ORACLE

MySQL Belgian Days 2024

MySQL Router

Explore The Secrets

Miguel Araújo
Senior Principal Software Engineer
MySQL, Oracle
February 2, 2024

Safe Harbor Statement

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 and timing of any features or functionality described for Oracle's product remains at the sole discretion of Oracle.

$ whoami && history

```
~/belgian_days_24
$ whoami
miguel_araujo
```

$ whoami && history

```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E

20.09.2015    MySQL Router 1st labs release
```

$ whoami && history

```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E

20.09.2015    MySQL Router 1st labs release

06.09.2016    2.1.0 labs: InnoDB Cluster integration
```

$ whoami && history

```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E
20.09.2015    MySQL Router 1st labs release
06.09.2016    2.1.0 labs: InnoDB Cluster integration
12.04.2017    2.1.3 GA: 1st GA release
```

$ whoami && history

```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E
20.09.2015    MySQL Router 1st labs release
06.09.2016    2.1.0 labs: InnoDB Cluster integration
12.04.2017    2.1.3 GA: 1st GA release
22.07.2019    8.0.17 GA: Support for GR notifications, REST API
```

$ whoami && history

```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E

20.09.2015    MySQL Router 1st labs release

06.09.2016    2.1.0 labs: InnoDB Cluster integration

12.04.2017    2.1.3 GA: 1st GA release

22.07.2019    8.0.17 GA: Support for GR notifications, REST API

19.10.2021    8.0.27 GA: Support for InnoDB ClusterSet
```

$ whoami && history

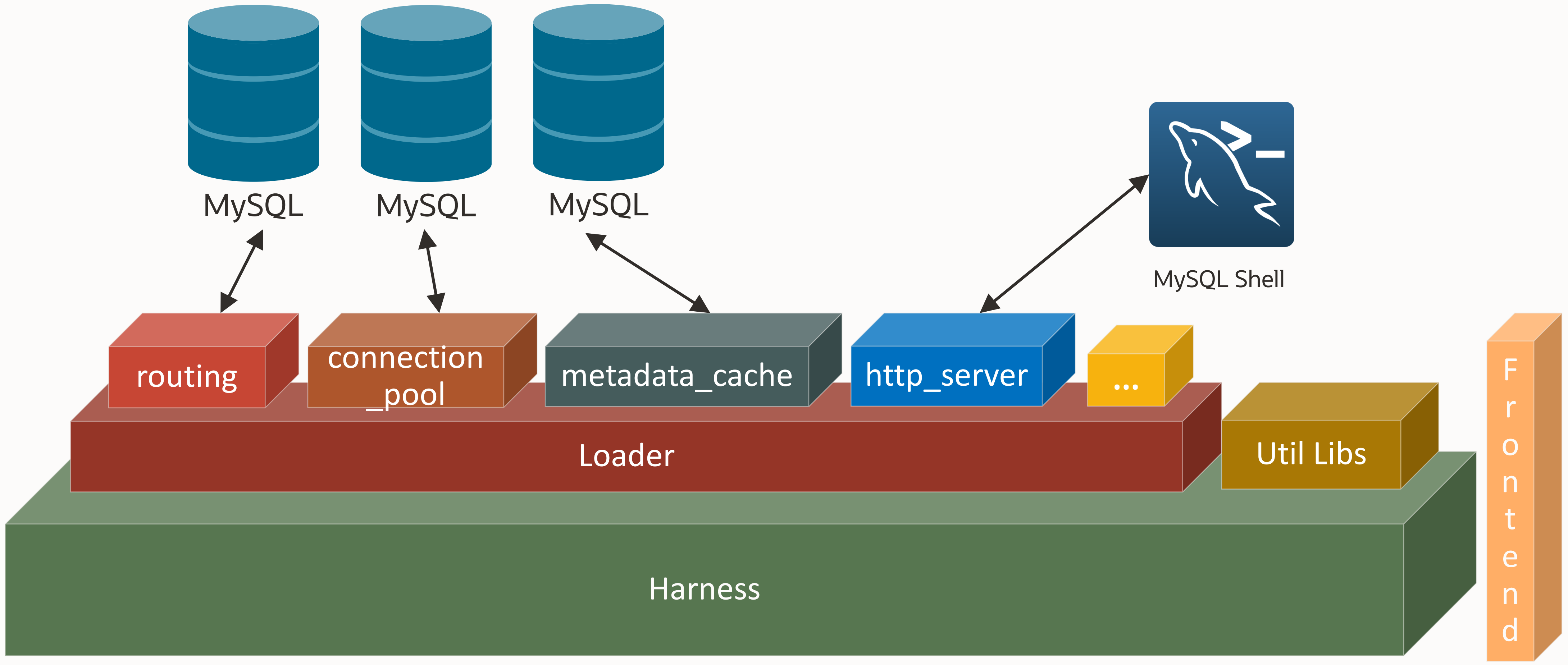
```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E

20.09.2015    MySQL Router 1st labs release

06.09.2016    2.1.0 labs: InnoDB Cluster integration

12.04.2017    2.1.3 GA: 1st GA release

22.07.2019    8.0.17 GA: Support for GR notifications, REST API

19.10.2021    8.0.27 GA: Support for InnoDB ClusterSet

18.07.2023    8.1.0: Support for InnoDB Cluster Read Replicas, Statement tracing
```

$ whoami && history

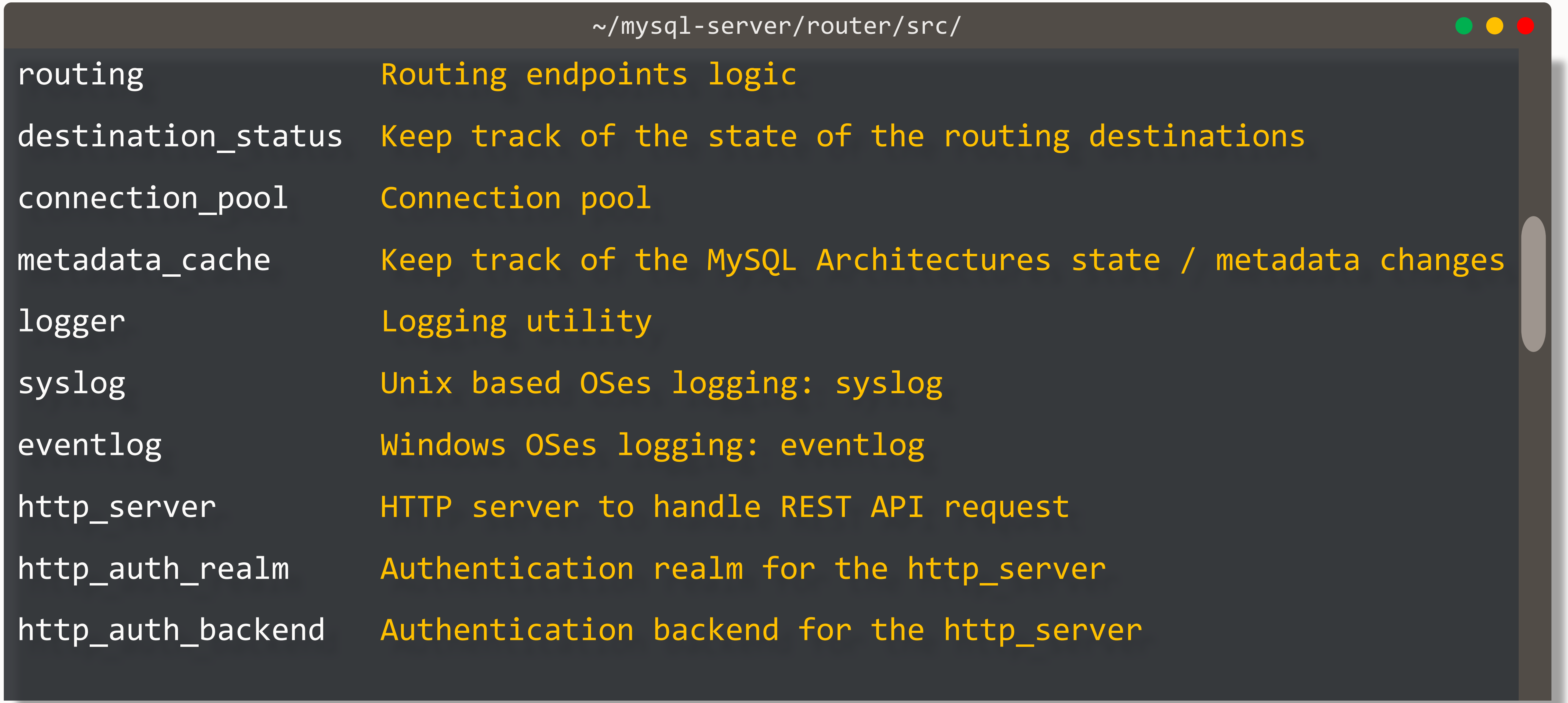
```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E
20.09.2015    MySQL Router 1st labs release
06.09.2016    2.1.0 labs: InnoDB Cluster integration
12.04.2017    2.1.3 GA: 1st GA release
22.07.2019    8.0.17 GA: Support for GR notifications, REST API
19.10.2021    8.0.27 GA: Support for InnoDB ClusterSet
18.07.2023    8.1.0: Support for InnoDB Cluster Read Replicas, Statement tracing
25.10.2023    8.2.0: R/W Splitting
```

