ORACLE

MySQL Routing Guidelines

Designing Smarter Query Routing Together

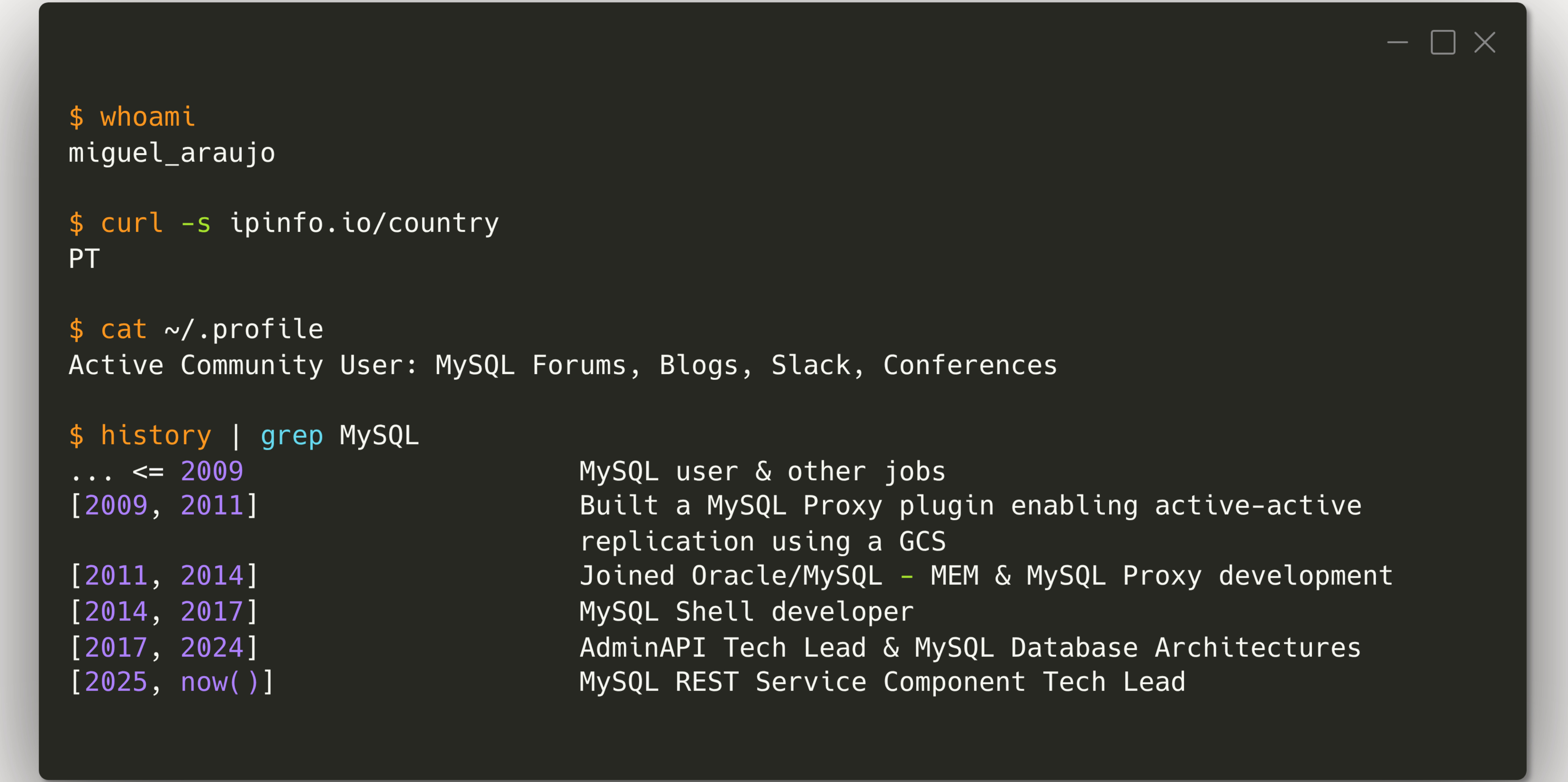
Miguel Araújo

Senior Principal Software Engineer

MySQL, Oracle

January 29, 2026

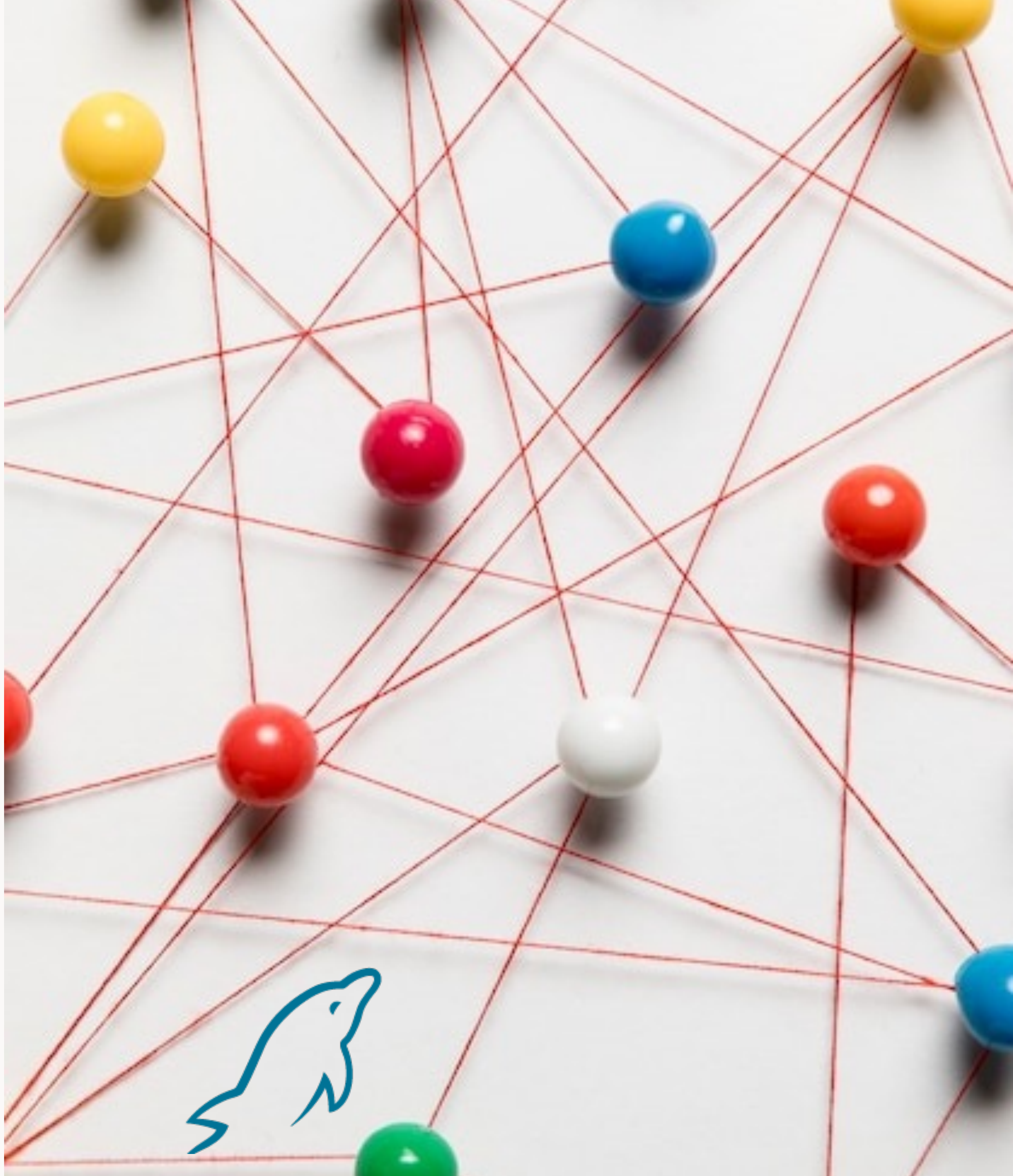
MySQL



```
$ whoami
miguel_araujo

$ curl -s ipinfo.io/country
PT

$ cat ~/.profile
Active Community User: MySQL Forums, Blogs, Slack, Conferences

$ history | grep MySQL
... <= 2009                     MySQL user & other jobs
[2009, 2011]                    Built a MySQL Proxy plugin enabling active-active
                                replication using a GCS
[2011, 2014]                    Joined Oracle/MySQL - MEM & MySQL Proxy development
[2014, 2017]                    MySQL Shell developer
[2017, 2024]                    AdminAPI Tech Lead & MySQL Database Architectures
[2025, now()]                   MySQL REST Service Component Tech Lead
```

