ORACLE

MySQL

MySQL 数据库架构

灾难恢复解决方案

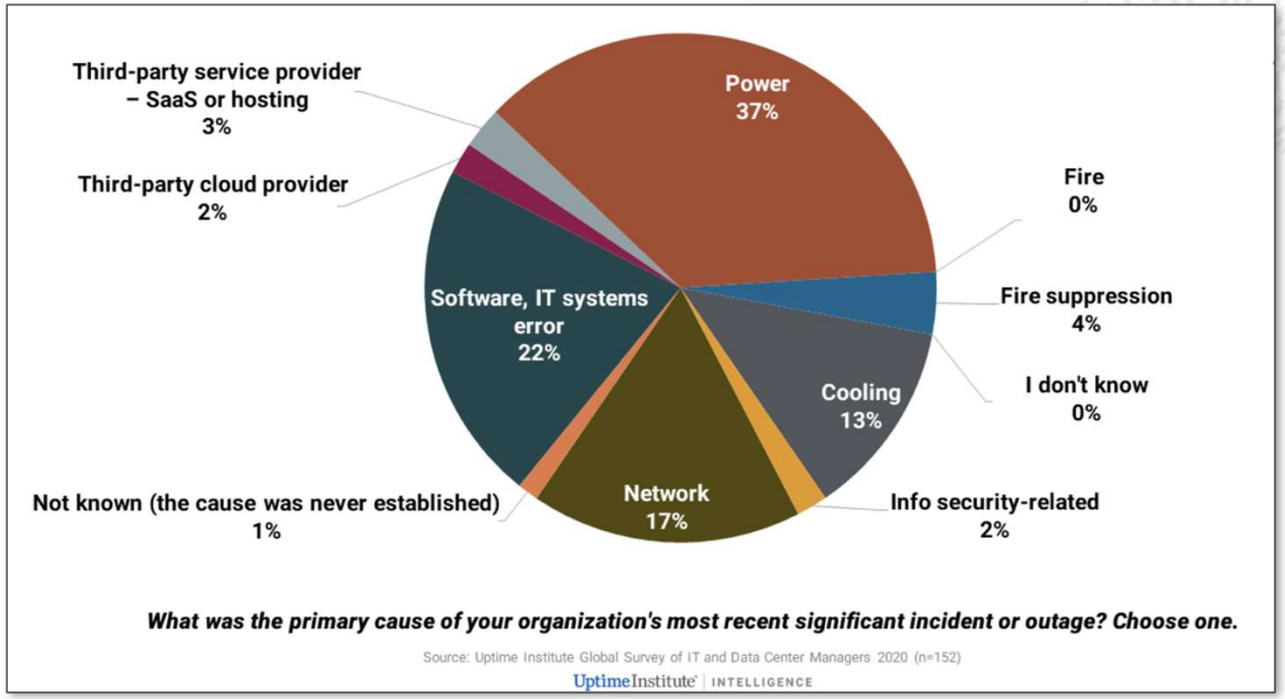
MySQL InnoDBClusterSet介绍

MySQL SE 罗伟文

Safe Harbor Statement

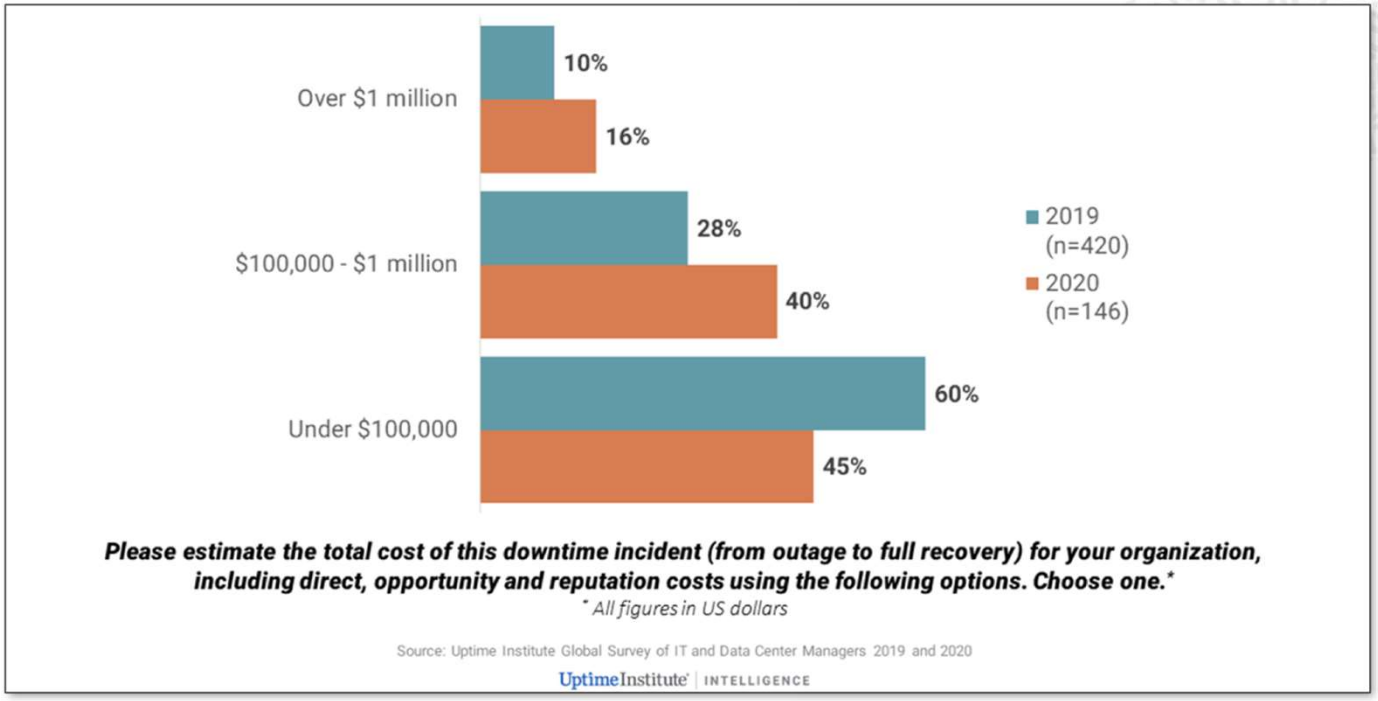
以下内容旨在概述我们的一般产品方向。它仅供参考，不得纳入任何合同。它并不提供任何材料、代码或功能的承诺，不应据此做出购买决策。 Oracle 产品描述的任何特性或功能的开发、发布和时间安排仍由 Oracle 自行决定。

IT 灾难和中断：主要原因



停电是导致重大中断的最大原因

IT 灾难和中断：代价不断上升



超过一半经历过代价超过10万美元的中断。

IT 灾难和中断：例子

5 小时的计算机中断代价 1.5 亿美元。该航空公司最终在停电当天取消了约1，000个航班，并在接下来的两天内停飞了另外1，000个航班。

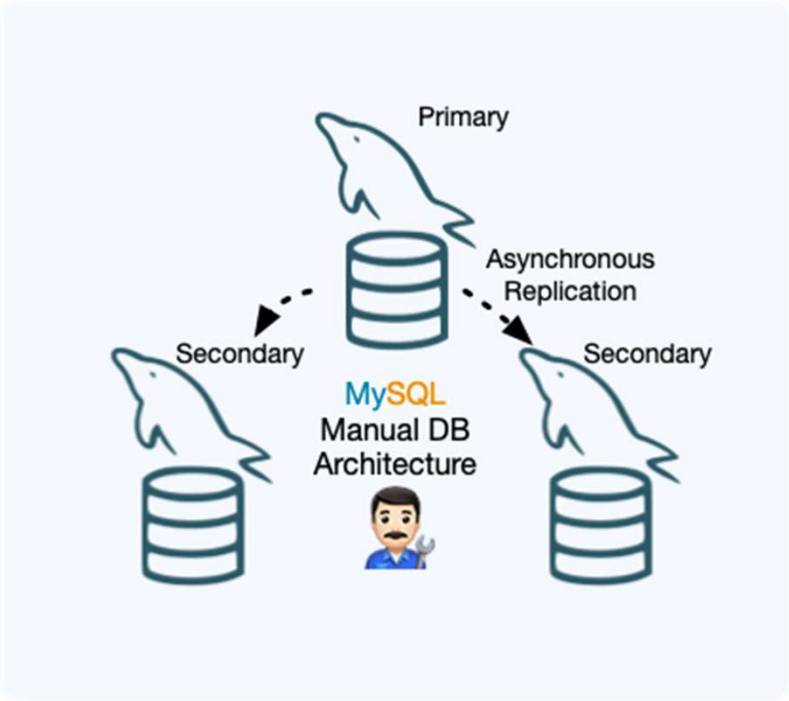
在法国云服务irm的故障后，数百万个网站下线了。Anger预计将使公司损失超过1.05亿欧元。

由于取消约130个航班和延误200个航班，成千上万的乘客滞留在世界各地。

数以百万计的银行客户无法访问在线帐户。银行花了将近2天的时间才恢复正常运作。

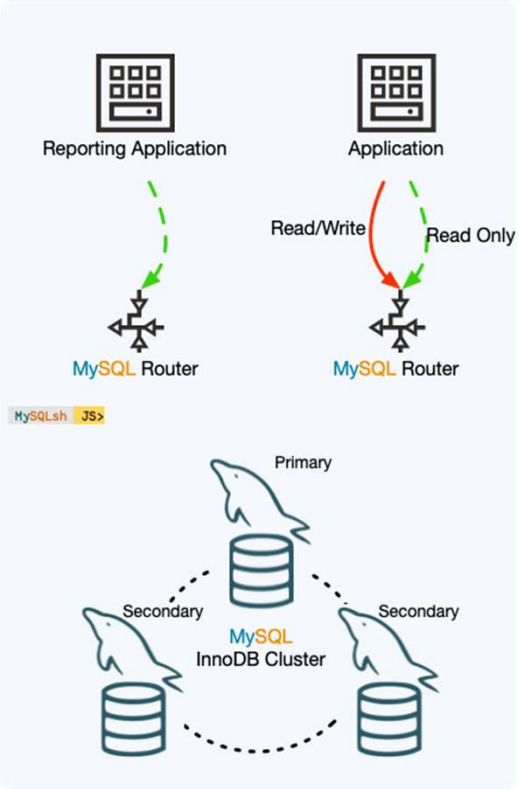
过去、现在和未来

"过去" - 手动



- 设置复制拓扑通常是手动完成的，需要执行许多步骤包括用户管理、恢复备份、配置复制...
- MySQL只提供技术部分，让用户来设置（始终自定义的）架构。
- 甚至需要其他软件...为DBA和专家带来了大量工作，他们花时间自动化和集成他们的定制架构

Present - Solutions!

