发挥MySQL 8.0的优势

马楚成
MySQL Principal Solution Engineer, APAC

ivan-cs.ma@oracle.com
20181215

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes 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 upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

2018年4月19日
8.0.11 GA

2018年7月27日
8.0.12 发布

2018年10月22日
8.0.13 发布

The world's most popular open source database

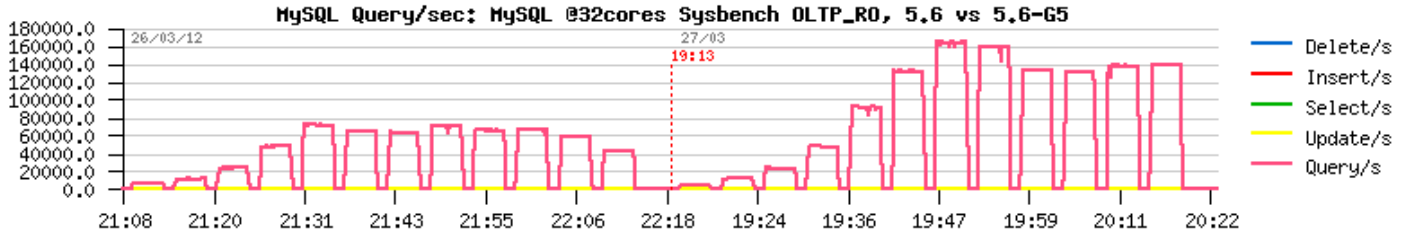
ORACLE

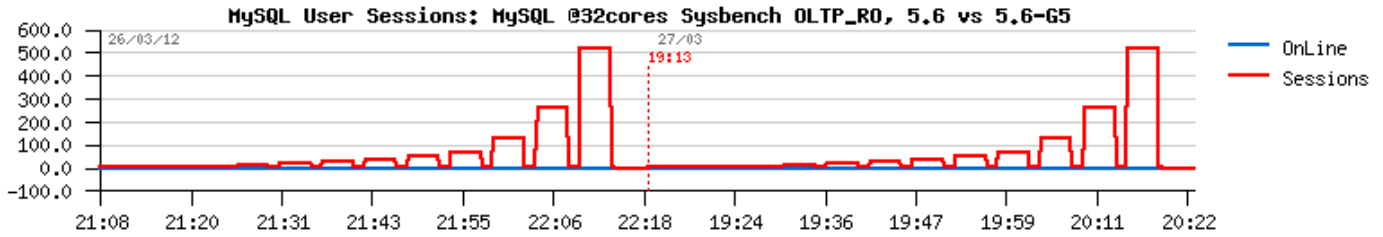
议程

1. MySQL 8.0 新功能
2. 文档存储的优势
3. MySQL 8.0 部署方案
4. Q & A

MySQL 扩展性优化阶段

- MySQL 5.5
  - Delivered “already known” solution (except Buffer Pool[BP] instances and few others)
- MySQL 5.6
  - 基本变化（kernel_mutex拆分，G5补丁，RO交易等...
  - http://dimitrik.free.fr/blog/archives/2012/04/mysql-performance-56labs-is-opening-a-new-era.html



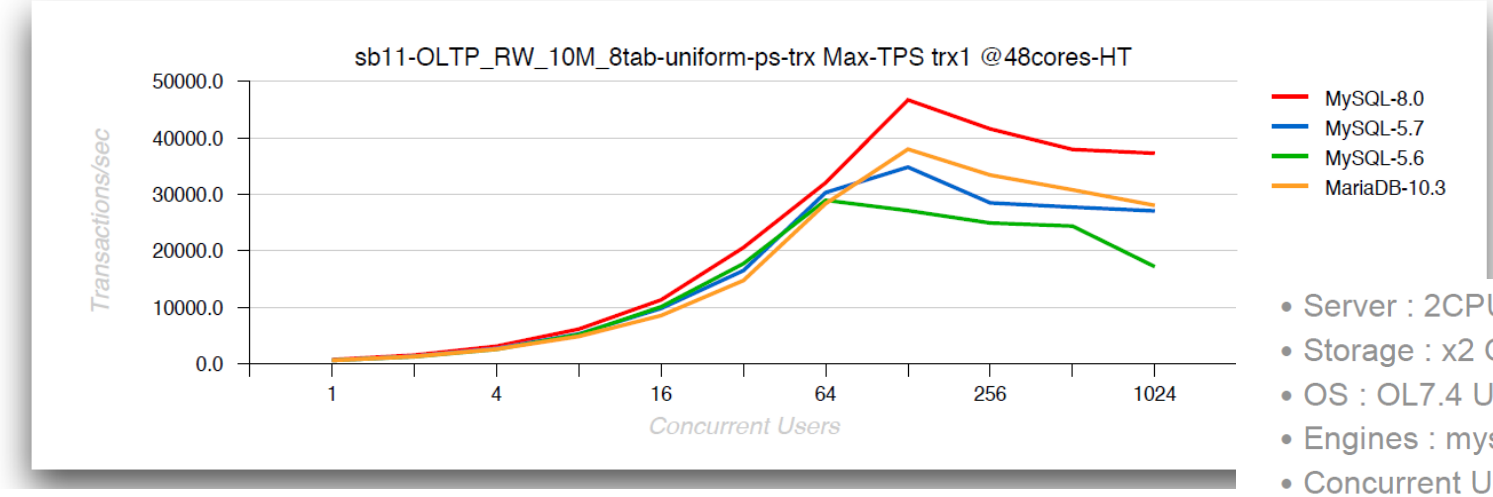


MySQL 扩展性优化阶段

- MySQL 5.5
  - Delivered “already known” solution (except Buffer Pool[BP] instances and few others)
- MySQL 5.6
  - 基本变化（kernel_mutex拆分，G5补丁，RO交易等...
- MySQL 5.7
  - 只读(readonly)性能提升+“服务器”层的锁定争用
- MySQL 8.0
  - 用于InnoDB REDO处理的新设计（RW更快）
  - Resource Group (CPU resource), etc…

OLTP_RW latin1 @ MySQL 8.0 (Apr.2018)

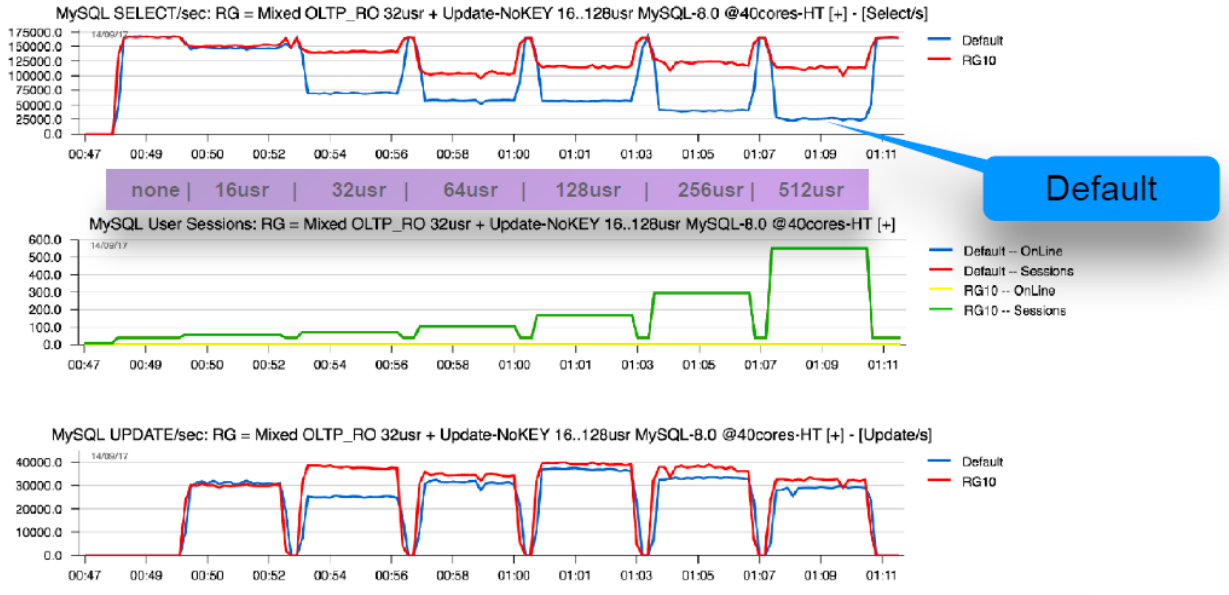
- **45K (!!) TPS** Sysbench OLTP_RW 10Mx8tab, trx_commit=1, 2S
  - **30**% gain vs MySQL 5.7
  - **50**% gain vs MySQL 5.6



- Server : 2CPU Sockets (2S) 48cores-HT Skylake
- Storage : x2 Optane NVMe, MDM-raid0, EXT4
- OS : OL7.4 UEK4
- Engines : mysql5.6, mysql5.7, mysql8.0, mariadb10.3.5
- Concurrent Users : **1**, **2**, **4**, .. **1024**
- charset : **latin1** (!! ;-))
- **trx_commit=1**
- REDO = 32GB
- O_DIRECT
- DBLWR=off, binlog=off, SSL=off, PFS=off, UNDO auto-truncate=off..

MySQL Resource Groups – 更少的资源争用

- Test case :
  - 40 cores – 4S (Broadwell) Server, OL7
  - 32 concurrent users running SELECT
  - UPDATE /*+ RESOURCE GROUP(RG10) */
    …
    - RG10 : CREATE RESOURCE GROUP RG10 type=user vcpu=0-9,40-49 thread_priority=0;



Derived Tables - 提取表

- Subquery in FROM clause

```
SELECT AVG(o_totalprice) FROM
        ( SELECT * FROM orders   ORDER BY o_totalprice DESC LIMIT 100000 ) td;
```

- MySQL 5.6 :单独执行并将结果存储在临时表中
- MySQL 5.7:处理提取表如视图：可以与外部查询块合并

MySQL 8.0 CTE : Readability / Performance

使用CTE获得更好的性能（MySQL 8.0）

- Derive tables cannot be referenced twice BUT CTEs can
- Better performance with materialization
  - Multiple CTE references are only materialized once
  - Dervied tables and views will be materialized once per reference

例：

Using view

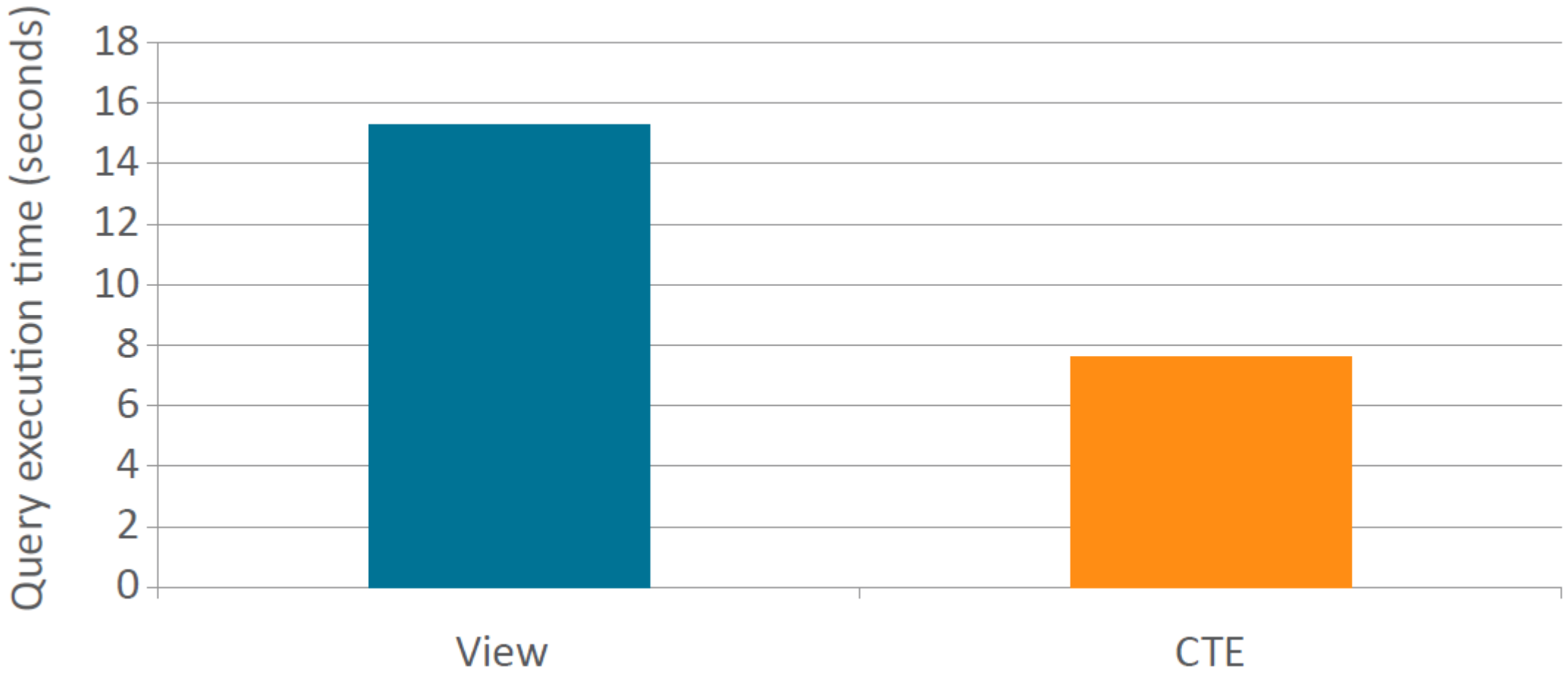
```
CREATE VIEW revenue0 (supplier_no, total_revenue) AS
(
   SELECT l_suppkey,
          SUM(l_extendedprice * (1- l_discount))
   FROM  lineitem
   WHERE l_shipdate >= '1996-07-01'
      AND l_shipdate < DATE_ADD('1996-07-01',
          INTERVAL '90' DAY)
   GROUP BY l_suppkey
);
```

