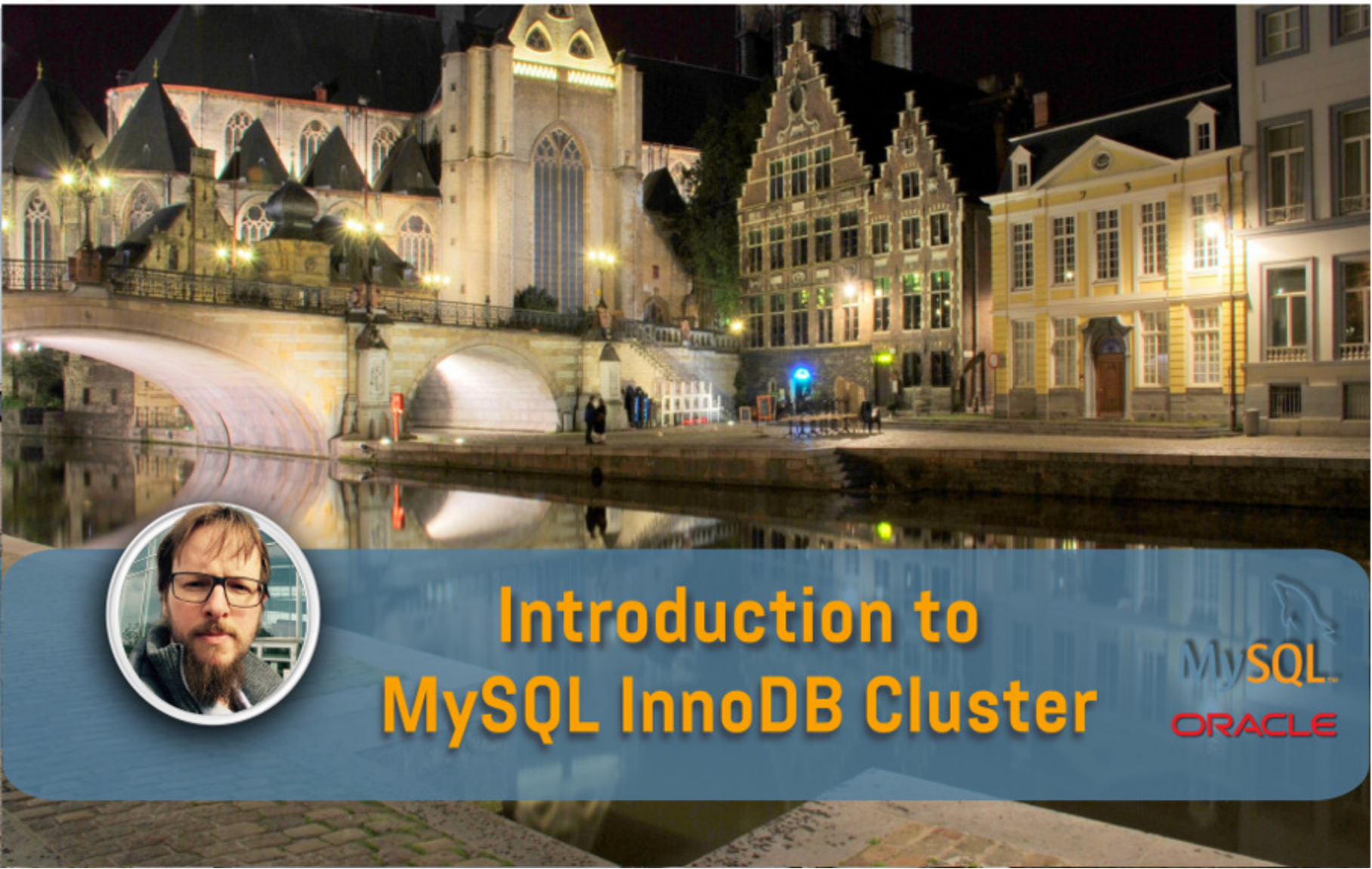
PERCONA UNIVERSITY

June 22,2017
5.00PM

Co.Station Ghent
Oktrooiplein 1, 9000 Ghent



Introduction to MySQL InnoDB Cluster

MySQL
ORACLE


ORACLE®

MySQL InnoDB Cluster

MySQL High Availability made easy

Percona University, Ghent, Belgium June 2017

Frédéric Descamps - MySQL Community Manager - Oracle

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purpose only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied up in making purchasing decisions. The development, release and timing of any features or functionality described for Oracle´s product remains at the sole discretion of Oracle.

about.me/lefred

Who am I ?

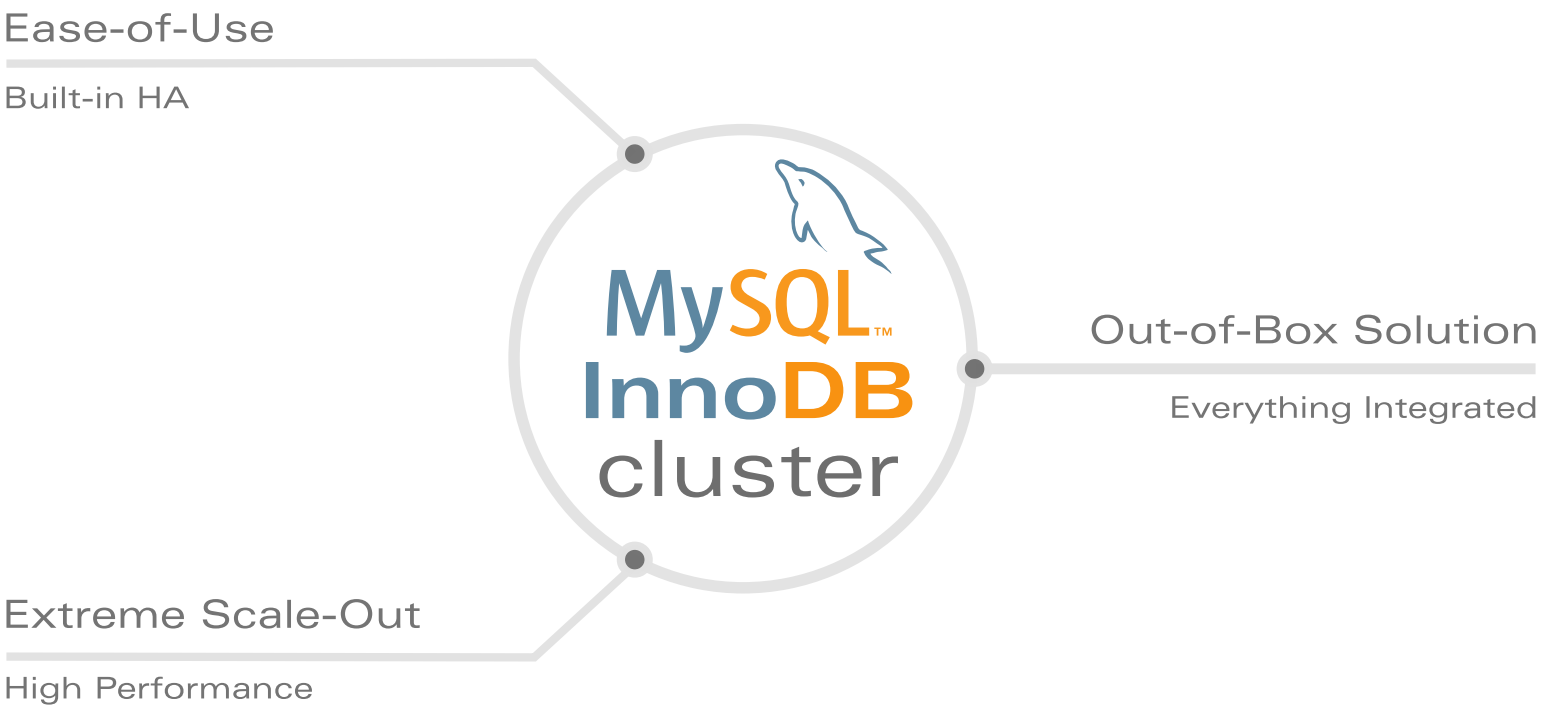
Frédéric Descamps

- @lefred
- MySQL Evangelist
- Hacking MySQL since 3.23
- devops believer
- MySQL Community Manager since May 2016
- living in Belgium **B E**

