MariaDB Cluster for High Availability
World Most Advanced Open Source Database Cluster

April 29, 2015
Sonali Minocha and Abdul Manaf

Solutions.
Developed.
Delivered.

OSSCube Introduction

Solutions.
Developed.
Delivered.

Introducing OSSCube

- Global Solutions Provider
  - Consulting, Development, Integration, and Support

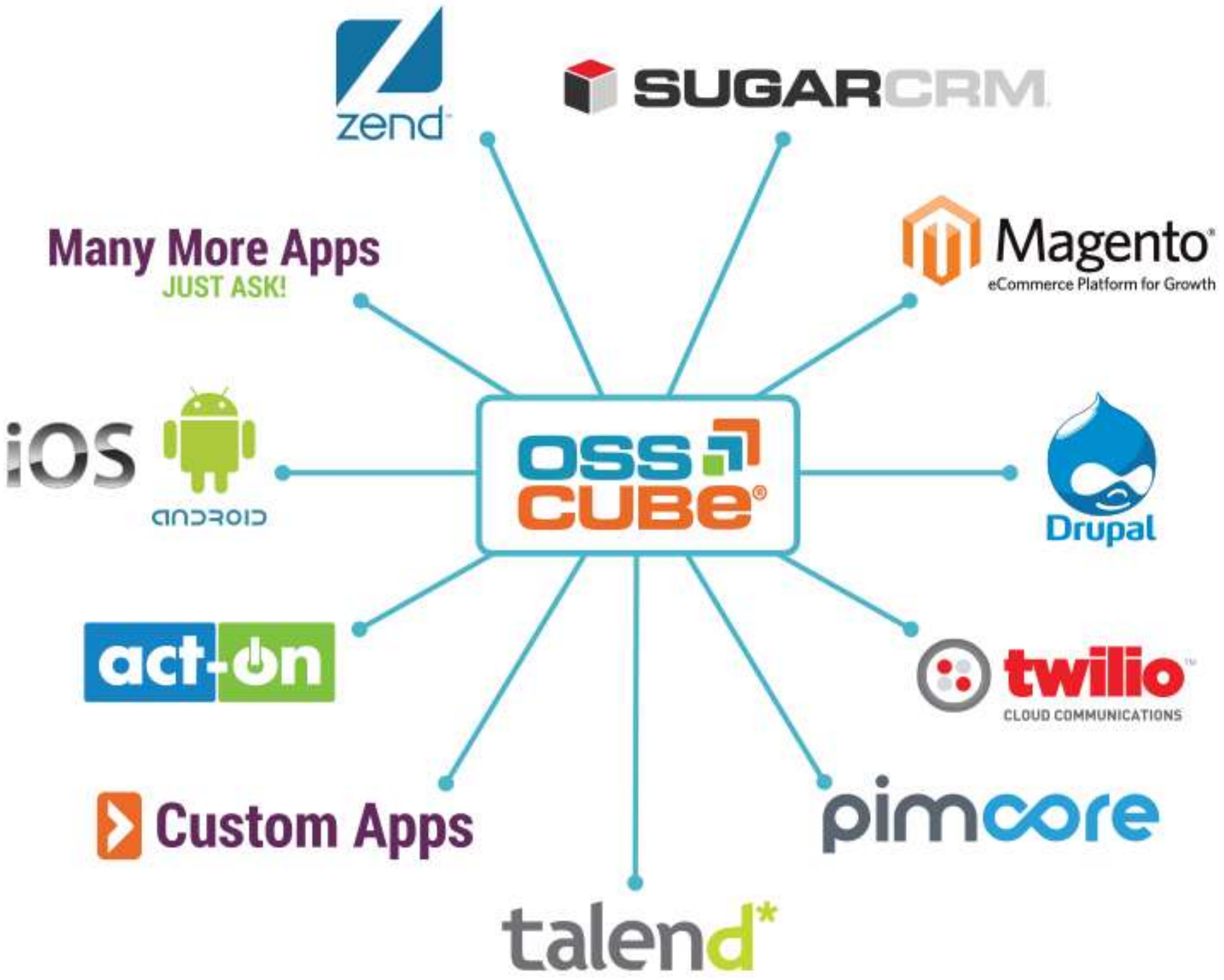
- Develop Integrated Enterprise Business Solutions
  - Open Source
  - CMMI Level 3 accredited
  - Continuous Delivery for fast, predictable projects

- Flexible Delivery Model
  - Support Projects or Staff Augmentation
  - Scale up or down as required

OSSCube Practice Areas

- PHP – The World’s Only Zend Center of Excellence
- CRM - SugarCRM
- Marketing Automation – Act-On
- Content Management System – Drupal
- Product Information Management - Pimcore
- Enterprise Mobile and Web applications
- E-Commerce – Magento
- BPM and Workflow – BonitaSoft and ProcessMaker
- Enterprise Architecture and Consulting
- BI and Data Integration – Pentaho and Talend
- Big Data – Hadoop, Cloudera, Spark
- Database – MySQL, MariaDB
- Cloud based large computing capacity - AWS

Integrated Business Solutions



Inc. 5000 OSS CUBE

Webinar Agenda

- Concept of High Availability
- MySQL native replication and problems
- Overview of Galera cluster
- Galera cluster architecture
- MariaDB Galera 3 Node Implementation

High Availability

- High availability is a system design protocol and associated implementation that ensures a certain degree of operational continuity during a given measurement period

- High availability refers to the ability of users to access a system without loss of service

Do you need high availability ? If you are not certain, consider the following question:

