAB **EXERCISES**



3-1 Performing a State Snapshot Transfer (SST)

3-2 Performing an Incremental State Transfer (IST)

CLUSTER SPECIFICS

MariaDB Training

LEARNING **OBJECTIVES**

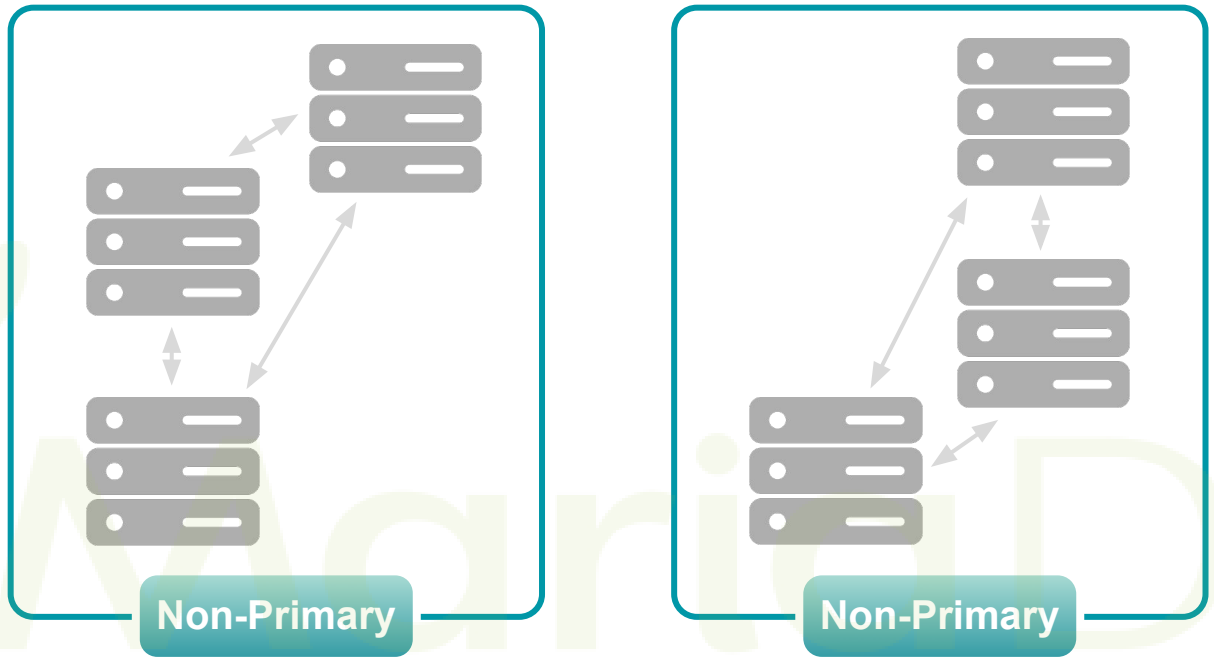
- Explain and avoid Split Brain scenarios
- Detect and tune slow nodes
- Activate consistent read settings
- Control the AUTO_INCREMENT attribute
- Understand true parallel replication

CLUSTER SPECIFICS

- Split Brain
- Primary Key is required
- Controlling `AUTO_INCREMENT`
- Causal Reads
- Flow Control
- Parallel Replication
- Streaming Replication
- XA Transactions (part of Galera 4)
- MyISAM Replication is experimental (do not use in production)

SPLIT BRAIN

Split Brain can occur with an even number of nodes



Distinguishing Server Crash from Network Failure in Shared Nothing Architecture

Decision Algorithm Used to Avoid Split Brain

Absolute Majority Needed in MariaDB Cluster (>50%)

Uneven Number of Nodes Safer

CONTROLLING **AUTO_INCREMENT**

MariaDB Cluster provides automatic control of AUTO_INCREMENT

```
# vi /etc/my.cnf.d/server.cnf
wsrep_auto_increment_control = ON
```

MariaDB Cluster will adjust auto_increment_increment and auto_increment_offset variables so INSERTs in separate nodes will interleave

AUTO_INCREMENT BEHAVIOR

Increments **AUTO_INCREMENT** by the number of nodes

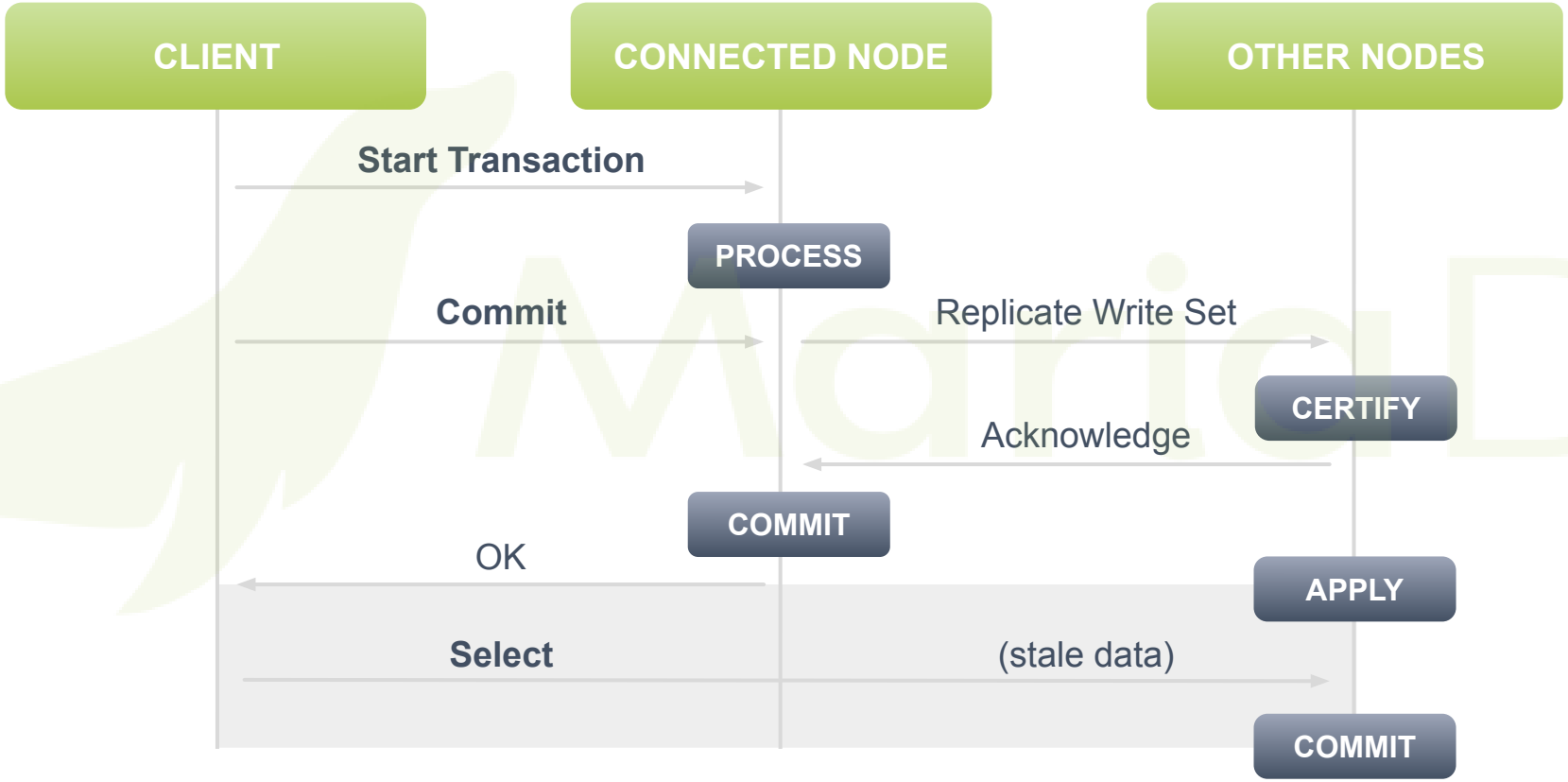
```
MariaDB [test]> SHOW VARIABLES LIKE
'%auto_increment%';
+------------------------------+-------+
| Variable_name                | Value |
+------------------------------+-------+
| auto_increment_increment     | 3     |
| auto_increment_offset        | 1     |
| wsrep_auto_increment_control | ON    |
+------------------------------+-------+


MariaDB [test]> CREATE TABLE t1
(c1 INT AUTO_INCREMENT PRIMARY KEY, c2 INT)
ENGINE=InnoDB;


MariaDB [test]> INSERT INTO t1(c2)
VALUES (1), (2), (3);
```

```
MariaDB [test]> SELECT * FROM t1;
+----+------+
| c1 | c2   |
+----+------+
|  1 |    1 |
|  4 |    2 |
|  7 |    3 |
+----+------+
```

CAUSAL READS



CAUSAL READS

Set

`wsrep_sync_wait = 1`

Every read (e.g., `SELECT`) waits for the slave queue to be fully applied

Set the timeout for maximum causal read wait

`repl.causal_read_timeout=PT90S`

Read locks only on local node

FLOW CONTROL

MariaDB Cluster implements Flow Control to protect cluster from node queue growth

Nodes should have equal capacity and use equivalent hardware

`wsrep_flow_control_sent` shows the number of times another node asked for flow control from the node

`wsrep_flow_control_recvd` shows the number of times the node received a flow control stop signal from other nodes

`wsrep_flow_control_paused` is the fraction of time the node had to pause for flow control; a value greater than `0.02` indicates a performance problem in cluster

`wsrep_local_recv_queue_avg` is the average size of the local received queue since the last status query; a value greater than `0.01` also indicates a performance problem in cluster

`wsrep_slave_threads`

- Number of parallel threads applying to slave queue
- Slave thread count = 1 to 3 x logical CPU cores

`gcs.fc_limit`

- High limit for Flow Control (default 16)
- Flow Control stops when reached
- 5 x wsrep_slave_threads

`gcs.fc_factor`

- Factor times limit is lower limit (default 1.0; recommended 0.8)
- Flow Control resumes when slave queue reduced to this level

TRUE PARALLEL REPLICATION

- Every application can benefit from a parallel replication
- Works on the row level not on the database or table level
- Assigns
- Set to the number of replicas that makes sense
- Only applying is done in parallel
- Commit order is dictated

For Best Configuration

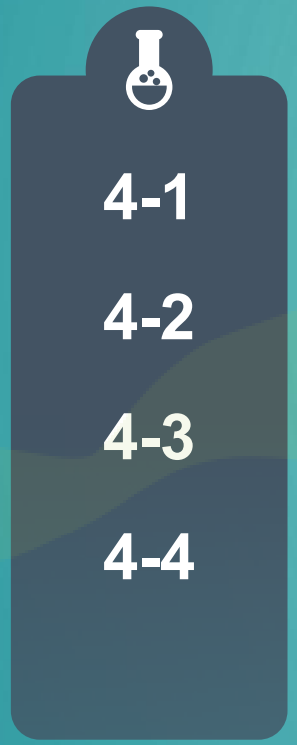
Check the number of cores the server has

Check the value of `wsrep_cert_deps_distance`

LESSON **SUMMARY**

- Explain and avoid Split Brain scenarios
- Detect and tune slow nodes
- Activate consistent read settings
- Control the AUTO_INCREMENT attribute
- Understand true parallel replication

LAB **EXERCISES**



4-1 Controlling AUTO_INCREMENT

4-2 Spotting Causality Failure

4-3 Configuring Flow Control

4-4 Configuring True Parallel Replication

BENCHMARKING

MariaDB Training

LEARNING **OBJECTIVES**

Check the performance of MariaDB Cluster using the sysbench benchmarking tool

Server variables that impact read/write performance

CLUSTER PERFORMANCE

Server variables that can impact the cluster's read/write performance

Performance versus cluster reliability and fault tolerance

Other considerations

`innodb_buffer_pool_size` is one of the most important parameters on the MariaDB server, used for buffering data into memory for faster access to the data.

