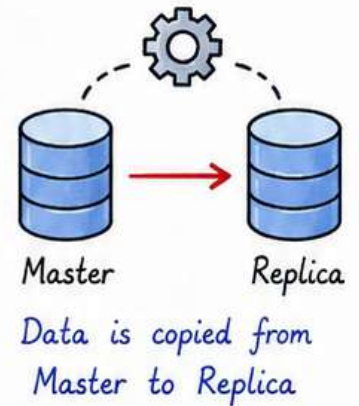


Chapter 31: MySQL Replication & High Availability

31.1 Introduction to Replication

- * Replication is the process of copying data from one MySQL server (**source**) to another MySQL server (**replica**).
- * It helps in improving **availability**, **reliability** and **performance** of databases.
- * Replication ensures that multiple servers have the **same data**.
- * It provides **fault tolerance** and **disaster recovery**.
- * MySQL uses **asynchronous replication** by default.



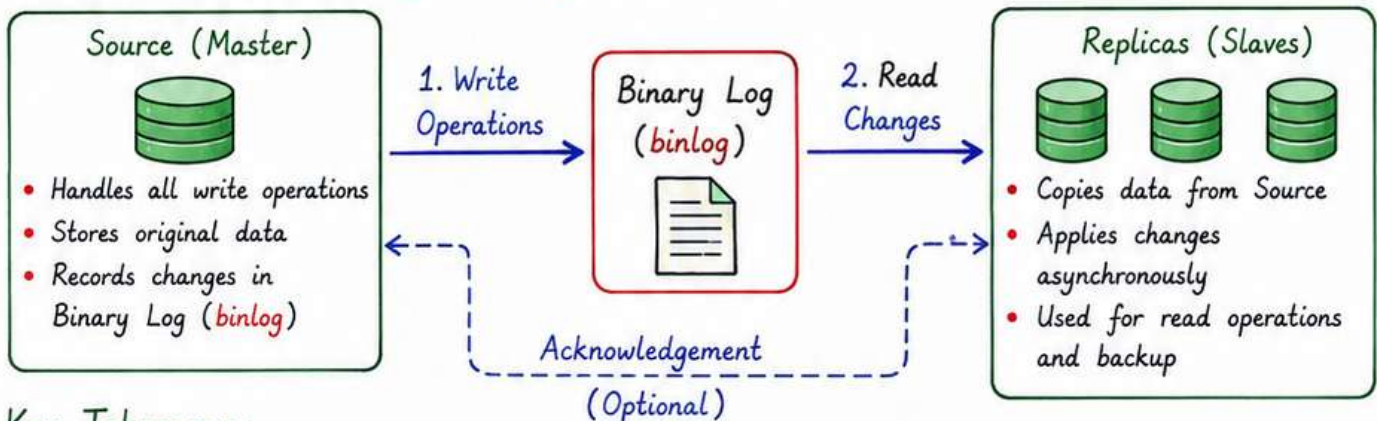
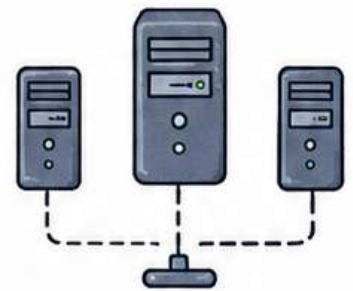
Key Points :

- ✓ Replication increases data availability.
- ✓ It allows read scaling by offloading read queries to replicas.
- ✓ It protects data in case of server failure.



31.2 Source-Replica Architecture

- * In MySQL replication, one server acts as the **Source (Master)** which has the original data.
- * Other servers act as **Replicas (Slaves)** which copy data from the Source.
- * All **write operations** (INSERT, UPDATE, DELETE) are performed on the **Source**.
- * **Read operations** can be performed on both Source and Replicas.
- * Replicas are updated **asynchronously** from the Source.