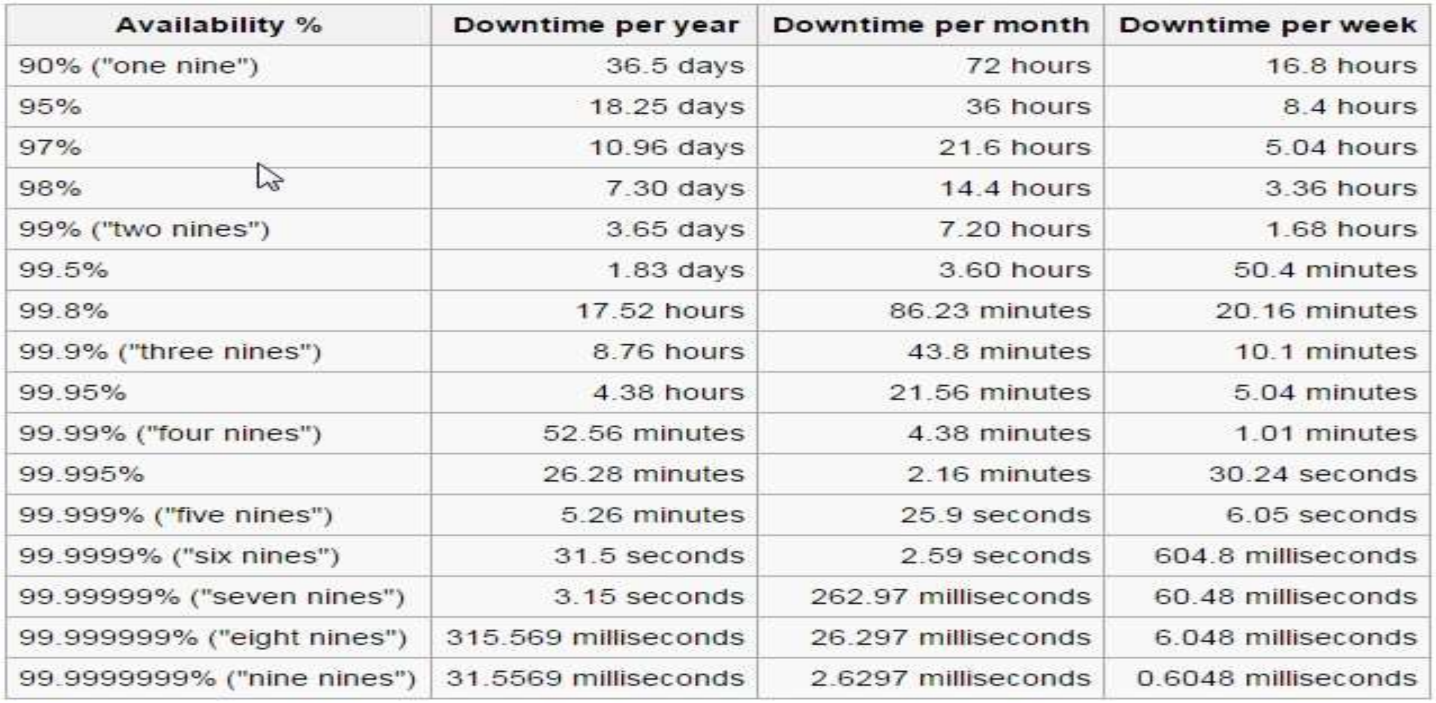
- Which level of availability do I need?
  - How many nines? or How much downtime can your business survive?

- Do I require no loss of data?
  - Could I loose some transactions?
  - Will my users notice or care?

- Do I need automatic fail over or is manual switchover OK?
  - How do I test this?

Availability Calculation

Availability = Uptime / (Uptime + Downtime)



Availability %	Downtime per year	Downtime per month	Downtime per week
90% ("one nine")	36.5 days	72 hours	16.8 hours
95%	18.25 days	36 hours	8.4 hours
97%	10.96 days	21.6 hours	5.04 hours
98%	7.30 days	14.4 hours	3.36 hours
99% ("two nines")	3.65 days	7.20 hours	1.68 hours
99.5%	1.83 days	3.60 hours	50.4 minutes
99.8%	17.52 hours	86.23 minutes	20.16 minutes
99.9% ("three nines")	8.76 hours	43.8 minutes	10.1 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes
99.99% ("four nines")	52.56 minutes	4.38 minutes	1.01 minutes
99.995%	26.28 minutes	2.16 minutes	30.24 seconds
99.999% ("five nines")	5.26 minutes	25.9 seconds	6.05 seconds
99.9999% ("six nines")	31.5 seconds	2.59 seconds	604.8 milliseconds
99.99999% ("seven nines")	3.15 seconds	262.97 milliseconds	60.48 milliseconds
99.999999% ("eight nines")	315.569 milliseconds	26.297 milliseconds	6.048 milliseconds
99.9999999% ("nine nines")	31.5569 milliseconds	2.6297 milliseconds	0.6048 milliseconds

Principles and Causes of loosing HA

Principles of High Availability

- Elimination of single points of failure
- Reliable crossover. In multi threaded systems, the crossover point itself tends to become a single point of failure. High availability engineering must provide for reliable crossover.
- Detection of failures as they occur. If the two principles above are observed, then a user may never see a failure. But the maintenance activity must.