Using CTE

```
WITH revenue0 (supplier_no, total_revenue) AS
(
   SELECT l_suppkey,
          SUM(l_extendedprice *  (1-l_discount))
   FROM  lineitem
   WHERE l_shipdate >= '1996-07-01'
      AND l_shipdate < DATE_ADD('1996-07-01',
          INTERVAL '90' DAY)
   GROUP BY  l_suppkey
)
```

```
SELECT s_suppkey, s_name, s_address, s_phone, total_revenue
FROM supplier, revenue0
WHERE s_suppkey = supplier_no AND total_revenue = (SELECT MAX(total_revenue) FROM revenue0)
ORDER BY s_suppkey;
```

性能



MySQL 复制

- MySQL 5.6
  - 快速传输：多线程从站（按数据库）轻松实现GTID故障切换
- MySQL 5.7
  - 多源复制，多线程从属，无损半同步复制
  - 崩溃安全复制(REPOSITORY=TABLE)
- MySQL InnoDB Cluster
  - 开箱即用的MySQL HA解决方案
- MySQL 8.0

高效的复制 - Applier

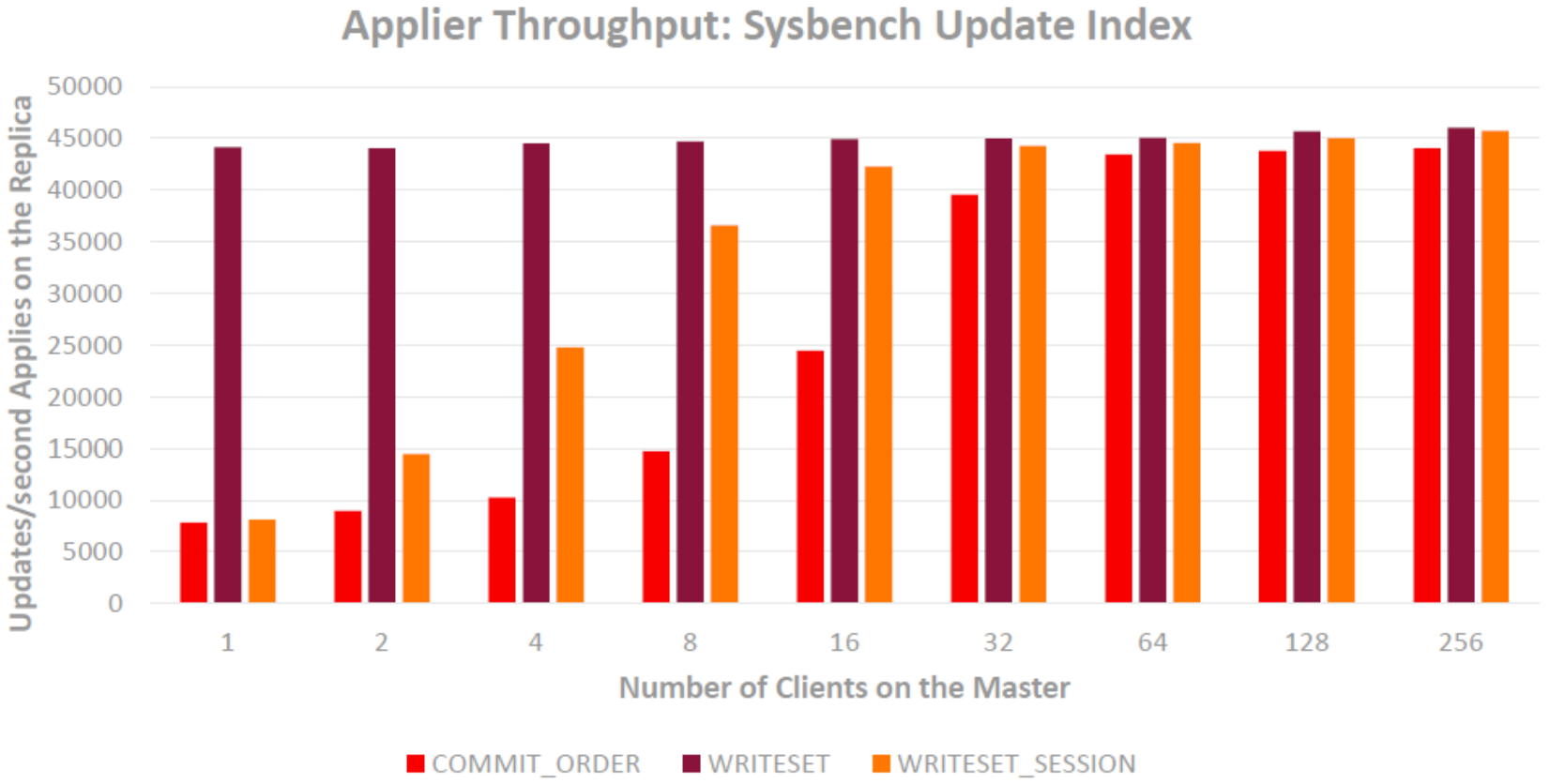
Write set 并发

- WRITESET 依赖关系跟踪允许并行应用单个线程工作负载
- WRITESET_SESSION in addition to writesets tracks sessions dependencies.
- 快速组复制恢复 - 追赶数据滞后

高效的复制 - Applier

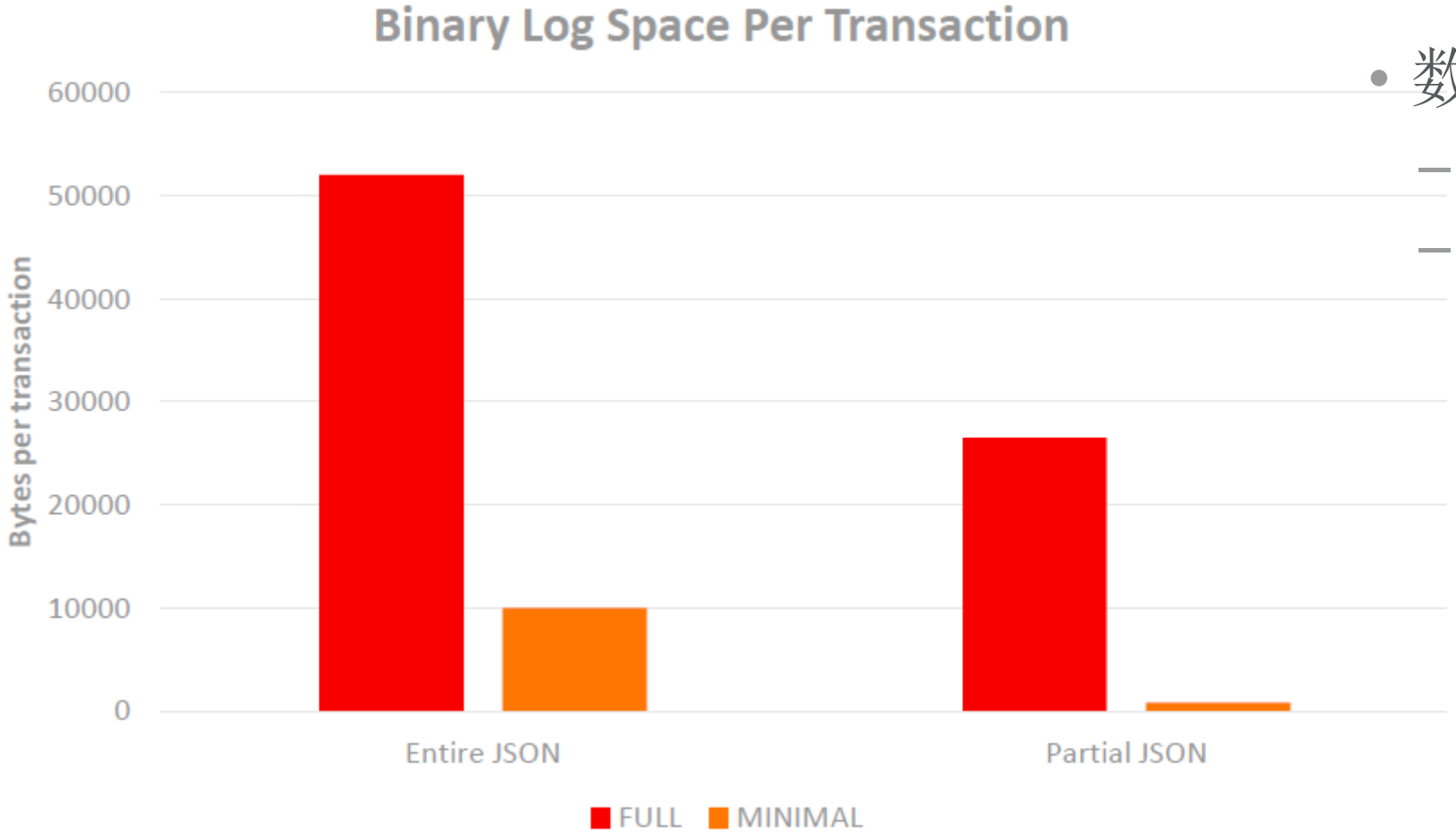
Write set 并行

- WRITESET & WRITESET_SESSION dependency tracking



高效复制JSON文档

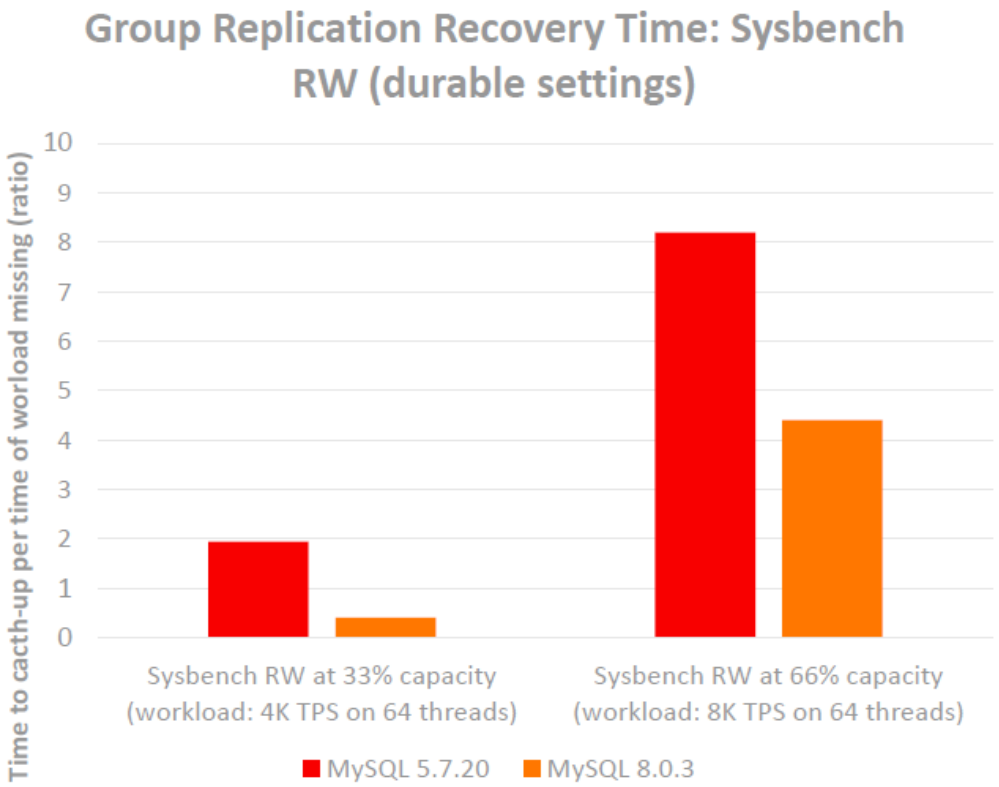
仅复制更改(Partial JSON Updates)

