Routing Guidelines

Unlocking Smarter Query Routing in MySQL Architectures

Miguel Araújo
Senior Principal Software Engineer
MySQL, Oracle
February 02, 2025

FOSDEM'25

```
$ whoami
miguel_araujo

$ curl -s ipinfo.io/country
PT

$ grep 'activity' ~/community_log.txt
Active Community User: MySQL Forums, Blogs, Slack, and Conferences

$ history -E
... <= 2009                     Casual MySQL user & other jobs
[2009, 2011]                    Implemented a MySQL-Proxy plugin to use a GCS (Spread) to
                                enable active-active replication
[2011, 2014]                    Joined Oracle MySQL. MEM & MySQL-Proxy developer
[2014, 2017]                    MySQL Shell developer
[2017, now()]                   AdminAPI Tech Lead + MySQL Database Architectures
```

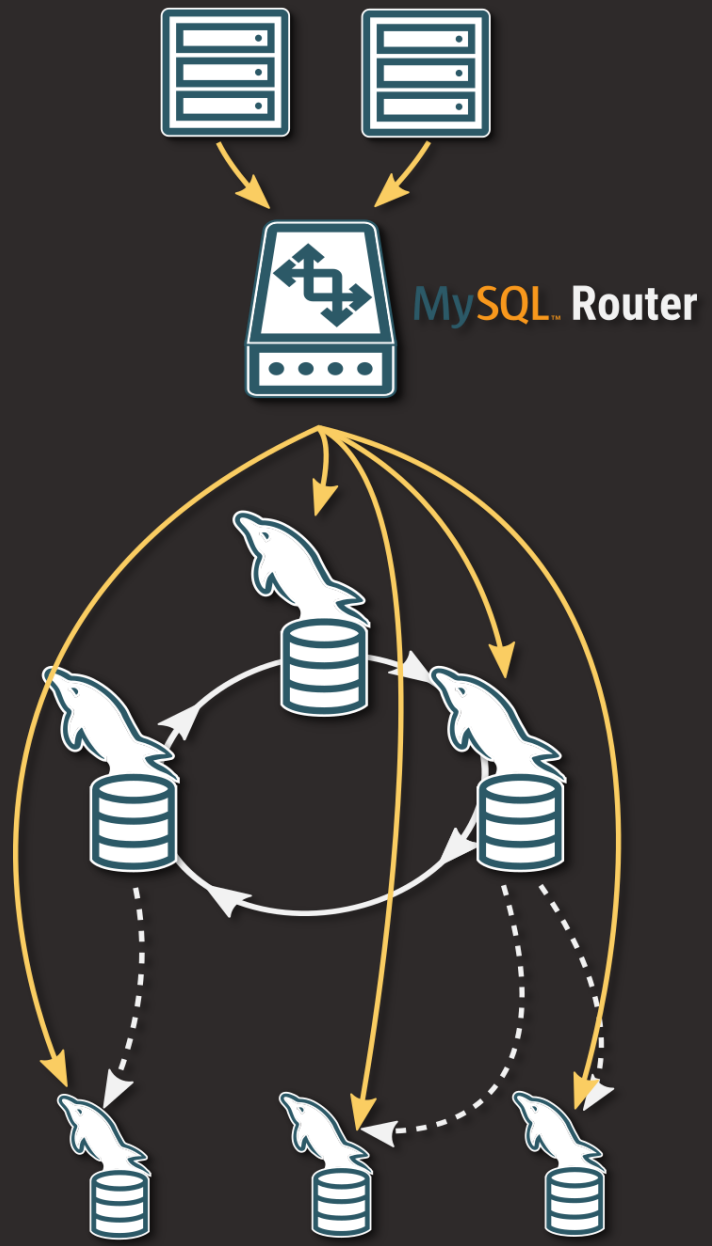
Setting the Stage

The Case for Smarter Query Routing

Current Routing Mechanism

The MySQL Router serves as a lightweight middleware that efficiently routes client requests to the appropriate MySQL Servers.

- Full integration into MySQL Architectures
- Transparent client connection routing
  - Load Balancing
  - Application connection failover
  - Automatically does what is expected from it with almost no setup (most of the time)



