

Chapter 30: MySQL Administration

30.1 MySQL Server Configuration

- * The MySQL server can be **customized** using configuration settings.
- * Configuration controls the **behavior**, **performance** and **security** of the server.
- * Proper configuration ensures **stability**, **efficiency** and **reliability**.
- * MySQL reads configuration settings from option files at startup.

Key Features :

- ✓ Control memory usage, connections and caching.
- ✓ Manage storage engines and data directories.
- ✓ Enable logging, replication and security options.
- ✓ Tune performance for different workloads.

Common Configuration Areas :

Area	Description
Server Settings	Port, connections, timeouts, character set, etc.
Memory Settings	Buffer size, cache size, sort buffer, etc.
Storage Settings	Data directory, log directory, temp directory, etc.
Logging Settings	Error log, general log, slow query log, etc.
Replication Settings	Binary log, server-id, relay log, etc.
Security Settings	User authentication, access control, SSL, etc.

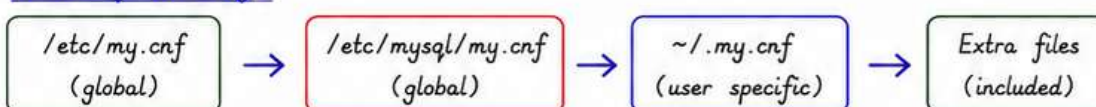
30.2 Configuration Files

- * MySQL reads configuration options from option (**my.cnf**) files.
- * The **location** of these files depends on the operating system.
- * You can have **multiple** option files; MySQL reads them in a **specific order**.

Default Locations of Option Files :

Operating System	Default Location
Linux / Unix	/etc/my.cnf, /etc/mysql/my.cnf, ~/.my.cnf
Windows	C:\ProgramData\MySQL\MySQL Server X.X\my.ini C:\my.ini (if exists)
macOS	/etc/my.cnf, /usr/local/etc/my.cnf
All Systems	~/.my.cnf (user specific options)

Order of Reading :



Key Points :

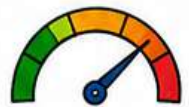
- ✓ Changes in configuration files require server **restart** to take effect.
- ✓ You can override settings in later files.
- ✓ Use '**my_print_defaults**' command to view loaded configuration.
- ✓ Always keep a backup before modifying configuration files.



Configuration



MySQL Server



Better Performance



Security



Config File



File Location



Settings



Tip

30.3 Server Variables

- * Server variables control the **behavior** of MySQL server.
- * They can be **dynamic** (change at runtime) or **static** (require restart).
- * Variables can be **Global** (affects entire server) or **Session** (affects current session).

Types of Server Variables :

Type	Description
Global Variables	Affect the entire MySQL server instance. Changed value affects new connections.
Session Variables	Affect only the current session . Changed value affects only current connection.
Dynamic Variables	Can be changed at runtime . No server restart required.
Static Variables	Can be changed only in configuration files . Require server restart to take effect.

Examples :

```
-- View all server variables
SHOW VARIABLES;

-- View a specific variable
SHOW VARIABLES LIKE 'max_connections';

-- View global value
SHOW GLOBAL VARIABLES
    LIKE 'max_connections';

-- View session value
SHOW SESSION VARIABLES
    LIKE 'max_connections';

-- Set global variable (dynamic)
SET GLOBAL max_connections = 500;

-- Set session variable
SET SESSION sql_mode = 'STRICT_ALL_TABLES';
```

Key Points :

- ✓ Use **SHOW VARIABLES** to view variables.
- ✓ Use **SET GLOBAL** to change global dynamic variables.
- ✓ Use **SET SESSION** to change session variables.
- ✓ Static variables must be changed in configuration files.
- ✓ Changes to global variables do not affect current sessions.



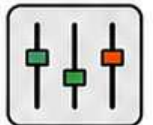
Server Variable



Global



Session



Configuration



MySQL Server

30.4 Status Variables

- * Status variables provide **information** about the current server activity and performance.
- * They are **read-only** and cannot be set manually.
- * Useful for **monitoring** MySQL server.