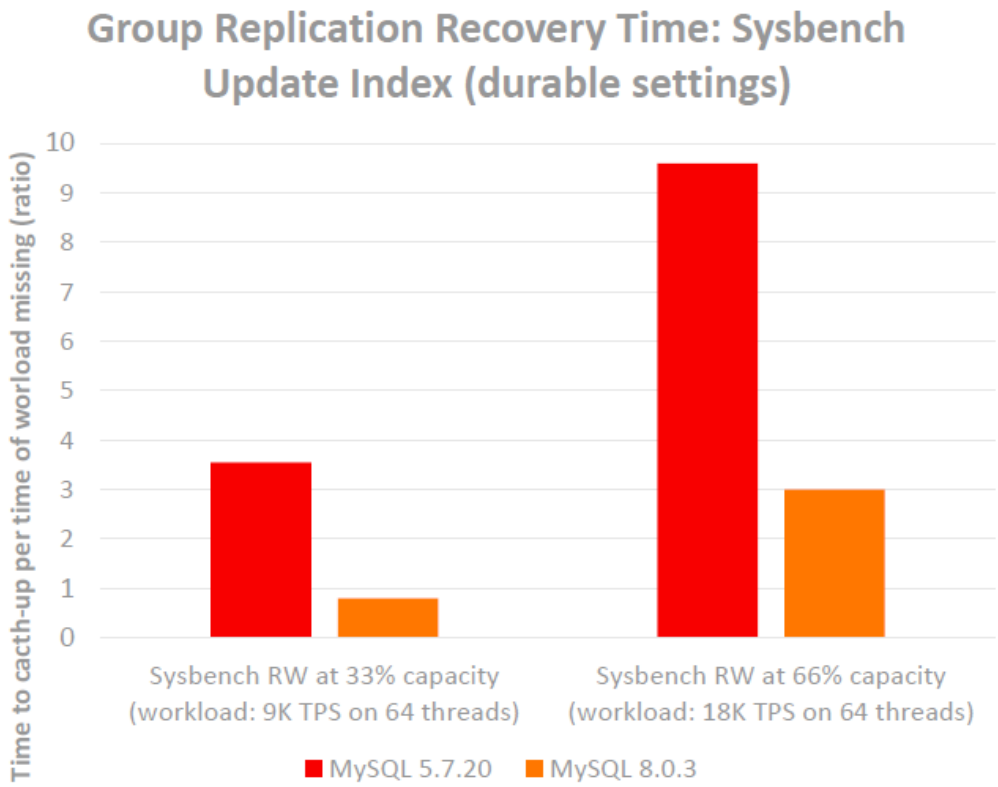


- 数据基准：：
  - 表有10个JSON字段，
  - 每笔交易修改~10%的数据

快速组复制恢复

使用WRITESET快速在线复制副本





All these features **plus…**

- Source code now documented with Doxygen
- New Plugin Infrastructure
- Improved BLOB Storage
- Improved Memcached Interface
- InnoDB Auto Increment Persists
- Parser Refactoring
- Improvements to Temporary Tables
- C++11 and Toolchain Improvements
- GTID_PURGED always settable
- Undo tablespaces now dynamic
- In memory storage engine
- SQL Roles Support
- Encryption for Redo and General Tablespaces
- InnoDB Cats lock scheduler algorithm
- Improved Parallel Replication
- SQL Grouping Function
- Optimizer Trace detailed sort statistics
- Smaller Package Downloads
- Improved usability of cost constant configuration

MySQL作为文档存储

NoSQL的魅力

- 越来越多 用户/公司 开始尝试NoSQL

对 RDBMS的疑惑	后来发现NoSQL的不足，限制和问题
RDBMS扩展性低	缺乏“参照完整性””Referential Integrity”
开发人员不喜欢SQL，等等	缺乏ACID支持

- 结果：
  - 对传统数据库上开发NoSQL接口（例如Uber使用Schemaless）
  - 开发存储引擎，实现NoSQL（例如Facebook使用MyRocks /RocksDB)
  - 使用一些支持JSON的RDBMS

MySQL文档存储【Document Store】

- 使用MySQL文档存储，轻松保存和检索“JSON”数据
- 提供JSON列为数据存储，以新语法 + 新JSON函数 (jsoncol->”$.item” / Generated Columns) 来处理文档，方便使用
- 提供跟MySQL的RDBMS的逻辑一致性支持
- 同时，跨文档/关系表的混合使用
- 灵活性是关键

JSON 与 TABLE 的转换

- TABLE → JSON

```
+------+------+------+
| id   | name | age  |
+------+------+------+
|    1 | John |   34 |
|    2 | Mary |   40 |
|    3 | Mike |   44 |
+------+------+------+
```

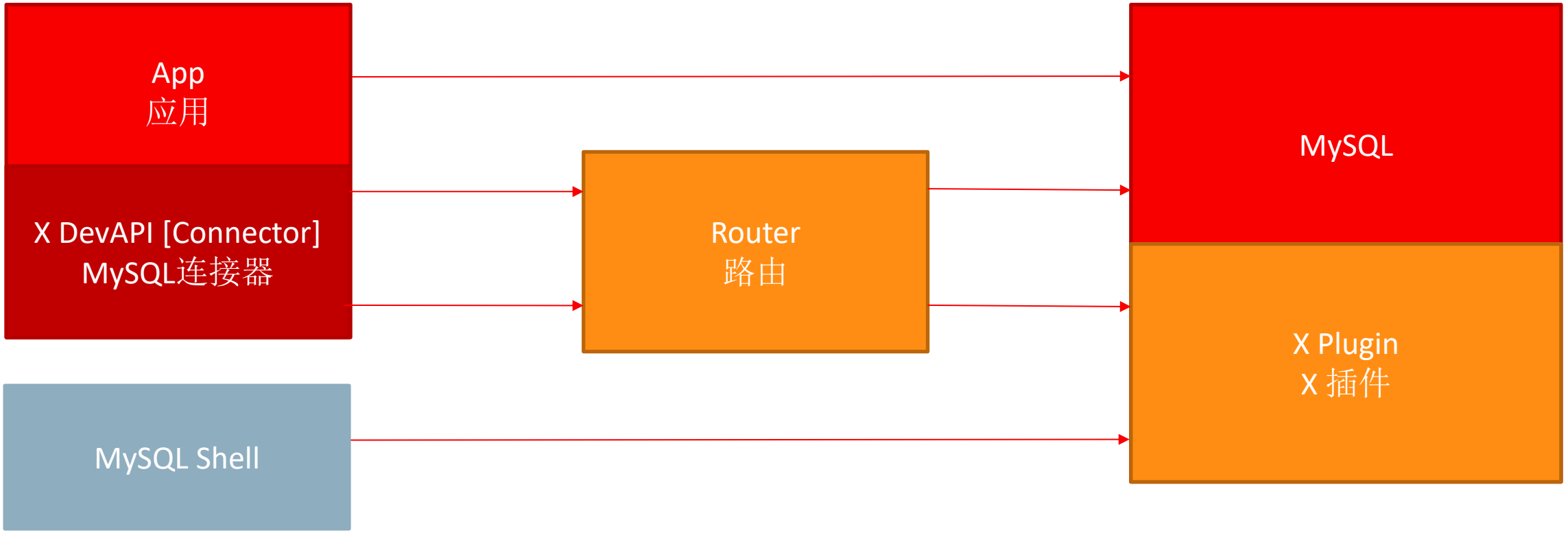
- SET @jsonempl=(
  SELECT JSON_ARRAYAGG(
- JSON_OBJECT("id", id,
  "name", name,
  "age", age))
- FROM mydb.employees);
- SELECT JSON_PRETTY(@jsonempl)\G

```
JSON_PRETTY(@jsonempl): [
  {
    "id": 1,
    "age": 34,
    "name": "John"
  },
  {
    "id": 2,
    "age": 40,
    "name": "Mary"
  },
  {
    "id": 3,
    "age": 44,
    "name": "Mike"
  }
]
```

- JSON → TABLE
- mysql> set @myjson=' [{ "id": 1, "age": 34, "name": "John" },
- { "id": 2, "age": 40, "name": "Mary" },
- { "id": 3, "age": 44, "name": "Mike" }]‘ ;
- create table mydb.newemployees as
- SELECT * from JSON_TABLE(@myjson, '$[*]'
- COLUMNS(
- id INT PATH '$.id',
- name VARCHAR(45) PATH '$.name',
- age INT PATH '$.age')) as emps;
- select * from mydb.newemployees;

```
+------+------+------+
| id   | name | age  |
+------+------+------+
|    1 | John |   34 |
|    2 | Mary |   40 |
|    3 | Mike |   44 |
+------+------+------+
```

主要组件



无模式化数据 【Schemaless Data】

- X DevAPI用上CRUD API，可以处理无模式数据
- 在文档存储方法上，允许在不用SQL的情况下处理数据

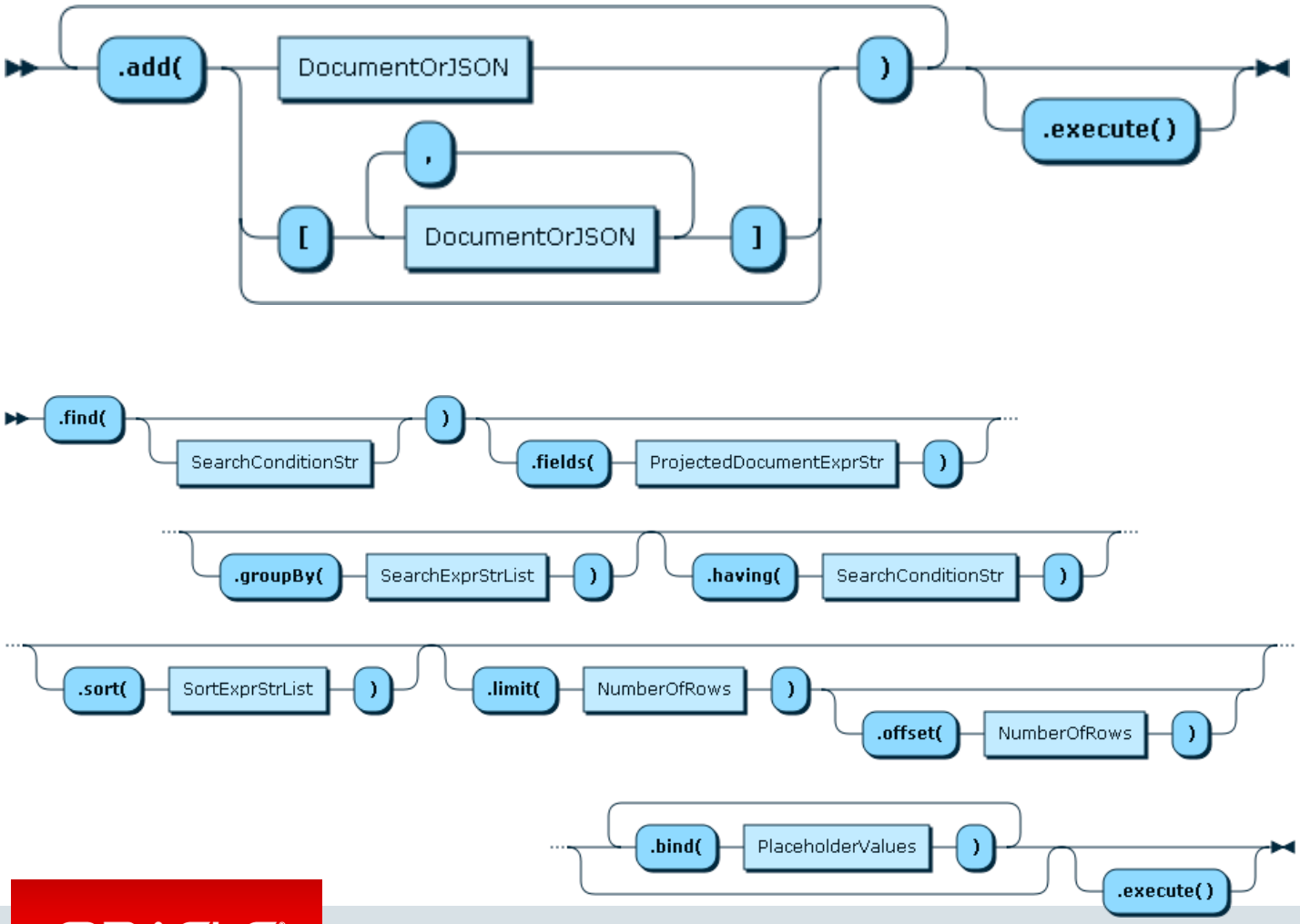
```
> users.add({ name: 'Rui' }).add({ name: 'Johannes' })
Query OK, 2 items affected (0.0373 sec)

> users.find()
[
    {
        "_id": "00005b50ced40000000000000001",
        "name": "Rui"
    },
    {
        "_id": "00005b50ced40000000000000002",
        "name": "Johannes"
    }
]
2 documents in set (0.0009 sec)
```

```
> users.modify('true').set('team', 'nodejs')
Query OK, 2 items affected (0.0751 sec)

> users.find('name = "Rui"')
[
    {
        "_id": "00005b50ced40000000000000001",
        "name": "Rui",
        "team": "nodejs"
    }
]
1 document in set (0.0026 sec)
```

易于编写和使用的 API

- 代码从 ***getSession() 开头***
- 更易读，可维护（甚至可测试）
- 操作封装在单行语义方法中
- 为重构任务提供很好的框架
- 提示和自动完成【auto-completion】提供支持

CollectionAdd / CollectionFind




```
collection.add({ name: 'foo', age: 42 })
  .add({ name: 'bar', age: 23 })
  .execute()

collection.add([
  { name: 'baz', age: 50 },
  { name: 'qux', age: 25 }
]).execute()
```

```
collection.find('name = :name')
  .bind('name', 'foo')
  .fields('COUNT(age) AS age')
  .groupBy('age')
  .having('age > 42')
  .sort('age DESC')
  .limit(10)
  .offset(5)
  .execute()
```

