A Deployment of MySQL-HA-DR

fanjunliang yuchangxi

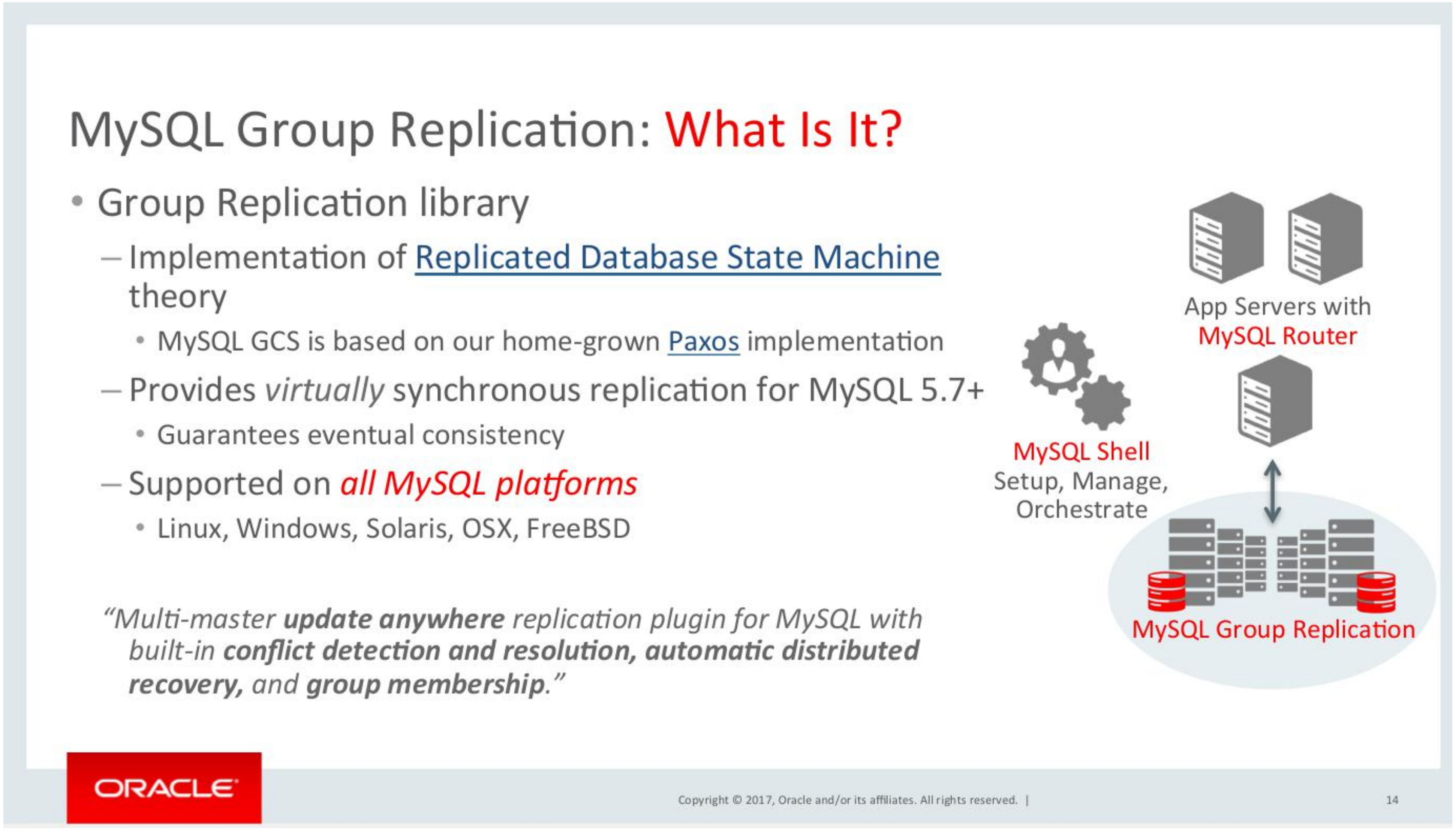
Agenda

- MySQL HA
    - Group-replication
    - MySQL-Router
- MySQL DR
    - Daul-Data-Centers solutions
    - Solution for MySQL 5.7
    - Solution for MySQL 8.0
- MySQL-Operator
    - Persistence
    - Trouble Shooting
- Reference

Agenda

- MySQL HA
  - Group-replication
  - MySQL-Router
- MySQL DR
  - Daul-Data-Centers solutions
  - Solution for MySQL 5.7
  - Solution for MySQL 8.0
- MySQL-Operator
  - Persistence
  - Trouble Shooting
- Reference

MySQL-HA Group-replication



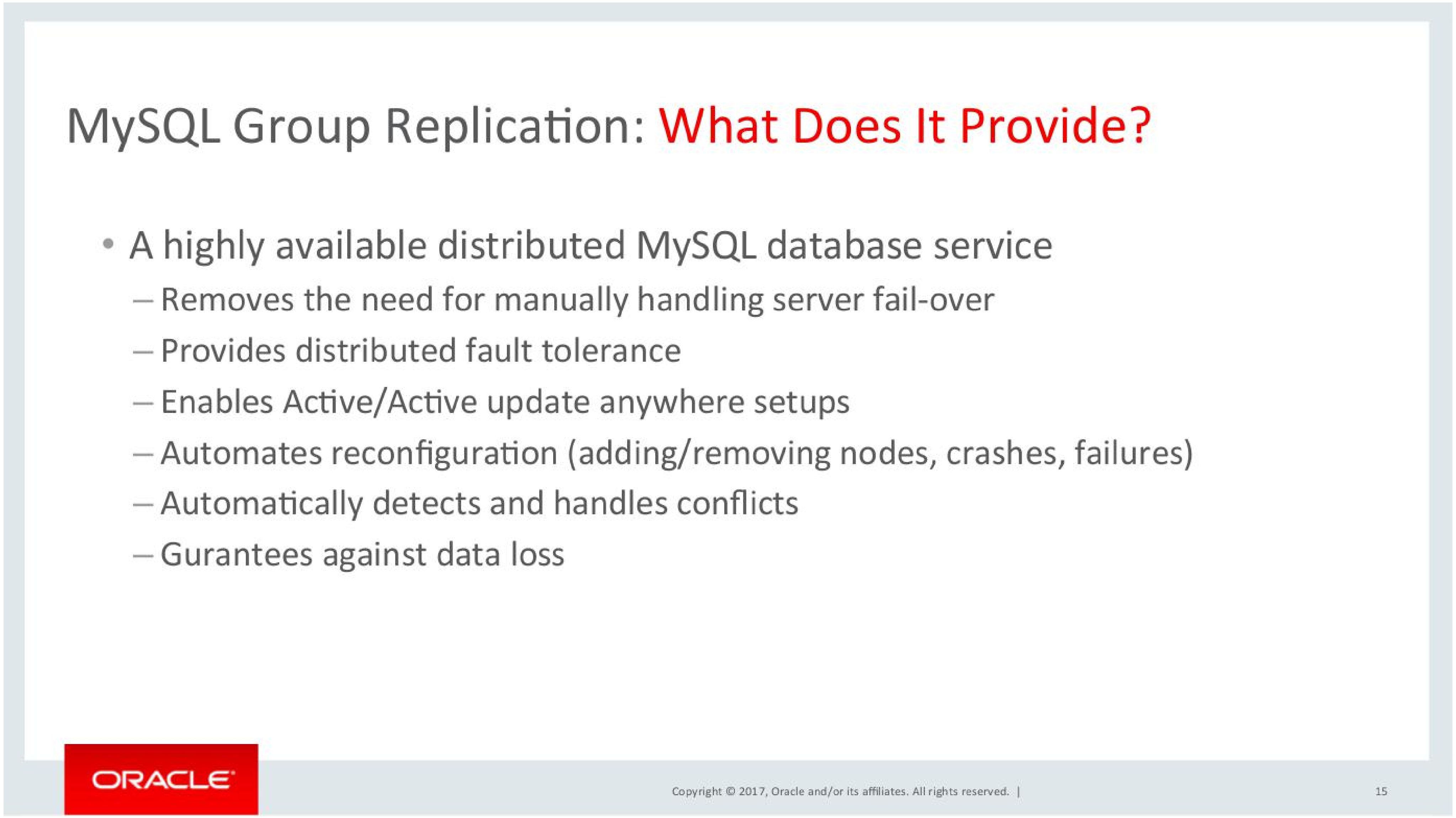
MySQL Group Replication: What Is It?

