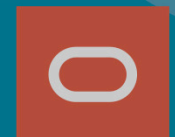
The Evolution of a MySQL Database Architecture

from single instance to mutli-regions DR with read replicas and transparent read/write splitting

Frédéric Descamps
Community Manager
Oracle MySQL
Confoo.CA - February 2024



The Journey To MySQL InnoDB ClusterSet

Who am I ?

about.me/lefred

Frédéric Descamps

- *@lefred*
- *MySQL Evangelist*
- *using MySQL since version 3.20*
- *devops believer*
- *likes* 🏀
- *living in* 🇧🇪
- *https://lefred.be*

Best Practices in 2024

configuration parameters

During this presentation I assume that...

- *your system is MySQL 8.0.36 or more recent (Innovation Release 8.3.0)*
- *you are using only InnoDB*
- *you have the binary logs enabled*
- *the binary log format is ROW*
- *you are using GTIDs*

Evolution to High Availability

MySQL 8.x

MySQL™ High Availability Evolution

One single MySQL instance

This is where it all begins...

reminder:

- *we **ONLY** use InnoDB*
- *we keep the default durability settings*

One single MySQL instance

					
					
	/				
	/				

Next Level (level up)

The database becomes more important, loosing it might be an issue...

Next Level (level up)

The database becomes more important, loosing it might be an issue…

RTO → hours

Next Level (level up)

The database becomes more important, loosing it might be an issue...

RTO → hours

RPO → 1 day

Next Level (level up)

The database becomes more important, loosing it might be an issue...

RTO → hours

RPO → 1 day

***RTO**: Recovery Time Objective (how long to recover)*
***RPO**: Recovery Point Objective (how much data can be lost)*

Backups

- *Physical Backups*
- *Logical Backups*

For logical backups, please use MySQL Shell ***Dump & Load Utility*** *!*

```
[fred@imac ~] $ mysqlsh mysql://root@localhost
MySQL Shell 8.0.28

Copyright (c) 2016, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a Classic session to 'root@localhost'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 18
Server version: 8.0.28 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL  localhost  2022-02-01 15:03:01
JS  util.dumpInstance('/tmp/logical_backup', {threads: 8})
```

One single MySQL instance with daily backups

Next Level (level up)

1 day RPO ? Really ?? We want to reduce it to minutes at least !

Next Level (level up)

1 day RPO ? Really ?? We want to reduce it to minutes at least !

RTO → hours

Next Level (level up)

1 day RPO ? Really ?? We want to reduce it to minutes at least !

RTO → hours

RPO → minutes

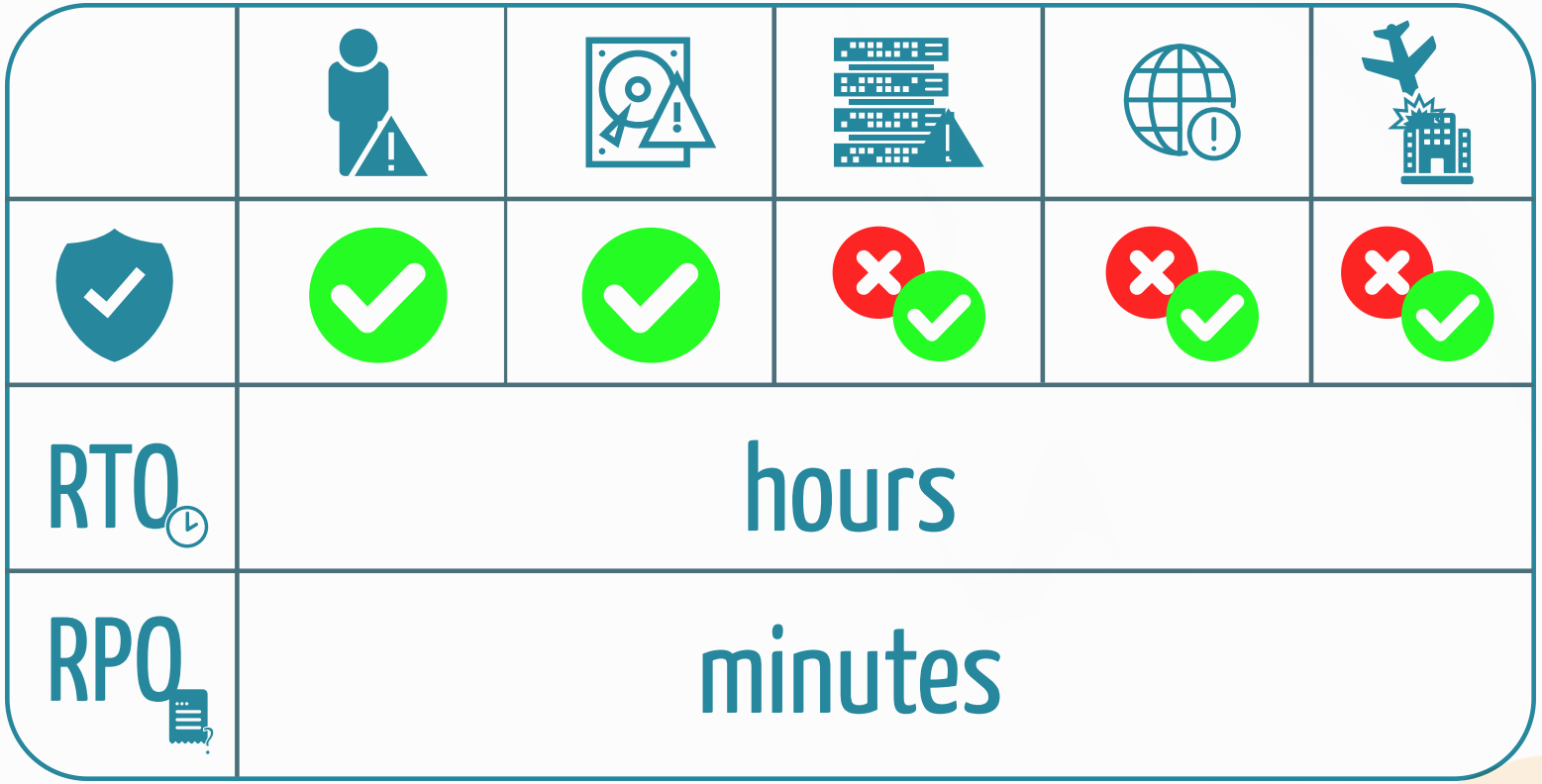
Durable Binlogs

We enable binary logs allowing to replay the modifications since the last backup:

These are the defaults in MySQL 8.0:

```
MySQL  localhost:33060+  2021-01-05 12:14:57
SQL  show global variables like 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
1 row in set (0.0020 sec)
MySQL  localhost:33060+  2021-01-05 12:15:03
SQL  show global variables like 'sync_binlog';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| sync_binlog   | 1     |
+---------------+-------+
1 row in set (0.0020 sec)
```

One single MySQL instance with daily backups and binlogs



Point-in-Time Recovery (PITR)

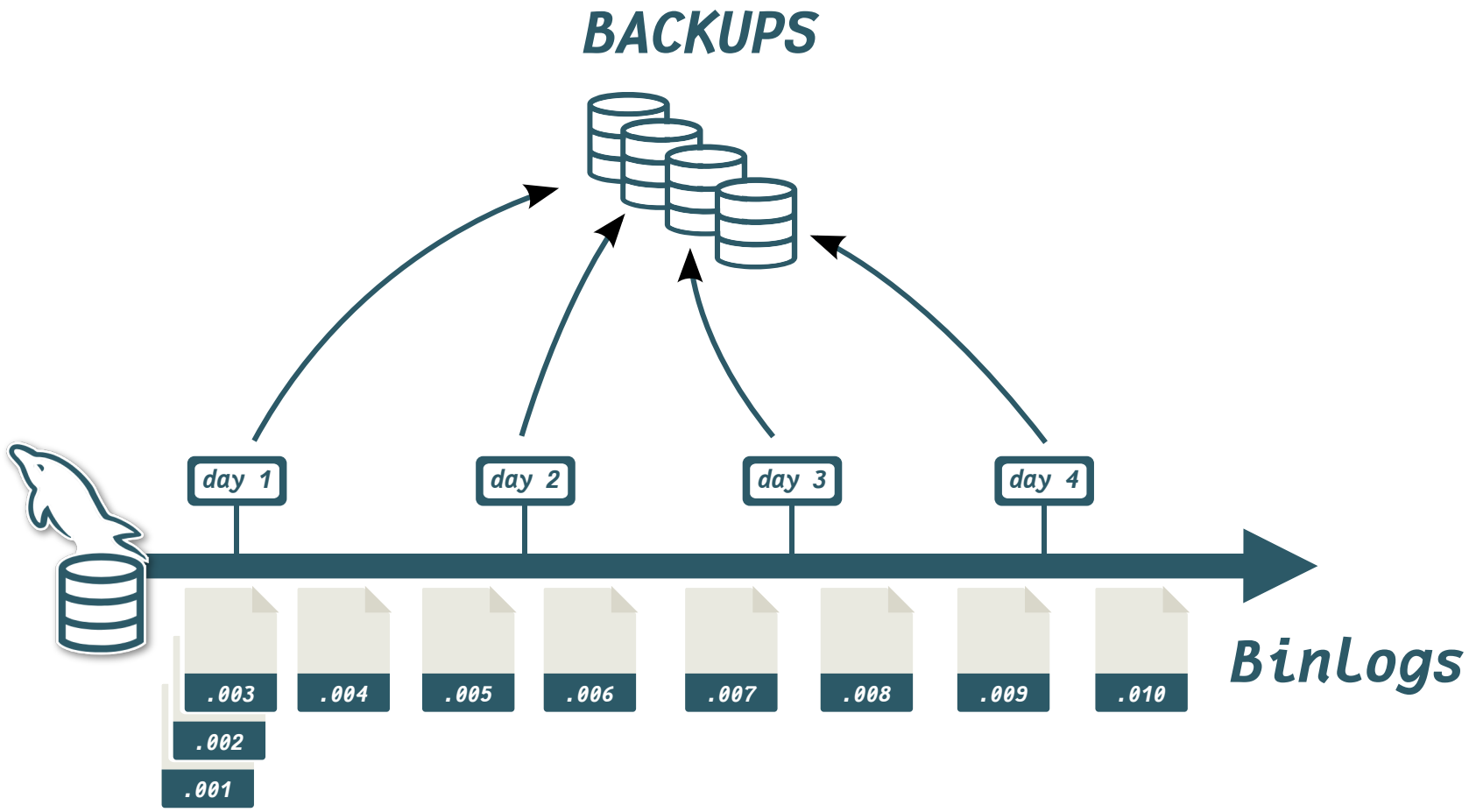
PITR is the technique by which an administrator can restore or recover a data set to a certain point in the past.

With MySQL, point-in-time recovery involves restoring a dump of the data and then replaying the binlogs from and to a specific point.

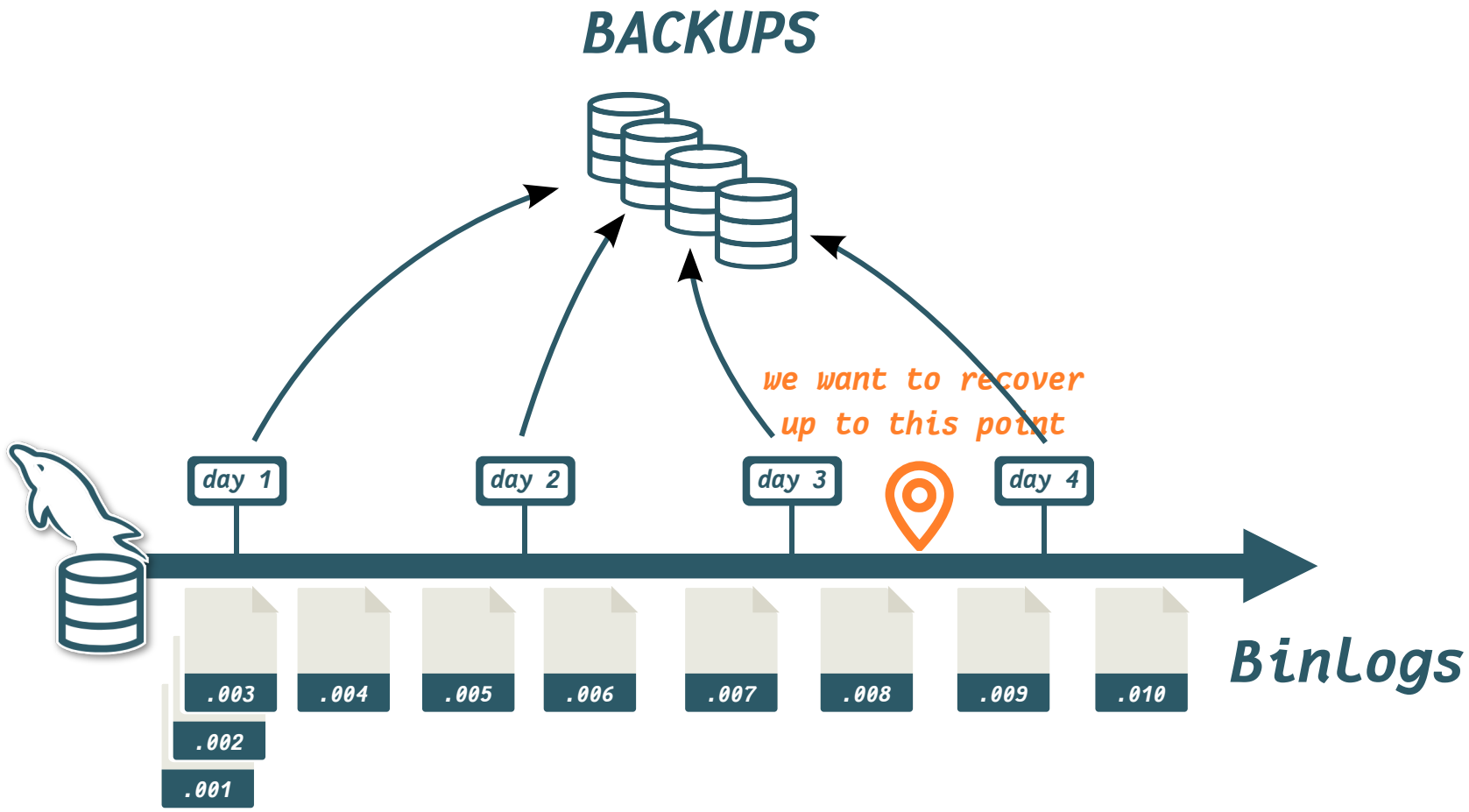
This technique is used to:

- *solve an issue*
- *perform a live migration*

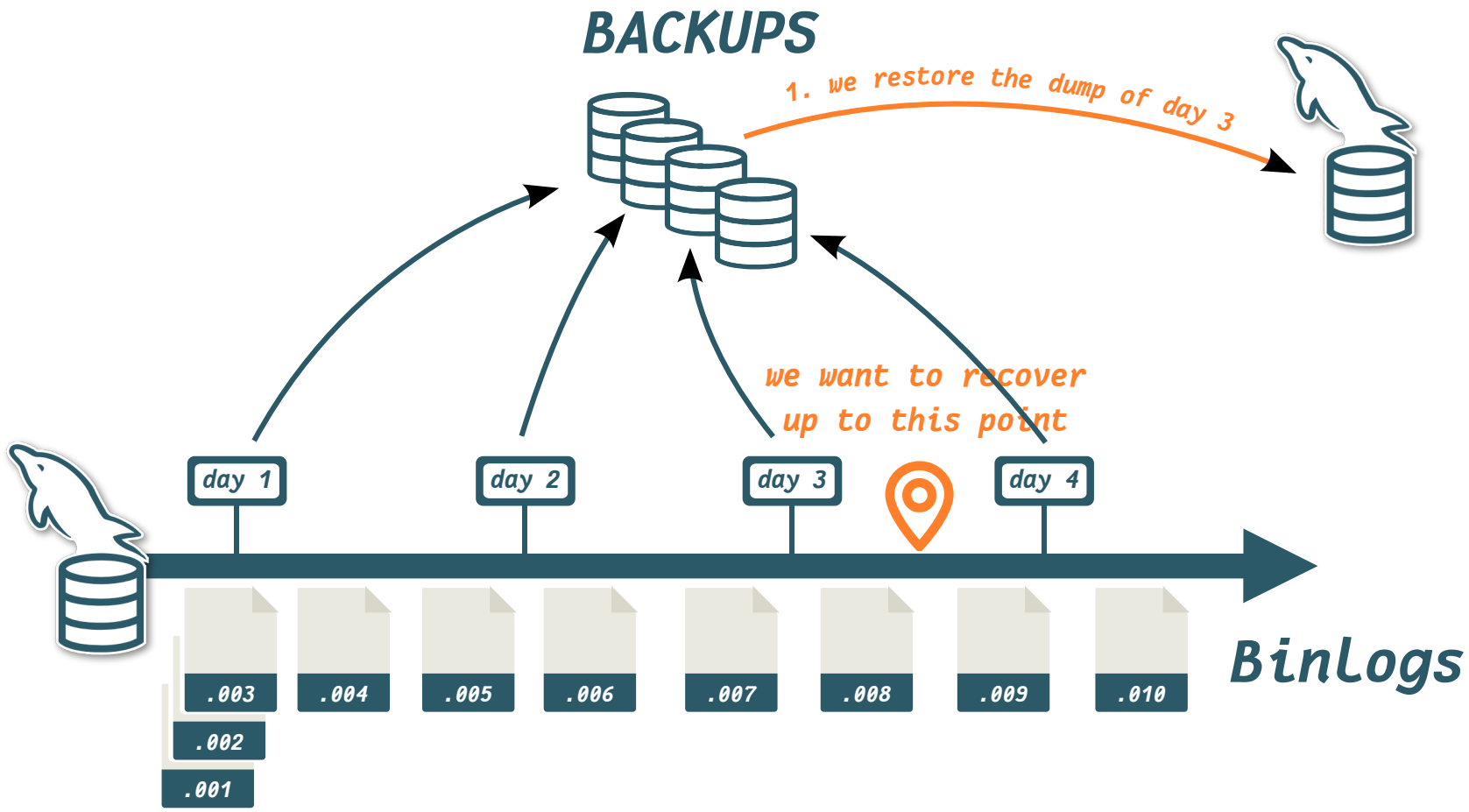
Point-in-Time Recovery : how does it work ?



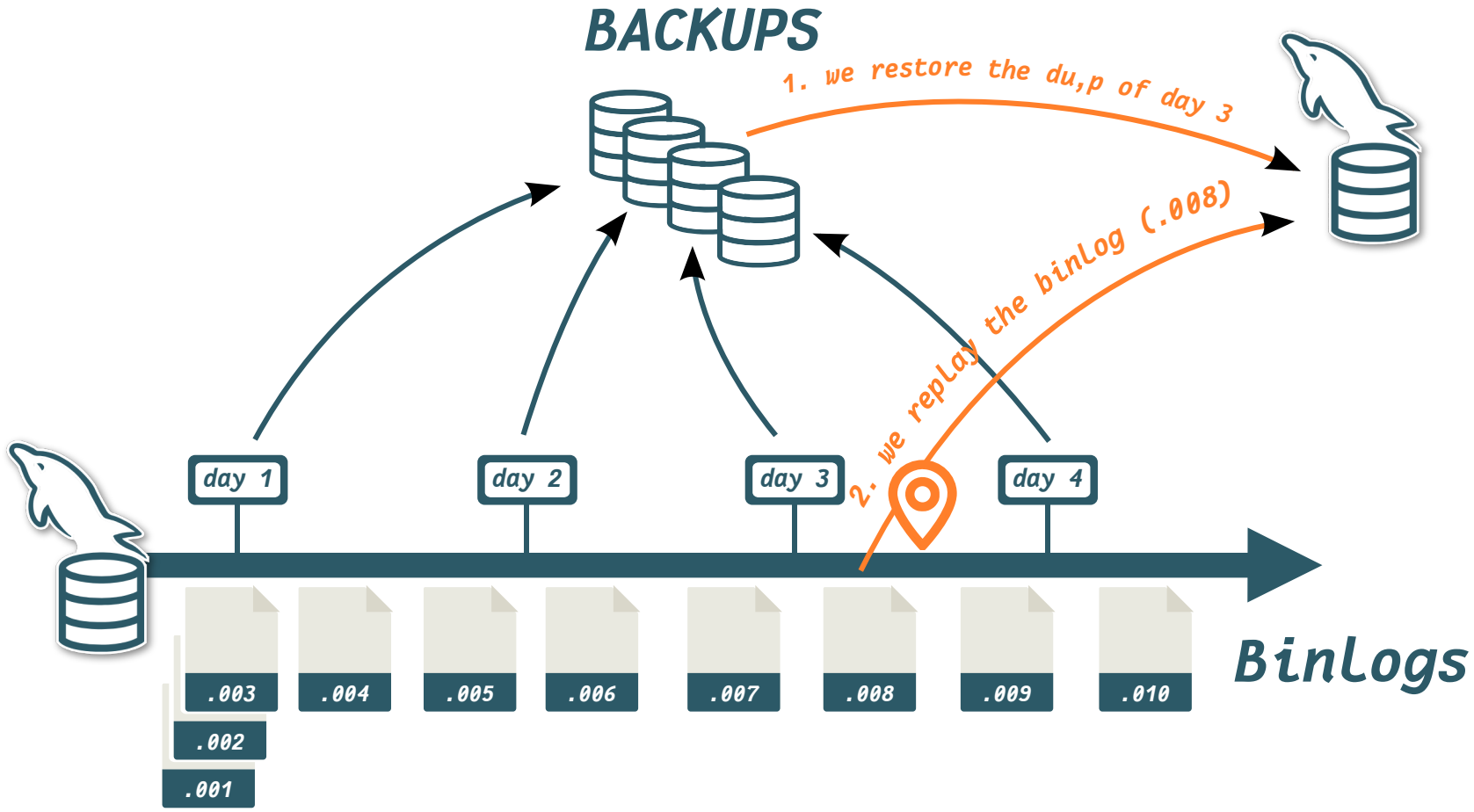
Point-in-Time Recovery : how does it work ?



Point-in-Time Recovery : how does it work ?

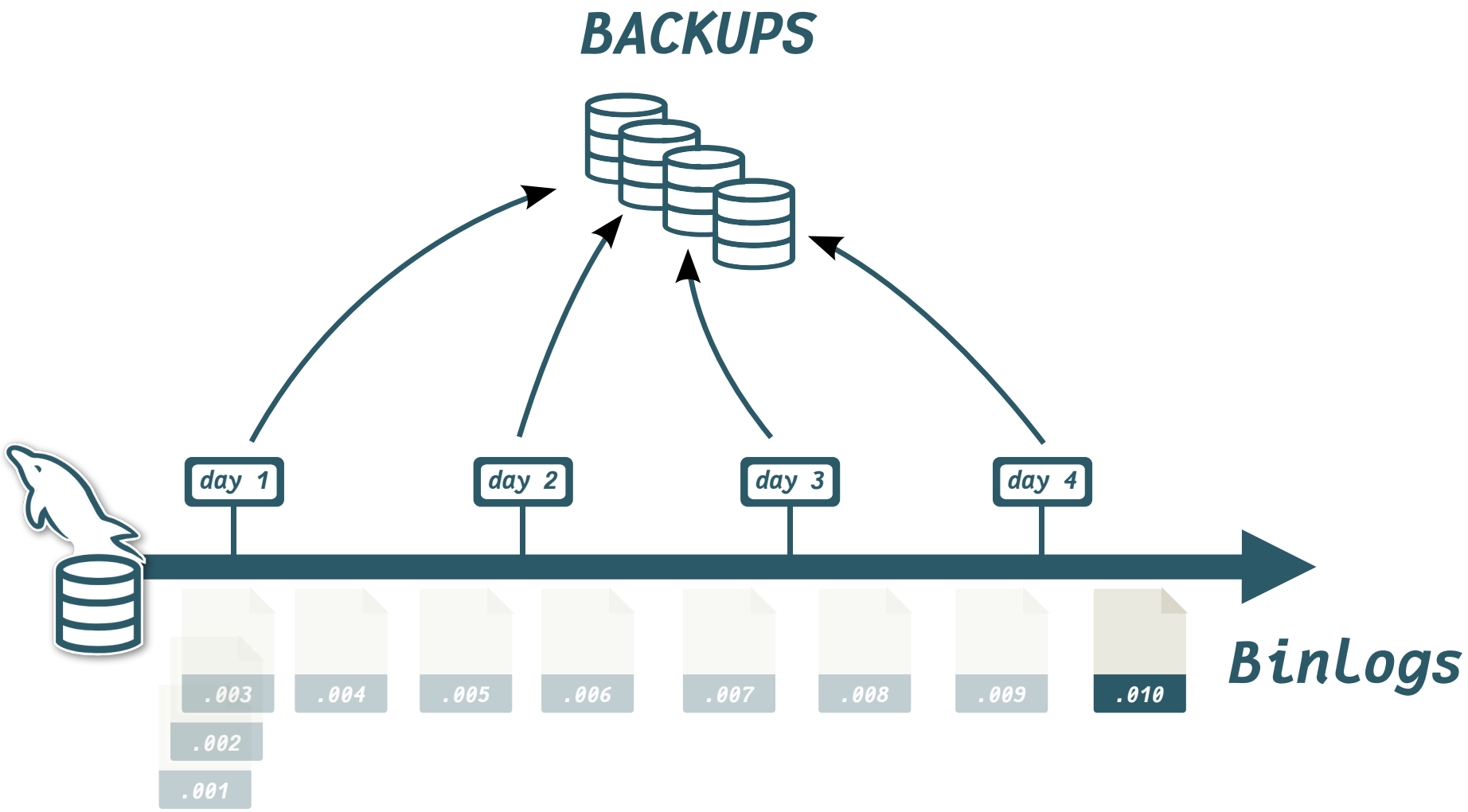


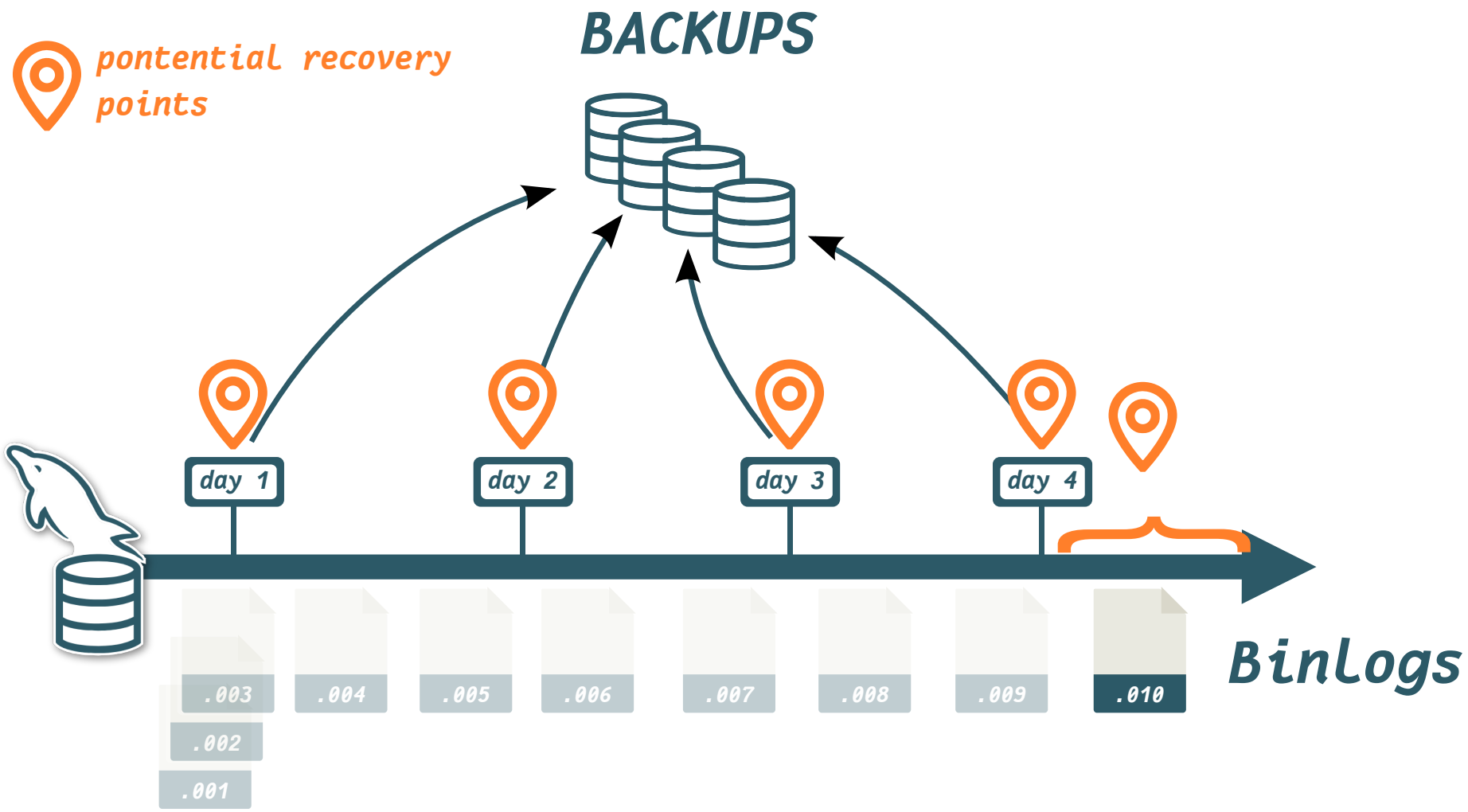
Point-in-Time Recovery : how does it work ?