A Shared Experience

Where Current Routing Falls Short

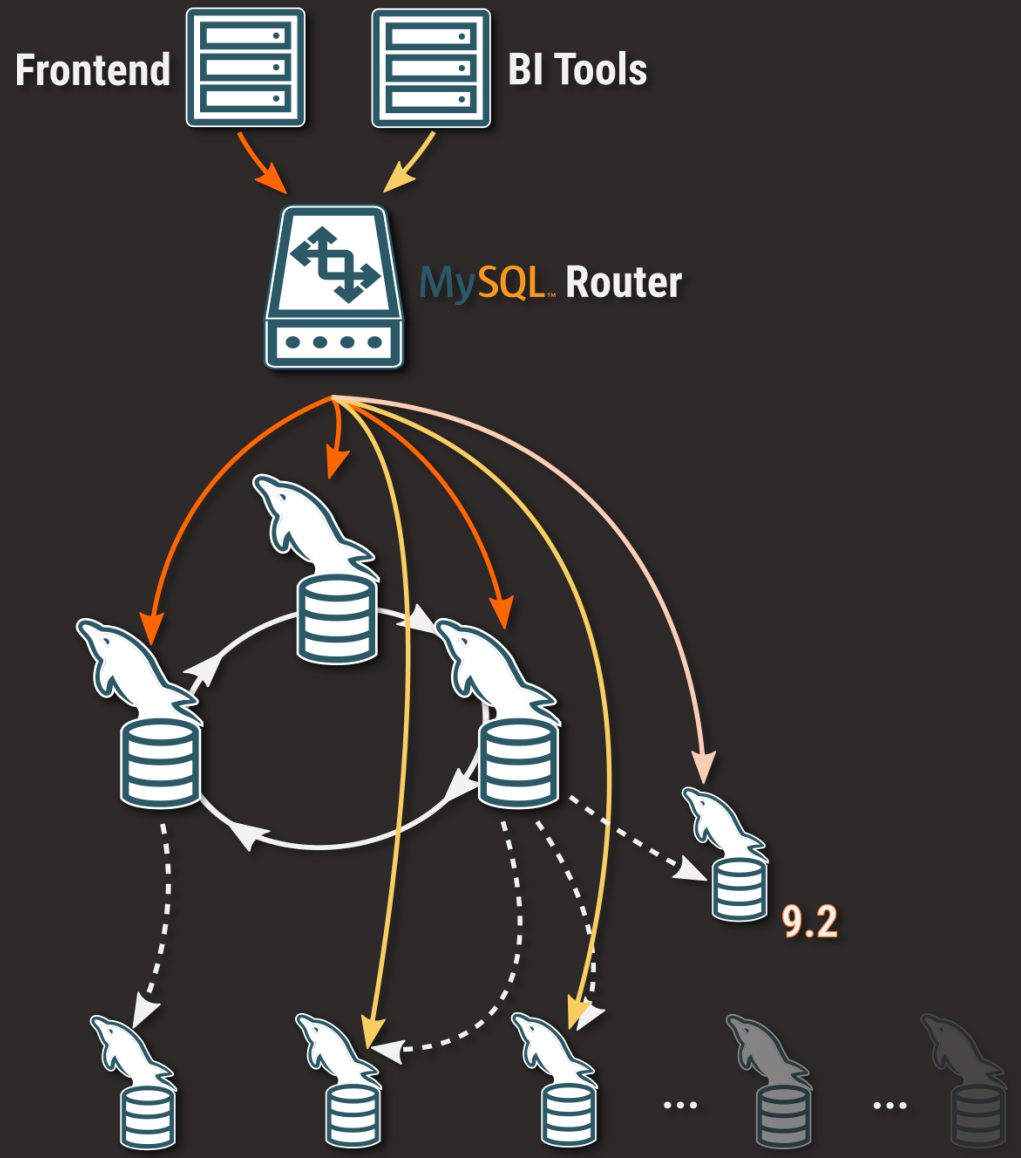
A friend running an e-commerce struggled and asked me for help...

- BI traffic is intensive
- Frontend traffic must avoid stale data

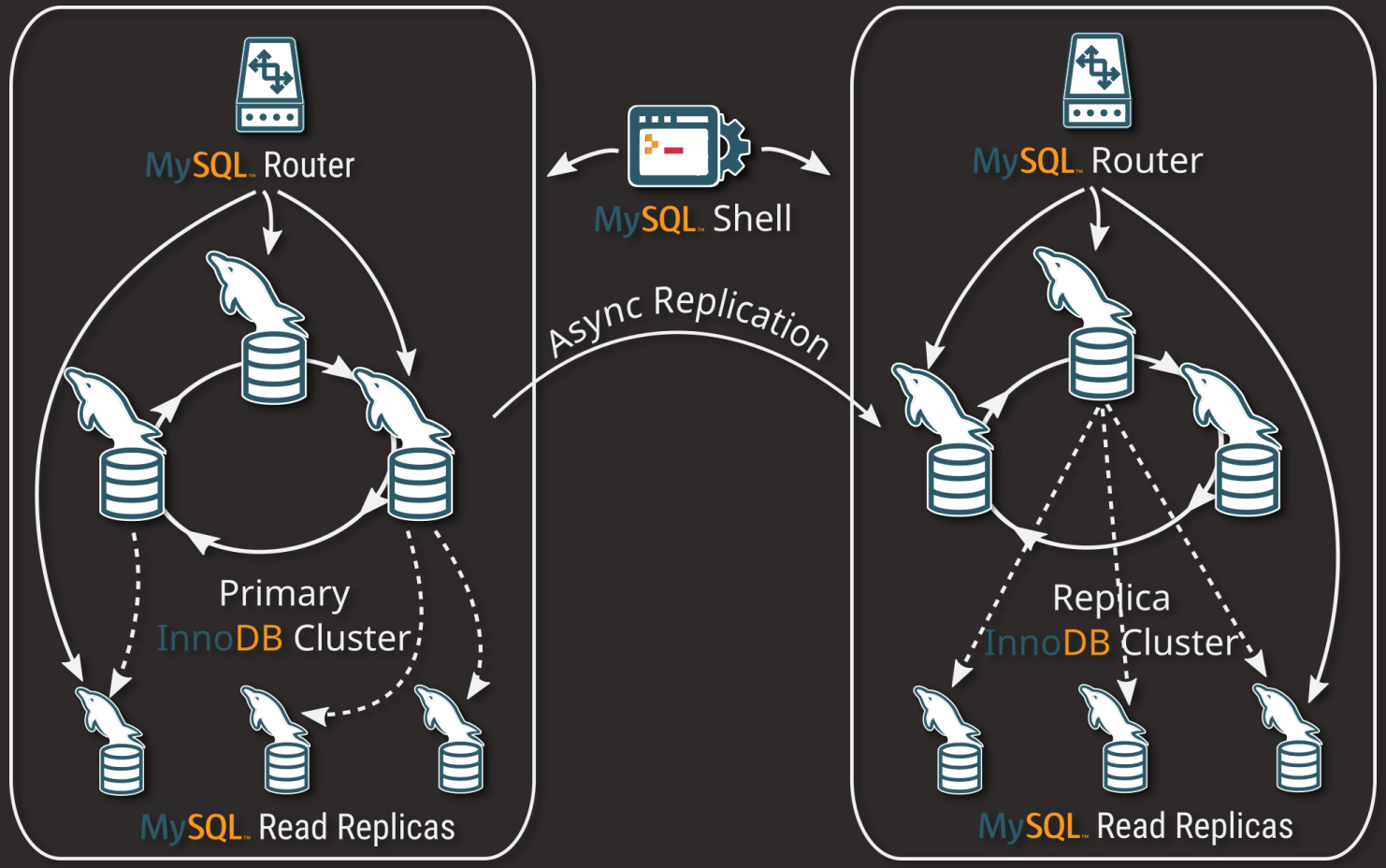
- Do manual Routing configuration? REJECTED
- Use "read_only_targets" and multiple Routers? REJECTED

"Write me a patch to support this, please?"