Routing Guidelines

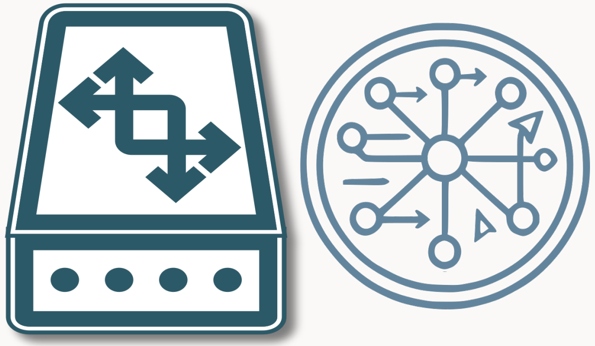
A quick refresher



Why Routing Guidelines

"In modern database architectures, efficient query routing is essential for achieving performance, scalability, resilience, and adaptability."

- Modern topologies run **multiple**, **conflicting workloads**
- Different applications have **different routing requirements**
- **Static** routing rules **cannot evolve** with the topology

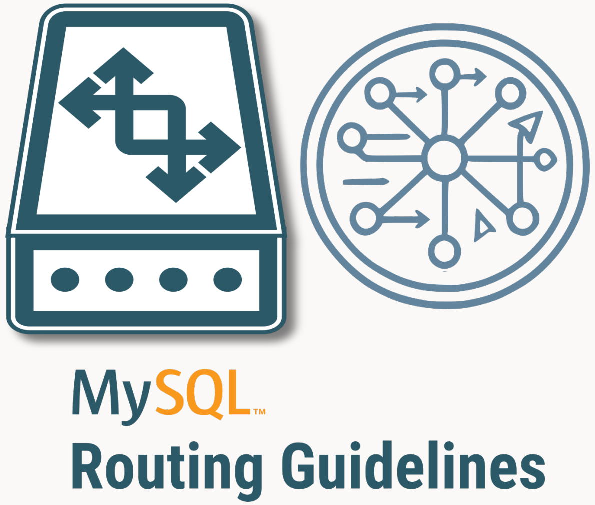


MySQL™
Routing Guidelines

What are Routing Guidelines?

> “Routing Guidelines are a declarative way to express routing intent, evaluated dynamically by MySQL Router.”

- Express routing intent using **rule-based matching**
- Use **session**, **router**, and **server** attributes as input
- Enable **workload-aware routing** beyond read/write splitting



How Routing Guidelines work

Destinations

- **Groups** servers with a shared **purpose**
- Defines eligible **targets** for routing

Routes

- **Rule-based matching** on context
- Uses **session**, **router**, and **server** attributes

