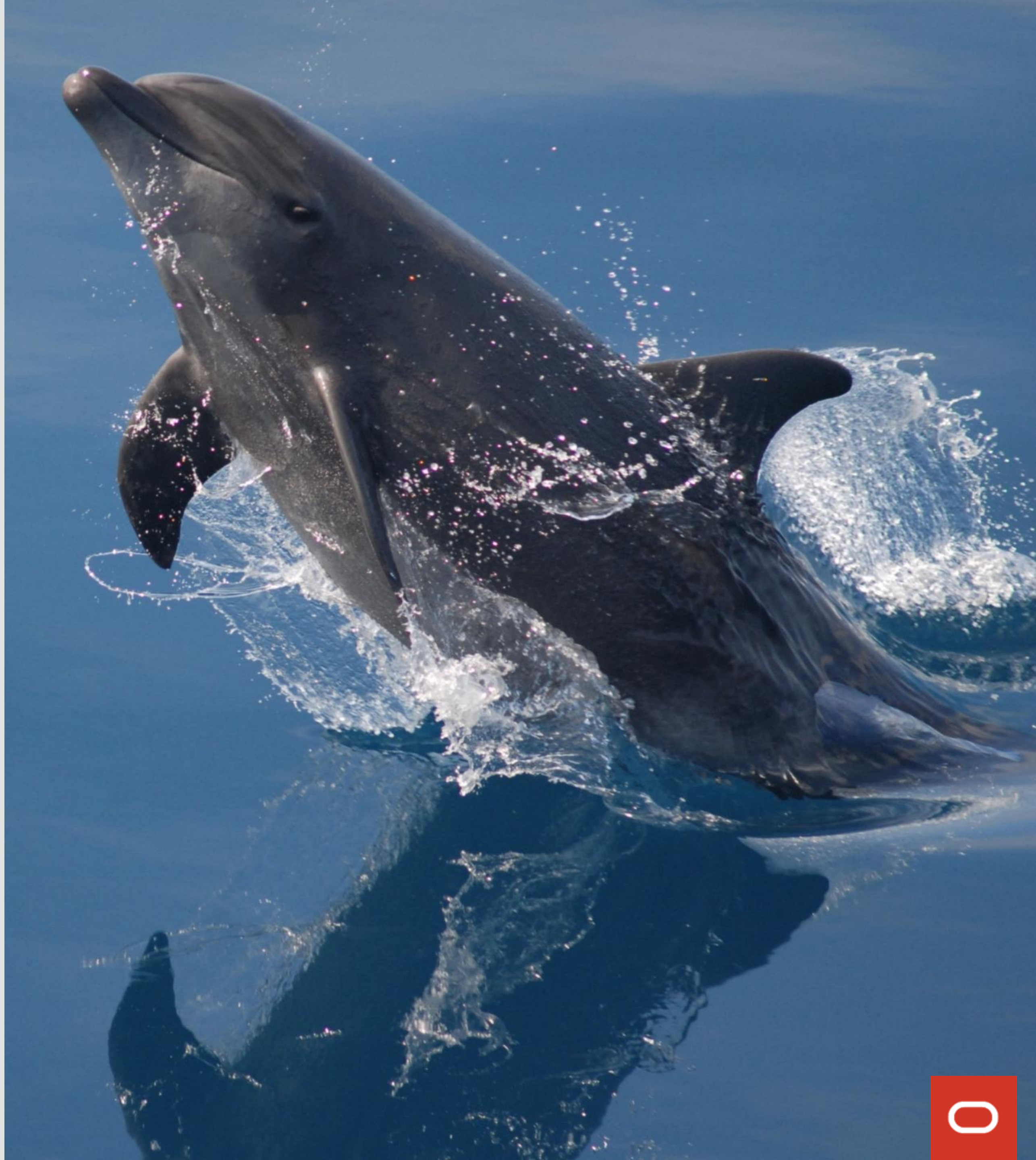
MySQL Goes to 8!

Geir Høydalsvik, MySQL Engineering

FOSDEM 2020, Database Track

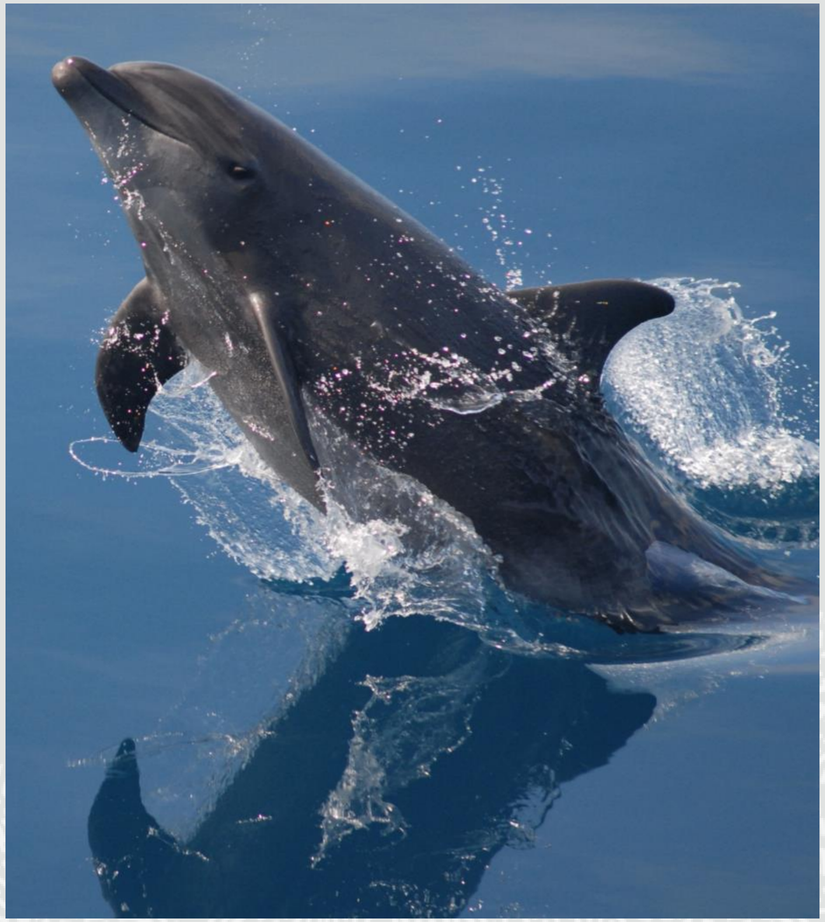
February 2nd, 2020, Brussels



Safe harbor statement

WHAT IS MySQL?

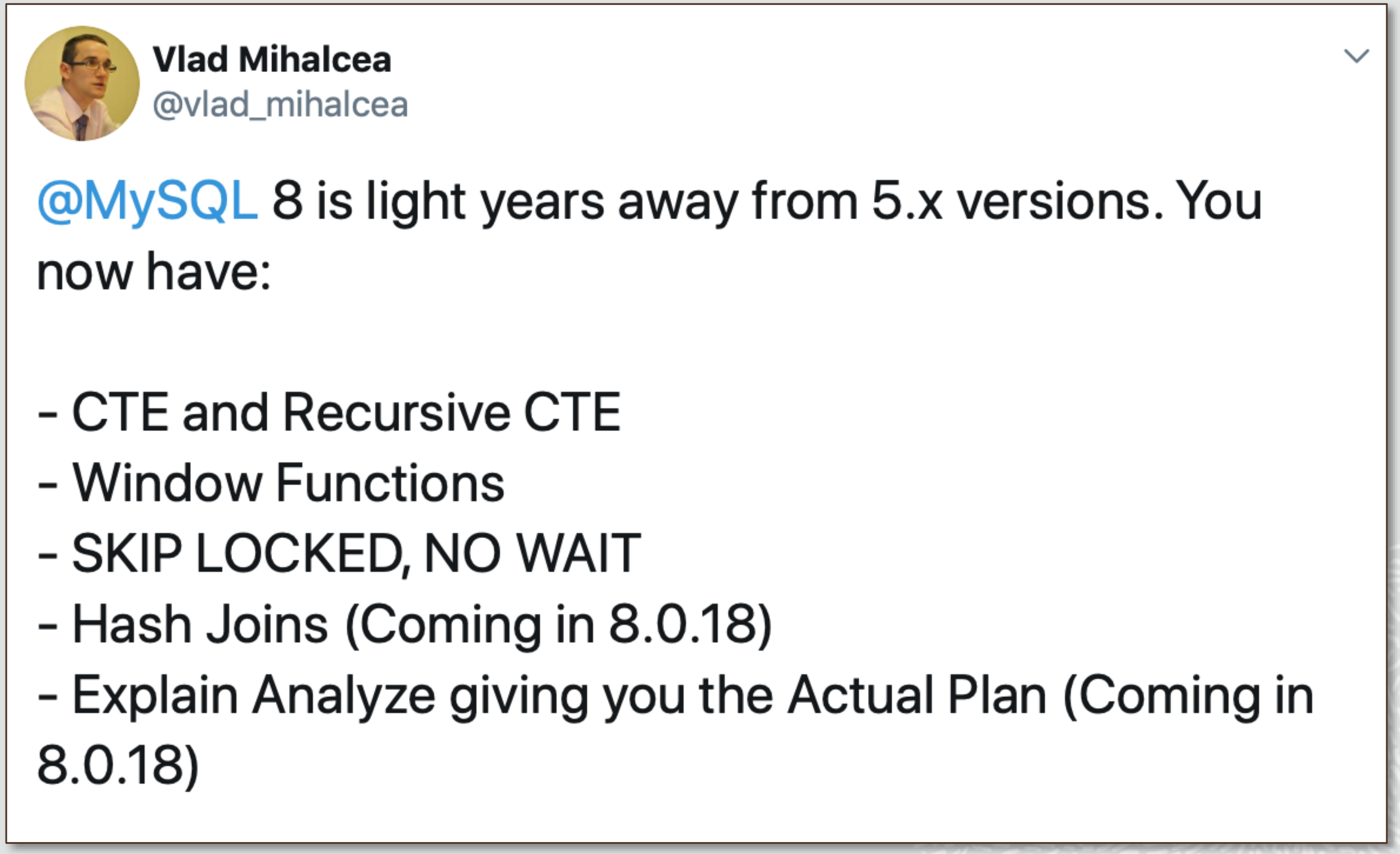
- Relational Database
  - Transactional, ACID
  - InnoDB storage engine: ARIES, MVCC
  - OLTP: low latency, high throughput
- Replication
  - Read scale out, High Availability
- Simple, Solid, Secure
  - Easy to use, Proven at scale



LAST 10 YEARS

- Major investments
  - Reengineering
  - Features
  - Quality
- Major releases
  - MySQL 5.5
  - MySQL 5.6
  - MySQL 5.7
  - MySQL 8.0

MySQL 8 - IS LIGHT YEARS AWAY FROM 5.X



Vlad Mihalcea
@vlad_mihalcea

@MySQL 8 is light years away from 5.x versions. You now have:

- CTE and Recursive CTE
- Window Functions
- SKIP LOCKED, NO WAIT
- Hash Joins (Coming in 8.0.18)
- Explain Analyze giving you the Actual Plan (Coming in 8.0.18)

The Basics

SQL, JSON, GIS, Character Sets, Collations, Functions

MySQL 8 - OPTIMIZER

Parse phase

- Parse step
- Contextualization step
- **Abstract Syntax Tree**

Optimize phase

- Range optimizer
- Join optimizer
- **Physical plan**

Prepare phase

- Resolve step
- Transform step
- **Logical Plan**

Execute phase

- Produce iterator tree
- Execute iterator
- **Resultset**

MySQL 8 - HISTOGRAM

- Provides the optimizer with information about column value distribution

  ```
  ANALYZE TABLE table UPDATE HISTOGRAM ON column WITH n BUCKETS;
  ```

- Table sampling for efficiency

MySQL 8 – ITERATOR EXECUTOR

- **Each operation is an iterator**
- **Execution loop reads from root node**

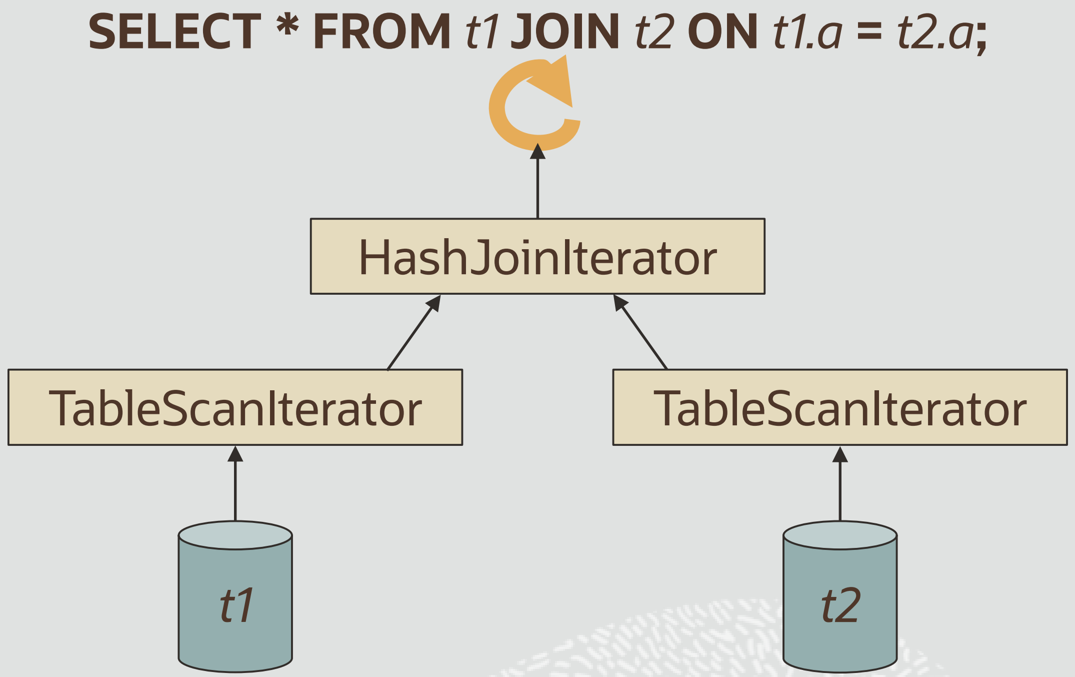
Row by row

May trigger multiple read calls further down

- **Common interface**

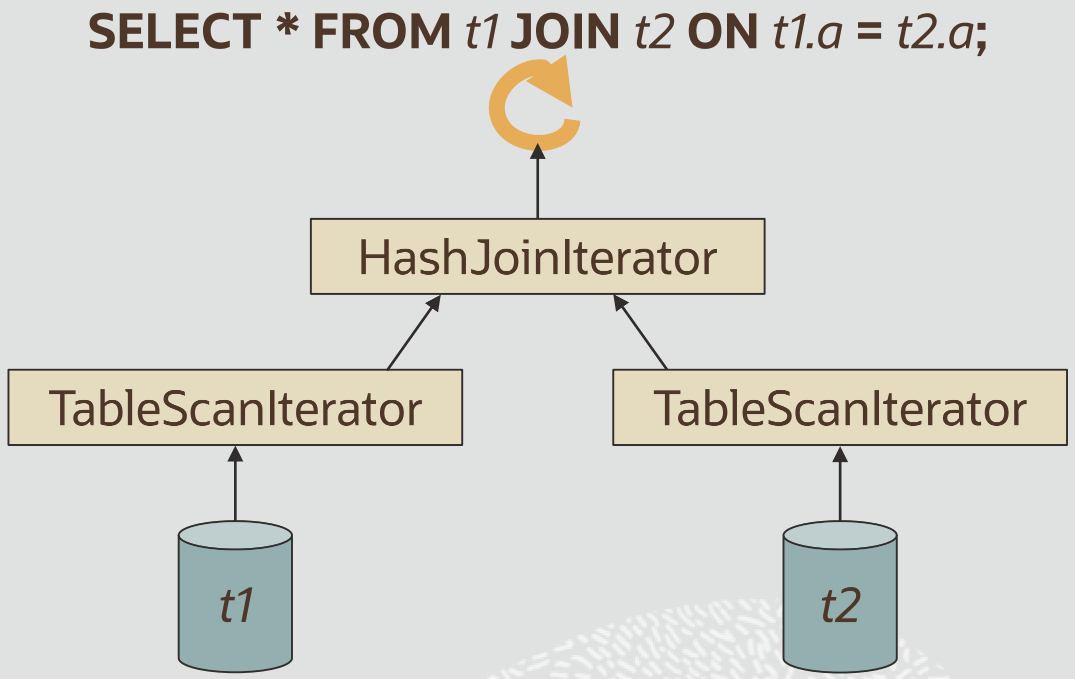
Init()

Read()



MySQL 8 - HASH JOIN

- “Just another iterator”
- Faster than Block Nested Loop
- In-memory if possible
- Spill to disk if necessary
- Used for inner equi-joins in 8.0.18
  - And also for outer, semi and anti joins in 8.0.20
- Hash Join replaces Block Nested Loop