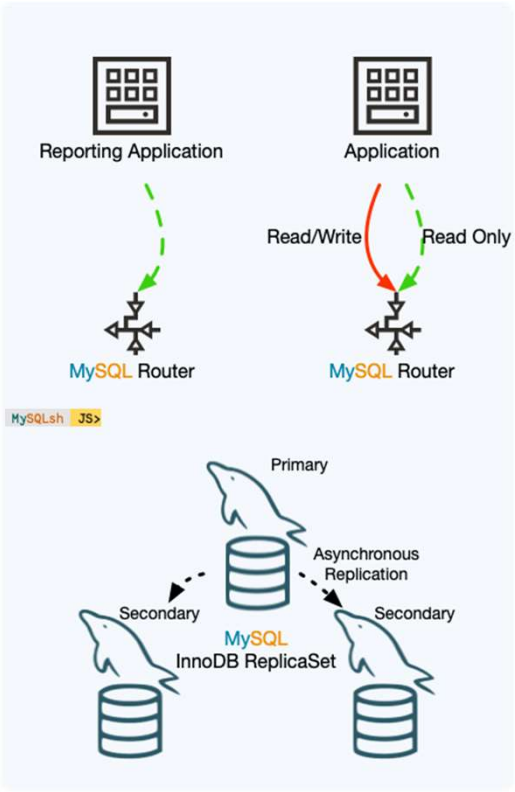


RPO = 0
RTO = seconds (自动故障转移)

2016 - MySQL InnoDB Cluster

- MySQL 组复制：自动成员身份更改、网络分区处理、一致性...
- MySQL Shell 提供强大的界面，有助于自动化和集成所有组件
- InnoDB克隆以自动生成成员，完全集成在InnoDB中
- MySQL Router
- MySQL Server

Present - Solutions!



RPO !=0
RTO =minutes (manual failover)

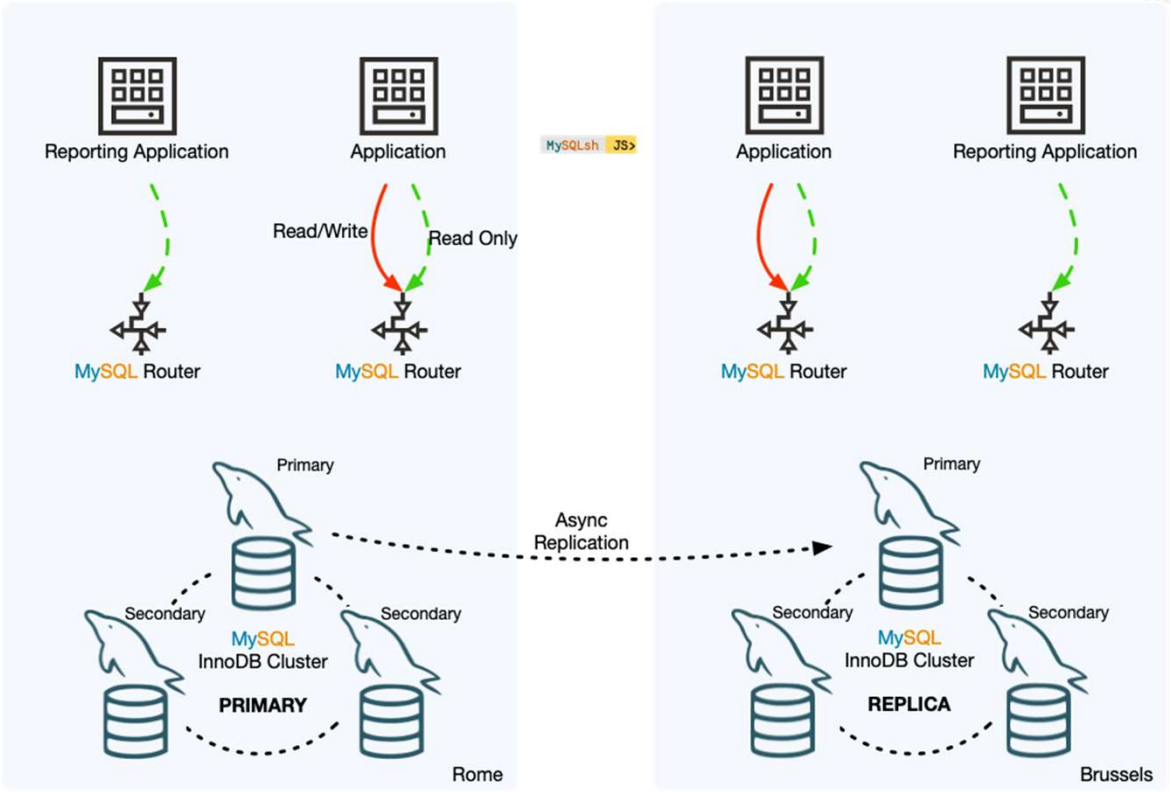
2020 - MySQL InnoDB Replicaset

- "经典"、"异步"的基于复制的解决方案，完全集成
- MySQL Shell
- MySQL Router
- MySQL Server

MySQL InnoDB ClusterSet

一个或者多个MySQL InnoDB Cluster副本连接到一个主MySQL InnoDB Cluster

高可用 (区域内的失败)



RPO = 0

RTO = 秒级 (自动故障转移)

