

ORACLE

ORACLE

MySQL Shell for Database Engineers

The Best MySQL Administration Tool





Herbert Menezes

MySQL Master Principal Solutions Engineer

MySQL 5.7 Database Administrator



OCI 2023 AI Certified Foundations Associate

OCI 2023 Foundations OCA

OCI 2022 Architect OCA

OCI 2020 Architect OCA

OCI 2018 Architect OCA



Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes 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 upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Context

MySQL turned 28 years old this year!

It's a relational database:

- SQL
- Has a command-line SQL client: mysql

But MySQL isn't just relational databases...



Meet MySQL Shell

Modern, new command-line client for MySQL

Written from scratch, supporting:

- MySQL Document Store (NoSQL, X DevAPI)
- JSON Documents / SQL Tables
- SQL, Python and JavaScript
- Fully customizable





There's much more in Shell than meets the eye!

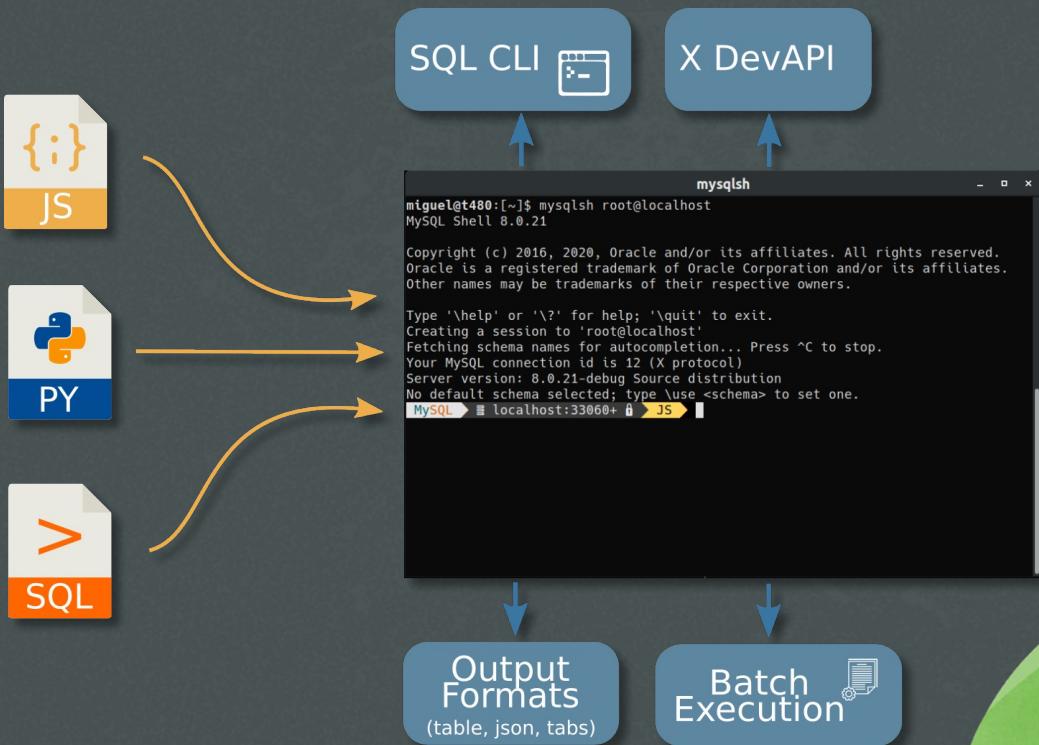
MySQL Shell

A DBA and Developer Toolbox:

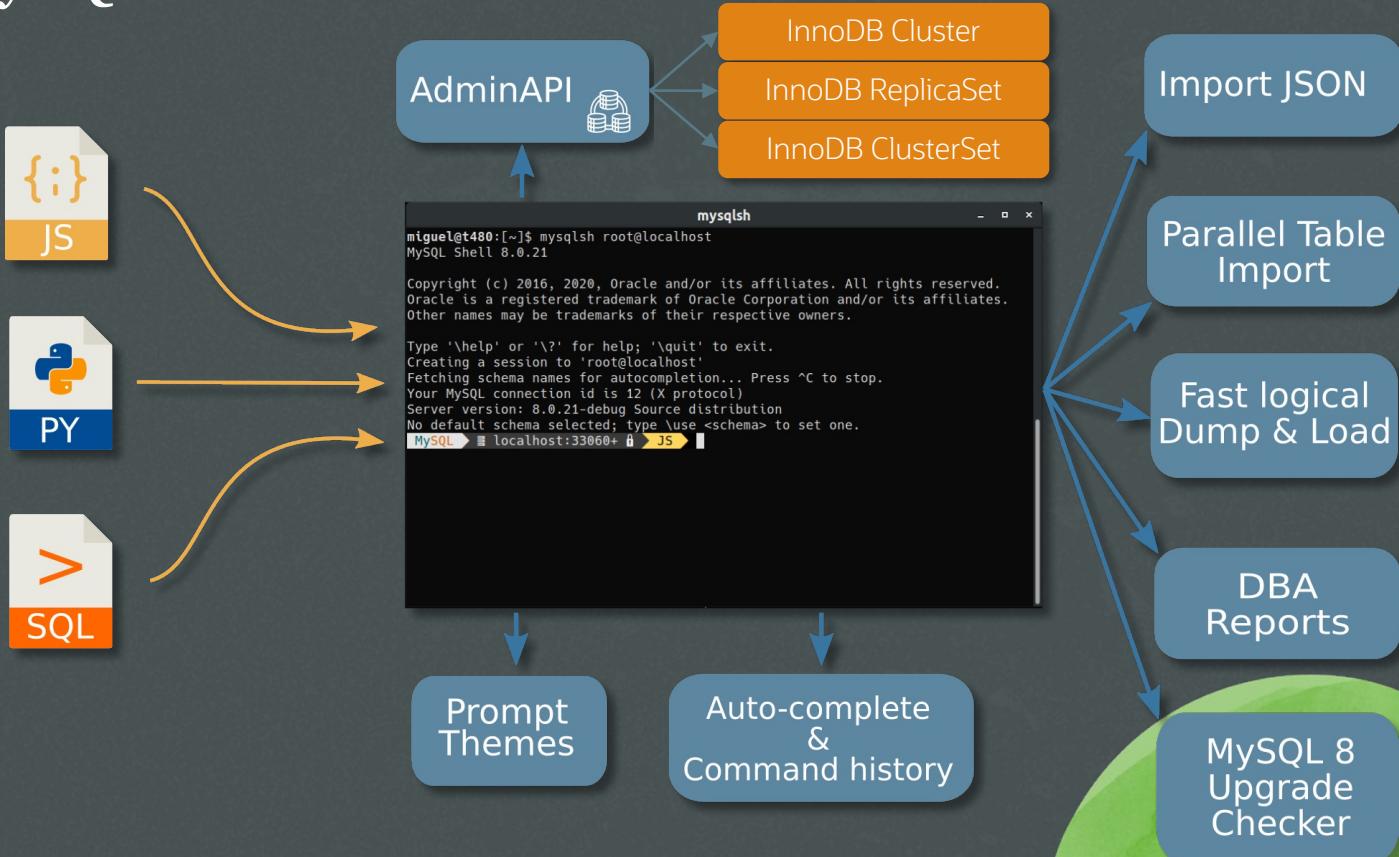
- SQL, JavaScript and Python support
- Auto-completion
- Command history
- Integrated built-in help system
- Customizable prompts / colors
- APIs and Utilities built-in
- Extensible
- Open-source!