Order

- Routes are **evaluated** top to bottom
- **First** matching rule is applied

Matching rules

Predefined variables

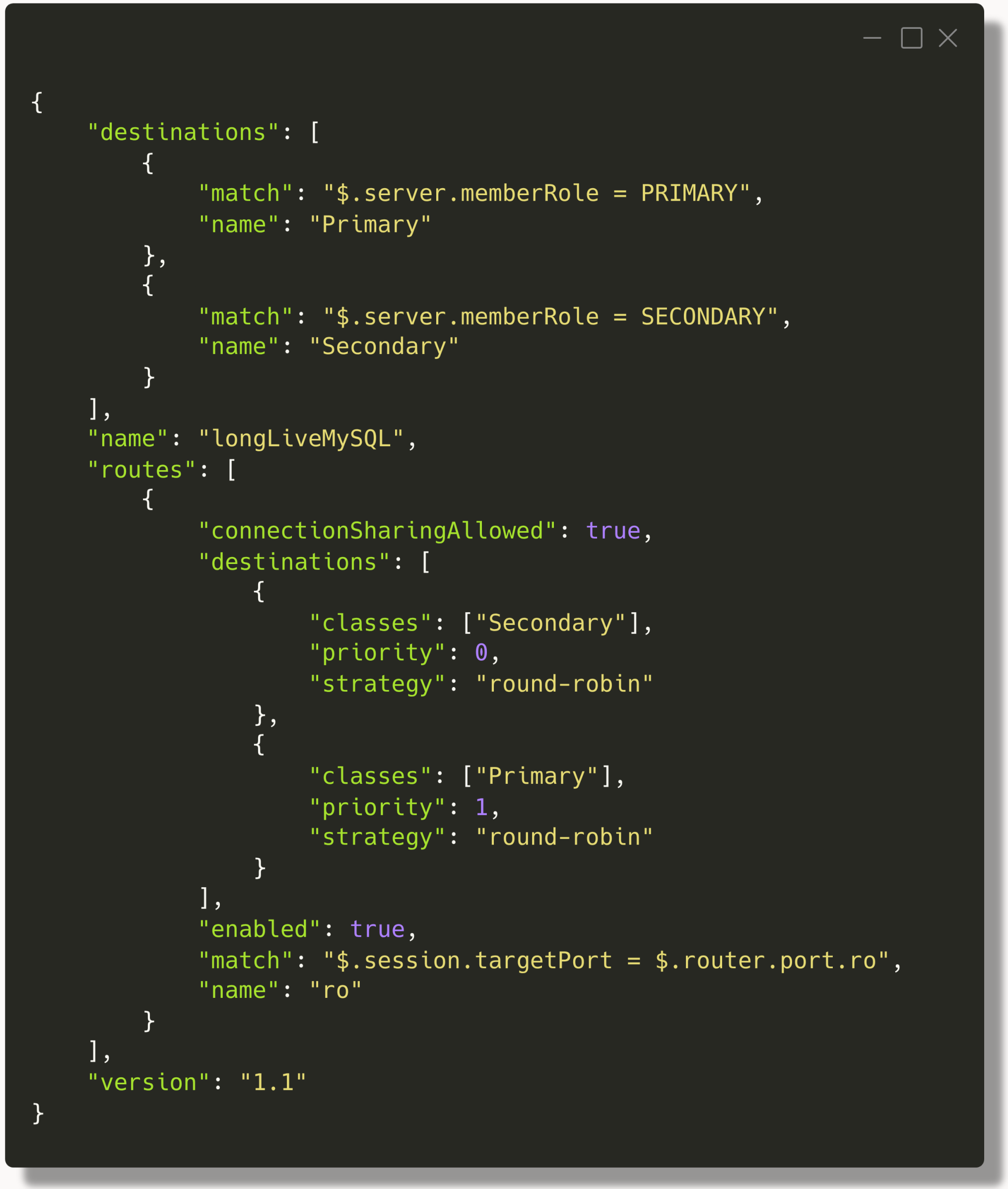
- `$.server.*`
  - Related to the MySQL Server
    - `$.server.version`,
    - `$.server.address`, etc.
- `$.session.*`
  - Related to the Client session
    - `$.session.user`,
    - `$.session.sourceIp`, etc.
- `$.router.*`
  - Related to the Router instance
    - `$.router.hostname`,
    - `$.router.port.rw`, etc.

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.`

A Routing Guideline example

- Destinations
  - `Primary`
  - `Secondary`
- Routes
  - `ro`
- Name
  - `longLiveMySQL`
- Version
  - `1.1`



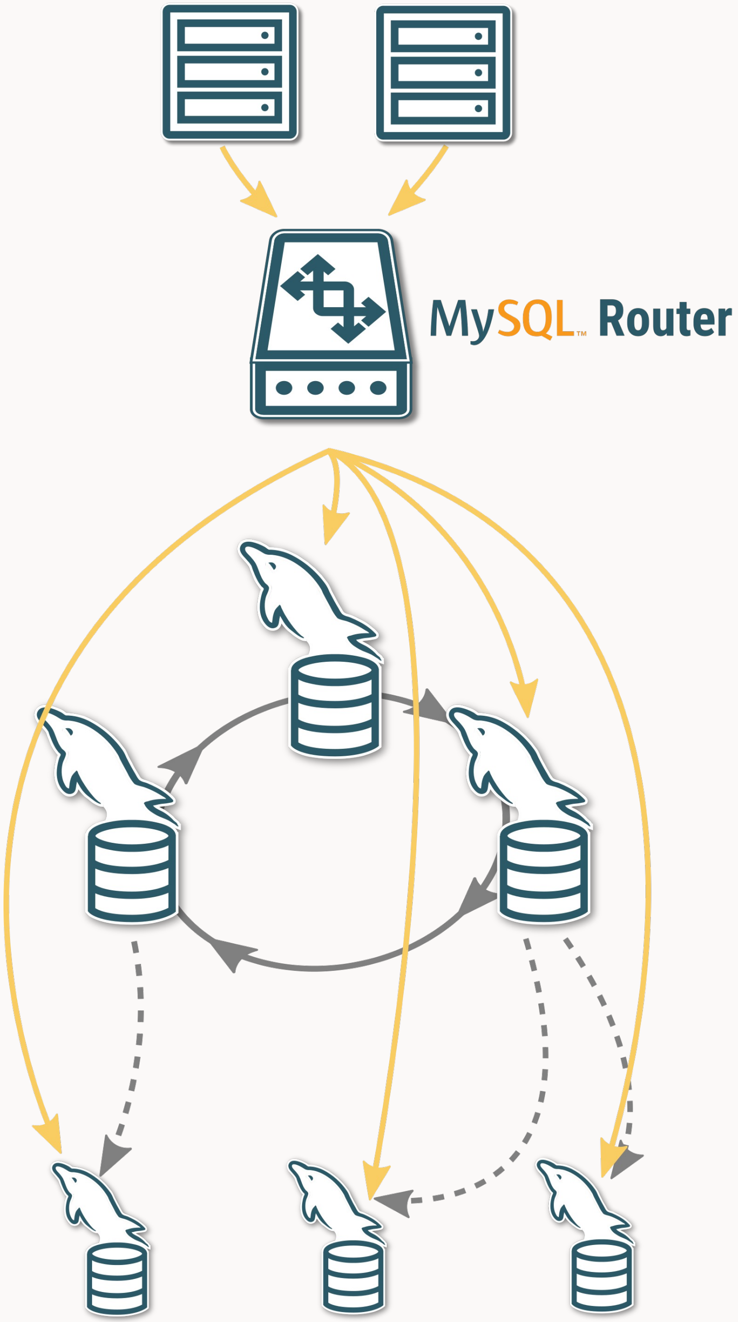
```
{
    "destinations": [
        {
            "match": "$.server.memberRole = PRIMARY",
            "name": "Primary"
        },
        {
            "match": "$.server.memberRole = SECONDARY",
            "name": "Secondary"
        }
    ],
    "name": "longLiveMySQL",
    "routes": [
        {
            "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": "1.1"
}
```