Technicalities

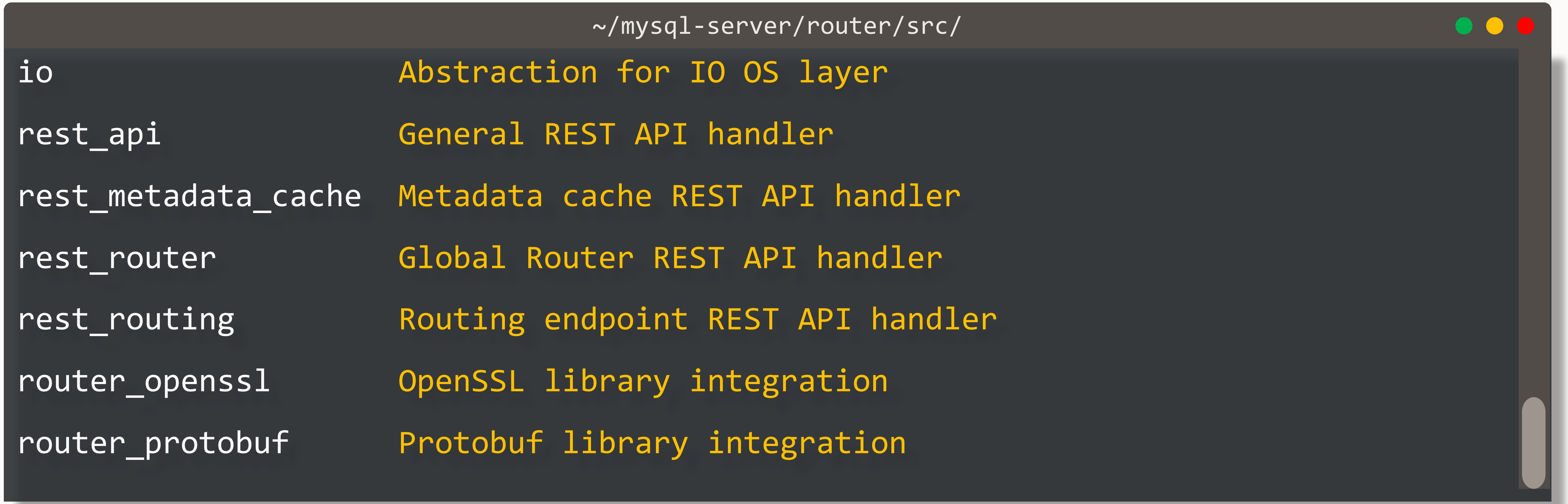
Architecture



Built-in plugins


```
~/mysql-server/router/src/

routing             Routing endpoints logic
destination_status  Keep track of the state of the routing destinations
connection_pool     Connection pool
metadata_cache      Keep track of the MySQL Architectures state / metadata changes
logger              Logging utility
syslog              Unix based OSes logging: syslog
eventlog            Windows OSes logging: eventlog
http_server         HTTP server to handle REST API request
http_auth_realm     Authentication realm for the http_server
http_auth_backend   Authentication backend for the http_server
```

Built-in plugins

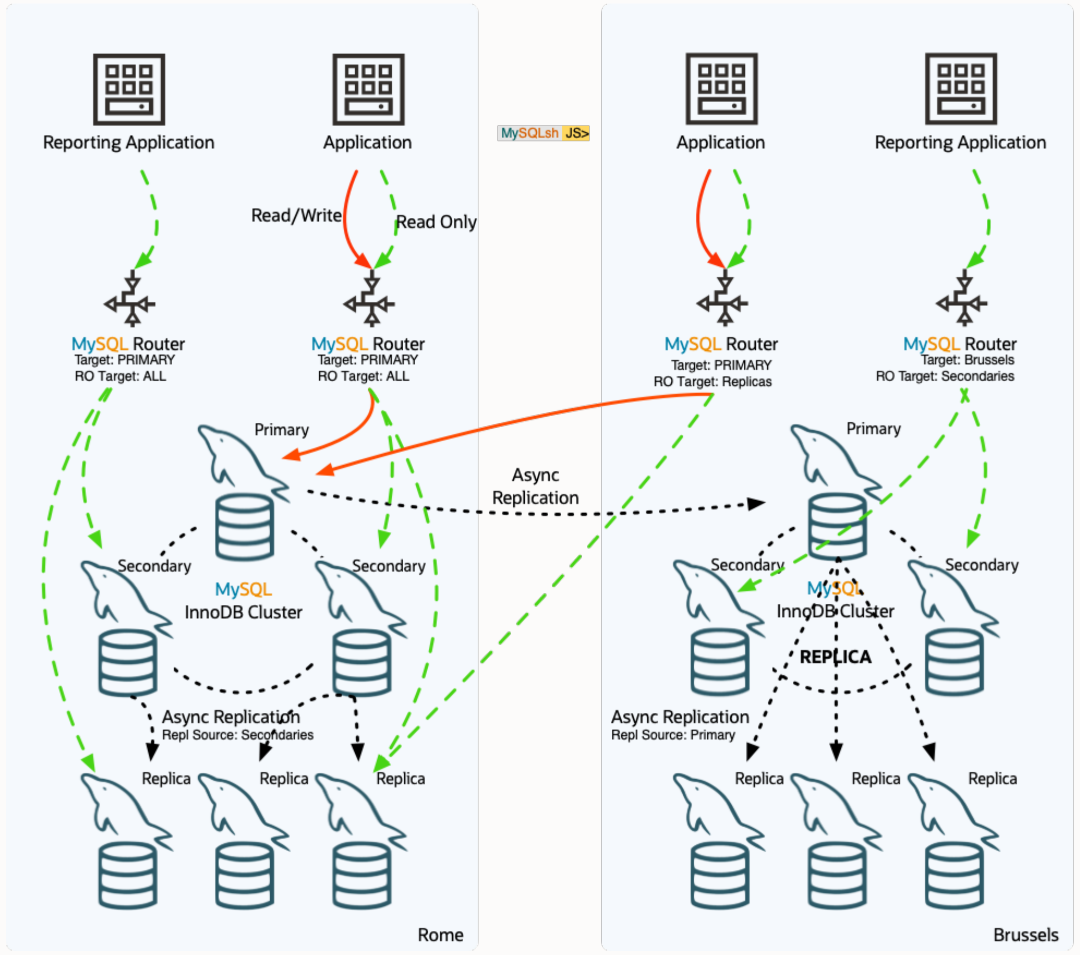


~/mysql-server/router/src/

```
io                  Abstraction for IO OS layer
rest_api            General REST API handler
rest_metadata_cache Metadata cache REST API handler
rest_router         Global Router REST API handler
rest_routing        Routing endpoint REST API handler
router_openssl      OpenSSL library integration
router_protobuf     Protobuf library integration
```

Router and MySQL Architectures

Core component of MySQL Architectures



Transparent access to Database Architecture

- Transparent client connection routing
  - Load balancing
  - Application connection failover
  - Little to no configuration needed
- Stateless design
  - Part of the application stack
- Full integration into MySQL Architectures
  - InnoDB Cluster
  - InnoDB ReplicaSet
  - InnoDB ClusterSet
  - InnoDB Cluster Read Replicas **New in 8.1.0 !**
- 3 TCP Ports:
  - PRIMARY traffic
  - SECONDARY traffic
  - RW splitting **New in 8.2.0 !**

Bootstrapping

Auto-configuration for the MySQL Architecture

- Fetches the topology Metadata information from one of the servers
- Stores it in a dynamic file (`data/state.json`)
- Registers itself in the Metadata schema
- Creates a configuration file ready to be used
- Creates daemon start/stop scripts