`innodb_io_capacity` should be configured according to the storage IOPS, setting too high for a slow disk IO can become a bottleneck

`innodb_flush_method` This controls the InnoDB flushing method. Depends on filesystem and must be tested/benchmarked

`innodb_log_file_size` This defines the size of the InnoDB redolog files size. Larger values mean less disk I/O due to less flushing checkpoint activity, but also slower recovery from a crash

`innodb_flush_log_at_trx_commit` When to flush logs on a transaction commit. Set to "1" for highest database durability (ACID compliance) and maximum fault tolerance but impacts performance

`innodb_autoinc_lock_mode` only important if using auto increment columns. Setting it to 0 or 2 gives best performance but chances of data loss are higher in case of power failure.

SYSBENCH INSTALLATION

Install sysbench

```
# sudo yum -y install sysbench
```

Setting up the benchmark database and database user account

```
MariaDB [(none)]> create database sbtest;
MariaDB [(none)]> create user sbuser@localhost identified by 'secretpassword';
MariaDB [(none)]> grant all on sbtest.* to sbuser@localhost;
```

BENCHMARKING **WITH SYSBENCH**

Two stages

Prepare

Prepare one time based on the required data volume and number of tables

Run

Run multiple times, record results, perform server configuration changes, and run again

Run from multiple cluster nodes in parallel to and evaluate the results

Prepare the benchmark

```
# sysbench --db-driver=mysql --oltp-tables-count=10
   --oltp-table-size=100000
   /usr/share/sysbench/tests/include/oltp_legacy/oltp.lua
   --mysql-host=<Node1-IP> --mysql-port=3306
   --mysql-user=sbuser --mysql-password=secretpassword
   prepare
```

Run the benchmark

```
# sysbench --db-driver=mysql --threads=4 --events=250000
   --oltp-tables-count=12 --oltp-table-size=100000
   --oltp-test-mode=complex --oltp-dist-type=uniform
   /usr/share/sysbench/tests/include/oltp_legacy/oltp.lua
   --mysql-host=<Node(n)-IP> --mysql-port=3306
   --mysql-user=sb_user --mysql-password=secretpassword
   --time=120 --report-interval=10 run
```

SYSBENCH OUTPUT – PREPARE

Prepare output will the table names that were created "`--oltp-tables-count`" and populated with data "`--oltp-table-size`"

```
# sysbench --db-driver=mysql --oltp-tables-count=10
  --oltp-table-size=100000
  /usr/share/sysbench/tests/include/oltp_legacy/oltp.lua
  --mysql-host=<Node1-IP> --mysql-port=3306
  --mysql-user=sbuser --mysql-password=secretpassword
  prepare

Creating table 'sbtest1'...
Inserting 100000 records into 'sbtest1'
Creating secondary indexes on 'sbtest1'...
Creating table 'sbtest2'...
Inserting 100000 records into 'sbtest2'
Creating secondary indexes on 'sbtest2'...
…
…
…
Creating table 'sbtest10'...
Inserting 100000 records into 'sbtest10'
Creating secondary indexes on 'sbtest10'...
```

SYSBENCH OUTPUT – RUN

The `run` argument tells sysbench to generate the load based on the configuration

Generate a report every `report-interval` seconds

At the end a summary is printed out

```
# sysbench --db-driver=mysql --threads=4 --events=250000 --oltp-tables-count=12
   --oltp-table-size=100000 --oltp-test-mode=complex --oltp-dist-type=uniform
   /usr/share/sysbench/tests/include/oltp_legacy/oltp.lua
   --mysql-host=<Node(n)-IP> --mysql-port=3306 --mysql-user=sb_user
   --mysql-password=secretpassword --time=60 --report-interval=10 run

Running the test with following options:
Number of threads: 4
Report intermediate results every 10 second(s)
Initializing random number generator from current time

Initializing worker threads...
Threads started!

[ 10s ] thds: 4 tps: 727.49 qps: 14555.75 (r/w/o: 10189.90/2910.47/1455.39) lat
   (ms,95%): 7.04 err/s: 0.00 reconn/s: 0.00
[ 20s ] thds: 4 tps: 749.47 qps: 14990.92 (r/w/o: 10493.20/2998.78/1498.94) lat
   (ms,95%): 6.55 err/s: 0.00 reconn/s: 0.00
[ 30s ] thds: 4 tps: 740.00 qps: 14798.69 (r/w/o: 10359.67/2959.02/1480.01) lat
   (ms,95%): 6.55 err/s: 0.00 reconn/s: 0.00
…
SQL statistics:
    queries performed:
        read:                            619094
        write:                           176881
        other:                           88441
        total:                           884416
    transactions:                        44220  (736.91 per sec.)
    queries:                             884416 (14738.53 per sec.)
    ignored errors:                      1      (0.02 per sec.)
    reconnects:                          0      (0.00 per sec.)
```

SYSBENCH OUTPUT – CLEANUP

The `cleanup` argument tells sysbench to remove any data that was created during the load test

```
# sysbench --db-driver=mysql --mysql-host=<Node1-IP>
--mysql-port=3306 --mysql-user=sbuser
--mysql-password=secretpassword cleanup

 Dropping table 'sbtest1'...
 Dropping table 'sbtest2'...
 Dropping table 'sbtest3'...
 Dropping table 'sbtest4'...
```

LESSON **SUMMARY**

Check the performance of MariaDB Cluster using the sysbench benchmarking tool

Server variables that impact read/write performance

LAB **EXERCISES**

5-1 Benchmarking Cluster Performance

SECURITY

MariaDB Training

LEARNING **OBJECTIVES**

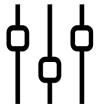
How to secure MariaDB Cluster with Data-In-Transit Encryption

Understand Data at Rest Encryption in MariaDB Cluster

FIREWALL SETTINGS

MariaDB Training

FIREWALL SETTINGS



MariaDB Cluster requires several **network ports** to maintain network connectivity between the nodes

Standard MariaDB Port (default: 3306)

```
# State Snapshot Transfers using mariadb-dump
port=3306
```

Galera Replication Port (default: 4567)

```
# MariaDB Cluster replication traffic.
# Multicast replication uses both TCP & UDP
wsrep_node_address=ip_address[:port]
```

Incremental State Transfer Port (default: 4568)

```
wsrep_provider_options="ist.recv_addr=ip_address:port"
```

State Snapshot Transfer Port (default: 4444)

```
wsrep_sst_receive_address=ip_address:port
```

FIREWALL CONFIGURATION – IPTABLES

LAN Configuration

```
# iptables --append INPUT --in-interface eth0 \
   --protocol tcp --match tcp --dport 3306 \
   --source 192.168.0.1/24 --jump ACCEPT

# iptables --append INPUT --in-interface eth0 \
   --protocol tcp --match tcp --dport 4567 \
   --source 192.168.0.1/24 --jump ACCEPT

# iptables --append INPUT --in-interface eth0 \
   --protocol tcp --match tcp --dport 4568 \
   --source 192.168.0.1/24 --jump ACCEPT

# iptables --append INPUT --in-interface eth0 \
   --protocol tcp --match tcp --dport 4444 \
   --source 192.168.0.1/24 --jump ACCEPT

# iptables --append INPUT --in-interface eth0 \
   --protocol udp --match udp --dport 4567 \
   --source 192.168.0.1/24 --jump ACCEPT
```

WAN Configuration

```
# iptables --append INPUT --protocol tcp \
    --source 64.57.102.34 --jump ACCEPT
# iptables --append INPUT --protocol tcp \
    --source 193.166.3.20  --jump ACCEPT
# iptables --append INPUT --protocol tcp \
    --source 193.125.4.10  --jump ACCEPT
```

Firewall Persistent Configuration (init)

```
# service iptables save
```

Firewall Persistent Configuration (systemd)

```
# vi /etc/sysconfig/iptables
or
# vi /etc/iptables/iptables.rules
# iptables-save > /etc/sysconfig/iptables.rules
```

FIREWALL CONFIGURATION – FIREWALLD

Enable MariaDB service for FirewallD

```
# firewall-cmd --zone=public
--add-service=mariadb
```

Open the TCP ports for Cluster

```
# firewall-cmd --zone=public --add-port=3306/tcp
# firewall-cmd --zone=public --add-port=4567/tcp
# firewall-cmd --zone=public --add-port=4568/tcp
# firewall-cmd --zone=public --add-port=4444/tcp
# firewall-cmd --zone=public --add-port=4567/udp
```

Persistent Configuration

```
# firewall-cmd --zone=public --add-service=mariadb
\
     --permanent
# firewall-cmd --zone=public --add-port=3306/tcp
--permanent
# firewall-cmd --zone=public --add-port=4567/tcp
--permanent
# firewall-cmd --zone=public --add-port=4568/tcp
--permanent
# firewall-cmd --zone=public --add-port=4444/tcp
--permanent
# firewall-cmd --zone=public --add-port=4567/udp
--permanent
# firewall-cmd --reload
```

KERNEL-LEVEL PROTECTION

MariaDB Training

KERNEL-LEVEL **PROTECTION FRAMEWORKS**

Security Enhanced Linux

Found typically on RHEL kernels and derivatives

Policy-based framework which governs access to file-based resources (including network port, sockets, directories) and system calls

