Native MySQL HA

An Introduction to InnoDB Clusters

ORACLE

MySQL

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.

Program Agenda

1. An Introduction to InnoDB Clusters
2. The Components
3. Setup, Monitoring, and Management
4. Reference Architectures
5. What's Next?

Program Agenda

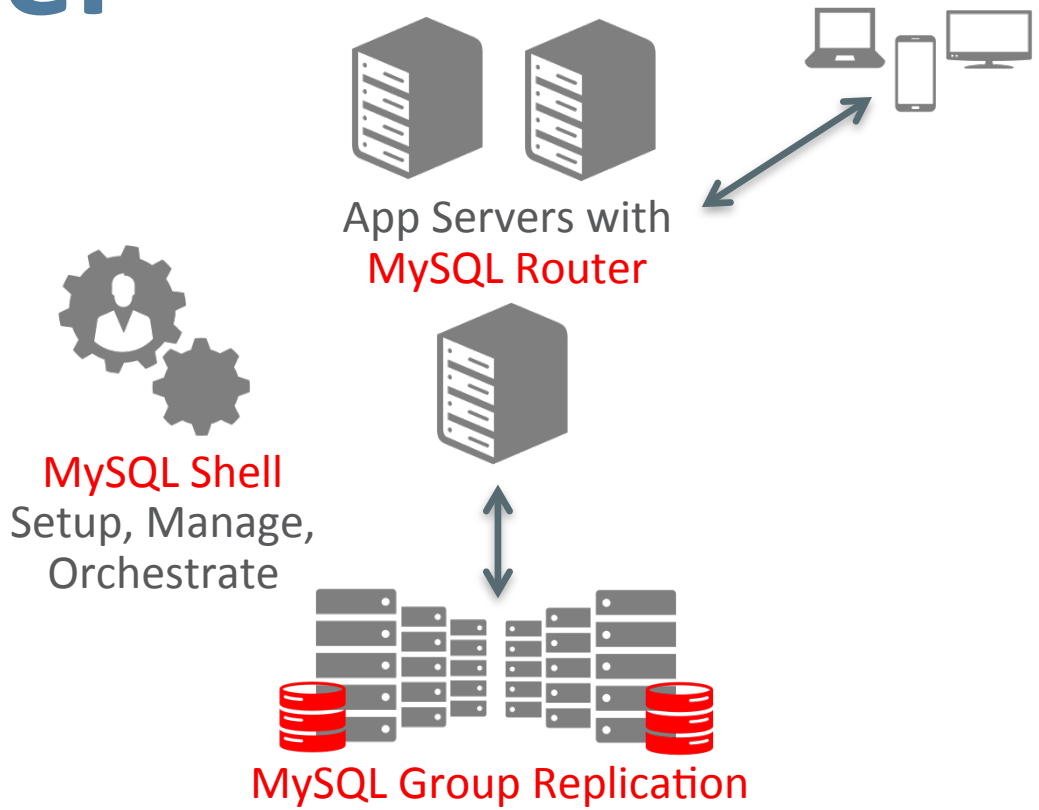
1. An Introduction to InnoDB Clusters
2. The Components
3. Setup, Monitoring, and Management
4. Reference Architectures
5. What's Next?

100%

Virtually all organizations require their most critical systems to be highly available

MySQL InnoDB Cluster

"High Availability becomes a core first class feature of MySQL!"

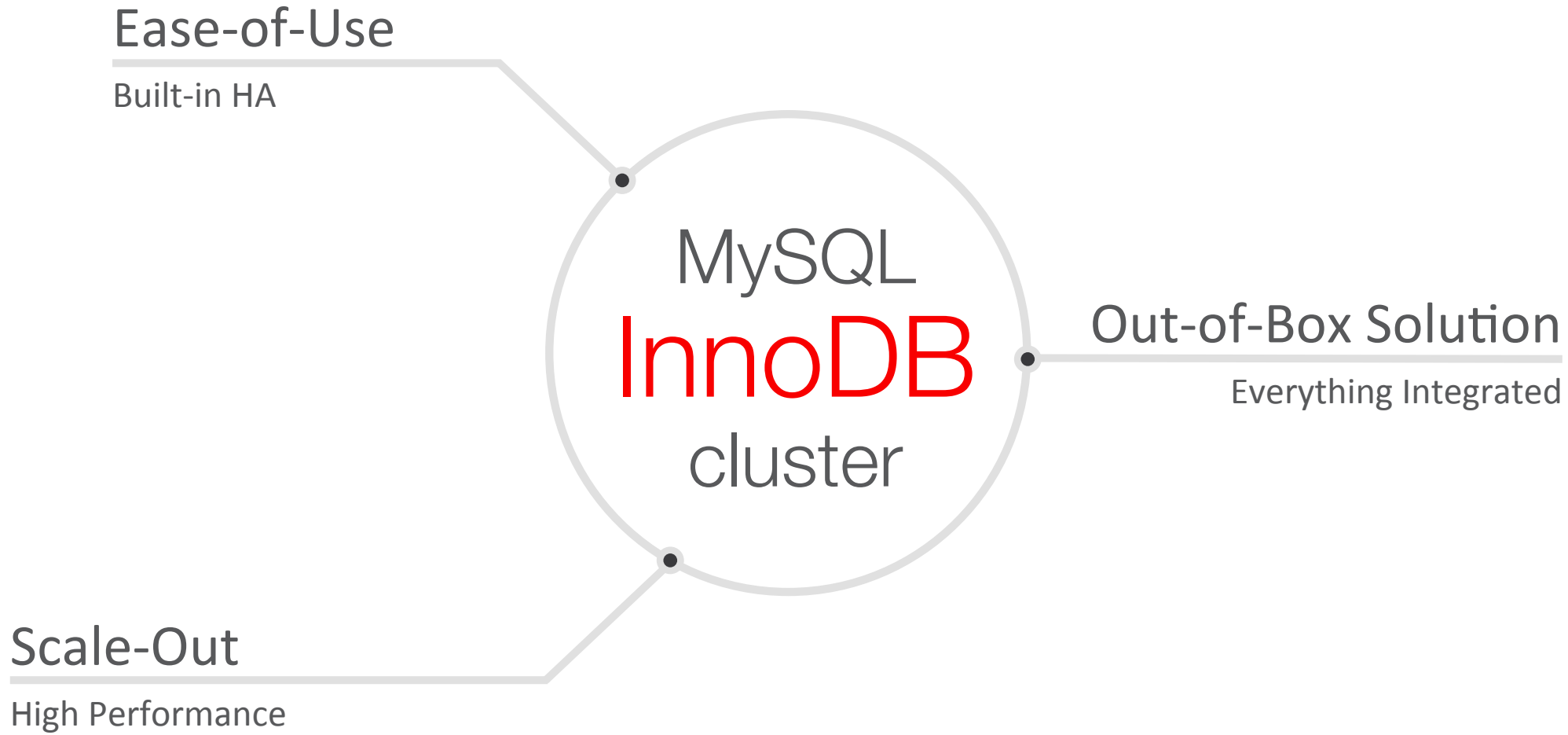


MySQL InnoDB Cluster: Vision

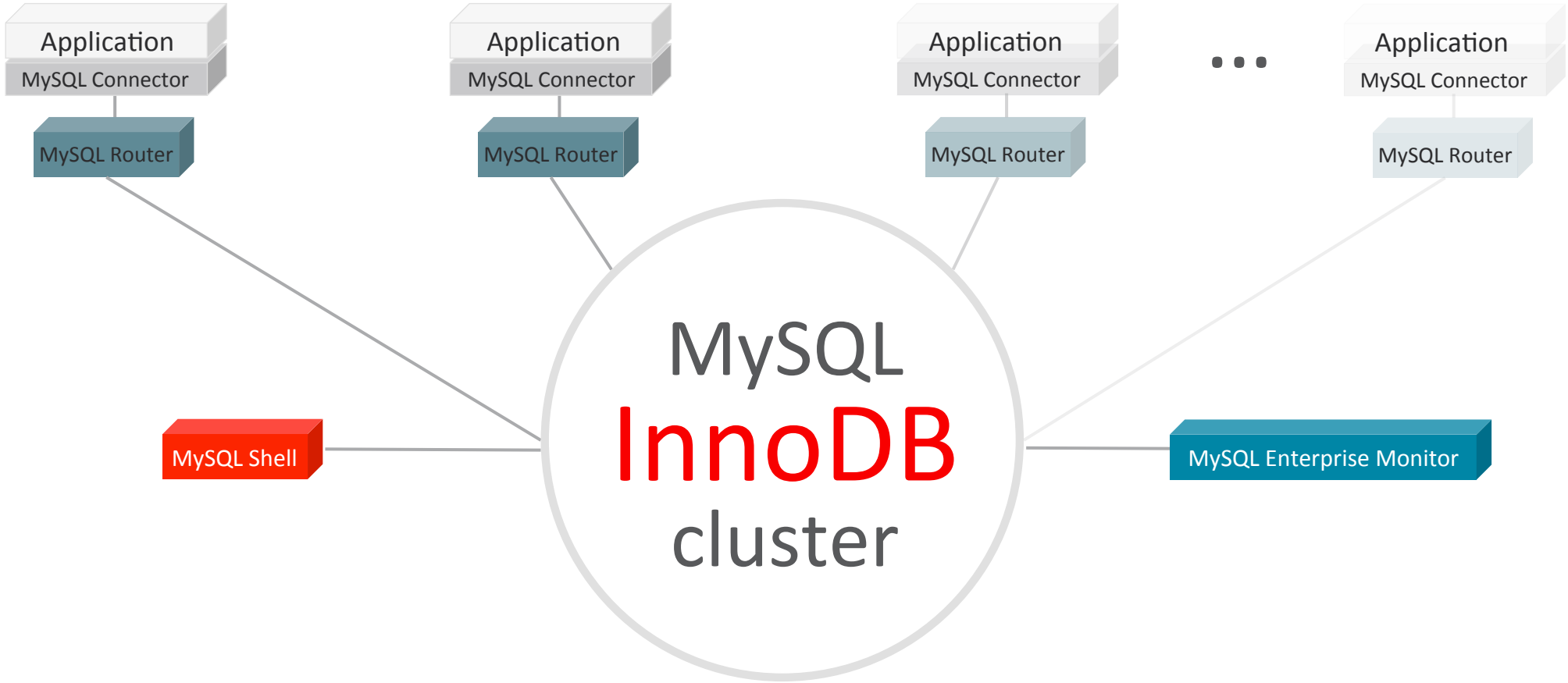
"A single product — MySQL — with high availability and scaling features baked in; providing an integrated end-to-end solution that is easy to use."