灾难恢复（区域故障）

RPO != 0

RTO = 分钟或更长时间（手动故障转移）

无写入性能影响

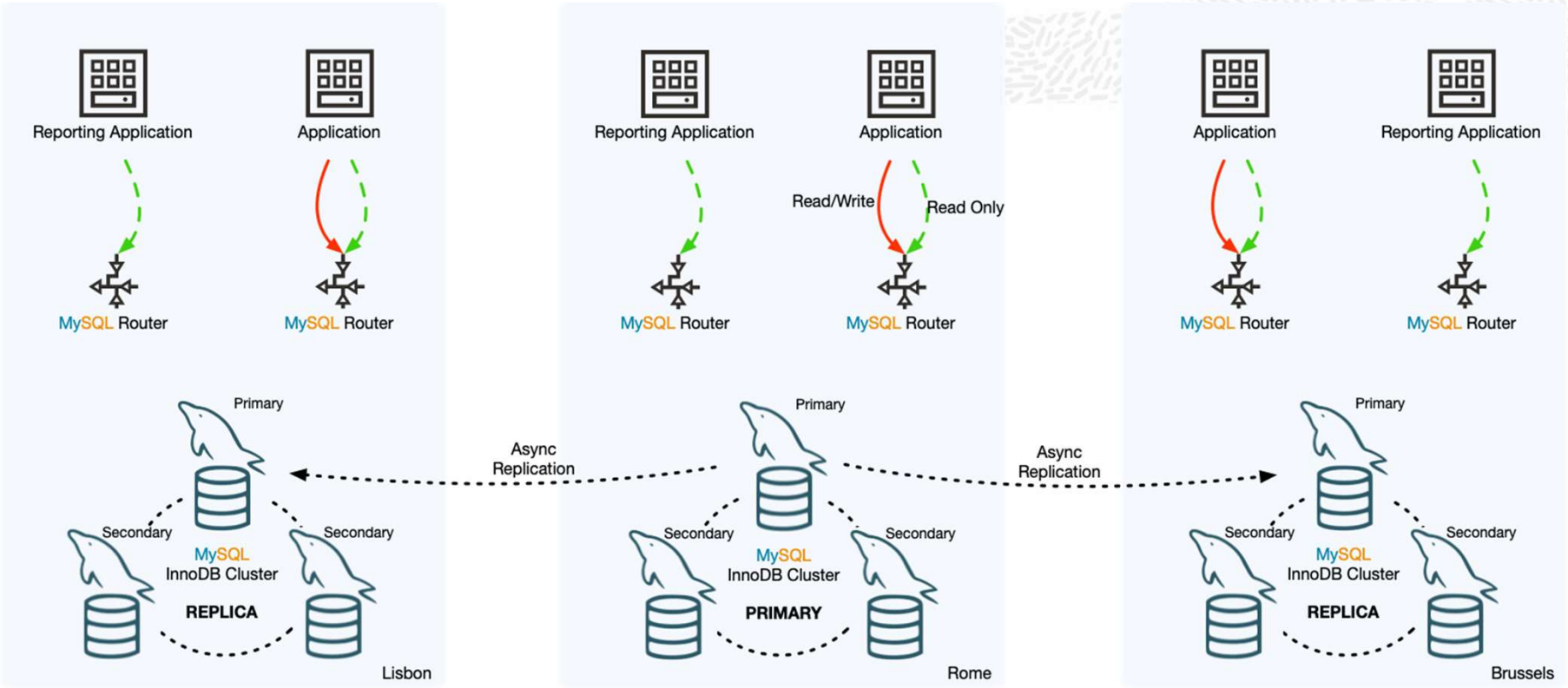
特点

- 简单易用
- 熟悉的界面和可用性

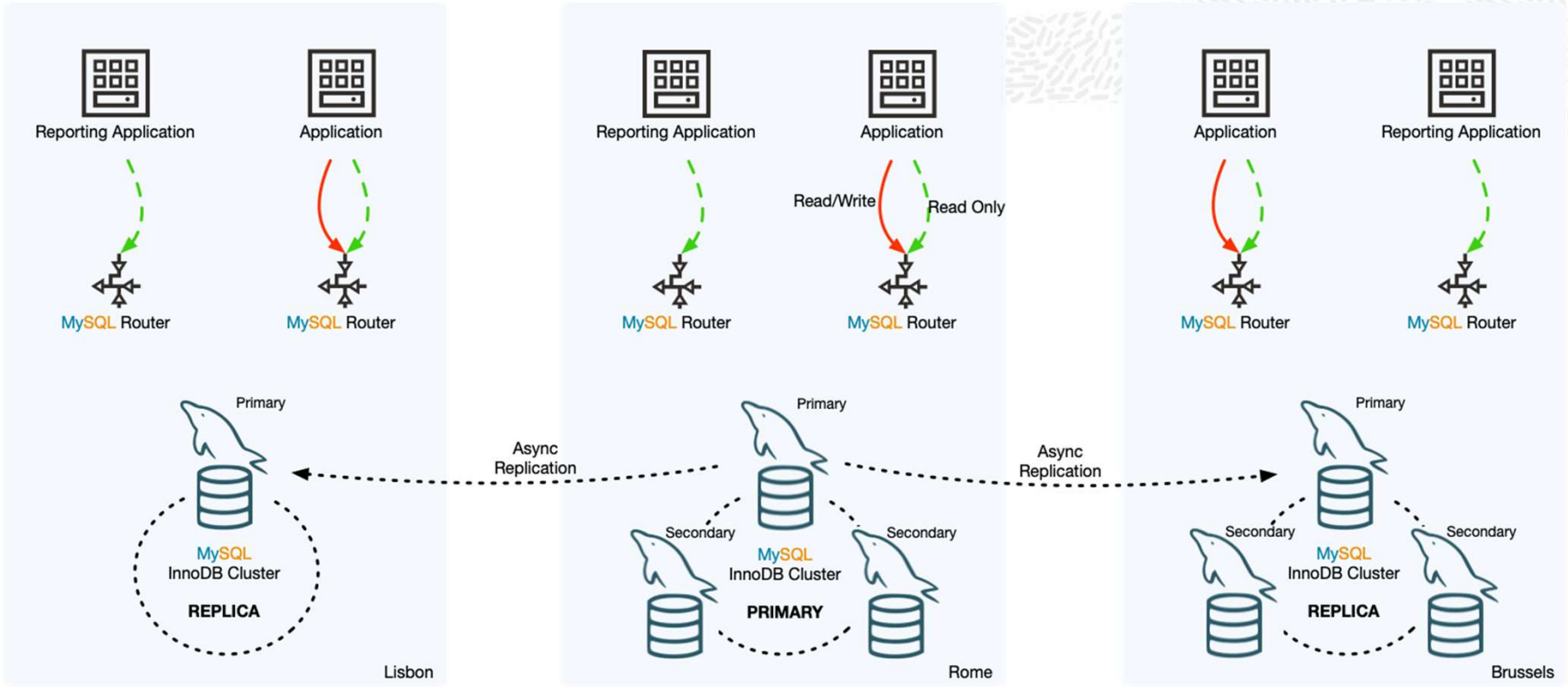
`mysqlsh, CLONE, …`

- 在线添加/删除节点/集群
- 路由器集成，拓扑结构发生变化时无需重新配置应用程序

MySQL InnoDB ClusterSet - 3个数据中心



MySQL InnoDB ClusterSet -并非每个集群都必须有 3 个节点



每个MySQL InnoDB集群副本可以有1-9个成员。

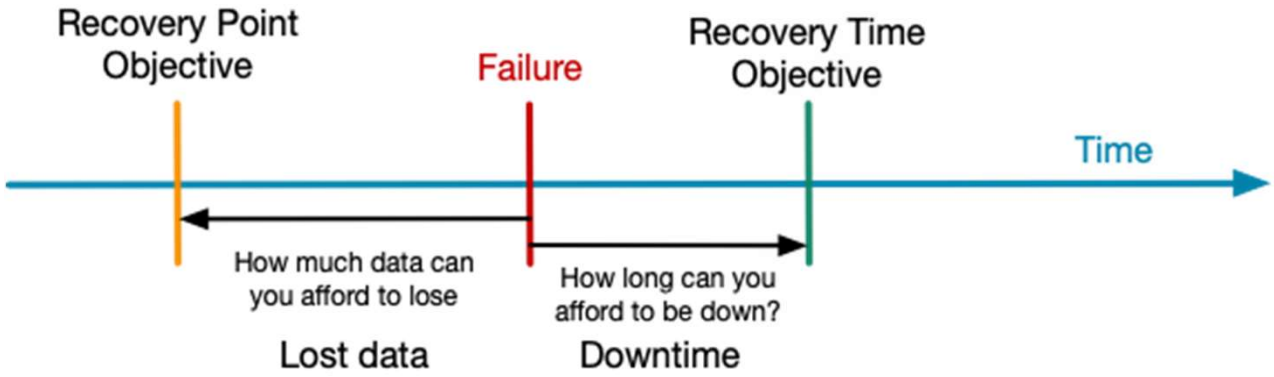
业务要求

业务需求

概念 - RTO & RPO
RTO：恢复时间目标
从单个故障中恢复需要多长时间
RPO：恢复点目标
发生故障时可能丢失多少数据

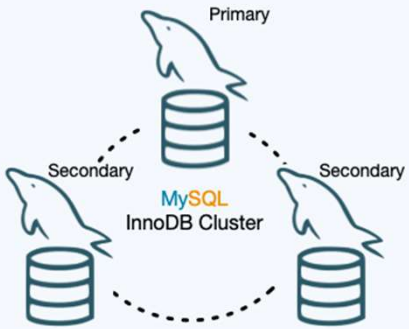
故障类型：

高可用性：单服务器故障，网络分区
灾难恢复：整个区域/网络故障
人为错误：个别表问题



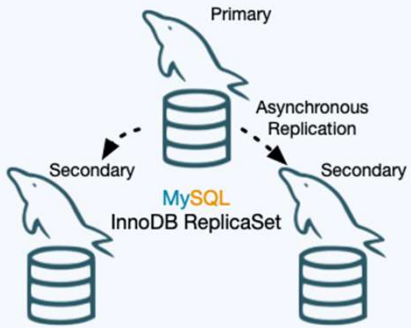
高可用性 - 单区域

MySQL InnoDB Cluster



- RPO = 0
- RTO = 秒级

MySQL InnoDB ReplicaSet



- RPO != 0
- RTO = 分钟 + (手动故障转移)

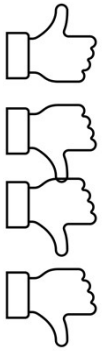

最佳写入性能


手动故障转移

灾难恢复 - 多区域

MySQL InnoDB Cluster

- RPO = 0
- RTO = 秒级

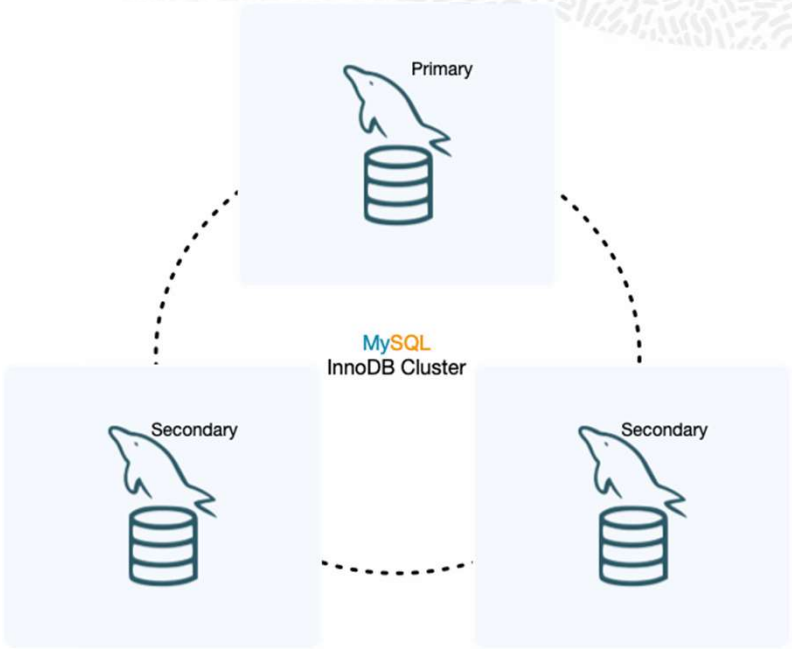


多区域 多主数据库

3DC

需要非常稳定的广域网

写入性能受DC之间的延迟影响



灾难恢复 - 多区域

MySQL InnoDB ClusterSet

- RPO !=0
- RTO =分钟+(手动故障转移)

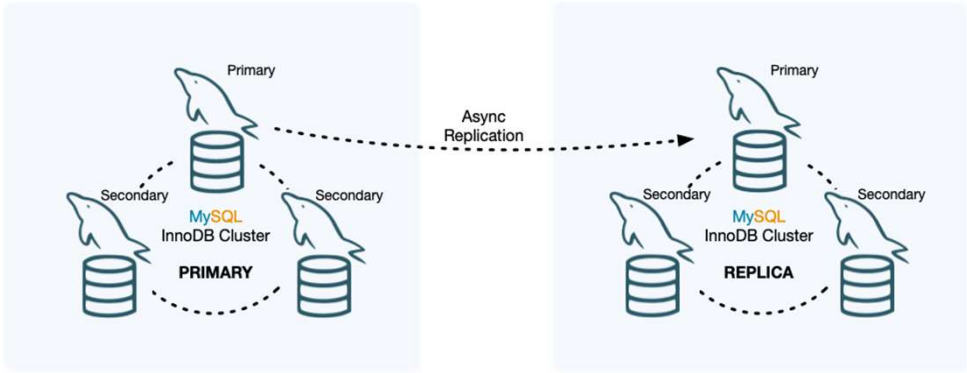
RPO =0 & RTO =区域内秒级(HA)

写入性能（无需同步到其他区域）

更高的 RTO：手动故障转移

RPO ！=0 （ 区域失败时）

MySQL InnoDB ClusterSet



复制增强功能

使 ClusterSet 成为可能的复制特性：

8.0.22：异步复制通道的自动连接故障转移
8.0.23：使用组复制的异步复制通道进行自动连接故障转移
8.0.24：使skip-slave-start成为全局、持久、只读的系统变量。
8.0.26：组复制成员操作（主成员上可配置super_read_only)
8.0.26：指定用于记录View_change_log_event的UUID
8.0.27：异步复制通道配置自动跟随主成员

MySQL InnoDB ClusterSet
配置命令

创建MySQL InnoDB Cluster

Start with setting up a regular MySQL InnoDB Cluster:

```
mysqlsh> \c root@localhost:3331
mysqlsh> \sql create schema sbtest;
mysqlsh> bru = dba.createCluster("BRU")
```

```
mysqlsh>  bru.addInstance('localhost:3332')
mysqlsh>  bru.addInstance('localhost:3333')
mysqlsh>  bru.status()
```


创建 ClusterSet

```
mysqlsh> clusterset = bru.createClusterSet('clusterset')
```

```
A new ClusterSet will be created based on the Cluster 'BRU'.
- Validating Cluster 'BRU' for ClusterSet compliance.
- Creating InnoDB ClusterSet 'clusterset' on 'BRU'...
- Updating metadata...

ClusterSet successfully created. Use ClusterSet.createReplicaCluster() to add Replica Clusters to it.
<ClusterSet:cluster>
```

检查ClusterSet状态

```
mysqlsh> clusterset.status()
```

```
{
    "clusters": {
        "BRU": {
            "clusterRole": "PRIMARY",
            "globalStatus": "OK",
            "primary": "127.0.0.1:3331"
        }
    },
    "domainName": "clusterset",
    "globalPrimaryInstance": "127.0.0.1:3331",
    "primaryCluster": "BRU",
    "status": "HEALTHY",
    "statusText": "All Clusters available."
}
```


添加副本集群

- Supports incremental recovery (binlog) & full recovery (CLONE)

```
mysqlsh> lis = clusterset.createReplicaCluster('localhost:4441', 'LIS')
mysqlsh> lis.addInstance('localhost:4442')
mysqlsh> lis.addInstance('localhost:4443')
```