REJECTED



As MySQL evolves with setups like ClusterSet, traditional routing faces significant challenges, requiring a smarter approach.



Challenges With Evolving Topologies

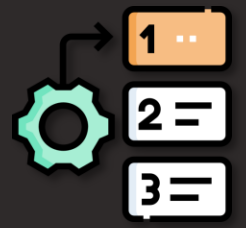
Geographically distributed instances (InnoDB ClusterSet)

- Current challenge: Routing doesn't consider latency differences across datacenters
- Need: Adaptable routing to optimize performance based on proximity

Challenges With Evolving Topologies

Resource prioritization

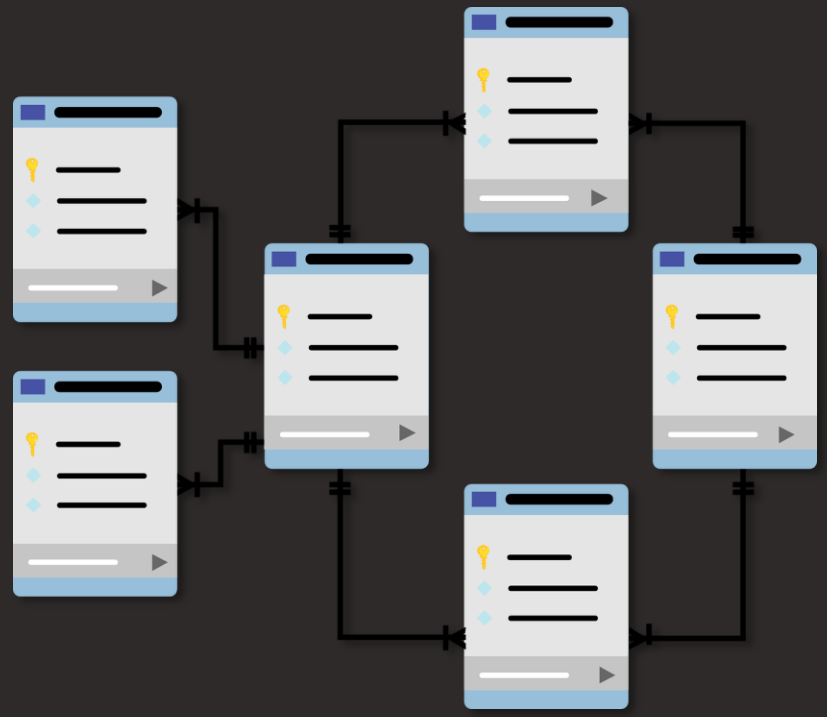
- Current challenge: Equal treatment of all sessions may cause inefficiencies during overload or partial outage
- Need: Prioritize critical operations during high load



Challenges With Evolving Topologies

Schema-specific routing (vertical partitioning)

- Current challenge: Manual connection management based on schema location
- Need: Automatic routing to the right Cluster based on schema or other criteria



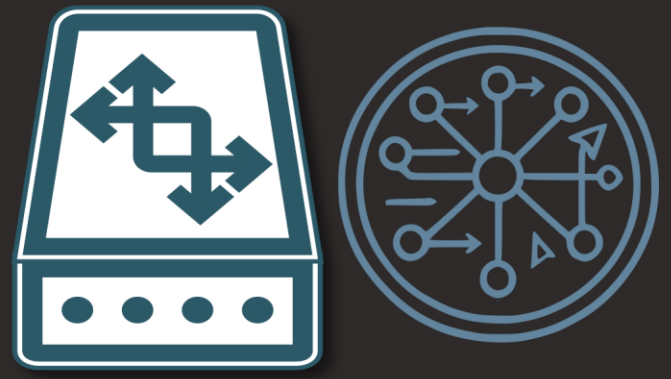
Complex Routing Factors

As the complexity of MySQL setups grows, routing must adapt:

- Support application-specific configurations
- Adapt to dynamic changes in topologies such as failovers, overloads, or maintenance
- Enable granular control, even down to individual statements within a session
- Custom routing behavior for very narrow and specific use cases

MySQL Routing Guidelines

Core Concepts And Technical Foundations



Routing Guideline Syntax

- Destinations
  - Primary
  - Secondary
- Routes
  - ro
- Name
  - FOSDEM25
- Version
  - 1.0

```
{
    "destinations": [
        {
            "match": "$.server.memberRole = PRIMARY",
            "name": "Primary"
        },
        {
            "match": "$.server.memberRole = SECONDARY",
            "name": "Secondary"
        }
    ],
    "name": "FOSDEM25",
    "routes": [
        {
            "connectionSharingAllowed": true,
            "destinations": [
                {
                    "classes": [
                        "Secondary"
                    ],
                    "priority": 0,
                    "strategy": "round-robin"
                },
                {
                    "classes": [
                        "Primary"
                    ],
                    "priority": 1,
                    "strategy": "round-robin"
                }
            ],
            "enabled": true,
            "match": "$.session.targetPort = $.router.port.ro",
            "name": "ro"
        }
    ],
    "version": "1.0"
}
```

Breaking It Down

Destinations

- Group MySQL instances in the topology using pattern-matching expressions
- The expressions define which servers are included in a destination class
- Each class forms a pool of candidate instances for routing

```
{
    "destinations": [
        {
            "match": "$.server.memberRole = PRIMARY",
            "name": "Primary"
        },
        {
            "match": "$.server.memberRole = SECONDARY",
            "name": "Secondary"
        },
        {
            "match": "$.server.memberRole = READ_REPLICA",
            "name": "ReadReplica"
        }
    ]
...
```