MySQL InnoDB Cluster: Goals

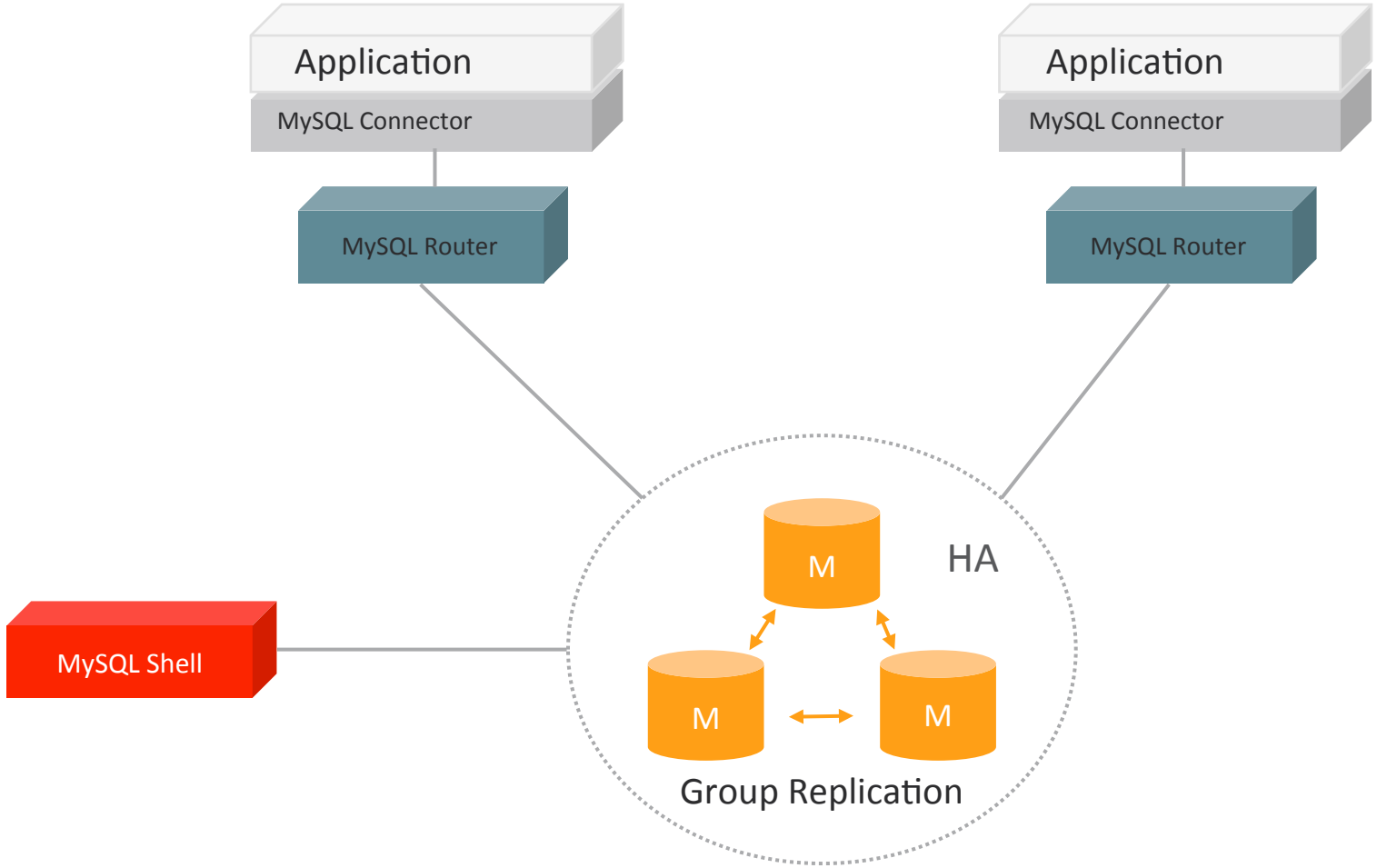
- One Product: MySQL
  - All components created together
  - Tested together
  - Packaged together
- Easy to Use
  - One client: MySQL Shell
  - Easy packaging
  - Integrated orchestration
  - Homogenous servers
- Flexible and Modern
  - SQL and NoSQL together
  - Protocol Buffers
  - Asynchronous API
  - Developer friendly
- Support Read/Write Scale Out
  - Sharded clusters
  - Federated system of N replica sets
    - Supporting cross shard operations
  - Each replica set manages a shard



MySQL InnoDB Cluster: High Level Architecture

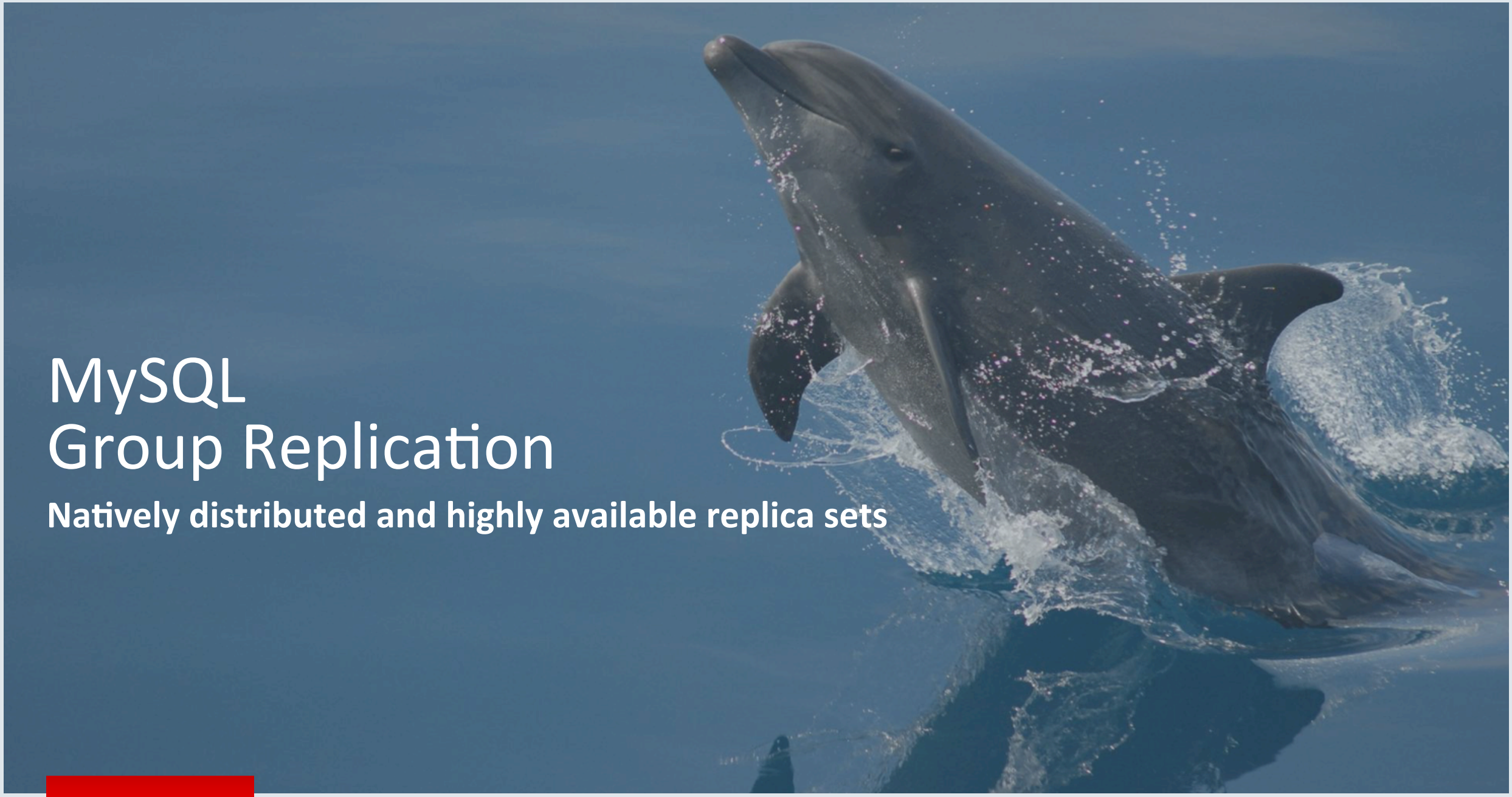


MySQL InnoDB Cluster: Internal Architecture



Program Agenda

1. An Introduction to InnoDB Clusters
2. The Components
3. Setup, Monitoring, and Management
4. Reference Architectures
5. What's Next?



MySQL Group Replication

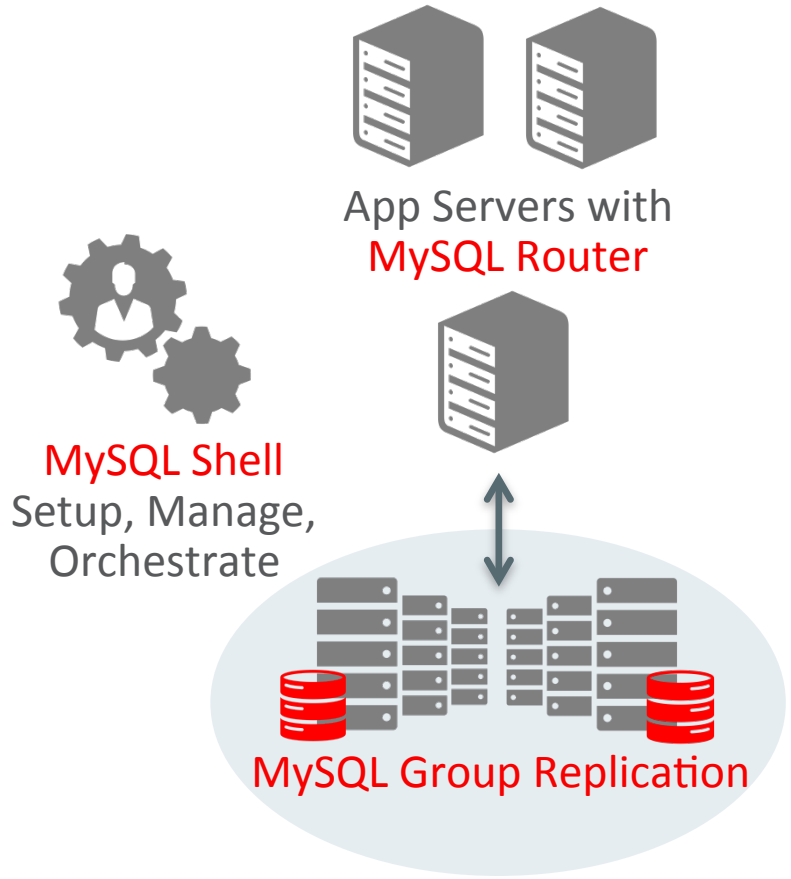
Natively distributed and highly available replica sets