Based on labels

Most MariaDB products come with a proper policy bundled, including MariaDB Cluster

Manual changes to filesystem locations require manual policy adjustments (e.g., moving the data directory)

AppArmor

Found typically on Debian Linux kernels and derivatives

Policy-based framework which governs access to file-based resources (including network port, sockets, directories) and system calls

Based on paths

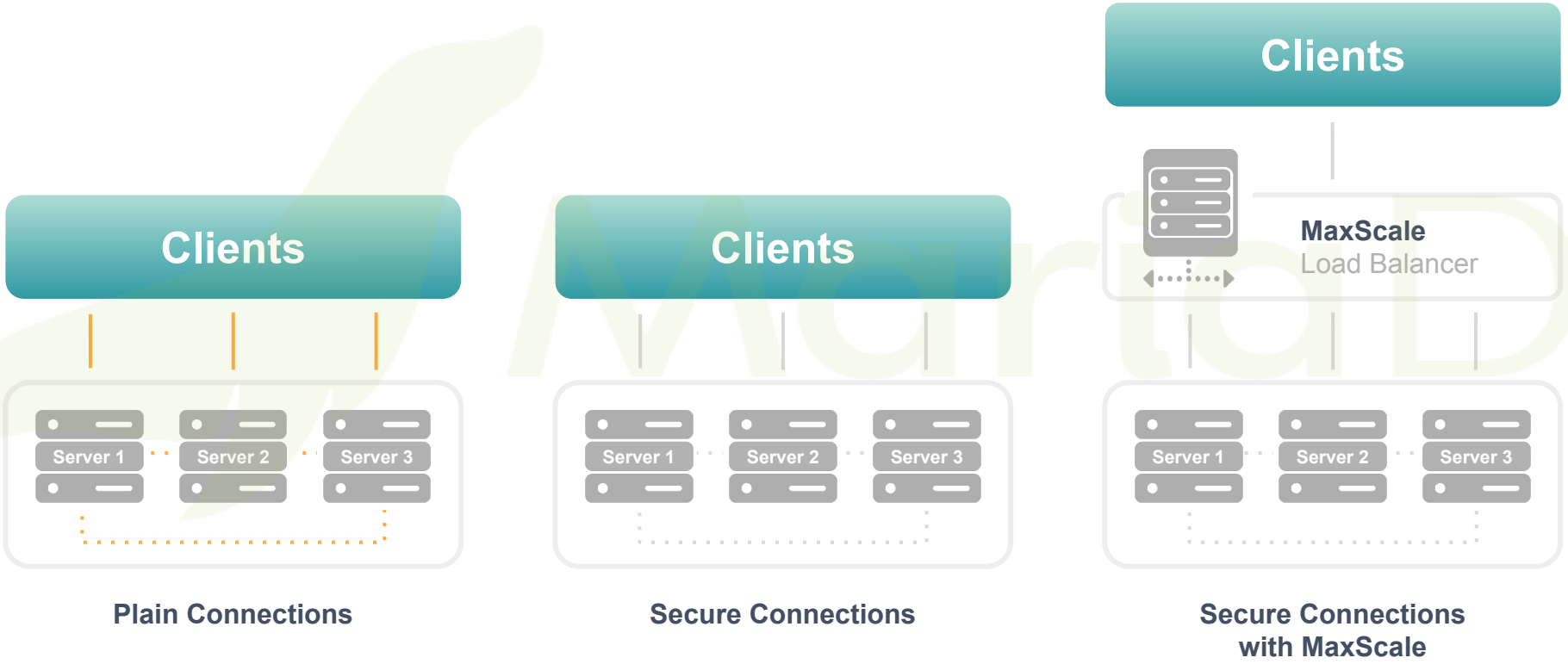
Most MariaDB products come with a proper policy bundled, including MariaDB Cluster

Manual changes to filesystem locations require manual policy adjustments (e.g., moving the data directory)

DATA-IN-TRANSIT ENCRYPTION

MariaDB Training

DATA-IN-TRANSIT ENCRYPTION REVIEW



CONFIGURING DATA-IN-TRANSIT ENCRYPTION

Enabling Transport Layer Security (TLS)

1) Securing the Database

```
# MariaDB Server
[mariadb]
ssl-ca = /path/to/ca-cert.pem
ssl-key = /path/to/server-key.pem
ssl-cert = /path/to/server-cert.pem

# MariaDB Client Configuration
[client-mariadb]
ssl-ca = /path/to/ca-cert.pem
ssl-key = /path/to/client-key.pem
ssl-cert = /path/to/client-cert.pem
```

2) Securing Replication Traffic

```
# vi /etc/my.cnf.d/server.cnf
...
wsrep_provider_options="socket.ssl_key=/path/to
/server-key.pem;socket.ssl_cert=/path/to/server
-cert.pem;socket.ssl_ca=/path/to/cacert.pem"
```

CONFIGURING DATA-IN-TRANSIT ENCRYPTION

Enabling Transport Layer Security (TLS)

3) State Snapshot Transfer

```
# mariabackup/xtrabackup
[sst]
encrypt=2
tca=/etc/my.cnf.d/certificates/sst.crt
tcert=/etc/my.cnf.d/certificates/sst.pem
```

```
# rsync
[sst]
tkey = /etc/my.cnf.d/certificates/client-key.pem
tcert = /etc/my.cnf.d/certificates/client-cert.pem
```

```
# mariabackup/xtrabackup
[sst]
encrypt=3
tkey=/etc/my.cnf.d/certificates/server1-key.pem
tcert=/etc/my.cnf.d/certificates/server1-cert.pem
```

DATA AT REST ENCRYPTION

MariaDB Training

DATA AT REST **ENCRYPTION REVIEW**

Only **data** and only **at rest** data is encrypted

Data on the wire needs to be encrypted using secure connections

Only the MariaDB server knows how to decrypt the data

MariaDB Backup can back up an encrypted database

mariadb-binlog can read encrypted binary logs only when `--read-from-remote-server` is used

The disk-based Galera gcache can now be encrypted (Enterprise Server 10.4+)

CONFIGURING **DATA AT REST ENCRYPTION**

Encrypting InnoDB Table Data

```
[mariadb]
# File Key Management
plugin_load_add                          = file_key_management
file_key_management_filename             = /etc/mysql/encryption/keyfile.enc
file_key_management_filekey              = FILE:/etc/mysql/encryption/keyfile.key
file_key_management_encryption_algorithm  = AES_CTR

# InnoDB Encryption
innodb_encrypt_tables                    = ON
innodb_encrypt_temporary_tables          = ON
innodb_encrypt_log                       = ON
innodb_encryption_threads                = 4
innodb_encryption_rotate_key_age         = 1
```

CONFIGURING **DATA AT REST ENCRYPTION**

The disk-based Cluster gcache

is not encrypted in the community version of MariaDB Server. However this file can be encrypted in MariaDB Enterprise Server

Encrypting GCache

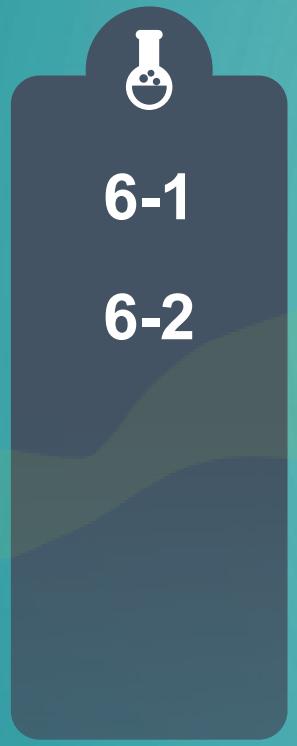
```
[mariadb]
...
# Controls Binary Log, Relay Log, and GCache Encryption
encrypt_binlog=ON
```

LESSON **SUMMARY**

How to secure MariaDB Cluster with Data-In-Transit Encryption

Understand Data at Rest Encryption in MariaDB Cluster

LAB **EXERCISES**



6-1 Setting Up Data-in-Transit Encryption

6-2 Setting Up Data-at-Rest Encryption

MONITORING AND TROUBLESHOOTING

MariaDB Training

LEARNING **OBJECTIVES**

- How to monitor the Cluster's status
- How to troubleshoot replication performance and when to use streaming replication
- How to analyze and resolve conflicts
- Gain an understanding of key performance indicators
- How to recover from node failure

DATABASE LOGS

MariaDB Training

LOG **VARIABLES**

Cluster replication can be monitored by configuring three variables (`wsrep_log_conflicts`, `wsrep_debug` and `cert.log_conflicts`) in the server configuration file

```
# Cluster replication logging
wsrep_log_conflicts=ON
wsrep_provider_options="cert.log_conflicts=ON"
wsrep_debug=ON
```

- `wsrep_log_conflicts` logs conflicts
- `cert.log_conflicts` logs certification failures
- `wsrep_debug` logs debug information

LOG **FILES**

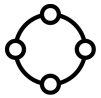
Same log files as MariaDB Server

Error log (enabled by default): host_name.err in datadir

General log (disabled by default): general_log for enable and general_log_file for file path are responsible variables

Slow query log (disabled by default): If enabled then slow queries above given threshold will be logged

SQL Error log and Audit log are available as plugins



Extra files if nodes fails on replication - GRA_*.log in datadir

STATUS VARIABLES

MariaDB Training

KEY PERFORMANCE INDICATORS

These are a few of the MariaDB Cluster specific status variables that are key performance indicators

Cluster status / integrity

Certification

Write set send/receive queue status

Write set application order

Flow Control activation

MONITORING CLUSTER STATUS

```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_%';
```

Cluster Integrity

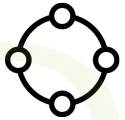
- wsrep_cluster_state_uuid
- wsrep_cluster_conf_id
- wsrep_cluster_size
- wsrep_cluster_status

Node Status

- wsrep_ready
- wsrep_connected
- wsrep_local_state_comment

Replication Health

- wsrep_local_recv_queue_avg
- wsrep_flow_control_paused
- wsrep_cert_deps_distance



Slow Network Issues

- wsrep_local_send_queue_avg

CHECKING **CLUSTER INTEGRITY**

```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_cluster_state_uuid' \G
*************************** 1. row ***************************
Variable_name: wsrep_cluster_state_uuid
        Value: 50b28ddf-b551-11ea-95f3-0e0c91ee2043

MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_cluster_conf_id' \G
*************************** 1. row ***************************
Variable_name: wsrep_cluster_conf_id
        Value: 11

MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_cluster_size' \G
*************************** 1. row ***************************
Variable_name: wsrep_cluster_size
        Value: 3

MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_cluster_status'\G
*************************** 1. row ***************************
Variable_name: wsrep_cluster_status
        Value: Primary
```