MySQL 8 - EXPLAIN ANALYZE

- **Wrap iterators in instrumentation nodes**
- **Measurements**

  Time (in ms) to first row

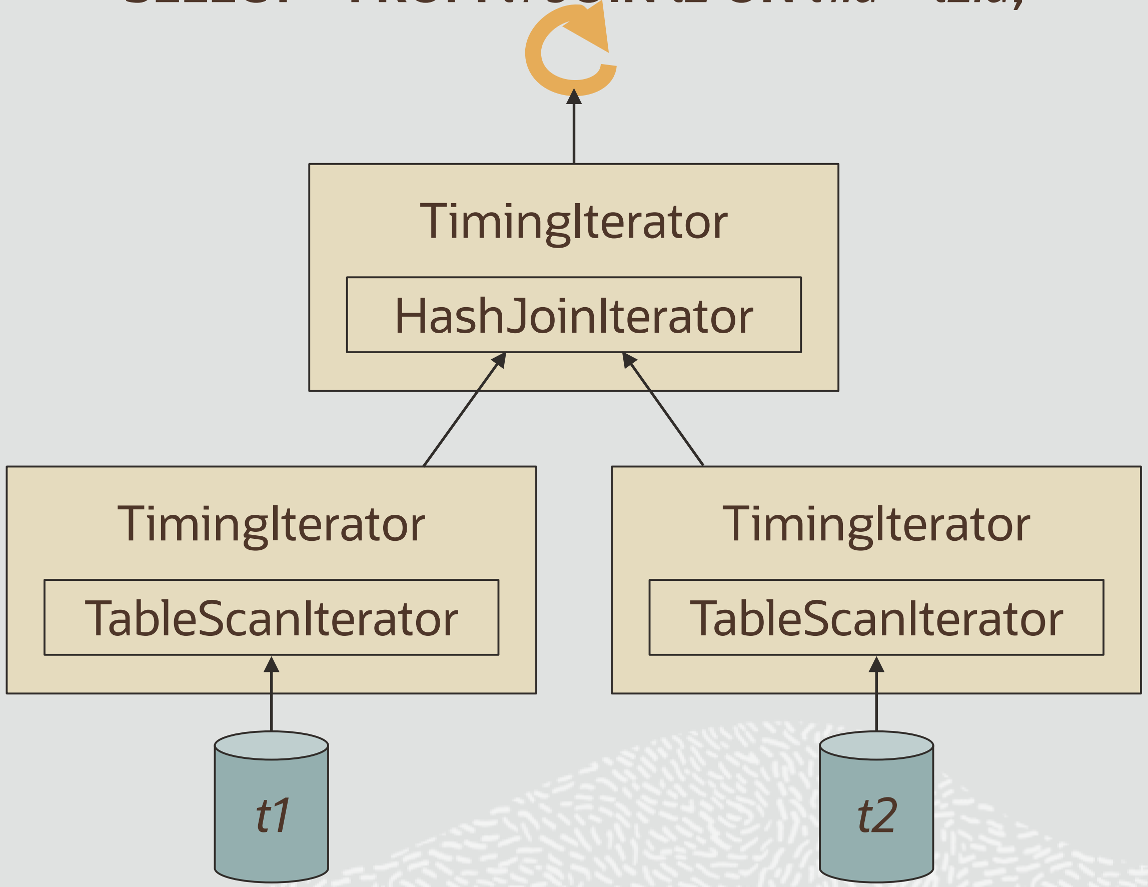
  Time (in ms) to last row

  Number of rows

  Number of loops
- **Execute the query and dump the stats**
- **Built on EXPLAIN FORMAT=TREE**

EXPLAIN ANALYZE

SELECT * FROM *t1* **JOIN** *t2* **ON** *t1.a* **=** *t2.a***;**



```
-> Inner hash join (t2.a = t1.a)  (cost=0.70 rows=1) (actual time=0.441..0.441 rows=0 loops=1)
    -> Table scan on t2  (cost=0.35 rows=1) (never executed)
    -> Hash
        -> Table scan on t1  (cost=0.35 rows=1) (actual time=0.220..0.220 rows=0 loops=1)
```

MySQL 8 – CHARACTER SET AND COLLATIONS

- MySQL 8 defaults to UTF-8
- Emoji, CJK characters, …
- Unicode 9.0 collations with accent, case, and kana sensitivity
- Unicode support for REGEXP

MySQL 8 - Common Table Expression (WITH clause)

- **Non-recursive**

```
WITH cte AS (subquery)
SELECT … FROM cte, t1…
```

- **Recursive**

```
WITH RECURSIVE cte AS
( SELECT … FROM table_name
  UNION [DISTINCT|ALL]
  SELECT … FROM cte, table_name )
SELECT … FROM cte;
```

A Common Table Expression (CTE) is **just like a derived table**, but its declaration is put before the query block instead of in the FROM clause

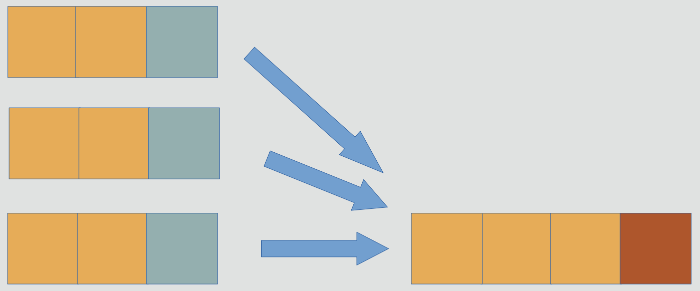
- Better readability
- Can be referenced multiple times
- Can refer to other CTEs
- Improved performance

MySQL 8 - Window Functions (OVER clause)

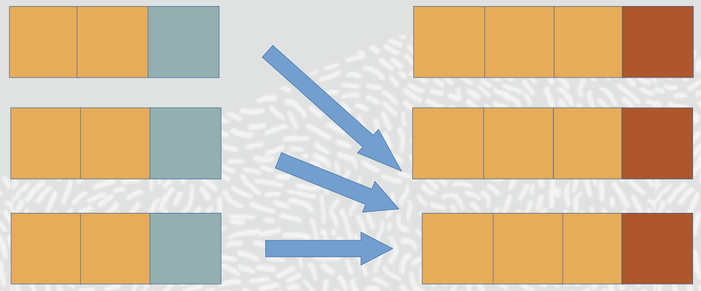
A window function performs a calculation across a set of rows that are related to the current row, similar to an aggregate function.

But unlike aggregate functions, a window function does not cause rows to become grouped into a single output row.

Window functions can access values of other rows “in the vicinity” of the current row.



Aggregate function



Window function

MySQL 8 - Window Functions (OVER clause)

Sum up total salary for each department:

```
SELECT name, dept_id, salary,
   SUM(salary)
       OVER (PARTITION BY dept_id)
       AS dept_total
FROM employee
ORDER BY dept_id, name;
```

PARTITION == disjoint set of rows in result set

name	dept_id	salary	dept_total
Newt	NULL	75000	75000
Dag	10	NULL	370000
Ed	10	100000	370000
Fred	10	60000	370000
Jon	10	60000	370000
Michael	10	70000	370000
Newt	10	80000	370000
Lebedev	20	65000	130000
Pete	20	65000	130000
Jeff	30	300000	370000
Will	30	70000	370000

MySQL 8 – LATERAL DERIVED TABLES

- Can refer to other tables in the same FROM clause
- Sometimes referred to as the SQL «for each» equivalent

```
SELECT … FROM t1, LATERAL (SELECT … FROM … WHERE … = t1.col)
AS derived, t2 …
```

MySQL 8.0 - FUNCTIONAL INDEXES

- Index over an expression

```
CREATE TABLE t1 (col1 INT, col2 INT);
CREATE INDEX idx1 ON t1 ((col1 + col2), (col1 - col2), col1) ;
```

- Document content, e.g. JSON array

```
CREATE TABLE lottery (data JSON);
CREATE INDEX ticket_idx ON lottery
        ((CAST(data->'$.lottery_tickets' AS UNSIGNED INT ARRAY))) ;
```

MySQL 8.0 – INVISIBLE INDEXES

- Indexes are “hidden” to the MySQL Optimizer
  - Not the same as “disabled indexes”
  - Contents are fully up to date and maintained by DML
- Two use cases:
  - Soft Delete: What will happen if I delete this index?
  - Staged Rollout: I will create this index over night and make it visible when I am at work tomorrow

MySQL 8 - CHECK CONSTRAINT

- Standard SQL Syntax

```
[ CONSTRAINT [symbol] ] CHECK ( condition) [ [ NOT ] ENFORCED ]
```

- Example

```
CREATE TABLE t1 (c1 INTEGER CONSTRAINT c1_chk CHECK (c1 > 0) ,
                 c2 INTEGER CONSTRAINT c2_chk CHECK (c2 > 0) ,
                 CONSTRAINT c1_c2_chk CHECK (c1 + c2 < 9999) );
```

MySQL 8 - Expressions as Default Values

- No longer limited to literal values

```
CREATE TABLE t1 (uuid BINARY DEFAULT (UUID_TO_BIN(UUID())) );
CREATE TABLE t2 (a INT, b INT, c INT DEFAULT (a+b) );
CREATE TABLE t3 (a INT, b INT, c POINT DEFAULT (POINT(0,0)) );
CREATE TABLE t4 (a INT, b INT, c JSON DEFAULT (‘[]’) );
```

- Useful for types without literal values
  - GEOMETRY, POINT, LINESTRING, POLYGON, ...

MySQL 8 - NOWAIT and SKIP LOCKED

```
SELECT * FROM tickets
WHERE id IN (1,2,3,4)
AND order_id IS NULL
FOR UPDATE
NOWAIT;
```

Error immediately if a row is already locked

```
SELECT * FROM tickets
WHERE id IN (1,2,3,4)
AND order_id IS NULL
FOR UPDATE
SKIP LOCKED;
```

Non deterministically skip over locked rows

MySQL 8.0 - NEW FUNCTIONS

- REGEXP
  - REGEXP_INSTR , REGEXP_LIKE, REGEXP_REPLACE, REGEXP_SUBSTR
- UUID
  - UUID_TO_BIN, BIN_TO_UUID, IS_UUID
- STATEMENT_DIGEST
  - STATEMENT_DIGEST , STATEMENT_DIGEST_TEXT
- Bit operations are now allowed on all binary data types
  - BINARY, VARBINARY, BLOB, TINYBLOB, MEDIUMBLOB and LONGBLOB

MySQL 8 - JSON Functions

JSON_ARRAY_APPEND()
JSON_ARRAY_INSERT()
JSON_ARRAY()
JSON_CONTAINS_PATH()
JSON_CONTAINS()
JSON_DEPTH()
JSON_EXTRACT()
JSON_INSERT()
JSON_KEYS()
JSON_LENGTH()

JSON_MERGE[_PRESERVE]()
JSON_OBJECT()
JSON_QUOTE()
JSON_REMOVE()
JSON_REPLACE()
JSON_SEARCH()
JSON_SET()
JSON_TYPE()
JSON_UNQUOTE()
JSON_VALID()

JSON_PRETTY()
JSON_STORAGE_SIZE()
JSON_STORAGE_FREE()
JSON_ARRAYAGG()
JSON_OBJECTAGG()
JSON_MERGE_PATCH()
JSON_TABLE()
JSON_OVERLAPS()
JSON Schema
JSON Array Indexes

MySQL 8 - JSON_TABLE()
From JSON Document to SQL Table

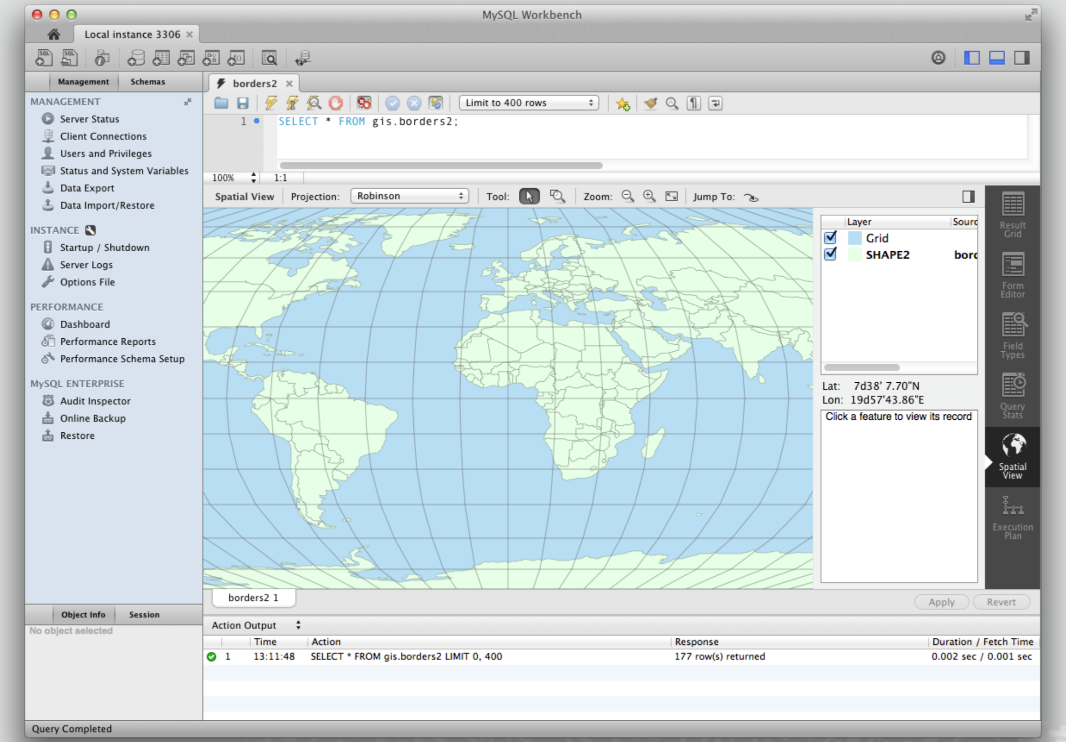
```
mysql> SELECT emps.* FROM JSON_TABLE(@jsonempl, "$[*]" COLUMNS
(id INT PATH "$.id", name VARCHAR(45) PATH "$.name", age INT
PATH "$.age")) emps;
+------+------+------+
| id   | name | age  |
+------+------+------+
| 1    | John | 34   |
| 2    | Mary | 40   |
| 3    | Mike | 44   |
+------+------+------+
3 rows in set (0,00 sec)
```

MySQL 8 - Full Geography Support

Full Geography Support

- Longitude, Latitude
- Projected – Flat/Across 2 dimensions
- Geographic – Spheroid
- 5107 predefined SRSs from the EPSG Dataset 9.2
- 4628 projected, 479 geographic

Built in and ready to use



```sql
SELECT * FROM gis.borders2;
```

Projection: Robinson

Lat: 7d38' 7.70"N
Lon: 19d57'43.86"E

Action Output

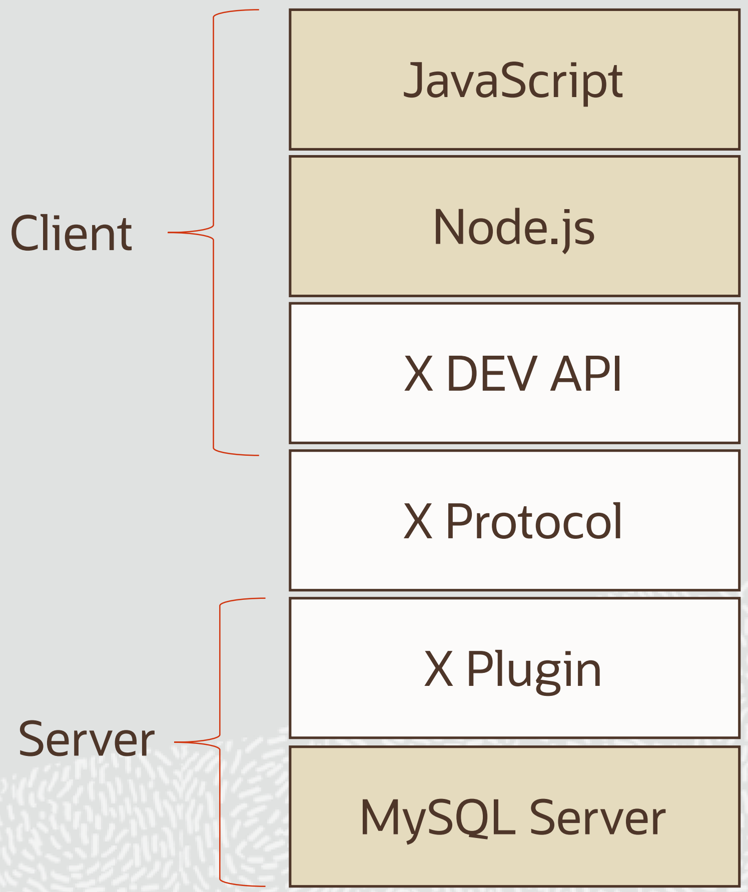
		Time	Action	Response	Duration / Fetch Time
✓	1	13:11:48	SELECT * FROM gis.borders2 LIMIT 0, 400	177 row(s) returned	0.002 sec / 0.001 sec

MySQL Document Store

SQL + NoSQL = MySQL

MySQL 8 – DOCUMENT STORE

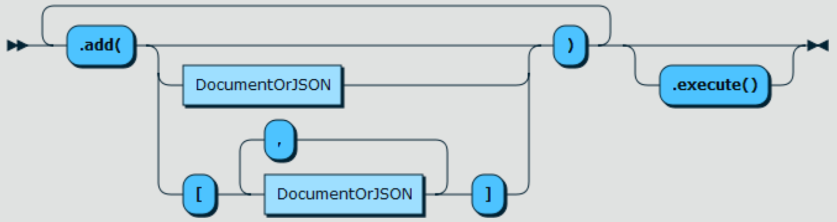
- JSON documents
- Schemaless
- Document collections
  - db.getCollections()
- CRUD operations
  - add(), find(), modify(), remove()
- Connectors (X DevAPI)
- Asynchronous protocol (X Protocol)
- Server front end (X Plugin)

