MySQL InnoDB Cluster

Easiest Tutorial !

Kenny Gryp
MySQL Product Manager
InnoDB & HA

Frédéric Descamps
MySQL Community Manager
EMEA & APAC

Safe Harbor

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, timing and pricing of any features or functionality described for Oracle´s product may change and remains at the sole discretion of Oracle Corporation.

Statement in this presentation relating to Oracle´s future plans, expectations, beliefs, intentions and ptospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle´s Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors". These filings are available on the SEC´s website or on Oracle´s website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

about us

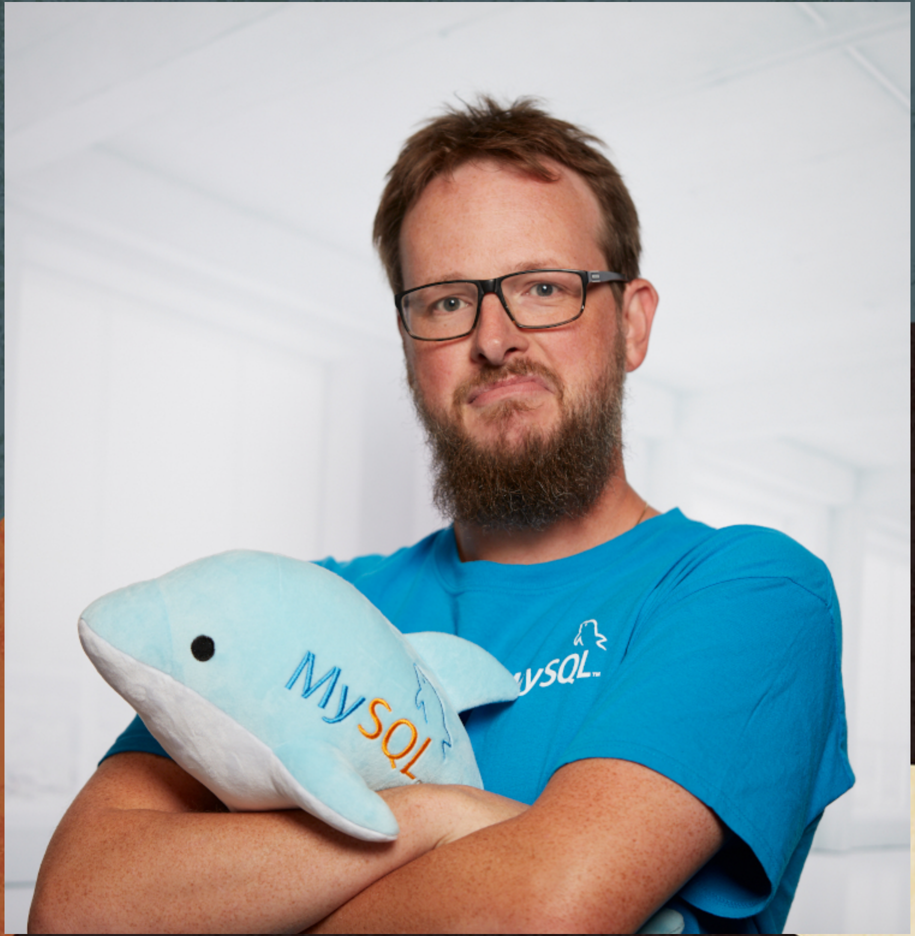
Who are we ?

Kenny Gryp

- @gryp
- MySQL Product Manager (InnoDB & HA)
- born in Belgium 🇧🇪

Frédéric Descamps

- @lefred
- MySQL Evangelist
- Managing MySQL since 3.23
- devops believer
- living in Belgium 🇧🇪
- https://lefred.be

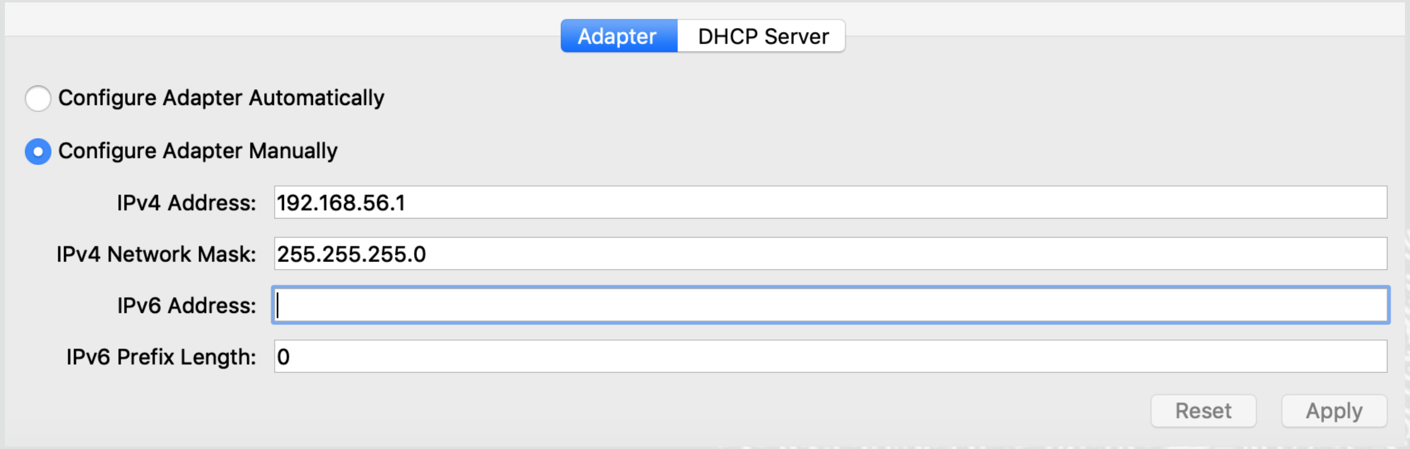


VirtualBox

Setup your workstation

Setup your workstation

- Install VirtualBox 6.0
- On the USB key, **COPY** `MySQLInnoDBCluster_PLEU19.ova` and click on it
- Ensure you have `vboxnet0` network interface
  - Older VirtualBox: VirtualBox Pref. -> Network -> Host-Only Networks -> **+**
  - New VirtualBox: Global Tools -> Host Network Manager -> **+**



( ) Configure Adapter Automatically
(●) Configure Adapter Manually

IPv4 Address:	192.168.56.1
IPv4 Network Mask:	255.255.255.0
IPv6 Address:	
IPv6 Prefix Length:	0

Login on your workstation

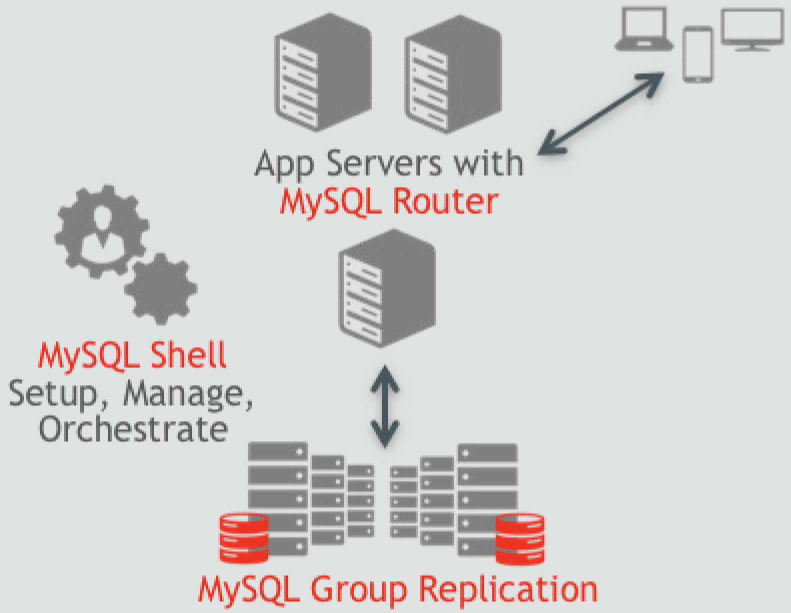
- Start all virtual machines (`mysql1`, `mysql2` & `mysql3`)
- Try to connect to all VM´s from your terminal or putty *(root password is **X**)* :
  - `ssh root@192.168.56.11` *mysql1*
  - `ssh root@192.168.56.12` *mysql2*
  - `ssh root@192.168.56.13` *mysql3*

MySQL InnoDB Cluster

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

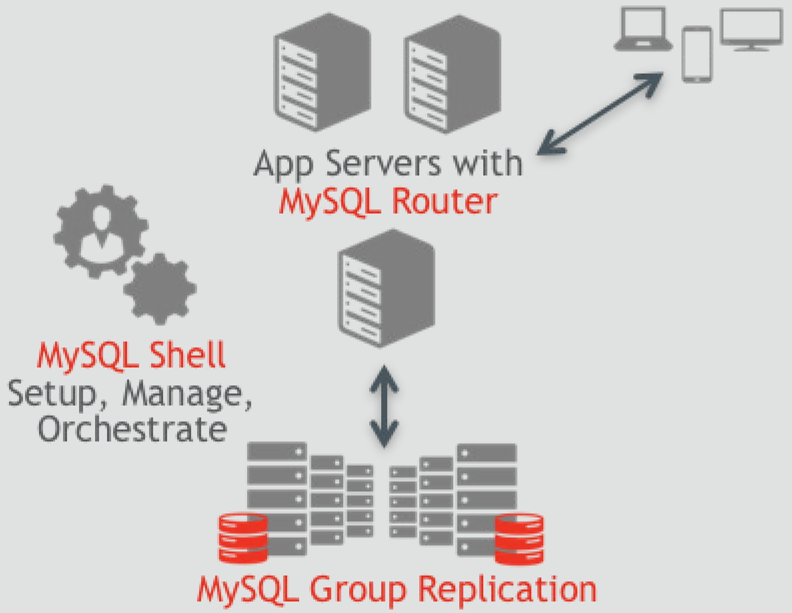
MySQL InnoDB Cluster

"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

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



Components:

- MySQL Server
- MySQL Group Replication
- MySQL Shell
- MySQL Router

MySQL InnoDB Cluster - Goals

One Product: MySQL

- All components developed together
- Integration of all components
- Full stack testing

MySQL InnoDB Cluster - Goals

One Product: MySQL

- All components developed together
- Integration of all components
- Full stack testing

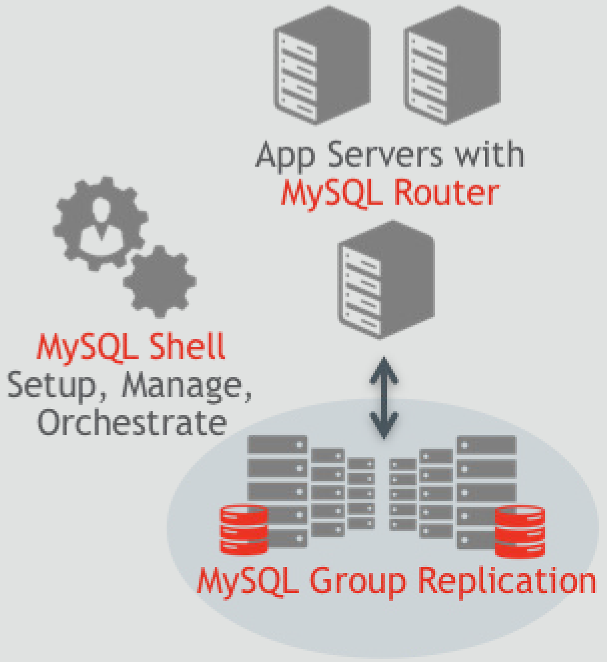
Easy to Use

- One client: MySQL Shell
- Integrated orchestration
- Homogenous servers

MySQL Group Replication

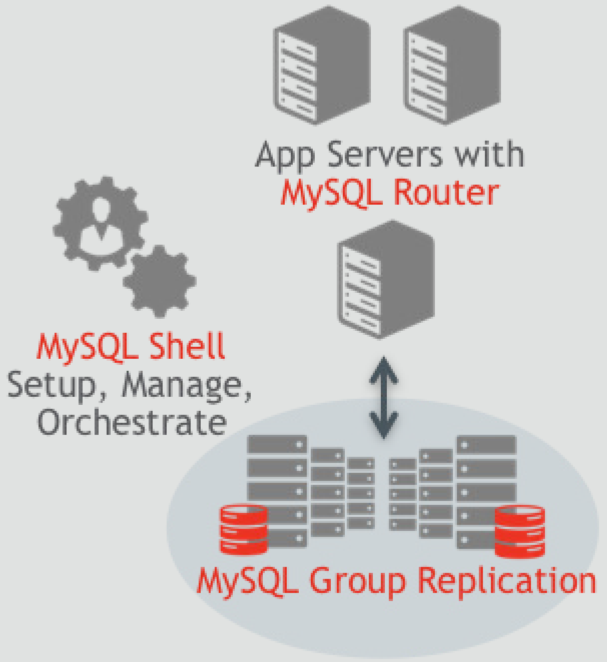
MySQL Group Replication

Highly Available **Distributed** MySQL Database



MySQL Group Replication

Highly Available **Distributed** MySQL Database



- Open Source
- Fault tolerance
- Automatic failover
- Active/Active update anywhere
- Automatic membership management
  - Adding/removing members
  - Network partitions, failures
- Conflict detection and resolution
- Consistency!