CHECKING **THE NODE STATUS**

```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_ready' \G
*************************** 1. row ***************************
Variable_name: wsrep_ready
        Value: ON

MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_connected' \G
*************************** 1. row ***************************
Variable_name: wsrep_connected
        Value: ON

MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_local_state_comment' \G
*************************** 1. row ***************************
Variable_name: wsrep_local_state_comment
        Value: Synced
```

CHECKING THE REPLICATION HEALTH

```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_local_recv_queue_avg'
\G
*************************** 1. row ***************************
Variable_name: wsrep_local_recv_queue_avg
        Value: 0.0058

MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_flow_control_paused' \G
*************************** 1. row ***************************
Variable_name: wsrep_flow_control_paused
        Value: 0.0144294

MariaDB [(none)]> SHOW GLOBAL STATUS LIKE 'wsrep_cert_deps_distance' \G
*************************** 1. row ***************************
Variable_name: wsrep_cert_deps_distance
        Value: 32.6937
```

DETECTING **SLOW NETWORK ISSUES**

```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE
'wsrep_local_send_queue_avg' \G
*************************** 1. row ***************************
Variable_name: wsrep_local_send_queue_avg
        Value: 0.130000
```

DETERMINING THE DEPTH **OF THE GALERA CACHE**

How long the oldest write set will persist in the gcache

Run this script on a regular basis to gather values from different load situations

```
set @start :=
     (select sum(VARIABLE_VALUE/1024/1024)from
information_schema.global_status
                    where VARIABLE_NAME like 'WSREP%bytes');
do sleep(60);
set @end :=
     (select sum(VARIABLE_VALUE/1024/1024)from
information_schema.global_status
                    where VARIABLE_NAME like 'WSREP%bytes');
set @gcache :=
     (select
SUBSTRING_INDEX(SUBSTRING_INDEX(@@GLOBAL.wsrep_provider_option
s,
                'gcache.size = ',-1), ';', 1));
set @gcache_size := @gcache * 1;
set @gcache_size_factor := if(right(@gcache,1)="M",1,1024);
set @difference := round((@end - @start),2);
select
     @difference as `MB/min`,
     @difference * 60 as `MB/hour`,
     @gcache as `gcache Size`,
     if(@difference = 0,0,
round(@gcache_size*@gcache_size_factor/@difference))
     as `Time to full(minutes)`;
```

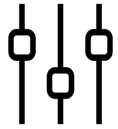
REPLICATION PERFORMANCE

MariaDB Training

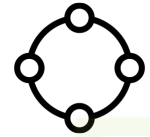
GROUP **COMMIT**

Introduced in MariaDB Server 5.3

Better performance for transactions-per-second (TPS)



Optimize expensive disk operations - fsync(), fdatasync()



MariaDB Cluster (Galera 4) integrates group commit

LARGE TRANSACTIONS

e.g. Delete/ Update millions of rows

Could affect cluster performance

Set `innodb_buffer_pool_size` to 80% of total memory

Less use of the disk → better performance

WRITE-SET CACHE SIZE

Set **gcache.size** as large as data directory

Extend the validity period of write-sets

Improve performance

WRITE-SET CACHING **DURING STATE TRANSFERS**

- **Much more memory is used then usual**
- **During IST higher gcache.size is recommended**
- **State snapshot is more efficient if write-set cache is greater than database size**

SLOW **APPLICATION OF WRITESETS**

Adjusting MariaDB Cluster settings online for faster applying of writesets

`wsrep_local_recv_queue_avg` > 0.01

or

`wsrep_flow_control_paused` > 0.001

then increase

`wsrep_slave_threads`

Recommended Values:

2 x number of CPUs <= `wsrep_slave_threads` <= 4 x number of CPUs

REPLICATION **THROTTLING BY SENDER**

Adjusting MariaDB Cluster settings online for faster applying of Write Sets

`wsrep_local_send_queue_avg` > 0.001
and
`wsrep_local_recv_queue_avg`
on other nodes ok

Check For Network Problems

DETECTING SLOW NODES

```
SELECT * FROM information_schema.GLOBAL_STATUS

WHERE VARIABLE_NAME LIKE 'wsrep_flow_control_sent'

OR VARIABLE_NAME LIKE 'wsrep_local_recv_queue_avg';
```

STREAMING REPLICATION

MariaDB Training

USING STREAMING REPLICATION

- Streaming replications breaks a transaction into smaller chunks
- Each chunk is sent to peers as soon as available
- As as a result, maximum transaction size limitation is removed
- Rollback also becomes lighter on the cluster as it may be initiated before the transaction is received in full

CONFIGURING STREAMING REPLICATION

Two parameters are needed to enable streaming replication:

- `wsrep_trx_fragment_unit` defines the units in which the chunk is measured - bytes, rows or statements.
- `wsrep_trx_fragment_size` defines the count of items in each chunk.

To set the fragment is set to three statements:

```
SET SESSION wsrep_trx_fragment_unit='statements';
SET SESSION wsrep_trx_fragment_size=3;
```

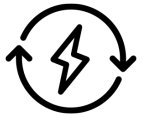
CONFLICT RESOLUTION

MariaDB Training

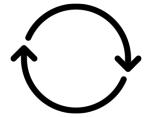
MULTI-PRIMARY CONFLICTS

Galera uses Optimistic Concurrency Control

When two transactions modify same row on different nodes at the same time, one transaction must abort



The aborted transaction will receive the Deadlock Error (1213) upon `COMMIT`



The application should retry deadlocked transactions

Not all applications have retrying logic built-in

...Or Use MaxScale!

DEADLOCK FOUND

Galera Cluster uses optimistic row locking
(as opposed to pessimistic locking used by MariaDB Server)

A deadlock will be resolved automatically by rolling back one of the transactions; the connected client will be notified about the rollback

Applications should be ready to handle
such conflicts, e.g. by retrying the transaction

Writing to multiple nodes may cause more deadlocks
than when writing to a single node

RETRYING **TRANSACTIONS**

MariaDB Cluster can retry an autocommit transaction on behalf of the client application, within the MariaDB server

MariaDB will not return the Deadlock Error but instead MariaDB will silently retry the transaction

Set `wsrep_retry_autocommit` to the number of retries MariaDB should attempt before returning the Deadlock Error

Retrying applies **only** to `autocommit` transactions since retrying is not safe for multi-statement transactions

DIAGNOSTICS

Look for database hot-spots such as: rows to which many transactions want to write simultaneously, or patterns like Queue or ID Allocation.

```
# tail -f
/var/lib/mysql/node1.err
```

Set `wsrep_log_conflicts` to send information on each cluster conflict to the MariaDB error log

```
MariaDB [(none)]> SET GLOBAL wsrep_log_conflicts=1;
```

Use `cert.log_conflicts` to log conflicting transactions

```
MariaDB [(none)]> SET GLOBAL
wsrep_provider_options="cert.log_conflicts=1";
```

Monitor the following status variables:

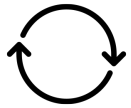
`wsrep_local_bf_aborts`

`wsrep_local_cert_failures`

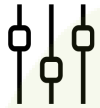
```
MariaDB [(none)]> SHOW LOCAL STATUS LIKE 'wsrep_local%';
```

Set `wsrep_debug` so all conflicts will be logged

ANALYZING **CONFLICTS**



Check if the application's logic can be changed to catch a deadlock exception and apply retrying logic in the application



Try setting `wsrep_retry_autocommit`

Limit the number of primary nodes or change to a primary-replica model

Treat writes only to hot-spot table as primary-replica, if can limit access to table

...Or Use MaxScale!

NODE FAILURE AND RECOVERY

MariaDB Training

MAINTENANCE **OR FAILURE USE CASES**

Maintenance Use Cases

Node stops

- Upgrade

Node becomes desynchronized

- Rolling schema upgrade

Failure Use Cases

Hardware failure

Software crash

Network errors

Failure of a state transfer

DETECTING SINGLE NODE FAILURES

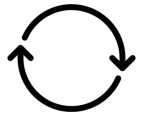
Various reasons – e.g. hardware failure, software crash, loss of connectivity

Node is failed when lose membership in Primary component

`wsrep_provider_options` options for node connectivite:

- evs.inactive_check_period
- evs.keepalive_period
- evs.suspect_timeout
- evs.inactive_timeout

RECOVERING FROM SINGLE NODE FAILURES



Incremental State Transfers (IST)

Partial recovery (deltas) of write sets

State Snapshot Transfer (SST)

Full recovery of data

Error Log

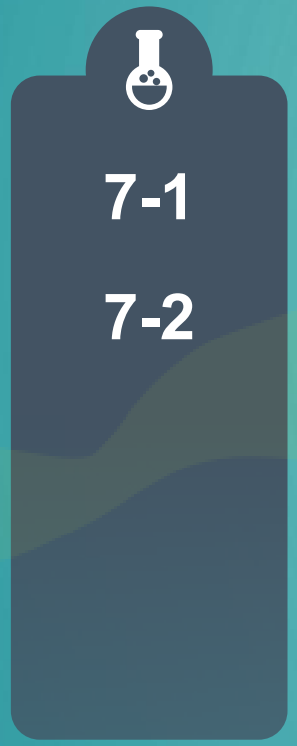
Shows method used and outcome. Check the error logs on both the joiner and donor nodes

Single node failures can also occur when a state snapshot transfer fails

LESSON **SUMMARY**

- How to monitor the Cluster's status
- How to troubleshoot replication performance and when to use streaming replication
- How to analyze and resolve conflicts
- Gain an understanding of key performance indicators
- How to recover from node failure

LAB **EXERCISES**



7-1 Creating and Observing Ad-Hoc Conflict

7-2 Monitoring the Galera Cache

MAINTENANCE

MariaDB Training

LEARNING **OBJECTIVES**

Gain an understanding of schema upgrade methods and best practices

How to backup MariaDB Cluster

How to explain MariaDB Cluster's recovery process

SCHEMA UPGRADES

MariaDB Training

ONLINE AND **OFFLINE** DDL

Offline DDL

Traditional DDL method

The altered resource is locked and rewritten with the updated schema

The more data in the resource, the slower the DDL

The only available method for most storage engines

Online DDL

A new method for applying some types of DDL statements

The alteration is done in-place without the need to rewrite the resource

Much faster than offline DDL (in practice, instantaneous)