Point-in-Time Recovery : important concepts

Usually, after a backup (and its verification), the binary log files are purged from the MySQL server:





Next Level (level up)

My data is very important and I've a heavy workload... I would like to loose less than a second in case of a problem !

Next Level (level up)

My data is very important and I've a heavy workload... I would like to loose less than a second in case of a problem !

RTO → hours

Next Level (level up)

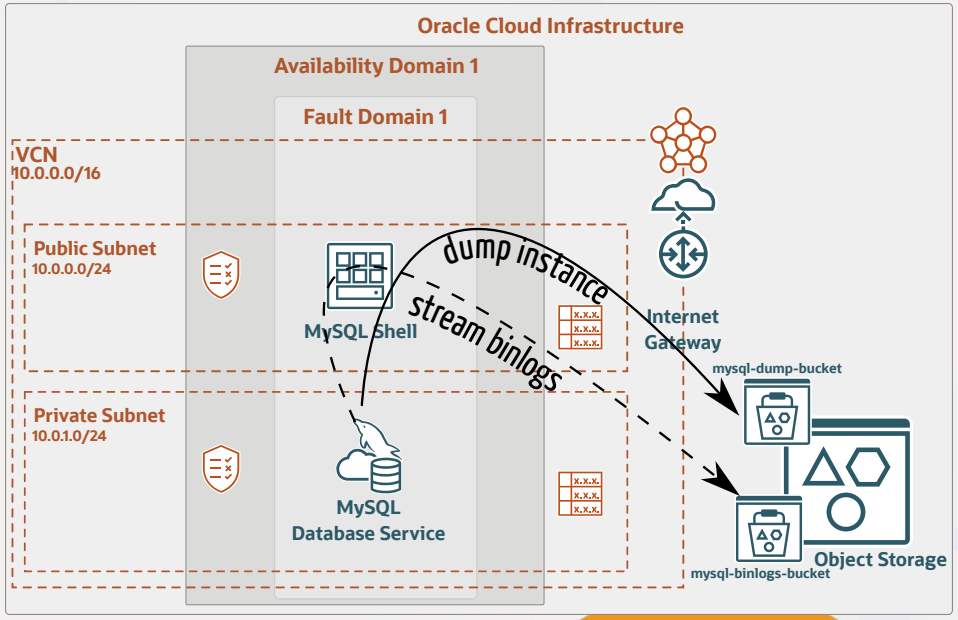
My data is very important and I've a heavy workload... I would like to loose less than a second in case of a problem !

RTO → hours

RPO → less than a second

Improved Point-in-Time Recovery (PITR)

- *Enable GTIDs (they should be already !)*
- *Save in realtime the binary logs to a different system (OCI Object Storage, S3, ...)*



Enabling GTIDs

with MySQL restart

```
MySQL  localhost:33060+  2021-01-05 12:27:49
SQL  show global variables like 'gtid_mode';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| gtid_mode     | OFF   |
+---------------+-------+
1 row in set (0.0040 sec)
MySQL  localhost:33060+  2021-01-05 12:28:16
SQL  set persist_only gtid_mode='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:28:34
SQL  set persist_only enforce_gtid_consistency='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:29:07
SQL  restart;
Query OK, 0 rows affected (0.0009 sec)
```

Enabling GTIDs

with MySQL restart

```
MySQL  localhost:33060+  2021-01-05 12:27:49
SQL  show global variables like 'gtid_mode';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| gtid_mode     | OFF   |
+---------------+-------+
1 row in set (0.0040 sec)
MySQL  localhost:33060+  2021-01-05 12:28:16
SQL  set persist_only gtid_mode='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:28:34
SQL  set persist_only enforce_gtid_consistency='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:29:07
SQL  restart;
Query OK, 0 rows affected (0.0009 sec)
```

without MySQL restart

```
MySQL  localhost:33060+  2021-01-05 12:31:00
SQL  show global variables like 'gtid_mode';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| gtid_mode     | OFF   |
+---------------+-------+
1 row in set (0.0043 sec)
MySQL  localhost:33060+  2021-01-05 12:31:01
SQL  set persist enforce_gtid_consistency='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:31:27
SQL  set persist gtid_mode='off_permissive';
Query OK, 0 rows affected (0.0101 sec)
MySQL  localhost:33060+  2021-01-05 12:32:00
SQL  set persist gtid_mode='on_permissive';
Query OK, 0 rows affected (0.0097 sec)
MySQL  localhost:33060+  2021-01-05 12:32:06
SQL  set persist gtid_mode='on';
Query OK, 0 rows affected (0.0138 sec)
MySQL  localhost:33060+  2021-01-05 12:32:10
SQL  show global variables like 'gtid_mode';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| gtid_mode     | ON    |
+---------------+-------+
1 row in set (0.0020 sec)
```

Saving Binlogs

```
MySQL  localhost:33060+  2021-01-05 13:07:33
SQL  create user getbinlog identified by 'password' require ssl;
Query OK, 0 rows affected (0.0073 sec)
MySQL  localhost:33060+  2021-01-05 13:08:05
SQL  grant replication slave on *.* to getbinlog;
Query OK, 0 rows affected (0.0044 sec)
```

Saving Binlogs

```
MySQL  localhost:33060+  2021-01-05 13:07:33
SQL  create user getbinlog identified by 'password' require ssl;
Query OK, 0 rows affected (0.0073 sec)
MySQL  localhost:33060+  2021-01-05 13:08:05
SQL  grant replication slave on *.* to getbinlog;
Query OK, 0 rows affected (0.0044 sec)
```

On another system:

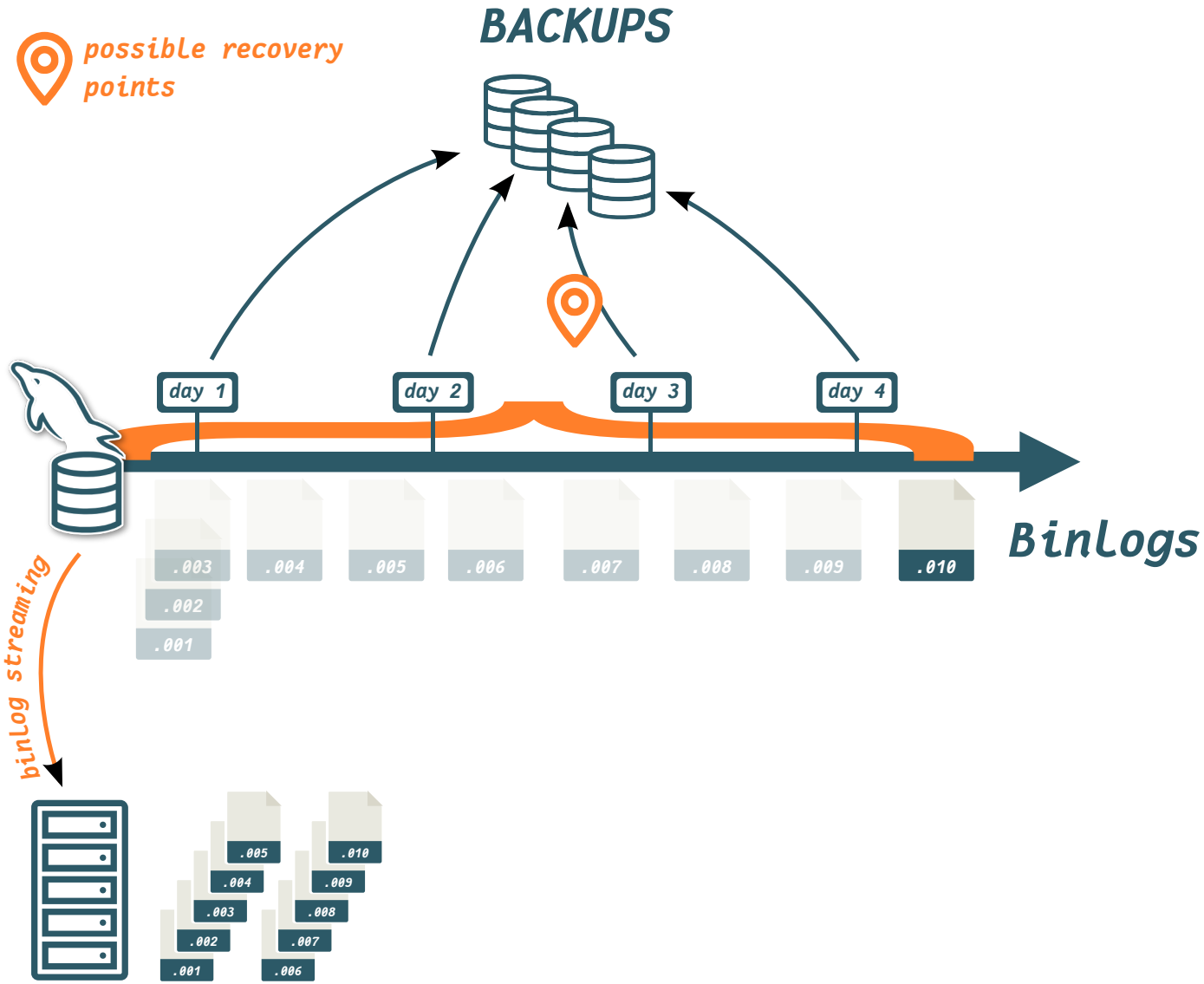
```
mysqlbinlog --raw --read-from-remote-server --stop-never --host 10.0.0.2 \
                        --port 3306 -u getbinlog -ppassword \
                         --ssl-mode='REQUIRED' binlog.xxxxxx
```

Point-in-Time Recovery : important concepts (2)

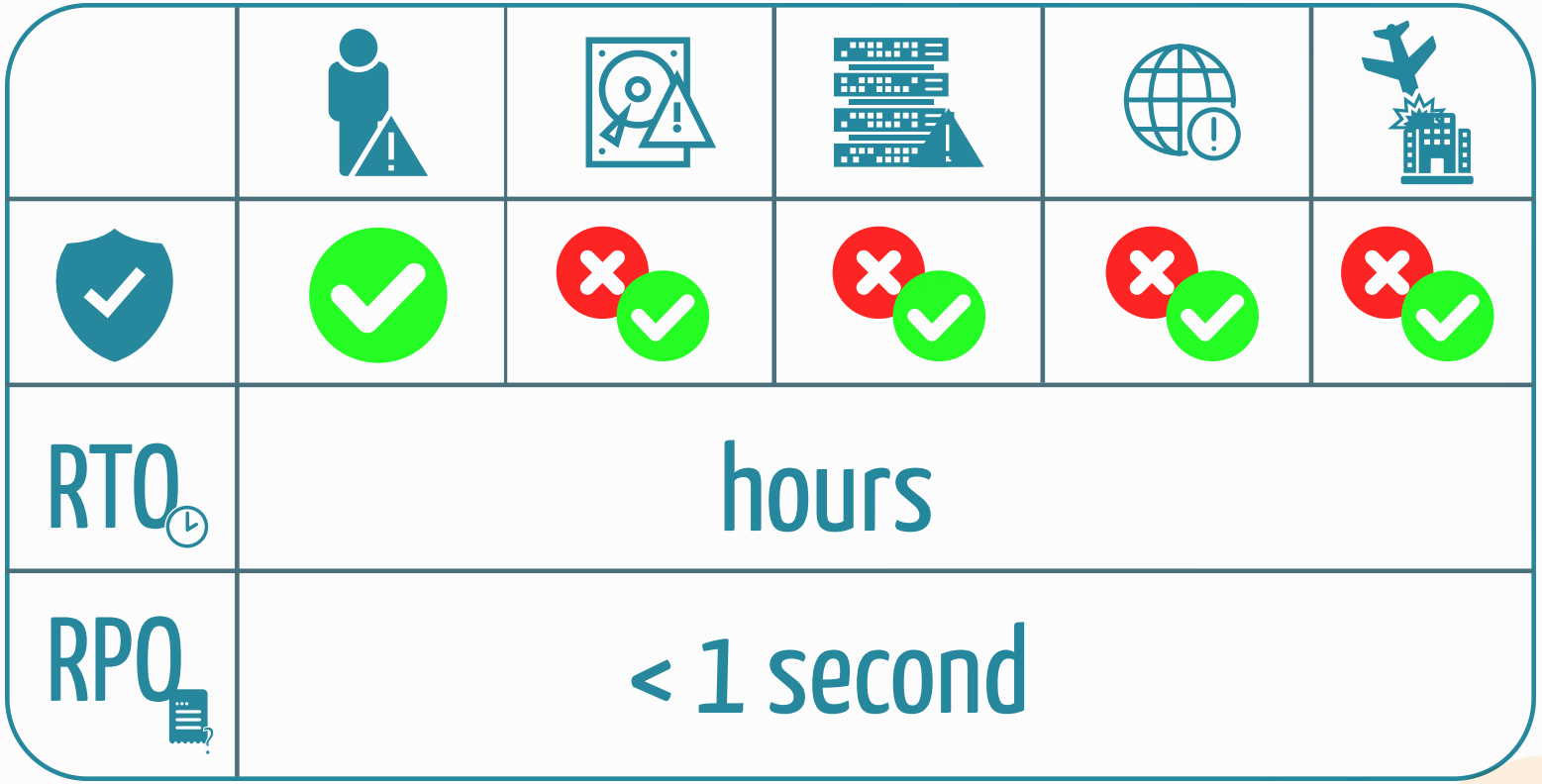
As you may have noticed, we can only recover at the exact time of the backups and we can only do a one-off recovery from the last one!

This is why it is recommended to relocate/offload the binlogs (to another server, a NAS, the cloud, ...).

This will allow for a point-in-time recovery at any moment in time:



One single MySQL instance with daily backups and external binlogs archiving



Next Level (level up)

My service is now very important, I would like my database to be available again within minutes in case of an incident.

Next Level (level up)

My service is now very important, I would like my database to be available again within minutes in case of an incident.

RTO → minutes

Next Level (level up)

My service is now very important, I would like my database to be available again within minutes in case of an incident.