MySQL Group Replication

- Implementation of Replicated Database State Machine
  - Total Order - Writes
  - XCOM - Paxos implementation
- 8.0+: Member provisioning using `CLONE` plugin.
- Configurable Consistency Guarantees
  - eventual consistency
  - 8.0+: per session & global read/write consistency
- Using MySQL replication framework by design
  - GTIDs, binary logs, relay logs
- Generally Available since MySQL 5.7
- Supported on all platforms: linux, windows, solaris, macosx, freebsd

MySQL Group Replication - Use Cases

Consistency: No Data Loss (RPO=0)

- in event of failure of (primary) member
- Split brain prevention (Quorum)

MySQL Group Replication - Use Cases

Consistency: No Data Loss (RPO=0)

- in event of failure of (primary) member
- Split brain prevention (Quorum)

Highly Available: Automatic Failover

- Primary members are automatically elected
- Automatic Network Partition handling

MySQL Group Replication - Use Cases

Consistency: No Data Loss (RPO=0)

- in event of failure of (primary) member
- Split brain prevention (Quorum)

Highly Available: Automatic Failover

- Primary members are automatically elected
- Automatic Network Partition handling

Read Scaleout

- Add/Remove members as needed
- Replication Lag handling with Flow Control
- Configurable Consistency Levels
  - Eventual
  - Full Consistency -- no stale reads

MySQL Group Replication - Use Cases

Consistency: No Data Loss (RPO=0)

- in event of failure of (primary) member
- Split brain prevention (Quorum)

Highly Available: Automatic Failover

- Primary members are automatically elected
- Automatic Network Partition handling

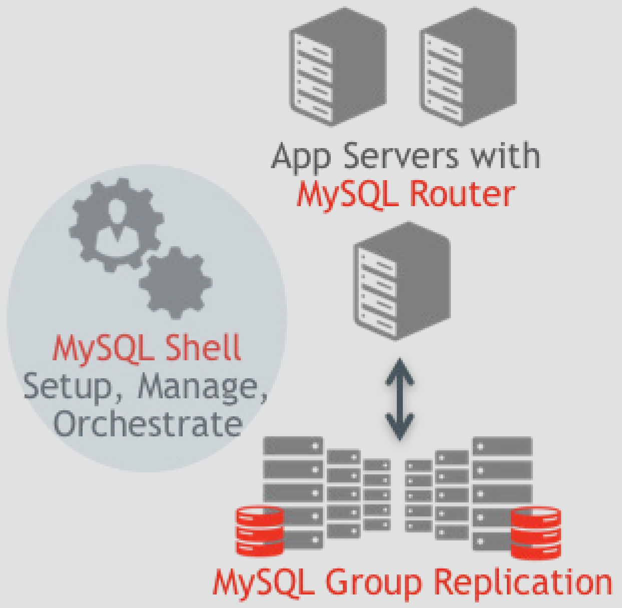
Read Scaleout

- Add/Remove members as needed
- Replication Lag handling with Flow Control
- Configurable Consistency Levels
  - Eventual
  - Full Consistency -- no stale reads

Active/Active environments

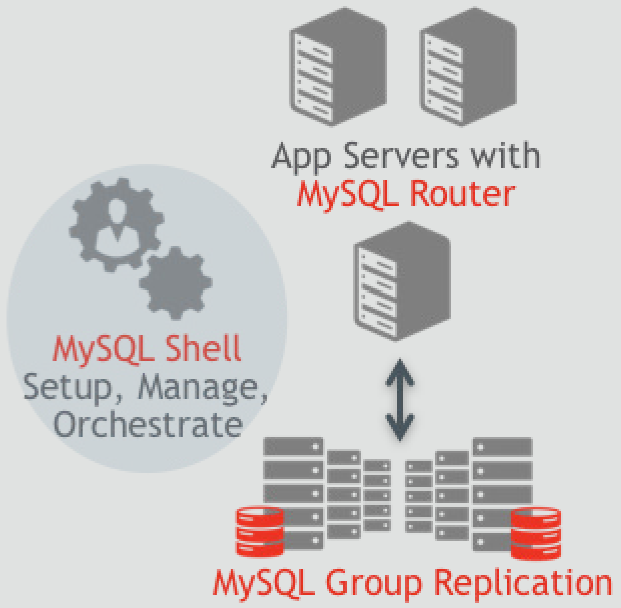
- Write to many members at the same time
  - ordered writes within the group (XCOM)
  - guaranteed consistency
- Good write performance
  - due to Optimistic Locking (workload dependent)

MySQL Shell : Database Admin Interface



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

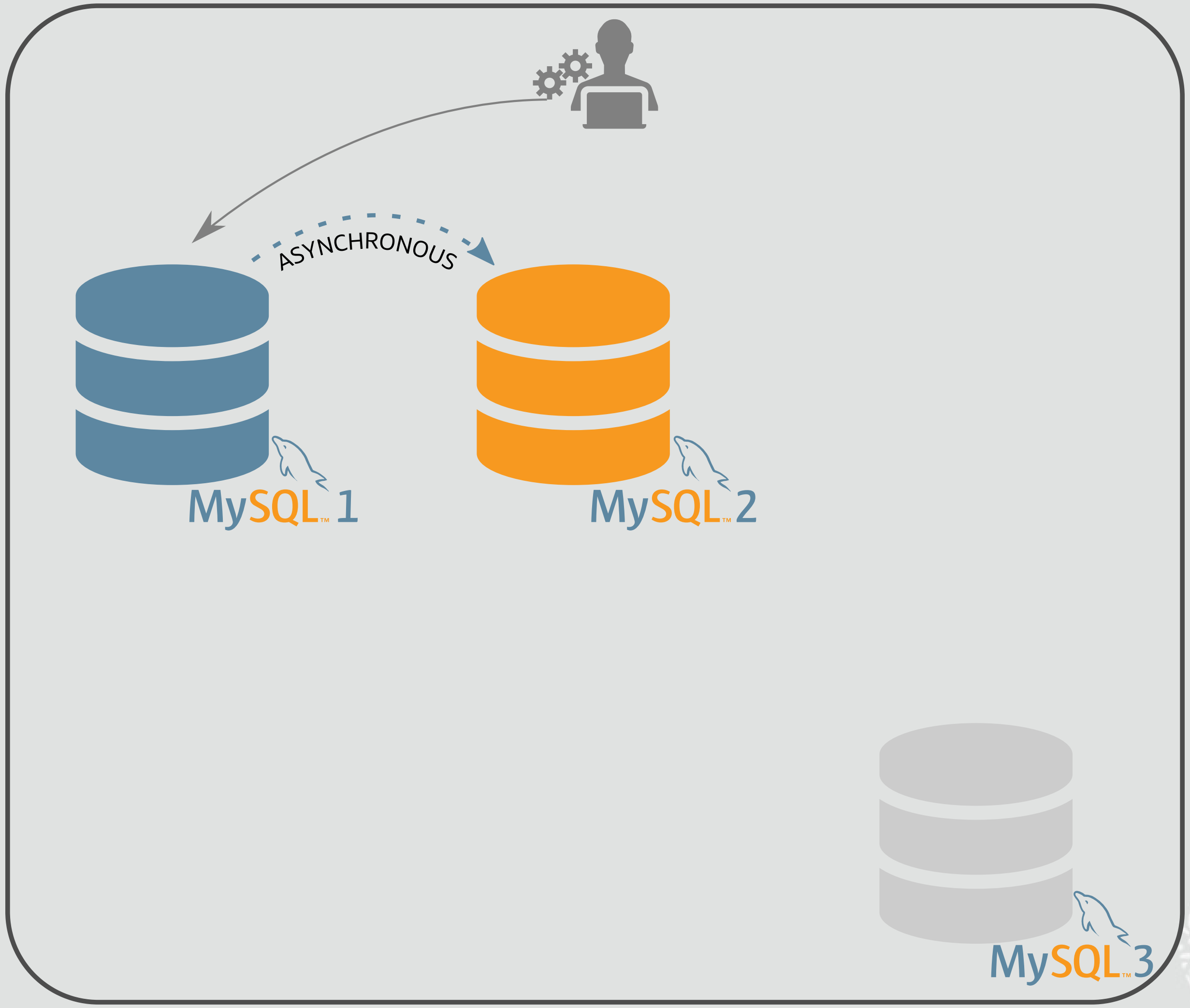
MySQL Shell : Database Admin Interface



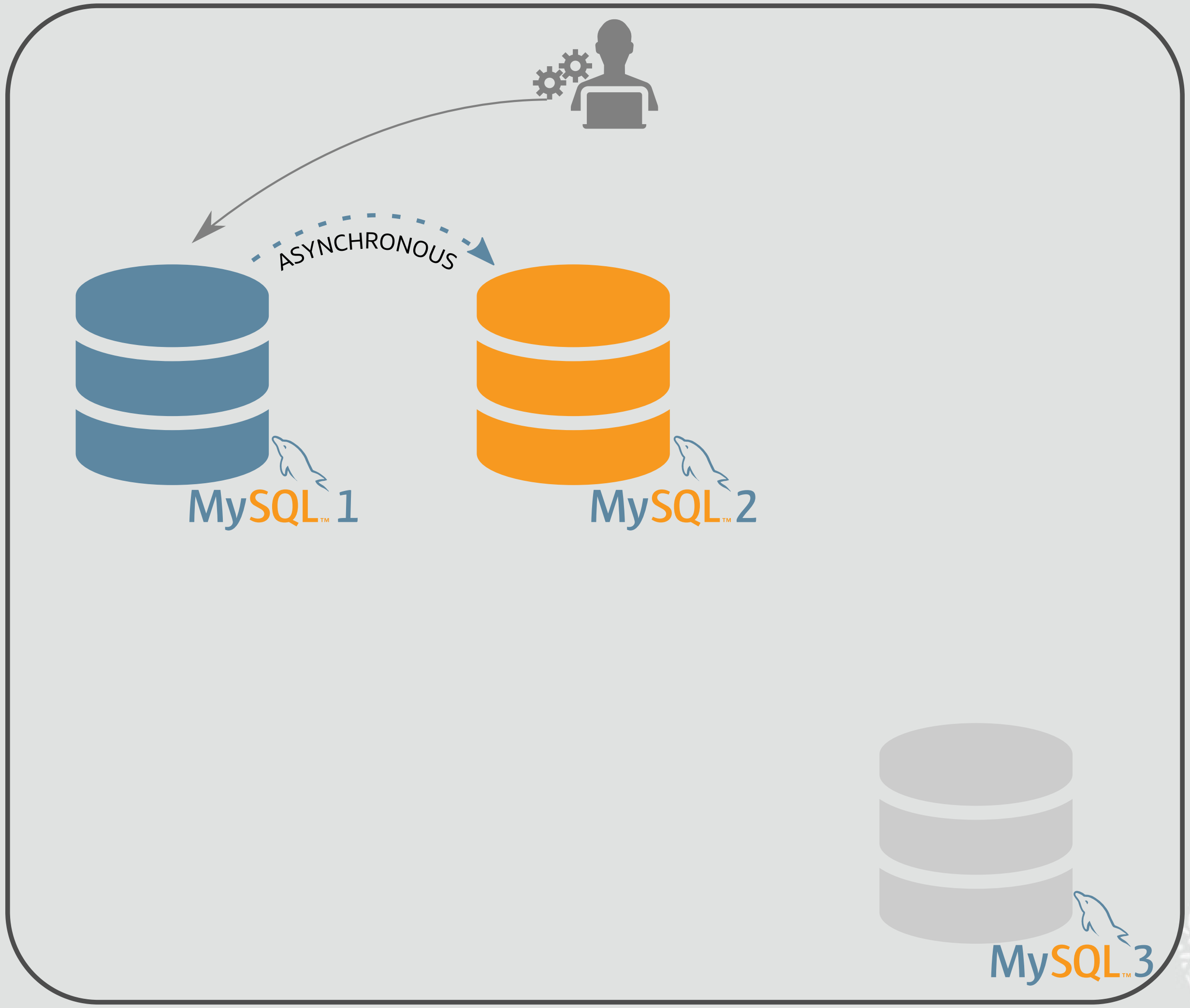
"MySQL Shell provides the developer and DBA with a single intuitive, flexible, and powerful interface for all MySQL related tasks!"

