Galera Cluster for MySQL and Master/Slave replication

DOAG K + A, 2022, Nürnberg (D)

Oli Sennhauser

CTO, FromDual GmbH

https://www.fromdual.com/presentations

Why mixing?

- Galera wsrep + MySQL replication

  Why mixing?

- Galera Cluster (wsrep) is SO COOL! Why to bother with this old Master/Slave crap?

- → Because both technologies have their strengths and weaknesses!
- And working together we can combine the advantages and reduce the disadvantages...

Differences

Galera wsrep	MySQL Replication
True multi-master Cluster (write to any node)	Master/Slave Cluster (write to 1, read from many
Active/active Cluster (write to any node)	write to several Masters possible but not recom.
(Virtual) synchronous Replication (= semi-sync!)	Asynchronous Replication (semi-sync is possible)
Tightly coupled (same state, no diverged data allowed but back-coupling and delays possible!)	Loosely coupled (different state, data divergence is possible, no back-coupling)
Multi-threaded replication	Multi-threaded Slave
No M/S failover or VIP needed (but LB!)	Failover done with scripts an VIP or LB
Hot standby (minimal downtime during failover)	Slave becomes Master (short downtime possible)
Automatic node provisioning (and joining)	No automatic node provisioning (easy scriptable)
Support InnoDB (only!)	Supports all Storage Engines!
Transparent to Application (no or minimal changes)	Somebody needs to do r/w split (if needed)
No read/write splitting (many do that after conflicts)	Most user do not need r/w split
Easy to use (until you have a problem)	Easy to use (also when you have a problem)
Easy to deploy (OK, yes)	Easy to deploy (a bit more difficult than Galera SST)
No replication lag (what about FC, back-coupling?)	Slave lag possible (what about no back-coupling?)
Read scalability	Read scalability
https://galeracluster.com/products/	.

Advantages – reality check

- Galera wsrep

- Active/active M/M Cluster
- Automatic failover
- Failover within seconds
- No SpoF
- Synchronous replication → No lost transaction
- Very strict

- .

- MySQL Replication

- Stop Slave is possible
- Upgrades / different versions
- Artificial delay is possible
- Async replication (no back-couplings from Slave to Master)
- Different Storage Engines possible
- Not so strict
- Copes well with unstable network