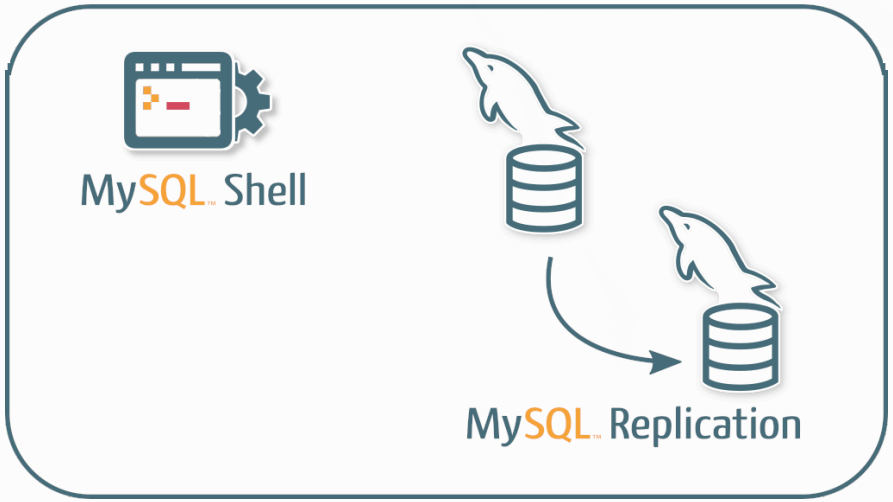
RTO → minutes

RPO → less than a second

MySQL InnoDB ReplicaSet

- *Based on the well known Asynchronous Replication*
- *But easier… easier is always better !*
- *Data provisioning included (clone)*
- *2 or more nodes*
- *Manual Failover*
- *Transparent/Automatic query routing with MySQL **Router***

MySQL InnoDB ReplicaSet



MySQL InnoDB ReplicaSet

Connected to the actual server (our single instance):

```
JS > dba.configureReplicaSetInstance()

JS > rs=dba.createReplicaSet('myreplicaset')
```

On the new extra server where only MySQL is installed (server and shell):

```
JS > dba.configureReplicaSetInstance()
```

And again on the main instance (our single instance):

```
JS > rs.addInstance('mysql-2')
```

MySQL InnoDB ReplicaSet - examples

```
 MySQL  localhost:33060+ ssl  JS  > dba.configureReplicaSetInstance()
Configuring local MySQL instance listening at port 3306 for use in an InnoDB ReplicaSet...

This instance reports its own address as mysql-2:3306
Clients and other cluster members will communicate with it through this address by default. If this is not correct, the report_host MySQL system variable should
 be changed.

ERROR: User 'root' can only connect from 'localhost'. New account(s) with proper source address specification to allow remote connection from all instances must
 be created to manage the cluster.

1) Create remotely usable account for 'root' with same grants and password
2) Create a new admin account for InnoDB ReplicaSet with minimal required grants
3) Ignore and continue
4) Cancel

Please select an option [1]: 2
Please provide an account name (e.g: icroot@%) to have it created with the necessary
privileges or leave empty and press Enter to cancel.
Account Name: clusteradmin
Password for new account: ********
Confirm password: ********

NOTE: Some configuration options need to be fixed:
+--------------------------+---------------+----------------+--------------------------------------------------+
| Variable                 | Current Value | Required Value | Note                                             |
+--------------------------+---------------+----------------+--------------------------------------------------+
| enforce_gtid_consistency | OFF           | ON             | Update read-only variable and restart the server |
| gtid_mode                | OFF           | ON             | Update read-only variable and restart the server |
| server_id                | 1             | <unique ID>    | Update read-only variable and restart the server |
+--------------------------+---------------+----------------+--------------------------------------------------+

Some variables need to be changed, but cannot be done dynamically on the server.
Do you want to perform the required configuration changes? [y/n]: y
Do you want to restart the instance after configuring it? [y/n]: y
Cluster admin user 'clusteradmin'@'%' created.
Configuring instance...
The instance 'mysql-2:3306' was configured to be used in an InnoDB ReplicaSet.
Restarting MySQL...
NOTE: MySQL server at mysql-2:3306 was restarted.
```

MySQL InnoDB ReplicaSet - examples

```
Please select a recovery method [C]lone/[I]ncremental recovery/[A]bort (default Clone): c
* Updating topology
Waiting for clone process of the new member to complete. Press ^C to abort the operation.
* Waiting for clone to finish...
NOTE: mysql-2:3306 is being cloned from single-mysql:3306
** Stage DROP DATA: Completed
** Clone Transfer
    FILE COPY  ##################################################################  100%  Completed
    PAGE COPY  ##################################################################  100%  Completed
    REDO COPY  ##################################################################  100%  Completed

NOTE: mysql-2:3306 is shutting down...

* Waiting for server restart... ready
* mysql-2:3306 has restarted, waiting for clone to finish...
** Stage RESTART: Completed
* Clone process has finished: 365.82 MB transferred in 2 sec (182.91 MB/s)

** Configuring mysql-2:3306 to replicate from single-mysql:3306
** Waiting for new instance to synchronize with PRIMARY...

The instance 'mysql-2:3306' was added to the replicaset and is replicating from single-mysql:3306.
```

MySQL InnoDB ReplicaSet - examples

```
MySQL  localhost:33060+  2021-01-05 16:38:38
JS  rs.status()
{
    "replicaSet": {
        "name": "myreplicaset",
        "primary": "single-mysql:3306",
        "status": "AVAILABLE",
        "statusText": "All instances available.",
        "topology": {
            "mysql-2:3306": {
                "address": "mysql-2:3306",
                "instanceRole": "SECONDARY",
                "mode": "R/O",
                "replication": {
                    "applierStatus": "APPLIED_ALL",
                    "applierThreadState": "Slave has read all relay log; waiting for more updates",
                    "receiverStatus": "ON",
                    "receiverThreadState": "Waiting for master to send event",
                    "replicationLag": null
                },
                "status": "ONLINE"
            },
            "single-mysql:3306": {
                "address": "single-mysql:3306",
                "instanceRole": "PRIMARY",
                "mode": "R/W",
                "status": "ONLINE"
            }
        },
        "type": "ASYNC"
    }
}
```

MySQL Router

MySQL Router is very easy to configure, you only need to use the **`bootstrap`** *command !*

```
[root@single-mysql ~]# mysqlrouter --bootstrap clusteradmin@localhost:3306 --user=mysqlrouter
Please enter MySQL password for clusteradmin:
# Bootstrapping system MySQL Router instance...

- Creating account(s) (only those that are needed, if any)
- Verifying account (using it to run SQL queries that would be run by Router)
- Storing account in keyring
- Adjusting permissions of generated files
- Creating configuration /etc/mysqlrouter/mysqlrouter.conf

Existing configuration backed up to '/etc/mysqlrouter/mysqlrouter.conf.bak'

# MySQL Router configured for the InnoDB ReplicaSet 'myreplicaset'

After this MySQL Router has been started with the generated configuration

    $ /etc/init.d/mysqlrouter restart
or
    $ systemctl start mysqlrouter
or
    $ mysqlrouter -c /etc/mysqlrouter/mysqlrouter.conf

the cluster 'myreplicaset' can be reached by connecting to:

## MySQL Classic protocol

- Read/Write Connections: localhost:6446
- Read/Only Connections:  localhost:6447

## MySQL X protocol

- Read/Write Connections: localhost:64460
- Read/Only Connections:  localhost:64470
```

MySQL Router

As usual, this is also visible in MySQL Shell

```
JS rs.listRouters()
{
    "replicaSetName": "myreplicaset",
    "routers": {
        "single-mysql.mysqlpub.mysqlvcn.oraclevcn.com::system": {
            "hostname": "single-mysql.mysqlpub.mysqlvcn.oraclevcn.com",
            "lastCheckIn": "2021-01-05 18:36:03",
            "roPort": 6447,
            "roXPort": 64470,
            "rwPort": 6446,
            "rwXPort": 64460,
            "version": "8.0.22"
        }
    }
}
```

MySQL InnoDB ReplicaSet - Manual Failover

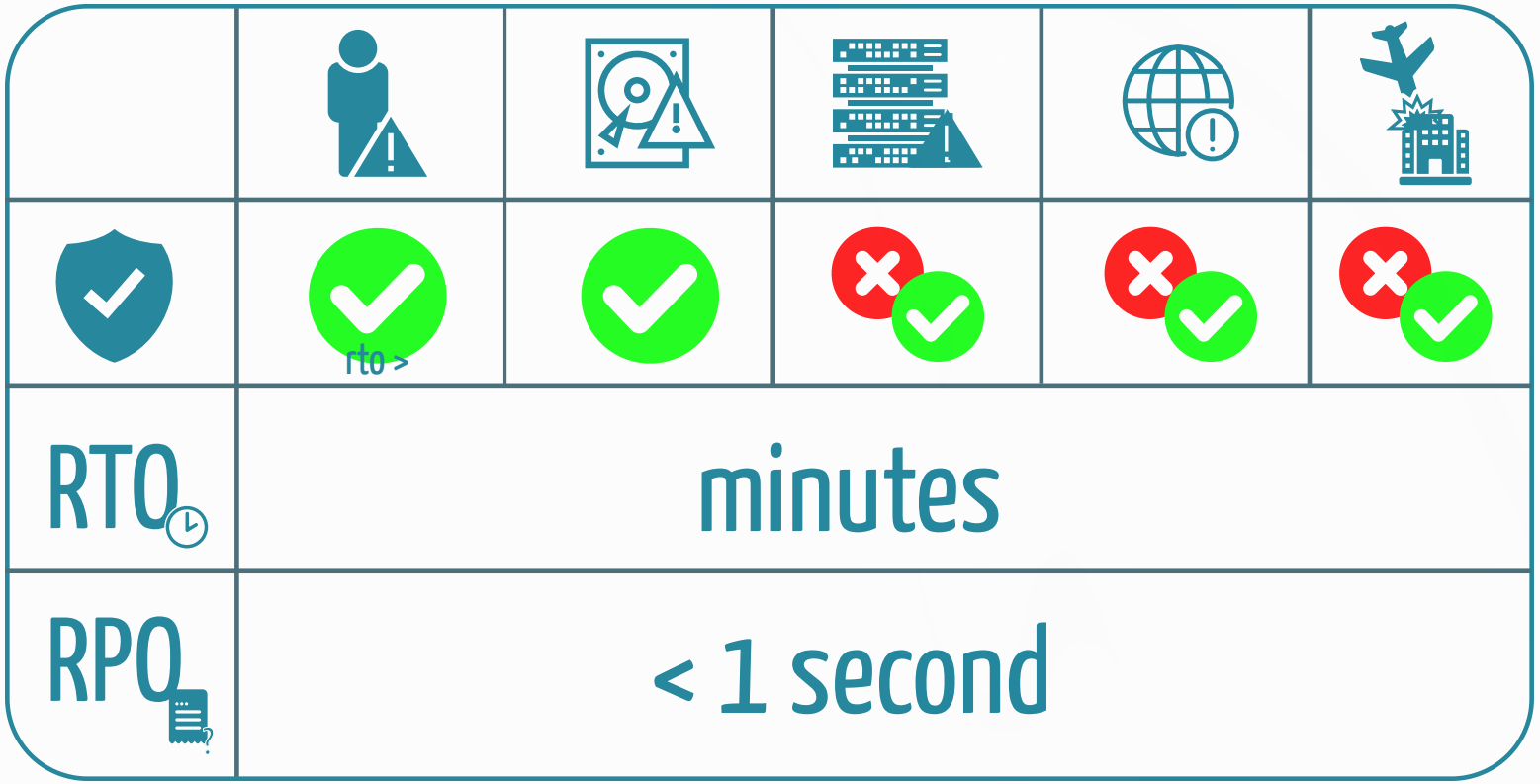
```
MySQL  localhost:6447  2021-01-05 18:41:28
 JS  rs.status()
ERROR: Unable to connect to the PRIMARY of the replicaset myreplicaset: MySQL Error 2003: Could not open connection to
 'single-mysql:3306': Can't connect to MySQL server on 'single-mysql' (111)
Cluster change operations will not be possible unless the PRIMARY can be reached.
If the PRIMARY is unavailable, you must either repair it or perform a forced failover.
See \help forcePrimaryInstance for more information.
WARNING: MYSQLSH 51118: PRIMARY instance is unavailable
{
    "replicaSet": {
        "name": "myreplicaset",
        "primary": "single-mysql:3306",
        "status": "UNAVAILABLE",
        "statusText": "PRIMARY instance is not available, but there is at least one SECONDARY that could be force-prom
oted.",
```

MySQL Shell is connected to MySQL InnoDB ReplicaSet via MySQL Router.

MySQL InnoDB ReplicaSet - Manual Failover

```
MySQL  localhost:6447  2021-01-05 18:41:32
JS  rs.forcePrimaryInstance()
* Connecting to replicaset instances
** Connecting to mysql-2:3306

* Waiting for all received transactions to be applied
** Waiting for received transactions to be applied at mysql-2:3306
* Searching instance with the most up-to-date transaction set
mysql-2:3306 has GTID set 5a47e14a-4f4b-11eb-97af-020017078dee:1-2639
mysql-2:3306 will be promoted to PRIMARY of the replicaset and the former PRIMARY will be invalidated.

* Checking status of last known PRIMARY
NOTE: single-mysql:3306 is UNREACHABLE
* Checking status of promoted instance
NOTE: mysql-2:3306 has status ERROR
* Checking transaction set status
* Promoting mysql-2:3306 to a PRIMARY...

* Updating metadata...

mysql-2:3306 was force-promoted to PRIMARY.
NOTE: Former PRIMARY single-mysql:3306 is now invalidated and must be removed from the replicaset.
* Updating source of remaining SECONDARY instances

Failover finished successfully.
```

MySQL RelicaSet



Next Level (level up)

*My service is **very** important, I'd like to be operational almost all the time (automatic failover) and never lose any data!*

Next Level (level up)

*My service is **very** important, I'd like to be operational almost all the time (automatic failover) and never lose any data!*

RTO → seconds

Next Level (level up)

*My service is **very** important, I'd like to be operational almost all the time (automatic failover) and never lose any data!*

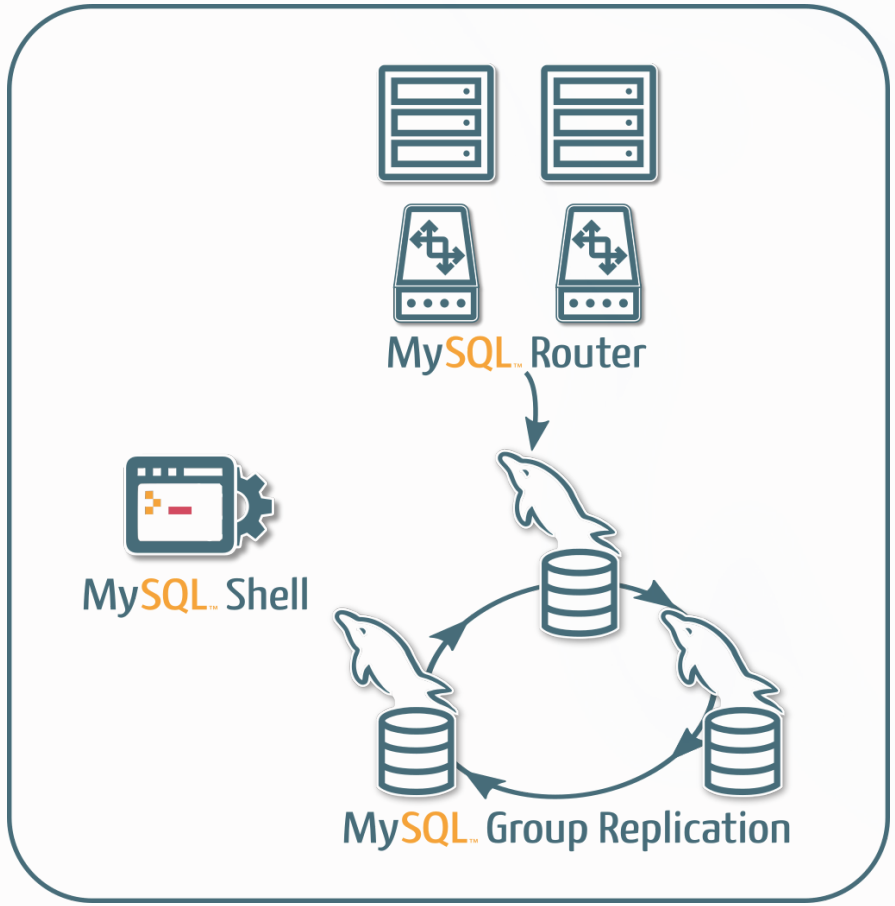
RTO → seconds

RPO → 0

MySQL InnoDB Cluster

- *Based on native Group Replication*
- *But easier.... easier is always better !*
- *Data provisioning included (clone)*
- *3 or more nodes (odd number)*
- ***Automatic Failover***
- *Uses MySQL Router*
- *Configurable Consistency Levels*