检查ClusterSet状态

```
mysqlsh> clusterset.status()
mysqlsh> bru.status()
mysqlsh> lis.status()
```


```
{
    "clusters": {
        "BRU": {
            "clusterRole": "PRIMARY",
            "globalStatus": "OK",
            "primary": "127.0.0.1:3331"
        },
        "LIS": {
            "clusterRole": "REPLICA",
            "clusterSetReplicationStatus": "OK",
            "globalStatus": "OK"
        }
    },
    "domainName": "clusterset",
    "globalPrimaryInstance": "127.0.0.1:3331",
    "primaryCluster": "BRU",
    "status": "HEALTHY",
    "statusText": "All Clusters available."
}
```


Or, to get everything in one command:

```
mysqlsh> clusterset.status({extended:1})
```

添加第二个副本集群

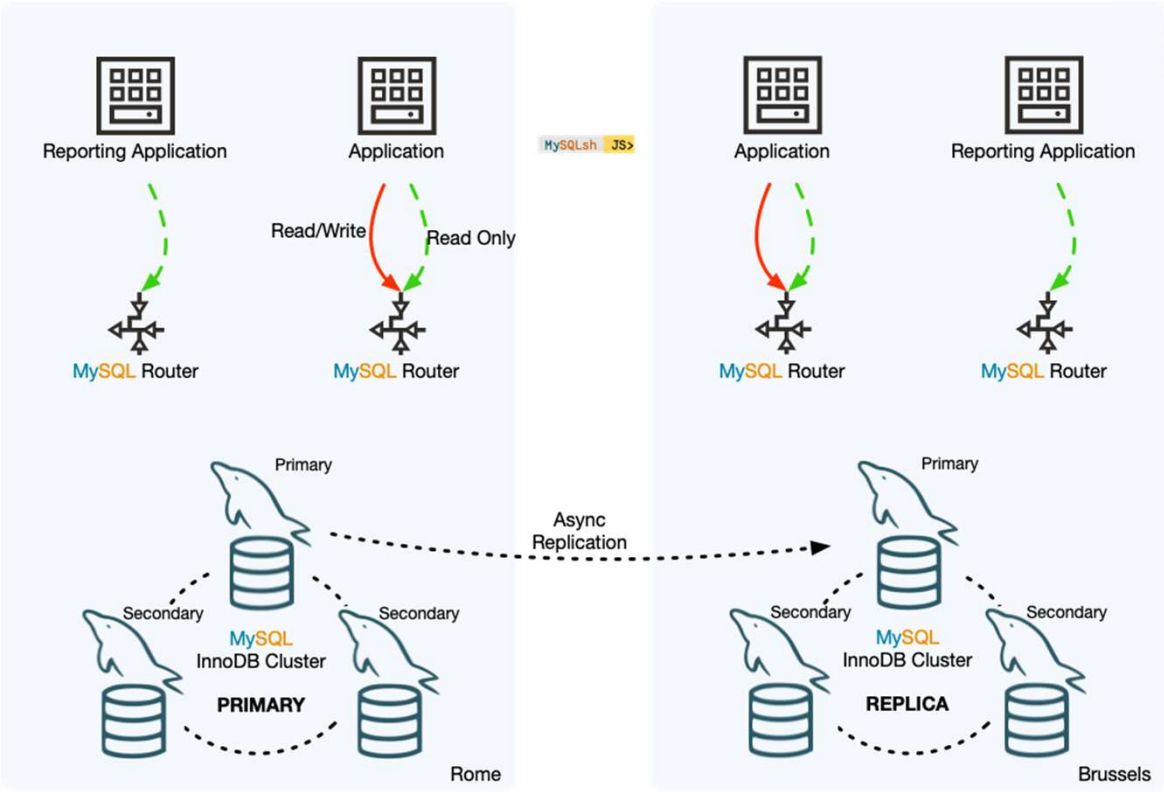
```
mysqlsh> rom = clusterset.createReplicaCluster(
                           'localhost:5551',
                           'ROM')
mysqlsh> rom.addInstance('localhost:5552')
mysqlsh> rom.addInstance('localhost:5553')
```

```
mysqlsh> rom.status()
mysqlsh> clusterset.status()
```


```
{
    "clusters": {
        "ROM": {
            "clusterRole": "REPLICA",
            "clusterSetReplicationStatus": "OK",
            "globalStatus": "OK"
        },
        "BRU": {
            "clusterRole": "PRIMARY",
            "globalStatus": "OK",
            "primary": "127.0.0.1:3331"
        },
        "LIS": {
            "clusterRole": "REPLICA",
            "clusterSetReplicationStatus": "OK",
            "globalStatus": "OK"
        }
    },
    "domainName": "clusterset",
    "globalPrimaryInstance": "127.0.0.1:3331",
    "primaryCluster": "BRU",
    "status": "HEALTHY",
    "statusText": "All Clusters available."
}
```

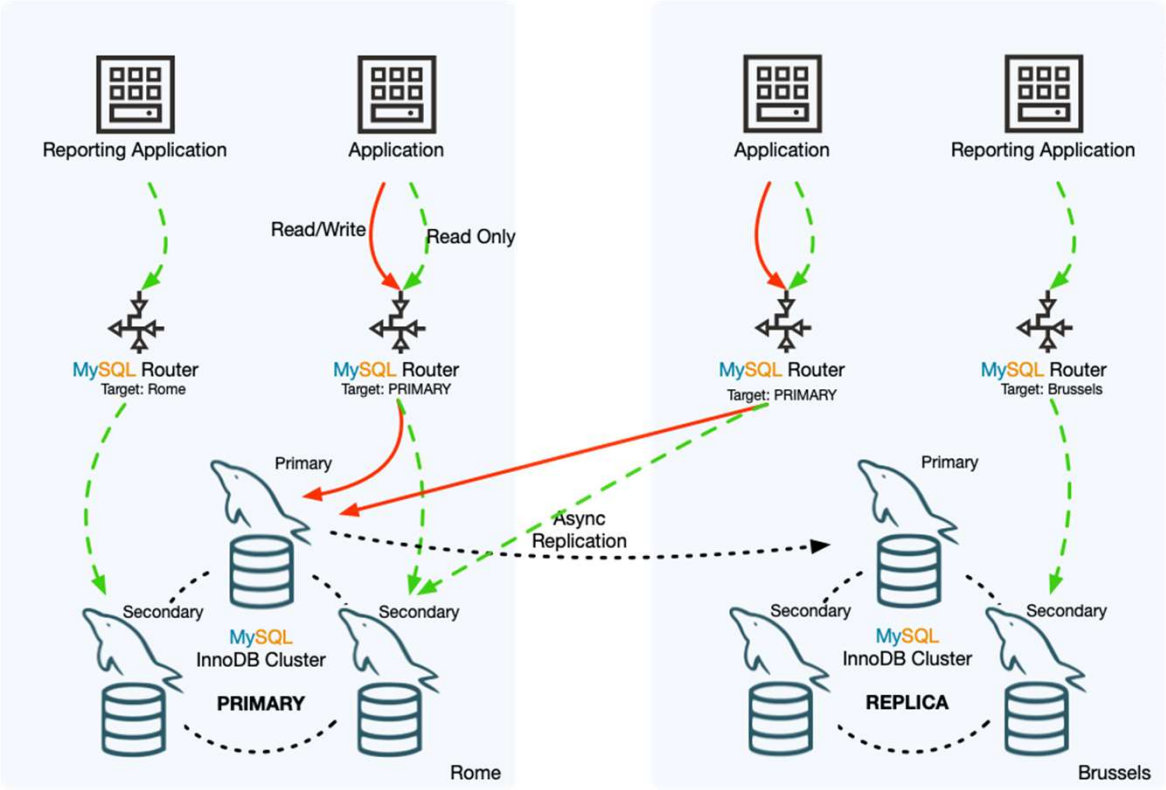
路由器集成

路由器集成



将应用程序配置为连接到本地 MySQL 路由器以连接到 ClusterSet

路由器集成



路由器目标模式：

1.跟从主集群

写入和读取到主集群

2.连接到配置的目标集群

当目标集群不是主集群时：仅打开读取流量

写入将被拒绝

当目标集群为主集群时：写入端口打开

特性：

可配置每个路由器实例

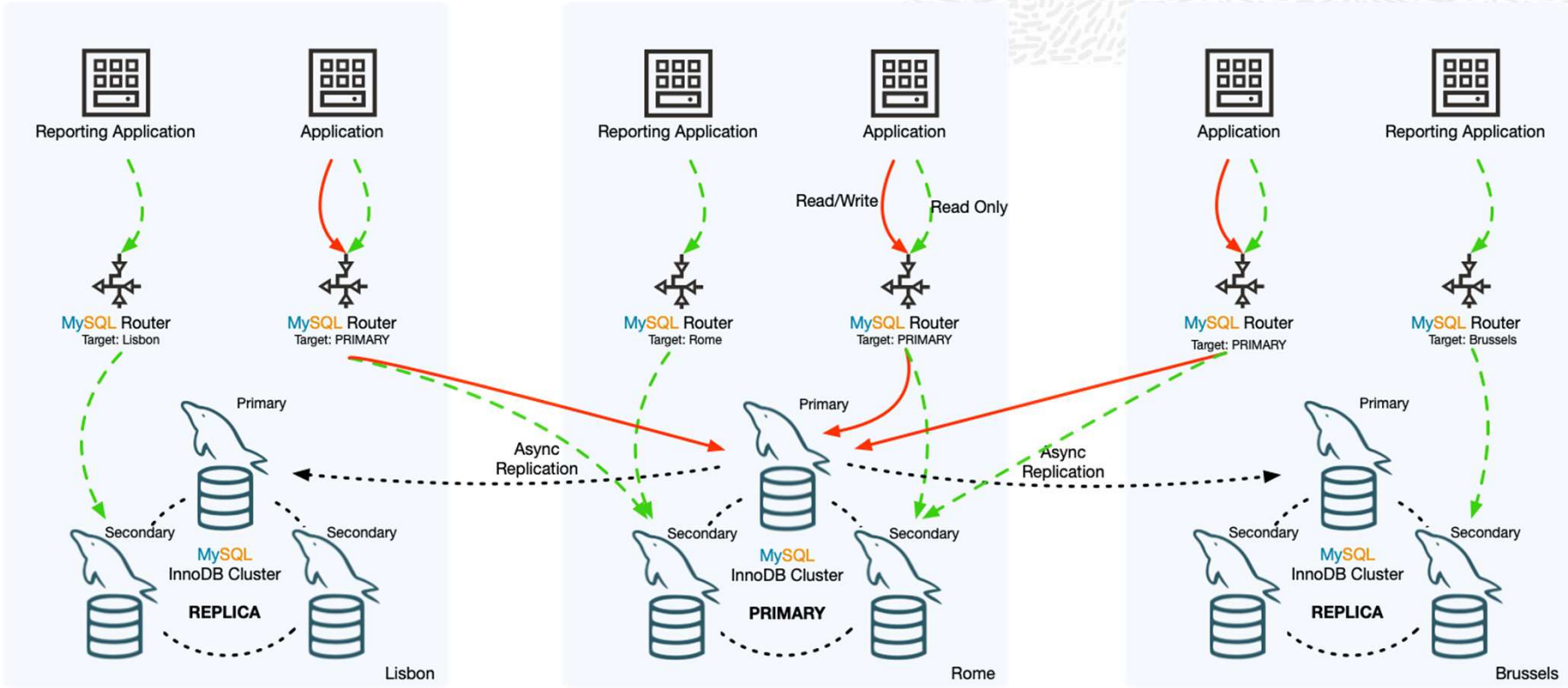
可以在mysqlsh中在线更改配置

部署 2 种类型的路由器：

- 以“主”为目标以将写入操作发送到“主数据库”
- 定义目标集群以便本地读取

INVALIDATED无效集群仍可用于只读（可自定义）

路由器集成- 3DC



MySQL InnoDB ClusterSet
路由器集成命令

引导路由器（Bootstrap Router）

- 和MySQL InnoDB Cluster & MySQL InnoDB ReplicaSet一样

```
$ sudo mysqlrouter --bootstrap root@localhost:3331 --user=mysqlrouter
$ sudo systemctl start mysqlrouter
$ sudo tail -f /var/log/mysqlrouter/mysqlrouter.log
```