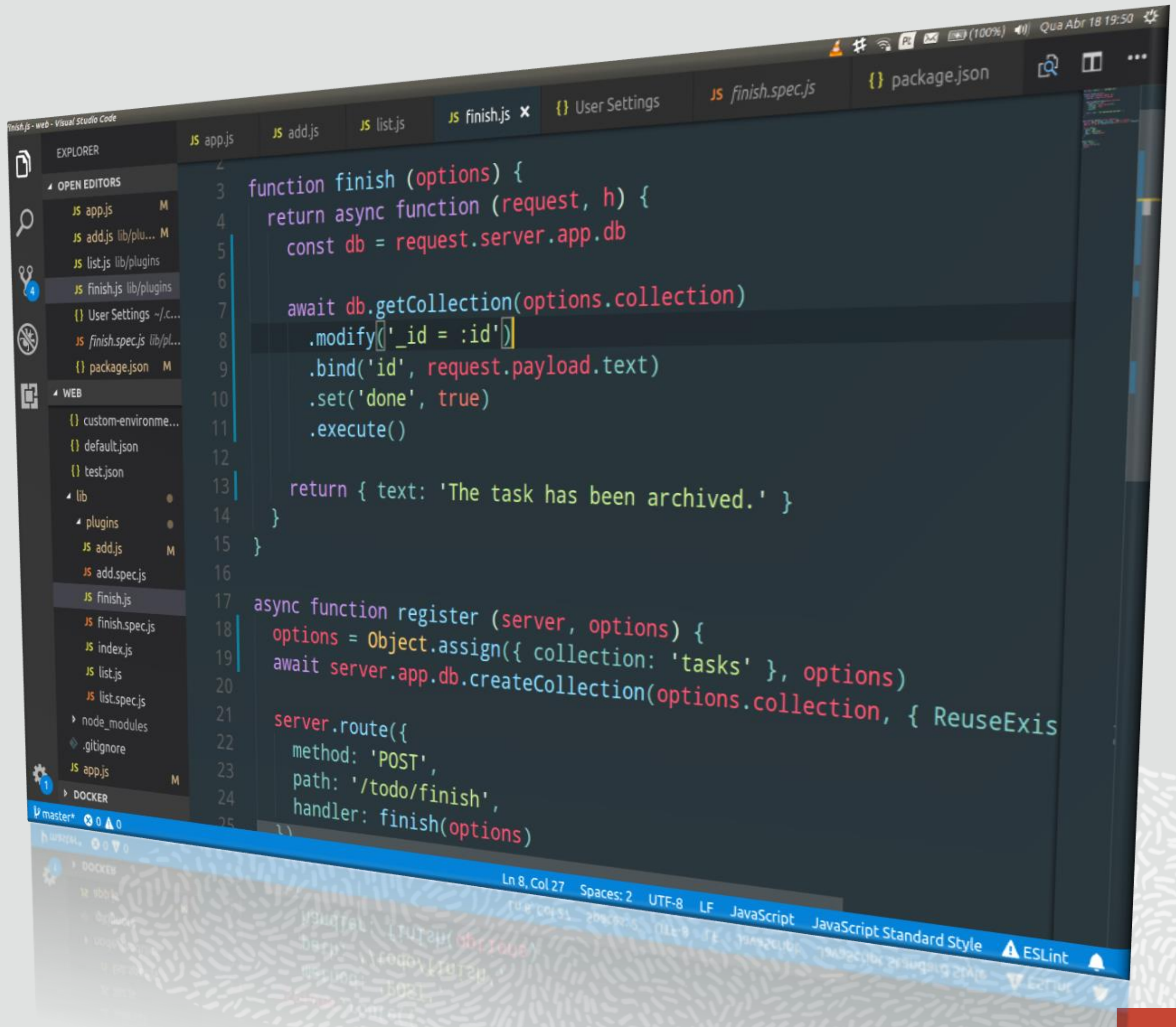


MySQL 8 – DOCUMENT STORE

- Full Node.js integration
  - Support for “Promises”
- Autocompletion support in IDEs
  - Due to method chaining support
- Intuitive Documentation & Tutorials
  - Example:

COLLECTION.add Function





```
function finish (options) {
  return async function (request, h) {
    const db = request.server.app.db

    await db.getCollection(options.collection)
      .modify('_id = :id')
      .bind('id', request.payload.text)
      .set('done', true)
      .execute()

    return { text: 'The task has been archived.' }
  }
}

async function register (server, options) {
  options = Object.assign({ collection: 'tasks' }, options)
  await server.app.db.createCollection(options.collection, { ReuseExis

  server.route({
    method: 'POST',
    path: '/todo/finish',
    handler: finish(options)
```

Operations

Secure, Monitor, Manage, and Upgrade

MySQL 8.0 - SECURE BY DEFAULT

- User should not have to “opt in” for security
- Minimize attack surface
- Minimize process permissions
- Minimize file permissions
- Minimize privileges
- Strong authentication
- Strong encryption methods

MySQL 8.0 - AUTHENTICATION

- Strong default authentication
  - caching_sha2_password

- Pluggable authentication
  - Client and server side
  - Support for intergation with external authentication systems

- Can use the OS login to authenticate
  - unix sockets

MYSQL 8.0 - PASSWORD MANAGEMENT

- Password rotation policies and enforcement
- Password history and reuse protection
- Password strength evaluation and enforcement
- Password generation
- Two passwords per user
- Brute-force attack protection

MySQL 8.0 - AUTHORIZATION

- Standards compliant user and roles management
  - Users, Roles, and Privileges
- SQL Standard Information Schema views for SQL Roles
  - APPLICABLE_ROLES, ENABLED_ROLES, ROLE_TABLE_GRANTS , …
- Fine grained permissions management
  - Admin Privileges

MYSQL 8.0 – FOCUS ON OpenSSL

- OpenSSL is linked dynamically
- OpenSSL versions can be patched at OS level
- Support for FIPs compliance
- Can reconfigure certificates without restarting the server
- Encrypt over the wire, TLS 1.3 support
- Encrypt data at rest

MYSQL 8.0 – MONITORING

- Information Schema tables
  - Persistent meta data
  - Views on the data dictionary tables
  - Data dictionary is implemented as system tables in InnoDB
- Performance Schema tables
  - Volatile meta data, lost upon restart
- SYS Schema
  - Stored routines
  - Task oriented reports

MySQL 8.0 - PERFORMANCE SCHEMA

- Statement latency statistics
  - What is the latency distribution for a given SQL statement?
- Data Locks
  - Which user threads are waiting for which locks? Who holds them?
- SQL Errors
  - Which errors have been sent back to clients? When? How often?
- Configuration Variables
  - What is the current value? Who set it? When?

MYSQL 8.0 – MANAGEMENT

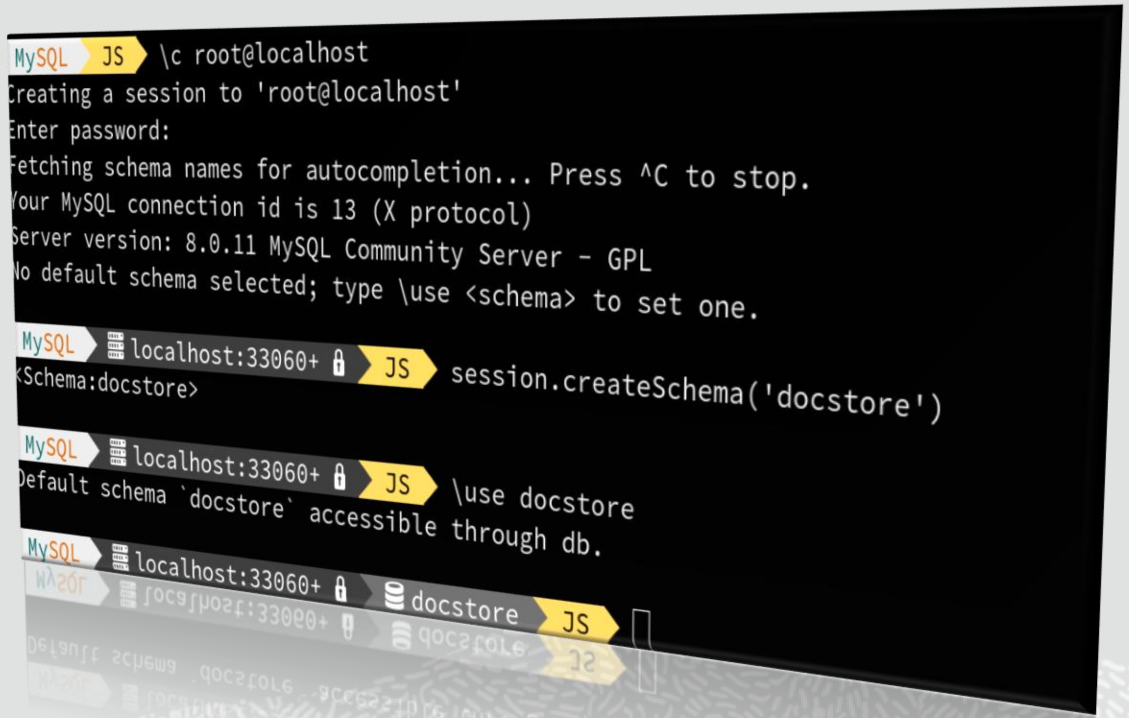
- Manage over a user connection or use the MySQL Shell
  - Eliminate the need to access the host machine
  - Eliminate the need to restart the server
- Configuration changes by SQL DDL
  - SET PERSIST (mostly online)
  - RESTART (still required in some cases)
- Auto Upgrade
  - The system reads the «on disk» version
  - Execute upgrade code if needed

MySQL Shell

Ease of Use
- To a new level

Meet Ada

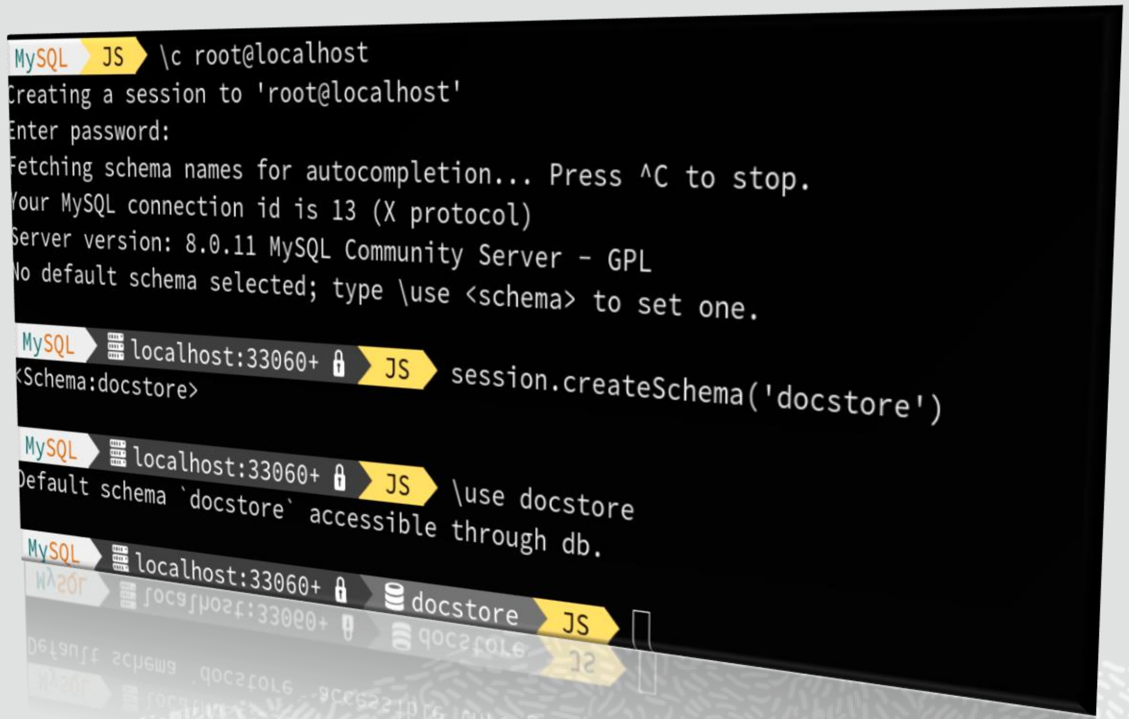
- Meet Ada, the DevOps
- Ada is smart
- Ada is using the MySQL Shell



```
MySQL  JS  \c root@localhost
Creating a session to 'root@localhost'
Enter password:
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 13 (X protocol)
Server version: 8.0.11 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL  localhost:33060+  JS  session.createSchema('docstore')
<Schema:docstore>
MySQL  localhost:33060+  JS  \use docstore
Default schema `docstore` accessible through db.
MySQL  localhost:33060+  docstore  JS
```

MySQL Shell : Modern

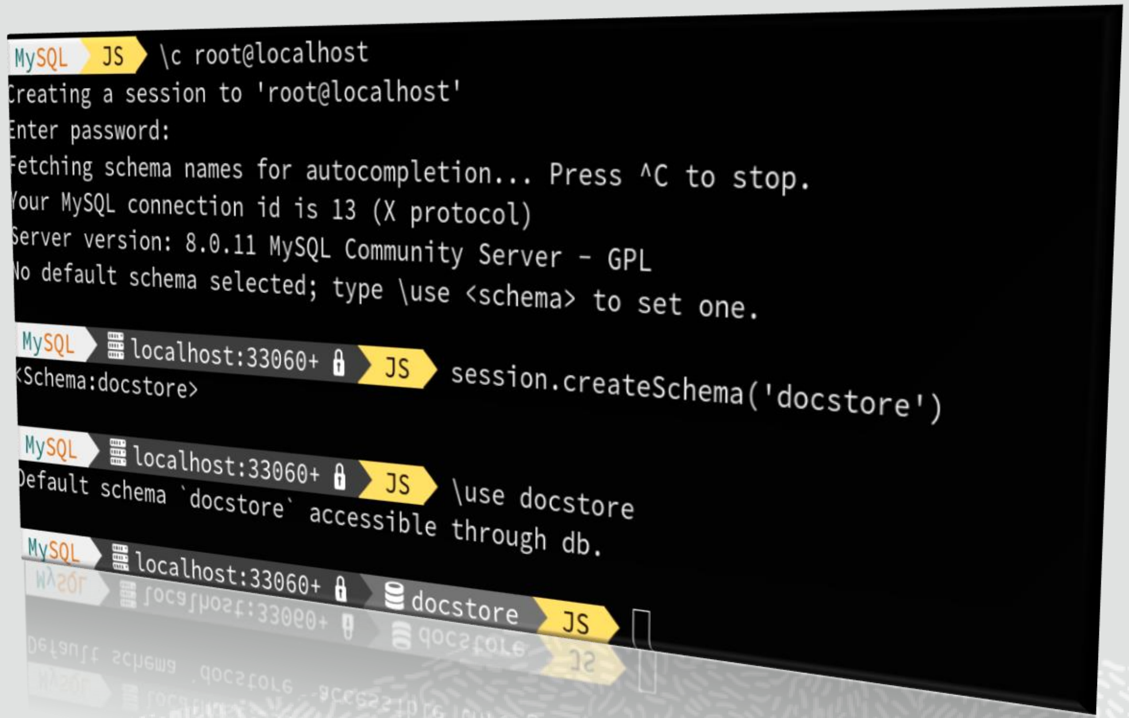
- Colorful Prompt Themes
- Autocompletion
- Syntax Highlighting
- Context Sensitive Help
- Command History
- Pager, less/more
- Output Formats



```
MySQL  JS  \c root@localhost
Creating a session to 'root@localhost'
Enter password:
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 13 (X protocol)
Server version: 8.0.11 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL  localhost:33060+  JS  session.createSchema('docstore')
<Schema:docstore>
MySQL  localhost:33060+  JS  \use docstore
Default schema `docstore` accessible through db.
MySQL  localhost:33060+  docstore  JS
```

MySQL Shell : Flexible

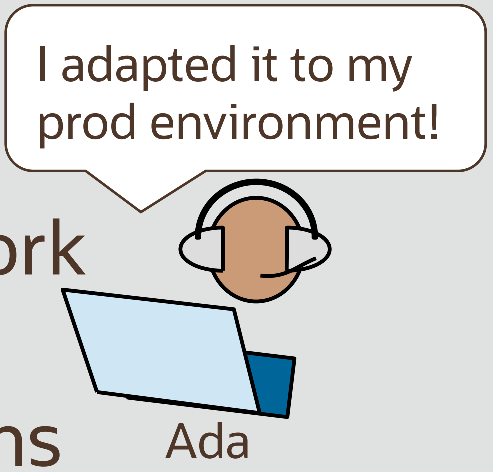
- SQL, JavaScript, Python
- Interactive & Batch
- SQL Client
- Document Store
- InnoDB Cluster Admin
- InnoDB ReplicaSet Admin