- Open Source
- Multi-Language: JavaScript, Python, and SQL
- Naturally scriptable
- Supports Document and Relational models
- Exposes full Development and Admin API
- Classic MySQL protocol and X protocol

LAB1: Current situation

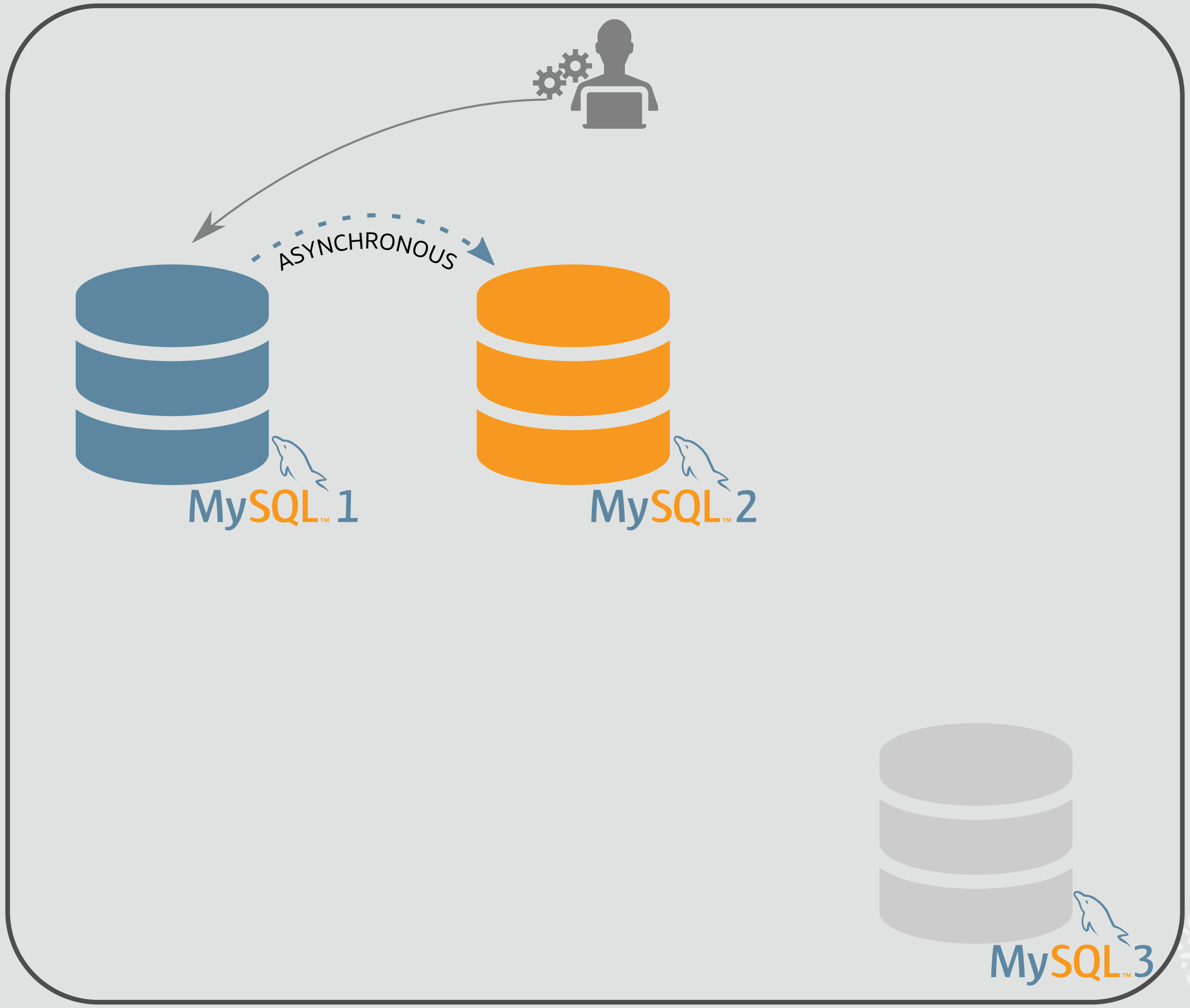


LAB1: Current situation



all database servers are using MySQL 8.0.17

LAB1: Current situation



- launch **`run_app.sh`** on **`mysql1`** into a **`screen`** session
- verify that **`mysql2`** is a running replica

Summary

	ROLE	INTERNAL IP
`mysql1`	master / app	`192.168.56.11`
`mysql2`	replica	`192.168.56.12`
`mysql3`	n/a	`192.168.56.13`

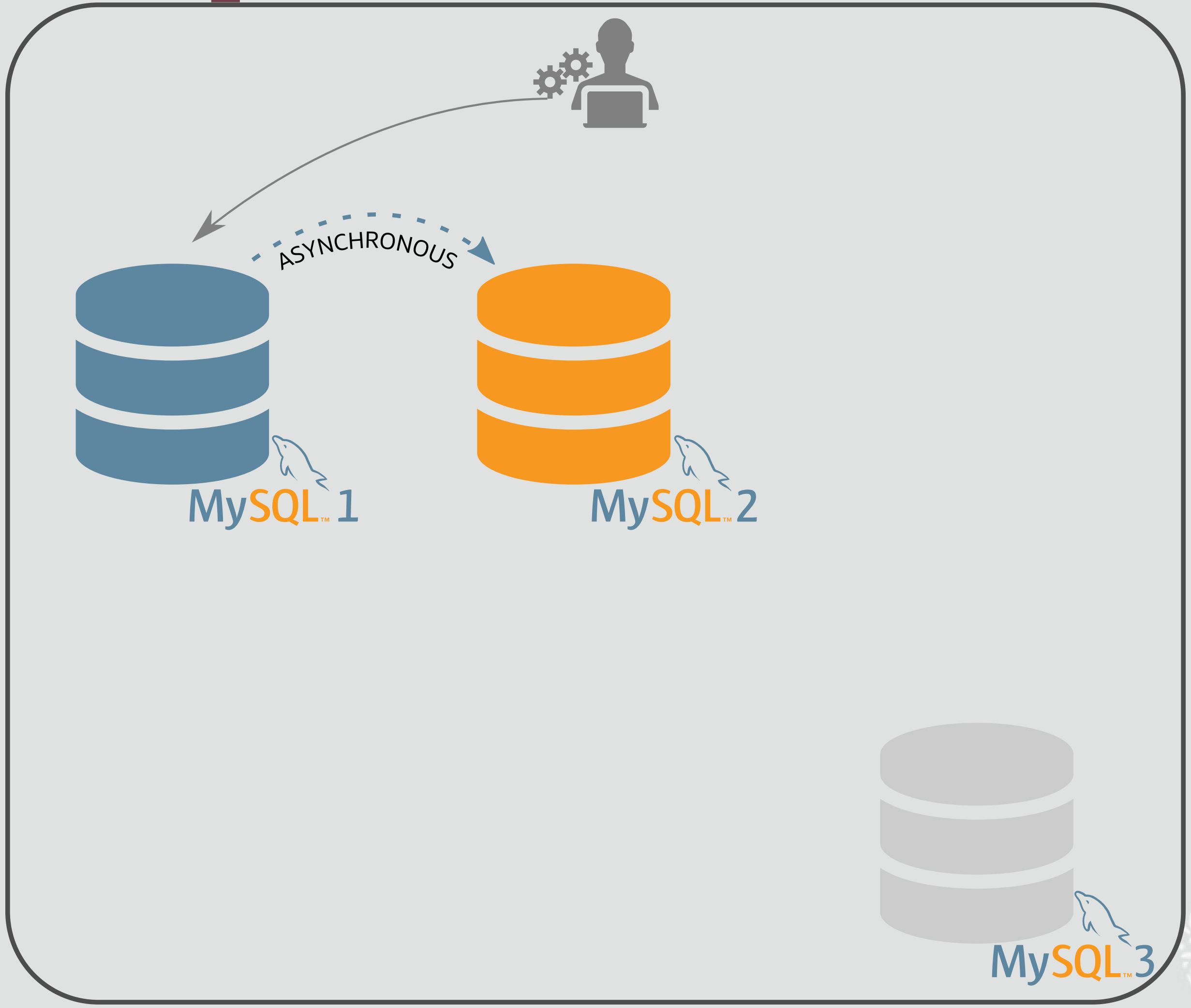
Summary

	ROLE	INTERNAL IP
`mysql1`	master / app	`192.168.56.11`
`mysql2`	replica	`192.168.56.12`
`mysql3`	n/a	`192.168.56.13`

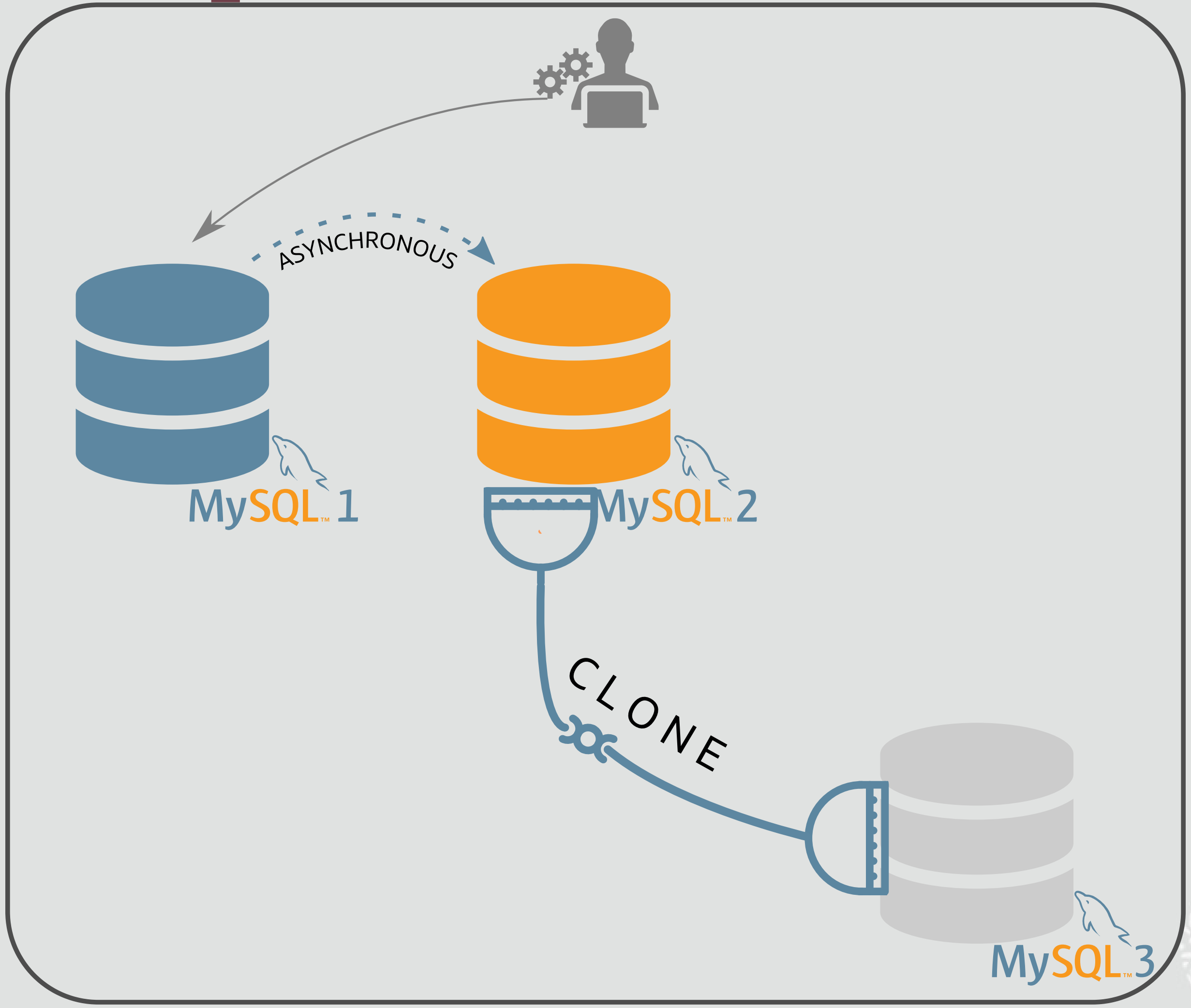
write this down somewhere !