MySQL InnoDB Cluster



MySQL InnoDB Cluster

On the ReplicaSet's Primary Instance:

```
JS > rs=dba.getReplicaSet()
JS > rs.dissolve()
```

```
JS > cluster=dba.createCluster('myCluster')
JS > cluster.addInstance('mysql-2')
```

MySQL InnoDB Cluster

On the ReplicaSet's Primary Instance:

```
JS > rs=dba.getReplicaSet()
JS > rs.dissolve()
```

```
JS > cluster=dba.createCluster('myCluster')
JS > cluster.addInstance('mysql-2')
```

We can now add a third instance to benefit from the automatic failover:

```
JS > dba.configureInstance()
```

From any member of the cluster:

```
JS > cluster=dba.getCluster()
JS > cluster.addInstance('mysql-3')
```

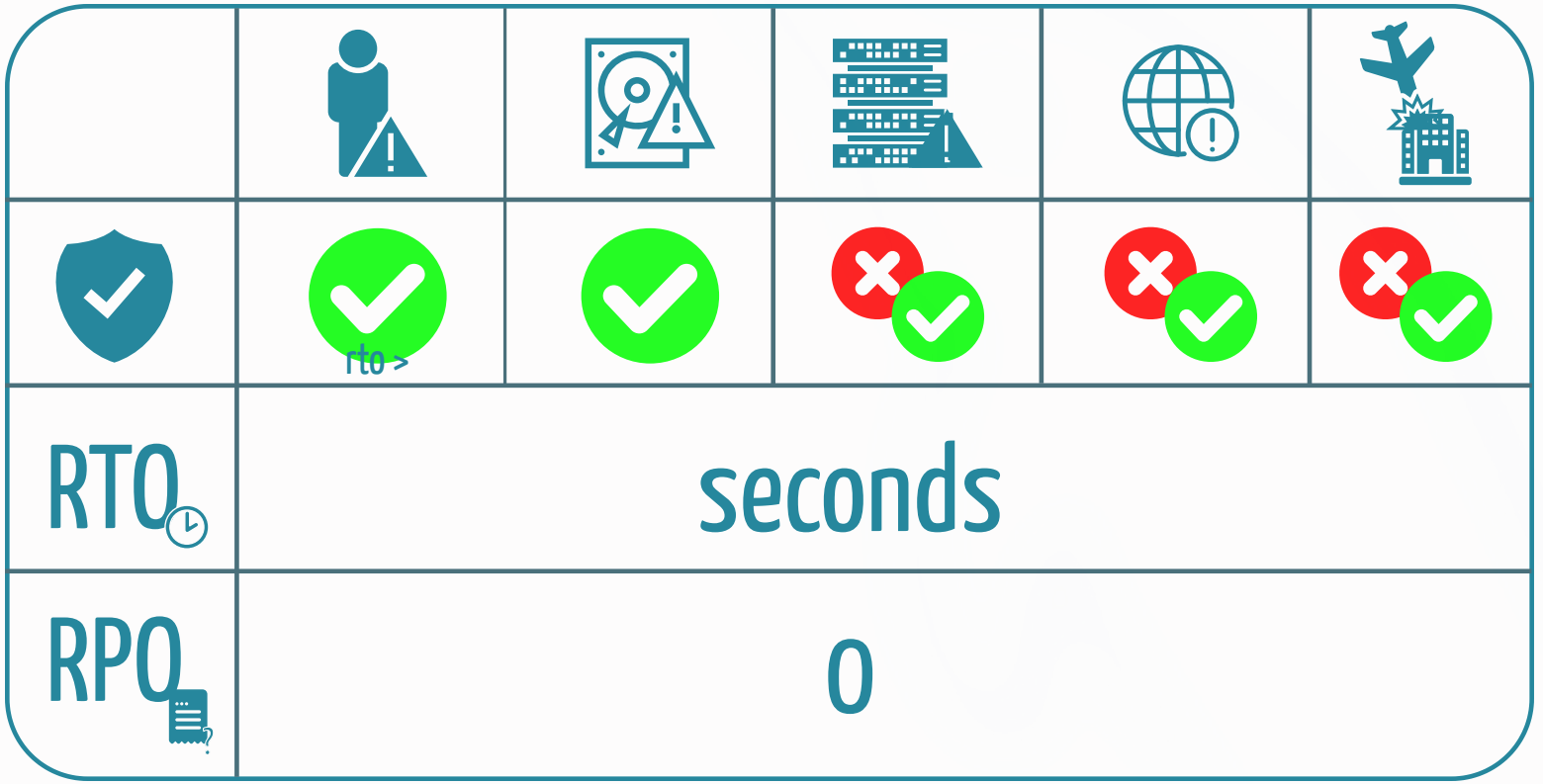
Finally, we can configure MySQL Router again:

```
# mysqlrouter --bootstrap \
  clusteradmin@single-mysql:3306 \
  --conf-use-gr-notifications \
  --user mysqlrouter --force
# systemctl restart mysqlrouter
```

MySQL InnoDB Cluster

```
JS > cluster.status()
{
    "clusterName": "mycluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "single-mysql:3306",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "mysql-2:3306": {
                "address": "mysql-2:3306",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.22"
            },
            "mysql-3:3306": {
                "address": "mysql-3:3306",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.22"
            },
            "single-mysql:3306": {
                "address": "single-mysql:3306",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.22"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "single-mysql:3306"
}
```

MySQL InnoDB Cluster



MySQL InnoDB Cluster: consistency levels

- *MySQL InnoDB Cluster operates, by default, in single-primary mode:*
  - *Apps and users write to the primary instance*
- *What guarantees do we have that:*
  - *We always read the most up-to-date data?*
  - *We do not read stale data of an evicted server?*
  - *Dirty reads do not happen?*

MySQL InnoDB Cluster: consistency levels

The default consistency level is **EVENTUAL**. *This means that there is no synchronization point for the transactions, when you perform a* ***write*** *on a node, if you immediately read the same data on another node, it is eventually there.*

```
mysql> show variables like
       'group_replication_consistency';
+--------------------------------+----------+
| Variable_name                  | Value    |
+--------------------------------+----------+
| group_replication_consistency | EVENTUAL |
+--------------------------------+----------+
```

MySQL InnoDB Cluster: consistency levels

The default consistency level is **EVENTUAL**. *This means that there is no synchronization point for the transactions, when you perform a* ***write*** *on a node, if you immediately read the same data on another node, it is eventually there.*

```
mysql> show variables like
       'group_replication_consistency';
+--------------------------------+----------+
| Variable_name                  | Value    |
+--------------------------------+----------+
| group_replication_consistency | EVENTUAL |
+--------------------------------+----------+
```

Since MySQL 8.0.16, we have the possibility to set the synchronization point at read or at write or both (globally or for a session).

Consistency: BEFORE (READ)

As a DBA, I want to load balance my reads without deploying additional restrictions on which server I read from to avoid reading stale data, my group writes are much more than my group reads.

Consistency: BEFORE (READ)

As a DBA, I want to load balance my reads without deploying additional restrictions on which server I read from to avoid reading stale data, my group writes are much more than my group reads.

As a developer, I want specific transactions in my workload to always read up-to-date data from the group, so that whenever that sensitive data is updated, I will enforce that reads shall read the most up to date value.

Consistency: BEFORE (READ)

```
MySQL 8.0.17  mysql1:33060+  2019-09-09 16:11:51
JS cluster.setOption("consistency","BEFORE")
Setting the value of 'consistency' to 'BEFORE' in all ReplicaSet members ...

Successfully set the value of 'consistency' to 'BEFORE' in the 'default' ReplicaSet.
```

Consistency: AFTER (WRITE)

I want to load balance my reads without deploying additional restrictions on which server I read from to avoid reading stale data, my group writes are much less than my group reads.

Consistency: AFTER (WRITE)

I want to load balance my reads without deploying additional restrictions on which server I read from to avoid reading stale data, my group writes are much less than my group reads.

I have a group that mostly does reads-only, I want my read-write transactions to be applied everywhere once they commit, so that subsequent reads are done on up-to-date data that includes my latest write. Without paying at reads.

Consistency: AFTER (WRITE)

```
MySQL 8.0.17  mysql1:33060+  2019-10-13 13:28:49
JS cluster.setOption("consistency","AFTER")
Setting the value of 'consistency' to 'AFTER' in all ReplicaSet members ...

Successfully set the value of 'consistency' to 'AFTER' in the 'default' ReplicaSet.
```

Consistency: BEFORE_AND_AFTER

I want that my application to replicate data as close as possible to synchronous.
And of course I'm OK to pay the required price !

Consistency: BEFORE_AND_AFTER

It's also possible and recommended for the most restrictive levels to define it at statement session level:

```
MySQL 8.0.17 mysql1:33060+ 2019-09-09 20:57:37
SQL set group_replication_consistency=BEFORE_AND_AFTER;
Query OK, 0 rows affected (0.0010 sec)
MySQL 8.0.17 mysql1:33060+ 2019-09-09 20:58:07
SQL insert into clusterdemo.demo (hostname) values ('lefred');
Query OK, 1 row affected (0.1135 sec)
```

Next Level (level up)

And if something happens to our data center ??

Next Level (level up)

And if something happens to our data center ??

How can we deploy a Disaster Recovery solution ?

Next Level (level up)

And if something happens to our data center ??

How can we deploy a Disaster Recovery solution ?

MySQL InnoDB ClusterSet !!

MySQL InnoDB ClusterSet

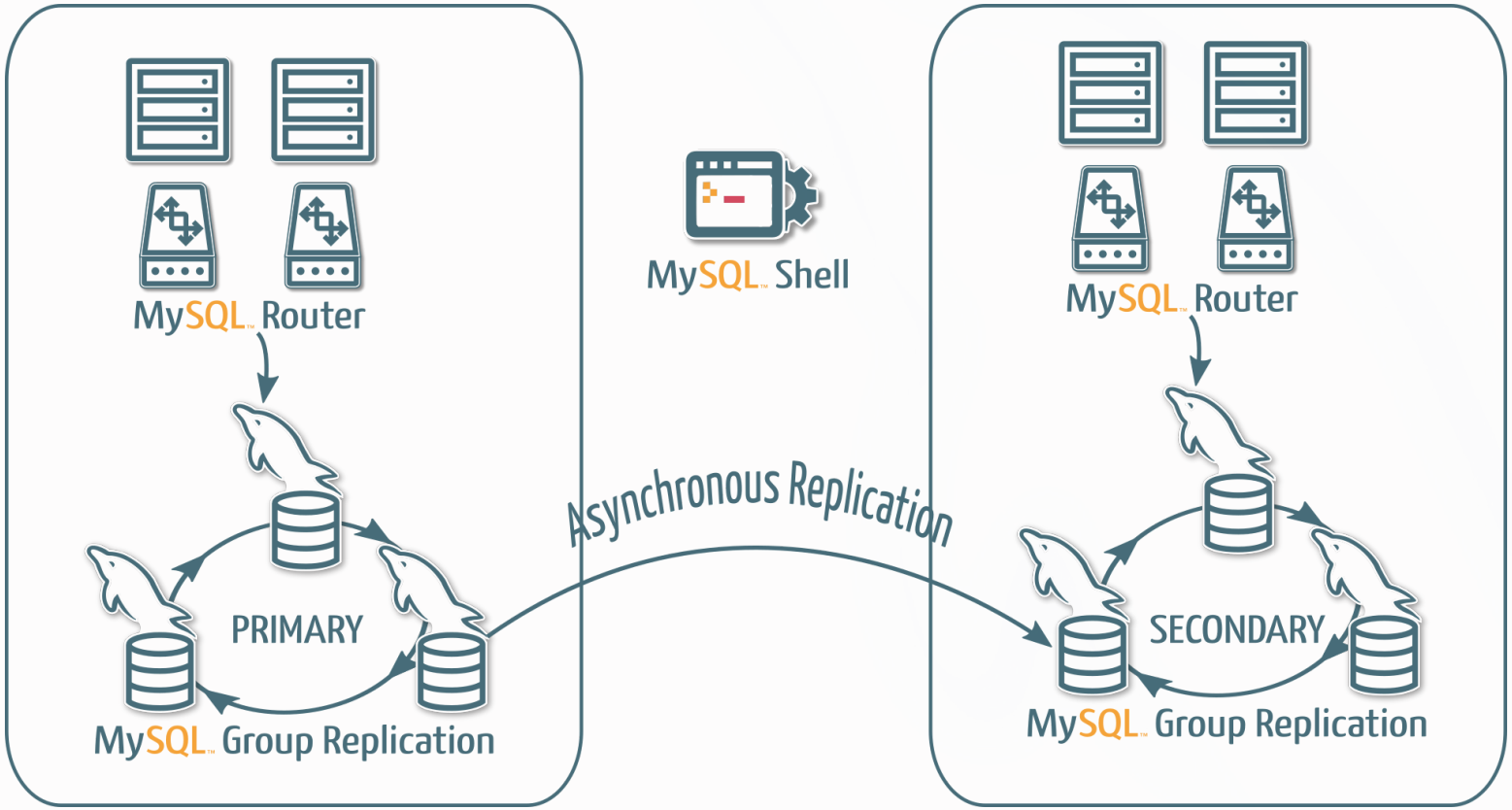
The Disaster Recovery Evolution !