Scheduled and unscheduled downtime

Causes of loosing it (unscheduled downtime)

- Failures of hosts
- Failures of Databases / MySQL
- Operating system
- The hardware
- Maintenance activity that may otherwise cause downtime
- And many more ……

MySQL HA Solutions

The primary solutions supported by MySQL include:

- MySQL Native Replication
- MySQL Cluster
- MySQL with DRBD
- Oracle VM Template for MySQL
- MySQL with Solaris Cluster
- MariaDB Galera Cluster

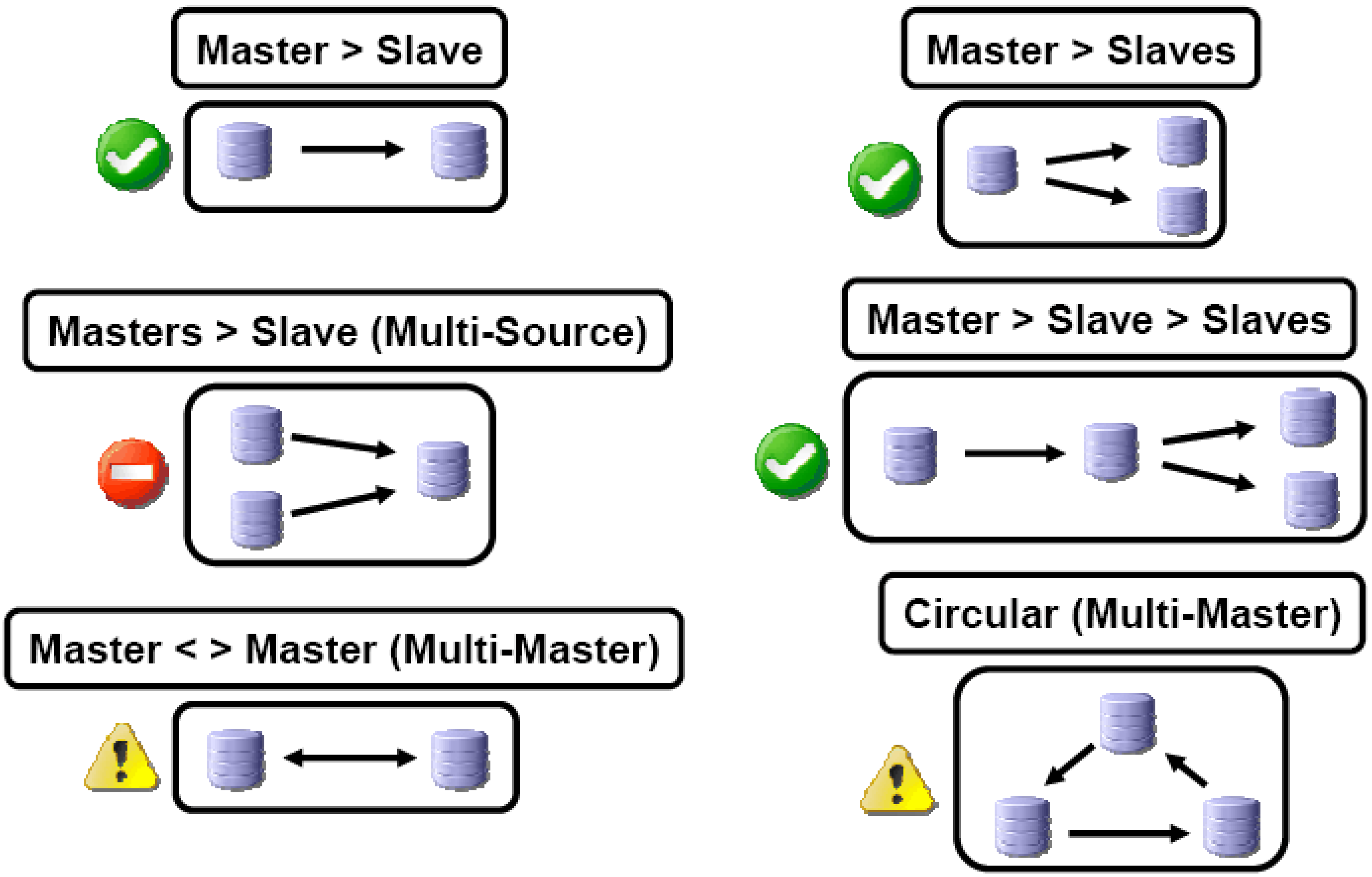
We will be covering:

- MySQL Native Replication Issues
- MariaDB Galera Cluster

Replication Overview and Components

- One of the “killer” features of MySQL
- Introduced in version 3.23
- Master
  - Binary logs
    - Row Format
    - Statement Format
  - Bin log dump thread on master
- Slave
  - Relay logs
  - IO and SQL thread on slave
  - master.info and relay-log.info files
- Row format was introduced in 5.1
- Lots of improvements in 5.6

Replication Topologies Summary



Limitations and Known Issues

- Human error: updating slave instead of master
- Non-deterministic SQL, bugs, and other
- Binary Logging format issue
  - Row
  - Statement
- Limited availability
  - Replication can break
  - Replication can lag behind
  - Replication can be out of sync
- Manual or at best semi-automatic fail over, tricky to automate.
- Limited write capacity: single threaded (Up to MySQL 5.5) causes slave lag
- Asynchronous = You will lose data