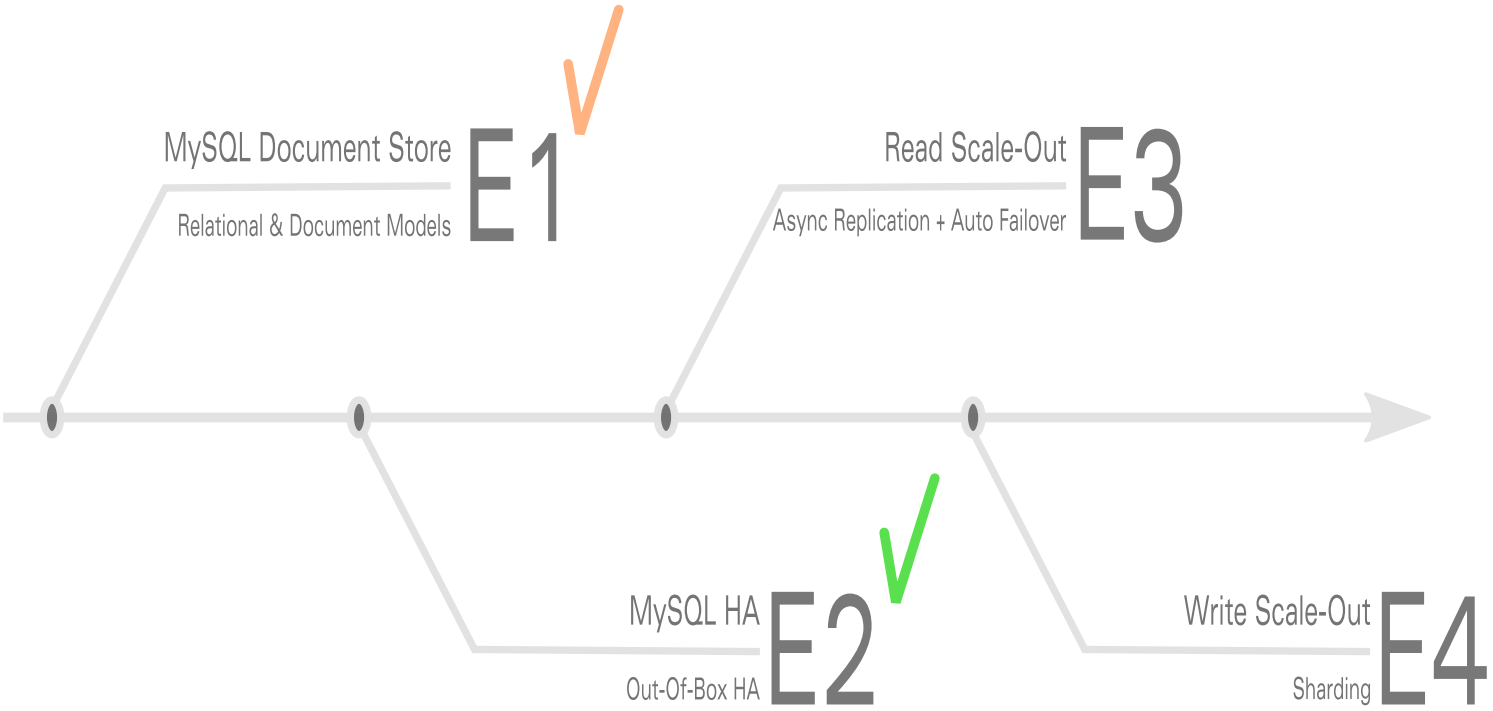
ORACLE

MySQL InnoDB Cluster

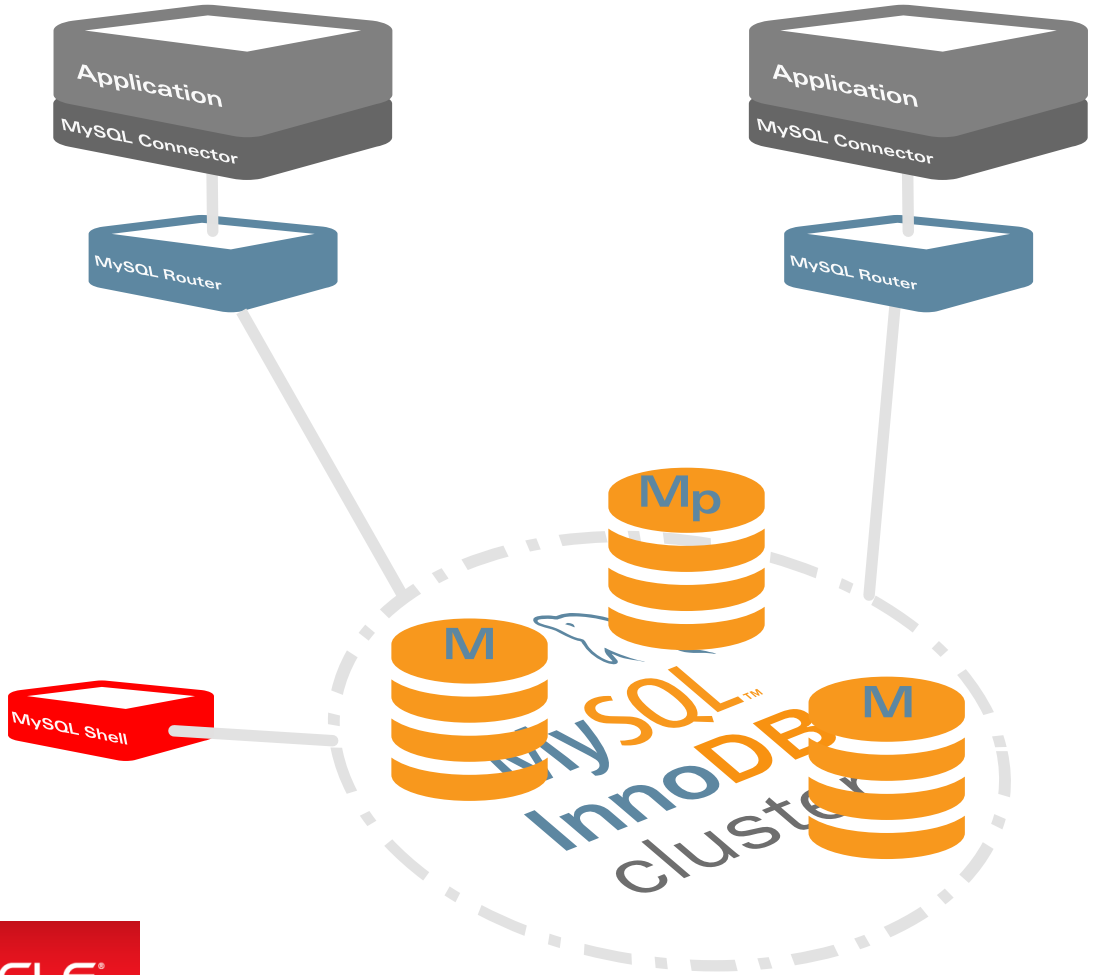
Easy MySQL High Availability



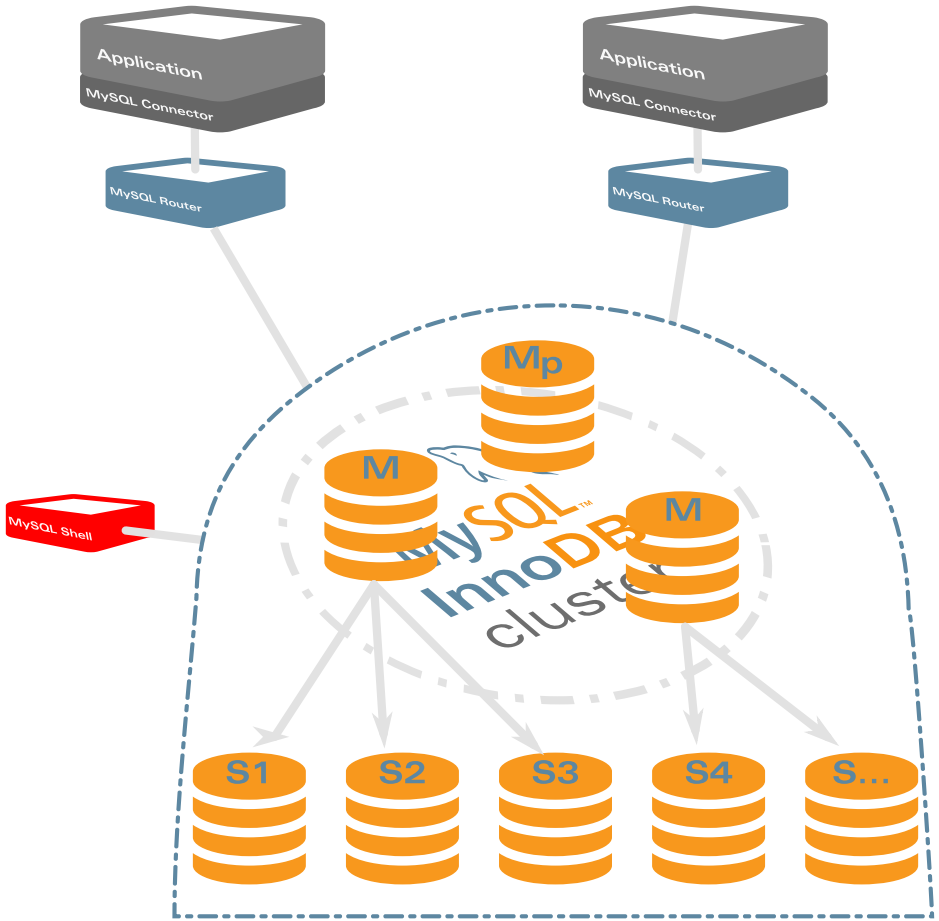
Our vision in 4 steps



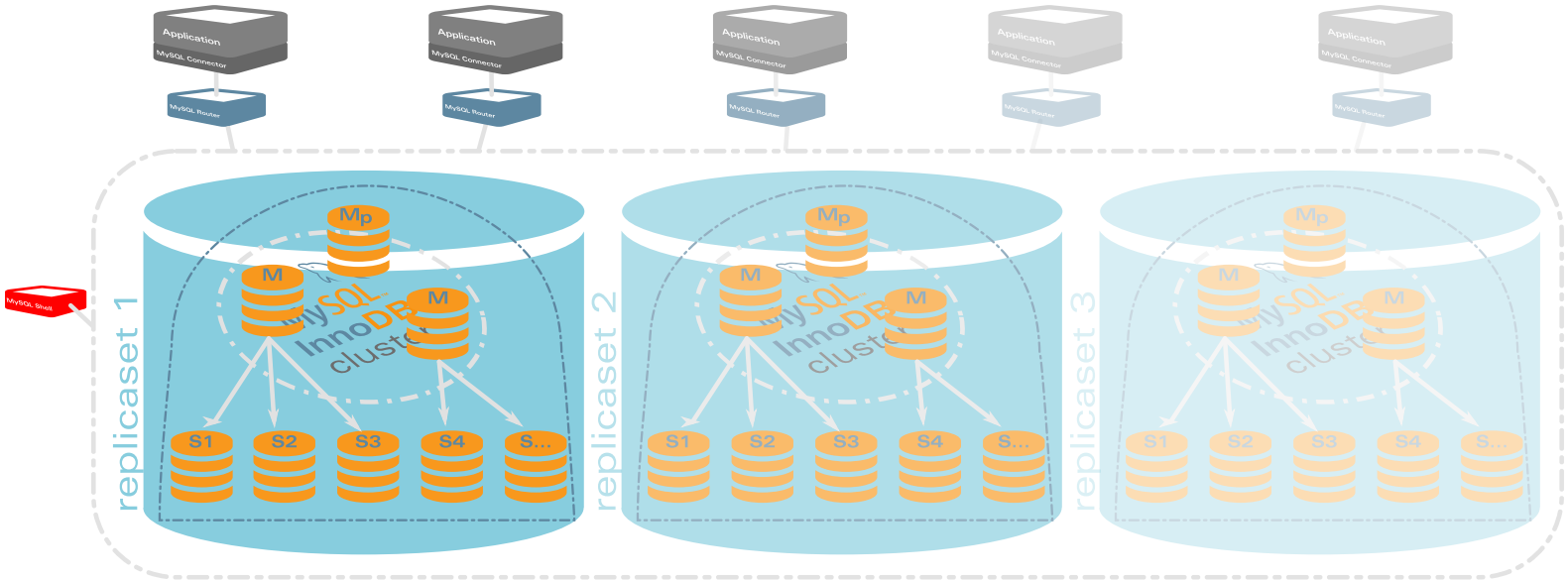
Step 2´s Architecture



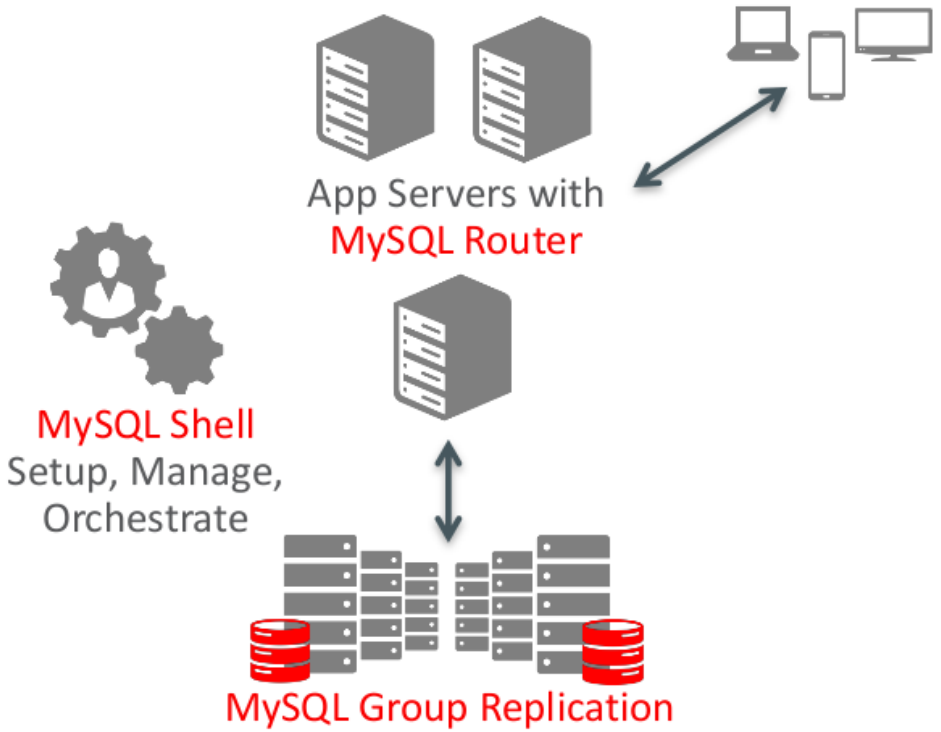
Step 3´s Architecture



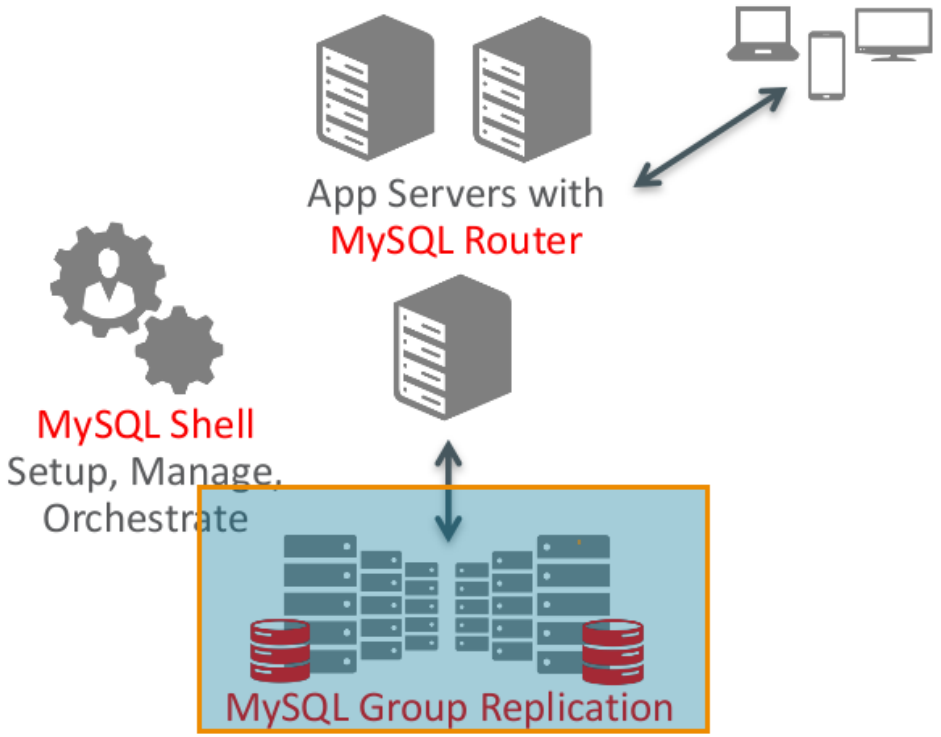
Step 4´s Architecture



Group Replication: heart of MySQL InnoDB Cluster



Group Replication: heart of MySQL InnoDB Cluster



But before going further...

What is

High Availability ?

High Availability

High availability is a characteristic of a system, which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.

There are three principles of systems design in reliability engineering which can help achieve high availability:

- *Elimination of single points of failure. This means adding **redundancy** to the system so that failure of a component does not mean failure of the entire system.*
- *Reliable crossover. In redundant systems, the crossover point itself tends to become a single point of failure. Reliable systems 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.*

Source: Wikipedia

Database Redundancy

How to achieve it ?

Database Redundancy

How to achieve it ?

Multiple solutions exist, some worse than the others ;-)

Database Redundancy

How to achieve it ?

Multiple solutions exist, some worse than the others ;-)

- use of share storage

Database Redundancy

How to achieve it ?

Multiple solutions exist, some worse than the others ;-)

- use of share storage
- use of share blocs by network (drbd)

Database Redundancy

How to achieve it ?

Multiple solutions exist, some worse than the others ;-)

- use of share storage
- use of share blocs by network (drbd)
- use of MySQL replication

Database Redundancy

How to achieve it ?

Multiple solutions exist, some worse than the others ;-)

- use of share storage
- use of share blocs by network (drbd)
- use of MySQL replication

Of course the last one is the technique most spread, more flexible and more reliable.

MySQL Replication

There are multiple types of MySQL replication:

MySQL Replication

There are multiple types of MySQL replication:

- asynchronous replication (async)

MySQL Replication

There are multiple types of MySQL replication:

- asynchronous replication (async)
- semi-synchronous replication (semi-sync)

MySQL Replication

There are multiple types of MySQL replication:

- asynchronous replication (async)
- semi-synchronous replication (semi-sync)
- group replication - *since (5.7.17) !!*

Asynchronous Replication(s)

- replicas (also called slaves) stream the replication logs from a unique master *(multi-source replication is possible since 5.7.6)*

Asynchronous Replication(s)

- replicas (also called slaves) stream the replication logs from a unique master *(multi-source replication is possible since 5.7.6)*
- slaves can be laging (having delay), there is no delivery guarantee *(unless for semi-sync)*

Asynchronous Replication(s)

- replicas (also called slaves) stream the replication logs from a unique master *(multi-source replication is possible since 5.7.6)*
- slaves can be laging (having delay), there is no delivery guarantee *(unless for semi-sync)*
- replicas provisioning is manual

Asynchronous Replication(s)

- replicas (also called slaves) stream the replication logs from a unique master *(multi-source replication is possible since 5.7.6)*
- slaves can be laging (having delay), there is no delivery guarantee *(unless for semi-sync)*
- replicas provisioning is manual
- data consistency on replicas is assumed and needs a manual verification

Asynchronous Replication(s)

- replicas (also called slaves) stream the replication logs from a unique master *(multi-source replication is possible since 5.7.6)*
- slaves can be laging (having delay), there is no delivery guarantee *(unless for semi-sync)*
- replicas provisioning is manual
- data consistency on replicas is assumed and needs a manual verification
- it´s even possible to create circular replication (**highly not recommended**)

Asynchronous Replication(s)

- replicas (also called slaves) stream the replication logs from a unique master *(multi-source replication is possible since 5.7.6)*
- slaves can be laging (having delay), there is no delivery guarantee *(unless for semi-sync)*
- replicas provisioning is manual
- data consistency on replicas is assumed and needs a manual verification
- it´s even possible to create circular replication (**highly not recommended**)
- write operations are safe only from a unique global master

Asynchronous Replication(s)

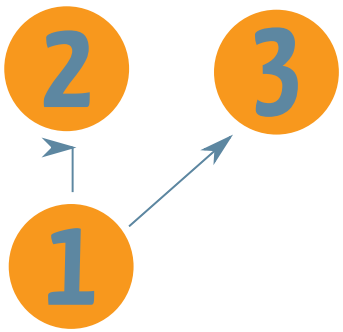
- complex topologies can be built

Asynchronous Replication(s)