MySQL Router

Manual configuration

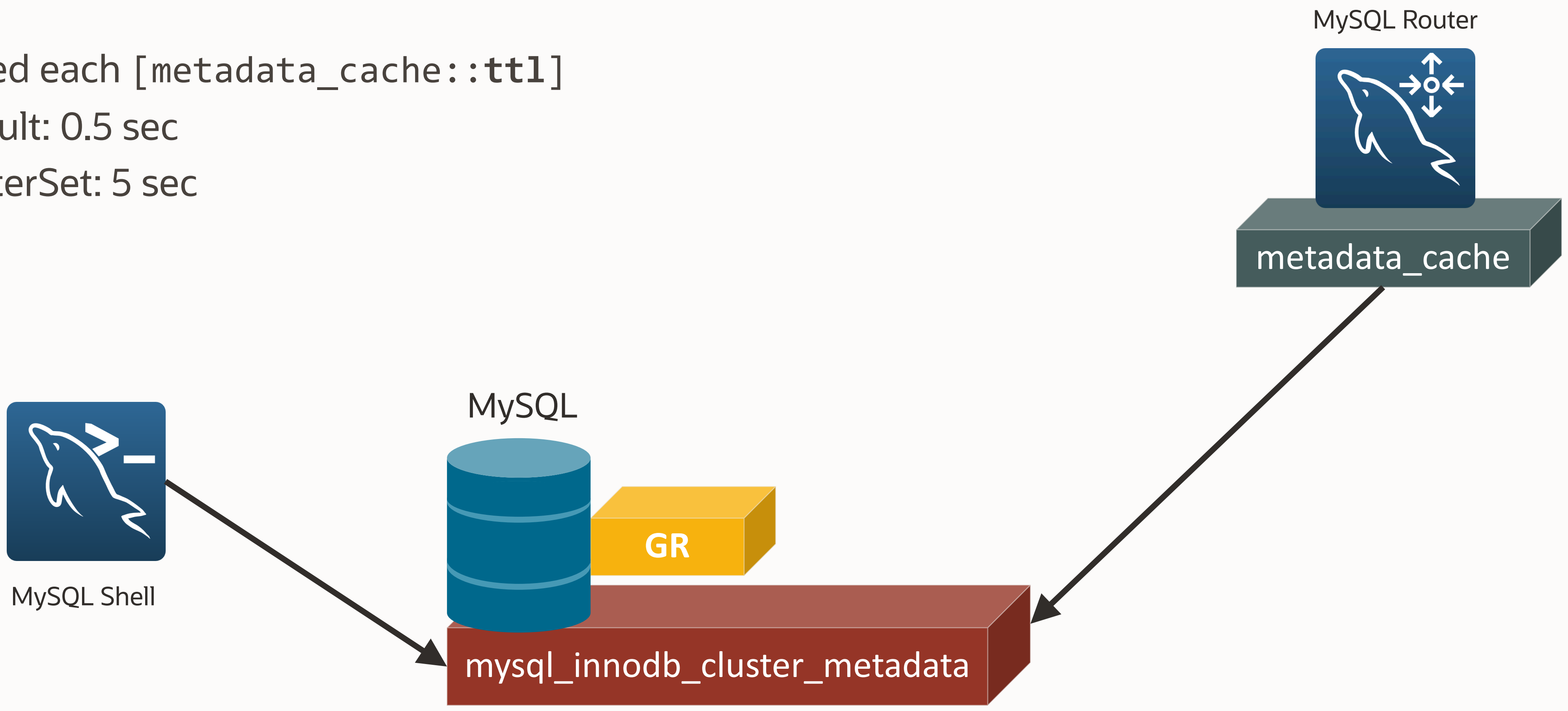
It’s possible to use Router without bootstrapping, however…

~/testbase/router/my.conf

```
[DEFAULT]
...
[routing:primary]
bind_address = localhost
bind_port = 3331
destinations = myserver_xyz:3306
routing_strategy = first-available
[routing:secondaries]
bind_address = localhost
bind_port = 3332
destinations = myserver_foo:3306, myserver_bar:3306
routing_strategy = round-robin-with-fallback
```

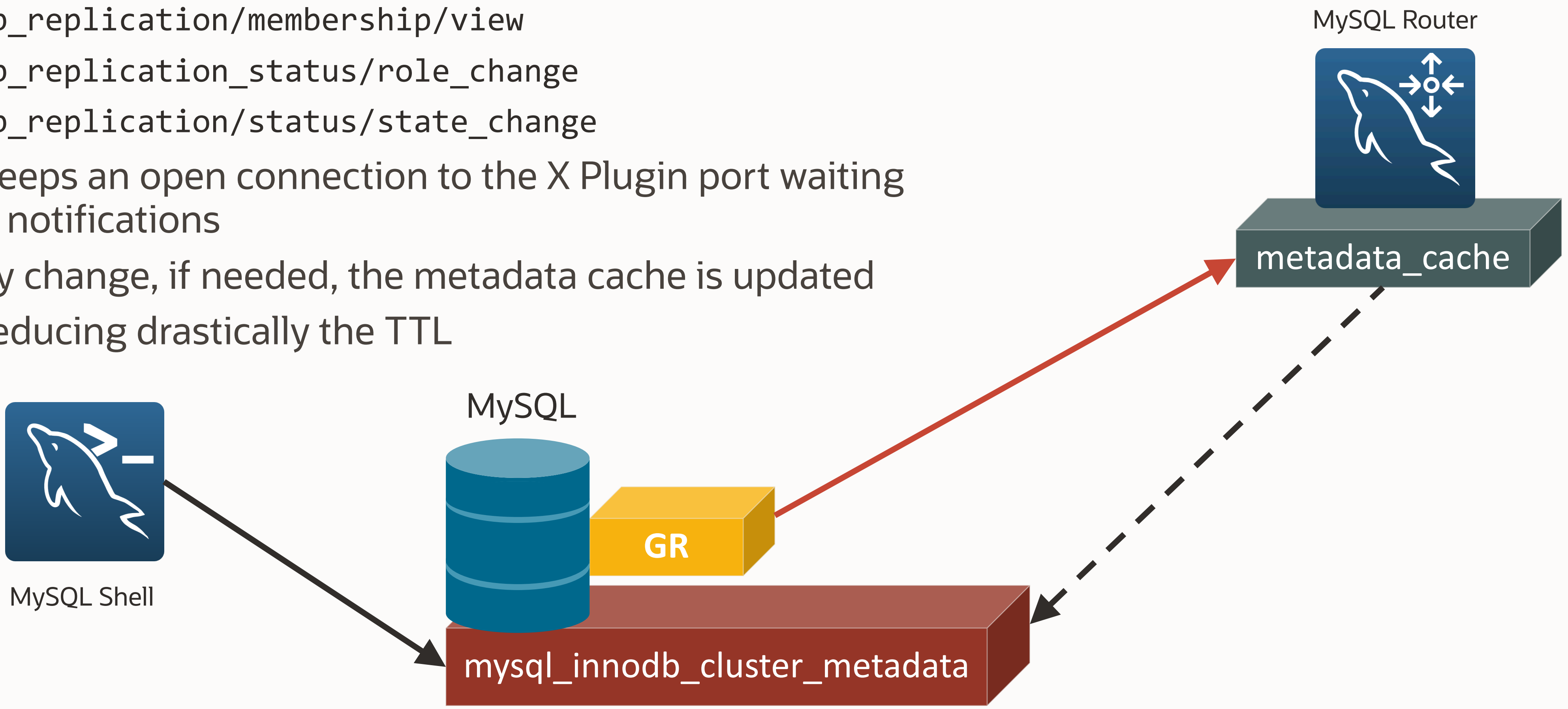
Metadata Cache

- Needed to route queries to the appropriate backend of the topology
- Refreshed each `[metadata_cache::ttl]`
    - Default: 0.5 sec
    - ClusterSet: 5 sec



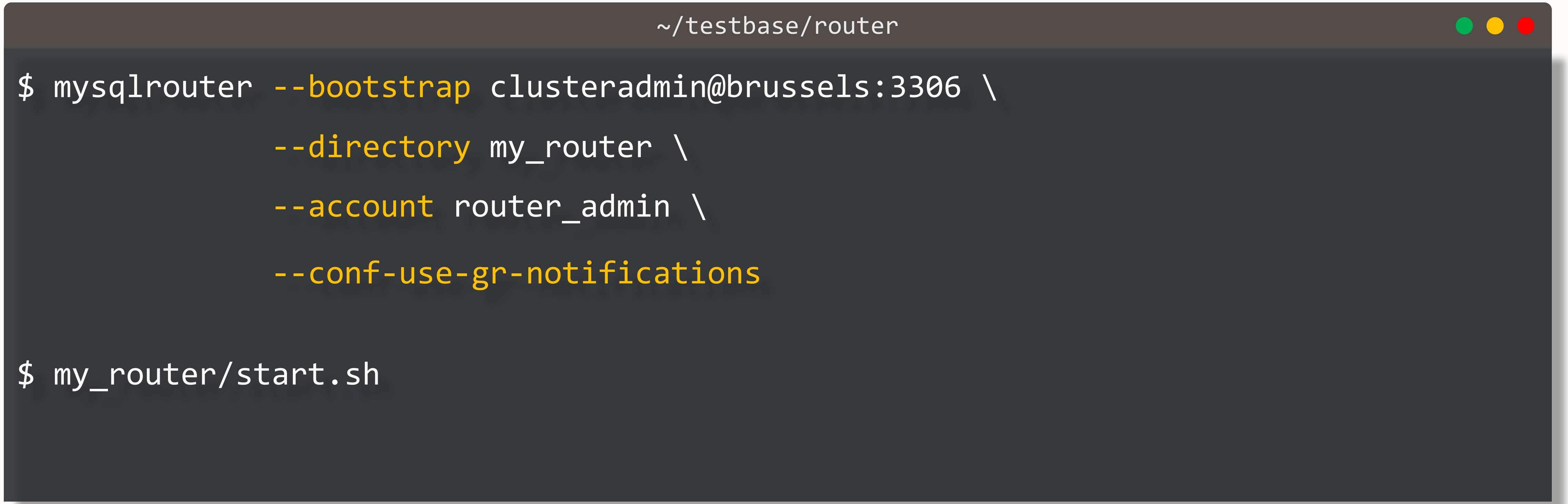
GR Notifications

- Push notifications sent via X protocol:
  - `group_replication/membership/quorum_loss`
  - `group_replication/membership/view`
  - `group_replication_status/role_change`
  - `group_replication/status/state_change`
- Router keeps an open connection to the X Plugin port waiting for push notifications
- For every change, if needed, the metadata cache is updated
- Allows reducing drastically the TTL