The scenario

One Cluster, multiple conflicting workloads

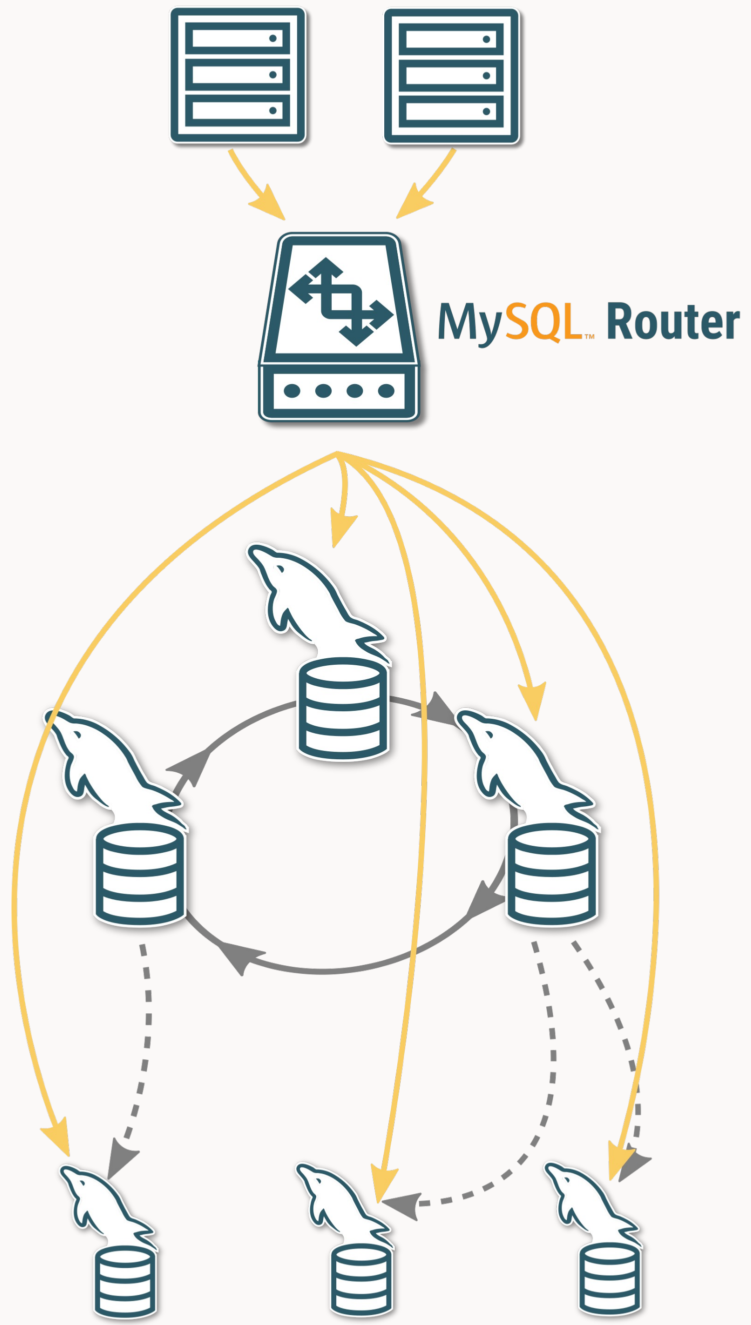
The scenario

- One MySQL **InnoDB Cluster** serving multiple, conflicting workloads
- OLTP traffic must stay **fast and predictable**
- Read replicas used for **backups, analytics,** and **reporting**
- Routing decisions handled centrally by MySQL **Router**



By default...

- **OLTP reads compete** with long-running analytics and reports
- **Backups, analytics, and reporting** introduce **latency** spikes on replicas
- **Read/write splitting alone** can’t express workload intent



Designing the Routing Guideline

Turning requirements into routing decisions

Design requirements

Operational pressures

- OLTP traffic is highly **latency-sensitive**
- Analytics, reporting, and backups are **long-running and disruptive**
- Routing logic must stay **outside applications**

Distinct workloads

- **OLTP** (latency-sensitive)
- **Backups** (long-running, bursty)
- **Analytics** (heavy scans)
- **Reporting** (large queries)

Decision #1:

What should OLTP traffic compete with?

Nothing
(fully isolated)

Other reads
(shared read replicas)

Analytics & Reporting
(long-running queries)

1 OR 2 OR 3

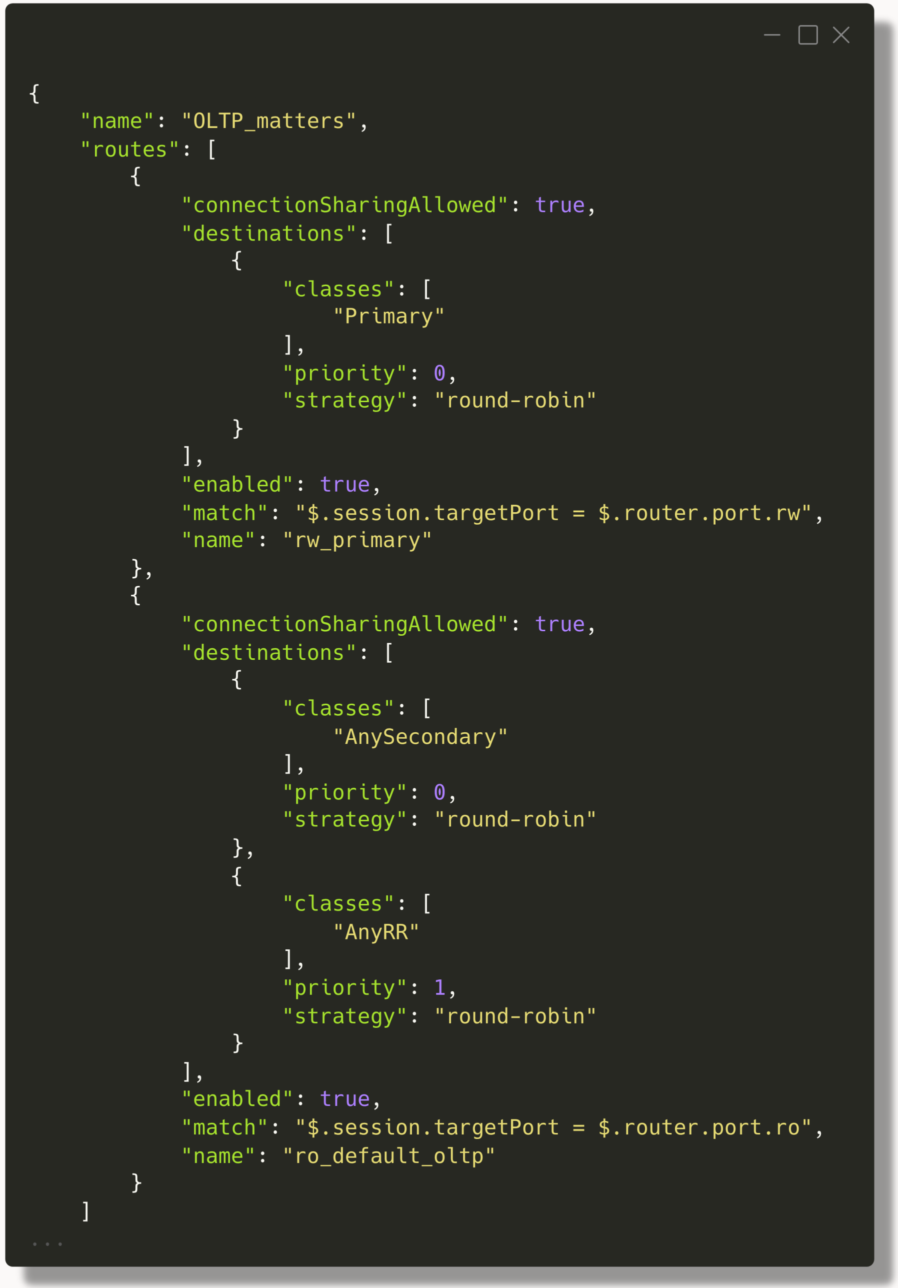
Decision #1: Isolate OLTP from heavy workloads

