SQL DM for MySQL 8.9

Table of Contents

SQL DM for MySQL is a monitor and advisor application for MySQL database administrations.

It provides you with tools to manage more database servers, to tune your current database infrastructure, and to aid you in finding and fixing problems with your database applications before they develop into more serious issues or costly outages.

Review the release notes for SQL Diagnostic Manager for MySQL, to get a peek of the most recent release:

- Access the SQL Diagnostic Manager for MySQL Release Notes(see page 11)

Or simply, start with the configuration of your product Get Started[1].

1 http://wiki.idera.com/x/qAD3BQ

Access the SQL Diagnostic Manager for MySQL Release Notes

To get a glimpse into the newest features, fixed issues, and known issues in this SQL Diagnostic Manager for MySQL release, review the following sections of the Release Notes:

- See New Features[2] in this release
- Review Fixed Issues[3] included in this release
- Review Previous Features and Fixed Issues[4]
- See Known Issues[5]

New Features and Fixed Issues

SQL Diagnostic Manager for MySQL provides the following new features and fixed issues:

8.9.3 New Features

- SQL DM for MySQL shows a notification in the UI when it is unable to send an Alert message due to wrong settings.

8.9.2 New Features

- SQL DM for MySQL now can analyze audit logging in Aurora Server.

8.9 New Features

- SQL DM for MySQL includes a query load profiling to Query Analysis along with a new interface for Query Details to make it easier to analyze query performance over time.
- Introduced new APIs for updating user passwords and LDAP bind user password.
- This version includes several Enterprise security improvements:
    - Restrict access to individual custom dashboards and the ability to create new dashboards to newly created users.
    - Capability to hide literals in SQL statements,
    - Ability to remove default admin user, and rename the default Admin name.
    - Improved audit log filtering, adding and excluding filter.
    - Configurable SNMP trap format.

8.9.3 Fixed Issues

- SQL DM for MySQL now connects to MySQL using SSL encryption with caching_sha2_password authentication plugin.
- On Chrome browser, the data is automatically filled for server names, tags, and other filters if the Save Password is turned ON.
- SQL DM for MySQL validates the email address properly, and it allows saving multiple emails using space and comma separator.

---

2 http://wiki.idera.com/x/aQEGBg
3 http://wiki.idera.com/x/aQEGBg
4 http://wiki.idera.com/x/agEGBg
5 http://wiki.idera.com/x/awEGBg

8.9.2 Fixed Issues

- Google Cloud Server logs are getting analyzed properly.
- Starting with version 8.9.2, modified UDOs will display as *Resolved* when there are no new changes in future upgrades.
- Query Analyzer, Realtime, and Dashboard statistics extracted files, now collect correct time/date.
- General Log Export as CSV does not show undefined and NAN in the CSV file anymore.
- MySQL Versions prior to 5.6.2, now display accurate disk space.
- Show/Hide literals in Long running query notifications, are now available for all notification channels.
- When the query contains special characters, the exported CSV file now is formatted properly.
- Resolving Conflicted CSO's now works properly.
- SQL DM for MySQL now sends pagerduty notifications correctly when the length of the message is greater than 1024.
- Editing a LDAP Password now sets the new password accordingly.
- MariaDB roles are listed correctly.
- The values plotted in the Monitor trend charts, are now converted to bytes.
- The column QPS now sorts accurately.
- Total Time column in Query Analyzer page, now shows the value properly.

8.9 Fixed Issues

- The Threads page now correctly displays queries that contains these "<" or ">" characters.
- The Event Details window now correctly displays the event values that contains special characters.
- The SSL settings now correctly save the changes after editing the servers.
- The client user filter option in the Sniffer settings now works correctly.
- The Locked and Locking Queries in Real-time are now available for MySQL 8.0+ servers.
- The Connection Type is no longer changing from SSL to Direct when editing a server using API.
- SQL DM for MySQL is no longer crashing when passing some arguments in the start command.
- SQL DM for MySQL now correctly exports CSV reports when applying filters in the Query Analyzer, Dashboard, and Real-time.
- All chart series are visible for MySQL 8.0+ version in the Dashboard and screens.

8.9.3 Miscellaneous

- MariaDB Connector is upgraded to 3.1.11.

8.9.2 Miscellaneous

- JQuery library is upgraded to 3.5.0.

Known Issues

IDERA strives to ensure our products provide quality solutions for your SQL Server needs. If you need further assistance with any issue, please contact Customer Support Portal(see page 12).

In this section you can find information of all the Known Issues existing for this version.

8.9 Known Issues

There are no Known Issues reported for this version.

Get started

SQL Diagnostic Manager for MySQL is a monitor and advisor application for MySQL database administrations. It provides you with tools to manage more database servers, to tune your current database infrastructure, and to aid you in finding and fixing problems with your database applications before they develop into more serious issues or costly outages.

With SQL Diagnostic Manager for MySQL, you can proactively monitor enterprise database environments and obtain expert advice so that even administrators new to MySQL can tighten security, optimize performance and reduce downtime of their MySQL powered systems.

Get Started with SQL Diagnostic Manager for MySQL

Introduction

SQL Diagnostic Manager for MySQL provides monitors and advisors for MySQL database administrators, helping you to manage more database servers, tune your database infrastructure, and identify any issues with your database applications. It monitors enterprise environments and provides expert advise so that even someone new to MySQL can tighten security, optimize performance, and reduce downtime of MySQL powered systems.

Features

In technical terms, SQL DM for MySQL is essentially a web server with a very low resource footprint. As such, it has much in common with popular web servers such as Apache, Nginx, IIS, Lighttpd and so on, but there are some important differences.

In order to access SQL DM for MySQL, you need a web browser that is capable of handling AJAX. This has been tested on Google Chrome, Internet Explorer (versions 11.0.96 and later), Mozilla Firefox and related browsers, and all recent releases of the Opera browser.

Monitoring Optimization

Where web servers are general purpose applications, SQL DM for MySQL is a more specialized server application. It is built for collecting and storing information on MySQL database servers and the systems that host them. You can not use it to host your own webpages as it is locked to only display those pages that the SQL DM for MySQL application was designed to serve.

SQL DM for MySQL uses the Asynchronous JavaScript and XML (AJAX) language to communicate with the server and generate pages divided into small, independent objects that it can then individually update as required. This reduces bandwidth consumption, given that you do not need to reload the entire page for updates and allows you to display data on as many MySQL servers as you want.

Each of the HTML objects in SQL DM for MySQL AJAX pages refresh automatically when new data becomes available. This refresh is independent for each MySQL server being displayed on the page. Refreshes happen at the server-level.

Unlimited Servers Side-by-Side

When using SQL DM for MySQL Enterprise, Charts show real-time graphs of all import metrics that provide a consolidated view into the availability and performance of all the MySQL servers across the enterprise. From these real-time charts, the MySQL database administrator can instantly see:

- The availability status of all MySQL servers,
- Import operating system metrics that may affect MySQL,
- Side-by-side comparison of similar servers,
- Which MySQL servers need attention, and
- Where and how they need to spend limited time.

Scripting with JavaScript

SQL DM for MySQL includes an embedded JavaScript engine, similar to most modern browsers. The entire SQL DM for MySQL application logic is handled through JavaScript objects, such as defining threshold values, calculating performance metrics and sending alerts. The complete source code for these JavaScript objects is available for customization, making them fully editable, extensible, and configurable as your use-cases require.

Agent-less Monitoring

Typically, monitoring and advisory tools use external PHP or Java agents to allow traditional web servers to connect to MySQL. Unlike these tools, SQL DM for MySQL is compiled with the MySQL client code through the C API, allowing to connect directly to MySQL database servers without needing agents.

This greatly simplifies SQL DM for MySQL deployments, given that you don not need to install and maintain agents on each MySQl host. Instead, SQL DM for MySQL uses the MySQL client to connect to and retrieve database information. To retrieve operating system-level metrics it uses SSH when connecting to Linux hosts.

You install SQL DM for MySQL on one host and that's it. It can retrieve whatever information it needs using the same sources and methods as a database administrator.

Built-in High Performance Database

Internally, SQL DM for MySQL manages the data it collects using a high-performance embedded database. Whatever server parameters it retrieves from MySQL servers are also stored in this database. Various methods in displaying counters or metrics are based on the data it provides.

Security and Authentication

The fact that SQL DM for MySQL is a dedicated and specialized web server also makes it very secure.

- **SQL DM for MySQL - by design is very secure:** It is practically immune to any kind of malicious code. Whether they be unwanted popups, advertisements, trojans, attempts at phishing, hijacking or the like. Any HTML pages with such content are not sent by SQL DM.
- **Password protection:** During install you (or your SysAdmin) provided a password. Your browser prompts you for this password whenever you connect to the SQL DM for MySQL service. You cannot access the stream of data from the SQL DM for MySQL service if you are not able to provide the correct password.
- **Additional security considerations:** Note that on the server the SQL DM for MySQL password is stored obfuscated in the MONyog.ini file . Also, the SQL DM for MySQL embedded database has stored credentials (user IDs and passwords for MySQL servers, SSH, and SMTP servers).

The SQL DM for MySQL authentication system to work, must have cookies enabled in the browser. A 'medium high' security setting are appropriate for most users. The exact setting (and how it is named) varies from browser to browser.

Architecture

In technical terms, SQL DM for MySQL is basically a very low footprint web server. As such it has much in common with popular web-servers like Apache, IIS, Lighttpd etc., but there are also important and significant differences.

SQL DM for MySQL is a dedicated and specialized server program for collecting and storing information concerning MySQL servers and the systems on which they are installed. You can not use it to display your own webpage. It is 'locked' to display the pages that SQL DM for MySQL program is designed for.

SQL DM for MySQL is basically a low footprint web server that is optimized for Monitoring MySQL servers.

SQL DM for MySQL uses state-of-the-art web technology

These pages use AJAX (Asynchronous JavaScript and XML) to communicate with SQL DM for MySQL server. With AJAX, it is possible to generate pages that are divided into small independent objects that are updated individually when required. This reduces the bandwidth consumption as compared to traditional method of reloading the whole page. And well-designed AJAX pages provide a much more pleasant user experience than traditional HTML pages. You can display and compare as many MySQL servers simultaneously as you want. It is handled by the browser itself. Also, you will need not reload the page to display the latest data. Each of the small HTML objects in SQL DMfor MySQL AJAX pages refresh automatically when new data are available. And this refresh is done independently for every MySQL server that is displayed. Actually, this refresh happens at the server level which means for "Server A" interval is 15 seconds to collect server data and for "Server B" it is 10 minutes, then the respective display of data for every server refreshes after that interval.

SQL DM for MySQL is configurable using the de-facto web scripting language: JavaScript

SQL DM for MySQL also includes an embedded JavaScript (JS) engine - not unlike most modern browsers. In fact, the entire SQL DM for MySQL "application logic" like defining threshold values, calculating performance metrics,

and sending alerts are defined as JS objects. The complete source code of these JS objects is available to the user for customization. The JavaScript objects are fully editable, extensible, and configurable.

SQL DM connects to MySQL directly: no JAVA, no PHP, no 'agents' install required

Also, unlike other web-servers SQL DM for MySQL has the MariaDB Connector/C 'compiled in'. It can connect to MySQL directly without depending on any sort of external code (PHP, JAVA or whatever) as other web-servers do. Also, it does not make use of any 'agents' on the computers where the MySQL servers are running. You install SQL DM for MySQL on one computer and that is it!

SQL DM for MySQL has a built-in high-performance database

Further! SQL DM for MySQL has a built-in high-performance embedded database. Whatever server parameters are retrieved from the MySQL servers are also stored in that database. Various ways to display 'counters' or 'metrics' based on these data are provided.

SQL DM for MySQL is compatible with all modern browsers

All you need, to view the results that SQL DM for MySQL throws up, is a web browser that is capable of handling AJAX. This has been tested on Google Chrome, Internet Explorer (versions 11.09 and later), Mozilla Firefox and related browsers, and all recent releases of the Opera browser.

SQL DM for MySQL Requirements

Consider the following requirements when installing SQL Diagnostic Manager for MySQL.

Type	Requirement
Operating System	Windows (32-bit): • Windows Server 2008 • Windows Server 2008 R2 • Windows Server 2012 • Windows Server 2012 R2 • Windows Server 2016 • Windows Vista • Windows 7 • Windows 8 • Windows 8.1 Linux (64-bit): • All distributions • Kernel 2.6.32

Type	Requirement
Database Platforms	Oracle MySQL Enterprise 5.1, 5.5, 5.6, 5.7, 5.8 MySQL Community Server 5.1, 5.5, 5.6, 5.7, 8.0 MariaDB (5.1, 5.2, 5.3) 5.5, 10, 10.1, 10.2, 10.3 Percona Server for MySQL 5.1, 5.5, 5.6, 5.7 Galera Cluster for MySQL 3 MariaDB Galera Cluster 5.5, 10.0 Amazon RDS for MySQL, Amazon RDS for MariaDB, Amazon for Amazon Aurora Azure MySQL Google Cloud SQL For MySQL Oracle MySQL Cloud Service

SQL Diagnostic Manager for MySQL hardware sizing guidelines

The following guidelines provide an estimation of the hardware resources required to deploy SQL Diagnostic Manager for MySQL depending on the number of servers you want to monitor.

Small and medium size deployments for less than 500 MySQL servers

Type	Requirement
RAM	4GB
CPU	2 Cores
Storage Space	5GB per monitored MySQL Server

Large size deployments for more than 500 MySQL Servers

Type	Requirement
RAM	32GB
CPU	16 Cores
Storage Space	1.5 TB

Installation

SQL DM for MySQL is supported on Microsoft Windows and Linux operating systems. Once installed, you can access it from any AJAX-supported web browser, including those available on mobile devices.

Once you have SQL DM for MySQL installed, it can connect to and collect data from MySQL database servers running on any platform. When connecting to a MySQL server running on Windows, SQL DM for MySQL can use the Windows Management Instrumentation to retrieve additional information. When connecting to a server running on Linux, it can do the same using SSH.

SQL DM for MySQL can only retrieve system information from Linux servers.

Downloading SQL Diagnostic Manager for MySQL

Currently, SQL DM for MySQL is supported on Microsoft Windows 2008 and later, and on Linux operating systems. It does not support specific distributions of Linux, but provides a package for those based on RPM standards.

Getting SQL Diagnostic Manager for MySQL

To obtain the download links for SQL DM for MySQL, you need to purchase a license or request a free trial from IDERA.

Once you have purchased the license or requested a free trial, you receive links to the installation media by email. For more information on the different levels of support see Pricing(see page 19) or Trial Center(see page 19).

Choosing the Correct Download

When you receive the download links, there are several available for you to use. The specific link you need depends on the system you want to run SQL DM for MySQL on, both in terms of hardware architecture and the operating system.

- Windows

  If you want to use SQL DM for MySQL on a Windows Server installation, there is only one link available to you. Select the **Executable (.exe)** link and follow the instructions on Windows Installation to install SQL DM for MySQL. For Linux installations, it is a little more complicated.

- Linux

  There are four download links available for Linux installations, divided by distribution and hardware architecture. Select the download link that matches your installation. Currently, IDERA only provides RPM-based packages. In the event that you use Fedora, CentOS, Red Hat Enterprise Linux, openSUSE, SUSE Linux Enterprise Server, or a similar RPM-based Linux distribution, select the **RPM** download link for your hardware. For all other distributions, select the **TAR** download link.

SQL DM for MySQL runs on both 32-bit and 64-bit operating systems. In the event that you want to install SQL DM on a server and do not know which architecture it uses, you can find out with the `uname` command:

```
$ uname -a
Linux actual 3.16.0-4-amd64 #1 SMP Debian 3.16.43-2(2017-04-30) x86_64 GNU/Linux
```

Reference to `amd64` or `x86_64` indicate a 64-bit operating system. `i386` or `i686` indicate 32-bit.

Installing SQL Diagnostic Manager for MySQL

Once you have downloaded the package for your system, you can install it on your system. The specific installation directions depends on the operating system you want to use to host SQL DM for MySQL:

- Linux(see page 20)
- Windows(see page 28)

Linux

SQL DM for MySQL runs as a web application on a server, from which it connects to and monitors MySQL database servers. The MySQL servers can run on any operating system. SQL DM for MySQL can run on Linux <install-windows> or any distribution of the Linux operating system.

Installing SQL DM for MySQL

Webyog provides two types of installation media for Linux operating systems: RPM and TAR. Each is available for 32-bit and 64-bit architectures. For RPM-based Linux distributions, you can install SQL DM for MySQL through the package manager or the `rpm` command. For all other distributions, you can manually install SQL DM for MySQL from a tarball.

Unlike the Windows installation, neither of these methods prompts the user to supply a default password or port number. By default, SQL DM for MySQL runs on port 5555, with the user admin and a blank password. You can change them through the web UI later.

Installation

The link downloads an `.rpm` package. The filename pattern is `IderaSQLdmforMySQL-<version>.<architecture>.rpm`. You can install SQL DM for MySQL by calling the package manager on this file.

- For Fedora and newer Red Hat-based distros, you can install SQL DM for MySQL with DNF:

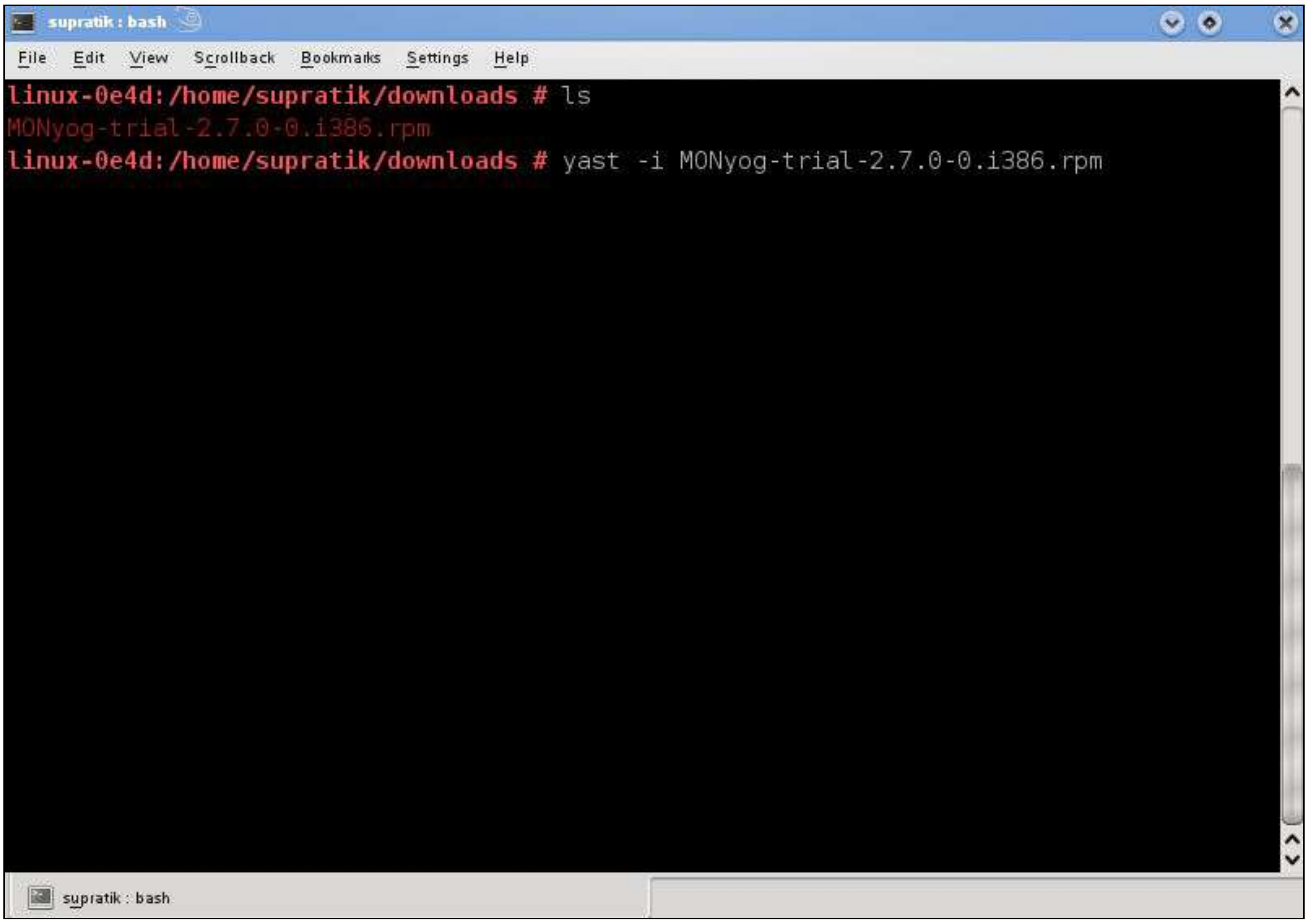
```
# dnf install /path/to/IderaSQLdmforMySQL-*.rpm
```

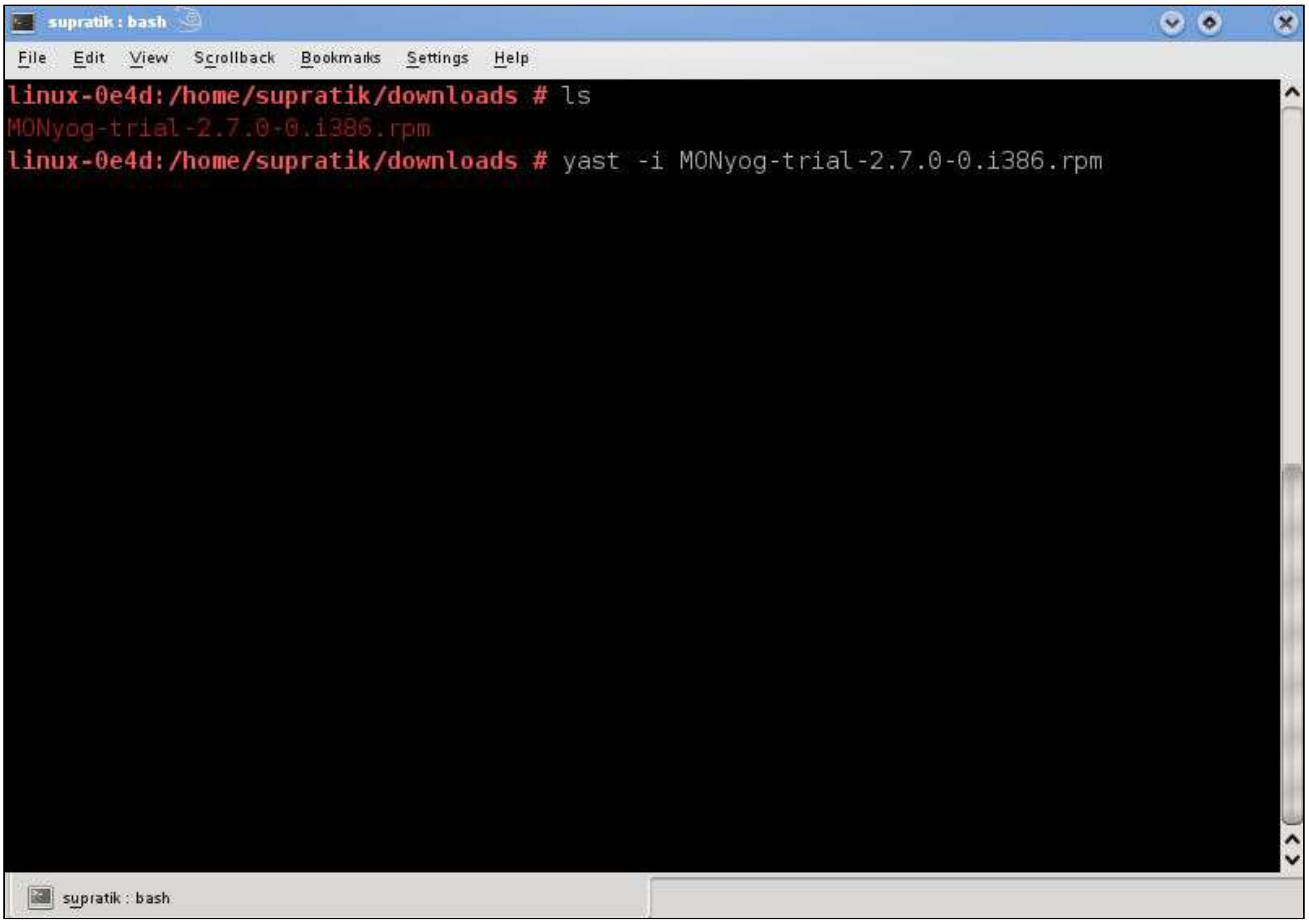
- For older Red Hat-based distros, you can use YUM:

```
# yum install /path/to/IderaSQLdmforMySQL-*.rpm
```

- For openSUSE and SUSE-based distros, use YaST:

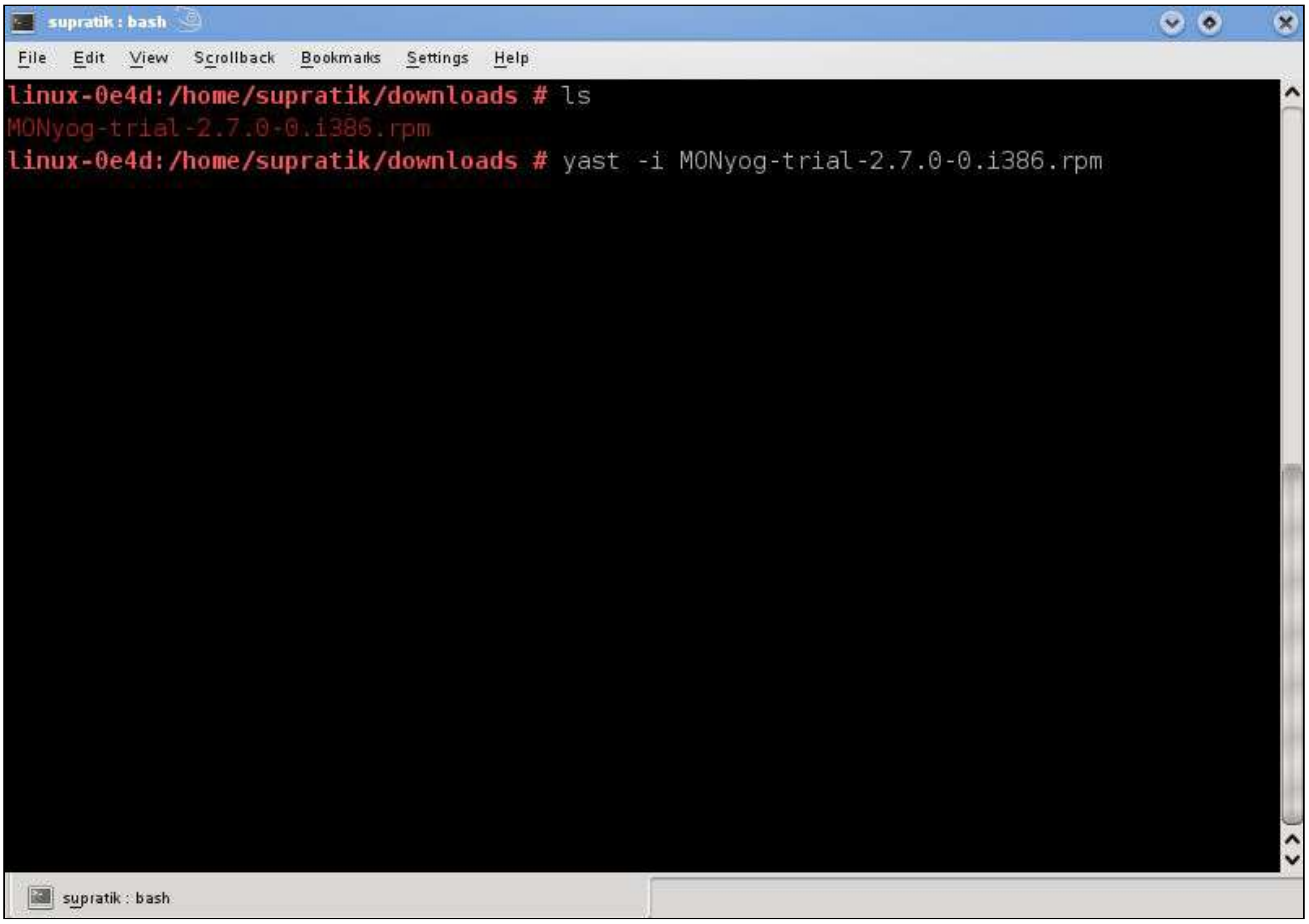
```
# yast -i /path/to/IderaSQLdmforMySQL-*.rpm
```


supratik : bash
File Edit View Scrollback Bookmarks Settings Help
linux-0e4d:/home/supratik/downloads # ls
MONyog-trial-2.7.0-0.i386.rpm
linux-0e4d:/home/supratik/downloads # yast -i MONyog-trial-2.7.0-0.i386.rpm
supratik : bash



- For any RPM-based distro, you can also use RPM itself:

```
# rpm -ivh /path/to/IderaSQLdmforMySQL-*.rpm
```



Whichever package manager you use, it installs SQL DM for MySQL in the `/usr/local/MONyog/` directory. You can now begin configuring and using SQL DM for MySQL.

Start/Stop

- In Redhat and Fedora systems if SQL DM for MySQL is installed from the RPM package, the daemon script 'MONyogd' can be used to start/stop the server. This script is in"/etc/init.d/" directory.

You can use one of the following command to start SQL DM for MySQL:

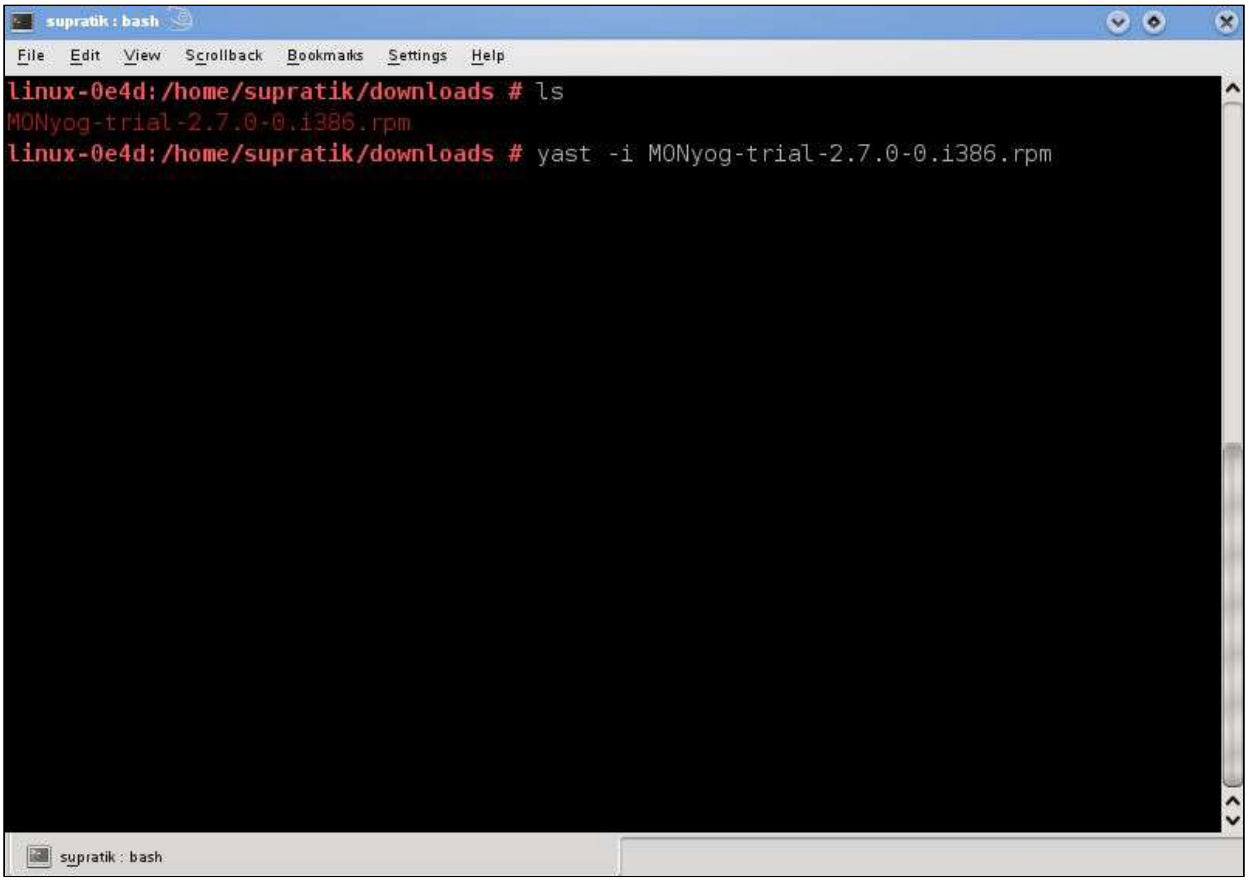
```
# service MONyogd start
```

or,

```
# /etc/init.d/MONyogd start
```

- To start the MONyog(SQL DM for MySQL) service on a system that uses systemd, run this command instead:

```
# systemctl start MONyog
```



similarly for stopping:

```
# service MONyogd stop
```

or,

```
# /etc/init.d/MONyogd stop
```

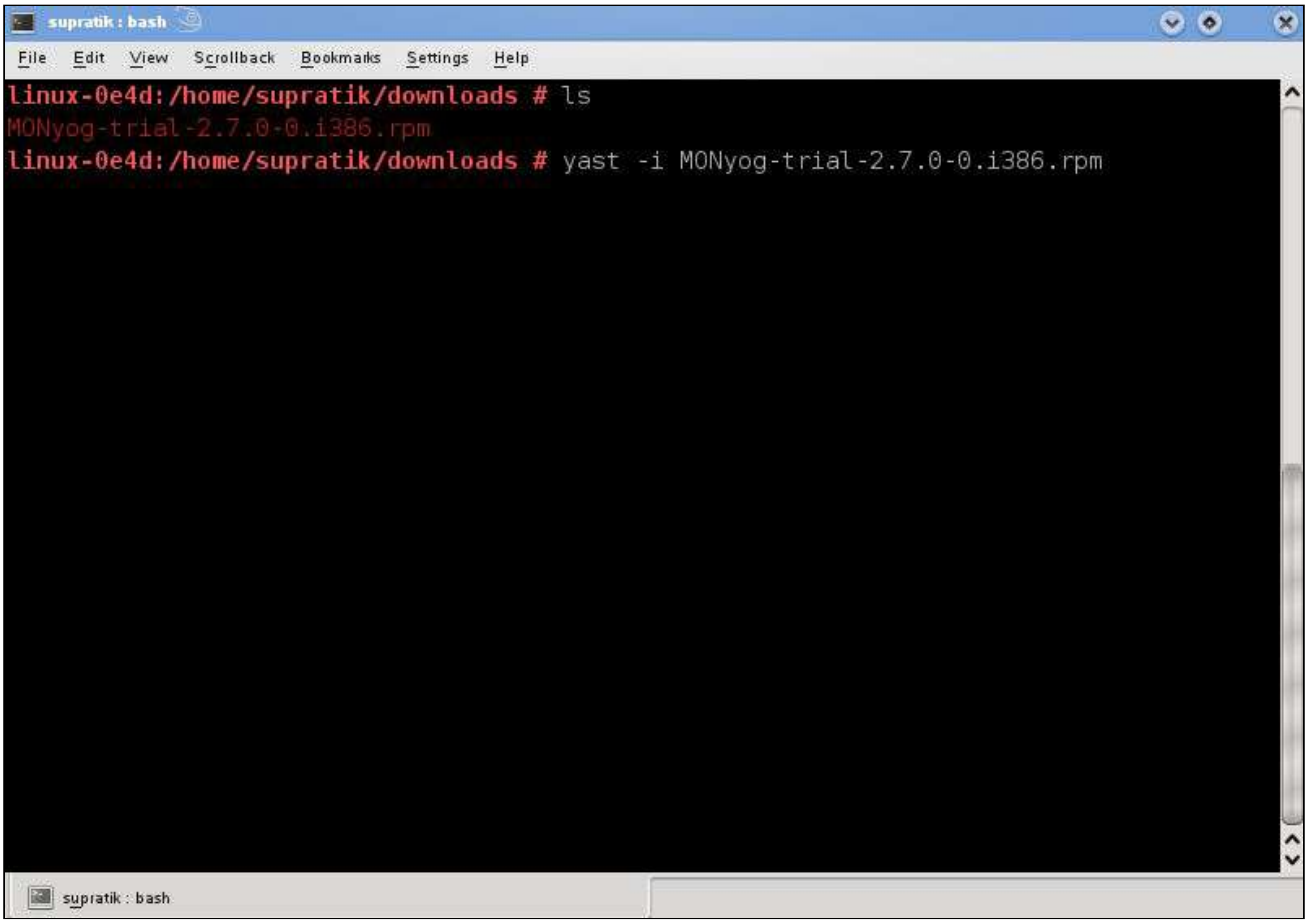
By default in these systems SQL DM for MySQL will be started automatically even after a restart.

- In SUSE systems if SQL DM for MySQL is installed from a RPM package, the YaST 'run-level tool' can be used to start/stop the server. You may enable/disable the MONyog(SQL DM for MySQL) service from **YaST (Administrative settings) > System > System Services**.

Upgrading

To upgrade SQL DM for MySQL in Redhat/Fedora from a previous installation the following command can be used:

```
# rpm -Uvh <IderaSQLdmforMySQL_package>.rpm
```



Uninstalling SQL DM for MySQL

To uninstall SQL DM for MySQL in Redhat/Fedora the following command can be used:

```
# rpm -e IderaSQLdmforMySQL
```

SQL DM for MySQL upgrade is supported from MONyog 2.7.0-0 onwards, currently SQL DM for MySQL supports upgrade from lower version of SQL DM for MySQL to upper version of SQL DM for MySQL.

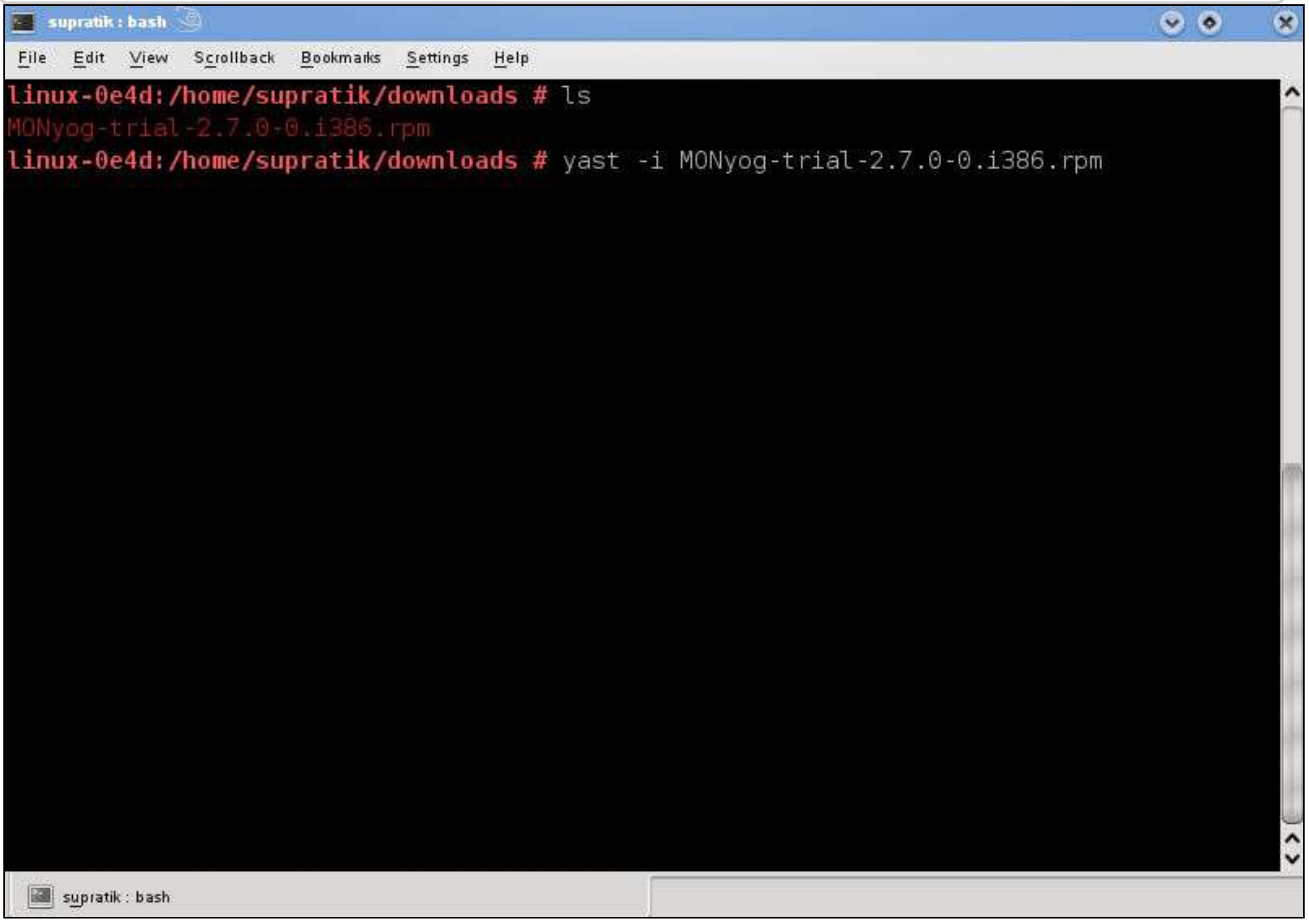
Tar Installation

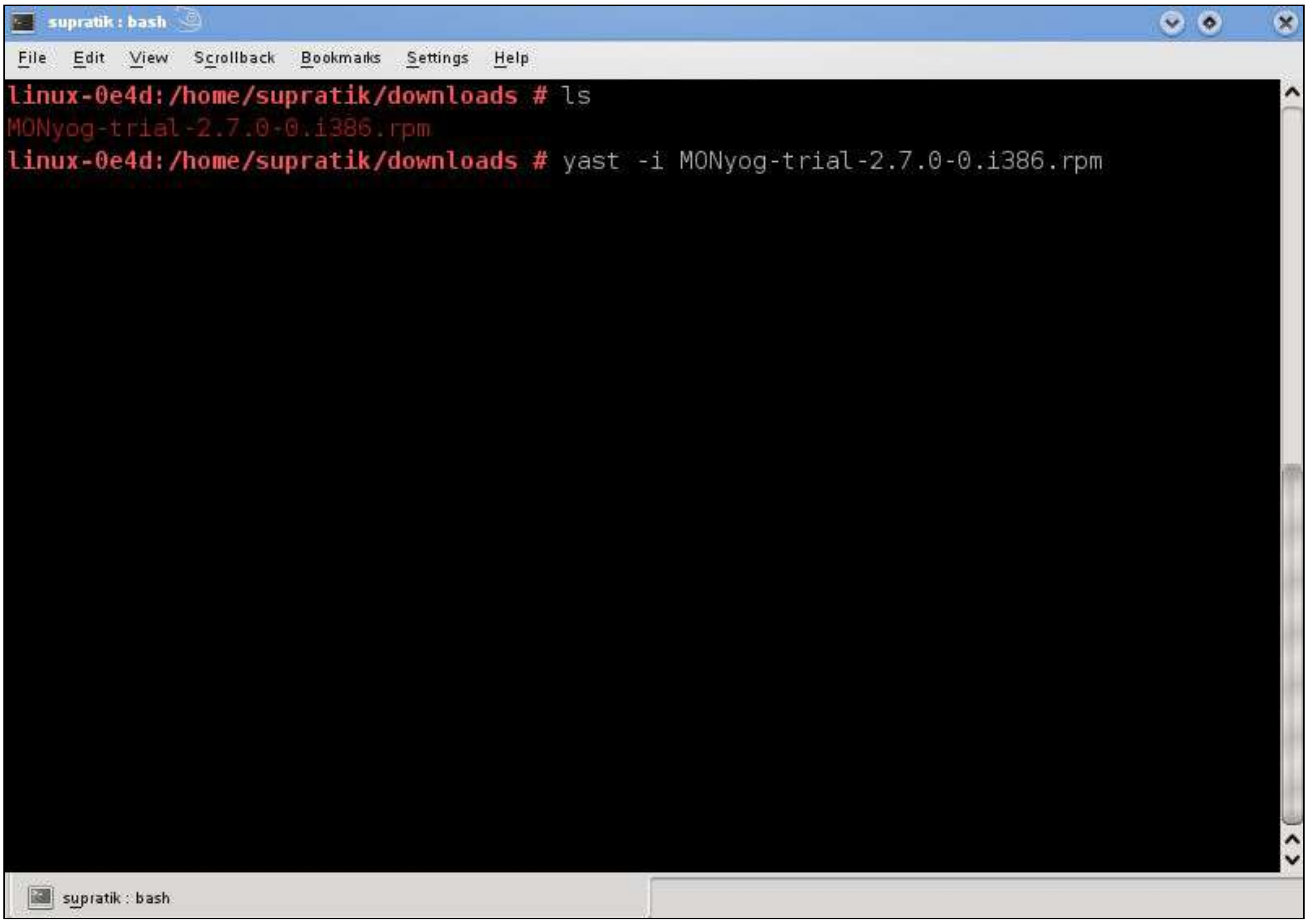
In the event that your server runs a Linux distribution that isn't RPM-based, such as Debian or Ubuntu, you can still manually install SQL DM for MySQL from a tarball. Note that SQL DM for MySQL does not require any sort of compiling. You just need to unpack it. Select the download link that corresponds to your hardware: **32 Bit Tar** or **64 Bit TAR**.

Installation

The download link retrieves a `.tar.gz` file from IDERA. The pattern for the filename is `IderaSQLdmforMySQL-<version>.<architecture>.tar.gz`. To install it run Tar setting the output to `/usr/local` or any prefix appropriate to your system:

```
# tar -xvf /path/to/IderaSQLdmforMySQL-*.tar.gz -C /usr/local
```





It installs SQL DM for MySQL in the same path as the RPM package. In order to use the MONyog(SQL DM for MySQL) service, you also need to copy the `MONyog` service script to `/etc/init.d/` directory.

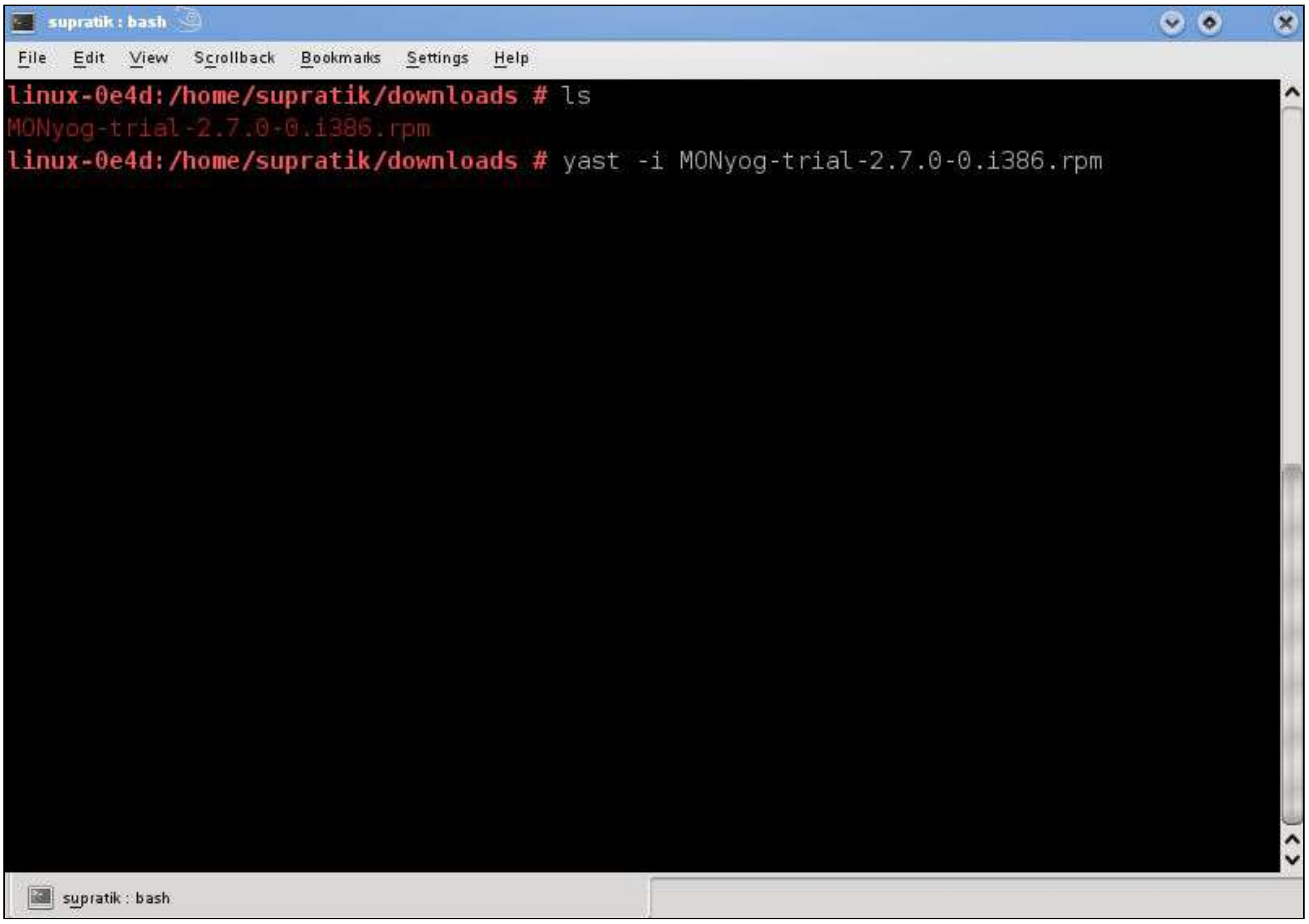
```
# cp /path/to/MONyog/bin/MONyog /etc/init.d/
```

Start/Stop

This description applies if you have extracted SQL DM for MySQL(Monyog) from zipped tar(.tar.gz) package.

There is one shell script named "MONyog" within "MONyog/bin" directory. For example if SQL DM for MySQL(Monyog) has been extracted to ~/MONyog/ directory, you can start SQL DM for MySQL by typing:

```
$ ~/MONyog/bin/MONyog start
```



Similarly, to stop:

```
$ ~/MONyog/bin/MONyog stop
```

Upgrading

Untar the SQL DM for MySQL package in the same directory where you had untarred the previous SQL DM for MySQL, use the command: 64 bit tar.

```
# tar -xvf /path/to/IderaSQLdmforMySQL-*.tar.gz -C /usr/local
```

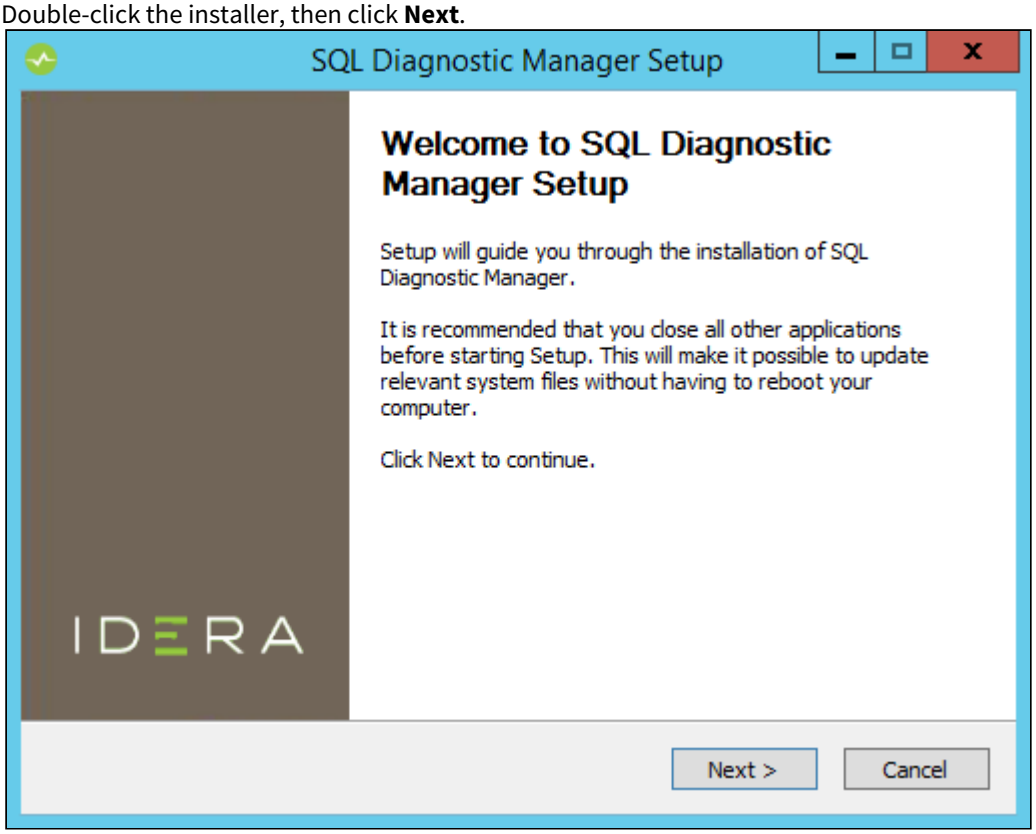
Windows

SQL DM for MySQL runs as a web application on a server, from which it connects to and monitors MySQL database servers. The MySQL servers can run on any operating system. SQL DM for MySQL can run on any distribution of the Linux <install-linux> operating system or it can run on Microsoft Windows Server.

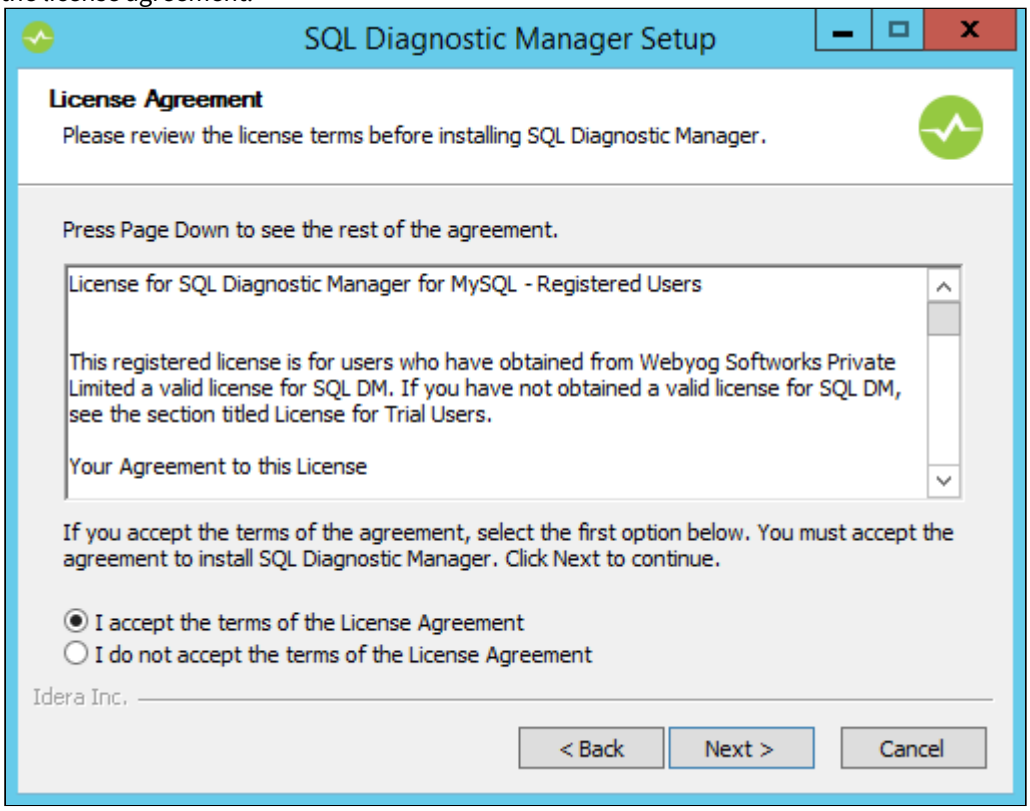
Installing SQL DM for MySQL

Webyog provides an executable file to install SQL DM for MySQL on Windows. When you receive the download link, select **Executable (.exe)** to get this installer. Once you have the installer, transfer it to the Windows Server instance you want to host SQL DM for MySQL.