Breaking It Down

Routes

- Match incoming client sessions to appropriate destination candidates, using expressions
- The expressions define how client sessions are classified and directed to MySQL Servers
- Candidate destinations are organized into tiers, with each tier containing one or more destination classes

```
"routes": [
    {
        "connectionSharingAllowed": true,
        "destinations": [
            {
                "classes": [
                    "Secondary"
                ],
                "priority": 0,
                "strategy": "round-robin"
            },
            {
                "classes": [
                    "Primary"
                ],
                "priority": 1,
                "strategy": "round-robin"
            }
        ],
        "enabled": true,
        "match": "$.session.targetPort = $.router.port.ro",
        "name": "ro"
    },
```

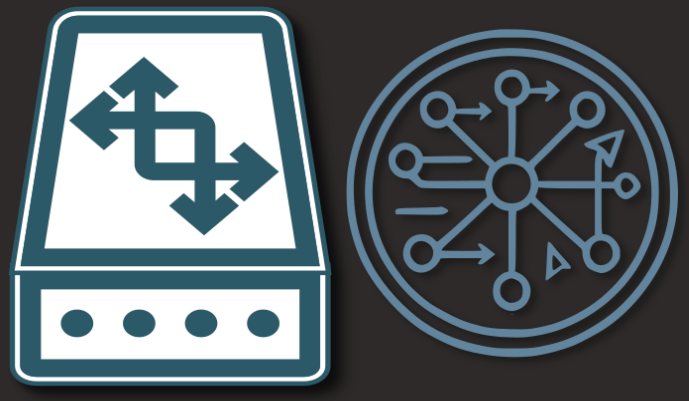
Matching Rules

Predefined variables

- `$.server.*`
  - Related to the MySQL Server
- `$.session.*`
  - Related to the Client session
- `$.router.*`
  - Related to the Router instance

Functions / Operators

- Logical operators
  - `AND | OR | NOT`
- Inclusion checks
  - `IN | NOT IN`
- `LIKE` operator
  - `Pattern matching | _ | %`
- Arithmetic operations
  - `+ | - | * | % | /`
- Comparisons
  - `> | >= | < | <= | = | <>`
- Functions
  - `SQRT() | CONCAT() | IS_IPV6() | etc.`



MySQL™

Routing Guidelines

Matching Expressions

- **Classify**: Identify servers, sessions, or routers with logical conditions
- **Compose**: Use variables, operators, and values to create matches
- **Chain**: Link conditions with AND, OR, or NOT for flexibility

```
(<function> | <variable>) [<operator> <value>] [<logical-operator> <expression>]*
```

Predefined Variables

`$.server.*`

VARIABLE	TYPE	EXAMPLE
`$.server.label`	String	"myserver"
`$.server.address`	String	"myhost123-portugal"
`$.server.port`	Integer	3306
`$.server.uuid`	String	"bae9454b-d8f3-11ef-8e0c-d08e7912e4ee"
`$.server.version`	Integer (MMmmpp)	90200
`$.server.memberRole`	Enum	SECONDARY
`$.server.tags`	String (key-value)	.performance = 'high'
`$.server.clusterName`	String	"FOSDEM25"
`$.server.clusterRole`	Enum	REPLICA
`$.server.clusterSetName`	String	"MyClusterSet"
`$.server.isClusterInvalidated`	Boolean	False

Predefined Variables

`$.session.*`

VARIABLE	TYPE	EXAMPLE
`$.session.targetIp`	String	"192.168.1.235"
`$.session.targetPort`	Integer	6446
`$.session.sourceIp`	String	"10.1.103.12"
`$.session.user`	String	"admin"
`$.session.connectAttrs`	String (key-value)	._os = 'Linux'
`$.session.schema`	String	"world"
`$.session.randomValue`	Double [0, 1]	0.3

Predefined Variables

`$.router.*`

VARIABLE	TYPE	EXAMPLE
`$.router.port.rw`	Integer	6446
`$.router.port.ro`	Integer	6447
`$.router.port.rw_split`	Integer	6448
`$.router.localCluster`	String	"FOSDEM25"
`$.router.hostname`	String	"domus"
`$.router.bindAddress`	String	"127.0.0.1"
`$.router.tags`	String (key-value)	"tag:router_foo"
`$.router.routeName`	String	"bootstrap_ro"
`$.router.name`	String	"myrouter123"