What we decided

- OLPT must stay fast and predictable
- Analytics and reporting must not interfere with OLTP

What this implies for routing

- OLTP traffic needs a **dedicated path**
- Heavy reads must be **explicitly routed elsewhere**
- "read-only" alone is **not enough** to express workload intent



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

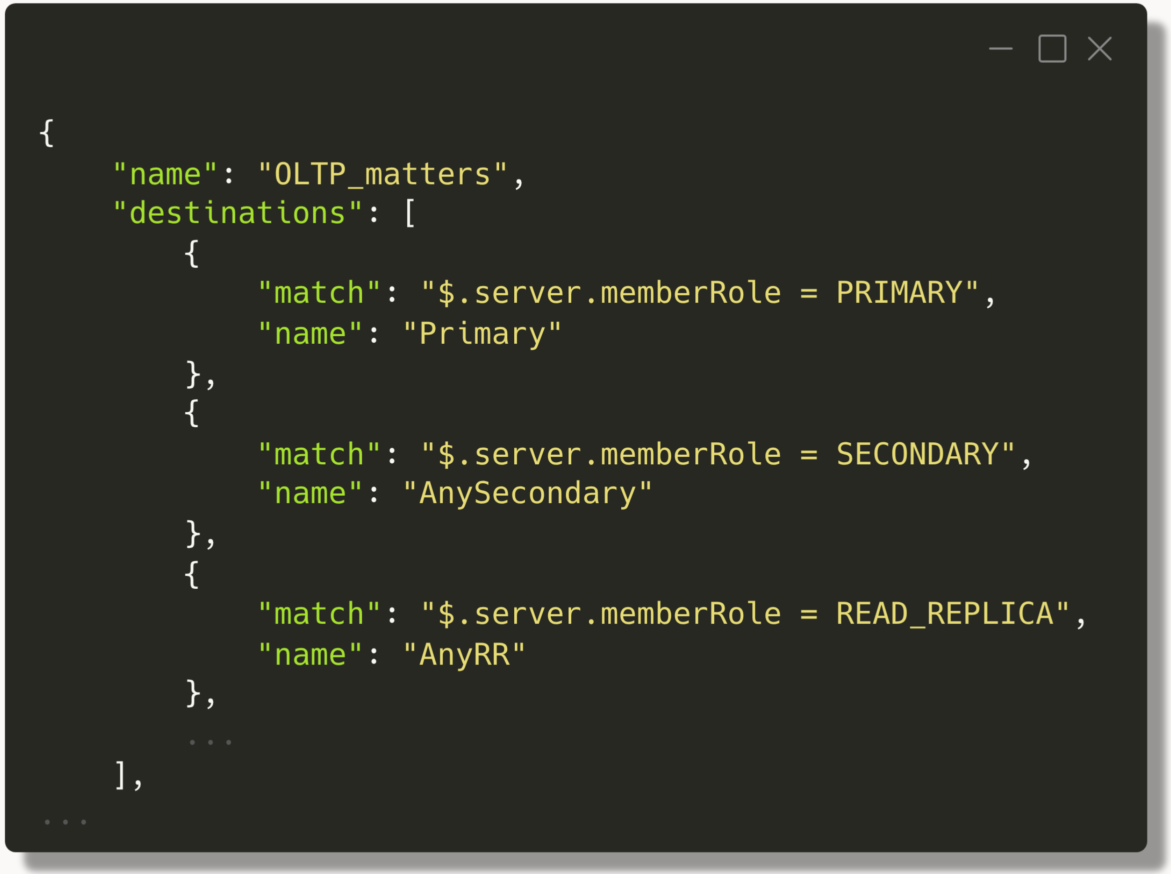
Decision #2: Define destinations

What we need

- Separate OLTP, backups, analytics, and reporting
- Keep routing independent of topology
- Avoid hardcoding hostnames or instance

What this implies

- Destinations must be logical groups
- Based on server attributes, not addresses
- Reusable across multiple routes