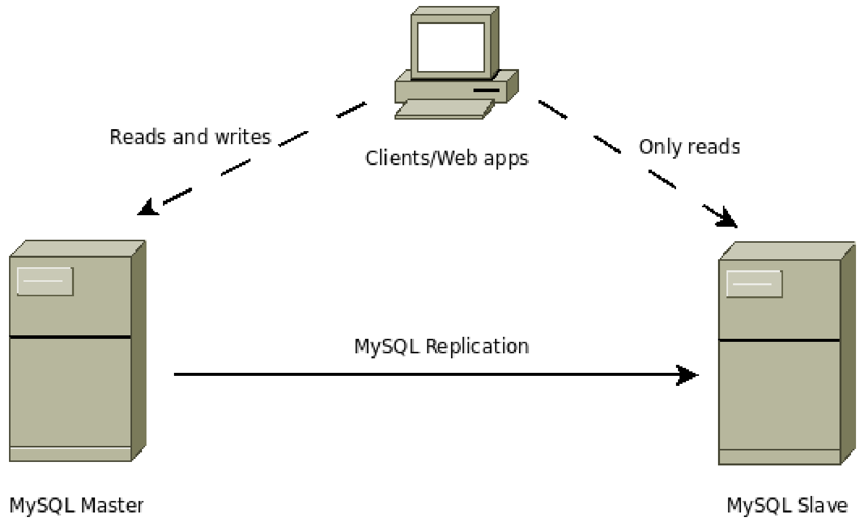
Master/Slave Based Failover

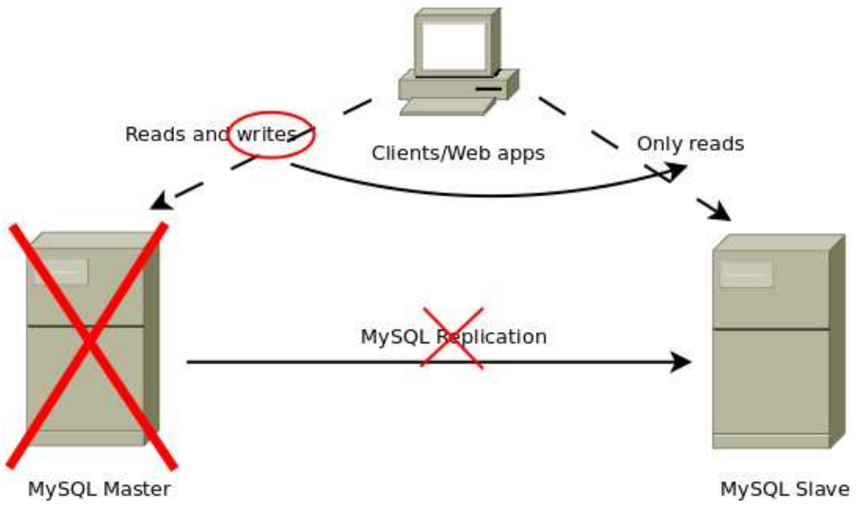
Simplest example, plain replication

Widely used

Manual failover

Manual operation required

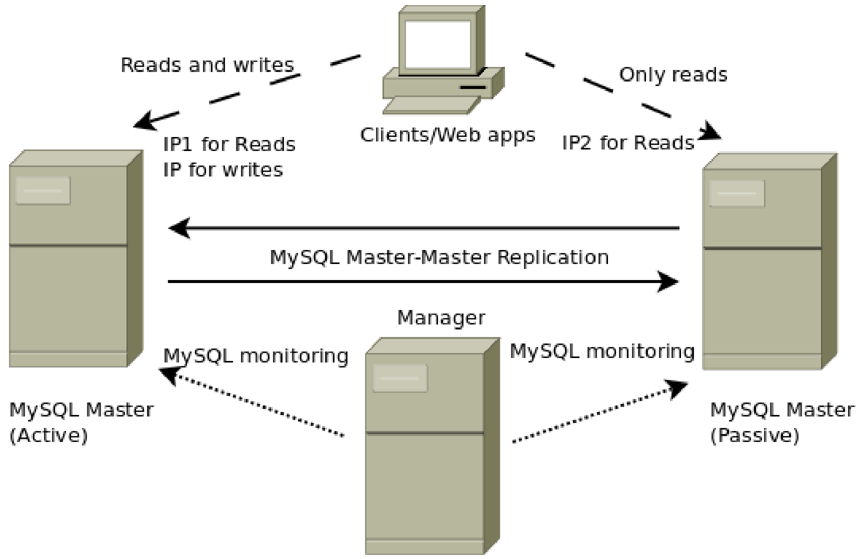


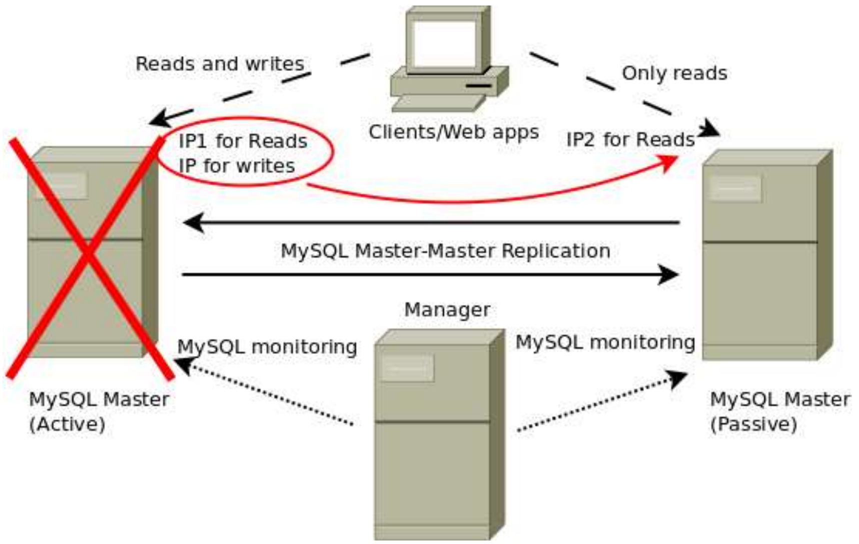


Master-Master Based Failover

Using MMM

Transfer IP1 and IP to the surviving server





Other Issues

- Duplicate Error on Slave
- Query caused different error on Master and Slave
- Temp Table doesn't exist after slave restarts
- Binary log and relay log corruption
- Slave read from wrong position after crash
- And many more...