- complex topologies can be built

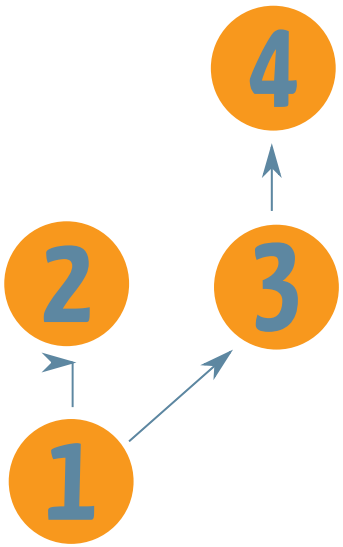
Asynchronous Replication(s)

- complex topologies can be built



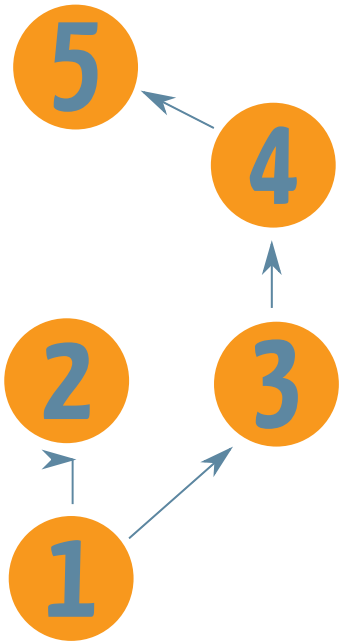
Asynchronous Replication(s)

- complex topologies can be built



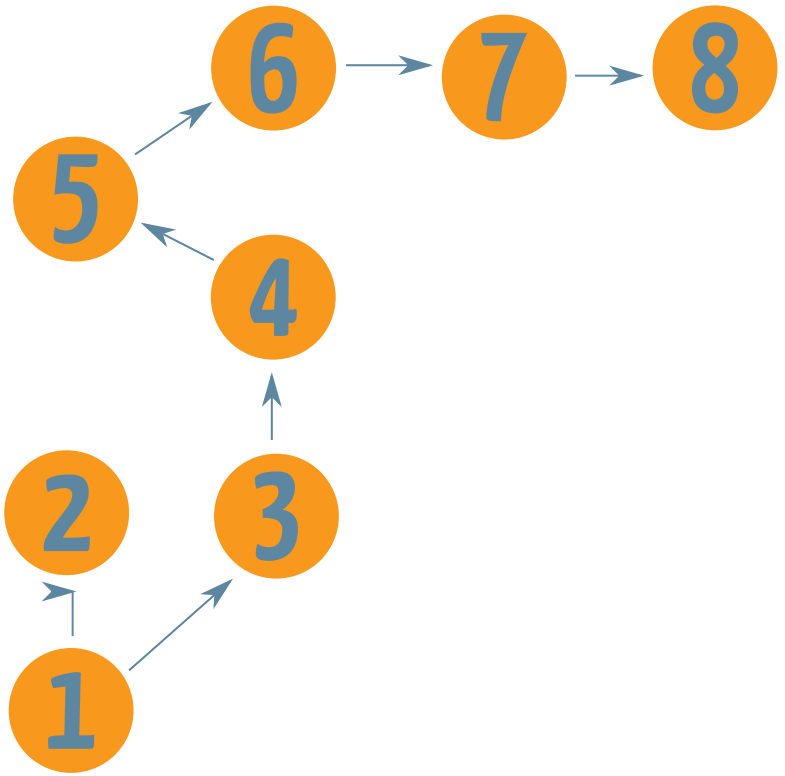
Asynchronous Replication(s)

- complex topologies can be built



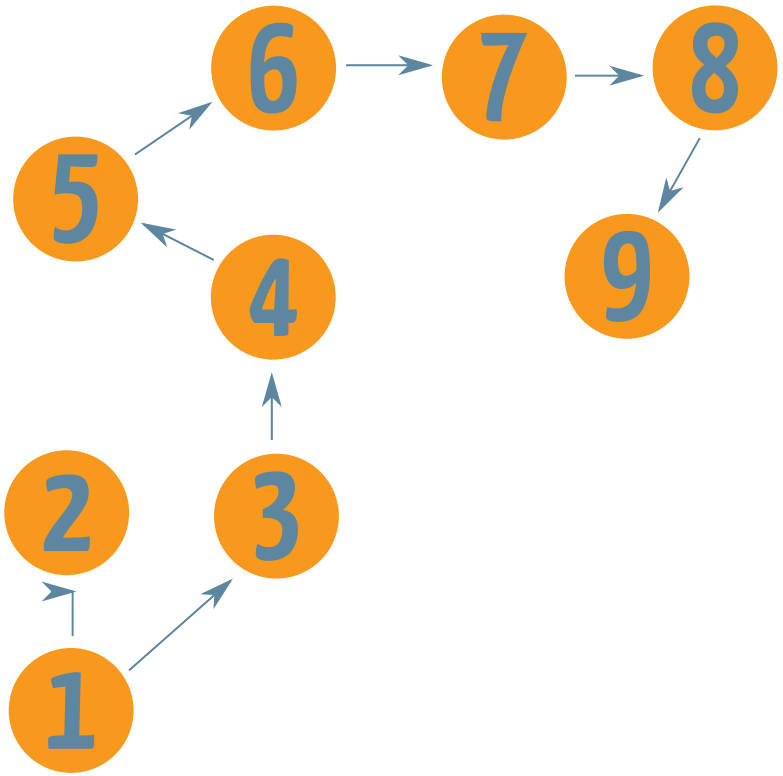
Asynchronous Replication(s)

- complex topologies can be built



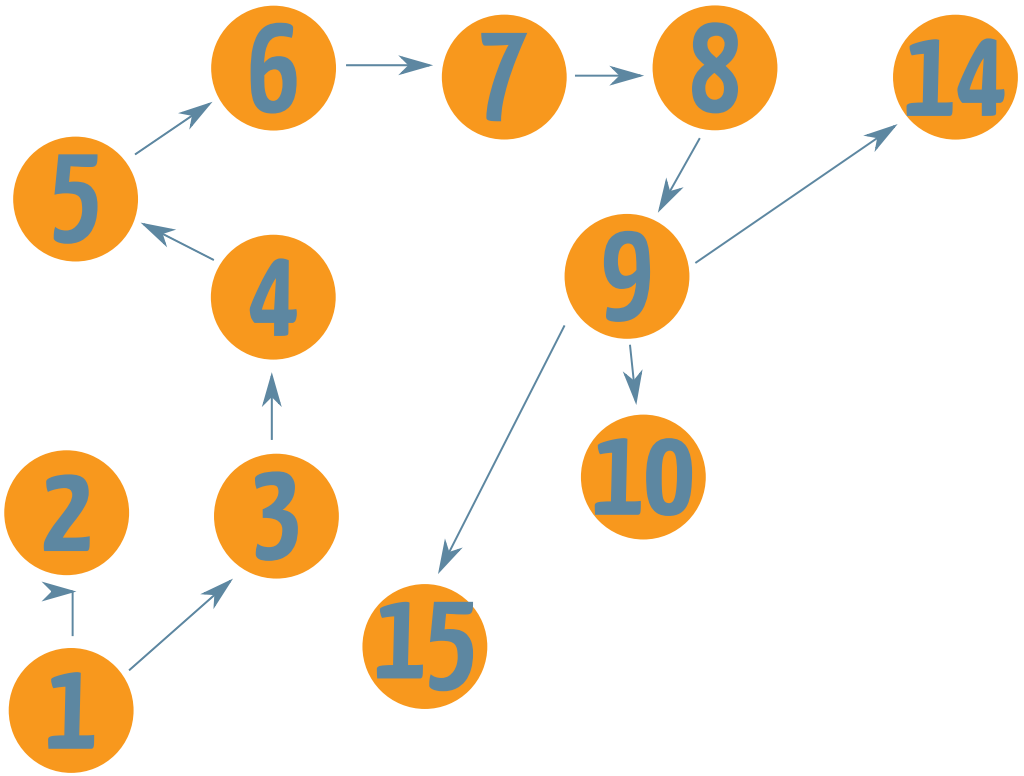
Asynchronous Replication(s)

- complex topologies can be built



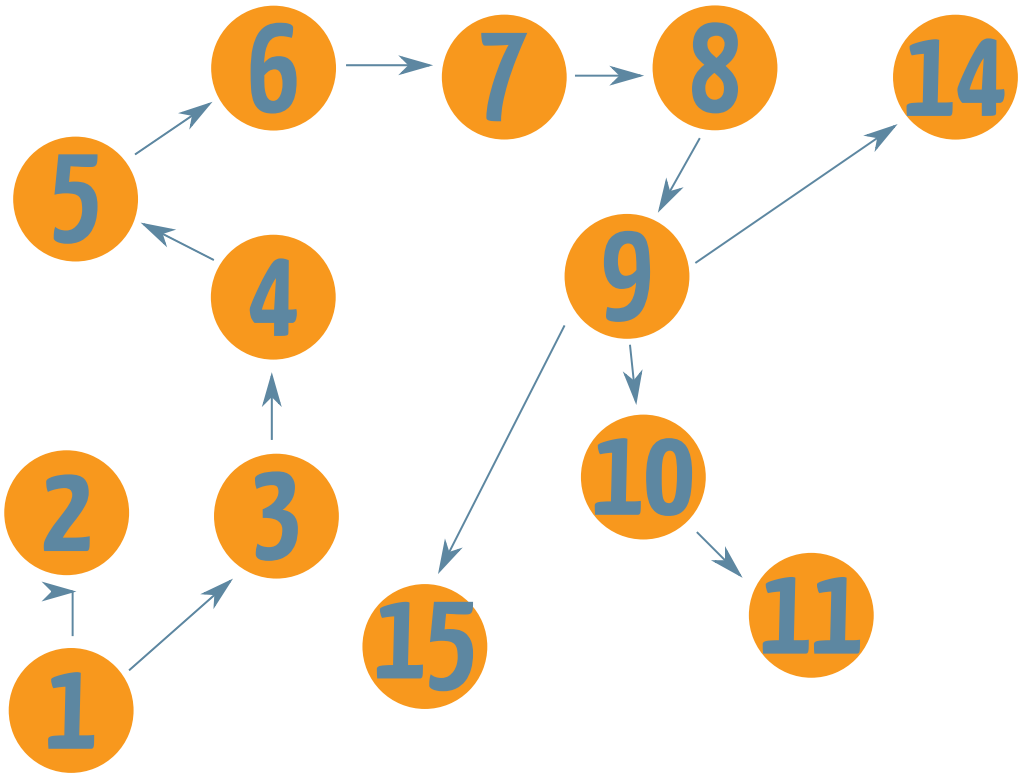
Asynchronous Replication(s)

- complex topologies can be built



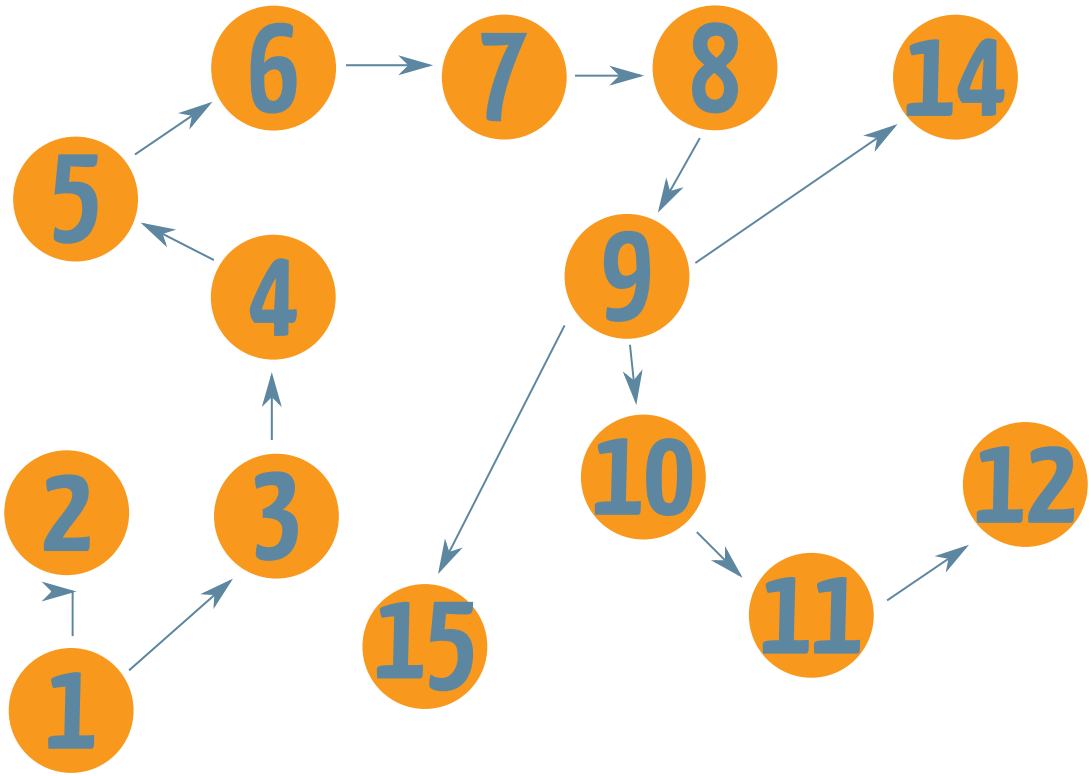
Asynchronous Replication(s)