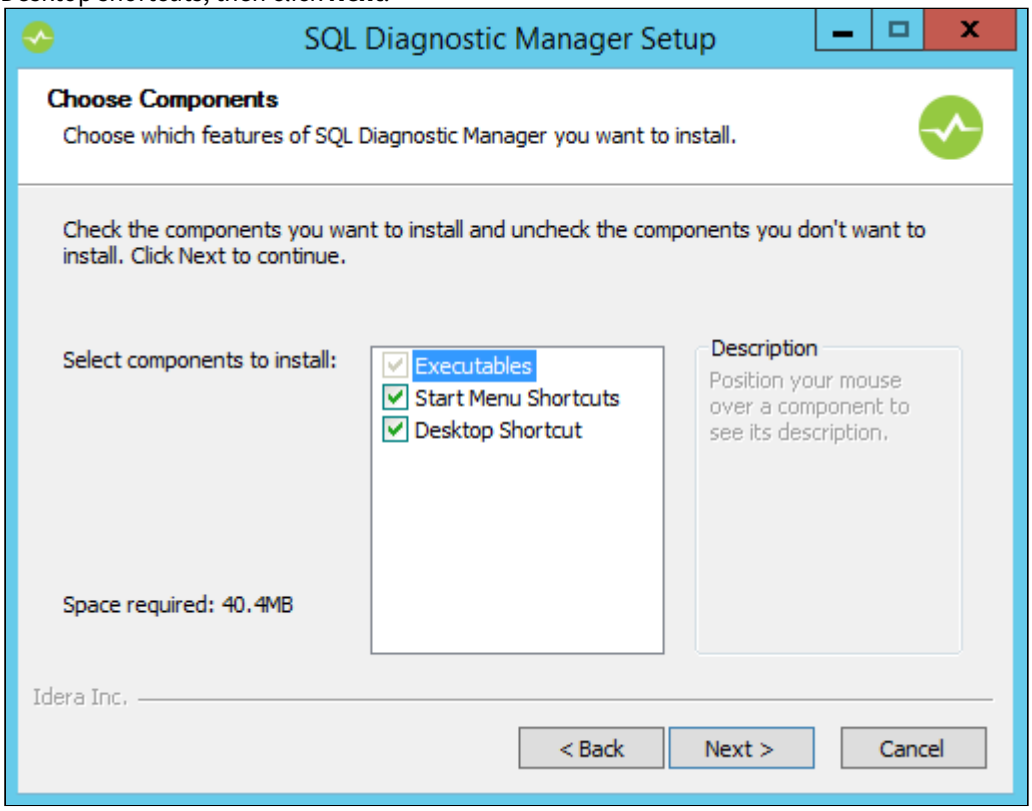
1. Double-click the installer, then click **Next**.



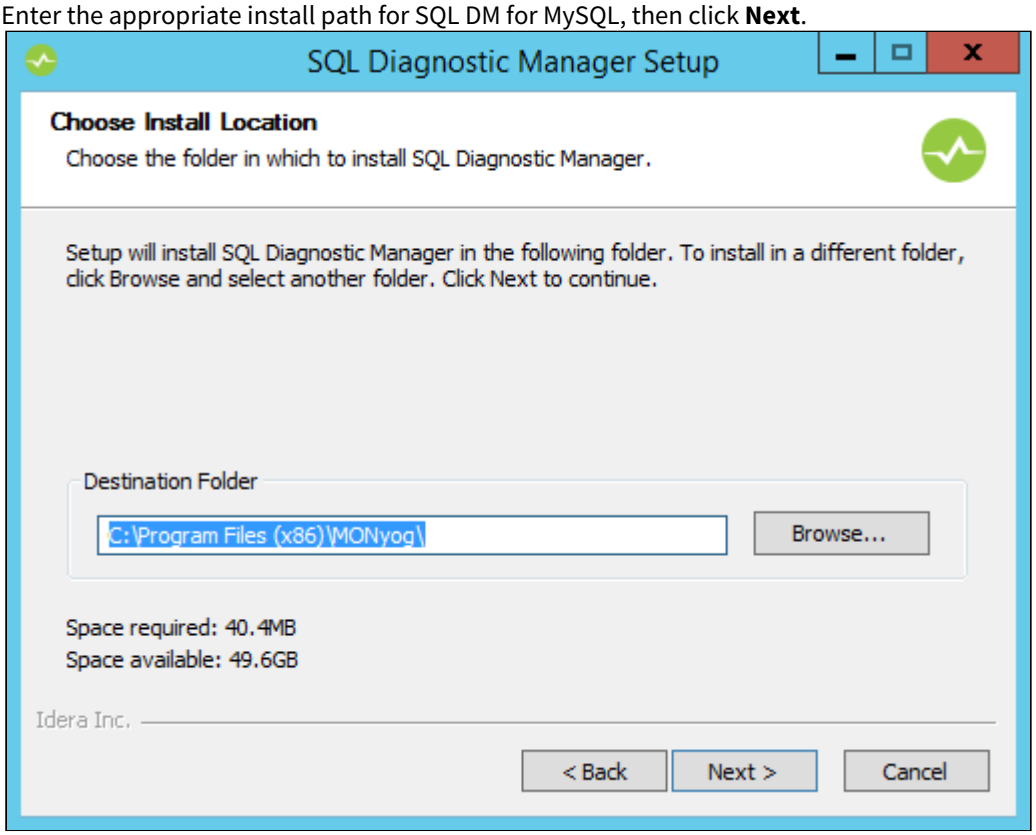
2. Read and accept the License Agreement, then click **Next**. Using SQL DM for MySQL requires that you accept the license agreement.



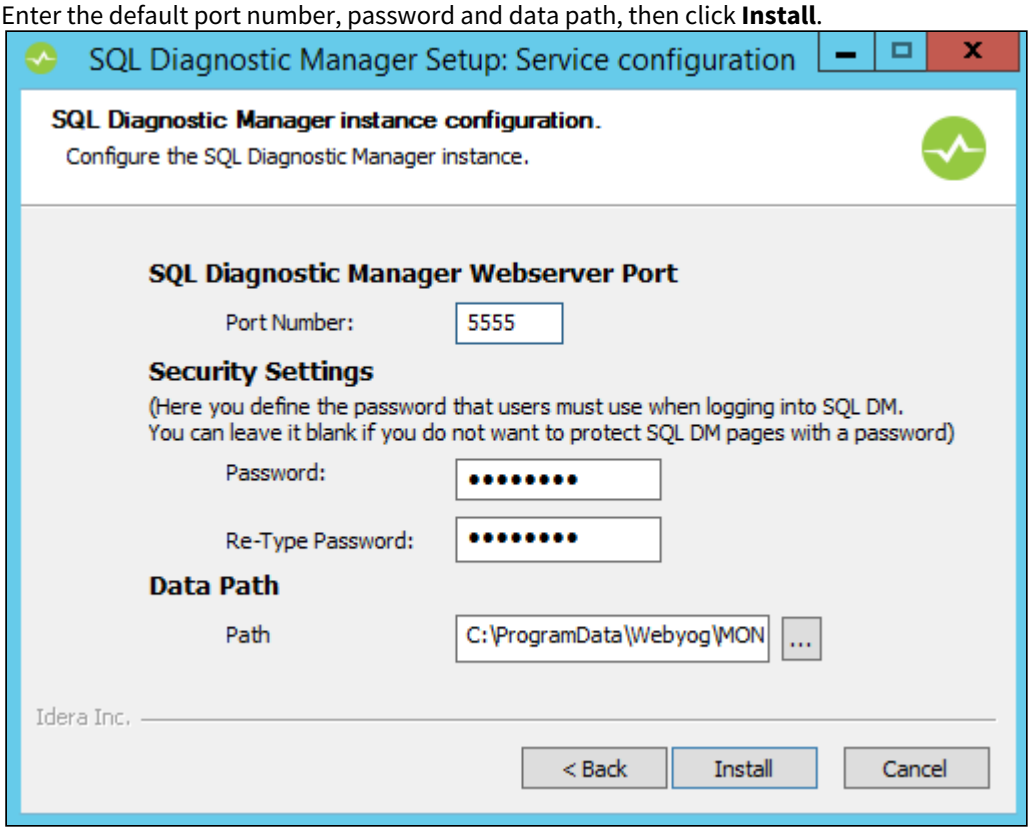
3. Select the components you want to install for SQL DM for MySQL, including executables, Start Menu and Desktop shortcuts, then click **Next**.



4. Enter the appropriate install path for SQL DM for MySQL, then click **Next**.



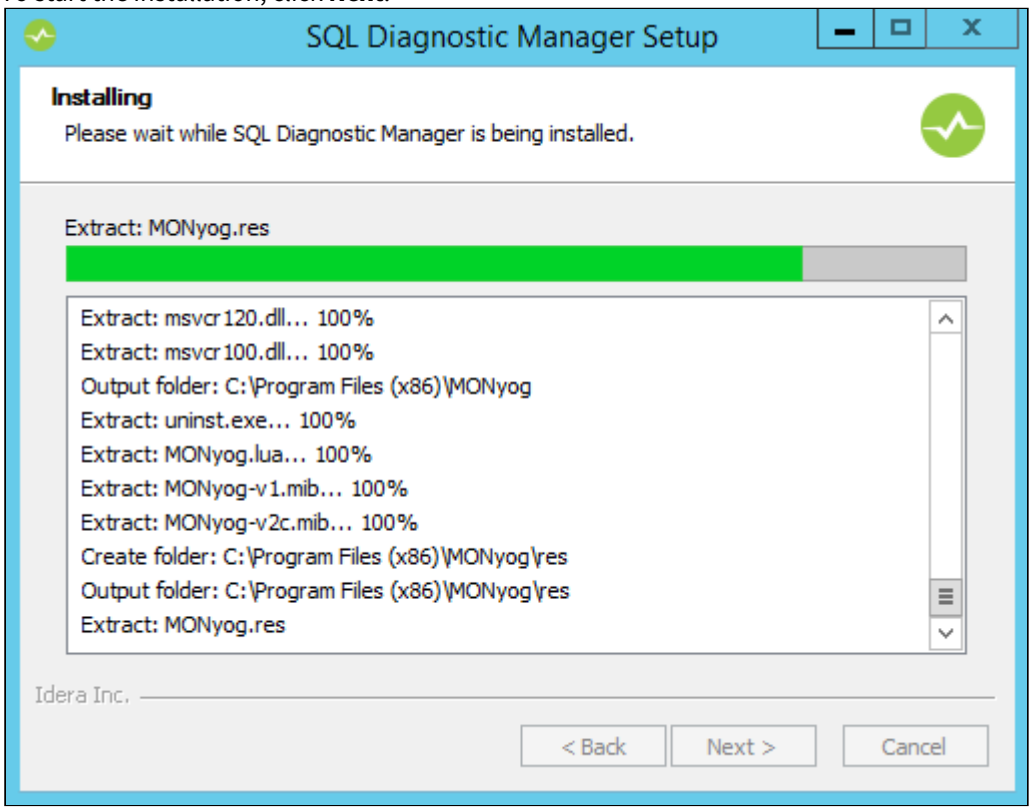
5. Enter the default port number, password and data path, then click **Install**.



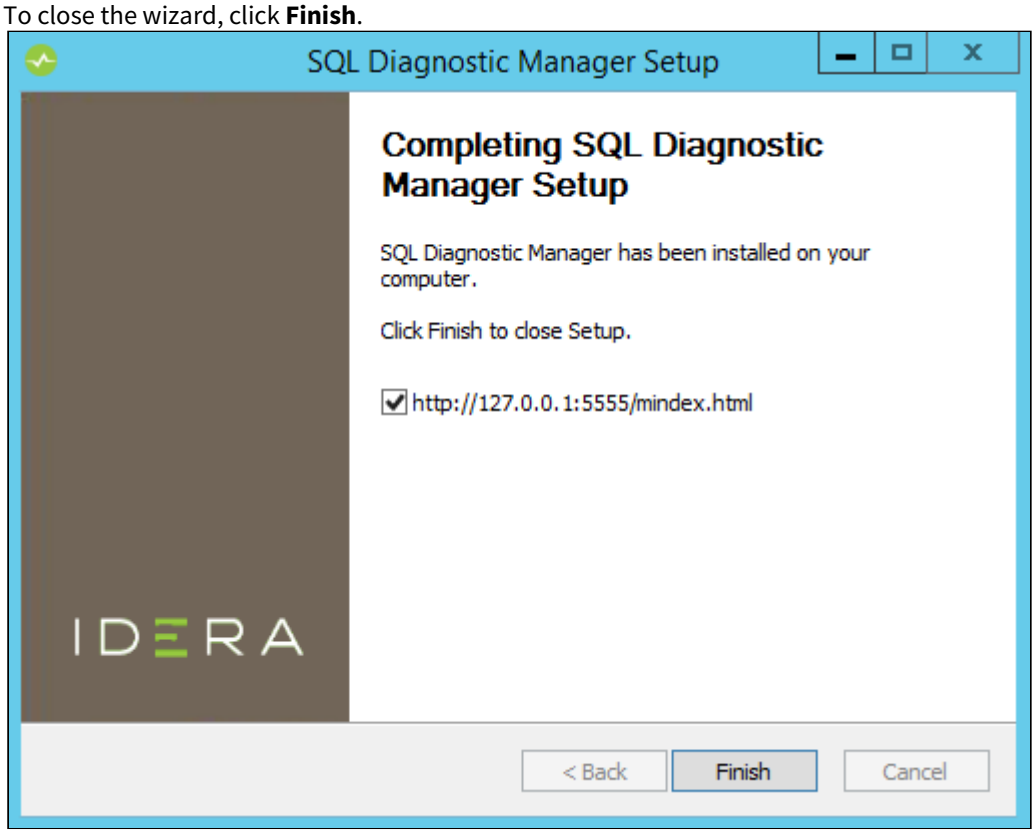
In the event that you decide later that you would like to use different port numbers or login credentials, you can change them by editing the `MONyog.ini` file and restarting MONyog(SQL DM for MySQL) Service. The location of this file varies depending on what version of Windows you use:

- **Windows 2008 and later:**`C:\ProgramData\Webyog\MONyog\`
- **Windows Vista and later:**`C:\ProgramData\Webyog\MONyog`

6. To start the installation, click **Next**.



7. To close the wizard, click **Finish**.



Managing the MONyog(SQL DM for MySQL) Service

SQL DM for MySQL installs as a service on Windows. You can manage the SQL DM for MySQL service, named MONyog from the Windows Service Manager as you would any other service. When the installation is complete, the installer automatically starts the MONyog service.

You can access SQL DM for MySQL through a web browser at `http://127.0.0.1:5555`, or at the port you configured it to use.

Monyog AMI

SQL DM for MySQL AMI comes with the default installation of 64 bit SQL DM for MySQL and nginx reverse proxy on Amazon Linux. You can find SQL DM for MySQL installer in the home directory.

If SQL DM for MySQL AMI default security group is used, SQL DM for MySQL User Interface can be accessed on port 80(http) and 443(https).

The URL to access SQL DM for MySQL would be:

- `http://<Public DNS of the Amazon instance>`
- `https://<Public DNS of the Amazon instance>`

Upgrading And Installing

Upgrading and installing SQL DM for MySQL can be either done by using:

- YUM to upgrade:

```
# yum update MONyog
```

or to install SQL DM for MySQL:

```
# yum install MONyog
```

- Downloading the binaries and upgrading it with RPM:

```
# rpm -Uvh MONyog-*.rpm
```

or, installing it with RPM:

```
# rpm -ivh MONyog-*.rpm
```

Nginx Configuration

SQL DM for MySQL AMI is configured with nginx reverse proxy. Nginx listens on ports 80 and 443 and forwards the incoming traffic to SQL DM for MySQL default port: 5555.

SQL DM for MySQL AMI comes with an SSL certificate which is not verfied by any Certificate Authority(CA).

Note

We highly recommend you to change the SSL certificates. You can either have a self-signed SSL certificate or get one signed by any Certificate Authority(CA). Refer the following to know more about generating SSL certificates with OpenSSL: CA.pl(see page 34). Also refer FAQ 14(see page 219) points 5-7.

To change SSL certificates in Nginx follow the FAQ 13(see page 217) section of the HELP file.Reset the admin password before continuing to use SQL DM for MySQL.

Reset the admin password before continuing to use SQL DM for MySQL.

Configuration

With SQL DM for MySQL installed on your server, you can begin to configure systems and databases in your infrastructure for its use.

Use SQL DM for MySQL to monitor a server, only grant its user access to MySQL and then register the server through the Web UI.

MySQL Configuration

When you register a server with SQL DM for MySQL, it collects information and monitors the server's health through standard MySQL client connections. You need to configure a user account with the appropriate permissions on each MySQL server you want to monitor. Fully enabling SQL DM for MySQL requires that you grant `SELECT`, `RELOAD`, `PROCESS`, and `SUPER` to this user. You can use whatever username you find convenient, with the host set to the server hosting SQL DM for MySQL.

For instance, say you have installed SQL DM for MySQL on a server at 192.168.1.150 and that you would like use the user `monyog` as the user name. To fully enable SQL DM for MySQL in this scenario, you would issue the following command:

```
GRANT SELECT, RELOAD, PROCESS, SUPER ON *.*

TO 'monyog'@'192.168.1.150';
```

Granting these permissions makes the MySQL server fully accessible to SQL DM for MySQL. To actually start collecting data on this server you also need to register it with the application.

Running SQL DM for MySQL as unprivileged user

By default, SQL DM for MySQL runs under the 'root' account in Linux. This may be a security nightmare for some.

Here's how you can create and use a user account exclusively for running SQL DM for MySQL:

- First add a new user and group; replace <GID>, with the group ID, and <PASSWORD>:

```
# groupadd monyog
```

```
# useradd -g <GID> -p <passwd> monyog
```

- Copy the original files to the new location and change the file ownership.

```
# cp -r /usr/local/MONyog/. /home/monyog/
```

```
# chown -R monyog:monyog /home/monyog
```

- Next, we need to change the initialization script. Be sure to make a backup first!

```
# cp /etc/init.d/MONyogd /etc/init.d/MONyogd.orig
```

```
# vi /etc/init.d/MONyogd
```

- Change the following lines for the new path.

```
PREFIX="/home/monyog"
```

```
MONYOGBIN="$PREFIX/bin/MONyog"
```

- In the same file, you also need to update the start command to the following.

```
# Start MONyog
...
else
action "`su - monyog -c \"$MONYOGBIN -s\"`" /bin/true
fi
```

- Now you need to alter the configuration file.

```
# vi /home/monyog/MONyog.ini
```

- Change the data path to the new directory.

```
Data_path=/home/monyog/data
```

- After this you should be able to start SQL DM for MySQL running as an unprivileged user.

```
# service MONyog start
```

Server Registration

At this point, you should have SQL DM for MySQL installed and configured on a Linux or Windows server and one or more servers running MySQL that you have configured to provide monitoring data to SQL DM for MySQL. However, in order to get SQL DM for MySQL to access that data, you need to register the server with the application.

A single instance of SQL DM for MySQL can monitor hundreds of MySQL database servers. With tagging, you can categorize them into logical groups, ensuring that they're easier to keep track of and identify. During the Trial period, you can register up to 1000 MySQL servers, after which you'll need to purchase a license.

For more information on licenses and pricing, see Pricing(see page 37).

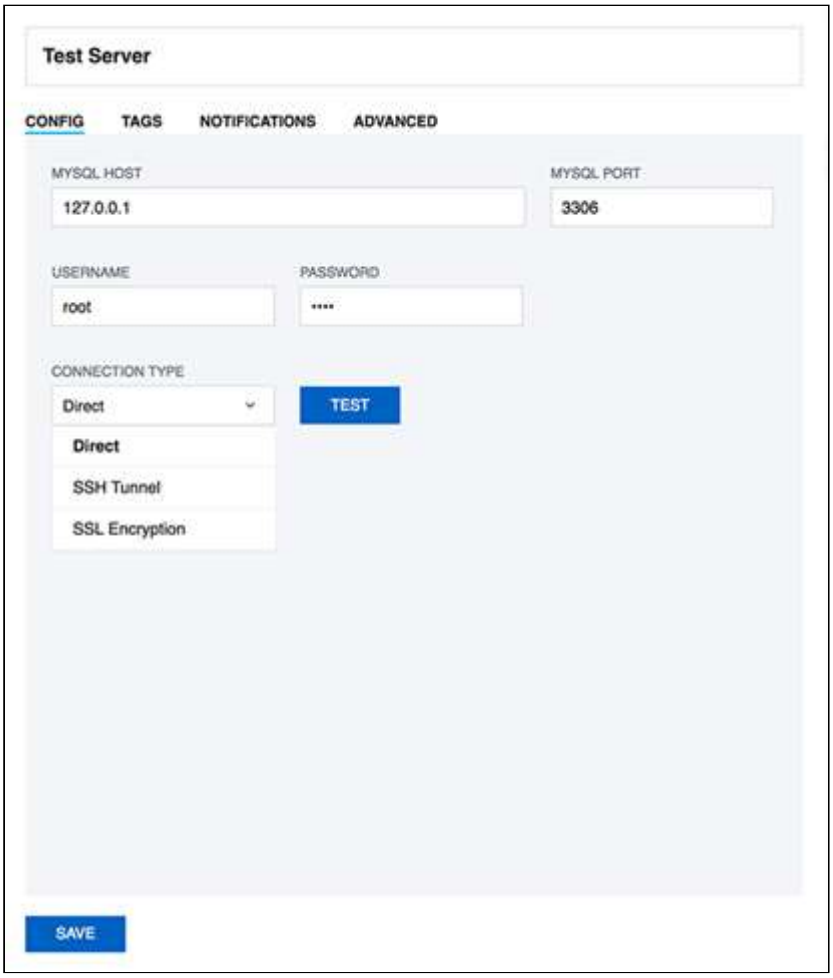
Adding Servers

The web interface for SQL DM for MySQL provides a convenient form for adding new MySQL database servers to monitor. The link to this form is found on the Servers tab. To get there, click the **Servers** icon on the dock.

On the Servers tab, click the **Add New Server** link at the top of the page:

Clicking this link opens an overlay form which you can use to configure a new server for SQL DM for MySQL.

For general information, you need to provide a name, the IP address or domain name, port number for the MySQL host, the username, and password for the SQL DM for MySQL user on the database. Also, you have the option of setting tags to group different servers together, and setting the connection type (direct, SSH tunneling, or SSL encryption).



When you finish configuring the new MySQL server, click **Save**.

SQL DM for MySQL now collects data on the new server. You can view its state, compare its configuration to another servers, and much more.

Register your servers

Connecting to SQL DM for MySQL

In this section it is assumed that SQL DM for MySQL is installed and running. Connect to it from a browser, start your browser, and enter it following the format: `http://host:port`

The host can be:

- An IP address (like "10.0.0.1")
- The host name of the system
- An alias being read from the local host-file (like `localhost`). The option Port is where you or your Sys Admin specified when installing SQL DM for MySQL. By default, SQL DM for MySQL UI can be accessed on port 5555.

Managing multiple servers

Single instance of SQL DM for MySQL is capable of monitoring hundreds of MySQL servers. With tagging, you can categorize your servers into logical groups, and avoid clutter.

You can register as many servers as your SQL DM for MySQL license allows. Also, SQL DM for MySQL installations registered with an Unlimited license, can register up to 1000 servers. If you are a customer with 'unlimited' license and want to register more licenses, please contact us by sending an email to IDERA Support[6].

The SQL Diagnostic Manager TRIAL installation can register up to 1000 servers.

Registering replicas

SQL DM for MySQL has an option to auto-register slaves. Please refer to Replication Settings(see page 0) in Advanced Settings for more details.

Review the following options to continue with the Registration of your servers:

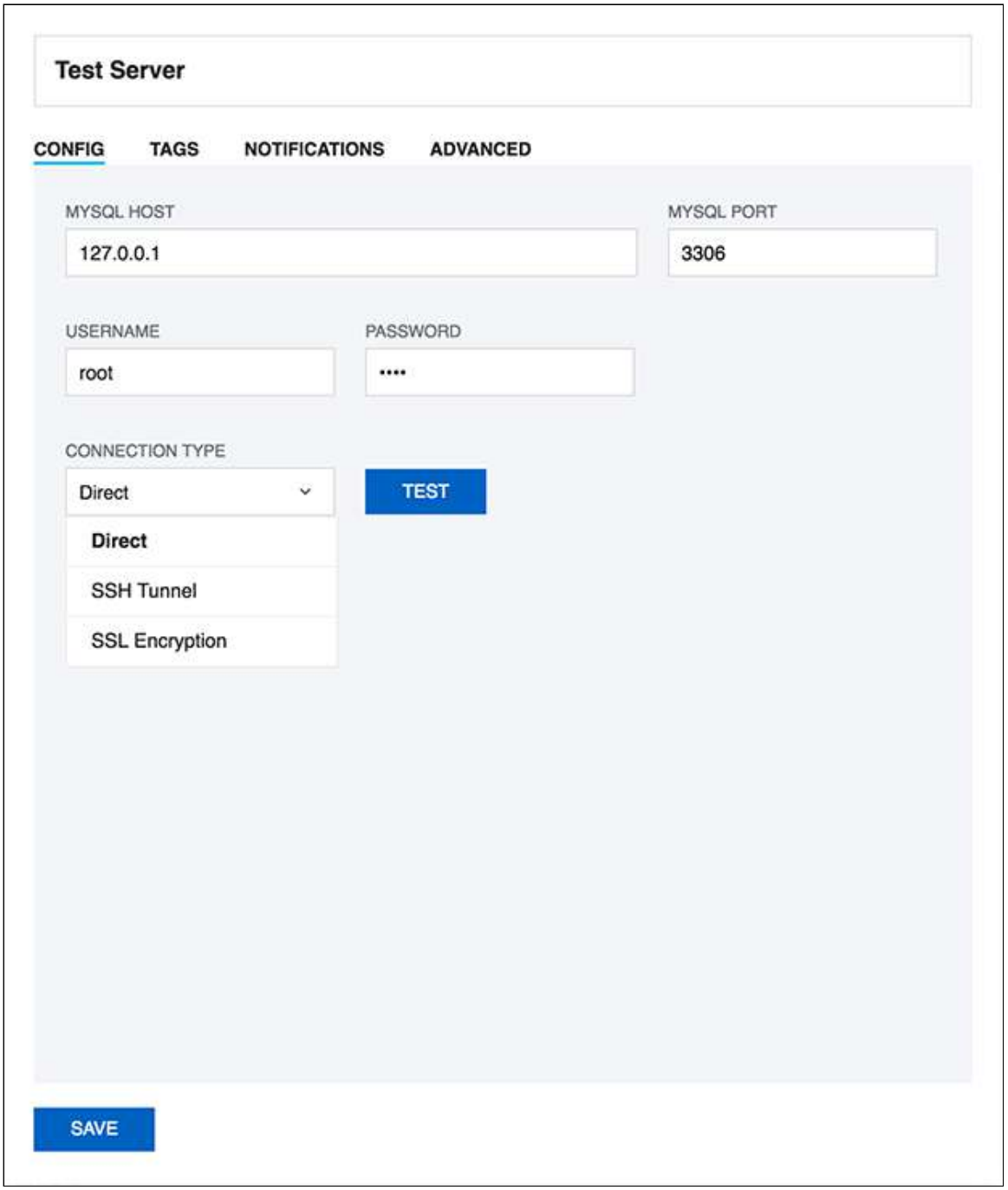
- Connecting to MySQL Server[7]
- Tags[8]
- Notification Settings[9]
- Advanced Settings[10]
- MySQL Privileges[11]
- System Privileges[12]
- RDS OS and File based Log Monitoring Privileges[13]

6 mailto:ideramysqlsupport@idera.com
7 http://wiki.idera.com/x/dwEGBg
8 http://wiki.idera.com/x/eAEGBg
9 http://wiki.idera.com/x/eQEGBg
10 http://wiki.idera.com/x/egEGBg
11 http://wiki.idera.com/x/ewEGBg
12 http://wiki.idera.com/x/fAEGBg
13 http://wiki.idera.com/x/fQEGBg

Connecting to MySQL Server

Registering Servers

First you must register the MySQL servers to which SQL DM for MySQL connects to. For every server you specify here, an embedded database is created. SQL DM for MySQL connects to those servers to retrieve and store information from that server.



For a **Direct connection**, enter the connection information as you would do from any client.

To retrieve system information (CPU, memory load etc.) you need to give the information required to create a remote command shell (Linux) from the server machine on which SQL DM for MySQL is running.

SSH tunneling

SQL DM for MySQL can send and receive encrypted authentication information as well as monitored data from MySQL using SSH tunneling. Also, it is possible to connect to MySQL with SSH tunneling even if the MySQL port (normally 3306) is blocked by a firewall or if users are not allowed to connect from remote hosts. An operating system user is required so that SQL DM for MySQL SSH client functionalities can use this user to connect to the SSHD daemon on the server.

If you want to use SSH tunneling to your MySQL server, you have to provide details for creating a SSH connection. Please refer to Using SSH connections, below.

To connect using SSH tunneling

- TCP port forwarding must be allowed in the SSH server. For openSSH servers it should be done in the "sshd_config" configuration file. In Linux usually it is in, `/etc/ssh/sshd_config`.

Set the "AllowTcpForwarding" option to "yes". So it should look like,

```
AllowTcpForwarding yes
```

- MySQL host should be specified 'relative' to the SSH server. Say, your SSH server is running in M1. Then you should ask yourself the following question: How should I connect to the target MySQL server from M1?

If your MySQL server is running in the same system M1, you can choose "**localhost**", "**127.0.0.1**" or the IP address of MySQL server which M1 can see.

SSH will listen on a specified port on the client machine, encrypt the data it receives, and forward it to the remote SSH host on port 22 (the SSH protocol port). The remote SSH host decrypts the data and forwards it to the MySQL server. The SSH host and the MySQL server do not have to be on separate machines, but separate SSH and MySQL servers are supported.

Using SSH connections

To create a SSH connection you need the following:

- **SSH Host**: Host of the machine on which SSH server is running.
- **SSH Port**: Port on which SSH server is listening. By default, it is 22.
- **SSH Username**: Username to access the SSH server (Note: not the MySQL server).
- **Authentication type**: Specify the type of authentication to use. This can be either key based or password based.
- If you have specified authentication type as Password - Provide the password.
- If you have specified authentication type as Key - You should note that SQL DM for MySQL only supports ''OpenSSH standard key format'' for key based authentication in SSH connections.
- **Private Key**: Paste the content of your private key file. Again, do not specify the path to your private key file.
- **Passphrase**: Enter the passphrase for your private key file (if any). This can be left blank, if no passphrase was given for the private key.

Using SSL Connection

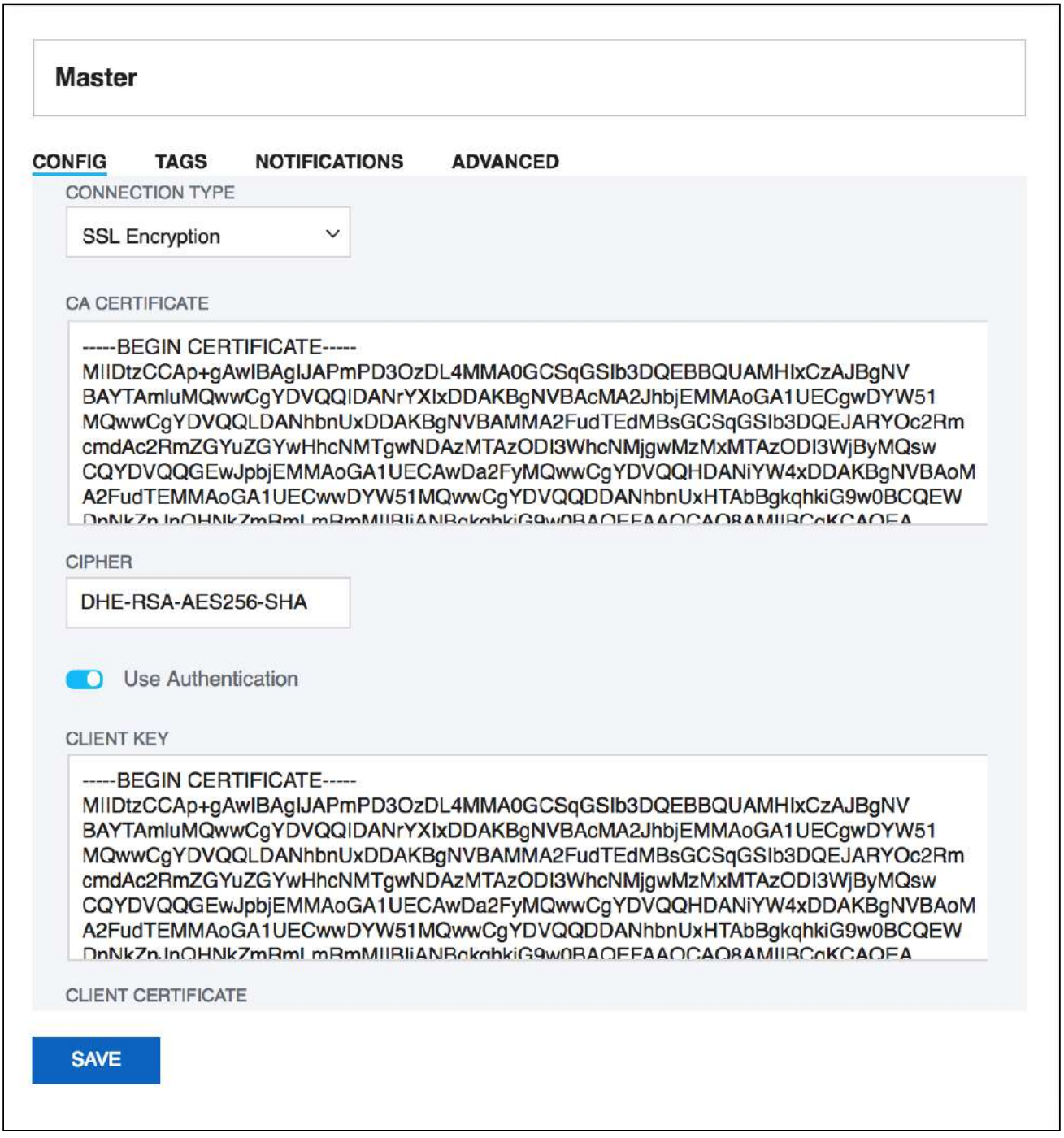
The Secure Sockets Layer (SSL) is a commonly-used protocol for managing the security of a message transmission on the Internet. SQL DM for MySQL provides native MySQL SSL encryption for direct MySQL connections.

To use SSL encryption you will be asked for:

- **CA Certificate**: The digital certificate issued by CA.
- **Cipher**: Encryption algorithm like DES, AES etc.

If you need client authentication then,

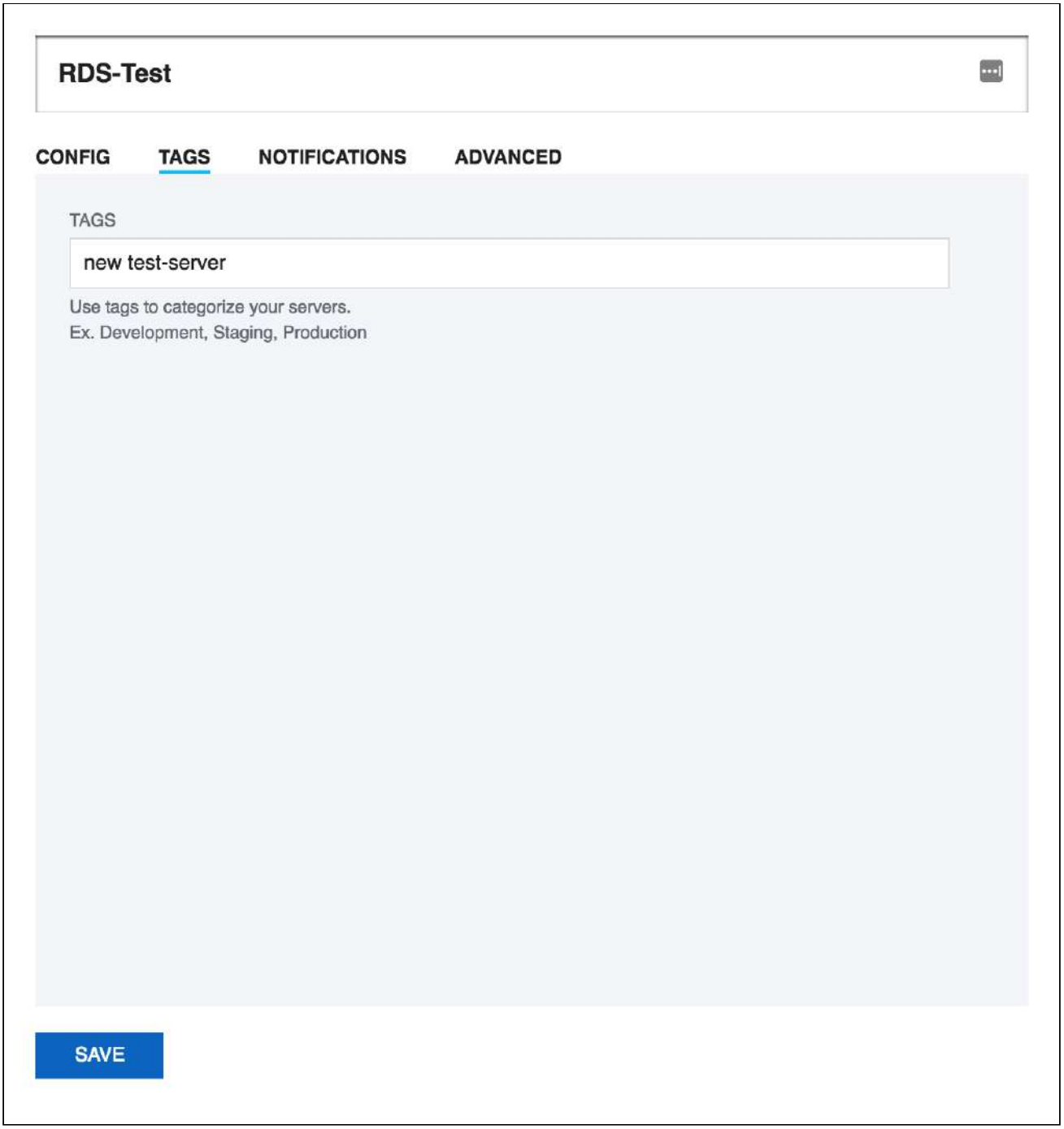
- **Client Key**: Private key of the client that is needed for encryption.
- **Client Certificate**: The client certificate.


Master
CONFIG
TAGS
NOTIFICATIONS
ADVANCED
CONNECTION TYPE
SSL Encryption
CA CERTIFICATE
-----BEGIN CERTIFICATE-----
MIIDtzCCAp+gAwIBAgIJAPmPD3OzDL4MMA0GCSqGSIb3DQEBBQUAMHIxCzAJBgNV
BAYTAmluMQwwCgYDVQQIDANrYXIxDDAKBgNVBAcMA2JhbjEMMAoGA1UECgwDYW51
MQwwCgYDVQQLDANhbnUxDDAKBgNVBAMMA2FudTEdMBsGCSqGSIb3DQEJARYOc2Rm
cmdAc2RmZGYuZGYwHhcNMTgwNDAzMTAzODI3WhcNMjgwMzMxMTAzODI3WjByMQsw
CQYDVQQGEwJpbjEMMAoGA1UECAwDa2FyMQwwCgYDVQQHDANiYW4xDDAKBgNVBAoM
A2FudTEMMAoGA1UECwwDYW51MQwwCgYDVQQDDANhbnUxHTAbBgkqhkiG9w0BCQEW
CIPHER
DHE-RSA-AES256-SHA
Use Authentication
CLIENT KEY
-----BEGIN CERTIFICATE-----
MIIDtzCCAp+gAwIBAgIJAPmPD3OzDL4MMA0GCSqGSIb3DQEBBQUAMHIxCzAJBgNV
BAYTAmluMQwwCgYDVQQIDANrYXIxDDAKBgNVBAcMA2JhbjEMMAoGA1UECgwDYW51
MQwwCgYDVQQLDANhbnUxDDAKBgNVBAMMA2FudTEdMBsGCSqGSIb3DQEJARYOc2Rm
cmdAc2RmZGYuZGYwHhcNMTgwNDAzMTAzODI3WhcNMjgwMzMxMTAzODI3WjByMQsw
CQYDVQQGEwJpbjEMMAoGA1UECAwDa2FyMQwwCgYDVQQHDANiYW4xDDAKBgNVBAoM
A2FudTEMMAoGA1UECwwDYW51MQwwCgYDVQQDDANhbnUxHTAbBgkqhkiG9w0BCQEW
CLIENT CERTIFICATE
SAVE

Tags

Tags are used for grouping servers. When you have few servers it is easy to keep track of them but, it is increasingly difficult to keep track of servers when there are too many of them. To reduce the difficulty in dealing with too many servers you can use tags.

Using tags you can categorize the servers by grouping a server under a single tag or you can also enter multiple tags separated by 'Space' for each server. Also, note that if you do not provide any tag then by default the server will be categorized under 'Not Tagged'.



Using tags, it is easy to search for a group of servers at once, simply enter the tag name in the server selector with **#** (ex: #test) and all the tags with the entered keyword are listed and you can further select the desired tag to select servers tagged under it.

Notifications Settings

To use the SQL DM for MySQL Notification alert functionality to send you alerts about a MySQL server you need to configure the different Notification channels from **Settings -> Notification & Maintenance**. Once you configure the notification channels from the Settings page, those channels are now be available for sending alerts for each of the servers. If you have enabled all the channels, then you get the following alert options: Email, SNMP, Slack, Pagerduty, and Syslog.

Different Notification Channels

Email

Use this option, to get SQL DM for MySQL alerts via Email. Once you enable it, enter the addresses to which SQL DM for MySQL is to send e-mails.

There is an Option to set a distinct email distribution list for warning and critical alerts. You can define a proper demarcation between the users who receives a particular alert based on the alert status. For instance, if you want only your on-call DBAs to receive critical alerts 24/7, this feature handles that easily. Please refer to Mail Settings(see page 182) for more information. Details about the SMTP server to use. SQL DM for MySQL does not have a SMTP server of its own. It needs to connect to a SMTP server on the Internet, your local network or your local computer. The settings are the same you enter when installing or configuring a mail client program like Outlook or Outlook Express.

SNMP

Enable this option to get alerts via SNMP traps. Refer to SNMP settings(see page 185) for more information

Write to Syslog

Enable this option if you want SQL DM for MySQL to write the alerts in the Syslog (of the machine where SQL DM for MySQL is installed). Refer to SYSLOG Settings(see page 188) for more information.

Slack Notifications

Use this option to send alerts to Slack. Like Pagerduty, you get a drop-down option to select a slack channel to which the alerts can be sent to. Refer to Slack Settings(see page 187) for more information.

Pagerduty Notifications

Enable this option to get alerts on Pagerduty. After enabling, you get a drop-down option to select a rule which you created at **Settings -> Notification & Maintenance -> PAGERDUTY**. All the alerts are sent to this rule in Pagerduty. Refer to Pagerduty Settings(see page 186) for more information.

MONITORS (CRITICAL & WARNING)

Notify When the Monitor is in Alert State For: Here, you can specify the number of times a counter should be in an alertable state, consecutively, before a notification is sent.

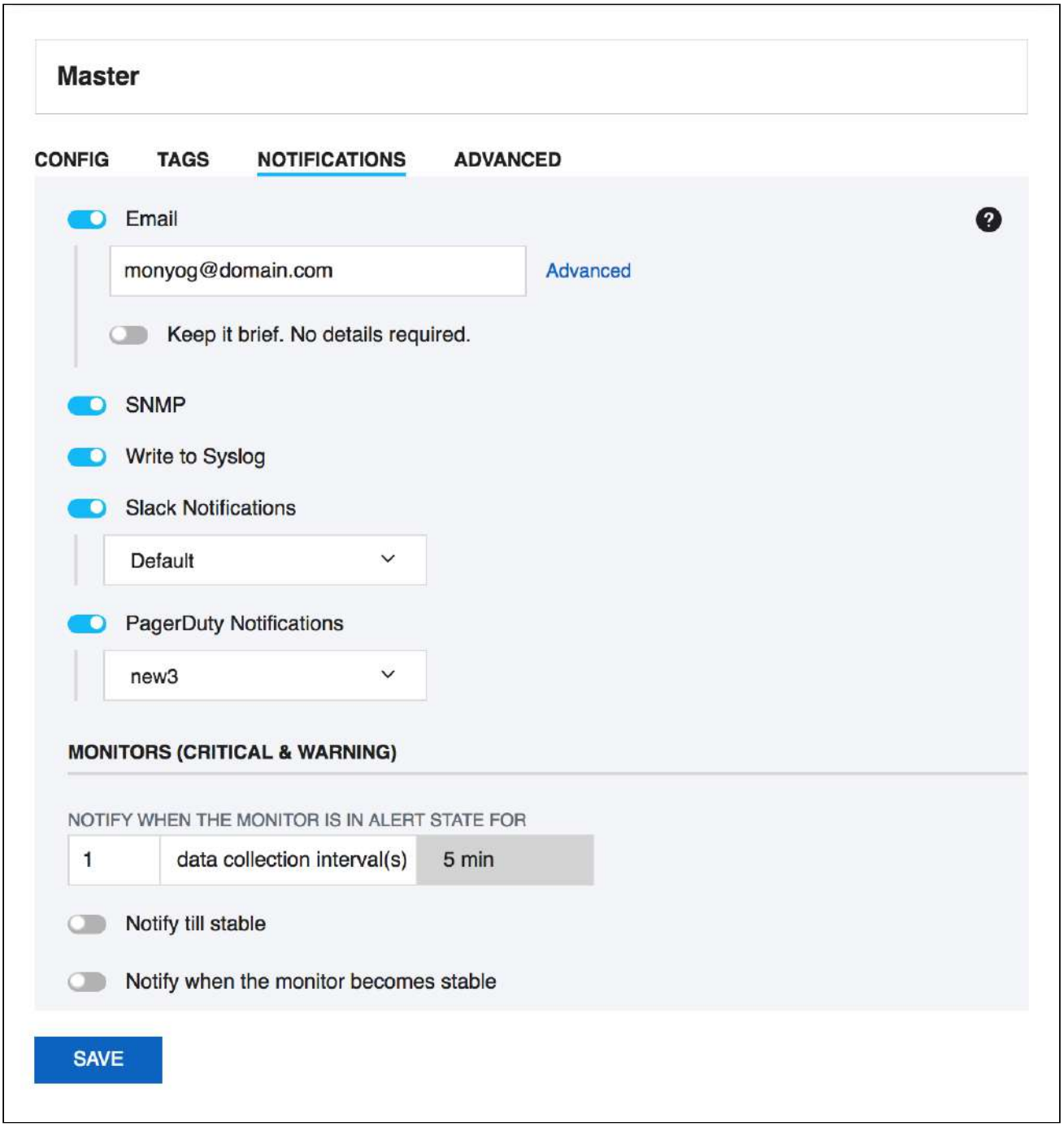
Notify till stable: SQL DM for MySQL sends mail alert notification until a given monitor becomes stable. The value in the "remind me after every" specifies how often you want to receive the mail alert notification i.e. if set to three, then for every three consecutive data collection for which the monitor is not stable, an e-mail alert is sent.

Notify when the monitor becomes stable: If a monitor goes into alertable state and then becomes stable, a stable e-mail or an SNMP trap is sent by SQL DM for MySQL.

OTHER

Notify when server restarts: If you enable this option, it sends an alert if the server restarts between two data retrievals. With managed hosting, the MySQL server may be restarted automatically as part of routine maintenance or after a server crash. Knowing when this has occurred can be useful.

Notify when changes to MySQL configuration is detected: If you enable this option, SQL DM for MySQL sends an alert whenever there is a change in MySQL configuration. These alerts are sent whenever SQL DM for MySQL detects a change in server variables using SET GLOBAL statements or in the MySQL configuration file.



Advanced Settings

You can configure the following Advanced settings when connecting to MySQL server:

- System Metrics(see page 49)

System Metrics

System Metrics (applicable for Linux based systems)

In the event that you would like SQL DM for MySQL to use SSH when communicating with this server, you can configure it from this tab. SQL DM for MySQL disables SSH communications by default. In order to use it, you need to click the **Enable System Metrics** switch. Doing so provides a series of configuration options needed to enable SSH on the server.

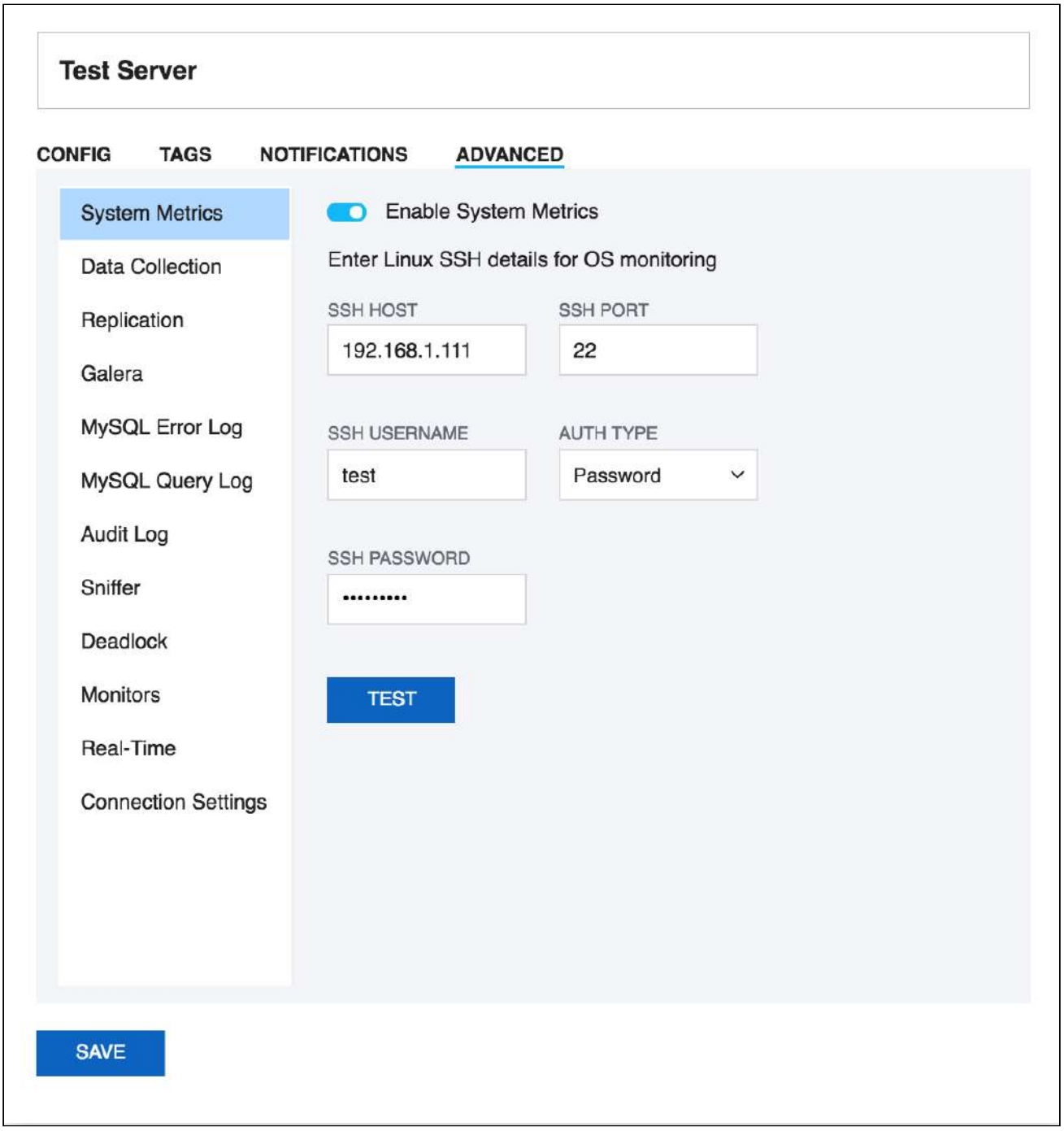
Please refer to System Privileges(see page 71) for further details of privileges needed for this feature.

If SSH tunneling to MySQL is configured successfully for this registration you can use those same details here too, provided that SSH tunnel user has enough privileges.

Using SSH connections

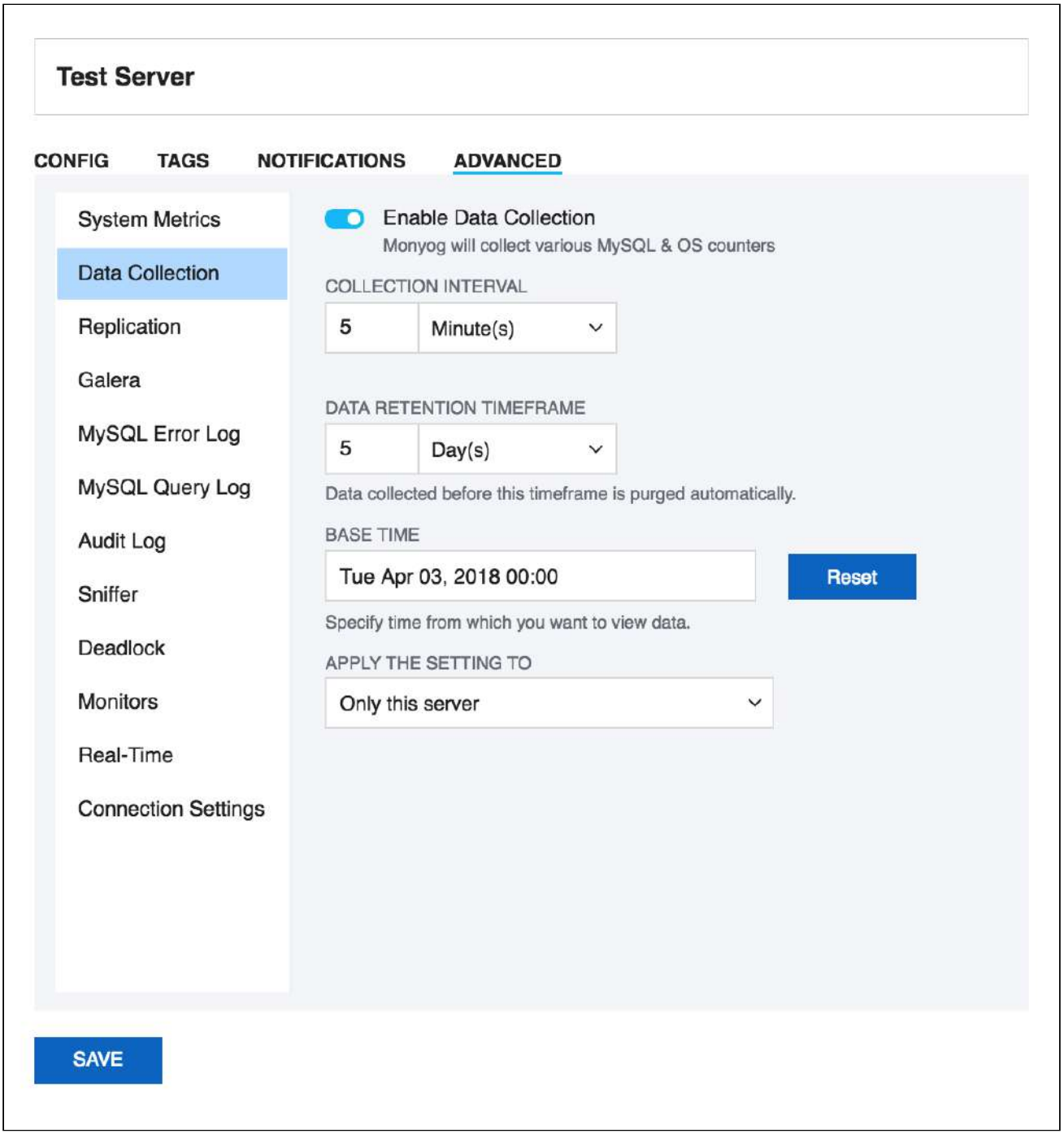
To create a SSH connection you need the following details:

- **SSH Host:** Host of the machine on which SSH server is running.
- **SSH Port:** Port on which SSH server is listening. By default, it is 22.
- **SSH Username:** Username to access the SSH server (Note: not the MySQL server).
- **Authentication type:** Specify the type of authentication to use. This can be either key based or password based.
- If you have specified Authentication type as Password - Provide the password.
- If you have specified, Authentication type as Key - You should note that SQL DM for MySQL only supports ''OpenSSH standard key format'' for key based authentication in SSH connections.
- **Private Key:** Paste the content of your private key file. Again, do not specify the path to your private key file.
- **Passphrase:** Enter the passphrase for your private key file (if any). This can be left blank, if no passphrase was given for the private key.



Data Collection

If you want to collect data from the server, you need to select **Enable Data Collection**, so SQL DM for MySQL collects and stores various MySQL and OS metrics.



Define the collection interval for every server as you want

You can also define the time interval between two successive retrievals of data. For production systems a setting between 2 and 10 minutes is a good place to start.

Data retention time frame

SQL DM for MySQL is designed for storing large amounts of data for long periods of time. Data collected before the specified time frame is purged automatically. Time frame may be specified in seconds, minutes, hours and days for a particular server.

Base time

For calculation of uptime-based counters the current value of each status variable is compared with either of those,

- server status variable 'uptime'
- server status variable 'uptime_since_flush_status'
- SQL DM for MySQL 'base time' setting

If SQL DM for MySQL 'base time' setting is defined and server status variable uptime_since_flush_status is available then, `uptime_since_flush_status` is used, if it is not available then base time is used.

The reason for this implementation is that if `FLUSH STATUS` is executed with a MySQL server, the server status variables are reset to the same value as after a server restart. There is one important exception however and that is the 'uptime' status variable itself. This single status variable is not affected by `FLUSH STATUS`.