Functions

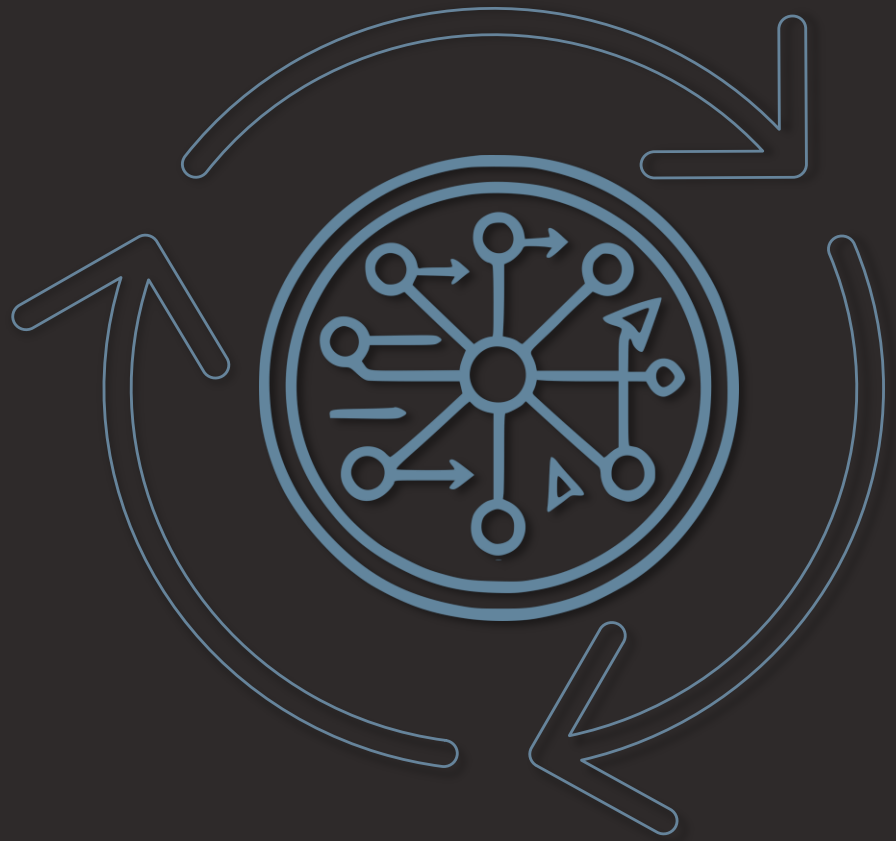
VARIABLE	EXAMPLE	
`CONCAT('str', 'str', ...)`	`CONCAT('a', 'b', 'cde')`	abcde
`SQRT('number')`	`SQRT(0.16)`	0.4
`NUMBER('str')`	`NUMBER(CONCAT('1', '2'))`	12
`NETWORK('str', 'int')`	`NETWORK('192.168.1.33', 24)`	192.168.1.0/24
`IS_IPV6('str')`	`IS_IPV6('192.168.1.33)`	FALSE
`IS_IPV4('str')`	`IS_IPV4('192.168.1.33)`	TRUE
`STARTSWITH('str1', 'str2')`	`STARTSWITH('foo', 'foobar')`	TRUE
`ENDSWITH('str1', 'str2')`	`CONCAT('a', 'b', 'cde')`	abcde
`CONTAINS('str1', 'str2')`	`CONTAINS('foobar', 'foo')`	TRUE
`RESOLVE_V4('str')`	`RESOLVE_V4('domus')`	127.0.0.1
`RESOLVE_V6('str'):`	`RESOLVE_V6('domus')`	2a02:26f0:d8:108e::a15
`REGEXP_LIKE('str1', 'str2')`	`REGEXP_LIKE('foobarbaz', 'foo.*baz')`	TRUE
`SUBSTRING_INDEX('str1', 'str2', 'int')`	`SUBSTRING_INDEX("test", "s", 1)`	te