ORACLE

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**."*



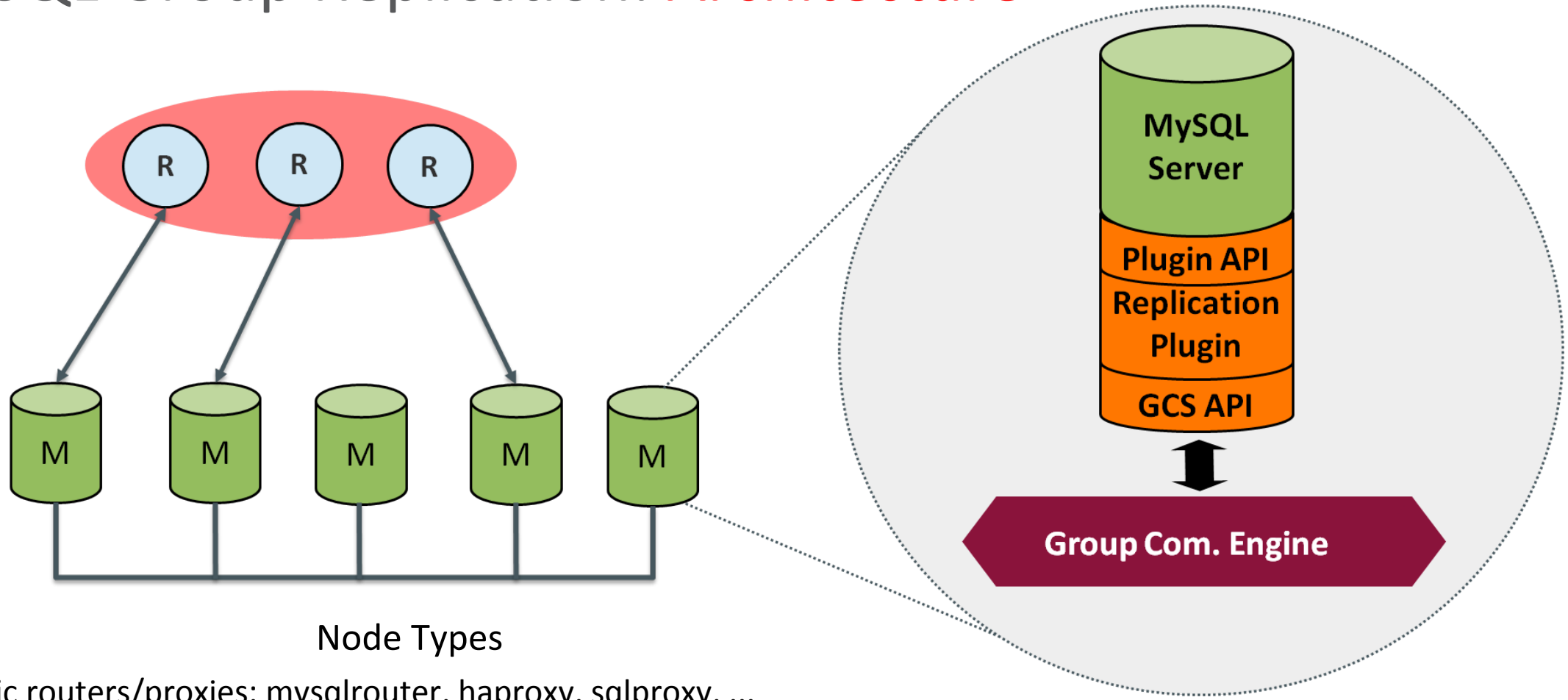
MySQL Group Replication: What Does It Provide?

- 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

MySQL Group Replication: Use Cases

- **Elastic Replication**
  - Environments that require a very fluid replication infrastructure, where the number of servers has to grow or shrink dynamically and with as little pain as possible.
- **Highly Available Shards**
  - Sharding is a popular approach to achieve write scale-out. Users can use MySQL Group Replication to implement highly available shards in a federated system. Each shard can map into a Replication Group.
- **Alternative to Master/Slave Replication**
  - It may be that a single master server makes it a single point of contention. Writing to an entire group may prove more scalable under certain circumstances.

MySQL Group Replication: Architecture

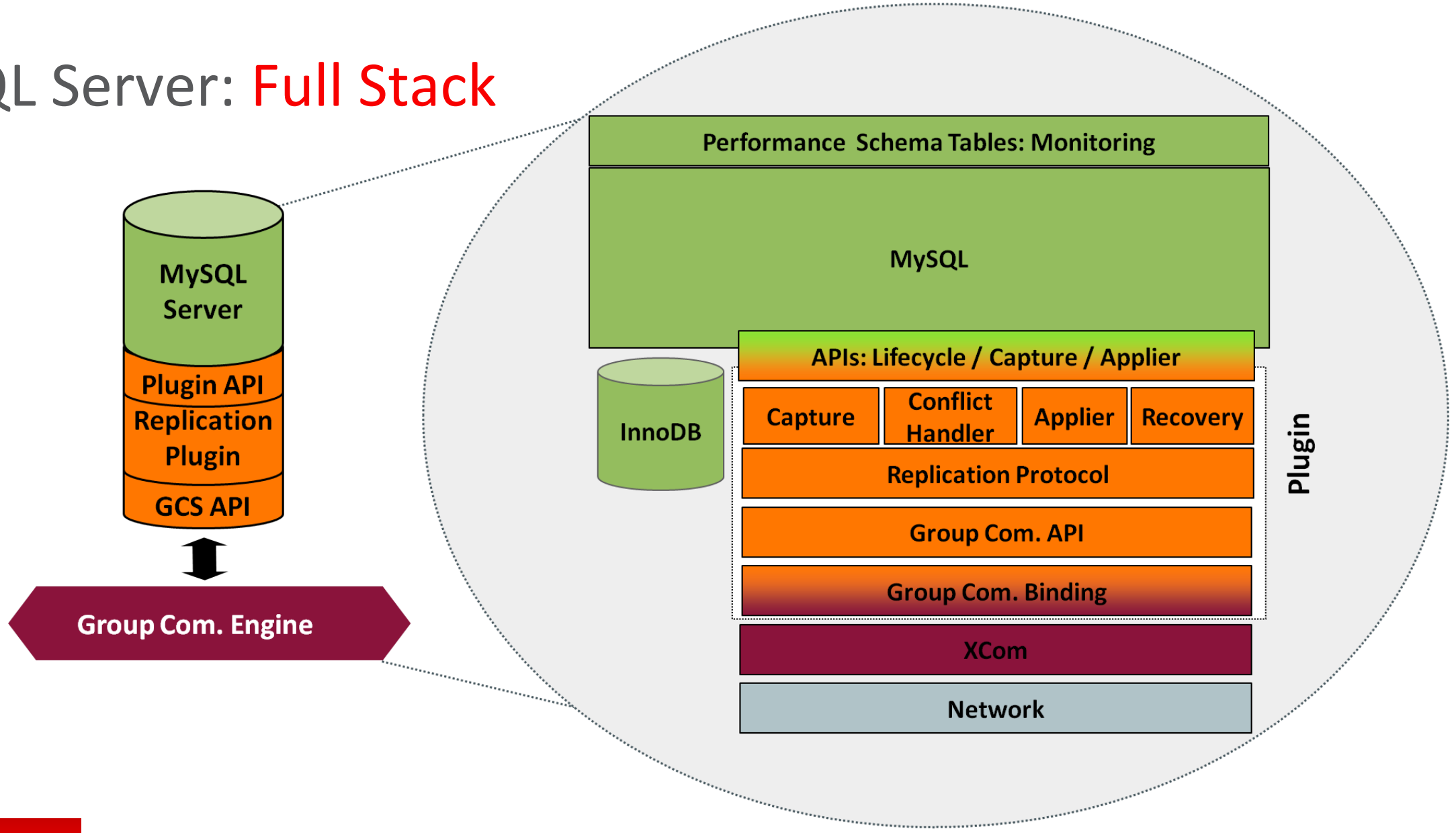


Node Types

R: Traffic routers/proxies: mysqlrouter, haproxy, sqlproxy, ...

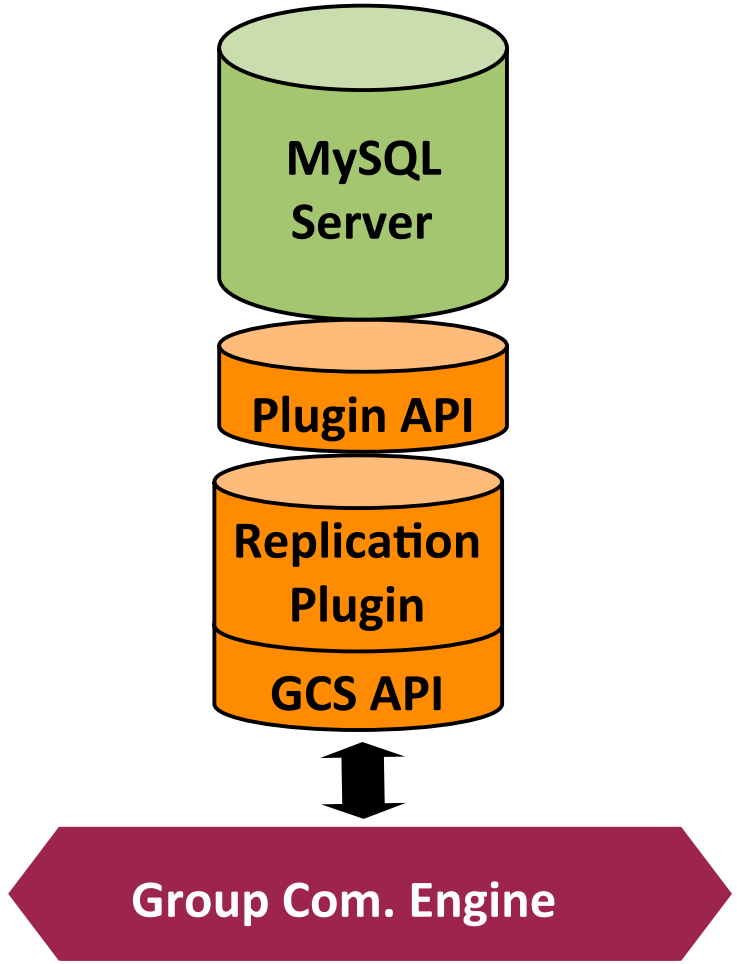
M: mysqld nodes participating in Group Replication