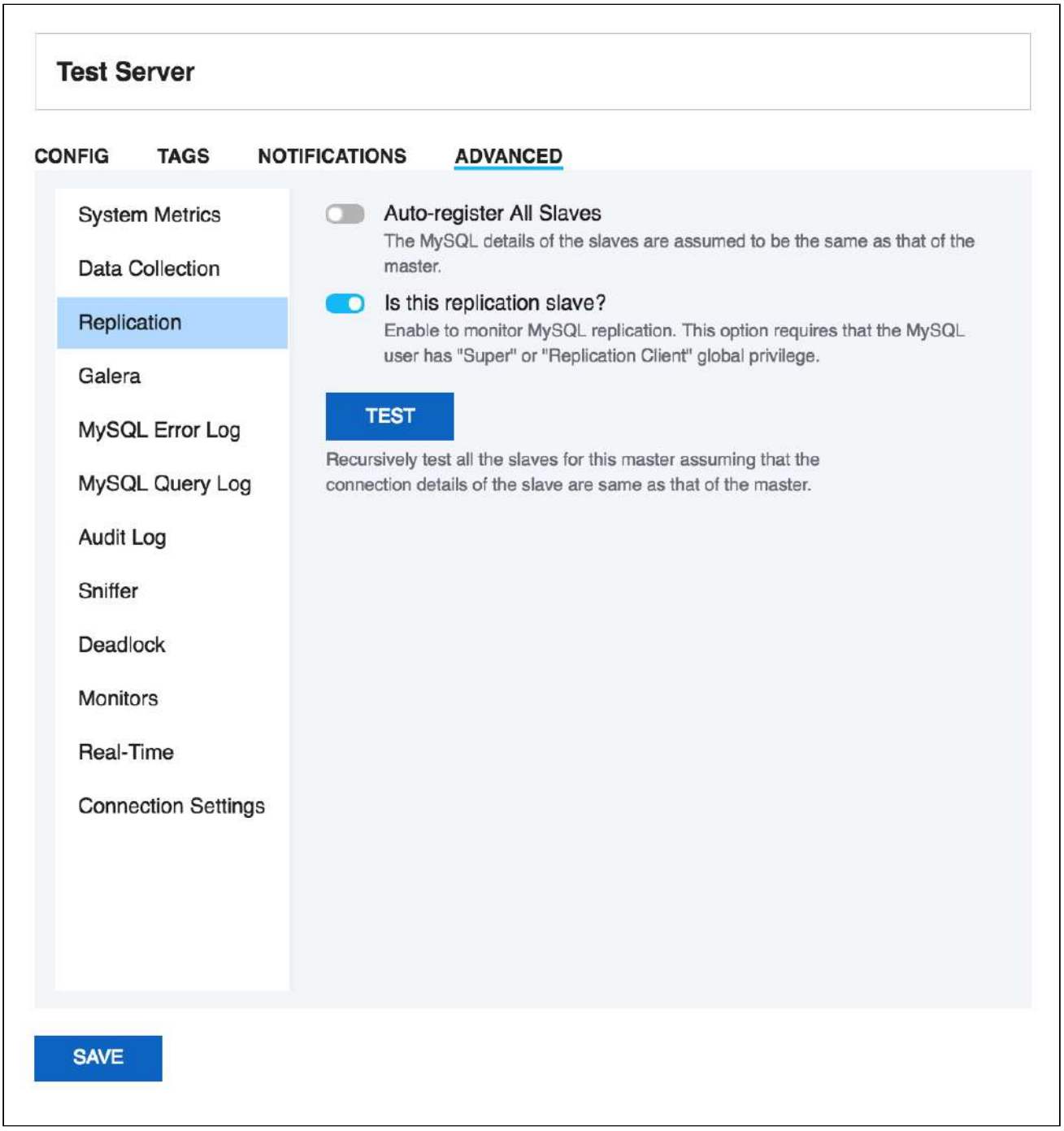
So, to get true uptime-based counters in SQL DM for MySQL with servers that do not support the `uptime_since_flush_status` variable you need to define a 'base time' in SQL DM for MySQL greater than or equal to the time where `FLUSH STATUS` was executed last time.

But also if `uptime` and/or `uptime_since_flush_status` is large ('old') you may use 'base time' setting to analyze uptime-based counters on an interval defined by you. For instance, if the server has been running for months you may choose to analyze uptime-base counters based on data collected from a specific time only as you have defined it.

Also, note that if the 'base time' is smaller than `uptime` (or `uptime_since_flush_status` if available), then 'base time' setting is ignored. Using a 'base time' larger than 'uptime' and/or `uptime_since_flush_status`, then base time is considered. If a base time is in future, then most recent collection time is considered (similar to Delta).

Replication Settings

SQL DM for MySQL monitors MySQL replicas by issuing a SHOW SLAVE STATUS on the slaves. SQL DM for MySQL can also auto-register slaves, given the master details.



Replication Slave

Select the toggle switch for Is this a replication slave? To monitor MySQL replication. This option requires that the MySQL user has "Super" or "Replication Client" global privilege.

Automatic registering of slaves

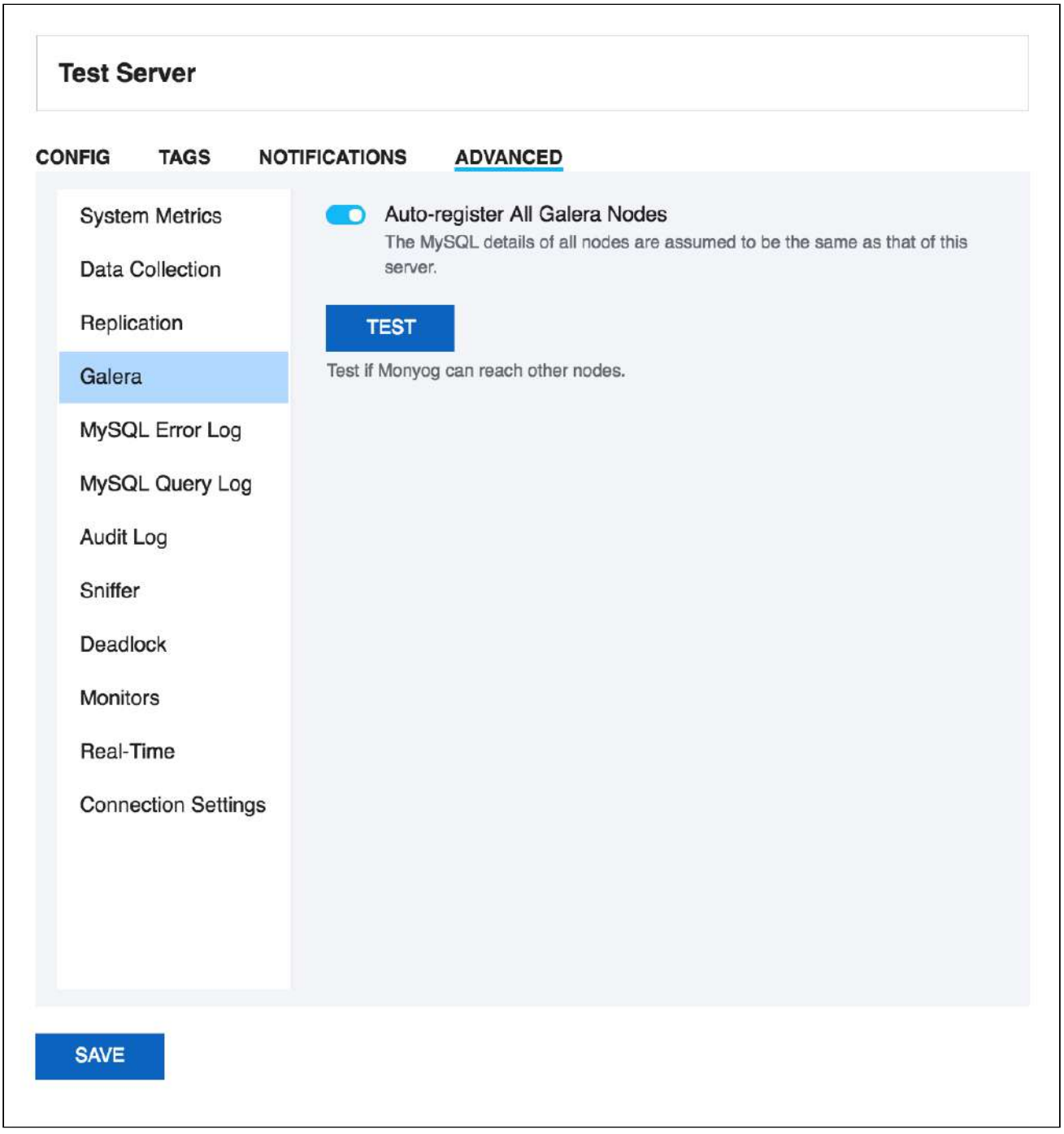
This feature of SQL DM for MySQL saves you time from registering each slave individually. In order for SQL DM for MySQL to auto-register all slaves, select the toggle switch to enable **Auto-Register slaves** in the Advanced settings tab while registering a slave. If a master is already registered, click **Edit Server,** and check the Auto-Register slaves in the Advanced settings tab. The MySQL and the SSH details of the slaves are assumed to be the same as that of the master. In case the slave details are different from that of the master, you have to manually edit that server and change details.

The auto-registering of slaves is extended to multiple levels of replication. For instance, lets say Server A is a Master that has Server B as the slave. And Server B has Server C as its slave. In such case, while registering Server A, if you check **Auto-Register Slaves**, then it registers A, B, and C provided the MySQL and the SSH details of the A is same as that of B.

How does SQL DM for MySQL auto-register all slaves of a given master? SQL DM for MySQL shoots a SHOW FULL PROCESSLIST on the master, and checks for all the slaves connected. (It assumes that MySQL and the SSH details of the slaves are the same as that of the master.) To view replication topology click the Replication tab(see page 148).

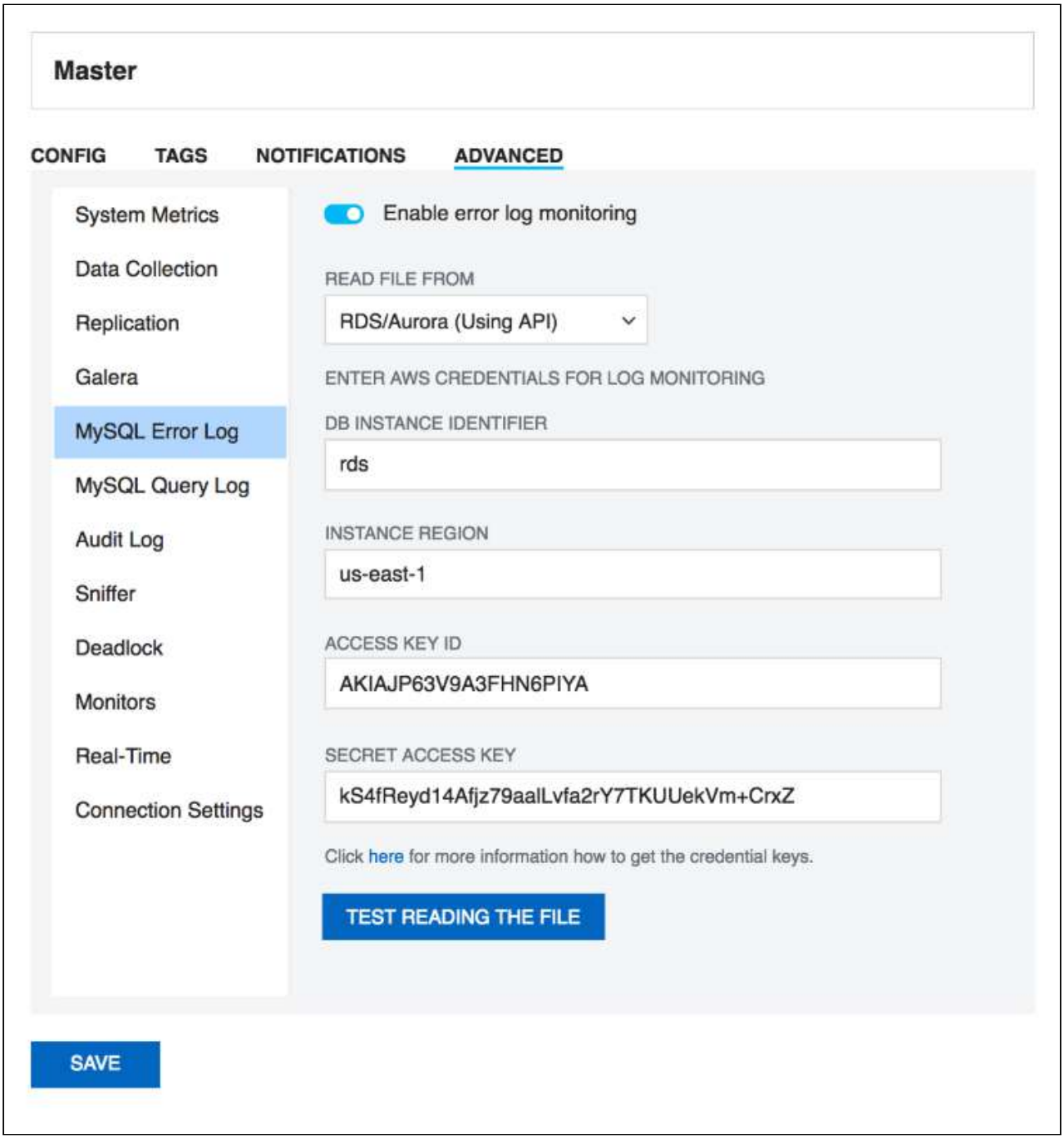
Galera

Use this option to Auto register all the Galera nodes of your cluster with SQL DM for MySQL. The MySQL and the SSH details of the nodes are assumed to be the same as that of the node on which you are enabling this option. In case the other node details are different from that of the node on which you are enabling this option, you need to manually edit that server, and change details. You can do a **Test** to check if SQL DM for MySQL is able to connect to the other nodes. If the **Test** gives a successful message then you can go ahead and click **Save**. SQL DM for MySQL registers the detected nodes and redirects you to the Servers page where you can see all the registered nodes.



MySQL Error Log Settings

The MySQL error log is quintessential in determining the health of the server. You can **Enable error log monitoring** to allow SQL DM for MySQL to keep an eye on your MySQL Error Log and notify you of important information.



Enable error log monitoring:

Select the toggle switch to **Enable error log monitoring**.

Read file from:

There are 3 ways of accessing the log files: Select ***"Local path"*** if the logs are in the machine where SQL DM for MySQL is running, or if they can be accessed by SQL DM for MySQL on a shared network drive. Choose ***"Via SFTP"*** if

you have configured SQL DM for MySQL to use SSH. Select ***"RDS/Aurora (Using API)"*** if your server is a RDS/Aurora instance. In case of RDS/Aurora (Using API) for file based logging, four additional fields have to be filled, which are:

- **DB instance identifier:** A unique name to identify your RDS/Aurora instance.
- **Instance region:** The region in which your instance is hosted, for e.g: us-east-1
- **Access key ID:** It is a 20 character long key ID which can be created from the AWS Management console. It is used to make programmatic request to AWS.
- **Secret access key:** It is 40 character long and can be created from the AWS Management console. You can refer to the documentation, on how to generate credential keys here: Getting Your Access Key ID and Secret Access Key(see page 55).

Fetch error log details:

SQL DM for MySQL can automatically get the path of the error log from the MySQL server. Just click the fetch button, and SQL DM for MySQL will do the rest for you.

File path:

If you choose to enter the error log file path manually, you may do so here.

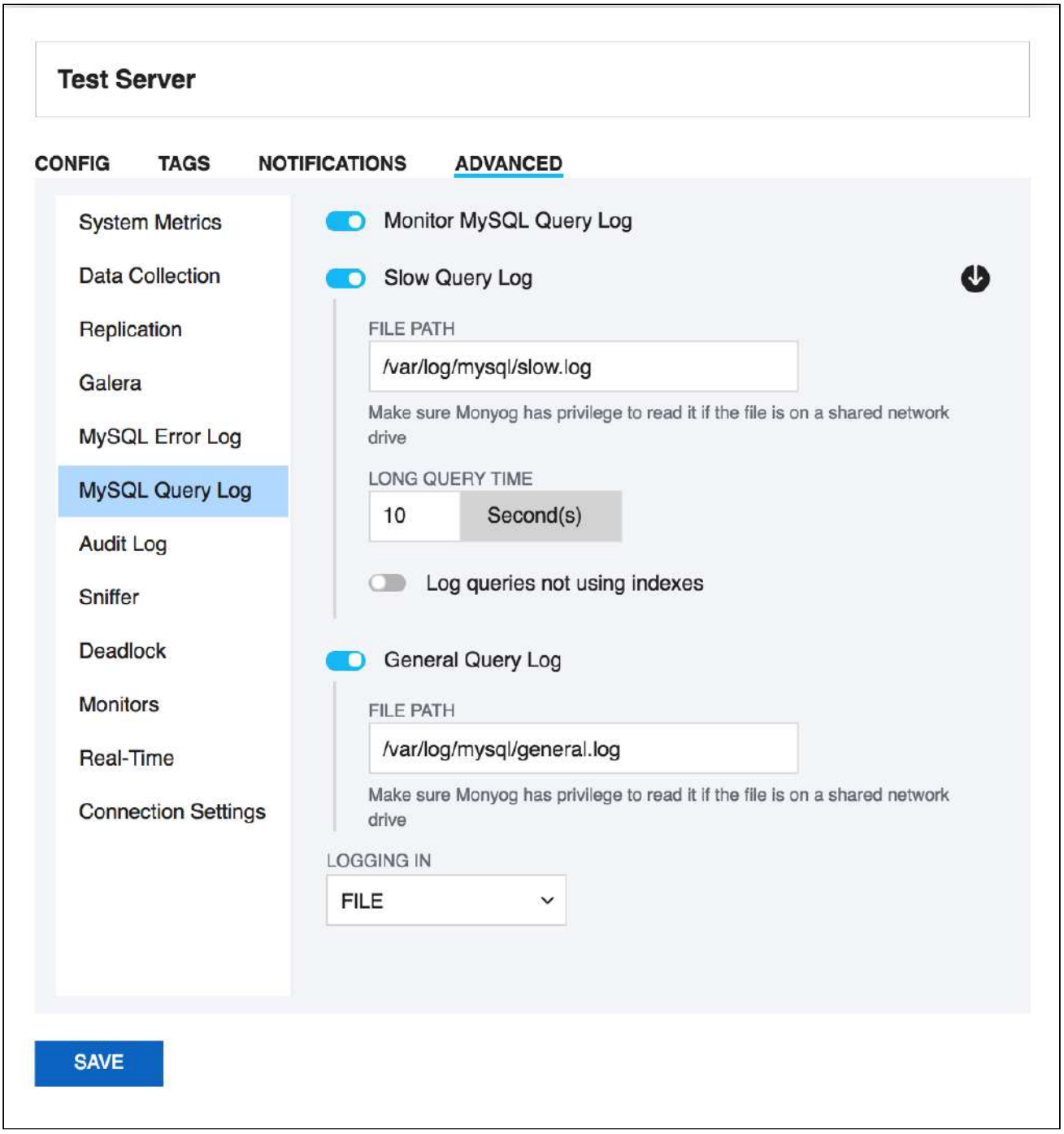
Test path:

Click this button to check if SQL DM for MySQL can access the file specified in the File path.

MySQL Query Log

SQL DM for MySQL retrieves (completely or partially) the General query log and the Slow query log from the MySQL servers it connects to, and analyzes them. Here, we see how to set up details for the connection, so that log analysis is available with SQL DM for MySQL. You have to set up details for the general query log and the slow query log independently. Enabling **Slow Query Log** 'log queries not using indexes' instead needs SUPER privilege. Refer to the MySQL documentation on how to enable and configure logging. MySQL server logs can be written to files on the server machine or to tables in the MySQL database itself.

The MySQL server (since version 5.0) has an option to log (in the slow log) queries that do not use an index. Such queries need not be slow if there are only a few hundred or few thousand records in the table(s) involved. But they are 'potentially slow' and should be identified if they access tables,which will continue to grow. You can enable and disable this from here too (SQL DM for MySQL will send the appropriate SET of statements to MySQL) Note: Only DML and DDL queries are recorded in the slow query log.



Logs written to files:

First, lets consider the situation where server logs are stored as files on the server machine. This is the most common situation and the only one available with MySQL servers before version 5.1. First time you configure a server with this option, click the **Fetch query log details** button. The MySQL server knows (it is stored in server variables) what logs are enabled and how logging is configured. Click **Test Path** to verify that the path SQL DM for MySQL connects and verifies the existence of the file (but not its content).

The log files can be accessed from the local file system (if SQL DM for MySQL and MySQL is running on the same computer) or by using SFTP (if SQL DM for MySQL and MySQL is running on different computers) or by using RDS/Aurora (Using API) if you are using a RDS/Aurora instance. Note that you must use the file and path syntax of the machine where the logs are.

If the log files can be accessed from a shared drive, over a network, or from a network enabled file system (like NFS on Linux), then SQL DM for MySQL can access them as if they were local files. No additional SSH/SFTP configuration is required in this case: the operating system takes care of the file transfer transparently.

When **via SFTP** option is chosen, then SSH server details as defined in SSH server details settings are used to read the file from the remote system. Note that the SSH user must have read access to the log files.

When RDS/Aurora (Using API) option is chosen, make sure that you have the required Access credentials with you to fetch the log files. The Access credentials can be generated from the AWS Management Console. You can refer to the documentation, on how to generate credential keys here: Getting Your Access Key ID and Secret Access Key(see page 57).

By default, MONyog(SQL DM for MySQL) service runs under Local System Account. If you have Slow query or General query logs in a Mapped Network Drive, SQL DM for MySQL is not be able to reach it. You need to use UNC notation for SQL DM for MySQL to be able to access them. See FAQ 31(see page 226) for details.

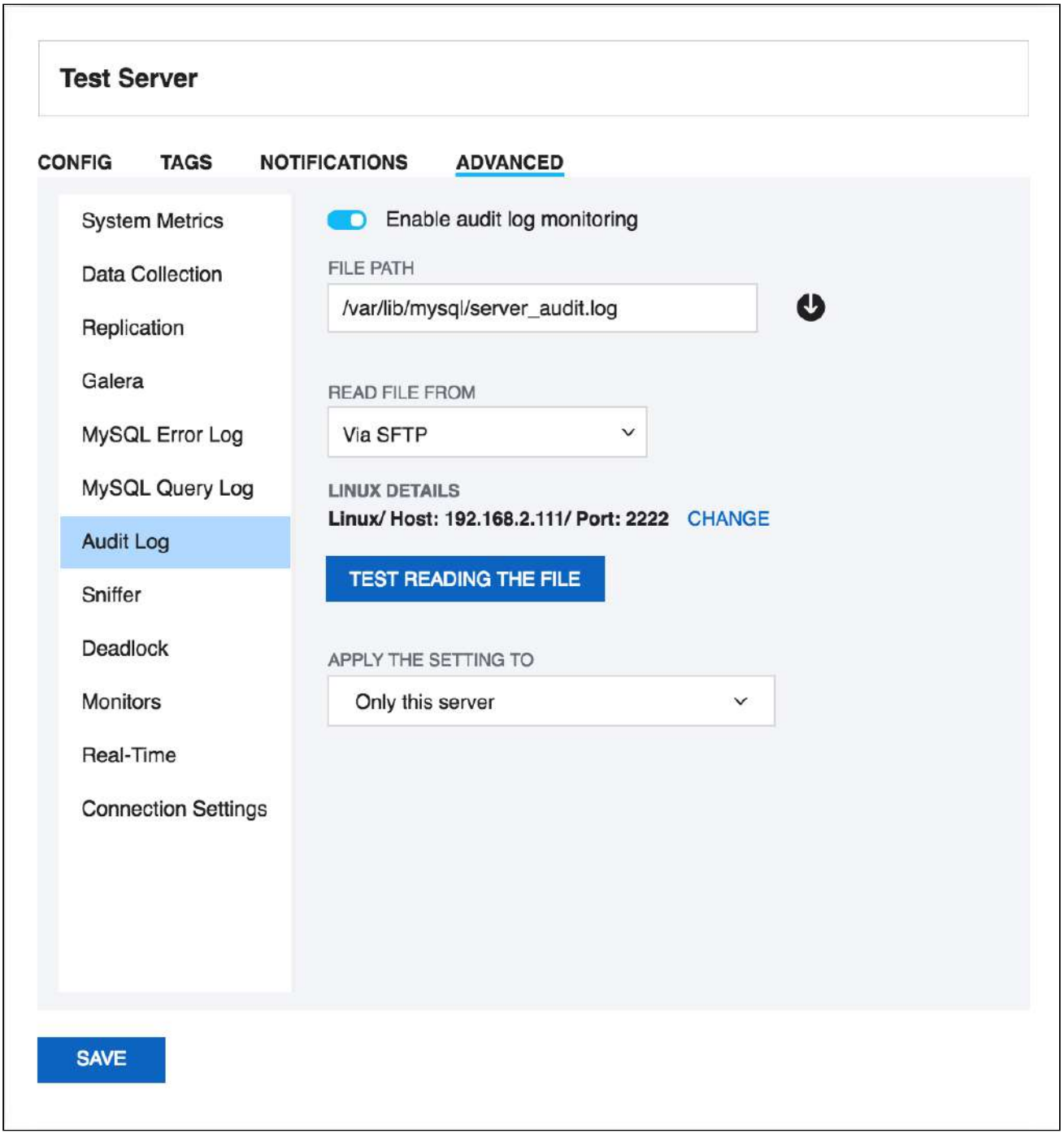
Logs written to MySQL tables:

This option is supported by MySQL from version 5.1. Also, SQL DM for MySQL supports when this option is available. Here, you click the **Fetch Log Details From MySQL** button. When this option is used there is no file path to configure and no SSH details to consider. SQL DM for MySQL can retrieve the server log by sending simple `SELECT` statements. Only the MySQL user used by SQL DM for MySQL to connect to MySQL must have `SELECT` privileges to the tables.

Audit Log Settings

This works exactly the same way as the MySQL Error log and MySQL Query Log. On enabling **audit log monitoring**, SQL DM for MySQL fetches the path from the server and displays it in the **File Path** box. Note that, we get the path from the variable "server_audit_file_path", and by default it just returns the audit log file name. In such cases, you have to manually enter the path for the audit log (by default, the path is same as datadir path).

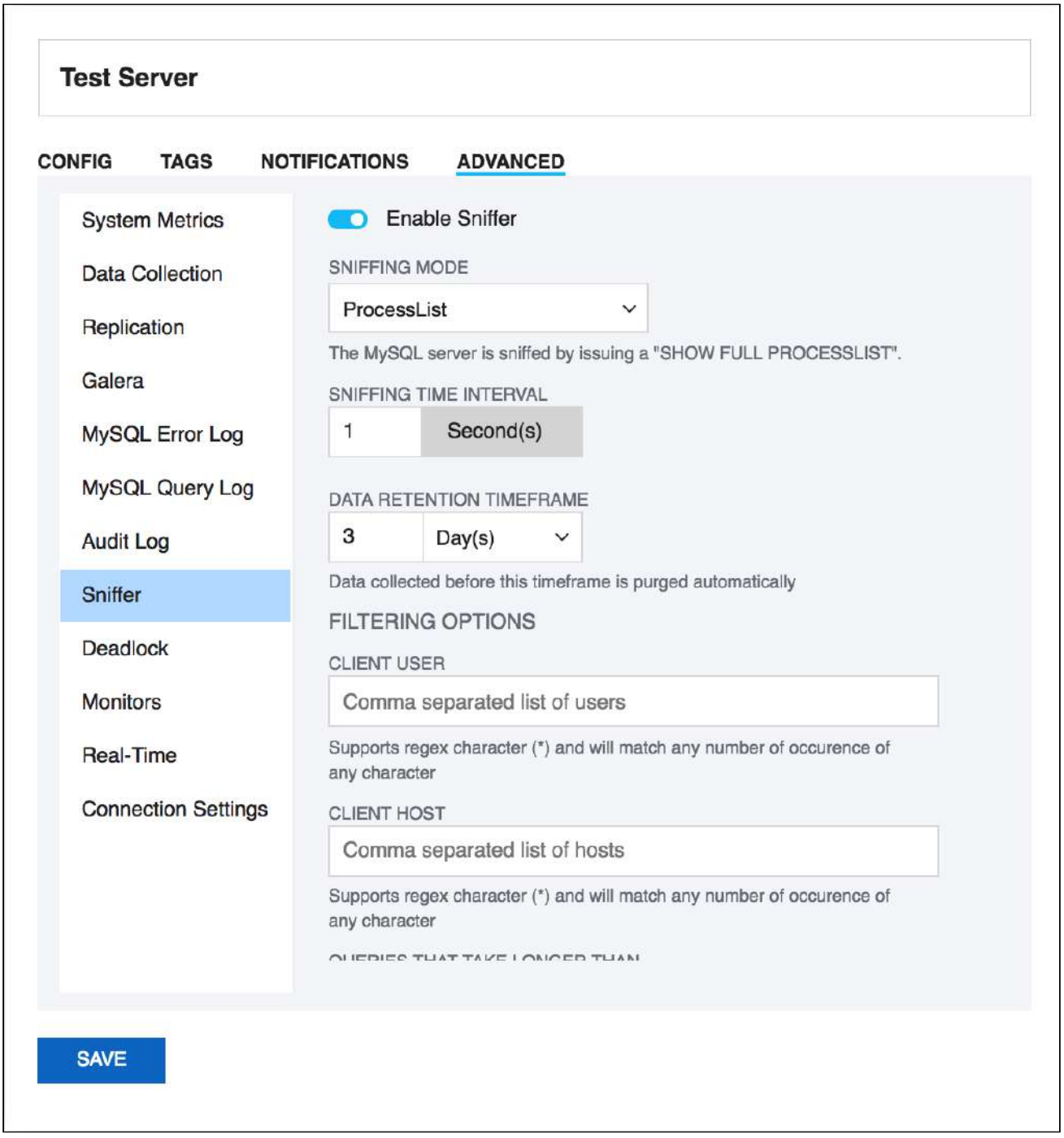
Next, depending on where you have the MySQL server running, select an appropriate option for **Read File From**. If the server is on the same machine as SQL DM for MySQL, choose **Local path**. If it is on a remote machine, choose **Via SFTP** and give the corresponding SSH details. If the server is a RDS/Aurora server, then choose **RDS/Aurora (Using API)**.


Test Server
CONFIG
TAGS
NOTIFICATIONS
ADVANCED
System Metrics
Data Collection
Replication
Galera
MySQL Error Log
MySQL Query Log
Audit Log
Sniffer
Deadlock
Monitors
Real-Time
Connection Settings
Enable audit log monitoring
FILE PATH
/var/lib/mysql/server_audit.log
READ FILE FROM
Via SFTP
LINUX DETAILS
Linux/ Host: 192.168.2.111/ Port: 2222
CHANGE
TEST READING THE FILE
APPLY THE SETTING TO
Only this server
SAVE

Sniffer Settings

SQL DM for MySQL query sniffer is a functionality that records a pseudo server log and stores it in the SQL DM for MySQL embedded database. With **Sniffer** enabled, SQL DM for MySQL can populate the pseudo server log in three different ways at the intervals you specify:

- By utilizing Performance Schema tables (events_statements_summary_by_digest, events_statements_history_long) and collecting snapshots at regular intervals.
- By sending the query SHOW FULL PROCESSLIST to the MySQL server.
- Or by connecting to a running instance of the MySQL-Proxy program that is used by one or more clients to connect to a MySQL server.



For MySQL 5.6.14 and above you can use Performance schema(if Performance Schema is enabled), Proxy and Processlist for query analysis. If using MySQL version less than 5.6.14 then you can use Processlist mode.

Performance Schema on MySQL contains queries executed on server along with other information

- Number of rows sent and examined
- Number of temporary tables created on disk
- Number of temporary tables created on memory
- Number of joins performed and the type of join

- Whether sorting happened and the type of sort
- Whether index used
- Whether good index used

SQL DM for MySQL uses performance schema statement digest `table(events_statements_summary_by_digest)` to get the above information and is dependent on the `statements_digest` in `setup_consumers` table. By default, this is enabled. If not, it can be enabled by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'statements_digest';
```

Example query is available in `events_statements_history_long` table and has to be enabled and is dependent on the `events_statements_history_long` in `setup_consumers` table. By default, this is not enabled and should be enabled by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'events_statements_history_long';
```

The `performance_schema.events_statements_summary_by_digest` table size is dependent on `performance_schema_digests_size` global variable. By default, this size is set to 5000 rows. Once it reaches this limit you may lose the queries. SQL DM for MySQL provides an option to truncate the performance schema digest table when it reaches 80% of `performance_schema_digests_size`.

Although configuring a Proxy instance is a little more complicated, the PROXY-based sniffer has several advantages over the PROCESSLIST-based, including:

1. All queries that was handled by the Proxy will be recorded by SQL DM for MySQL sniffer when PROXY option is used. When PROCESSLIST option is used very fast queries may execute completely between two SHOW FULL PROCESSLIST queries and will then not be recorded.
2. You can choose to analyze queries from specific client(s)/application(s) only. Simply let (only) the clients that you want to focus on at the moment connect through the Proxy.
3. When using the PROXY option you can distribute part of the load generated by the sniffer on the machine that fits best in your deployment scenario (like on the one that has most free resources available) by deciding where to have the PROXY: The MySQL machine, the SQL DM for MySQL machine (if not the same) or quite another machine. The machine running MySQL will have no additional load due to the sniffer if the Proxy is not running on that machine.

Also note that, if more SQL DM for MySQL instances use the same PROXY they use the same data collected, when the Proxy Sniffing is enabled by the first SQL DM for MySQL instance. To work with SQL DM for MySQL sniffer the MySQL Proxy instance must be started with the name of a LUA script called MONyog.LUA (LUA is a scripting/programming language) as argument and is distributed with SQL DM for MySQL. You can find it in the MONyog program folder after installing (Windows and Linux RPM) or unpacking (Linux .tar.gz) the SQL DM for MySQL program package as downloaded from the IDERA website. The MySQL Proxy program however you need to download from MySQL website (we cannot include it for license reasons). SQL DM for MySQL works with Proxy versions from 0.61 to 0.81(latest currently) with the exception of 0.7x versions for windows and Mac due to a bug in those specific builds. For more information on Proxy, see MySQL Proxy(see page 61).

To start a Proxy instance for use with SQL DM for MySQL use the command:

- For v0.81(Alpha) and later, run the following common from the Proxy installation folder:

```
# mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 \
          --proxy-address=192.168.y.y:4045 \
          --admin-username=root \
          --admin-password=root \
          --admin-lua-script=MONyog.lua \
          --proxy-lua-script=MONyog.lua
```

- For Older versions, from the Proxy installation folder, run:

```
# mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 \
          --proxy-address=192.168.y.y:4045 \
          --proxy-lua-script=MONyog.lua
```

(It is assumed that the 'MONyog.LUA' was copied to the folder where the PROXY binary is). Also note that, if no port is specified the PROXY listens on port 4040. Now, you can connect to the Proxy from one or more clients/ applications. The Proxy sends queries to MySQL and the results back to the client. But when started with the LUA script for SQL DM for MySQL sniffer it also sends information to SQL DM for MySQL that SQL DM for MySQL uses to populate the sniffer 'pseudo log'.

Once this 'seudo log has been recorded (in either of the three ways described: Performance Schema, PROCESSLIST or PROXY-based) the SQL DM for MySQL log analysis functionalities can operate on the pseudo log as well as the real logs. The data recorded in the pseudo log is purged automatically based on the data retention timeframe option set by you.

Further some filtering options are provided. This filtering happens before storing to the SQL DM for MySQL database. This prevents the sniffer database to grow out of control. The filtering options are as follow:

- **User and host**: You can choose to store queries executed only by a specific combination of users and/or hosts.
- **Minimum time taken**: For every `PROCESSLIST` returned to SQL DM for MySQL, the queries are recorded in the embedded database only if they have been executing for a time greater than the specified minimum execution time. Furthermore, if a query with the same structure and details (like process ID) as one already recorded is encountered, the embedded database is UPDATED, and the statement is recorded only once.

> ⚠ This setting should be somewhat larger than the sample interval (and also consider the latency of the connection). If set lower it would not really make much sense.

- **Queries starting with**: Enter any string and only queries starting with that string are recorded. Examples: `SELECT *`, `UPDATE Customer_Base`.

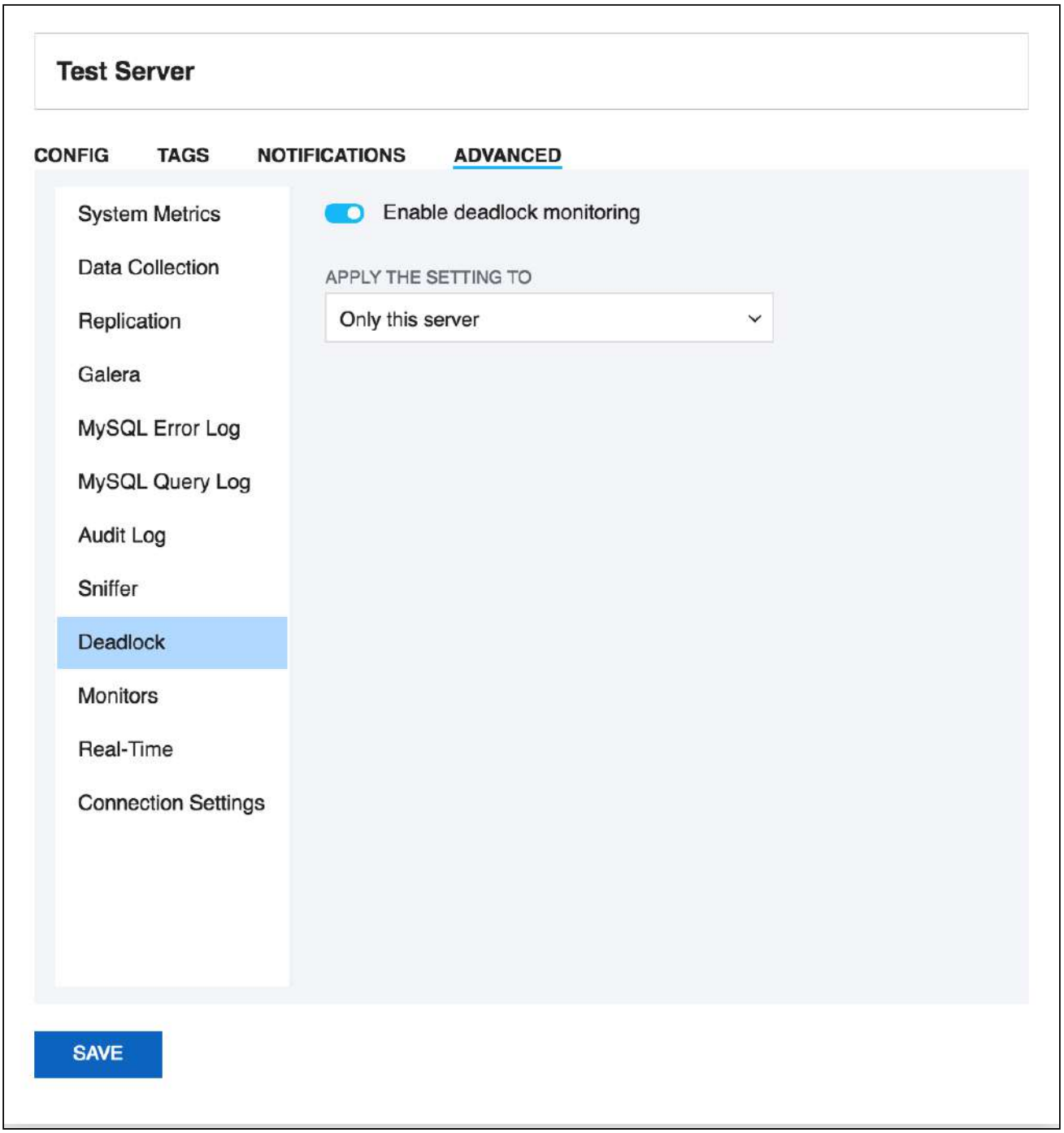
Also note that in `PROCESSLIST` Sniffer we have an option **Long Running Query Options** where you can monitor the long running queries by notifying or killing a query which takes more than a time specified by you. You can also specify users whose queries will be ignored (i.e. queries by such user are never killed by SQL DM for MySQL) and never raise an alert even if they take a longer than the time specified under 'LONG RUNNING QUERY TIME' you specified.

Clicking **Monitor only locked queries** would only monitor those long queries that are locked.

You should note that the query sniffer is not a complete 'general log'. Very fast statements may or may not be recorded as they may or may not finish executing between two PROCESSLISTs generated. The time interval between subsequent data collections for the 'pseudo log' depends on the connection to the MySQL server.

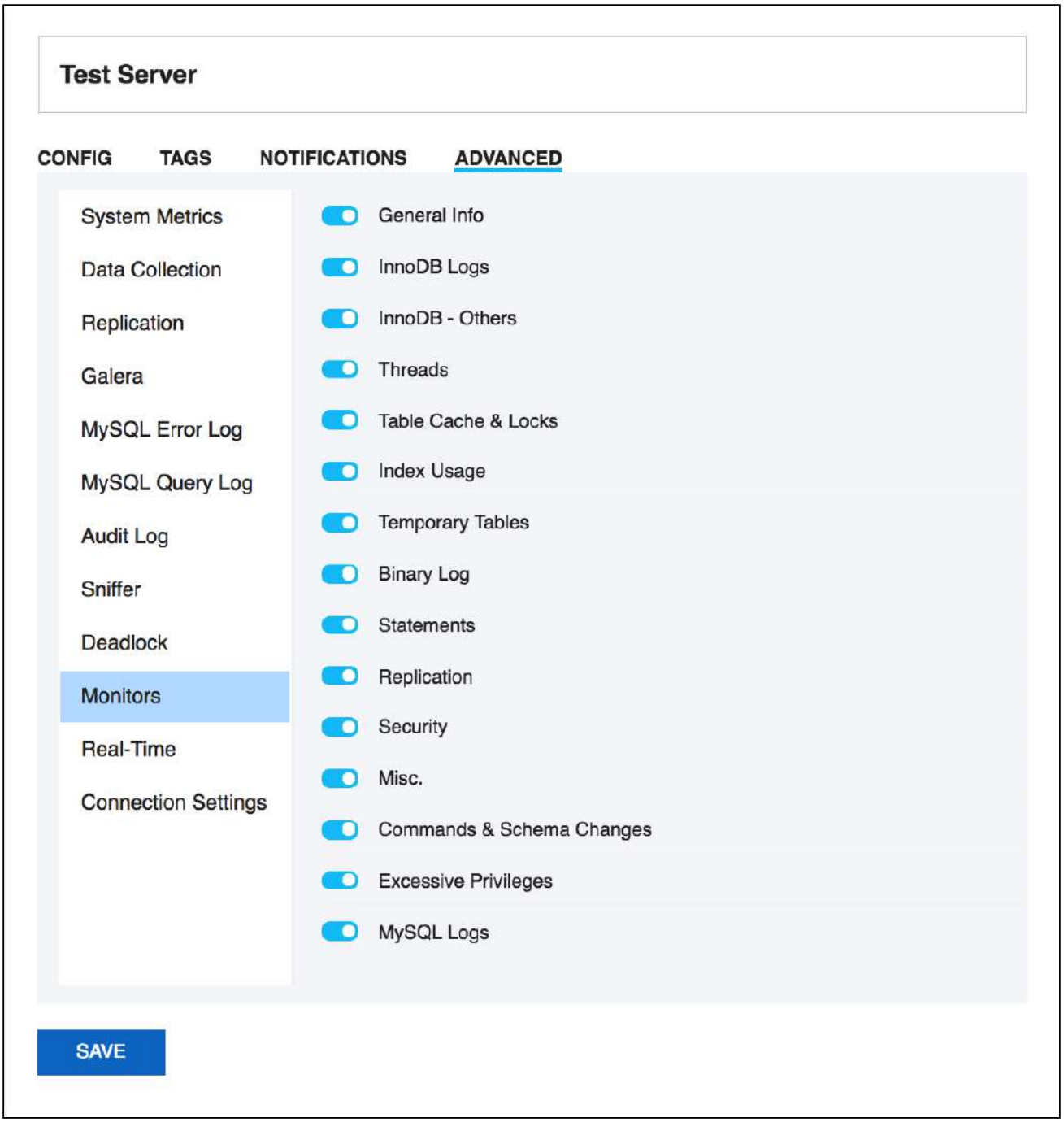
Deadlock Settings

In transactional databases deadlocks are a classic problem, but these deadlocks are not too dangerous unless they are so frequent that you cannot run certain transactions at all. To trace the deadlocks reported by `INNODB STATUS`, you can select the toggle switch to "**Enable deadlock monitoring**" option.



Monitors Settings

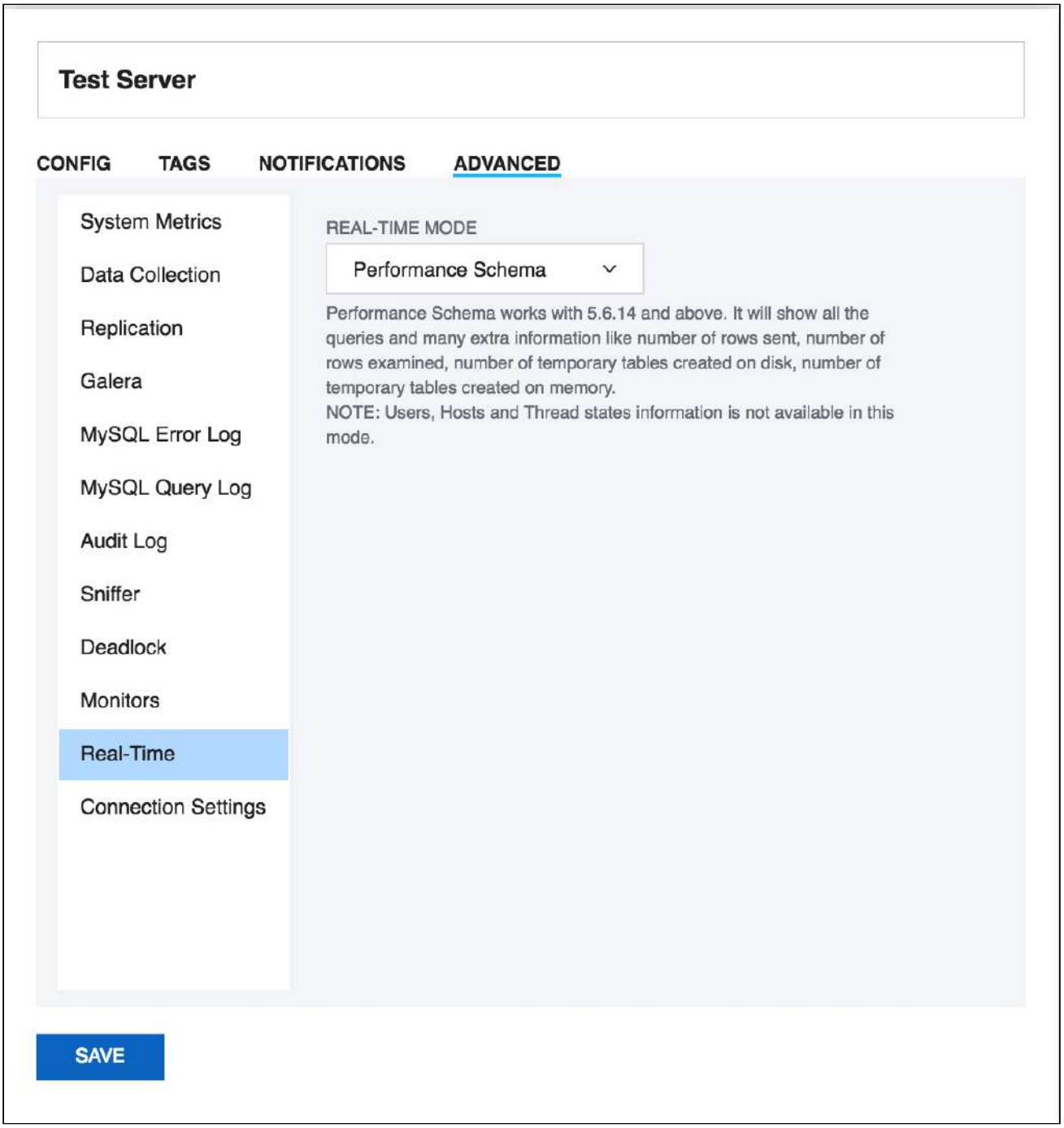
SQL DM for MySQL provides a way of disabling an entire group of Monitors. For instance if a MySQL server is not a replication slave, then the replication group can be disabled.



Real-Time Settings

SQL DM for MySQL provides you the option to choose the data collection mode for Real-time monitoring.

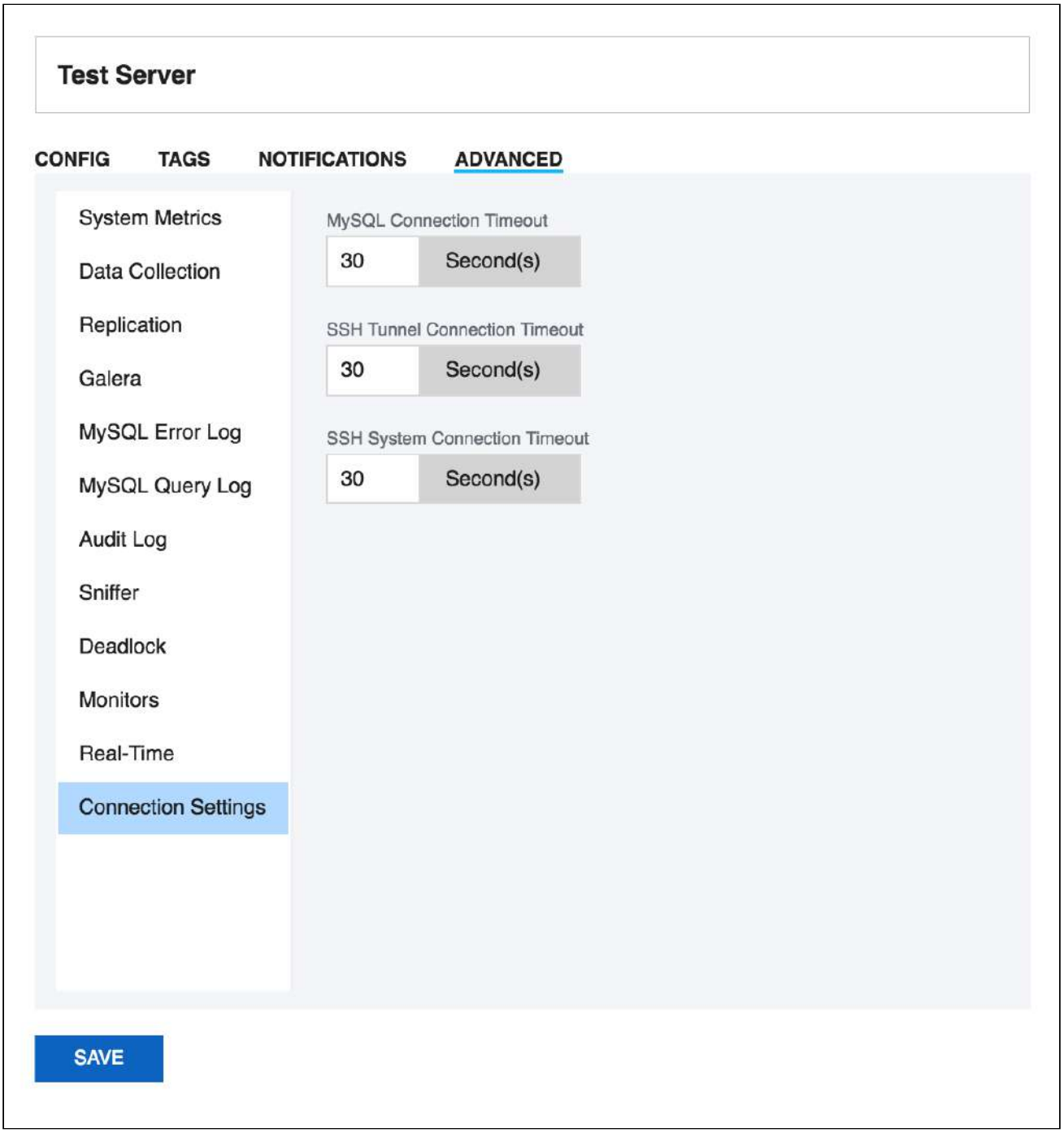
You can choose between **Processlist** and **Performance schema**. Select **Performance schema** mode if your MySQL version is 5.6.14 or above and if performance schema is enabled, otherwise you can go with **Processlist** mode.



Connection Settings

You can specify the **MySQL Connection Timeout** value for your server. This option simply means that SQL DM for MySQL waits for this long to get a response from the server before it throws an error. This comes handy to avoid false positives when connection to some specific servers is slow. You can setup a larger timeout in such cases. If you

have SSH Tunneling enabled to the MySQL then you can specify a **SSH Tunnel Connection Timeout**, and a **SSH System Connection Timeout** if System Metrics is enabled. The default value for all being 30 seconds.



MySQL Privileges

MySQL Privileges

Whether you are running a single MySQL database server or multiple servers in a cluster, SQL DM for MySQL provides a comprehensive list of performance variables that you can monitor in real-time to check the health and performance of your MySQL server or servers.

In order to access the data for these metrics, SQL DM for MySQL requires a dedicated user on each MySQL database server it monitors, with the appropriate privileges to collect the information it needs.

Privileges

Most SQL DM for MySQL functionalities do not require any privileges at all for the user it uses in establishing client connections with MySQL. You can create it without any special global or object privileges, using the following command:

```
GRANT USAGE ON *.*
TO 'user'@'host';
```

Fully enabling all SQL DM for MySQL functionality requires that you grant the user some additional privileges, ensuring that it can access all the data it needs.

> ⚠ For an example of granting full functionality to the SQL DM for MySQL user, see Database Configuration <db-config> below.

Security Counters

SQL DM for MySQL user requires `SELECT` privileges to the `mysql.user` table to enable SQL DM for MySQL security counters.

Additionally, if the server was started with the `skip_show_databases` configuration variable, you must also explicitly grant the `SHOW DATABASES` privilege.

Replication Counters

In order to collect replication metrics from a slave, the MySQL user on the slave requires the `REPLICATION_CLIENT` privilege. You can also enable this functionality using the `SUPER` privilege.

When registering MySQL servers with SQL DM for MySQL, there is an option **Is this a Replication Slave?** to include replication information. You must set this to **Yes**, when registering the server. It defaults to **No**, when set to No, SQL DM for MySQL ignores replication data.

When testing the connection from the server registration page, SQL DM for MySQL displays the error message:

```
Access denied.  You need the SUPER/REPLICATION CLIENT
privilege for retrieving REPLICATION details!
```

You can ignore this error if the given MySQL server is not a replication slave or if you do not want to monitor replication.

InnoDB Deadlock Monitoring

Depending on the MySQL server version, the SUPER or PROCESS privileges allow you to monitor for deadlocks with the InnoDB storage engine.

> ⚠ Beginning in the 5.1.24 version of MySQL, you can enable this feature using PROCESS. Older versions require SUPER.

Processlist Feature

In order to collect data on the MySQL server processes for all users, you must enable the PROCESS privilege. Additionally, when granted the SUPER privilege, you can use SQL DM for MySQL to kill running processes.

In order to use the EXPLAIN option on the processlist, you need to grant SELECT and SHOW VIEW to the objects accessed by the statements you want to explain, or simply, grant the global SELECT privilege.

> ⚠ SQL DM for MySQL displays N/A on counters where the user lacks the required privileges and logs a record of the MySQL server error for every attempt to retrieve those counters from the server. Be aware the log may grow considerably if this is the case.

Performance Schema-based Sniffer

Collecting data on the Performance schema requires SELECT, DROP and UPDATE privileges.

SQL DM for MySQL uses the SELECT privilege to read the Performance Schema tables. The DROP privilege allows it to truncate these tables. The UPDATE privilege enables the statement digest and statement history log in the performance schema tables.

Log Retrieval

In order to retrieve log information when stored in a table, (which is supported in version 5.1 and newer of MySQL), the SQL DM for MySQL user requires the SELECT privilege on the log tables.

Flush Status

In order to execute FLUSH STATUS commands, the SQL DM for MySQL user also requires the RELOAD privilege.

System Privileges

To fully enable all SQL DM for MySQL functionalities you may also need to create one or more operating system users for specific purposes.

- **SSH tunneling:** SQL DM for MySQL can send and receive encrypted authentication information as well as monitoring data from MySQL using SSH tunneling. Also, it is possible to connect to MySQL with SSH tunneling even if the MySQL port (normally 3306) is blocked by a firewall or if users are not allowed to connect from remote hosts. An operating system user is required so that SQL DM for MySQL SSH client functionalities can use this user to connect to the SSHD daemon on the server.

- **System information (currently available for Linux servers):** For retrieving system information you need an Operating System user. The user must be a 'shell' user (it must be possible for this user to issue system commands). This user is specified for the SSH-based connection to the host.

This user must have read-access to the Linux '/proc' folder. Normally, this is always the case for Linux users. Also to retrieve memory related information this user must have read access to the .pid files created by the MySQL server. An easy way to ensure this is to add the user to the `mysql` user group.

- **MySQL log information:** For retrieving MySQL log information (when stored as files - not when stored as MySQL tables) read-access to the MySQL log files are also required.

You may (but need not) use the same operating system user for all the three above ways that SQL DM for MySQL connects to the OS.

RDS OS and File based Log Monitoring Privileges

AWS IAM Policy can be used to create permissions that specify which RDS actions a user, or a group of users in your AWS account can perform. IAM Policy is basically a JSON document that consists of one or more statements which defines the action to be taken on AWS resources. It can be used to determine who is allowed to create, delete, or modify RDS instances.