更改路由器配置选项

更改target_cluster：

```
mysqlsh> clusterset.setRoutingOption('instance-....com::system', 'target_cluster', 'ROM')
mysqlsh> clusterset.setRoutingOption('instance-....com::system', 'target_cluster', 'BRU')
mysqlsh> clusterset.setRoutingOption('instance-....com::system', 'target_cluster', 'LIS')
mysqlsh> clusterset.setRoutingOption('instance-....com::system', 'target_cluster', 'primary')
```

target_cluster也可以在 mysqlrouter --bootstrap 期间进行配置
--conf-target-cluster 或者 --conf-target-cluster-by-name

更改invalidated_cluster_policy：
当target_cluster集群失效时，它是否仍应接受读取还是丢弃?

```
mysqlsh> clusterset.setRoutingOption('instance-....com::system', 'invalidated_cluster_policy', 'accept_ro')
mysqlsh> clusterset.setRoutingOption('instance-....com::system', 'invalidated_cluster_policy', 'drop_all')
```

更改路由器的默认设置：

```
mysqlsh> clusterset.setRoutingOption('target_cluster', 'LIS')
```


MySQL InnoDB ClusterSet
管理命令

更改主集群中的主成员

```
mysqlsh> bru.setPrimaryInstance('localhost:3332')
```

```
Setting instance 'localhost:3332' as the primary instance of cluster 'BRU'...

Instance '127.0.0.1:3331' was switched from PRIMARY to SECONDARY.
Instance '127.0.0.1:3332' was switched from SECONDARY to PRIMARY.
Instance '127.0.0.1:3333' remains SECONDARY.

WARNING: The cluster internal session is not the primary member anymore. For cluster management operations
please obtain a fresh cluster handle using dba.getCluster().

The instance 'localhost:3332' was successfully elected as primary.
```


更改副本集群中的主成员

```
mysqlsh> lis.setPrimaryInstance('localhost:4442')

Setting instance 'localhost:4442' as the primary instance of cluster 'LIS'...

Instance '127.0.0.1:4442' was switched from SECONDARY to PRIMARY.
Instance '127.0.0.1:4443' remains SECONDARY.
Instance '127.0.0.1:4441' was switched from PRIMARY to SECONDARY.

WARNING: The cluster internal session is not the primary member anymore. For cluster management operations
please obtain a fresh cluster handle using dba.getCluster().

The instance 'localhost:4442' was successfully elected as primary.
```


Switchover - 更改主集群- setPrimaryCluster()

```
 mysqlsh> clusterset.setPrimaryCluster('LIS')

Switching the primary cluster of the clusterset to 'LIS'
- Verifying clusterset status
-- Checking cluster BRU -  Cluster 'BRU' is available
-- Checking cluster ROM -  Cluster 'ROM' is available
-- Checking cluster LIS -  Cluster 'LIS' is available

- Refreshing replication account of demoted cluster
- Synchronizing transaction backlog at 127.0.0.1:4442

- Updating metadata

- Updating topology
-- Changing replication source of 127.0.0.1:3331 to 127.0.0.1:4442
-- Changing replication source of 127.0.0.1:3333 to 127.0.0.1:4442
-- Changing replication source of 127.0.0.1:3332 to 127.0.0.1:4442
- Acquiring locks in replicaset instances
-- Pre-synchronizing SECONDARIES
-- Acquiring global lock at PRIMARY & SECONDARIES

- Synchronizing remaining transactions at promoted primary

- Updating replica clusters
-- Changing replication source of 127.0.0.1:5552 to 127.0.0.1:4442
-- Changing replication source of 127.0.0.1:5553 to 127.0.0.1:4442
-- Changing replication source of 127.0.0.1:5551 to 127.0.0.1:4442
Cluster 'LIS' was promoted to PRIMARY of the clusterset. The PRIMARY instance is '127.0.0.1:4442'
```


Failover到另一个集群

```
mysqlsh> \c root@localhost:3331
mysqlsh> clusterset=dba.getClusterSet()
mysqlsh> clusterset.forcePrimaryCluster('BRU')
```

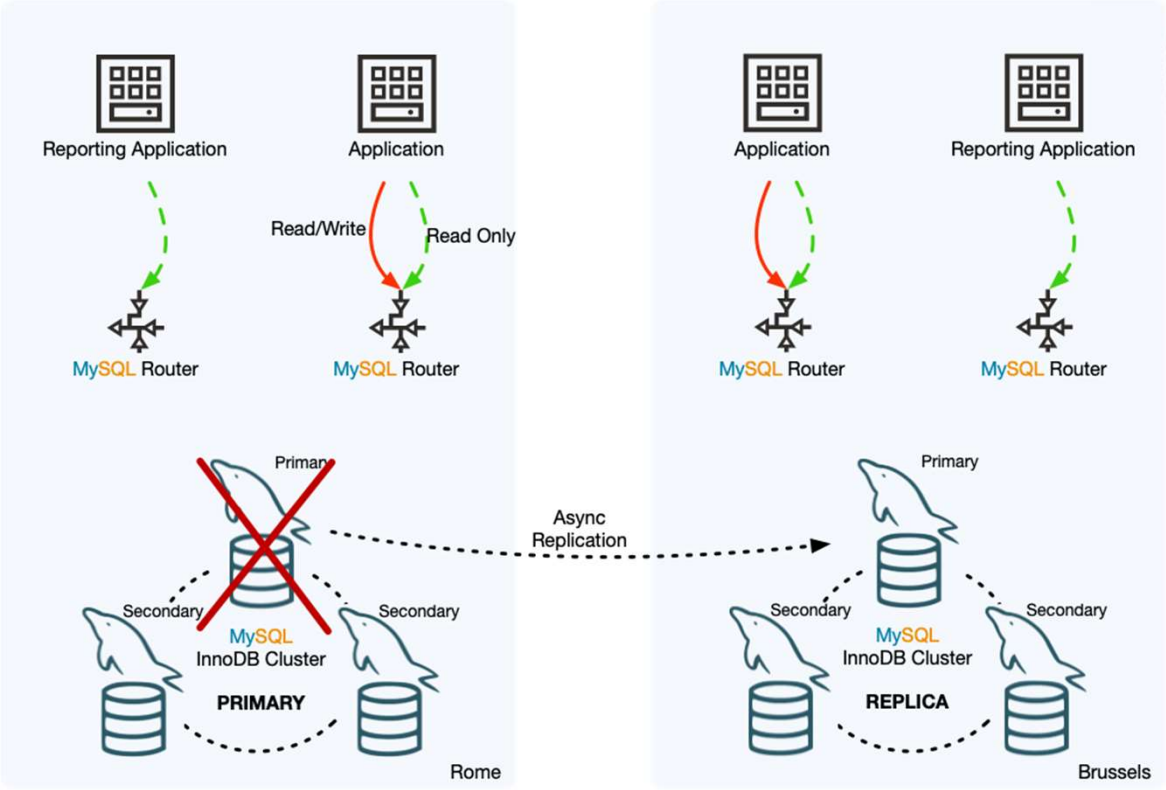
```
Failing-over primary cluster of the clusterset to 'BRU'
- Verifying primary cluster status
None of the instances of the PRIMARY cluster 'LIS' could be reached.
- Verifying clusterset status
-- Checking cluster BRU
  Cluster 'BRU' is available
-- Checking cluster ROM
  Cluster 'ROM' is available
-- Checking whether target cluster has the most recent GTID set
- Promoting cluster 'BRU'
- Updating metadata

-- Changing replication source of 127.0.0.1:5552 to 127.0.0.1:3331
-- Changing replication source of 127.0.0.1:5553 to 127.0.0.1:3331
-- Changing replication source of 127.0.0.1:5551 to 127.0.0.1:3331
PRIMARY cluster failed-over to 'BRU'. The PRIMARY instance is '127.0.0.1:3331'
Former PRIMARY cluster was INVALIDATED, transactions that were not yet replicated may be lost.
```