Examples of Status Variables :

Variable Name	Description
Uptime	Time since the server started (in seconds).
Threads_connected	Current number of open connections.
Threads_running	Currently executing threads.
Questions	Total number of queries executed.
Slow_queries	Number of slow queries.
Bytes_sent	Total number of bytes sent to clients.
Bytes_received	Total number of bytes received from clients.
Innodb_buffer_pool_reads	Number of physical reads from disk (InnoDB).

Examples :

```
-- Show all status variables
SHOW STATUS;

-- Show a specific status variable
SHOW STATUS LIKE 'Uptime';

-- Show multiple status variables
SHOW STATUS LIKE 'Threads_%';

-- Check slow queries count
SHOW STATUS LIKE 'Slow_queries';
```

Key Points :

- ✓ Use **SHOW STATUS** to monitor server activity.
- ✓ Status variables help in **performance tuning** and **troubleshooting**.
- ✓ Values are reset when the server restarts.
- ✓ Monitor key metrics like **uptime**, **threads**, **queries** and **slow_queries**.



Status Info



Monitoring



Performance



Analysis

30.5 SHOW VARIABLES

- * `SHOW VARIABLES` is used to display the **current value** of MySQL server variables.
- * Variables can be **Global** or **Session** scope.
- * Helps in checking and tuning server configuration.
- * Useful for database administrators and developers.

Syntax :

```
SHOW VARIABLES;           -- Show all variables
SHOW VARIABLES LIKE 'pattern'; -- Show variables matching pattern
SHOW GLOBAL VARIABLES;    -- Show all global variables
SHOW SESSION VARIABLES;   -- Show all session variables
```

Examples :

```
-- Show all variables
SHOW VARIABLES;
-- Show variable matching pattern
SHOW VARIABLES LIKE 'max_%';
-- Show a specific variable
SHOW VARIABLES LIKE 'max_connections';
-- Show global variables
SHOW GLOBAL VARIABLES;
-- Show session variables
SHOW SESSION VARIABLES;
```

Common Variables and Description :

Variable Name	Description
<code>max_connections</code>	Maximum number of client connections allowed.
<code>innodb_buffer_pool_size</code>	Size of the buffer pool for InnoDB.
<code>query_cache_size</code>	Size of the query cache.
<code>sql_mode</code>	SQL mode for parsing and validation.
<code>wait_timeout</code>	Time in seconds to wait for activity on a non-interactive connection.
<code>default_storage_engine</code>	Default storage engine for new tables.

Key Points :

- ✓ Use **SHOW VARIABLES** to view current configuration values.
- ✓ Use **LIKE** clause to filter specific variables.
- ✓ **Global** variables affect the entire server; **Session** variables affect only current session.
- ✓ Variable changes may require **server restart** (if static) to take effect.

30.6 SHOW STATUS

- * `SHOW STATUS` is used to display the current **server status information**.
- * Status values are **read-only**.
- * Helps in **monitoring performance** and server activity.
- * Useful for troubleshooting and performance tuning.

Syntax :

```
SHOW STATUS;           -- Show all status variables
SHOW STATUS LIKE 'pattern'; -- Show status matching pattern
```

Examples :

```
-- Show all status variables
SHOW STATUS;
-- Show specific status variable
SHOW STATUS LIKE 'Uptime';
-- Show status variables matching pattern
SHOW STATUS LIKE 'Threads_%';
```

Common Status Variables and Description :

Status Variable	Description
<code>Uptime</code>	Time since the server started (in seconds).
<code>Threads_connected</code>	Current number of open connections.
<code>Threads_running</code>	Number of threads that are not sleeping.
<code>Queries</code>	Total number of queries executed.
<code>Slow_queries</code>	Number of slow queries.
<code>Bytes_sent</code>	Total number of bytes sent to clients.
<code>Bytes_received</code>	Total number of bytes received from clients.
<code>Innodb_rows_read</code>	Total number of rows read from InnoDB.