SQL DM for MySQL needs the following permissions to fetch the log files:

1. DescribeDBLogFiles: This API fetches a list of log files available for your instance.

2. DownloadDBLogFilePortion: This API downloads the specified log file.

For fetching the OS metrics, the following permission is needed:

1. GetMetricStatistics: This API fetches the average of CloudWatch metrics.

You can create a simple IAM Policy, giving the above permissions, for e.g:

```
{
   "Version":"2012-10-17",
   "Statement": [{
      "Effect":"Allow",
      "Action": [
         "rds:DescribeDBLogFiles",
         "rds:DownloadDBLogFilePortion"
         "cloudwatch:GetMetricStatistics"
      ],
      "Resource":"*"
   }]
}
```

You can restrict the Resource for the above policy using this link(see page 72). In case, you want to perform either OS monitoring, or file-based log monitoring for your RDS/Aurora instance then you can include only those actions in the above policy.

You can use the default policy **CloudWatchReadOnlyAccess** provided by AWS for OS monitoring, in case you do not want to create a custom policy. Keep in mind that this policy grants more permissions than SQL DM for MySQL requires to fetch your RDS/Aurora metrics.

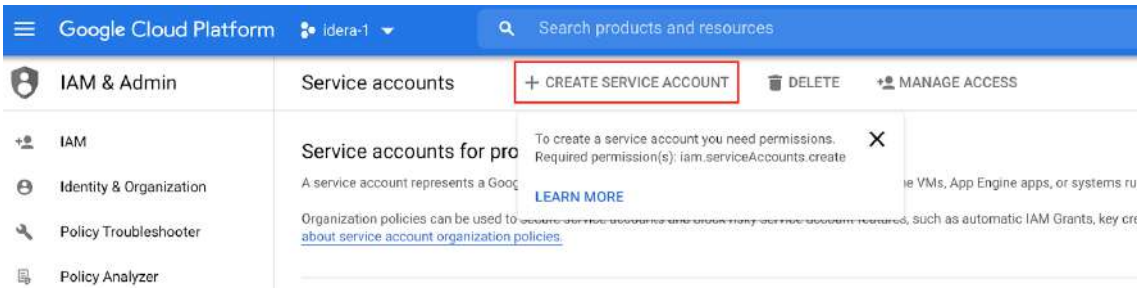
Google Cloud SQL Monitoring

For Google Cloud Monitoring you need the following prerequisites:

1. to create a service account in your Google Cloud account and download the content of the JSON file

   To generate service-account credentials and view the public credentials already created, follow these steps:

   - Open the Service accounts page[14]. When prompted, select a project.
   - Click **CREATE SERVICE ACCOUNT**.



   - In the **CREATE SERVICE ACCOUNT** window, type a name for the service account, and select **Furnish** a new private key. Click **Create**.
   - When you create a key, your new public/private key pair is generated and downloaded to your machine.

2. To configure access to your Cloud SQL instance

   Go to the Cloud SQL Instances page[15] in the **Google Cloud Console**.

   - From the client machine, use What's my IP[16] to see the IP address of the client machine.
   - Copy that IP address.
   - Go to the Cloud SQL Instances page[17] in the **Google Cloud Console**.
   - Open the instance **Overview** page, and record the IP address.
   - Select the **Connections** tab.
   - Under **Authorized networks**, click **Add Network** and enter the IP address of the client machine.

     The IP address of the instance and the MySQL client IP address you authorize must be the same IP version: either IPv4 or IPv6.

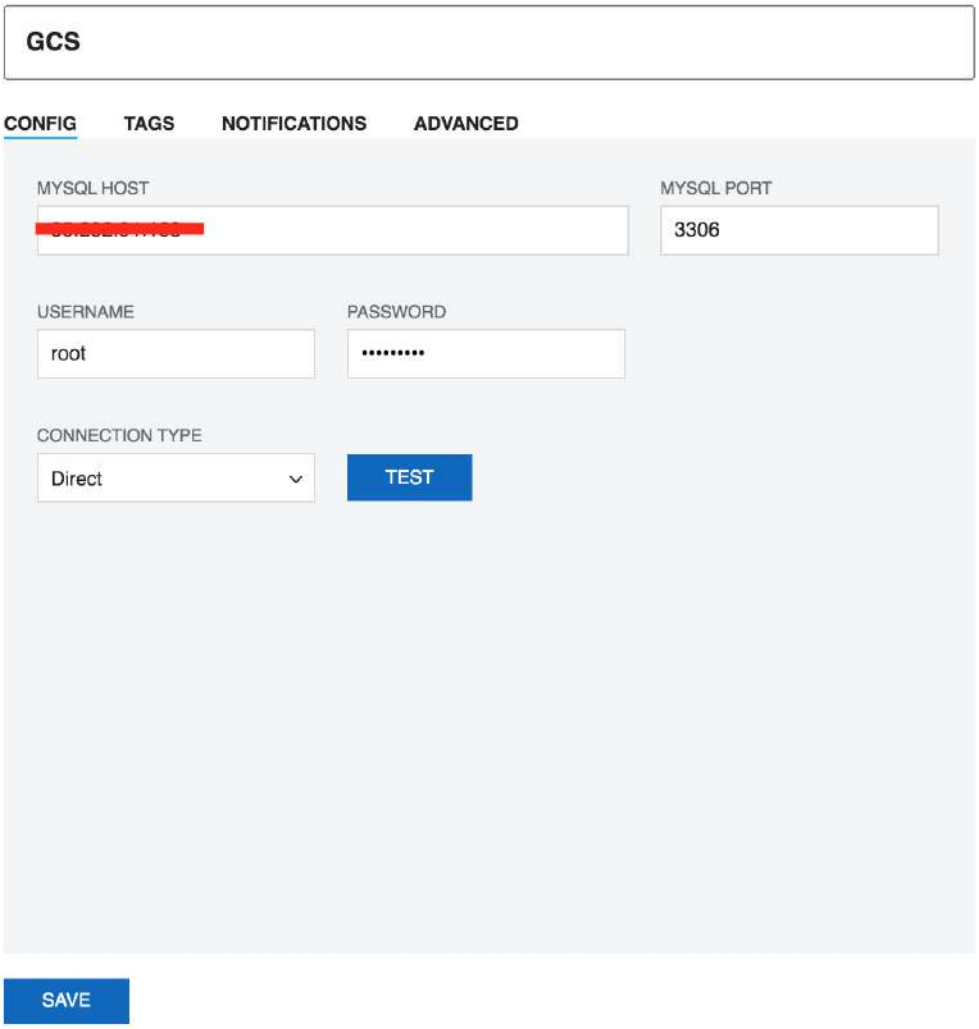
14 https://console.cloud.google.com/iam-admin/serviceaccounts
15 https://console.cloud.google.com/sql/instances
16 http://ipv4.whatismyv6.com/
17 https://console.cloud.google.com/sql/instances

- Click **Done** and then **Save** at the bottom of the page to save your changes.

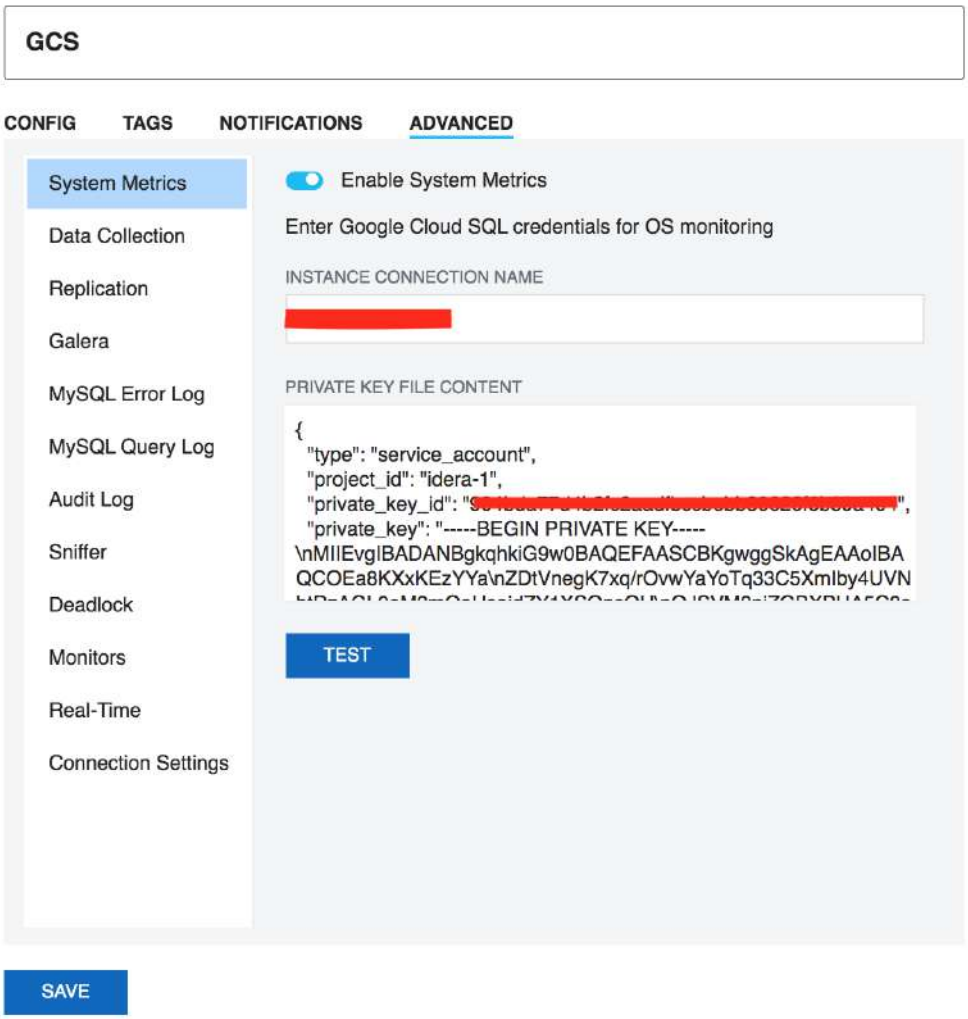
Adding your Google Cloud Server

To add your **Google Cloud Server** to SQL DM for MySQL, you need to follow these steps:

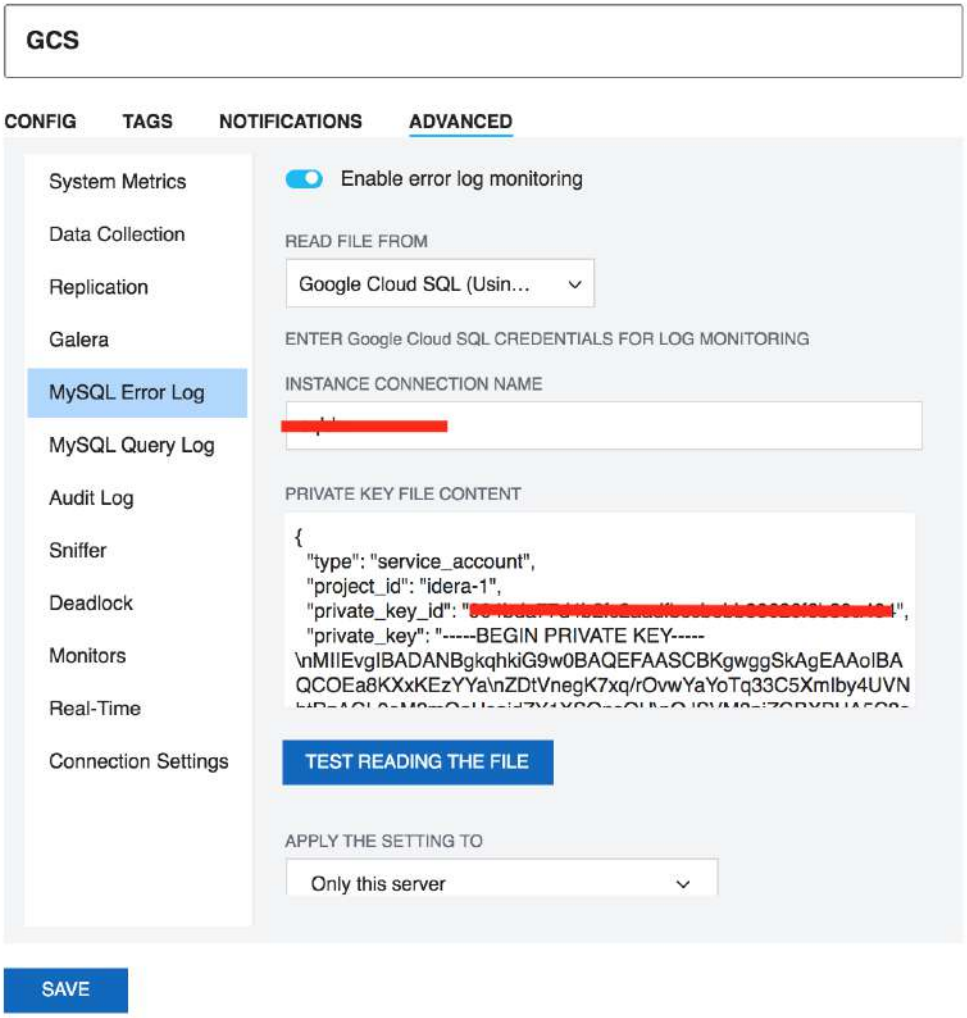
1. Provide the Host IP, MySQL Username, and Password to register your Google cloud SQL server in SQL DM for MySQL.



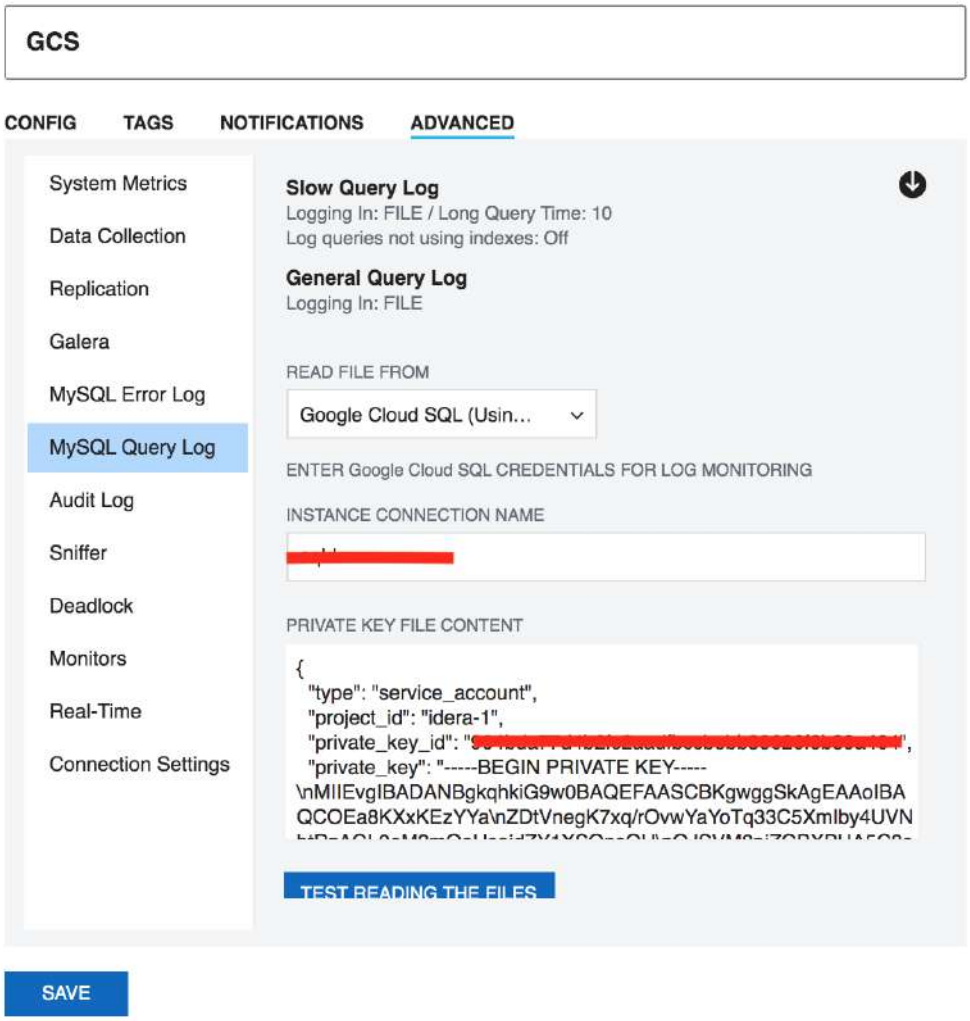
2. Enable **System Monitoring** and enter the instance name with the content of the private key JSON file. Go to Edit Server > Advanced > System Metrics.



3. To enable **MySQL Error Log** monitoring, review the below screenshot:

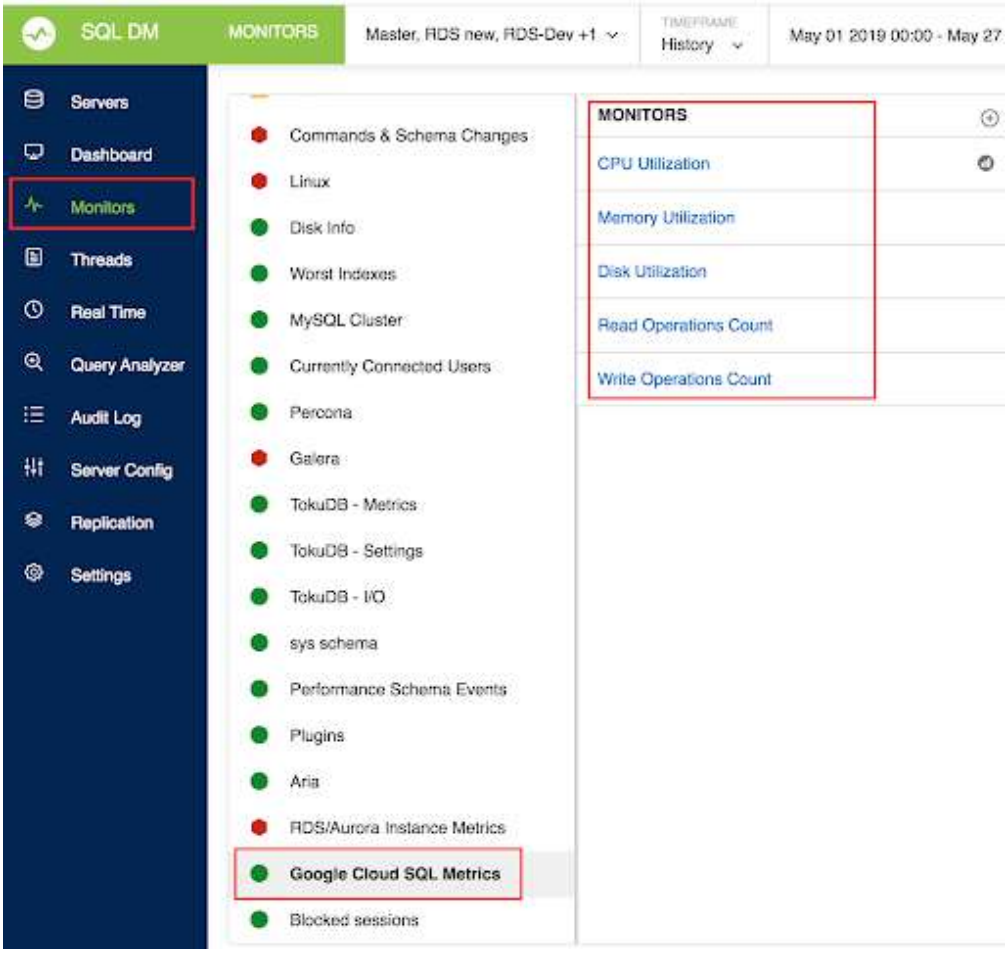


4. To enable **MySQL Query Log** monitoring, review the below screenshot:



5. To enable **GCS Objects**, go to the **Monitors** page, click "**+**" near Monitors, and select **Manage Google Cloud SQL Objects**.

6. You can enable the below Monitors to monitor the **Google Cloud SQL Metrics**:
   - database/cpu/utilization
   - database/memory/utilization
   - database/disk/utilization
   - database/disk/read_ops_count
   - database/disk/write_ops_count


SQL DM
MONITORS
Master, RDS new, RDS-Dev +1
TIMEFRAME
History
May 01 2019 00:00 - May 27
Servers
Dashboard
Monitors
Threads
Real Time
Query Analyzer
Audit Log
Server Config
Replication
Settings
Commands & Schema Changes
Linux
Disk Info
Worst Indexes
MySQL Cluster
Currently Connected Users
Percona
Galera
TokuDB - Metrics
TokuDB - Settings
TokuDB - I/O
sys schema
Performance Schema Events
Plugins
Aria
RDS/Aurora Instance Metrics
Google Cloud SQL Metrics
Blocked sessions
MONITORS
CPU Utilization
Memory Utilization
Disk Utilization
Read Operations Count
Write Operations Count

Navigate SQL DM for MySQL

SQL DM for MySQL is an agentless MySQL monitoring tool that helps DBAs reduce downtimes, tighten security, and optimize performance of your MySQL powered systems. Other characteristics can include reducing excessive CPU usage, query execution time, fast identification of queries causing performance issues, and high availability among others.

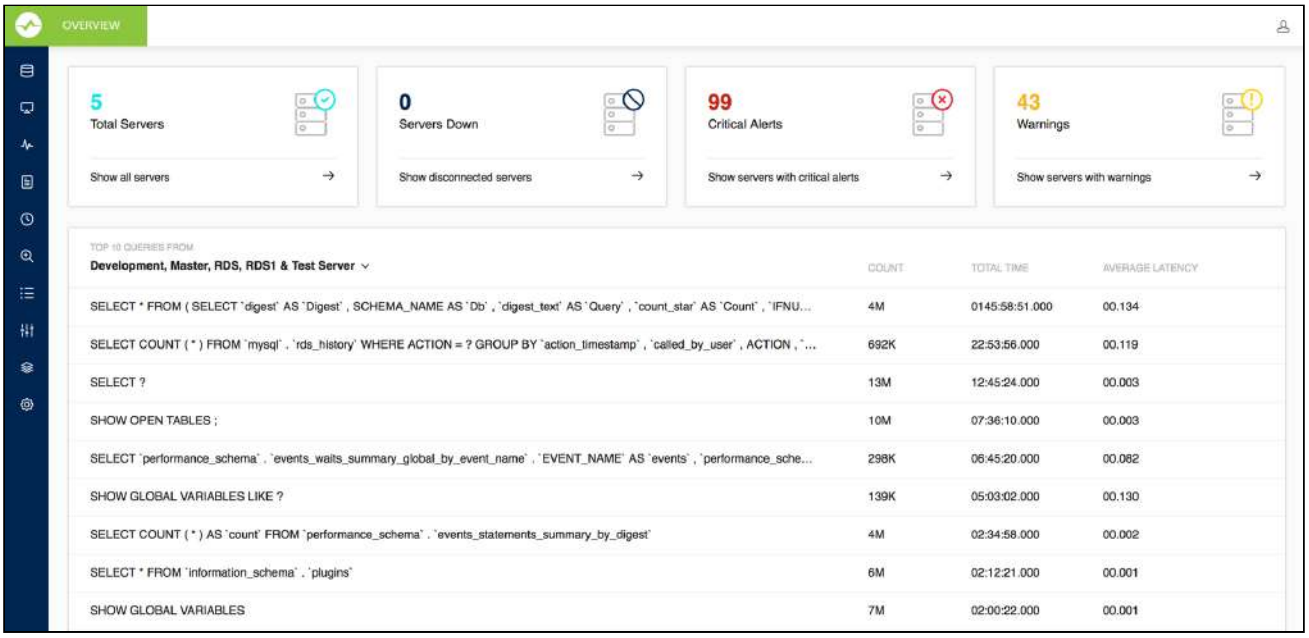
Below, review the different views and options that SQL DM for MySQL offers:

- Overview[18]
- Servers[19]
- Dashboard[20]
- Monitors[21]
- Threads[22]
- Real-Time[23]
- Query Analyzer[24]
- Audit Log[25]
- Server Configuration[26]
- Replication[27]

Overview

The Overview gives the top level picture of all the servers registered with SQL DM for MySQL. It gives the count of the total servers registered with SQL DM for MySQL, total number of disconnected servers, servers having critical alerts, and the servers having warnings. You can further drill down to Servers page with the corresponding categories from the Overview page.

18 http://wiki.idera.com/x/fwEGBg
19 http://wiki.idera.com/x/gAEGBg
20 http://wiki.idera.com/x/gwEGBg
21 http://wiki.idera.com/x/iQEGBg
22 http://wiki.idera.com/x/mwEGBg
23 http://wiki.idera.com/x/oQEGBg
24 http://wiki.idera.com/x/pAEGBg
25 http://wiki.idera.com/x/pgEGBg
26 http://wiki.idera.com/x/qAEGBg
27 http://wiki.idera.com/x/qwEGBg



Overview also gives you the Top 10 Queries across all the servers registered with SQL DM for MySQL based on total execution time and also lists the count and Average latency for each of the queries. The queries are basically filtered from the *"sniffer*.data" file of all the registered servers. You can click the queries to get the query details which also gives you the list of server names on which the particular query was executed. Clicking the server names to open Sniffer for the selected server, with the time-range selected as the first seen and the last seen of the query.

Servers

Servers provides an unified view of all the servers registered with SQL DM for MySQL. Users can view which servers are running at the optimum level and identify servers which require critical attention. The servers page allows you to add or delete a server as required.

With the All-New SQL DM for MySQL, you are to able to easily filter MySQL servers and find the details of your preferred server.

Server Block

The Server Block lets you identify the state of the MySQL server:

Status	Description	Server Block
Alerting	MySQL servers that has critical alerts and warnings.	
Disconnected	SQL DM for MySQL is unable to connect with the MySQL server.	

Status	Description	Server Block
Stable	MySQL servers that do not have critical alerts and warnings.	
Stopped	SQL DM for MySQL is not collecting data.	

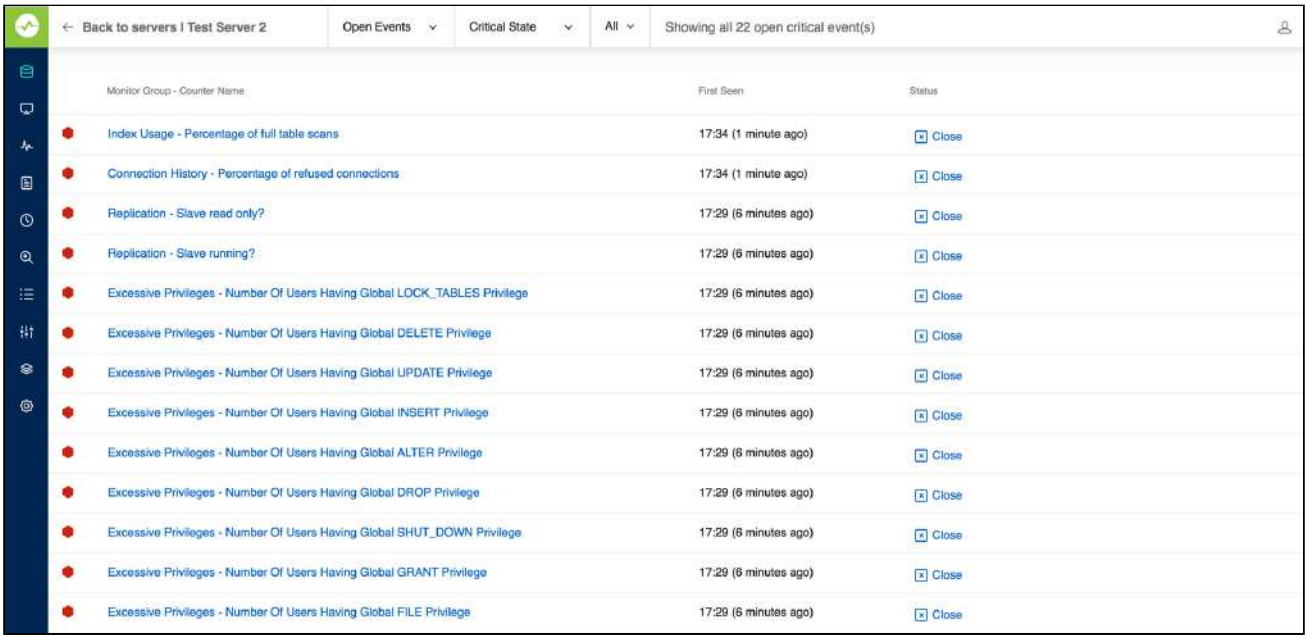
Server options

The server tab provides different options to the user to manage their registered MySQL servers. Users can edit, duplicate or delete MySQL server(s), command SQL DM for MySQL to stop the data collection for a particular server, rebuild database (please refer to SQL DM for MySQL's Data Maintenance[28]), check disk info, analyze all the events, and run diagnosis reports.

Events

An EVENT happens when any counter is changing its status to (yellow) WARNING alert level or to (red) CRITICAL alert level. And this EVENT is over when the counter becomes stable. The EVENT table gives an overall view of all such EVENTS that have happened on each server.

28 http://wiki.idera.com/x/sgEGBg



An alert (WARNING or CRITICAL) can be closed (and re-opened) for a specific server from both the EVENTS overview page as well as the Monitors page.

Closing an event is temporarily disabling an EVENT meaning, disabling alerting with the yellow/red on the monitors page or sending notifications until it becomes stable, and then goes to the CRITICAL/WARNING (yellow/red) state.

EVENTS can be closed from both the Monitors page (by clicking on the red/yellow alert) and the EVENTS page. Opening the closed EVENTS can be done from the EVENTS page. Closing ALL EVENTS for a certain server from the Monitors page is possible.

> ⚠ Opening/Closing EVENTS requires user-level privileges.

Disk Info

The Disk Info is an easy to use disk space usage analyzer that allows you to see which MySQL server takes up the most space. It displays number of databases with their data and index sizes. It gives you the snapshot of the disk space occupied by MySQL database objects on your servers as well as a graphical chart display to quickly spot the largest databases.

The Disk information is available at two information levels:

- Database level(see page 84)
- Table level(see page 84)

In Database and Table information levels the data are illustrated by a graphical chart (a 'Doughnut' chart type) for an easy overview. If there are few objects only (databases on server or tables in database) they display in the graph.

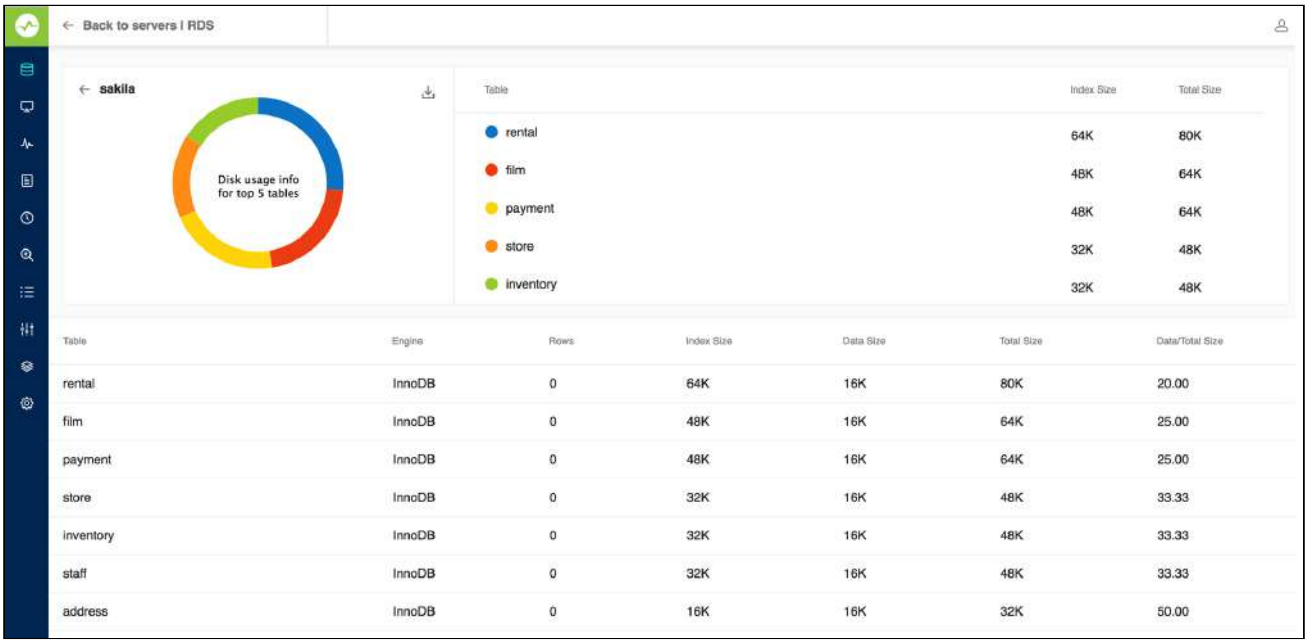
Overview

Database Level

You reach at this level by clicking the Disk Info option provided for each server in the Servers page. In this view, you get an overview of the space occupied by individual MySQL databases on the server you selected.

Table Level

You reach at this level by clicking one of the databases in the Database Level view. In this screen, you get an overview of the space occupied by the individual tables in the MySQL database you selected.



Also, you get a detailed view of the disk space occupied by data and indexes, along with values indicating how efficiently disk space for every object is utilized. Note that for specific table ENGINES such information is not always available and N/A displays.

For table ENGINES having a shared tablespace for all databases like INNODB (if not the file_per_table option is set in configuration) the Connection Level returns the size of the tablespace itself not the cumulative sum of the databases as shown in the Database Level. This may be rather misleading as the tablespace does not shrink when objects are dropped. Same may apply to third-party engines (like SolidDB) also having a shared tablespace for all databases. Also, with third-party ENGINES and ENGINES in beta, other issues may occur that can affect the accuracy and usability of the information. Finally, navigate back to the Database level from the Table level by clicking the database name adjacent to the chart. You can navigate back to the servers page by clicking the **Back to servers** link.

Disk Info charts displayed can now be exported as JPEG/PNG/PDF formats.

Dashboard

Real time Charts for Monitoring All MySQL Servers

One of the biggest challenges the MySQL DBA faces is managing an ever-growing number of MySQL servers and databases. Regardless of the size of the MySQL environment, each server requires specific attention when it comes to basic administration, security, performance monitoring, and availability. To give the MySQL DBA a proactive advantage in all of these areas, SQL DM for MySQL provides the Dashboard.

The Dashboard gives the flexibilty to display only a particular set of charts and create as many dashboards as you want. Using the Dashboards, DBAs can create their own set of charts and monitor MySQL and OS specific metrics for single or groups of servers. The Charts are designed so DBAs can easily understand the complete security, availability, and performance picture of all their MySQL servers in one place, all from a slick AJAX interface.

Default Dashboards

The Dashboard is a page displaying a graphical view of the server parameters and metrics that in most situations will give you a quick overview of the server load and performance. SQL DM for MySQL ships with a default dashboard called "Performance metrics" which includes the metrics:

MySQL metrics:

- Connections
- Cache misses
- Statements
- Database Throughputs

System metrics:

- Disk IO
- CPU usage

How is the refresh interval defined?

Every graph displays related metrics using different colors. In the Charts interface the X-axis timestamps are not printed, and the time interval between each reflects the server-specific setting for the sample interval. By default, the chart displays the last one-hour timeframe. If no data is available for a period because MONyog(SQL DM for MySQL) service was stopped, two empty sample points display in the grid no matter how long time has passed.

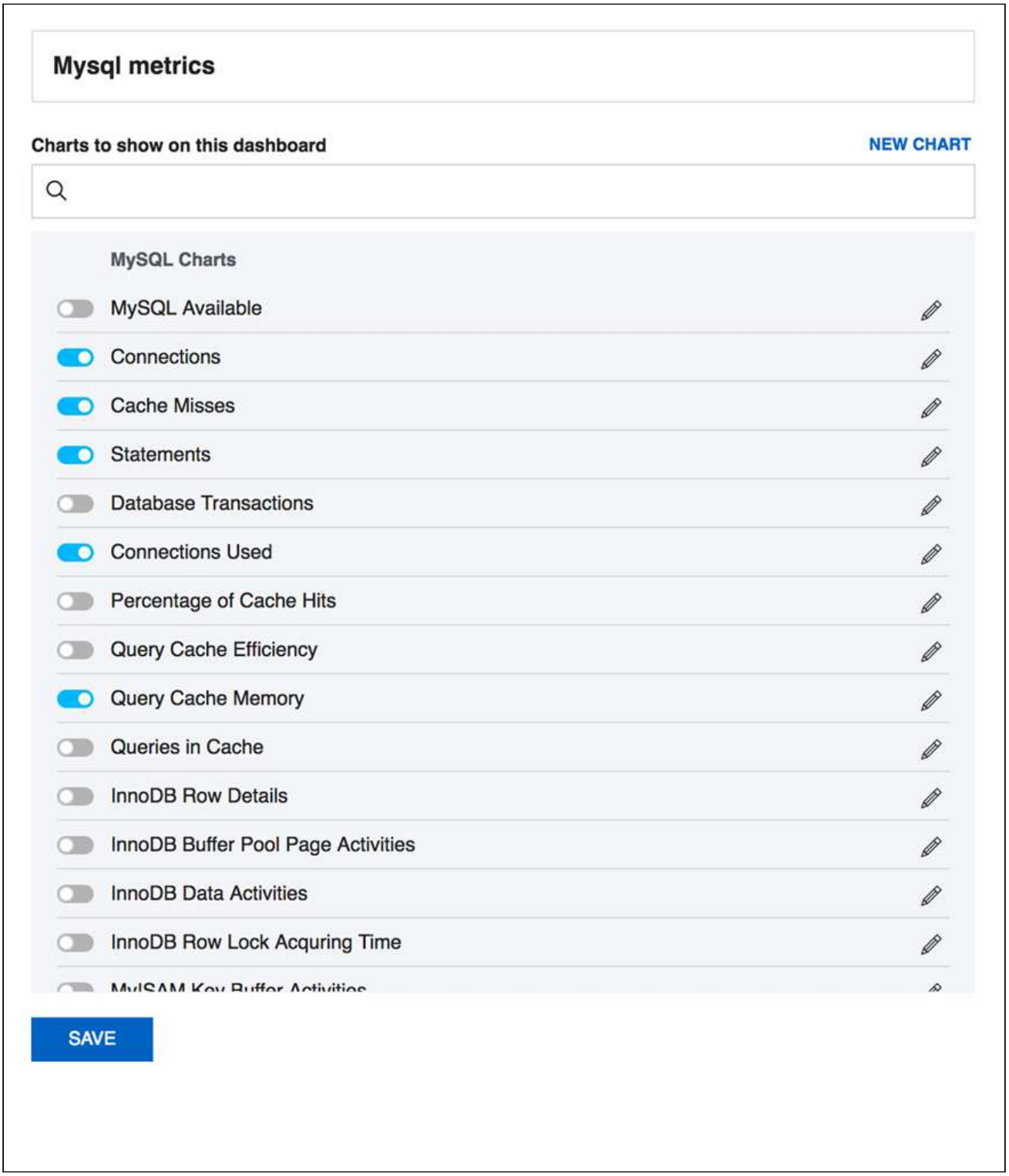
Adding Dashboards

Just click the drop down menu containing the Dashboard names, and select the **Add New Dashboard** option. This opens the page where you can give a suitable dashboard name and decide which charts to enable for your dashboard.

Dashboards settings

Dahboard settings can be changed from the dashboard page itself. These are local (browser specific) settings that are stored in a cookie.

- The size of the Chart is configurable, you can stretch/reduce the charts size.
- You can hover on the anchor points on the charts to see the actual values.
- You can customize the look of charts by changing the chart color under **Settings -> General -> CHART COLOR**
- You can rename the dashboard name, add/remove charts to the dashboard, and delete the dashboard from the dashboard name drop-down menu.



Charts recommendation

Recommended number of collections is 50 for better visibility.

> ⚠ This recommendation is based on 1280 x 1024 screen resolution. So, depending upon your screen resolution the number of collections may vary to some extent.

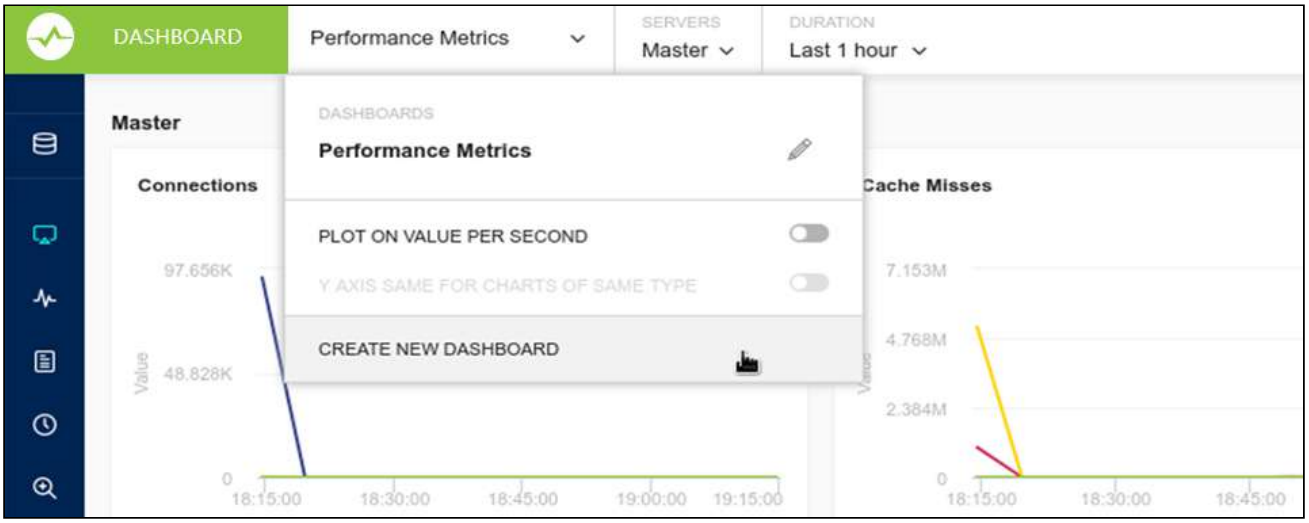
Also, charts can be exported as PDF/JPG/PNG formats, for more information review Exporting graphs.[29]

Customizing Dashboard

SQL DM for MySQL provides a series of charts that address general monitoring needs for MySQL systems. It also ships with a general purpose dashboard for monitoring database performance. Also, you can create your own custom dashboards and charts.

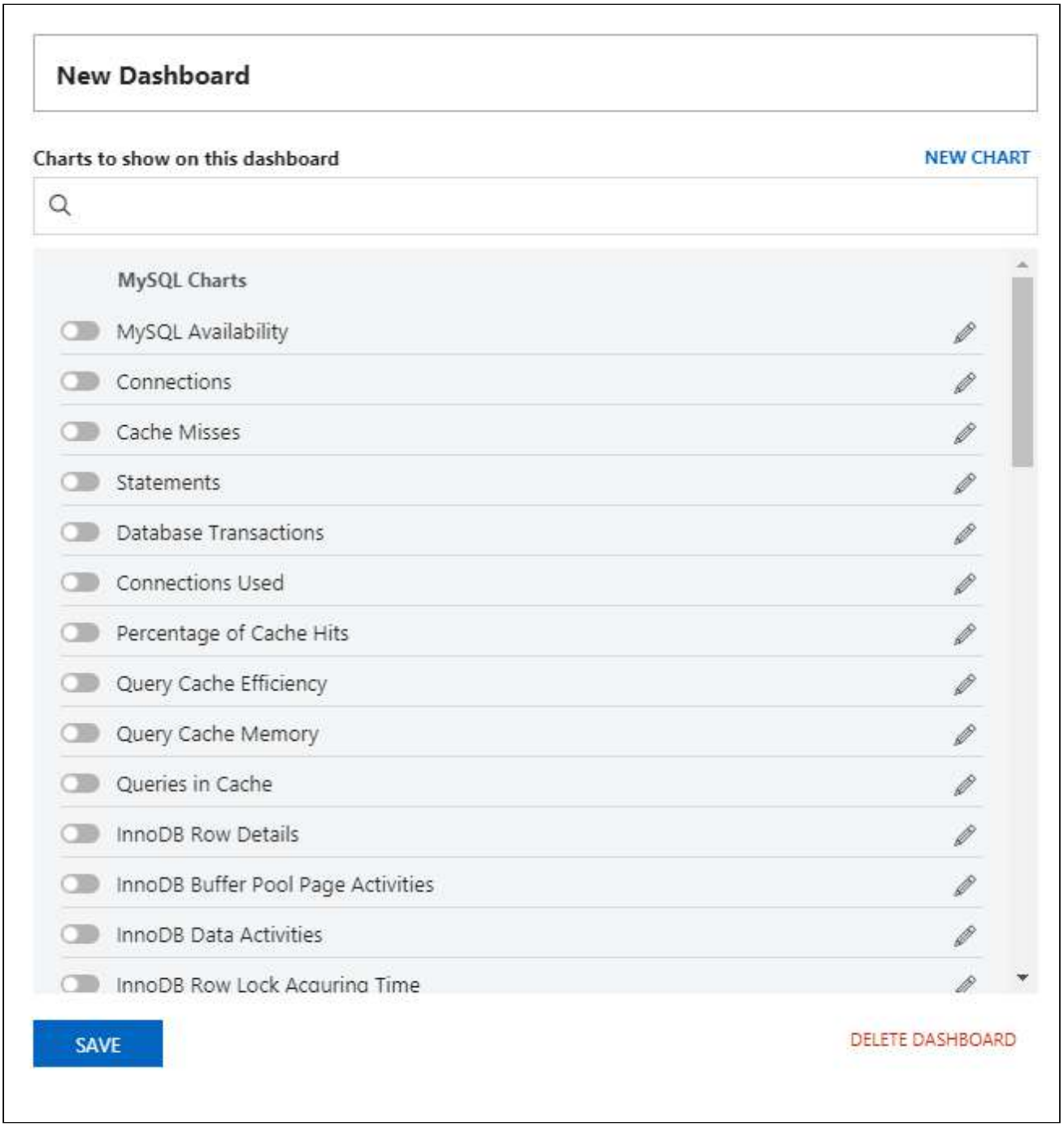
Custom Dashboards

To create a custom dashboard, open the dashboard selection context menu. Then click **Create New Dashboard**. It opens an overlay to configure the dashboard, select the charts you want to display. In the event that you would like to modify an existing dashboard, click the edit icon beside it in the same menu.



From the overlay, enter a logical name for the dashboard in the text field, then select the charts that you want to be added to the dashboard. In the event that you need a chart for something that is not available, you can create a new chart. Click **Save** to make the new dashboard available in SQL DM for MySQL.

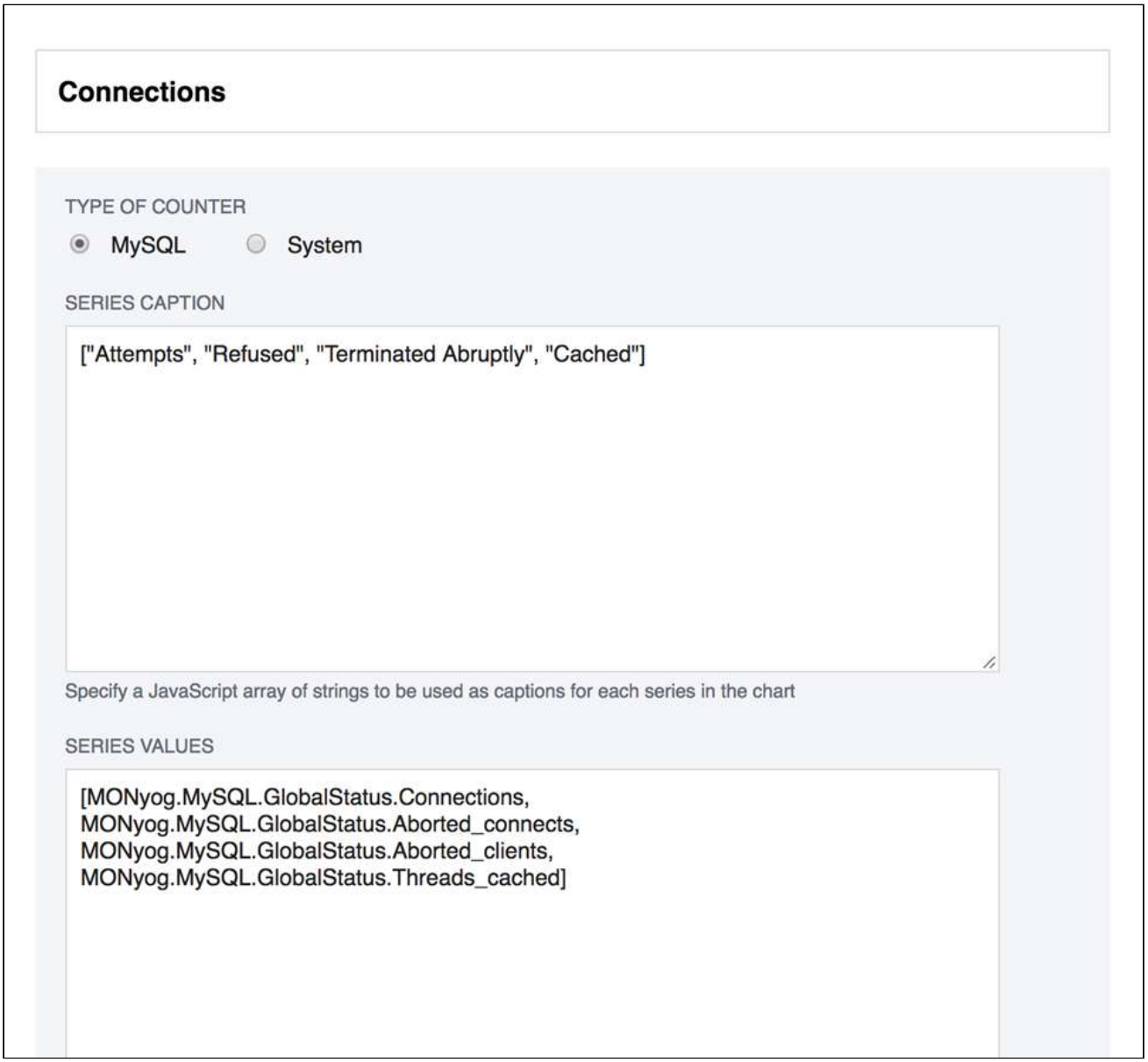
29 http://wiki.idera.com/x/iAEGBg



Custom Charts

In addition to the many charts that SQL DM for MySQL provides by default, you can create custom charts to suit your own particular needs. To create a chart, create a new dashboard or edit an existing one, click the **New Chart** link in the overlay.

When clicking **New Chart**, the second overlay opens to configure the new chart.



Charts are of the MySQL or System type. MySQL charts get their data from the MySQL client interface. System charts get their data from the operating system, (through SSH, for instance). For each chart you can define a Series Caption, indicating the usual one to four values shown on the chart and series values to specify how to generate those values. Both are written as JavaScript expressions.

Once your new chart is ready, click **Save**. SQL DM for MySQL adds the chart to those available on the configuration overlay, where you can enable it for any dashboard.

Query Details

You can zoom into spikes in the charts and get to see the queries involved in the charts, global status, and global system variables for that time range. Just hover over the chart, and click the **Show details** icon to open the page with the above mentioned information.

If sniffer was running during this interval, aggregated sniffer information is displayed for the time interval. Also, you can see first and last value of (optionally) all or changed variables aggregated values. The graph is zoomable by selecting a sub-interval with the mouse.

When user clicks a row in the query list, a pop-up opens with information about thread-id, user, and host along with full query.

The list of queries will not be rendered if there are more than 2000 queries in the time period. There is a user control that can be activated in such case. The reason is that every 1000 queries take around 1 seconds to render on an average desktop system. And, it displays the list of queries before zooming (provided still that there not more than 2000 queries to display).

Adding New Charts

Click **New Chart** option on the Create New Dashboard page.

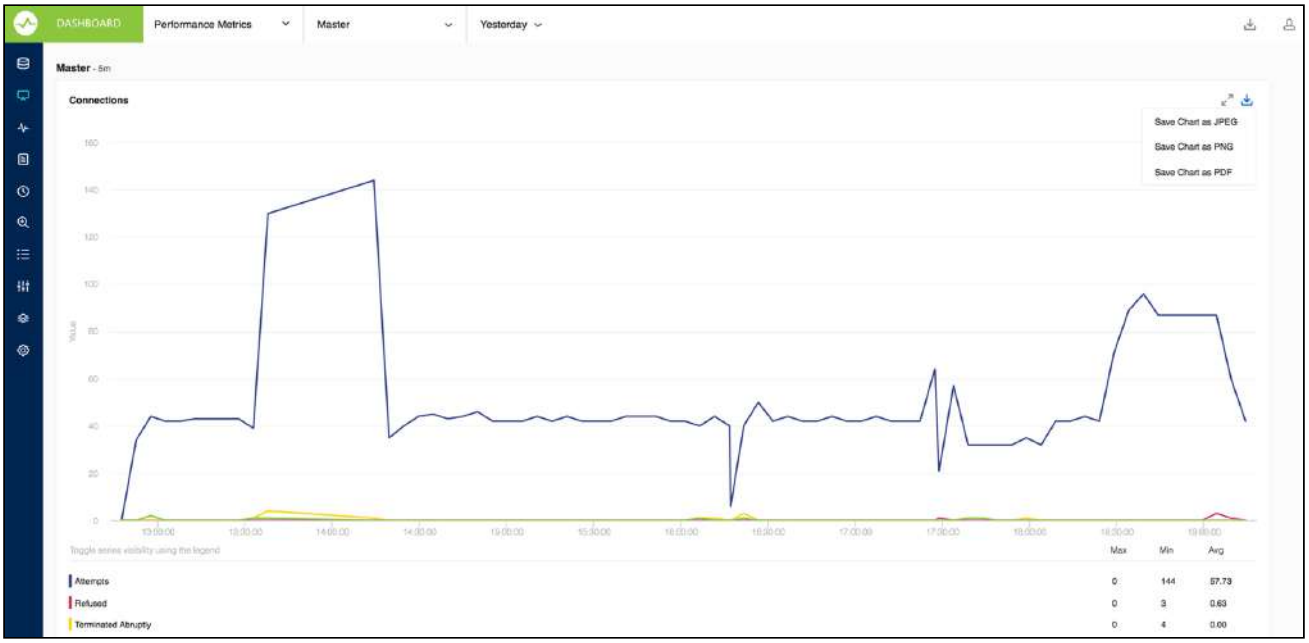
Select the desired type of counter - MySQL or System.

- **Caption:** Enter an appropriate name for your chart.
- **SeriesCaption:** Specify the names of all the graphs that are going to be plotted. Example: ["Attempts", "Refused", "Terminated abruptly"] Here 'Attempts', 'Refused' and 'Terminated abruptly' are the names of the graphs that are going to be plotted on this new chart being created.
- **SeriesValues:** Specify a JavaScript array of strings to be used as captions for each series in the chart. Please refer Monyog object model for more details.
- **This chart shows boolean values:** Enable this option if the chart displays only on and off status.

- **Unit:** Specify a JavaScript array of strings to be used as y-axis unit in the chart. Ex: ['','KB','MB','GB','TB']
- **Unit Factor:** Define the limit on reaching which the unit should be incremented. Ex: if this value is 1024 then, 1024 = 1KB, 1024KB=1MB and so on.
- **ChartValue:** Controls how values in the series are plotted. Specifying "Delta" causes the difference between values from the last two data collections to be plotted. Specifying "Current" causes the current value to be plotted as it is.
- **Uptime:** Selecting "Yes" indicates that this Charts chart will plot cumulative values which increase with time.

Exporting Graphs

All charts displayed by SQL DM for MySQL in Dashboard can be exported as PDF/JPG/PNG formats. To export a chart select the option from the drop-down context menu.



Monitors

Server monitoring is one of the base components of server management. The critical nature of servers require someone or something to constantly monitor the status of the managed servers. SQL DM for MySQL easily automates this tedious job and detects server problems as soon as they occur.