Deploying MySQL Router

1. Install MySQL Router
2. Bootstrap
3. Start it!



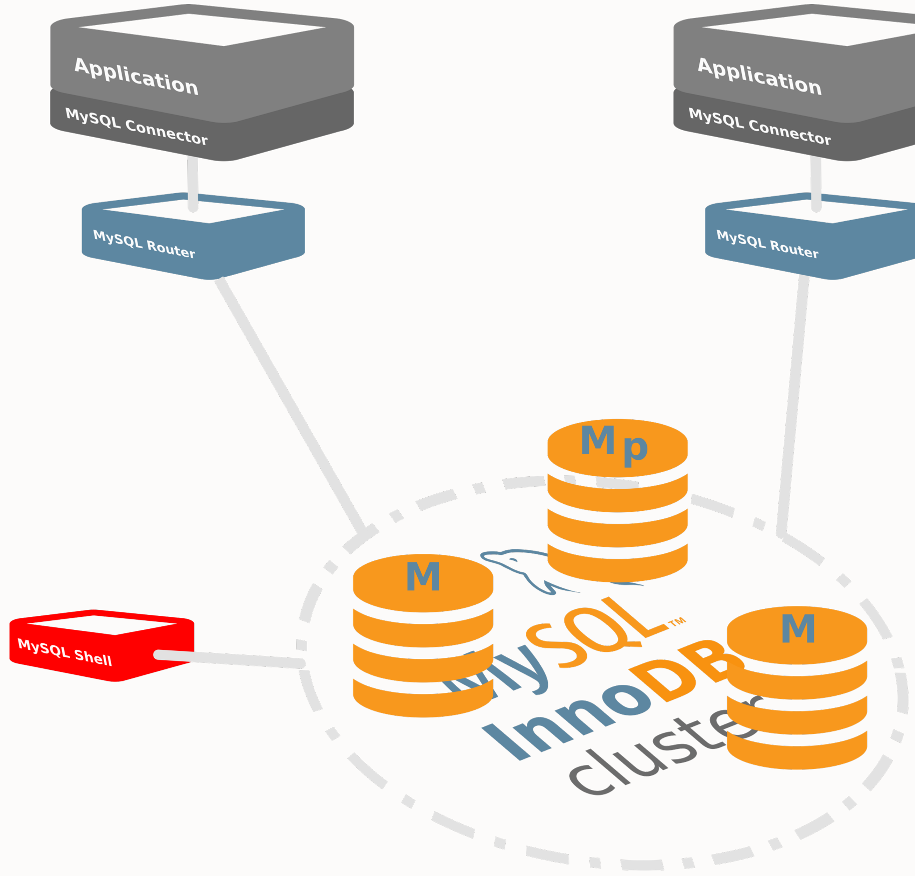
~/testbase/router

```
$ mysqlrouter --bootstrap clusteradmin@brussels:3306 \
              --directory my_router \
              --account router_admin \
              --conf-use-gr-notifications

$ my_router/start.sh
```

Deploying MySQL Router

It's **recommended** to deploy Router on the **same host** as the application and **enable GR notifications**. That allows:

- Using sockets instead of TCP/IP
- Decreasing network latency
- Fine-grained account access
- Scaling-out!

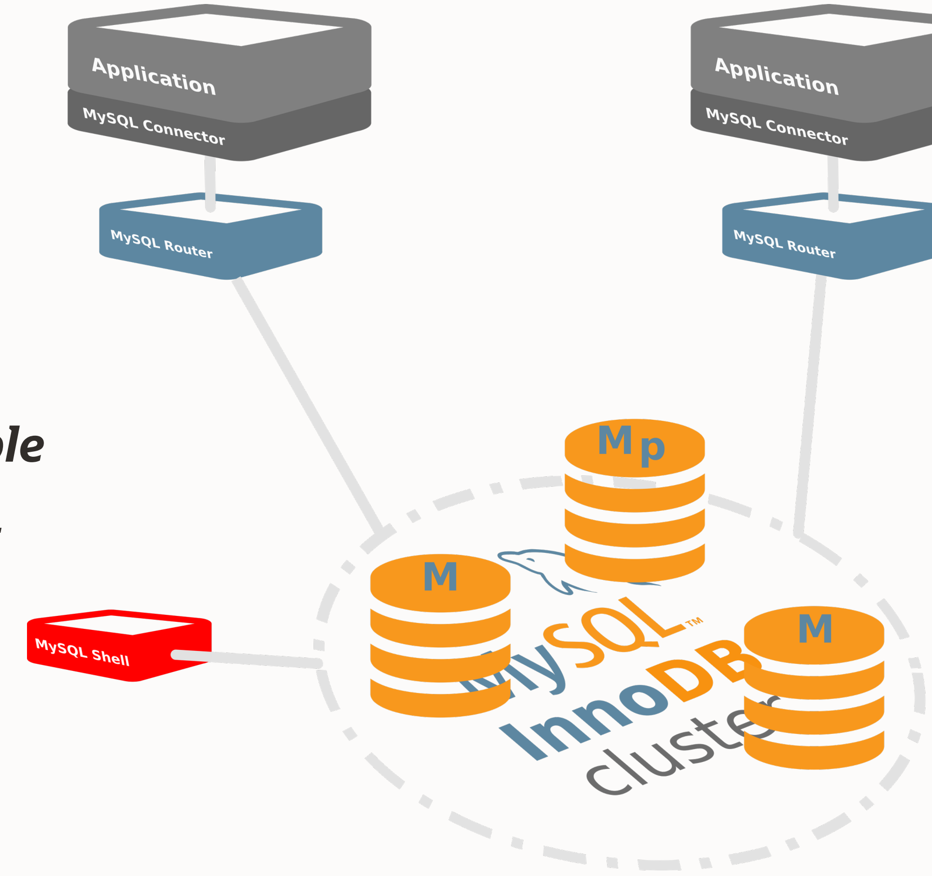


Deploying MySQL Router

It’s **recommended** to deploy Router on the **same host** as the application and **enable GR notifications**. That allows:

- Using sockets instead of TCP/IP
- Decreasing network latency
- Fine-grained account access
- Scaling-out!

*Alternatively, it’s possible to deploy **multiple** Routers in **multiple** machines under a **VIP**.*



Connection Sharing and Reuse

Connection Sharing and Reuse

- Based on a **Connection Pool**
- **Re-use** server-side connections that the client wanted to close, saving the setup costs of establishing new ones
- **Share** server-side connections where the client is idling on an active connection, to reduce the number of open server-side connections freeing up resources bound to those idle connections

MySQL Router

Connection Sharing and Reuse

- Configurable with
  - **`connection_sharing`** (disabled by default)
  - **`connection_sharing_delay`** (1 by default)
    - Seconds to wait before moving an idle connection to the pool
  - **`idle_timeout`** (5 by default)
    - How many seconds to keep a connection in the pool after the client disconnects
  - **`max_idle_server_connections`** (disabled by default)
    - How many open connections can be kept in the pool after the client disconnects

MySQL Router

TLS Session Caching

TLS Session Caching

New in 8.1.0 !