MySQL Server: Full Stack



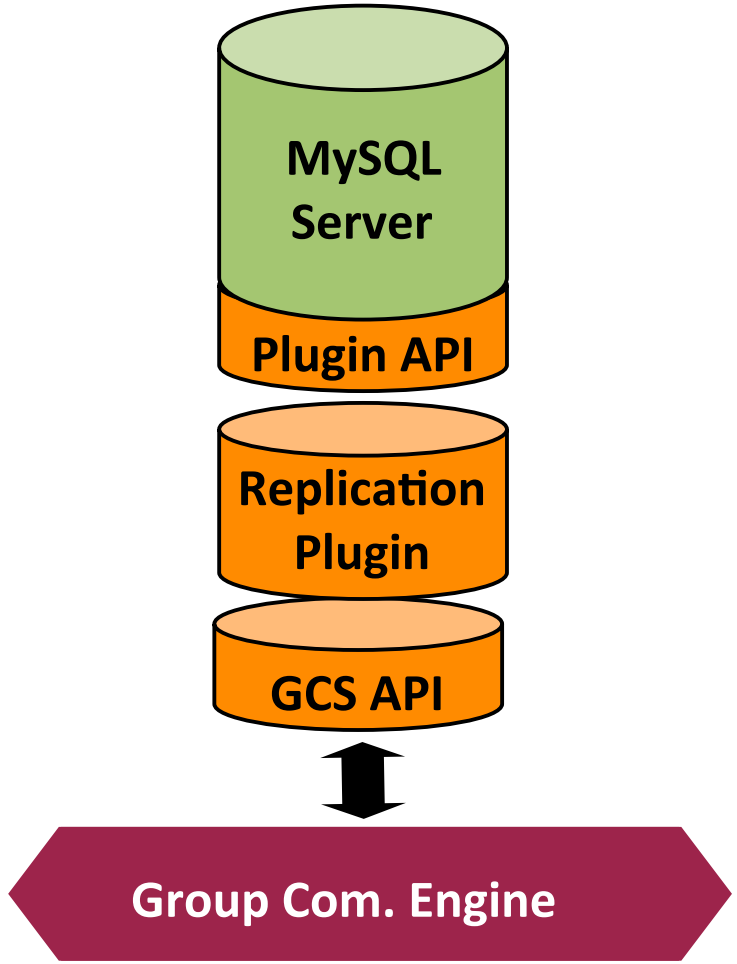
MySQL Server: Core

- Server calls into the plugin through a generic interface
  - Most server internals are hidden from the plugin
- Plugin interacts with the server through a generic interface
  - Replication plugin determines the fate of the commit operation through a well defined server interface
  - The plugin makes use of the relay log infrastructure to inject changes in the receiving server



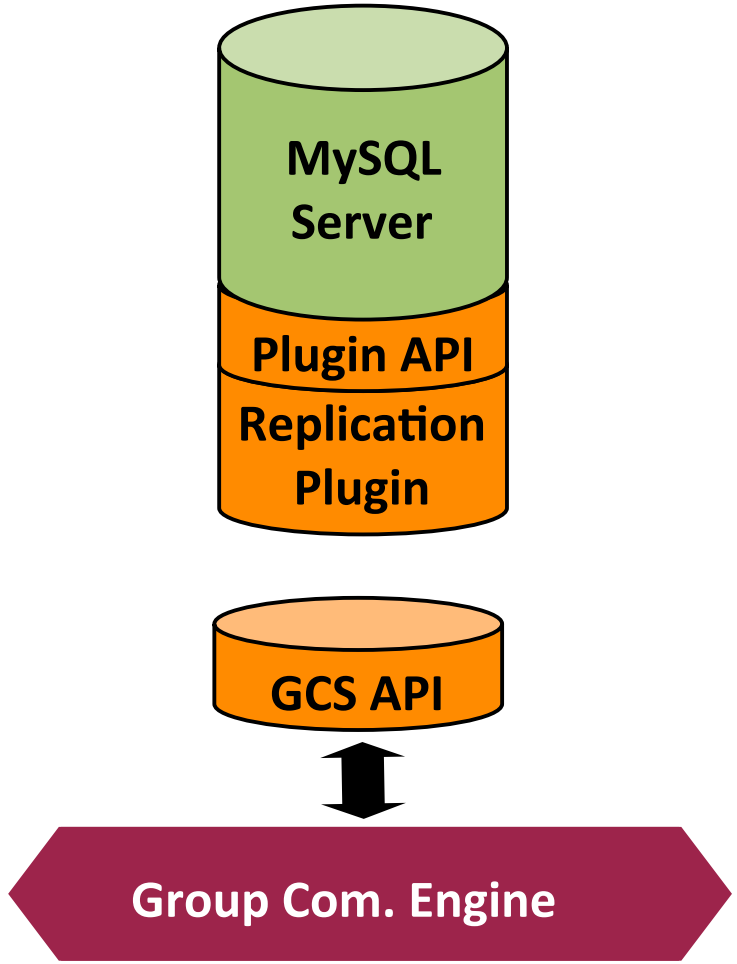
MySQL Server: Group Replication Plugin

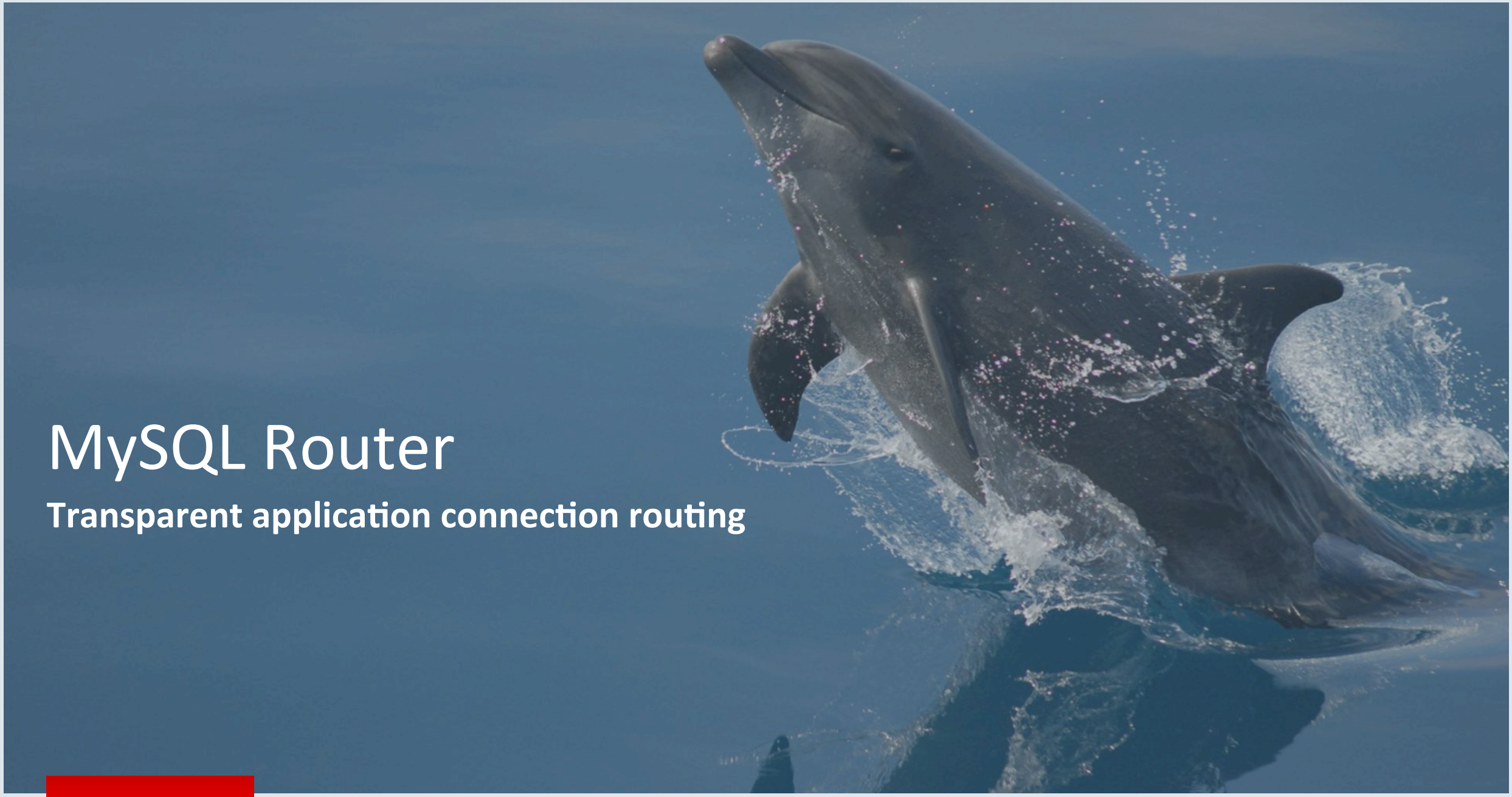
- The plugin is responsible for
  - Maintaining distributed execution context
  - Detecting and handling conflicts
  - Handling distributed recovery
    - Detect membership changes
    - Donate state if needed
    - Collect state if needed
  - Proposing transactions to other members
  - Receiving and handling transactions from other members
  - Deciding the ultimate fate of transactions
    - commit or rollback



MySQL Server: GCS

- The communication API (and binding) is responsible for:
  - Abstracting the underlying group communication system implementation from the plugin itself
  - Mapping the interface to a specific group communication system implementation
- The Group Communication System engine:
  - Variant of Paxos developed at MySQL
  - Building block to provide distributed agreement between servers