- complex topologies can be built



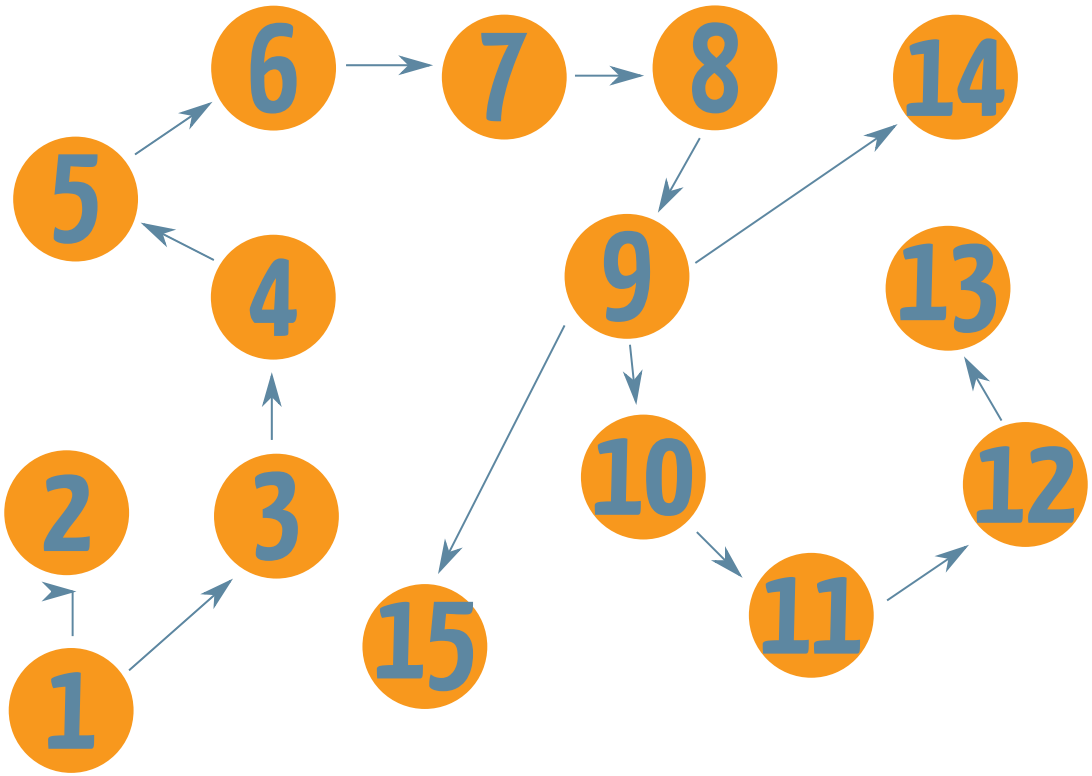
Asynchronous Replication(s)

- complex topologies can be built



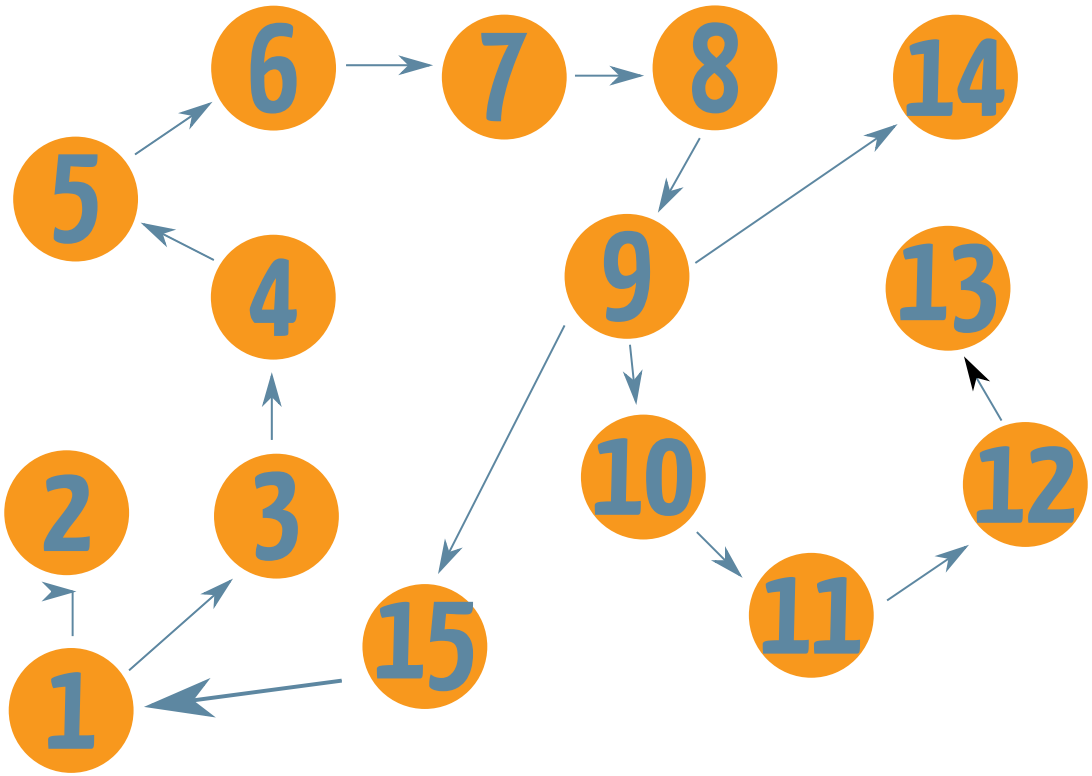
Asynchronous Replication(s)

- complex topologies can be built



Asynchronous Replication(s)

- complex topologies can be built



MySQL HA Architectures with asynchronous replication

It exists a lot of possible architectures and as much tools to enforce them.

MySQL HA Architectures with asynchronous replication

It exists a lot of possible architectures and as much tools to enforce them.

They are all based on the same principle:

MySQL HA Architectures with asynchronous replication

It exists a lot of possible architectures and as much tools to enforce them.

They are all based on the same principle:

- 1 master

MySQL HA Architectures with asynchronous replication

It exists a lot of possible architectures and as much tools to enforce them.

They are all based on the same principle:

- 1 master
- 1 or many replicas (slaves)

MySQL HA Architectures with asynchronous replication

It exists a lot of possible architectures and as much tools to enforce them.

They are all based on the same principle:

- 1 master
- 1 or many replicas (slaves)

If the master as an issue, the most accurate slave needs to takeover the role and be promoted as master for all the remaining slaves still online.

Automation

Automation

And this is the most complicated part...

Automation

And this is the most complicated part...

It exists a large amount of external tools that can be used to achieve this task. Most of them are only compatible with GNU/Linux.

Automation

And this is the most complicated part...

It exists a large amount of external tools that can be used to achieve this task. Most of them are only compatible with GNU/Linux.

Their problem is that they create complexity in the architecture's design. You need to be DBA & sysadmin to manage those solutions:

Automation

And this is the most complicated part...

It exists a large amount of external tools that can be used to achieve this task. Most of them are only compatible with GNU/Linux.

Their problem is that they create complexity in the architecture's design. You need to be DBA & sysadmin to manage those solutions:

- MySQL-Utilities (`mysqlrplcheck`, `mysqlrpladmin`)

Automation

And this is the most complicated part...

It exists a large amount of external tools that can be used to achieve this task. Most of them are only compatible with GNU/Linux.

Their problem is that they create complexity in the architecture's design. You need to be DBA & sysadmin to manage those solutions:

- MySQL-Utilities (`mysqlrplcheck`, `mysqlrpladmin`)
- Pacemaker with MySQL dedicated OCF

Automation

And this is the most complicated part...

It exists a large amount of external tools that can be used to achieve this task. Most of them are only compatible with GNU/Linux.

Their problem is that they create complexity in the architecture's design. You need to be DBA & sysadmin to manage those solutions:

- MySQL-Utilities (`mysqlrplcheck`, `mysqlrpladmin`)
- Pacemaker with MySQL dedicated OCF
- custom solutions...

MySQL Group Replication

but what is it ?!?

MySQL Group Replication

but what is it ?!?

- GR is a plugin for MySQL, made by MySQL and packaged with MySQL

MySQL Group Replication

but what is it ?!?

- GR is a plugin for MySQL, made by MySQL and packaged with MySQL
- GR is an implementation of Replicated Database State Machine theory

MySQL Group Replication

but what is it ?!?

- GR is a plugin for MySQL, made by MySQL and packaged with MySQL
- GR is an implementation of Replicated Database State Machine theory
- GR uses a Paxos based protocol

MySQL Group Replication

but what is it ?!?

- GR is a plugin for MySQL, made by MySQL and packaged with MySQL
- GR is an implementation of Replicated Database State Machine theory
- GR uses a Paxos based protocol
- GR allows to write on all Group Members (cluster nodes) simultaneously while retaining consistency

MySQL Group Replication

but what is it ?!?

- GR is a plugin for MySQL, made by MySQL and packaged with MySQL
- GR is an implementation of Replicated Database State Machine theory
- GR uses a Paxos based protocol
- GR allows to write on all Group Members (cluster nodes) simultaneously while retaining consistency
- GR implements conflict detection and resolution

MySQL Group Replication

but what is it ?!?

- GR is a plugin for MySQL, made by MySQL and packaged with MySQL
- GR is an implementation of Replicated Database State Machine theory
- GR uses a Paxos based protocol
- GR allows to write on all Group Members (cluster nodes) simultaneously while retaining consistency
- GR implements conflict detection and resolution
- GR allows automatic distributed recovery

MySQL Group Replication

but what is it ?!?

- GR is a plugin for MySQL, made by MySQL and packaged with MySQL
- GR is an implementation of Replicated Database State Machine theory
- GR uses a Paxos based protocol
- GR allows to write on all Group Members (cluster nodes) simultaneously while retaining consistency
- GR implements conflict detection and resolution
- GR allows automatic distributed recovery
- Supported on **all MySQL platforms** !!
  - Linux, Windows, Solaris, OSX, FreeBSD

And for users ?

And for users ?

- no longer necessary to handle server fail-over manually or with a complicated script

And for users ?

- no longer necessary to handle server fail-over manually or with a complicated script
- GR provides fault tolerance

And for users ?

- no longer necessary to handle server fail-over manually or with a complicated script
- GR provides fault tolerance
- GR enables update-everywhere setups

And for users ?

- no longer necessary to handle server fail-over manually or with a complicated script
- GR provides fault tolerance
- GR enables update-everywhere setups
- GR handles crashes, failures, re-connetcs automatically

And for users ?

- no longer necessary to handle server fail-over manually or with a complicated script
- GR provides fault tolerance
- GR enables update-everywhere setups
- GR handles crashes, failures, re-connetcs automatically
- **Allows an easy setup of a MySQL service high available !**

OK, but how does it work ?

OK, but how does it work ?

it's just ...

OK, but how does it work ?

it's just ...

OK, but how does it work ?

it's just ...

... no, in fact the writesets replication is **synchronous** and then certification and apply of the changes are local to each nodes and asynchronous.

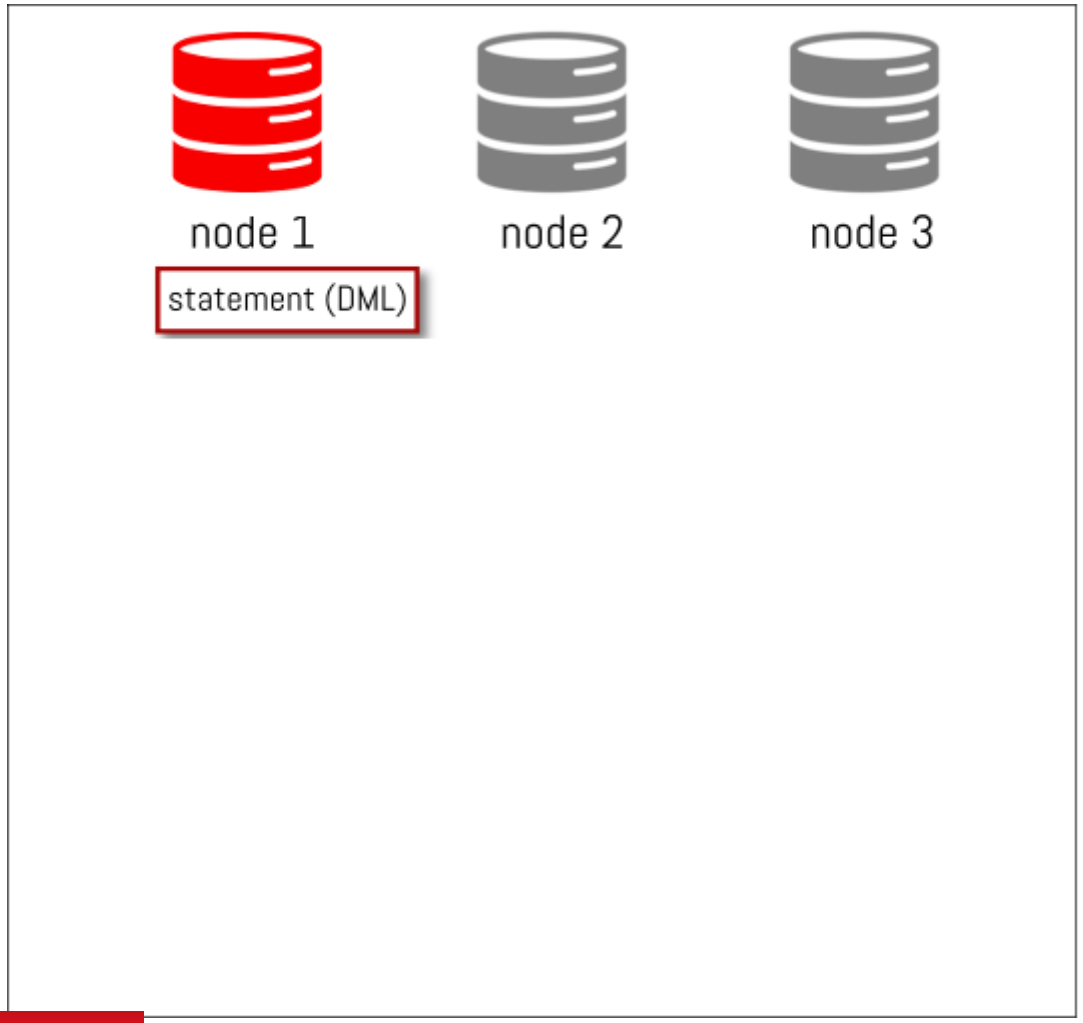
OK, but how does it work ?

it's just ...