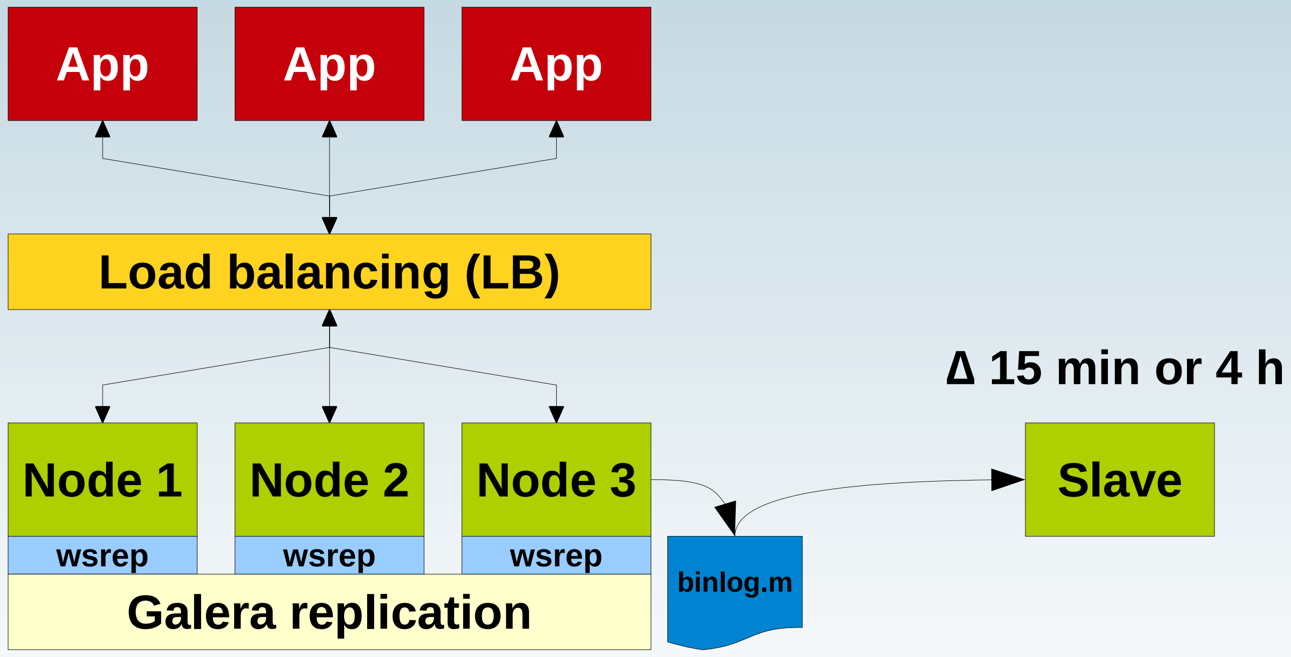
Reasons to combine

- Some features the other technology does not provide!

- Stopping Replication for some reasons (no back-couplings)
- Back-couplings because of flowcontrol
  - Backup or Reporting
- Artificially delayed replication
- Filtering on Schema/Table level
- Different table definitions (attribute promotion/demotion)
- Different Storage Engine (Column Store, etc.)