MySQL Router

Transparent application connection routing

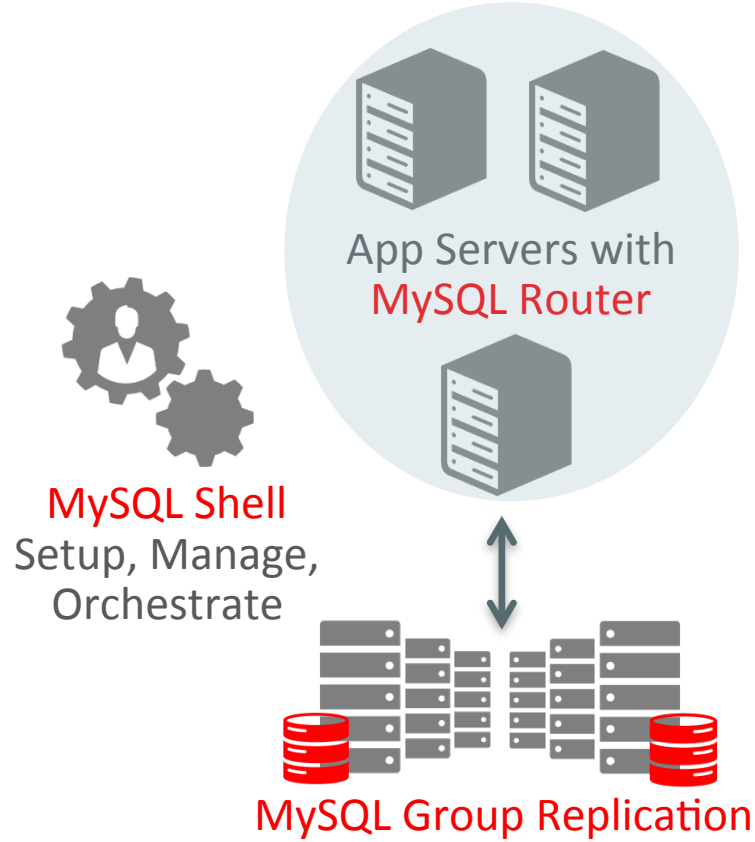
ORACLE

MySQL Router

Transparent access to HA databases for MySQL Applications

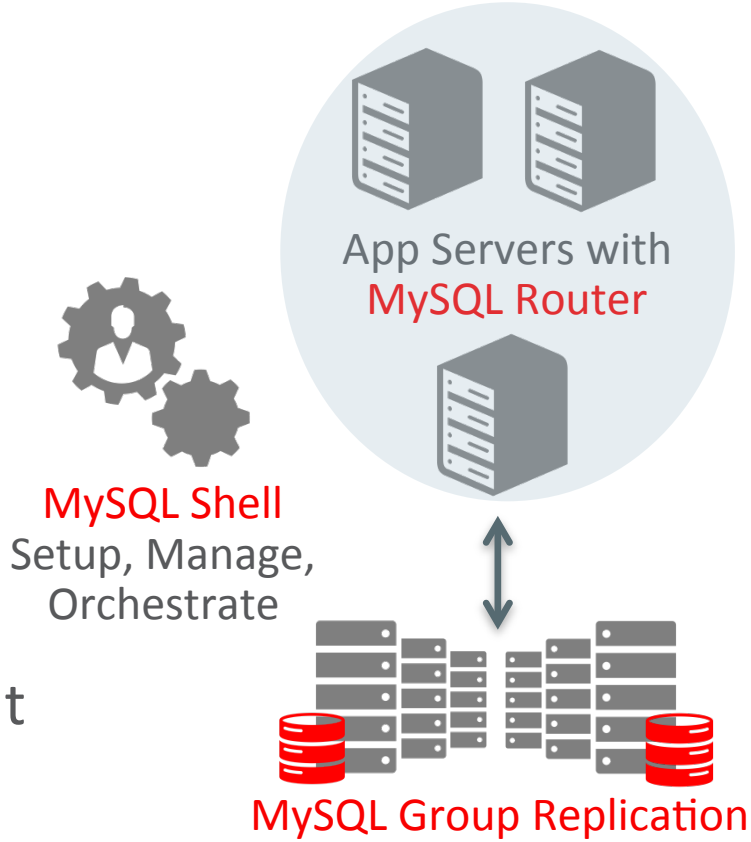
- Transparent client connection routing
  - Load balancing
  - Application connection failover
- Stateless design offers easy HA client routing
  - A local Router becomes part of the application stack

"MySQL Router allows you to easily migrate your standalone MySQL instances to natively distributed and highly available InnoDB clusters without affecting existing applications!"



MySQL Router: 2.1

- Native support for InnoDB clusters
  - Understands Group Replication topology
  - Utilizes metadata schema stored on each member
    - Bootstraps itself and sets up client routing for the InnoDB cluster
    - Allows for intelligent client routing into the InnoDB cluster
  - Supports multi-master and single primary modes
- Core improvements
  - Built-in keyring for easy and secure password management



*"MySQL Router 2.1, with the new metadata_cache plugin, provides **transparent** client connection **routing and failover** into your InnoDB clusters!"*

MySQL Shell

Single tool for development, setup, management, orchestration, and monitoring

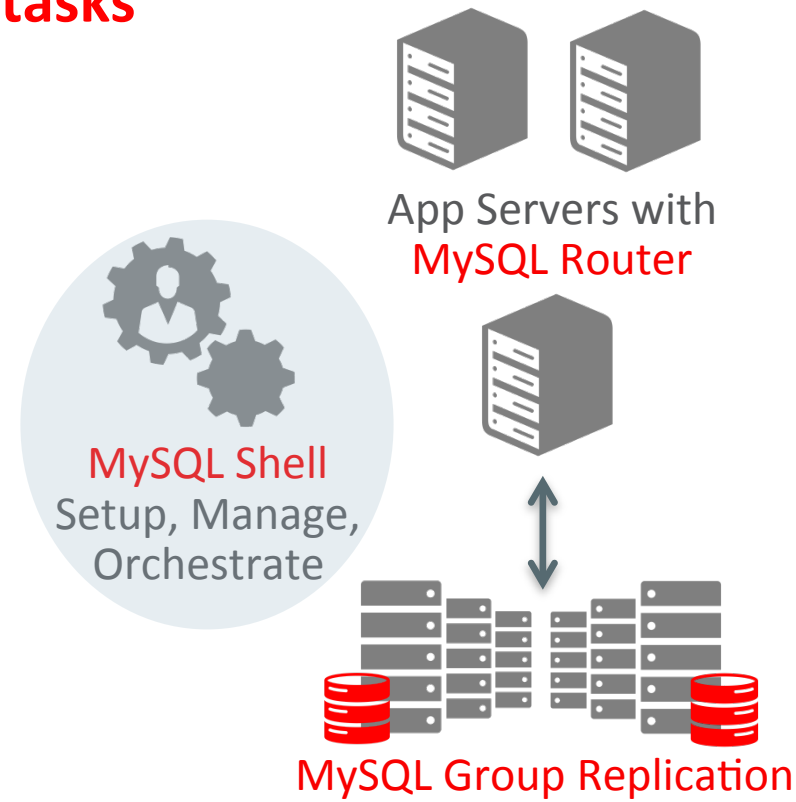
ORACLE

MySQL Shell

A single unified client for all administrative and operations tasks

- Multi-Language: JavaScript, Python, and SQL
  - Naturally scriptable
- Supports both Document and Relational models
- Exposes full Development and Admin API

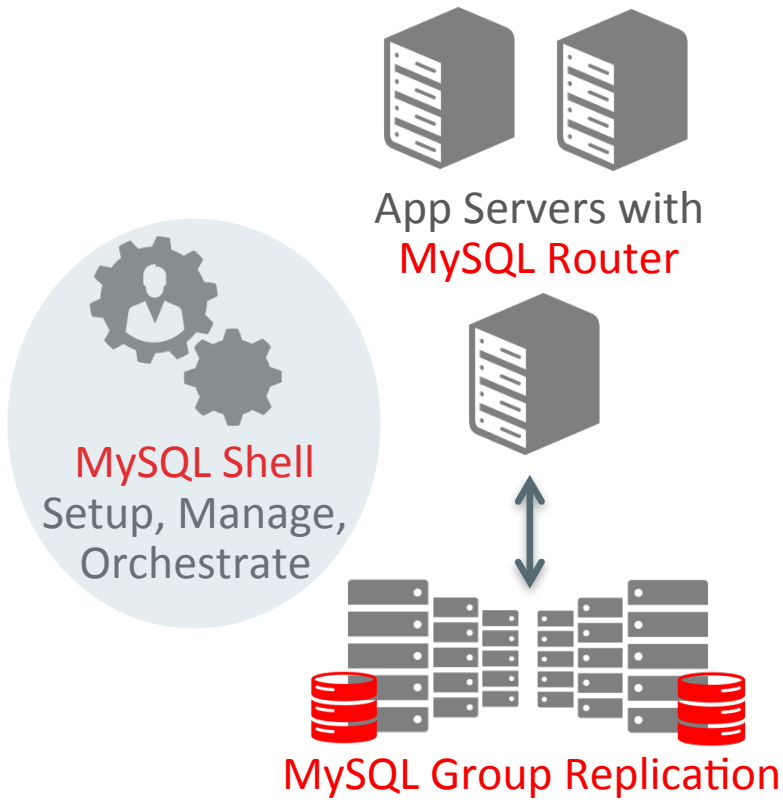
”MySQL Shell provides the developer and DBA with a single intuitive, flexible, and powerful interface for all MySQL related tasks!”



MySQL Shell: Admin API

Database Administration Interface

- `mysql-js> dba.help()`
- The global variable 'dba' is used to access the MySQL AdminAPI
- Perform DBA operations
  - Manage MySQL InnoDB clusters
    - Create clusters
    - Validate MySQL instances
    - Configure MySQL instances
    - Get cluster info
    - Modify clusters



Program Agenda

1. An Introduction to InnoDB Clusters
2. The Components
3. Setup, Monitoring, and Management
4. Reference Architectures
5. What's Next?

MySQL Shell: Configure MySQL Instances

```
shell> mysqlsh root@localhost

mysql-js> dba.checkInstanceConfiguration('localhost:3306')

mysql-js> dba.configureLocalInstance('localhost:3306')
```

MySQL Shell: Create an InnoDB Cluster

```
mysql-js> cluster = dba.createCluster('NewAppCluster')

mysql-js> cluster.addInstance('root@hanode1:3306')
mysql-js> cluster.addInstance('root@hanode2:3306')
mysql-js> cluster.addInstance('root@hanode3:3306')

# to persist the config changes on each machine
shell> mysqlsh root@localhost
mysql-js> dba.configureLocalInstance('localhost:3306')
```

MySQL Shell: Add a MySQL Router

```
shell> mysqlrouter --bootstrap hanode1:3306

shell> mysqlrouter &

shell> mysqlsh root@localhost:6446
```

MySQL Shell: Check Status

```
shell> mysqlsh root@localhost:6446

mysql-js> cluster = dba.getCluster()

mysql-js> cluster.status()
```

MySQL Shell: Check Status

```
mysql-js> cluster.status()
{
    "clusterName": "mylabcluster",
    "defaultReplicaSet": {
        "name": "default",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "hanode1:3306": {
                "address": "hanode1:3306",
                "mode": "R/W",
                "readReplicas": {},
                "role": "HA",
                "status": "ONLINE"
            },
…
```