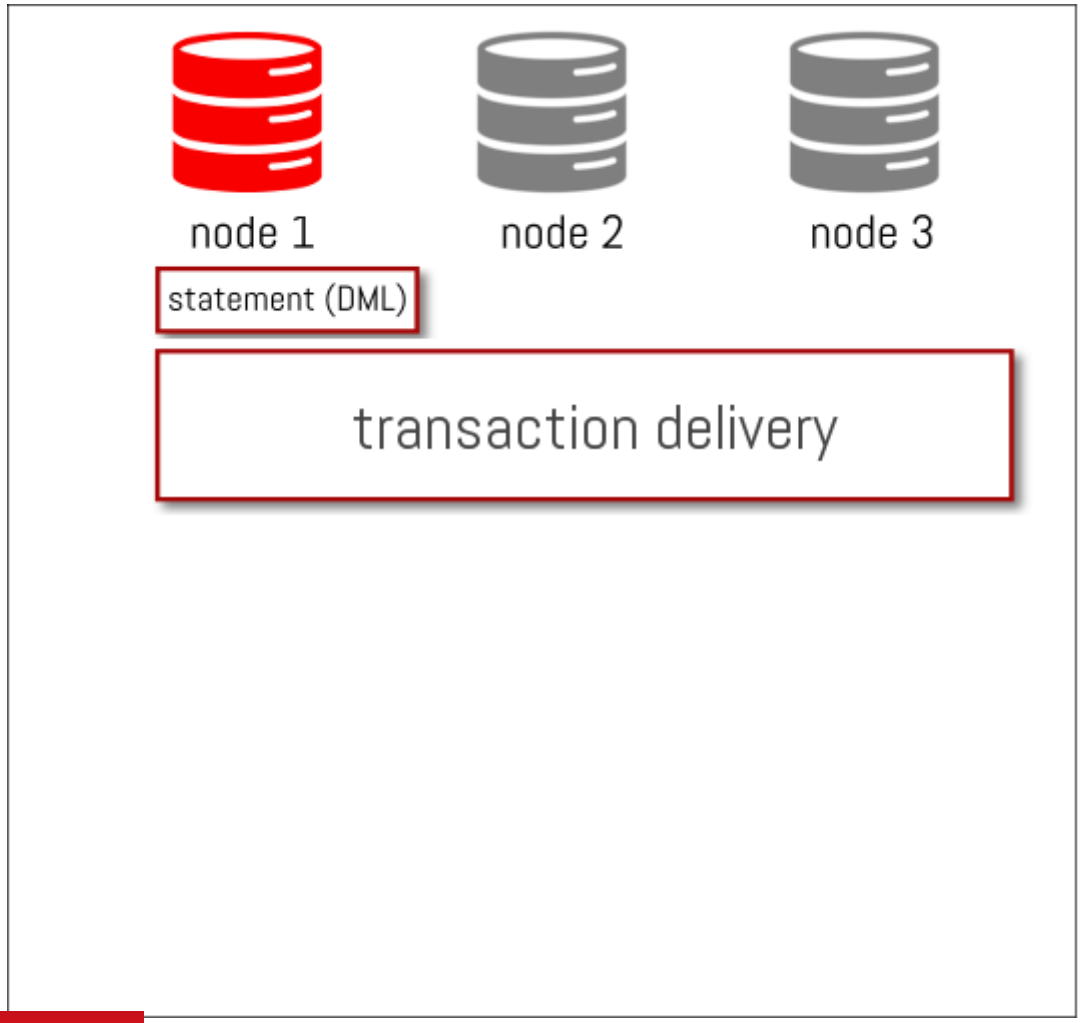
... no, in fact the writesets replication is **synchronous** and then certification and apply of the changes are local to each nodes and asynchronous.

not that easy to understand... right ? As a picture is worth a 1000 words, let's illustrate this...