Galera cluster

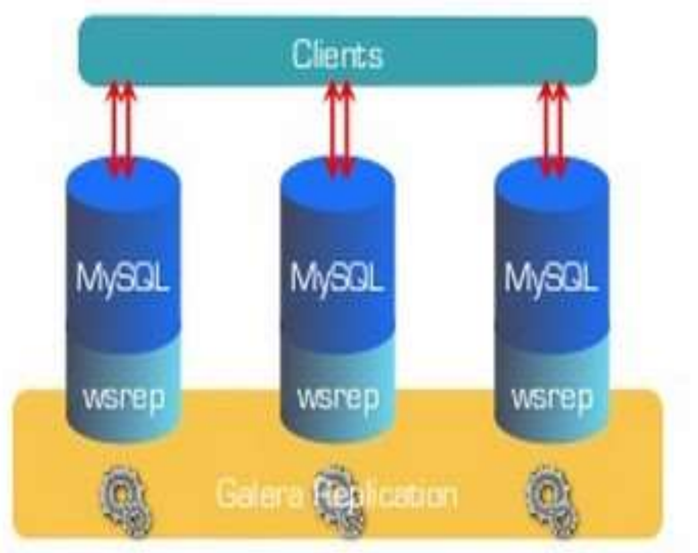
Overview of Galera Cluster

Galera Cluster Features & Benefits

- Synchronous replication
- Active-Active multi-master topology
- Read and write to any cluster node
- Automatic membership control, failed nodes drop from the cluster
- Automatic node joining
- True parallel replication, on row level
- Direct client connections, native MySQL look & feel
- Available on Linux only
- No slave lag
- No lost transactions
- Both read and write scalability
- Smaller client latencies

Galera Cluster

- Provides virtually synchronous replication
- Works with InnoDB
- No slave lag
- Transactions are validated by slave on Transaction commit
  - **Certification based replication**
- Master – Master or Master – Slave is possible



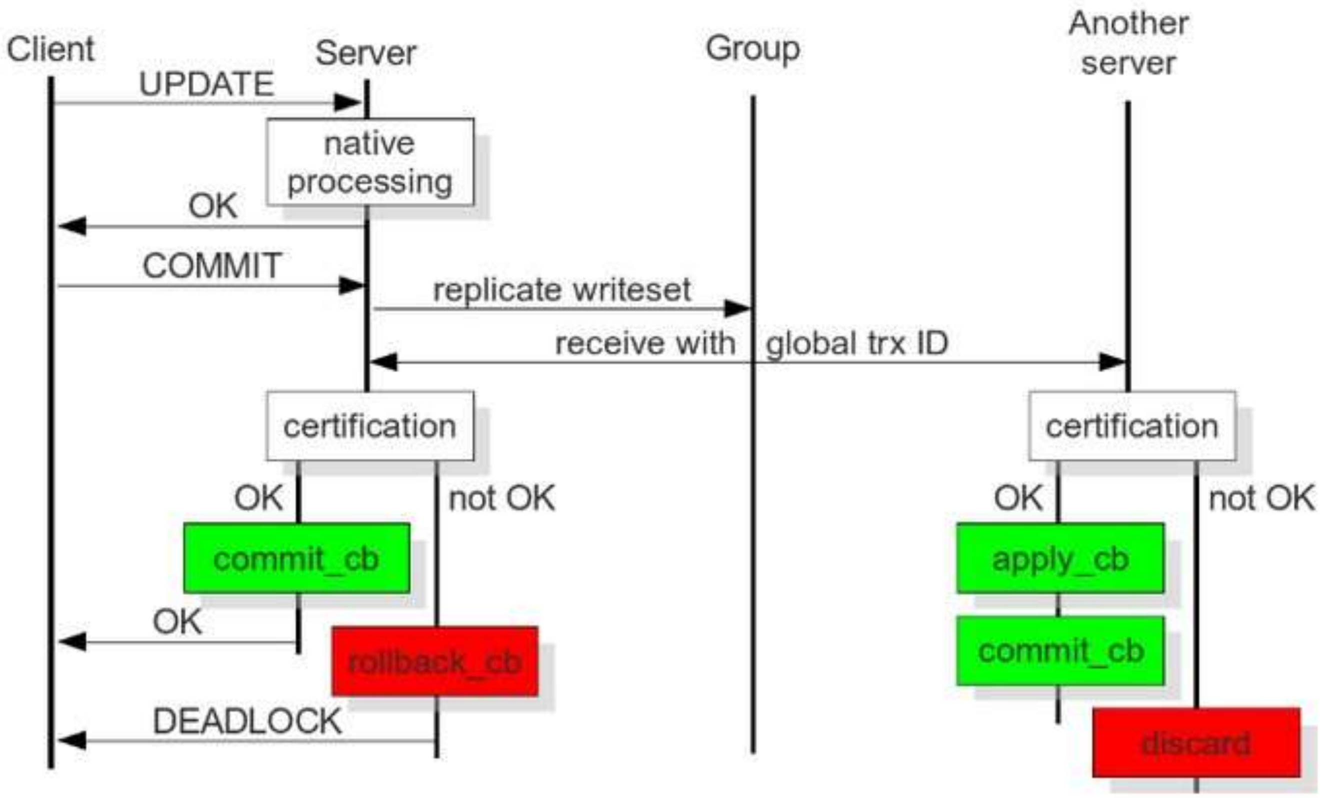
Certification Based Replication

- Certification-based replication uses group communication and transaction ordering techniques to achieve synchronous replication.