从ClusterSet中删除集群

```
mysqlsh> clusterset.removeCluster('LIS')
```


ClusterSet场景

主集群主成员崩溃/分区

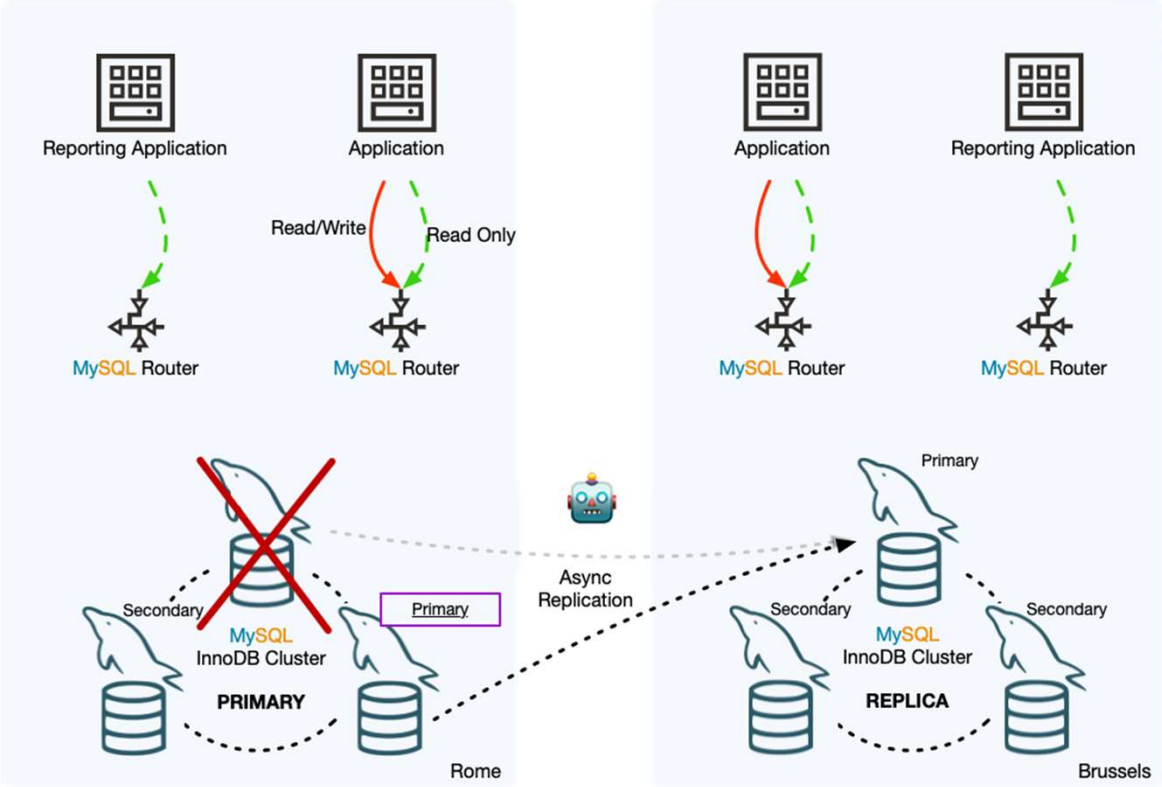


- 当集群中有新当选的主成员时
- 适用于主集群和副本集群中的故障

自动处理 InnoDB 集群状态更改

- 异步复制在主更改后自动重新配置

主集群 主成员崩溃/分区 - 自动!

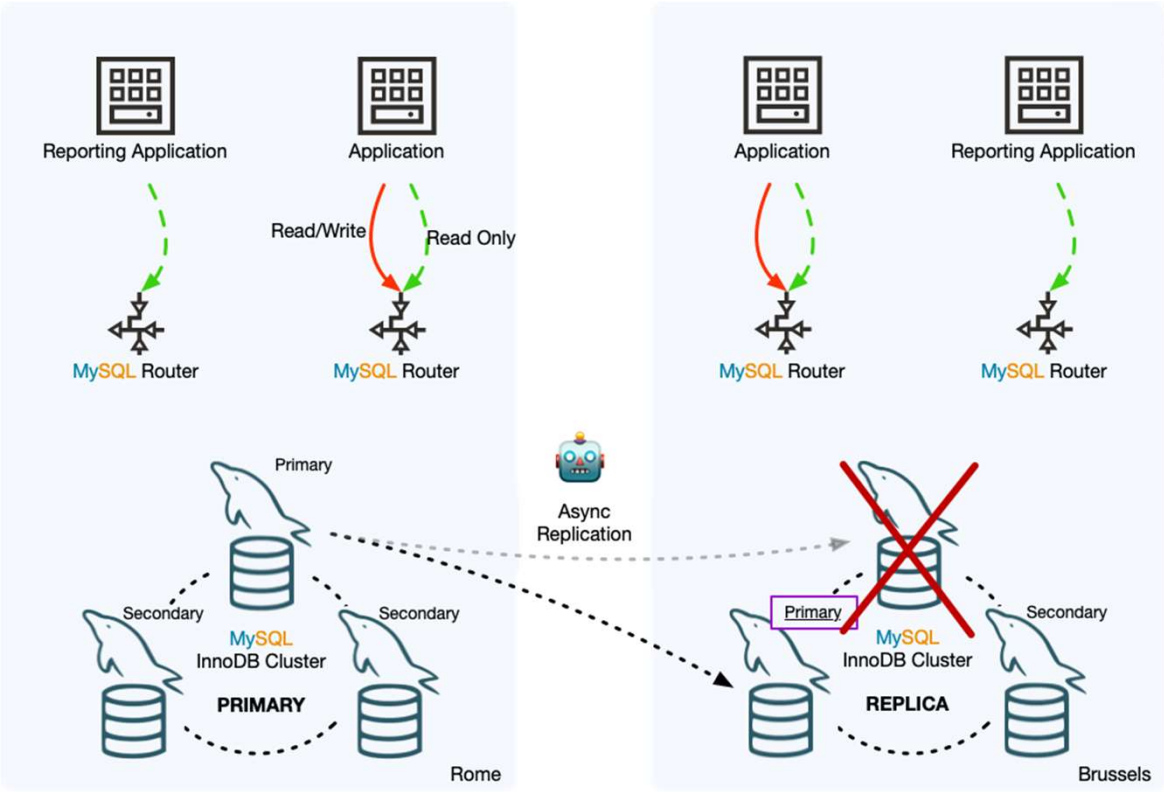


- 当集群中有新当选的主成员时
- 适用于主集群和副本集群中的故障

自动处理 InnoDB 集群状态更改

- 异步复制在主更改后自动重新配置

副本集群主成员崩溃/分区 - 自动!

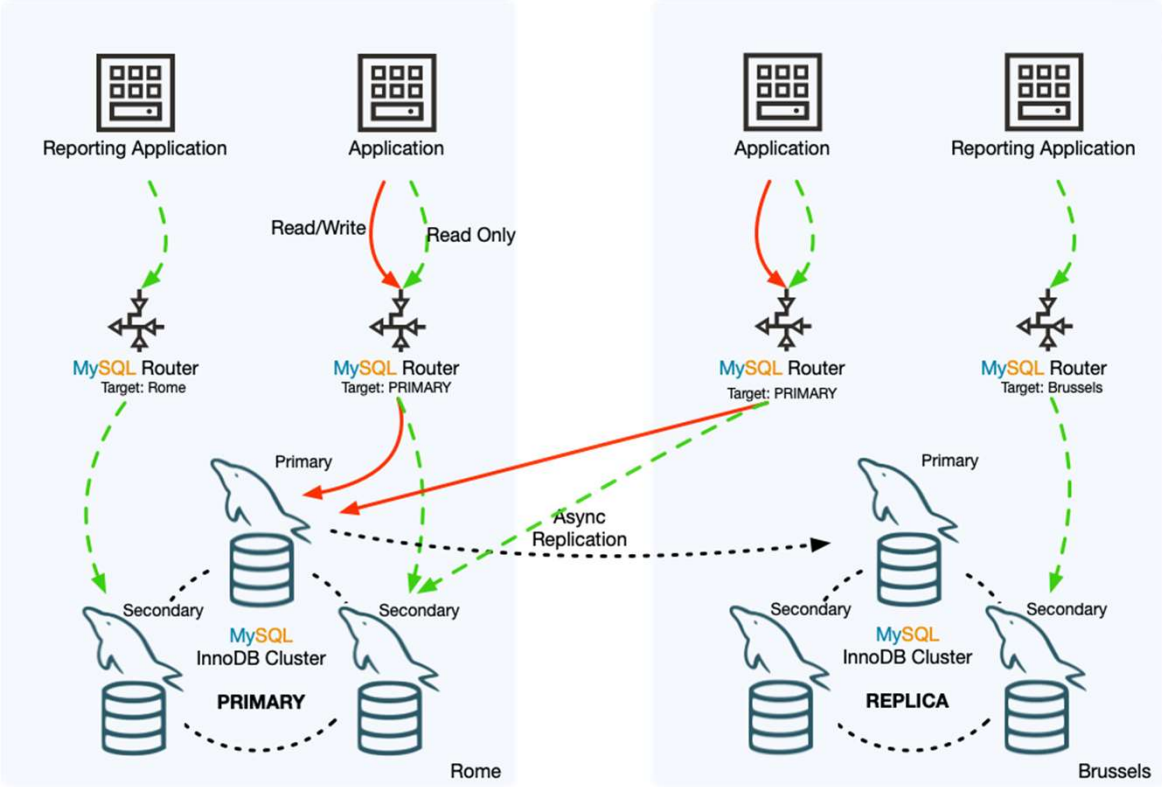


- 当集群中有新当选的主成员时
- 适用于主集群和副本集群中的故障

自动处理 InnoDB 集群状态更改

- 异步复制在主更改后自动重新配置

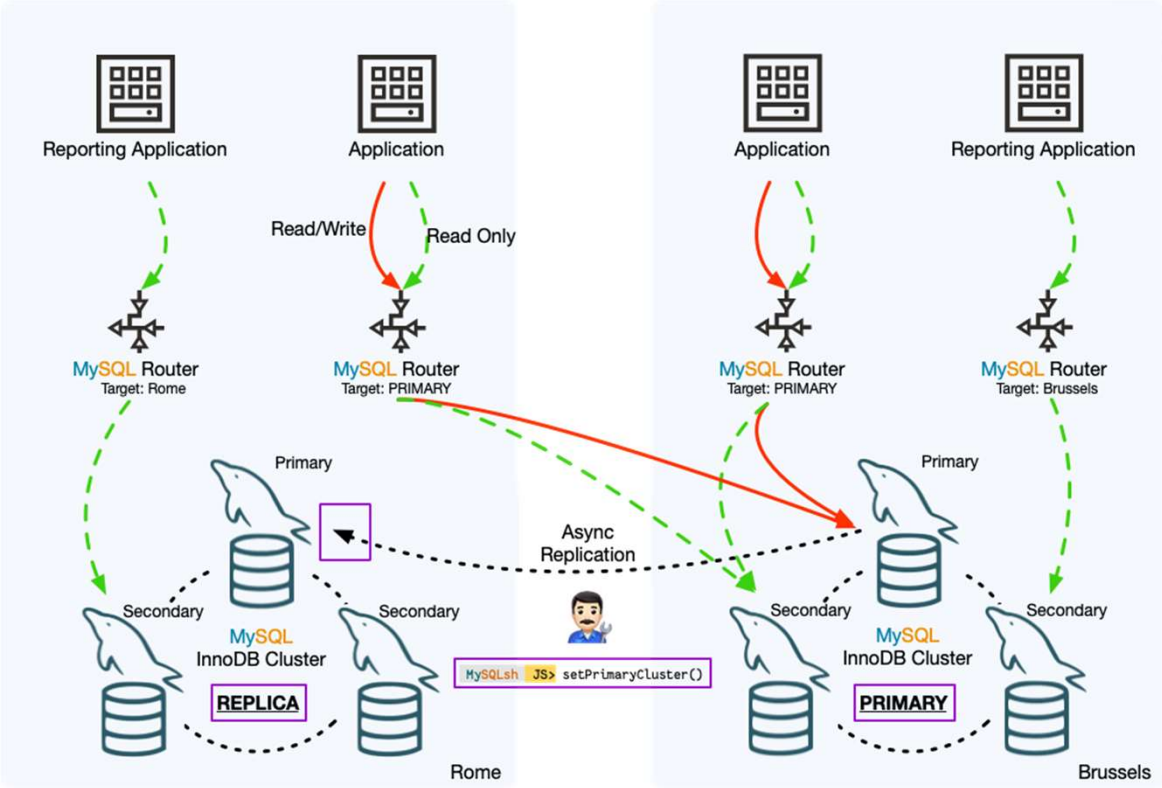
改主 - 更改正常系统上的主集群



切换

- 一个可以完成所有操作的命令：setPrimaryCluster（）
- 集群之间的异步复制通道会自动重新配置
- 保证一致性
- 如果需要，所有路由器将立即重定向（取决于目标模式）