MySQL Shell Overview



MySQL Shell Overview



The MySQL Toolbox

MySQL Protocols

- mysql
- mysqlx



The MySQL Toolbox

MySQL Protocols

- mysql
- mysqlx

ShellAPI

- OS utilities
- General purpose functions
- Create Reports
- Create Plugins
- Manage Credentials



The MySQL Toolbox

MySQL Protocols

- mysql
- mysqlx

ShellAPI

- OS utilities
- General purpose functions
- Create Reports
- Create Plugins
- Manage Credentials

AdminAPI

- InnoDB Cluster
- InnoDB ReplicaSet
- InnoDB ClusterSet
- Sandboxes



The MySQL Toolbox



MySQL Protocols

- mysql
- mysqlx

ShellAPI

- OS utilities
- General purpose functions
- Create Reports
- Create Plugins
- Manage Credentials

AdminAPI

- InnoDB Cluster
- InnoDB ReplicaSet
- InnoDB ClusterSet
- Sandboxes

Utilities

- Upgrade checker
- JSON import
- Parallel import table
- Instance and Schema dump
- Dump loading



Multi-language Support

- SQL
- JavaScript
 - Full JS engine (V8)
 - All core language features
- Python
 - 3.6+

SQL

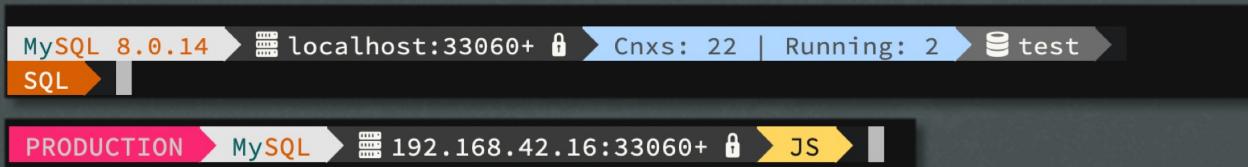
JS

Python



Pleasant experience

- Auto-complete
- Customizable prompt with colors and state information



Pleasant experience



- Auto-complete
- Customizable prompt with colors and state information

```
MySQL 8.0.14 ➔ localhost:33060+ 🔒 ➔ Cnxs: 22 | Running: 2 ➔ test ➔  
SQL ➔  
  
PRODUCTION ➔ MySQL ➔ 192.168.42.16:33060+ 🔒 ➔ JS ➔
```

- Command history; Pager: more/ less

```
MySQL ➔ t480:33060+ 🔒 ➔ SQL ➔ \pager less  
Pager has been set to 'less'.  
MySQL ➔ t480:33060+ 🔒 ➔ SQL ➔ SHOW STATUS;
```

Pleasant experience

- Integrated built-in Help system
- Accessible via:
 - `\? <topic>`
 - `\help <topic>`
- All functionality available

```
MySQL ➤ SQL ➤ \help \connect
NAME
  \connect - Connects the shell to a MySQL server and assigns the global
session.

SYNTAX
  \connect [<TYPE>] <URI>
  \c [<TYPE>] <URI>

DESCRIPTION
  TYPE is an optional parameter to specify the session type. Accepts the
following values:
  - --mc, --mysql: create a classic MySQL protocol session (default port
    3306)
  - --mx, --mysqlx: create an X protocol session (default port 33060)

  If TYPE is omitted, automatic protocol detection is done, unless the
  protocol is given in the URI.

  URI format is: [user[:password]@]hostname[:port]

EXAMPLE
  \connect --mx root@localhost
  Creates a global session using the X protocol to the indicated URI.
```

Built-in APIs

- APIs for managing and interacting with MySQL
- All APIs available in JavaScript and Python:
 - DevAPI
 - ShellAPI
 - AdminAPI
- Flexibility and power of choice





Database APIs

Classic protocol

```
// Print a welcome message
println('Create a local MySQL account with special privileges.');

// Prompt for the username if not passed by argument
if (sys.argv[1] == undefined) {
    username = shell.prompt('Please enter the username for the account:');
} else {
    username = sys.argv[1];
}

// Prompt for the password
pwd = shell.prompt('Please enter a password for the account ' + username + ':',
{type: 'password'});
```

Database APIs

Classic protocol

```
// List of required grants
var grantsList = [
    'USAGE ON *.*',
    'SELECT, INSERT, UPDATE, DELETE, CREATE TEMPORARY TABLES ON world_x.*'
];

try {
    // Create the account
    session.runSql('CREATE USER ?@\'localhost\' IDENTIFIED BY ?',
        [username, pwd]);
    for (var grant in grantsList) {
        session.runSql('GRANT ' + grantsList[grant] + ' TO ?@\'localhost\'',
            [username])
    }
} catch (err) { (...) }
```

Multiple MySQL connections

- Work with multiple MySQL sessions simultaneously
- Using ShellAPI and/or X DevAPI

Example in Python:

```
session1 = mysql.get_session("admin@t480:3330")
session2 = mysql.get_session("root@localhost:3306")

session1.run_sql("SELECT @@hostname, @@port;")

result = session2.run_sql("SELECT * FROM performance_schema.global_status;")
row = result.fetch_all()
print(row[10])
```

Utilities

- DBA focused:
 - Upgrade Checker
 - JSON import
 - Parallel table import
 - Dump & load
 - Session inspector
- Exposed on the **ShellAPI util** module



Upgrade Checker

- Check for compatibility issues for upgrading to MySQL 8
 - **5.7 → 8.0+** (current Shell version)
- Warns about required actions:
 - Obsolete/conflicting settings
 - Schema properties requiring changes
- Can check .cnf files

```
// Available through the util object
mysqlsh-js> util.checkForServerUpgrade("root@localhost");
```