Only available on InnoDB storage engine

Only for certain types of DDL

SCHEMA UPGRADES

DDL is non-transactional

Requires caution when mixing DDL and DML

MariaDB Cluster has two modes for DDL statements

Total Order Isolation (TOI)

Rolling Schema Upgrade (RSU)

Online Schema Change (OSC)

Use `wsrep_osu_method` to choose either upgrade method

TOTAL ORDER ISOLATION (TOI)

DDL is replicated up-front

Each node gets the DDL statement and must process the DDL at the same slot in the transaction stream

MariaDB Cluster isolates and locks the entire cluster for the duration of DDL processing

If the DDL is not an online one and the tablespace file is big, this may block the cluster for a prolonged period of time

This could cause a timeout for the application depending on how long the DDL processing takes

ROLLING SCHEMA UPGRADE (RSU)

DDL is not replicated but is run on one node at a time, and the remaining nodes operate as usual

MariaDB Cluster will remove the node from replication for the duration of DDL processing, but the clients should not connect and use this node

When done with DDL, node will catch up with missed transactions (IST)

- Galera cache must be large enough to cover the duration of DDL processing +50%

The DBA needs to roll the RSU operation over all nodes

Requires backward compatible schema changes

Use only under certain conditions

- Planned SQL is not conflicting
- SQL will not generate inconsistency

ONLINE SCHEMA CHANGE (OSC)

Works in TOI mode without blocking the cluster

A copy of the table's structure is made and the DDL is applied

Triggers are installed on the original table to replicate all writes to it onto the new table

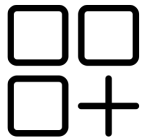
Background copy of the data from the original table to the new one is executed

Once tables are fully synchronized, they are atomically renamed and the old tables are dropped

An external script is used to implement and orchestrate the workflow. This often means manual application of the DDL

Any existing FK will be renamed (with an underscore prepended to its name) as InnoDB does not allow neither to have two FK inside the same schema with the same name nor to rename a FK

SCHEMA UPGRADE STRATEGY



Best Practices

Plan upgrades

Try to make upgrades backwards compatible

Practice upgrades

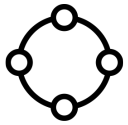
Determine DDL execution time

Use RSU if possible

`ALTER TABLE` to create new `AUTO_INCREMENT` column will cause problems

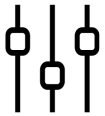
Every node has different `AUTO_INCREMENT` and offset settings

CLUSTER STALLS ON ALTER TABLE



Monitor closely the DDL execution.

Be prepared to act in case something goes wrong.



In most cases patience is the best strategy.

Make sure you have backups available should you need to terminate a DDL manually and recover the cluster.

USER CHANGES NOT REPLICATING

The service schemas like *mysql* are not replicated.

Do not do any changes there by manually inserting or updating rows.

Always use proper DML like CREATE USER or ALTER USER, which will be replicated by the cluster.

BACKING UP MARIADB CLUSTER

MariaDB Training

BACKING UP MARIADB CLUSTER

Best Practices

Dedicate a reference node for backups

Assign the Global Transaction Identifier (GTID) with the backup

All MariaDB Cluster nodes are continuously up-to-date

For consistent chronological binary logs set `log_slave_updates = ON`

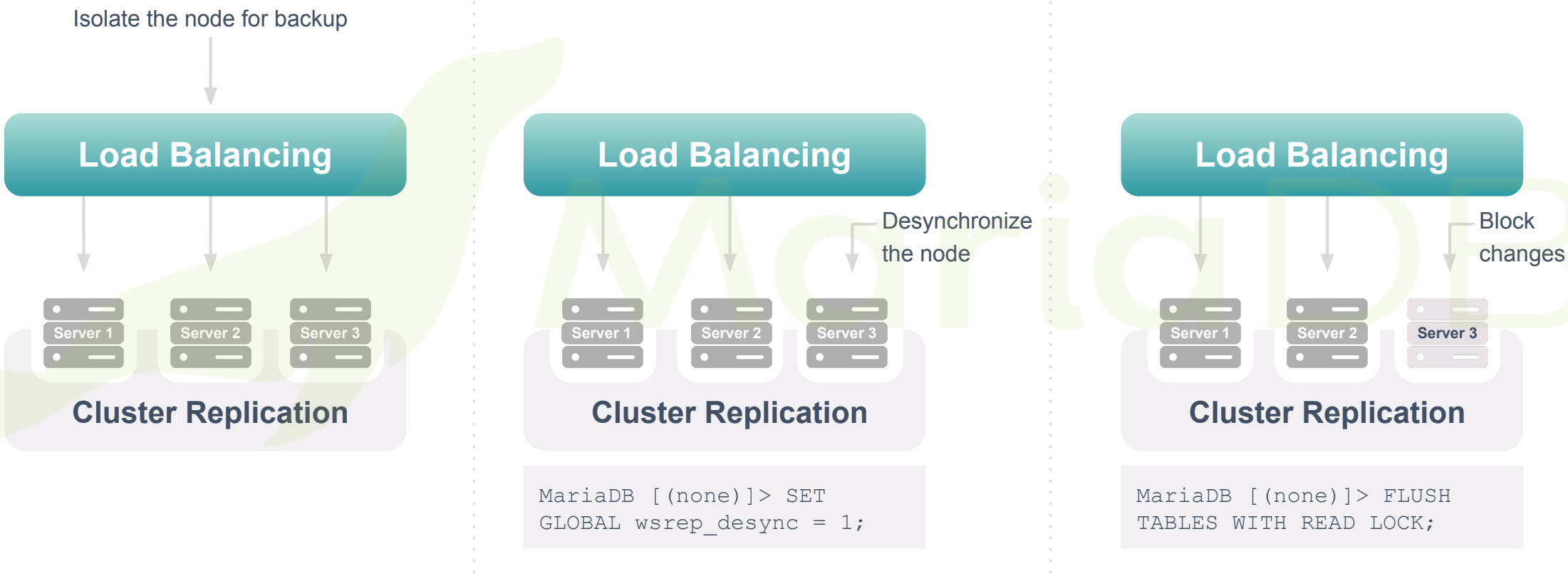
Possible Methods

- Desynchronize a node for backup
- Perform the backup with MariaDB Backup (Recommended)

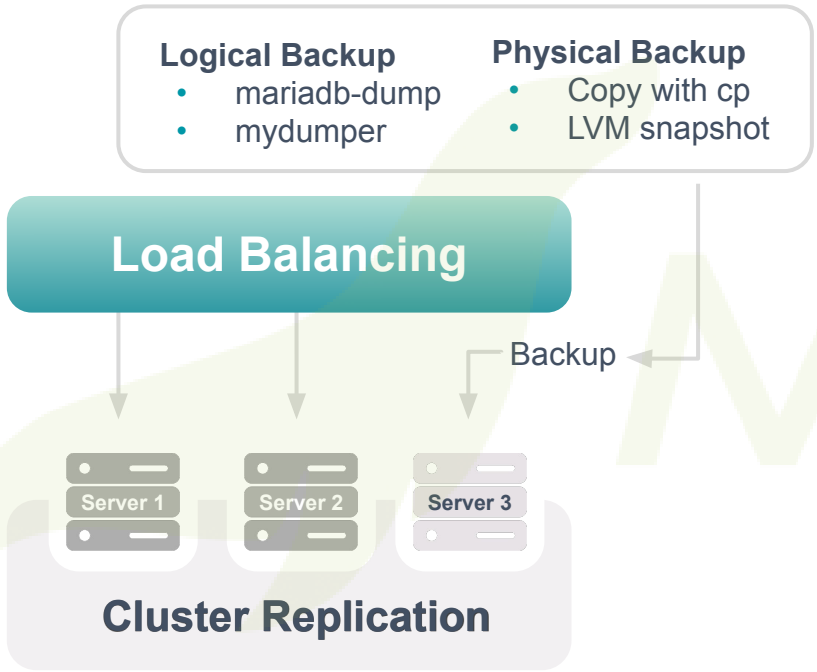
Backups with GTID

- Use MariaDB Backup with `--galera-info` option
- The GTID marks its position in the Cluster Transaction Stream
- A backup with a known GTID enables IST instead of SST, which is useful for:
  - Recovering a node
  - Provisioning new nodes

DESYNCHRONIZING A NODE FOR BACKUP

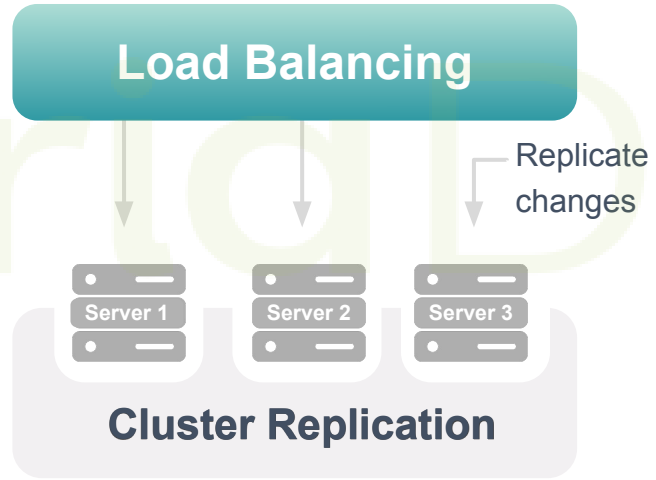


DESYNCHRONIZING A NODE FOR BACKUP




```
MariaDB [(none)]> SHOW GLOBAL STATUS LIKE
'wsrep_cluster_uuid';

MariaDB [(none)]> SHOW GLOBAL STATUS LIKE
'wsrep_last_committed';
```




```
MariaDB [(none)]> UNLOCK TABLES;

MariaDB [(none)]> SET GLOBAL wsrep_desync=0;
```

CLUSTER RECOVERY

MariaDB Training

RECOVERING FROM DOWNTIME

Scheduled

Scheduled maintenance

Clean shutdowns

Unexpected

Power outage

Network outage

Out-of-sync nodes (e.g., split-brain)

RECOVERY PROCESS

Galera GTID (seqno) is written to persistent InnoDB storage after each `COMMIT`

Galera will use InnoDB GTID if `seqno` in `grastate.dat` is undefined (-1)