- Group Replication library
  - Implementation of Replicated Database State Machine theory
    - MySQL GCS is based on our home-grown Paxos implementation
  - Provides *virtually* synchronous replication for MySQL 5.7+
    - Guarantees eventual consistency
  - Supported on *all MySQL platforms*
    - Linux, Windows, Solaris, OSX, FreeBSD

*"Multi-master **update anywhere** replication plugin for MySQL with built-in **conflict detection and resolution, automatic distributed recovery,** and **group membership**."*

ORACLE

MySQL-HA Group-replication



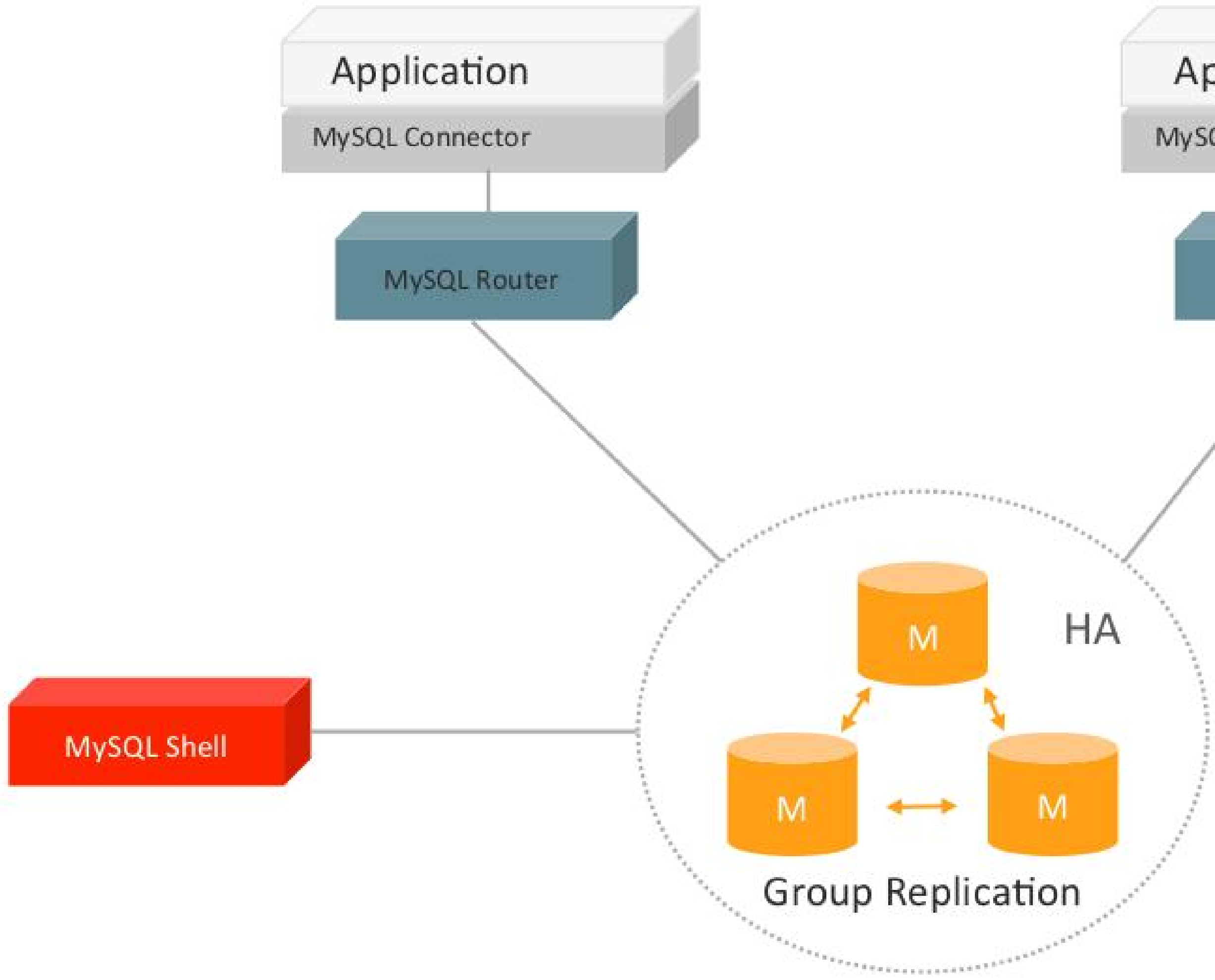
MySQL Group Replication: What Does It Provide?

- A highly available distributed MySQL database service
  - Removes the need for manually handling server fail-over
  - Provides distributed fault tolerance
  - Enables Active/Active update anywhere setups
  - Automates reconfiguration (adding/removing nodes, crashes, failures)
  - Automatically detects and handles conflicts
  - Gurantees against data loss