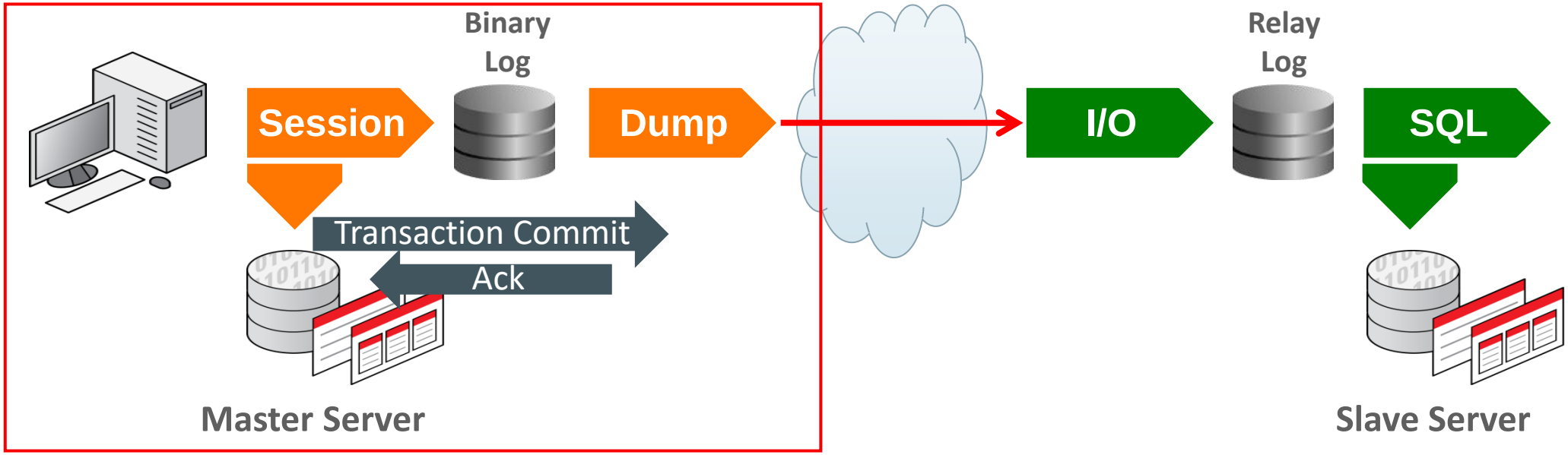
事务和保存点

- 在会话范围内，提供原子性操作
  - 在会话范围内创建，提交或回滚事务
  - 在这些事务中创建，释放或回滚到中间保存点

```
try {
  session.startTransaction()
  // run some operations (1)
  session.createSavepoint('foo')
  // run more operations (2)
  session.releaseSavepoint('foo')
  session.commit()
} catch (err) {
  try {
    session.rollbackTo('foo') // go to (2)
  } catch (err) {
    session.rollback() // revert the entire thing
  }
}
```

MySQL 8.0 不同部署方案

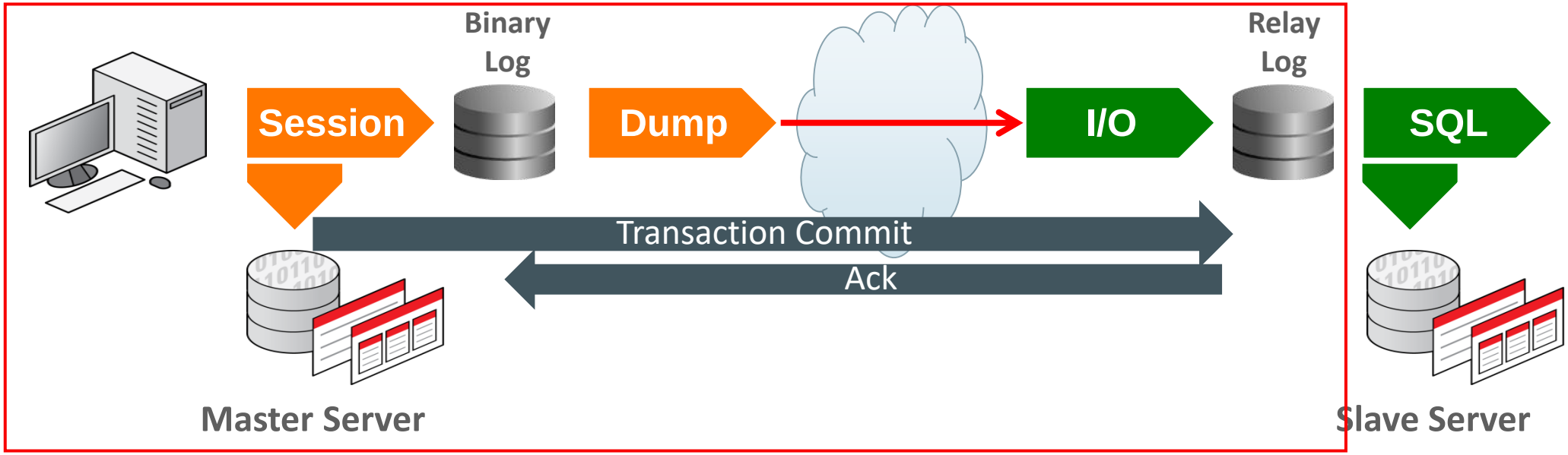
MySQL Replication – 异步复制



- Session thread updates requests to Storage Engine from application, and changes are written to binlog before apply to storage engine

- Dump thread reads event from binlog and propagate them to slave server
- I/O thread read replication events, stores them to relay log
- SQL thread: reads relay log and applies them to storage engines

MySQL Replication – 半同步复制

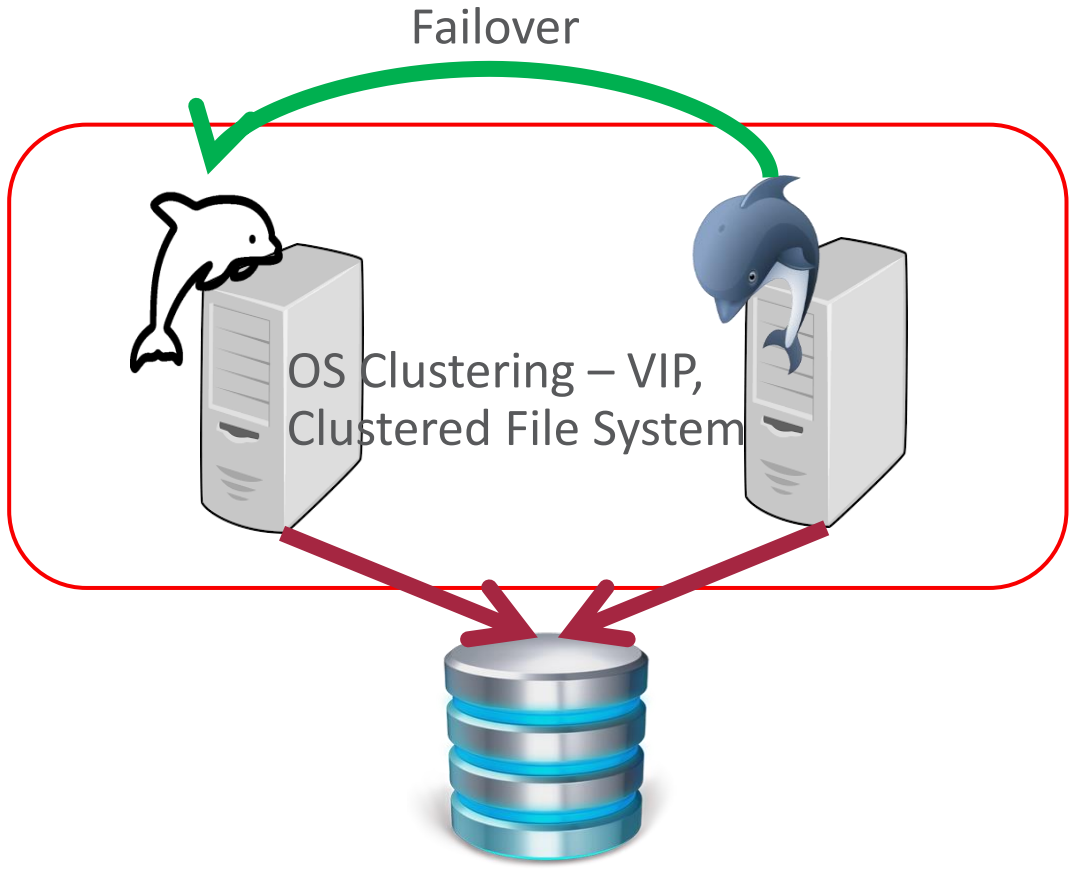


- Session thread updates requests to SE from application, and changes are written to binlog before apply to storage engine
- Dump thread reads event from binlog and propagate them to slave server
- I/O thread read replication events, stores them to relay log

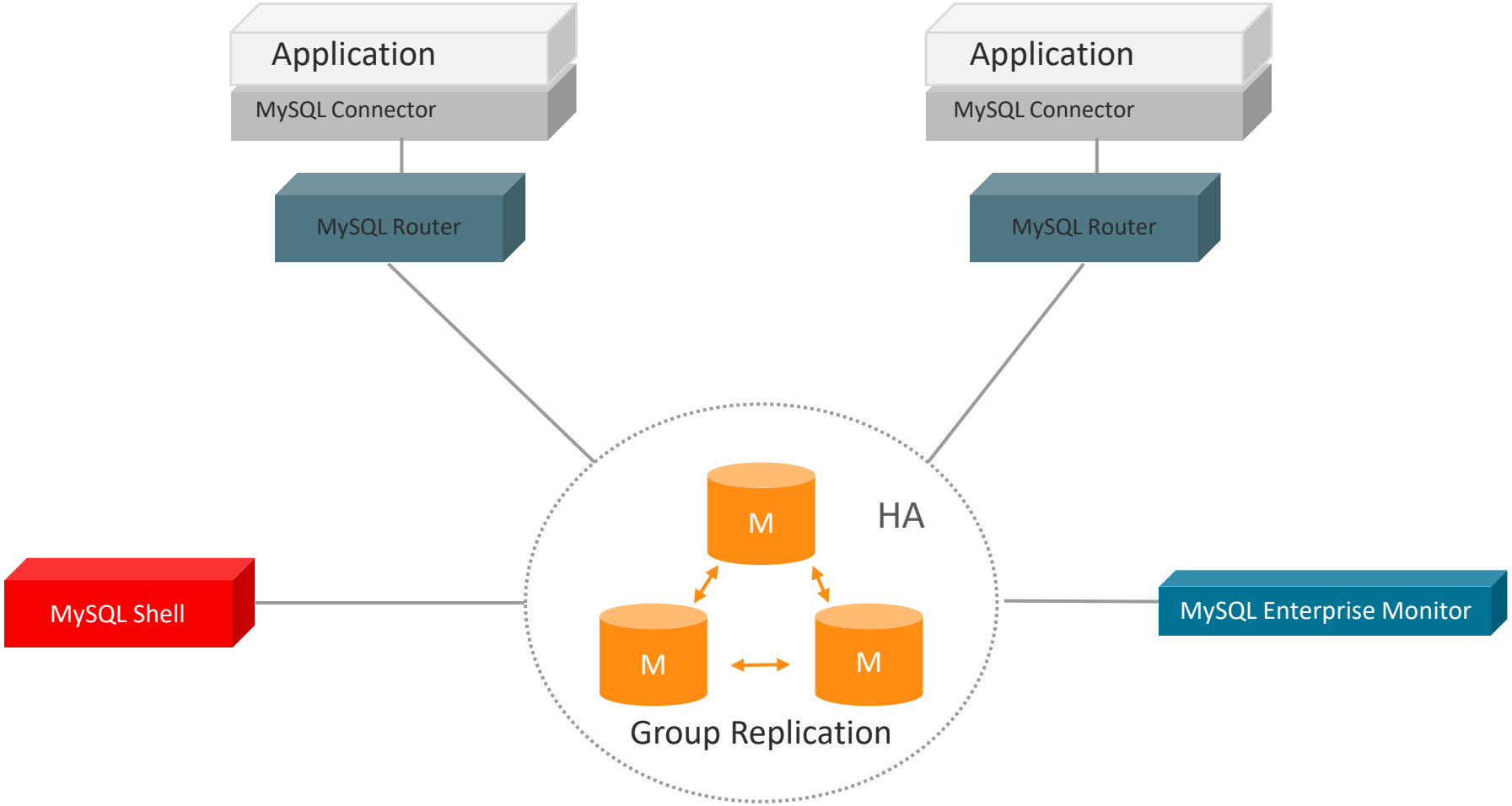
- SQL thread: reads relay log and applies them to storage engines