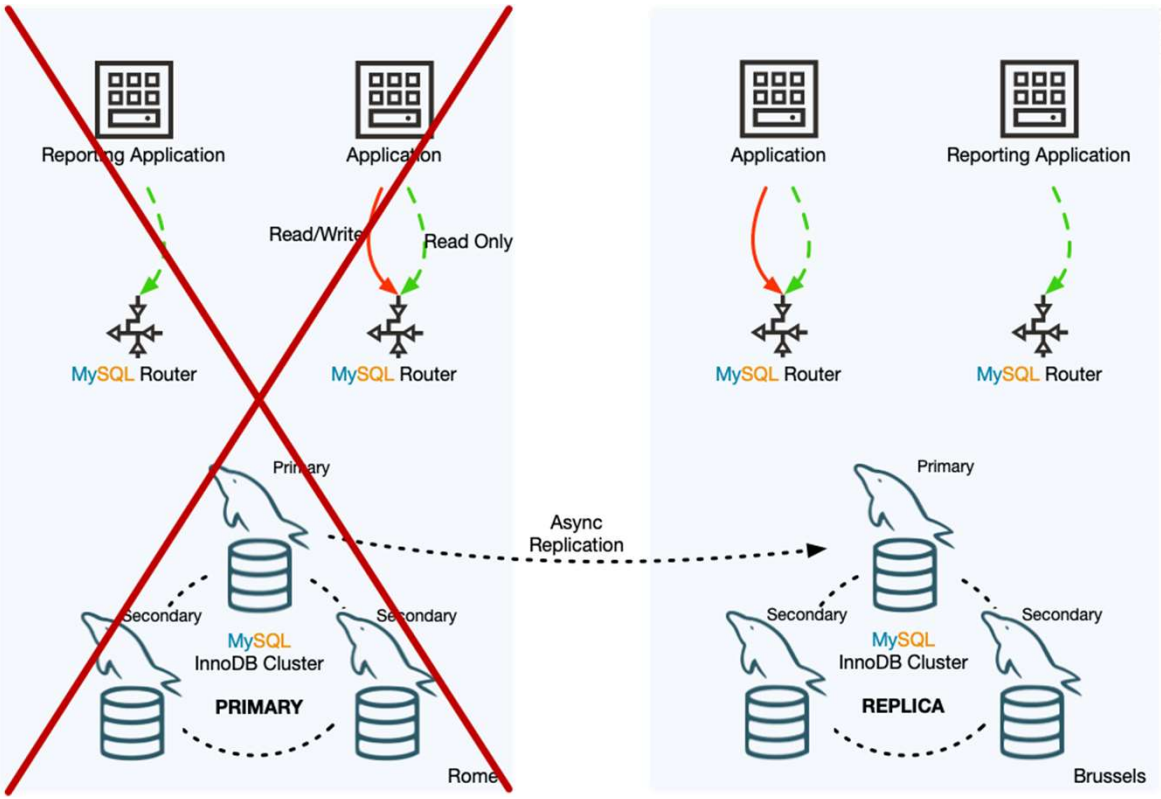
改主 - setPrimaryCluster()



切换

- 一个可以完成所有操作的命令：setPrimaryCluster（）
- 集群之间的异步复制通道会自动重新配置
- 保证一致性
- 如果需要，所有路由器将立即重定向（取决于目标模式）

数据中心崩溃/分区



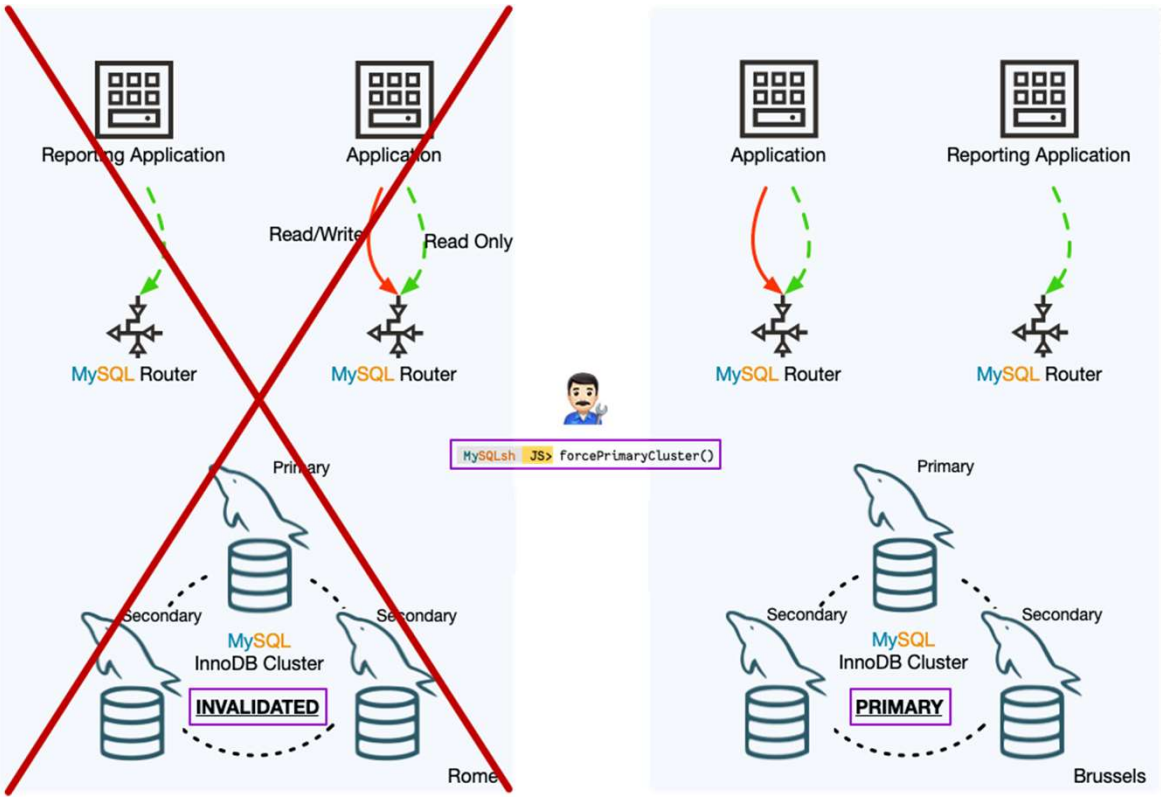
故障转移到另一个集群

- 一个命令使主集群失效并提升新的主集群：forcePrimaryCluster（）
- 将重新配置其它副本集群复制

脑裂警告

- 无法连接到其他集群的本地路由器将无法了解新拓扑
- 如果数据中心被网络分区，它将继续作为主数据库运行

数据中心崩溃/分区- forcePrimaryCluster()