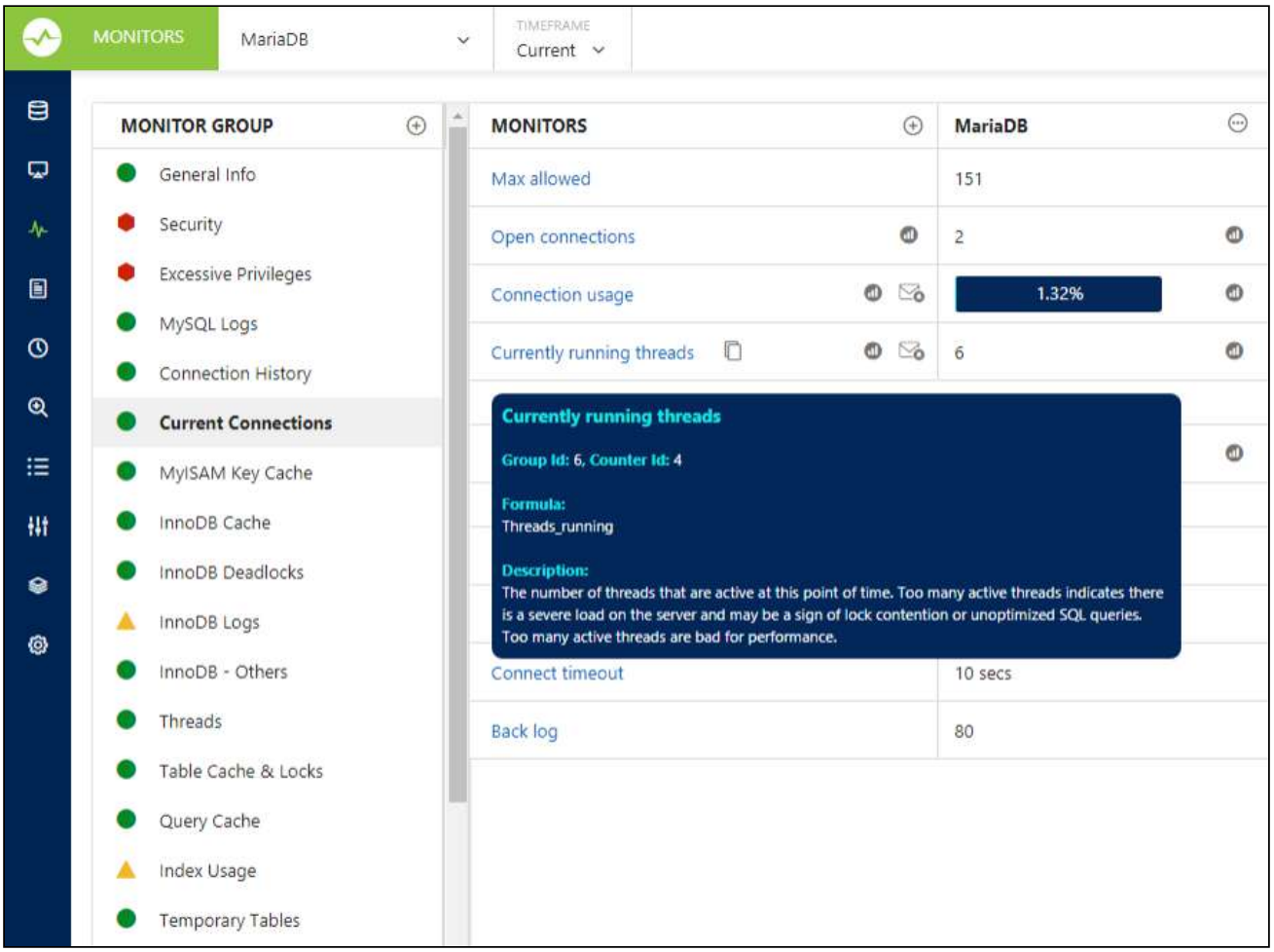
'Monitors' interface lets you access any server metric you can imagine.

The Monitors page shows a detailed display of server parameters and metrics. The left column displays the metrics that SQL DM for MySQL displays per server. For every server that you registered with SQL DM for MySQL, a column, with data pertinent to that server, is displayed. Some data is displayed as cumulative values (accumulated over a time frame that you can specify) while others are averaged over a time interval (typically per second).

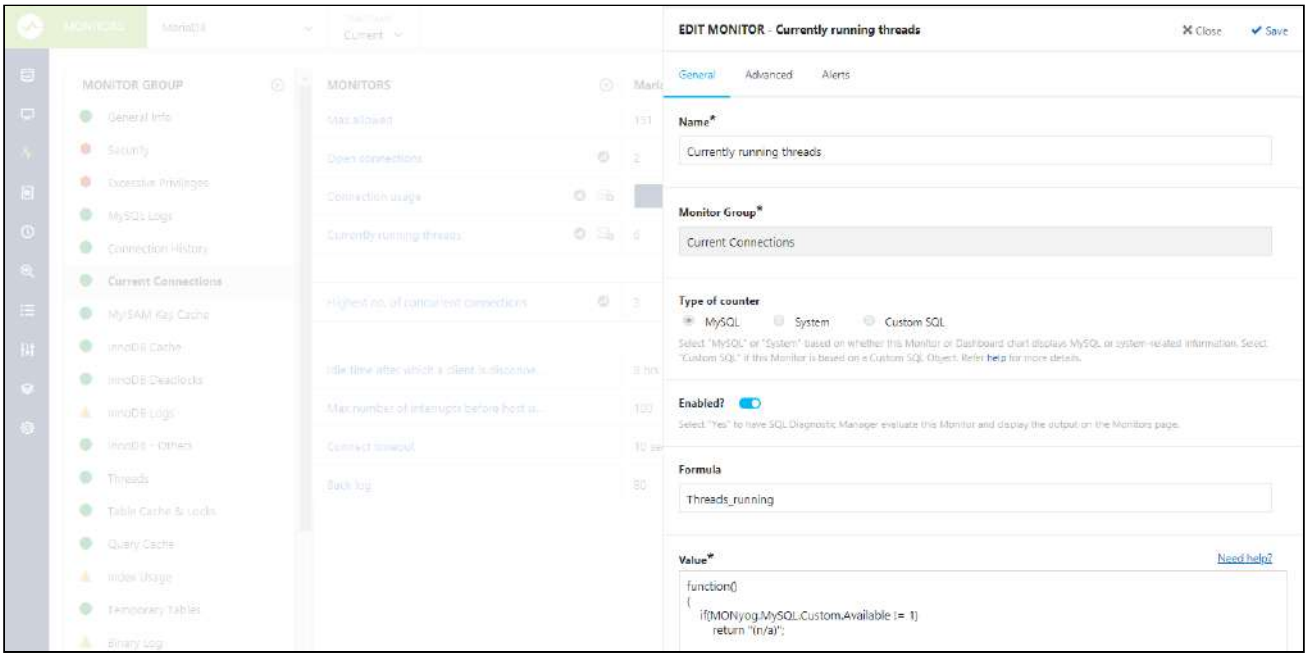
Layout_View of Monitors

To see how every metric is calculated by SQL DM for MySQL you need to position the mouse over the name of each metric. The name of the monitor, formula used by the calculation displays with an explanation as a 'tool tip popup'. The tool-tip also explains how the result shown is evaluated, and how the value relates to an overall or some specific server performance metric.

Every metric is documented in the SQL DM for MySQL web interface itself.



If you require more details on the metric, click the hyperlink to get a detailed view.



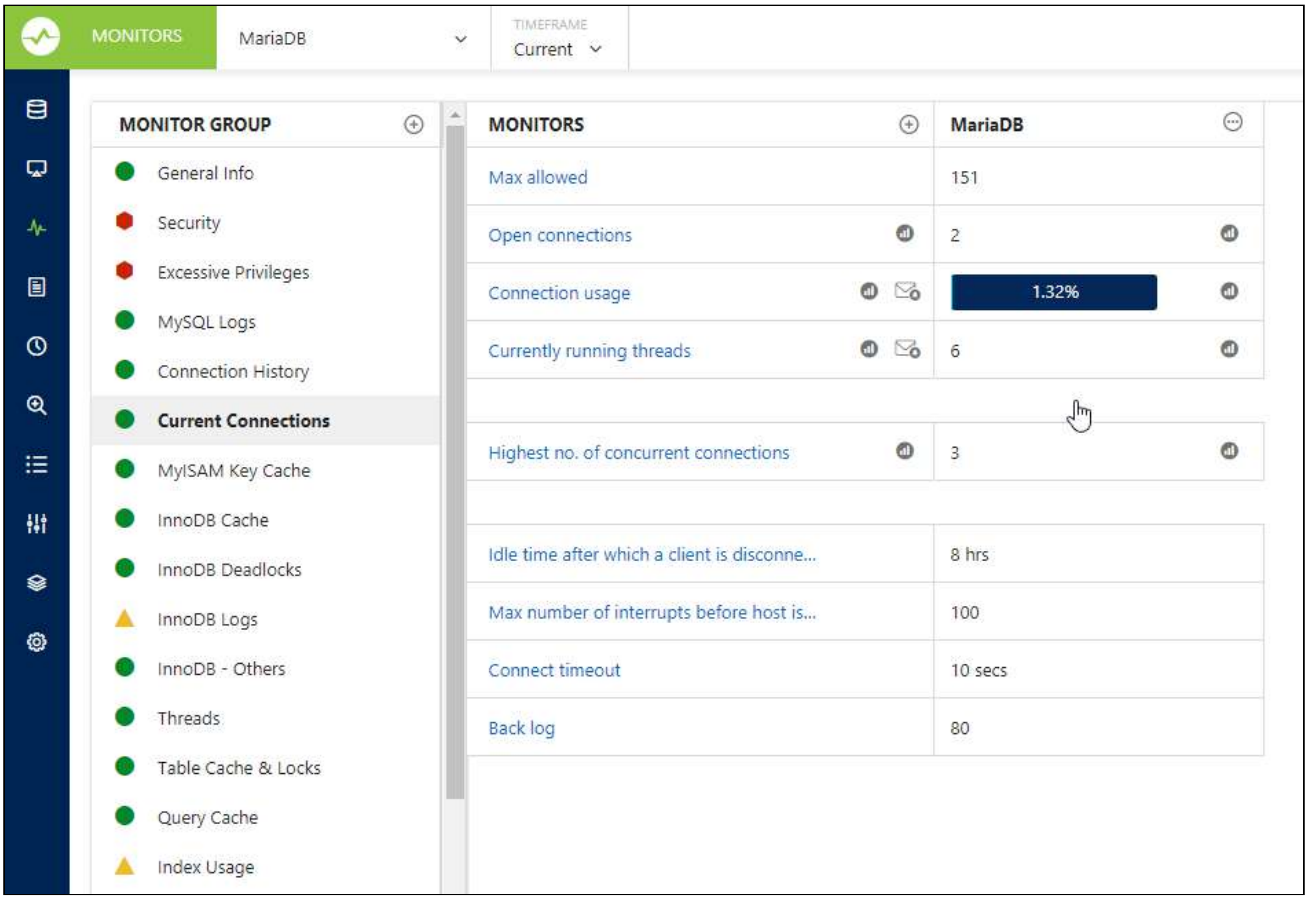
The monitors can be edited or duplicated by clicking the icons adjacent to the name of the monitor. The mail alert icon is displayed adjacent to the name of the monitor specifying an alert condition. The blue coloured mail alert icon indicates that the notification has been enabled with the alert condition specified.

Mail and Graph icon

Mail icon

The Mail icon  indicates that the corresponding counter will send you an alert when:

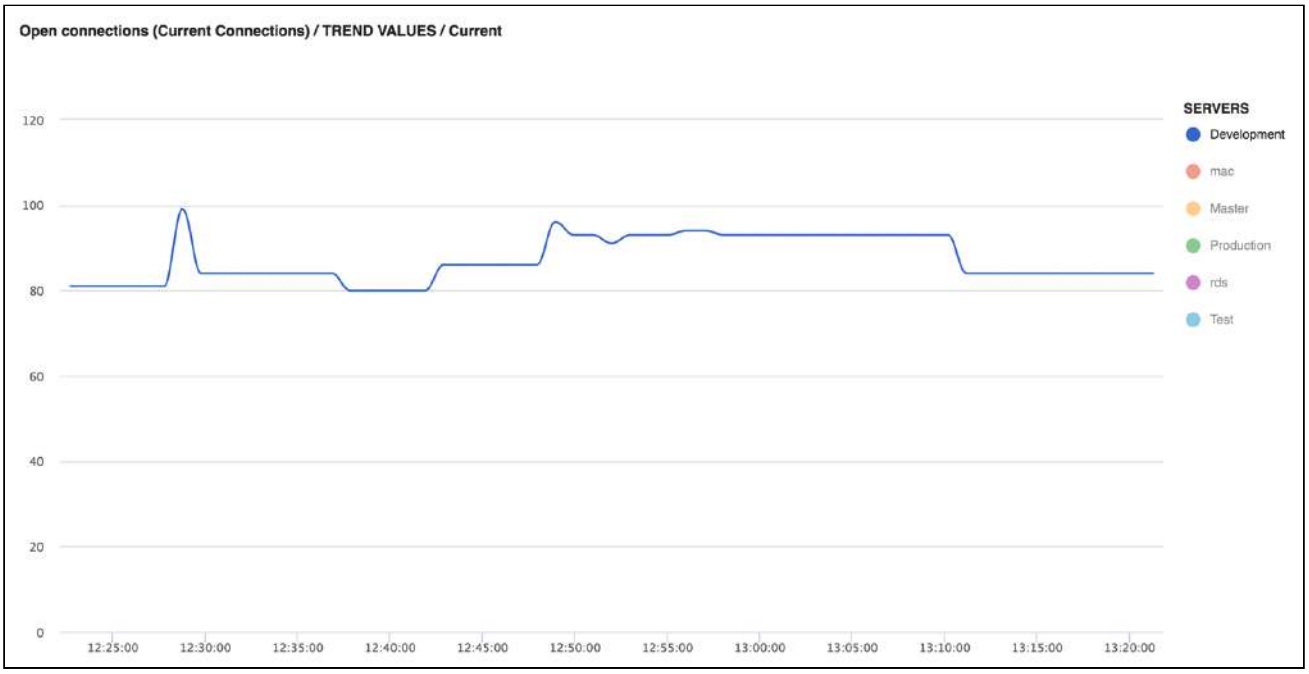
- You selected the option to send alerts for the specific server.
- The value for either the 'current' or the latest time frame exceeds threshold level.



Graph icon

A graph or chart is a type of information graphic or graphic organizer that represents tabular numeric data and/or functions. Graphs are often used to make it easier to understand large quantities of data and the relationship between different parts of the data. Graphs can usually be read more quickly than the raw data that they come from.

When clicking the graph icon a graph displays, depicting the current status of some metric of the server, which is updated in real-time.



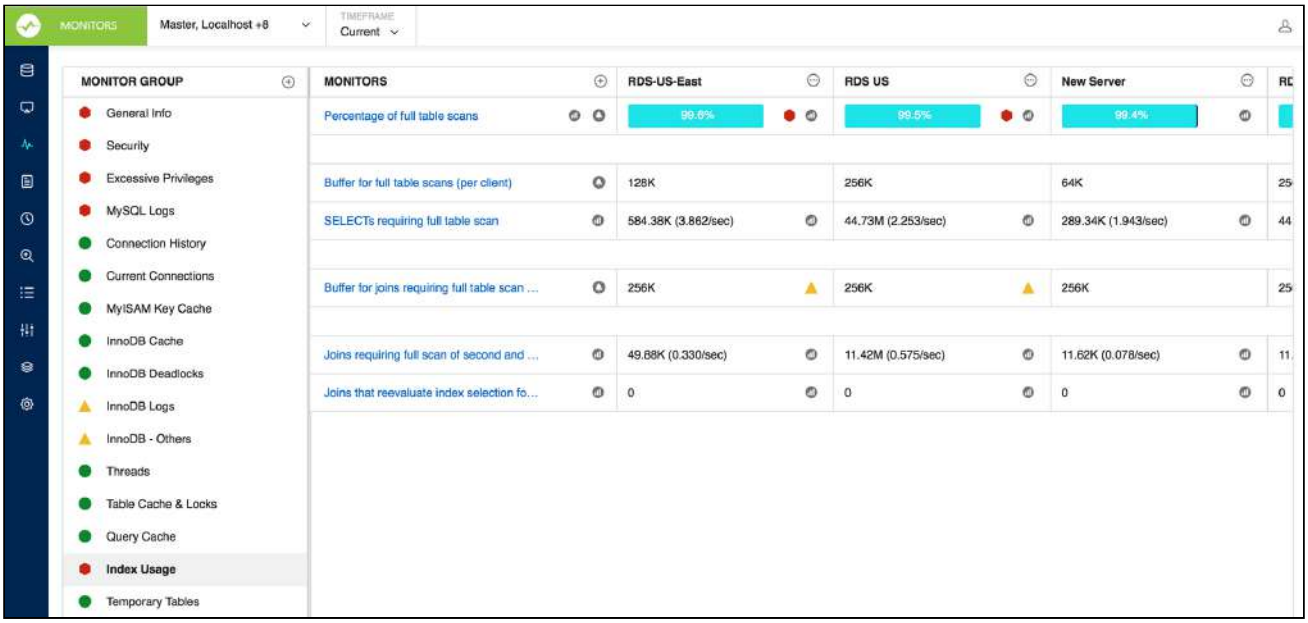
Charts and graphs displayed by the Monitors can now be exported in PDF/JPG/PNG formats. To export a chart select the option from the drop-down context menu.

Critical and Warning Alerts

In case a red hexagon or yellow triangle appears beside a metric, it means that the current value of the metric exceeds the set threshold value, ie., the symbols signify alert conditions. The definition of what should trigger a red hexagon or yellow triangle alert is defined in the JavaScript component defining every metric. The default values that SQL DM for MySQL ships with have been configured for average servers used for mixed purposes: a mix of websites and databases with corporate data, for example.

You may want to change the values depending on the type of databases. Threshold values for the metrics (and also actual, realizable, values) varies depending, at the very least, on the type of applications that use the database. For instance with certain web applications like Forums a very high 'cache hit rate' typically can be reached (as a large percentage of queries return the most popular post in the Forums) whereas in a production planning system it is typically not possible to achieve a very high hit rate (as identical queries and results only rarely occur here).

Red and Yellow alerts will quickly tell what could be issues to investigate.



Difference between All time_Current, Delta

Define yourself the timeframe of data in the database that shall be used for generating the display.

The Monitors page can be defined to operate in 2 'dynamic modes' and 1 'cumulative mode'. The dynamic modes auto-update with the most recent data without user interaction. The 'cumulative mode' is History/Trend Analysis where you can define any time interval for the actual display of the data.

- **Current:** Display values based on Current data values collected by SQL DM for MySQL for every specific server. Some server metrics will only be meaningful after the server has been running for a while (ie, after the server has been 'warmed-up').
- **Delta:** Display results based on data for the period between the last data collection and the collection before it. The result gives you a better idea of the current situation, and how much this current situation differs from the average or normal situation.

Difference between Cumulative and Point in time counters

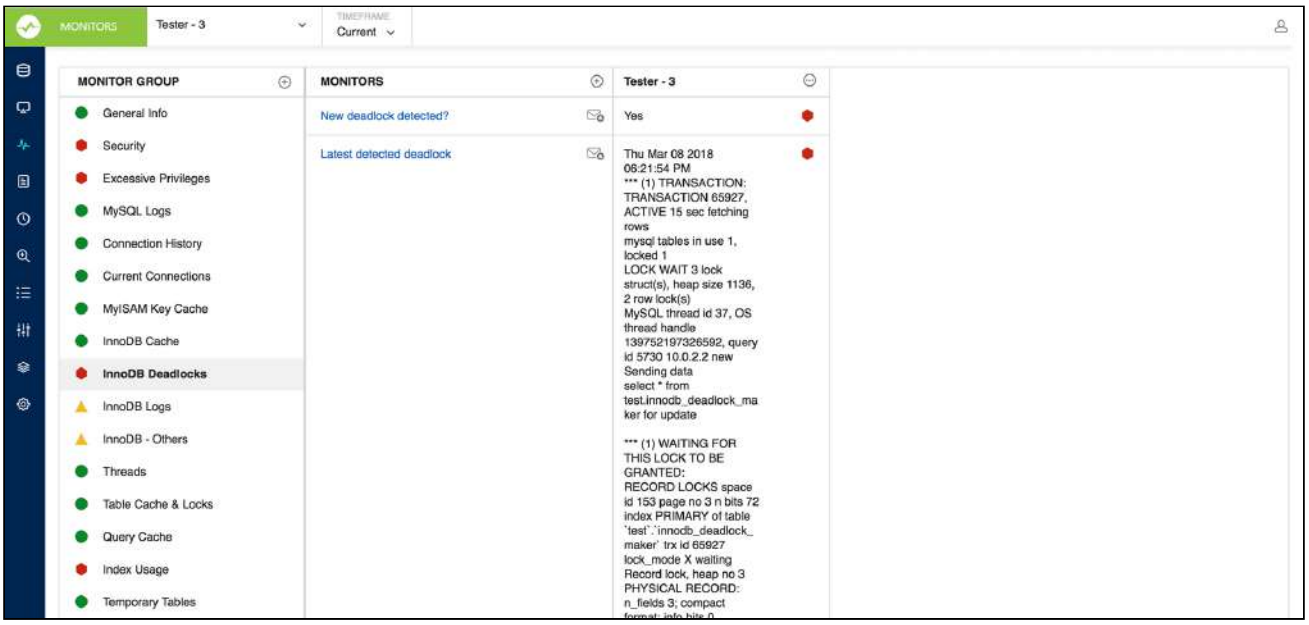
Cumulative Metrics (increase with time and can be recorded for a period (like 'Key_reads', 'Maximum no. of Connections that Ever Was')) are calculated differently instead of point-in-time metrics (like 'Server Availability', 'Currently Running Threads').

- **Cumulative metrics:** SQL DM for MySQL send alert if thresholds are crossed for last collection interval.
- **Point-in-time metrics:** SQL DM for MySQL send alert only based on 'Current' values (for instance current no. of connections is close to Max Allowed Connections - there is nothing more to consider!).

Innodb Deadlock Monitoring

The InnoDB deadlock monitor is a little different than the rest of the monitors and require a little additional explanation. Whereas all other monitors are based on the return of `SHOW GLOBAL STATUS`, `SHOW GLOBAL VARIABLES` and `SHOW SLAVE STATUS`, the 'InnoDB deadlock' monitor is based on output from `SHOW ENGINE INNODB STATUS`. This statement differs from the first-mentioned in the sense that it does not return as single discrete value (like a single number), but a rather long string instead with lots of information. From this long string we extract what is related to deadlocks.

With this monitor, it does not make sense to distinguish between 'Current' and 'Delta' timeframes. These timeframes display the same in SQL DM for MySQL:



- **New Deadlock Detected** indicates if a new deadlock was reported by INNODB STATUS between the last two data retrievals.
- **Latest Deadlock Detected** shows information about the last deadlock detected (if any). To view all deadlock situation that SQL DM for MySQL has information about, use the 'History' timeframe.

There is a limitation with deadlock information returned by `SHOW ENGINE INNODB STATUS` that will also affect SQL DM for MySQL: It returns only information about the last deadlock. So if more than one deadlock has occurred between two data retrievals, SQL DM for MySQL only has information about the latest.

Getting automatic alert notifications

Get alerts for what you want and when you want

SQL DM for MySQL can send you alerts over mail, SNMP, Slack,PagerDuty, and Syslog. Stay on top of significant events and avoid many sleepless nights!

Some features of the alerting system are:

- It uses the concept of "Delayed alert notifications". It can now be defined that a problem must have existed for a number of sample intervals continuously (in a row) for an alert to be sent. A global setting for each server is available from GUI. For individual counters the global setting can be overridden by defining the RetryOverride (like 'RetryOverride:3') property of the (JavaScript) counter definition.
- Also, you can choose to be notified when SQL DM for MySQL detects that a problem, which existed previously, has been resolved.

⚠ Alerts for every registered server are sent independently.

Configuring alerts.

Please refer to Notification & Maintenance Settings[30] for more information regarding Notification Settings.

What alerts contain and when is it sent?

In the Alert Condition field there is a condition defined to decide when the monitor will be in critical or warning state. SQL DM for MySQL will evaluate AlertCondition and if AlertCondition is "Critical" or "Warning" then only it will trigger an Alert. Further, if WarmUpRequired is defined and it is "Yes", then SQL DM for MySQL evaluates the AlertCondition if the server is up for at least three hours (the default warmup time). "Critical" and "Warning" correspond to red or yellow dots in the GUI interface respectively.

The mails are optimized for display in a HTML enabled mail client but also readable in a client supporting text mails only.

The alert contains the following information:

- The MySQL Host.
- There is a detailed rule defining if and when a alert is being sent like All Time/Current, Delta - Last two collections.
- Name of the alerting Monitor.
- The corresponding Monitor Group.
- Type of Alert: Critical, Warning, etc.
- The defined Threshold Limit.
- Value of the Monitor when the alert was generated.
- The Advice text.

This is an example of a typical SQL DM for MySQL mail alert.

Server: LMT	
Sampling timeframe: **All Time/Current**	
Name	Available?
Group	General Info
Type	Critical
Value	No

30 http://wiki.idera.com/x/uAEGBg

Server: LMT	
Advice	If MONyog service is not able to connect to MySQL, it simply means that connection is not possible for one of the following (or similar) reasons: • There is no MySQL server running at the specified host. • Connection to the MySQL server is not all.

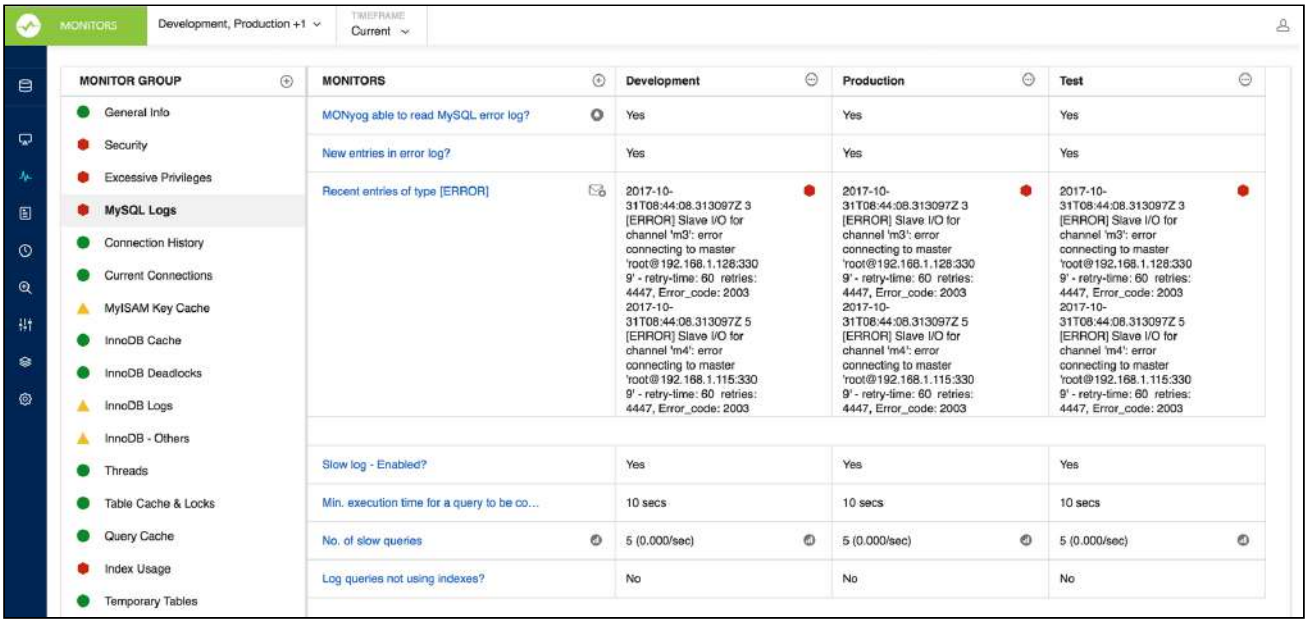
MySQL Error Log

MySQL error log contains information indicating when MySQL was started and stopped and also any critical errors that occur while the server is running. As such, monitoring the error log is crucial – changes to the log are indicative of disastrous outages.

SQL DM for MySQL makes the task of monitoring the error log very simple for you.

SQL DM for MySQL alerts you of changes in the error log, and if there is an entry of type `[ERROR]` in the log, SQL DM for MySQL extracts the corresponding message and send it to you.

> ⚠ If there are more entries of type `[ERROR]`, then SQL DM for MySQL will show only the last 1024 characters of type `[ERROR]`.



For more information, see Advanced Settings.[31]

31 http://wiki.idera.com/x/egEGBg

History_Trend and Graph Analysis

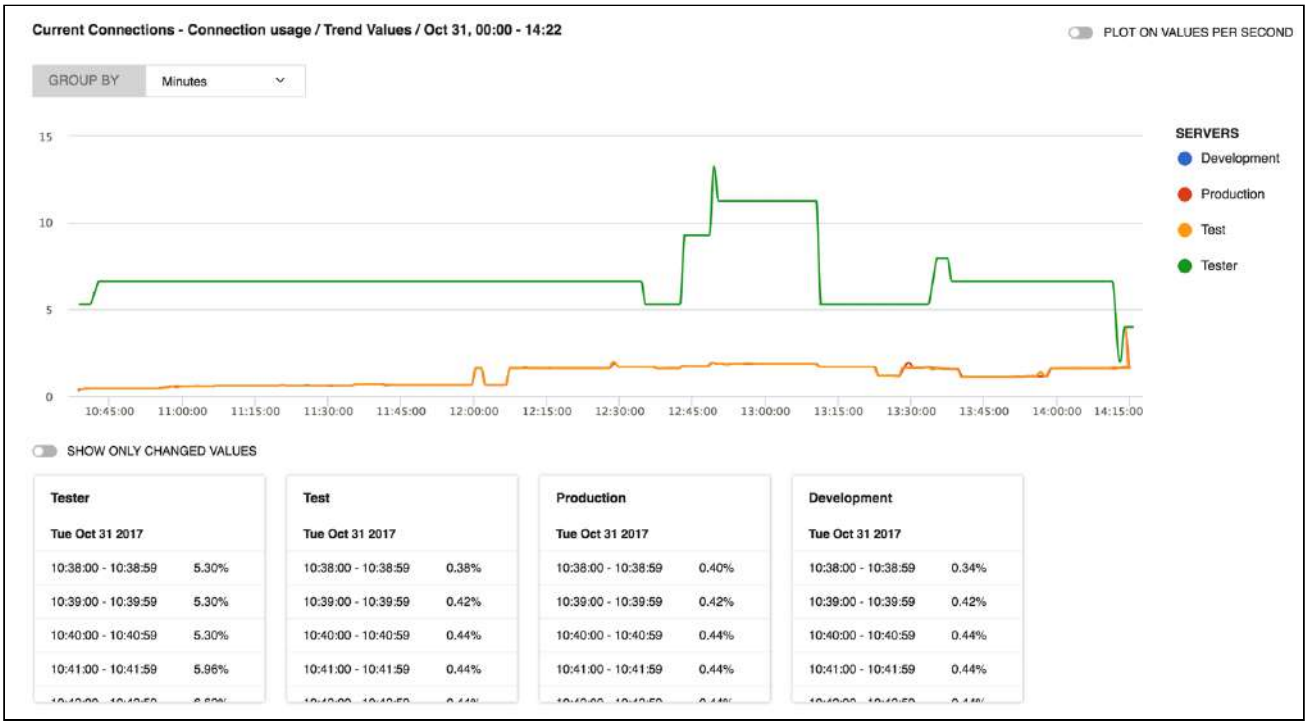
The term Trend Analysis refers to the concept of collecting information and attempting to spot a pattern, or trend, in the information. In some fields of study, the term Trend Analysis has more formally-defined meanings. Today, trend analysis often refers to the science of studying changes in social patterns, including fashion, technology, and the consumer behavior.

Analyzing history reports give you the means of tracking trends and identifying problem areas in your network. You can use this information to find recurring problems, which helps you prevent future problems. This data can also help you plan maintenance schedules and equipment replacement.

SQL DM for MySQL concerns itself with answering the following question:

- **Was your MySQL down in the last one month?**

- **When did your MySQL traffic peak in the last few days?**



And these are the few instances where we can find the answers for the following questions:

- When replication slave went down?
- Whether their was a hacking attempt?
- Which counters do you want to analyze?
- What time period do you want to analyze?

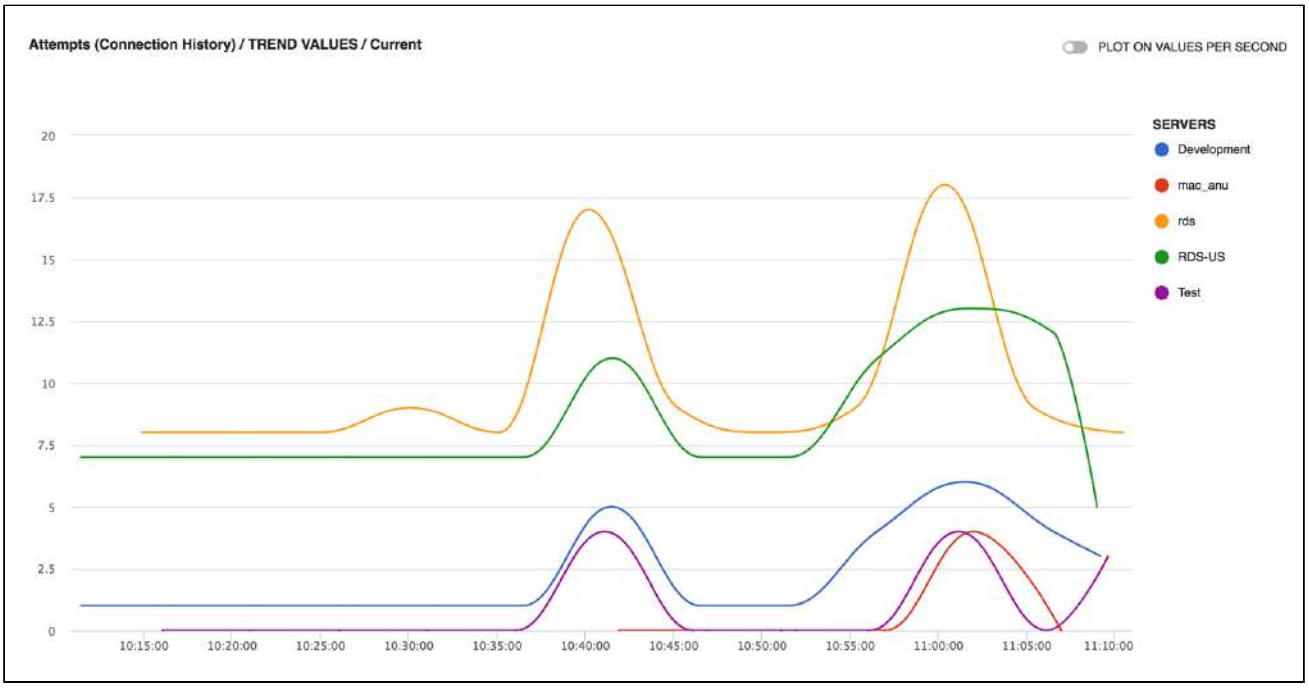
Answering the question thus becomes a matter of comparing different results and of analyzing their differences.

Graph Analysis

This feature is a further extension of the History/Trend analysis and lets you view the graphical representation for a monitor across different servers in one unified chart. You can either see the Current trend analysis (shows the graph for last 1 hour) or the Historical trend analysis for a selected time-frame.

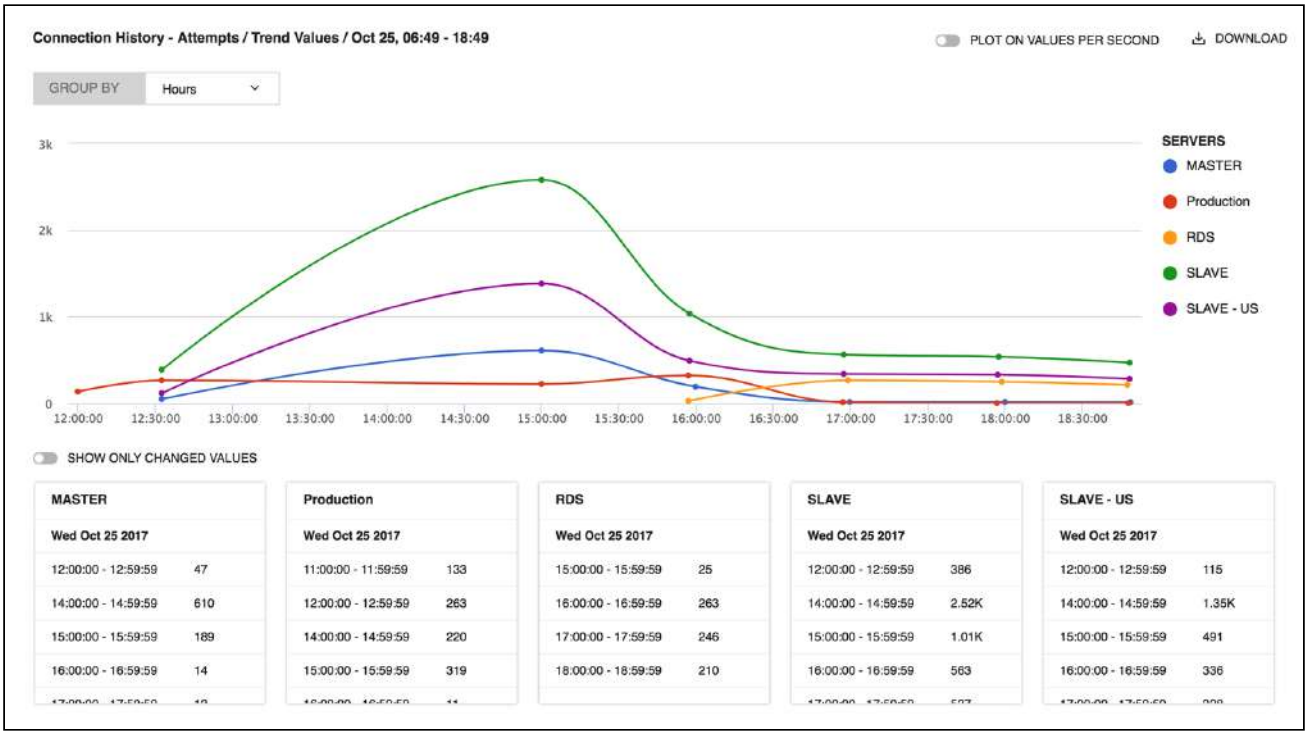
Current Trend Graph

With the Current time-frame selected, the Trend graph icon is present just beside the monitor name. When clicking, it displays the trend graph of last 1 hour duration for the selected servers in the Monitors page. The selected servers list is displayed on the right hand side, the servers can be selected or unselected by clicking the server name.



Historical Trend Graph

This is similar to the Current trend graph, the difference being that you can select any desired time-frame for which you want to see the graph for. You can choose the time-frame as History and then click the **Trend Graph** icon beside the counter name to see the graph. Along with the graph, it also shows the values of the monitor in the tabular form. Enable the option **Show Only Changed Values** to check when the monitor value was changed.



Custom SQL Objects (CSOs)

The CSO feature is to utilize information available in Information_Schema (as well as Performance_Schema of MySQL 5.5+) that are not exposed in the basic `SHOW` statements we have been using in the monitors/advisors. In addition to the `Performance_Schema` `SELECT` queries any query which returns a result set can be monitored.

CSOs not only lets you monitor server metrics but also lets you monitor server data.

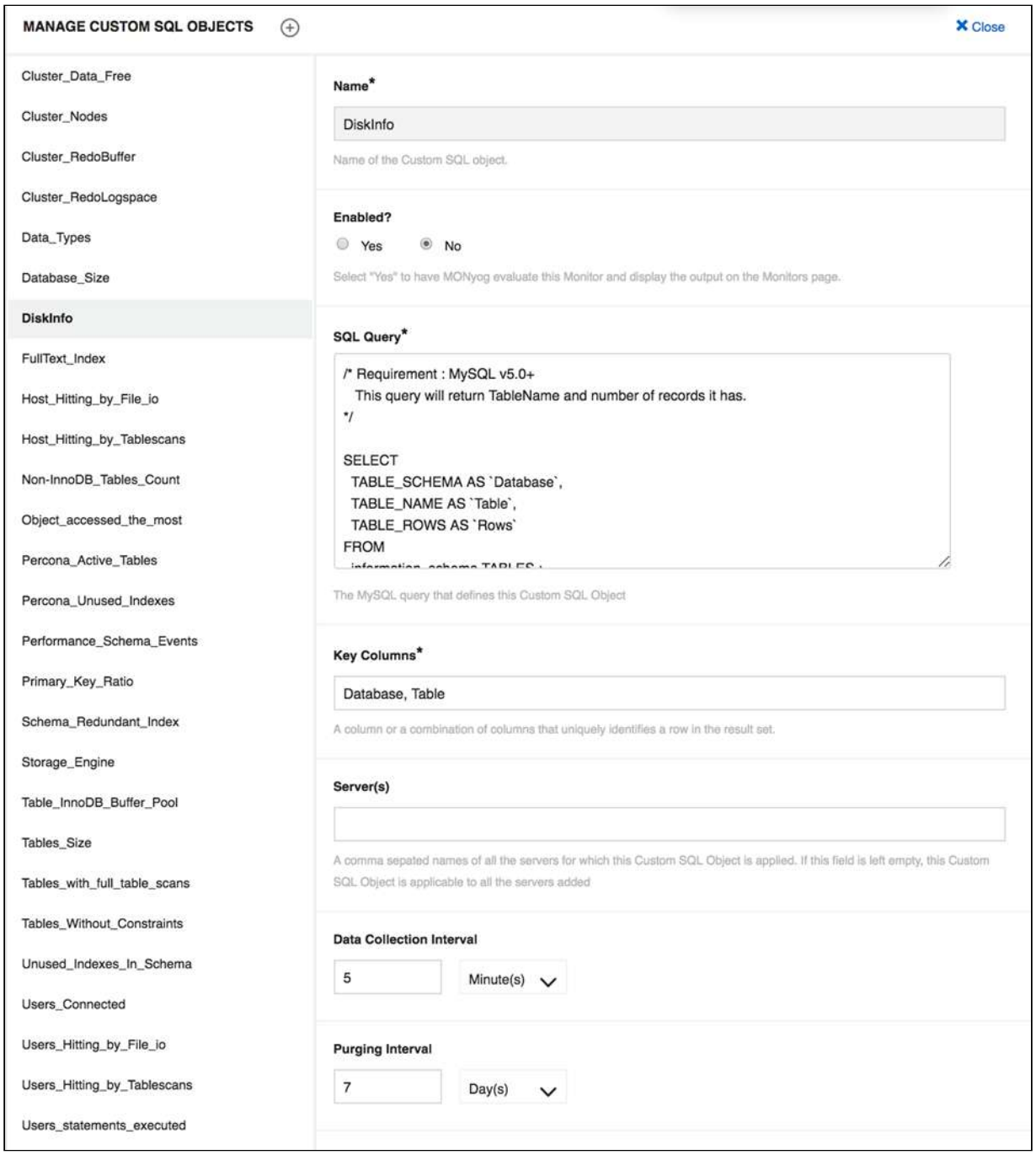
A CSC is based on any user-defined SQL query returning a result set. The array returned by MySQL from the SQL query populates a SQL DM for MySQL Object (a "Custom SQL Object" (CSO) in this case). This is exposed as a javascript array that may be referenced in SQL DM for MySQL counter definitions like any SQL DM for MySQL object.

Enabling pre-defined CSCs and CSOs:

In order to monitor CSOs you need to create a Custom SQL Counter(CSC). SQL DM for MySQL comes shipped with a bunch of pre-defined CSCs with their respective CSOs. By default, all pre-defined CSOs and CSCs are disabled. To enable some of these samples follow these steps:

1. Click the drop-down icon beside the title Monitors -> Manage CSO. The twenty-eight pre-defined CSOs display in the left menu.
   As an example select the **DiskInfo** item. The User Defined SQL-query displays in the SQL box. Sample interval and retention timeframe specific for this CSO may be changed as per your preference and you may specify for which MySQL server(s) this particular CSO should be collected. Also note that one or more Key columns are defined. It must be a column or a set of columns returning (a) unique (set of) value(s) (similar to a UNIQUE KEY in MySQL). Without defining a Key Column, the result monitors might now show proper

values.



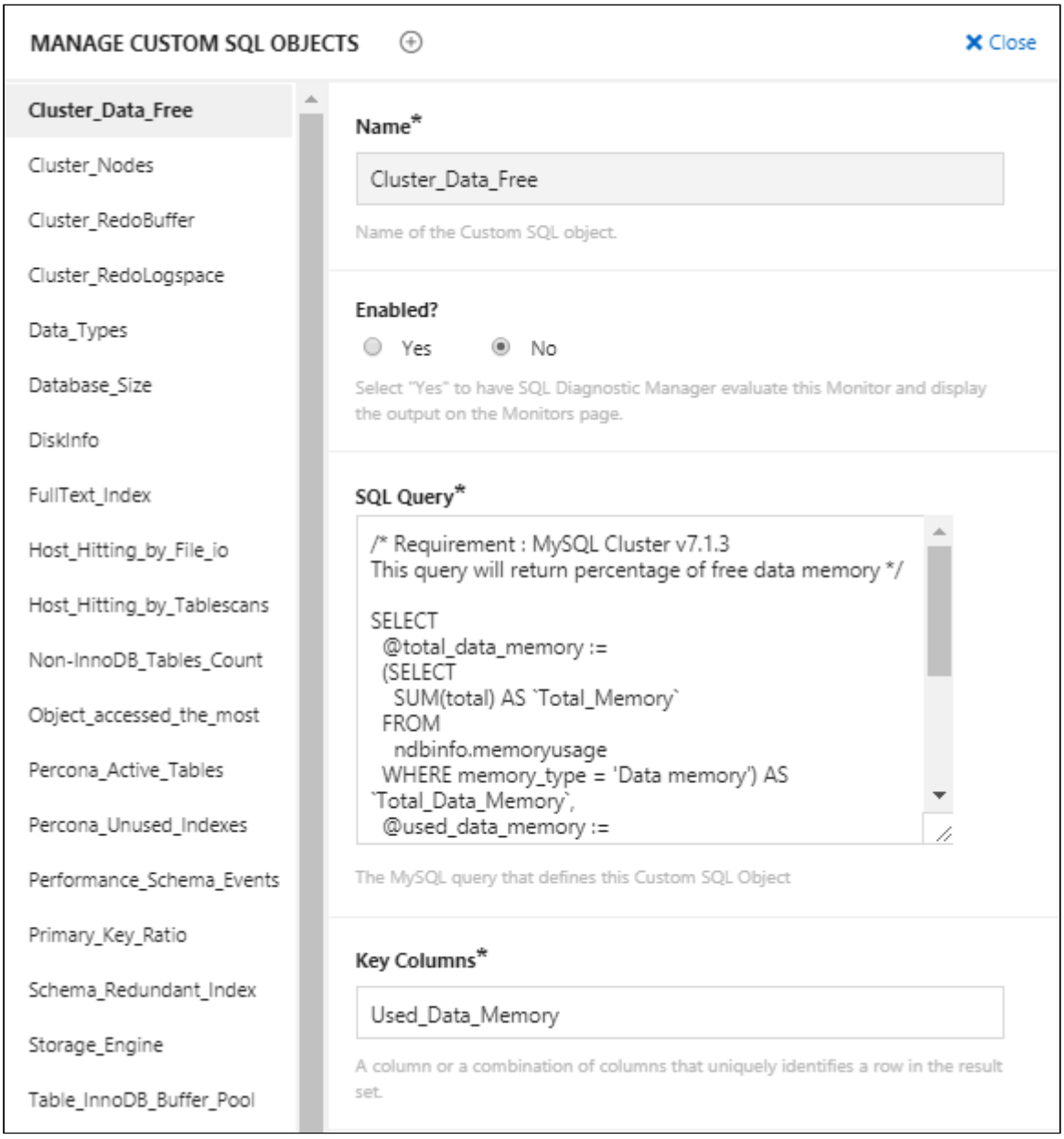
2. Go to Monitors page, select **Manage Monitor Groups**, enable the **DiskInfo** Group, and **Save** the changes. This pre-defined group contains pre-defined CSC's using the CSOs you enabled in the previous step.
3. Go to Monitors page, select the **DiskInfo** group that now displays at the bottom. You can see five new counters in that group that in various ways reference the CSOs that we just enabled (click the counter name and next 'Customize' as usually to see the javascript code). Customize those further as you want to do with any counter in SQL DM for MySQL.

New CSOs and CSCs

What makes a CSO?

Choose Manage CSO from the drop-down. This opens up a form with the following details:

- **SQL**: Any user defined SQL which returns a result set.
- **Key Columns:** CSOs work with a result set which has unique rows. A combination of one or more columns of the result set can be made as a key column as long as long as this key column identifies a unique row in the result set.
- **Server(s):** Comma separated names of the servers for which this SQL needs to be queried every Collection Interval.
- **Data Collection Interval:** Interval in which this SQL is queried periodically. We recommend five minutes which also happens to be the default value.
- **Purging Interval:** The data retention time & we recommend seven days.



Making its CSC

We create a CSC like any other monitor. Go to monitors page and select **Add new monitor** from the drop-down

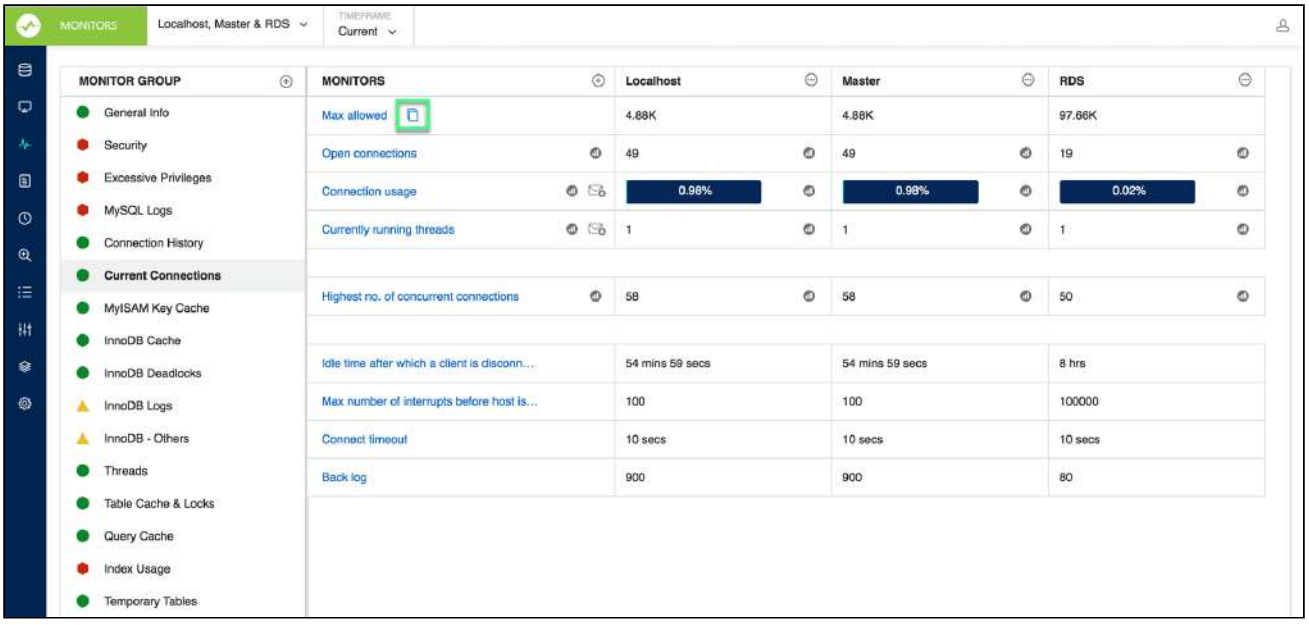
- Enter the name of the counter being added.
- Type in the name of the group to which this counter is being added.
- Choose the type of counter as Custom SQL.
- **Formula**: A MySQL server parameter that is needed to compute the value of this counter.
- **Value**: This defines a function that computes the value. Below, you can find a template:

```
function() {
   var sqlObject = MONyog.UserObject('<Name of your Custom SQL Object>');
   if (!sqlObject || !sqlObject.isEnabled() || !MONyog.MySQL.Custom.Available)
      return '(n/a)';
   /* You will have to call select here to fetch the resultset. */
   var resultSet = sqlObject.select();
   var results = ''; /* results holds the resultset in the form of array of row(s).*/
   /*Get column(s) for each row from the result set */
   for (i in resultSet) {
      if (resultSet.length > 0)
         results += '<br>';
      results += resultSet[i].<Column name in resultset> +
         '.' + resultSet[i].<Column name in resultset>;
      }
   if (results.length == 0)
         results = 'None';
   return results;
}
```

- **Description**: Description of Monitor/Advisor.
- **Advice Text**: Advise text to the Monitor/Advisor being added.

Duplicate Monitors

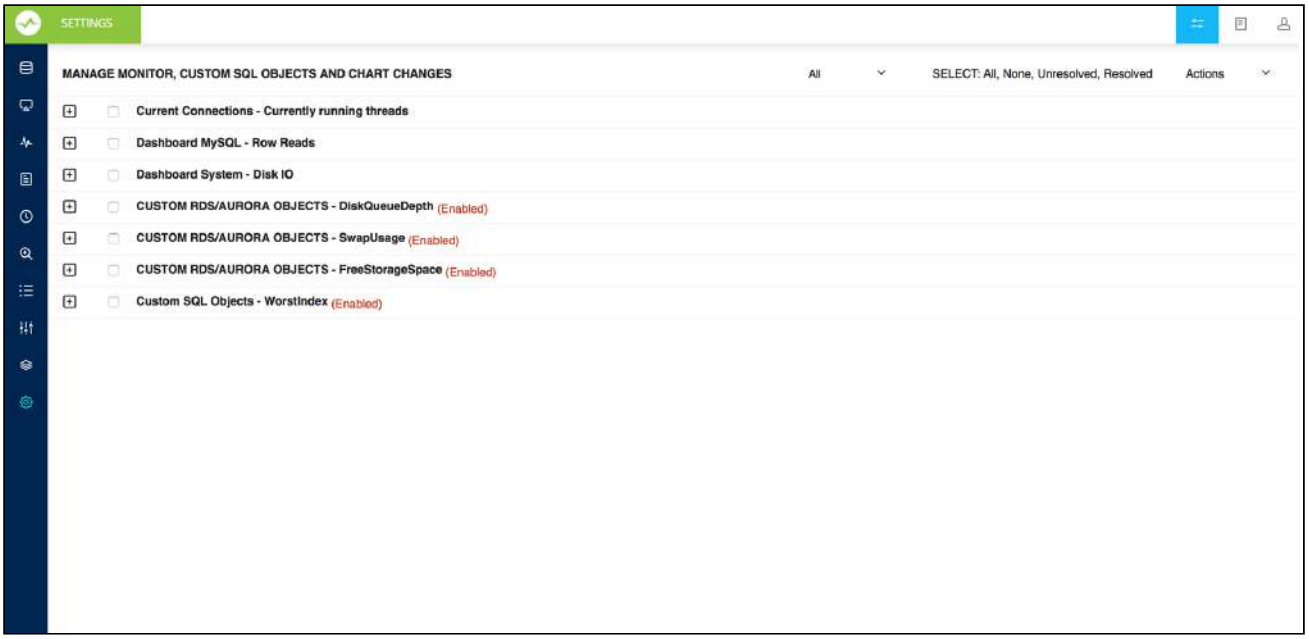
You can duplicate a Monitor and build on that template to make a new Monitor. This is possible by clicking the **Duplicate Monitor** icon adjacent to the respective Monitor name. Refer to Adding or Editing Monitors[32] for more information on adding new Monitors.



32 http://wiki.idera.com/x/mQEGBg

Managing your changes

To review the applied changes in Monitors and/or in Charts, go to Settings and select **Manage changes**.

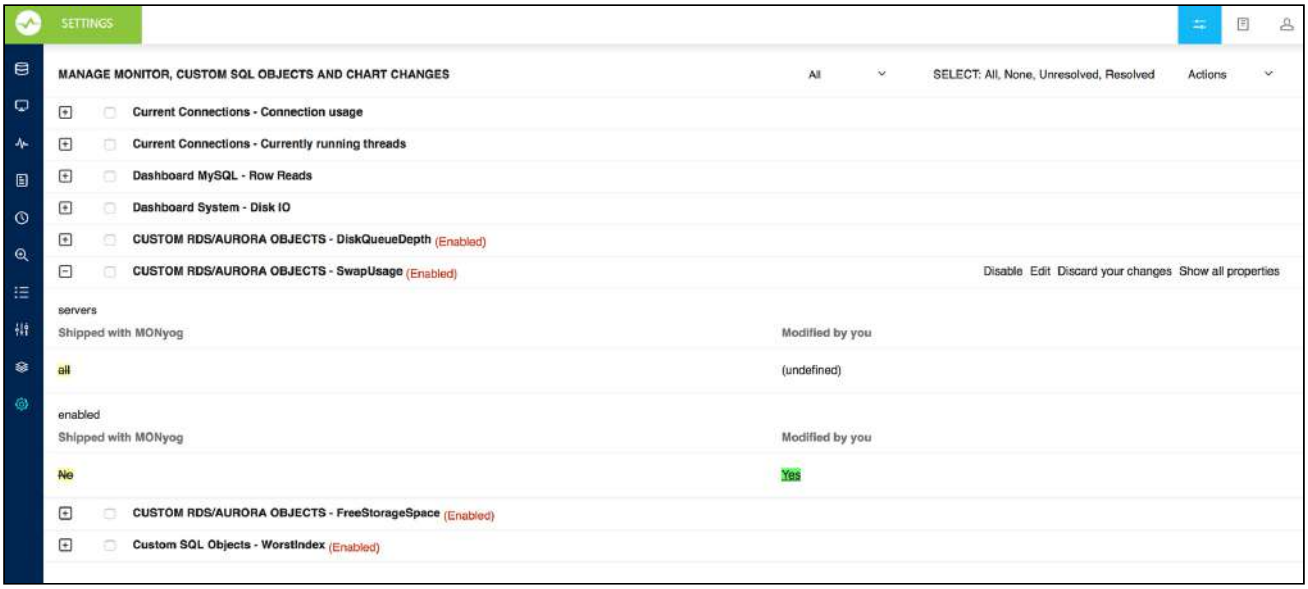


The Manage Changes page lists all the applied changes you made. If the modification is in Monitors or in Dashboard Charts shipped with SQL DM for MySQL, just click it and it shows you the differences between your definition and the original.

In the event that you need to revert any of your changes to existing Monitors or Charts you can just click **Discard your changes**. SQL DM for MySQL reverts to the original definition of the Monitors or Charts supplied with it. Discarding Monitors or Charts that you have created can cause a permanent deletion.