What Certification Based Replication Requires

- Transactional Database
- Atomic Changes
- Global Ordering

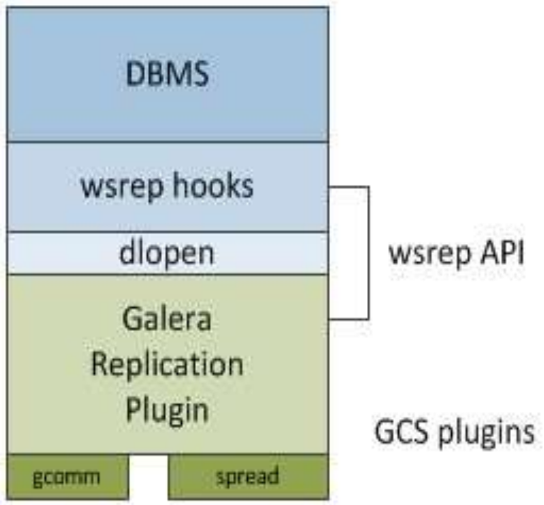
Certification Based Replication



Internal Architecture of Galera Cluster

Galera Cluster revolves around four components:

- **Database Management System (DBMS)**
- **wsrep API**
  - wsrep hooks
  - dlopen()
- **Galera Replication Plugin**
- **Group Communication Plugins**



State Transfers

The process of replicating data from the cluster to the individual node, bringing the node into sync with the cluster, is known as provisioning.

Methods available in Galera Cluster to provision nodes

- State Snapshot Transfers (SST) Where a snapshot of the entire node state transfers.
- Incremental State Transfers (IST) Where only the missing transactions transfer.

State Snapshot Transfers (SST)

The cluster provisions nodes by transferring a full data copy from one node to another.

- Logical
  - mysqldump
- Physical
  - rsync
  - xtrabackup

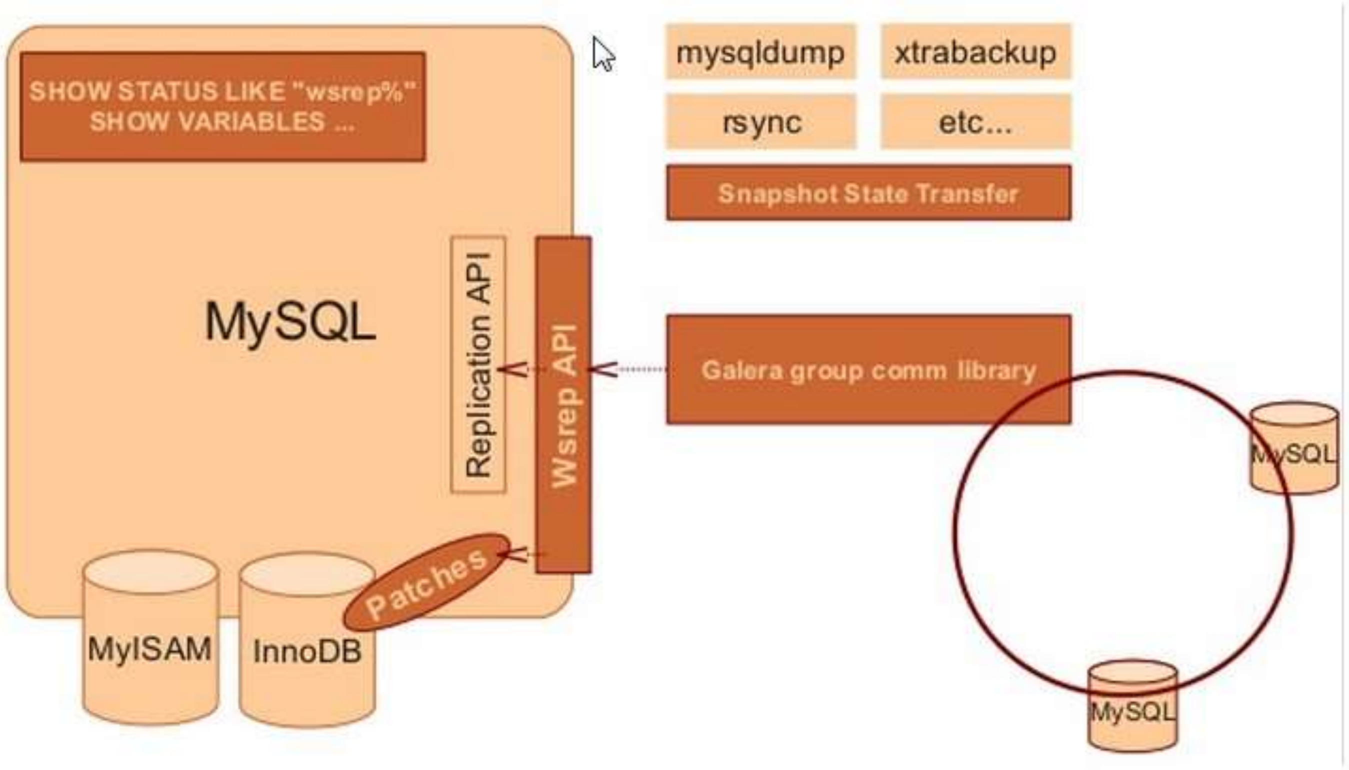
Incremental State Transfers (IST)

The cluster provisions a node by identifying the missing transactions on the joiner and sends them only, instead of the entire state.

- This provisioning method is only available under certain conditions:
  - Where the joiner node state UUID is the same as that of the group.
  - Where all missing write-sets are available in the donor's write-set cache.