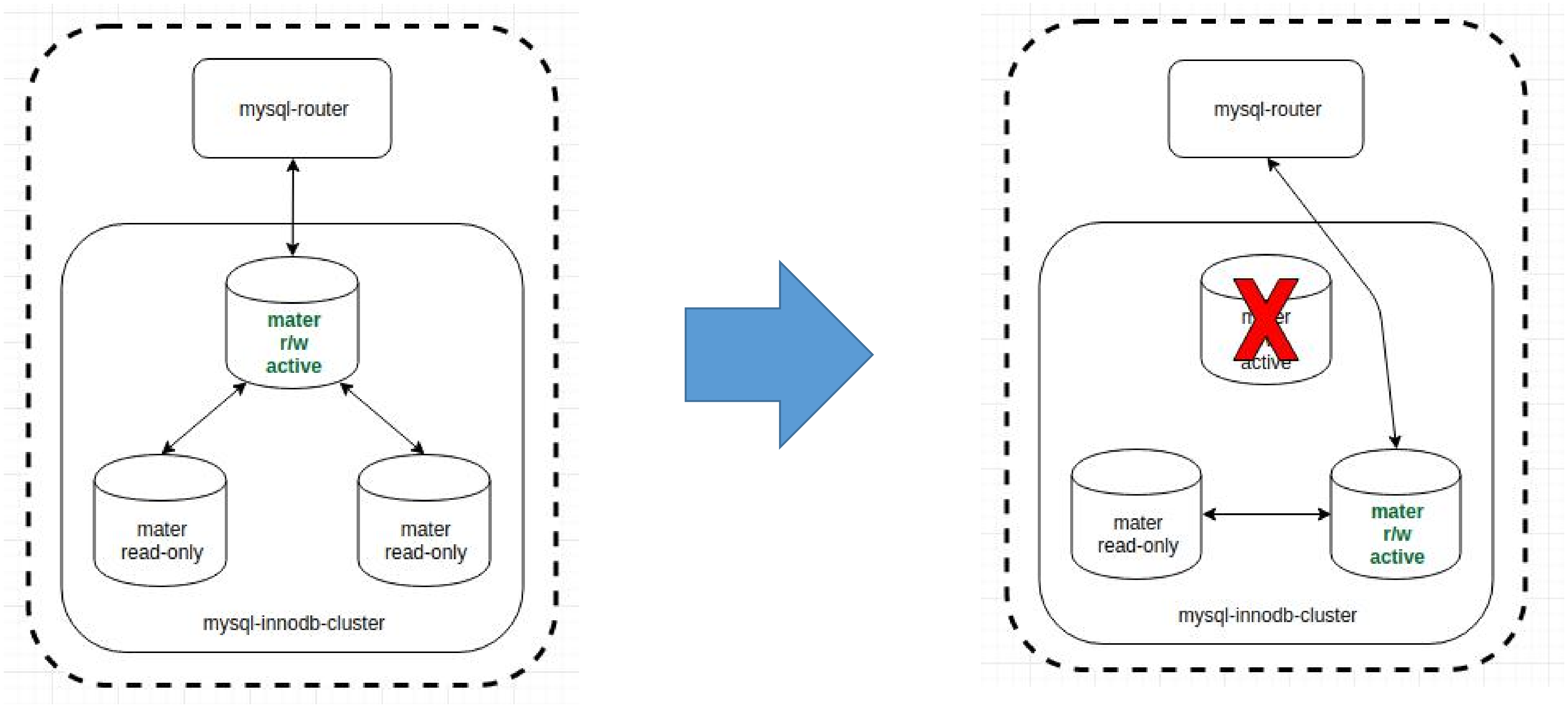
ORACLE

MySQL-HA MySQL-Router

- MySQL Router is part of InnoDB cluster, and is lightweight middleware that provides transparent routing between your application and back-end 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 back-end MySQL Servers.



MySQL-HA MySQL-Router



Agenda

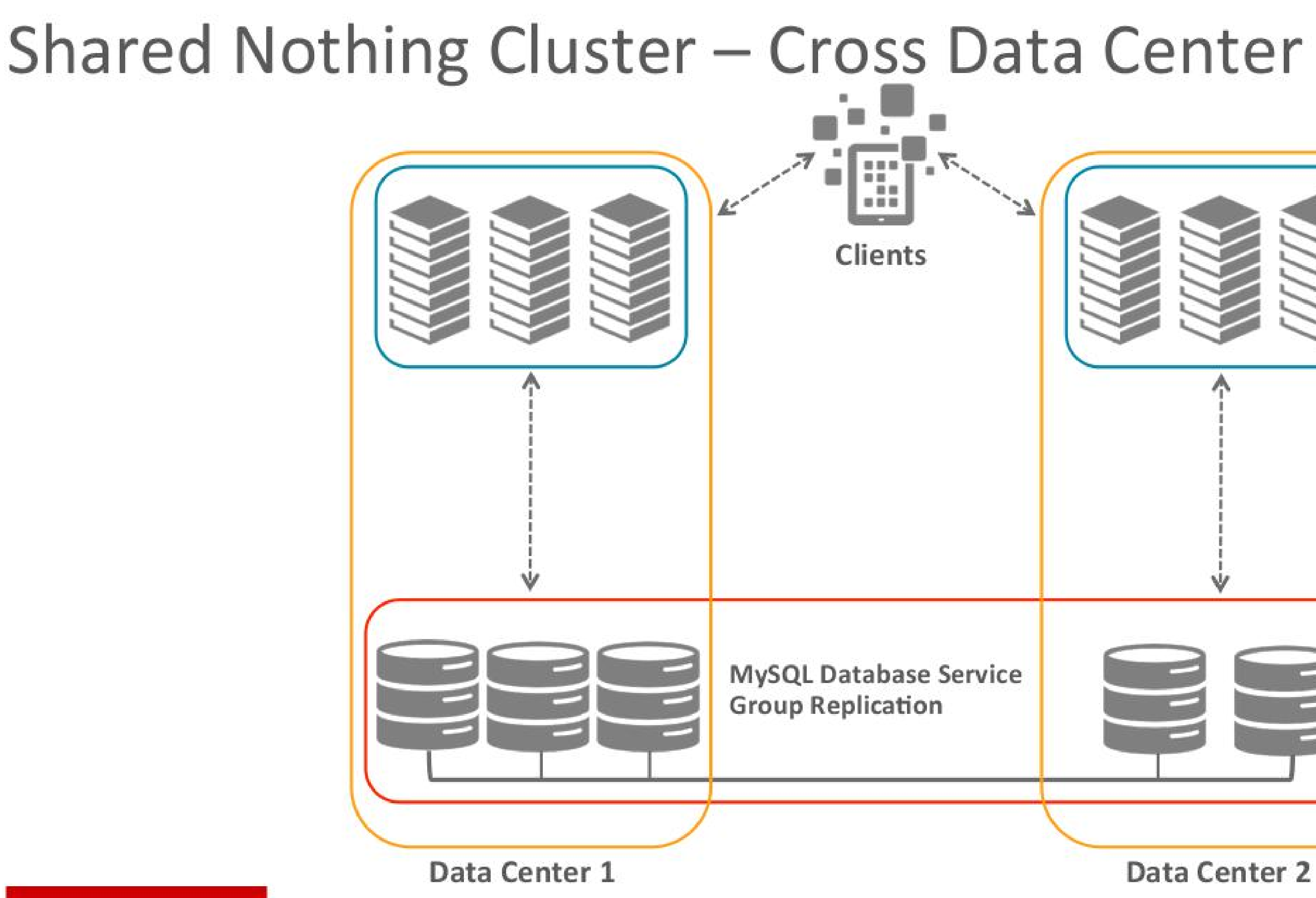
- MySQL HA
  - Group-replication
  - MySQL-Router

- MySQL DR (Geographically Redundant)
  - Daul-Data-Centers solutions
  - Solution for MySQL 5.7
  - Solution for MySQL 8.0