Key Points :

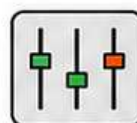
- ✓ Use **SHOW STATUS** to monitor server activity and performance.
- ✓ Values help in identifying bottlenecks and issues.
- ✓ Regularly monitor key metrics like **Uptime**, **Threads_running**, **Slow_queries**.
- ✓ Status values are reset when the server restarts.



Show Variables



Server Settings



Configuration



Inspect



Tip



Show Status



Monitor Performance



Troubleshoot



Optimize

30.7 MySQL Logs

- * MySQL maintains **different types of logs** to record events and activities.
- * Logs help in **debugging, performance tuning, recovery and auditing**.
- * You can enable or disable logs based on requirements.

Types of MySQL Logs :

Log Type	Description	Purpose
Error Log	Records server errors, warnings and startup/shutdown messages.	Troubleshooting and monitoring server issues.
General Query Log	Records all client requests.	Debugging and auditing.
Slow Query Log	Records queries that take more than a specified time.	Performance tuning.
Binary Log (Binlog)	Records changes to data.	Replication and point-in-time recovery.
Relay Log	Records events received from master in replication.	Replication (used by slaves).
Redo Log	Records changes in InnoDB before they are written to disk.	Crash recovery and durability.
Undo Log	Records before-images of data changes in InnoDB.	Transaction rollback and MVCC.

Enable / Disable Logs (Examples) :

```
-- Enable General Query Log
SET GLOBAL general_log = 'ON';
-- Enable Slow Query Log
SET GLOBAL slow_query_log = 'ON';
SET GLOBAL long_query_time = 2; -- in seconds
-- Enable Binary Log
SET GLOBAL log_bin = 'ON';
```

View Log Status (Examples) :

```
SHOW VARIABLES LIKE 'general_log';
SHOW VARIABLES LIKE 'slow_query_log';
SHOW VARIABLES LIKE 'log_bin';
SHOW VARIABLES LIKE 'log_error';
```

How to View Logs Location

```
SHOW VARIABLES LIKE 'log_error';
-- Error log file

SHOW VARIABLES LIKE 'general_log%';
-- General query log settings

SHOW VARIABLES LIKE 'slow_query_log%';
-- Slow query log settings

SHOW VARIABLES LIKE 'log_bin%';
-- Binary log settings
```

Key Points :

- ✓ Logs are essential for monitoring and troubleshooting.
- ✓ Enable only required logs to avoid performance impact.
- ✓ Regularly monitor and rotate log files.
- ✓ Always keep sufficient disk space for logs.



MySQL Logs



Monitor



Performance



Server



Audit



Tip

30.8 Error Log

- * The Error Log records **errors, warnings and informational messages**.
- * It is the most important log for troubleshooting MySQL server issues.
- * It is enabled by default.

Contents of Error Log :

Information	Details
Startup Messages	MySQL server start, version, OS details etc.
Shutdown Messages	MySQL server shutdown information.
Error Messages	Syntax errors, connection errors, access denied, etc.
Warning Messages	Potential issues, deprecated features, slow operations etc.
InnoDB Messages	InnoDB initialization, recovery, checkpoint information.
Replication Messages	Replication errors and status.
Other Information	General server events and activities.

Common Error Log Uses :

- * Identify server startup or shutdown problems.
- * Find errors in queries and connections.
- * Check disk issues, permissions, and configuration problems.
- * Monitor replication and InnoDB status.