Manage Custom SQL Objects

Review the applied changes of the Custom SQL Objects, go to Settings and select **Manage changes**.



Upgrading SQL DM for MySQL

Before explaining how SQL DM for MySQL manages users changes when upgrading, we need to go over two terms:

- **Resolved:** Monitors or Charts or Custom SQL Objects marked as resolved (displayed in white rows in the Settings -> Manage changes page) are currently in use by SQL DM for MySQL. You can see these Monitors or Charts or Custom SQL Objects either on the Monitors page or the Charts page as the case may be.
- **Unresolved**: Monitors or Charts or Custom SQL Objects marked as unresolved (displayed in red rows in the Settings -> Manage changes page) are neither loaded nor used by SQL DM for MySQL. You have to explicitly

resolve such conflicts. Resolving a conflict is as simple a task as deciding whether to keep your changes or discarding them.

SQL DM for MySQL customization framework allows users changes to propagate from one version to another while upgrading. There are certain cases that you need to keep in mind:

- Integrating a Monitor or a Charts or Custom SQL Object which is shipped with SQL DM for MySQL but has been modified by you in another version causes a conflict, i.e. SQL DM for MySQL does not know which definition to use. To resolve this conflict click on **Settings -> Manage changes**, select the unresolved rows, and either select **Use your changes** to keep your changes or **Discard your changes** to revert back to original definition of the Monitor or a Charts.

> ⚠ You can perform the same action for an individual conflicted Monitor or Charts by clicking **Discard your changes**, by expanding that particular conflicted Monitor or Charts or Custom SQL Object.

- When integrating a Monitor or a Charts or Custom SQL Objects created by you, SQL DM for MySQL automatically assimilates, and start using it without your intervention.

Monyog object model

The Monyog Object model (MOM) abstracts all OS/MySQL values required for calculating all performance metric. It relieves the user from performing low level tasks like connecting to the servers, executing SQL statement, checking return codes, etc.

Values from MOM are used to calculate and display metric, check thresholds, send notification mails, etc.

All output returned from the "SHOW GLOBAL STATUS" is available as MONyog.MySQL.GlobalStatus.*. For example: to get the value of Uptime status variable, you can use MySQL.GlobalStatus.Uptime.

Similarly the following Object Models are defined as follows:

Variable	MOM Object
`SHOW GLOBAL VARIABLES`	`MONyog.MySQL.GlobalVariables.*`
`SHOW GLOBAL STATUS`	`MONyog.MySQL.GlobalStatus.*`
`SHOW SLAVE STATUS`	`MONyog.MySQL.Slave.*`
OS-level counters	`MONyog.System.*`

> ⚠ Operating system-level counters measure CPU, memory usage, and so on.

MOM variable for SQL DM for MySQL tags and server names(see page 113)

AlertCondition:

This attribute expects an expression that should evaluate to one the following 3 values: Critical, Warning, or None. Generally, a JS expression or function is specified which compares some of the MOM values to the Warning or Critical threshold values defined above. (Optional) Consider two examples of a monitor to receive alerts based on an alert condition.

```
function()
{
    if(ToInt(MONyog.MySQL.GlobalStatus.Select_scan) > 1000)
        return GetWarnStatusInt(this.Value, this.Critical, this.Warning, true);
    else
        return "None";
}
```

```
function()
{
    if(this.Value != "(n/a)" && this.Value > 1)
        return "Critical";
    else
        return "None";
}
```

WarmUpRequired: A value of Yes specifies that this metric makes sense only if the server is running for a minimum period of time. If the server is not running for the minimum period, AlertCondition is not evaluated and no alerts would be displayed or notified. The default value is 3 hours, if you want to change this value; go to Settings, select **General** you find MYSQL WARMUP PERIOD.

MailAlert: Specifies whether the user wants mail alerts for this metric in case the thresholds are crossed. (Optional)

Graph: This value defines whether real-time graphs are shown for this metric. (Optional)

Bargraph: This value defines whether Percentage type value should be plotted as Bar Graph. (Optional)

Uptime: This value determines whether this metric contain cumulative values. Cumulative values are those values which always increase continuously since server startup (or since last FLUSH STATUS). For example: Connection Attempts. The value of Connection Attempts is always incremented by the MySQL server. Cumulative values are treated differently from "point in time" values like "Currently Running Threads".

If a metric is Cumulative you should always set the value of this attribute to a constant expression: "MONyog.MySQL.GlobalStatus.Uptime"

Format: The display format of various counters. The only possible value currently applicable is - NumCounterWithSeconds. This specifies whether the metric values should also be displayed in "per second" value.

AdviceText: The advice text that is shown to a user whenever any AlertCondition evaluates to "Critical" or "Warning". This text is also shown as the tool tip when the user points the mouse over the alert icons.

RetryOverride: A MOM variable that takes an integer value and overrides the server-level "Send notification when alert-able" setting at the counter-level. Note: It does not take the value "0".

NotifyStableOverride: A MOM variable that takes either "Yes" or "No" as a value and overrides the server-level "Notify when stable" setting at the counter-level.

MOM variables for SQL DM for MySQL tags and server names

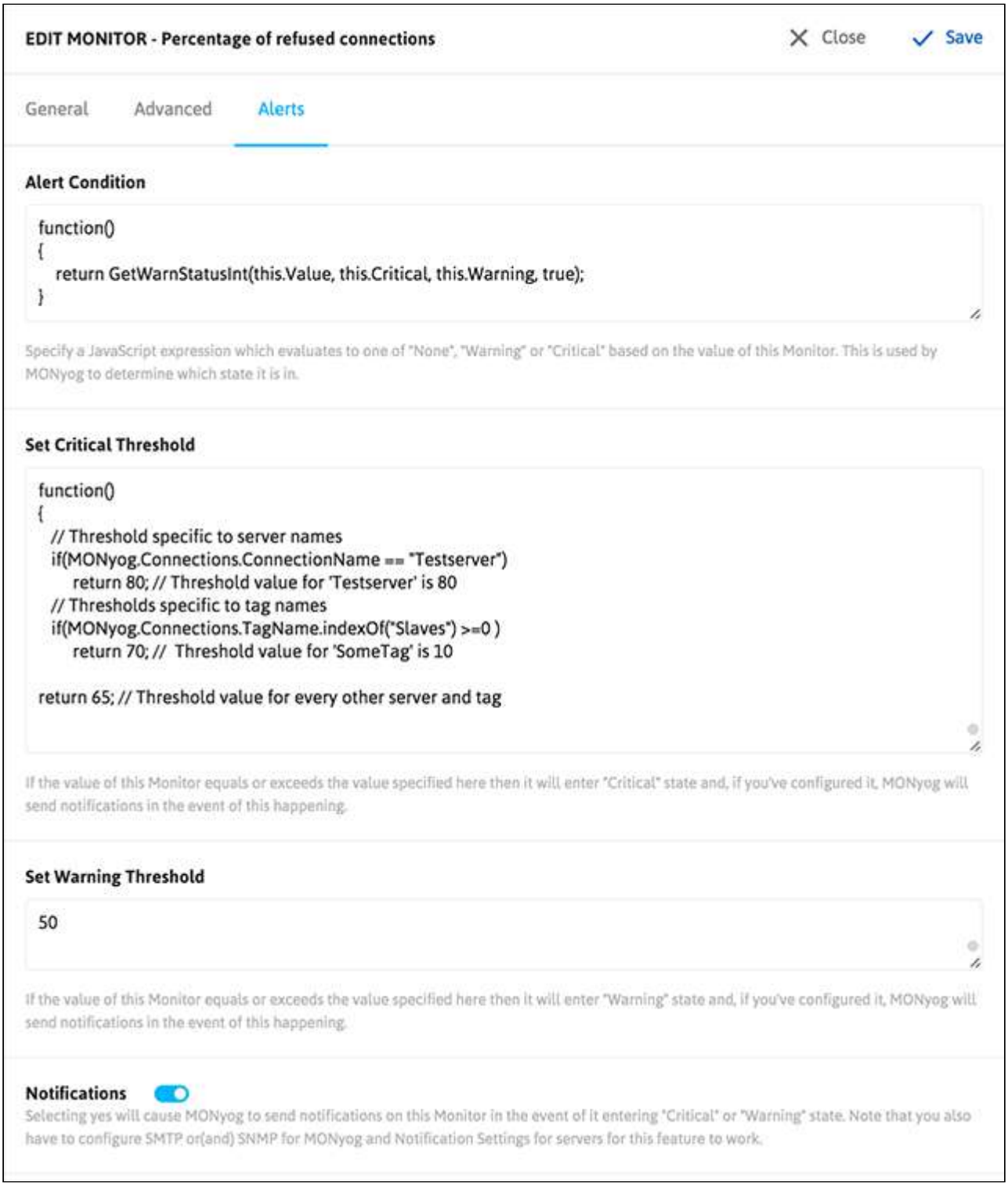
Tags and server names that are used in SQL DM for MySQL are exposed as Monyog Object Model (MOM) variables. MONyog.Connections.TagName returns an array of tags for that server.

`MONyog.Connections.ConnectionName` Gives the name of that server. This can be extremely useful while setting different threshold levels (based on tags or server names) for a monitor. For example:

The following can be added to the critical/warning field in **Add/Edit server->View Advanced** to set server and tag specific thresholds.

```
// Threshold based on server names
if(MONyog.Connections.ConnectionName == "Testserver")
   return 80; // Threshold value for 'Testserver' is 80
if(MONyog.Connections.ConnectionName == "Productionserver")
   return 50;
// Thresholds based on tag names
if(MONyog.Connections.TagName.indexOf("SomeTag") >=0 )
   return 10; //  Threshold value for 'SomeTag' is 10
if(MONyog.Connections.TagName.indexOf("SomeOtherTag") >=0 )
   return 39;
```

Example of a Monitor(Percentage of max allowed reached) that is customized to receive alerts based on server names and tags.



SQL DM for MySQL Attribute Reference - Charts Interface

- **Chart Name:** The name of the new Chart.
- **ChartType:** The type of chart can be MySQL or System.

- **SeriesCaption:** Array containing labels for every series in a graph.
- **SeriesValues:** Array containing the values of each series in a graph.
- **ChartValue:** The type to plot the actual values of the seconds_behind_master where it is the difference between the 2 intervals. It can plot the values in 2 ways:
    - **Delta**
    - **Current**

  And the default is **Delta**.
- **This chart shows boolean values:** Possible values are "OnOff" only. This is a special time of Y-Axis plotting that has only 2 possible values - "On" or "Off". This type of graph is useful for plotting Availability status of MySQL/OS across a timeframe. (Optional)
- **Unit:** Specify y-axis unit in the chart. Ex: ['','KB','MB','GB','TB']
- **Unit Factor:** The limit when the unit should be incremented. Ex: if this value is 1024 then, 1024 = 1KB, 1024KB=1MB and so on.
- **Uptime:** See the section for "SQL DM for MySQL Attribute Reference - Monitors Interface". (Optional)

System Information Populated by MOM

The System Information is populated by MOM is divided into the following categories:

General

- `MONyog.System.General.version`: The Linux kernel version.

Physical Memory (in Kilobytes)

- `MONyog.System.Mem.sys_mem_total`: Total physical memory.
- `MONyog.System.Mem.sys_mem_free`: Available physical memory.
- `MONyog.System.Mem.proc_mem_vmrss`: Physical memory being used by MySQL.

Swap memory (in Kilobytes)

- `MONyog.System.Swp.sys_swp_total`: Total swap memory.
- `MONyog.System.Swp.sys_swp_free`: Free Swap memory.
- `MONyog.System.Swp.proc_swp_vmsize`: Swap memory being used by MySQL.

CPU

Below are the CPU related metrics. Each gives the number of jiffies spent in various modes, since the last capture.

- `MONyog.System.Cpu.sys_cpu_user`: User mode
- `MONyog.System.Cpu.sys_cpu_nice`: Nice mode
- `MONyog.Systeem.Cpu.sys_cpu_system`: System/Kernel mode
- `MONyog.System.Cpu.sys_cpu_idle`: Spent Idly
- `MONyog.System.Cpu.sys_cpu_iowait`: Spent in waiting for IO
- `MONyog.System.Cpu.sys_cpu_hi`: Spent in hardware interrupts
- `MONyog.System.Cpu.sys_cpu_si`: Spent in Software interrupts

I/O

Below are the block devices related metrics. Each gives the number of blocks read and written to the devices attach to the system.

- `MONyog.System.Io.blocks_in`: Total number of blocks read from the devices.
- `MONyog.System.Io.blocks_out`: Total number of blocks written to the devices.

Custom

- `MONyog.System.Custom.Available`: If the system is available to Monyog or not.

> ⚠ Currently "Timeframe" does not have any effect on system related values.

Disk

- `MONyog.System.Disk.sys_disk_free_mysql`: Amount of free space left on the volume where MySQL data resides.
- `MONyog.System.Disk.sys_disk_freepercent_mysql:` Percentage of free space left.
- `MONyog.System.Disk.sys_disk_total_mysql`: Total size of the volume where MySQL data resides.
- `MONyog.System.Disk.sys_disk_used_mysql`: Space being used by various files on the volume.
- `MONyog.System.Disk.sys_disk_free_innodb`: Amount of free space left on the volume where InnoDB data resides.
- `MONyog.System.Disk.sys_disk_freepercent_innodb`: Percentage of free space left.
- `MONyog.System.Disk.sys_disk_total_innodb`: Total size of the volume where InnoDB data resides.
- `MONyog.System.Disk.sys_disk_used_innodb`: Space being used by various files on the volume.

Connection:

Connection name, MySQL user, SSH user, SSH tunneling user which are saved in connection details are exposed for customization in Monitors. (For instance 'connection name' can be accessed using MONyog.Connections.ConnectionName, 'MySQL user' as-MONyog.connections.MySQLUser etc.)

- `MONyog.connections.ConnectionName`: Name of that server
- `MONyog.connections.TagName`: Returns an array of tags for that server
- `MONyog.connections.MySQLUser`: Using this MySQL user can be accessed
- `MONyog.connections.MySQLHost`: MySQL host can be accessed
- `MONyog.connections.MySQLPort`: MySQL port can be accessed

> ⚠ To retrieve system counters from Linux and access log files from remote system on all platforms, SSH server uses below variables:

- `MONyog.connections.SSHHostSystem`: To access SSH host
- `MONyog.connections.SSHPortSystem`: To access SSH port
- `MONyog.connections.SSHUserNameSystem`: To access SSH username

> ⚠ If you have used SSH tunneling to your MySQL server below variables can be used:

- `MONyog.connections.SSHHostTunnel`: To access SSH host
- `MONyog.connections.SSHUserNameTunnel`: To access SSH username
- `MONyog.connections.SSHPortTunnel`: To access SSH port

MySQL Information Populated by MOM:

MySQL Error log

- `MONyog.MySQL.ErrorLog.Is_accessible`: To access the MySQL error log
- `MONyog.MySQL.ErrorLog.Total_size`: Size of MySQL error log
- `MONyog.MySQL.ErrorLog.Size_changed`: To determine any new entry is there or not in MySQL error log
- `MONyog.MySQL.ErrorLog.Last_error`: Last error in the MySQL error log

Innodb Deadlock

- `MONyog.MySQL.InnodbStatus.Deadlock_detected`: Any new Innodb deadlock is found
- `MONyog.MySQL.InnodbStatus.Last_detected_time`: Period in which last deadlock is detected
- `MONyog.MySQL.InnodbStatus.Latest_deadlock`: Latest deadlock is detected

Customizing Monitors

Everything that you see in the Monitors interfaces is defined as JavaScript (JS) objects using Monyog Object Model (MOM) and are editable through a simplified interface from within SQL DM for MySQL.

The SQL DM for MySQL daemon/web server has an embedded JS execution engine. Whenever a request for a report is received by SQL DM for MySQL, a new JS runtime context is created. This runtime compiles and evaluates the JS objects defined by various scripts generated from user input, formats the result, and sends the formatted report to the client. The following sections get into the details of each and every aspect of SQL DM for MySQL JS environment.

Each of the SQL DM for MySQL Advisor rules allows the MySQL DBA to customize the thresholds that are acceptable for specific MySQL servers. As an example, a DBA using the supplied Advisor Rule ''MyISAM Key Cache Misses'' may use lower threshold values for their MySQL servers running OLTP applications, while higher thresholds may be acceptable for OLAP applications.

SQL DM for MySQL customization framework empowers users with the ability to add new advisors, modify existing advisors, and even disable some of the predefined advisors supplied by IDERA.

Adding or Editing Monitors

To add a new Monitor do the following:

- Click the **Add new Monitor** link from the drop-down in Monitors page.
- Choose the type of counter - MySQL, System, or Custom SQL.
- Type in the name of the group to which this monitor is being added.
- Enter the name of the counter being added.

For editing a monitor, click **Edit Monitor** icon beside the monitor in Monitors page.

Formula:

A MySQL server parameter that is needed to compute the value of this counter.

Value: Defines a function that computes the value.

Example:

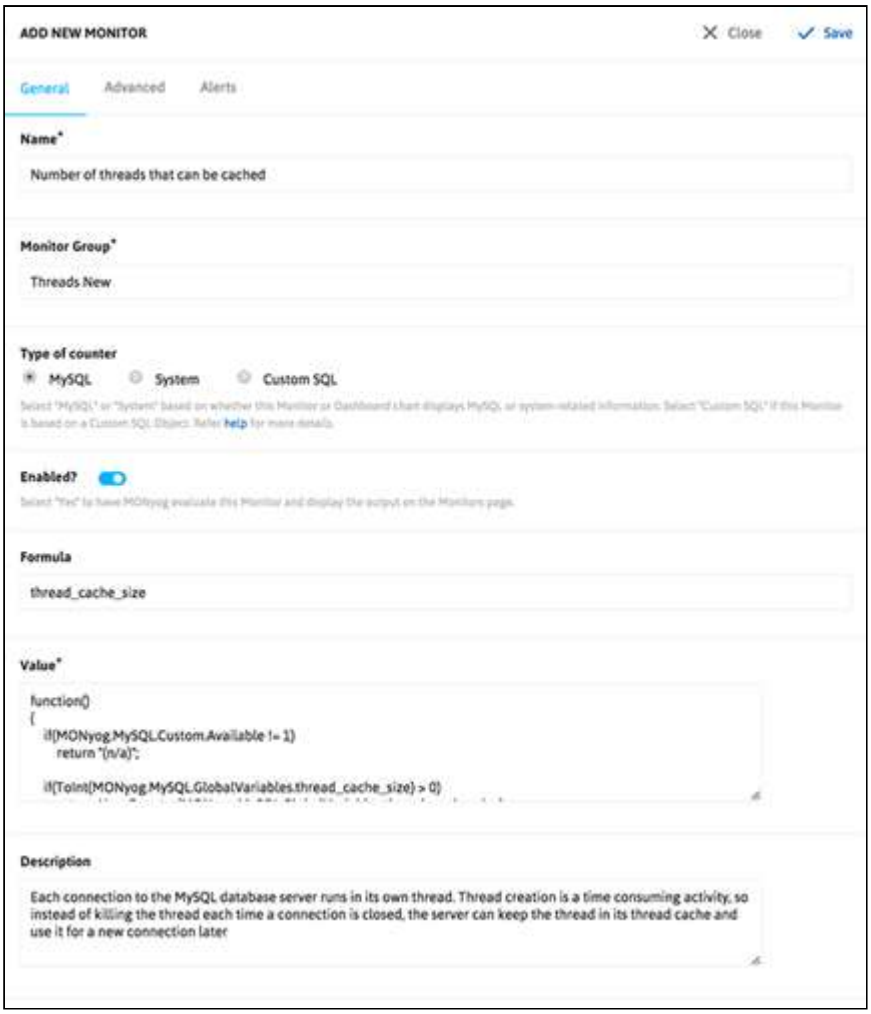
```
function()
{
    if(MONyog.MySQL.Custom.Available != 1)
        return "(n/a)";

    return MONyog.MYSQL.GlobalVariables.max_allowed_packet;
}
```

For a detailed description of values, see Monyog Object Model[33] for more information.

DocText: Description of Monitor.

Advise Text: Advise text to the Monitor being added.



In this customization form each field corresponds to a valid MOM property. Some fields take on valid JavaScript expressions while others take on text/HTML as values. The values entered in these fields are used to generate JS objects which are basically name-value pairs.

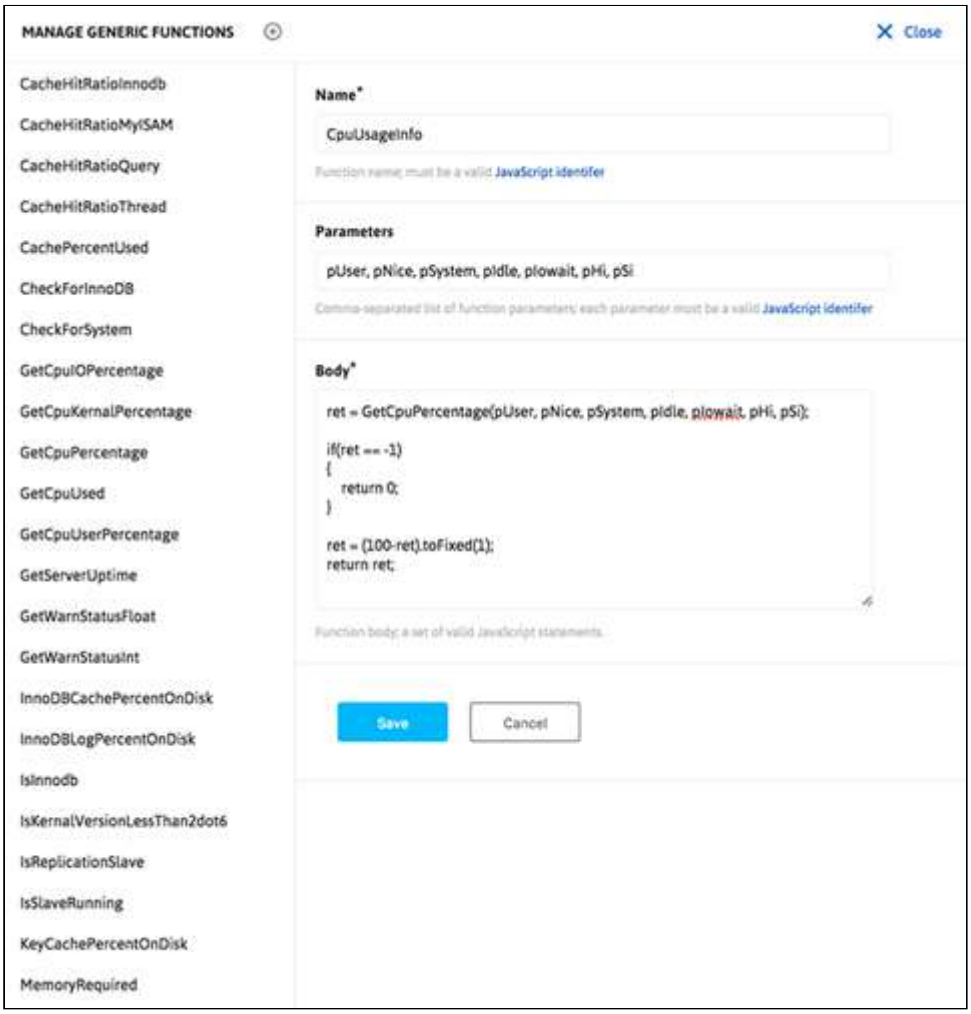
33 http://wiki.idera.com/x/lwEGBg

The General tab displays properties which are basically necessary to define a Monitor. Properties with asterisk(*) are mandatory. To customize the behavior of the Monitor further, click the **Advanced** that displays all MOM properties.

> For more information, see New CSO's and CSC's[34]

Add generic functions

Common functions required across multiple Monitors can be put into generic functions. The generic functions shipped with SQL DM for MySQL cannot be edited. Only the user added generic functions will be editable. For example, the function StorageUnits defined in generic functions takes a numeric value as a parameter and returns a string which is the value specified in terms of K(ilo), M(ega), B(ytes), etc. To add a generic function click the icon adjacent to the title.



34 http://wiki.idera.com/x/lAEGBg

Enable/Disable notification for a particular Monitor.

SQL DM for MySQL allows user to disable notification for a monitor for a particular server. When notification is enabled for a monitor it generates alerts for all the servers whose values are alertable which may include servers that are not of interest for that monitor. Hence, receiving unwanted alerts is completely irritable. Now, you can avoid notification for a particular server or group of servers by using the below API.
The API action is to disableNotification and enableNotification.

Example API:

To disable notifications for “Percentage of max allowed reached” monitor under “connection history” monitor group for the servers under the tag “Prod”.

```
$ Curl “http://192.168.1.1:5555/?
_object=MONyogAPI&_action=disableNotification&_groupid=5&_counterid=10&_tag=Prod”
```

To enable notifications for “Percentage of max allowed reached” monitor under “connection history” monitor group for a particular server named “Staging-DB”.

```
$ Curl
“http://192.168.1.1:5555/?
_object=MONyogAPI&_action=enableNotification&_groupid=5&_counterid=10&_server=Staging
-DB”
```

You can check the Group ID and Counter ID of a monitor by hovering over the name of the monitor on the monitors page.

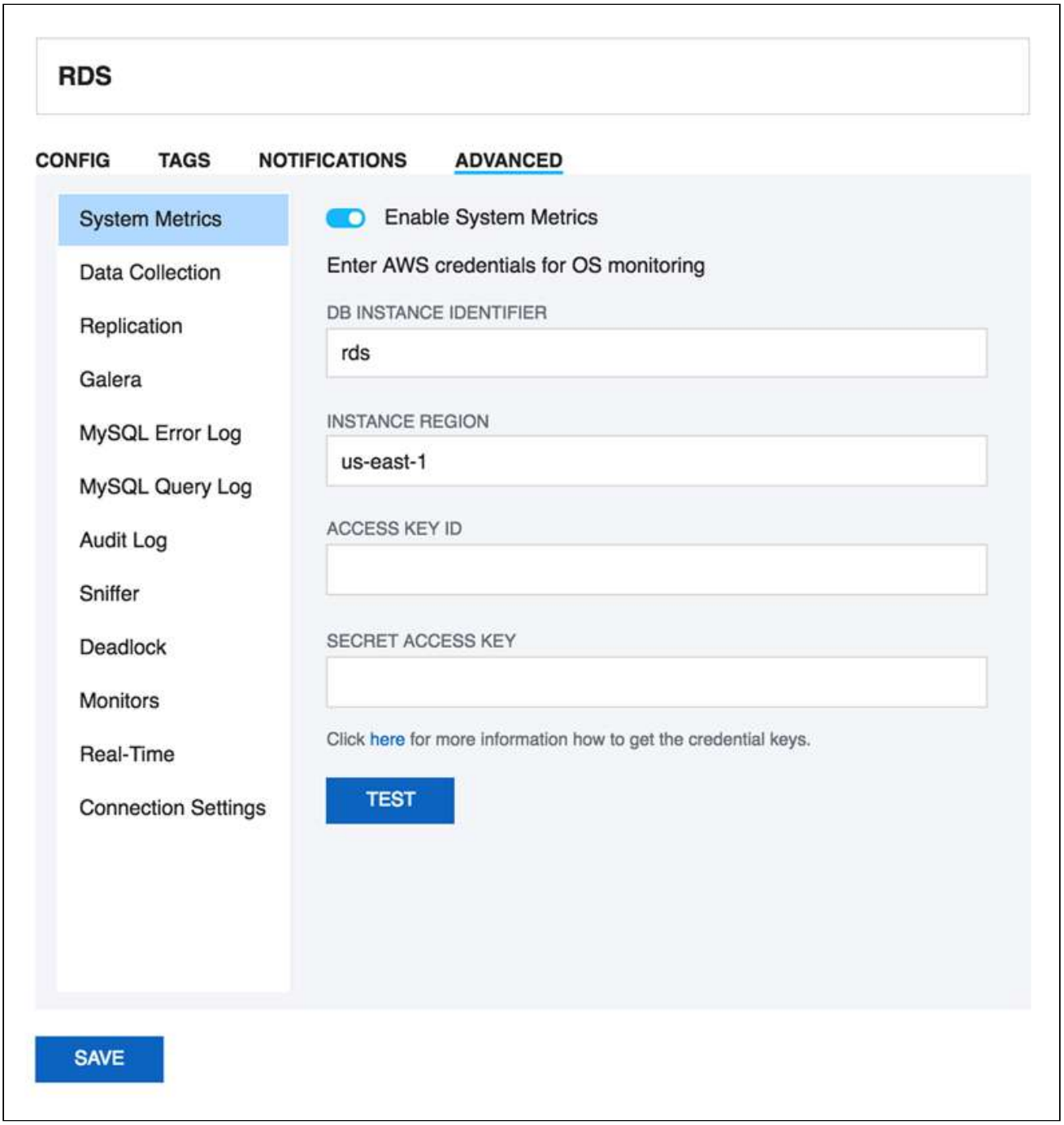
How to RDS_Aurora OS monitoring in SQL DM for MySQL

RDS/Aurora OS monitoring feature introduced with Monyog-8.1.0. SQL DM for MySQL makes use of the CloudWatch API and uses the different OS metrics available with the API to fetch and display the data. All the RDS/Aurora OS monitors are shown under the monitor group "RDS/Aurora Instance Metrics" in Monitors page and the corresponding charts are available on the Dashboard page. In order to be able to see the OS data, you should first enable system metric for the RDS/Aurora instance

Enabling System Metrics

Go to **Servers**, select **Edit server**, and choose **Advanced** of the RDS/Aurora instance and **Enable System Metrics**. Once enabled, the user should enter the following four parameters:

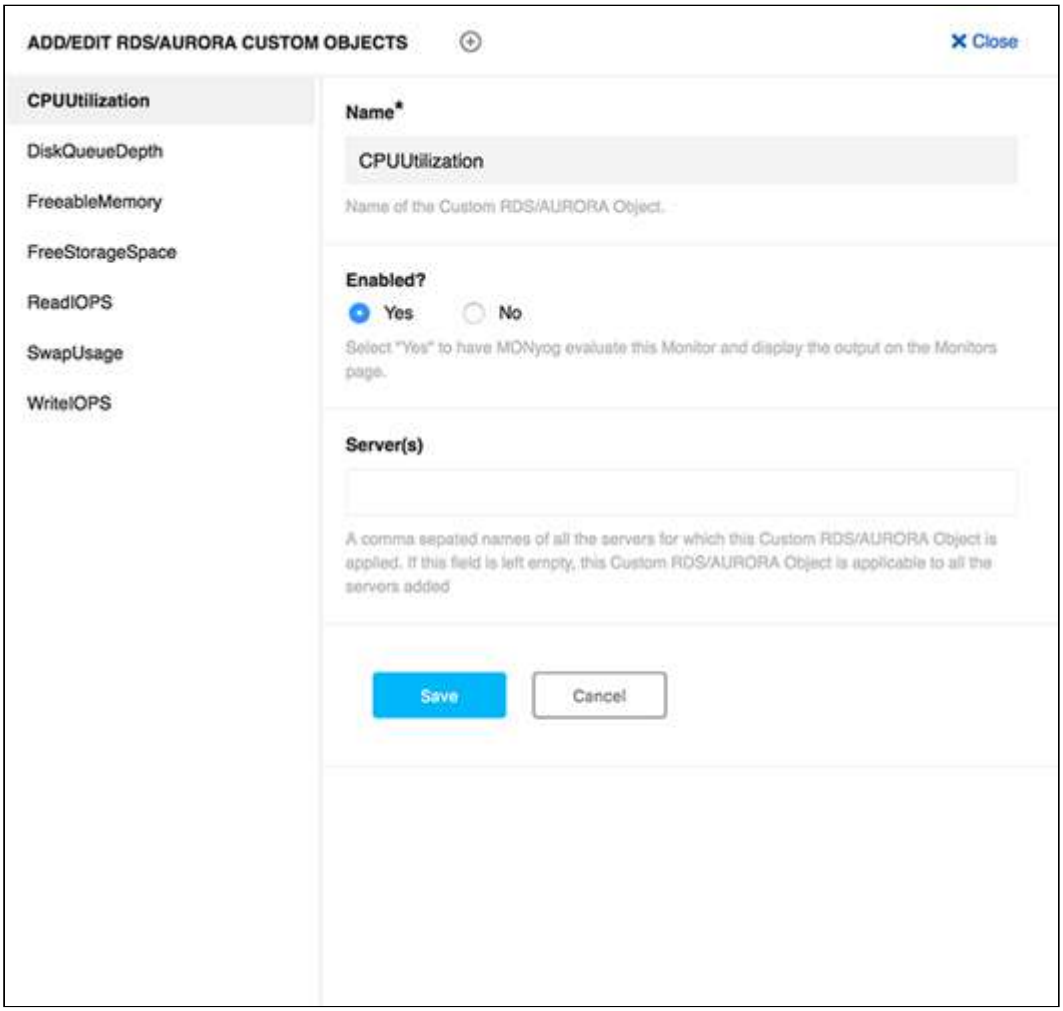
1. **DB instance identifier:** A unique name to identify your RDS/Aurora instance.
2. **Instance region:** The region in which your instance is hosted, for e.g: us-east-1
3. **Access key ID:** It is a twenty character long key ID which can be created from the AWS Management Console. It is used to make a programmatic request to AWS.
4. **Secret access key:** It is a forty characters long and can be created from the AWS Management Console. You can refer to the documentation, on how to generate credential keys in the following page: Getting your Access Key ID and Secret Access Key(see page 121).



Enabling RDS/Aurora Custom Objects

Once the System metrics is enabled, you start getting OS data in the monitor group **RDS/Aurora Instance Metrics.** The custom objects CPU Utilization, Freeable memory, Read IOPS, and Write IOPS are enabled by default, while rest are disabled. In order to enable the others:

Go to **Monitors,** select **RDS/Aurora Instance Metrics**, click the ⊕ icon, select **Manage RDS/Aurora Custom Objects,** and select **Yes** for the Enabled option for any of the listed custom objects.



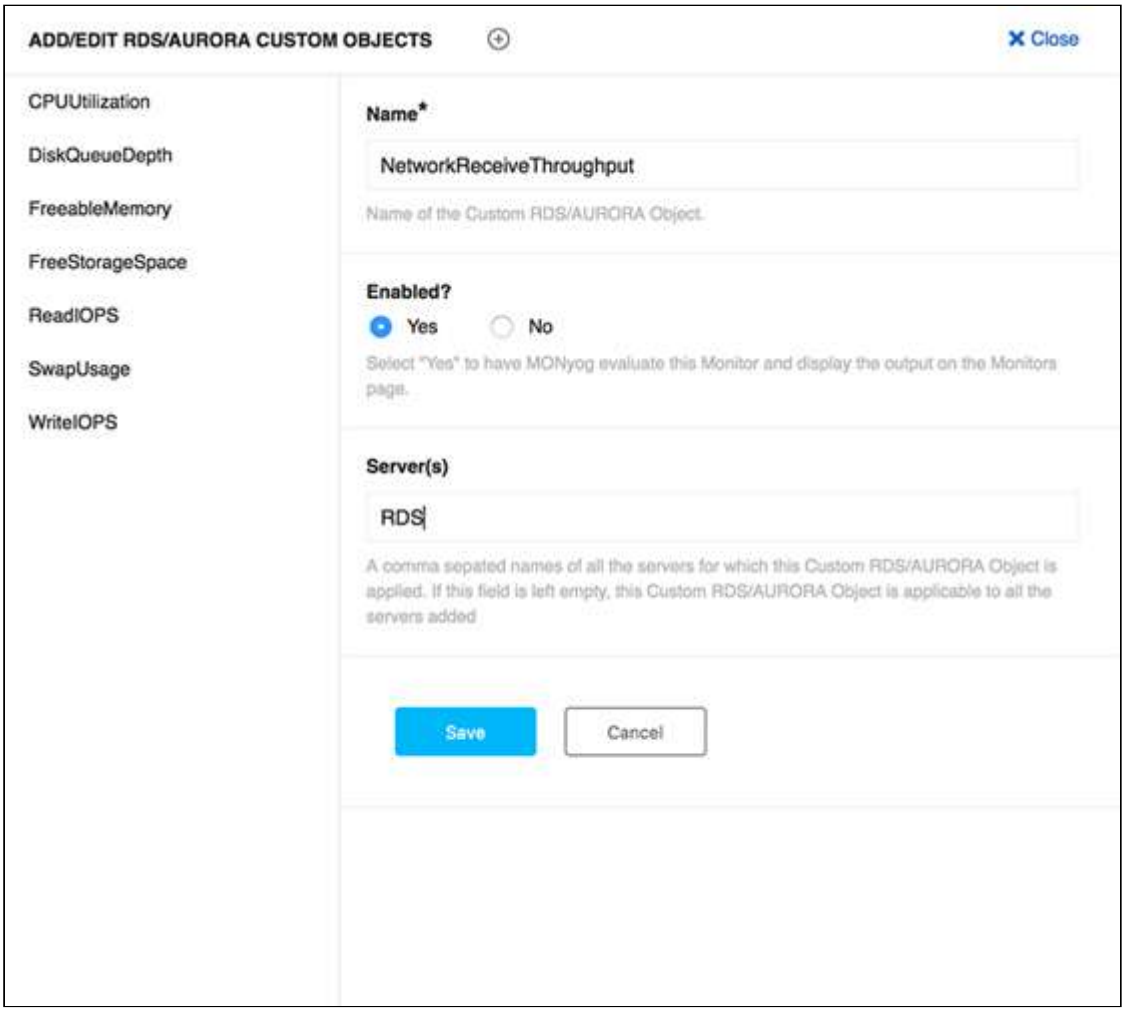
Adding New RDS/Aurora Custom Objects

Apart from the default metrics shipped with SQL DM for MySQL, any user can also add an RDS/Aurora Custom Object from the list of available CloudWatch metrics (Refer here for the CloudWatch(see page 121)

metrics available for RDS and here for the Aurora instances(see page 121)). Adding RDS/Aurora Custom Objects is a 2-step process:

Adding a Custom Object

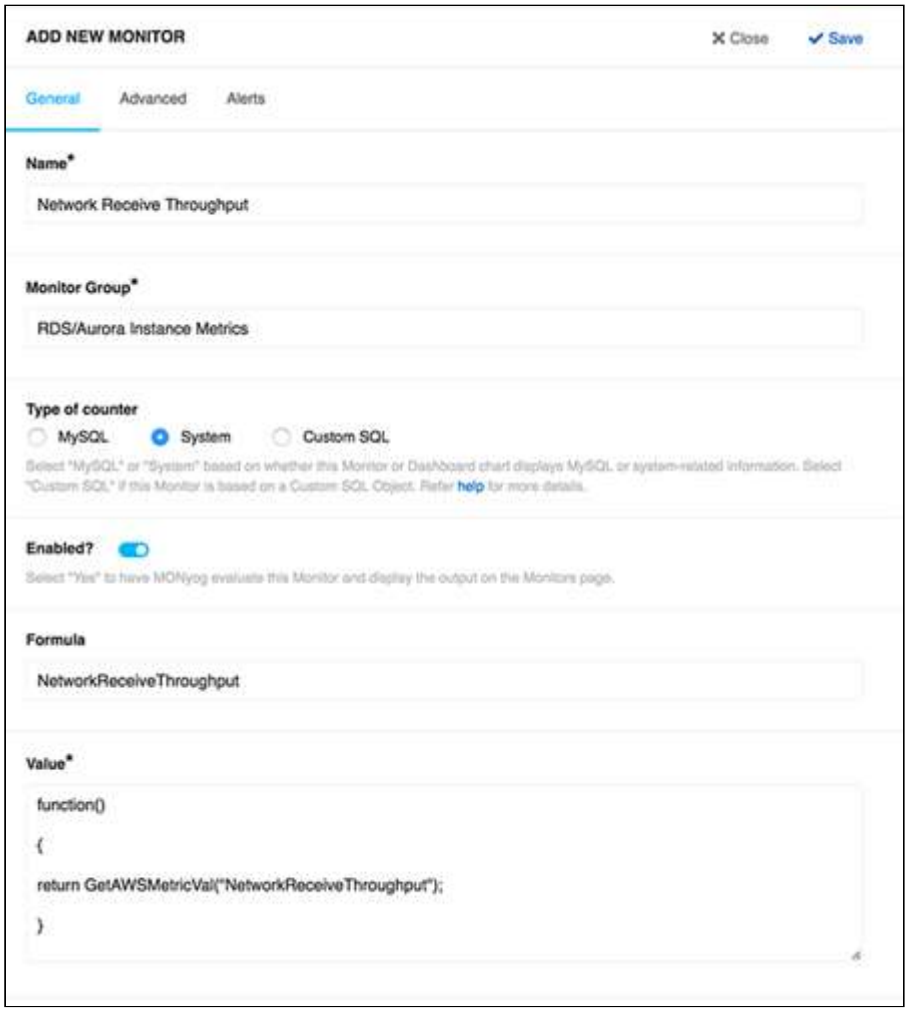
1. Go to Monitors, select **RDS/Aurora Instance Metrics**, click the ⊕ icon, select **Manage RDS/Aurora Objects**, and click **Add new RDS/Aurora custom object (+)** .
2. Enter the name of the CloudWatch Metric which you want to add under the Name field. You can also give the name of the servers comma separated for which you want this metric to be evaluated for.



Adding the Monitor

1. Go to Monitors, select **RDS/Aurora Instance Metrics**, click the ⊕ icon, and select **Add new monitor.**
2. Enter Monitor name and the Monitor group name in which you want to add this new monitor to (use "RDS/ Aurora Instance metrics" if you want to add it in this group). Select **System** as "Type of counter" if you are adding a system metric.
3. Enter a simple JavaScript function in the Value field using the Cloudwatch metric like:

```
function()
{
 return GetAWSMetricVal("NetworkReceiveThroughput");
}
```



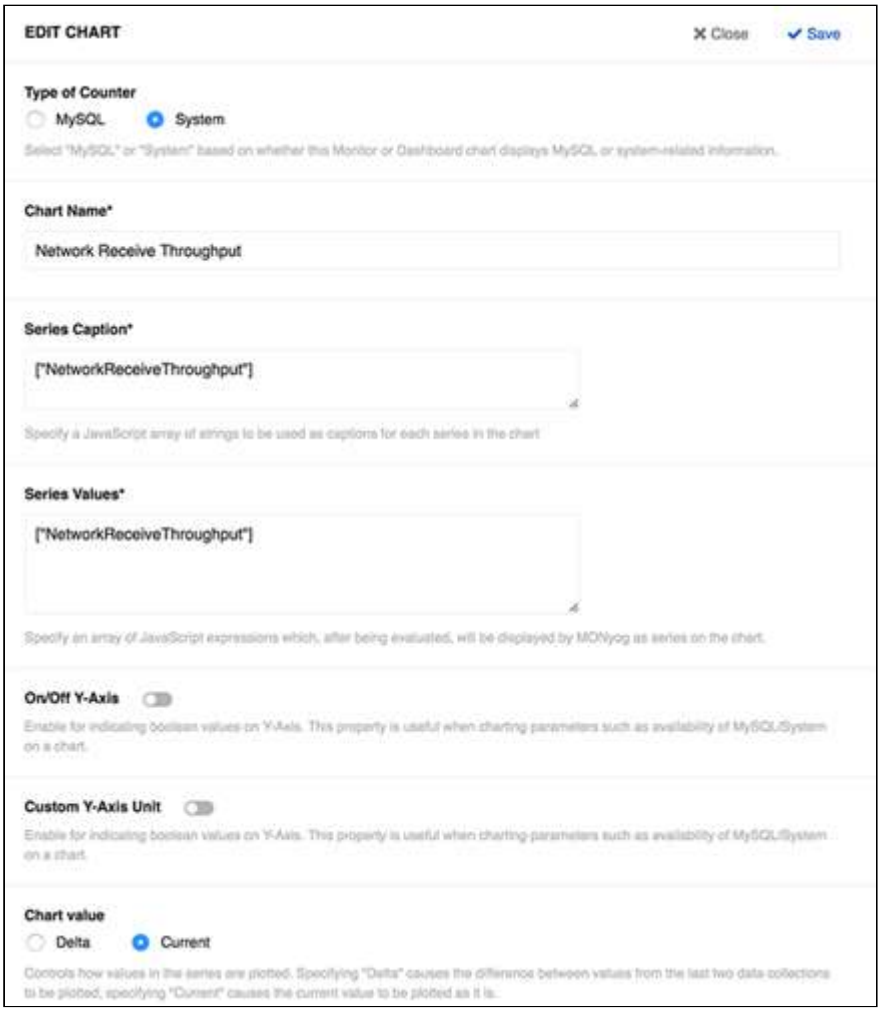
Enabling RDS/Aurora Dashboard charts

Go to **Dashboard**, select **Manage Dashboard**, and **Enable the listed RDS/Aurora charts** under System Charts. The Dashboard page gives the flexibility to create a dashboard with a particular set of charts, so a user can create a dashboard with only RDS OS metrics charts for ease of monitoring.

Adding New RDS/Aurora Dashboard charts

To add a RDS/Aurora chart in the dashboard page, the corresponding RDS/Aurora custom objects should be defined and enabled at **Monitors -> RDS/Aurora Instance Metrics -> click** the ⊕ icon **-> select Manage RDS/Aurora Objects**. Once enabled, follow the steps below to add the chart:

1. Go to **Dashboard** , select **Manage Dashboard**, and click **Add new chart** ⊕ .
2. Select **System** as the "Type of counter" and give a proper chart name.
3. Enter the **RDS/Aurora custom object** in the series caption field and the corresponding MOM (Monyog Object Model) variable in Series values field.



Threads

The Threads page shows you the number of threads currently being executed by MySQL. Each query sent to MySQL is executed in a thread. The Threads feature is used to check which queries are being executed presently. It gives you a general sense of what is keeping your server busy at that very moment.

However, this feature is not the right tool to review statistical information about the queries executed on the MySQL server over a period of time. For finding problem SQL, for example, you should use the Query Analyzer which takes a snapshot of queries, their execution times, etc. over a period of time and presents you with a consolidated report. You can then use the report for further analysis; you can even export it as CSV and run it through other tools.

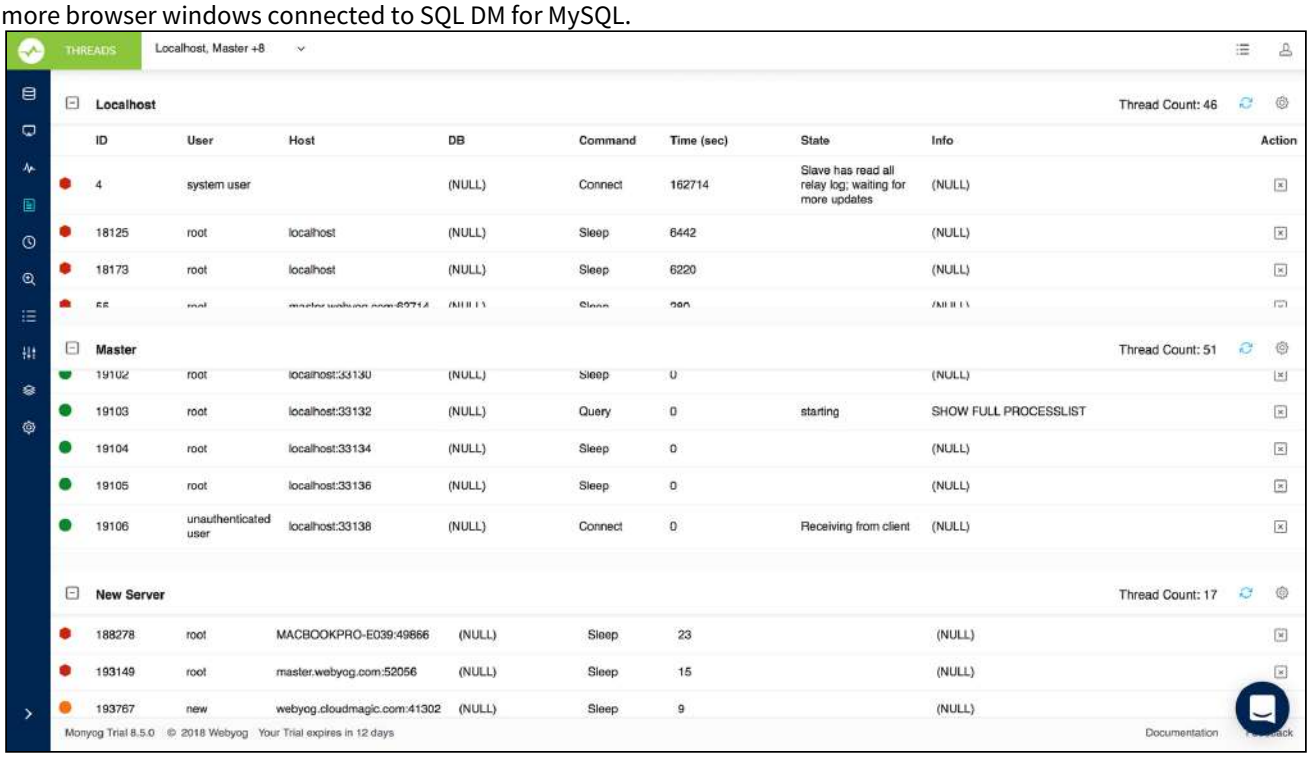
The view can be filtered and sorted as described subsequently. Columns include the connection ID, the connection status, and the text of the current query. From this display you can select a query to view, copy, explain, or kill a query. The **Settings** icon for the respective server thread table allows you to change the PROCESSLIST query to display information you want. This gives you unprecedented flexibility to filter data.

Layout_View of Threads

View Queries in real time

The rows returned by the `SHOW FULL PROCESSLIST` query to the MySQL server are stored in a table (named `processlist`) in the embedded database - one column for every column returned plus one additional column named `action`. The Action column contains an HTML string that is used for the various options that are provided by SQL DM for MySQL. This table is stored in a `MEMORY` type database (no disk storage) and only the data returned by the latest `SHOW FULL PROCESSLIST` for each server is saved.

Also, `PROCESSLIST` data is only retrieved when the corresponding page in the web interface is showing in one or more browser windows connected to SQL DM for MySQL.



Actions on currently running queries

Query Details and Kill

SQL DM for MySQL enables you to do the following actions on your threads:

- `VIEW` the current query for the selected process.
- `EXPLAIN` the query currently executed by the selected process (EXPLAIN works for DML queries only) and this shows the optimized query returned by EXPLAIN EXTENDED if MySQL server supports.
- `KILL` a process.

Privileges required for optimal use of this feature

You should notice that to retrieve information about processes of other users, you need the PROCESS (or SUPER) privilege. However, to kill a process you need the SUPER privilege. SQL DM for MySQL notifies you of the limitations of your permission level (that is, if you do not have full privileges to use all the options provided).

Filtering

Advanced filtering options

The display of the Threads is implemented as a SELECT - query against the embedded database. The initial query is as follows:

```
SELECT *
FROM processlist
ORDER BY time DESC;
```

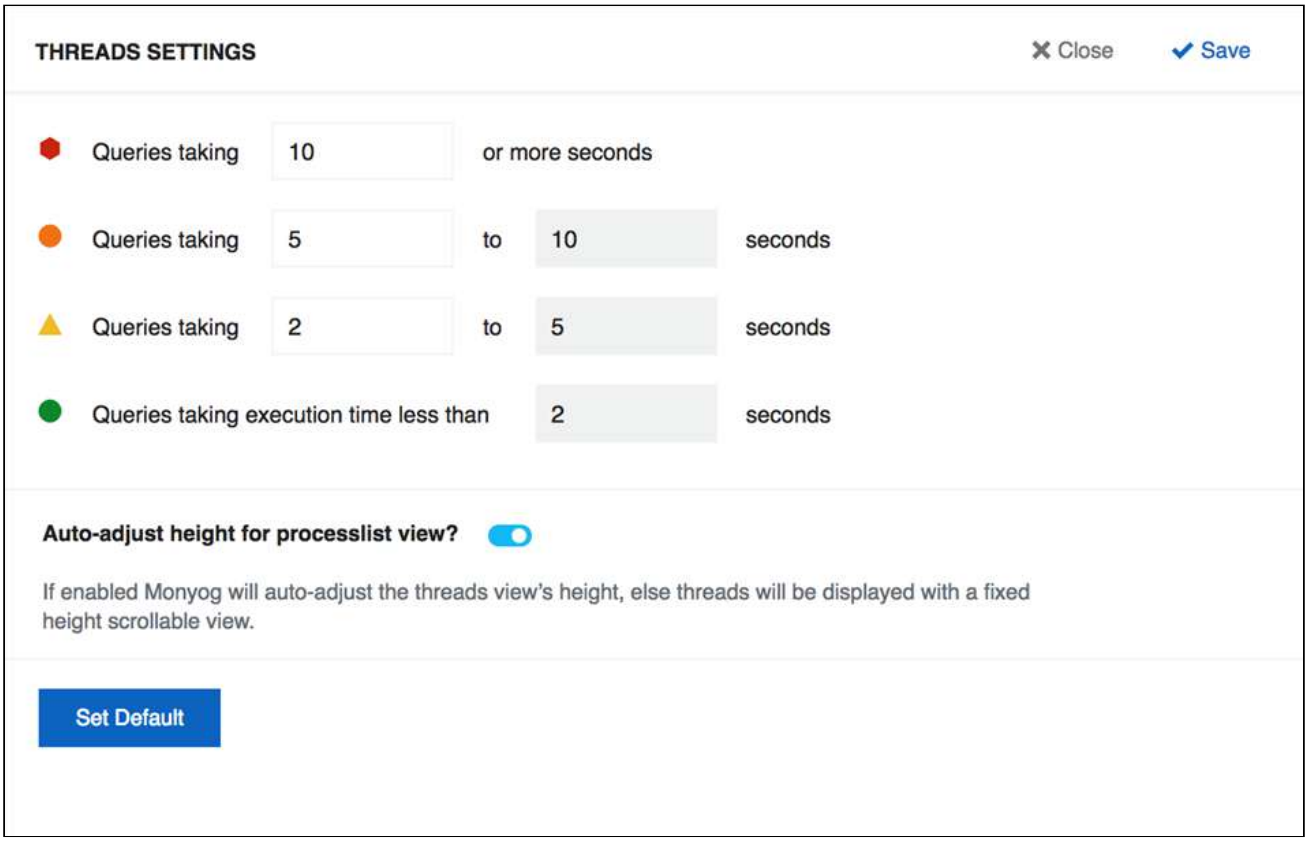
The ORDER BY clause, here, will display the slowest queries at the top (which is convenient for further investigation). You can define your own sort and filter criteria by editing this query using the WHERE, ORDER BY, and even GROUP BY clauses and AGGREGATES as you prefer. To change the query simply click on the 'Settings' icon.

For example a user query is as follows:

```
SELECT user, count(user) as no_of_queries, max(time) as running_longest
FROM [processlist]
WHERE ((user <> 'Monyog_user') and (command <> 'Sleep'))
GROUP BY user;
```

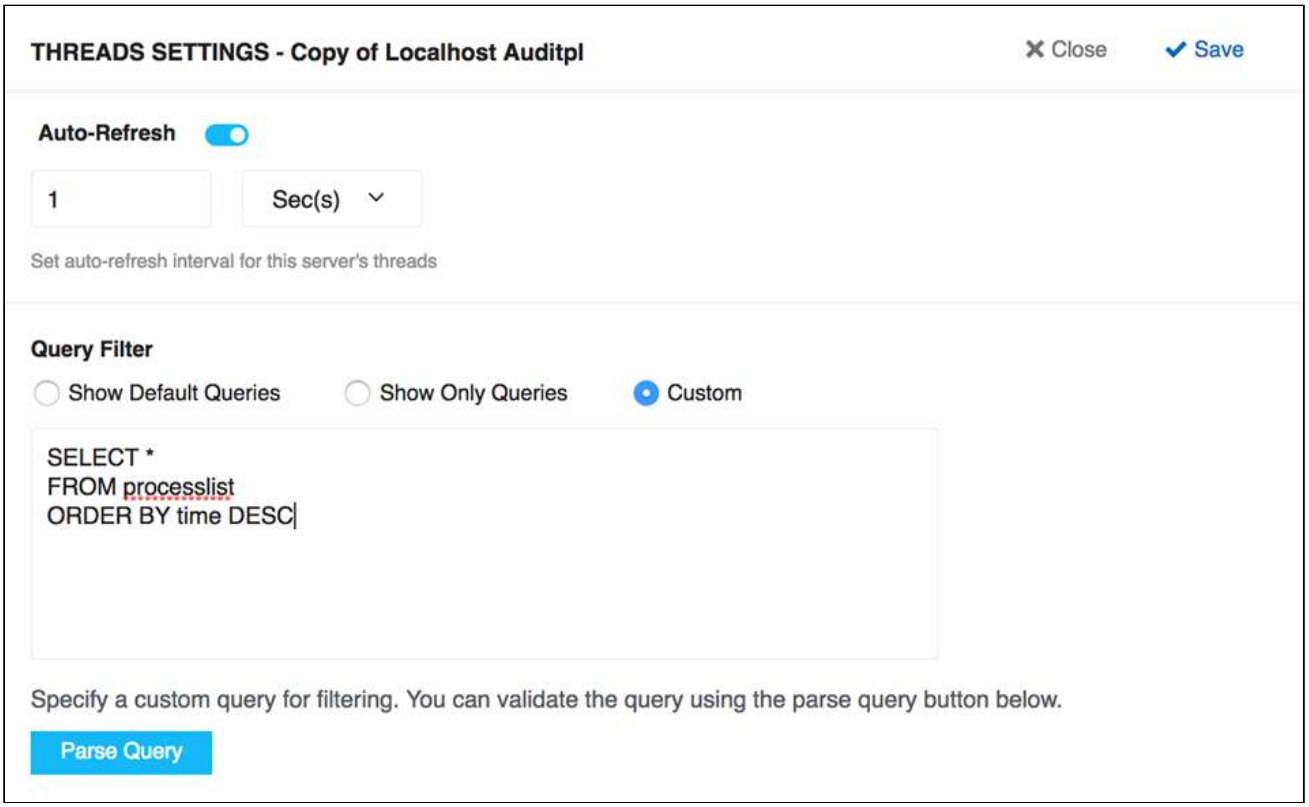
Note that string comparison is case-sensitive in SQLite and also note the use of [square brackets] in SQLite similar to backquotes in MySQL.