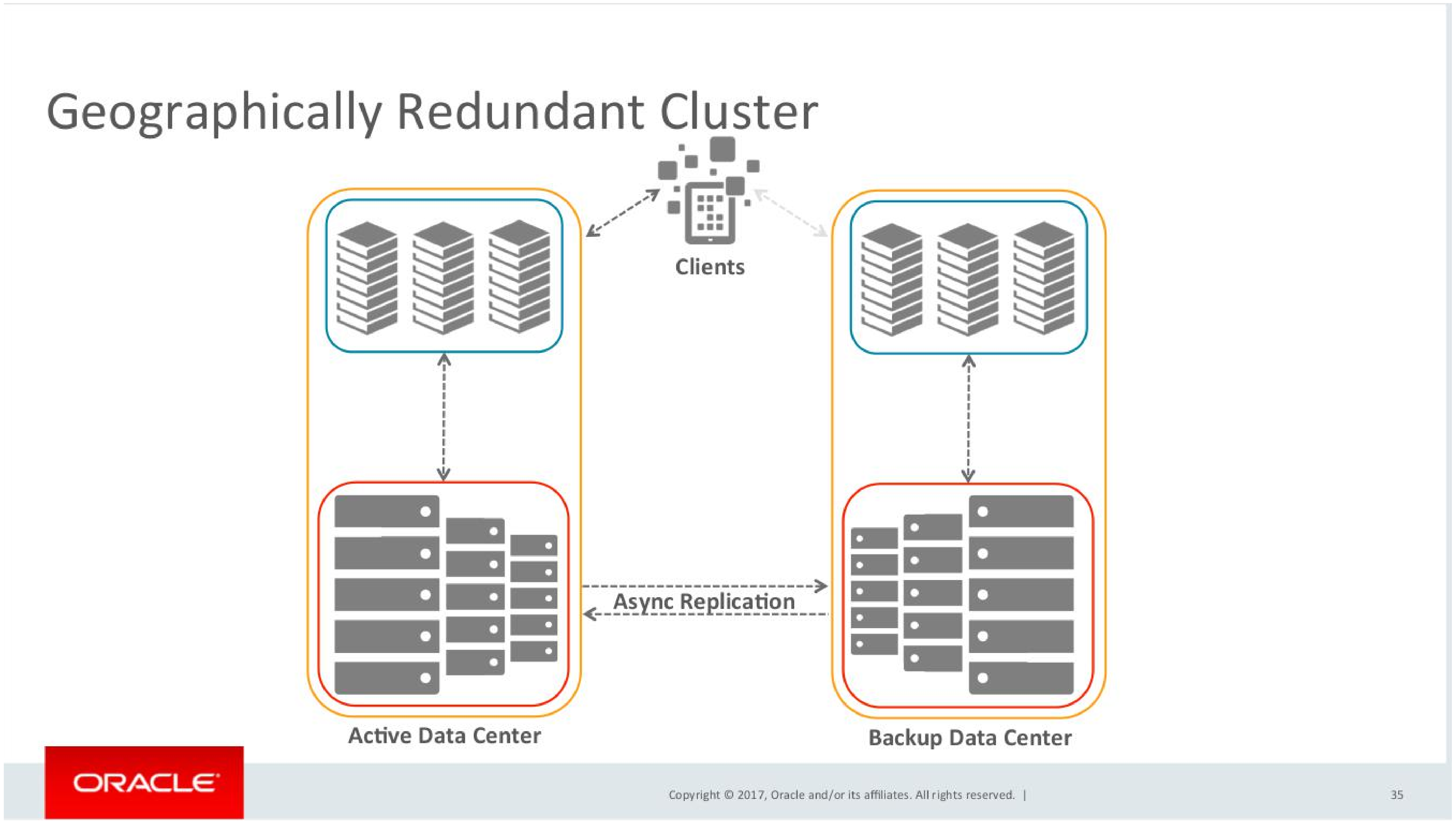
- MySQL-Operator
  - Persistence
  - Trouble Shooting