- TLS handshakes are slow

- **Cache and resume TLS sessions** from:
  - Client to Router
  - Router to Server
- **Saves time and resources** by reducing the connection handshake
  - Enabled by default

- **Client** and **Server** side caches, configurable with:
  - `_ssl_session_cache_mode`: enable/disable
  - `_ssl_session_cache_size`: max number of cached sessions
  - `_ssl_session_cache_timeout`: cache timeout

MySQL Router

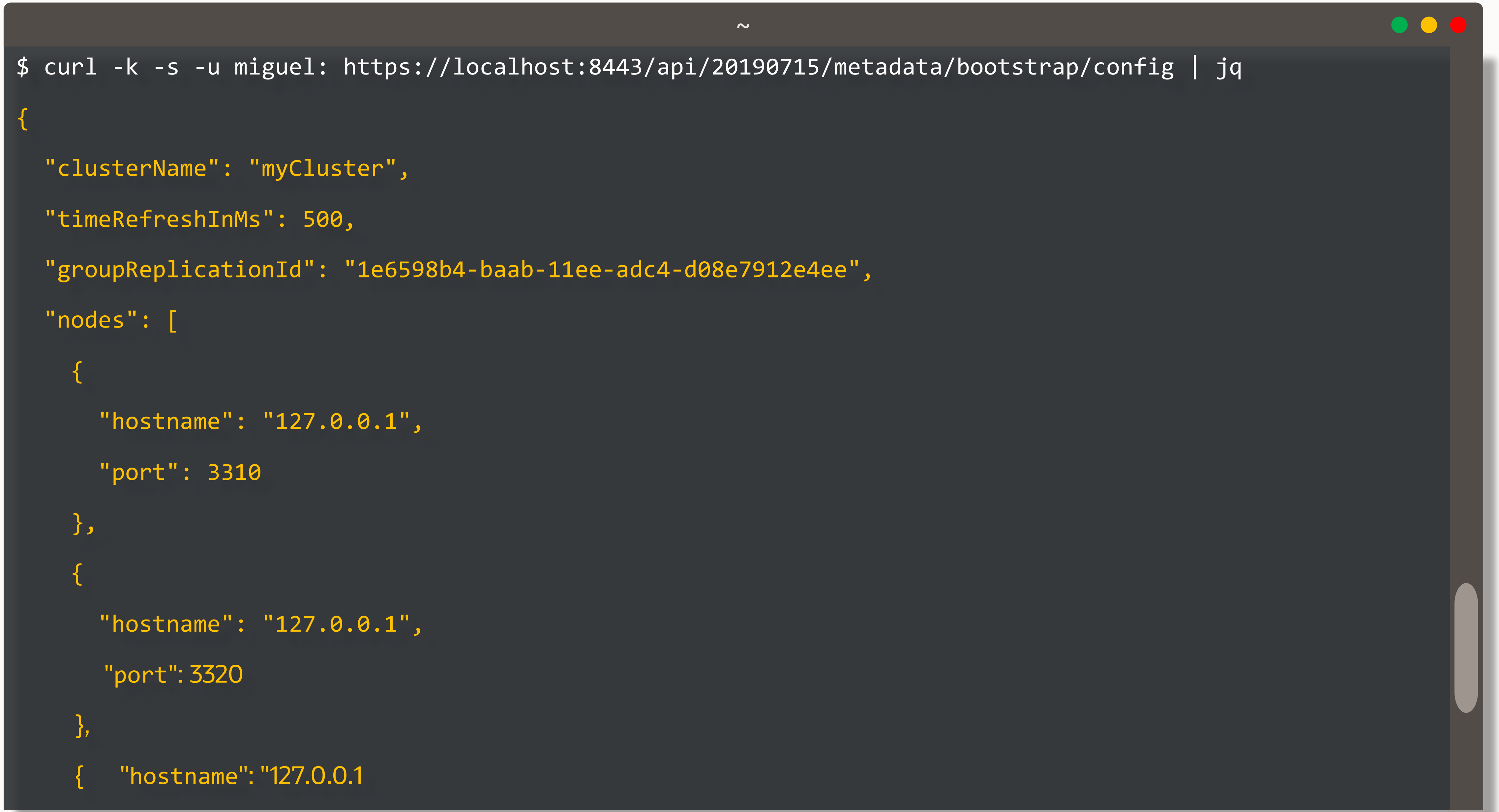
REST API

REST API

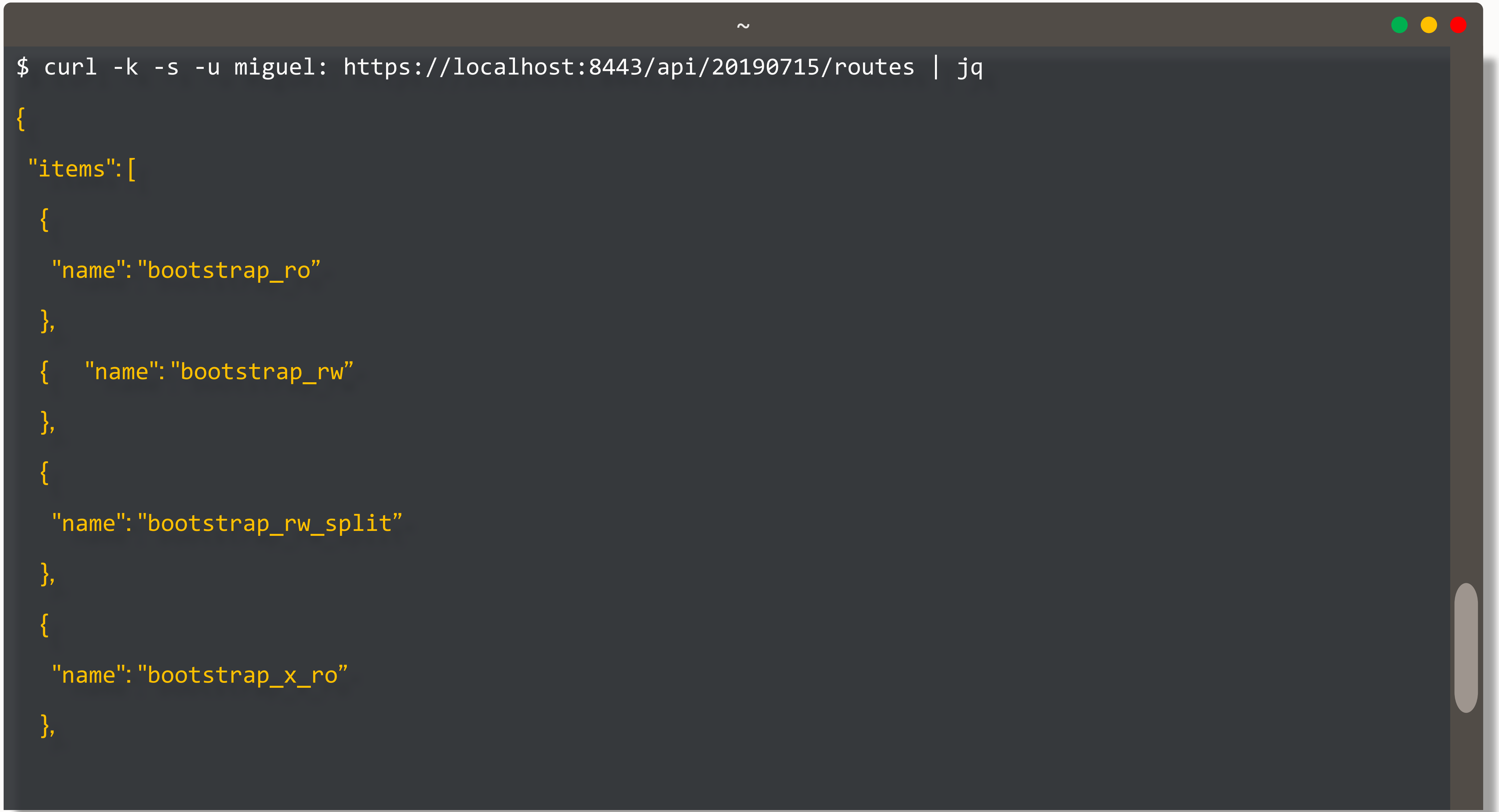
- Built on top of the **HTTP Server plugin**