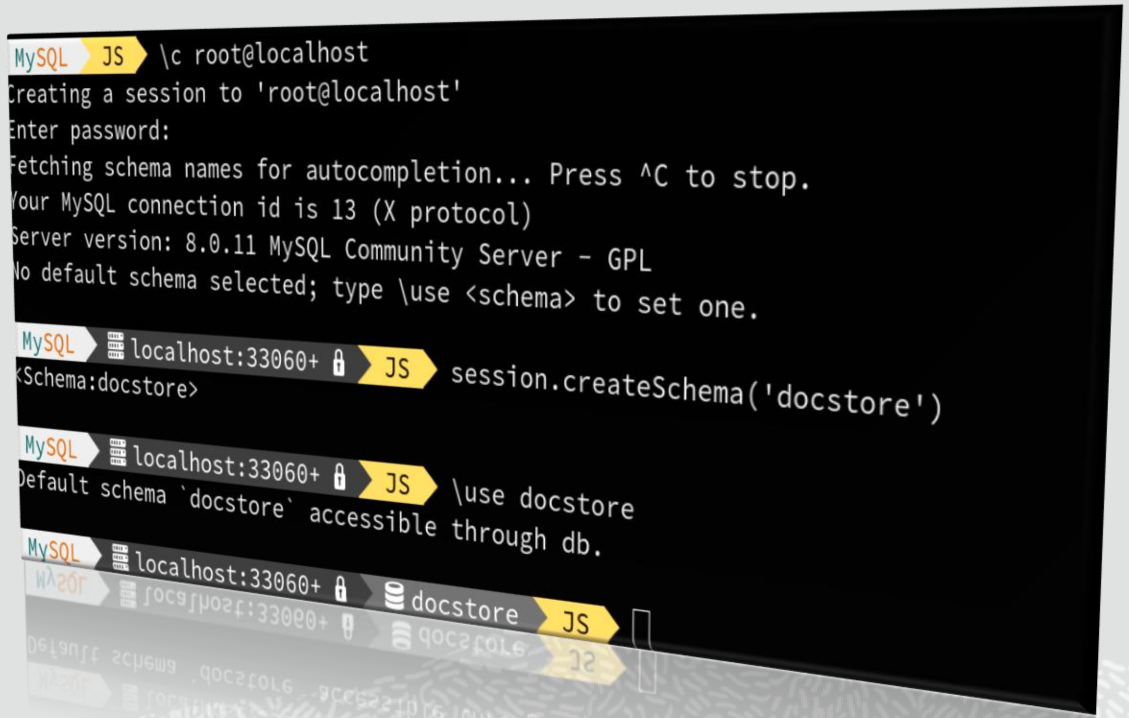


```
MySQL  JS  \c root@localhost
Creating a session to 'root@localhost'
Enter password:
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 13 (X protocol)
Server version: 8.0.11 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL  localhost:33060+  JS  session.createSchema('docstore')
<Schema:docstore>

MySQL  localhost:33060+  JS  \use docstore
Default schema `docstore` accessible through db.
MySQL  localhost:33060+  docstore  JS
```

MySQL Shell : Extendible

- Utilities
  - upgradeChecker()
  - importJSON()
  - importTable()
- Reporting Framework
  - \show \watch
- User Defined Plugins
  - JS or Python





```
MySQL  JS  \c root@localhost
Creating a session to 'root@localhost'
Enter password:
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 13 (X protocol)
Server version: 8.0.11 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL  localhost:33060+  JS  session.createSchema('docstore')
<Schema:docstore>
MySQL  localhost:33060+  JS  \use docstore
Default schema `docstore` accessible through db.
MySQL  localhost:33060+  docstore  JS
```

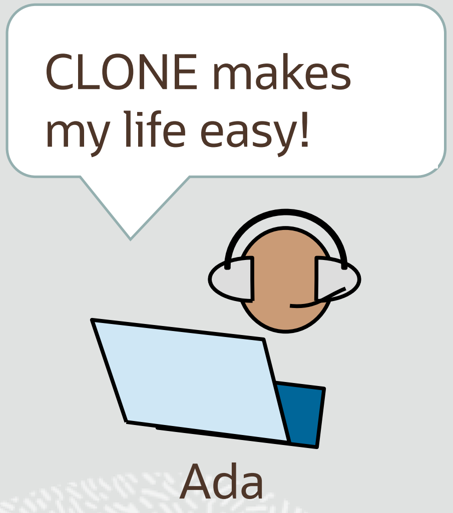
MySQL CLONE

Fast instance provisioning

MySQL 8 – CLONE

WHY IS CLONE SUCH A BIG DEAL?

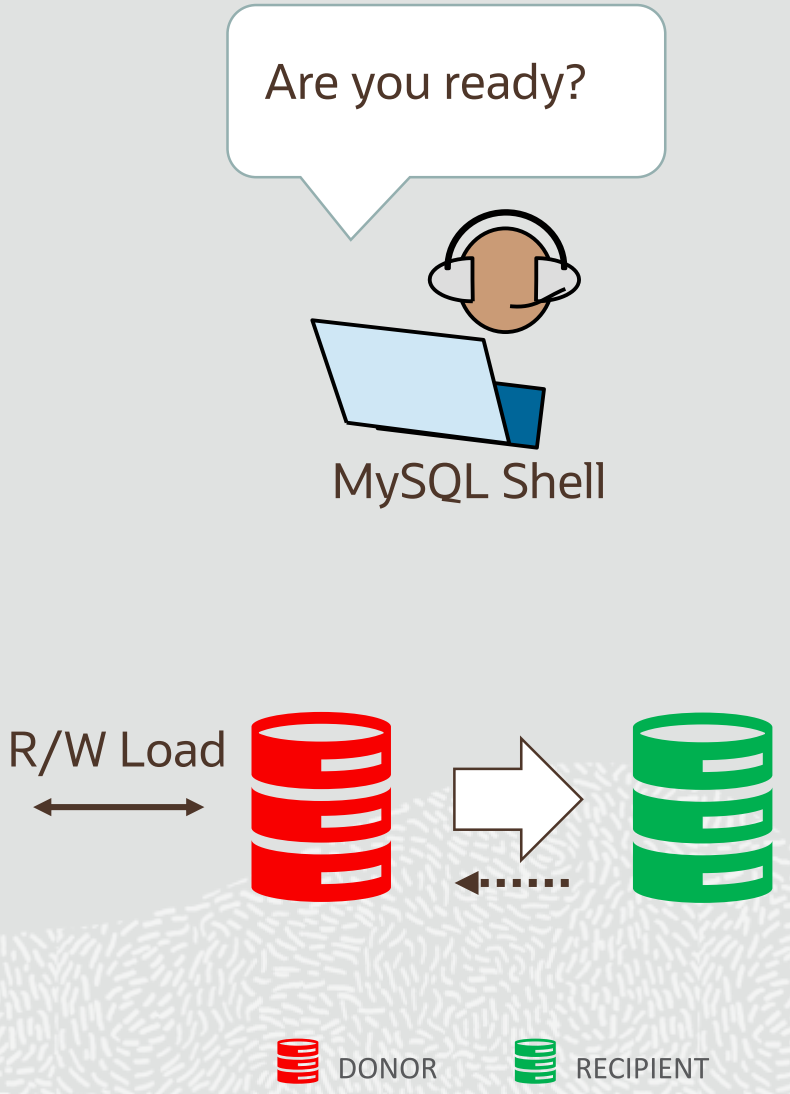
- Puts the power of **fast instance provisioning** into the hands of everybody
- Reduces the complex provisioning procedure to a few simple steps
- Even happens automatically when needed when using **InnoDB Cluster**
- Can be done fully remotely



MySQL 8 - CLONE Directly from SQL

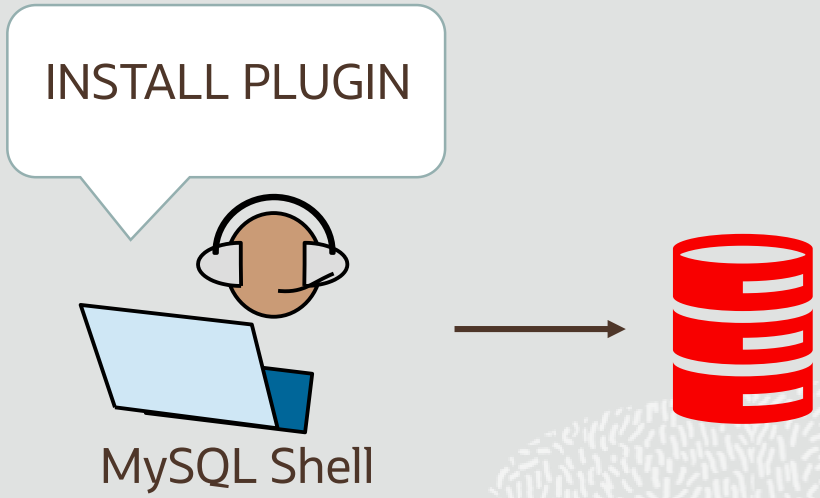
“User traffic is growing and I need a new read replica”

Provision a *new slave* (RECIPIENT) from an *existing master* (DONOR)



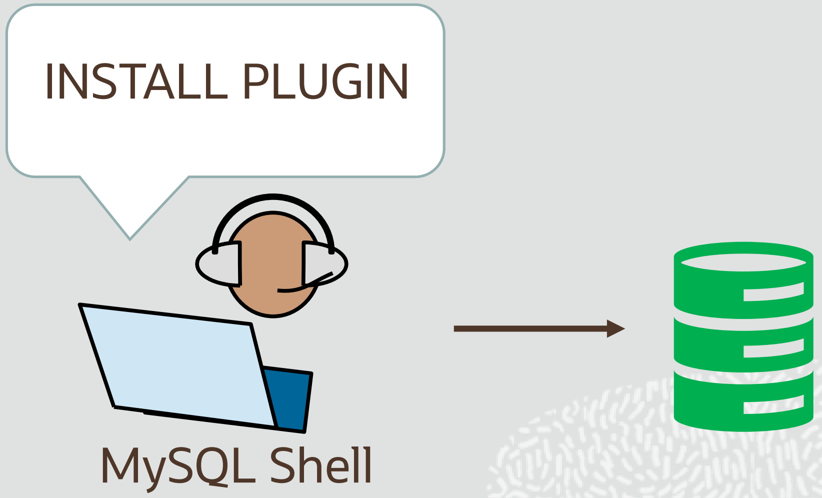
MySQL – 8 CLONE Setup the DONOR

```
mysql> INSTALL PLUGIN CLONE SONAME "mysql_clone.so";
mysql> CREATE USER clone_user IDENTIFIED BY "clone_password";
mysql> GRANT BACKUP_ADMIN ON *.* to clone_user;
```



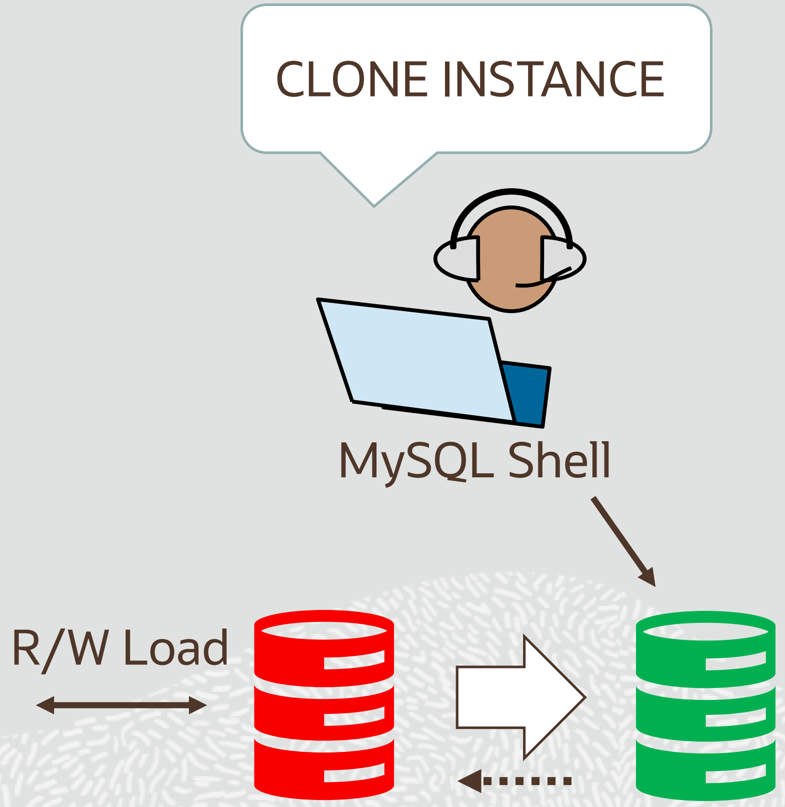
MySQL 8 - CLONE Setup the RECIPIENT

```
mysql> INSTALL PLUGIN CLONE SONAME "mysql_clone.so";
mysql> SET GLOBAL clone_valid_donor_list = "donor.host.com:3306";
mysql> CREATE USER clone_user IDENTIFIED BY "clone_password";
mysql> GRANT BACKUP_ADMIN ON *.* to clone_user;
```



MySQL 8 - Connect to RECIPIENT and execute CLONE SQL statement

```
mysql> CLONE INSTANCE
        -> FROM clone_user@donor.host.com:3306
        -> IDENTIFIED BY "clone_password";
```



MySQL 8 - CLONE Check Status

```
mysql> select STATE, …
     > from performance_schema.clone_status;

+-------------+---------------------+------------+
| STATE       | START TIME          | DURATION   |
+-------------+---------------------+------------+
| In Progress | 2019-07-17 17:23:26 |     4.84 m |
+-------------+---------------------+------------+
```

MySQL 8 - CLONE Check Progress

```
mysql> select STATE, …
     > from performance_schema.clone_progress;
```

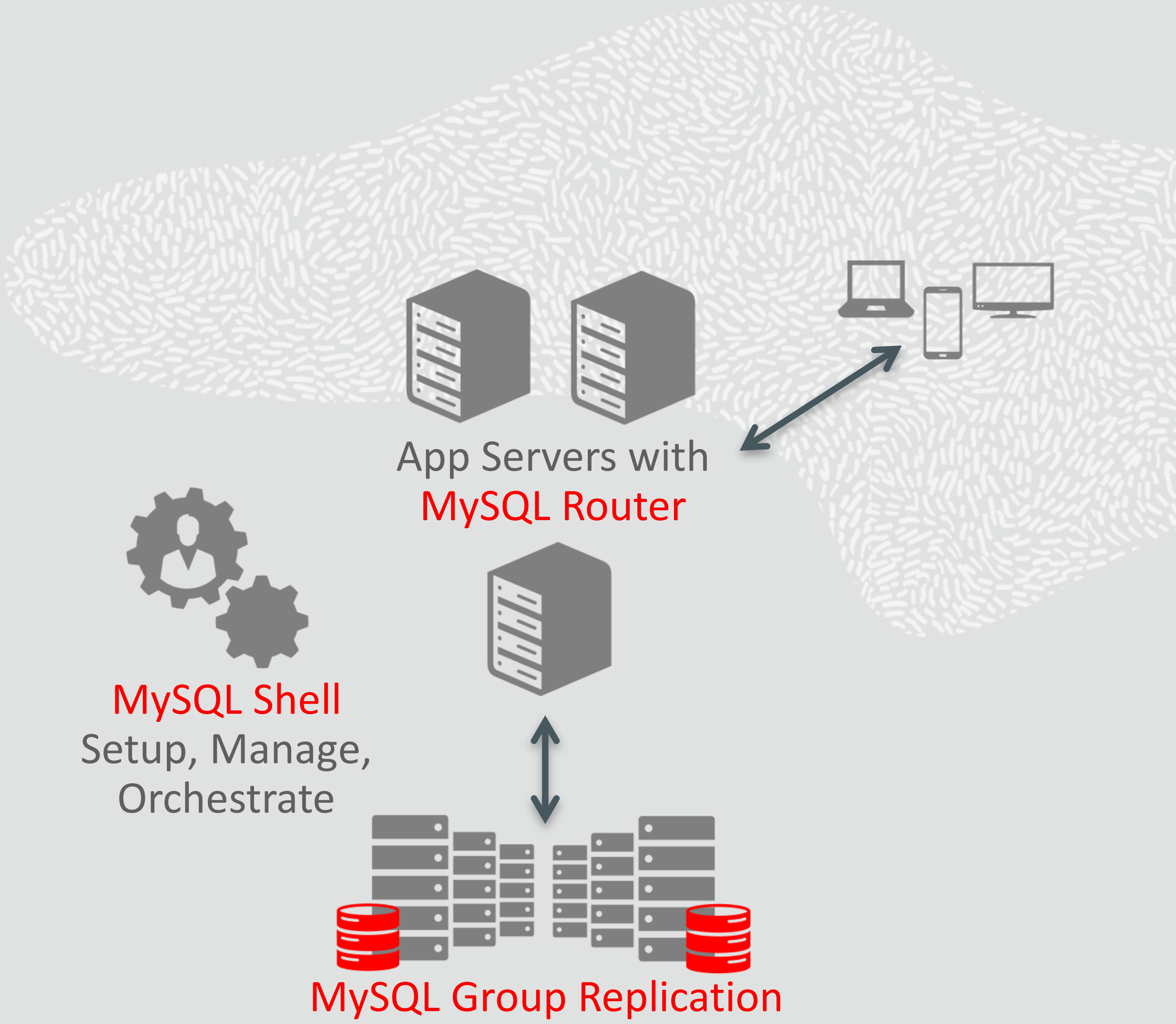
```
+-----------+-------------+------------+----------+------------+---------+
| STAGE     | STATE       | START TIME | DURATION | Estimate   | Done(%) |
+-----------+-------------+------------+----------+------------+---------+
| DROP DATA |   Completed |   17:23:26 | 790.86 ms |      0 MB |    100% |
| FILE COPY |   Completed |   17:23:27 |  10.33 m  | 94,729 MB |    100% |
| PAGE COPY |   Completed |   17:33:47 |  15.91 s  | 11,885 MB |    100% |
| REDO COPY |   Completed |   17:34:03 |   1.07 s  |    293 MB |    100% |
| FILE SYNC | In Progress |   17:34:04 |  51.68 s  |      0 MB |      0% |
| RESTART   | Not Started |       NULL |      NULL |      0 MB |      0% |
| RECOVERY  | Not Started |       NULL |      NULL |      0 MB |      0% |
+-----------+-------------+------------+-----------+-----------+---------+
```