View Error Log Location :

```
SHOW VARIABLES LIKE 'log_error';
-- Returns the error log file path
```

Example Error Log Entry :

```
2025-05-10T10:15:23.456789Z 0 [Note] Starting MySQL
Community Server 8.0.36
2025-05-10T10:15:23.789012Z 1 [Note] InnoDB: Initialization
has started.
2025-05-10T10:15:24.123456Z 1 [Warning] CA certificate
ca.pem is self signed.
2025-05-10T10:16:10.987654Z 12 [ERROR] Access denied for
user 'root'@'localhost' (using password: YES)
2025-05-10T10:17:05.345678Z 15 [Note] Shutdown complete.
```

Key Points :

- ✓ Always check the error log first when troubleshooting.
- ✓ Regularly review the error log for warnings and errors.
- ✓ Ensure the log file does not grow beyond available disk space.
- ✓ You can change the log file location in my.cnf.



Error Log



Troubleshoot



Debug



Reliability



Tip

30.9 General Query Log

- * The General Query Log records **all client requests** and statements received by the MySQL server.
- * It is useful for **debugging, auditing** and monitoring.
- * The log includes **all queries**, regardless of execution time.



Query Log

How to Enable General Query Log :

```
-- Enable General Query Log
SET GLOBAL general_log = 'ON';
-- (Optional) Write log to a file
SET GLOBAL log_output = 'FILE';
SET GLOBAL general_log_file = 'C:/mysql/logs/general_query.log';
-- Disable General Query Log
SET GLOBAL general_log = 'OFF';
```

View General Query Log Location :

```
SHOW VARIABLES LIKE 'general_log'; -- Status (ON/OFF)
SHOW VARIABLES LIKE 'log_output'; -- Output (FILE/TABLE)
SHOW VARIABLES LIKE 'general_log_file'; -- Log file path
```



Monitor



Audit



Server



Tip

Key Points :

- ✓ Records every query and command received by the server.
- ✓ Useful for debugging, auditing and tracking application behavior.
- ✓ Can generate a large log file, may impact performance.
- ✓ Disable after use in production environments.

Example - General Query Log Entries :

Time	Log Entry
2025-05-10T10:15:23.456789Z	Connect root@localhost on using TCP/IP
2025-05-10T10:15:23.456912Z	Query SHOW DATABASES;
2025-05-10T10:15:23.457123Z	Query USE college;
2025-05-10T10:15:23.457345Z	Query SELECT * FROM students;
2025-05-10T10:15:23.457789Z	Quit

30.10 Slow Query Log

- * The Slow Query Log records queries that take **longer** than a specified time (**long_query_time**) to execute.
- * It helps identify **performance bottlenecks** in the database.
- * Only long-running queries are logged.



Slow Queries

How to Enable Slow Query Log :

```
-- Enable Slow Query Log
SET GLOBAL slow_query_log = 'ON';
-- Set time (in seconds) above which queries are logged
SET GLOBAL long_query_time = 2; -- e.g., 2 seconds
-- Write log to file
SET GLOBAL log_output = 'FILE';
SET GLOBAL slow_query_log_file = 'C:/mysql/logs/slow_query.log';
-- Disable Slow Query Log
SET GLOBAL slow_query_log = 'OFF';
```

View Slow Query Log Location :

```
SHOW VARIABLES LIKE 'slow_query_log'; -- Status (ON/OFF)
SHOW VARIABLES LIKE 'long_query_time'; -- Time threshold
SHOW VARIABLES LIKE 'slow_query_log_file'; -- Log file path
```



Performance



Diagnose



Optimize



Tip

Notes :

- * `long_query_time = 0` → logs all queries.
- * Lower values increase log size.
- * Review slow queries regularly and optimize.

Example - Slow Query Log Entries :

Time	Query Time	Lock Time	Rows Sent	Rows Examined	SQL Query
2025-05-10T10:20:15.123456Z	3.215	0.000	1500	25000	SELECT * FROM orders WHERE status='Pending';
2025-05-10T10:21:47.654321Z	5.782	0.001	0	125000	SELECT * FROM logs WHERE created_at < '2024-01-01';

Key Points :

- ✓ Logs only queries that exceed the specified time limit.
- ✓ Helps in performance tuning and query optimization.
- ✓ Adjust `long_query_time` carefully.
- ✓ Large log files may impact disk space.
- ✓ Monitor and clean logs regularly.