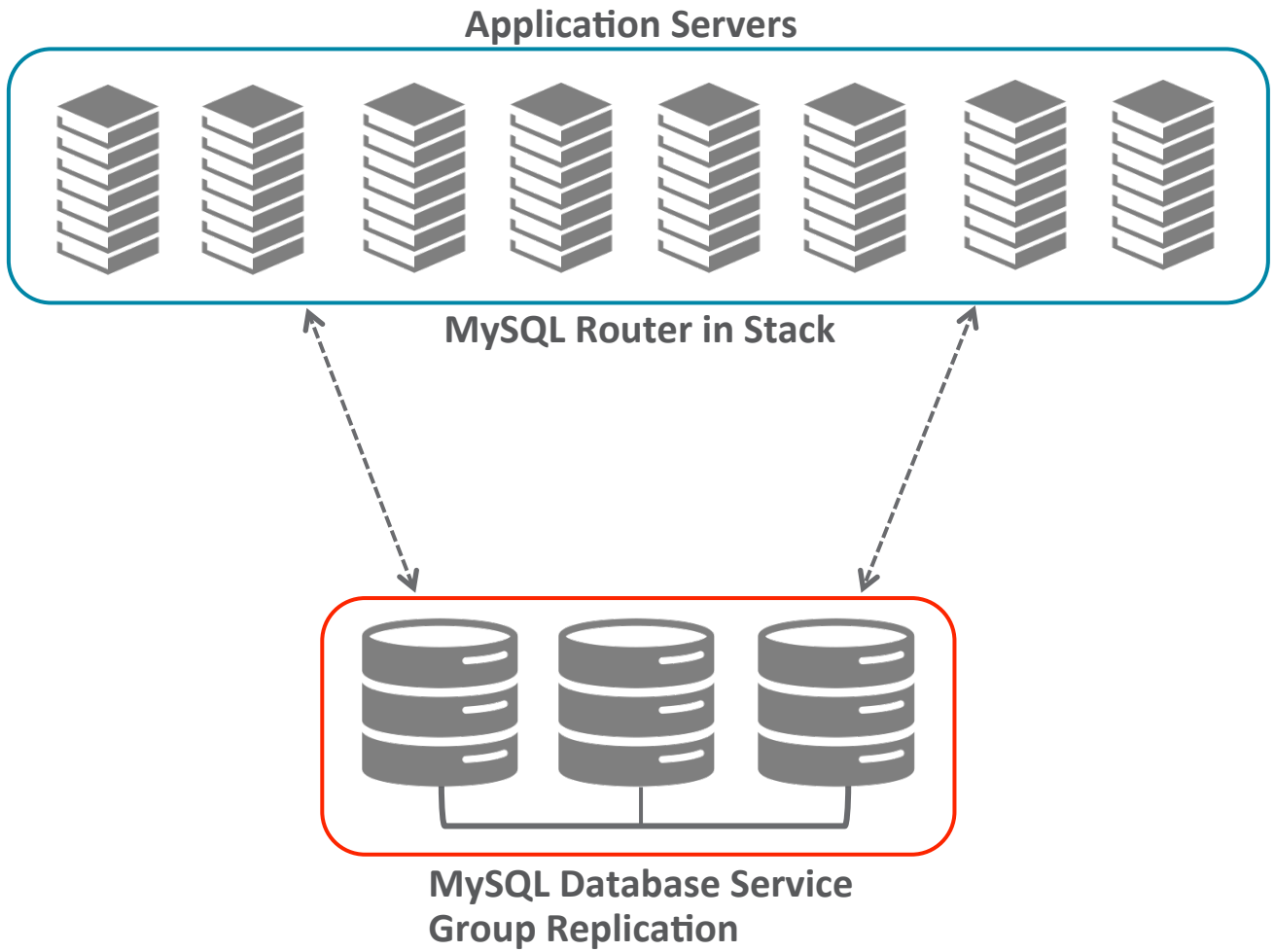
Program Agenda

1. An Introduction to InnoDB Clusters
2. The Components
3. Setup, Monitoring, and Management
4. Reference Architectures
5. What's Next?

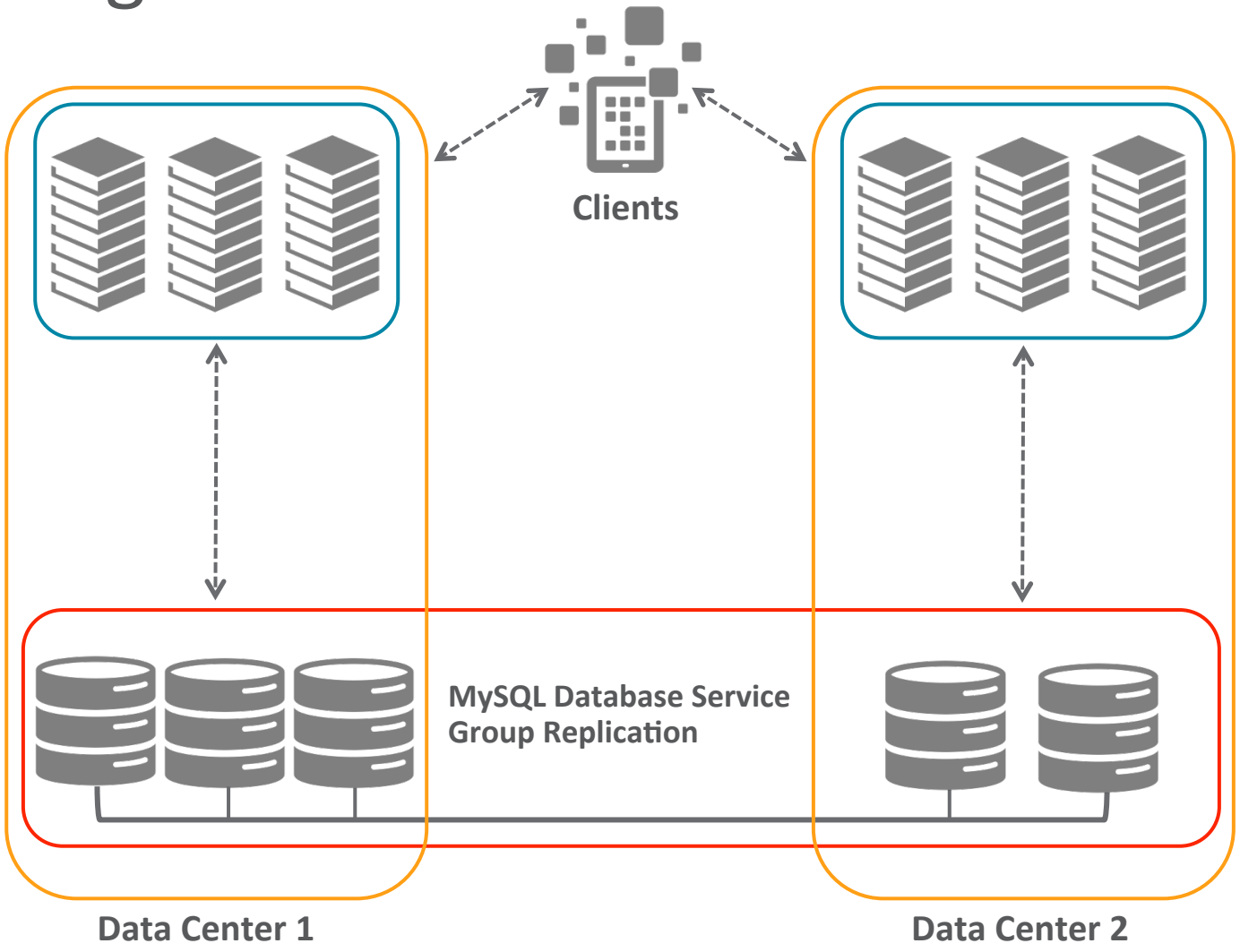
Hardware and Infrastructure Notes

- 3, 5, or 7 machines per group
  - Isolate machine resources as much as possible
  - Limit virtualization layers
  - Machines configured for dedicated database server role
    - Recommended configuration for HPC setups
      - 32-64 vCPUs with fast CPU clock (2.5GHz+)
      - SSDs (for data and replication logs)
  - High quality network connection between each machine
    - Low latency, high throughput, reliable
    - Limit network routers and hubs as much as possible
    - Isolated and dedicated network when possible

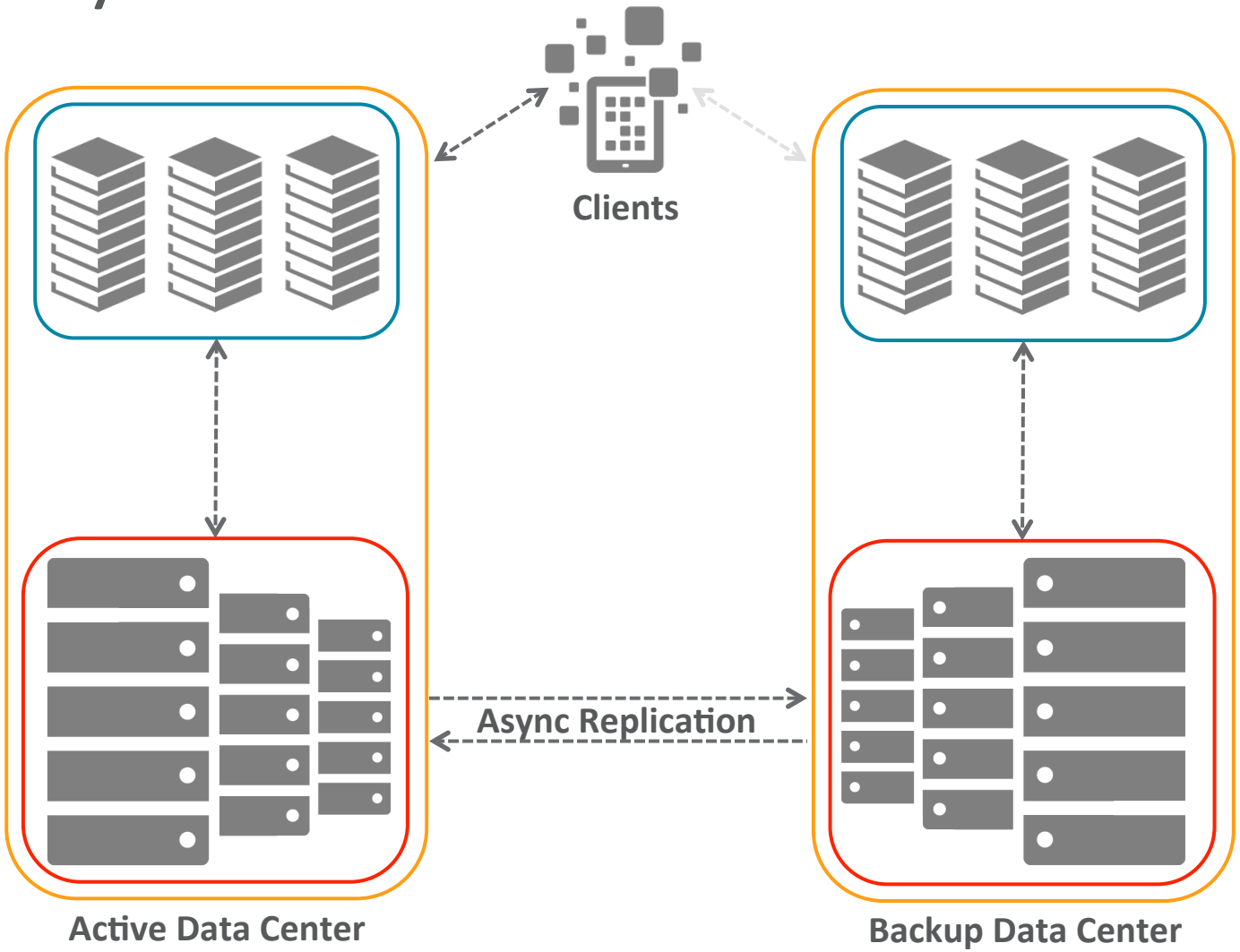
Shared Nothing Cluster – Single Data Center