Notice that implementing support for the SHOW FULL PROCESSLIST command using a SELECT statement gives you much more flexibility over the output than the command does by itself.



Customize Threads

SQL DM for MySQL provides a novel and unique way to customize its features. For instance, you can write your own query to filter the output of the processlist. You can issue a SELECT query on this table to filter your Processlist display. This gives you unprecedented flexibility to filter data. The table name is 'processlist' and the column names are Id, User, Host, Db, Command, Time, State, Info, and Action.

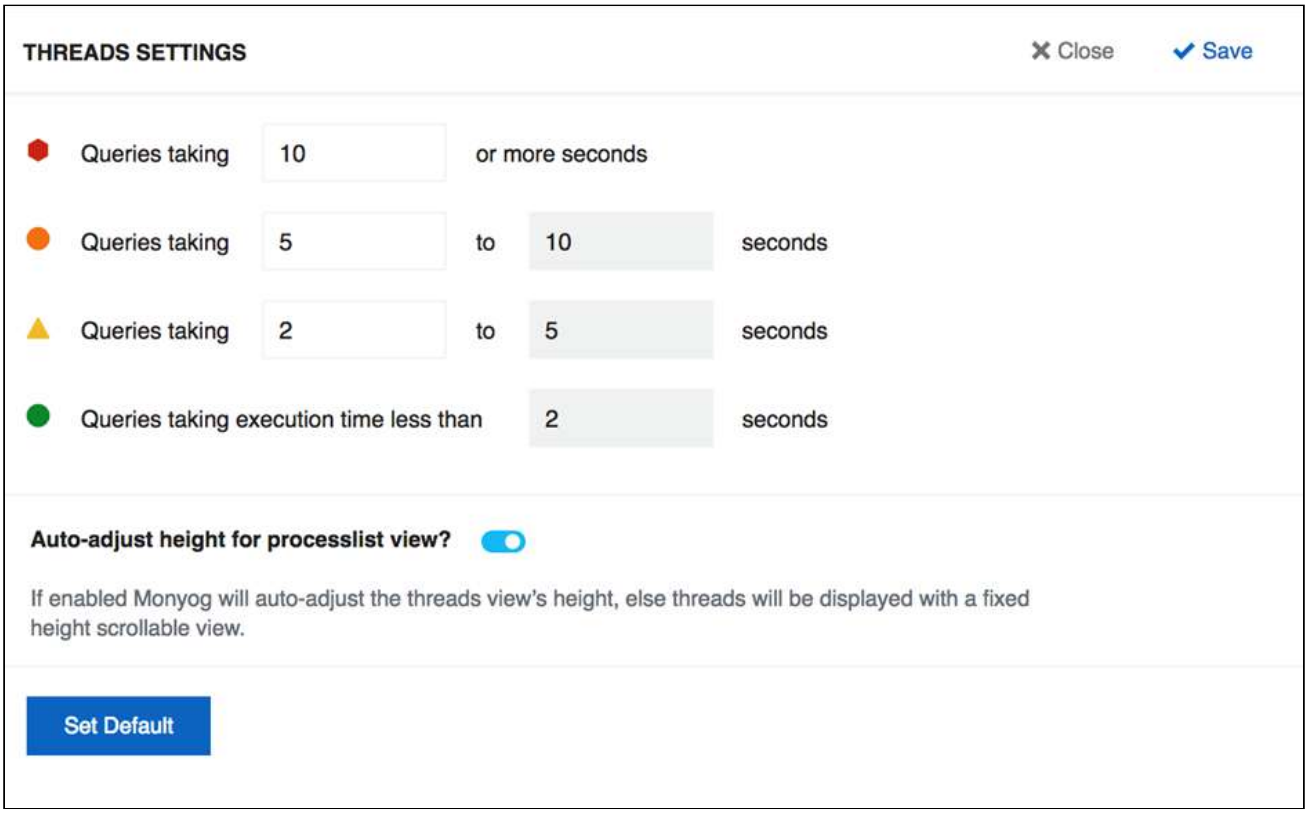


Threads Settings

Using the Threads option, you can customize the timeframe for the currently running queries based on their execution time.

How To Change?

Users can change the Threads option by clicking the Threads Settings icon. A new window displays where we can set the time for currently running queries in decreasing order.



An option is given to select the view for the processlist window. You can either select to have a fixed height or dynamic (full screen) height for your processlist window.

Real-Time

Real-Time continuously executes a bunch of queries on a server and fetches information on the top queries, tables, databases, user, hosts, queries that are locked, queries that are locking, etc. Since the information is retrieved in real time, you get to see what your server is up to at any point in time. You have to start **Recording** a session in order to see your server activity. This action can be stopped and saved for later analysis.

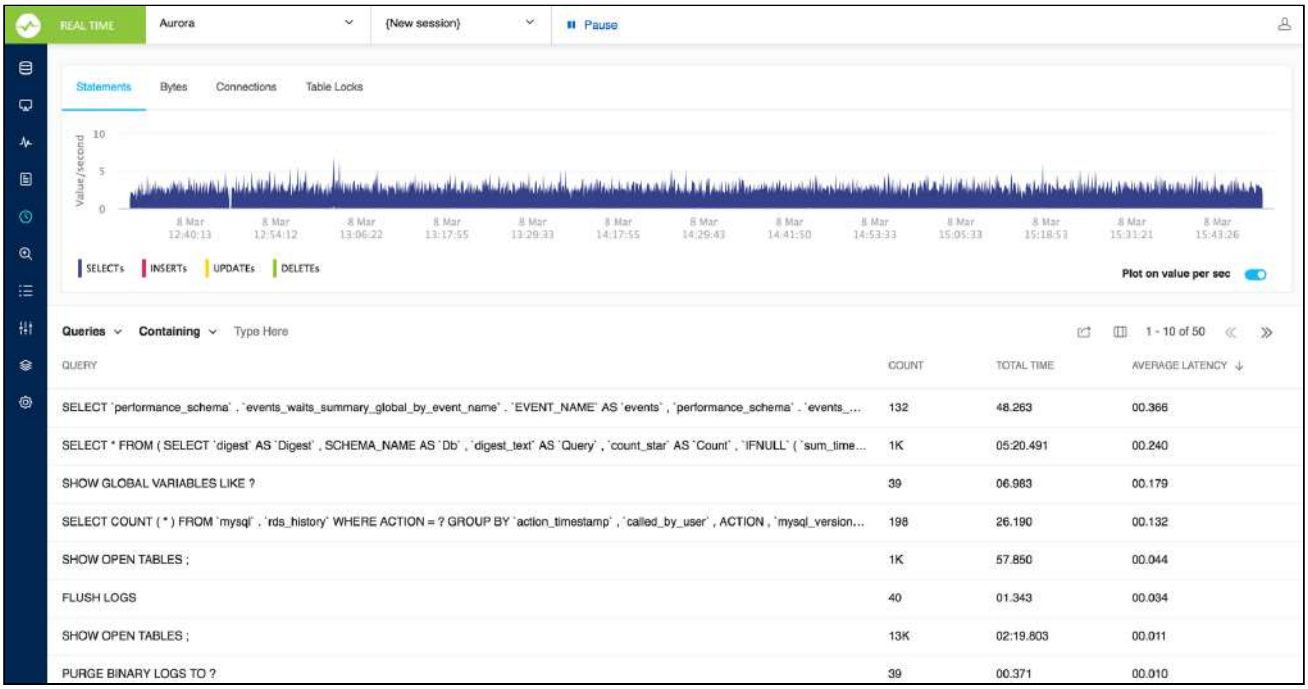
Real-Time is what you need to use if you do not have Slow or General query log or sniffer, and still want to know what is happening on your server in real time.

Real-Time collection mode

Real-Time monitoring can be done in two modes: Processlist and Performance Schema.

- Processlist mode, SQL DM for MySQL executes the query `SHOW FULL PROCESSLIST` every second to fetch the queries. Performance schema based Real-time makes use of the performance_schema database of the MySQL server. Performance schema database logs each and every query in its table. SQL DM for MySQL queries the performance_schema and retrieves the queries, including the short-lived ones.

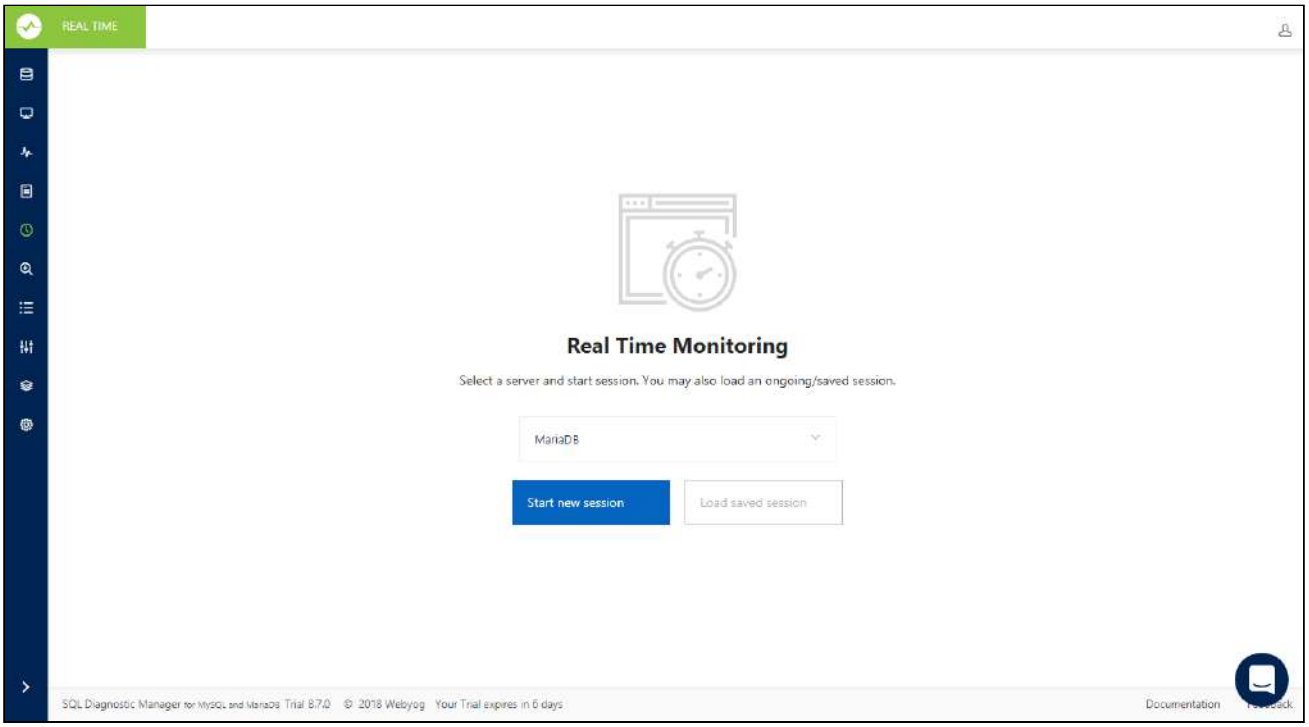
- Performance schema based Real-time, you can get extra information like Number of full table scans done by the query, Success, Error, and Warning count for each query.



Using Real-Time

You can pause recording and save a session for later analysis. The charts are zoomable.

To start using Real-Time, select a server from the dropdown, choose a saved session, or select **Start new session.**



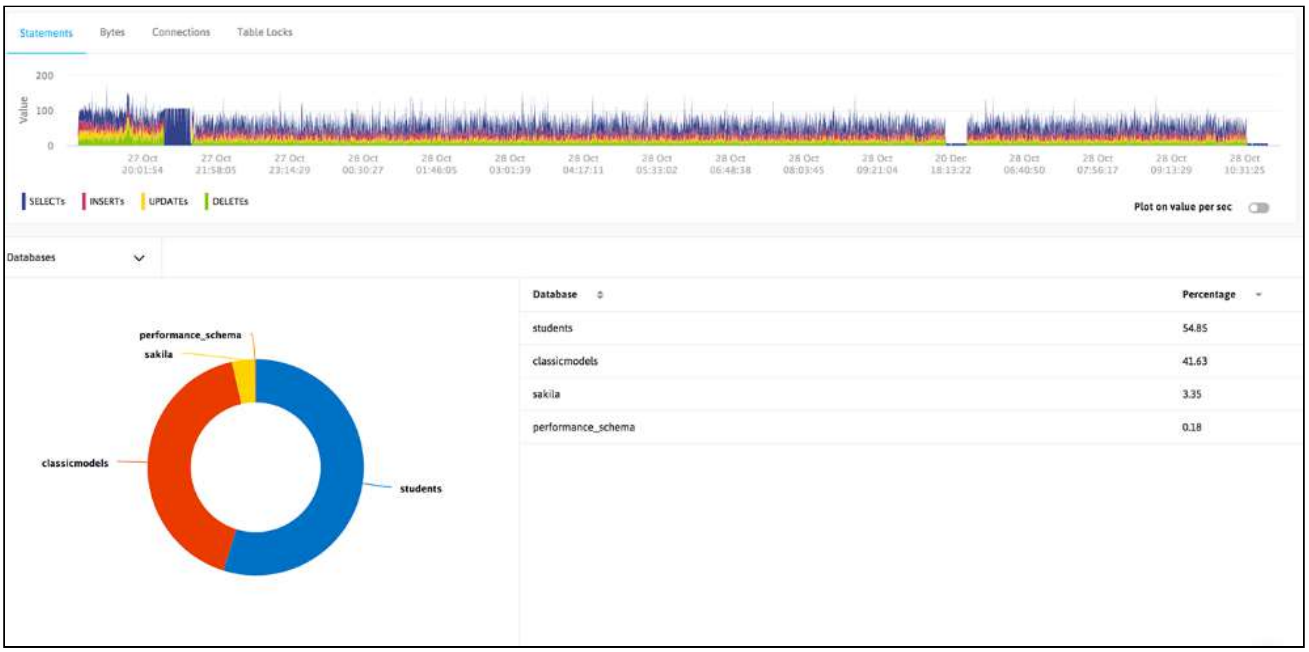
SQL DM for MySQL stores Real-Time data gathered in a session in SQLite. You can find them in the `data` folders under `realtimedata`. Real-Time data collection for the server stops and save after three days if no activity happens inside that session.

Query	Count	Total Time	Average Latency	Full Table Scan	Success Count
SHOW GLOBAL VARIABLES	1	00.002	00.002	1	1
SHOW GLOBAL STATUS	1	00.001	00.001	1	1
SHOW OPEN TABLES ;	40	00.053	00.001	40	40
SELECT 'UNIX_TIMESTAMP' (DATE_SUB (NOW ( ) , INTERVAL ? SQL_TSI_SECOND)) AS 'starttime'	1	00.001	00.001	0	1
SELECT * FROM (SELECT 'digest' AS 'Digest' , SCHEMA_NAME AS 'Db' , 'digest_text' AS 'Query' , 'count_star' AS 'Count' , 'IFNULL' (.	4	00.004	0	4	4
SHOW OPEN TABLES ;	188	00.111	0	188	188
SHOW OPEN TABLES ;	40	00.022	0	40	40
SELECT COUNT (*) AS 'count' FROM 'performance_schema' . 'events_statements_summary_by_digest'	4	00.001	0	4	4
SHOW OPEN TABLES ;	240	00.074	0	240	240
SELECT * FROM 'mysql' . 'user'	1	0	0	1	1

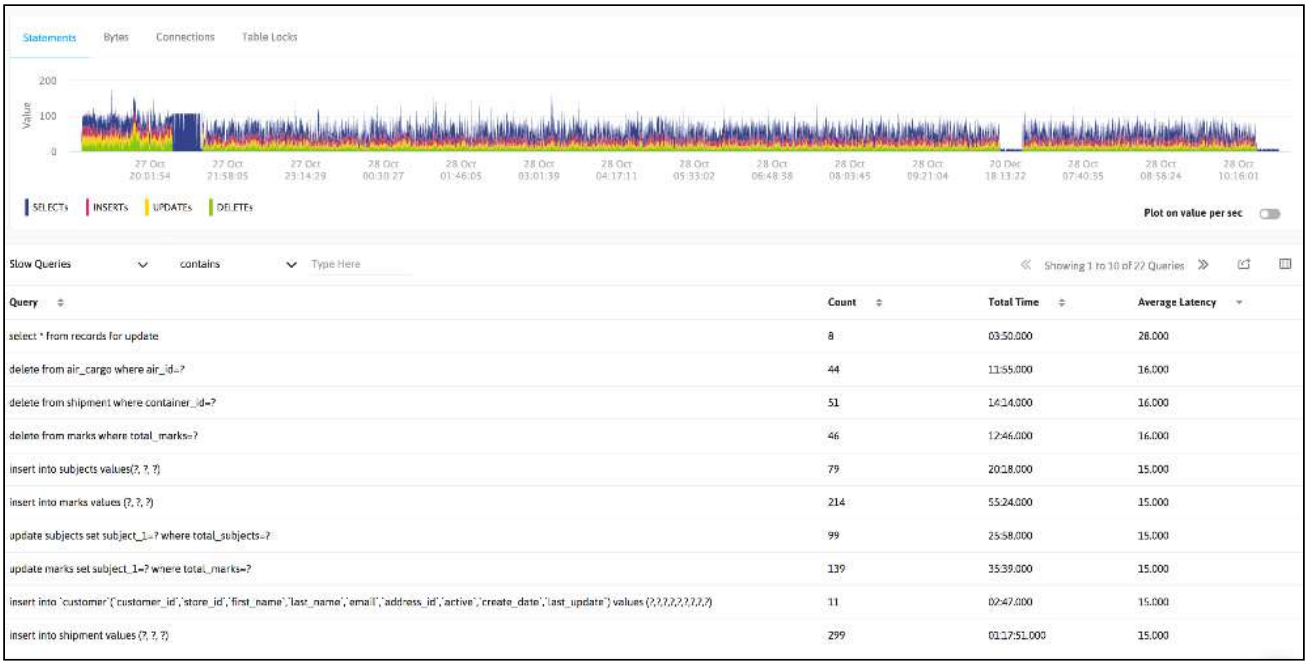
Real-Time gives you details of your server activity.

The Queries tab gives you an aggregated view of the queries that are executed in a session. This information is obtained from the `SHOW FULL PROCESSLIST`. You get additional information like execution time, user/host information, and the total count for each query. You do not have to enable general log to retrieve this information.

The Tables and Databases tabs detail information retrieved using `SHOW OPEN TABLES`:

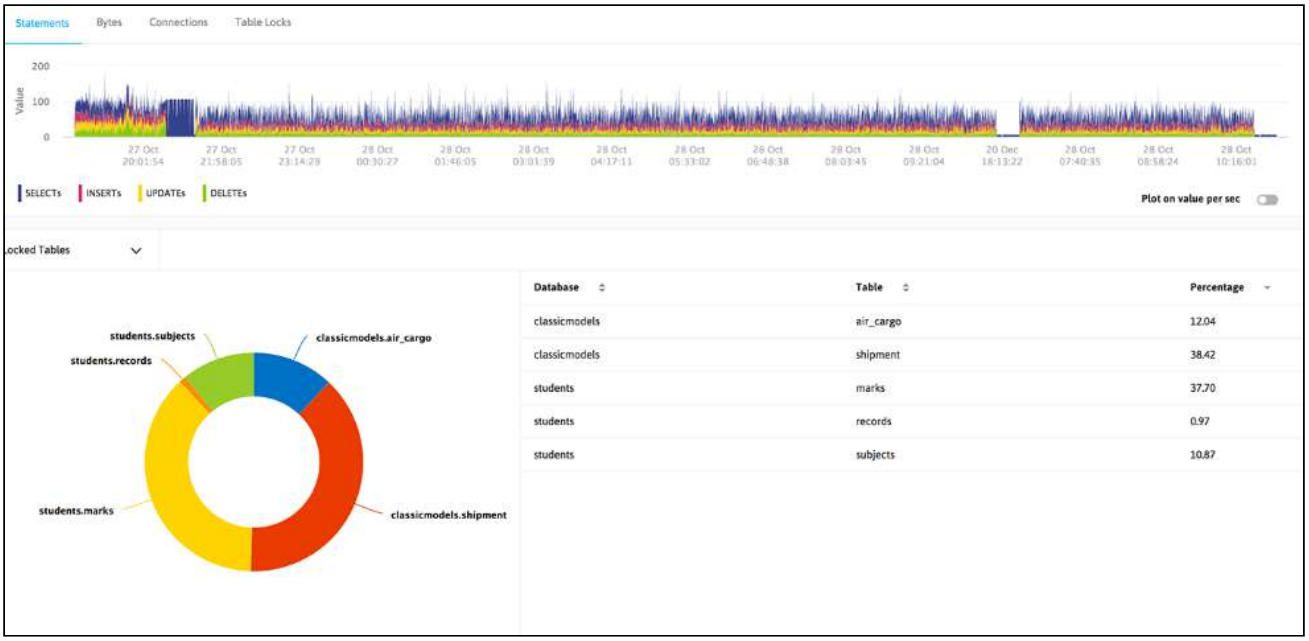


The Hosts, Users, and Slow queries are again fetched from the `SHOW FULL PROCESSLIST` and aggregated information is displayed. There is an option to change the slow query time of Real-Time profiler. On selecting Slow queries, Real-Time considers the queries taking more than the specified time as slow queries.



The Locked queries, Locking queries, and Locked tables tabs are fetched from InnoDB transaction tables.

For MySQL versions above 5.1, InnoDB plugin has to installed to retrieve locked and locking query information. But for 5.5 onwards MySQL comes with InnoDB storage engine. Refer Innodb Introduction (see page 132) for more details.



Query Analyzer

The Query Analyzer feature of SQL DM for MySQL helps you identify problem SQL. SQL DM for MySQL can find problem SQL by one or more of the following methods:

- Taking `SHOW PROCESSLIST` snapshots at regular intervals (using SQL DM for MySQL Sniffer)
- Using MySQL Proxy to collect profiling data (using SQL DM for MySQL Sniffer)
- Utilising Performance schema tables (`events_statements_summary_by_digest`, `events_statements_history_long`)
- Parsing Slow Query Log and General Query Log (using SQL DM for MySQL Log Parser)

There are several advantages and disadvantages of each approach.

`SHOW PROCESSLIST` is available in all MySQL versions and it is the easiest to setup. However, taking a snapshot of `SHOW PROCESSLIST` does not guarantee that all queries are captured. Many short-lived queries can be missed between two successive snapshots. It is a quick and easy way to find long running queries.

Log parsing requires some additional setup. Also, switching on the General Query Log puts a significant amount of load on the server. You should always keep the Slow Query Log switched on. Parsing the Slow Query Log is an effective way to find bad queries.

Using MySQL Proxy gives you the most accurate information on profiling SQL. However, during profiling you have to configure your clients to connect to MySQL Proxy, which in turn connects to MySQL server. Using MySQL Proxy

ensures that all queries are profiled. It helps you to find problematic queries that don not take much time, but are executed thousands of times. Eliminating such queries can significantly improve the performance of your application.

Performance schema based sniffer makes use of the performance_schema database of the MySQL server. SQL DM for MySQL queries the performance schema database and collects snapshots at regular interval. Since each and every query is logged in the Performance schema database, SQL DM for MySQL displays even the short lived queries using this method.

To use the SQL DM for MySQL Query Analyzer functionality for a specific server, the server General query log or Slow query log details must be configured in **Connection Settings** or a **Query Sniffer** must be enabled for that server.

Using the above tools to find problem SQL is almost always a post-mortem excercise. In certain situations you may want real-time notifications for long-running queries. SQL DM for MySQL can continuously monitor queries in real-time and send notifications (on mail or SNMP) for queries that take more than a specified amount of time to execute. You can also specify an option to kill such queries instantly.

> ⚠ The SQL DM for MySQL Sniffer taking snapshots of `SHOW PROCESSLIST` is different from the the Processlist feature in that the Sniffer retains the information retrieved in a database for generation of reports and further analysis, whereas the Processlist feature just displays that information as is, without manipulating or storing it.

Different Types of Logs Supported

MySQL query log settings

SQL DM for MySQL retrieves (completely or partially) the General query log and the Slow query log from the MySQL servers it connects to, and analyzes them. Here, you see how to set up details for the connection, so that log analysis are available with SQL DM for MySQL. You have to set up details for the general query log and the slow query log independently. Enabling slow query log 'log queries not using indexes' instead needs `SUPER` privilege. Refer to the MySQL Documentation(see page 136) on how to enable and configure logging. MySQL server logs can be written to files on the server machine or to tables in the MySQL database itself.

The MySQL server (since version 5.0) has an option to log (in the slow log) queries that do not use an index. Such queries need not be slow if there are only a few hundred or few thousand records in the table(s) involved. But they are 'potentially slow' and should be identified if they access tables, which continue to grow. You can enable and disable it, as well (SQL DM for MySQL sends the appropriate `SET` of statements to MySQL).

> ⚠ Only DML and DDL queries are recorded in the slow query log.

Logs written to files

First, consider the situation where server logs are stored as files on the server machine. This is the most common situation and the only one available with MySQL servers before version 5.1.

If it is the first time you configure a server with this option, click the **Fetch query log details** button. MySQL server detects (it is stored in server variables) what logs are enabled and how logging is configured. Click **Test Path** to verify the path. SQL DM for MySQL connects and verify the existence of the file (but not its content).

The log files can be accessed from the local file system (if SQL DM for MySQL and MySQL is running on the same computer) or by using SFTP (if SQL DM for MySQL and MySQL is running on different computers). Note that you must use the file and path syntax of the machine where the logs are.

If the log files can be accessed from a shared drive, over a network, or from a network enabled file system (like NFS on Linux), then SQL DM for MySQL can access them as if they were local files. No additional SSH/SFTP configuration is required in this case: the operating system takes care of the file transfer transparently.

When **Via SFTP** option is chosen, then SSH server details as defined in SSH server details settings are used to read the file from the remote system.

The SSH user must have read access to the log files.

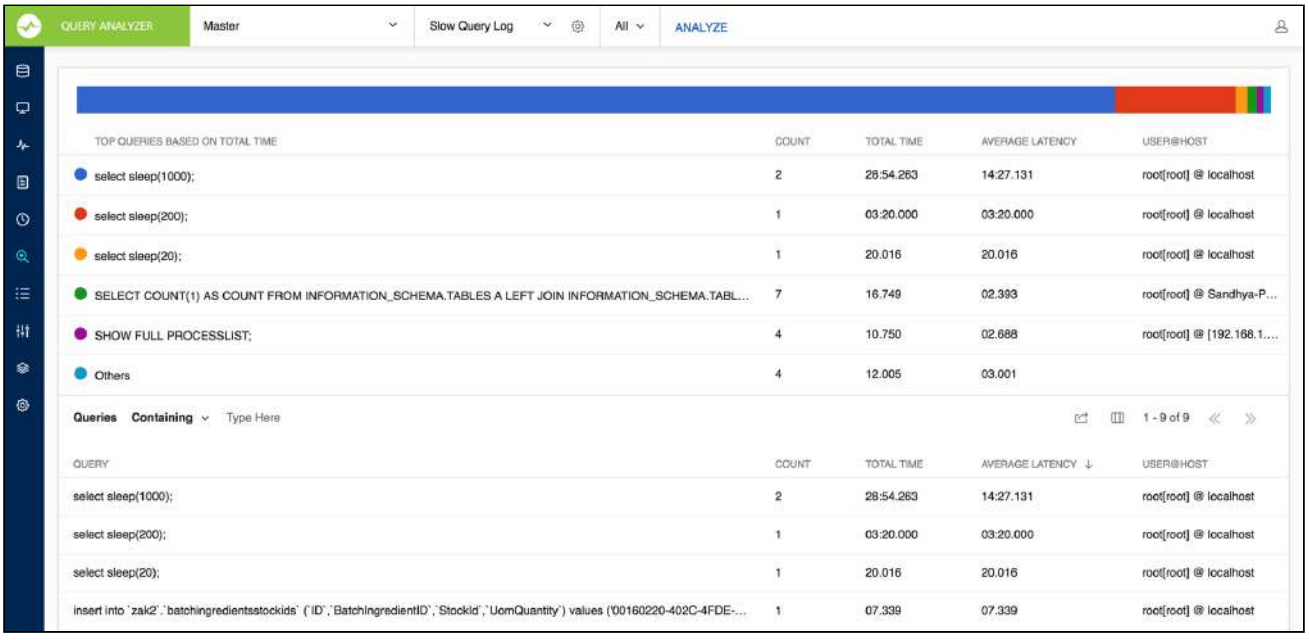
If MySQL server version is greater than 5.1.6 then all the fields mentioned in log analyzer would be editable i.e. if a user changes and saves the settings by clicking **Save** a pop up is displayed where a user can set the new value to corresponding MySQL Server.

By default, MONyog(SQL DM for MySQL) service runs under Local System Account. If you have Slow query or General query logs in a Mapped Network Drive, SQL DM for MySQL is not able to reach it. You need to use UNC notation for SQL DM for MySQL to be able to access them. See FAQ 31(see page 226) for details.

Logs written to MySQL tables

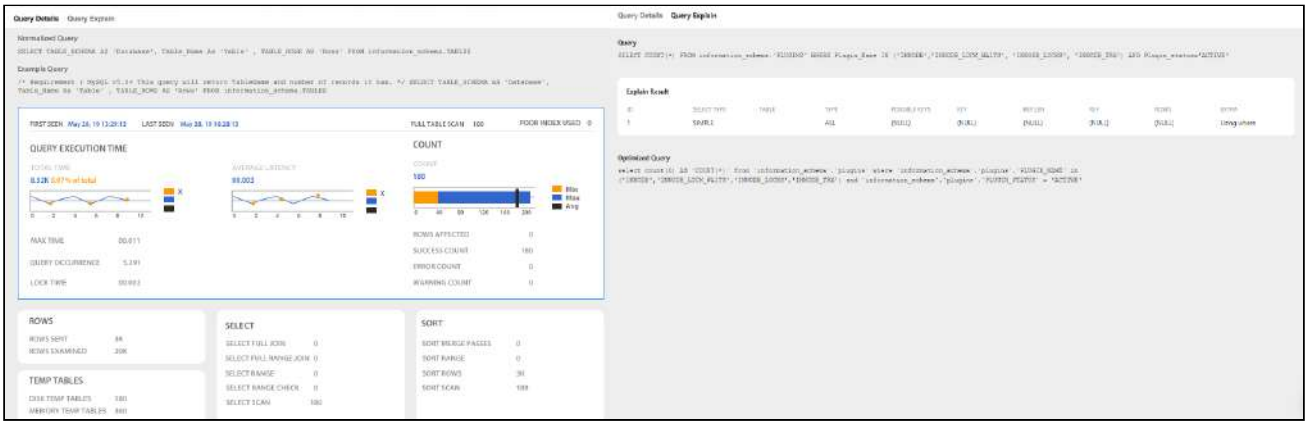
It is supported by MySQL from version 5.1. Also, SQL DM for MySQL supports when this option is available. Here, click the **Fetch Log Details From MySQL** button. When this option is used there is no file path to configure and no SSH details to consider. SQL DM for MySQL can retrieve the server log by sending simple `SELECT` statements. Only the MySQL user used by SQL DM for MySQL to connect to MySQL must have `SELECT` privileges to the tables.

In the Query Analyzer tab select the MySQL server, the type of log (including the 'pseudo log') you want to analyze (Slow Query Log, General Query Log, or Sniffer(see page 140)), and click **Analyze** to start the analysis**:**



SQL DM for MySQL includes an improved Top Query Overview to ease the query performance overtime analysis.

After getting the results of your Analysis, click a query to obtain the Query Details and Query Explain tabs with detailed information about the selected query:



Explain plan is available in Query analyzer for Slow_log table based logging, Processlist based sniffer and performance schema based sniffer.

With the General Query Log there are few specific problems:

1. With multi-line queries only record the first line of the statement. The reason is that, as the log does not record the statement DELIMITERs, there is no way to tell where a multi-line statement ends. Even the option to 'Show full' is not displayed more than one line as SQL DM for MySQL has only stored one line. Refer to FAQ 23(see page 224).
2. It is not always possible to tell what user executed a specific query. When this is the case the User column displays empty in the Query Analyzer output. It is not a bug in SQL DM for MySQL but a limitation with the general log itself.

You can sort the display by clicking on the column header. Note that statement grouping/counting and sorting is case insensitive.

Filter settings

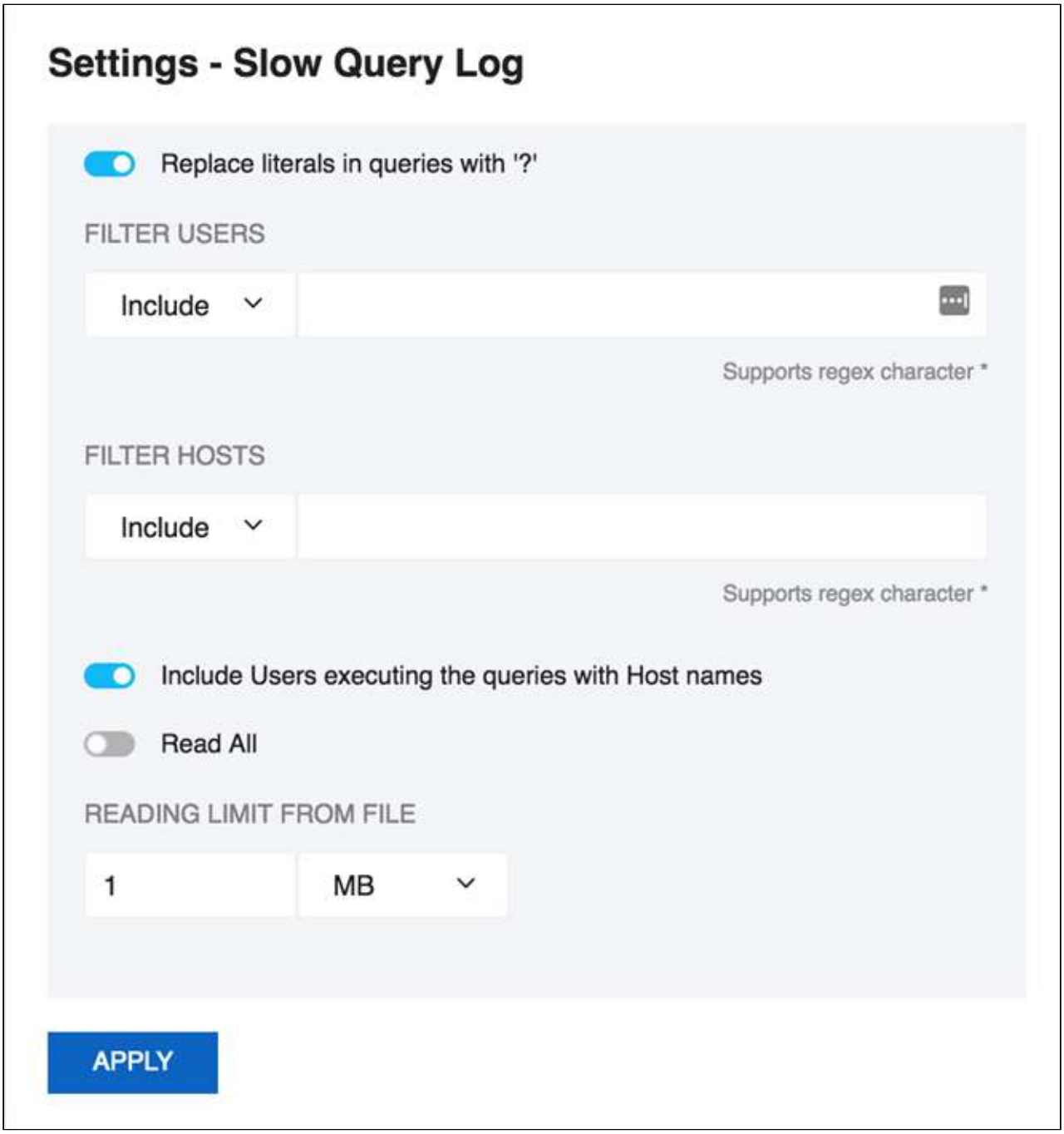
There is an option to **Replace literals from the query.** The purpose of this option is to eliminate small differences between almost identical queries. Currently, quoted strings and numbers are replaced with the dummy string '?' only. The filtering settings are stored for that particular session which is not permanent.

For example:

```
SELECT * FROM customer_master
WHERE cust_id = 23 AND address = 'r;#23 fleet street';
```

becomes,

```
SELECT * FROM customer_master
WHERE cust_id = ? AND address = ?;
```



The reading limit **All** is selected it considers the whole file for analyzing but if the option is **Last**, it reads the last specified chunk in KB, MB, or bytes out of the whole log file. Also, you can define a timeframe to be analyzed and the size of the 'log chunk' (in KB, MB and Bytes for file based logs and in rows for table-based logs) to be transferred to SQL DM for MySQL.

If **All** is selected in the list, is not considering any timeframe and just displays all queries within the specified size/ chunk. Also note, it is the smallest of those two settings that have effect for the analysis. For analyzing the sniffer pseudo log there is no chunk size to be defined as the complete pseudo log as stored in the SQL DM for MySQL

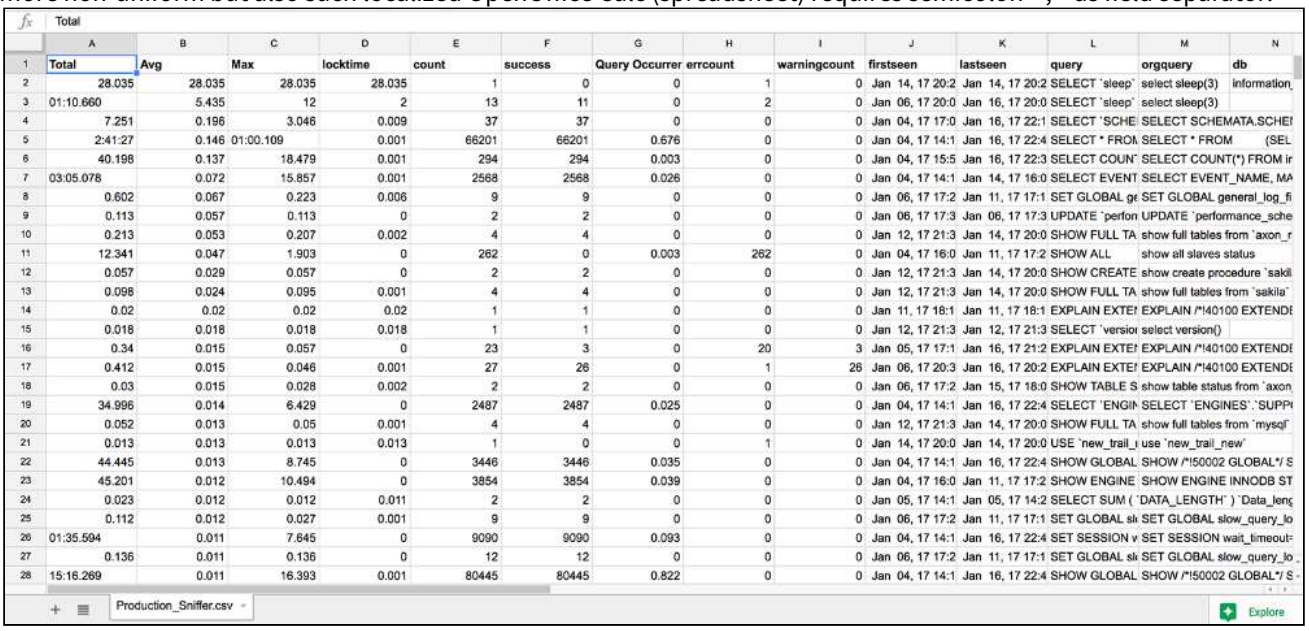
database that is considered. The selected log chunk needs to have statements for the selected period. If not, then SQL DM for MySQL of course only display data from the first log record available.

Include User and Host Information: If this option is selected it displays the User and Host of that particular query and it groups the query analyzer table based on user@host[35] and query.

> If SQL DM for MySQL is already installed in the machine and Sniffer is enabled, it only displays the User info in Query Analyzer table because old SQL DM for MySQL never used to store the Host information in sniffer.data table. Also note, that this option is not supported by MySQL Proxy. In General query log, if connect string is not included in the specified chunk, it is not display the user@host[36] information which is just left as an empty space.

Export As CSV

The option to define the field delimiter is provided because some localized Windows programs for LOCALEs where comma " , " is used as a decimal sign and requires a semicolon " ; " as field separator. This includes Microsoft Office programs (Excel and Access) and Microsoft text-ODBC driver on such localized Windows. On Linux, the situation is more non-uniform but also such localized OpenOffice Calc (spreadsheet) requires semicolon " ; " as field separator.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Total	Avg	Max	locktime	count	success	Query Occurrer	errcount	warningcount	firstseen	lastseen	query	orgquery	db
2	28.035	28.035	28.035	28.035	1	0	0	1	0	Jan 14, 17 20:2	Jan 14, 17 20:2	SELECT `sleep`	select sleep(3)	information_
3	01:10.660	5.435	12	2	13	11	0	2	0	Jan 06, 17 20:0	Jan 16, 17 20:0	SELECT `sleep`	select sleep(3)	
4	7.251	0.196	3.046	0.009	37	37	0	0	0	Jan 04, 17 17:0	Jan 16, 17 22:1	SELECT `SCHE	SELECT SCHEMATA,SCHEI	
5	2:41:27	0.146	01:00.109	0.001	66201	66201	0.676	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SELECT * FRO	SELECT * FROM (SEL	
6	40.198	0.137	18.479	0.001	294	294	0.003	0	0	Jan 04, 17 15:5	Jan 16, 17 22:3	SELECT COUN	SELECT COUNT(*) FROM ir	
7	03:05.078	0.072	15.857	0.001	2568	2568	0.026	0	0	Jan 04, 17 14:1	Jan 14, 17 16:0	SELECT EVENT	SELECT EVENT_NAME, MA	
8	0.602	0.067	0.223	0.006	9	9	0	0	0	Jan 06, 17 17:2	Jan 11, 17 17:1	SET GLOBAL ge	SET GLOBAL general_log_fi	
9	0.113	0.057	0.113	0	2	2	0	0	0	Jan 06, 17 17:3	Jan 06, 17 17:3	UPDATE `perfon	UPDATE `performance_sche	
10	0.213	0.053	0.207	0.002	4	4	0	0	0	Jan 12, 17 21:3	Jan 14, 17 20:0	SHOW FULL TA	show full tables from `axon_r	
11	12.341	0.047	1.903	0	262	0	0.003	262	0	Jan 04, 17 16:0	Jan 11, 17 17:2	SHOW ALL	show all slaves status	
12	0.057	0.029	0.057	0	2	2	0	0	0	Jan 12, 17 21:3	Jan 14, 17 20:0	SHOW CREATE	show create procedure `sakil	
13	0.098	0.024	0.095	0.001	4	4	0	0	0	Jan 12, 17 21:3	Jan 14, 17 20:0	SHOW FULL TA	show full tables from `sakila`	
14	0.02	0.02	0.02	0.02	1	1	0	0	0	Jan 11, 17 18:1	Jan 11, 17 18:1	EXPLAIN EXTE	EXPLAIN /*!40100 EXTENDE	
15	0.018	0.018	0.018	0.018	1	1	0	0	0	Jan 12, 17 21:3	Jan 12, 17 21:3	SELECT `versio	select version()	
16	0.34	0.015	0.057	0	23	3	0	20	3	Jan 05, 17 17:1	Jan 16, 17 21:2	EXPLAIN EXTE	EXPLAIN /*!40100 EXTENDE	
17	0.412	0.015	0.046	0.001	27	26	0	1	26	Jan 06, 17 20:3	Jan 16, 17 20:2	EXPLAIN EXTE	EXPLAIN /*!40100 EXTENDE	
18	0.03	0.015	0.028	0.002	2	2	0	0	0	Jan 06, 17 17:2	Jan 15, 17 18:0	SHOW TABLE S	show table status from `axon	
19	34.996	0.014	6.429	0	2487	2487	0.025	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SELECT `ENGIN	SELECT `ENGINES`.`SUPP	
20	0.052	0.013	0.05	0.001	4	4	0	0	0	Jan 12, 17 21:3	Jan 14, 17 20:0	SHOW FULL TA	show full tables from `mysql`	
21	0.013	0.013	0.013	0.013	1	0	0	1	0	Jan 14, 17 20:0	Jan 14, 17 20:0	USE `new_trail_	use `new_trail_new`	
22	44.445	0.013	8.745	0	3446	3446	0.035	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SHOW GLOBAL	SHOW /*!50002 GLOBAL*/ S	
23	45.201	0.012	10.494	0	3854	3854	0.039	0	0	Jan 04, 17 16:0	Jan 11, 17 17:2	SHOW ENGINE	SHOW ENGINE INNODB ST	
24	0.023	0.012	0.012	0.011	2	2	0	0	0	Jan 05, 17 14:1	Jan 05, 17 14:2	SELECT SUM (	`DATA_LENGTH`) `Data_leng	
25	0.112	0.012	0.027	0.001	9	9	0	0	0	Jan 06, 17 17:2	Jan 11, 17 17:1	SET GLOBAL sl	SET GLOBAL slow_query_lo	
26	01:35.594	0.011	7.645	0	9090	9090	0.093	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SET SESSION v	SET SESSION wait_timeout=	
27	0.136	0.011	0.136	0	12	12	0	0	0	Jan 06, 17 17:2	Jan 11, 17 17:1	SET GLOBAL sl	SET GLOBAL slow_query_lo	
28	15:16.269	0.011	16.393	0.001	80445	80445	0.822	0	0	Jan 04, 17 14:1	Jan 16, 17 22:4	SHOW GLOBAL	SHOW /*!50002 GLOBAL*/ S	

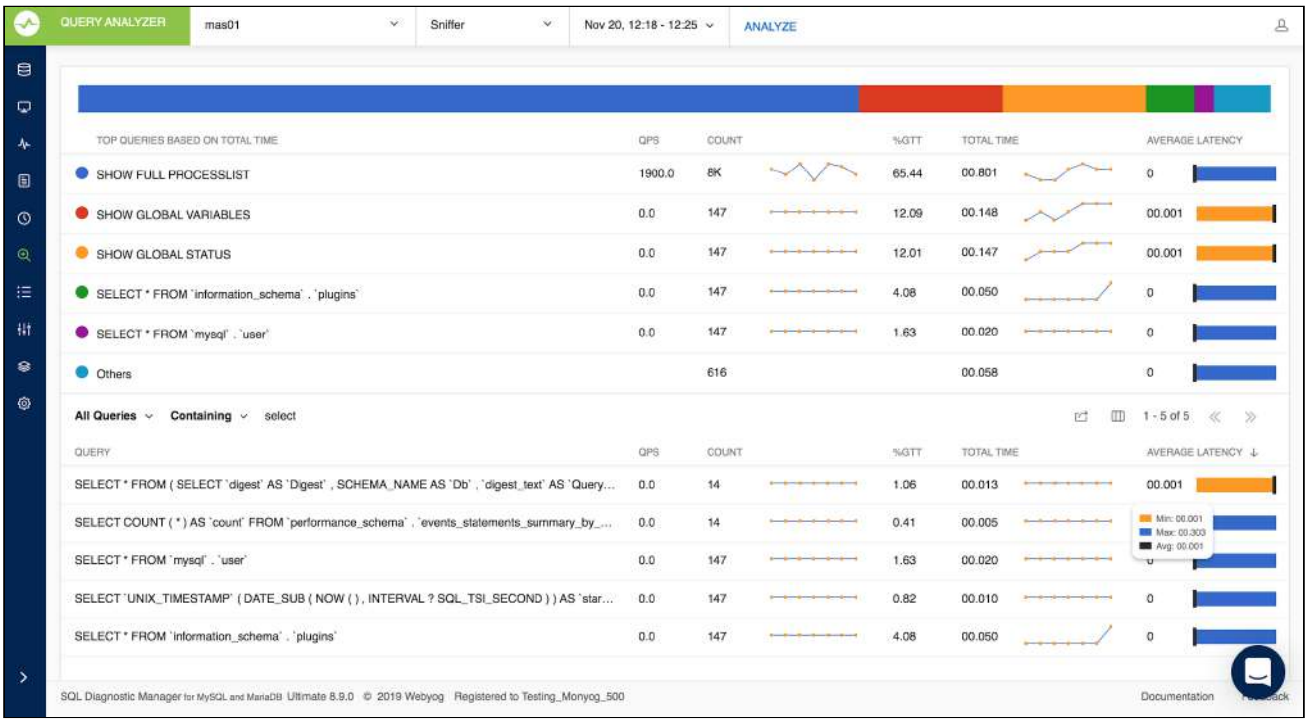
Production_Sniffer.csv

Sniffer

SQL DM for MySQL Query Sniffer is a functionality that records a pseudo server log and stores it in the SQL DM for MySQL embedded database. With **Query sniffer** enabled, SQL DM for MySQL can populate the pseudo server log in three different ways at the intervals you specify:

35 mailto:user@host
36 mailto:user@host

- By utilizing Performance Schema tables (`events_statements_summary_by_digest`, `events_statements_history_long`) and collecting snapshots at regular intervals.
- By sending the query `SHOW FULL PROCESSLIST` to the MySQL server.
- Or, by connecting to a running instance of the MySQL-Proxy program that is used by one or more clients to connect to a MySQL server.



For MySQL 5.6.14 and above you can use Performance schema, Proxy, and Processlist for query analysis. If using MySQL version less than 5.6.14 then only Proxy or Processlist can be used in SQL DM for MySQL.

Performance Schema on MySQL contains queries executed on server along with other information:

- Number of rows sent and examined
- Number of temporary tables created on disk
- Number of temporary tables created on memory
- Number of joins performed and the type of join
- Whether sorting happened and the type of sort
- Whether index used
- Whether good index used

SQL DM for MySQL uses performance schema statement digest `table(events_statements_summary_by_digest)` to get the above information and is dependent on the statements_digest in setup_consumers table. By default, this option is enabled, you can disable it by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'statements_digest';
```

Example query is available in `events_statements_history_long` table and has to be enabled and is dependent on the `events_statements_history_long` in setup_consumers table. By default, this is not enabled and should be enabled by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'events_statements_history_long';
```

The `performance_schema.events_statements_summary_by_digest` table size is dependent on `performance_schema_digests_size` `global` variable. By default, this size is set to 5000 rows. Once it reaches this limit you may lose the queries. SQL DM for MySQL provides an option to truncate the performance schema digest table when it reaches 80% of `performance_schema_digests_size`.

Performance schema based sniffer comes with different filters like: Queries with errors, Queries with warnings, Queries with missing indexes, and Queries with poor indexes.

> ⚠ `performance_schema` truncates queries after 1024 characters and always replaces literals with a wildcard (in other words: P_S contains a summary/an aggregation only). So query listing not replacing literals is not possible with this option. And finally also observe that no other tool (or user) should be writing (including deleting or truncating) to `events_statements_summary_by_digest` and `events_statements_history_long` tables if this option is used as there is only one of each table for all users (it is not a temporary table or a materialized view or similar private for the user). This is a design limitation with the tables in P_S as such and not a Monyog issue.

If using MySQL version less than 5.6.14 then only Proxy or Processlist can be used in SQL DM for MySQL. Although, configuring a Proxy instance is a little more complicated, the PROXY-based sniffer has several advantages over the PROCESSLIST-based, including the following:

1. All queries that was handled by the Proxy are recorded by SQL DM for MySQL sniffer when PROXY option is used. When PROCESSLIST option is used very fast queries may execute completely between two SHOW FULL PROCESSLIST queries andis not recorded.
2. You can choose to analyze queries from specific client(s)/application(s) only. Simply let (only) the clients that you want to focus on at the moment connect through the Proxy.
3. When using the PROXY option you can distribute part of the load generated by the sniffer on the machine that fits best in your deployment scenario (like on the one that has most free resources available) by deciding where to have the PROXY: The MySQL machine, the SQL DM for MySQL machine (if not the same), or quite another machine. The machine running MySQL have no additional load due to the sniffer if the Proxy is not running on that machine.

Also, if more SQL DM for MySQL instances use the same PROXY they use the same data collected, when the Proxy Sniffing is enabled by the first SQL DM for MySQL instance. To work with SQL DM for MySQL sniffer the MySQL Proxy instance must be started with the name of a LUA script called **MONyog.LUA** (LUA is a scripting/programming language) as argument and is distributed with SQL DM for MySQL. You find it in the Monyog program folder after installing (Windows and Linux RPM) or unpacking (Linux .tar.gz) the SQL DM for MySQL program package as downloaded from the IDERA website. The MySQL Proxy program however you need to download from MySQL website (we cannot include it for license reasons). SQL DM for MySQL works with Proxy versions from 0.61 to 0.81(latest currently) with the exception of 0.7x versions for windows and Mac due to a bug in those specific builds. For more information on Proxy MySQL Proxy(see page 140).

To start a Proxy instance for use with SQL DM for MySQL use the command:

- **For Older version:**

```
Proxy installation folder>mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 --
proxy-address=192.168.y.y:4045 --proxy-lua-script=SQL DM for MySQL.lua
```

- **For v0.81 and later:**

```
Proxy installation folder>mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 --
proxy-address=192.168.y.y:4045 --admin-username=root --admin-password=root --admin-
lua-script=MONyog.lua --proxy-lua-script=MONyog.lua
```

(It is assumed that the 'MONyog.LUA' was copied to the folder where the PROXY binary is). Also, if no port is specified the PROXY listens on port 4040. Now, you can connect to the Proxy from one or more clients/applications. The Proxy sends queries to MySQL and results back to the client. But when started with the LUA script for SQL DM for MySQL sniffer it also send information to SQL DM for MySQL that SQL DM for MySQL uses to populate the sniffer 'pseudo log'.

Once this 'pseudo log' has been recorded (in either of the two ways described: PROCESSLIST or PROXY-based) the SQL DM for MySQL log analysis functionalities can operate on the 'pseudo log' as well as the 'real logs'. The data recorded in the pseudo log is purged automatically based on the 'data retention timeframe' option set by you.

Further some filtering options are provided. This filtering happens before storing to the SQL DM for MySQL database. It prevents the sniffer database to grow out of control. The filtering options are as follows:

- **User and host**: You can choose to store queries executed only by a specific combination of users and/or hosts.
- **Minimum time taken:** For every PROCESSLIST returned to SQL DM for MySQL, the queries are recorded in the embedded database only if they have been executing for a time greater than the specified minimum execution time. Furthermore, if a query with the same structure and details (like process ID) as one already recorded is encountered, the embedded database is UPDATED, and the statement is recorded only once. This setting should be somewhat larger than the sample interval (and also consider the latency of the connection).
- **Queries starting with:** Enter any string and only queries starting with that string is recorded. Examples: `SELECT *`, `UPDATE Customer_Base`.

Also note, in the `PROCESSLIST` Sniffer there is an option 'Long Running Query Options' where you can monitor the long running queries by notifying or killing a query which takes more than a time specified by you. You can also specify users whose queries are ignored (i.e. queries) by such user are not killed by SQL DM for MySQL and never raise an alert even if they take a longer time to execute than the alert/kill setting time you specified.

Clicking the **monitor only locked queries** would only monitor those long queries that are locked.

> ⚠ Note that the query sniffer is never a complete General Log. Very fast statements may or may not be recorded as they may or may not finish executing between two `PROCESSLIST`'s generated. The time interval between subsequent data collections for the 'pseudo log' depends on the connection to the MySQL server.