MySQL InnoDB Cluster

High Availability - Out of the Box

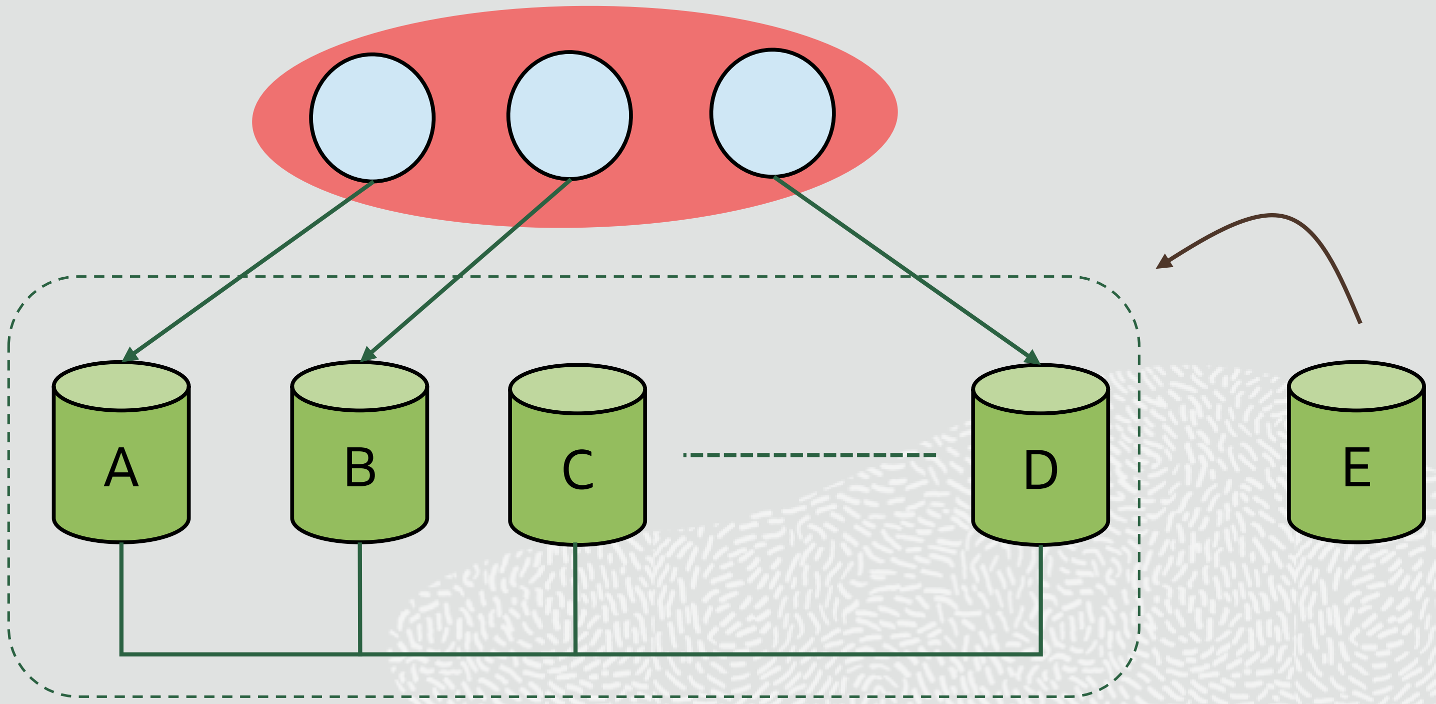
MySQL 8 - MySQL InnoDB Cluster

- MySQL Group Replication
  - High Availability
  - Elastic, Fault Tolerant, Self Healing
- MySQL Router
  - Connection Routing, Load Balancing
- MySQL Shell
  - Easy Setup & Administration



MySQL 8 – GROUP REPLICATION

Initialize group
Detect node failure
Reestablish group
Elect new primary
Recover from failure
Rejoin group
Grow and shrink group
Provision new members
Topology Meta-data
Router Notification
Observability



MySQL 8 – REPLICATION TECHNOLOGY

- Multi-threaded Replication Applier With WRITESETs
  - Faster Slaves - higher end-to-end throughput
- Global Transaction Identifiers (GTIDs)
  - Track replication changes seamlessly across replication chains
- Replicated state machines
  - Coordinate, synchronize and execute distributed operations using well known and proven distributed algorithms such as Paxos

MySQL 8 – DISTRIBUTED AND COORDINATED AUTOMATION

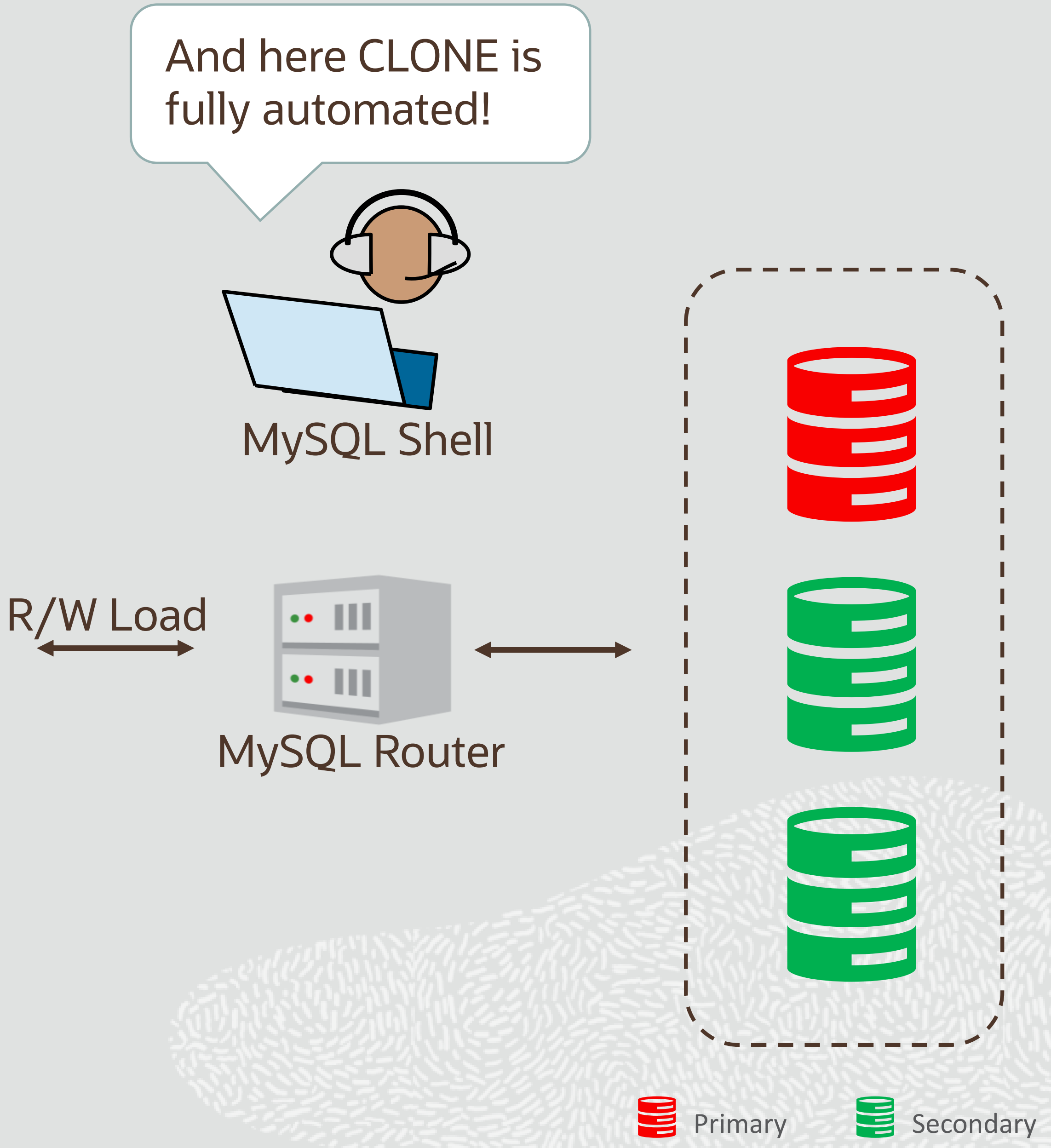
- Fault-detection
  - Automatic detection of failed servers in the cluster
- Server fencing
  - Automatic isolation of faulty servers from the app and the cluster
- Data consistency levels
  - Distributed commit protocol enabling reading your own writes
- Distributed recovery
  - Automatic (re)syncing procedure for servers joining a cluster
- Flow control
  - Automatic server throttling preventing unbounded secondary lag
- Membership services
  - Automatic, dynamic list of servers in the cluster and their status

MySQL InnoDB Cluster

Mini-tutorial

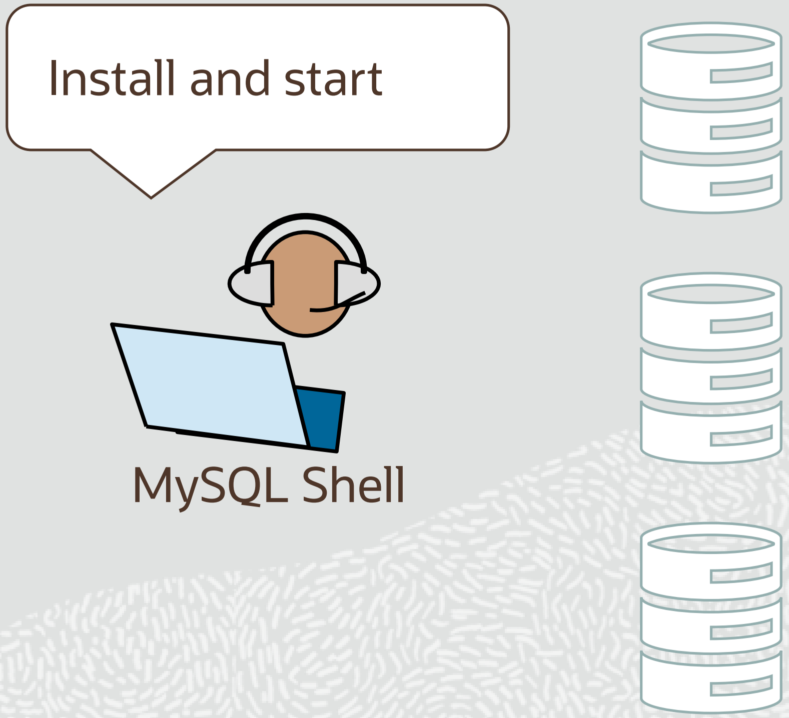
MySQL InnoDB Cluster

- configureInstance()
- createCluster()
- addInstance()
- removeInstance()
- rejoinInstance()



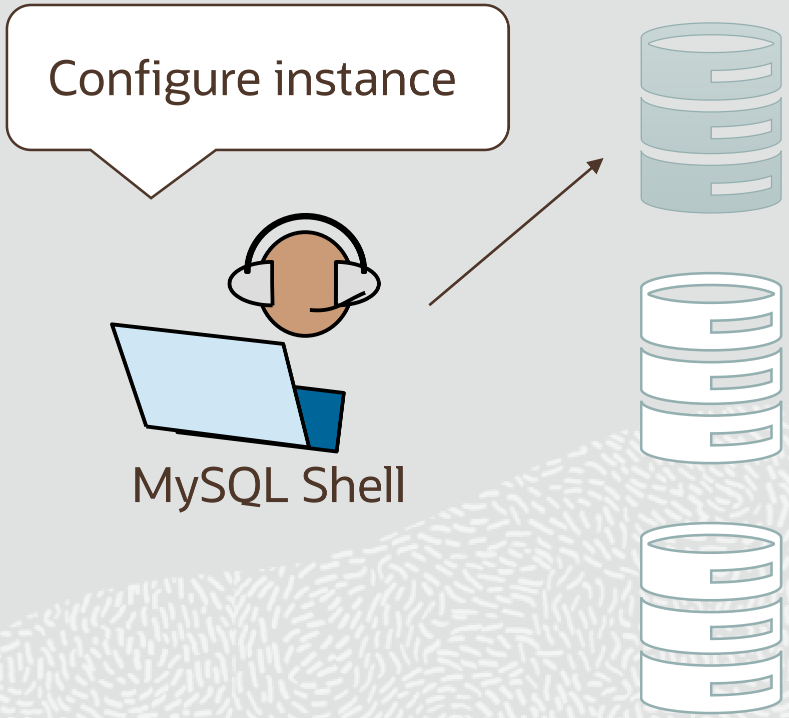
Pre-requisites : Install and start MySQL on 3 servers

Note: mysqld is managed by Linux systemd



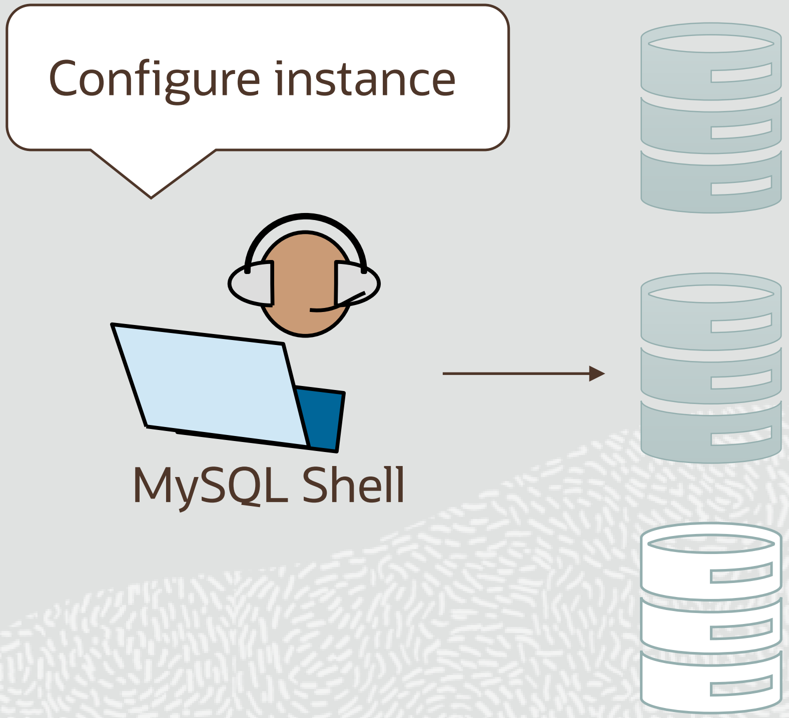
mysql-js>dba.configureInstance('clusteradmin@mysql1')

```
binlog_checksum = NONE
enforce_gtid_consistency = ON
gtid_mode=ON
server_id= <unique ID>
```



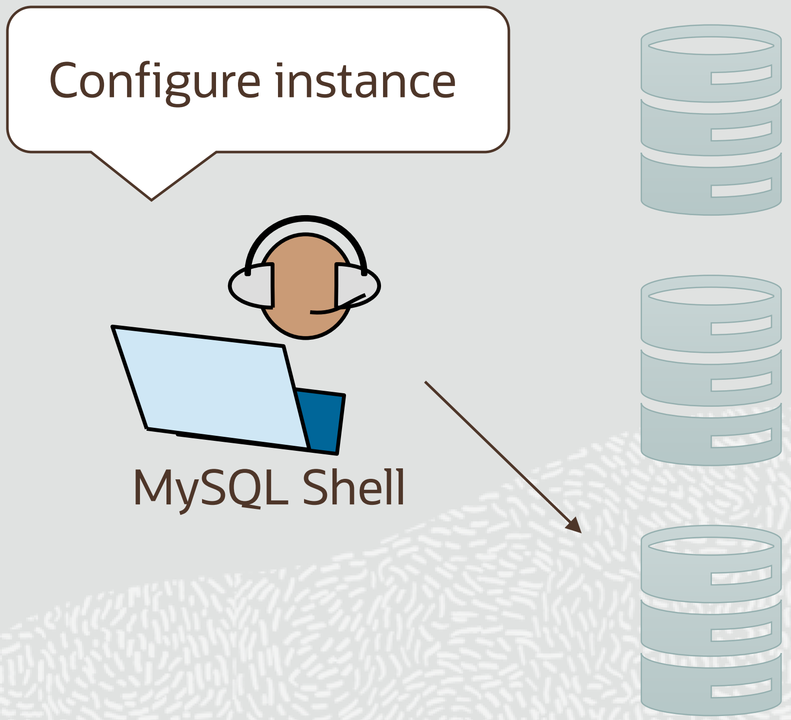
mysql-js>dba.configureInstance('clusteradmin@mysql2')

```
binlog_checksum = NONE
enforce_gtid_consistency = ON
gtid_mode=ON
server_id= <unique ID>
```

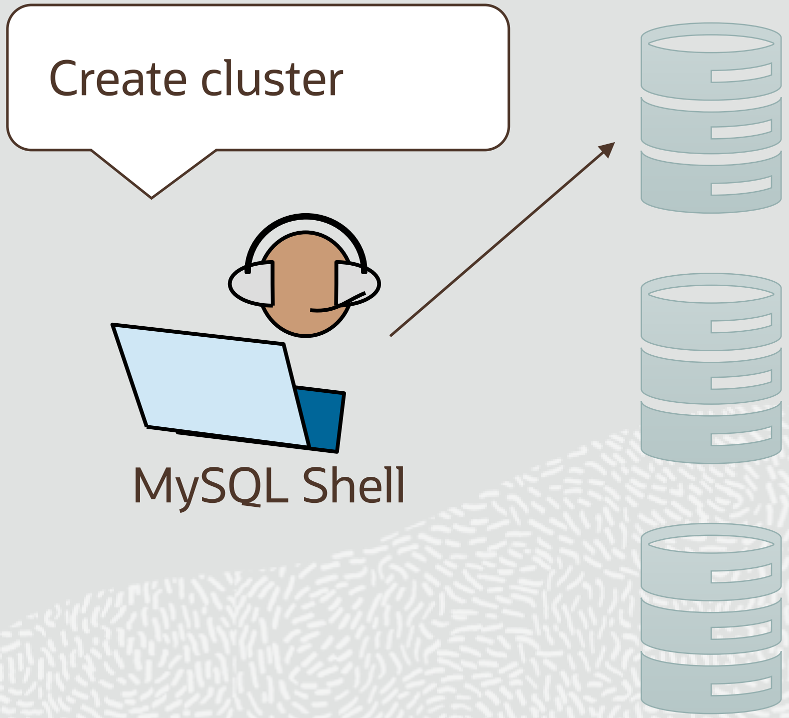


mysql-js>dba.configureInstance('clusteradmin@mysql3')

```
binlog_checksum = NONE
enforce_gtid_consistency = ON
gtid_mode=ON
server_id= <unique ID>
```