- Follows the OPENAPI 2.0 spec

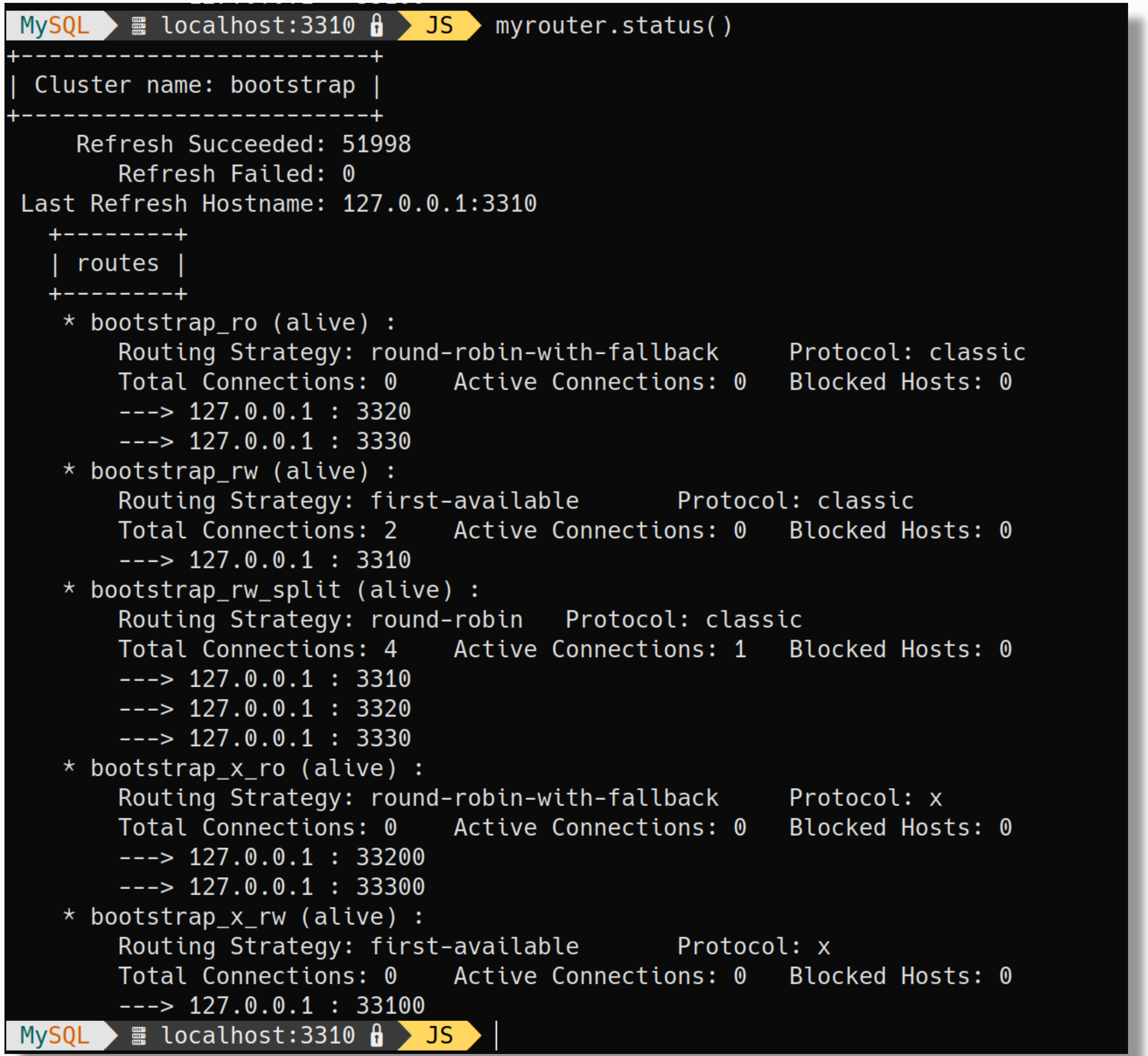
- Exposes a **Swagger** file to describe the REST API:
  - ✓ Metadata cache config
  - ✓ Metadata cache status
  - ✓ Metadata cache instances list
  - ✓ Router status
  - ✓ Routing plugin status
  - ✓ Routes config / status / health / destination / connections
  - ✓ Routes list
  - ✓ Blocked hosts



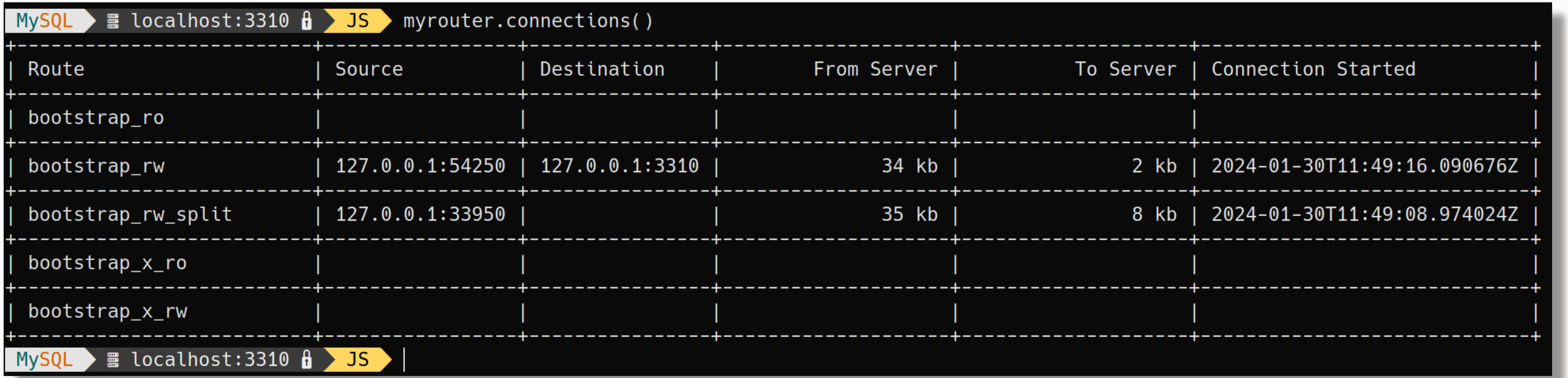
```
$ curl -k -s -u miguel: https://localhost:8443/api/20190715/metadata/bootstrap/config | jq
{
  "clusterName": "myCluster",
  "timeRefreshInMs": 500,
  "groupReplicationId": "1e6598b4-baab-11ee-adc4-d08e7912e4ee",
  "nodes": [
    {
      "hostname": "127.0.0.1",
      "port": 3310
    },
    {
      "hostname": "127.0.0.1",
      "port": 3320
    },
    {   "hostname": "127.0.0.1
```



```
$ curl -k -s -u miguel: https://localhost:8443/api/20190715/routes | jq
{
 "items":[
  {
   "name":"bootstrap_ro”
  },
  {   "name":"bootstrap_rw”
  },
  {
   "name":"bootstrap_rw_split”
  },
  {
   "name":"bootstrap_x_ro”
  },
```



```
MySQL  localhost:3310  JS  myrouter.status()
+-------------------------+
| Cluster name: bootstrap |
+-------------------------+
     Refresh Succeeded: 51998
        Refresh Failed: 0
 Last Refresh Hostname: 127.0.0.1:3310
   +---------+
   | routes |
   +---------+
    * bootstrap_ro (alive) :
      Routing Strategy: round-robin-with-fallback     Protocol: classic
      Total Connections: 0     Active Connections: 0   Blocked Hosts: 0
      ---> 127.0.0.1 : 3320
      ---> 127.0.0.1 : 3330
    * bootstrap_rw (alive) :
      Routing Strategy: first-available       Protocol: classic
      Total Connections: 2     Active Connections: 0   Blocked Hosts: 0
      ---> 127.0.0.1 : 3310
    * bootstrap_rw_split (alive) :
      Routing Strategy: round-robin   Protocol: classic
      Total Connections: 4     Active Connections: 1   Blocked Hosts: 0
      ---> 127.0.0.1 : 3310
      ---> 127.0.0.1 : 3320
      ---> 127.0.0.1 : 3330
    * bootstrap_x_ro (alive) :
      Routing Strategy: round-robin-with-fallback     Protocol: x
      Total Connections: 0     Active Connections: 0   Blocked Hosts: 0
      ---> 127.0.0.1 : 33200
      ---> 127.0.0.1 : 33300
    * bootstrap_x_rw (alive) :
      Routing Strategy: first-available       Protocol: x
      Total Connections: 0     Active Connections: 0   Blocked Hosts: 0
      ---> 127.0.0.1 : 33100
MySQL  localhost:3310  JS  |
```



```
MySQL  localhost:3310  JS  myrouter.connections()
+------------------------+-----------------+----------------+-----------------+-----------------+-----------------------------+
| Route                  | Source          | Destination    |     From Server |       To Server | Connection Started          |
+------------------------+-----------------+----------------+-----------------+-----------------+-----------------------------+
| bootstrap_ro           |                 |                |                 |                 |                             |
+------------------------+-----------------+----------------+-----------------+-----------------+-----------------------------+
| bootstrap_rw           | 127.0.0.1:54250 | 127.0.0.1:3310 |           34 kb |            2 kb | 2024-01-30T11:49:16.090676Z |
+------------------------+-----------------+----------------+-----------------+-----------------+-----------------------------+
| bootstrap_rw_split     | 127.0.0.1:33950 |                |           35 kb |            8 kb | 2024-01-30T11:49:08.974024Z |
+------------------------+-----------------+----------------+-----------------+-----------------+-----------------------------+
| bootstrap_x_ro         |                 |                |                 |                 |                             |
+------------------------+-----------------+----------------+-----------------+-----------------+-----------------------------+
| bootstrap_x_rw         |                 |                |                 |                 |                             |
+------------------------+-----------------+----------------+-----------------+-----------------+-----------------------------+
MySQL  localhost:3310  JS
```

MySQL Rest Service (MRS)

Key takeaways

- Fast and powerful way to serve data to client applications via a **HTTPS REST interface**
- Implemented as a **MySQL Router feature**
- Built on the concepts of ORDS, focusing on **the strengths of MySQL**
  - Not as powerful as ORDS (PL/SQL based)
  - Focus on MySQL performance
  - Focus on MySQL scalability
  - Using MySQL/HeatWave as metadata storage, not depending on an OracleDB instance
- **Auto REST** for tables, views, and procedures
- **GUI Frontend** with MySQL Shell for VSCode

MySQL REST Service

Key takeaways

RESTful Web Services

- Auto REST for tables, views, procedures and functions
- {JSON} responses with paged results
- Developer support (GUI, CLI, API)
- Support for popular OAuth2 services