30.11 Binary Log

- * The Binary Log (Binlog) records **all changes to data** (INSERT, UPDATE, DELETE, DDL).
- * It is essential for **replication, point-in-time recovery** and **auditing**.
- * Binlog is enabled by default in most MySQL installations.



Binary Log

How to Enable Binary Log :

```
-- Enable Binary Log
SET GLOBAL log_bin = 'ON';

-- Set binary log file name
SET GLOBAL log_bin_basename = 'mysql-bin';

-- Set binary log format (STATEMENT / ROW / MIXED)
SET GLOBAL binlog_format = 'ROW';

-- (Optional) Expire binary logs automatically
SET GLOBAL expire_logs_days = 7;

-- Disable Binary Log
SET GLOBAL log_bin = 'OFF';
```

Key Points :

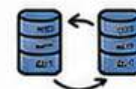
- ✓ Binlog is required for replication and backup.
- ✓ It records data-changing statements.
- ✓ Use ROW format for better data consistency.
- ✓ Keep binlog size under control and purge old logs.
- ✓ Ensure sufficient disk space for binlog files.

View Binary Log Status and Location :

```
SHOW VARIABLES LIKE 'log_bin';           -- Check if enabled
SHOW VARIABLES LIKE 'log_bin_basename';   -- Binlog file name
SHOW VARIABLES LIKE 'binlog_format';      -- Binlog format
SHOW BINARY LOGS;                         -- List all binlog files
SHOW MASTER STATUS;                      -- Current binlog status
SHOW VARIABLES LIKE 'expire_logs_days';   -- Auto purge setting
```

Example - Binary Log Files :

Log File Name	File Size	Last Modified
mysql-bin.000001	12.4 MB	2025-05-10 10:15:22
mysql-bin.000002	11.8 MB	2025-05-10 12:45:30
mysql-bin.000003	9.6 MB	2025-05-11 09:20:11



Replication



Point-in-Time Recovery



Audit



Server

30.12 MySQL Server Monitoring

- * Monitoring helps in tracking **performance, resource usage** and **server health**.
- * It helps in identifying **bottlenecks** and potential issues.
- * MySQL provides **built-in tools** and **performance schema** for monitoring.

Tools for Monitoring :

Tool / Method	Description
SHOW STATUS	View server status variables.
SHOW PROCESSLIST	View current connections and running queries.
Performance Schema	Detailed performance and instrumentation data.
MySQL Enterprise Monitor	Advanced monitoring and alerting tool.
Sys Schema	Provides useful views for monitoring.
Third-party Tools	Percona Monitoring, Zabbix, Prometheus, Grafana.

Common Monitoring Metrics :

- ✓ CPU Usage
- ✓ Memory Usage
- ✓ Disk Usage
- ✓ Connections
- ✓ Query Performance
- ✓ Slow Queries
- ✓ Threads
- ✓ Replication Status



Monitor



Performance



Health



Analytics



Alerts

Useful Commands for Monitoring :

```
SHOW STATUS;           -- All status variables
SHOW STATUS LIKE 'Threads%'; -- Thread related status
SHOW STATUS LIKE 'Com_%';  -- Command statistics
SHOW STATUS LIKE 'InnoDB%'; -- InnoDB status
SHOW PROCESSLIST;        -- Current running queries
SHOW VARIABLES LIKE 'max_connections'; -- Max connections
```

Key Points :

- ✓ Regular monitoring ensures better performance and reliability.
- ✓ Track key metrics and set alerts for critical issues.
- ✓ Use Performance Schema for detailed insights.
- ✓ Analyze slow queries and optimize your database.
- ✓ Monitoring helps in capacity planning and problem resolution.