Key Takeaway :

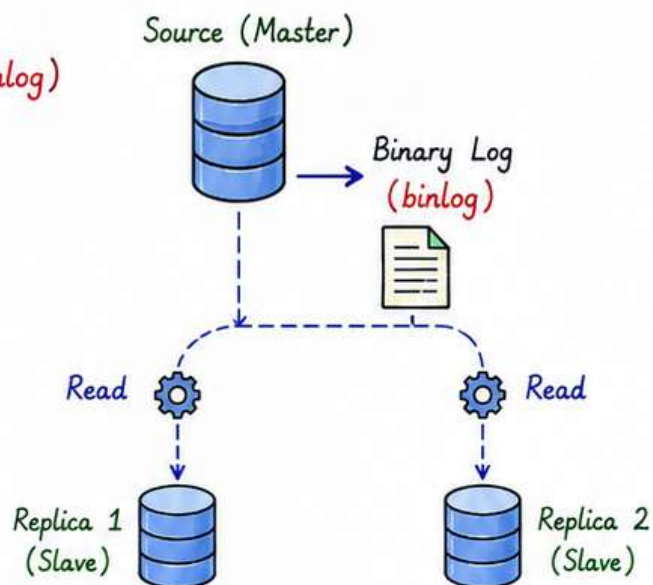


MySQL Replication helps in improving **high availability**, **backup** and **scalability** by maintaining multiple copies of data.



31.3 Binary Log Replication

- * MySQL replication works by using the **Binary Log (binlog)** on the Source server.
- * The binlog records all changes made to the data (INSERT, UPDATE, DELETE).
- * Replicas read these changes from the **binlog** and replay them on their own databases.
- * This ensures that replicas have the same data as the source.



Replication Flow:

1. Client sends write request to Source

2. Source executes the transaction and writes to Binary Log

3. Replicas (IO Thread) read events from Binary Log

4. Replicas (SQL Thread) apply events to their databases

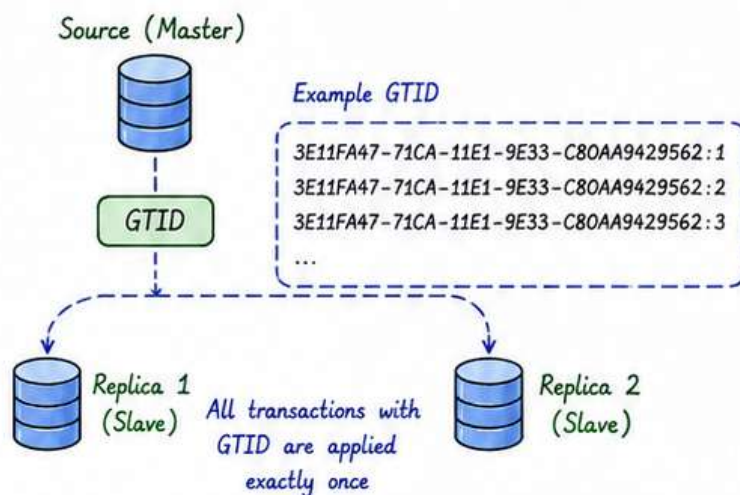
Key Points :

- ✓ Binary Log is the heart of MySQL replication.
- ✓ Replication is **asynchronous** by default.
- ✓ Network issues may cause **replication delay**.



31.4 GTID-Based Replication

- * GTID (Global Transaction ID) uniquely identifies each transaction.
- * It makes replication more **reliable** and **easier** to manage.
- * With GTID, you **don't need to track** binary log positions.
- * It prevents **data inconsistency** and **simplifies failover**.



Advantages of GTID :



Improves reliability of replication



Prevents data inconsistency



Simplifies failover and recovery



Easier to manage large environments



Reduces manual intervention

Key Takeaway :



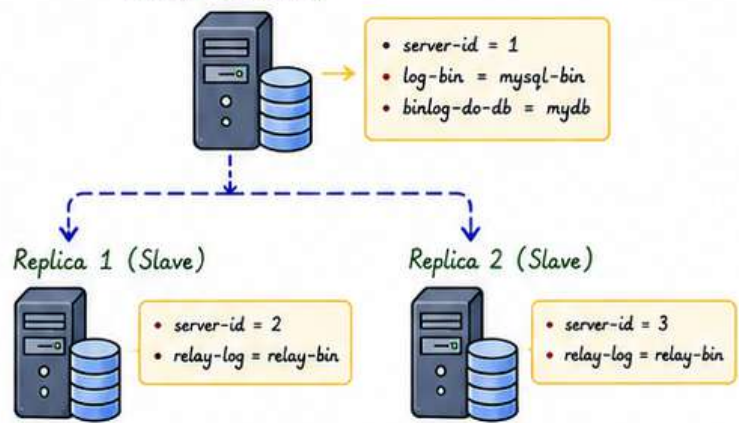
GTID-based replication enhances **reliability**, avoids **missing transactions** and makes MySQL replication **easier** to monitor and manage.