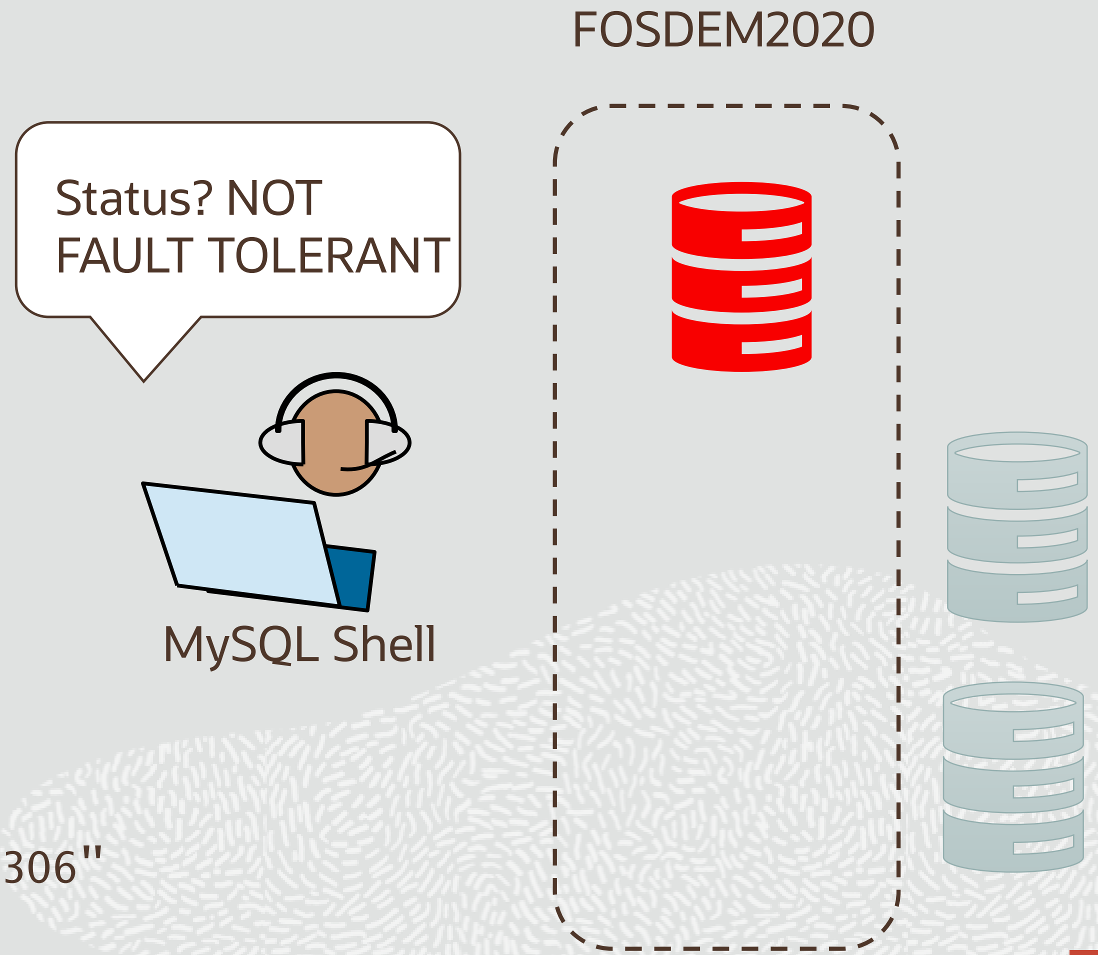


mysql-js> cluster=dba.createCluster(‘FOSDEM2020')

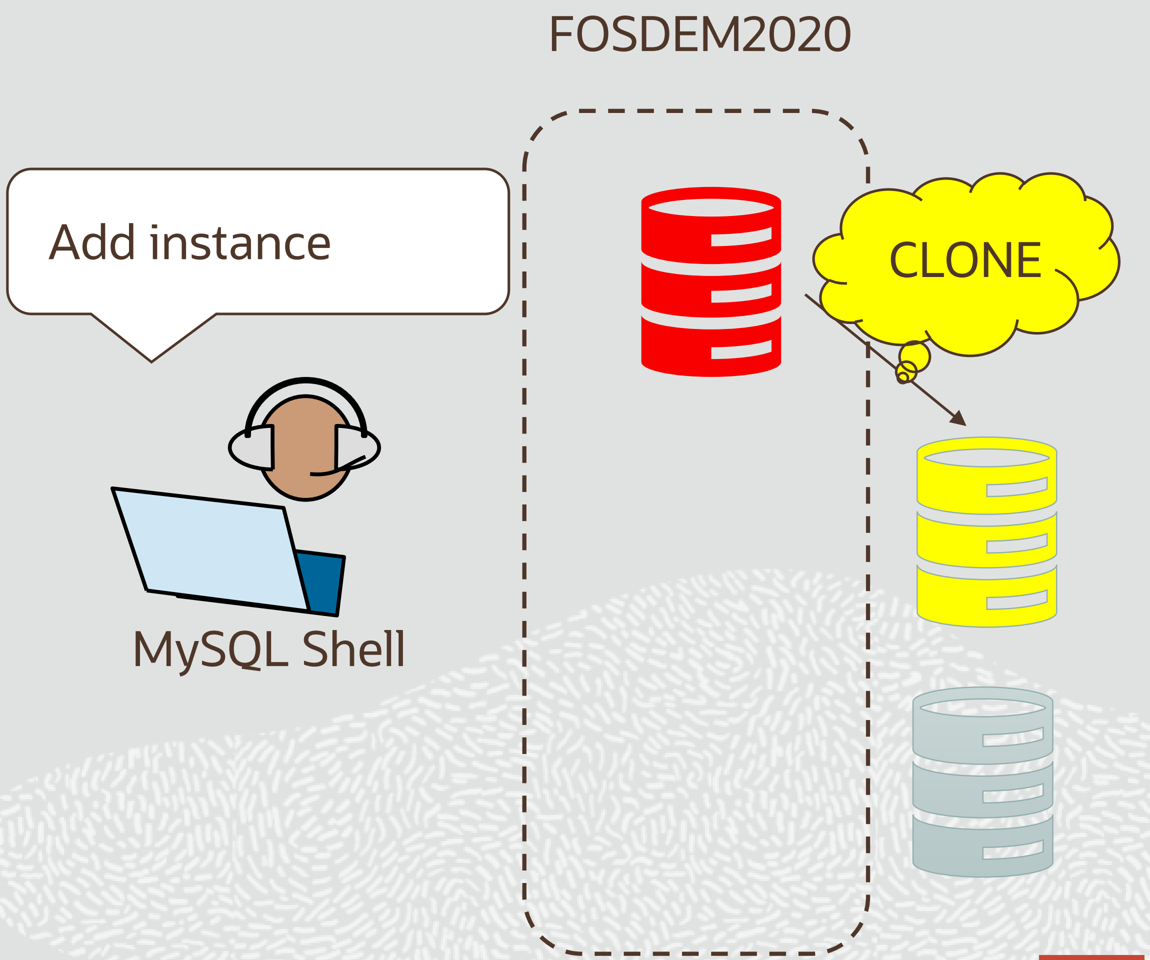


mysql-js> cluster.status()

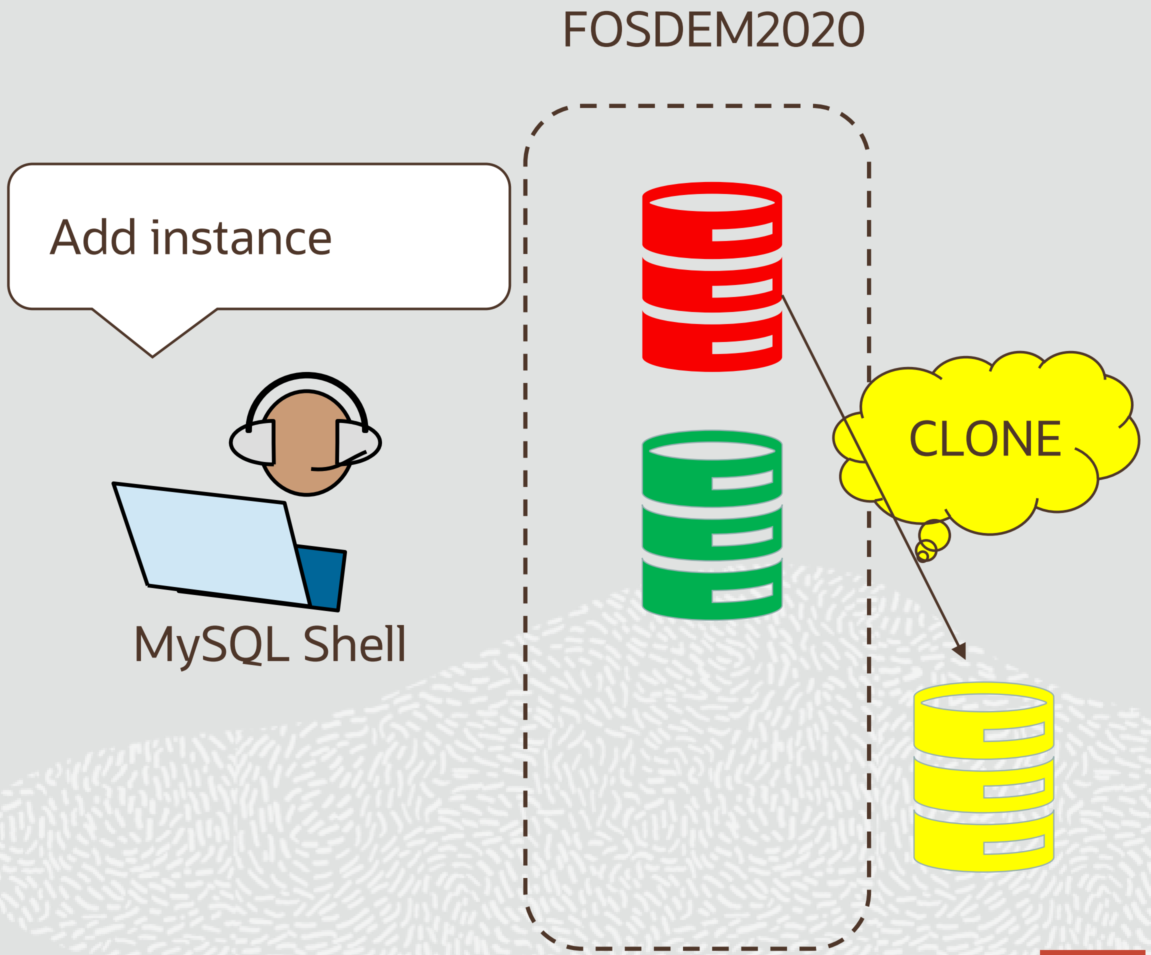
```
mysql-js> cluster.status()
{
    "clusterName": “FOSDEM2020",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mysql1:3306",
        "ssl": "REQUIRED",
        "status": "OK_NOT_TOLERANT",
        "statusText": "Cluster is NOT
         tolerant to any failures.",
        "topology": {
            …
        },
        "topologyMode": "Single Primary"
    },
    "groupInformationSourceMember": "mysql1:3306"
```



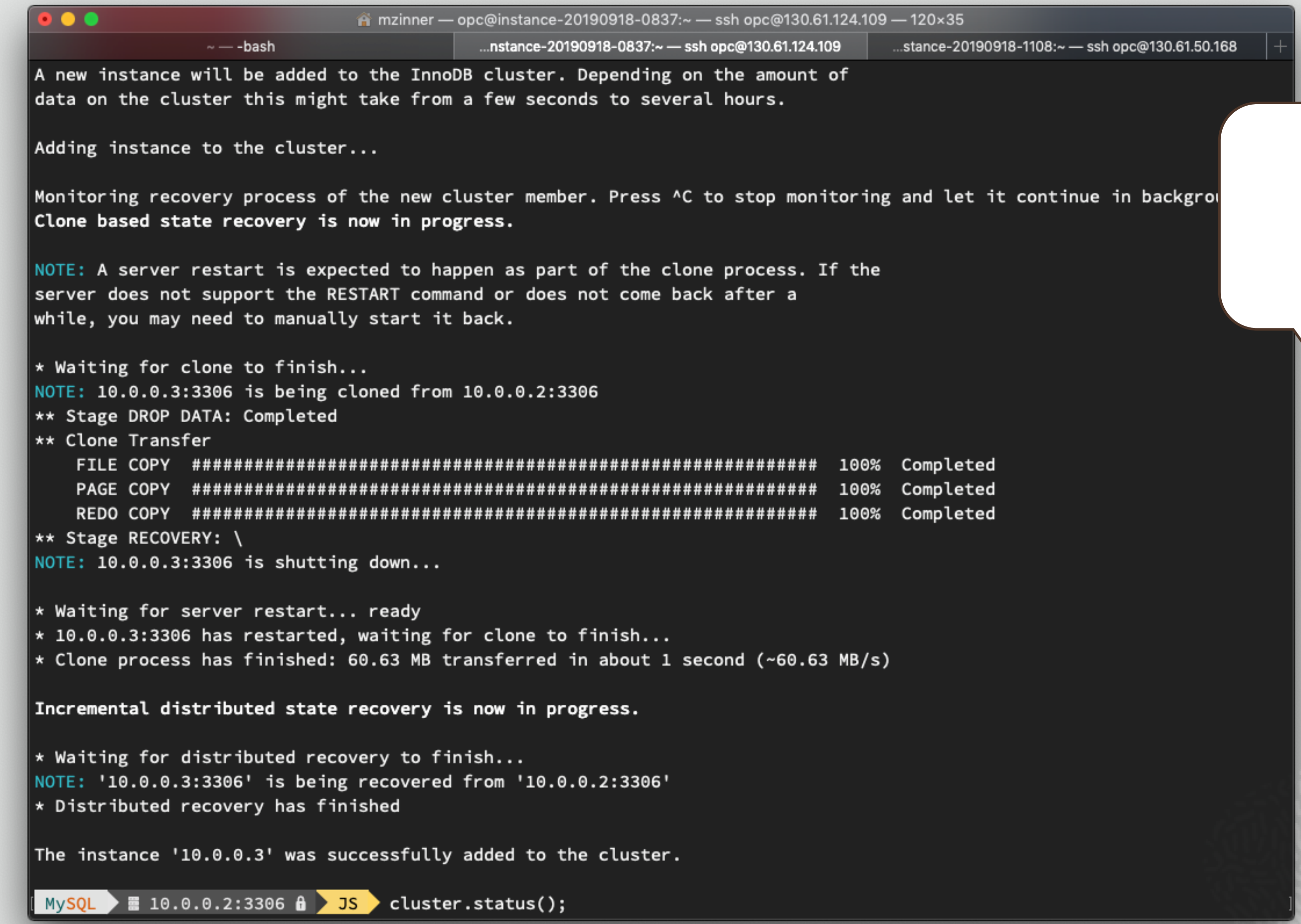
mysql-js> cluster.addInstance('clusteradmin@mysql2')



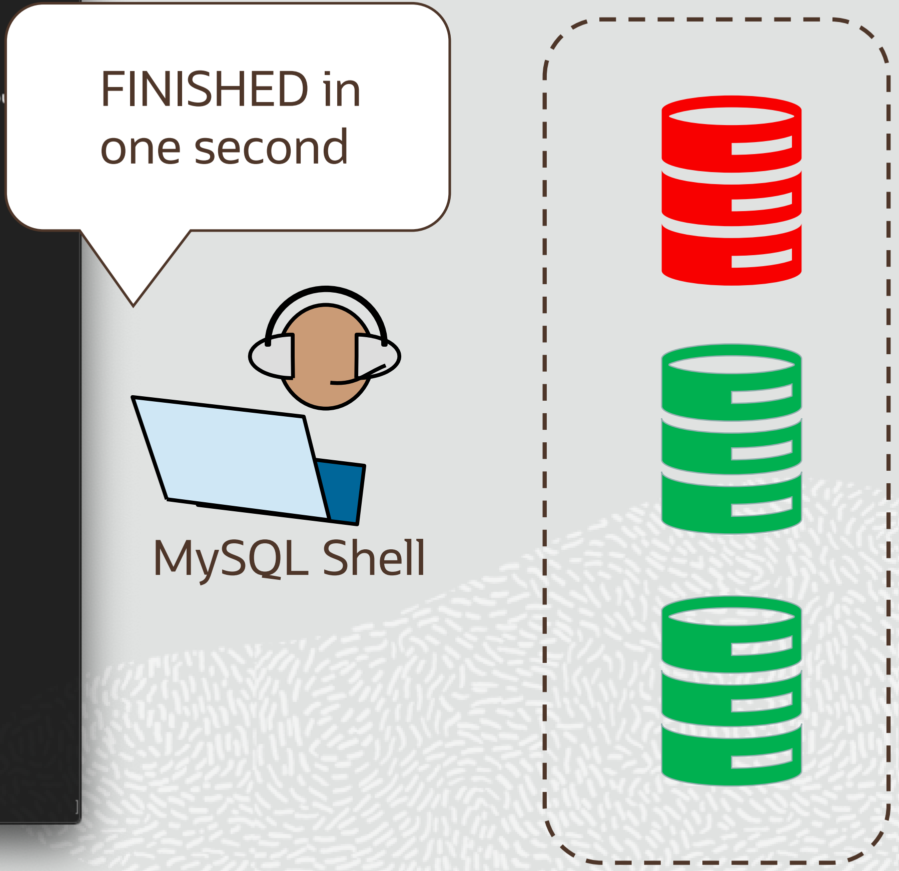
mysql-js> cluster.addInstance('clusteradmin@mysql3')