Upgrade Checker

```
MySQL ➔ t480:3361 ➔ JS ➔ util.checkForServerUpgrade()
The MySQL server at t480:3361, version 5.7.26 - MySQL Community Server (GPL),
will now be checked for compatibility issues for upgrade to MySQL 8.0.22...

1) Usage of old temporal type
   No issues found

2) Usage of db objects with names conflicting with new reserved keywords
   No issues found

3) Usage of utf8mb3 charset
   No issues found

4) Table names in the mysql schema conflicting with new tables in 8.0
   No issues found

5) Partitioned tables using engines with non native partitioning
   No issues found

6) Foreign key constraint names longer than 64 characters
```

API Command-line Integration

- All APIs are available on the command-line
- Accessible using the special delimiter character `--`
- Arguments follow a syntax suitable for command-line use:

```
mysqlsh [shell options] -- <object> <command>  
[command options]
```

Example:

```
mysqlsh admin@t480 -- cluster status --extended=1
```

API Command-line Integration

- Integrating Shell's Upgrade Checker with Puppet:

```
service { 'mysql':
  ensure => latest,
  enable => true,
  require => Package['mysql-server-community'],
  before => Exec['upgrade checker']
}

exec { "upgrade checker":
  command => "mysqlsh -- util check-for-server-upgrade --user=root
--host=localhost --port=33100 --password='mypassword'"
  require => service["mysql"]
}
```



JSON Import

- Import JSON documents into a collection or relational table
- Removes the need to use multiple INSERT statements or write scripts
- Supports BSON data types

```
// Available through the util object
mysqlsh-js> util.importJson("/home/miguel/banks.json", {schema: "docstore",
collection: "banks"})
```

```
// Very handy to be used with the API command-line integration
$ mysqlsh admin@localhost/docstore -- util import-json banks.json --collection=banks
```

Dump & Load

- Powerful utilities to make it easy to do logical dumps and load of:
 - A whole database instance
 - A set of schemas
- Focus on ease of use:

```
// Dump an entire database instance, including users
util.dumpInstance(...)
```

```
// Dump a set of schemas
util.dumpSchemas(...)
```

```
// Load a dump into a target database
util.loadDump(...)
```

```
// Streaming a dump into a target database
util.copyInstance(...)
```

Dump & Load

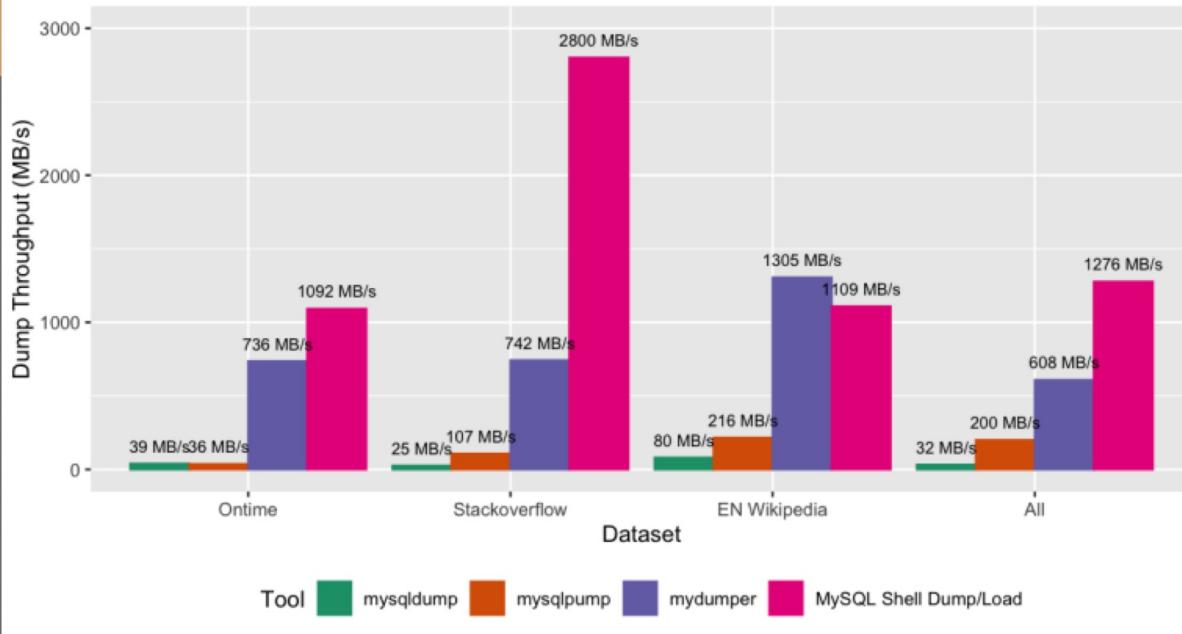
- Focus on performance:
 - Multi-threaded dump and load
 - Concurrent execution of dump and load
 - Built-in compression (zstd & gzip)
- Focus on integration:
 - Dump & Load straight to/from OCI Object Storage/Amazon S3
 - Compatibility modes for importing into OCI MySQL Database Service



Dump & Load

MySQL Database Logical Dump Throughput - Comparison

- MySQL 8.0.21 - Redo Log Disabled, Oracle Linux 7.8
- OCI BM.Standard.B1.44 - 44x Intel Xeon E5-2699 v4 - 512GB RAM
- Storage: 8x 400GB - 240MB/s in RAID-0



Replication & High Availability

- MySQL offered individual components to achieve Highly Available setups
 - Leaving it up to the user to set-up the architectures...
 1. Provisioning, restoring backups
 2. User management
 3. Replication configuration
 4. Deployment of client proxies
 5. Manually changing topologies (or rely on external tools)
 6. Use additional monitoring tools



A lot of work for DBA's and SysAdmins,
who spend their time automating

AdminAPI

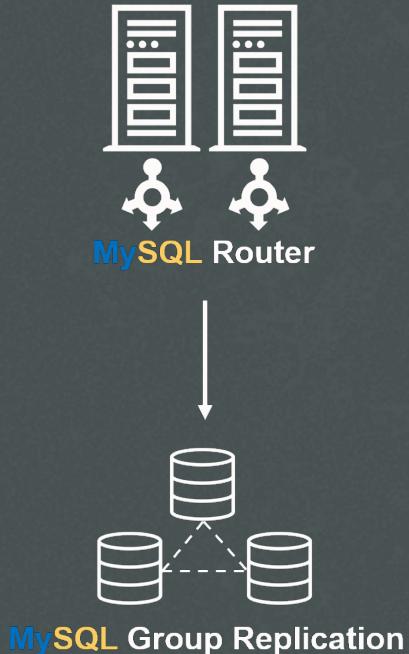
- Setup and Management of InnoDB Cluster / ReplicaSet / ClusterSet
- Hides the complexity of:
 - Configuration
 - Provisioning
 - Orchestration
- Simple and straight-forward
- Doesn't require MySQL expertise
- Flexible, powerful and secure



MySQL InnoDB Cluster



MySQL Shell



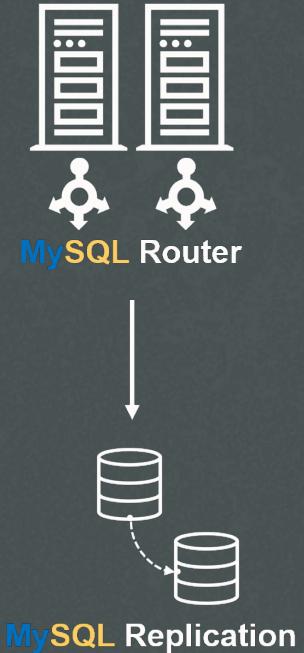
- MySQL Group Replication
- MySQL Shell
- MySQL Router

Complete HA Solution!