31.5 Replication Configuration

- * Replication requires configuration on both **Source (Master)** and **Replica (Slave)** servers.
- * We need to enable **binary logging** on the Source and configure the Replica to read from the Source.
- * Each server must have a unique **server-id**.
- * Replicas connect to the Source using **IP address, port, username** and **password**.

Source (Master)



Steps to Configure Replication:

1. Enable Binary Log on Source.
2. Create Replication User on Source.
3. Get Binary Log File and Position from Source.
4. Configure Replica to connect to Source.
5. Start Replication on Replica.

Example Configuration:

On Source (Master)

```
-- Enable Binary Log
log-bin = mysql-bin
server-id = 1

-- Create Replication User
CREATE USER 'repl_user'@'%' IDENTIFIED BY 'repl_pass';
GRANT REPLICATION SLAVE ON *.* TO 'repl_user'@'%';
FLUSH PRIVILEGES;

-- Get Binary Log File and Position
SHOW MASTER STATUS;
```

On Replica (Slave)

```
-- Set unique server-id
server-id = 2

-- Configure replication
CHANGE REPLICATION SOURCE TO
SOURCE_HOST='source_ip',
SOURCE_USER='repl_user',
SOURCE_PASSWORD='repl_pass',
SOURCE_LOG_FILE='mysql-bin.000001',
SOURCE_LOG_POS=154;

-- Start replication
START REPLICA;
```

Key Points:

- ✓ Source writes changes to binary log.
- ✓ Replica reads the binary log and applies changes.
- ✓ All servers must have unique **server-id**.



31.6 Monitoring Replication

- * Monitoring replication helps to ensure that **Replicas** are running correctly and in sync with the **Source**.
- * We can check replication status, **lag** and **errors**.
- * Regular monitoring helps in early detection of issues.

Important Parameters:

Parameter	Description
Replica_IO_Running	Shows if Replica is connecting to Source.
Replica_SQL_Running	Shows if Replica is applying events.
Seconds_Behind_Source	Replication delay in seconds.
Last_IO_Error	Last error while reading from Source.
Last_SQL_Error	Last error while applying events.

Key Takeaway:



Monitoring replication ensures **data consistency**, helps in quick issue resolution and maintains **high availability** of MySQL databases.



Replication Status Check:

```
SHOW REPLICA STATUS\G
```



Healthy Replication:

- ✓ Replica_IO_Running: Yes
- ✓ Replica_SQL_Running: Yes
- ✓ Seconds_Behind_Source: 0 (or low)



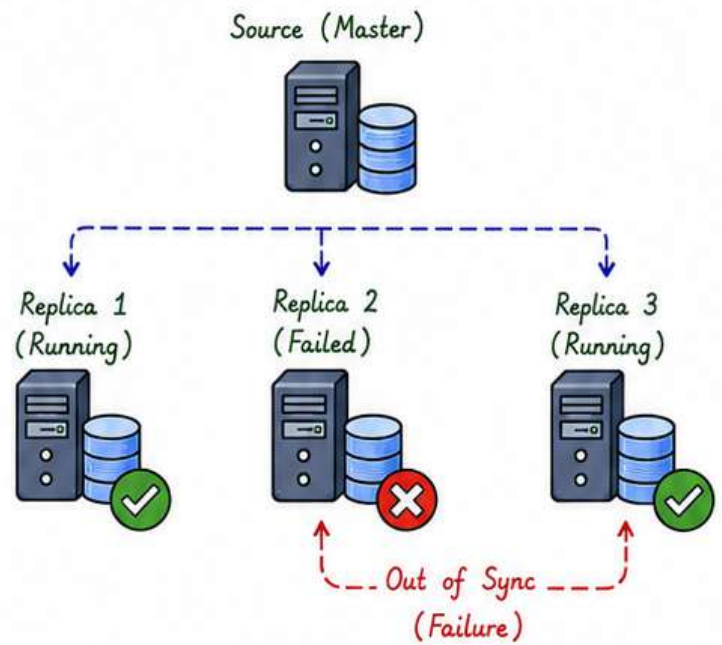
If there is an Issue:

- Check error messages.
- Restart replication.
- Verify network connectivity.
- Ensure binary log is available.