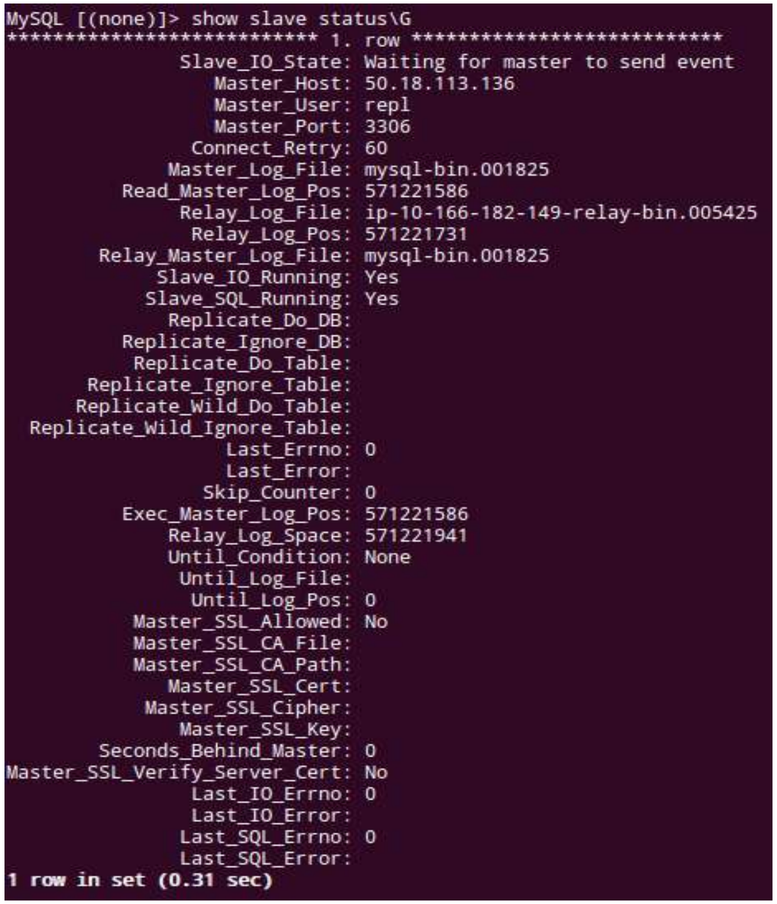
- For example, say that you have a node in your cluster that falls behind the cluster.
- This node carries a node state that reads: **5a76ef62-30ec-11e1-0800-dba504cf2aab:197222** Meanwhile, the current node state on the cluster reads **5a76ef62-30ec-11e1-0800-dba504cf2aab:201913**

Galera Cluster is



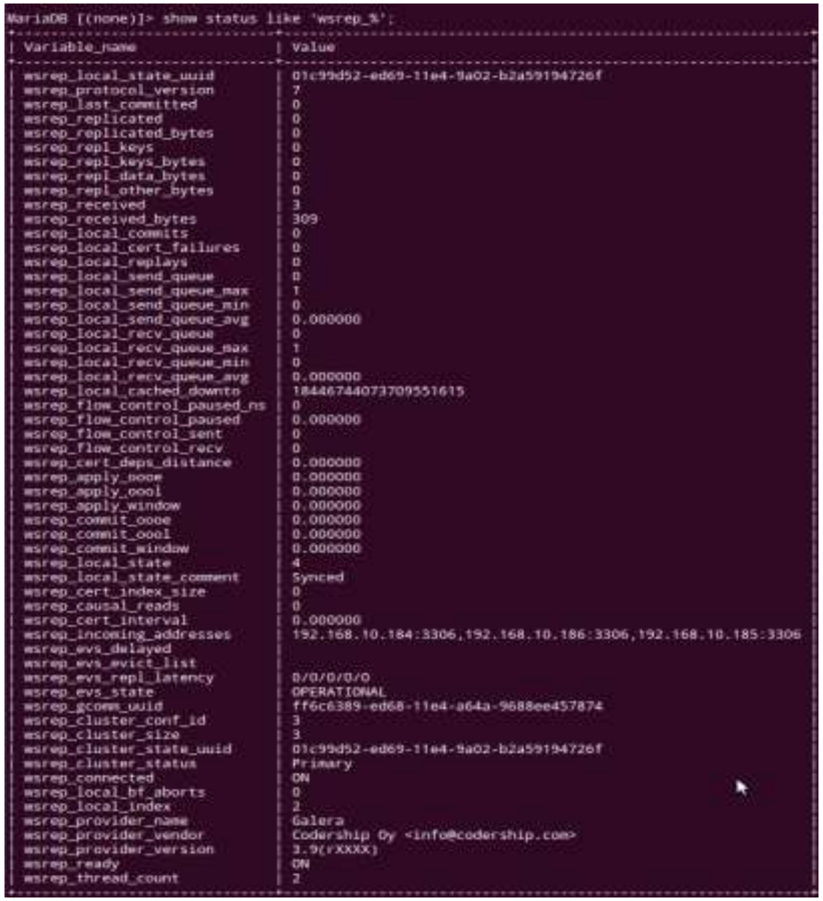
Important Status Variables

MySQL Native Replication



```
MySQL [(none)]> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 50.18.113.136
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.001825
          Read_Master_Log_Pos: 571221586
               Relay_Log_File: ip-10-166-182-149-relay-bin.005425
                Relay_Log_Pos: 571221731
        Relay_Master_Log_File: mysql-bin.001825
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 571221586
              Relay_Log_Space: 571221941
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
1 row in set (0.31 sec)
```