- Read-only Node/Slave
  - I did not try if this would work with a Galera node at all?
- Upgrade and/or fail-back over many releases!
- 2 DC with high latency or unstable network in between

Use cases I

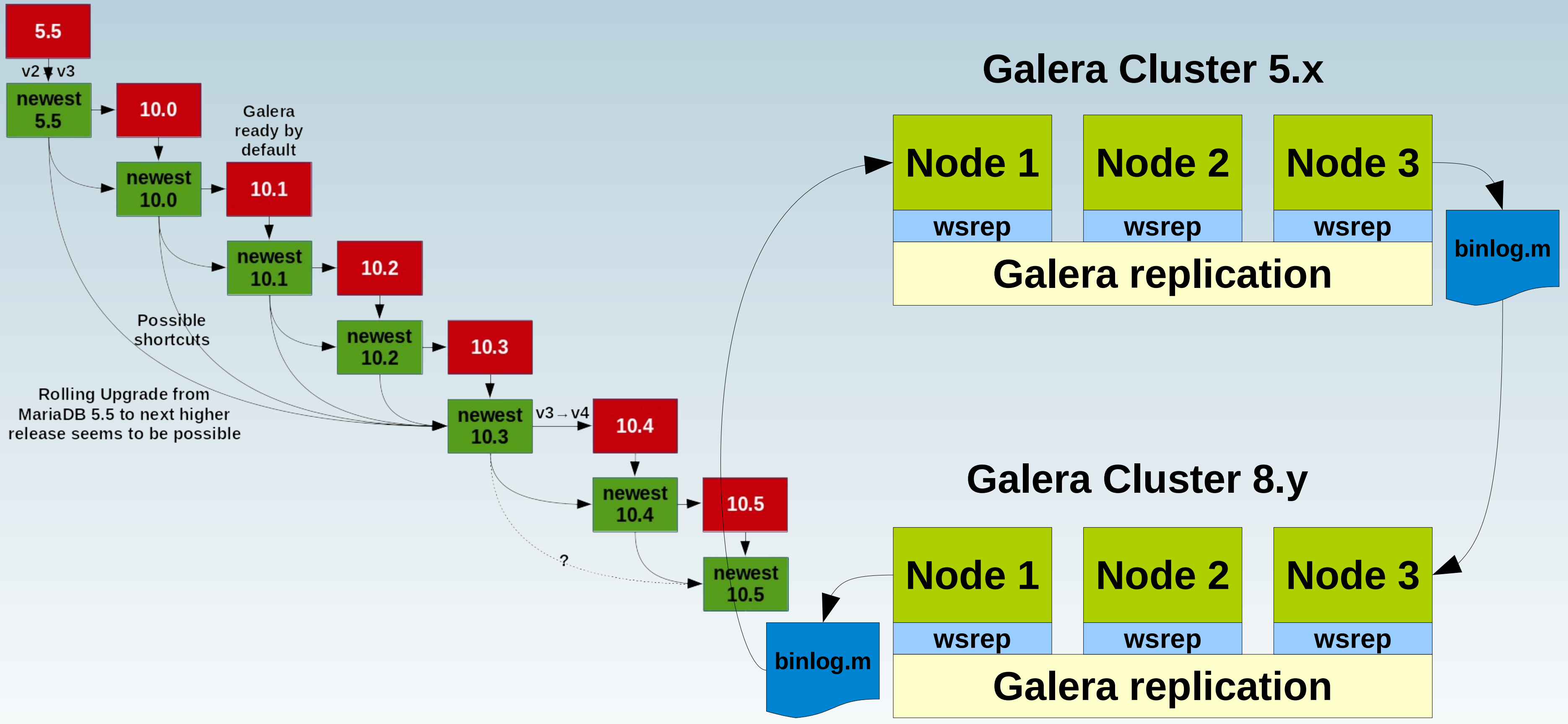
- Artificially delayed replication
  - Stock trading (pro vs. free (15 min delay))
  - Logical errors / Oops! queries (4 h time)