MySQL Shell : Add Instance with CLONE Progress Reporting

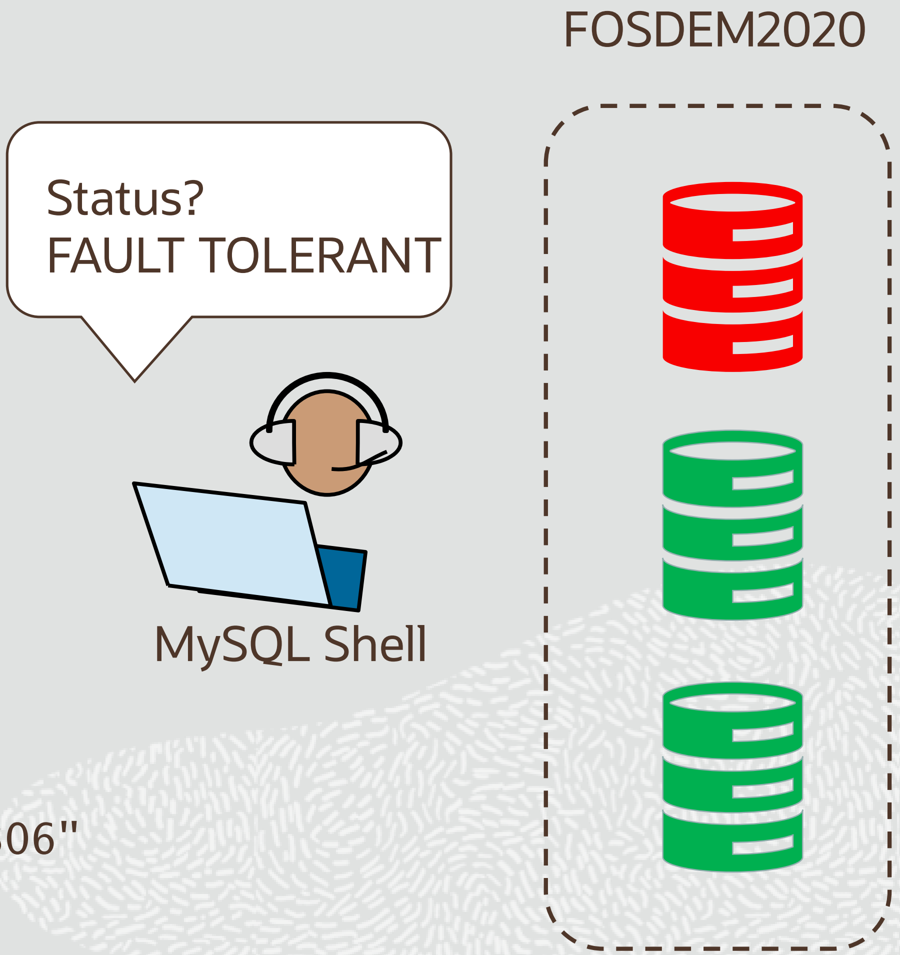

mzinner — opc@instance-20190918-0837:~ — ssh opc@130.61.124.109 — 120×35
~ — -bash
...nstance-20190918-0837:~ — ssh opc@130.61.124.109
...stance-20190918-1108:~ — ssh opc@130.61.50.168

```
A new instance will be added to the InnoDB cluster. Depending on the amount of
data on the cluster this might take from a few seconds to several hours.

Adding instance to the cluster...

Monitoring recovery process of the new cluster member. Press ^C to stop monitoring and let it continue in backgro
Clone based state recovery is now in progress.

NOTE: A server restart is expected to happen as part of the clone process. If the
server does not support the RESTART command or does not come back after a
while, you may need to manually start it back.

* Waiting for clone to finish...
NOTE: 10.0.0.3:3306 is being cloned from 10.0.0.2:3306
** Stage DROP DATA: Completed
** Clone Transfer
    FILE COPY  ############################################################  100%  Completed
    PAGE COPY  ############################################################  100%  Completed
    REDO COPY  ############################################################  100%  Completed
** Stage RECOVERY: \
NOTE: 10.0.0.3:3306 is shutting down...

* Waiting for server restart... ready
* 10.0.0.3:3306 has restarted, waiting for clone to finish...
* Clone process has finished: 60.63 MB transferred in about 1 second (~60.63 MB/s)

Incremental distributed state recovery is now in progress.

* Waiting for distributed recovery to finish...
NOTE: '10.0.0.3:3306' is being recovered from '10.0.0.2:3306'
* Distributed recovery has finished

The instance '10.0.0.3' was successfully added to the cluster.

 MySQL  10.0.0.2:3306  JS  cluster.status();
```

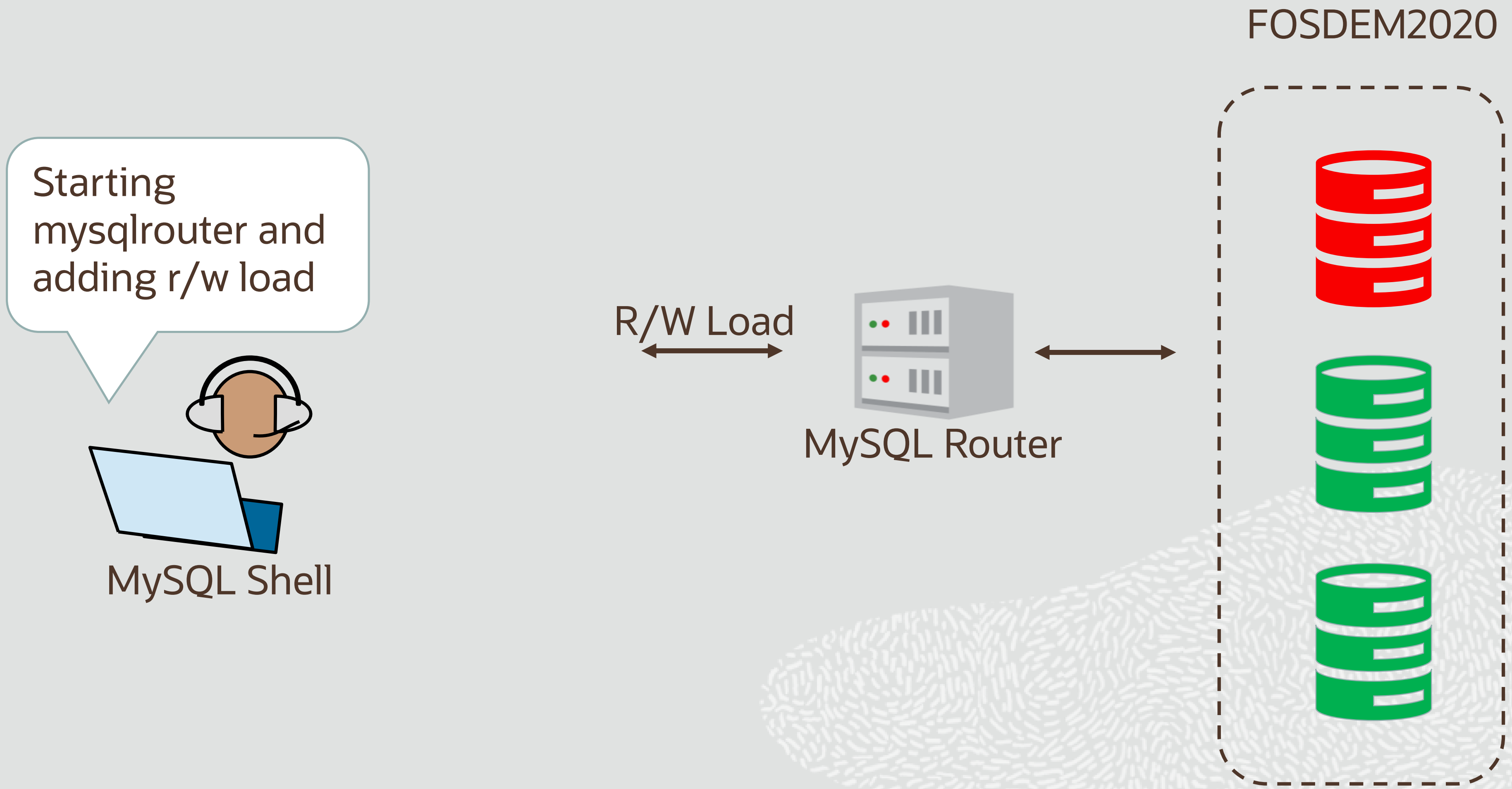


mysql-js> cluster.status()

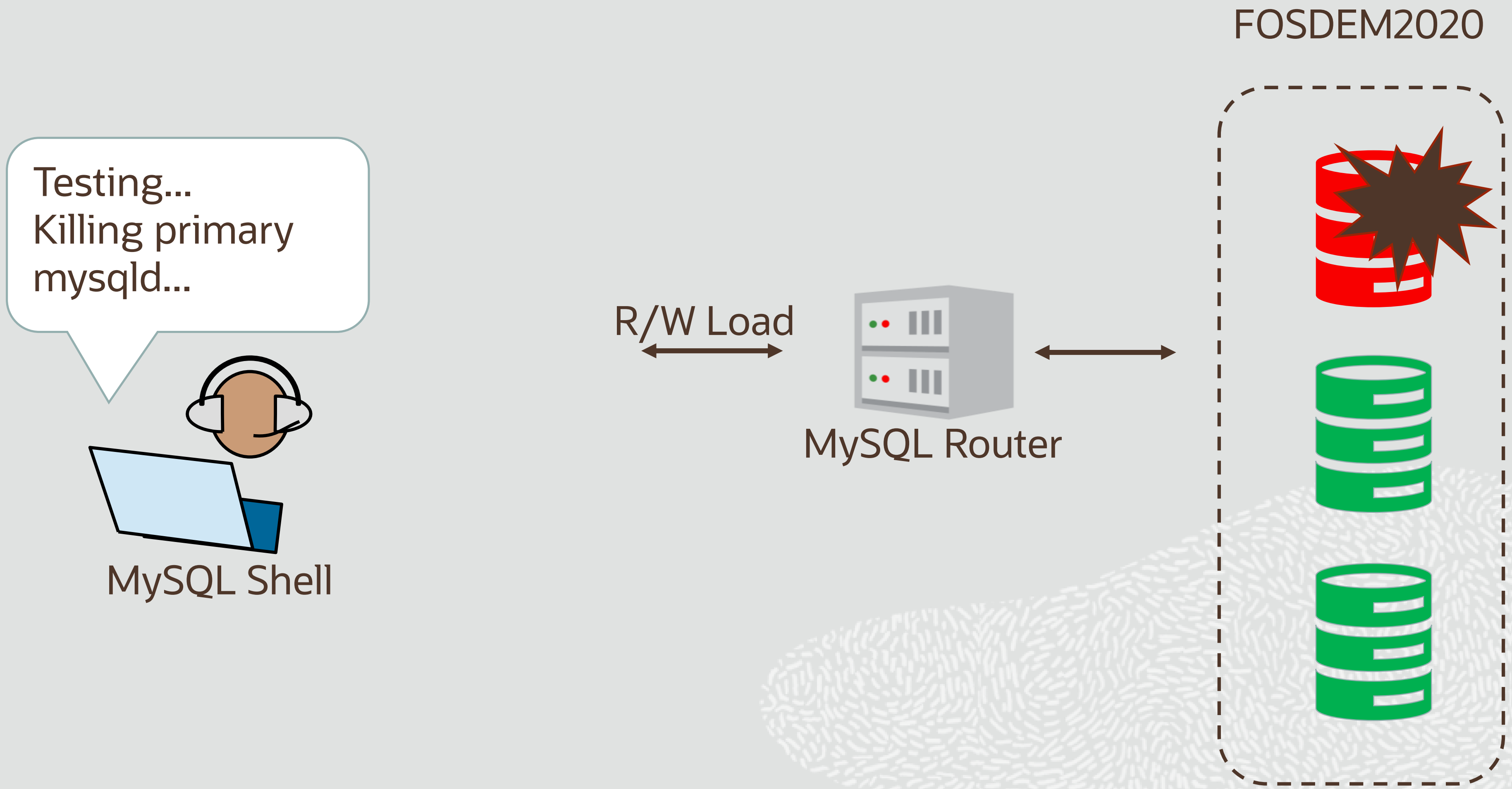
```
mysql-js> cluster.status()
{
    "clusterName": “FOSDEM2020",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mysql1:3306",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE
         and can tolerate up to ONE failure.",
        "topology": {
             ...CUT...
        },
        "topologyMode": "Single Primary"
    },
    "groupInformationSourceMember": "mysql1:3306"
}
```



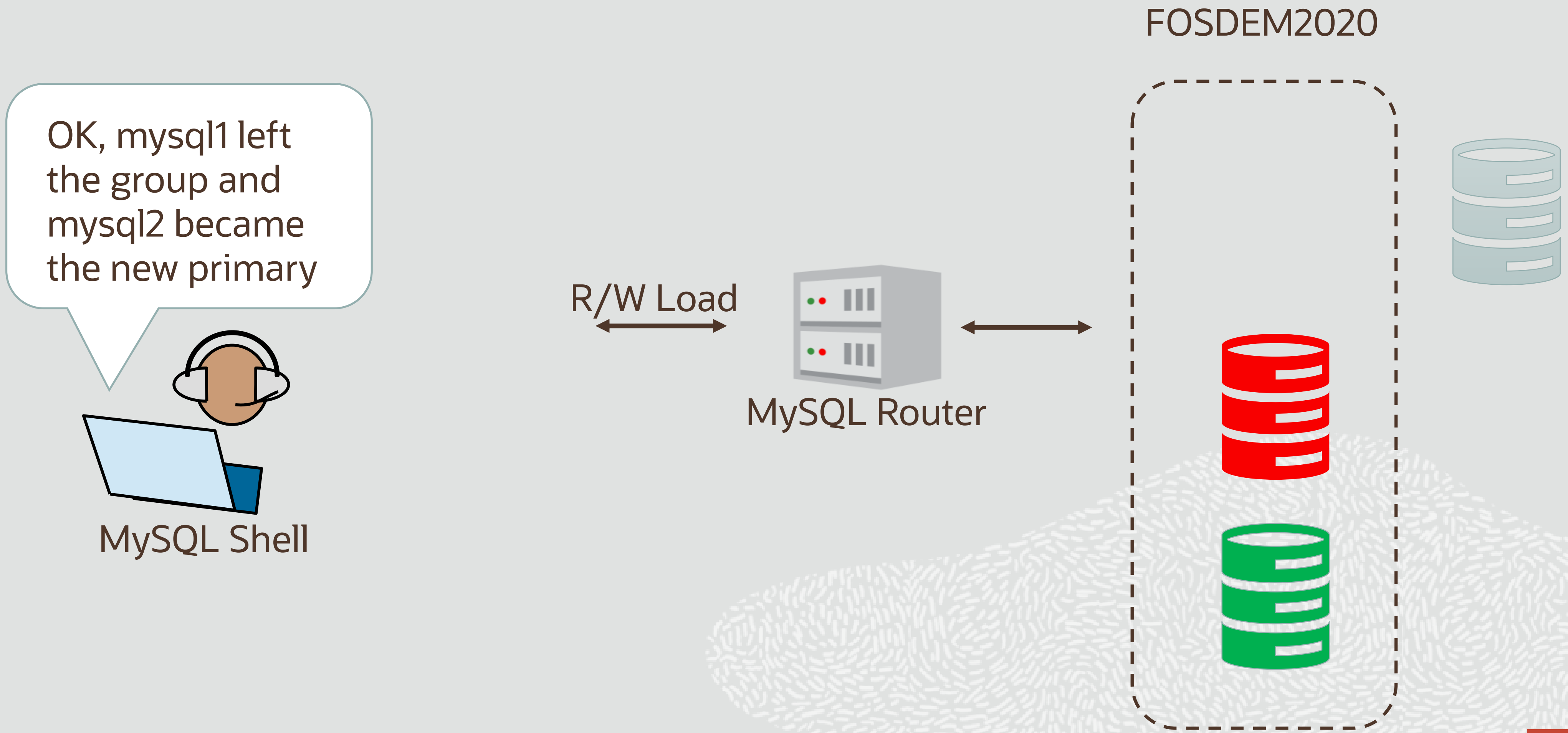
# mysqlrouter --bootstrap clusteradmin@mysql1 --user=routeradmin
# systemctl start mysqlrouter



mysql1# kill -9 $(pidof mysqld)