- *available since MySQL 8.0.27*
- *fencing support since 8.0.28*

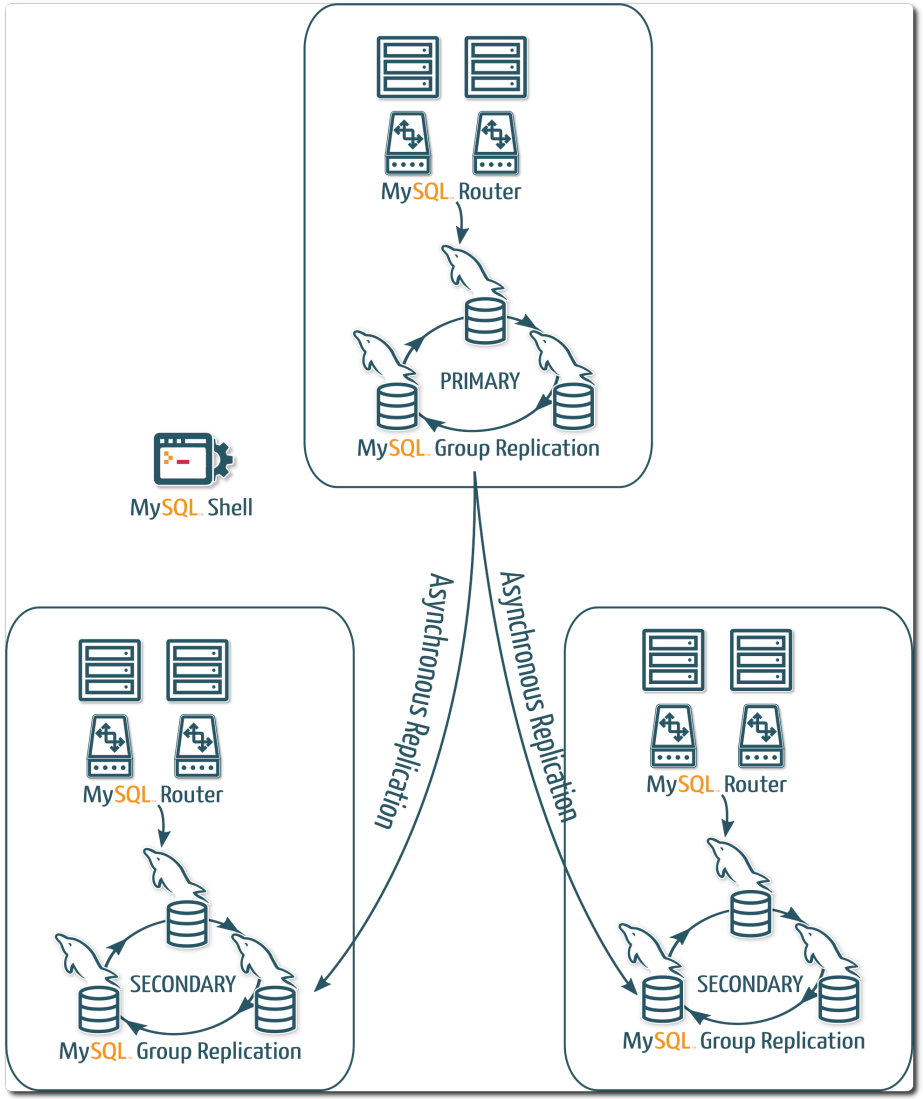
The result of several transformations during previous releases:

- *8.0.22: Automatic Connection Failover (Async Replication Channels)*
- *8.0.23: Automatic Connection Failover (Async Replication Channels using Group Replication)*
- *8.0.24: transformation of* `skip-replica-start` *en global, persistent and read-only variable*
- *8.0.26: Actions added for Group Replication's members (ex:* `super_read_only` *configurable)*
- *8.0.26: Specific UUID added for structural events change of the Group (*`View_change_log_event`*)*
- *8.0.27: Asynchronous Replication Channel automatically follows the Primary.*

MySQL InnoDB ClusterSet



More complex architectures can be deployed !



MySQL InnoDB ClusterSet

On the Cluster's Primary Instance:

```
JS > cluster=dba.getCluster()
JS > cs=cluster.createClusterSet('mydomain')
```

We can already check the status:

```
JS > cs.status()
```

Now let's create the Replica Cluster:

```
JS > cluster2=cs.createReplicaCluster(
       'dc2-mysql-4', 'mycluster2')
```

MySQL InnoDB ClusterSet

On the Cluster's Primary Instance:

```
JS > cluster=dba.getCluster()
JS > cs=cluster.createClusterSet('mydomain')
```

We can already check the status:

```
JS > cs.status()
```

Now let's create the Replica Cluster:

```
JS > cluster2=cs.createReplicaCluster(
       'dc2-mysql-4', 'mycluster2')
```

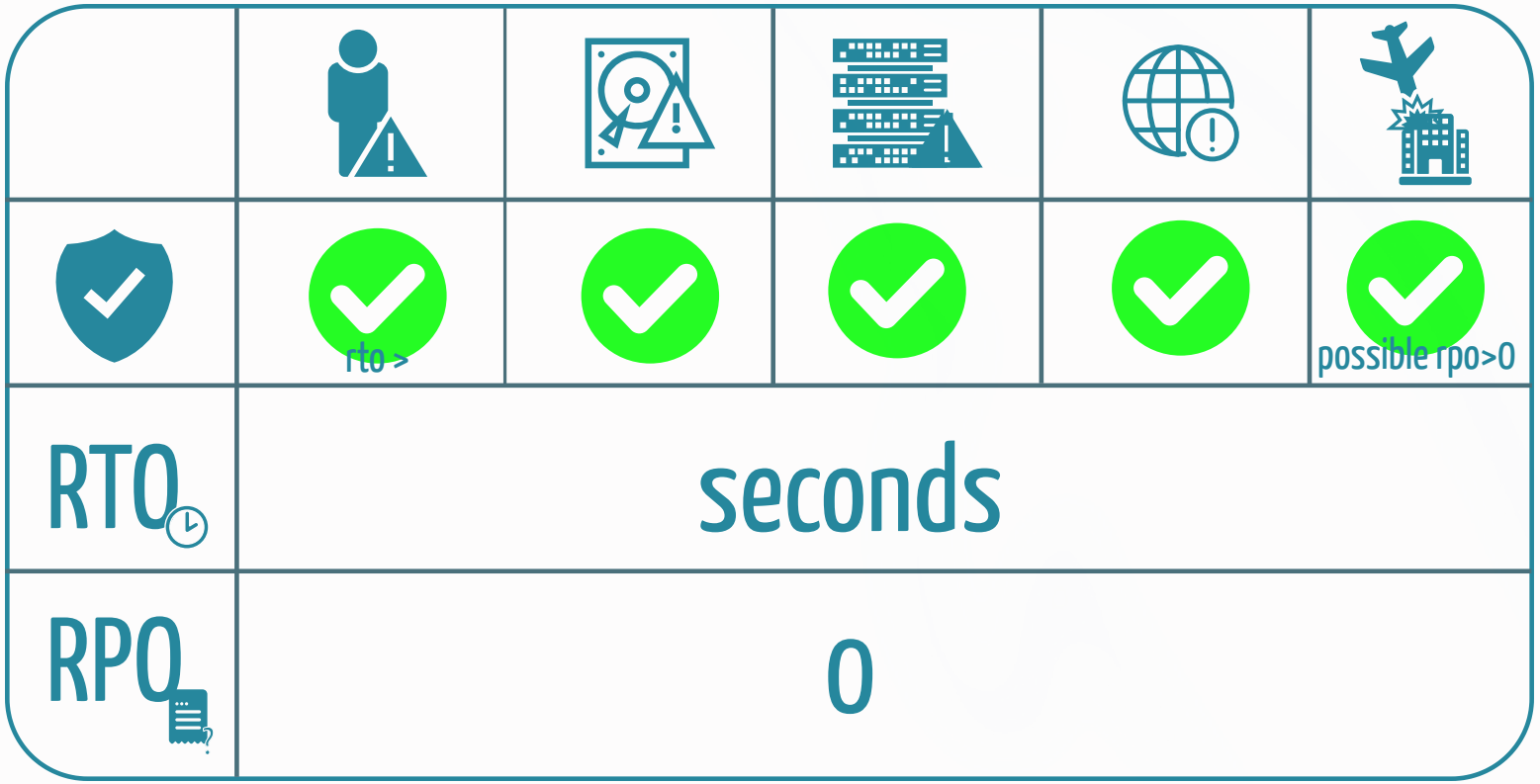
And we add the other 2 instances:

```
JS > cluster2.addInstance(
       'dc2-mysql-5')

JS > cluster2.addInstance(
       'dc2-mysql-6')
```

Finally, we need to bootstrap another time MySQL Router:

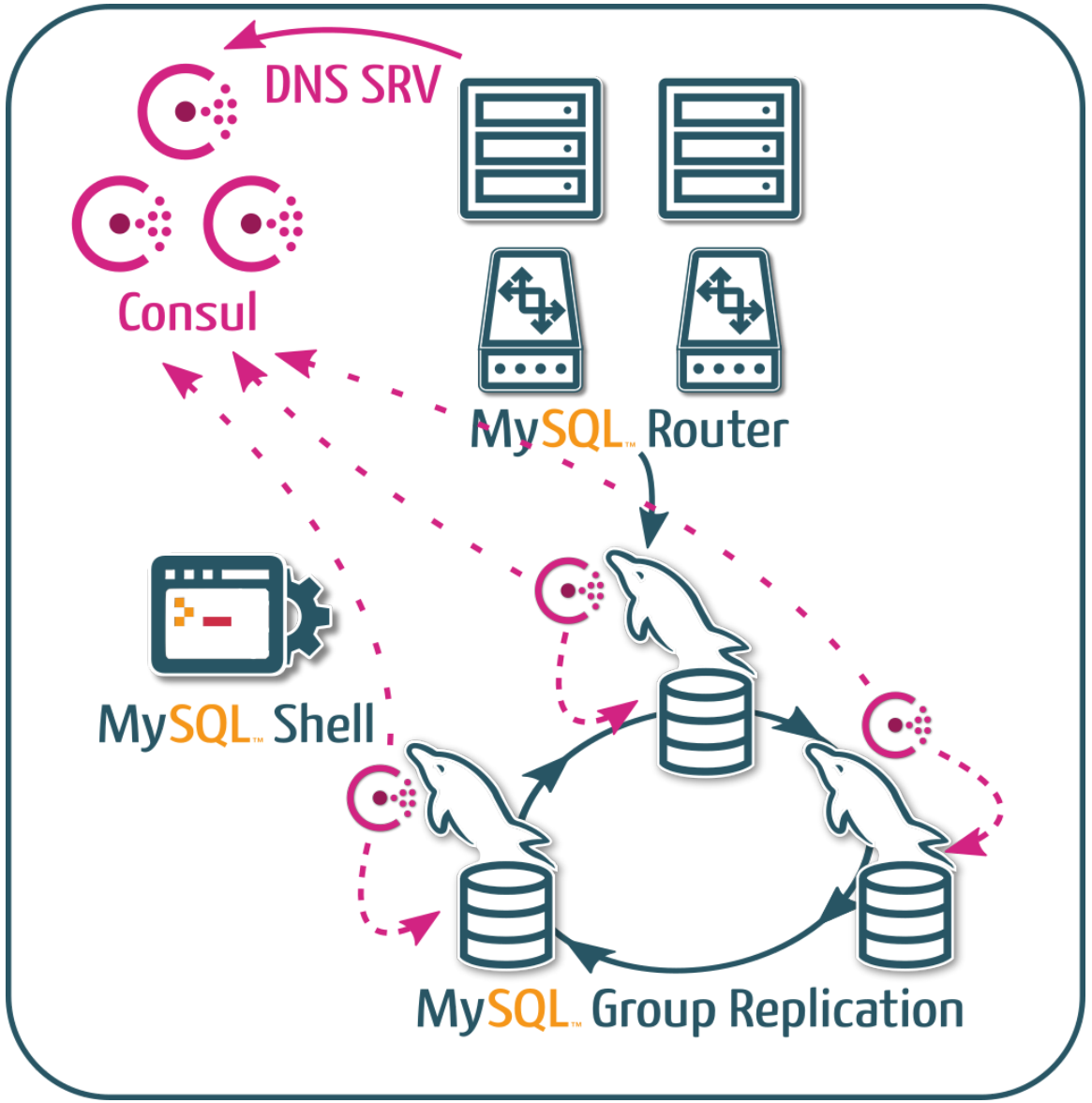
```
# mysqlrouter --bootstrap \
  clusteradmin@single-mysql:3306 \
  --conf-use-gr-notifications \
  --user mysqlrouter --force
# systemctl restart mysqlrouter
```

MySQL InnoDB ClusterSet



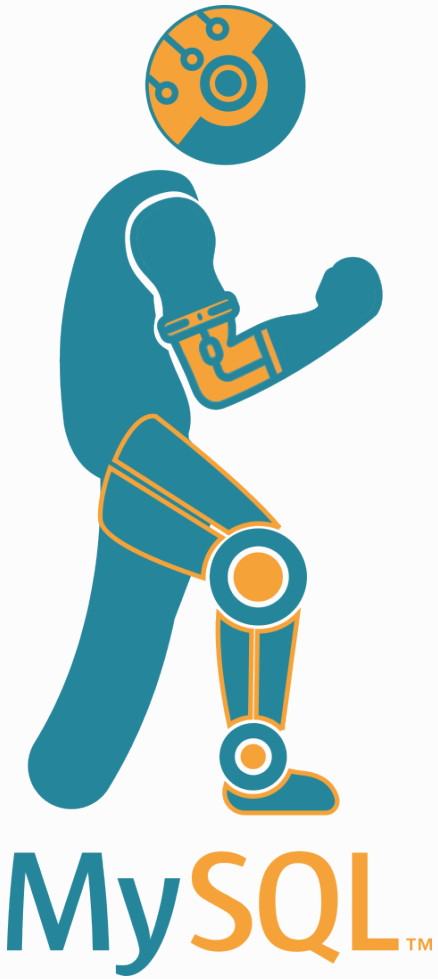
Extra

Since MySQL 8.0.19, the connectors also support **`dns-srv`** *which, together with a discovery service such as Consul, can replace MySQL Router when it is impossible to install it on the application servers:*