Use cases II

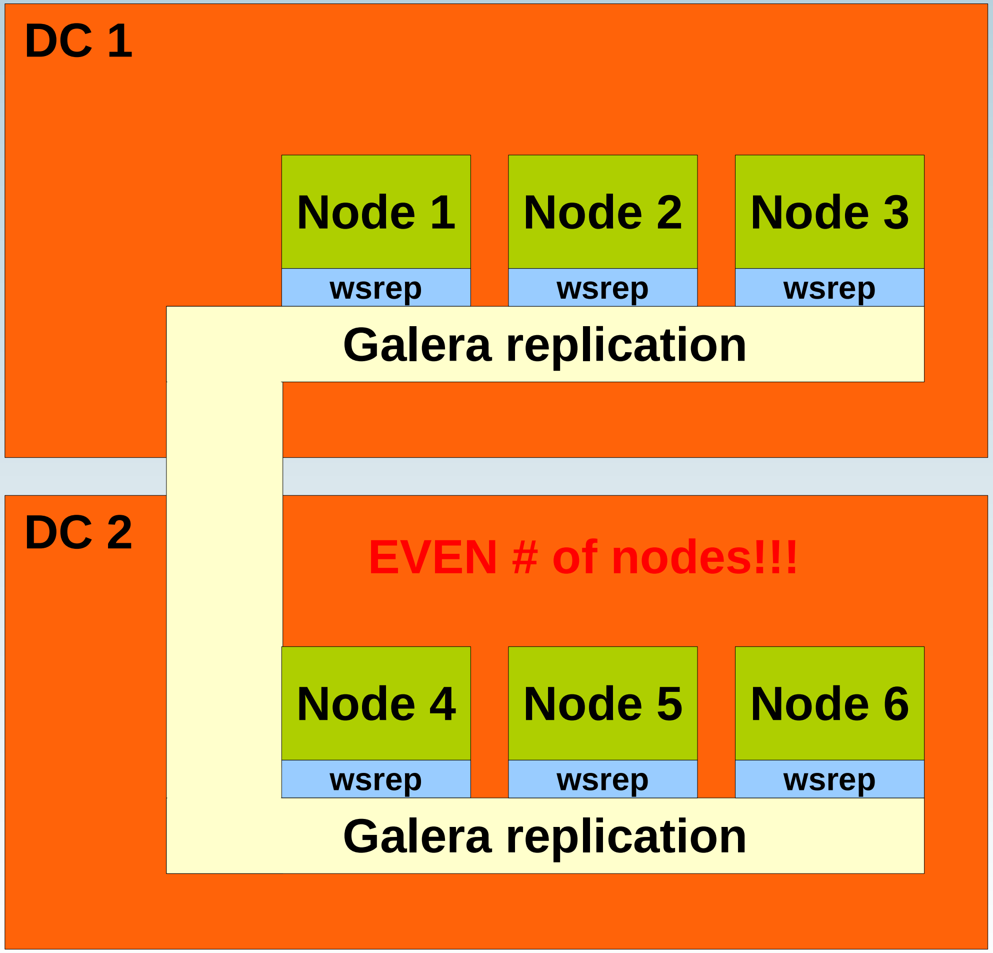
www.fromdual.com

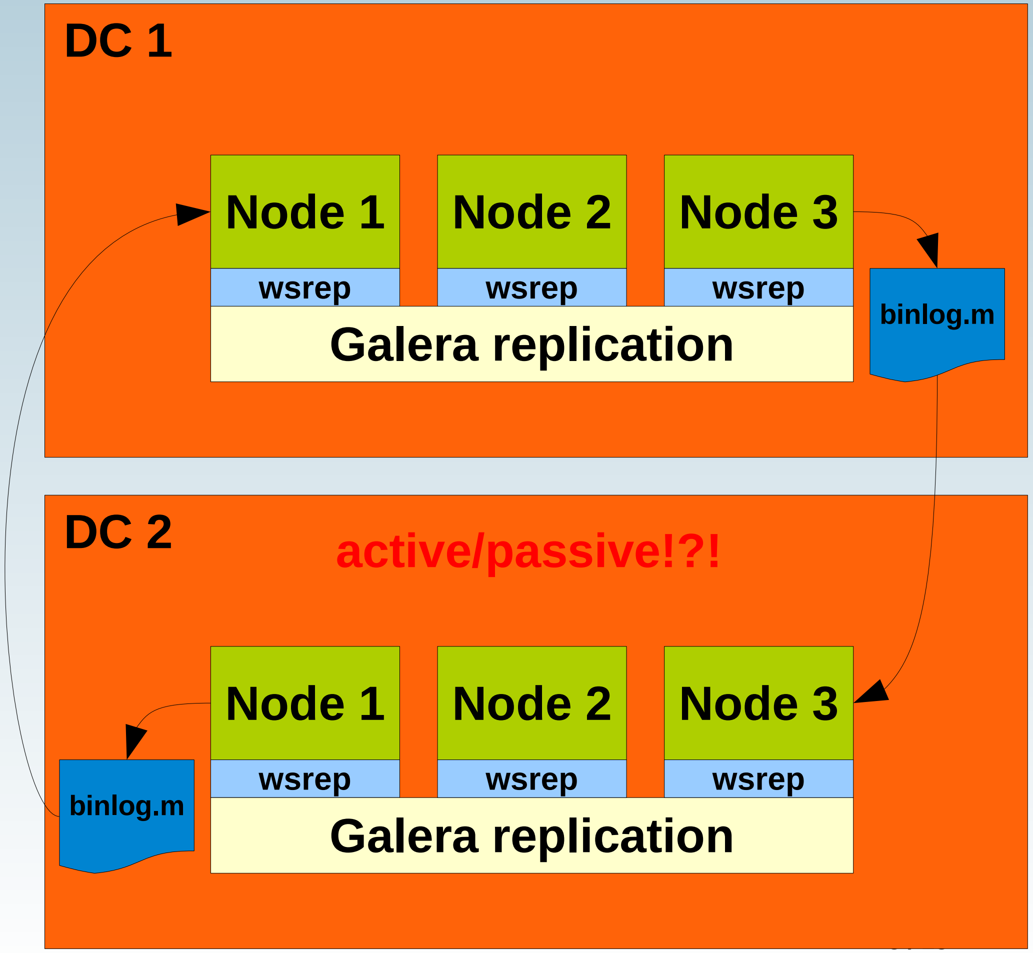
- Upgrade and/or
- Fail-back over many releases!