`seqno` is undefined in case of crash

Persistent GTID can be queried with `galera_recovery`

Old non-systemd installations use `--wsrep_recover`

Use InnoDB GTID if unsure which node was last updated

SHUTDOWN RECOVERY

Cluster Start Order is Important

Always start with node that was last shutdown

Which node was the last with latest writes or changes?

```
# GALERA saved state
version: 2.1
uuid:    c414d03a-3d83-11e8-acaa-cf4bfafd35a4
seqno:   23369
safe_to_bootstrap: 1
```

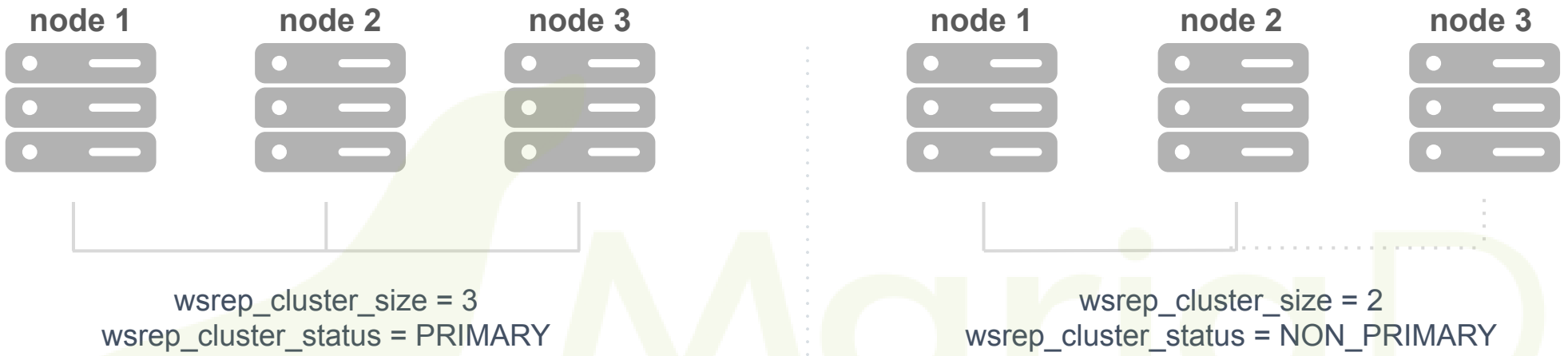
```
# GALERA saved state
version: 2.1
uuid:    c414d03a-3d83-11e8-acaa-cf4bfafd35a4
seqno:   23368
safe_to_bootstrap: 0
```

Only the last node has safe_to_bootstrap: 1 in grastate.dat

Galera prevents you from starting cluster from other nodes

```
[ERROR] WSREP: It may not be safe to bootstrap the cluster from this node. It was not the last
one to leave the cluster and may not contain all the updates. To force cluster bootstrap with
this node, edit the grastate.dat file manually and set safe_to_bootstrap to 1 .
```

SPLIT-BRAIN RECOVERY



Nodes 1 and 2 do not know whether node 3 is dead or just temporarily unreachable

Primary cluster status is lost

Manually promote the cluster to **PRIMARY**

```
SET GLOBAL wsrep_provider_options = "pc.bootstrap=1";
```

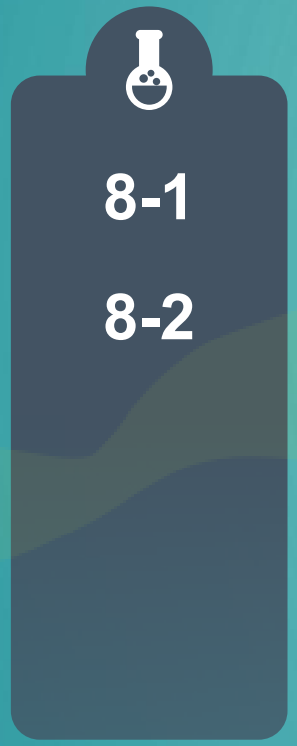
LESSON **SUMMARY**

Gain an understanding of schema upgrade methods and best practices

How to backup MariaDB Cluster

How to explain MariaDB Cluster's recovery process

LAB **EXERCISES**



8-1 Comparing Online and Offline DDL Statements

8-2 Testing a Rolling Schema Upgrade

HIGH AVAILABILITY

MariaDB Training

LEARNING **OBJECTIVES**

- Gain an understanding of Cluster Topologies
- Gain an understanding of Load Balancing Tools such as MariaDB MaxScale
- Describe the advantages and disadvantages of Geo-Replication
- Explain Failover and Failback

CLUSTER TOPOLOGIES

MariaDB Training

SINGLE PRIMARY CLUSTER

If a cluster uses only one Node as a primary (i.e. writing node)

Set
`wsrep_sync_wait = 1`

Can relax Flow Control before pausing replication

```
MariaDB [(none)]> SET GLOBAL VARIABLE wsrep_provider_options =
                  "gcs.fc_limit = 256;
                   gcs.fc_factor = 0.99;
                   gcs.fc_master_slave = YES";

-- Defaults
-- gcs.fc_limit = 16
-- gcs.fc_factor = 0.5
-- fc_master_slave = NO
```

Add settings to `wsrep_provider_options` option in the configuration file so that they persist a server reboot

TWO NODE CLUSTER

If a cluster uses **Two-Node Cluster**, there are two scenarios that should be considered:

- Split-Brain
- Non-Operational Cluster

Recommendation for Split-Brain - Galera Arbitrator

Configuration for Galera Arbitrator

```
group="galera-testing"
address="gcomm://172.31.30.39,172.31.18.53,172.31.
26.106"
options="gmcast.listen_addr=tcp://0.0.0.0:4444"
log="/var/log/garbd.log"
```

Starting Galera Arbitrator

```
# garbd --cfg /etc/garbd.cnf
```

Recommendation for Non-Operational Cluster

```
mariadb> SET GLOBAL
wsrep_provider_options='pc.bootstrap=YES';
```

MULTI-PRIMARY CLUSTER

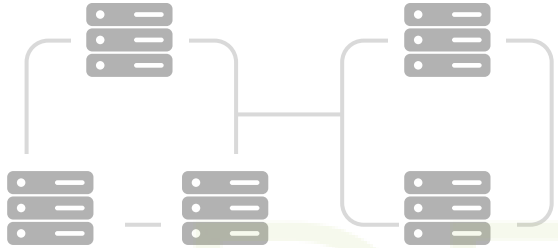
Multi-Primary for Load Balancing

Nodes are typically hosted within 1 data center

(Multi-)Primary - (Multi-)Replica

Read scale-out without the typical slave lag known from replication

Replicas for reporting or data mining



Multi-Primary for Business Continuity & Disaster Recovery

Nodes are distributed across 2 or more data centers

LOAD BALANCING

MariaDB Training

LOAD BALANCING TOOLS

gldb daemon

Used for load balancing through a daemon process

libglb

Used for load balancing in the application process

```
# LD_PRELOAD=src/.libs/libglb.so \
GLB_OPTIONS="--random 3306
192.168.0.1 192.168.0.2
192.168.0.3" \
mysql -u root —p fido123 -h
127.0.0.1 -P 3306
```

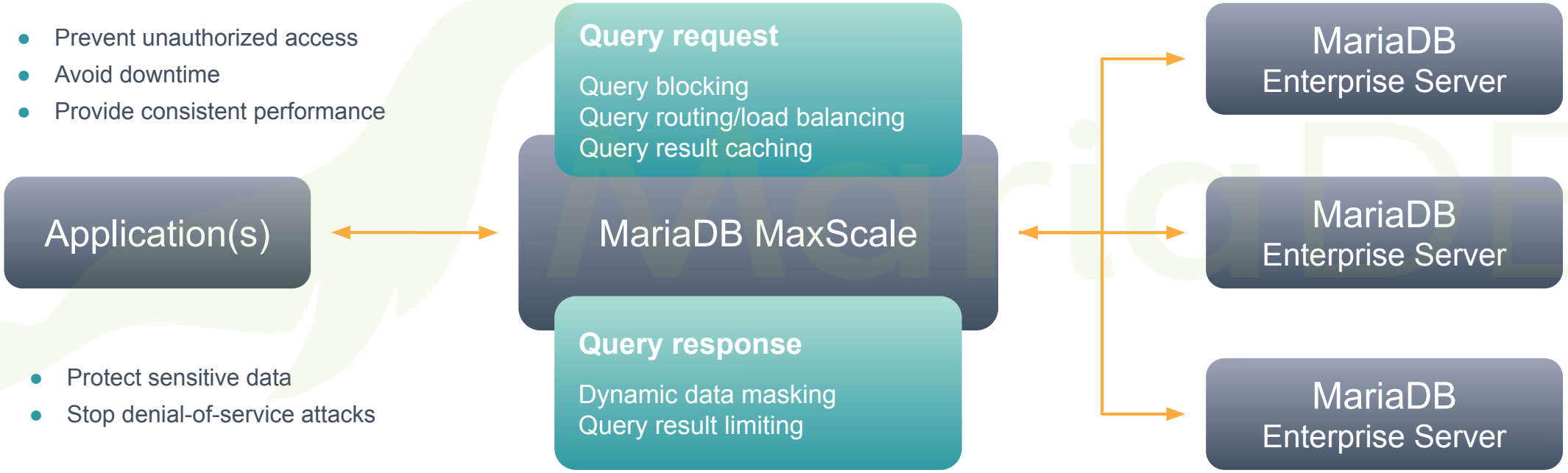
MaxScale

Used for load balancing and rule-based request routing

```
# yum install maxscale
```