- Reference

MySQL-DR Multi-Data-Center solutions

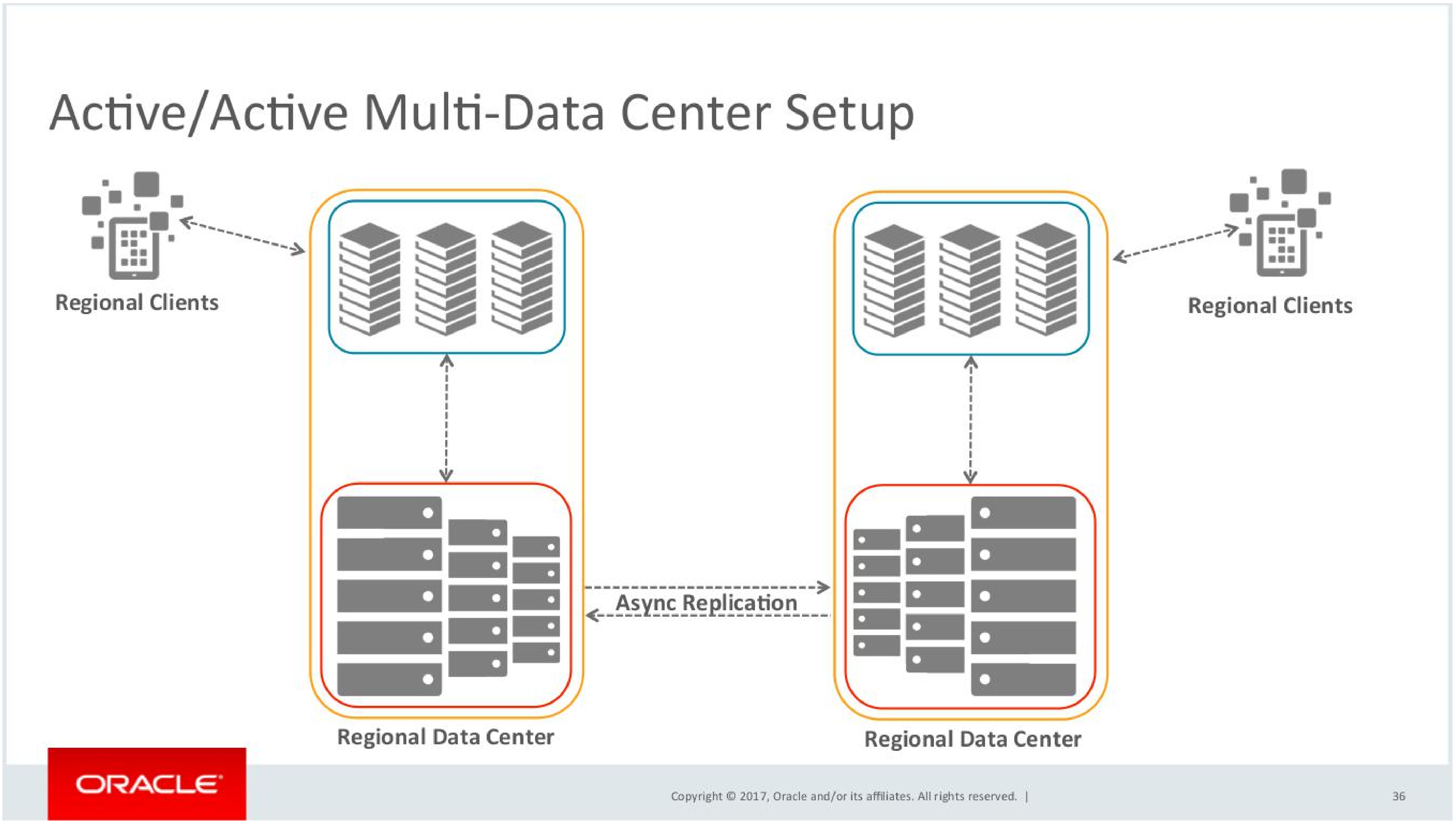


MySQL-DR Multi-Data-Center solutions



Geographically Redundant Cluster

MySQL-DR Multi-Data-Center solutions



Active/Active Multi-Data Center Setup

MySQL-DR Multi-Data-Center solutions

上面三种方法都是常见的双中心部署方案，依先后顺序称作A，B，C，各自优劣如下：

	A	B	C
优	1. 部署简单	1. 部署复杂 2. 容灾能力强	1. 部署复杂 2. 容灾能力强 3. 充分使用硬件资源，就近服务用户，真正的异地双活
劣	1. 容灾能力一般, 如果多数节点所在数据中心挂掉，不可写。 2. 网络要求较高，比如多数节点所在数据中心中的某个节点挂掉，需要在另外一个数据中心的节点写入数据后才算写成功。	1. 其中一个数据中心作为备份，有点浪费 2. 数据中心之间同步，如果使用异步同步，可能会丢部分数据	1. 数据中心之间同步，如果使用异步同步，可能会丢部分数据 2. 需要DNS或者额外的导流服务 3. 导流失败可能会导致用户看到过期数据
结论	云顶项目可以使用，需要注意group-replication 使用paxos进行数据同步，所以mysql实例必须是个奇数个节点。	可以使用。	没有必要

MySQL-DR

mysql 集群间复制时，会出现一个问题：
将集群信息当成数据进行复制，此时会出现mysql_innodb_cluster_metadata库已存在，无法复制的问题。

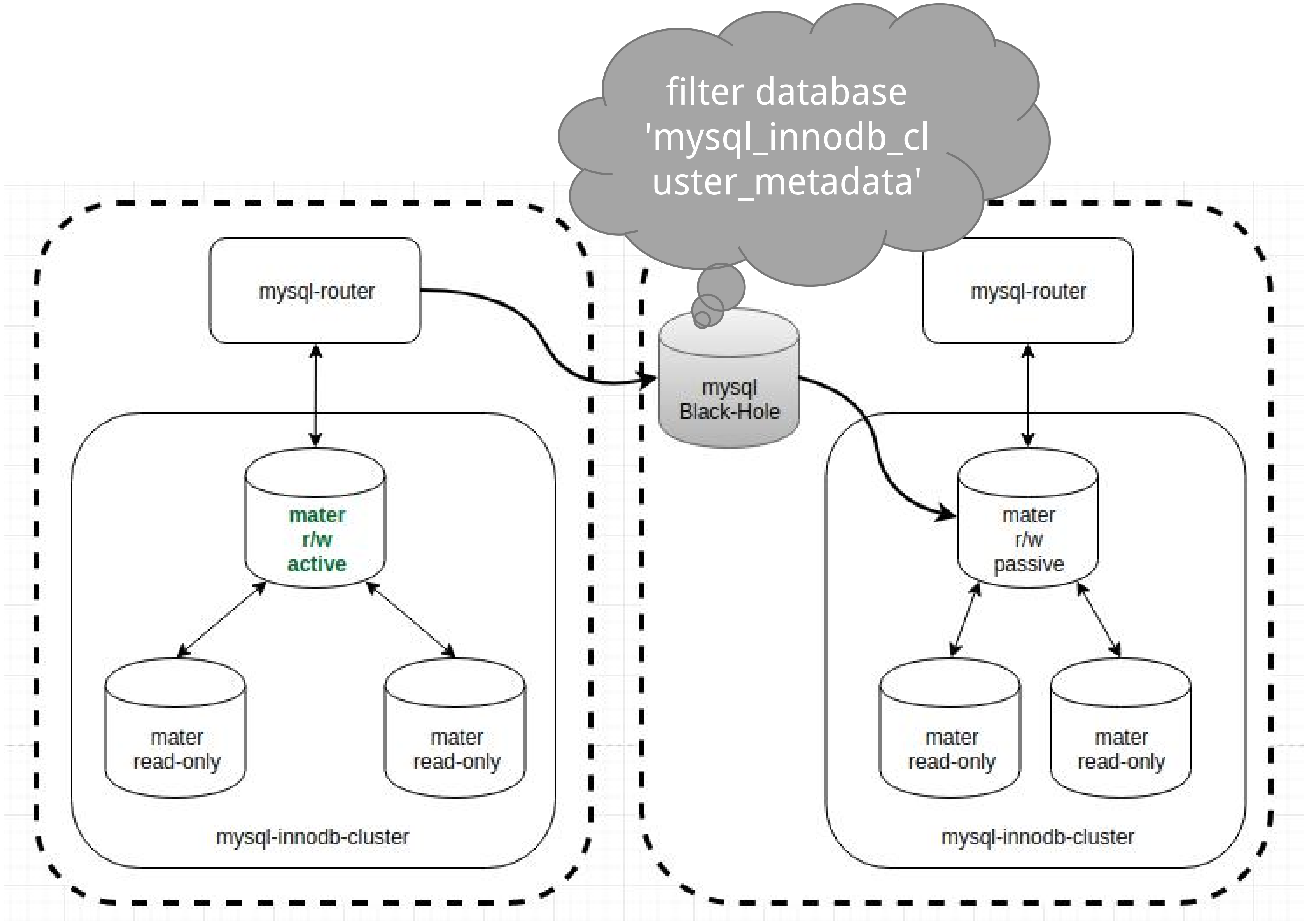
针对不同版本的mysql，有几种解决方案

MySQL-DR Solution for MySQL 5.7(1)

```
select * from clusters\G
*************************** 1. row *********
         cluster_id: 1
       cluster_name: MyGroupDC1
 default_replicaset: 1
        description: Default Cluster
mysql_user_accounts: NULL
            options: null
         attributes: {"default": true}
*************************** 2. row *********
         cluster_id: 2
       cluster_name: MyGroupDC2
 default_replicaset: 2
        description: Default Cluster
mysql_user_accounts: NULL
            options: null
         attributes: {"default": true}
```