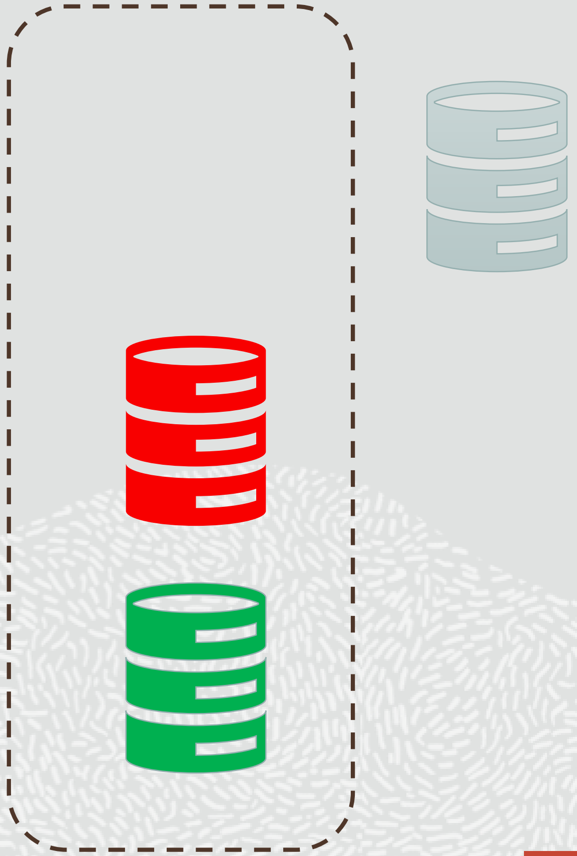
MySQL: New Primary



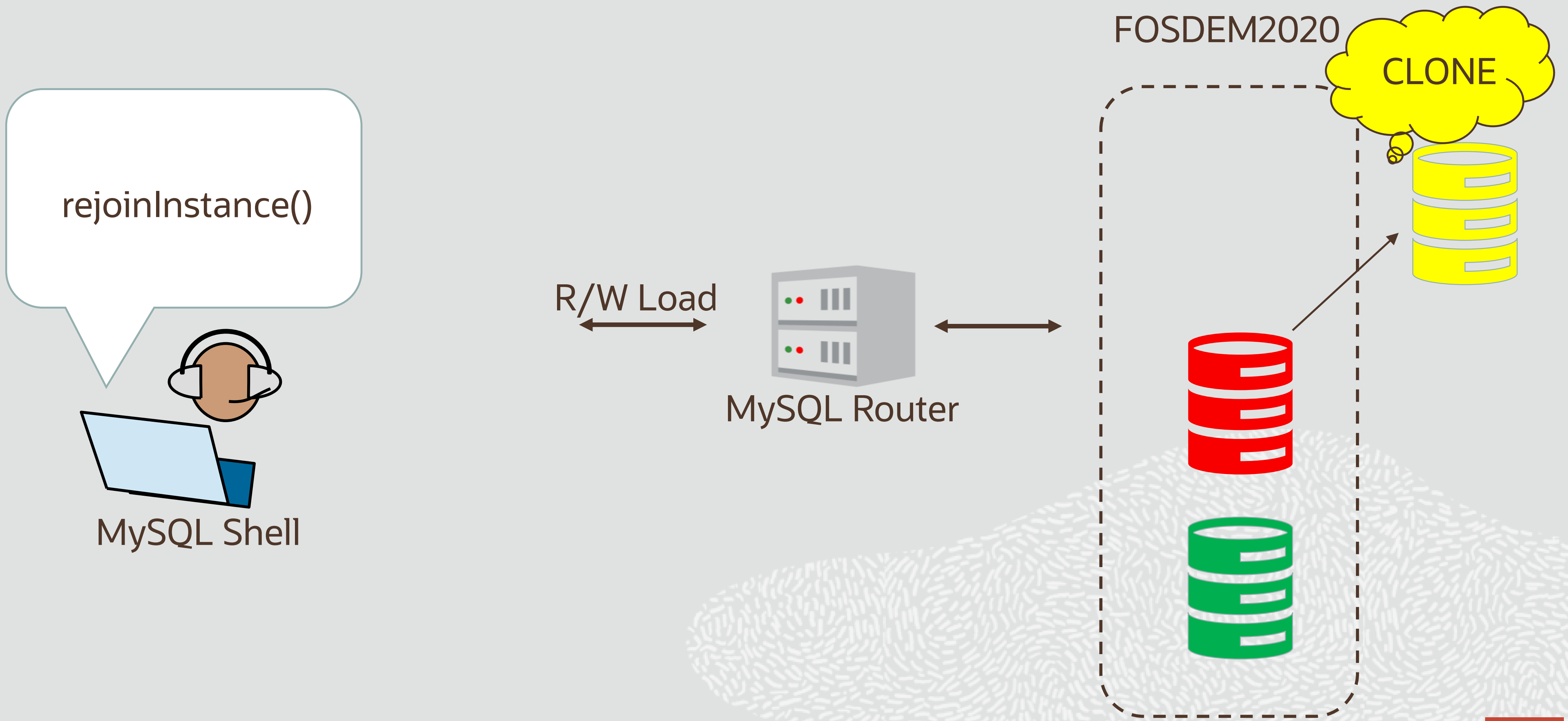
MySQL: How to regain Fault Tolerance?

FOSDEM2020

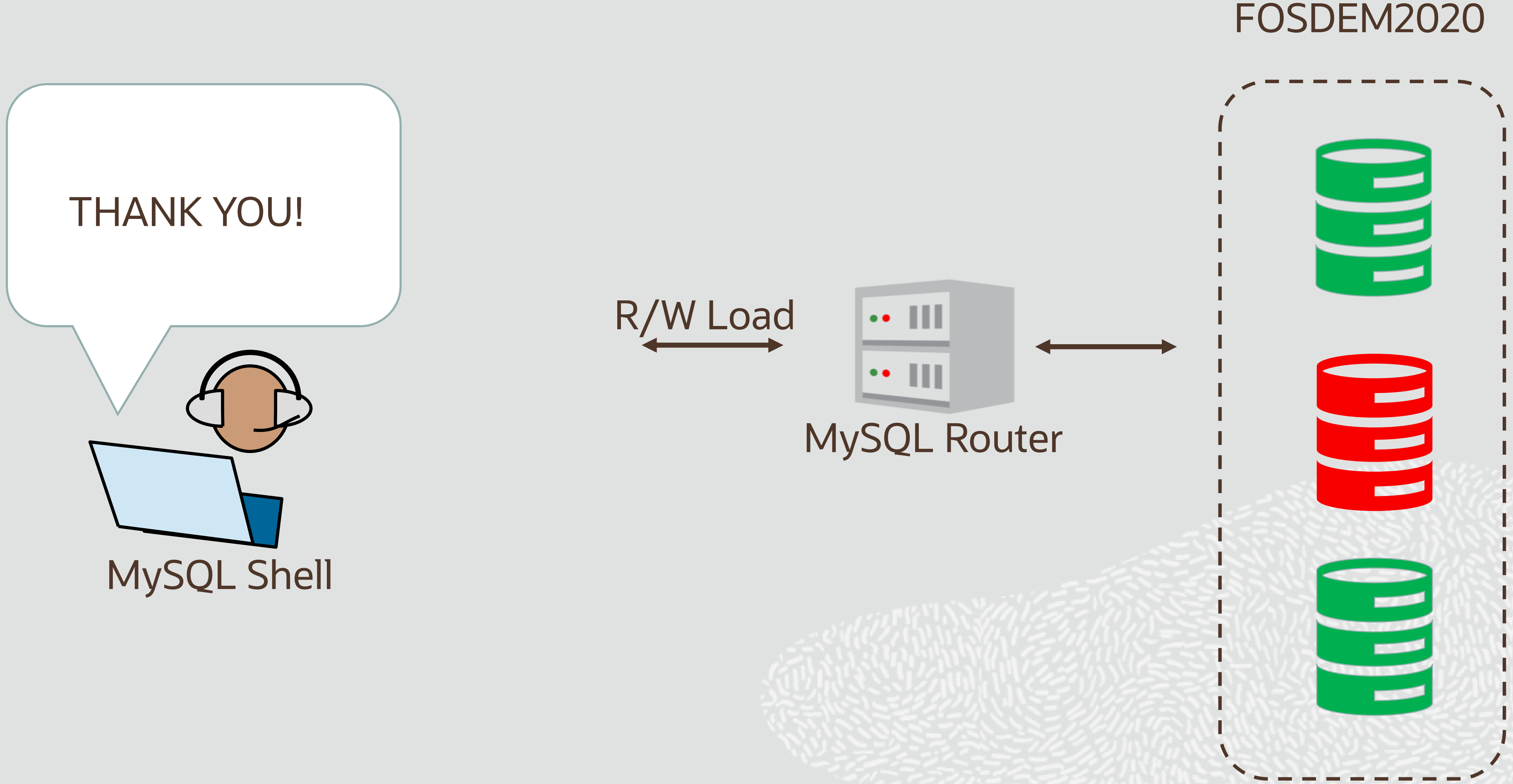
1. **Automatic, self healing**
   - Binlog (if GTID available in group)
   - CLONE (otherwise)
2. **Manual fix**
   - Self healing failed, e.g. network failure
   - rejoinInstance()
   - CLONE
3. **Replace with new instance (permanent failure)**
   - removeInstance()
   - addInstance()
   - CLONE



mysql-js> cluster.rejoinInstance('clusteradmin@mysql1')



MySQL: Fault Tolerant Again



MySQL 8 – CONNECTORS AND DRIVERS

MySQL Engineering

- Node.js Driver (Connector/Node.js)
- Python Driver (Connector/Python)
- C++ Driver (Connector/C++)
- C Driver (Connector/C)
- C API (mysqlclient)
- ADO.NET (Connector/NET)
- ODBC (Connector/ODBC)
- JDBC (Connector/J)

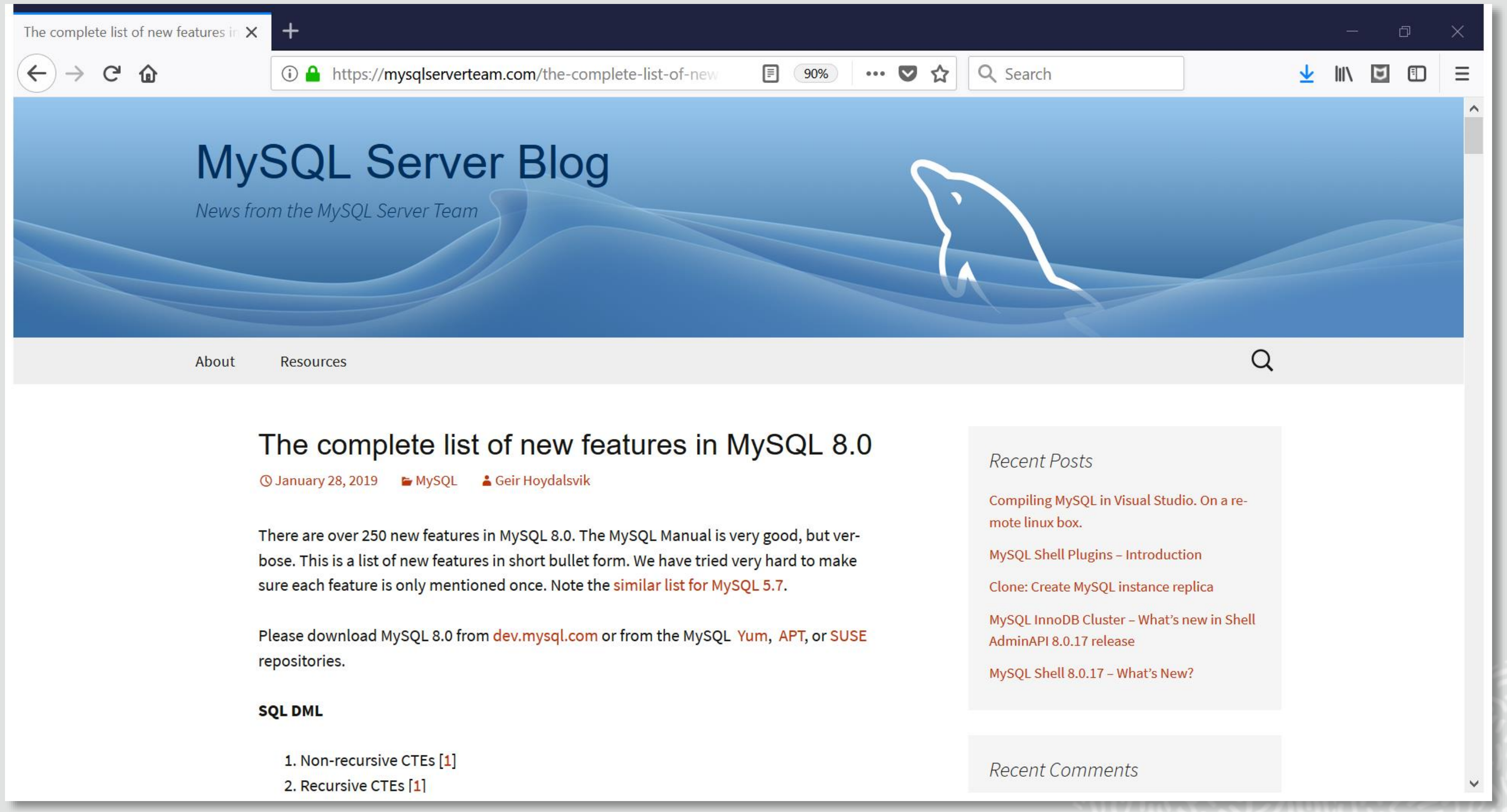
Community

- PHP Drivers for MySQL
  - (mysqli, ext/mysqli, PDO_MYSQL, PHP_MYSQLND)
- Perl Driver for MySQL (DBD::mysql)
- Ruby Driver for MySQL (ruby-mysql)
- C++ Wrapper for MySQL C API (MySQL++)
- Go MySQL Driver
- NodeJS (mysql, mysql2)

MySQL 8 – SOURCE CODE

- Open Source (GPL)
- GitHub https://github.com/mysql/mysql-server
- Wide platform coverage
- C++ 14 , Use of standard constructs, e.g. std::atomic
- Cleaning up header files dependencies
- Warning-free with GCC 8 and Clang 6
- Asan and Ubsan clean
- Google C++ Style Guide
- MySQL Source Code Documentation

MySQL 8 - The complete list of new features


The complete list of new features in
https://mysqlserverteam.com/the-complete-list-of-new
90%
Search
MySQL Server Blog
News from the MySQL Server Team
About
Resources

The complete list of new features in MySQL 8.0

January 28, 2019 MySQL Geir Hoydalsvik

There are over 250 new features in MySQL 8.0. The MySQL Manual is very good, but verbose. This is a list of new features in short bullet form. We have tried very hard to make sure each feature is only mentioned once. Note the similar list for MySQL 5.7.

Please download MySQL 8.0 from dev.mysql.com or from the MySQL Yum, APT, or SUSE repositories.

SQL DML

1. Non-recursive CTEs [1]
2. Recursive CTEs [1]

Recent Posts

Compiling MySQL in Visual Studio. On a remote linux box.

MySQL Shell Plugins – Introduction

Clone: Create MySQL instance replica

MySQL InnoDB Cluster – What's new in Shell AdminAPI 8.0.17 release

MySQL Shell 8.0.17 – What's New?

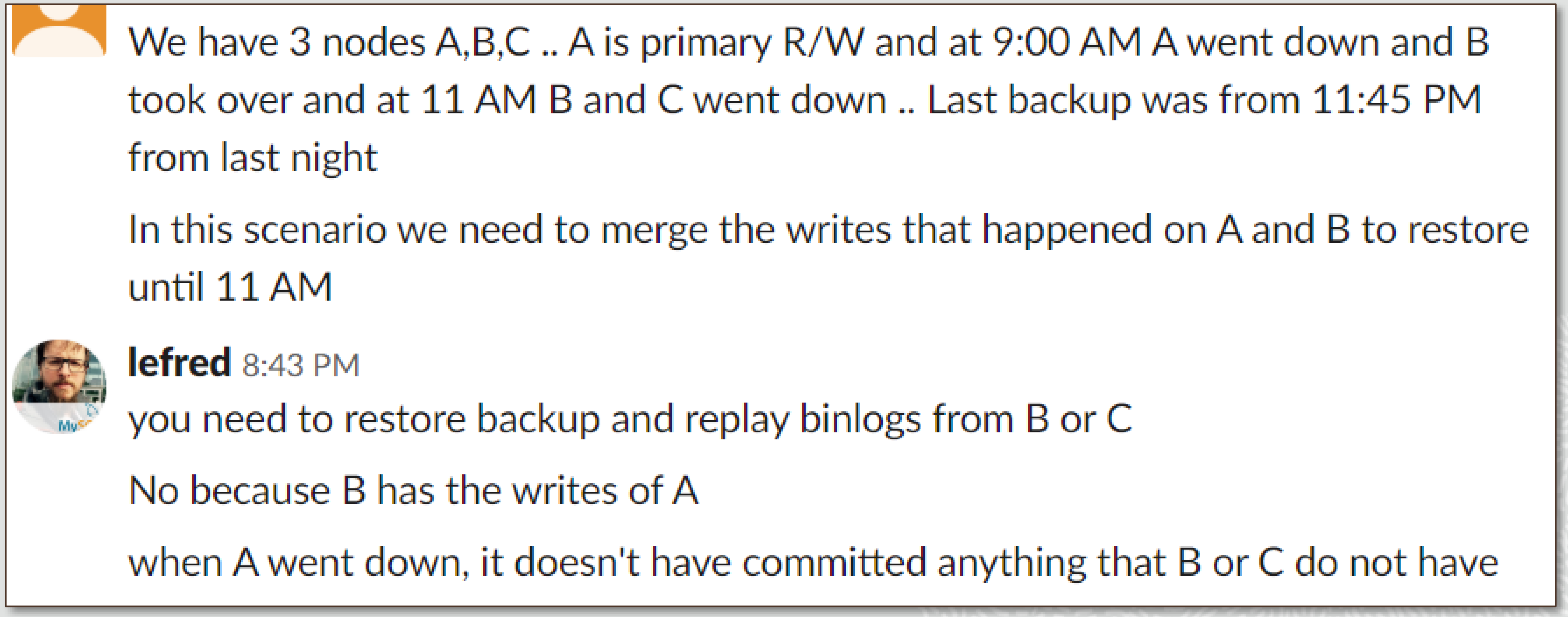
Recent Comments

https://mysqlserverteam.com/the-complete-list-of-new-features-in-mysql-8-0/

MySQL Community on Slack

MySQL
mysqlcommunity.slack.com

https://lefred.be/mysql-community-on-slack/



We have 3 nodes A,B,C .. A is primary R/W and at 9:00 AM A went down and B took over and at 11 AM B and C went down .. Last backup was from 11:45 PM from last night

In this scenario we need to merge the writes that happened on A and B to restore until 11 AM

lefred 8:43 PM
you need to restore backup and replay binlogs from B or C

No because B has the writes of A

when A went down, it doesn't have committed anything that B or C do not have

MySQL on Social Media

https://www.facebook.com/mysql

https://twitter.com/mysql

https://www.linkedin.com/company/mysql

ORACLE