Migrating
from Asynchronous Replication to Group Replication

The plan



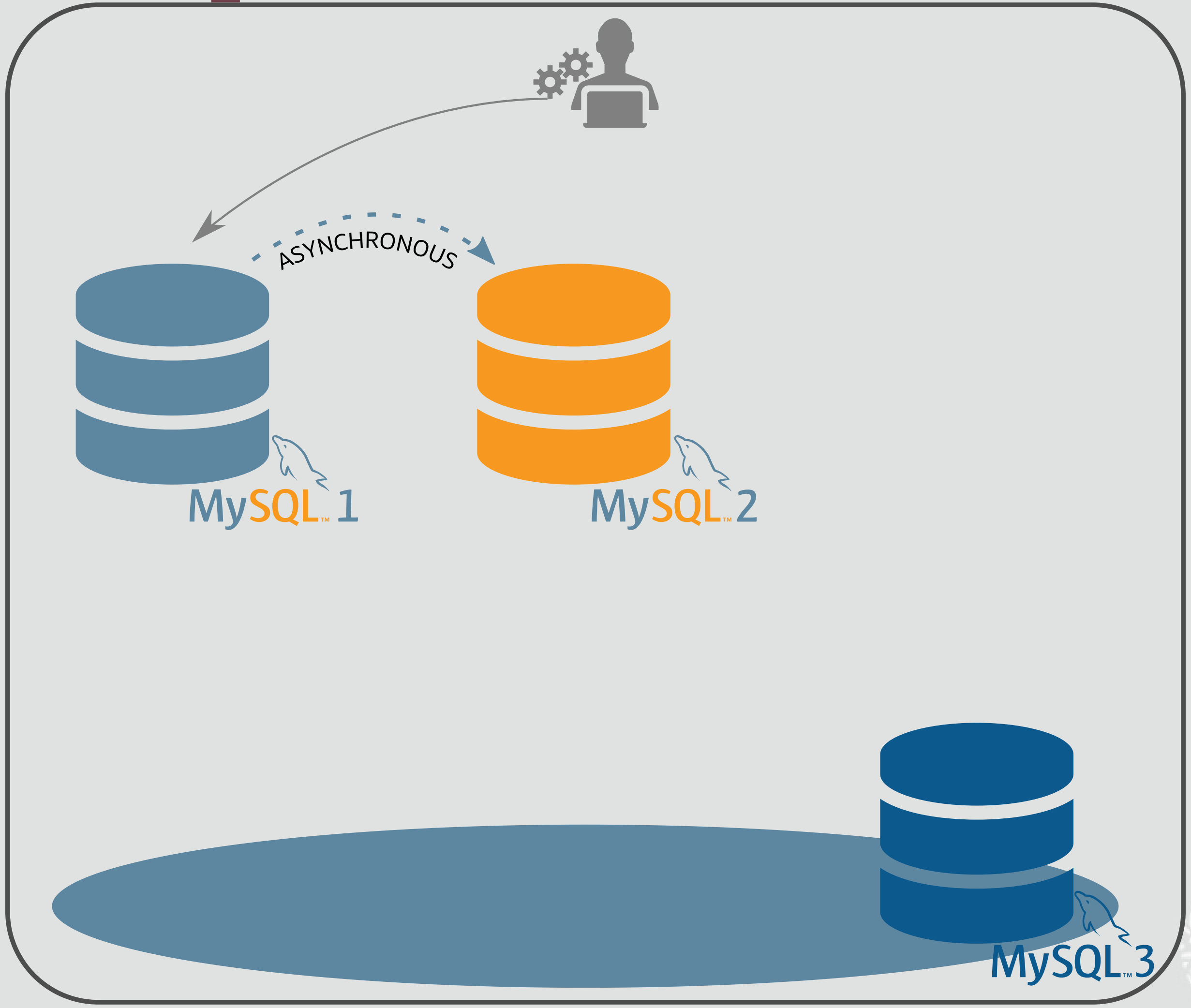
The plan



These are the 7 steps for the migration:

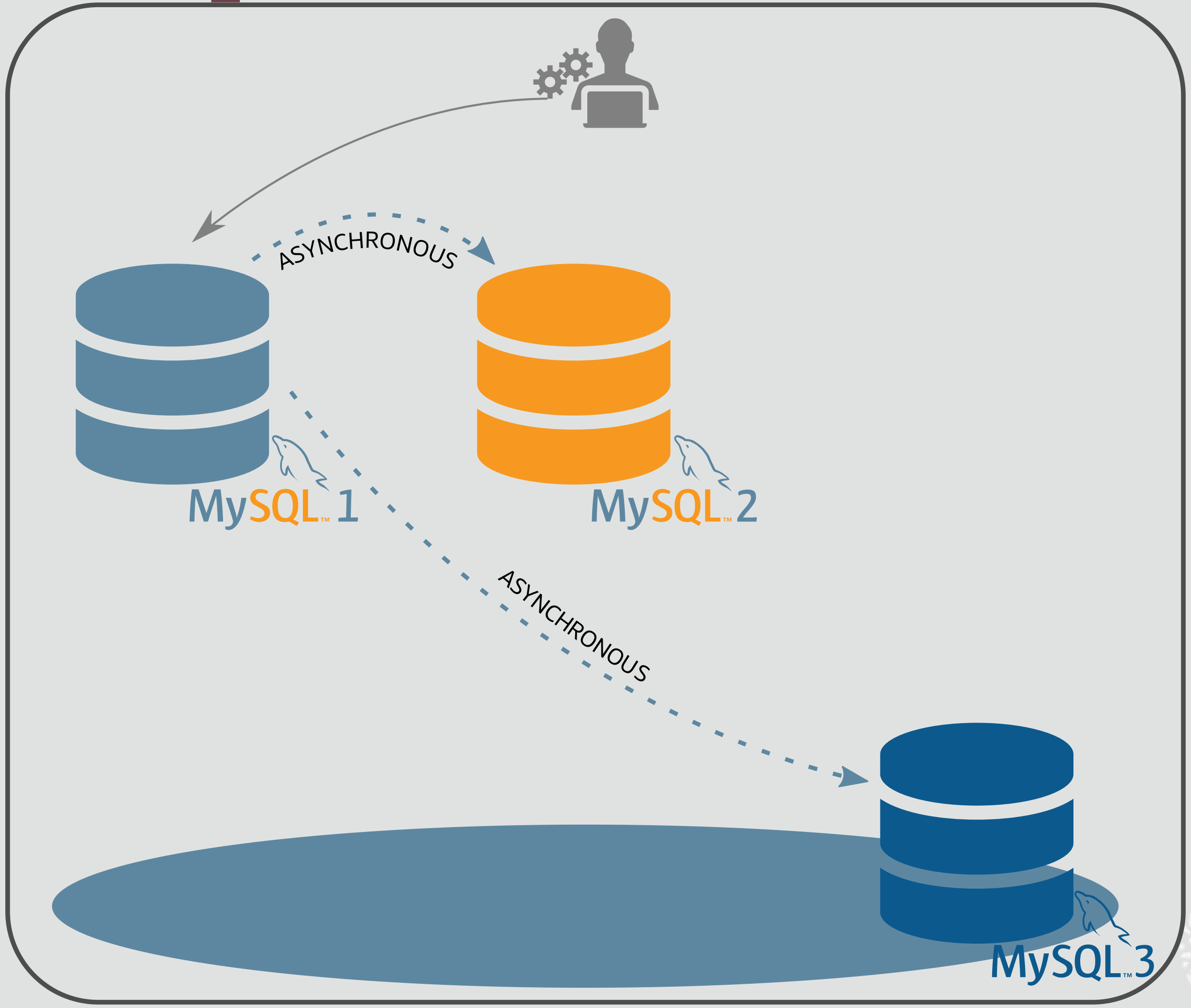
1) We transfer the data from `mysql2` (replica) to our new MySQL Server `mysql3`

The plan



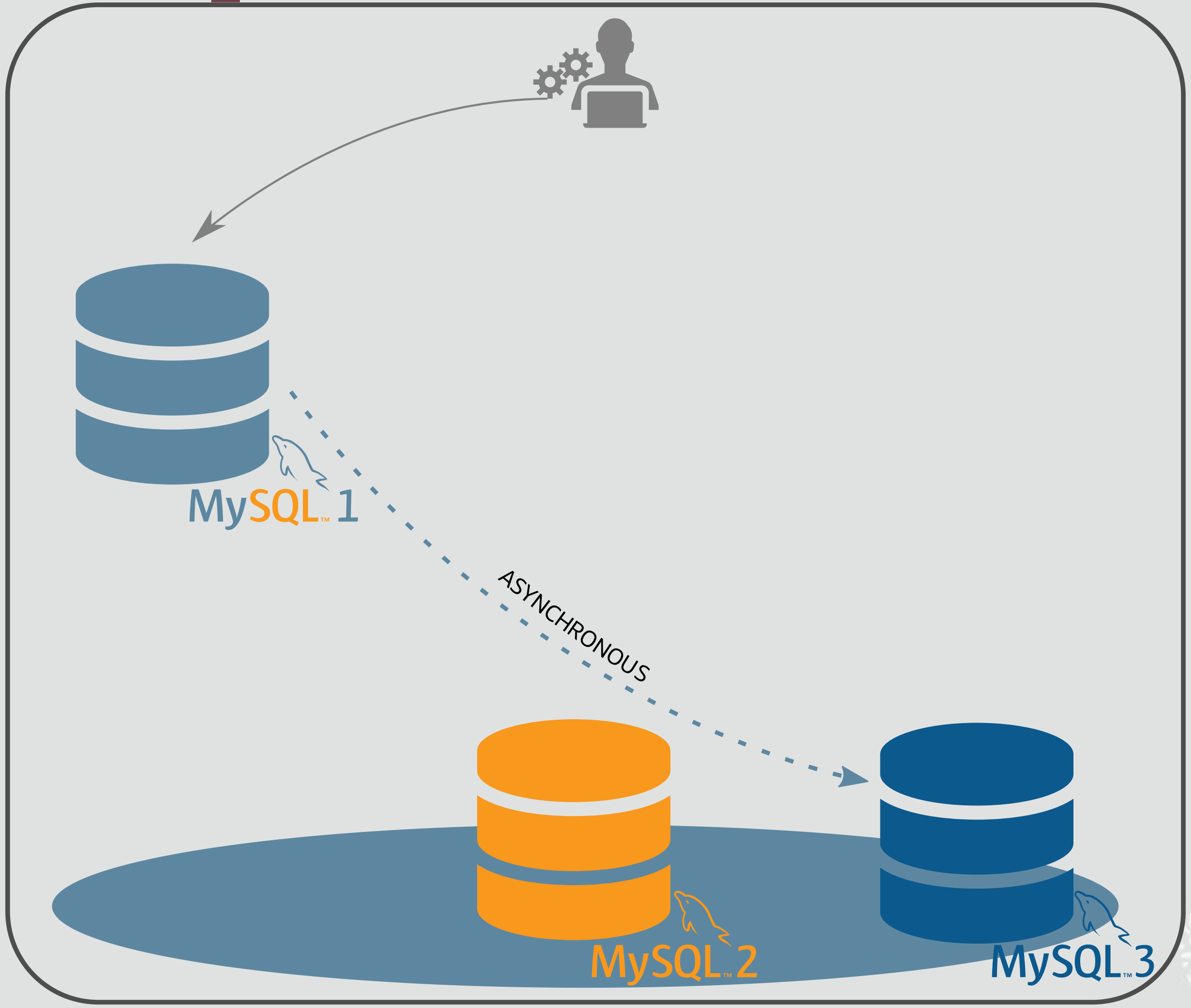
2) We create a MySQL InnoDB Cluster on `mysql3`