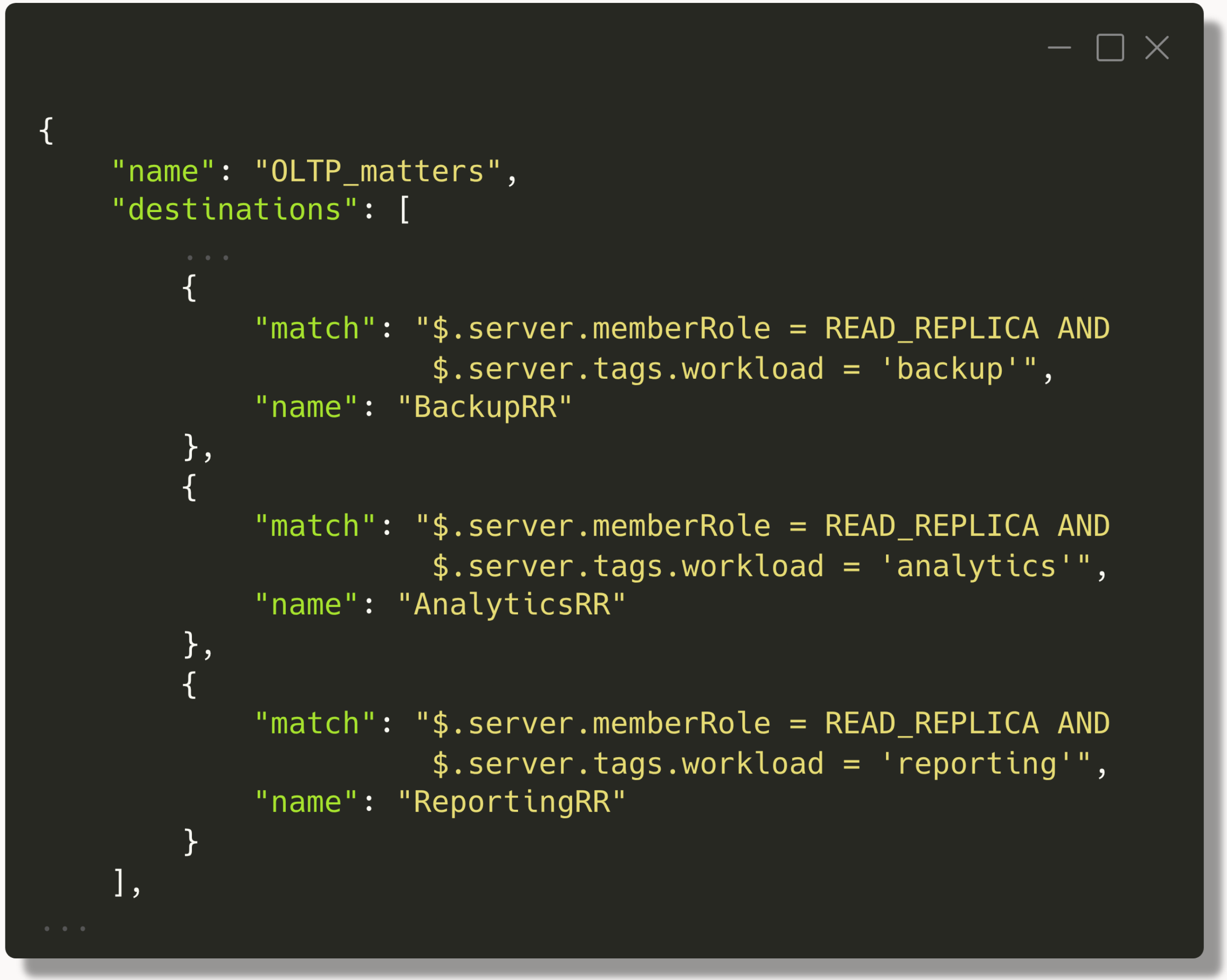


```
{
    "name": "OLTP_matters",
    "destinations": [
        {
            "match": "$.server.memberRole = PRIMARY",
            "name": "Primary"
        },
        {
            "match": "$.server.memberRole = SECONDARY",
            "name": "AnySecondary"
        },
        {
            "match":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

Decision #2: Define destinations

Destinations

- `Primary`
- `AnySecondary`
- `AnyRR`
- `BackupRR`
- `AnalyticsRR`
- `ReportingRR`



```
{
    "name": "OLTP_matters",
    "destinations": [
        ...
        {
            "match": "$.server.memberRole = READ_REPLICA AND
                      $.server.tags.workload = 'backup'",
            "name": "BackupRR"
        },
        {
            "match": "$.server.memberRole = READ_REPLICA AND
                      $.server.tags.workload = 'analytics'",
            "name": "AnalyticsRR"
        },
        {
            "match": "$.server.memberRole = READ_REPLICA AND
                      $.server.tags.workload = 'reporting'",
            "name": "ReportingRR"
        }
    ],
...
```

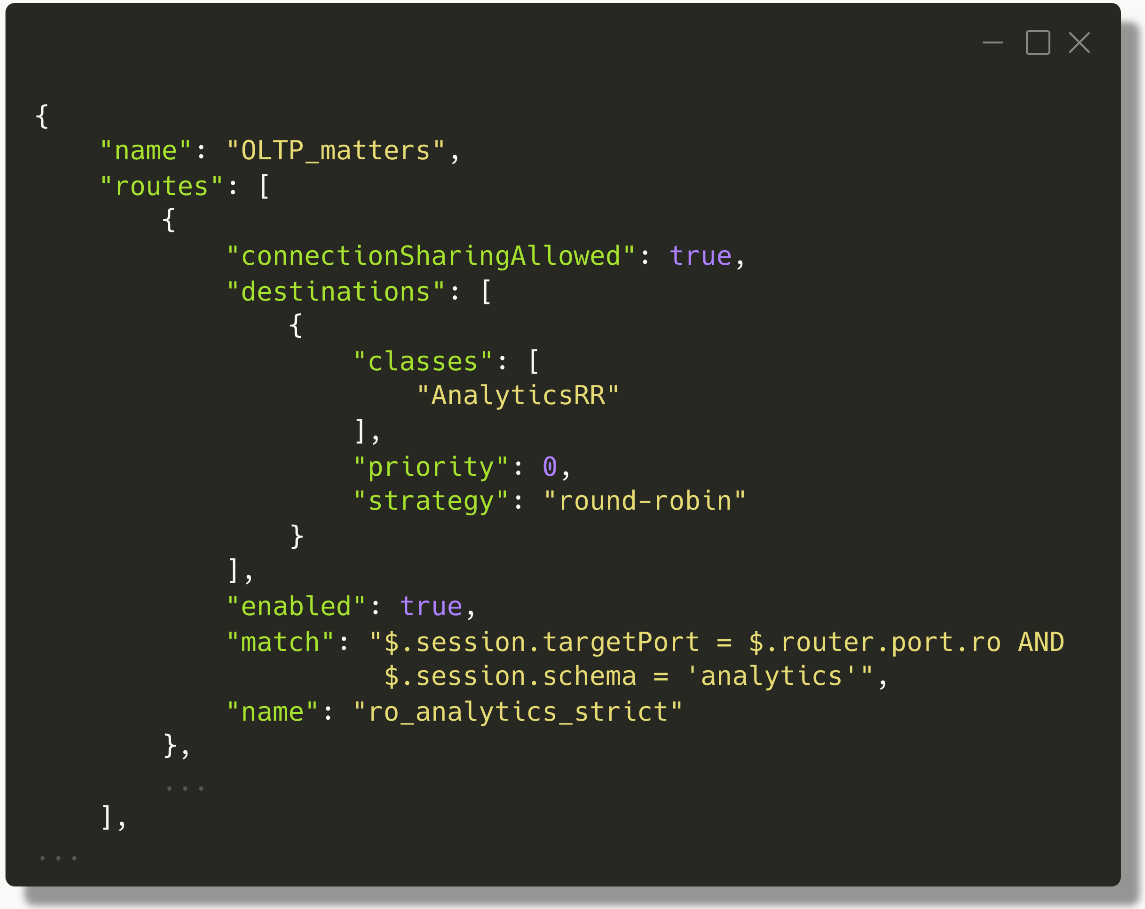
Decision #3: Express workload intent

The problem

- All traffic is just connections
- **RO** vs **RW** is not enough
- Analytics, reporting, backups all look like **reads**

The decision

- Workload intent must come from the **session**
- Not from server roles
- Not from hardcoded ports