Example - Key Metrics (Sample Output) :

Variable Name	Value	Description
Uptime	86400	Server running time (in seconds)
Threads_connected	35	Current open connections
Threads_running	3	Currently running threads
Questions	152345	Total queries executed
Slow_queries	12	Total slow queries
InnoDB_buffer_pool_reads	1234	Disk reads for InnoDB
Com_select	120345	Total SELECT commands

Tip :



Automate monitoring using tools like Prometheus + Grafana or Percona Monitoring and set email / Slack alerts.

30.13 Managing Connections

- * MySQL handles **multiple client connections**.
- * Proper connection management improves **performance** and **security**.
- * You can **view, control** and **limit** connections.



Connections

View Current Connections :

Command	Description
SHOW PROCESSLIST;	Shows all current connections and queries.
SHOW STATUS LIKE 'Threads_connected';	Number of currently open connections.
SHOW STATUS LIKE 'Max_used_connections';	Maximum number of connections used since server start.

Example - SHOW PROCESSLIST Output :

Id	User	Host	db	Command	Time	State	Info
5	root	localhost	NULL	Sleep	10	NULL	
8	root	localhost	testdb	Query	0	executing	SELECT * FROM users;
9	appuser	192.168.1.10	shop	Query	2	sorting result	SELECT * FROM orders;



Monitor



Manage

Control Connections :

```
-- Kill a connection by Id
KILL 8;

-- Kill all connections of a user
KILL USER 'appuser'@'%';

-- Set maximum allowed connections
SET GLOBAL max_connections = 200;

-- Set wait timeout (in seconds)
SET GLOBAL wait_timeout = 28800;
```

Important Status Variables :

Variable	Description
Threads_connected	Current number of open connections.
Threads_running	Number of threads that are not sleeping.
Max_used_connections	Maximum number of connections used.
Max_connections	Maximum allowed client connections.
Connection_errors_internal	Internal connection errors.
Aborted_connects	Failed connection attempts.



Security



Server

Key Points :

- ✓ Monitor connections regularly using **SHOW PROCESSLIST**.
- ✓ Set **max_connections** according to server capacity.
- ✓ Kill idle or long-running connections if required.
- ✓ Proper connection management improves **security** and **performance**.

30.14 MySQL Service Management

- * MySQL runs as a **service (daemon)** in the background.
- * Service management commands **vary by OS**.
- * Used to **start, stop, restart** and check **status** of MySQL server.



Service

Windows (Using Services or Command Prompt) : Linux (Using systemctl) :

```
-- Start MySQL service
net start MySQL80

-- Stop MySQL service
net stop MySQL80

-- Restart MySQL service
net stop MySQL80
net start MySQL80

-- Check MySQL service status
sc query MySQL80
```

```
-- Start MySQL service
sudo systemctl start mysql

-- Stop MySQL service
sudo systemctl stop mysql

-- Restart MySQL service
sudo systemctl restart mysql

-- Check MySQL service status
sudo systemctl status mysql

-- Enable MySQL service at boot
sudo systemctl enable mysql
```

macOS (Using brew services) :

```
-- Start MySQL service
brew services start mysql

-- Stop MySQL service
brew services stop mysql

-- Restart MySQL service
brew services restart mysql

-- Check MySQL service status
brew services list
```



Start / Stop



Restart



Status



Reliability

Check Service Status Examples :

OS	Command	Description
Windows	sc query MySQL80	Shows service status (RUNNING / STOPPED)
Linux	sudo systemctl status mysql	Shows detailed service status and logs
macOS	brew services list	Lists MySQL service and its status

Key Points :

- ✓ Ensure MySQL service is running before connecting.
- ✓ Enable service at boot for automatic start.
- ✓ Regularly check service status and restart if needed.
- ✓ Service management helps in **stability** and **high availability**.

Common Issues & Solutions :

- Service won't start → Check error log and configuration.
- Access denied → Run commands with administrator/root privileges.
- Port already in use → Change port in my.ini / my.cnf.
- Service not found → Check installation and service name.