The plan



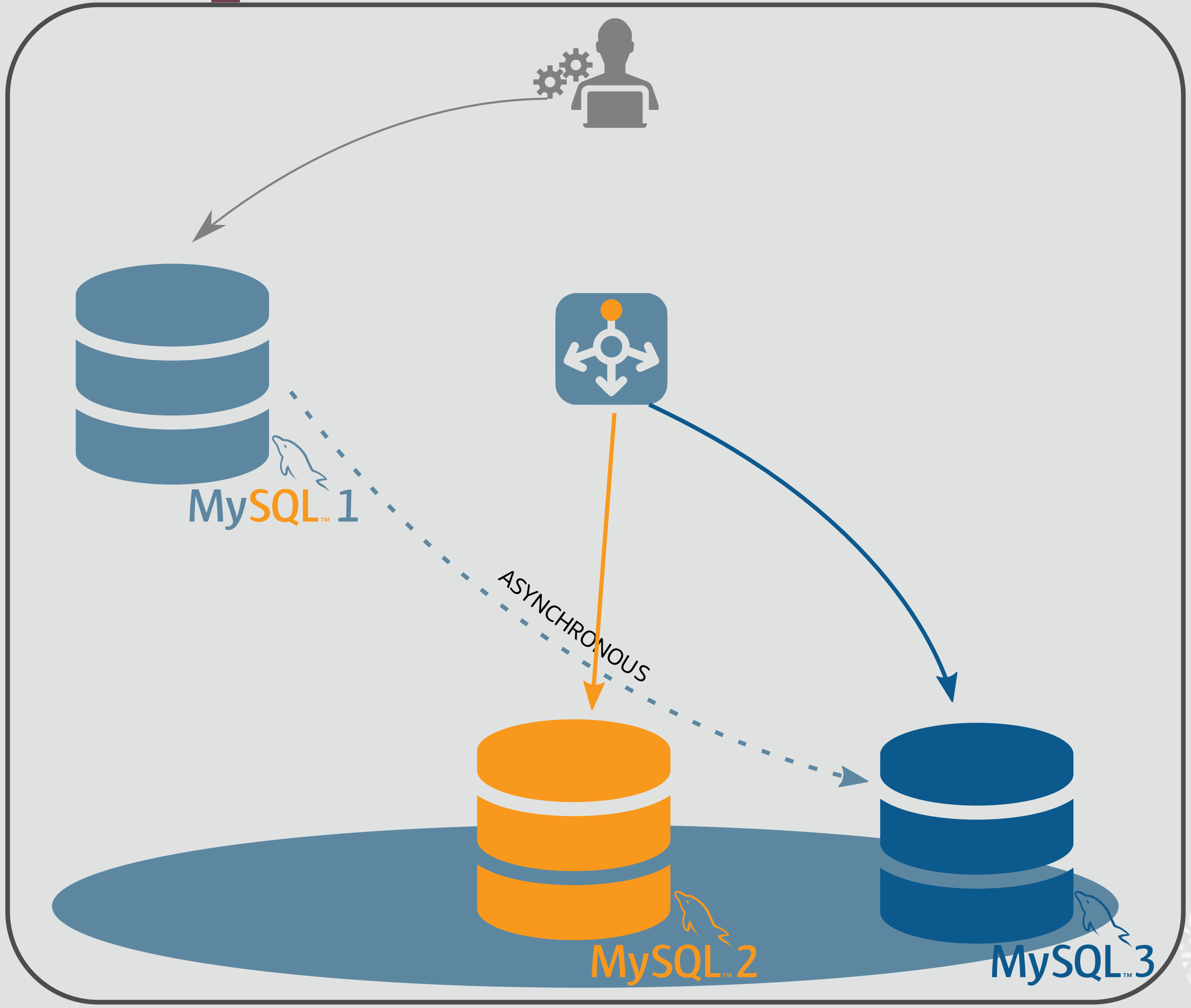
3) We setup our MySQL InnoDB Cluster to become a replica from the production server (`mysql1`).

The plan



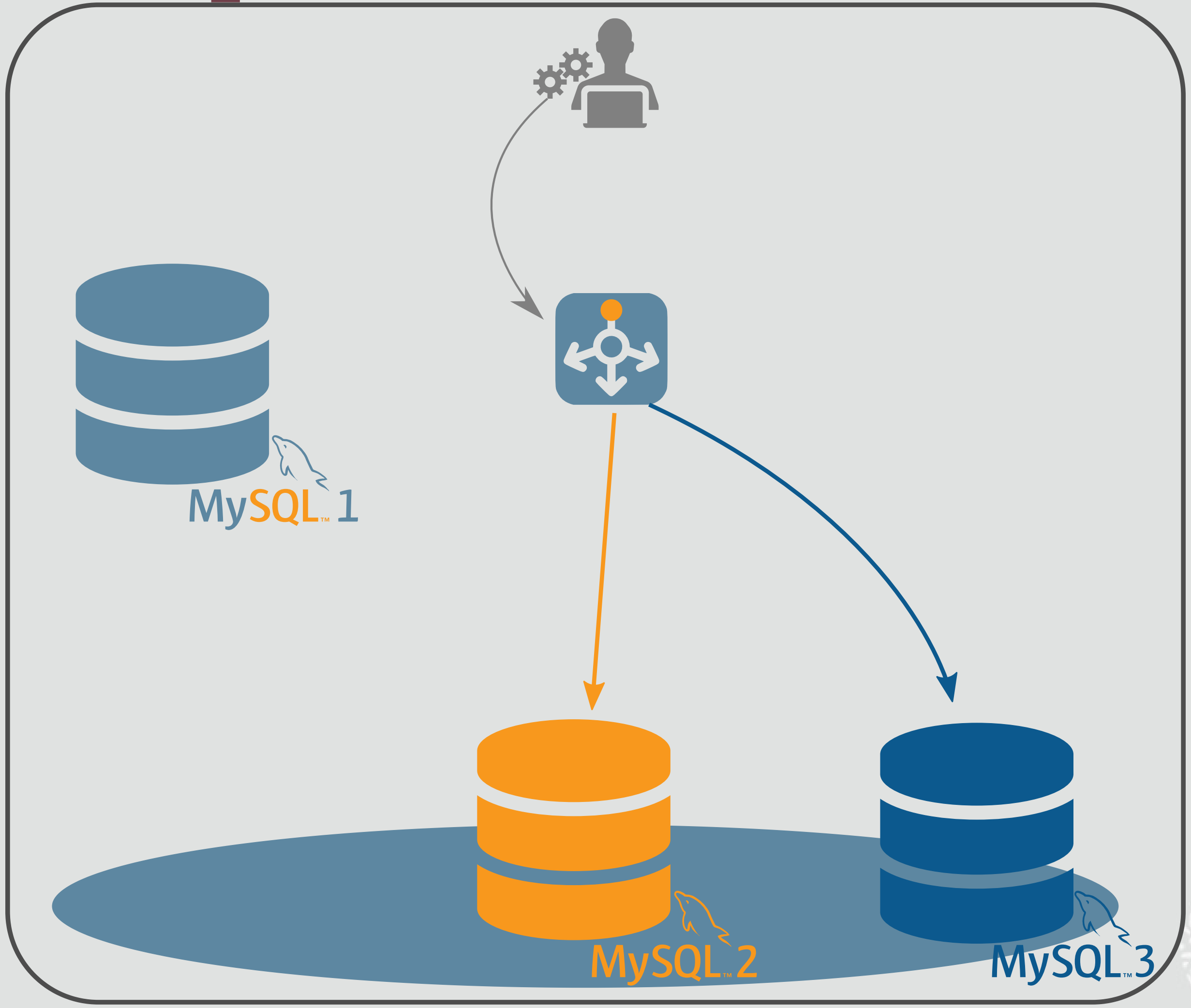
4) We remove `mysql2` as replica of `mysql1` and we add it in the MySQL InnoDB Cluster

The plan



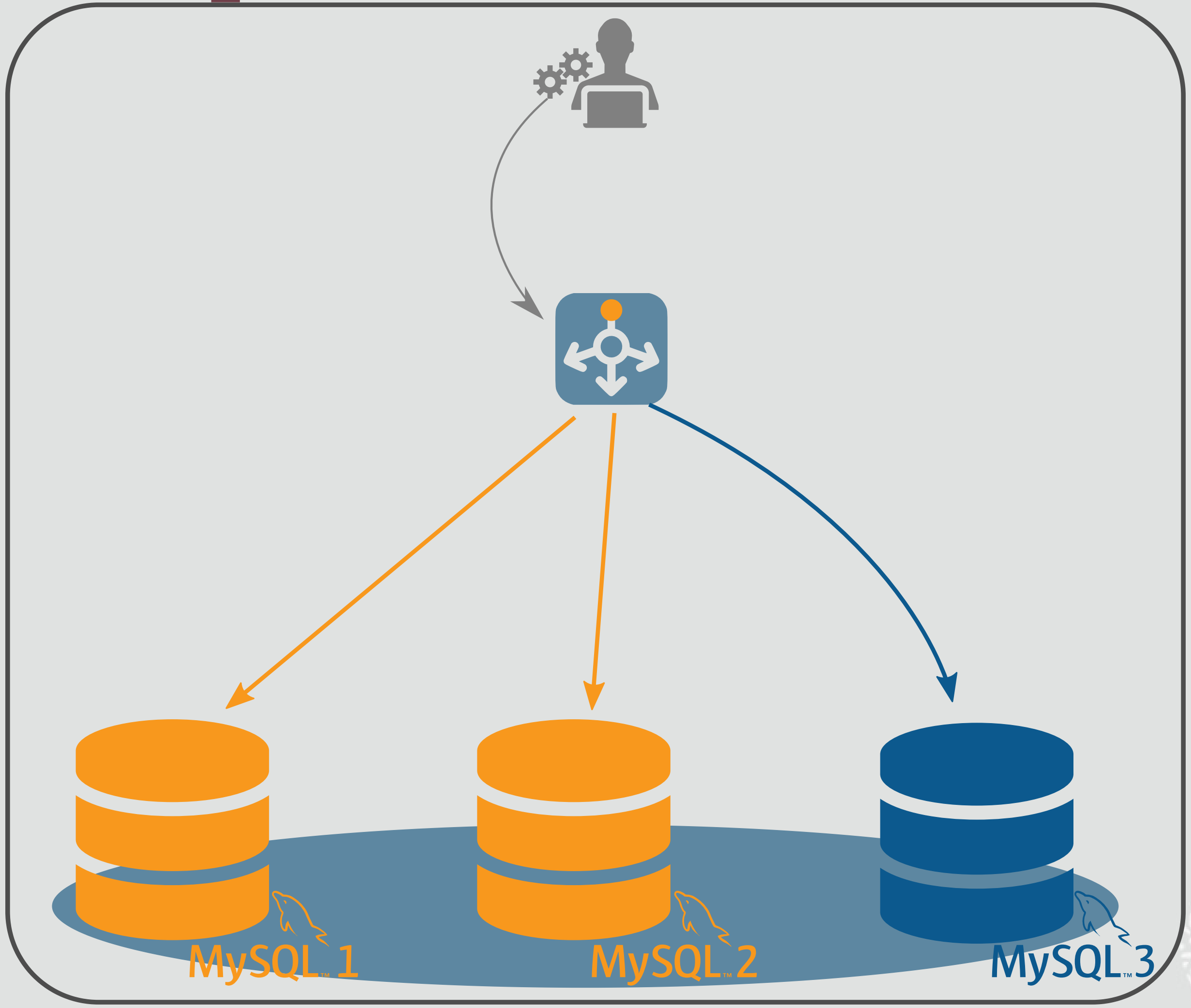
5) We bootstrap and start a MySQL Router (on `mysql1`).