HA （共享存储） – Active / Passive

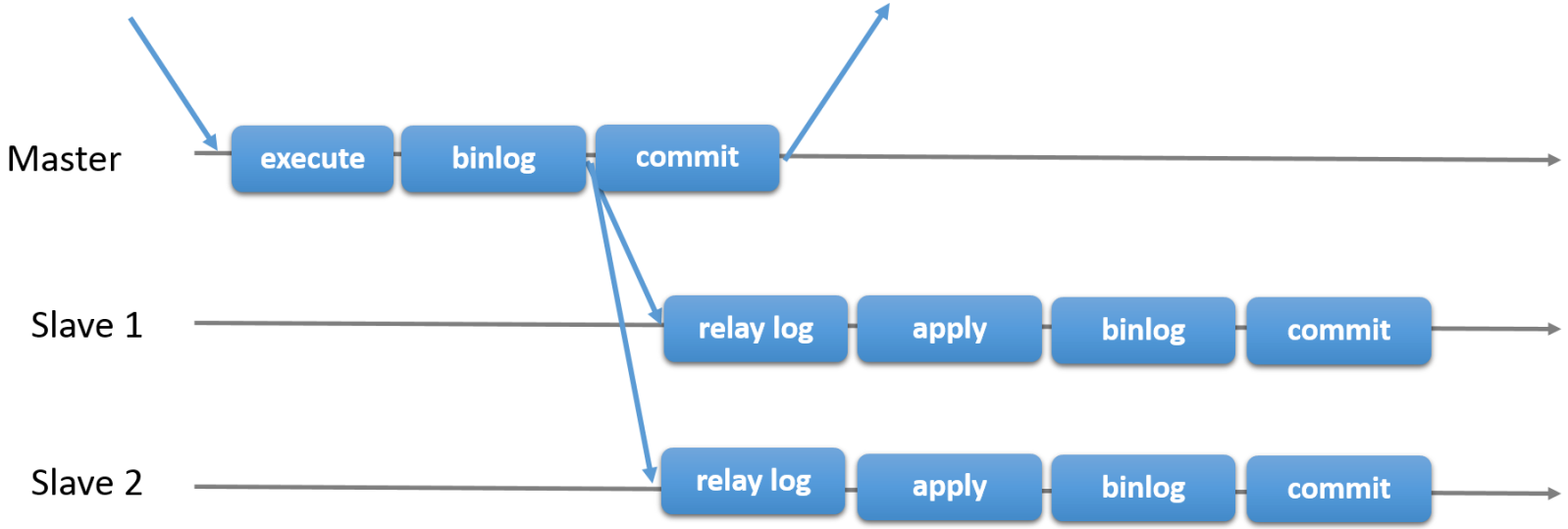
- 条件
  - 需要群集软件或在磁盘层上配置DRBD（可能需要额外的成本）
- 好处
  - VIP – 通用 / 成熟
  - 大多数情况下停机时间非常短
  - 故障时无数据丢失
- 坏处
  - 磁盘是单点故障
  - 在大批量DML失败时，崩溃恢复可能要更长停机时间
  - 没有水平可扩展性



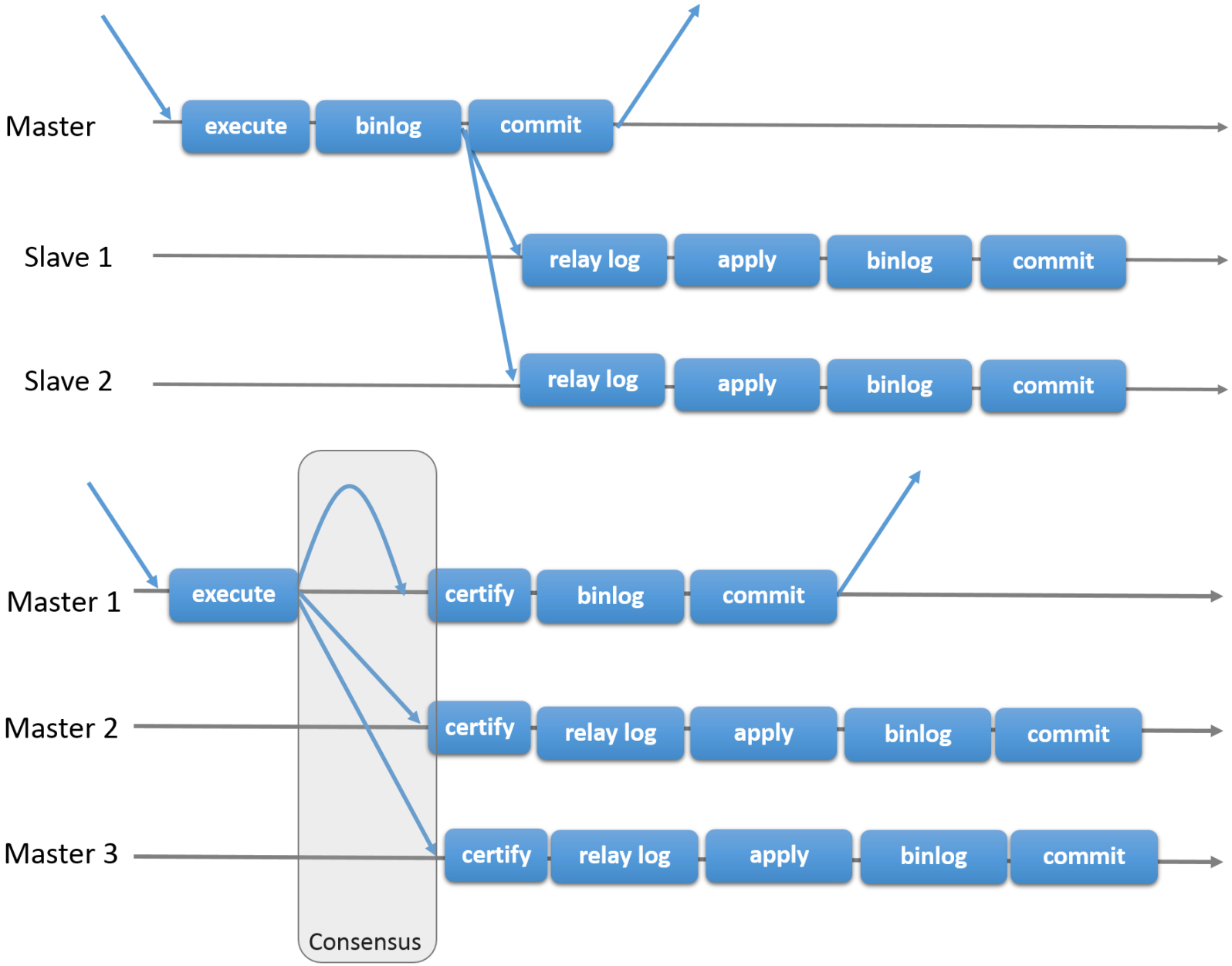
MySQL InnoDB Cluster: 架构



半同步复制

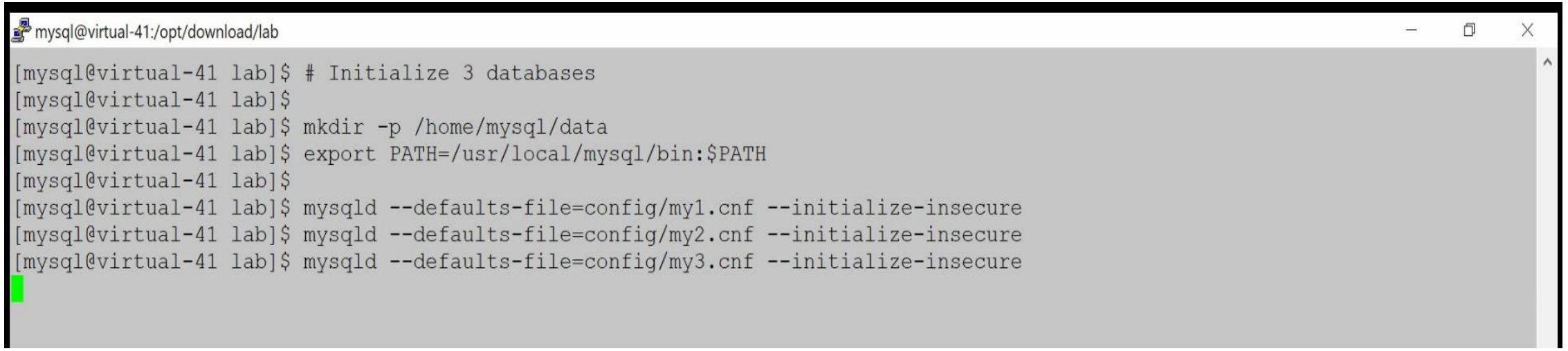


半同步复制 vs 组复制



Initialize 3 database

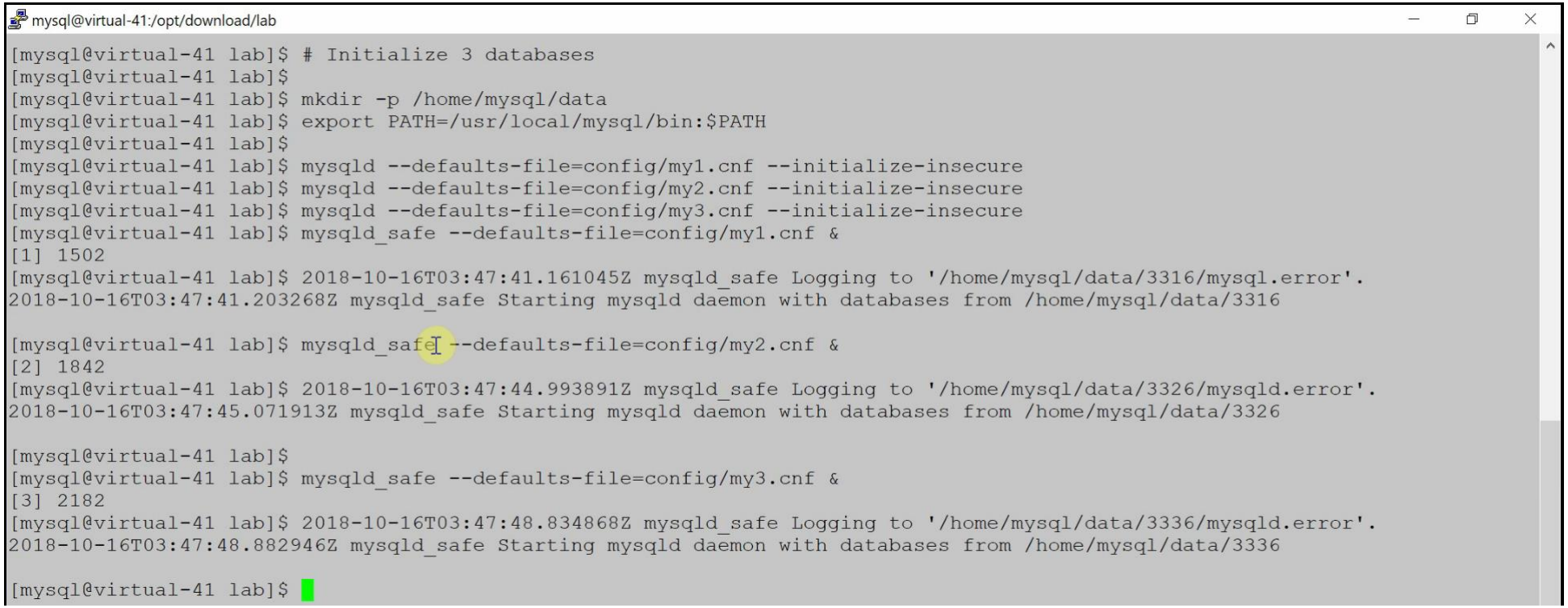
mysqld --defaults-file=<config file> --initialize-insecure



Start Up 3 database

mysqld_safe --defaults-file=<config file>

(3 port # - 3316, 3326, 3336)



Configure REMOTE Group Replication ADMIN User to each of the MySQL Server (gradmin@% / grpass) - using MySQL Shell (mysqlsh)

```
mysqlsh>dba.configureInstance('<connection>',
{clusterAdmin:'gradmin', clusterAdminPassword:'grpass'})
```

```
MySQL  localhost:3316 ssl  JS  dba.configureInstance('root@localhost:3316', {clusterAdmin:'gradmin',clusterAdminPassword:'g
rpass'})
Please provide the password for 'root@localhost:3316':
Save password for 'root@localhost:3316'? [Y]es/[N]o/Ne[v]er (default No):
Configuring local MySQL instance listening at port 3316 for use in an InnoDB cluster...

This instance reports its own address as virtual-41.localhost
Clients and other cluster members will communicate with it through this address by default. If this is not correct, the repor
t_host MySQL system variable should be changed.
Assuming full account name 'gradmin'@'%' for gradmin

The instance 'localhost:3316' is valid for InnoDB cluster usage.

Cluster admin user 'gradmin'@'%' created.
```

```
 MySQL  localhost:3316 ssl  JS  dba.configureInstance('root@localhost:3326', {clusterAdmin:'gradmin',clusterAdminPassword:'g
rpass'})
Please provide the password for 'root@localhost:3326':
Save password for 'root@localhost:3326'? [Y]es/[N]o/Ne[v]er (default No):
Configuring local MySQL instance listening at port 3326 for use in an InnoDB cluster...

This instance reports its own address as virtual-41.localhost
Clients and other cluster members will communicate with it through this address by default. If this is not correct, the repor
t_host MySQL system variable should be changed.
Assuming full account name 'gradmin'@'%' for gradmin

The instance 'localhost:3326' is valid for InnoDB cluster usage.

Cluster admin user 'gradmin'@'%' created.

 MySQL  localhost:3316 ssl  JS  dba.configureInstance('root@localhost:3336', {clusterAdmin:'gradmin',clusterAdminPassword:'g
rpass'})
Please provide the password for 'root@localhost:3336':
Save password for 'root@localhost:3336'? [Y]es/[N]o/Ne[v]er (default No):
Configuring local MySQL instance listening at port 3336 for use in an InnoDB cluster...

This instance reports its own address as virtual-41.localhost
Clients and other cluster members will communicate with it through this address by default. If this is not correct, the repor
t_host MySQL system variable should be changed.
Assuming full account name 'gradmin'@'%' for gradmin

The instance 'localhost:3336' is valid for InnoDB cluster usage.

Cluster admin user 'gradmin'@'%' created.

 MySQL  localhost:3316 ssl  JS
```

```
**************************************************************************
3 MySQL Instances are up and running.
**********************************************
REMOTE admin user (gradmin/grpass) is configured using mysqlsh
**************************************************************************
```