Workflow

1. Classify Destinations

Groups Servers into destination classes

Server 1 : n destination class



2. Match Route Rules

Classifies incoming client request to routes

Route 1 : n candidate destination class

3. Apply Routing Strategy

first-available

round-robin

4. Monitor Topology

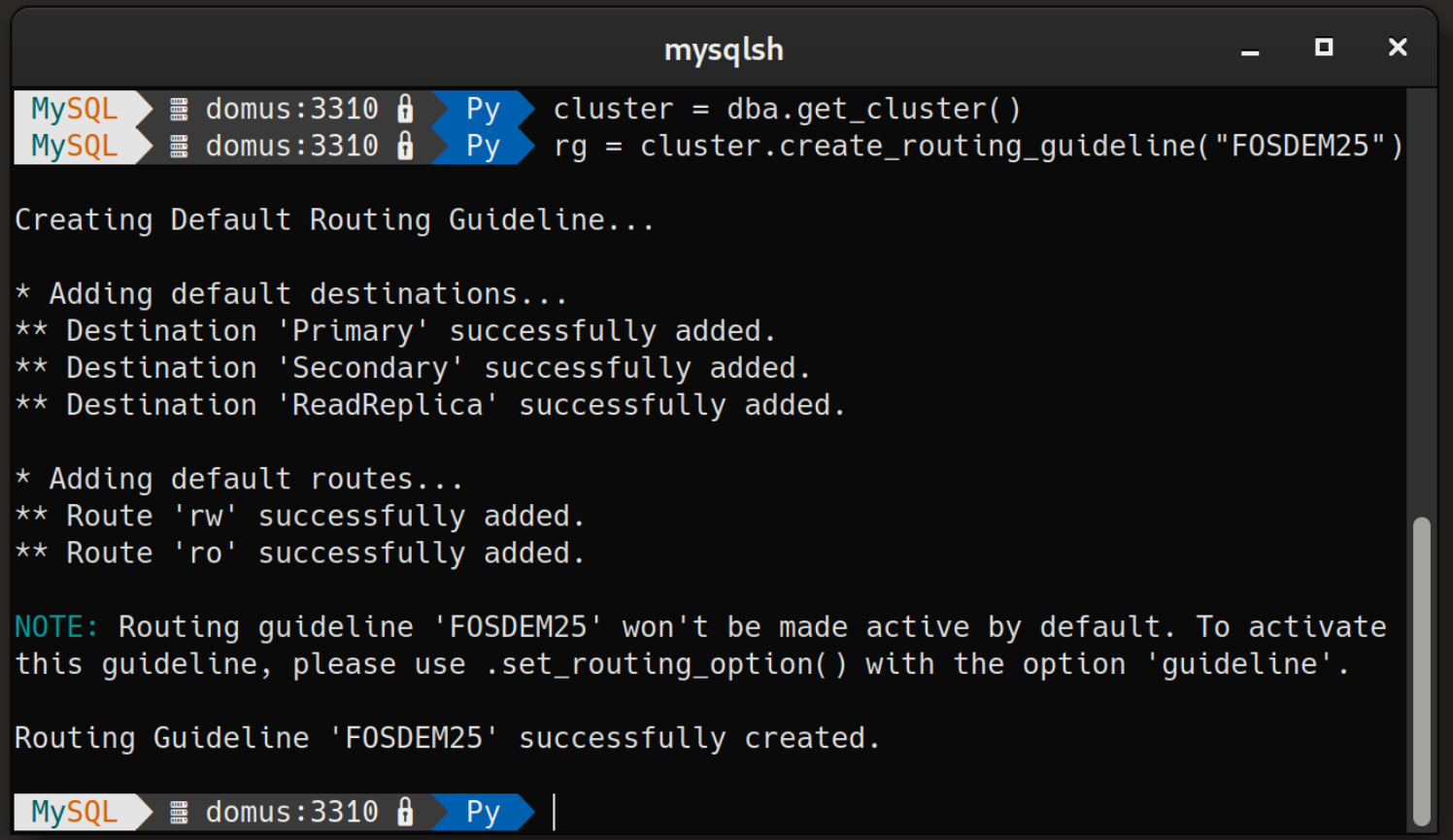
Reclassify servers

Update Routes

Disconnect invalid connections

AdminAPI

- Extended to support Routing Guidelines
- Define, manage, visualize
- New `<RoutingGuideline>` class
- Seamless support in all MySQL Architectures:
  - InnoDB Cluster
  - InnoDB ReplicaSet
  - InnoDB ClusterSet



AdminAPI

<Cluster>, <ReplicaSet>, <ClusterSet>

COMMAND	PURPOSE
`.create_routing_guideline(name[, json[, options]])`	Obvious
`.set_routing_option("guideline", name)`	New option "guideline" to activate a guideline in the target topology.
`.get_routing_guideline([name])`	Evident
`.remove_routing_guideline(name)`	Clear
`.routing_guidelines()`	Lists all Routing Guidelines of the topology with some info
`.import(filePath)`	Imports a Routing Guideline stored in a .json file into the topology

AdminAPI

<RoutingGuideline>

COMMAND	PURPOSE
`.show([options])`	Displays a comprehensive summary of the Routing Guideline
`.as_json()`	Unambiguous
`.destinations()`	Explicit
`.routes()`	Unmistakable
`.add_route(name, match, destinations [, options])`	Clear
`.add_destination(name, match, [, options])`	Self-evident
`.remove_route(name)`	Lucid
`.remove_destination(name)`	Straightforward
`.set_destination_option(destinationName, option, value)`	Decipherable
`.set_route_option(routeName, option, value)`	Perceptible
`.copy(name)`	Visible
`.export(filePath)`	Exports the target Routing Guideline to a .json file
`.rename(name)`	Logical

Impact And Use Cases

Unlocking The Potential Of Routing Guidelines

High Availability And Disaster Recovery

Goals

1. Seamless Failover:
   - ❑ Redirect traffic to alternate nodes during outages for uninterrupted service
2. Local-First:
   - ❑ Prefer local nodes for routing, using remote nodes as fallback options
3. Optimized Read-Write Routing:
   - ❑ Routes write traffic to primary and distributes read traffic across secondaries and read-replicas for scale-out
4. Fallback levels:
   - ❑ Implement fallback tiers to ensure maximum availability

High Availability And Disaster Recovery

mysqlsh

```
MySQL  domus:3310  Py  rg.destinations()
+----------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| destination                | match                                                                                                                                                                  |
+----------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Primary_Local              | $.server.clusterRole = PRIMARY AND $.server.memberRole = PRIMARY AND $.server.isClusterInvalidated = false AND $.server.clusterName = $.router.localCluster            |
| Primary_Remote             | $.server.clusterRole = PRIMARY AND $.server.memberRole = PRIMARY AND $.server.isClusterInvalidated = false AND $.server.clusterName <> $.router.localCluster           |
| Secondary_Local            | $.server.memberRole = SECONDARY AND $.server.isClusterInvalidated = false AND $.server.clusterName = $.router.localCluster                                             |
| Secondary_Remote           | $.server.memberRole = SECONDARY AND $.server.isClusterInvalidated = false AND $.server.clusterName <> $.router.localCluster                                            |
| Scale_Out_Local            | $.server.memberRole = READ_REPLICA AND $.server.isClusterInvalidated = false AND $.server.clusterName = $.router.localCluster                                          |
| Scale_Out_Remote           | $.server.memberRole = READ_REPLICA AND $.server.isClusterInvalidated = false AND $.server.clusterName <> $.router.localCluster                                         |
| Read_Only_Fallback_Local   | $.server.isClusterInvalidated = true AND ($.server.memberRole = SECONDARY OR $.server.memberRole = READ_REPLICA) AND $.server.clusterName = $.router.localCluster      |
| Read_Only_Fallback_Remote  | $.server.isClusterInvalidated = true AND ($.server.memberRole = SECONDARY OR $.server.memberRole = READ_REPLICA) AND $.server.clusterName <> $.router.localCluster     |
+----------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
8 rows in set (0.0000 sec)
MySQL  domus:3310  Py  |
```

mysqlsh

```
MySQL  domus:3310  Py  rg.add_route(
                         "rw_traffic",
                         "$.session.targetPort in ($.router.port.rw, $.router.port.rw_split)", ["first-available(Primary_Local, Primary_Remote)"], {"connectionSharingAllowed": True, "enabled": True});

Route 'rw_traffic' successfully added.
MySQL  domus:3310  Py  rg.add_route(
                         "ro_traffic",
                         "$.session.targetPort = $.router.port.ro",
                         [
                           "round-robin(Secondary_Local, Scale_Out_Local)",
                           "round-robin(Secondary_Remote, Scale_Out_Remote)",
                           "round-robin(Primary_Local, Primary_Remote)",
                           "round-robin(Read_Only_Fallback_Local, Read_Only_Fallback_Remote)"
                         ],
                         {"connectionSharingAllowed": True, "enabled": True});

Route 'ro_traffic' successfully added.
MySQL  domus:3310  Py  |
```