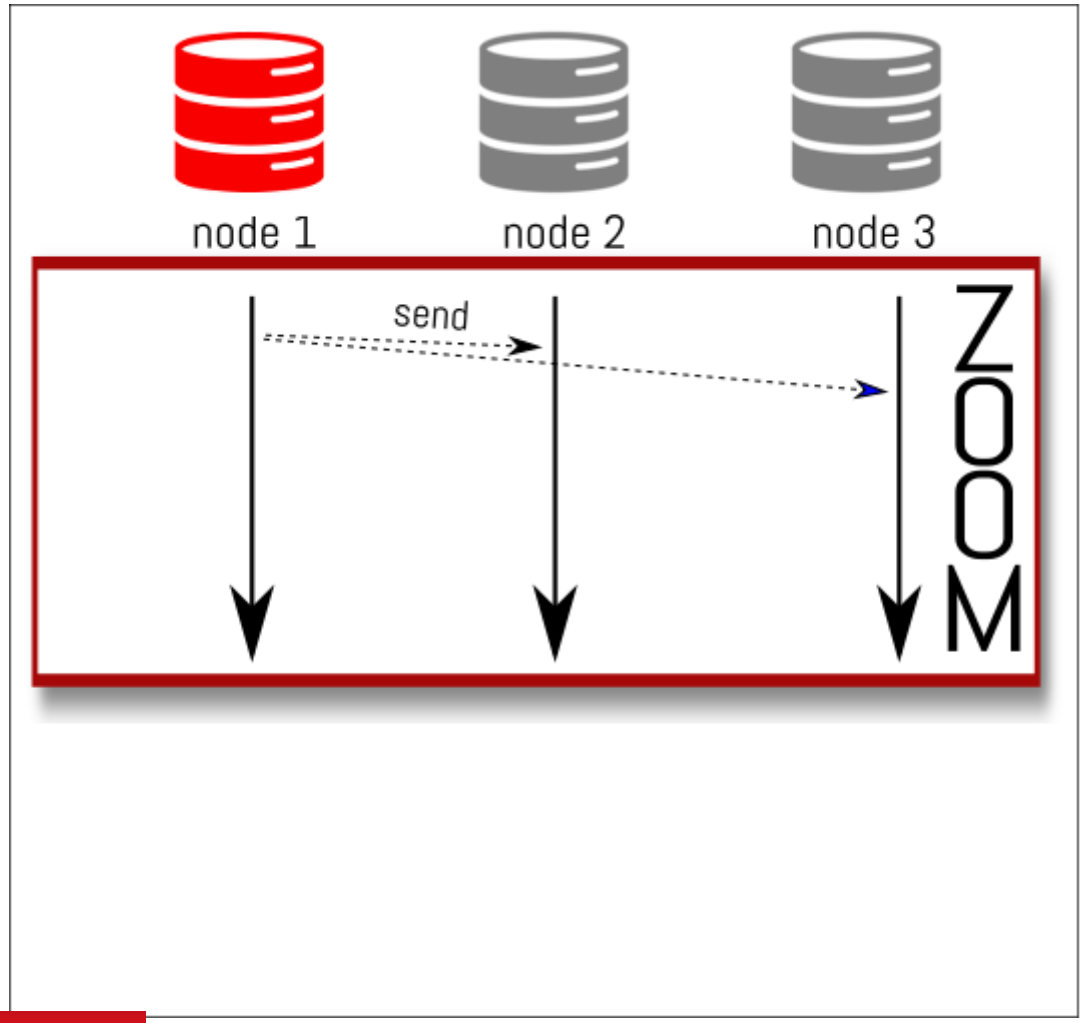
MySQL Group Replication



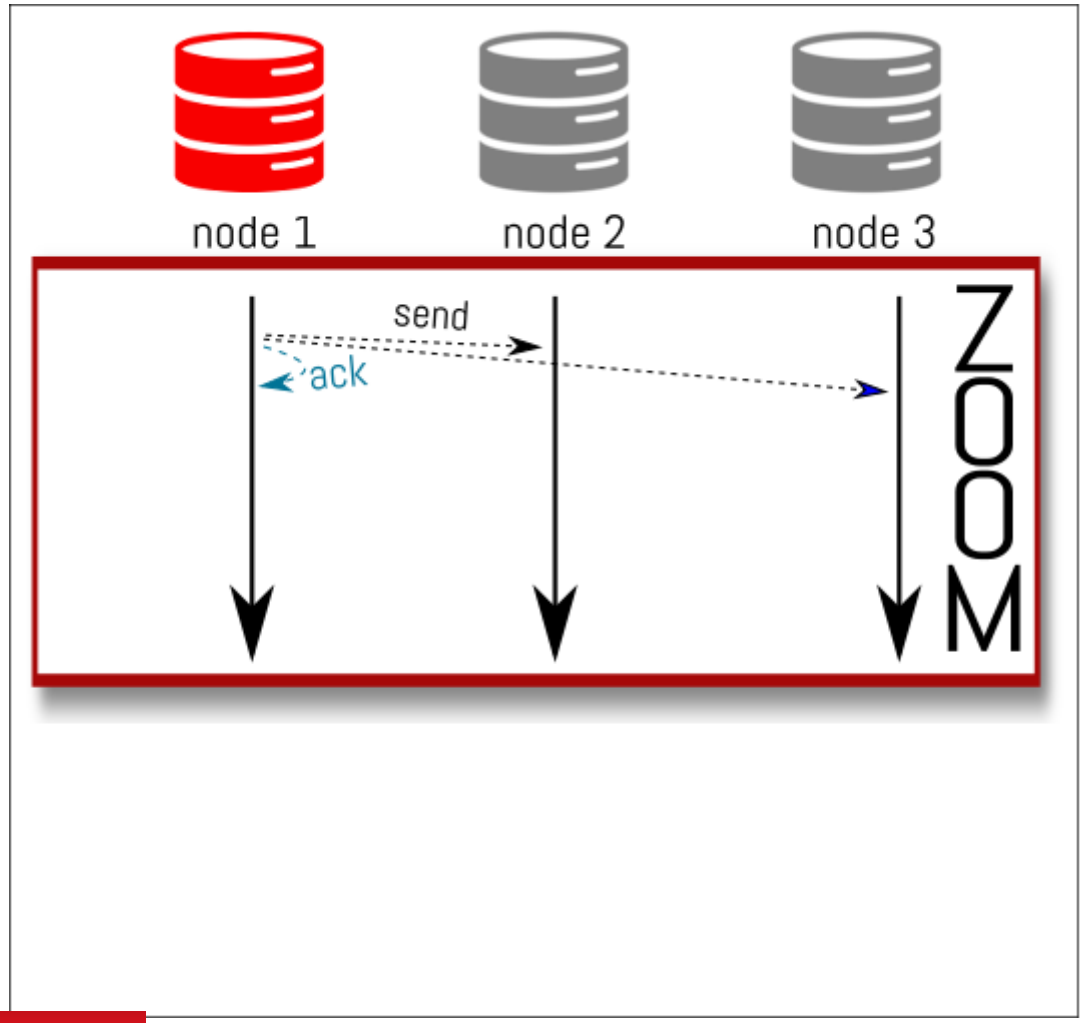
MySQL Group Replication



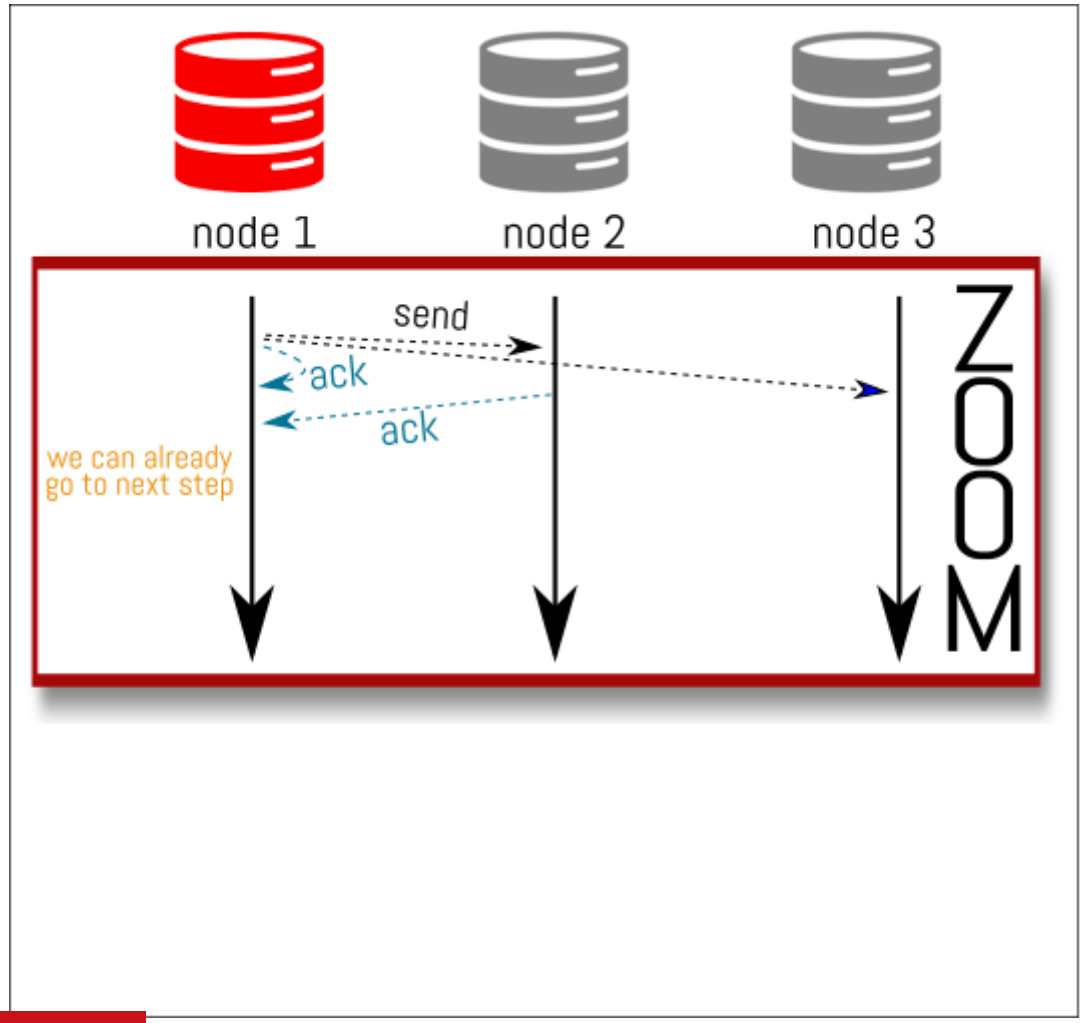
MySQL Group Replication



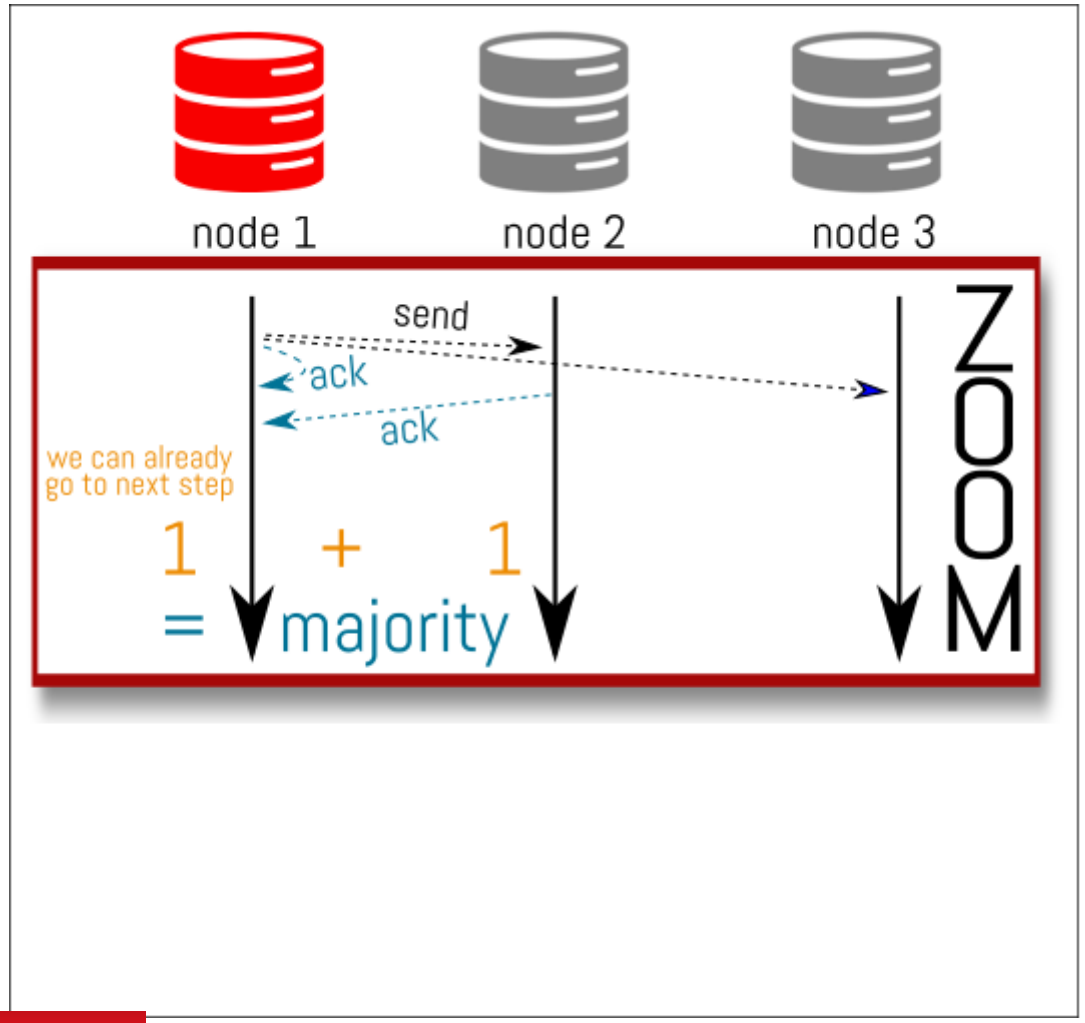
MySQL Group Replication



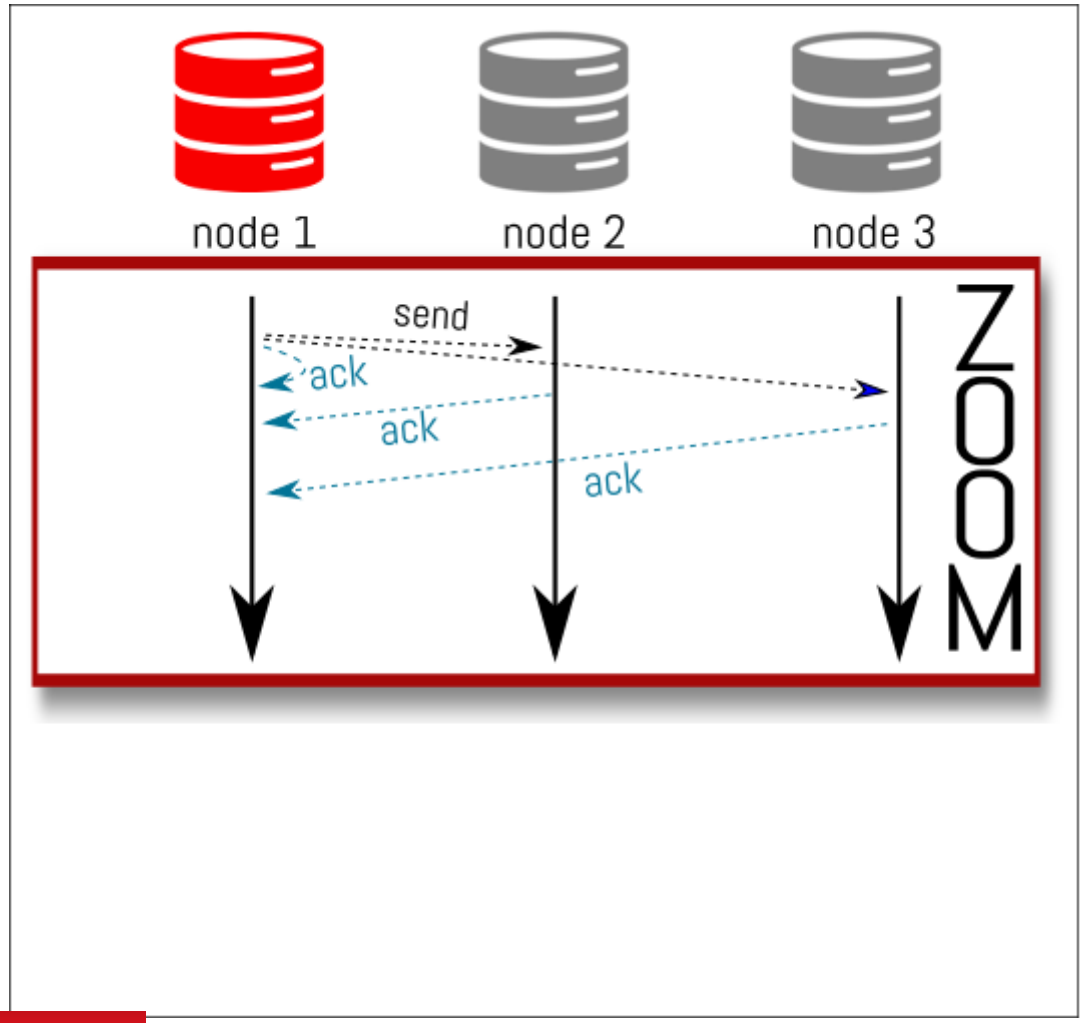
MySQL Group Replication



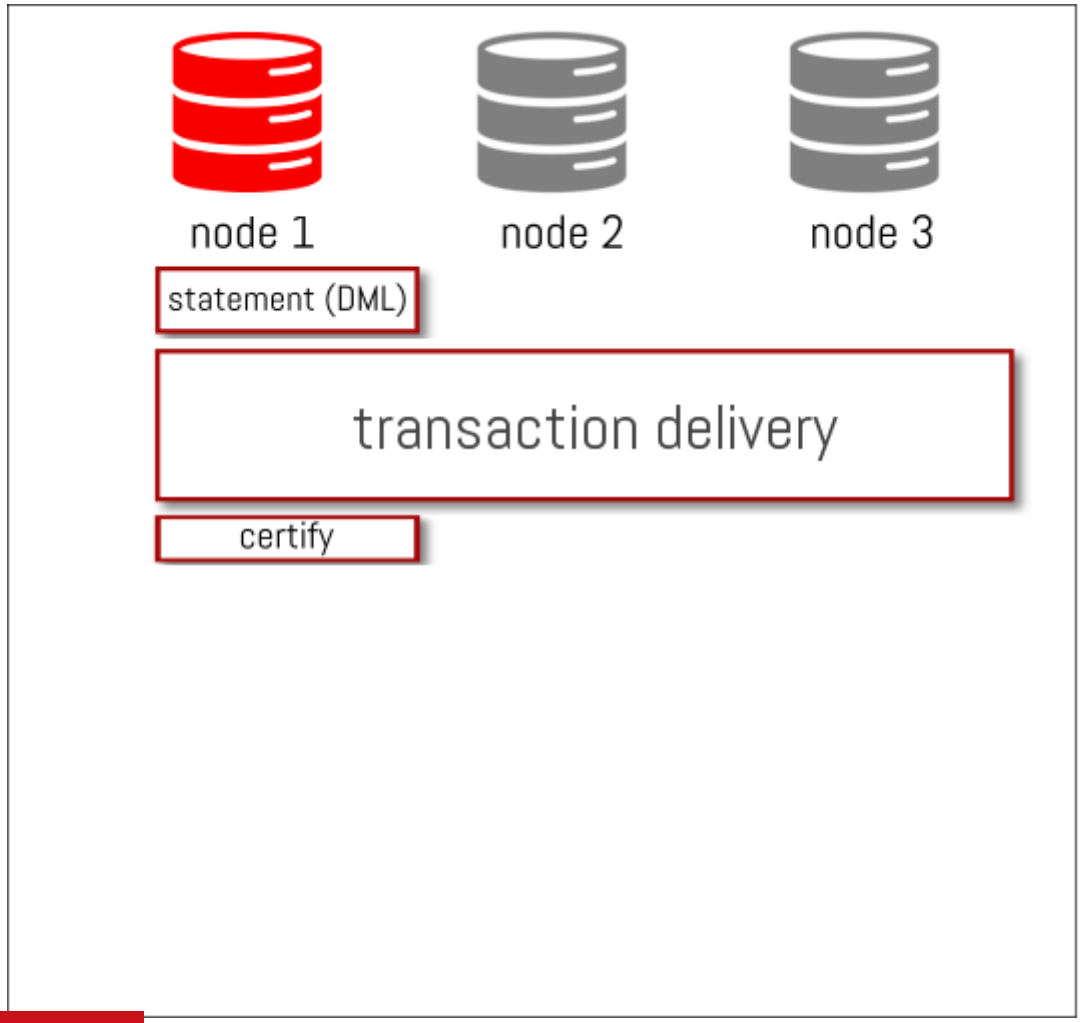
MySQL Group Replication



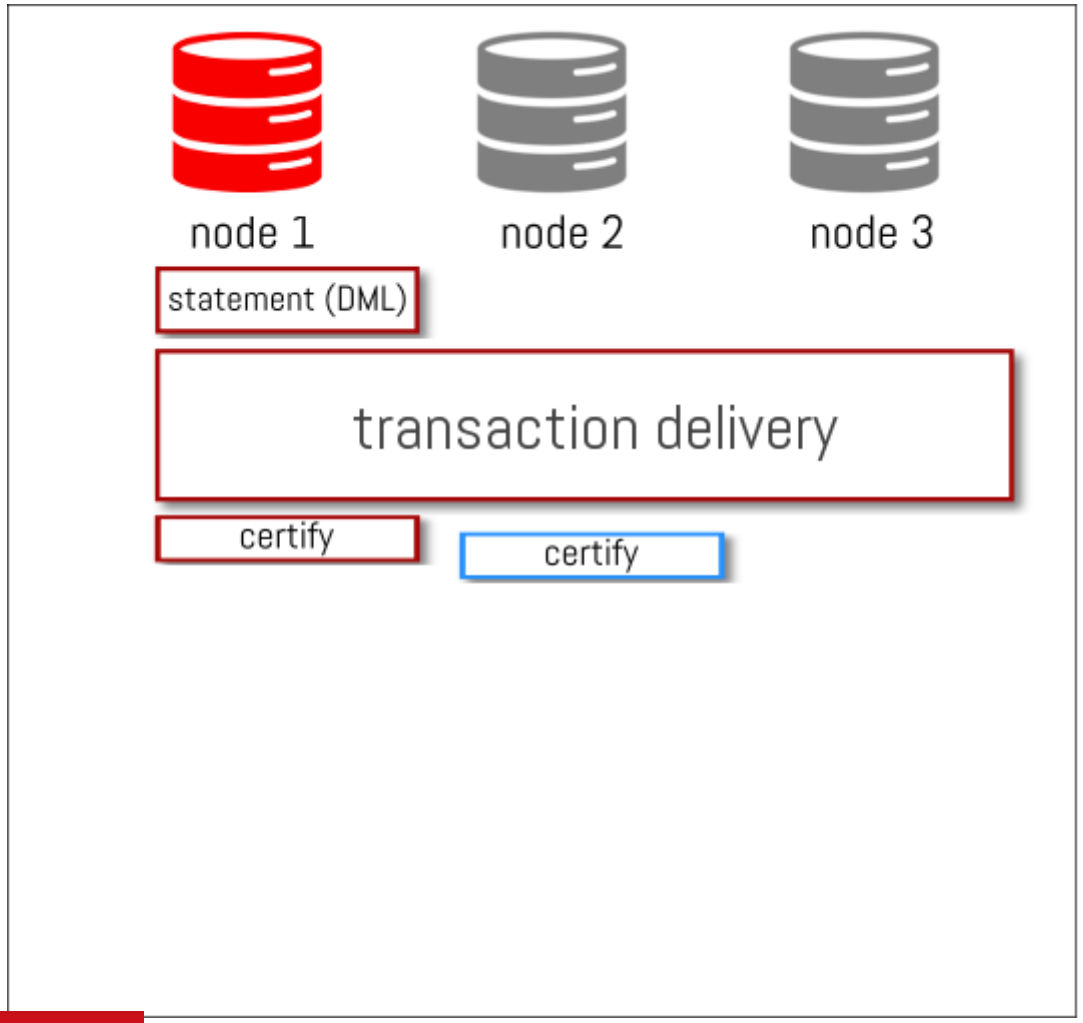
MySQL Group Replication



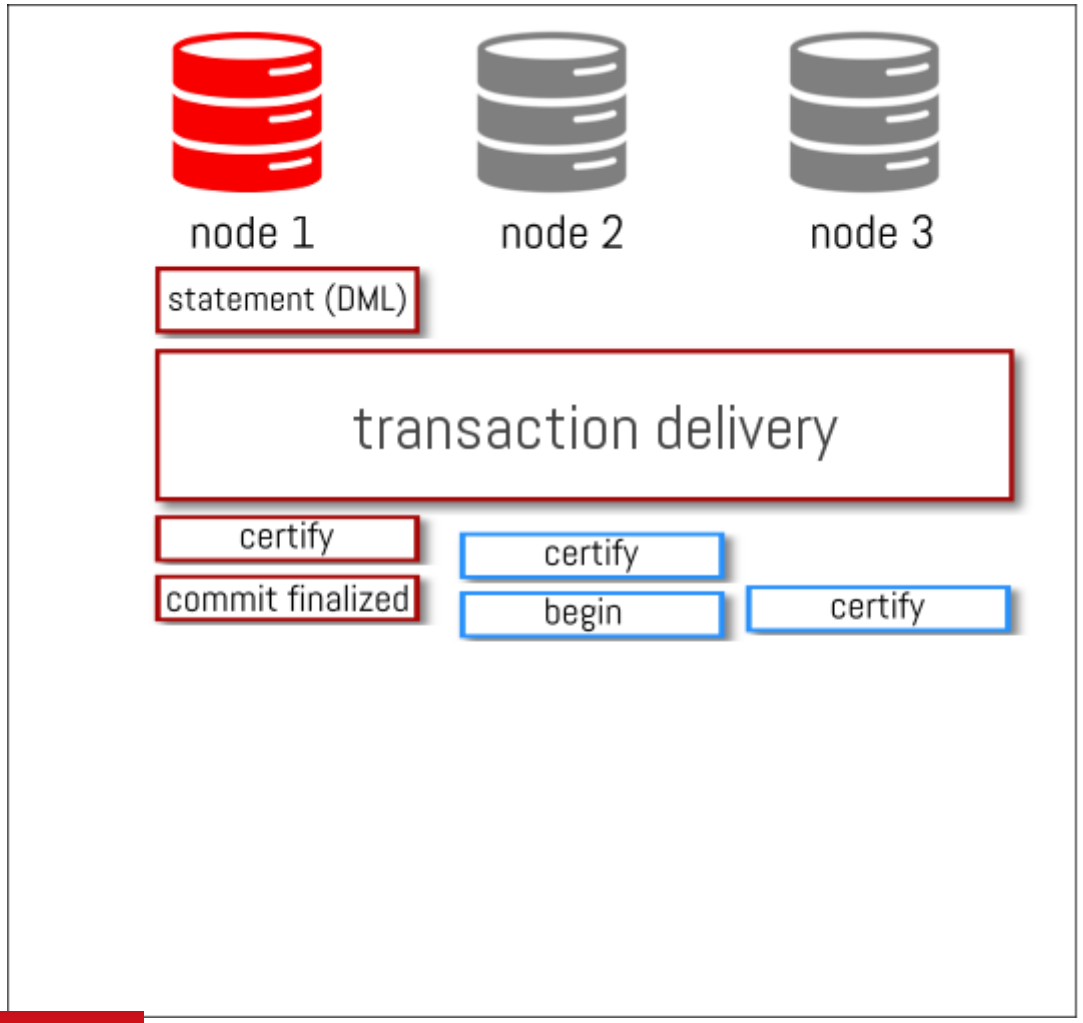
MySQL Group Replication



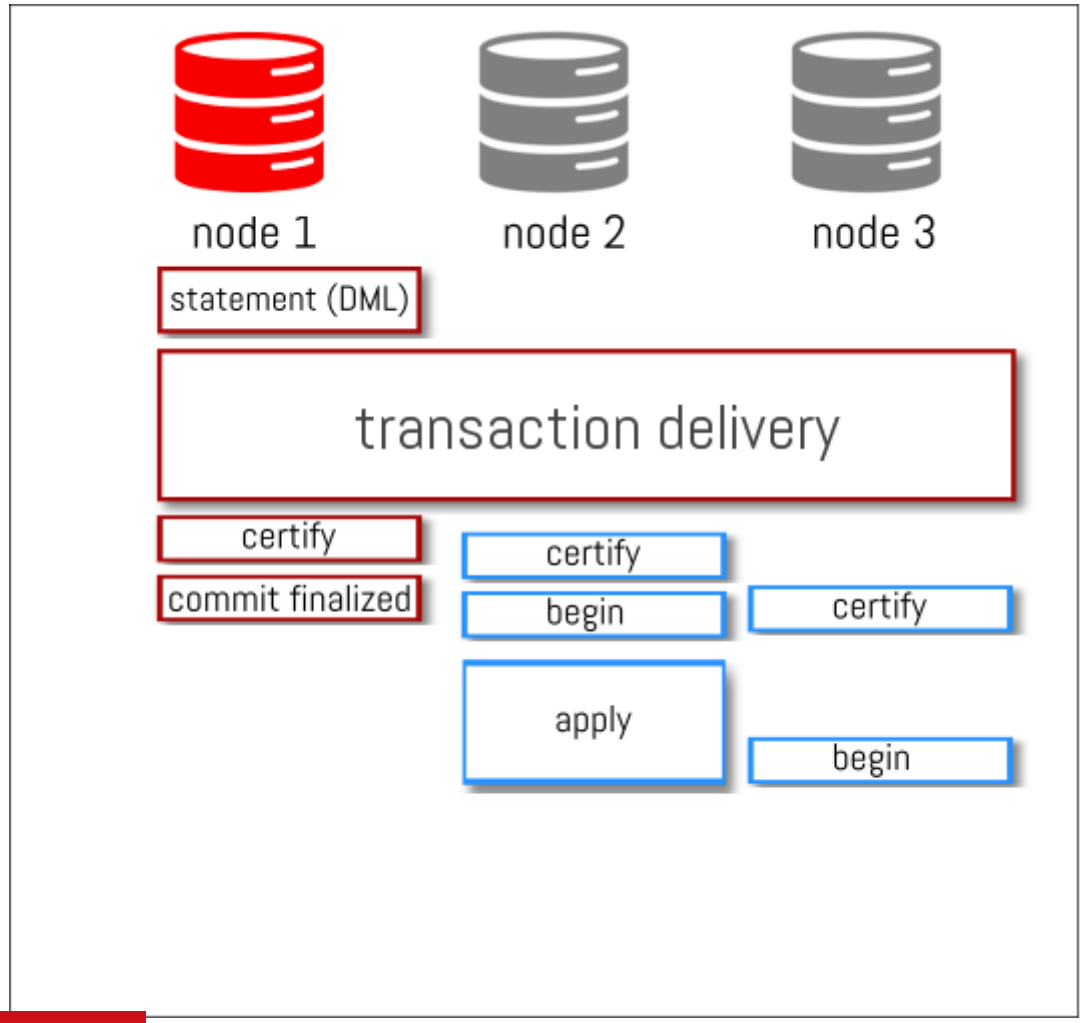
MySQL Group Replication



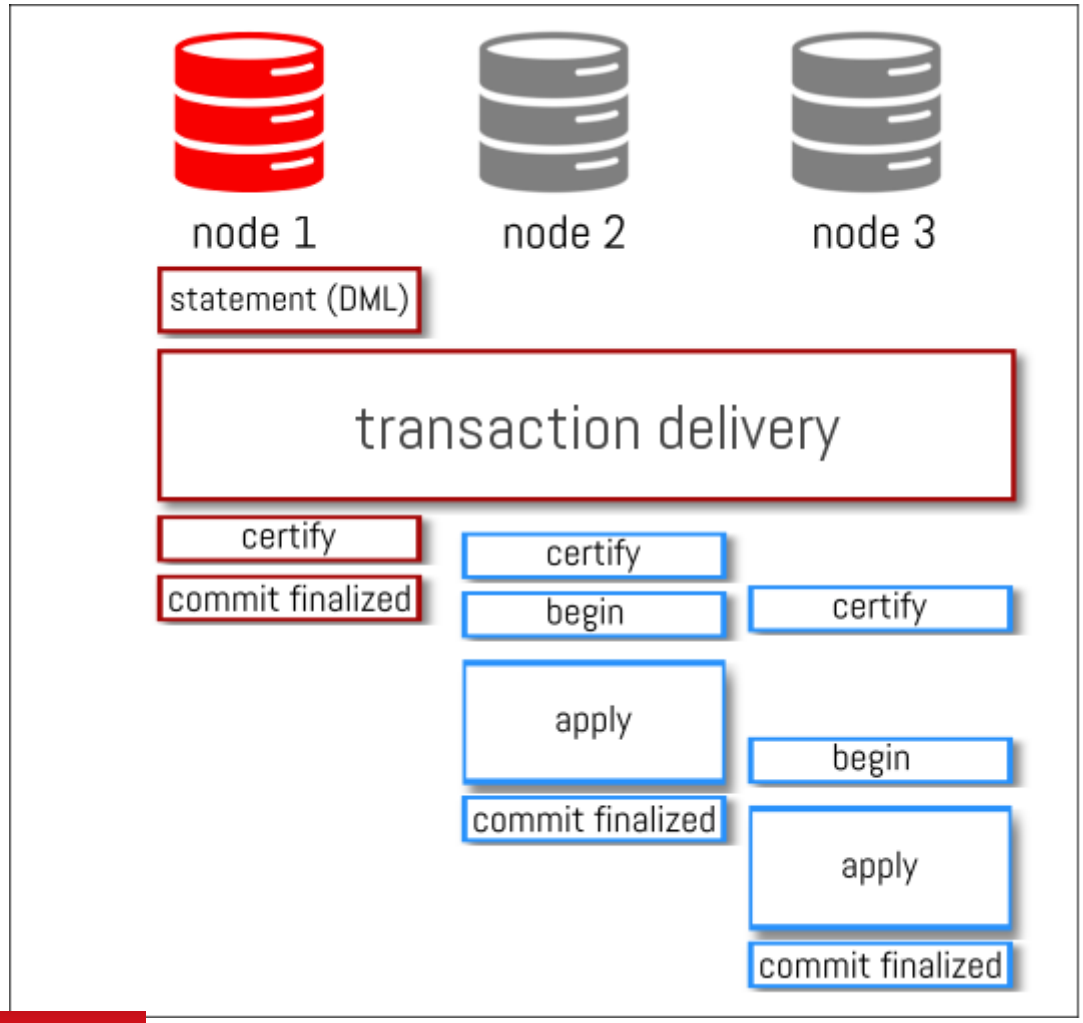
MySQL Group Replication



MySQL Group Replication



MySQL Group Replication



Certification

Certification is the process that only needs to answer the following unique question:

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*
  - based on unapplied earlier transactions

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*
  - based on unapplied earlier transactions
  - such conflicts must come for other members/nodes

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*
  - based on unapplied earlier transactions
  - such conflicts must come for other members/nodes
- happens on every member/node

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*
  - based on unapplied earlier transactions
  - such conflicts must come for other members/nodes
- happens on every member/node
- should be deterministic on every node

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*
  - based on unapplied earlier transactions
  - such conflicts must come for other members/nodes
- happens on every member/node
- should be deterministic on every node
- results are not reported to the group

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*
  - based on unapplied earlier transactions
  - such conflicts must come for other members/nodes
- happens on every member/node
- should be deterministic on every node
- results are not reported to the group
  - pass: enter in the apply queue

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*
  - based on unapplied earlier transactions
  - such conflicts must come for other members/nodes
- happens on every member/node
- should be deterministic on every node
- results are not reported to the group
  - pass: enter in the apply queue
  - fail: drop the transaction

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*
  - based on unapplied earlier transactions
  - such conflicts must come for other members/nodes
- happens on every member/node
- should be deterministic on every node
- results are not reported to the group
  - pass: enter in the apply queue
  - fail: drop the transaction
- serialized by the total order in GCS/XCOM + GTID

Certification

Certification is the process that only needs to answer the following unique question:

- *can the write (transaction) be applied ?*
  - based on unapplied earlier transactions
  - such conflicts must come for other members/nodes
- happens on every member/node
- should be deterministic on every node
- results are not reported to the group
  - pass: enter in the apply queue
  - fail: drop the transaction
- serialized by the total order in GCS/XCOM + GTID
- cost is based on trx size (# rows & # keys)

GTID

- GTIDs are the same as those used by asynchronous replication.

```
mysql>  SELECT * FROM performance_schema.replication_connection_status\G
*************************** 1. row ***************************
             CHANNEL_NAME: group_replication_applier