MARIADB MAXSCALE

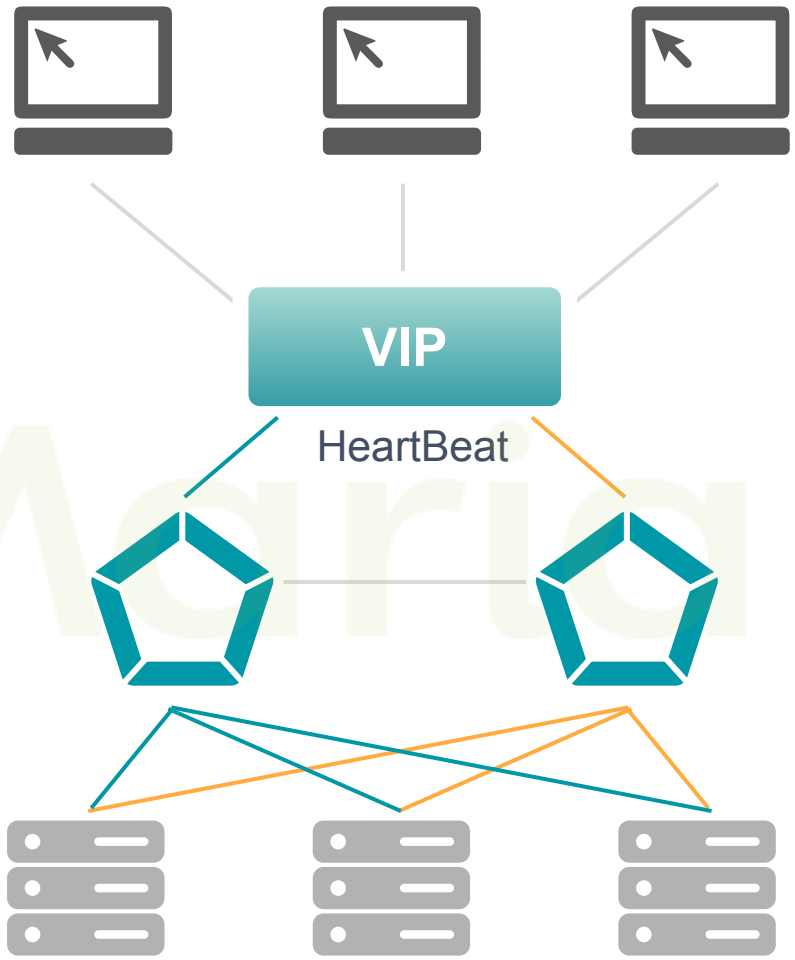
- Prevent unauthorized access
- Avoid downtime
- Provide consistent performance



- Protect sensitive data
- Stop denial-of-service attacks

MAXSCALE

MaxScale can be used for load balancing and high availability.



GEO-DISTRIBUTED DATABASE CLUSTERS

MariaDB Training

GEO-DISTRIBUTED TOPOLOGIES

One MariaDB Cluster spanning different locations

Features

- All nodes are in sync
- Automatic failover to different location if one location fails
- `AUTO_INCREMENT` is automatically managed by cluster

One MariaDB Cluster per location

Features

- In case of WAN failure all locations can continue working
- Automatic failover and failback possible with MaxScale

SEGMENTS

Define how MariaDB Cluster nodes are physically grouped

Nodes in Data Center #1

```
wsrep_provider_options = "gmcast.segment=1"
```

Nodes in Data Center #2

```
wsrep_provider_options = "gmcast.segment=2"
```

DEPLOYMENT ARCHITECTURE

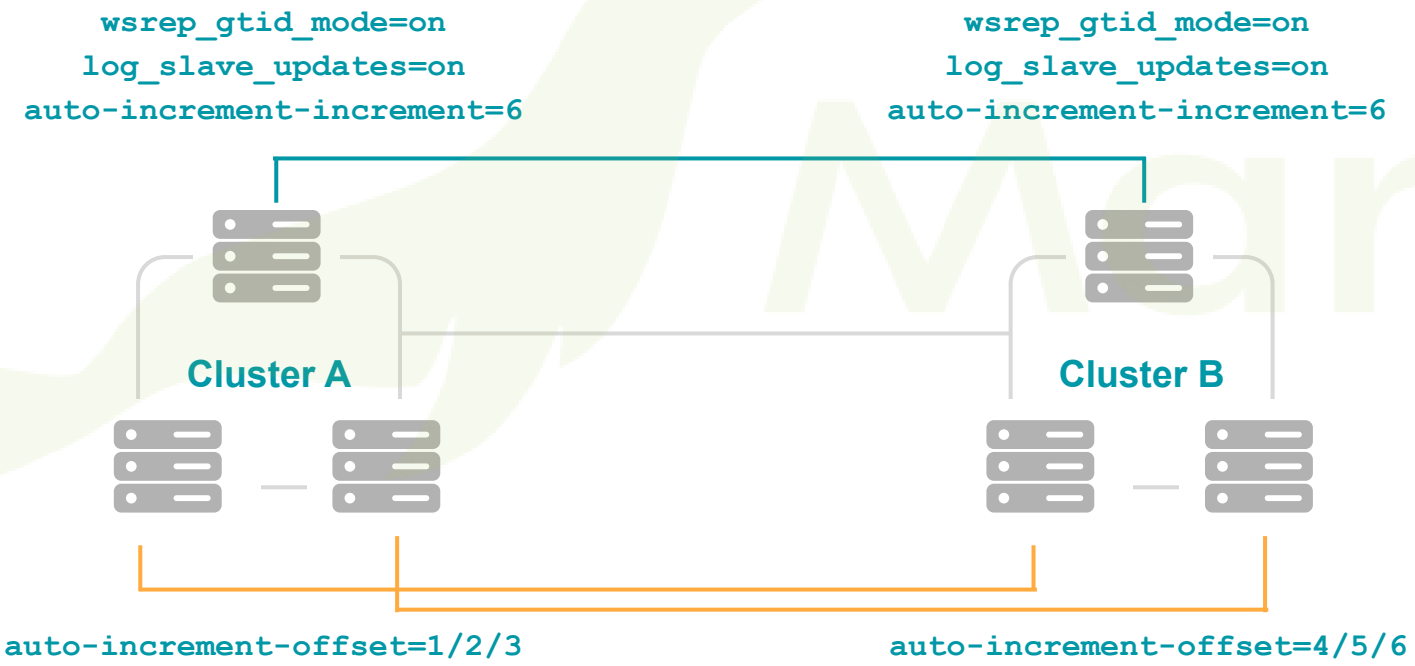
One MariaDB Cluster spanning different locations

- Segments select one node on each side to promote write sets to the other segment
- If actual link is down a new one is elected automatically
- With equal number of nodes use arbitrator or higher weight on one segment
- Flow Control can be configured for better performance

```
wsrep_provider_options="evs.send_window=8;evs.user_send_window=4"
```

DEPLOYMENT ARCHITECTURE

One MariaDB Cluster per location



Redundant bi-directional or circular MariaDB Replication lines setup between the clusters with one being active at a given time

`server-ids` per cluster identical

separate `domain-ids` per cluster

LESSON **SUMMARY**

- Gain an understanding of Cluster Topologies
- Gain an understanding of Load Balancing Tools such as MariaDB MaxScale
- Describe the advantages and disadvantages of Geo-Replication
- Explain Failover and Failback

ADVANCED FEATURES

MariaDB Training

LEARNING **OBJECTIVES**

Gain an understanding of Weighted Quorums

How to set up an Arbitrator as an alternative to weighted quorums

How and when to use:

- Multicasting
- Multiple Clusters
- WAN Replication

How to set up MariaDB Cluster Notifications

WEIGHTED QUORUM

MariaDB Training

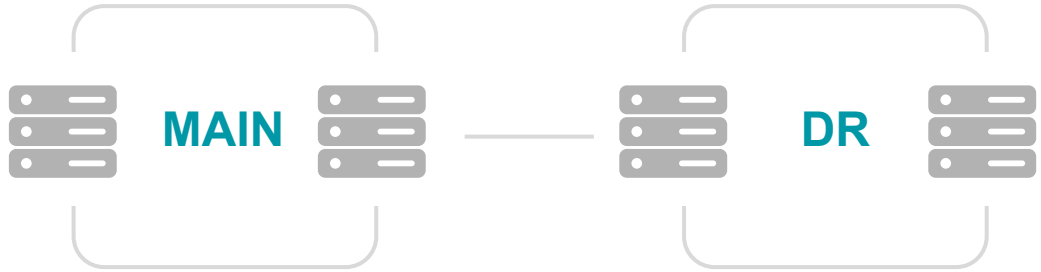
WEIGHTED QUORUM

Better Control on Primary & Non-Primary State in Case of Failure

For Link Failure, MAIN remains Primary

Manual Intervention to make DR Primary

```
server MAIN1: pc.weight=1
server MAIN2: pc.weight=1
server DR1: pc.weight=0
server DR2: pc.weight=0
```

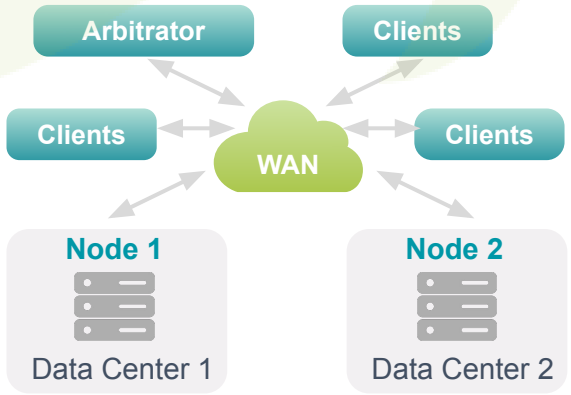


ARBITRATOR

MariaDB Training

USING AN ARBITRATOR TO ENABLE QUORUM

In locally distributed clusters with identical number of nodes per location an arbitrator can be used to define which side should stay PRIMARY.



Using an arbitrator is an alternative to set weighted quorums

To start garbd as an arbitrator service (System V or systemd) setup the configuration file

```
# Copyright (C) 2013-2015 Codership Oy
# This config file is to be sourced by garbd service script.
# A space-separated list of node addresses (address[:port])
# in the cluster:
GALERA_NODES="192.168.1.1:4567,192.168.1.2:4567"
# Galera cluster name, should be the same as on the rest of the
node.
GALERA_GROUP="example_wsrep_cluster"
# Optional Galera internal options string (e.g. SSL settings)
# see
#
https://galeracluster.com/library/documentation/galera-paramete
rs.html
GALERA_OPTIONS="socket.ssl_cert=/etc/galera/cert/cert.pem;socke
t.ssl_key=/$"
# Log file for garbd. Optional, by default logs to syslog
LOG_FILE="/var/log/garbd.log"
```

NOTIFICATIONS

MariaDB Training

MARIADB CLUSTER NOTIFICATIONS

MariaDB Cluster can trigger notifications

Notification can be sent to MaxScale

Script API for notification syntax

`wsrep_notify_cmd` defines the script to handle notifications

- Load balancer configuration
- Monitoring

Notification script (`/usr/bin/wsrep_notify.sh`)

```
#!/bin/bash
# Command to call when node status or cluster
membership changes.
# Some or all of the following options will be
passed:
# --status  - New status of node
# --uuid    - UUID of cluster
# --primary - Whether component is Primary ("yes"
or "no")
# --members - Comma-separated list of members
# --index   - Index of node in list

echo "$@" >> /tmp/wsrep_notifications.log
```

MULTICASTING

MariaDB Training

MULTICASTING

Value of Multi-Casting — WAN vs. LAN

Recommended to set Multicast **before** **Bootstrapping Cluster**

Problems may occur if Migrating from Unicast to Multicast

```
# ip route add 224.0.0.0/4 dev eth0

[galera]
wsrep_provider_options="gmcast.mcast_addr=239.192.0.11"
wsrep_cluster_address="gcomm://239.192.0.11"
```

WAN REPLICATION

MariaDB Training

WAN REPLICATION

Link Failures happens More Frequently

No Need to Arbitrate Always
Increase the Timeouts

Writes Stall for Longer Time while Link is Down
Monitor the Network

```
wsrep_provider_options =
   "evs.keepalive_period = PT3S;
    evs.suspect_timeout = PT30S;
    evs.inactive_timeout = PT1M;
    evs.install_timeout = PT1M"
```

```
Defaults:
evs.keepalive_period = PT1S
evs.suspect_timeout = PT5S
evs.inactive_timeout = PT15S
evs.install_timeout = PT15S
```

WAN LATENCY

In terms of ingestion capability, the cluster is as slow as is its slowest (or farthest) member.