Geolocation-Based Routing And Compliance

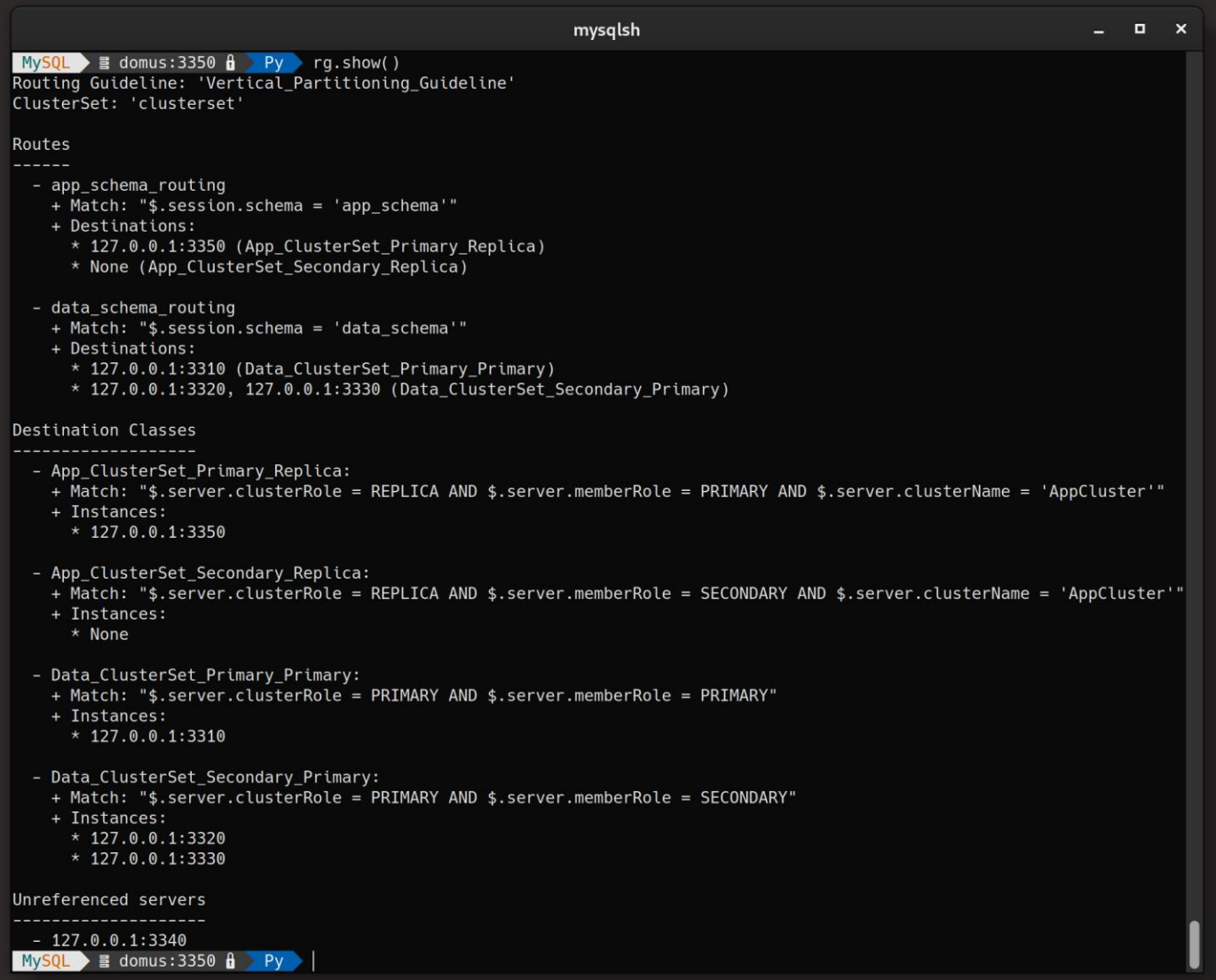
```
{
    "destinations": [
        {
            "match": "$.server.address IN ('us-east-
                    1.example.com', 'us-west-2.example.com')",
            "name": "US_Regions"
        },
        {
            "match": "$.server.address IN ('eu-central-
                    1.example.com', 'eu-west-1.example.com')",
            "name": "EU_Regions"
        },
        {
            "match": "$.server.tags.compliance = 'GDPR'",
            "name": "GDPR_Compliant"
        }
    ],
    "name": "Geo_Based_Guideline",
```

- Routes traffic based on IP to specific regional destinations
- Directs traffic requiring a specific compliance to servers tagged with it

```
    "routes": [
        {
            "connectionSharingAllowed": true,
            "destinations": [
                {
                    "classes": [
                        "US_Regions",
                        "EU_Regions"
                    ],
                    "strategy": "round-robin",
                    "priority": 0
                }
            ],
            "enabled": true,
            "match": "NETWORK($.session.sourceIP, 24) =
                      NETWORK('192.168.1.0', 24) OR
                      NETWORK($.session.sourceIP, 8) =
                      NETWORK('10.0.0.0', 8)",
            "name": "geo_based"
        },
        {
            "connectionSharingAllowed": true,
            "destinations": [
                {
                    "classes": [
                        "GDPR_Compliant"
                    ],
                    "strategy": "round-robin",
                    "priority": 0
                }
            ],
            "enabled": true,
            "match": "$.session.connectAttrs.region = 'EU'",
            "name": "compliance_based"
        }
    ],
    "version": "1.0"
}
```

Schema-Based Routing

- Application schema Traffic: Sessions using 'app_schema' are directed to AppCluster
- Data schema traffic: Sessions using 'data_schema' are directed to the main data cluster
- $.session.schema