31.7 Replication Failures

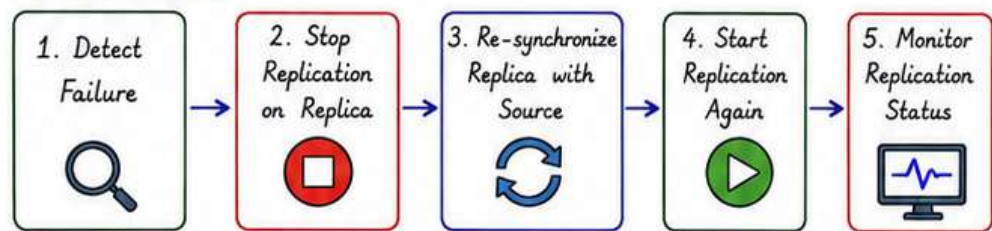
- * Replication failures can occur due to **network issues**, server crashes, or data inconsistencies.
- * If a **Replica stops**, it will fall behind the **Source**.
- * Failures can lead to **data loss** and **downtime**.
- * We need to detect failures and take corrective actions quickly.
- * MySQL provides tools to recover and re-synchronize **Replicas**.



Common Causes of Failure:

- Network connectivity problems.
- Replica server crash or shutdown.
- Disk full on Replica.
- Data inconsistency.
- Long running transactions on Source.

Recovery Steps:



Key Points:

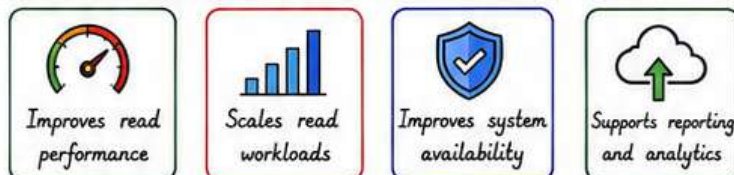
- ✓ Failures are normal; quick recovery is important.
- ✓ Regular monitoring helps in early detection.
- ✓ Keep backups to recover data if needed.



31.8 Read Replicas

- * Read Replicas are used to handle **read-heavy** workloads.
- * They offload read queries from the **Source (Master)**.
- * This improves **performance** and **scalability**.
- * Read Replicas **do not accept write operations**.
- * They replicate data asynchronously from the Source.

Benefits of Read Replicas:



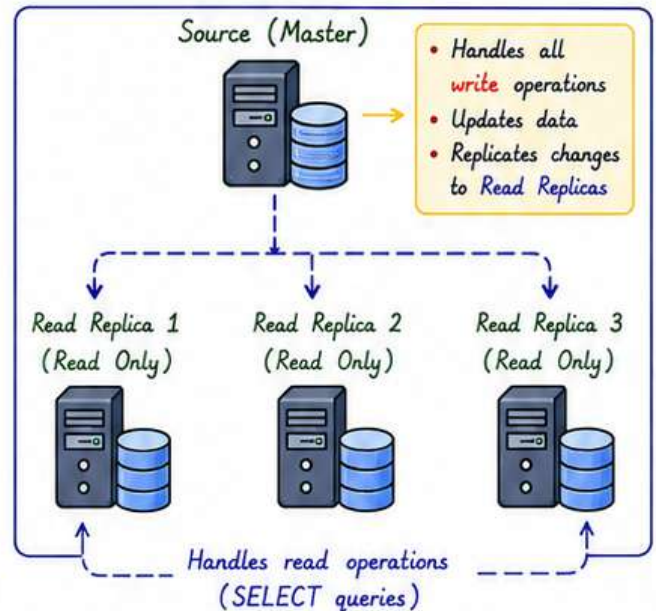
Use Cases:

- Handling large number of read queries.
- Reporting, analytics and business intelligence.
- Content delivery and caching.
- Geographically distributed read operations.

Key Takeaway:



Read Replicas improve **scalability** and **performance** by handling read queries, while the **Source** focuses on write operations and **data consistency**.



Key Points:

- ✓ Read Replicas are read-only.
- ✓ They help in scaling and performance.
- ✓ Writes should always go to the Source.

31.9 High Availability Concepts

- * High Availability (HA) ensures that the database system is available and accessible with **minimal downtime**.
- * It involves **redundancy**, **failover**, **load balancing** and **automated recovery**.
- * The goal is to minimize service disruption in case of failures.
- * MySQL provides multiple HA solutions and architectures.