Full SQL Support & SDK API

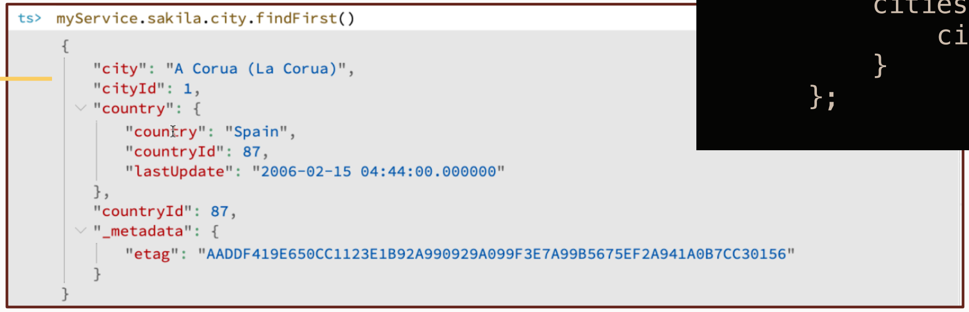
- Fully manageable through SQL
- CREATE REST DUALITY VIEW statements
- Tailored SDK for all RESTful Endpoints
- Popular, Prisma-like API, live prototyping

JSON/Relational Duality

- Full support SQL support for JSON/Relational REST endpoints
- Visual Duality Editor - Build complex JSON structures with a few clicks
- SQL & SDK interface preview

```
sql> CONFIGURE REST METADATA;

sql> CREATE REST SERVICE /myService;

sql> CREATE REST SCHEMA /sakila FROM `sakila`;
```

```
sql> CREATE OR REPLACE REST DUALITY VIEW /country
     ON SERVICE /myService SCHEMA /sakila
     AS sakila.country {
         countryId: country_id @SORTABLE,
         country: country,
         lastUpdate: last_update,
         cities: sakila.city @INSERT @UPDATE @DELETE {
             city: city
         }
     };
```

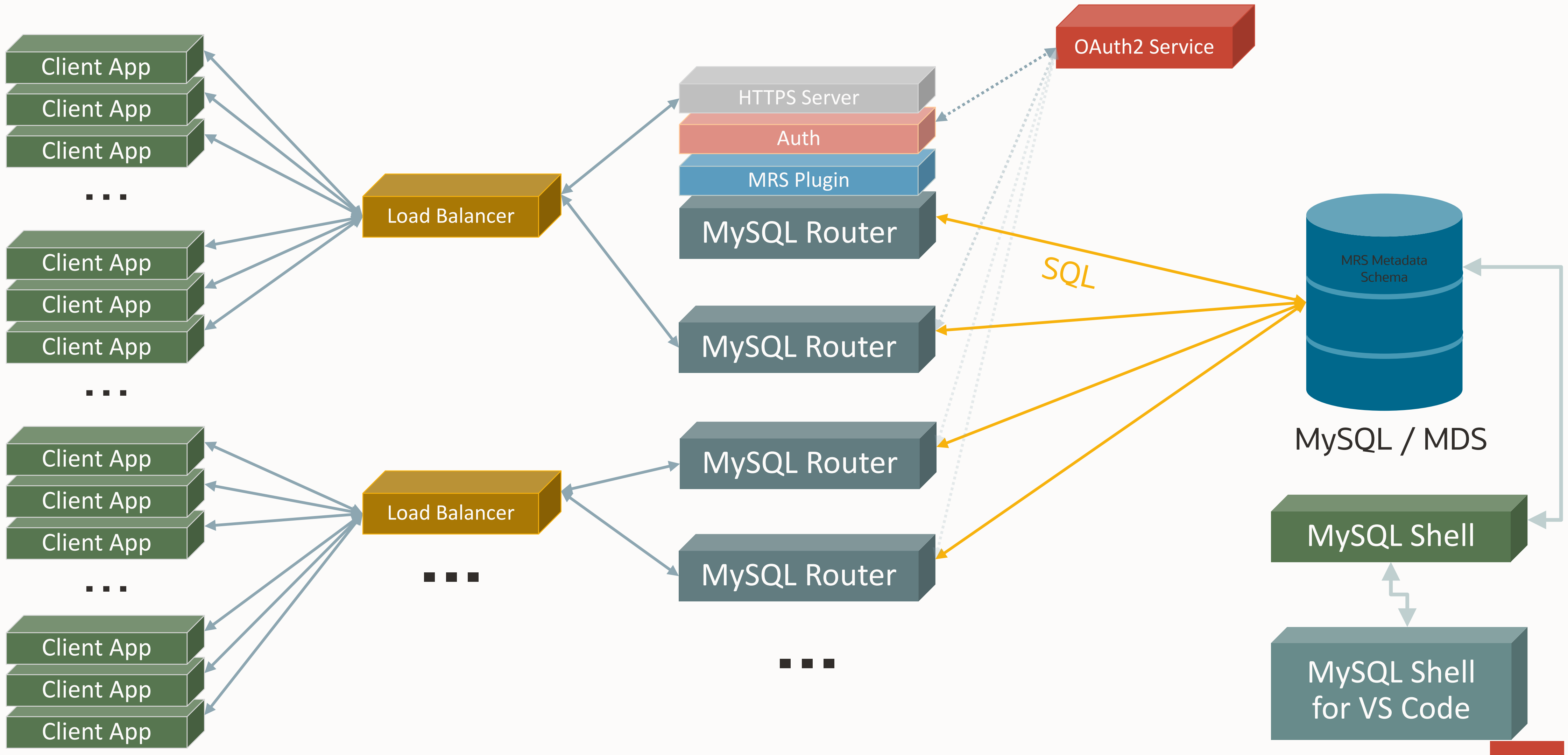


```
ts> myService.sakila.city.findFirst()
{
    "city": "A Corua (La Corua)",
    "cityId": 1,
    "country": {
        "country": "Spain",
        "countryId": 87,
        "lastUpdate": "2006-02-15 04:44:00.000000"
    },
    "countryId": 87,
    "_metadata": {
        "etag": "AADDF419E650CC1123E1B92A990929A099F3E7A99B5675EF2A941A0B7CC30156"
    }
}
```