```
{
    "name": "OLTP_matters",
    "routes": [
        {
            "connectionSharingAllowed": true,
            "destinations": [
                {
                    "classes": [
                        "AnalyticsRR"
                    ],
                    "priority": 0,
                    "strategy": "round-robin"
                }
            ],
            "enabled": true,
            "match": "$.session.targetPort = $.router.port.ro AND
                      $.session.schema = 'analytics'",
            "name": "ro_analytics_strict"
        },
        ...
    ],
...
```

Decision #4:

When `analyticsRR` is unavailable, what's the fallback?

Fail fast
(no fallback)

1

OR

Fallback to AnyRR
(best-efforts reads)

2

OR

Fallback to Secondaries
(keeps service up, risk OLTP)

3

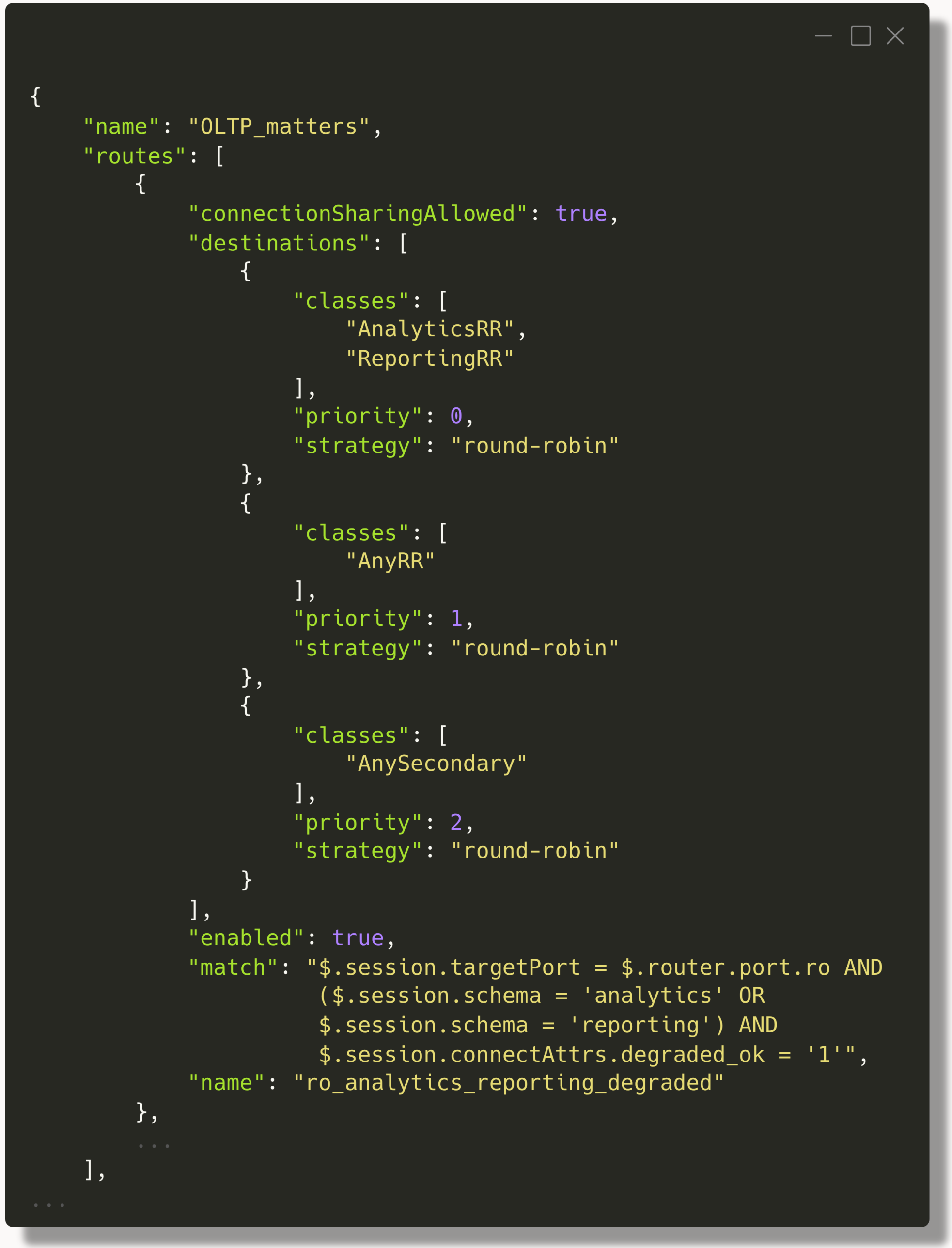
Decision #4: Controlled degradation

Trade-off

- Availability **vs** OLTP predictability

Policy

- Degradation is **explicit opt-in**
- Triggered via **connection attribute**

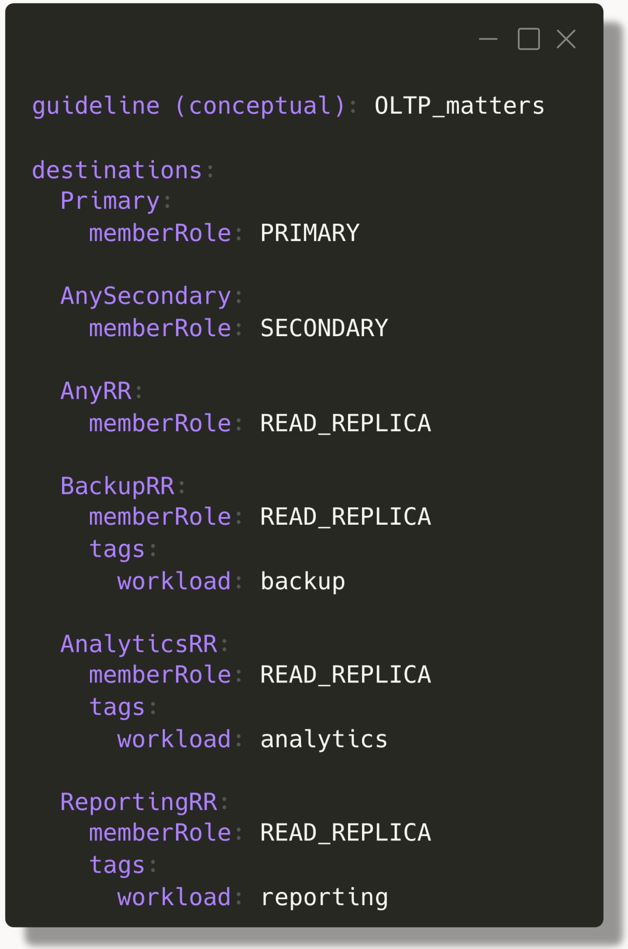