Key HA Concepts:

Concept	Description
Redundancy	Duplicate components to avoid single points of failure.
Failover	Automatic switch to a standby system when primary fails.
Replication	Keep multiple copies of data across servers.
Monitoring	Continuously check system health and performance.
Backup & Recovery	Regular backups and quick recovery in disasters.
Load Balancing	Distribute client requests for better performance.
Automation	Automate failover, recovery and management tasks.

Common HA Solutions in MySQL:

- MySQL Replication with automatic failover.
- MySQL **InnoDB Cluster**.
- MySQL **Group Replication**.
- Using external tools like MHA (**Master High Availability**).
- Load balancing with ProxySQL or HAProxy.
- Cloud-based HA solutions (AWS RDS Multi-AZ, etc.).

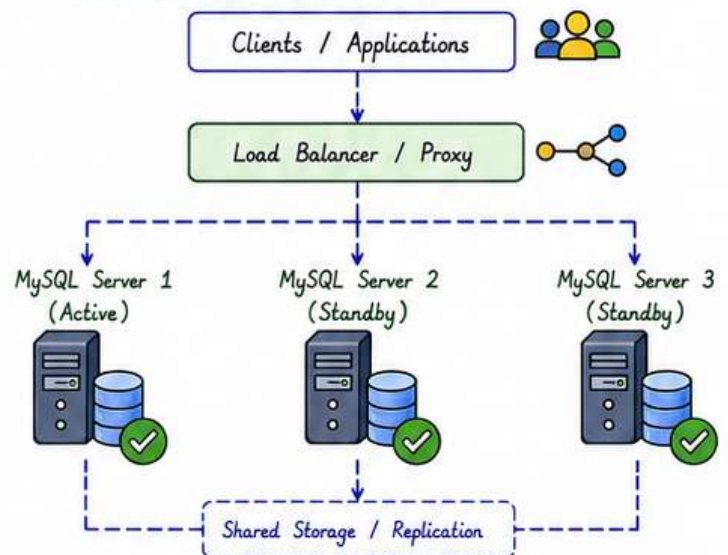
Key Takeaway:



High Availability ensures that MySQL databases remain **available**, **reliable** and **performant** by using redundancy, replication, failover and continuous monitoring.



HA Architecture Overview:



Benefits of High Availability:



Key Points:

- ✓ HA ensures continuous database availability.
- ✓ Use redundancy and failover to avoid downtime.
- ✓ Replications and clustering are key to HA.
- ✓ Monitoring and automation are critical for HA.



31.10 MySQL InnoDB Cluster

- * MySQL InnoDB Cluster is a fully managed HA solution built on **Group Replication**.
- * It provides **automatic failover**, **data consistency** and **scalability**.
- * It combines Group Replication, **MySQL Router** and **InnoDB Storage Engine**.
- * It is easy to deploy, manage and monitor.
- * It supports **read scaling** and **high availability**.

Key Features:



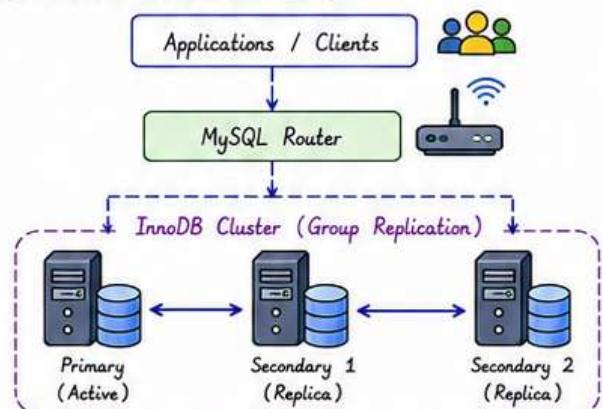
Key Takeaway:



MySQL InnoDB Cluster is a powerful, easy-to-manage HA solution that provides **automatic failover**, **data consistency**, **scalability** and **high availability** for MySQL databases.



InnoDB Cluster Architecture:



Key Points:

- ✓ InnoDB Cluster provides built-in HA.
- ✓ It uses Group Replication for data consistency.
- ✓ MySQL Router manages client connections.
- ✓ Automatic failover ensures minimal downtime.



