Standard Replication & Galera Cluster
Codership Training
Introduction

Standard Replication & Galera Cluster
Introductions

**Codership Oy**

Creators & Developers of Galera Cluster
Employees in Multiple Countries

**Galera Cluster**

Released Initially in May 2007
Over 1.5 Million Downloads

**Russell Dyer, Presenter**

KB Editor, Documentation, Instructor
(MySQL, MariaDB)

Writer (O'Reilly Books)
Tutorial Outline

Standard Replication
- Purpose & Advantages
- Standard Replication Layout
- Configuring Replication

Galera Cluster
- Galera Basics
- Configuring Galera
- Deploying a Cluster
Purpose & Advantages of Replication

Standard Replication & Galera Cluster
Standard Replication for Maintenance

Back-Ups
- Continuous — Never Old; Always Current
- Without Locking Tables — No Performance Drain

Upgrades & Schema Changes
- Set Slave to Read-Only
- Make Slave Master
- Redirect Traffic away from Master to Slave — New Master
- Apply Changes to Original Master
- Redirect Traffic back to Master

Standard Replication for High Availability

Master Fails
- Network Outage
- Physical Problems

Swappable Database Servers
- Redirect Traffic to Slave
Replication for Load Balancing

Distribute Read Traffic
- Move Slow, Heavy Queries to Slave

Redirect for Maintenance
- Take Slave Off-Line to for Back-ups
- Direct away from Failed Servers
- Redirect Traffic while Upgrading
Enhanced Replication with Galera Cluster

Multiple Masters

Transactional Writes

  Conflict Detection & Resolution upon Commit

Seamlessly Remove & Add Servers

  Nodes Isolated Easily

Automatic Provisioning

Standard Replication Layout

Standard Replication & Galera Cluster
Basic Replication Elements & Process

Master
- Handles Client Writes – Possibly Also Reads
- Daemon Sends Writes to Storage Engines
- Write Queries Recorded in Binary Log
- Sends Binary Log Entries to Slave – When Asked

Slaves
- Handles Only Client Reads
- Queries Master for Binary Log Entries
- Writes to Storage Engines and Relay Log
Slave Master

Set `log_slave_updates` in Configuration File

Add Replication Filtering Rules – Optional

Change Storage Engine on Intermediate Slaves – Optional
Circular Replication

Multiple Masters for Load Balancing

 Writes – No True Multi-Master

Circular Replication Simulates Multi-Masters

 Each Slave is Master to Another

 Requires Binary Log on Each Slave

 Set `log_slave_updates` Option

Asynchronous

Master Doesn’t Wait for Slaves

IO Thread may Slow to Receive Binary Log Packets
  Network Congestion or Disconnects

SQL Thread may Slow in Processing Relay Log
  Load on Slave or Network Problems
Semi–Synchronous Replication Mode

Implemented with an optional Plug-In
Master waits for a Slave to Acknowledge Receipt
Slave waits to Write to Relay Log – Not to Execute
Only One Slave Response Needed
Can Affect Significantly Performance of Master

```
INSTALL PLUGIN rpl_semi_sync_master SONAME 'semisync_master.so';

INSTALL PLUGIN rpl_semi_sync_slave SONAME 'semisync_slave.so';
```

Executed from `mysql` Client

Parallel Replication

Replication Process on Slaves

Events Received from Master (IO Thread) and
Queued in Relay Log
Each Relay Log Entry is Retrieved by SQL
Thread
Each Transaction is *Applied* to Slave

Application Performed in Pool of Separate
Worker Threads — Not Sequentially by SQL Thread

Configuring Standard Replication

Standard Replication & Galera Cluster
Master Configuration

Enable Binary Log with `log-bin`

Set `server-id` to Unique Value

Create Replication User Account on Master

Make a Consistent Snapshot of Data on Master

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
server-id = 1
log-bin
log-error=/var/log/mysqld.log
```

Excerpt from Database Configuration File

```
GRANT REPLICATION SLAVE ON *.*
TO 'replicator'@'34.211.1.12'
IDENTIFIED BY 'rover123';
```

Executed from `mysql` Client

```
mysqldump -p -u root \
   --master-data --flush-logs \
   --all-databases > full-dump.sql
```

Executed from Command-Line

Replication Threads

Master Dump Thread
Sends Binary Log Entries to Slave

Slave IO Thread
Requests and Receives Master Binary Log Entries
Writes Entries to its Relay Log

Slave SQL Thread
Reads Relay Log & Executes Queries Locally
Checks Query Result Codes Match Master

Multiple Execution Threads on Slave
Separates Entries by Database
Updates Applied in Parallel — Not Sequence

Replication Files

Master Binary Log Files

Master Records Write to File
Rotated when Flushed or Periodically to New Log File

Slave Relay Log File

Log of Master Binary Log Entries
Rotated Periodically or when Flushed

Replication Configuration Stored in `master.info` (Slave)

Name of Relay Log File in `relay-log.info` (Slave)

Slave Configuration

Configuration File

Set `server-id` to Unique Value

Add `read-only` on to Prevent Writes

Restart `mysqld`

Load Data from Master

Execute `CHANGE MASTER` Statement

Execute `START SLAVE` on Slave

MySQL `CHANGE MASTER TO`: https://dev.mysql.com/doc/refman/5.5/en/change-master-to.html
MariaDB `CHANGE MASTER TO`: https://mariadb.com/kb/en/mariadb/change-master-to/

Excerpt from Database Configuration File