The plan



6) We stop our application and we restart it pointing to the MySQL Router

The plan



7) We add `mysql1` in the MySQL InnoDB Cluster

LAB2: Admin User Creation

We will first create an admin account on `mysql1` and `mysql3` (`mysql2` will get it replicated) that we will use throughout the tutorial:

```
mysql1> CREATE USER clusteradmin@'%' identified by 'mysql';
mysql1> GRANT ALL PRIVILEGES ON *.* TO  clusteradmin@'%'
        WITH GRANT OPTION;
```

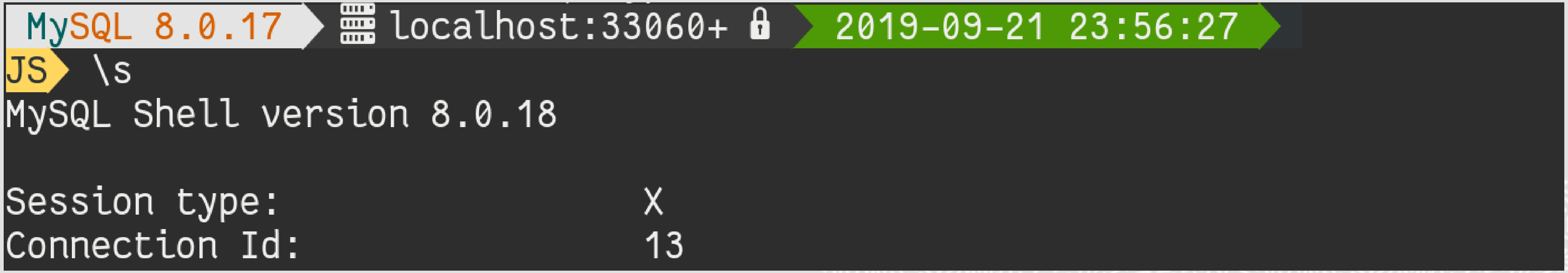
```
mysql3> CREATE USER clusteradmin@'%' identified by 'mysql';
mysql3> GRANT ALL PRIVILEGES ON *.* TO  clusteradmin@'%'
        WITH GRANT OPTION;
```

LAB2: Beautiful MySQL Shell

You can setup a better shell environment :

```
# mysqlsh
mysql-js> shell.options.setPersist('history.autoSave', 1)
mysql-js> shell.options.setPersist('history.maxSize', 5000)
```

```
# cp /usr/share/mysqlsh/prompt/prompt_256pl+aw.json ~/.mysqlsh/prompt.json
# mysqlsh clusteradmin@mysql2
```



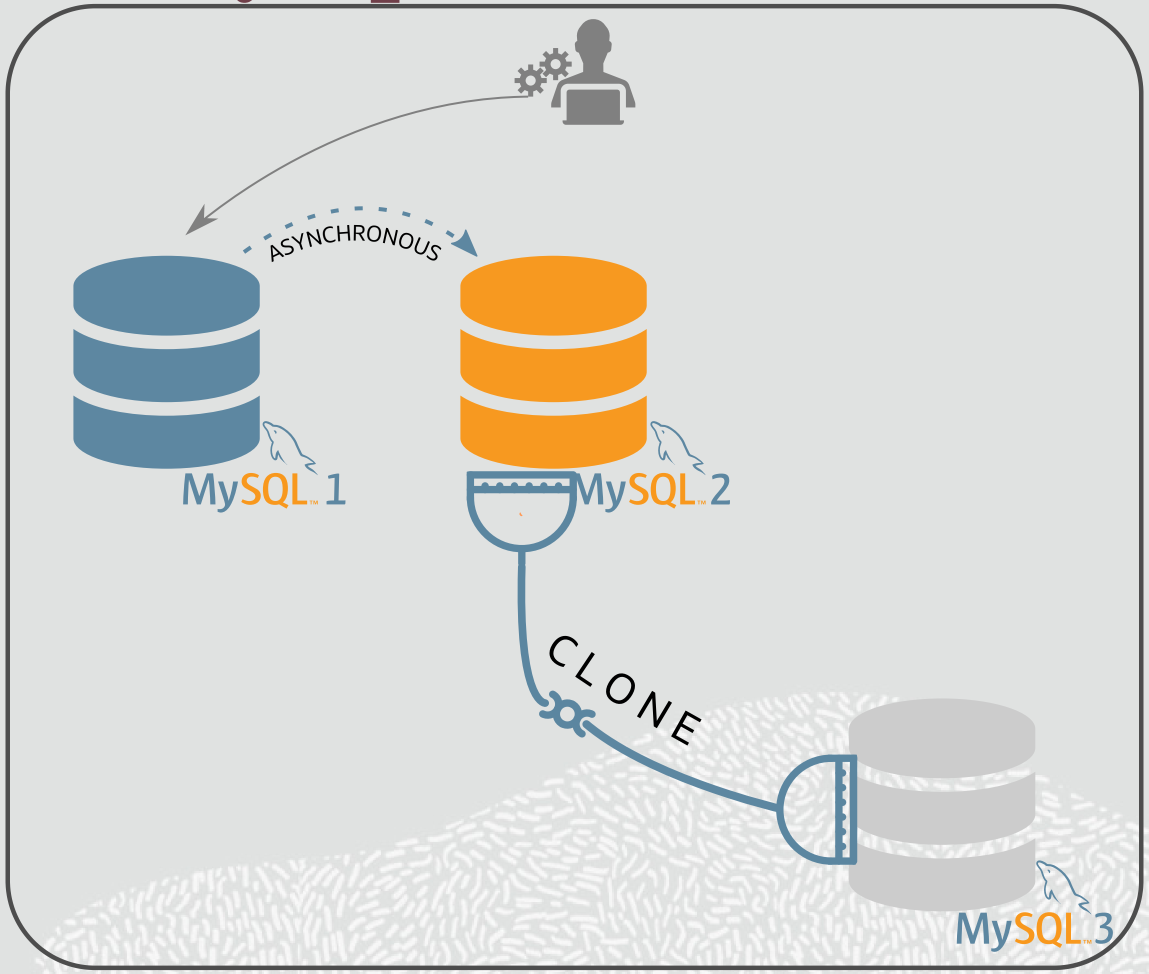
```
MySQL 8.0.17   localhost:33060+   2019-09-21 23:56:27
JS  \s
MySQL Shell version 8.0.18

Session type:                  X
Connection Id:                 13
```

LAB2: Transfer the data to mysql3

We will use the new `CLONE` plugin to transfer the data from `mysql2` to `mysql3`.

Please note that installing a plugin requires a connection using the classic protocol (3306)



LAB2: Transfer the data to mysql3

We start by installing the CLONE plugin on **`mysql2`**:

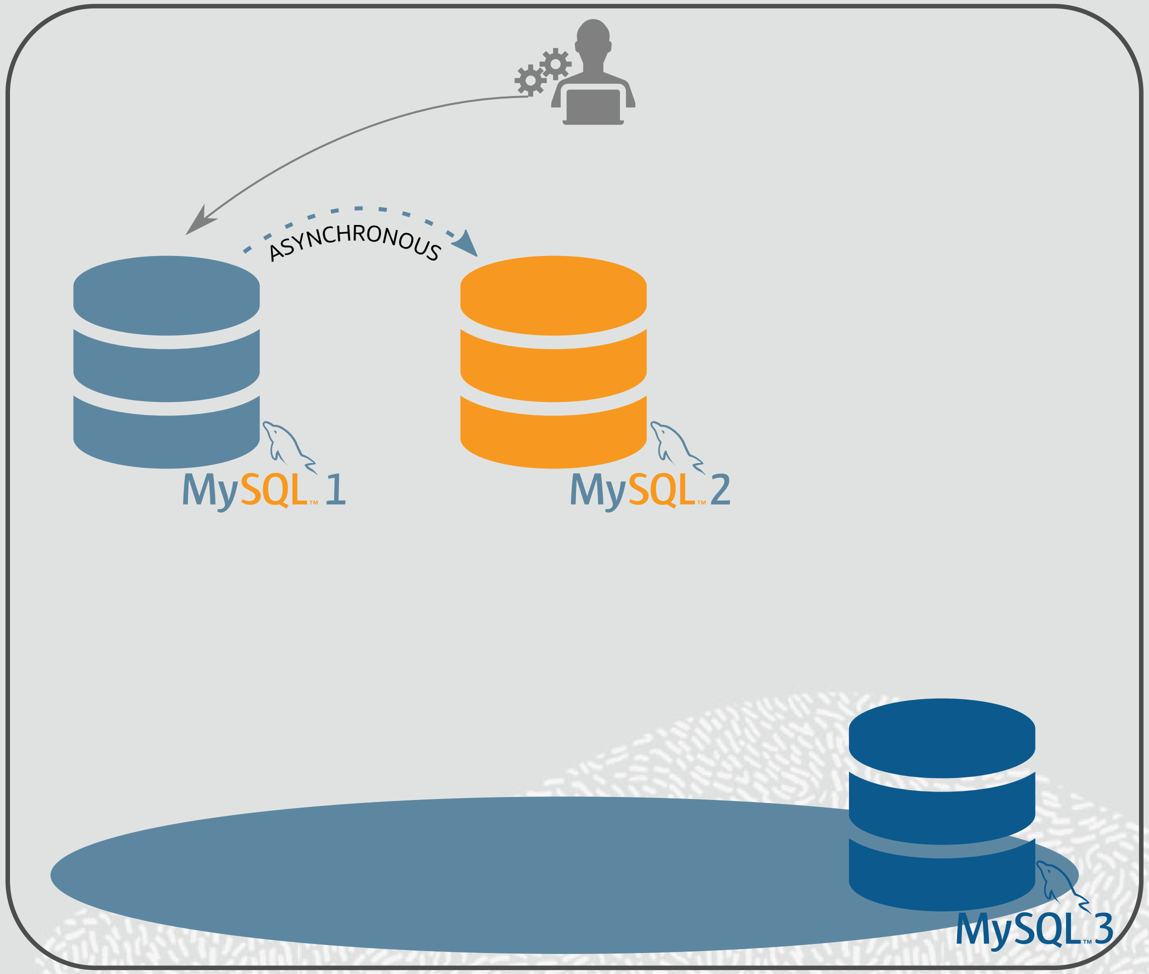
```
JS> \c clusteradmin@mysql2:3306
mysql2>JS> \sql
mysql2>sql> INSTALL PLUGIN clone SONAME 'mysql_clone.so';
```

Then we install it on **`mysql3`** and we configure it to use **`mysql2`** as donor and we start the cloning process:

```
mysql3>sql> \c clusteradmin@mysql3:3306
mysql3>sql> INSTALL PLUGIN clone SONAME 'mysql_clone.so';
mysql3>sql> SET GLOBAL clone_valid_donor_list='mysql2:3306';
mysql3>sql> CLONE INSTANCE FROM clusteradmin@mysql2:3306 IDENTIFIED BY 'mysql';
```

LAB3: MySQL InnoDB Cluster Creation

Let's create our cluster !