More with Innovation Releases

MySQL InnoDB Read Replicas

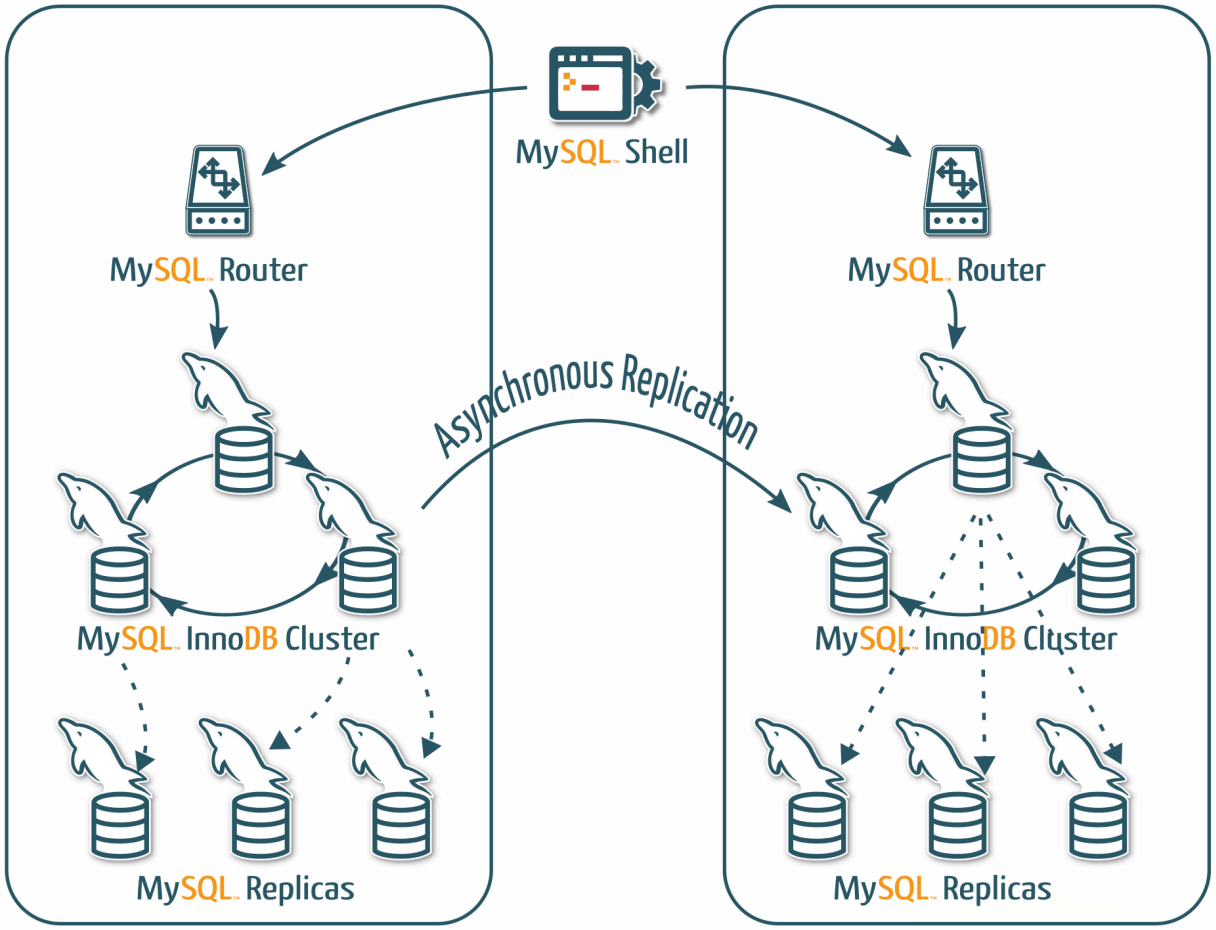


Scaling Reads Evolution !

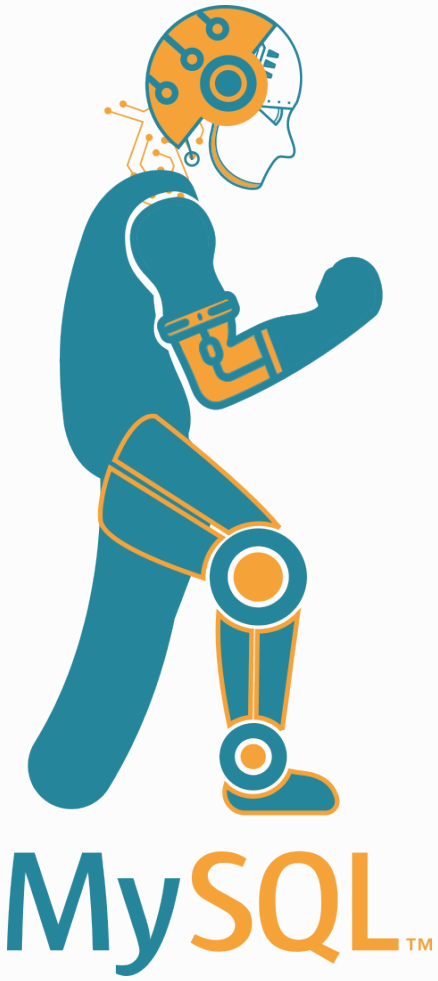
- *available since MySQL 8.1.0*
- *single or multiple replicas*
- *replicate from Primary or not*

MySQL 8.1 - Read Replicas

*Added support for **Read Replicas** with MySQL InnoDB Cluster and MySQL InnoDB ClusterSet:*



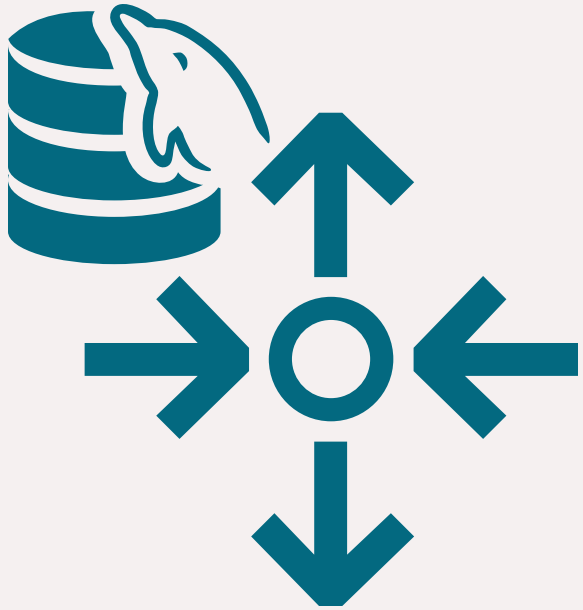
MySQL Transparent Read/Write Splitting



Read / Write Split Evolution !

- *available since MySQL 8.2.0*
- *long expected feature*
- *totally transparent*

MySQL 8.2 - Transparent Read/Write Splitting

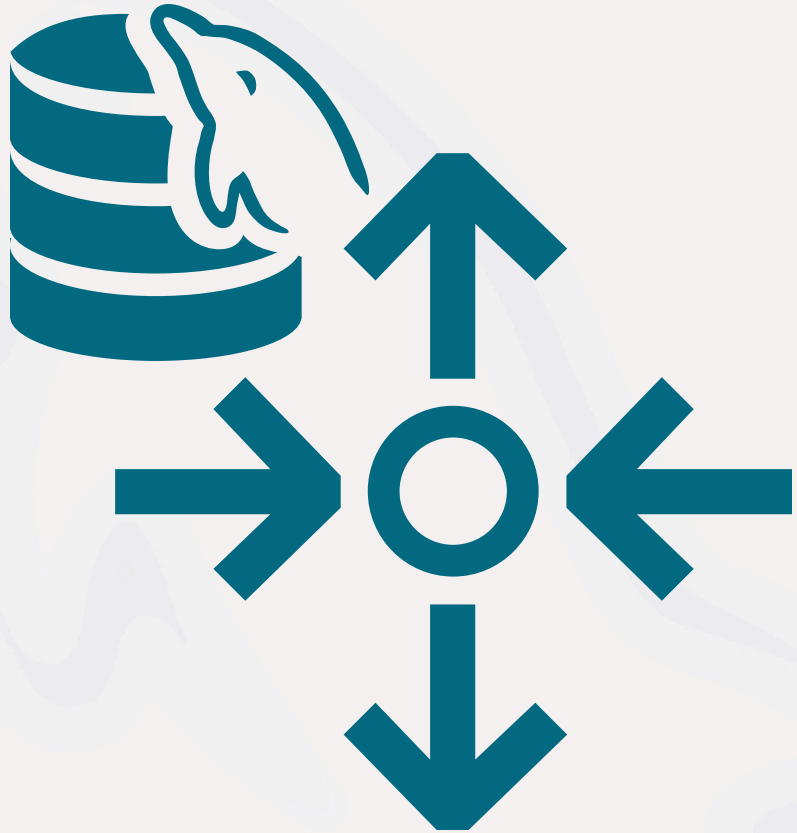


New routing policy with transparent read/write splitting:

```
[routing:bootstrap_rw_split]
bind_address=0.0.0.0
bind_port=6450
destinations=
  metadata-cache://myClusterSet/?role=PRIMARY_AND_SECONDARY
routing_strategy=round-robin
protocol=classic
connection_sharing=1
client_ssl_mode=PREFERRED
server_ssl_mode=PREFERRED
access_mode=auto
```

MySQL 8.2 - Transparent Read/Write Splitting

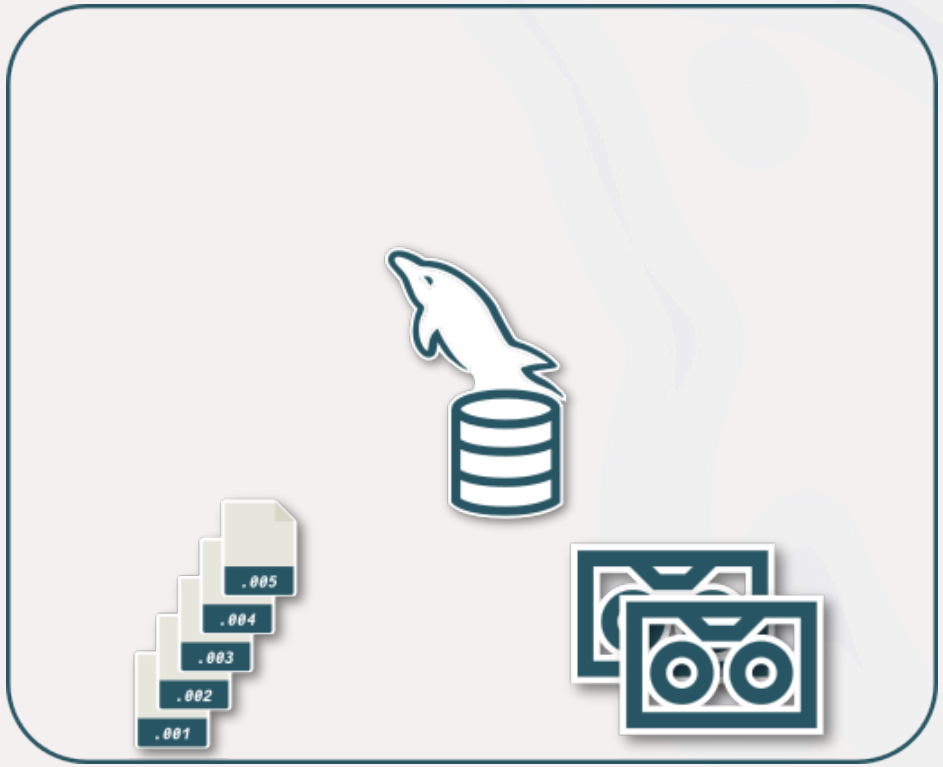
```
JS > cs.listRouters()
{
 "domainName": "myClusterSet",
 "routers": {
   "dell::system": {
      "hostname": "dell",
      "lastCheckIn":
         "2023-09-17 11:40:36",
      "roPort": "6447",
      "roXPort": "6449",
      "rwPort": "6446",
      "rwSplitPort": "6450",
      "rwXPort": "6448",
      "targetCluster": null,
      "version": "8.2.0"
   }
 }
}
```

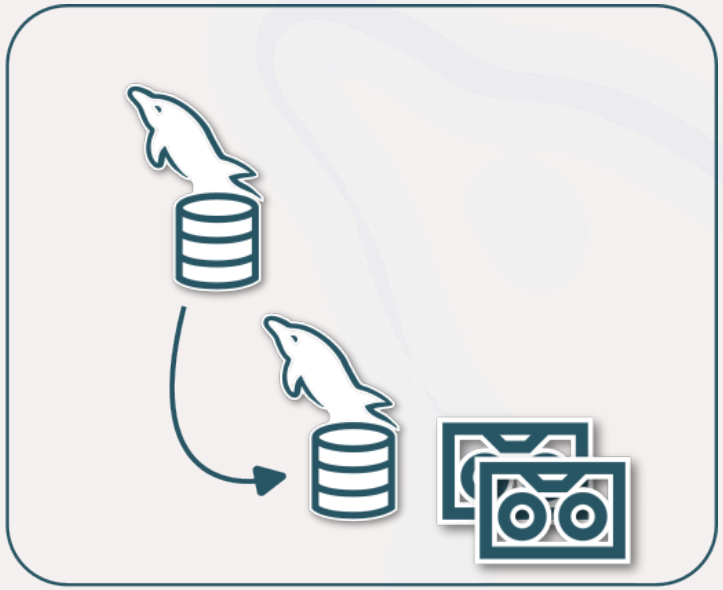


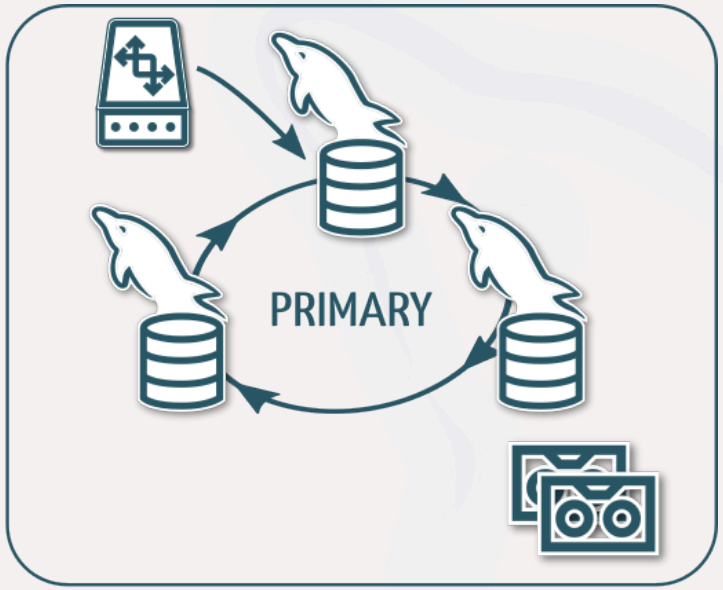
MySQL 8.2 - Transparent Read/Write Splitting