MySQL InnoDB ReplicaSet



MySQL Shell



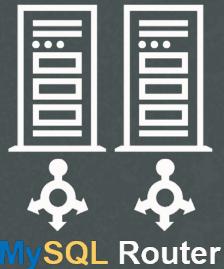
- MySQL Replication
- MySQL Shell
- MySQL Router

Simple but complete Replication Solution!

MySQL InnoDB ClusterSet



MySQL Shell



MySQL Router



MySQL Group Replication

Assinc Replication



MySQL Group Replication

- MySQL Replication
- MySQL Shell
- MySQL Router

Simple but complete HA + DR Solution!

AdminAPI

Creating a Cluster

```
// Configure Instances for Cluster usage
mysqlsh-js> dba.configureInstance("root@mysql1", {clusterAdmin: "admin"})
(...)

// Create the Cluster
mysqlsh-js> \c admin@mysql1
mysqlsh-js> cluster = dba.createCluster("myCluster")

// Add Instances to the Cluster
mysqlsh-js> cluster.addInstance("mysql2")
```



Couldn't be easier!

Powerful solutions supported by Shell's ease of use

ShellAPI

- General purpose functions and properties
- OS utilities
- Credential Manager
- Extend MySQL Shell
 - Reports
 - Plugins



Credential Manager

- Store passwords using a secret store or keychain
- Greatly improves usability and saves time
- Built-in support for:
 - MySQL login-path (`mysql_config_editor`)
 - macOS keychain
 - Windows Credentials Management API

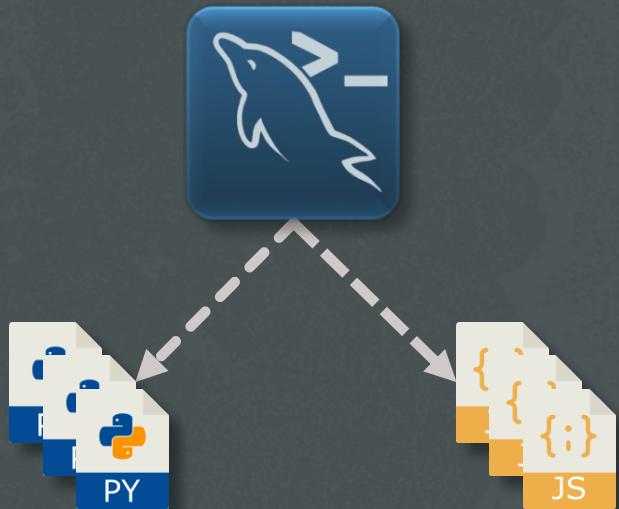


Credential Manager

```
// List available Secret Stores
mysqlsh-js> shell.listCredentialHelpers();
[
    "keychain",
    "login-path"
]
// List stored credentials
mysqlsh-js> shell.listCredentials();
[
    "miguel@t480:3306",
    "admin@10.0.2.15:3306",
    "customer@10.0.2.15:3306"
]
```

Extend MySQL Shell

- Extend Shell functionality through plugins
- Create new global reports or extensions objects
- Available in JavaScript or Python
- All Shell APIs available
- Automatically loaded at start-up



Extend MySQL Shell

- API to register extensions:

```
// Register a user-defined report
```

```
shell.registerReport([name, type, report[, description]]);
```

```
// Create an extension object
```

```
shell.createExtensionObject();
```

```
// Add a new member into an extension object
```

```
shell.addExtensionObjectMember([object, name, member[, definition]]);
```

```
// Register an extension object as a Shell global object
```

```
shell.registerGlobal(name, object[, definition]);
```

Create a Report to get the uptime

```
.mysqlsh/init.d/uptime.py:0

def uptime(session):
    stmt = "SELECT TIME_FORMAT(SEC_TO_TIME(VARIABLE_VALUE ), '%Hh %im %ss') AS Uptime
FROM performance_schema.global_status WHERE VARIABLE_VALUE='Uptime';"

    result = session.run_sql(stmt)
    report = [result.get_column_names()]

    for row in result.fetch_all():
        report.append(list(row))

    return {'report': report}
```

Create a Report to get the uptime

```
.mysqlsh/init.d/uptime.py:13  
shell.register_report(  
    'uptime',  
    'list',  
    uptime,  
    {  
        'brief': 'Shows Server Uptime.',  
        'details': ['You need the SELECT privileges on performance_schema.*'],  
        'arg': '0'  
    }  
}
```

Create a Report to get the uptime

```
MySQL ➔ localhost:3361 ➔ JS ➔ \? shell.reports.uptime
NAME
  uptime - Shows Server Uptime.

SYNTAX
  shell.reports.uptime(session)

WHERE
  session: Object. A Session object to be used to execute the report.

DESCRIPTION
  This is a 'list' type report.

  You need the SELECT privilege on performance_schema.global_status.

  The session parameter must be any of ClassicSession, Session.

MySQL ➔ localhost:3361 ➔ JS ➔ \show uptime
+-----+
| Uptime      |
+-----+
| 29h 00m 07s |
+-----+
```

Create a Plugin for system information

```
.mysqlsh/plugins/ext/system_info/init.py
```

```
from mysqlsh.plugin_manager import plugin, plugin_function  
@plugin
```

```
class system_info:
```

```
    """
```

```
System Information
```

```
A collection of tools to gather system information.
```

```
"""
```

```
@plugin_function("system_info.get_public_key")
```

```
def get_public_key(session=None, verbose=False):
```

```
    """
```

```
Get RSA Public Key of (...)
```