```
[mysqld]
...
server-id = 2
read-only
```

Execution from Command-Line

```
mysql -p -u root < full-dump.sql
```

Excerpt from `mysql` Client

```
CHANGE MASTER TO
MASTER_HOST='172.31.31.202',
MASTER_PORT=3306,
MASTER_USER='replicator',
MASTER_PASSWORD='rover123';
```

Execution from `mysql` Client

```
START SLAVE;
```
Configuring & Starting Replication
Monitoring Replication

Check Regularly Status on Master
Includes Current Binary Log File Name & Position

Check More Often Status of Replication on Slave

SHOW MASTER STATUS;
Executed from mysql Client on Master

SHOW SLAVE STATUS \G
Executed from mysql Client on Slave

MariaDB SHOW MASTER STATUS: https://mariadb.com/kb/en/show-master-status/
MariaDB SHOW SLAVE STATUS: https://mariadb.com/kb/en/mariadb/show-slave-status/
Troubleshooting Replication

Check Slave Error Log for Replication Entries

- Network Disconnects
- Binary or Relay Log Event Corruption – Stops Slave SQL Thread
- Different Error Codes indicates Not Synchronized

Rebuild Slave from a Fresh Snapshot
Galera Cluster Basics

Standard Replication & Galera Cluster
Galera Cluster Features

Virtual Synchronous Replication
True Multi-Master Solution
Almost No Slave Lag
Conflict Detection & Resolution on Commit
Easy Maintenance
  Automatic Provisioning
  Node Isolation
  Rolling Upgrades
Data Centric

Data Doesn’t Belong to a Node – Nodes Belong to Data
Data is Synchronized among Multiple Nodes
Galera Nodes are Anonymous – All are Equal
Galera Cluster is a Distributed Master
Node Provisioning Tool

State Transfers for New Nodes

State Snapshot Transfer (SST)
Incremental State Transfers (IST)

Methods for State Transfers

Logical — `mysqldump`
Physical — `rsync`

Galera Node Provisioning: https://galeracluster.com/library/documentation/node-provisioning.html
Galera State Transfers: https://galeracluster.com/library/documentation/state-transfer.html
Configuring Galera Cluster

Standard Replication & Galera Cluster
Database Configuration

Set Bind Address – Not Local Host
Set Default Storage Engine to InnoDB
Set Binary Log Format to Row
Enable Error Logging

Excerpt from MySQL Configuration File.

```ini
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
bind-address=0.0.0.0
user=mysql

default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
innodb_flush_log_at_trx_commit=0
innodb_buffer_pool_size=128M

binlog_format=ROW
log-error=/var/log/mysqld.log
```
Galera Configuration

Identify & Enable Galera
Node Name & Address
Cluster Name & Addresses
Slave Threads
State Transfer Method

Excerpt from Database Configuration File.

Galera Configuration:  https://galeracluster.com/library/training/tutorials/wsrep-configuration.html
Galera Options:  https://galeracluster.com/library/documentation/mysql-wsrep-options.html
Galera Ports

MySQL Default Traffic – TCP 3306
Galera Cluster – TCP & UDP 4567
Incremental State Transfers – TCP 4444
State Snapshot Transfers – TCP 4568
Open Ports or Disable SELinux & Firewall

Open Ports on SELinux

```
semanage port -a -t mysqld_port_t -p tcp 3306
semanage port -a -t mysqld_port_t -p tcp 4444
semanage port -a -t mysqld_port_t -p tcp 4567
semanage port -a -t mysqld_port_t -p udp 4567
semanage port -a -t mysqld_port_t -p tcp 4568
semanage permissive -a mysql_d_t
```

Disables SELinux

```
setenforce 0
```

Firewall Settings: https://galeracluster.com/library/documentation/firewall-settings.html
SELinux Configuration: https://galeracluster.com/library/documentation/selinux.html
Configuring firewalld: https://galeracluster.com/library/documentation/firewalld.html
Caveats of Starting a Cluster

A Cluster is made of Multiple Nodes
   — Not Stand-Alone

New Nodes Unassuming — Look for
   Primary Component

Tell First Node it's the Primary Component
Starting Nodes

Bootstrap Database & Galera on Seed Node

MySQL — `mysqld_bootstrap`

MariaDB — `galera_new_cluster`

Start Database & Galera Normally on Additional Nodes

```
Seed Node
```
```
mysqld_bootstrap
```
```
Seed Node
```
```
galera_new_cluster
```
```
Additional Nodes
```
```
systemctl start mariadb
```
```
Additional Nodes
```
```
systemctl start mysqld
```

**Starting MySQL Nodes**

**Starting MariaDB Nodes**
Configuring & Starting a Galera Cluster
Conclusion

Standard Replication & Galera Cluster
Additional Resources

Codership Library (galeracluster.com/library)

Documentation (/library/documentation)
Knowledge Base (/library/kb)
FAQ (/library/faq)
Training (/library/training)
  Videos (/library/training/videos)
  Tutorials (/library/training/tutorials)