               GROUP_NAME: afb80f36-2bff-11e6-84e0-0800277dd3bf
            SOURCE_UUID: afb80f36-2bff-11e6-84e0-0800277dd3bf
              THREAD_ID: NULL
          SERVICE_STATE: ON
COUNT_RECEIVED_HEARTBEATS: 0
 LAST_HEARTBEAT_TIMESTAMP: 0000-00-00 00:00:00
  RECEIVED_TRANSACTION_SET: afb80f36-2bff-11e6-84e0-0800277dd3bf:1-57,
f037578b-46b1-11e6-8005-08002774c31b:1-48937
        LAST_ERROR_NUMBER: 0
       LAST_ERROR_MESSAGE:
     LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00
```

- but transactions use the Group´s GTID

```
mysql> show master status\G
*************************** 1. row ***************************
             File: mysql4-bin.000001
         Position: 1501
     Binlog_Do_DB:
 Binlog_Ignore_DB:
 Executed_Gtid_Set: afb80f36-2bff-11e6-84e0-0800277dd3bf:1-57,
f037578b-46b1-11e6-8005-08002774c31b:1-48937
```

Requirements

- exclusively works with InnoDB tables only

Requirements

- exclusively works with InnoDB tables only
- every tables must have a PK defined

Requirements

- exclusively works with InnoDB tables only
- every tables must have a PK defined
- only IPV4 is supported

Requirements

- exclusively works with InnoDB tables only
- every tables must have a PK defined
- only IPV4 is supported
- a good network with low latency is important

Requirements

- exclusively works with InnoDB tables only
- every tables must have a PK defined
- only IPV4 is supported
- a good network with low latency is important
- maximum of 9 members per group

Requirements

- exclusively works with InnoDB tables only
- every tables must have a PK defined
- only IPV4 is supported
- a good network with low latency is important
- maximum of 9 members per group
- `log-bin` must be enabled and only **ROW** format is supported

Requirements (2)

- enable GTIDs

Requirements (2)

- enable GTIDs
- replication meta-data must be stored in system tables

```
--master-info-repository=TABLE
--relay-log-info-repository=TABLE
```

Requirements (2)

- enable GTIDs
- replication meta-data must be stored in system tables

```
--master-info-repository=TABLE
--relay-log-info-repository=TABLE
```

- writesets extraction must be enabled

```
--transaction-write-set-extraction=XXHASH64
```

Requirements (2)

- enable GTIDs
- replication meta-data must be stored in system tables

```
--master-info-repository=TABLE
--relay-log-info-repository=TABLE
```

- writesets extraction must be enabled

```
--transaction-write-set-extraction=XXHASH64
```

- `log-slave-updates` must also **be enabled**

Limitations

- binlog checksum is not supported *(but we have checksum in the replication channel)*

```
--binlog-checksum=NONE
```

Limitations

- binlog checksum is not supported *(but we have checksum in the replication channel)*

```
--binlog-checksum=NONE
```

- savepoints **were** not supported before 5.7.19 & 8.0.1

Limitations

- binlog checksum is not supported *(but we have checksum in the replication channel)*

```
--binlog-checksum=NONE
```

- savepoints **were** not supported before 5.7.19 & 8.0.1
- *SERIALIZABLE* is not supported as transaction isolation level

Limitations

- binlog checksum is not supported *(but we have checksum in the replication channel)*

```
--binlog-checksum=NONE
```

- savepoints **were** not supported before 5.7.19 & 8.0.1
- *SERIALIZABLE* is not supported as transaction isolation level
- http://lefred.be/content/mysql-group-replication-limitations-savepoints/
- http://lefred.be/content/mysql-group-replication-and-table-design/

Default = Single Primary Mode

By default, MySQL InnoDB Cluster runs in `Single Primary Mode`.