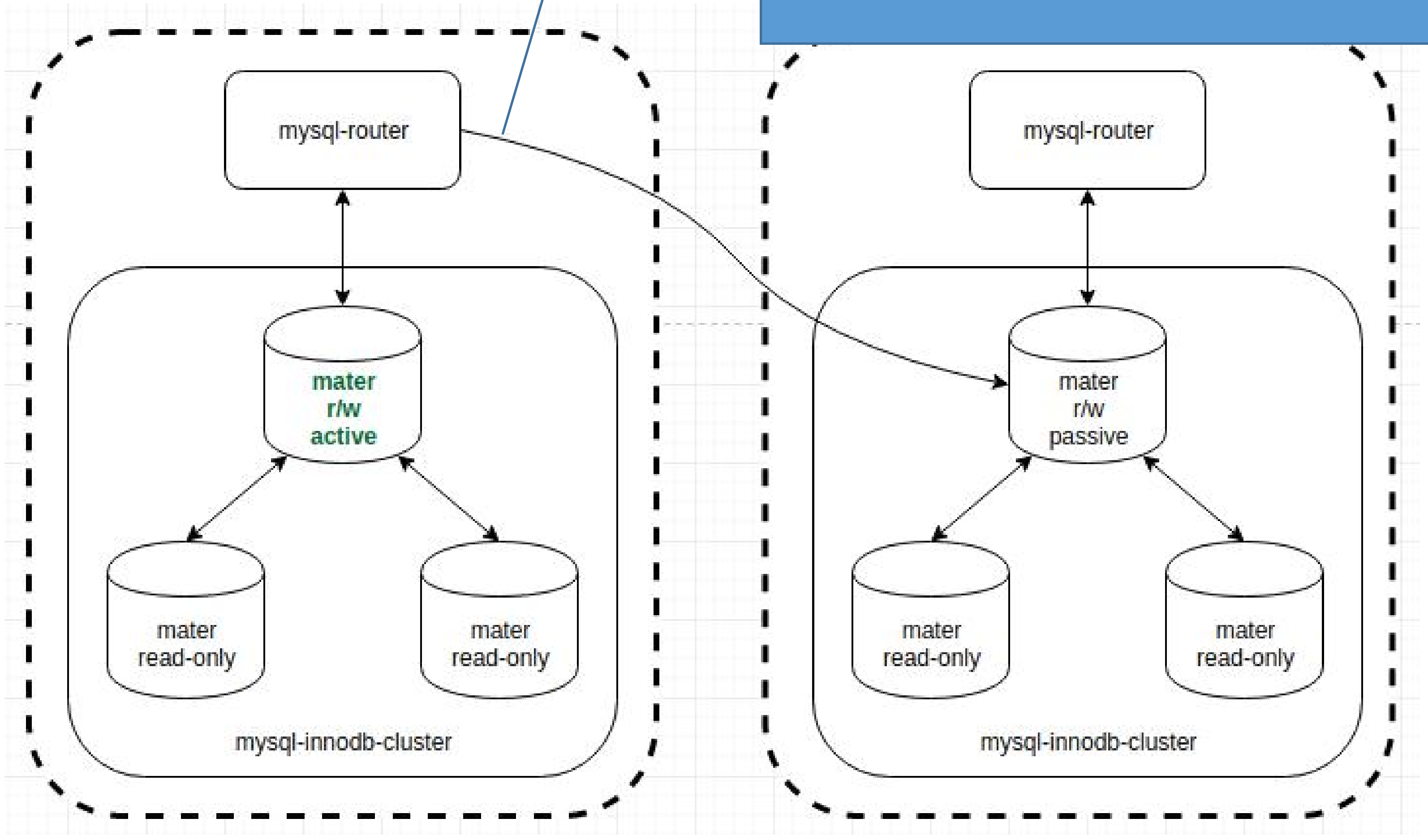
```
select instance_id, host_id, replicaset_id, instance_name
from instances;
+-------------+---------+---------------+----------------+
| instance_id | host_id | replicaset_id | instance_name  |
+-------------+---------+---------------+----------------+
|           1 |       1 |             1 | mysql1dc1:3306 |
|           2 |       2 |             1 | mysql2dc1:3306 |
|           3 |       3 |             1 | mysql3dc1:3306 |
|           7 |       4 |             2 | mysql4dc2:3306 |
|          12 |       5 |             2 | mysql5dc2:3306 |
|          19 |       6 |             2 | mysql6dc2:3306 |
+-------------+---------+---------------+----------------+
```

MySQL-DR Solution for MySQL 5.7(2)



MySQL-DR Solution for MySQL 8.0

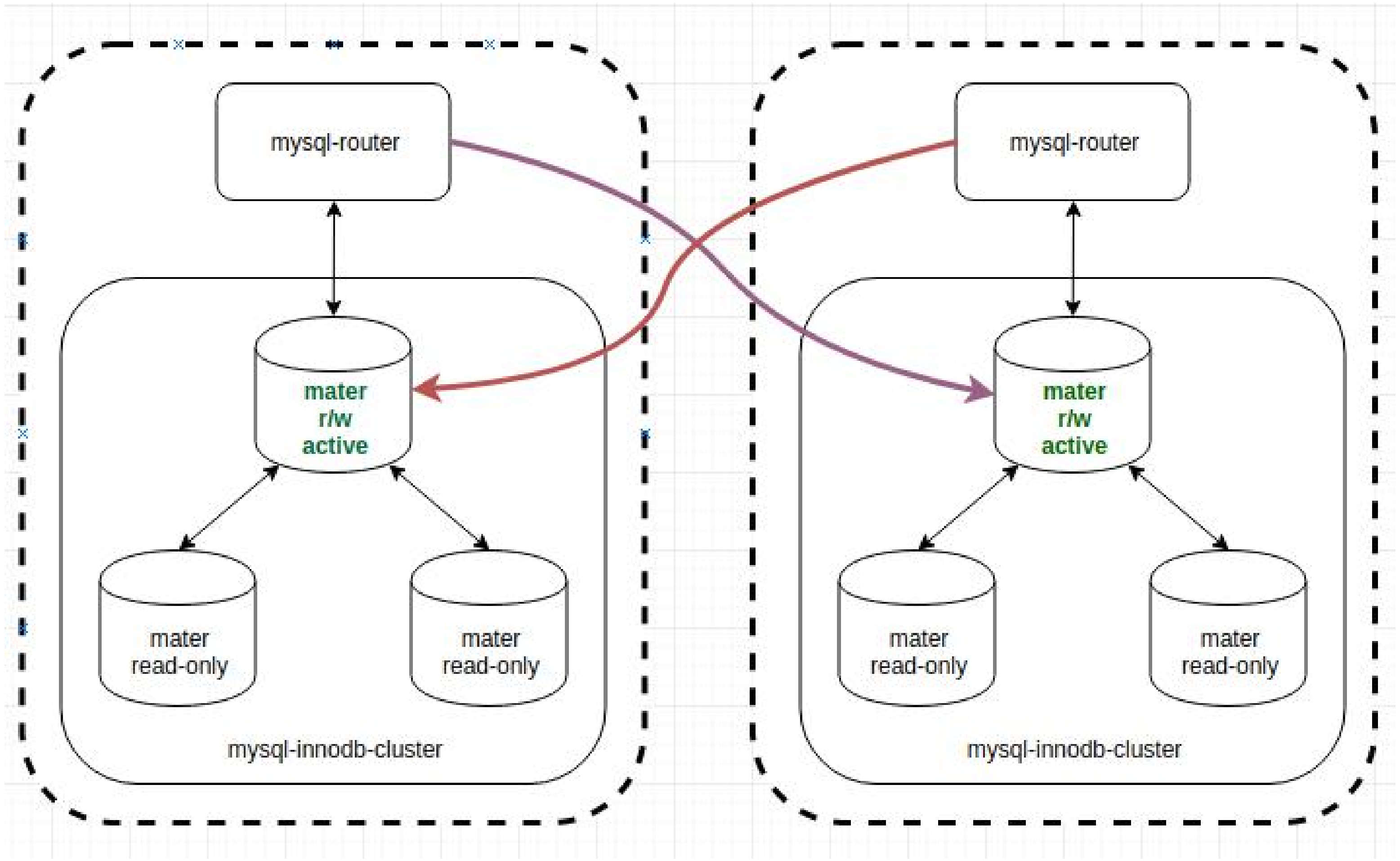
1. CHANGE MASTER TO MASTER_HOST="xxx" FOR CHANNEL "ic_to_dr"
2. CHANGE REPLICATION FILTER REPLICATE_IGNORE_DB=(mysql_innodb_cluster_metadata) FOR CHANNEL "ic_to_dr";