Create a Plugin for system information



```
MySQL ➔  localhost:3320  Py  \? system_info
NAME
    system_info - System Information

DESCRIPTION
    A collection of tools to gather system information.

FUNCTIONS
    get_public_key([session][, verbose])
        Get RSA Public Key of the caching_sha2_password auth plugin

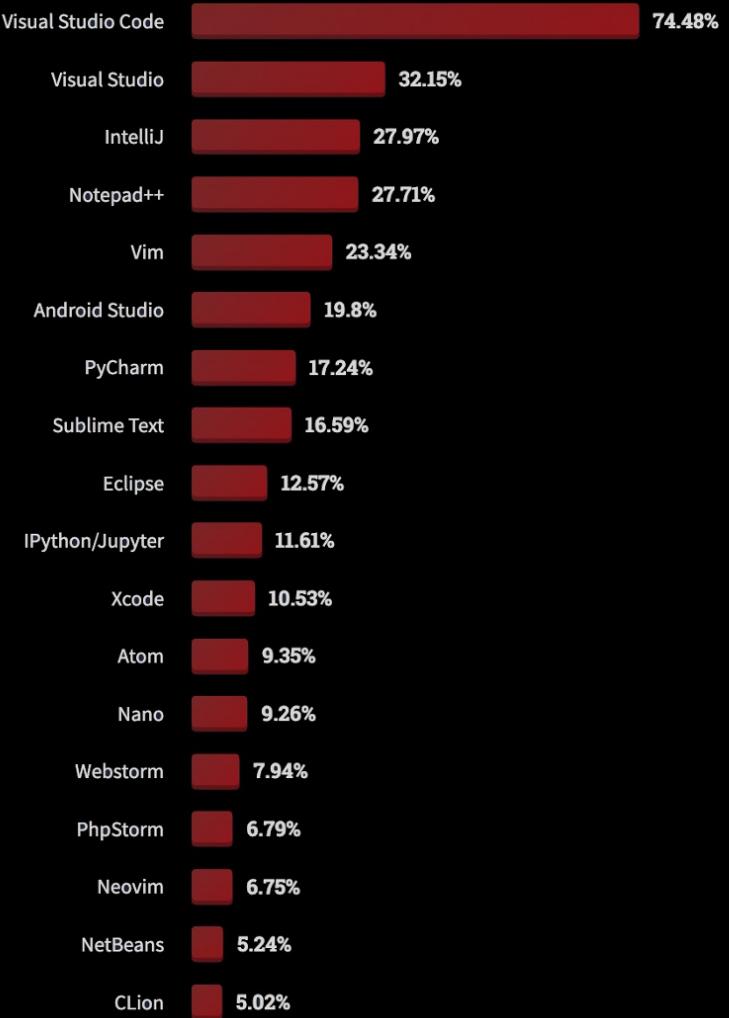
    help([member])
        Provides help about this object and it's members

    uptime([session][, verbose])
        Get the server uptime

MySQL ➔  localhost:3320  Py  system_info.get_public_key(session, True)
-->  SELECT VARIABLE_VALUE FROM performance_schema.global_status WHERE VARIABLE_NAME = 'Caching_sha2_password_rsa_public_key';
----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEAAQ8AMIIBCgKCAQEAYwoRH7uBILAyIiI31gWGj
dqq34fLqADgZFcP2m+F9I1TqUfBGWGVAMAzXM8IvTmqq+wvzkFHrvYtcZWEmdIU
0fVQNGd9eFc1tqezdjE5bjUCV3ohW5T/Z4NY0Rs1RRWKJGj3krdXMgVCj+Y0kGo0
rqZ/S2VYraRP0DLZPqREWxajgYrqaEaRSZN++YVb1CU/cXoIDCbiGvJrd5NwyxSM
1Wb4VCUXZrpvvNALuaJD+WIOPYQUYta6FnG01/Jm6nMOXk+MDnni4C0VR6x7qR0
i7k0avCFvWD9MJ05e8PrFYJtY1AW80U6ciR3Wka6KIkuS14bLmg9CJa/wzEzuup
7QIDAQAB
----END PUBLIC KEY-----
```

MySQL Shell for VSCode

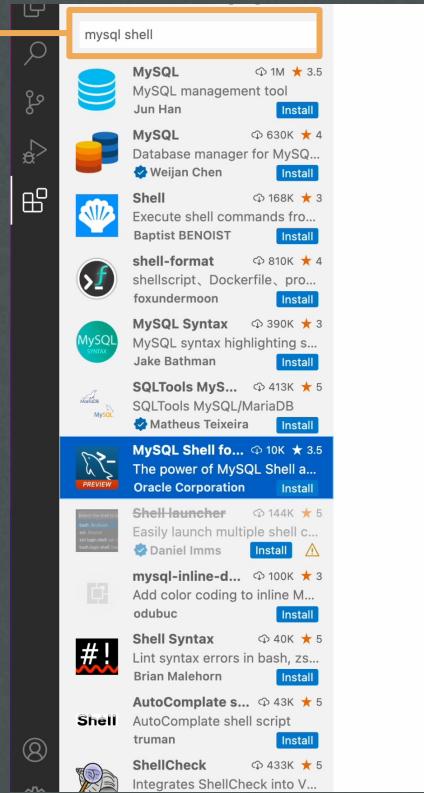




“In the Stack Overflow 2022 Developer Survey, **Visual Studio Code** was ranked the most popular developer environment tool, with 74% of 71,000 respondents reporting that they use it.”

Extension Installation

Install
MySQL Shell for VSCode
directly in Marketplace

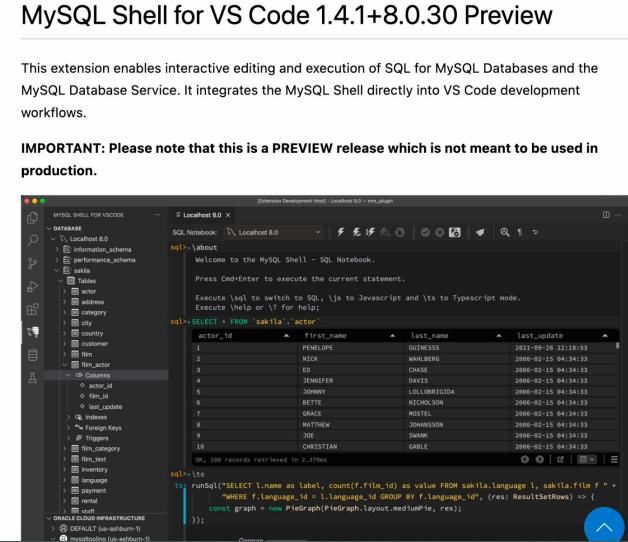


The screenshot shows the VS Code Marketplace search results for "mysql shell". The top result is "MySQL Shell" by Jun Han, version v1.4.1, with 3.5 stars and 1M installs. Below it are other extensions like "MySQL" by Weijan Chen, "Shell" by Baptist BENOIST, "shell-format" by foxundermoon, "MySQL Syntax" by Jake Bathman, "SQLTools MySQL" by Matheus Teixeira, and "MySQL Shell for VS Code" by Oracle Corporation, which is currently selected and shown in preview mode.