Networks have a physical limit.

TCP adds more restriction on top of this: a trans-continental link can only sustain a few megabits per second.

Monitoring the link delay becomes of high importance.

ICMP is the proper protocol for this with `ping` a common command-line utility.

LESSON **SUMMARY**

Gain an understanding of Weighted Quorums

How to set up an Arbitrator as an alternative to weighted quorums

How and when to use:

- Multicasting
- Multiple Clusters
- WAN Replication

How to set up MariaDB Cluster Notifications

LAB **EXERCISES**

10-1 Setting Up Cluster Notifications

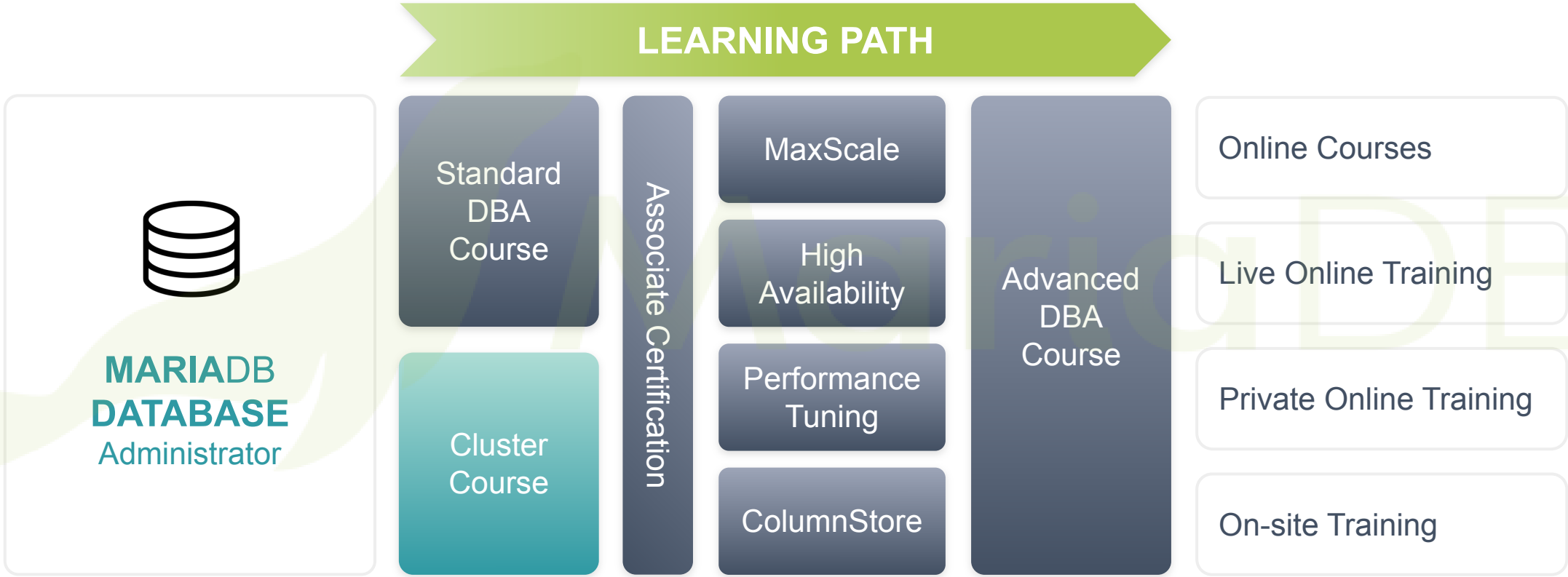
CONCLUSION

MariaDB Training

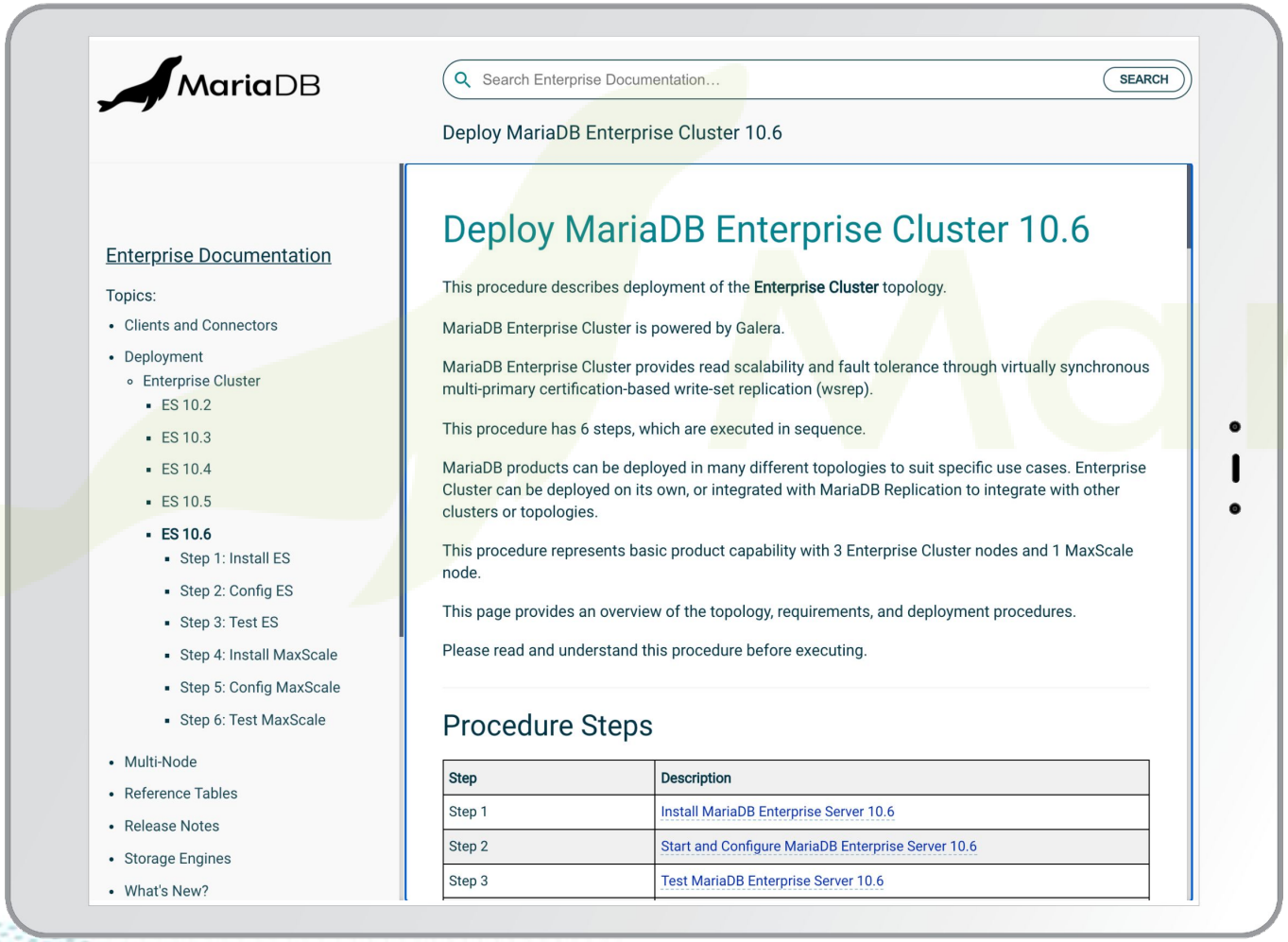
COURSE SUMMARY

- Understand concepts such as cluster replication associated with MariaDB Cluster
- Explain the architecture of MariaDB Cluster
- Install and configure a basic setup of MariaDB Cluster
- Plan and perform an upgrade of MariaDB Cluster
- Understand how to migrate standalone or replication topologies to MariaDB Cluster
- Perform backup and restore operations using MariaDB Backup
- Secure MariaDB Cluster
- Monitor and improve MariaDB Cluster performance based on key performance indicators
- Understand and troubleshoot common MariaDB Cluster problems
- Describe common cluster topologies
- Understand how to load balance MariaDB Cluster using MariaDB MaxScale
- Explain and understand how to manage some known cluster specifics of such as Split Brain, consistent reads, controlling the AUTO_INCREMENT attribute, detecting slow nodes and parallel replication
- Use best practices to perform schema upgrades
- Explain how State Snapshot Transfers (SSTs) and Incremental State Transfers (ISTs) work in MariaDB Cluster
- Describe the advantages and disadvantages of Geo-Replication
- Set up advanced features such as MariaDB Cluster Notifications and Arbitrator
- Explain how and when to use other advanced features such as multicasting, multiple clusters, and WAN replication

MARIADB CERTIFICATION & TRAINING



ENTERPRISE DOCUMENTATION



Covers all MariaDB products

Detailed release notes, instructions, reference tables

Left navigation and search help you discover/find

Right navigation helps you jump within page

TOOLS AND **RESOURCES**

MariaDB Forums

https://cloud.mariadb.com/forums

MariaDB Blog

https://mariadb.com/resources/blog/

Knowledge Base

Simply search for

"MariaDB kb <topic>"

Open-Source Community

Post Questions on Forums

Ask Questions on IRC

MARIADB SUPPORT

MariaDB Support is here for when you have problems that are beyond your ability to resolve.

- 24 x 7 Online, Live Assistance
- Performance Tuning
- Login Support
- General Consulting
- Code Review
- Bug and Hot fixes

Thank you for completing this course!

Have feedback? course.feedback@mariadb.com