31.11 MySQL Router

- * MySQL Router is a lightweight **middleware** that routes client connections to MySQL servers.
- * It is part of the **MySQL InnoDB Cluster**.
- * It provides **intelligent routing**, **load balancing** and **high availability**.
- * It automatically detects the state of cluster members.
- * It can handle **read/write splitting**.

Key Features :



Key Points :

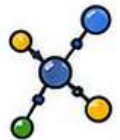
- ✓ MySQL Router improves performance and scalability.
- ✓ It provides high availability and automatic failover.
- ✓ It simplifies application connection management.
- ✓ It works seamlessly with InnoDB Cluster.



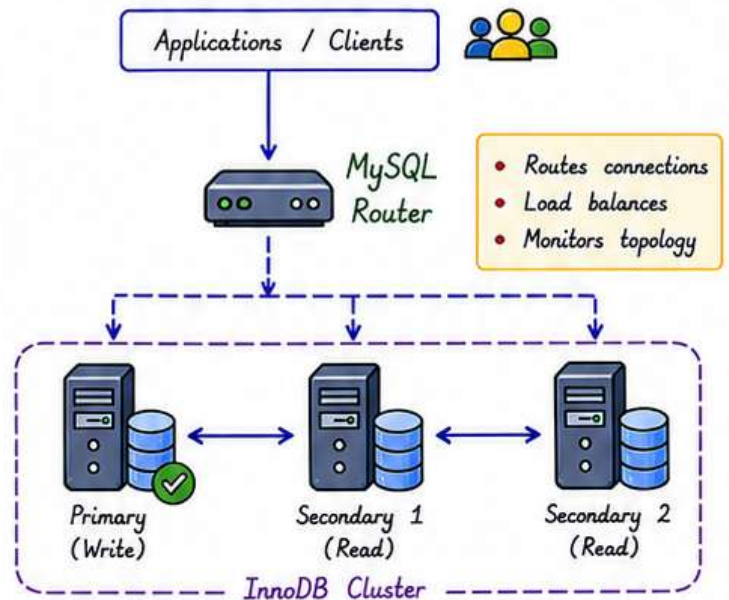
Key Takeaway :



MySQL Router acts as the connection layer between applications and MySQL servers, providing **intelligent routing**, **load balancing** and **high availability**.



MySQL Router Architecture :



Routing Modes :

Mode	Description
Read/Write Splitting	Write requests go to Primary, read requests are load balanced across Secondaries.
Read-Only	All requests are routed to Secondaries.
Read-Write	All requests are routed to Primary.

31.12 MySQL Shell

- * MySQL Shell is a modern, **JavaScript** and **Python** based client for MySQL.
- * It provides an **enhanced interface** for administering MySQL servers and clusters.
- * It supports both **SQL** and **scripting**.
- * It is used for **administration**, **automation** and **development** tasks.
- * It works with **InnoDB Cluster**, **MySQL Router** and standalone MySQL servers.

Common Use Cases :

- Managing InnoDB Clusters.
- User and privilege management.
- Backup and restore operations.
- Monitoring server status.
- Automating administrative tasks.

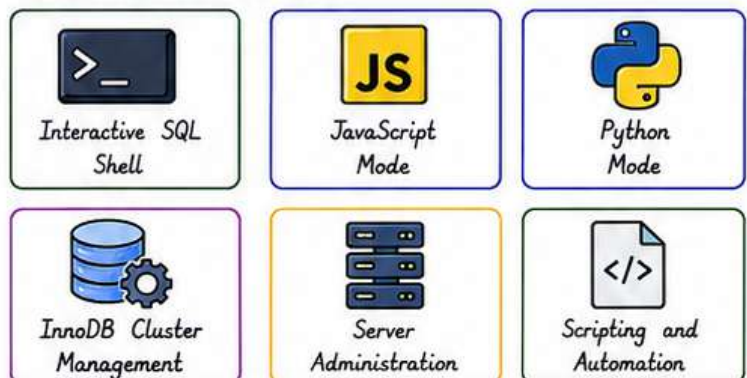
Key Takeaway :



MySQL Shell provides a modern, feature-rich interface for managing MySQL servers, clusters and automation using **SQL**, **JavaScript** and **Python**.



MySQL Shell Capabilities :



Key Points :

- ✓ MySQL Shell is powerful and versatile.
- ✓ Supports SQL, JavaScript and Python.
- ✓ Simplifies cluster management and administration.
- ✓ Automates many MySQL tasks.