The extension page for "MySQL Shell for VS Code" shows the following details:

- MySQL Shell for VS Code** v1.4.1
- Oracle Corporation | 10,182 installs | ★★★★☆ (10)
- The power of MySQL Shell as part of your VS Code workflow.
- Install button
- Preview icon
- Details, Feature Contributions, Changelog tabs

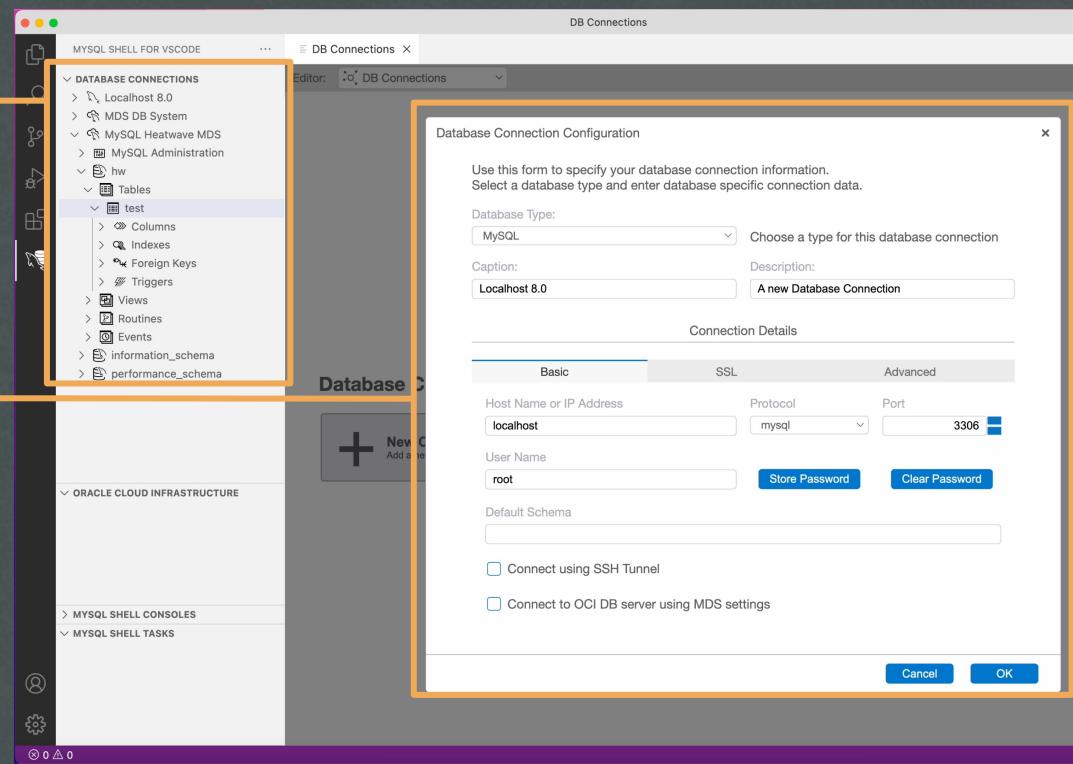


A screenshot of the MySQL Shell for VS Code extension in VS Code. It shows a sidebar with database structures like "information_schema", "performance_schema", and "sakila". The main area displays a SQL query: "SELECT * FROM `sakila`.`actor`" and its results, showing 19 rows of actor information. A status bar at the bottom indicates "OK, 1988 records retrieved in 2.37ms".

Adding DB Connections

Database Editor
Connections View

Create Remote
MySQL Connections



SQL Notebook Interface

Write, Execute, Edit,
Execute within Notebook

The screenshot shows a SQL Notebook interface with two main sections. The top section displays the results of a query against the `sakila` database:

```
sql> SELECT * FROM `sakila`.`actor`
 WHERE actor_id > 35
```

actor_id	first_name	last_name	last_update
36	BURT	DUKAKIS	2006-02-15 04:34:33
37	VAL	BOLGER	2006-02-15 04:34:33
38	TOM	MCKELLEN	2006-02-15 04:34:33
39	GOLDIE	BRODY	2006-02-15 04:34:33
40	JOHNNY	CAGE	2006-02-15 04:34:33
41	JODIE	DEGENERES	2006-02-15 04:34:33
42	TOM	MIRANDA	2006-02-15 04:34:33
43	KIRK	JOVOVICH	2006-02-15 04:34:33
44	NICK	STALLONE	2006-02-15 04:34:33
45	REESE	KILMER	2006-02-15 04:34:33
46	DARREN	COPPER	2006-02-15 04:34:33

The bottom section shows another query and the start of a TypeScript block:

```
sql> SELECT * FROM `sakila`.`address`
```

address_id	address	address2	district	city_id	postal_code
1	47 MySakila Drive	NULL	Alberta	300	1
2	28 MySQL Boulevard	NULL	QLD	576	6
3	23 Workhaven Lane	NULL	Alberta	300	1
4	1411 Lillydale Drive	NULL	QLD	576	6
5	1913 Hanoi Way		Nagasaki	463	35200
6	1121 Loja Avenue		California	449	17886
7	692 Joliet Street		Attika	38	83579
8	1566 Inegl Manor		Mandalay	349	53561
9	53 Idfu Parkway		Nantou	361	42399
10	1795 Santiago de Compostela Way		Texas	295	18743

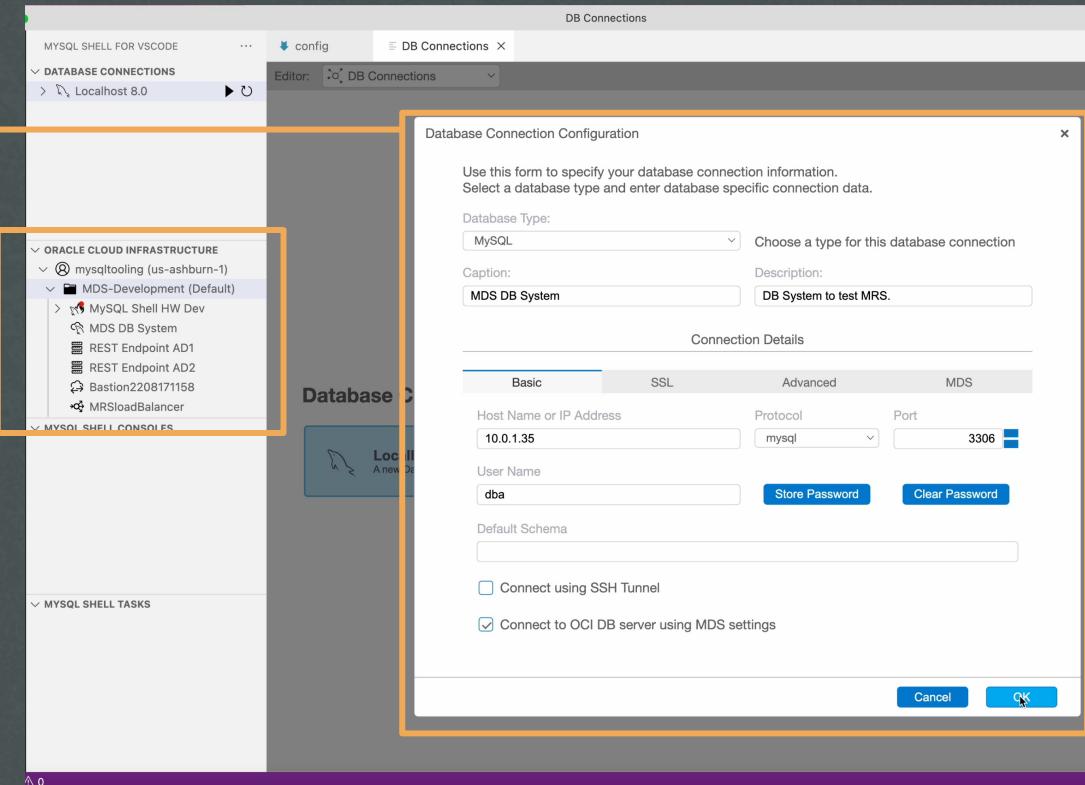
```
sql> \ts
Switched to TypeScript mode
ts> runSql("SELECT * FROM sakila.actor", (res) => {})
```