LAB3: MySQL InnoDB Cluster Creation

We will operate everything from one Shell, for the example, we will use **`mysql1`**. Let's start by configuring **`mysql3`** to be have all required settings to be part of a MySQL InnoDB Cluster:

```
JS> dba.configureInstance('clusteradmin@mysql3')
```

Then we connect to it and we create our cluster:

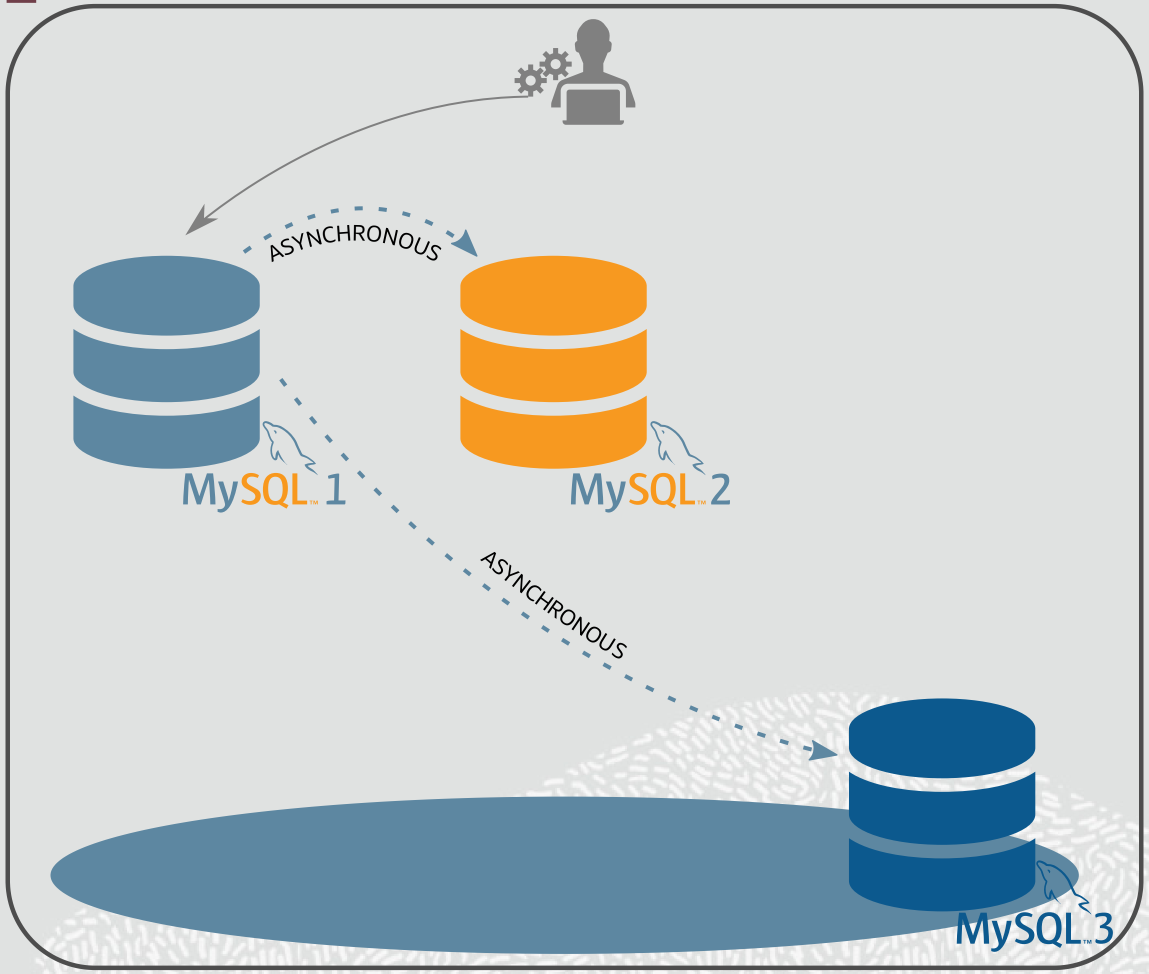
```
JS> \c clusteradmin@mysql3
mysql3>JS> cluster=dba.createCluster('PLEU19')
```

You can check the result of the cluster like this:

```
mysql3>JS> cluster.status()
```

LAB4: Asynchronous Replication

Let's configure asynchronous replication on `mysql3` to get the traffic from `mysql1`.



LAB4: Asynchronous Replication

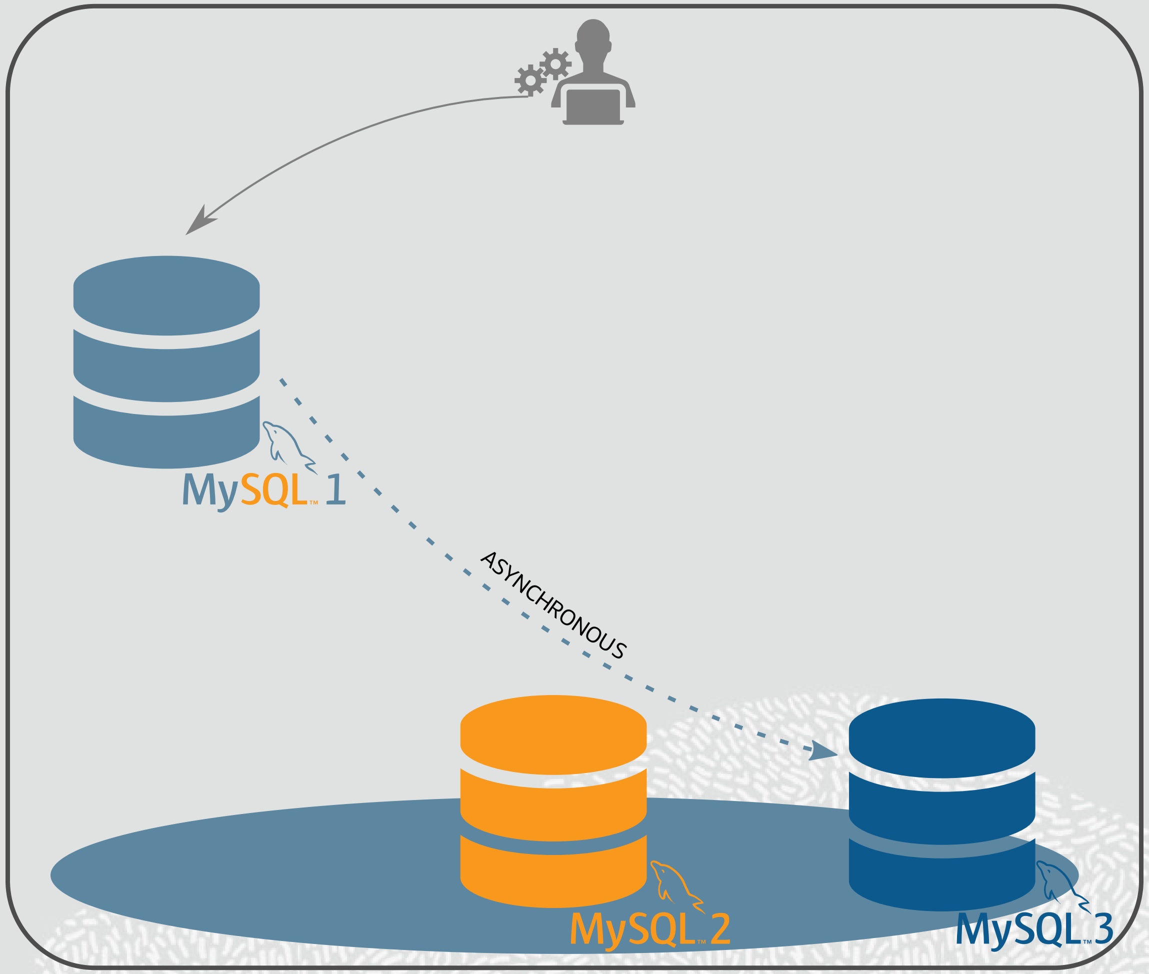
Configure & start asynchronous replication on `mysql3`:

```
sql> \c clusteradmin@mysql3
mysql3>sql> CHANGE MASTER TO MASTER_HOST='mysql1', MASTER_PORT=3306,
     MASTER_USER='repl', MASTER_PASSWORD='X',
     MASTER_AUTO_POSITION=1, MASTER_SSL=1;

mysql3>sql> START SLAVE;
```

LAB5: Add a new instance to our cluster

Let's add now **`mysql2`** to the MySQL InnoDB Cluster.



LAB5: Add a new instance to our cluster

Now we will join `mysql2` to the MySQL InnoDB Cluster.

LAB5: Add a new instance to our cluster

Now we will join **`mysql2`** to the MySQL InnoDB Cluster.

The first step is stop the asynchronous replication on **`mysql2`**:

```
js> \c clusteradmin@mysql2
mysql2>js> \sql stop slave;
mysql2>js> \sql reset slave all;
```

LAB5: Add a new instance to our cluster

Now we will join **`mysql2`** to the MySQL InnoDB Cluster.

The first step is stop the asynchronous replication on **`mysql2`**:

```
js> \c clusteradmin@mysql2
mysql2>js> \sql stop slave;
mysql2>js> \sql reset slave all;
```

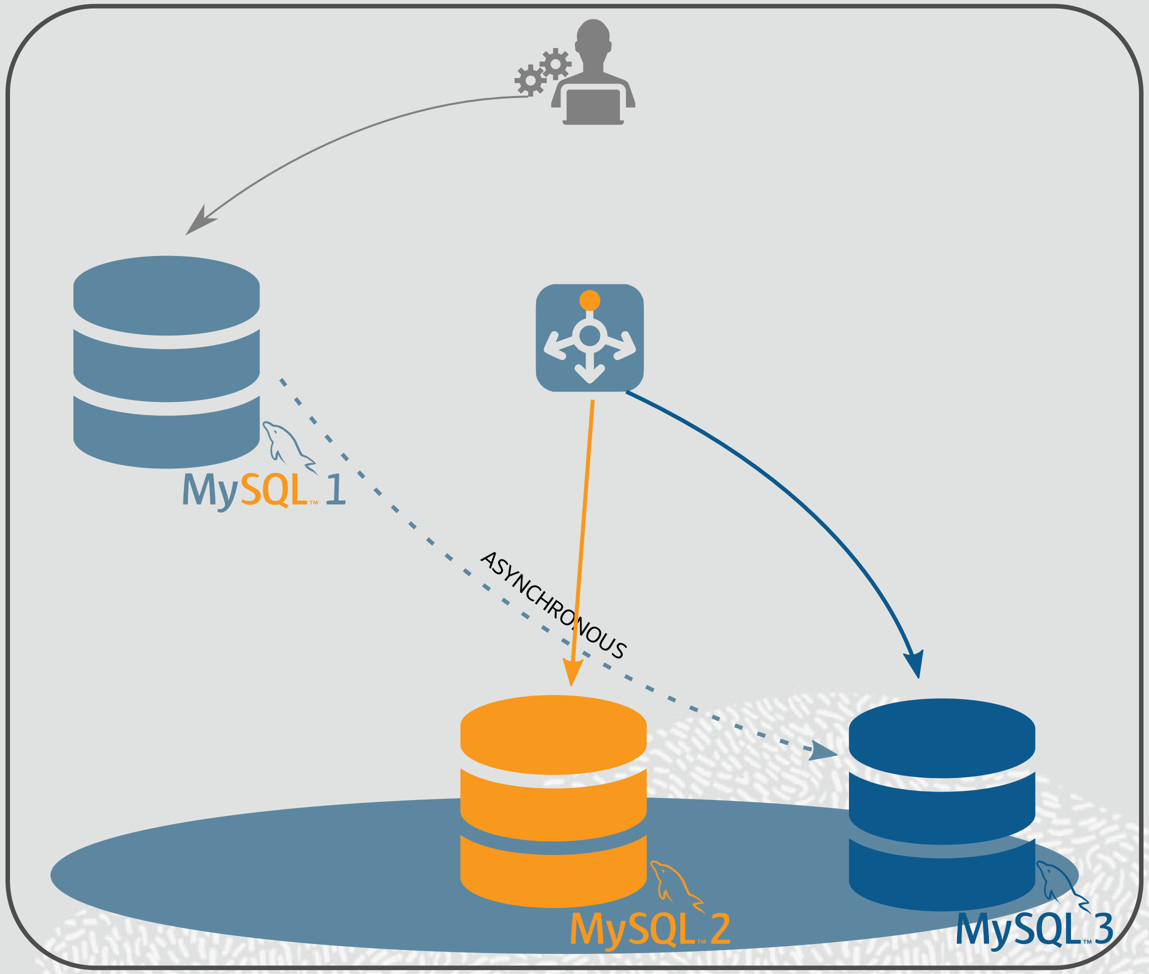
In the MySQL Shell of **`mysql1`**, we configure the **`mysql2`** and we add it to the cluster:

```
JS> \c clusteradmin@mysql3
mysql3>JS> cluster=dba.getCluster()
mysql3>JS> dba.configureInstance('clusteradmin@mysql2')

mysql3>JS> cluster.addInstance('clusteradmin@mysql2')
```

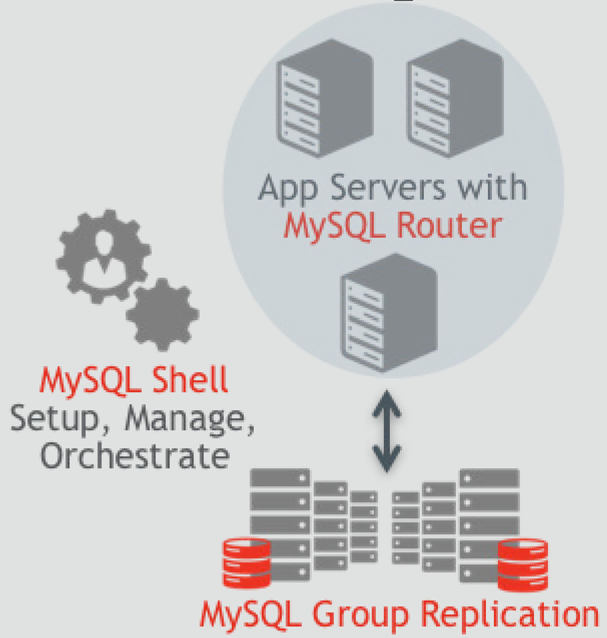
LAB6: MySQL Router

We have now already a cluster of 2 nodes running. We will bootstrap a MySQL Router