Galera Cluster



```
MariaDB [(none)]> show status like 'wsrep_%';
+--------------------------------+------------------------------------------------------------------+
| Variable_name                  | Value                                                            |
+--------------------------------+------------------------------------------------------------------+
| wsrep_local_state_uuid         | 01c99d52-ed69-11e4-9a02-b2a59194726f                             |
| wsrep_protocol_version         | 7                                                                |
| wsrep_last_committed           | 0                                                                |
| wsrep_replicated               | 0                                                                |
| wsrep_replicated_bytes         | 0                                                                |
| wsrep_repl_keys                | 0                                                                |
| wsrep_repl_keys_bytes          | 0                                                                |
| wsrep_repl_data_bytes          | 0                                                                |
| wsrep_repl_other_bytes         | 0                                                                |
| wsrep_received                 | 3                                                                |
| wsrep_received_bytes           | 309                                                              |
| wsrep_local_commits            | 0                                                                |
| wsrep_local_cert_failures      | 0                                                                |
| wsrep_local_replays            | 0                                                                |
| wsrep_local_send_queue         | 0                                                                |
| wsrep_local_send_queue_max     | 1                                                                |
| wsrep_local_send_queue_min     | 0                                                                |
| wsrep_local_send_queue_avg     | 0.000000                                                         |
| wsrep_local_recv_queue         | 0                                                                |
| wsrep_local_recv_queue_max     | 1                                                                |
| wsrep_local_recv_queue_min     | 0                                                                |
| wsrep_local_recv_queue_avg     | 0.000000                                                         |
| wsrep_local_cached_downto      | 18446744073709551615                                             |
| wsrep_flow_control_paused_ns   | 0                                                                |
| wsrep_flow_control_paused      | 0.000000                                                         |
| wsrep_flow_control_sent        | 0                                                                |
| wsrep_flow_control_recv        | 0                                                                |
| wsrep_cert_deps_distance       | 0.000000                                                         |
| wsrep_apply_oooe               | 0.000000                                                         |
| wsrep_apply_oool               | 0.000000                                                         |
| wsrep_apply_window             | 0.000000                                                         |
| wsrep_commit_oooe              | 0.000000                                                         |
| wsrep_commit_oool              | 0.000000                                                         |
| wsrep_commit_window            | 0.000000                                                         |
| wsrep_local_state              | 4                                                                |
| wsrep_local_state_comment      | Synced                                                           |
| wsrep_cert_index_size          | 0                                                                |
| wsrep_causal_reads             | 0                                                                |
| wsrep_cert_interval            | 0.000000                                                         |
| wsrep_incoming_addresses       | 192.168.10.184:3306,192.168.10.186:3306,192.168.10.185:3306      |
| wsrep_evs_delayed              |                                                                  |
| wsrep_evs_evict_list           |                                                                  |
| wsrep_evs_repl_latency         | 0/0/0/0/0                                                        |
| wsrep_evs_state                | OPERATIONAL                                                      |
| wsrep_gcomm_uuid               | ff6c6389-ed68-11e4-a64a-9688ee457874                             |
| wsrep_cluster_conf_id          | 3                                                                |
| wsrep_cluster_size             | 3                                                                |
| wsrep_cluster_state_uuid       | 01c99d52-ed69-11e4-9a02-b2a59194726f                             |
| wsrep_cluster_status           | Primary                                                          |
| wsrep_connected                | ON                                                               |
| wsrep_local_bf_aborts          | 0                                                                |
| wsrep_local_index              | 2                                                                |
| wsrep_provider_name            | Galera                                                           |
| wsrep_provider_vendor          | Codership Oy <info@codership.com>                                |
| wsrep_provider_version         | 3.9(rXXXX)                                                       |
| wsrep_ready                    | ON                                                               |
| wsrep_thread_count             | 2                                                                |
+--------------------------------+------------------------------------------------------------------+
```

Implementation

MariaDB 3 Node Cluster Implementation

How To Configure a Galera Cluster with MariaDB on Ubuntu 12.04 Servers

- Basic set up for MariaDB Galera Cluster requires minimum of 3 Nodes
- In our demo we will be using 3 Linux ubuntu 12.04 nodes
- In order to configure cluster we will be following some basic steps on every node , on Node1 execute following steps
- **STEP 1 : Remove old MySQL if already installed for fresh installation**
  - apt-get remove --purge mysql-server mysql-client mysql-common
  - apt-get autoremove
  - apt-get autoclean
- **STEP 2 : Add the MariaDB Repositories**
  - apt-get install python-software-properties
  - apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 0xcbcb082a1bb943db
  - add-apt-repository 'deb http://mirror3.layerjet.com/mariadb/repo/10.0/ubuntu precise main‘
  - apt-get update
- **STEP 3 : Install MariaDB with Galera Patches**
  - apt-get install rsync
  - apt-get install galera
  - apt-get install mariadb-galera-server

How To Configure a Galera Cluster with MariaDB on Ubuntu 12.04 Servers

- **STEP 4 : Configure MariaDB and Galera**
  - Create Galera configuration file under /etc/mysql/conf.d/
  - Most basic Galera configuration

```
[mysqld]
#mysql settings
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
query_cache_size=0
query_cache_type=0
bind-address=0.0.0.0
#galera settings
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_cluster_name="my_wsrep_cluster"
wsrep_cluster_address="gcomm://192.168.10.184,192.168.10.185,192.168.10.186“    ## IP of nodes who will be part of cluster
wsrep_sst_method=rsync
```

- **STEP 1 to STEP 4 will be followed on every Node**

Starting the Galera Cluster

- After Step 4 MariaDB is ready to run individually on every node

Starting nodes to run as part of cluster

- Stop MySQL on all nodes
  - service mysql stop
- On Node1 Start MySQL to run as cluster
  - service mysql start --wsrep-new-cluster
- On All other nodes start mysql simply as
  - service mysql start
  - Check the variable “**wsrep_cluster_size%**”

Thank You!

We love to connect with you.

Twitter	(@OSSCubeIndia)
Facebook	(www.facebook.com/osscubeindia)
LinkedIn	(www.linkedin.com/company/osscube)
Google+	(plus.google.com/u/0/+OSSCubeIndia/posts)