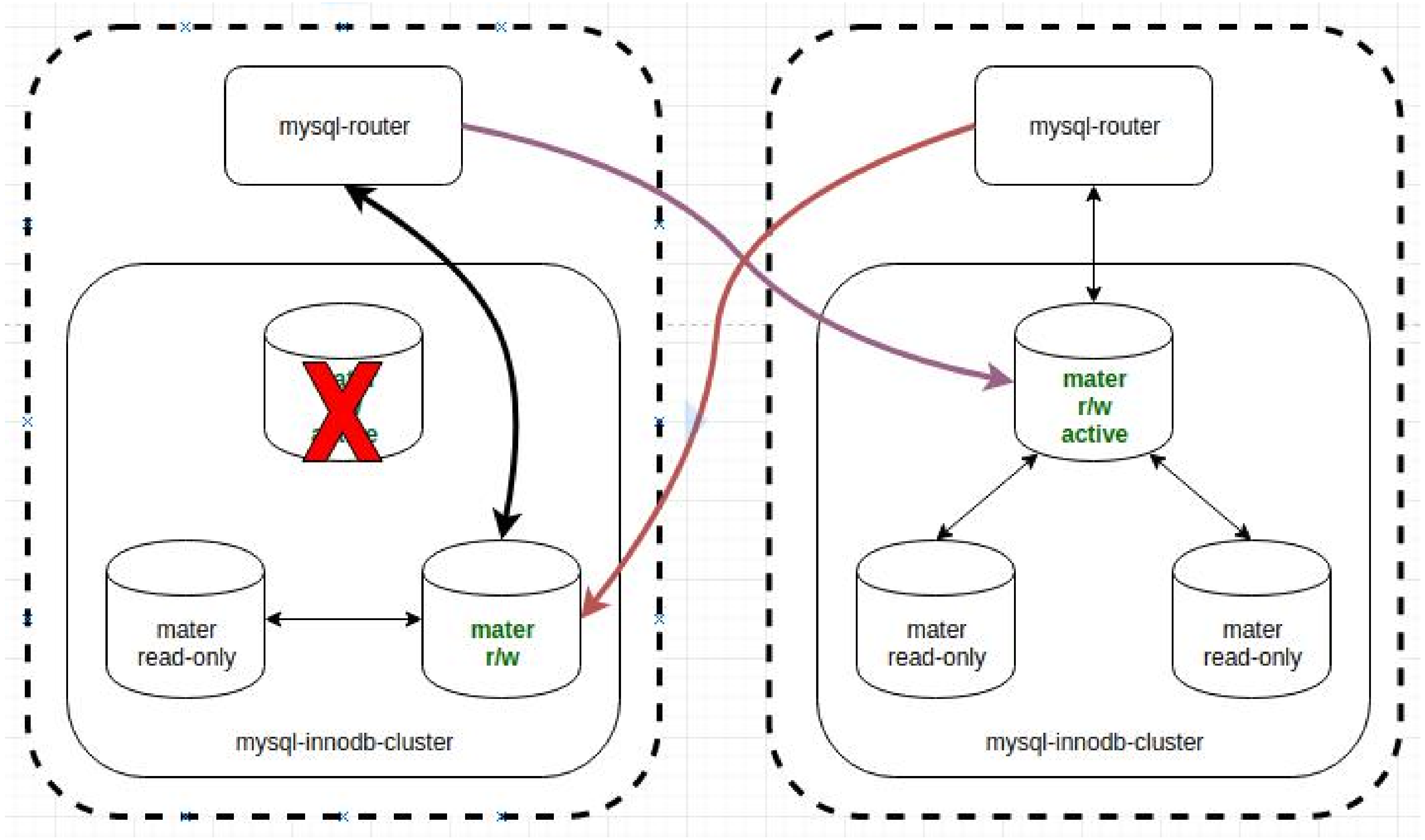
MySQL-DR Solutions conclusion

	MySQL 5.7 (1) Use public cluster-info-db	MySQL 5.7 (2) Use Black-Hole to filter source cluster-info-db	MySQL 8.0 Use Replication-Channel
优	1. 稳定的MySQL版本 2. 无额外组件 3. 无filter隐患	1. 稳定的MySQL版本 2. 部署隔离	1. 部署隔离 2. 无额外组件 3. 无filter隐患
劣	1. 部署麻烦，如果跨DC或者k8s-cluster，需要额外知道各实例的IP, 端口。	1. 可能有filter隐患 (操作失误) 2. 增加额外的组件 3. Black-Hole是单点	1. 需要高版本MySQL支持 2. 较多issue
结论	不考虑	可以考虑	暂时选定该方案

MySQL-DR final version



MySQL-DR final version



MySQL-DR final version

由于我们有一个数据中心是没有写流量的，所以双中心双active模式没有什么影响。
当某个中心挂了2个节点或者全部挂掉后，可以直接将服务切换到另外一个服务。

Agenda

- MySQL HA
  - Group-replication
  - MySQL-Router
- MySQL DR
  - Daul-Data-Centers solutions
  - Solution for MySQL 5.7
  - Solution for MySQL 8.0
- MySQL-Operator
  - Persistence
  - Trouble Shooting
- Reference

MySQL-Operator

- The MySQL Operator creates, configures and manages MySQL clusters running on Kubernetes.

- The MySQL Operator is opinionated about the way in which clusters are configured. We build upon InnoDB cluster and Group Replication to provide a complete high availability solution for MySQL running on Kubernetes.

MySQL-Operator persistent

- MySQL的持久化使用k8s local volumes
  1. 建立 StoragetClass, volumeBindingMode使用WaitForFirstConsumer. This setting tells the PersistentVolume controller to not immediately bind a PersistentVolumeClaim. Instead, the system waits until a Pod that needs to use a volume is scheduled. The scheduler then chooses an appropriate local PersistentVolume to bind to, taking into account the Pod’s other scheduling constraints and policies. This ensures that the initial volume binding is compatible with any Pod resource requirements, selectors, affinity and anti-affinity policies, and more.
  2. 建立PV. StorageClassName与 1 中name绑定。spec.local 与物理机上的预留的data路径绑定。
  3. kind: cluster volumeClainTemplate.spec.storageClassName 与1中name绑定。

MySQL-Operator trouble shooting

1. 从mysql 5.7 升级到 mysql 8.0的时候，出现了很多问题
   - mysql-agent 版本不匹配 -> 更新agent Docker file
   - 无法将第二个节点加入到集群 -> 更新operator
   - crd无法找到 -> 更改resource-definition-version
   - 域名长度超过60字节 -> 缩短cluster-name
   - 直接使用mysql-operator master最新代码，问题会少很多

2. 14集群上分配给mysql节点的ip为172.96.0.0./16, 不在集群的ip白名单上
   - 最好pod分配的ip为172.96.0.0./16
   - 暂时修改了agent的默认白名单