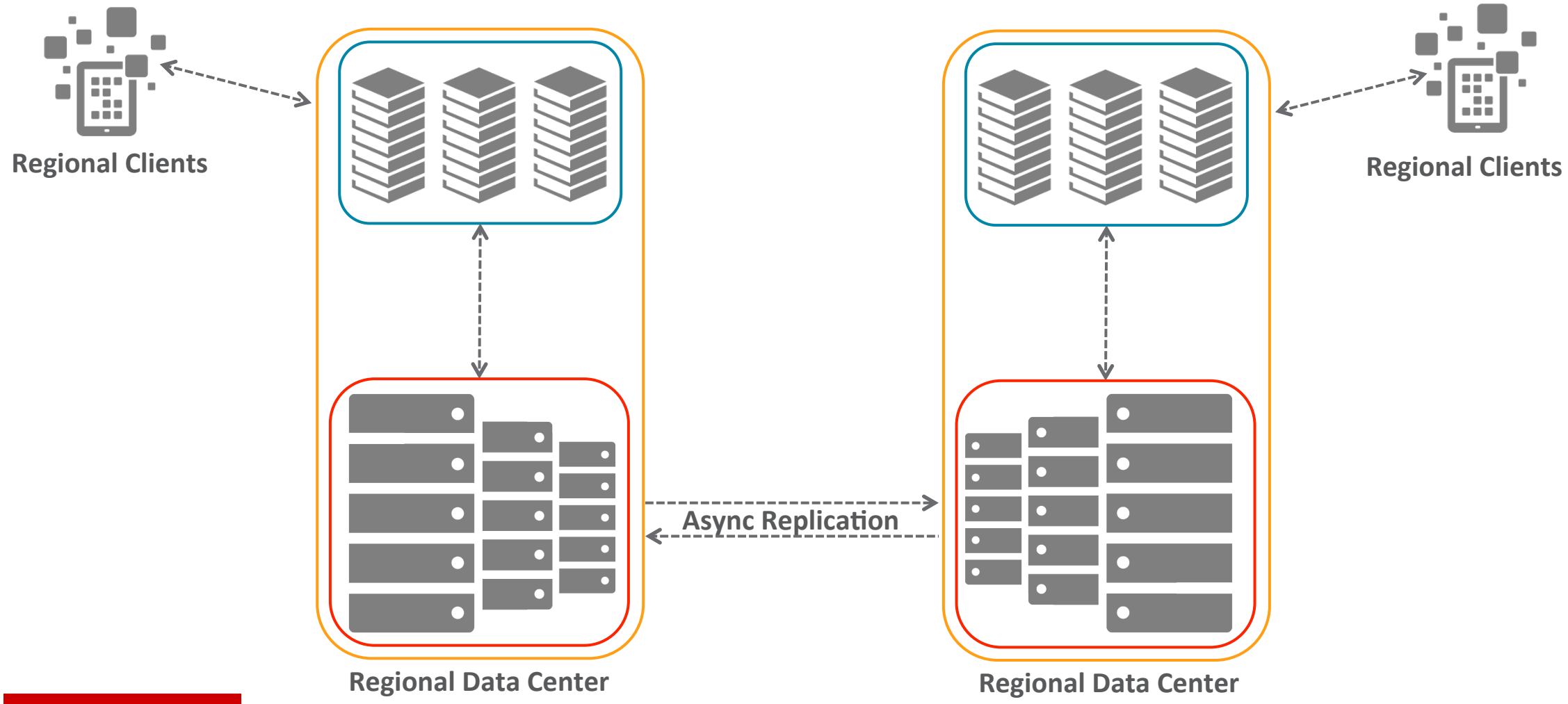
Shared Nothing Cluster – Cross Data Center



Geographically Redundant Cluster



Active/Active Multi-Data Center Setup

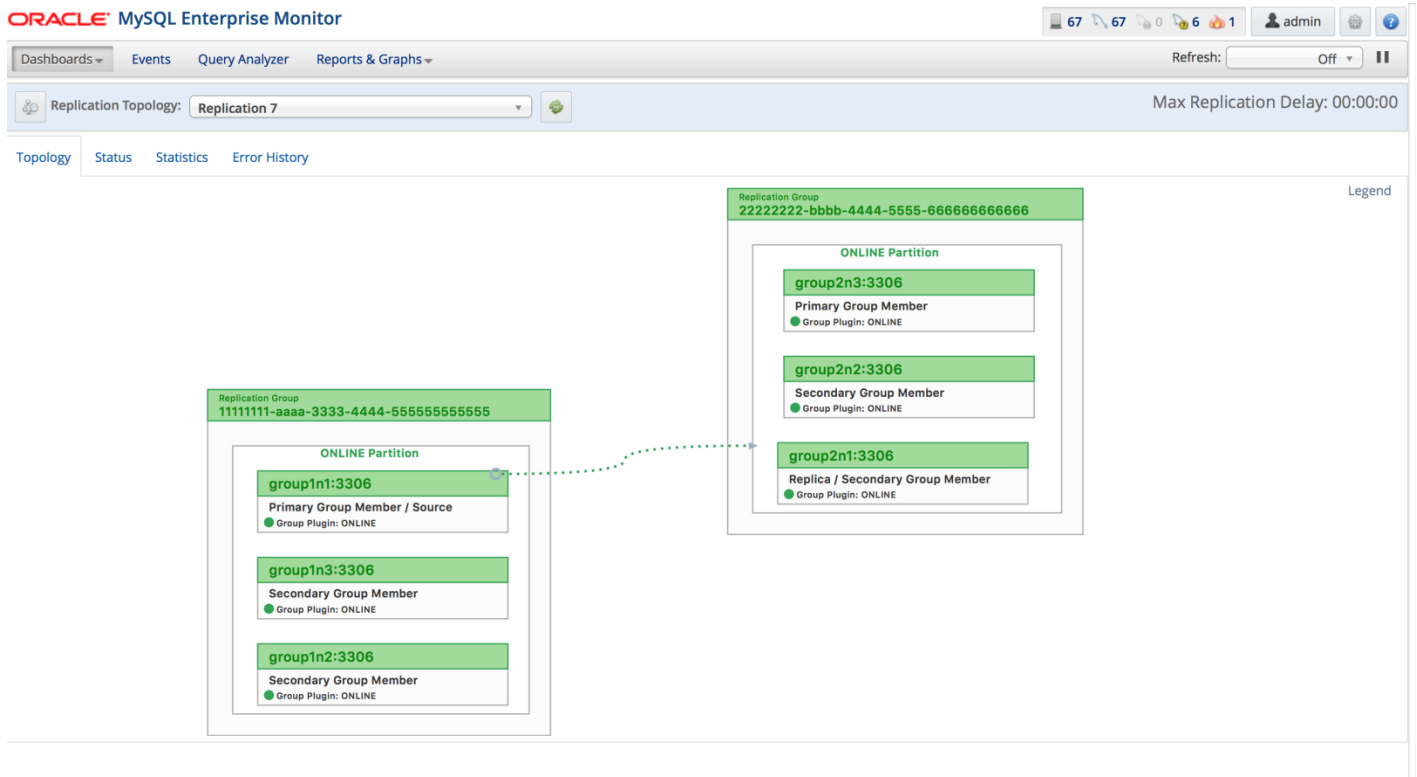


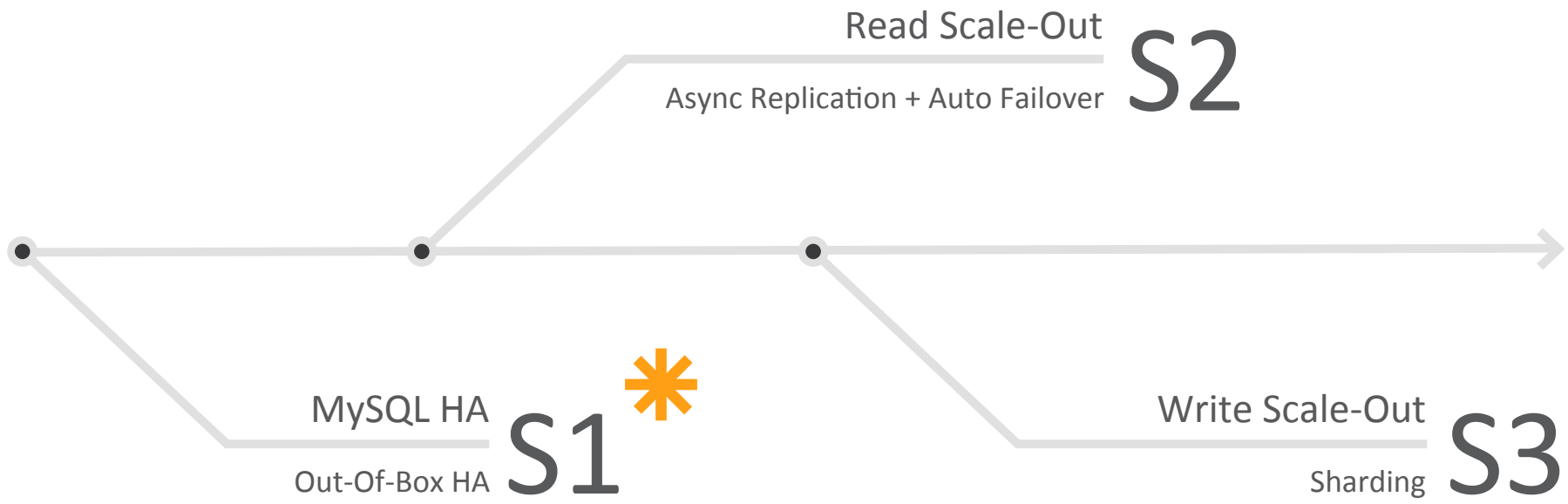
Program Agenda

1. An Introduction to InnoDB Clusters
2. The Components
3. Setup, Monitoring, and Management
4. Reference Architectures
5. What's Next?