Default = Single Primary Mode

By default, MySQL InnoDB Cluster runs in `Single Primary Mode`.

```
mysql> show global variables like 'group_replication_single_primary_mode';
+----------------------------------------+-------+
| Variable_name                          | Value |
+----------------------------------------+-------+
| group_replication_single_primary_mode | ON    |
+----------------------------------------+-------+
```

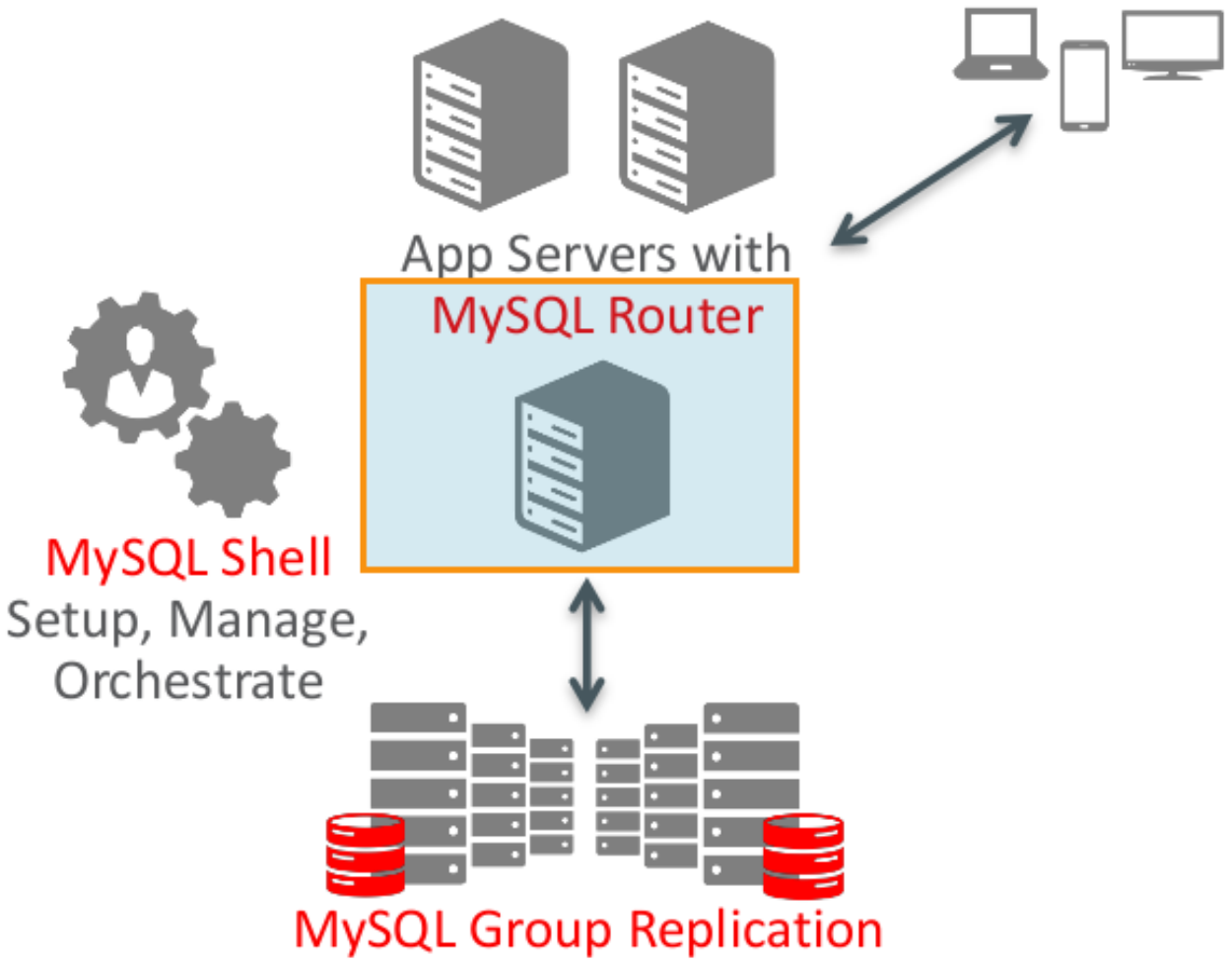
Default = Single Primary Mode

By default, MySQL InnoDB Cluster runs in `Single Primary Mode`.

```
mysql> show global variables like 'group_replication_single_primary_mode';
+----------------------------------------+-------+
| Variable_name                          | Value |
+----------------------------------------+-------+
| group_replication_single_primary_mode | ON    |
+----------------------------------------+-------+
```

In Single Primary Mode, a single member acts as the writable master (`PRIMARY`) and the rest of the members act as hot-standbys (`SECONDARY`).

The group itself coordinates and configures itself automatically to determine which member will act as the `PRIMARY`, through a leader election mechanism.



MySQL Router (GA!)

MySQL Router is lightweight middleware that provides transparent routing between your application and backend MySQL Servers. It can be used for a wide variety of use cases, such as providing high availability and scalability by effectively routing database traffic to appropriate backend MySQL Servers.

MySQL Router (GA!)

MySQL Router is lightweight middleware that provides transparent routing between your application and backend MySQL Servers. It can be used for a wide variety of use cases, such as providing high availability and scalability by effectively routing database traffic to appropriate backend MySQL Servers.

MySQL Router doesn´t require any specific configuration. It configures itself automatically (bootstrap) using MySQL InnoDB Cluster´s metadata.

MySQL Router (GA!)

MySQL Router is lightweight middleware that provides transparent routing between your application and backend MySQL Servers. It can be used for a wide variety of use cases, such as providing high availability and scalability by effectively routing database traffic to appropriate backend MySQL Servers.

MySQL Router doesn´t require any specific configuration. It configures itself automatically (bootstrap) using MySQL InnoDB Cluster´s metadata.

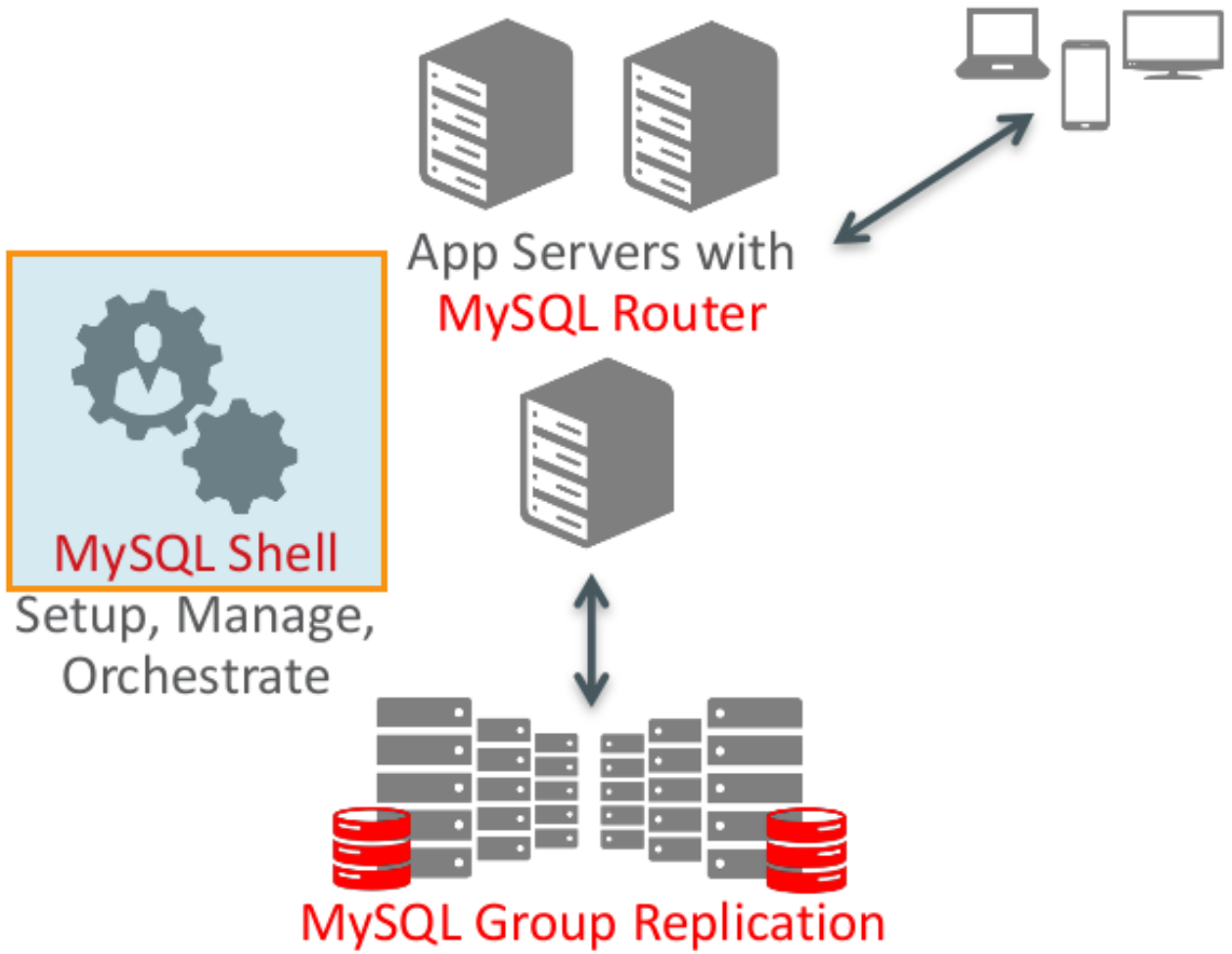
The MySQL Router development will be focusing on sharding.

ProxySQL

If you need some specific features that are not yet available in MySQL **Router**, like transparent R/W splitting, then you can use your software of choice.

We are also collaborating with ProxySQL.

ProxySQL has native support for Group Replication which makes it a good choice for advanced users.



MySQL Shell (GA!)

The MySQL Shell is an interactive Javascript, Python, or SQL interface supporting development and administration for the MySQL Server and is a component of the MySQL Server. You can use the MySQL Shell to perform data queries and updates as well as various administration operations.

MySQL Shell (2)

The MySQL Shell provides:

MySQL Shell (2)

The MySQL Shell provides:

- Both Interactive and Batch operations

MySQL Shell (2)

The MySQL Shell provides:

- Both Interactive and Batch operations
- Document and Relational Models

MySQL Shell (2)

The MySQL Shell provides:

- Both Interactive and Batch operations
- Document and Relational Models
- CRUD Document and Relational APIs via scripting

MySQL Shell (2)

The MySQL Shell provides:

- Both Interactive and Batch operations
- Document and Relational Models
- CRUD Document and Relational APIs via scripting
- Traditional Table, JSON, Tab Separated output results formats

MySQL Shell (2)

The MySQL Shell provides:

- Both Interactive and Batch operations
- Document and Relational Models
- CRUD Document and Relational APIs via scripting
- Traditional Table, JSON, Tab Separated output results formats
- MySQL Standard and X Protocols

MySQL Shell (2)

The MySQL Shell provides:

- Both Interactive and Batch operations
- Document and Relational Models
- CRUD Document and Relational APIs via scripting
- Traditional Table, JSON, Tab Separated output results formats
- MySQL Standard and X Protocols
- and more...

MySQL Shell (3)

PREVIEW: the new Shell´s prompt is nice and clear !

```
 MySQL   JS  dba.checkInstanceConfiguration(i2)
Please provide the password for 'root@mysql2:3306':
Validating instance...

The instance 'mysql2:3306' is valid for Cluster usage
{
    "status": "ok"
}

 MySQL   JS  dba.checkInstanceConfiguration(i3)
Please provide the password for 'root@mysql3:3306':
```

don´t forget, this is a HA solution !

Deploying a MySQL InnoDB Cluster using MySQL Shell´s adminAPI:

```
mysql-js> var i1 = 'root@instance01:3306';
mysql-js> var i2 = 'root@instance02:3306';
mysql-js> var i3 = 'root@instance03:3306';

mysql-js> dba.checkInstanceConfiguration(i1);
mysql-js> dba.checkInstanceConfiguration(i2);
mysql-js> dba.checkInstanceConfiguration(i3);

mysql-js> shell.connect(i1);
mysql-js> var cluster = dba.createCluster('GhentCluster');

mysql-js> cluster.checkInstanceState(i2);
mysql-js> cluster.addInstance(i2);

mysql-js> cluster.checkInstanceState(i3);
mysql-js> cluster.addInstance(i3);
```

DEMO !

https://youtu.be/y3WywG7Zhks

Thanks to the MySQL Shell we can automate everything ;-)

https://github.com/lefred/puppet-lefred-innodbcluster

Super easy using hiera:

`common.yaml`:

```
innodbcluster::mysql_root_password:  StRongP4ssw0rD!
innodbcluster::mysql_bind_interface: eth1
innodbcluster::cluster_name:         GhentCluster
innodbcluster::grant::user:          fred
innodbcluster::grant::password:      fred
innodbcluster::seed:                 instance01
```

`instance01.yaml` :

```
classes:
    - innodbcluster

innodbcluster::mysql_serverid:    1
```

example (2)

With `hiera-eyaml` you can even encrypt your password:

```
innodbcluster::mysql_root_password: >
    ENC[PKCS7,MIIBeQYJKoZIhvcNAQcDoIIBajCCAWYCAQAxggEhMIIBHQIBADAFMAACAQEw
    DQYJKoZIhvcNAQEBBQAEggEAhqKUTXZ/4L8/aL3XARMfDBEI+s5HPshPg9BI
    ...
    FLfovstrb8zmcbk5yb/KD0lDM8Elas0lrnpk8MxwNfKw+hB299JFp8ldAtUk
    ODIieTA8BgkqhkiG9w0BBwEwHQYJYIZIAWUDBAEqBBBJEeoyzHtW/WGpbiUz
    gcXTgBAnb1gGrBZAfAiv/ztwuZ9z]
innodbcluster::mysql_bind_interface:    eth1
innodbcluster::cluster_name: GhentCluster
innodbcluster::grant::user: root
innodbcluster::grant::password: >
    ENC[PKCS7,MIIBeQYJKoZIhvcNAQcDoIIBajCCAWYCAQAxggEhMIIBHQIBADAFMAACAQEw
    DQYJKoZIhvcNAQEBBQAEggEAhqKUTXZ/4L8/aL3XARMfDBEI+s5HPshPg9BI
    ...
    FLfovstrb8zmcbk5yb/KD0lDM8Elas0lrnpk8MxwNfKw+hB299JFp8ldAtUk
    ODIieTA8BgkqhkiG9w0BBwEwHQYJYIZIAWUDBAEqBBBJEeoyzHtW/WGpbiUz
    gcXTgBAnb1gGrBZAfAiv/ztwuZ9z]
```

DEMO !

https://www.youtube.com/watch?v=skwlmBNE7ts


MySQL
Oracle OpenWorld
October 1–5, 2017
San Francisco
#oow17

Resources

http://lefred.be/content/mysql-innodb-cluster-mysql-shell-starter-guide/

http://lefred.be/content/mysql-innodb-cluster-automated-installation-with-puppet/

http://lefred.be/content/category/mysql/group-replication/

https://www.slideshare.net/lefred.descamps/mysql-devops-webinar

http://mysqlserverteam.com/innodb-cluster-in-opc/

Thank you !

Kenny & Vadim for their tests, bugs and feature requests

Thank you !

Any Questions ?