MySQL Router

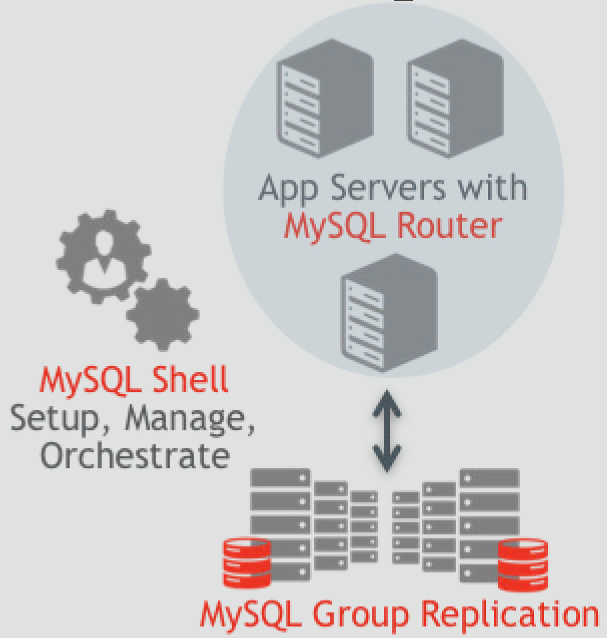
Transparent Access to Database Architecture



"provide transparent routing between your application and back-end MySQL Servers"

MySQL Router

Transparent Access to Database Architecture



"provide transparent routing between your application and back-end MySQL Servers"

- Transparent client connection routing
  - Load balancing
  - Application connection failover
- Stateless design offers easy HA client routing
  - Router as part of the application stack
- Little to no configuration needed
- Native support for InnoDB clusters
  - Understands Group Replication topology
  - Utilizes metadata schema on each member
- Currently 2 TCP Ports: `PRIMARY` and `NON-PRIMARY` traffic (for Classic and X Protocol)

LAB6: MySQL Router

MySQL Router will be used on the application server (`mysql1`):

```
[root]# mysqlrouter --bootstrap=clusteradmin@mysql3:3306 --user=mysqlrouter
```

LAB6: MySQL Router

MySQL Router will be used on the application server (**`mysql1`**):

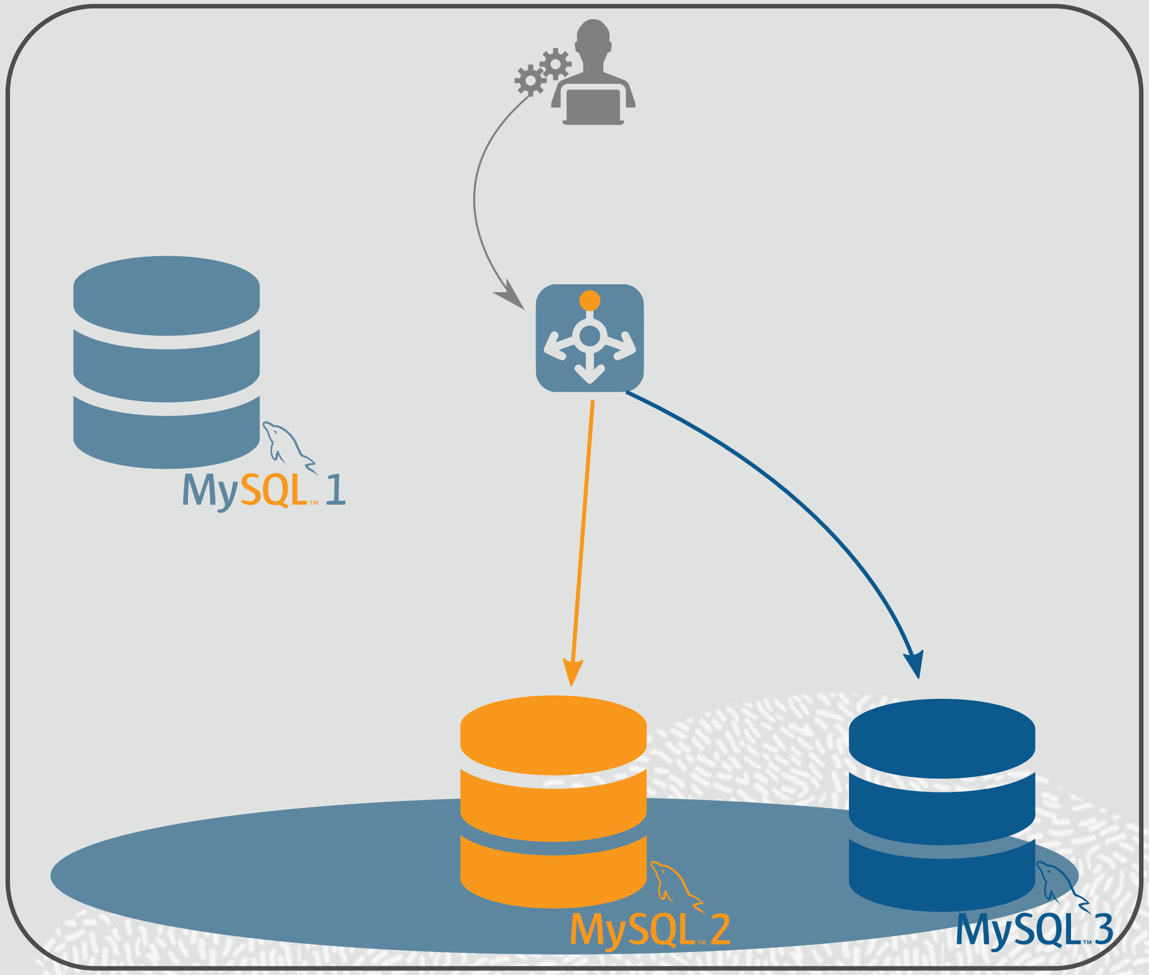
```
[root]# mysqlrouter --bootstrap=clusteradmin@mysql3:3306 --user=mysqlrouter
```

We can start the Router:

```
[root]# systemctl start mysqlrouter
```

LAB7: Using the MySQL InnoDB Cluster

Now we will point our application to the MySQL Router, this is the **only** downtime during our migration.



LAB7: Using the MySQL InnoDB Cluster

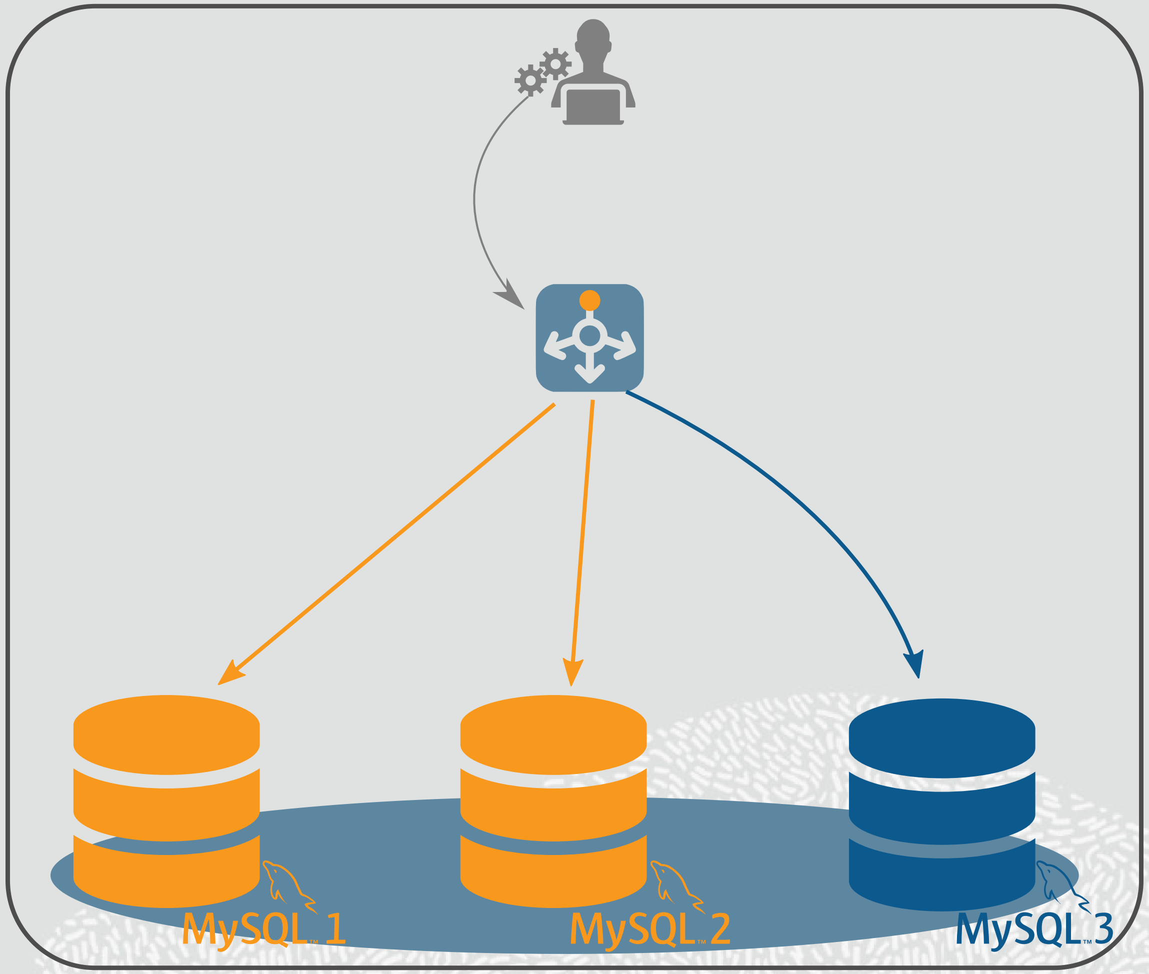
In the `screen` session where the application is running (`screen -rx`), we stop it (`^C`) and we restart it connecting to the Router:

```
[root]# run_app.sh mysql1 router
```

LAB8: 3 nodes MySQL InnoDB Cluster

It's time to add the last node in our MySQL InnoDB Cluster.

And don't forget to stop slave on `mysql3` !



LAB8: 3 nodes MySQL InnoDB Cluster

We will now add the **`mysql1`** to the cluster exactly as we did with **`mysql2`**:

```
JS> dba.configureInstance('clusteradmin@mysql1')

mysql3>JS> \c clusteradmin@mysql3
mysql3>JS> cluster=dba.getCluster()
mysql3>JS> cluster.addInstance('clusteradmin@mysql1')
```

LAB8: 3 nodes MySQL InnoDB Cluster

We will now add the **`mysql1`** to the cluster exactly as we did with **`mysql2`**:

```
JS> dba.configureInstance('clusteradmin@mysql1')

mysql3>JS> \c clusteradmin@mysql3
mysql3>JS> cluster=dba.getCluster()
mysql3>JS> cluster.addInstance('clusteradmin@mysql1')
```

```
JS> cluster.status()
{
    "clusterName": "PLEU19",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mysql3:3306",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
...
```

Play Time!

Upgrade to MySQL 8.0

It's time to upgrade to MySQL 8.0, the fastest MySQL adoption release ever !

October 1st 5:00PM-5:50PM ROOM B

MySQL InnoDB Cluster
Advanced Configuration & Operations

OPEN SOURCE DATABASE CONFERENCE PERCONA LIVE

Pedro Gomes
MySQL Senior Software Developer

Frédéric Descamps
MySQL Community Manager

Meet the MySQL Team at the Conference

Sunny Bains

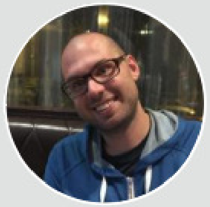
Frédéric Descamps

Pedro Gomes

Luis Soares

Dimitri Kravtchuk

Georgi Kodinov


Kenny Gryp

Ståle Deraas

Norvald Ryeng

Geir Høydalsvik

Thank you !