TypeScript SDK API with live prototyping of REST queries

Full JSON/Relational Duality Support via SQL and GUI

MySQL REST Service
Architecture



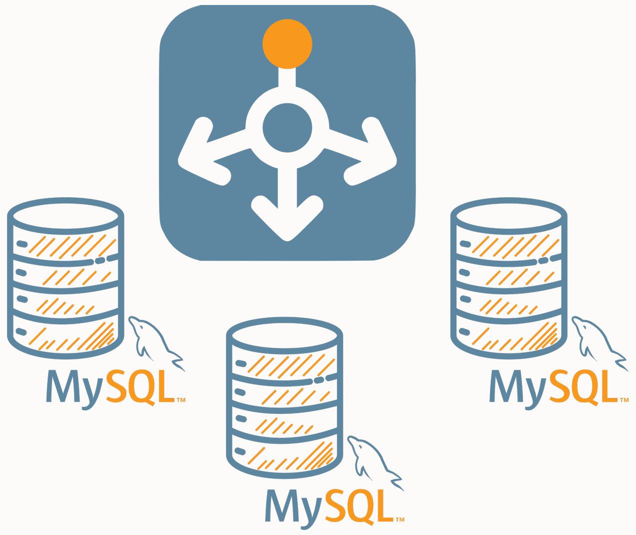
R/W Splitting

R/W Splitting

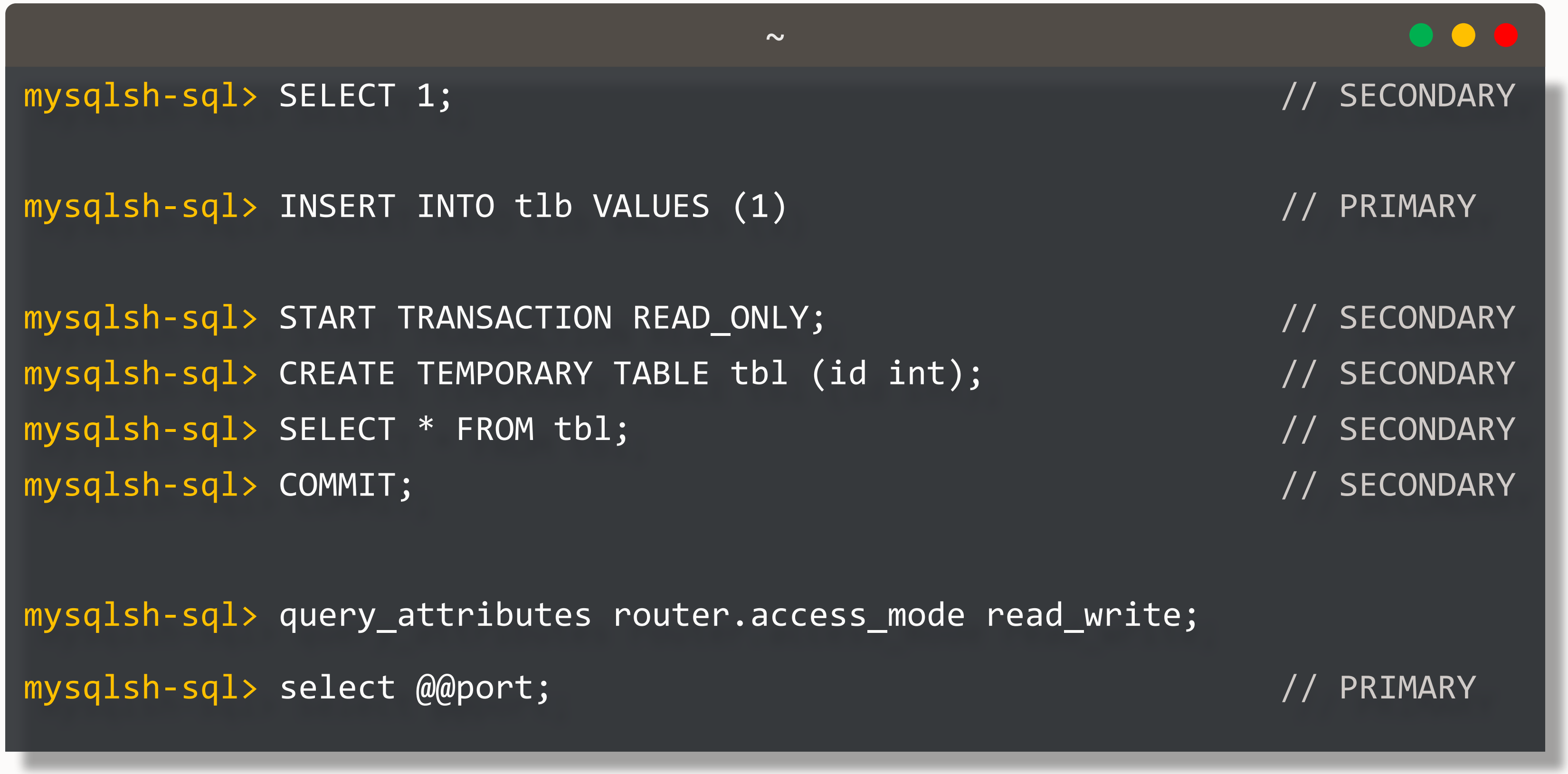
- **Motivation:**
  - **1 port to rule them all**
  - Up to now, the application had to be aware of the type of transaction to use either the RW or the RO port
  - It's performant, but limits the usage of Router

- **Work it works:**
  - Router classifies each query as **read** or **write automatically** and forwards to the appropriate backend
  - It's also possible to manually or programmatically to specify the type of query using
    - ROUTER SET
    - query_attributes

New in 8.2.0!

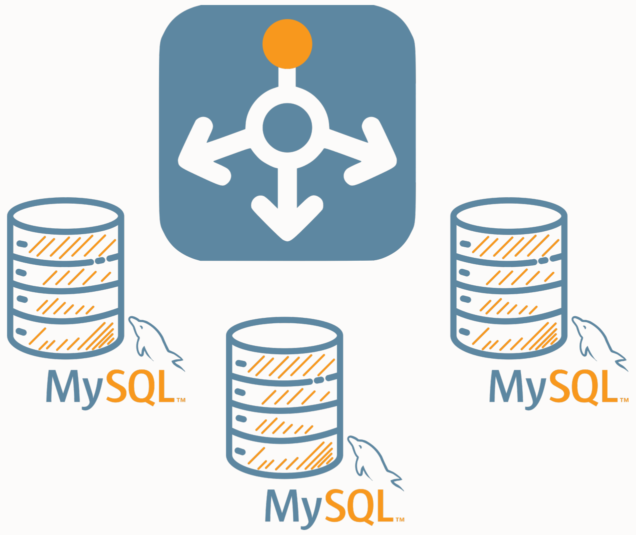


R/W Splitting

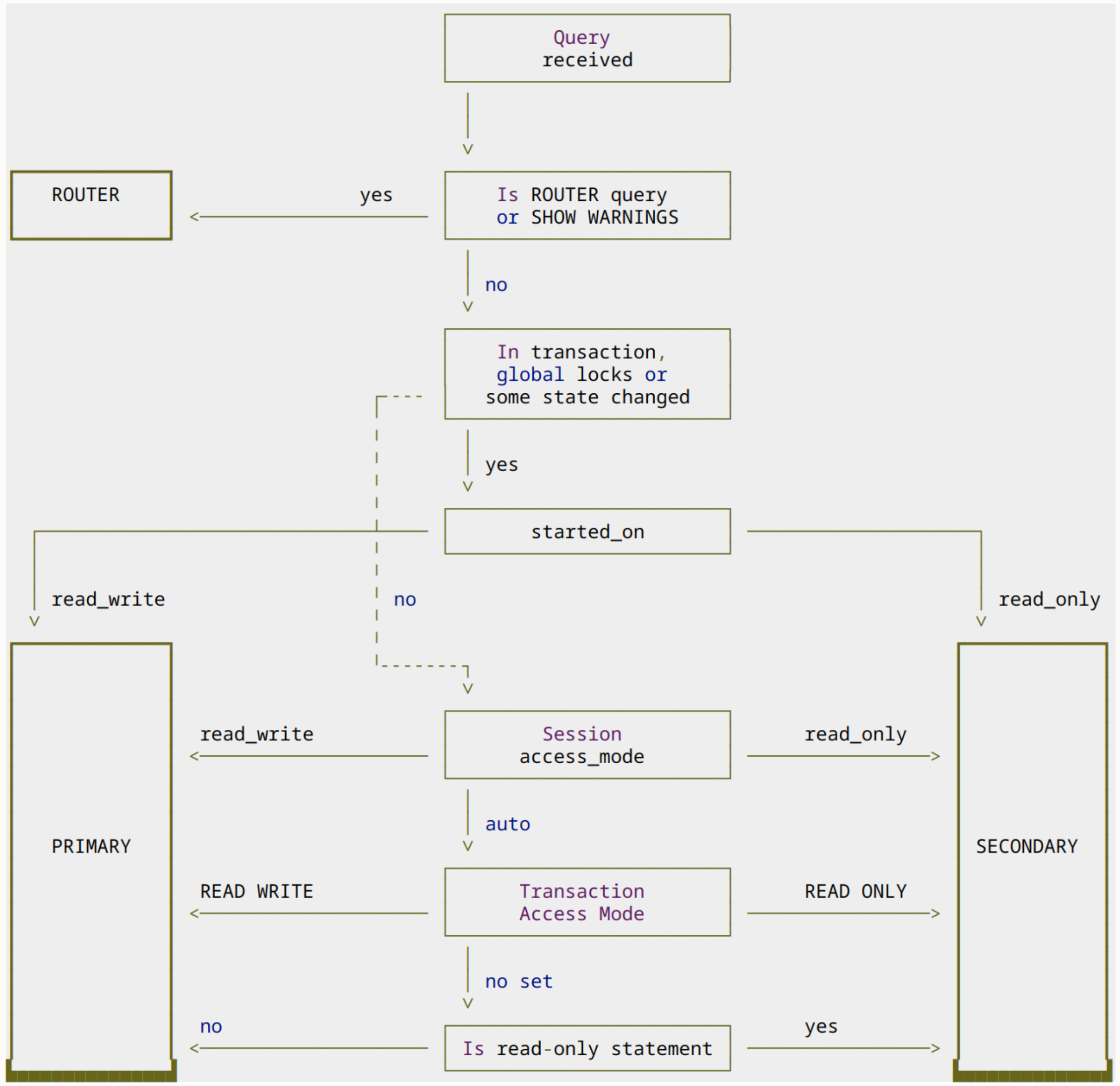


```
mysqlsh-sql> SELECT 1;                                            // SECONDARY

mysqlsh-sql> INSERT INTO tlb VALUES (1)                           // PRIMARY

mysqlsh-sql> START TRANSACTION READ_ONLY;                         // SECONDARY
mysqlsh-sql> CREATE TEMPORARY TABLE tbl (id int);                 // SECONDARY
mysqlsh-sql> SELECT * FROM tbl;                                   // SECONDARY
mysqlsh-sql> COMMIT;                                              // SECONDARY


mysqlsh-sql> query_attributes router.access_mode read_write;

mysqlsh-sql> select @@port;                                       // PRIMARY
```

New in 8.2.0!



R/W Splitting



- The query is sent to the **`PRIMARY`** if the **`access_mode`** is **`read_write`**
- The query is sent to the **`SECONDARY`** if the **`access_mode`** is **`read_only`**
- If the **`access_mode`** is **`auto`**, a query is sent to a **`SECONDARY`** if:
  - Inside a **`READ_ONLY`** transaction, or
  - Router attribute for access mode set (**`router.access_mode`**), or
  - Outside a transaction, the connection allows sharing and the statement is a “read-only” statement
- If none of the above is met, the query is sent to the **`PRIMARY`**

Thank you!

Questions?

MySQL™ Belgian Days 2024

February 1 & 2

ICAB Incubator

4 Rue des Pères Blancs 1040 Bruxelles

ORACLE