TypeScript
(JS & py Support)

OCI MySQL HeatWave

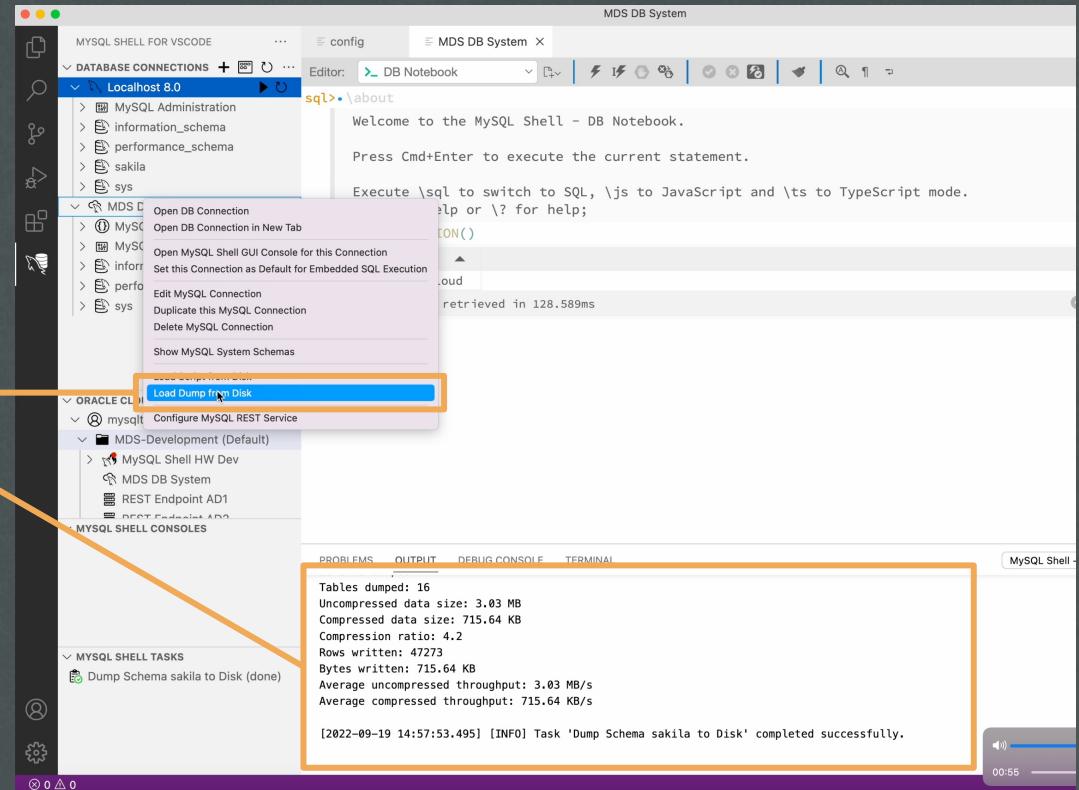
Connect Directly to
DBSystem via
OCI Bastion Service

Access OCI MySQL
Infrastructure



Lift and Shift

MySQL Shell Dump & Load
Within VS Code



Integration MySQL HeatWave for OLAP

Execute Directly with
HeatWave

Connect Directly from
Customers On-Premise
Network via Bastion

MDS/Heatwave
Management

HeatWave Cluster Tuning

The screenshot shows the MySQL Shell for VS Code interface. On the left, there's a sidebar with various connection and task options. The main area is a DB Notebook where SQL statements are executed. The interface includes a toolbar at the top with several icons.

MySQL Shell for VS Code:

- DATABASE CONNECTIONS:** MySQL Heatwave MDS (selected), MySQL Administration, hw, Tables (test selected), information_schema, performance_schema, ORACLE CLOUD INFRASTRUCTURE, mysqltooling (us-ashburn-1), MDS Development (Default).
- MYSQL SHELL CONSOLES:** MySQL Heatwave MDS (selected), MySQL DB System, REST Endpoint AD1, REST Endpoint AD2, Bastion220817158, MRSloadBalancer, / (Root Compartment).
- MYSQL SHELL TASKS:** Rescale HeatWave Cluster (done).

MySQL Heatwave MDS:

- Toolbar:** Includes icons for Run, Refresh, Stop, and others.
- SQL Statement:** sql> \about
- Output:** Welcome to the MySQL Shell - DB Notebook. Press Cmd+Enter to execute the current statement. Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript. Execute \help or \? for help;
- Table Output:** SELECT * FROM `hw`.`test` (partial data shown)
- SQL Statement:** sql> .SELECT COUNT(*) FROM hw.test;
- Output:** COUNT(*) ▲
2000792
OK, 1 record retrieved in 144.559ms
- SQL Statement:** sql> .SELECT age % 7, COUNT(age) FROM `hw`.`test` GROUP BY age % 7 ORDER BY age % 7;
- SQL Statement:** sql> |

Executing Embedded SQL

Execute SQL Snippets in
MySQL Shell

DB Notebook interface

Update SQL
Snippets into source

The screenshot shows a DB Notebook interface with several panes:

- EXPLORER** pane: Shows the project structure with files like `__pycache__`, `.vscode`, `db_schema`, `internal`, `tests`, `__init__.py`, `.coveragerc`, `.gitignore`, `auth_apps.py`, `content_files.py`, `content_sets.py`, `core.py`, `cspell.json`, `db_objects.py` (selected), `general.py`, `init.py`, `LICENSE`, `pytest-coverage.ini`, `pytest.ini`, `python.env`, `readme.md`, `requirements.txt`, `run_tests.py`, `schemas.py`, and `services.py`.
- db_objects.py 2** pane: A code editor showing Python code for interacting with MySQL. It highlights a snippet of SQL code:

```
else:
    sql = """
        SELECT ROUTINE_NAME AS OBJECT_NAME
        FROM INFORMATION_SCHEMA.ROUTINES
        WHERE ROUTINE_SCHEMA = ? /*=sakila*/
        AND ROUTINE_TYPE = 'PROCEDURE'
        ORDER BY ROUTINE_NAME
    """

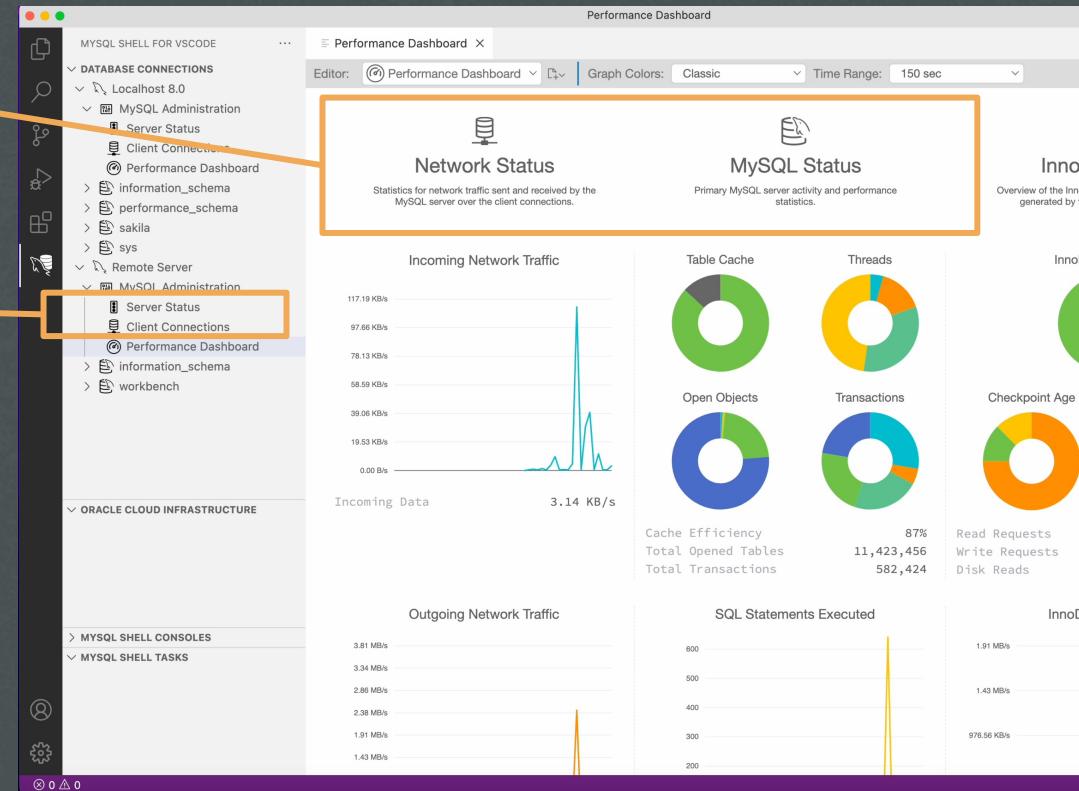
    res = session.run_sql(
        sql, [schema.get("name")])
    rows = res.fetch_all()
    db_objects = [
        row.get_field("OBJECT_NAME") for row in rows]
```
- Localhost 8.0** pane: A MySQL Shell session window showing the execution of the SQL snippet. The command entered is:

```
sql>+SELECT ROUTINE_NAME AS OBJECT_NAME
FROM INFORMATION_SCHEMA.ROUTINES
WHERE ROUTINE_SCHEMA = /*=sakila*/
AND (ROUTINE_TYPE = 'PROCEDURE'
OR ROUTINE_TYPE = 'FUNCTION')
ORDER BY ROUTINE_NAME
```
- Editor** pane: A context menu is open over the SQL command, with the option **Update SQL in Original Source File** highlighted.

MySQL Administration

MySQL Performance Metrics

Quick overview of Server Status



MySQL REST Service

Fast, Secure HTTPS Access for your MySQL Data

RESTful Web Services

- Auto REST for tables, views and procedures
- {JSON} responses
- Paged results
- Developer support (GUI, CLI, API)

MySQL Shell for VS Code

- GUI frontend for MRS management
- RESTful Web Service creation
- Interactive documentation
- CLI & scripting support

Built in User Management

- Support for popular OAuth2 services
- Use Role, Group & Hierarchy Management
- User management GUI
- CLI & scripting support

MySQL REST Service

Manage REST Service

The screenshot shows the MySQL Shell interface for VS Code. On the left, the sidebar displays database connections, including 'localhost 8.0' and 'MySQL REST Service'. The 'MySQL REST Service' connection is expanded, showing its schema, tables, and sub-tables like 'actor', 'address', etc. A yellow box highlights the 'MySQL REST Service' connection in the sidebar. An orange arrow points from the text 'Manage REST Service' to this highlighted connection.

MySQL REST Object

Adjust the MySQL REST Object Configuration

MRS Service Path: /mrs

MRS Schema Path: /sakila

Request Path: /actor

Database Object Name: actor

CRUD Operations: READ

Flags: Enabled, Requires Authentication

Basic Parameters Advanced Options

Result Format: FEED

Items per Page:

Comments:

Row Ownership: Enforce Row User Ownership

Row Ownership Field:

Cancel OK

The central part of the screenshot shows the 'MySQL REST Object' configuration dialog. The 'Request Path' field is highlighted with an orange border. The entire dialog box is also outlined with an orange border. Another orange arrow points from the text 'Create REST Objects by mapping them to database objects' to the 'Request Path' field in the dialog.

Summary

- MySQL Shell's goal is to be the primary frontend for most of MySQL functionality
- It's an unified interface for Developers and DBAs
- Brings ease of use for complex tasks
- Frontend manager for InnoDB Cluster and InnoDB ReplicaSet
- Extensible and open-source!

- 
- ✓ Give it a try!
dev.mysql.com/downloads/shell/
 - ✓ Check and contribute to lefred's collection
of MySQL Shell plugins:
github.com/lefred/mysqlshell-plugins
 - ✓ Reach us out on Slack!
mysqlcommunity.slack.com
#shell | #mysql_innodb_cluster

Thank you!

ORACLE