Use cases III

- 2 DC with high latency or unstable network in between



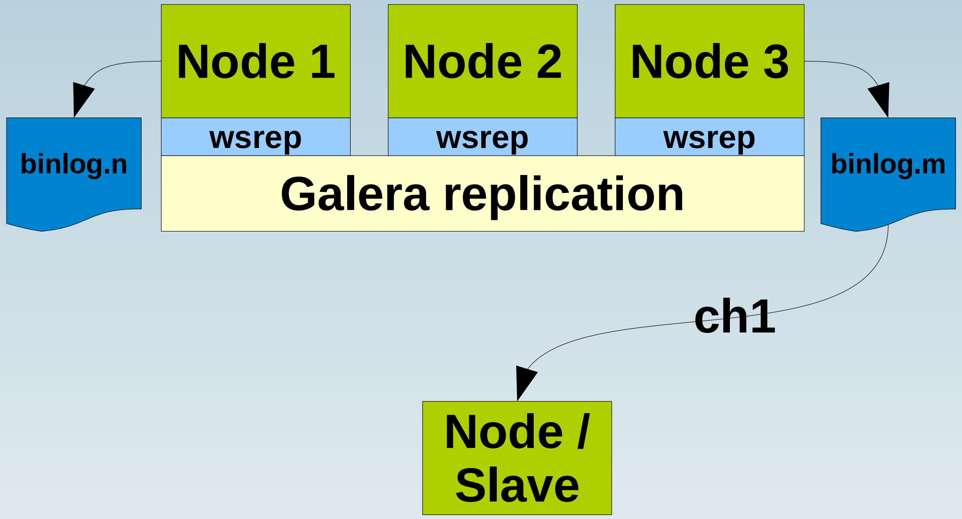


Challenge?

- Binary Logs
  - `log_slave_updates = ON` !!!
- What happens in case of an IST?
  - Binary Logs will be continued → No problem!
- What happens in case of a SST?
  - Binary Logs are lost!
  - You have to re-setup/fix your Slave-Cluster :-(
- → Switch the Binary Log Channel

Switch Binary Log Channel

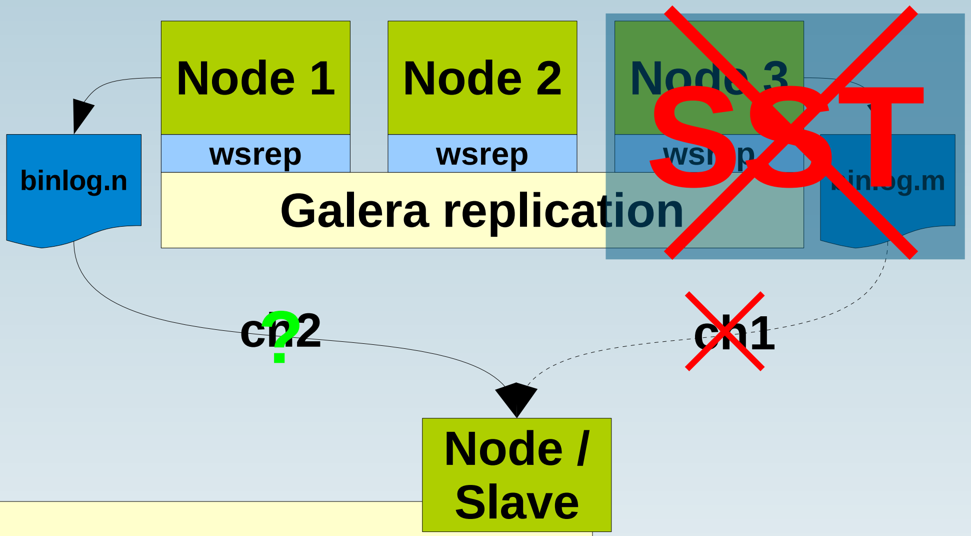
- To make it easier: Just one slave node



Switch binary log channel

www.fromdual.com

- To make it easier: Just one slave node



```
SQL> SHOW SLAVE STATUS\G
    Master_Log_File: mygal-80-c_binlog.000005
Read_Master_Log_Pos: 447
   Slave_IO_Running: No
      Last_IO_Errno: 1236
      Last_IO_Error: Got fatal error 1236 from
master when reading data from binary log: 'Could
not find first log file name in binary log index file'
```

```
SQL> SHOW MASTER STATUS;
+----------------------------+----------+
| File                       | Position |
+----------------------------+----------+
| mygal-80-a_binlog.000001   | 99999999 |
+----------------------------+----------+
```