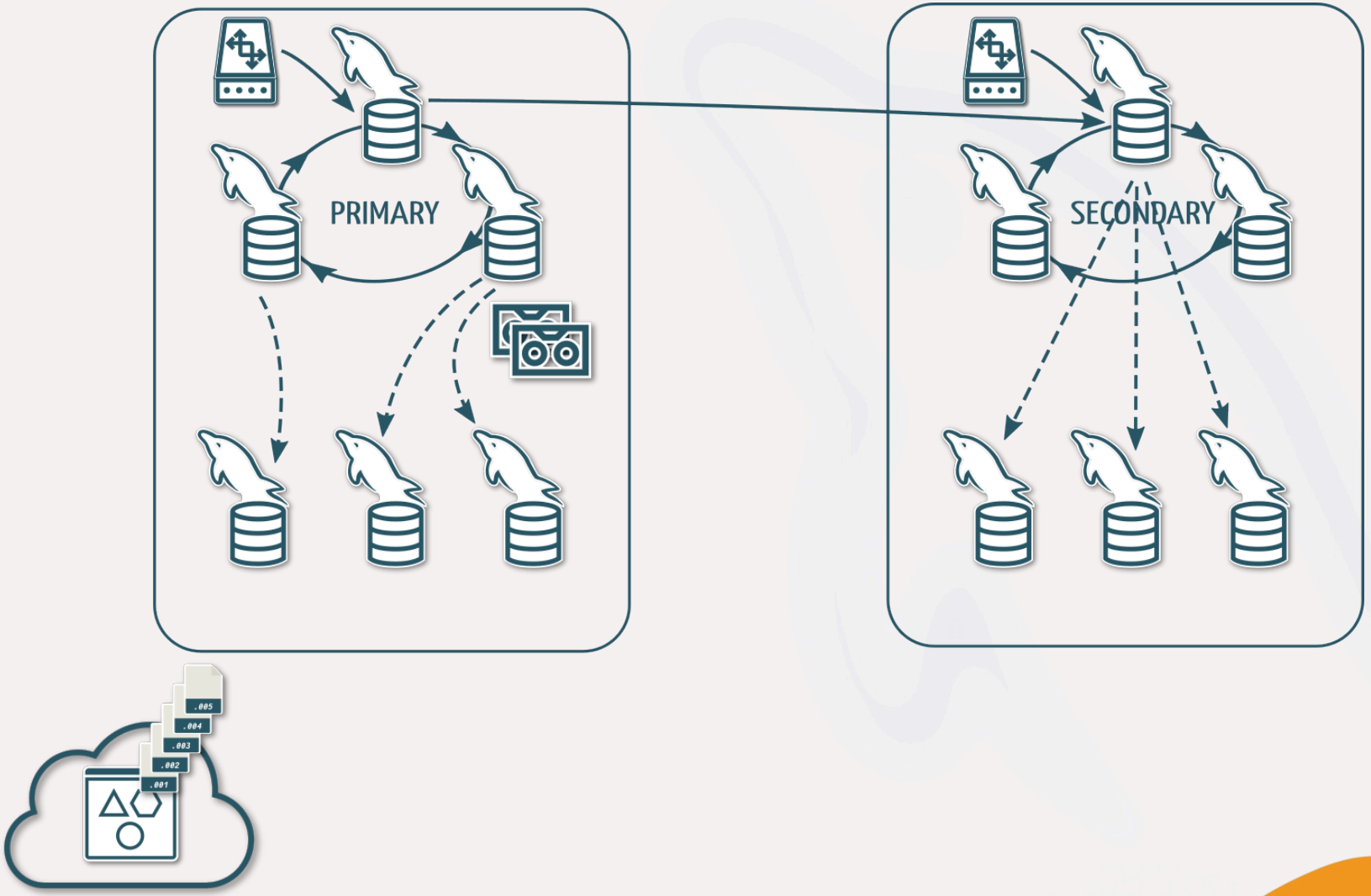
```
MySQL> \c root@127.0.0.1:6450
MySQL> 127.0.0.1:6450 > select @@port, member_role
 from performance_schema.replication_group_members where member_id=@@server_uuid;
+--------+-------------+
| @@port | member_role |
+--------+-------------+
|   3330 | SECONDARY   |
+--------+-------------+
1 row in set (0.0034 sec)

MySQL> 127.0.0.1:6450 > start transaction;

MySQL> 127.0.0.1:6450 > select @@port, member_role
 from performance_schema.replication_group_members where member_id=@@server_uuid;
+--------+-------------+
| @@port | member_role |
+--------+-------------+
|   3310 | PRIMARY     |
+--------+-------------+
1 row in set (0.0016 sec)
```

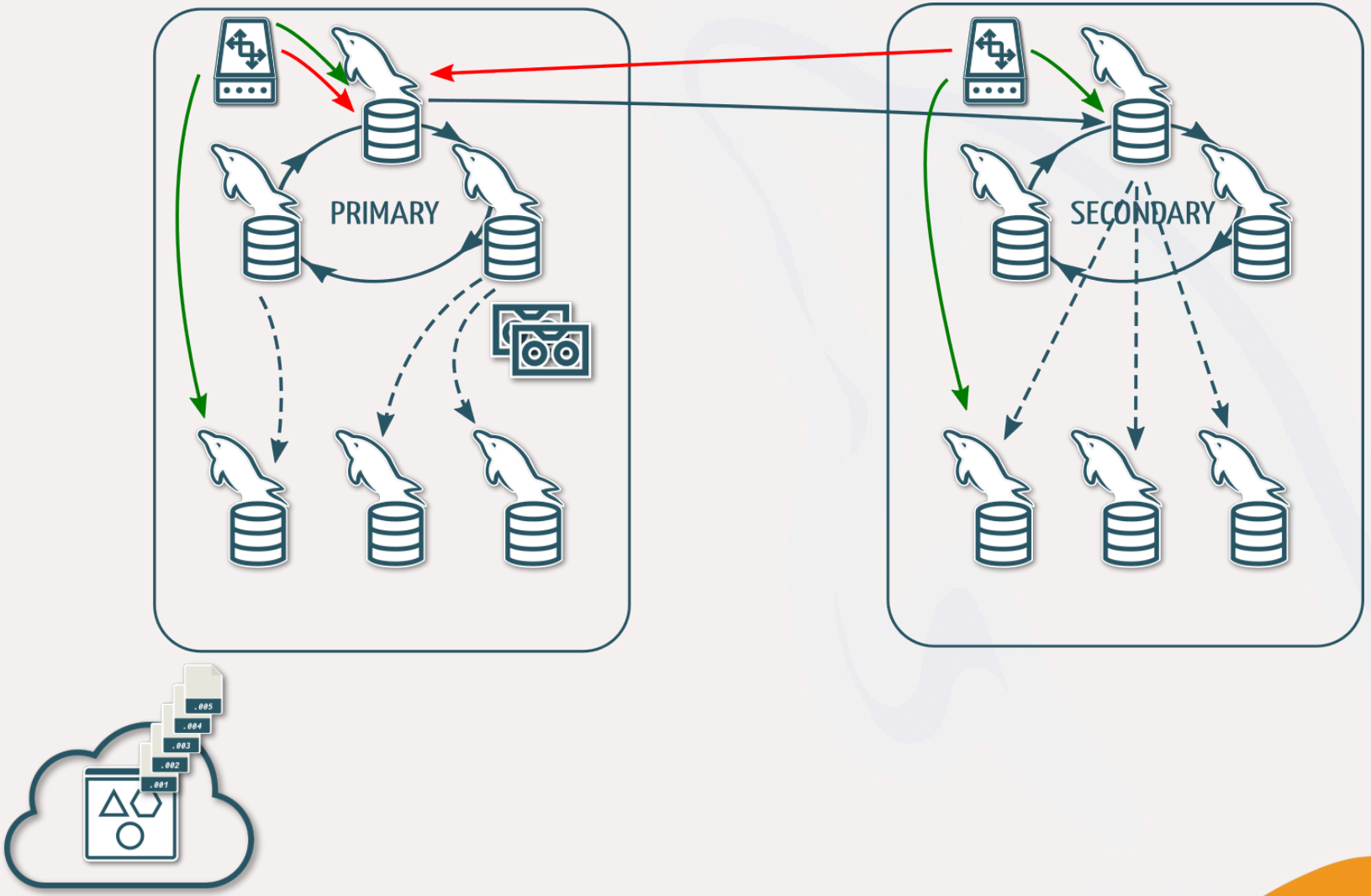
Evolution Summary



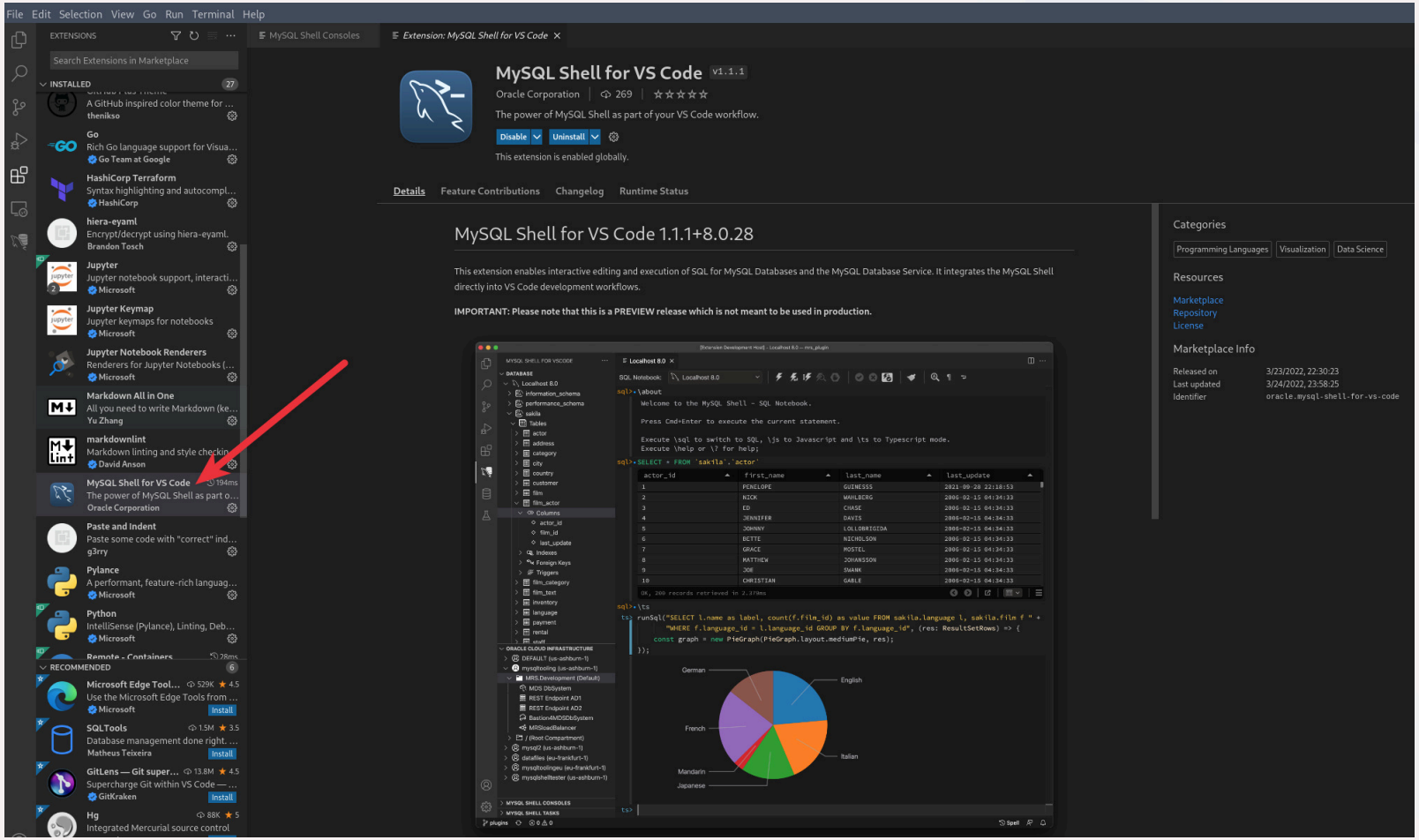


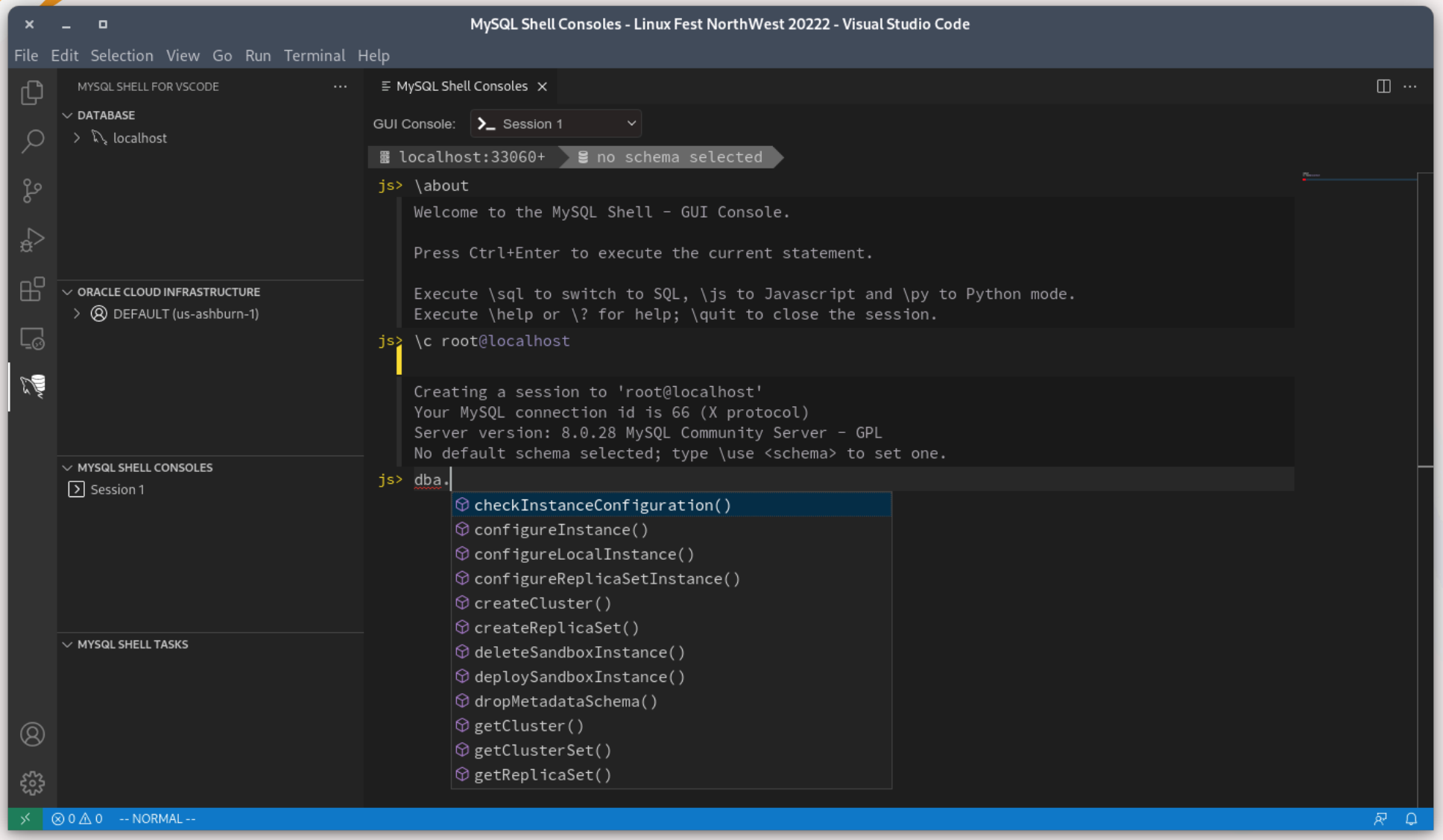






MySQL Shell is now included in Visual Studio Code:





Share your ❤️ to MySQL

#mysql #MySQLCommunity

Join our slack channel!

bit.ly/mysql-slack

Questions ?