MySQL Enterprise Monitor

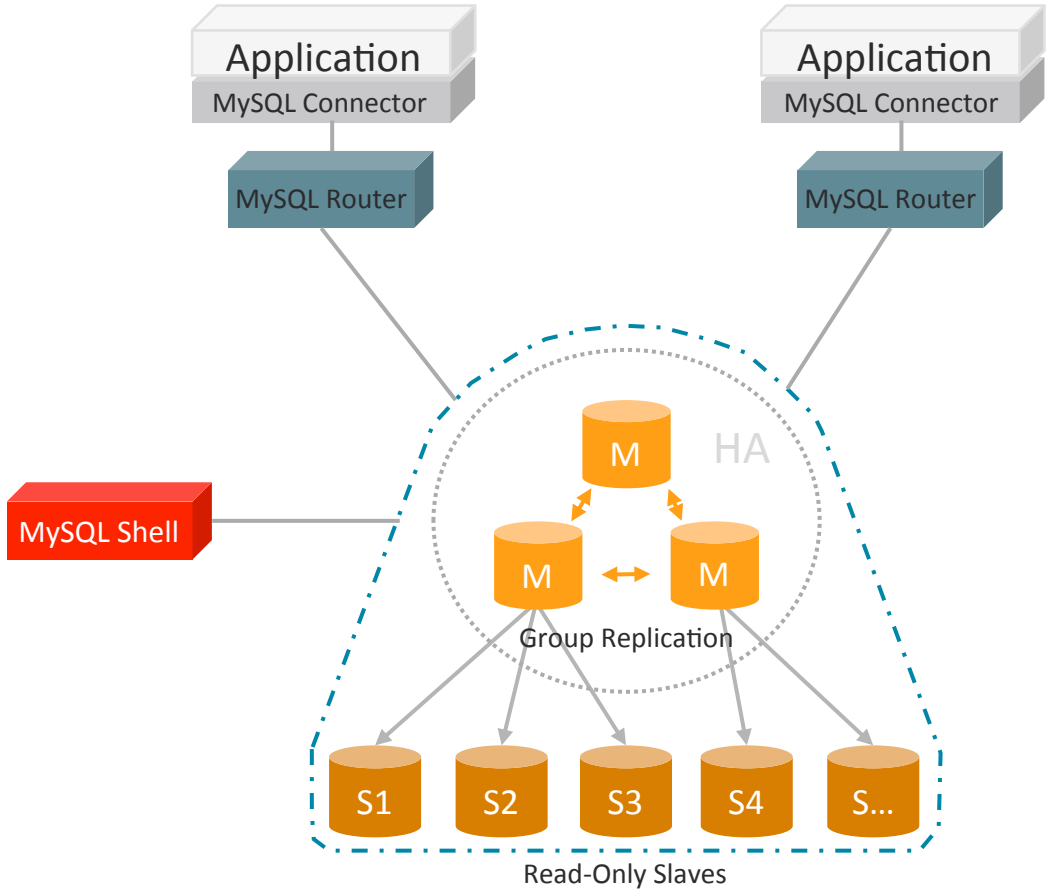
- Native holistic support for Group Replication / InnoDB clusters (GA in 3.4!)
  - Topology views
  - Detailed metrics and graphs
  - Best Practice advice
- Monitoring of MySQL Routers



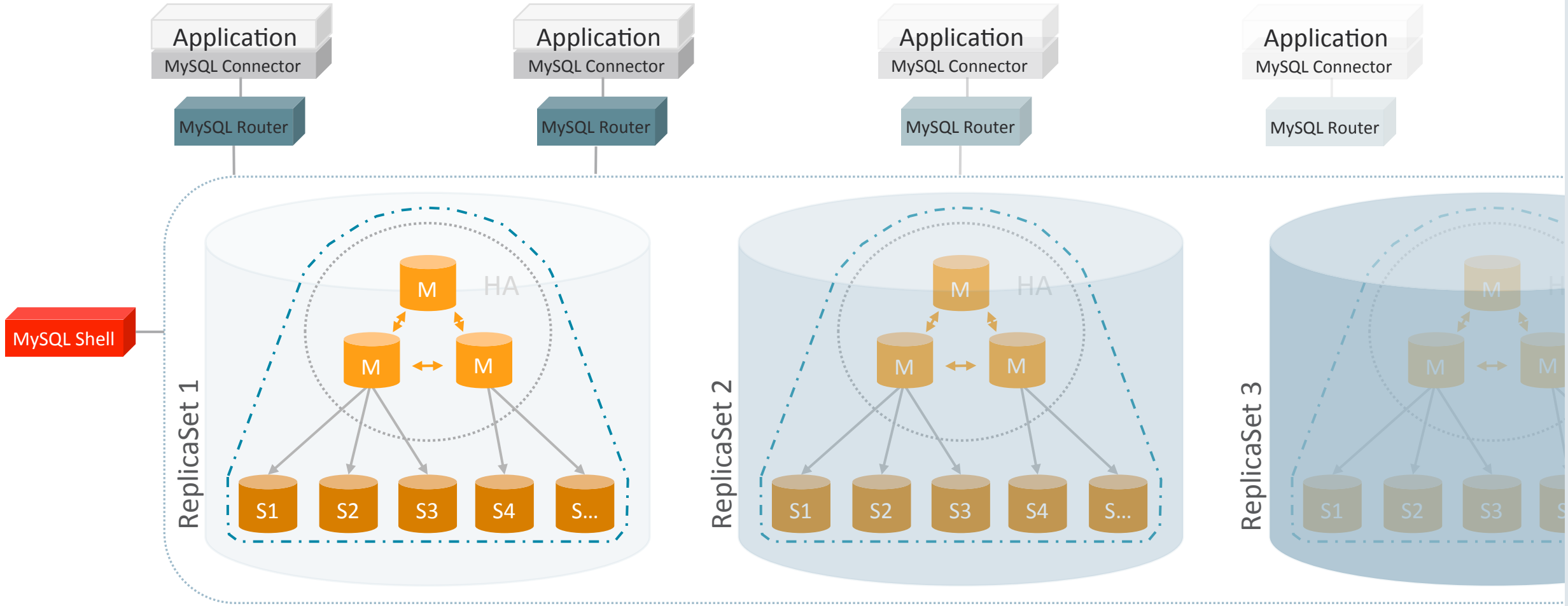


* InnoDB Cluster Now GA!

MySQL InnoDB Cluster: Architecture – Step 2



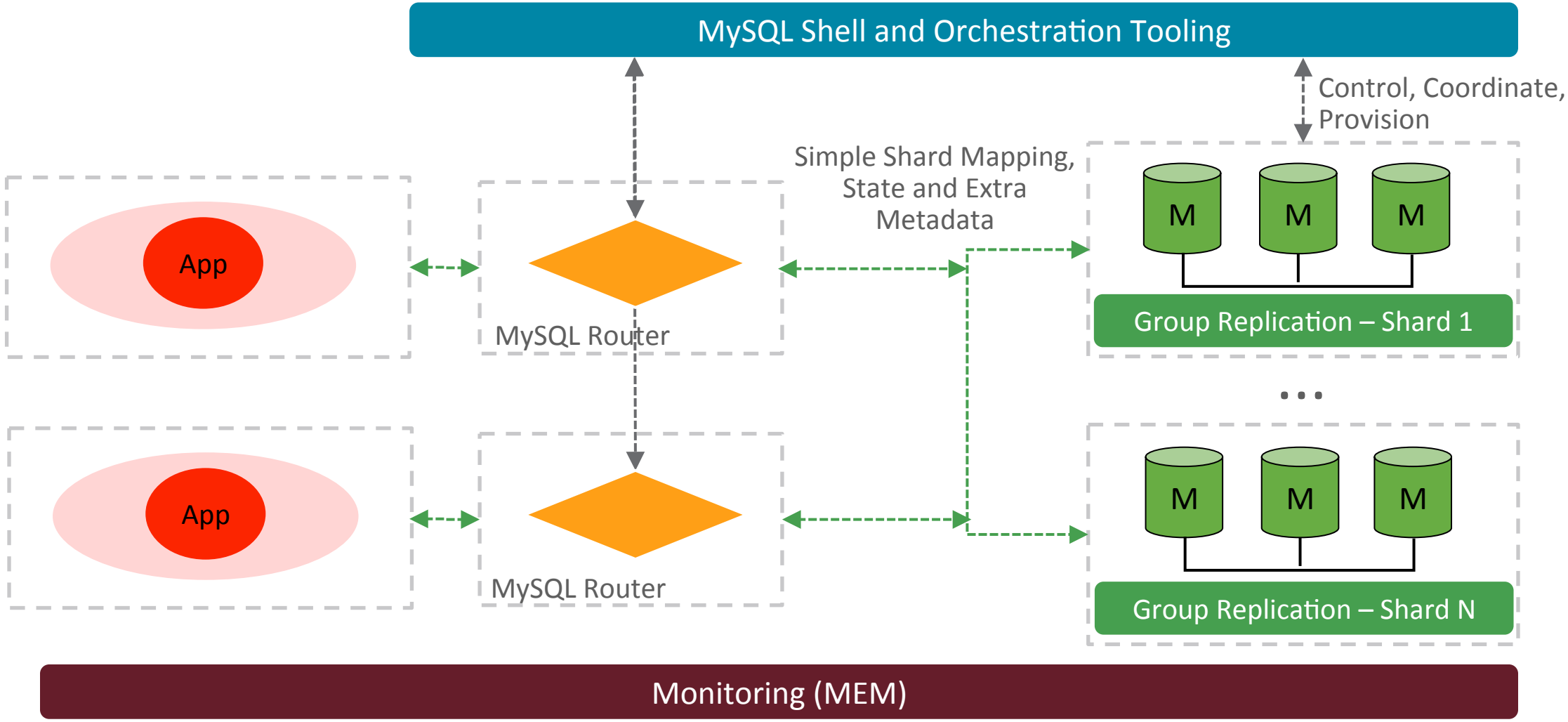
MySQL InnoDB Cluster: Architecture – Step 3



Sharded InnoDB Clusters

- Group Replication
  - Each shard is a highly available replica set
- MySQL Router
  - Manages shard mappings and related metadata
  - Manages client routing
  - Provides cross shard execution framework
    - On top of distributed query execution facilities present in Server
- MySQL Shell
  - Exposes management and orchestration features

MySQL InnoDB Cluster: The End Goal



Oracle Cloud: MySQL Cloud Service

- MySQL Enterprise Edition
- Web based console to manage your MySQL Cloud instances
- Self-Service Provisioning
- Elastic Scalability
- Multi-Layered Security
- Unified Cloud Management
- Oracle PaaS and IaaS Integration
- Premier Technical Support included

Integrated HA and DR service options – utilizing InnoDB clusters – coming soon!


ORACLE®