Create InnoDB Cluster using mysqlsh

```
mysqlsh> \connect gradmin:grpass@primary:3316
mysqlsh> dba.createCluster('mycluster')
mysqlsh> var x = dba.getCluster()
mysqlsh> x.addInstance('gradmin:grpass@primary:3326')
mysqlsh> x.addInstance('gradmin:grpass@primary:3336')
mysqlsh> x.status()
```

```
MySQL  primary:3316 ssl  JS  dba.createCluster('mycluster')
A new InnoDB cluster will be created on instance 'gradmin@primary:3316'.

Validating instance at primary:3316...

This instance reports its own address as virtual-41.localhost

Instance configuration is suitable.
Creating InnoDB cluster 'mycluster' on 'gradmin@primary:3316'...
Adding Seed Instance...

Cluster successfully created. Use Cluster.addInstance() to add MySQL instances.
At least 3 instances are needed for the cluster to be able to withstand up to
one server failure.

<Cluster:mycluster>
```

```
**************************************************************************
3 MySQL Instances are up and running.
**********************************************
REMOTE admin user (gradmin/grpass) is configured using mysqlsh
**************************************************************************
```

Create InnoDB Cluster using mysqlsh

```
mysqlsh> \connect gradmin:grpass@primary:3316
mysqlsh> dba.createCluster('mycluster')
mysqlsh> var x = dba.getCluster()
mysqlsh> x.addInstance('gradmin:grpass@primary:3326')
mysqlsh> x.addInstance('gradmin:grpass@primary:3336')
mysqlsh> x.status()
```

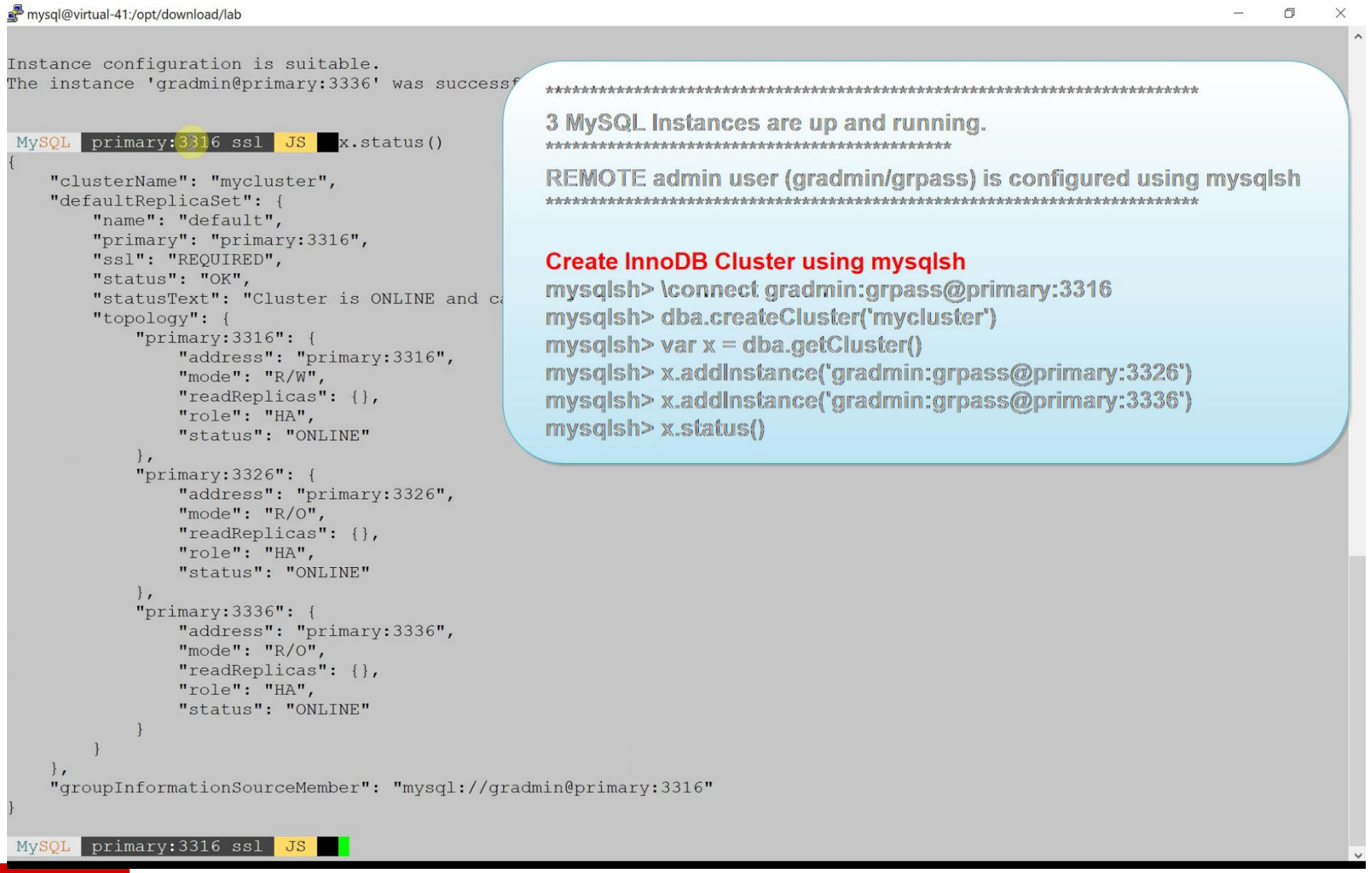
```
 MySQL  primary:3316 ssl  JS  var x = dba.getCluster()
 MySQL  primary:3316 ssl  JS  x.addInstance('gradmin:grpass@primary:3326')
A new instance will be added to the InnoDB cluster. Depending on the amount of
data on the cluster this might take from a few seconds to several hours.

Adding instance to the cluster ...

Validating instance at primary:3326...

This instance reports its own address as virtual-41.localhost

Instance configuration is suitable.
The instance 'gradmin@primary:3326' was successfully added to the cluster.
```

```
**************************************************************************
3 MySQL Instances are up and running.
**********************************************
REMOTE admin user (gradmin/grpass) is configured using mysqlsh
**************************************************************************
```

Create InnoDB Cluster using mysqlsh

```
mysqlsh> \connect gradmin:grpass@primary:3316
mysqlsh> dba.createCluster('mycluster')
mysqlsh> var x = dba.getCluster()
mysqlsh> x.addInstance('gradmin:grpass@primary:3326')
mysqlsh> x.addInstance('gradmin:grpass@primary:3336')
mysqlsh> x.status()
```

```
 MySQL  primary:3316 ssl  JS  x.addInstance('gradmin:grpass@primary:3336')
A new instance will be added to the InnoDB cluster. Depending on the amount of
data on the cluster this might take from a few seconds to several hours.

Adding instance to the cluster ...

Validating instance at primary:3336...

This instance reports its own address as virtual-41.localhost

Instance configuration is suitable.
The instance 'gradmin@primary:3336' was successfully added to the cluster.
```



Setup the MySQL Router

1. Creating the MySQL Router Configuration files by bootstrapping

   # mysqlrouter --bootstrap gradmin:grpass@primary:3316 --directory config/mysqlrouter01

2. Starting the MySQL Router

Within the config/mysqlrouter01, there is a 'start.sh' script created. Execute the 'start.sh' to bring the MySQL Router up and running with PORT# 6446 for RW and 6447 for RO

```
[mysql@virtual-41 lab]$ mysqlrouter --bootstrap=gradmin:grpass@primary:3316 --directory config/myrouter01 --force

Bootstrapping MySQL Router instance at '/opt/download/lab/config/myrouter01'...
Checking for old Router accounts
Creating account mysql_router4_a8j3jng2oazw@'%'
MySQL Router  has now been configured for the InnoDB cluster 'mycluster'.

The following connection information can be used to connect to the cluster.

Classic MySQL protocol connections to cluster 'mycluster':
- Read/Write Connections: localhost:6446
- Read/Only Connections: localhost:6447
X protocol connections to cluster 'mycluster':
- Read/Write Connections: localhost:64460
- Read/Only Connections: localhost:64470
[mysql@virtual-41 lab]$
```

Setup the MySQL Router

1. Creating the MySQL Router Configuration files by bootstrapping

```
# mysqlrouter --bootstrap gradmin:grpass@primary:3316 --directory config/mysqlrouter01
```

2. Starting the MySQL Router

Within the config/mysqlrouter01, there is a 'start.sh' script created. Execute the 'start.sh' to bring the MySQL Router up and running with PORT# 6446 for RW and 6447 for RO

```
[mysql@virtual-41 config]$ cd myrouter01
[mysql@virtual-41 myrouter01]$ ll
total 28
drwx------ 2 mysql mysql 4096 Oct 16 11:55 data
drwx------ 2 mysql mysql 4096 Oct 16 11:55 log
-rw------- 1 mysql mysql 1318 Oct 16 11:55 mysqlrouter.conf
-rw------- 1 mysql mysql  106 Oct 16 11:55 mysqlrouter.key
drwx------ 2 mysql mysql 4096 Oct 16 11:55 run
-rwx------ 1 mysql mysql  167 Oct 16 11:55 start.sh
-rwx------ 1 mysql mysql  215 Oct 16 11:55 stop.sh
[mysql@virtual-41 myrouter01]$ cat start.sh
#!/bin/bash
basedir=/opt/download/lab/config/myrouter01
ROUTER_PID=$basedir/mysqlrouter.pid /usr/local/router/bin/mysqlrouter -c $basedir/mysqlrouter.conf &
disown %-
[mysql@virtual-41 myrouter01]$ ./start.sh
[mysql@virtual-41 myrouter01]$
```

Start Scripts in the FOLDER

Connecting to ROUTER : 6446 : PRIMARY Routing

SELECT @@port;

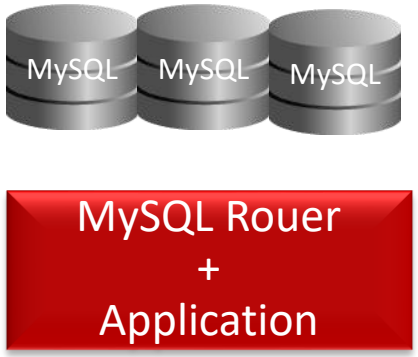
```
[mysql@virtual-41 myrouter01]$ mysql -ugradmin -pgrpass -h127.0.0.1 -P6446
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 128
Server version: 8.0.12-commercial MySQL Enterprise Server - Commercial

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select @@port;
+--------+
| @@port |
+--------+
|   3316 |
+--------+
1 row in set (0.00 sec)
```

生产系统部署的场景
MySQL InnoDB Cluster

https://dev.mysql.com/doc/refman/8.0/en/mysql-innodb-cluster-production-deployment.html