```
{
    "name": "OLTP_matters",
    "routes": [
        {
            "connectionSharingAllowed": true,
            "destinations": [
                {
                    "classes": [
                        "AnalyticsRR",
                        "ReportingRR"
                    ],
                    "priority": 0,
                    "strategy": "round-robin"
                },
                {
                    "classes": [
                        "AnyRR"
                    ],
                    "priority": 1,
                    "strategy": "round-robin"
                },
                {
                    "classes": [
                        "AnySecondary"
                    ],
                    "priority": 2,
                    "strategy": "round-robin"
                }
            ],
            "enabled": true,
            "match": "$.session.targetPort = $.router.port.ro AND
                      ($.session.schema = 'analytics' OR
                      $.session.schema = 'reporting') AND
                      $.session.connectAttrs.degraded_ok = '1'",
            "name": "ro_analytics_reporting_degraded"
        },
        ...
    ],
...
```

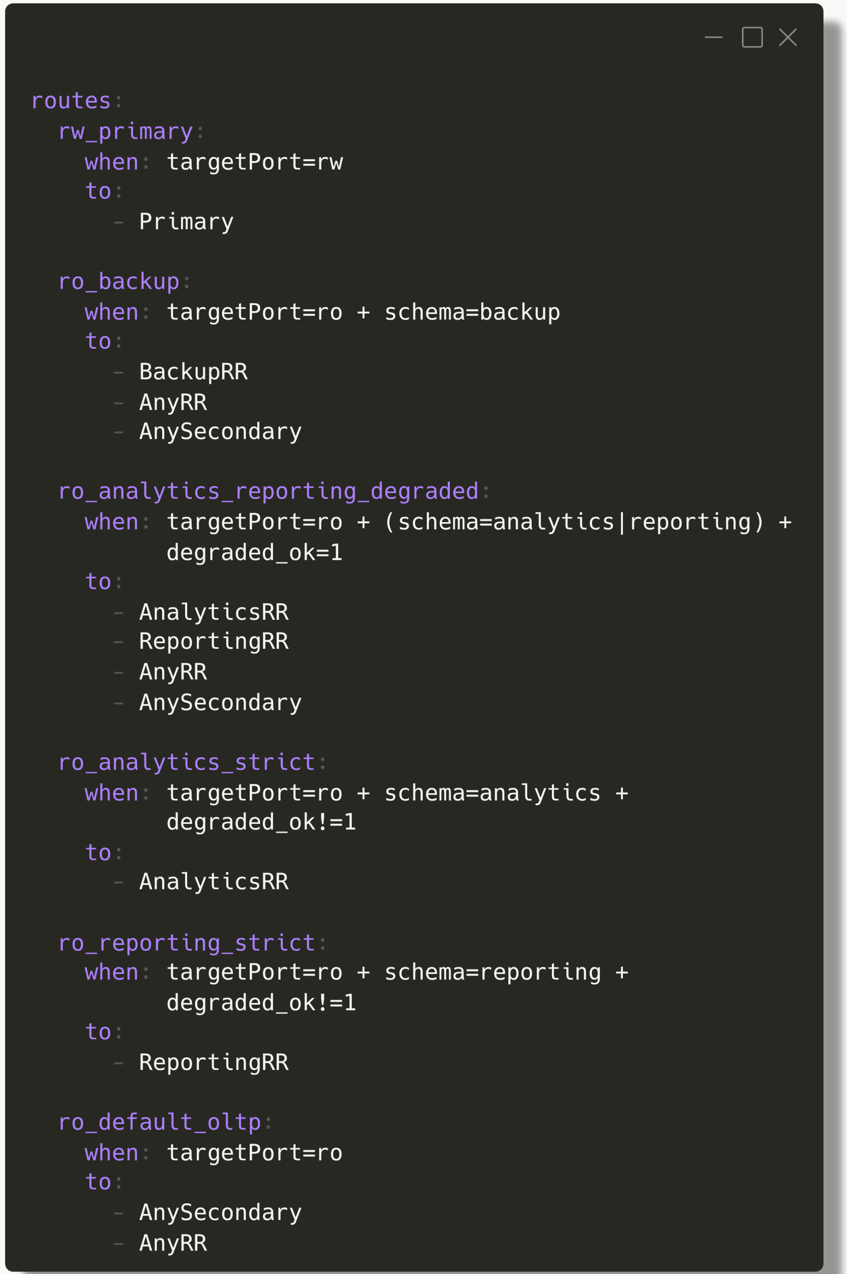
Summary: Putting it all together

What we designed together

1. OLTP isolation
2. Destination classes
3. Explicit workload intent
4. Controlled degradation



```
guideline (conceptual): OLTP_matters

destinations:
  Primary:
    memberRole: PRIMARY

  AnySecondary:
    memberRole: SECONDARY

  AnyRR:
    memberRole: READ_REPLICA

  BackupRR:
    memberRole: READ_REPLICA
    tags:
      workload: backup

  AnalyticsRR:
    memberRole: READ_REPLICA
    tags:
      workload: analytics

  ReportingRR:
    memberRole: READ_REPLICA
    tags:
      workload: reporting
```



```
routes:
  rw_primary:
    when: targetPort=rw
    to:
      - Primary

  ro_backup:
    when: targetPort=ro + schema=backup
    to:
      - BackupRR
      - AnyRR
      - AnySecondary

  ro_analytics_reporting_degraded:
    when: targetPort=ro + (schema=analytics|reporting) +
          degraded_ok=1
    to:
      - AnalyticsRR
      - ReportingRR
      - AnyRR
      - AnySecondary

  ro_analytics_strict:
    when: targetPort=ro + schema=analytics +
          degraded_ok!=1
    to:
      - AnalyticsRR

  ro_reporting_strict:
    when: targetPort=ro + schema=reporting +
          degraded_ok!=1
    to:
      - ReportingRR

  ro_default_oltp:
    when: targetPort=ro
    to:
      - AnySecondary
      - AnyRR
```

Conceptual representation!

Full guideline at:

https://github.com/miguelaraujo/mysql-routing-guidelines-examples

What Routing Guidelines are (and are not)

What they ARE	What they are NOT
Declarative routing **policy**	Not a SQL optimizer
Evaluated per **session**	Not per-query routing
Expresses **intent**, not topology	Not application logic
Centralized and **external to applications**	Not a replacement for schema design
Dynamic and **topology-aware**	Not "just" read/write splitting

Thank you!

Questions?