Channel failover classic

- With classical/physical method
  - Binary Log File and Pos
- How?
  - Search last transaction/statement in Relay Log of Slave:
    - `SHOW SLAVE STATUS\G`
    - `SHOW RELAYLOG EVENTS IN ...` or
    - `mysqlbinlog --verbose relay-bin.*`
  - Search equivalent in Binary Log of Master of Channel 2:
    - `SHOW BINLOG EVENTS IN ...` or
    - `mysqlbinlog --verbose binlog.*`
  - Point Slave to same position of new Master of Channel 2:
  - `STOP SLAVE;`
  - `CHANGE MASTER TO ...`
  - `START SLAVE;`

- Difficult to automatize!?!
- Laboriously!

Search last trx on Slave

- **On Slave:**

```
SQL> SHOW SLAVE STATUS\G
                Relay_Log_File: chef-relay-bin.000004
                 Relay_Log_Pos: 761

SQL> SHOW RELAYLOG EVENTS IN 'chef-relay-bin.000004';
+-----------------------+ ... +------------------------------------------+
| Log_name              | ... | Info                                     |
+-----------------------+ ... +------------------------------------------+
...
| chef-relay-bin.000004 | ... | table_id: 86 (test.test)                 |
| chef-relay-bin.000004 | ... | table_id: 86 flags: STMT_END_F           |
| chef-relay-bin.000004 | ... | COMMIT /* xid=17542 */                   |
+-----------------------+ ... +------------------------------------------+
```

Search last trx on new Master

- **On new Master of Channel 2:**

```
SQL> SHOW BINLOG EVENTS IN 'chef_mygal-80-c__binlog.000010';
+---------------------------------+------+ ... +---------------------------------+
| Log_name                        | Pos  | ... | Info                            |
+---------------------------------+------+ ... +---------------------------------+
...
| chef_mygal-80-c__binlog.000010  |  331 | ... | chef_mygal-80-c__binlog.000009  |
| chef_mygal-80-c__binlog.000010  |  447 | ... | COMMIT /* xid=17542 */          |
| chef_mygal-80-c__binlog.000010  | 1234 | ... | BEGIN                           |
...
```

- **Change Replication Channel:**

```
SQL> STOP SLAVE;
SQL> CHANGE MASTER TO master_host='192.168.1.1', master_port=3306
                    , master_log_file='chef_mygal-80-c__binlog.000010'
                    , master_log_pos=1234;
SQL> START SLAVE;
```

CAUTION!!!

- In MySQL 8.0 with GTID DISABLED this will lead to data inconsistencies between Master and Slave!!!
- Why: Binary Log is NOT purged/deleted and Slave will silently continue working.
- So you risk a gap in your replication stream.
- If binary logs a located in $datadir it may work correctly if you are lucky!
- So do NOT do this WITHOUT GTID!
- It may have worked in 5.5 to 5.7…
- https://github.com/codership/mysql-wsrep/issues/408

Channel failover with GTID

- With "modern" method: GTID
  - On Master and Slave:

```
gtid_mode                         = ON
enforce_gtid_consistency          = 1
```

- On Slave:

```
SQL> STOP SLAVE;
SQL> CHANGE MASTER TO master_host='192.168.1.1', master_port=3306
                    , master_auto_position = 1;
SQL> START SLAVE;
```

Literature

- **MySQL Cluster - Cluster circular replication with 2 replication channels**

  https://fromdual.com/mysql-cluster-circular-replication-with-channel-failover

- **Replication channel fail-over with Galera Cluster for MySQL**

  https://fromdual.com/replication-channel-fail-over-with-galera-cluster-for-mysql

Thank you!

Questions ?

Discussion?

We have some time for a personal talk...

FromDual provides neutral and independent:

- Consulting
- remote-DBA
- Support for MariaDB and Galera Cluster
- Training

www.fromdual.com/presentations