场景

- 一个数据中心

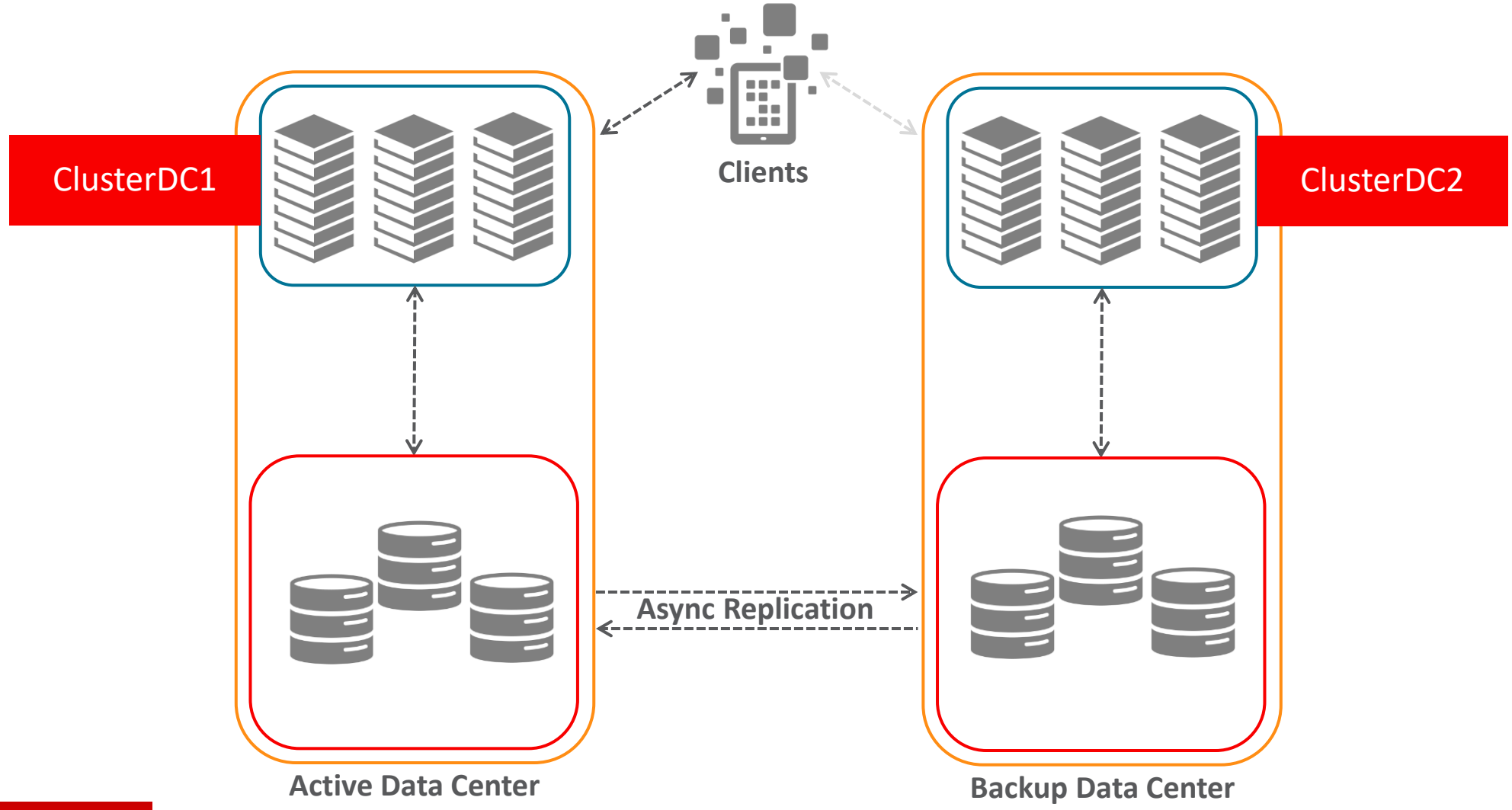


单主模式 - Single Primary Mode

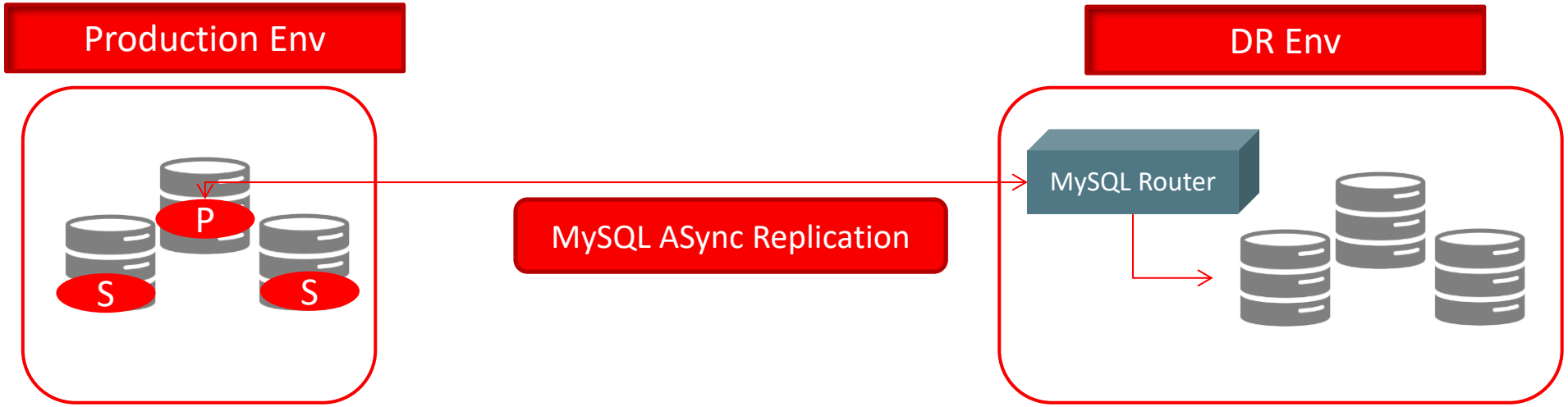
- 主服务器故障
  - 服务器自动切换
  - 应用程序连接到路由器[无影响]
- 辅助服务器故障
  - 冗余服务器可用
  - 应用程序连接到路由器[无影响]
- 故障服务器恢复
  - 数据赶上并开始作为辅助数据

- 主服务切换
  - SELECT group_replication_set_as_primary(uuid)
  - 可能机器有更多更大-资源
  - 用于维护目的

场景 – 异地机房

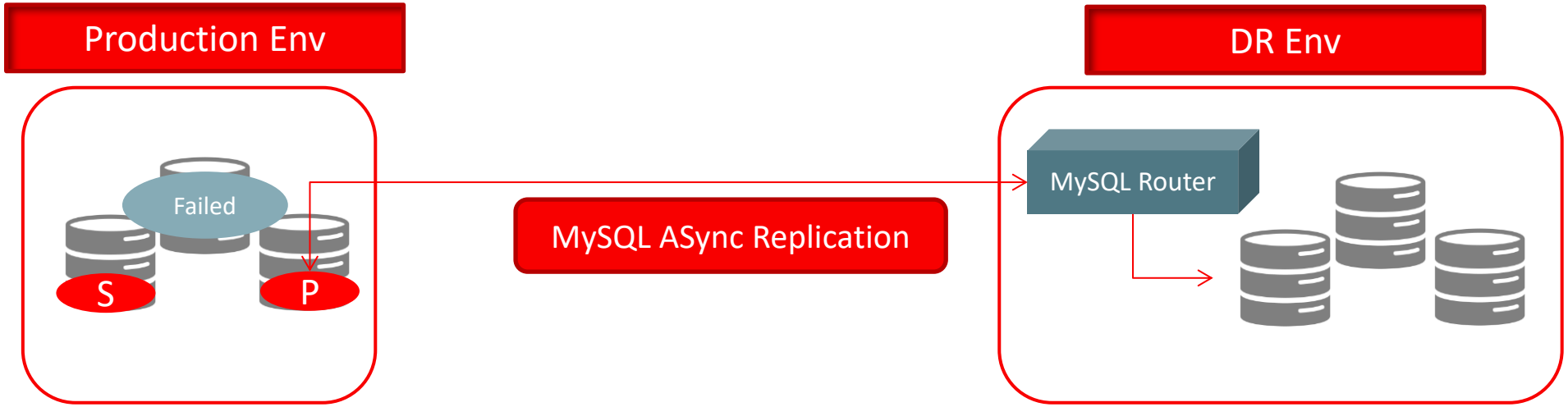


通过路由复制



在DR环境中安装MySQL路由器。 用于复制以指向生产环境中MySQL InnoDB集群的PRIMARY节点。

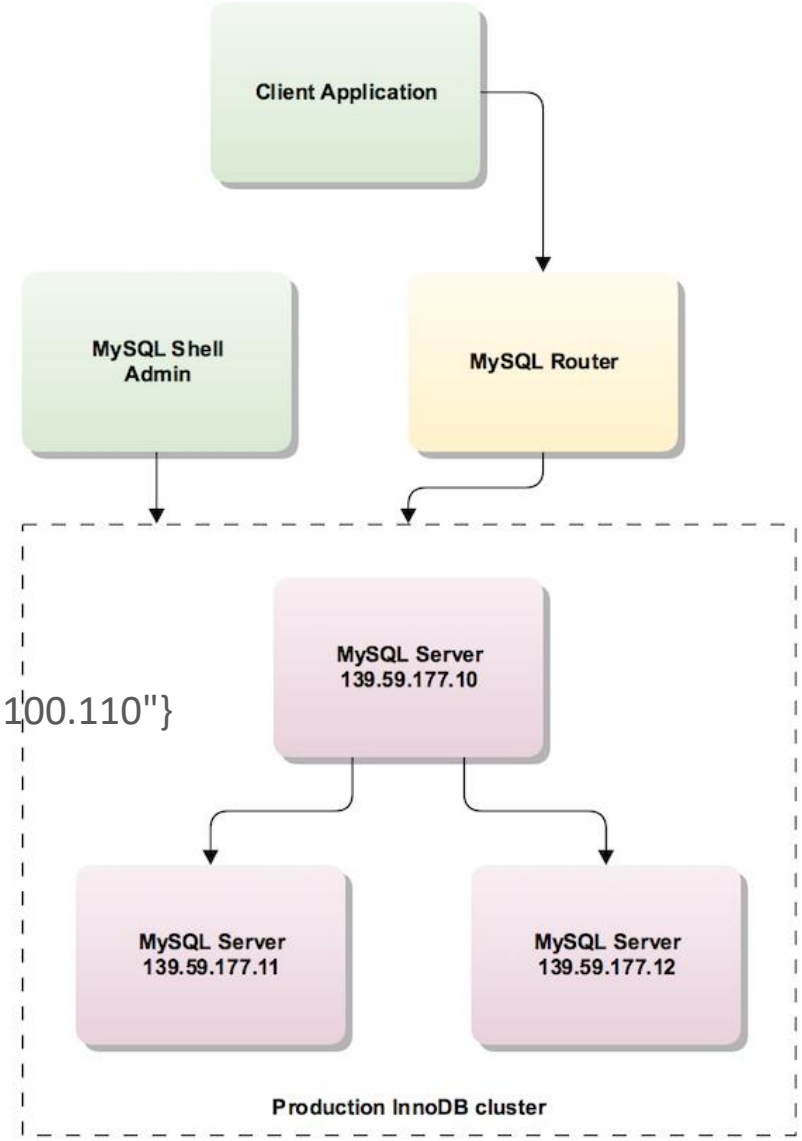
通过路由复制



如果主节点发生故障，则其中一个辅助节点将提升为主节点。 复制通道仍可以连接到活动服务器，以便在两个站点之间进行数据复制。

生产系统部署

- User Privileges
  - Remote Admin User
  - Created By <cluster>.configureInstance(…)
- IP/Hostname
  - configure ***report_host*** MySQL parameter
  - Configure WhiteList of Servers
  - mysql-js> cluster.addInstance("ic@ic-3:3306", {ipWhitelist: "203.0.113.0/24, 198.51.100.110"}
- if MySQL 8.0
  - persisted_globals_load is set to ON
- Create the Cluster
  - addInstance (…)



升级MySQL InnoDB集群
https://dev.mysql.com/doc/refman/8.0/en/group-replication-upgrade.html

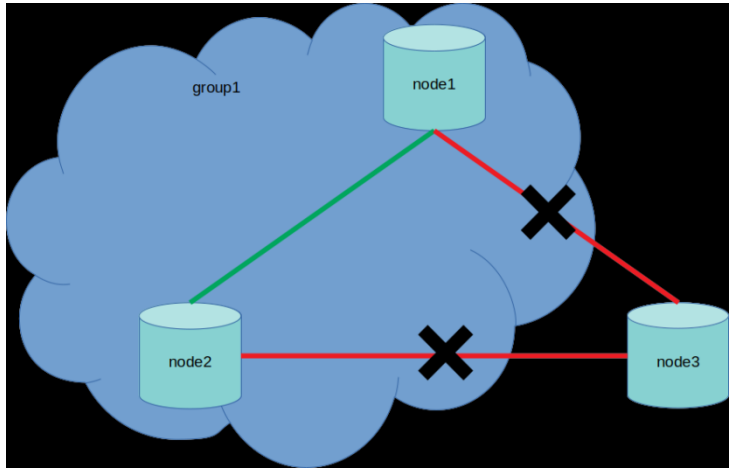
- 离线升级
  - 全部关机
  - 升级每个成员
  - 重新开始

- 在线升级
  - 注意： 具有较低版本的组+具有较高版本的新节点（检查版本兼容性）
  - 仅在低版本上写入（不是新加入更高版本）
- Upgrade
  - 首先升级RO节点（最后在RW节点上）
  - set persist group_replication_start_on_boot = 0
  - shutdown
  - my.cnf更改basedir指向NEW VERSION
  - restart，确认节点不在群组内
  - mysql_upgrade（来自New Binary）
  - set persist group_replication_start_on_boot = 1
  - restart - 加入群组

网络可靠性

https://mysqlhighavailability.com/group-replication-coping-with-unreliable-failure-detection/

- Default the heart beat is 5 seconds.
- In MySQL 8.0.13, new setting ：*group_replication_member_expel_timeout*

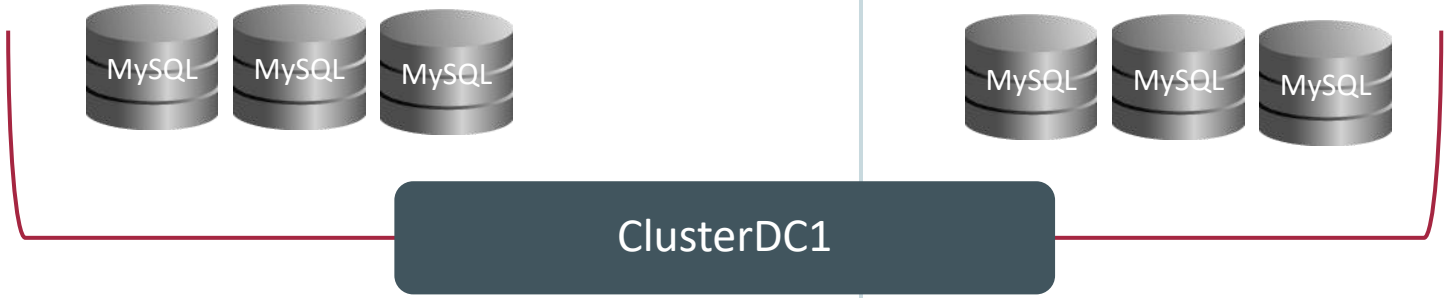


Node3如果在group_replication_member_expel_timeout
时间内重新加入，则GROUP将接受。

执行TIMEOUT，节点将根据
group_replication_exit_state_action
中定义的值执行操作

MySQL InnoDB Cluster - 异地复制场景

- DC1
- DC2

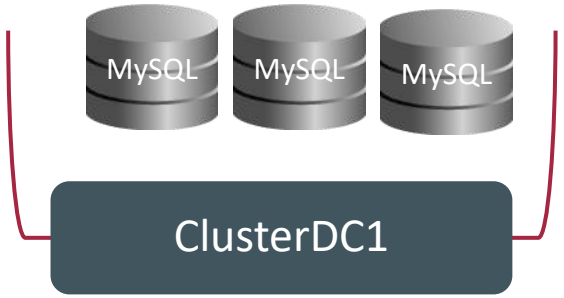


mysql_innodb_cluster_metadata
- clusterDC1

https://lefred.be/content/mysql-innodb-cluster-with-2-data-centers-for-disaster-recovery-howto/

MySQL InnoDB Cluster - 异地复制场景

- DC1



mysql_innodb_cluster_metadata
- clusterDC1

- DC2

cluster.removeInstance (...)