Testing, Staging, And Session Affinity

mysqlsh

```
MySQL  domus:3350  Py  rg.destinations()
+--------------------+-----------------------------------------+
| destination        | match                                   |
+--------------------+-----------------------------------------+
| Production_Servers | $.server.tags.environment = "production" |
| Staging_Servers    | $.server.tags.environment = "staging"    |
| Testing_Servers    | $.server.tags.environment = "testing"    |
+--------------------+-----------------------------------------+
3 rows in set (0.0000 sec)
MySQL  domus:3350  Py  rg.routes()
+--------------------+---------+-----------+-----------------------------------------------------------------+-----------------------------------------------------------------------------+-------+
| name               | enabled | shareable | match                                                           | destinations                                                                | order |
+--------------------+---------+-----------+-----------------------------------------------------------------+-----------------------------------------------------------------------------+-------+
| testing_traffic    |       1 |         1 | $.session.randomValue < 0.1                                     | first-available(Testing_Servers)                                            |     0 |
| staging_traffic    |       1 |         1 | $.session.randomValue >= 0.1 AND $.session.randomValue < 0.3    | first-available(Staging_Servers)                                            |     1 |
| production_traffic |       1 |         1 | $.session.randomValue >= 0.3                                    | first-available(Production_Servers)                                         |     2 |
| session_affinity   |       1 |         0 | $.session.user = 'persistent_user'                              | round-robin(Production_Servers, Staging_Servers, Testing_Servers)           |     3 |
+--------------------+---------+-----------+-----------------------------------------------------------------+-----------------------------------------------------------------------------+-------+
4 rows in set (0.0000 sec)
MySQL  domus:3350  Py
```

- Testing traffic: Routes ~10% of requests to testing servers for isolated testing
- Staging traffic: Routes ~20% of requests to staging servers for validation
- Production traffic: Routes remaining requests to production servers for stability
- Session affinity: Ensures '`persistent_user`' sessions maintain continuity

Client Characteristics Routing

- Backup Traffic: Routes sessions coming from `'mysqldump'` to the Backup Servers
- Linux traffic: Routes sessions coming from Linux clients to Servers running on Linux
- Connection attributes
  - ._os = 'Linux'
  - .program_name = 'mysqldump'
- Metadata tags
  - $.server.tags

```
{
    "destinations": [
        {
            "match": "$.server.tags.backup = 'true'",
            "name": "Backup_Servers"
        },
        {
            "match": "$.server.tags.os = 'Linux'",
            "name": "Linux_Servers"
        },
        ...
    ],
    ...
    "routes": [
        {
            "connectionSharingAllowed": true,
            "destinations": [
                {
                    "classes": [
                        "Linux_Servers"
                    ],
                    "strategy": "round-robin",
                    "priority": 0
                }
            ],
            "enabled": true,
            "match": "$.session.connectAttrs._os = 'Linux'",
            "name": "linux_traffic"
        },
        ...
        {
            "connectionSharingAllowed": true,
            "destinations": [
                {
                    "classes": [
                        "Backup_Servers"
                    ],
                    "strategy": "round-robin",
                    "priority": 0
                }
            ],
            "enabled": true,
            "match": "$.session.connectAttrs.program_name =
                      'mysqldump'",
            "name": "backup_traffic"
        },
        ...
```

From A Friend's Challenge To Smarter Routing

- Primary and Secondaries:
  - `$.server.memberRole`
- Split Read-Replicas: By network ranges and member-role
  - `NETWORK()`
  - `$.server.memberRole`
- Testing Servers: Match by MySQL Server version
  - `$.server.version`

```
{
  "destinations": [
    {
      "name": "bi_read_replicas",
      "match": "NETWORK($.server.address, 24) = NETWORK('10.10.10.0', 24) AND
                $.server.memberRole = 'READ_REPLICA'"
    },
    {
      "name": "frontend_read_replicas",
      "match": "NETWORK($.server.address, 24) = NETWORK('10.10.20.0', 24) AND
                $.server.memberRole = 'READ_REPLICA'"
    },
    {
      "name": "secondaries",
      "match": "$.server.memberRole = 'SECONDARY'"
    },
    {
      "name": "primary",
      "match": "$.server.memberRole = 'PRIMARY'"
    },
    {
      "name": "testing",
      "match": "$.server.version = 90020"
    }
  ]
  ...
```

From A Friend's Challenge To Smarter Routing

```
{
  "name": "frontend_traffic",
  "match": "$.session.user IN ('frontend', 'app')",
  "destinations": [
    {
      "classes": ["secondaries"],
      "strategy": "round-robin",
      "priority": 0
    },
    {
      "classes": ["frontend_read_replicas"],
      "strategy": "round-robin",
      "priority": 1
    },
    {
      "classes": ["primary"],
      "strategy": "first-available",
      "priority": 2
    }
  ],
  "connectionSharingAllowed": true,
  "enabled": true
}
```

- Frontend traffic: Prioritize secondaries, fallback to read-replicas and primary
  - `$session.user`

```
{
  "name": "bi_traffic",
  "match":"CONTAINS($.session.connectAttrs['_platform'], 'arm64') AND
           STARTS_WITH($.session.user, 'bi_acct_')",
  "destinations": [
    {
      "classes": ["bi_read_replicas"],
      "strategy": "round-robin",
      "priority": 0
    },
    {
      "classes": ["frontend_read_replicas"],
      "strategy": "round-robin",
      "priority": 1
    }
  ],
  "connectionSharingAllowed": true,
  "enabled": true
}
```

- BI traffic: Prioritize read-replicas dedicated for it
  - `$.session.connectAttrs`
  - `$.session.user`

Closing and Q&A

Takeaways, Resources, And Discussion

Takeaways

- Smarter Routing: Routing Guidelines enable dynamic, flexible, and declarative query routing

- Effortless Management: MySQL Shell / AdminAPI makes defining and managing Routing Guidelines straightforward

- Future-Ready Architectures: Routing Guidelines empower scalable, resilient, and flexible MySQL setups, seamlessly handling complex topologies.

Resources

- Cookbook: https://github.com/mysql/mysql-shell/blob/master/ROUTING_GUIDELINES.md
- Documentation: https://dev.mysql.com/doc/mysql-shell/en/admin-api-routing-guidelines.html
- Community: Join our Slack workspace: bit.ly/mysql-slack
  - #mysql_innodb_cluster
  - #router
  - #shell

Thank you!

Questions?