3. MySQL Router作为复制源，分配权限后导致router query功能丢失
   - 添加权限时注意GRANT ALL, 否则GRANT REPLICATION会取消掉query等其它权限

MySQL-Operator trouble shooting

1. 集群加入第二个节点期间出现 "this member has more executed transactions than those present in the group"
   - 需要额外清理该节点事务信息，reset master
   - 已提issue, https://github.com/oracle/mysql-operator/issues/160; 后面更新mysql镜像后现象消除

2. 加入PV持久化后，出现 "getClusterStatusFromGroupSeeds: no cluster found on any of the seed nodes, localMSHErr: signal: aborted"
   - /var/lib/mysql 权限为755, 上级目录为owner为mysql. 需要将host上的local-volume权限设置为777

3. 重启单个mysql节点或整个mysql集群失败
   - kubernete secrets重新生成，导致密码不对。按照wiki固定secrets。
   - mysql-agent需要的日志目录/var/lib/mysql权限不对 -> 修改mysql-agent的$HOME env

MySQL-Operator trouble shooting

1. 集群A，B之间的通信在A中master挂掉的情况下丢失
   - 需要修改插件逻辑，让replication重新创建

2. [Repl] Plugin group_replication reported: 'Can't start group replication on secondary member with single-primary mode while asynchronous replication channels are running.'
   - 有集群间replication的master必须关闭replication, 否则无法启动group replication
   - 如果在关闭replication前，被复制集群有新的更新同步了过来。挂掉的master启动后同步了新的更新数据，导致重新加入时发现领先于现有集群。因此需要先建立现有集群的master与对面集群的replication，当现有集群赶上时，才能将重新启动的节点加入集群。

3. 集群A，B需要有不同的base server id

MySQL-Operator trouble shooting (推测)

1. 3节点k8s集群上部署, 拔掉某台机器网线或重启某台机器, 经常会出现mysql-cluster无法恢复的情况. 典型场景有以下几种:
   - 拔掉网线的机器恢复后, 出现本pod DNS解析失败的现象, pod ip解析为老ip
   - 拔掉其中一台机器的网线后, 可能会影响到其它机器, 具体表现为ks8集群判断两个物理节点 not ready
   - 如果mysql-cluster有三个节点 (A, B, C), B掉线或关机后, A认为C掉线, C则认为A在线. 实际上期望B的异常并不会影响到A,C 两个节点.

   - ==> 通过mysql-gcs日志会发现理论上的节点间的心跳信息会在有异常情况出现后断掉, 因此迫于无奈只能先怀疑到k8s的(flannel)网络问题, 使用k8s host-network. 经验证还不错, 比使用cluster-network要好很多. 关掉重启两个物理节点也能恢复.

MySQL-Operator trouble shooting (推测)

1. 想将cluster-network下的mysql-cluster直接迁移到host-network下, 第二个节点会出现一直recovering的状态,加入不了集群.

    ==> 观察 mysql-clsuter-0的状态(show master status), 会发现比mysql-cluster-1多出了6个建立session的transaction. 怀疑与 mysql-cluster-0 与自身33061的连接有关, 连接数也为6个, 可以看下详细的transcation binlog继续追查下. 因为这几个transcation无法进行replication, 所以mysql-clsuter-1一直处于recovering状态.

    ==> 迁移方式

    a. 在k8s集群里面创建另外一个mysql-host-cluster

    b. 从mysql-cluster导数据到mysql-host-cluster, 注意ignore-gtid. --set-gtid-purged=OFF, 使用mysqldump! mysqlpump会出现@SESSION.SQL_LOG_BIN=0;导致read-only-master 没有数据

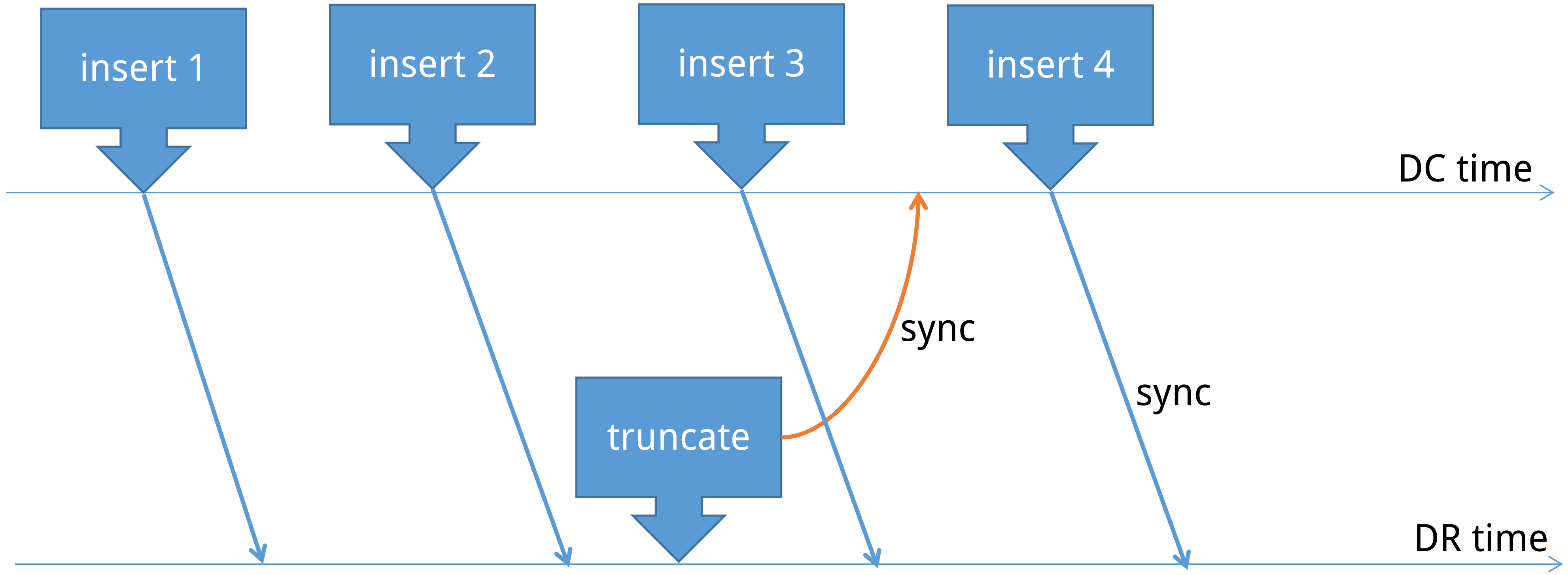
    c. 重启mysql-router, 指向mysql-host-cluster, 应用层无感知.

MySQL-Operator 警告！！！

DC正常写入，DR使用truncate table操作。导致数据不一致。

DC中只有4, DR中有3和4。

所以如果DC, DR对任一公用数据进行操作，都有可能会导致最终不一致性。所以在**“双主”**的架构下，需要保证只对一中心进行访问。



Agenda

- MySQL HA
    - Group-replication
    - MySQL-Router
- MySQL DR\
    - Daul-Data-Centers solutions
    - Solution for MySQL 5.7
    - Solution for MySQL 8.0
- MySQL-Operator
    - Persistence
    - Trouble Shooting
- Reference

https://lefred.be/content/mysql-ha-architecture-1-innodb-cluster-consul/

https://lefred.be/content/reduce-human-interaction-when-using-an-asynchronous-slave-to-a-mysql-innodb-cluster/

https://github.com/oracle/mysql-operator