MySQL InnoDB Cluster - 异地复制场景

- DC1

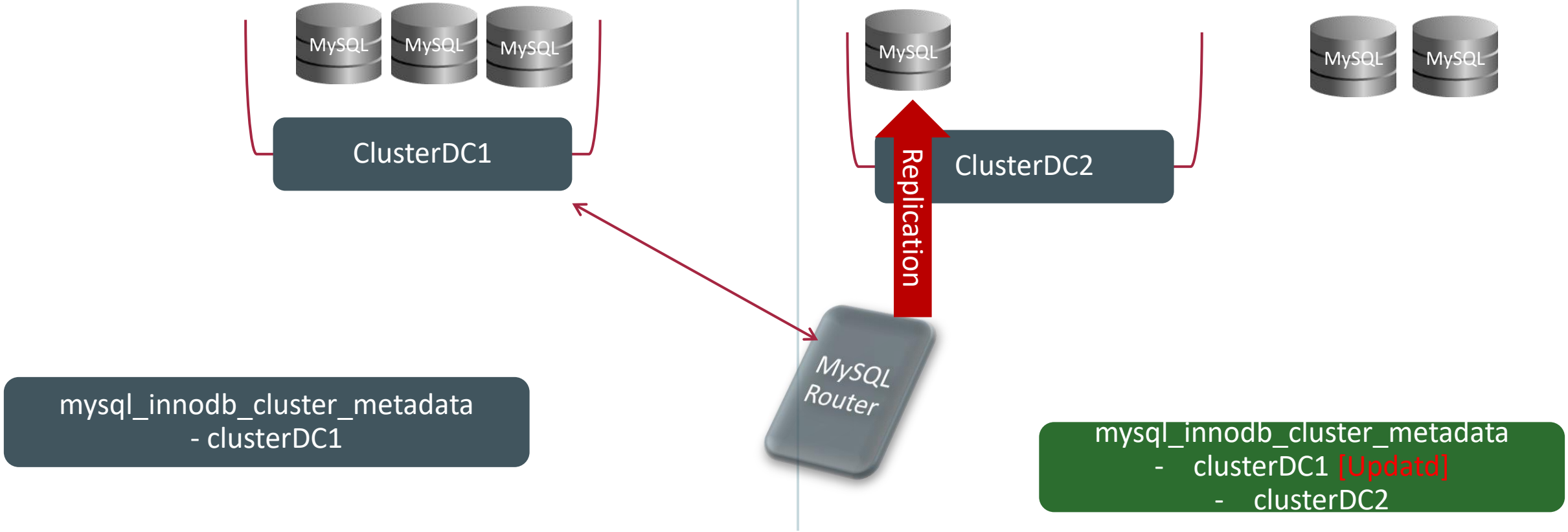
mysql_innodb_cluster_metadata
- clusterDC1

- DC2

mysql_innodb_cluster_metadata
- clusterDC1
- clusterDC2

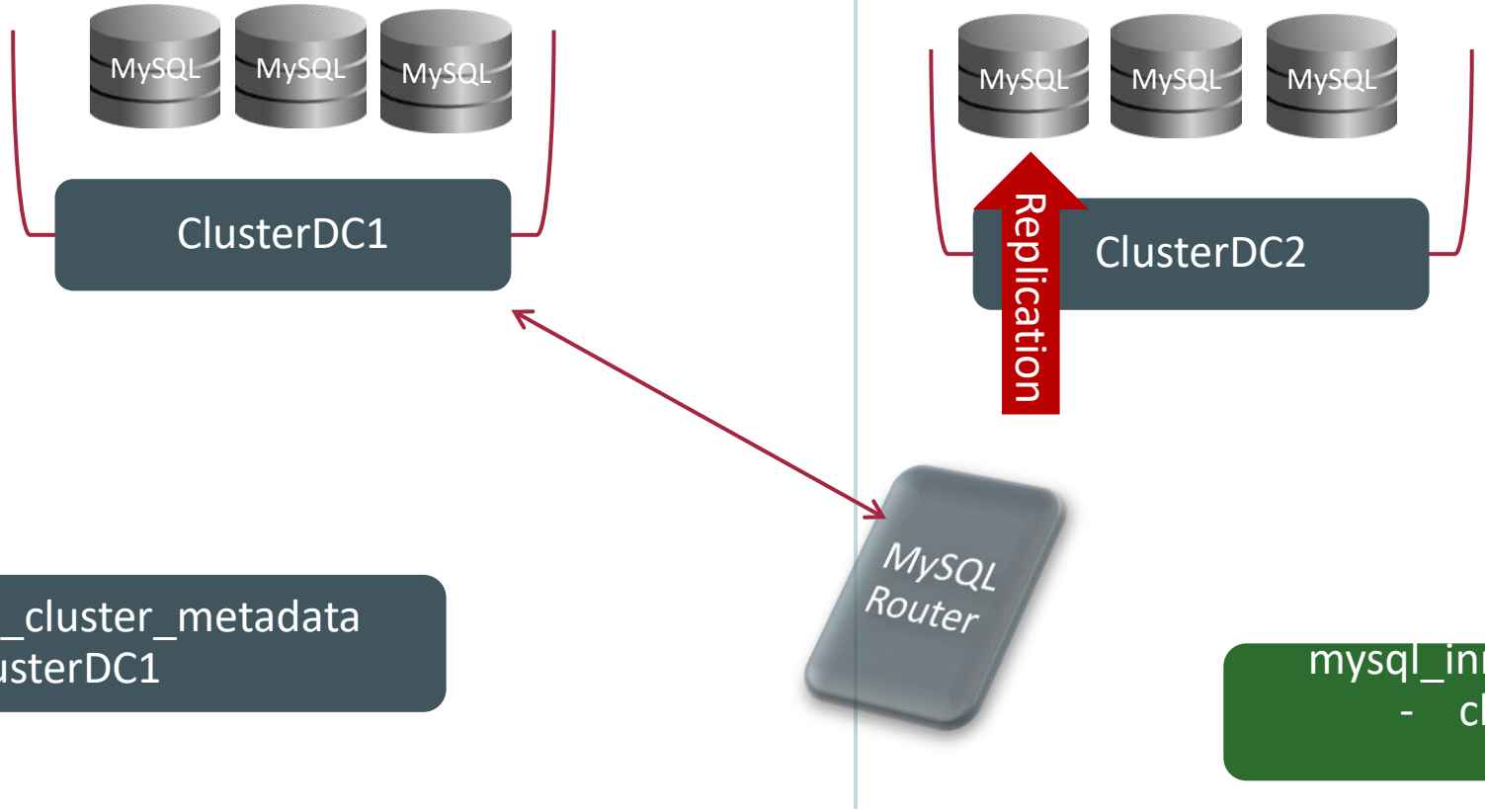
MySQL InnoDB Cluster - 异地复制场景

- DC1
- DC2



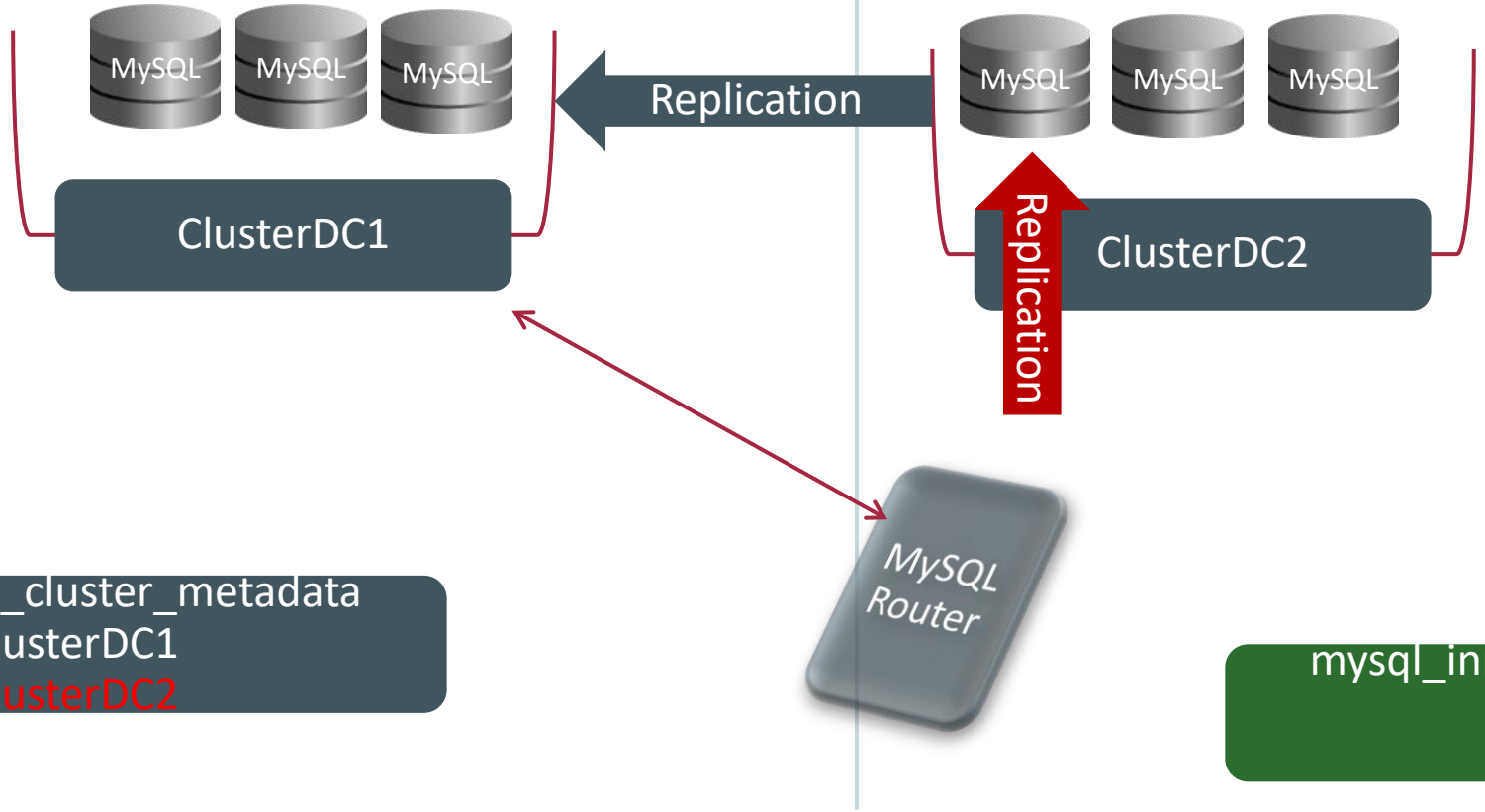
MySQL InnoDB Cluster - 异地复制场景

- DC1
- DC2



MySQL InnoDB Cluster - 异地复制场景

- DC1
- DC2



mysql_innodb_cluster_metadata
- clusterDC1
- clusterDC2

mysql_innodb_cluster_metadata
- clusterDC1
- clusterDC2

总结

- Oracle commits to MySQL and delivered Quality Product and Supporting Open Source.
- HA solutions to MySQL – SIMPLE, EASY and Perform!
- MySQL 8.0 新功能 – 更好用，更易于使用，高可用更方便，更可靠！
- Q &A


ORACLE®

关于「3306π」社区

围绕MySQL核心技术，将互联网行业中最重要的数据化解决方案带到传统行业中

囊括其他开源技术，Redis、MongoDB、Hbase、Hadoop、ElasticSearch、Storm、Spark等

在全面互联网化的大趋势下，将互联网新鲜的核心技术理念带到传统行业里，构建良好交流互动环境

分享干货知识，即便是赞助商，也要求如此，拒绝放水

加入「3306π」社区

社区公众号

社区QQ群