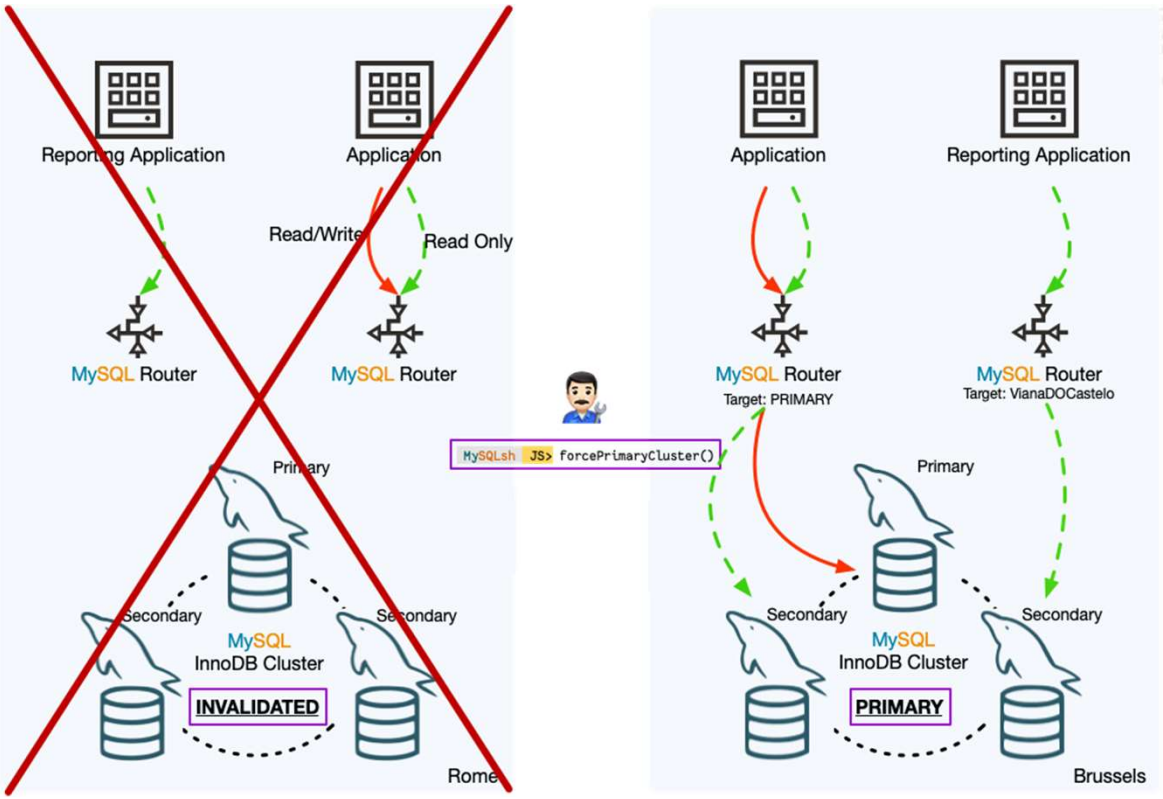
故障转移到另一个集群

- 一个命令使主集群失效并提升新的主集群：forcePrimaryCluster（）
- 将重新配置其它副本集群复制

脑裂警告

- 无法连接到其他集群的本地路由器将无法了解新拓扑
- 如果数据中心被网络分区，它将继续作为主数据库运行

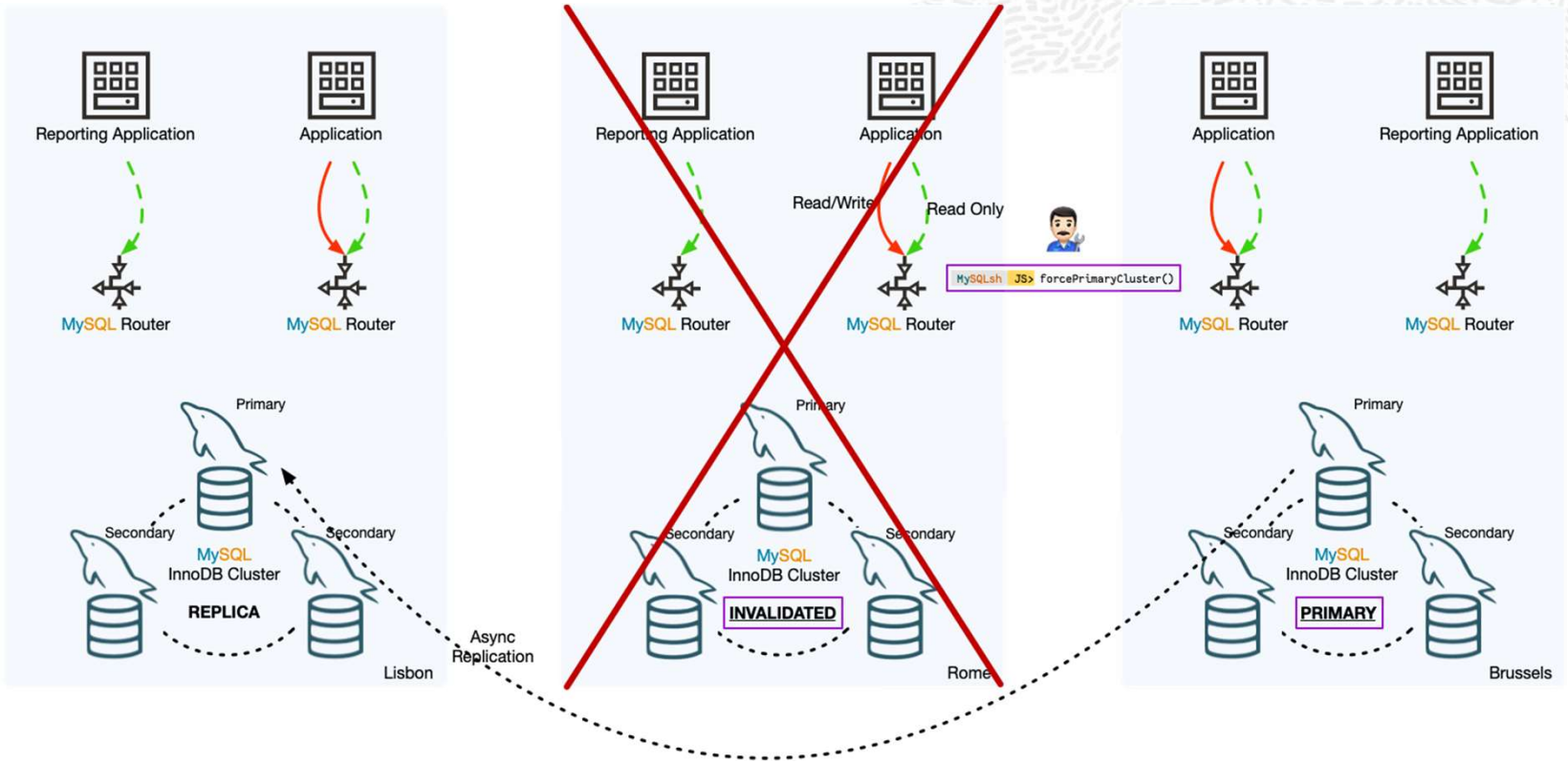
数据中心崩溃/分区 - 路由器集成



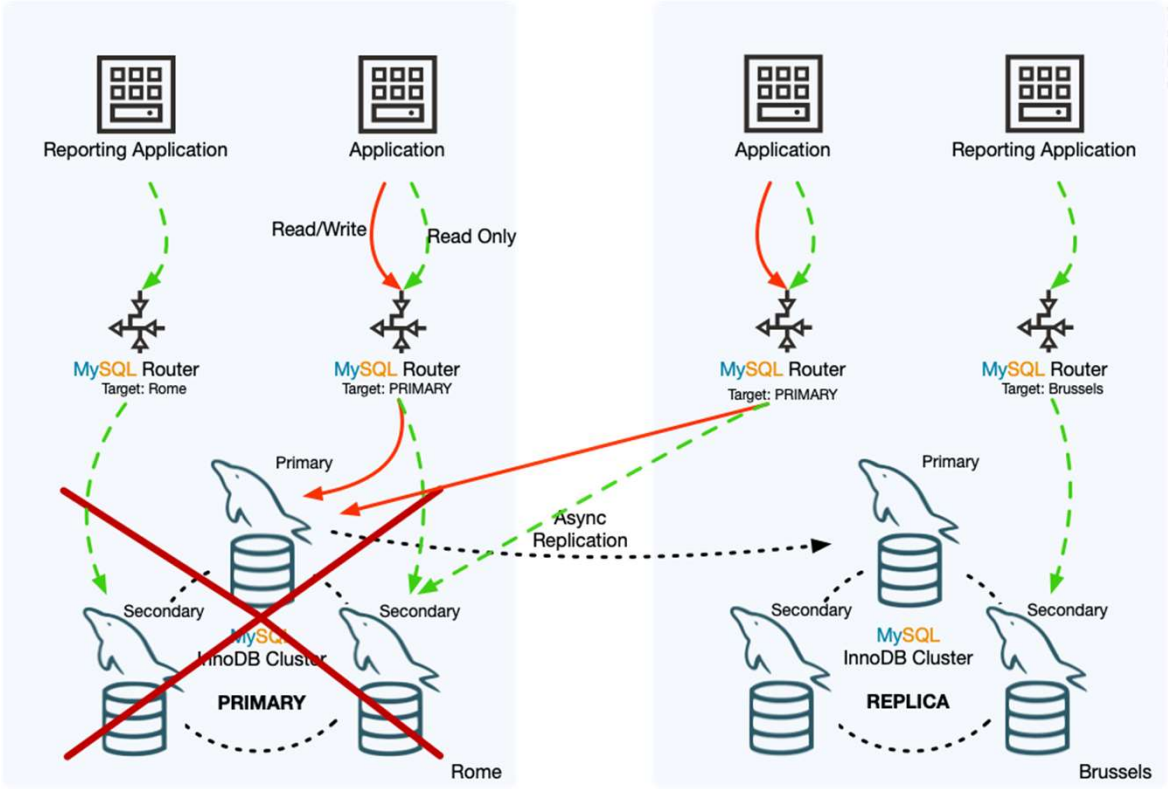
路由器集成

- 路由器将了解新的拓扑结构并重定向流量
- 恢复的路由器将了解新的拓扑并放弃旧的主集群（例如，故障DC重新上线）

数据中心崩溃/分区 - 支持多个副本集群



组复制崩溃/分区



路由器集成

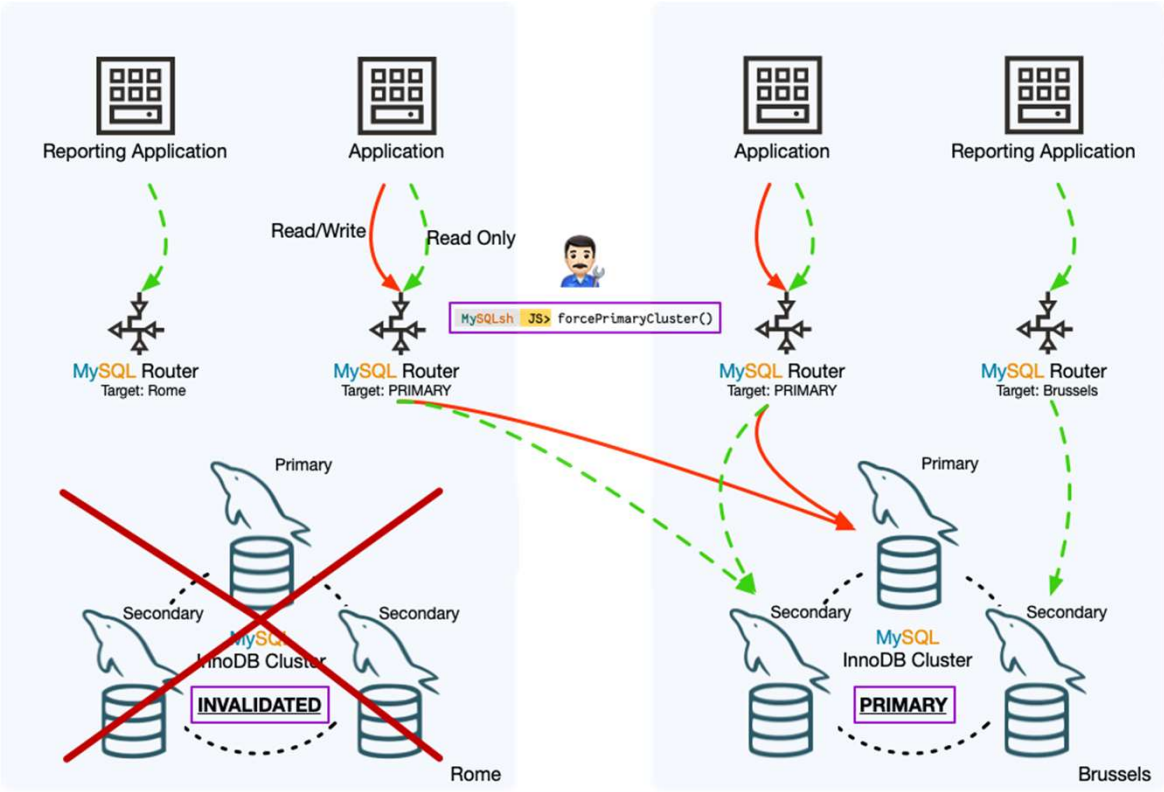
当 GR 处于脱机状态时：

- 网络分区
- 无仲裁
- 整个集群丢失（例如断电）

故障转移到另一个集群

- 一个命令使主集群失效并提升新的主集群：forcePrimaryCluster（）
- 路由器实例将跟从主（取决于目标模式）

组复制崩溃/分区- forcePrimaryCluster() & Router



路由器集成

当 GR 处于脱机状态时：

- 网络分区
- 无仲裁
- 整个集群丢失（例如断电）

故障转移到另一个集群

- 一个命令使主集群失效并提升新的主集群：forcePrimaryCluster（）
- 路由器实例将跟从主（取决于目标模式）

MySQL InnoDB ClusterSet -限制

- 需要服务器、路由器和Shell版本 8.0.27 或更高
- InnoDB 集群仅支持单主模式
- 集群之间使用异步复制，而不是半同步（如果要求 RPO=0，则使用跨区域分布的单个集群）


ORACLE
MySQL
MySQL InnoDB ClusterSet