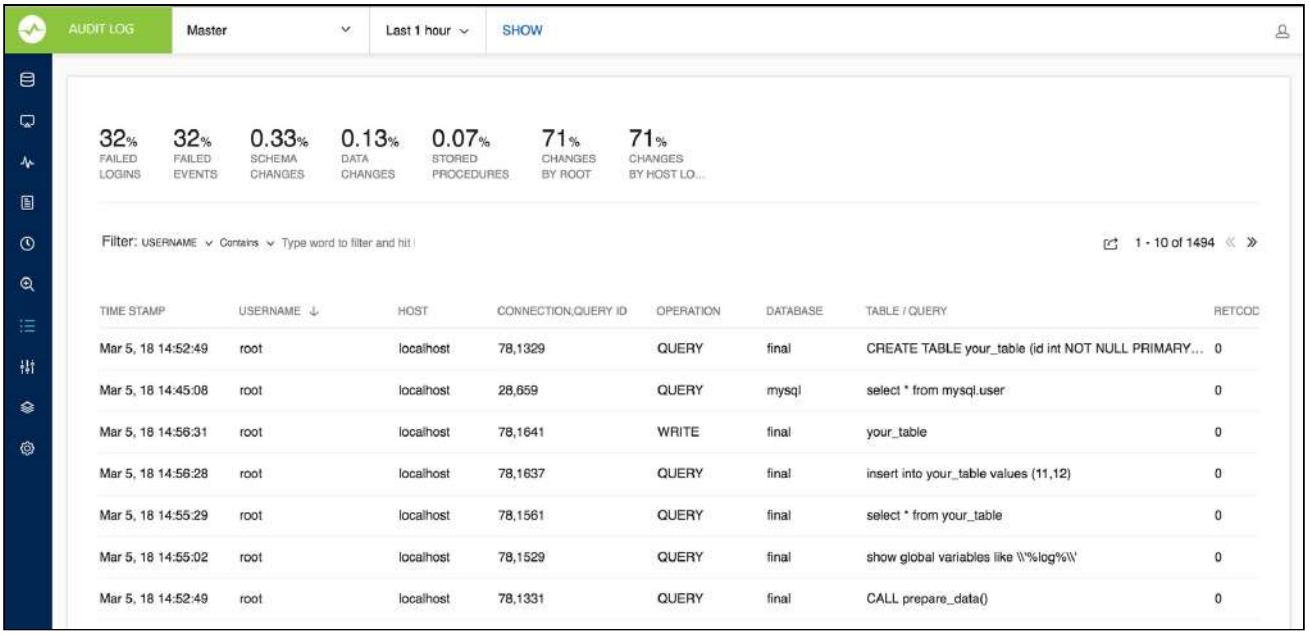
The identical queries are only listed once and the Count column tells how many times this query was executed.

Audit Log

This feature parses the audit log maintained by the server and displays the content in clean tabular format. SQL DM for MySQL supports both MySQL Enterprise and MariaDB audit logs.

SQL DM for MySQL accesses the audit log file, the same way it does for other MySQL log files: Slow Query, General Query, and Error log. Please review Advanced Settings(see page 48) to learn how to enable the audit log on your server, and configure SQL DM for MySQL.

After selecting the server and the time-frame for which you want the audit log to be seen from, click **SHOW AUDIT LOG** to get the content of the log. The limit on the number of rows which can be fetched in one time-frame is 10000 (This limit is calculated from the beginning of the audit log), so in case you have more than 10000 entries in your audit log for the selected time frame, then the remaining entries are not displayed.



The section on the top gives you quick summary of the audit log in percentage like Failed Logins, Failed Events, Schema changes, Data Changes, and Stored Procedure. All these legends are clickable and shows the corresponding audit log entries.

Furthermore, you can use the filter option to fetch audit log based on Username, Host, Operation, Database, and Table/Query, use the Contains or Does not contain options to have more specific Audit log results.

ⓘ Export the fetched audit log content in CSV format.

Server Configuration

Maintaining server configuration and tracking changes, plays a vital role in the maintenance of MySQL servers. DBAs may be responsible for hundreds of servers and keeping an eye on the configuration settings for all of them could be difficult to say the least.

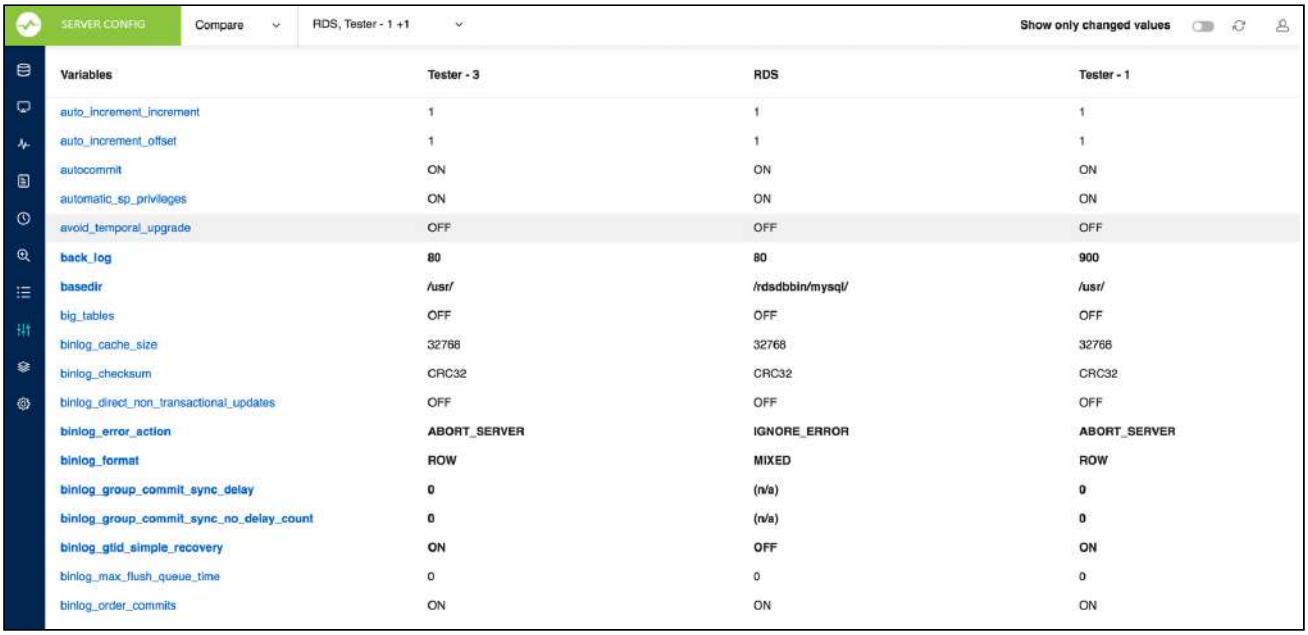
Server Config allows you to compare MySQL configurations of multiple servers side-by-side, with all changes highlighted so that differences are visually discernible at a glance.

> Wondering why server A is not performing as well as server B when they share the exact same load? The answer could lie in the configuration files.

Server Config also allows you to track changes to your server configuration files over a period of time. Now you are in full control of what goes into those files and the impact they have on your MySQL server

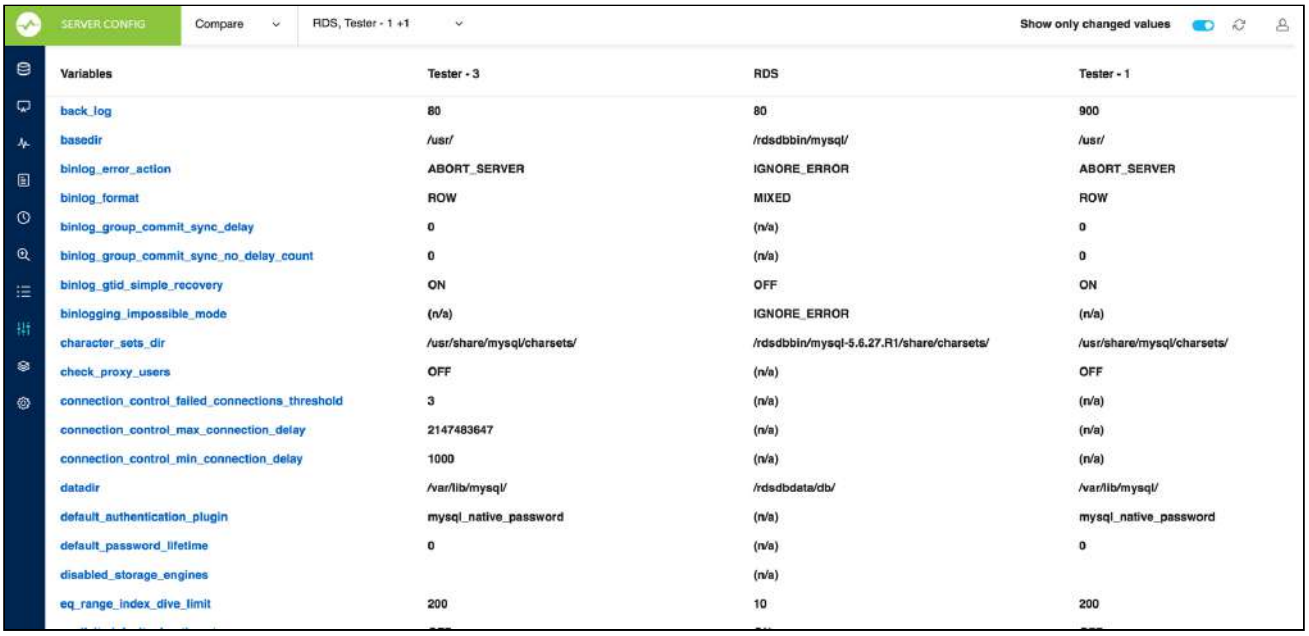
Compare Configuration

Compare Configuration allows you to compare MySQL Global Variables of multiple servers side-by-side. This comes in handy when trying to determine why one MySQL server is not performing as well as another one with similar conditions of load. Differences in the Global Variables are highlighted.



Variables	Tester - 3	RDS	Tester - 1
auto_increment_increment	1	1	1
auto_increment_offset	1	1	1
autocommit	ON	ON	ON
automatic_sp_privileges	ON	ON	ON
avoid_temporal_upgrade	OFF	OFF	OFF
back_log	**80**	**80**	**900**
basedir	**/usr/**	**/rdsdbbin/mysql/**	**/usr/**
big_tables	OFF	OFF	OFF
binlog_cache_size	32768	32768	32768
binlog_checksum	CRC32	CRC32	CRC32
binlog_direct_non_transactional_updates	OFF	OFF	OFF
binlog_error_action	**ABORT_SERVER**	**IGNORE_ERROR**	**ABORT_SERVER**
binlog_format	**ROW**	**MIXED**	**ROW**
binlog_group_commit_sync_delay	**0**	**(n/a)**	**0**
binlog_group_commit_sync_no_delay_count	**0**	**(n/a)**	**0**
binlog_gtid_simple_recovery	**ON**	**OFF**	**ON**
binlog_max_flush_queue_time	0	0	0
binlog_order_commits	ON	ON	ON

To view only changed values, enable the **Show only changed values** toggle-bar on the tool bar at the top.

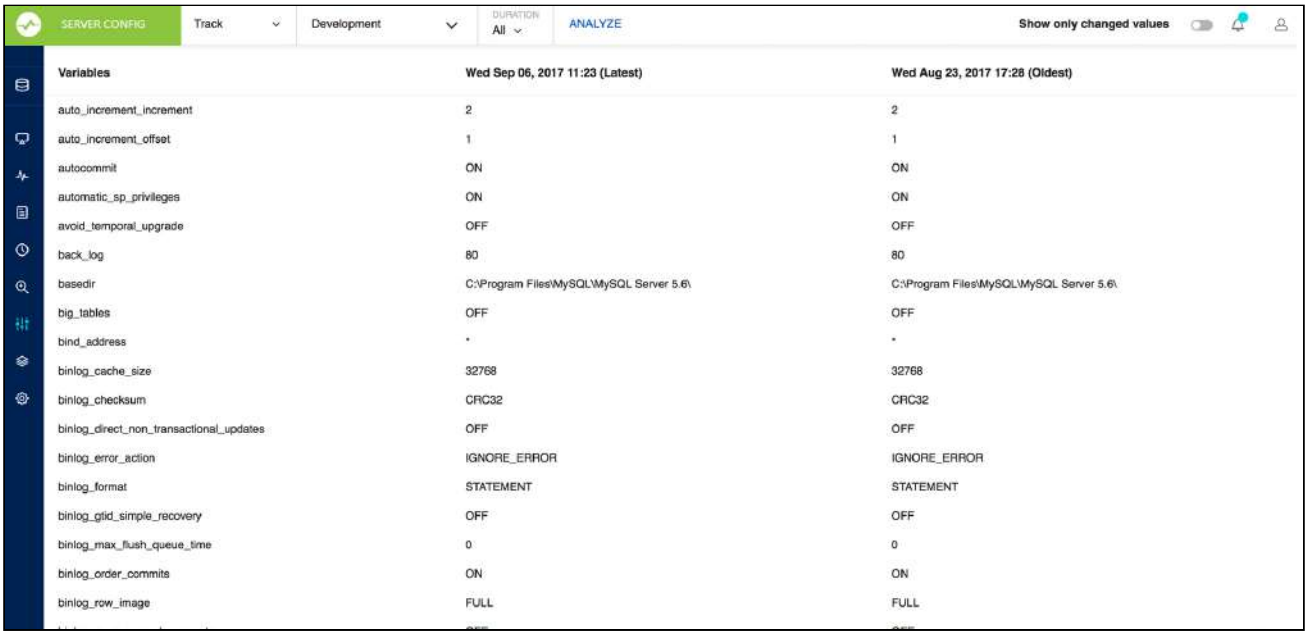


Variables	Tester - 3	RDS	Tester - 1
back_log	80	80	900
basedir	/usr/	/rdsdbbin/mysql/	/usr/
binlog_error_action	ABORT_SERVER	IGNORE_ERROR	ABORT_SERVER
binlog_format	ROW	MIXED	ROW
binlog_group_commit_sync_delay	0	(n/a)	0
binlog_group_commit_sync_no_delay_count	0	(n/a)	0
binlog_gtid_simple_recovery	ON	OFF	ON
binlogging_impossible_mode	(n/a)	IGNORE_ERROR	(n/a)
character_sets_dir	/usr/share/mysql/charsets/	/rdsdbbin/mysql-5.6.27.R1/share/charsets/	/usr/share/mysql/charsets/
check_proxy_users	OFF	(n/a)	OFF
connection_control_failed_connections_threshold	3	(n/a)	(n/a)
connection_control_max_connection_delay	2147483647	(n/a)	(n/a)
connection_control_min_connection_delay	1000	(n/a)	(n/a)
datadir	/var/lib/mysql/	/rdsdbdata/db/	/var/lib/mysql/
default_authentication_plugin	mysql_native_password	(n/a)	mysql_native_password
default_password_lifetime	0	(n/a)	0
disabled_storage_engines		(n/a)	
eq_range_index_dive_limit	200	10	200

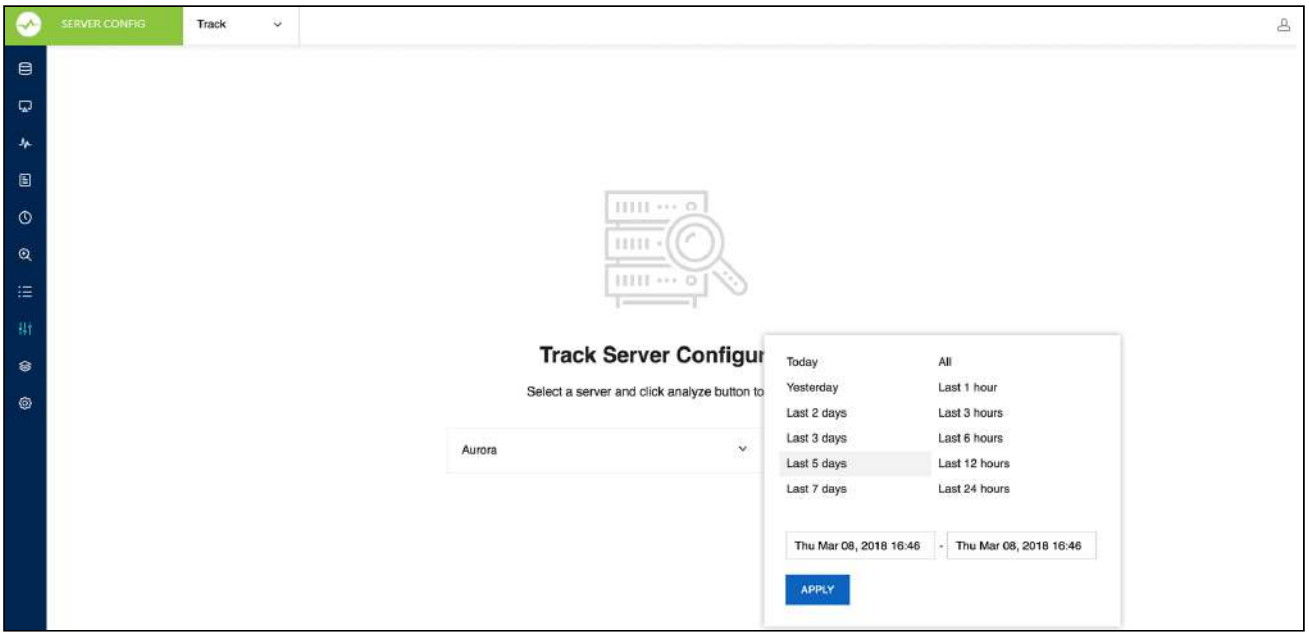
Track Configuration Changes

Track Configuration Changes is version control for your MySQL server Global Variables. SQL DM for MySQL tracks changes to global configuration no matter if the configuration parameters were specified in my.ini/my.cnf, are server defaults or if somebody with SUPER privilege has executed a SET GLOBAL statement. If you notice your MySQL server performance degraded from the last time you checked, you can track, and compare changes in the configuration file to determine whether a change in the Global Variables has brought about this degradation.

To view data for a server, select the server from the drop-down menu and click **ANALYZE**. All timestamps when changes were detected are listed in the drop-down with the latest timestamp and oldest timestamp compared by default.



The time-frame can be changed and the respective changes to the configuration are displayed. To track changes between two timestamps select the original timestamp on the left drop-down menu. Next, select the timestamp to compare with from the right drop-down menu, enable the **Show only changed values** toggle-bar, which hides all values that have not changed between the two timestamps.



Replication

This interface shows the replication graph and relationship of all registered MySQL servers, as well as `SLAVE STATUS` and `MASTER STATUS` where it applies on hovering over each of the server block. The display gets updated after every one minute.

The servers are color-coded to represent the different states of the replication servers: green denotes stable, red disconnected servers and yellow not in sync. The yellow color is decided if any of the following variables return the below value:

```
'SLAVE IO RUNNING' = No
```

or

```
'SLAVE SQL RUNNING' = No
```

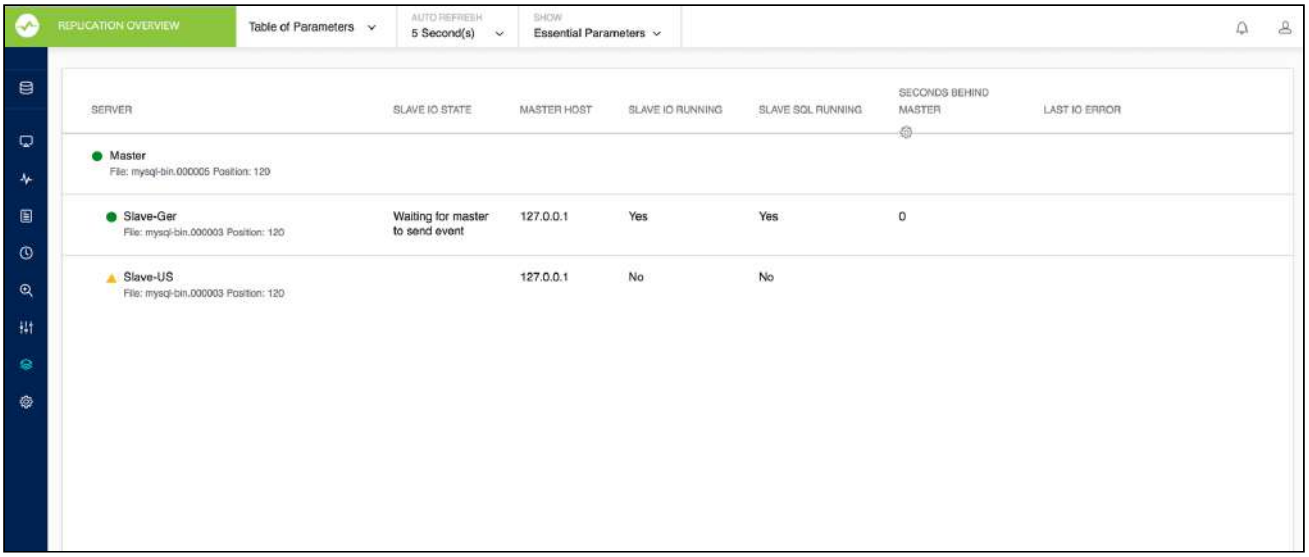
or

```
 'SECONDS BEHIND MASTER' >= 450 Secs (this is the default global value for all the
servers. A user can also set a value according to their MySQL environment)
```

From Monyog 8.1.0, the user can set the **Seconds Behind Master** threshold value for all the servers in SQL DM for MySQL by clicking the icon in the **Seconds Behind Master** column or in the dialog pop-up upon clicking the respective server in the table/topology.

There are two interfaces that the user can choose to see the information about the replication setup:

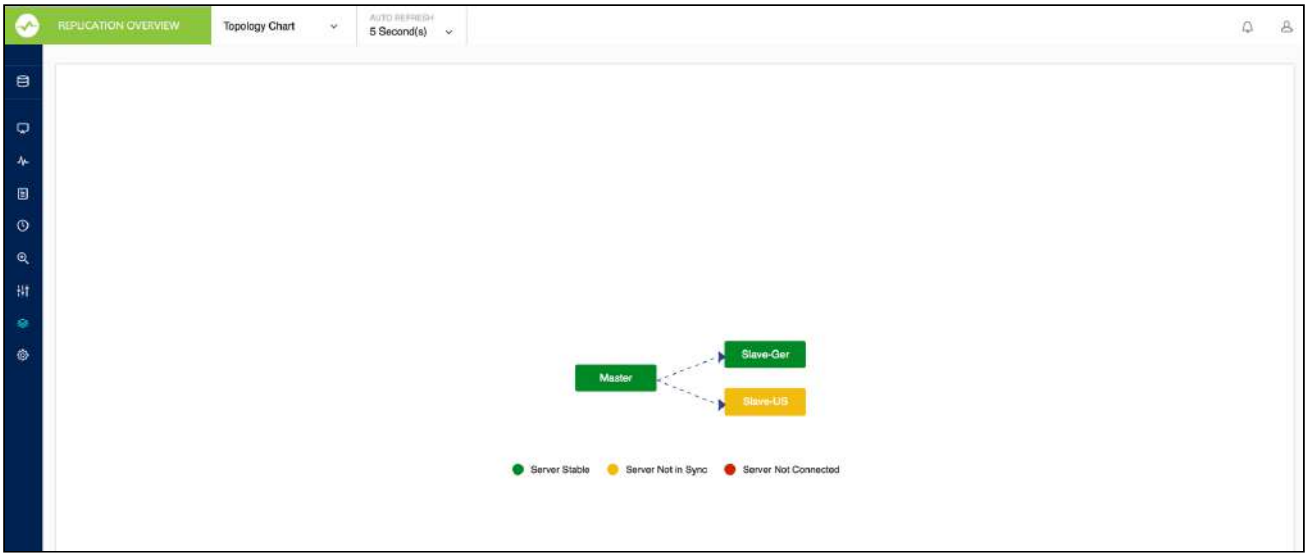
- **Table of Parameters:** This table contains the result set of SHOW [ALL] SLAVE STATUS for servers which have been marked as replication slaves in SQL DM for MySQL. The first time the number of columns in the table is condensed to only display 'Essential Parameters'. To get the detailed view, you can select the option 'All Parameters' from the drop-down menu in the header.



You can sort any column which removes the hierarchy listing of servers. To view slaves in the hierarchy, click the **SHOW HIERARCHY** link.

Also, if the I/O thread ('Slave_IO_Running') is not running, it dynamically brings out the Last IO Error column for a quick view on what went wrong with the Slave IO Thread. It works similarly for SQL thread ('Slave_SQL_Running') as well.

- **Topology Chart:** This interface shows the replication graph and relationship of all registered MySQL servers which are marked as master/slave, as well as SLAVE STATUS and MASTER STATUS while hovering over each of the server blocks.The display gets updated at the user-specified refresh interval which is a browser-specific setting. The default interval is 5 seconds.



The server blocks on clicking gives the result set of SHOW [ALL] SLAVE STATUS for the slave and SHOW MASTER STATUS for the master servers.

SQL DM for MySQL Database Schema

Database Schema reveals the SQLite schema details which are used for managing SQL DM for MySQL connections (copying connections/creating new connections etc.). Also, with this information you get the basic understanding of how data is stored so that you get to query the database for information without using SQL DM for MySQL.

> ⚠ If multiple applications use the same database one issue LOCKS that cause the other to wait, so generally suggested that you use a copy of the database.

What is SQL DM for MySQL Data?

SQL DM for MySQL uses SQLite for storing all the data. With 'data' - we are referring to:

- Data collected from MySQL servers (stored in the database file mysql.data).
- Data collected from the operating system (currently available for Linux only, stored in the database file system.data).
- Data captured from 'sniffing' (stored in the database file sniffer.data).
- Data captured from 'CSO's (stored in database file udo.data).
- Data captured for events (stored in events.data).
- Data captured when a Real-Time session is started and saved (stored in the realtimedata folder: rt_name.data).
- Data related to MySQL server connection (stored in connection.data).
- Also, for faster retrieval in future, SQL DM for MySQL stores the cached data (stored in cache.data).

There is one of each of those database files for every connection except for events.data that is common to all the servers.

Where can you find this data?

Below, you can find the default paths for the data collected by SQL DM for MySQL for the first connection created by SQL DM for MySQL:

In windows systems

```
C:\ProgramData\Webyog\MONyog\Data\0001
```

In Linux systems

```
-   RPM: /usr/local/MONyog/data/0001
-   Tar: In the same directory where MONyog was 'untarred'.
```

How to view existing schema and data?

You can view the existing schema and data by using a SQLite client. In addition to the official SQLite command-line client there are simple GUI clients available SQLite Manager (see page 150) (This is a plugin for the Firefox browser and works on all platforms, but there are more GUI clients available for download - mostly for Windows. Most Linux distributions ship with some database client software that handles SQLite).

> ⚠ Schemas may be subject to change. We may add/remove columns, change data types, change indexes etc. with new releases. When we do that, you can check-in release notes and you can open the database with the tools mentioned to see the columns.

Preferences and Connection Settings

Preferences and global settings

In addition to the data stored on a per-server basis; SQL DM for MySQL has a database storing global user preferences (preferences.config database) and also a very tiny text file (MONyog.ini), that only has what minimal information is required for SQL DM for MySQL to start.

connection.data

There is a Preferences table in connection.data which is used for storing the default processlist query.

```
CREATE TABLE IF NOT EXISTS [preferences] (
   [name] VARCHAR(50) DEFAULT '' NOT NULL PRIMARY KEY UNIQUE,
   [value] TEXT DEFAULT '');
```

The server_names table is used for storing all the connection details.

```
CREATE TABLE [server_names]
   (id INTEGER DEFAULT 0 NOT NULL PRIMARY KEY UNIQUE,
    key VARCHAR(255) DEFAULT '' NOT NULL,
    value VARCHAR(255) DEFAULT '' NOT NULL);
```

MySQL variables and System data

mysql.data and system.data

These two databases have a completely identical structure:

```
CREATE TABLE IF NOT EXISTS [metric_master] (
   [metric_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   [metric_desc] TEXT ASC UNIQUE )
```

```
CREATE TABLE IF NOT EXISTS [snapshot_master] (
   [timestamp_id] INTEGER NOT NULL,
   [metric_id] INTEGER NOT NULL,
   [metric_now] TEXT,
   [metric_diff] TEXT,
   PRIMARY KEY (metric_id, timestamp_id))
```

```
CREATE INDEX IF NOT EXISTS [timestamp_id_index] ON [snapshot_master] ([timestamp_id])
```

```
CREATE TABLE IF NOT EXISTS [timestamp_master] (
   [timestamp_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   [server_timestamp] INTEGER,
   [server_start_time] INTEGER,
   [server_uptime] INTEGER,
   [server_uptime_diff] INTEGER,
   [server_is_connected] INTEGER)
```

```
  CREATE INDEX IF NOT EXISTS [server_timestamp_idx] ON [timestamp_master]
([server_timestamp])
```

What is most important to understand here is the [timestamp_id] column occurring in both [snapshot_master] and [timestamp_master] tables. Actually, with MySQL and InnoDB you would probably create a Foreign Key from [snapshot_master] to [timestamp_master] for constraining and clarity. To get meaningful results you need to `JOIN` the two in the query or use a `SUBQUERY`.

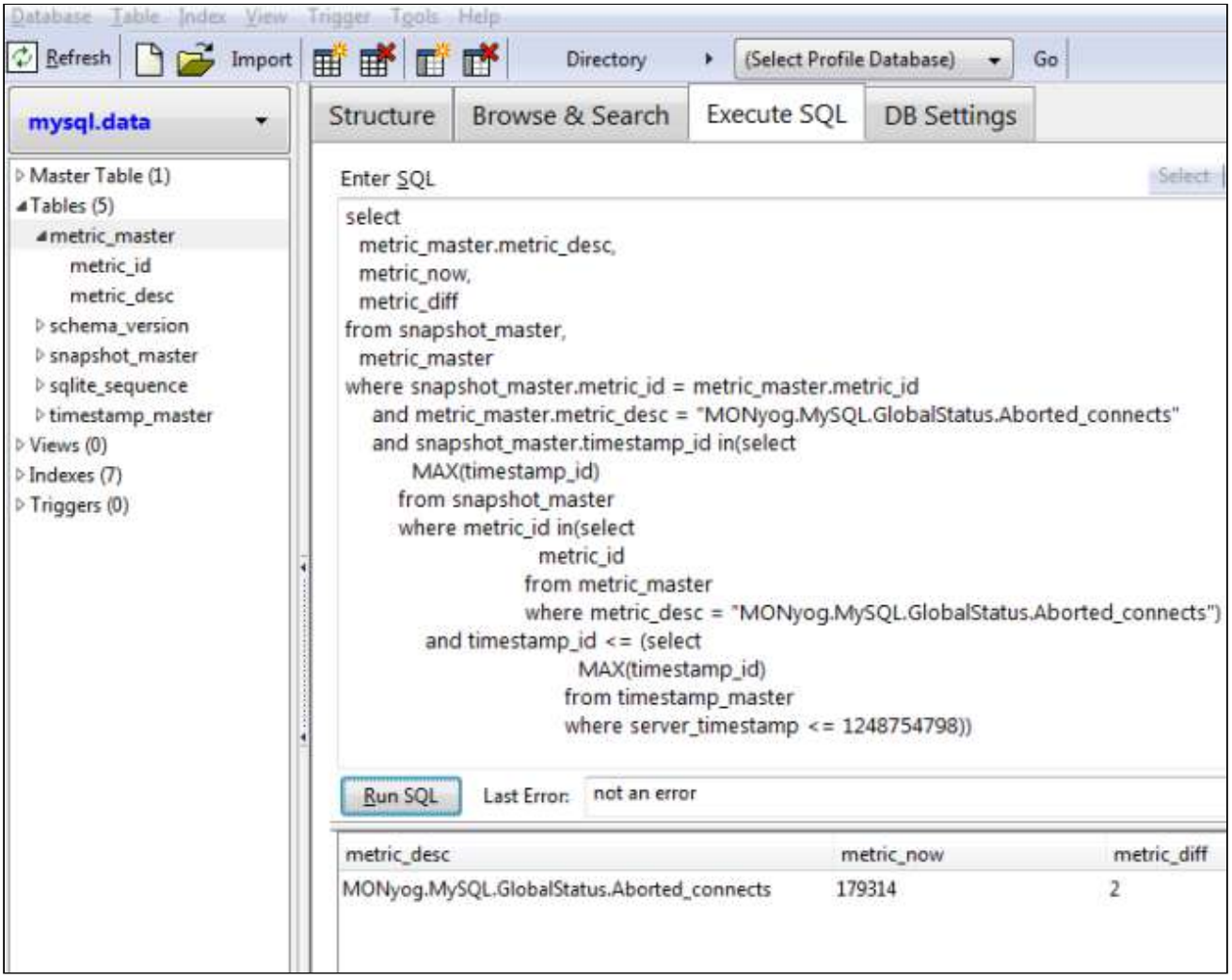
This is basically how they work:

- Everytime SQL DM for MySQL sends a statement like `SHOW VARIABLES/STATUS` of some kind (or fetching a OS metric from Linux/proc folder) one row is INSERTED into [timestamp_master] table with information about current time and the metrics retrieved for each such statement is INSERTED into [snapshot_master]. The [snapshot_master] table contains the metrics details. The [timestamp_id] column identifies when metric details were like that. Also, that timestamps in SQL DM for MySQL databases are unix_timestamps.
- And actually, we do not always `INSERT` into [snapshot_master]. Only if the particular metric was changed since last time something was INSERTED for that metric we will INSERT again. So if you want to find the value of a metric at some particular time you need to find the most recent value stored before that particular time for that particular metric.
- Finally, the [snapshot_master] does not have the names of the metrics. It is not possible to know in advance what metrics the server returns as it depends on server details (version and configuration). And actually a server may be upgraded.
  Saving each textual description only once which would save disk space. So [snapshot_master] only contains a number in the [metric_id] column referring to the textual description the [metric_master] table. So, if the query returns the name of the metric or the metric name should be used in a WHERE-clause also

[metric_master] table must be referenced in the query. If you are familiar with the SHOW statements and what information they return you identify the meaning of each row in [metric_master] table.

> ⚠ The term Metric here refers to the discrete values returned for `SHOW` statements themselves (SHOW GLOBAL VARIABLES; SHOW GLOBAL STATUS; SHOW SLAVE STATUS etc.). Whatever calculations SQL DM for MySQL does in its web interface are done after and not before storage. But we do one calculation before storing however: whenever a metric is INSERTED we also retrieve that latest stored value for the same metric and calculate the difference. Both the current value and this difference is stored (in [metric_now] and [metric_diff] columns respectively).

An example of an easily understandable query doing all this could look like:



An example of a query that we actually execute (optimized for large SQLite databases) to populate a graph is as follows:

```
SELECT metric_now
FROM snapshot_master
WHERE snapshot_master.metric_id = my_metric_id
   AND snapshot_master.timestamp_id IN(
      SELECT MAX(timestamp_id)
      FROM snapshot_master
      WHERE metric_id = my_metric_id
      AND timestamp_id <= (
         SELECT MAX(timestamp_id)
         FROM timestamp_master
         WHERE server_timestamp <= my_metric_timestamp)
   )
```

Actually SQLite support has recommended using SUBQUERIES and not JOINS in most cases with SQLite for best performance with big databases. That is also the experience we have ourselves when profiling different queries returning same results.

udo.data

This is the database where we store data from the Custom SQL Objects(CSOs). This database has 3 different tables- Snapshot_master, Column_master and timestamp_master

```
CREATE TABLE [column_master]
   (id INTEGER NOT NULL PRIMARY KEY,
    timestamp_id INTEGER NOT NULL,
    udo_id INTEGER NOT NULL,
    key_column_value VARCHAR(255),
    column VARCHAR(255),
    UNIQUE([udo_id], [key_column_value], [column], [timestamp_id]));
```

```
CREATE TABLE [snapshot_master] ([timestamp_id] INTEGER NOT NULL,
   [metric_id] INTEGER NOT NULL,
   [metric_now] TEXT,
   [metric_diff] TEXT,
   PRIMARY KEY (metric_id, timestamp_id));
```

```
CREATE TABLE [timestamp_master] (
   [timestamp_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   [server_timestamp] INTEGER,
   [server_start_time] INTEGER,
   [server_uptime] INTEGER,
   [server_uptime_diff] INTEGER,
   [server_is_connected] INTEGER, [udo_id] INTEGER);
```

events.data

This database holds information related to events in SQL DM for MySQL.

```
CREATE TABLE [events]  (id INTEGER PRIMARY KEY AUTOINCREMENT,
   first_seen INTEGER(5) NOT NULL,
   last_seen INTEGER(5) NOT NULL,
   down_count INTEGER(5) NOT NULL DEFAULT 0,
   server_id VARCHAR(255) NOT NULL,
   server_name VARCHAR(255) NOT NULL,
   group_id INTEGER(5) NOT NULL DEFAULT 0,
   counter_id INTEGER(5) NOT NULL DEFAULT 0,
   grp VARCHAR(255),
   name VARCHAR(255),
   sampling_time_frame VARCHAR(255),
   type INTEGER NOT NULL DEFAULT 0,
   threshold VARCHAR(255),
   value VARCHAR(255),
   advice TEXT,
   mail_alert VARCHAR(10) NOT NULL DEFAULT '',
   down_count_override INTEGER(5) NOT NULL DEFAULT 0,
   notify_stable_override VARCHAR(10) NOT NULL DEFAULT '',
   smtp_alert_cnt INTEGER(5) NOT NULL DEFAULT 0,
   snmp_alert_cnt INTEGER(5) NOT NULL DEFAULT 0,
   ignored_timestamp INTEGER NOT NULL DEFAULT 0);
```

```
CREATE TABLE [timestamp_master](
   [timestamp_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   [server_timestamp] INTEGER UNIQUE);
```

Query Analyzer and Processlist data

sniffer.data

```
CREATE TABLE IF NOT EXISTS [query_master](
   [id] INTEGER PRIMARY KEY AUTOINCREMENT,
   [query] TEXT,
   UNIQUE([query]))

CREATE TABLE IF NOT EXISTS [query_snapshot] (
   [pkeyid] INTEGER PRIMARY KEY AUTOINCREMENT,
   [id] INTEGER,
   [threadid] INTEGER,
   [user] TEXT,
   [querytime] INTEGER,
   [uptime] INTEGER,
   [host] TEXT DEFAULT)
```

Here, you see the same pattern as above: the [id] column in the [query_snapshot] table identifies a row in the [query_master] where the actual/textual query is saved. Also note, that a UNIQUE KEY is defined on the [query] column so that we can use an INSERT ... ON DUPLICATE KEY construction and thus ensure that the [query_master] table only has the same query stored once. But in [query_snapshot] table there is one row for every instance of the query.

Actually, with general/slow query log analyzers we use identical tables. The log CHUNK retrieved from the server isparsed and the tables populated like - you see in your sniffer.data database. The tables used for log analysis, however, are MEMORY tables and they are only available in SQL DM for MySQL and only for as long as they are needed.

Processlist

Also, SQL DM for MySQL processlist feature uses a SQLite MEMORY table (for every server). The table structure is as follows:

```
CREATE TEMPORARY TABLE IF NOT EXISTS [processlist](
  [Id] INTEGER NOT NULL PRIMARY KEY,
  [User] TEXT,
  [Host] TEXT,
  [Db] TEXT,
  [Command] TEXT,
  [Time] INTEGER,
  [State] TEXT,
  [Info] TEXT,
  [Action] TEXT)
```

So, that is how the MySQL processlist displays in SQL DM for MySQL - unlike when connected to MySQL directly - it can be filtered, sorted etc. by using WHERE, ORDER BY, GROUP BY etc. with a SELECT query against the SQL DM for MySQL [processlist] table. But as it is a MEMORY table you can only query it from inside the SQL DM for MySQL processlist interface.

Information about the SQL DM for MySQL database schema itself

There is a schema_version table in all databases created by SQL DM for MySQL. Every time SQL DM for MySQL starts it checks here if the database is up to date with the current program version. If it is not, SQL DM for MySQL will perform the necessary schema upgrades at start-up. Schema definition reads as follows:

```
CREATE TABLE IF NOT EXISTS [schema_version] (
   [schema_desc] TEXT,
   [schema_major_version] TEXT,
   [schema_minor_version] TEXT,
   PRIMARY KEY ([schema_major_version], [schema_minor_version]))
```

SQL DM for MySQL Data Maintenance

You can rebuild the embedded SQLite database for a server by clicking the arrow icon on the left panel next to the server name and selecting the **Rebuild Database** option. Using this option periodically may result in better performance - including shorter startup time when OS is rebooted. Basically using this option defragments the database including indexes. It is not possible to provide an absolute advice on how this option should be used. But the larger the databases and the shorter the sample interval the faster there is a chance of fragmentation occurring after a huge amount of INSERTs and DELETEs to the database (Only with databases files of around 1 GB and larger we have seen the need for this. A general advice could be to execute monthly with large database files).

Also note, if the error "Database or Disk is full" (on Windows) prompts when VACUUM is executing, it refers to the file system hosting the temporary files location. Try setting the TEMP environment variable to a location with more free space, restart your system and it works.

Location in Windows 2008: `C:\ProgramData\Webyog\MONyog\Data\`

Real-Time data

The session data that are recorded by SQL DM by MySQL Real-Time are stored in SQLite in Monyog directories data folder.

The Real-time database has many tables. Schema details are as follows:

inndob_locks:

```
CREATE TABLE `innodb_locks` (
   `row_id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   `lock_id` VARCHAR (81) NOT NULL DEFAULT '',
   `lock_type` VARCHAR (32) NOT NULL DEFAULT '',
   `lock_table` VARCHAR (1024) NOT NULL DEFAULT '',
   `lock_index` VARCHAR (1024) DEFAULT NULL,
   `lock_data` VARCHAR (8192) DEFAULT NULL,
   `lock_mode` VARCHAR (8192) DEFAULT NULL)
```

innodb_transactions:

```
CREATE TABLE `innodb_transactions` (
   `row_id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   `trx_id` VARCHAR (18) NOT NULL DEFAULT '',
   `trx_state` VARCHAR (13) NOT NULL DEFAULT '',
   `trx_start_time` INTEGER NOT NULL,
   `trx_query_id` INTEGER (128) NOT NULL DEFAULT '',
   `trx_query_starttime` INTEGER NOT NULL,
   `trx_query_endtime` INTEGER NOT NULL,
   `trx_user_host` VARCHAR (20) NOT NULL DEFAULT '',
   `trx_db` VARCHAR (64) NOT NULL DEFAULT '',
   `blocking_trx_id` VARCHAR (18) NOT NULL DEFAULT '',
   `blocking_query_id` INTEGER NOT NULL DEFAULT '')
```

metric_master:

```
CREATE TABLE [ metric_master ] (
   [ metric_id ] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   [ metric_desc ] TEXT ASC UNIQUE)
```

profiler_timestamps:

```
CREATE TABLE `profiler_timestamps` (
   [ timestamp_id ] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   [ server_timestamp ] INTEGER UNIQUE)
```

query_master:

```
CREATE TABLE 'query_master' (
   'id' INTEGER PRIMARY KEY AUTOINCREMENT,
   'query' TEXT,
   UNIQUE ('query'))
```

query_snapshot:

```
CREATE TABLE 'query_snapshot' (
   'pkeyid' INTEGER PRIMARY KEY AUTOINCREMENT,
   'id' INTEGER,
   'threadid' INTEGER,
   'user' TEXT,
   'querytime' INTEGER,
   'uptime' INTEGER,
   'host' TEXT,
   'state' TEXT,
   'db' TEXT)
```

schema_master:

```
CREATE TABLE [ schema_master ] (
   [ schema_id ] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   [ schema_name ] TEXT UNIQUE)
```

schema_version:

```
CREATE TABLE [ schema_version ] (
   [ schema_desc ] TEXT,
   [ schema_major_version ] TEXT,
   [ schema_minor_version ] TEXT,
   PRIMARY KEY (
      [ schema_major_version ],
      [ schema_minor_version ]))
```

snapshot_master:

```
CREATE TABLE [ snapshot_master ] (
   [ timestamp_id ] INTEGER NOT NULL,
   [ metric_id ] INTEGER NOT NULL,
   [ metric_now ] TEXT,
   [ metric_diff ] TEXT,
   PRIMARY KEY (metric_id, timestamp_id))
```

sqlite_sequence:

```
CREATE TABLE sqlite_sequence(name,seq)
```

table_master:

```
CREATE TABLE [ table_master ] (
   [ table_id ] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
   [ schema_id ] INTEGER,
   [ table_name ] TEXT,
   UNIQUE (schema_id, table_name))
```

table_snapshot:

```
CREATE TABLE \[ table\_snapshot \] (
    \[ timestamp\_id \] INTEGER NOT NULL, \[ table\_id \] INTEGER NOT
    NULL, \[ COUNT\] INTEGER, PRIMARY KEY (timestamp\_id, table\_id))
```

timestamp_master:

```
CREATE TABLE \[ timestamp\_master \] (
    \[ timestamp\_id \] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT
    UNIQUE, \[ server\_timestamp \] INTEGER UNIQUE, \[
    server\_start\_time \] INTEGER, \[ server\_uptime \] INTEGER, \[
    server\_uptime\_diff \] INTEGER, \[ server\_is\_connected \]
    INTEGER)
```

SQL DM for MySQL ini parameters

The MONyog.ini is the configuration file used by SQL DM for MySQL to read the configuration and is created when MONyog(SQL DM for MySQL) service is started for the first time. It consists of different parameters and can be used to customize a few settings as well. MONyog.ini consists of the following parameters:

- **Port:** The port at which SQL DM for MySQL listens to the HTTP requests. The default port is 5555, but you can change it to any open port available on the system where SQL DM for MySQL is running.
- **Password:** This field stores the obfuscated form of the admin password for SQL DM for MySQL UI. Hence, you can not change/edit the admin password from here, you can do it from the SQL DM for MySQL UI ("User profile" icon at the top right corner)
- **Data_path:** This gives the default data path at which SQL DM for MySQL stores the collected data from all the registered servers. A user can change the data path from here in order to store data to the desired location.
- **Registration_name:** The registration name used while registering the license keys in SQL DM for MySQL.
- **Key:** The license key used to register SQL DM for MySQL.
- **Install_ID:** This gives the unique ID associated with each SQL DM for MySQL installation.
- **JSRuntime_size:** It gives the amount of memory allocated to run all the Javascript (Monitors and Dashboard page) in SQL DM for MySQL. The default value is 1024 MB, it may need to be changed to higher values like 2048, 4096 when monitoring 100+ servers
- **Overview_Available:** It determines whether the Overview page is available in SQL DM for MySQL or not. The default value is 1, indicating Overview page will be available. User can make it 0, if they do not want this page to be displayed.
- **MONyogLogPath(Not present in the default MONyog.ini file):** It can be used to change the default MONyog.log path, for e.g: MONyogLogPath=/home/ubuntu/MONyogLog/

> ⚠ Please stop MONyog(SQL DM for MySQL) service before making any changes to any of the below parameters.

Depending on the mode of SQL DM for MySQL installation, you can find the MONyog.ini file in the below locations:

- **Windows**

```
<MONyog installation drive>\ProgramData\Webyog\MONyog\MONyog.ini
```

- **Linux System**

```
- for RPM: /usr/local/MONyog/MONyog.ini
- for .tar:<MONyog extracted directory>/MONyog/MONyog.ini
```

It is created when MONyog(SQL DM for MySQL) service is started for the first time. A default MONyog.ini looks like this:

```
[GENERAL]

Port=5555

Password=

Data_path=/home/pankaj/Downloads/MONyog/bin/..//data

Registration_name=Webyog

Key=7AD04B55-8898-4DEFG90-8D37-D88243971EF3

Install_id=b11c77ad-e5ea-439e-99bc-8f551a1380b7

JSRuntime_size=1024

Overview_Available=1

Intercom_Available=1
```

Apart from the above default parameters, you can also use MONyog.ini file to change the default MONyog-bin.pid file location from "/var/run/" to some other directory. A user can give the path in the MONyog.ini file like: “Pid_file_path=/abc/xyz".

The SQL DM for MySQL API

SQL DM for MySQL API is an application programming interface that enables SQL DM for MySQL to interact with other software like a command shell or a script or application. To use the SQL DM for MySQL API you send a HTTP request simply (specifying the SQL DM for MySQL URL with parameters) and thus even the address line of any browser can be used. Configure SQL DM for MySQL and perform other actions (like you will be able to from GUI). The APIs are capable of managing the servers (Adding, Editing, and Removing) along with managing data collection and alert settings.

Using the SQL DM for MySQL API

You can access the API by passing parameters to Monyog through its base URL.

For example, if Monyog is running on a system with IP 192.168.1.1, then the parameters need to be passed to: `http://192.168.1.1:5555/`

You can use either of the HTTP methods GET and POST.

The Parameters

The parameters that you will need to pass are:

- `_object`: This basically addresses the logical object in Monyog that you want to direct your request to.The only acceptable value is MONyogAPI.
- `_action`: This specifies the part of the object specified above that you want to direct your request to. The acceptable values are:
    - `Alerts`
    - `DataCollection`
    - `Sniffer`
    - `LongRunningQueries`
    - `LockedQueries`
    - `LongRunningQueryAction`
    - `AddServer`
    - `EditServer`
    - `RemoveServer`
- `_value`: The operation that you want to perform for the action specified in the _action field. Acceptable values include for:
    - `Alerts`, `DataCollection`, `Sniffer`, `LongRunningQueries`, `LockedQueries`: `enable` and `disable`
    - `LongRunningQueryAction`: `notify`, `kill` and `notifyandkill`
- `_user`: It may be Monyog user, LDAP user or LDAP Group user. In case, no user is supplied, admin account is used by default.
- `_password`: The password for the specified _user.
- `_server`: Name or data directory number of the servers separated by a comma(‘,’) for which the operation to be performed.
- `_tags`: Name of the tag separated by a comma(‘,’) for which the operation to be performed for all the servers under the specified tag.

API's for server management

To manage servers in Monyog via API, the following parameters need to be passed:

- `_server`: The name or data directory number of the server to be registered
- `_mysqlhost`: MySQL host/ip address
- `_mysqluser`: MySQL user name
- `_mysqlport`: MySQL port
- `_mysqlpassword`: MySQL user password

For example, to add a server:

```
$ curl "192.168.1.1:5555/?_object=MONyogAPI&_action=addserver&_user=admin
       &_server=Production&_mysqlhost=127.0.0.1&_mysqluser=admin
       &_mysqlport=3306&_mysqlpassword=adminpassword"
```

Additional parameters for registering servers are listed here.

For example, suppose you have a server named Production001 registered with Monyog. To stop data collection for this server using the HTTP GET method, the URL would look like:

```
$ curl “http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
       &_value=disable&_password=mypassword&_server=Production001”
```

In summary, the various URLs that you can use with curl:

Change LDAP bind user password:

```
$ curl "127.0.0.1:5555/?_object=MONyogAPI&_action=changeldapbindpassword&_user=<sqldm
user name>&_password=<sqldm user password>&_currentpassword=<current ldap user
password>&_newpassword=<new ldap user password>"
```

Change user password:

```
$ curl "127.0.0.1:5555/?_object=MONyogAPI&_action=changepassword&_user=<sqldm
username>&_password=<currentpassword>&_newpassword=<newpassword>"
```

Starts data collection for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
       &_value=enable&_user=admin&_password=Password&_server=Production001"
```

Starts data collection for <multiples servers>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
       &_value=enable&_user=admin&_password=Password&_server=Production001,Test"
```

Stops data collection for <server name>(Slave Of Production)

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
       &_value=disable&_user=admin&_password=Password&_server=Slave+Of+Production"
```

Starts data collection for <tag>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
       &_value=enable&_user=admin&_password=Password&_tag=Production"
```

Stops data collection for <tag>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
       &_value=disable&_user=admin&_password=Password&_tag=Production"
```

Stops data collection globally for all the servers (Maintenance)

```
$ curl “http://192.168.1.1:5555/?
_object=MONyogAPI&_action=datacollection&_value=enable&_user=admin&_password="
```

Enables alerts for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=Alerts
       &_value=enable&_user=admin&_password=Password&_server=Production001"
```

Disables alerts for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=Alerts
       &_value=disable&_user=admin&_password=Password&_server=Production001"
```

Enables alerts for <tag>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=Alerts
       &_value=enable&_user=admin&_password=Password&_tag=Production"
```

Disables alerts for <tag>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=Alerts
       &_value=disable&_user=admin&_password=Password&_tag=Production"
```

Disables alerts globally for all the servers(Maintenance)

```
$ curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=Alerts&_value=disable&_user=admin&_password="
```

Enables alerts globally for all the servers(Maintenance)

```
$ curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=Alerts&_value=enable&_user=admin&_password="
```

Enables Sniffer for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=sniffer
       &_value=enable&_server=Production001"
```

Disables Sniffer for <server name>

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=sniffer
       &_value=disable&_server=Production001"
```

Add Server

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=addserver
       &_mysqluser=msandbox&_mysqlhost=127.0.0.1&_mysqlport=3306&_tags=Production
       &_server=Test&_mysqlpassword=msandbox&_connectontype=direct
       &_user=admin&_password=Password"
```

Add Server with SSH Tunnel

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=addserver
       &_mysqluser=msandbox&_mysqlhost=127.0.0.1&_mysqlport=3306
       &_tags=Production&_server=Test&_mysqlpassword=msandbox
       &_connectiontype=ssh&_sshhost=192.168.1.86&_sshuser=username
       &_sshpassword=sshpassword&_sshport=22&_user=admin&_password=Password"
```

Edit Server

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=editserver
       &_mysqluser=msandbox&_mysqlhost=127.0.0.1&_mysqlport=3306
       &_tags=Production&_server=Test&_mysqlpassword=msandbox
       &_connectontype=direct&_user=admin&_password=Password"
```

Delete Server

```
$ curl "http://192.168.1.1:5555/?_object=MONyogAPI&_action=removeserver&_server=Test"
```

Delete all the servers under <tag>

```
curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=removeserver&_tag=Production”
```

Delete multiple servers

```
curl "http://192.168.1.1:5555/?
_object=MONyogAPI&_action=removeserver&_server=Server1,Server2”
```

Return Codes

Assuming that the connection to SQL DM for MySQL was successful, it returns a text message. The message is in the JSON format:

```
{“STATUS”: “SUCCESS/FAILURE”, “RESPONSE” : “<Response text>”}
```

Your application can parse this message and determine whether the operation was successfully carried out or not.

⚠ Since version 5.21 we have deprecated the API calls to `_object=ConnectionMgr`. Instead use `_object=MONyogAPI`.

Applications

The Monyog API is very flexible and can be accessed from other programming languages including scripting languages such as Perl, VBScript, etc. Here is a very generic Perl script that accepts the required parameters from the command line and executes the specified action:

```
#! /usr/bin/perl
use LWP 5.64;
# USAGE: MONyog.pl <hostname>:<port> <user> <password> <connection_name/ID> <action>
<value>
# $ARGV[0] = hostname:port of server running Monyog
# $ARGV[1] = Monyog user
# $ARGV[2] = Monyog password
# $ARGV[3] = connection name
# $ARGV[4] = action
# $ARGV[5] = value
my $numArgs = $#ARGV + 1;
if($numArgs < 5) {
   die 'USAGE: MONyog.pl <hostname>:<port> <user> <password> <connection_name/ID>
<action>';
}

my $browser = LWP::UserAgent->new;

# The request URL
my $url = URI->new('http://' . $ARGV[0] . '/');

# The form data pairs:
$url->query_form(
   '_object' => 'MONyogAPI',
   '_action' => $ARGV[4],
   '_user' => $ARGV[1]
   '_password' => $ARGV[2],
   '_server' => $ARGV[3],
   '_value' => $ARGV[5]
);

# The response object
$response = $browser->post($url);

if (!$response->is_success) {# Error connecting to MONyog
   die $response->status_line . "\n";
} else {

   # Successfully connected to MONyog; print MONyog's response
   print $response->content . "\n";

}
```

Additional Parameters for API

SSH tunneling for MySQL

Property name	Description
`_sshhost`	Host machine on which SSH server is running.
`_sshuser`	Username to access the SSH server.
`_sshport`	Port on which SSH server is listening. **Range:** 1 .. 65535 **Defaluts:** "22"
`_sshauthtype`	SSH authentication type. **Permitted values:** `key,password` **Default:** `password`
`_sshpassword`	SSH user's password for password based authentication.
`_sshprivatekey`	SSH user's private 'key' for key based authentication. **Note** The key must be url encoded. For example line endings `\r\n` must be encoded to `%0A` and " " (white space) to `+`.
`_sshpassphrase`	Passphrase for private key file (if any) for key based authentication.

SSL encryption for MySQL

Property name	Description
`_cacertificate`	The digital certificate issued by CA. **Note** The key must be url encoded. For example line endings 'rn' must be encoded to '%0A' and " " (white space) to "+".
`_sslcipher`	Encryption algorithm like DES, AES etc. **Note** The key must be url encoded. For example line endings 'rn' must be encoded to '%0A' and " " (white space) to "+".
`_usesslauth`	SSL client authentication type. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"

Property name	Description
`_clientkey`	Private key of the client that is needed for encryption.
`_clientcertificate`	The client certificate.

SSH settings

To enable and use SSH for reading logs via SFTP and/or OS monitoring, the following query parameters are required:

Property name	Description
`_osmonitoring`	Enable or disable SSH settings for OS monitoring/reading MySQL logs via SFTP. **Permitted values:** yes/no; true,false;1/0 **Default:** no
`_ostype`	OS monitoring is available for Linux systems only. Setting to "Linux", SQL DM for MySQL can monitor system related metrics like CPU consumption, Memory Usage, etc. **Permitted values:** `linux,rds` .
`_ossameasmysqltunnel`	Uses the same SSH details provided for SSH tunneling. Setting to "yes" will use the details provided for SSH tunneling to MySQL for this registration. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_sshsystemhost`	Host machine on which SSH server is running.
`_sshsystemuser`	Username to access the SSH server.
`_sshsystemport`	Port on which SSH server is listening. **Range:** 1 ..65535 **Default:** "22"
`_sshsystemauthtype`	SSH authentication type to be used.
`_sshsystempassword`	SSH user's password for password based authentication.

Property name	Description
_sshsystemprivatekey	SSH user's private 'key' for key based authentication. **Note** The key must be url encoded. For example line endings 'rn' must be encoded to '%0A' and " " (white space) to "+".
_sshsystempassphrase	Passphrase for your private key file (if any) for 'key' based authentication.

Notification settings

Property name	Description
_mailalerts	Setting to "yes" will send alerts/notifications via email. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
_snmpalerts	Setting to "yes" will trigger SNMP traps on event of alerts. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
_mailaddress	SQL DM for MySQL will send alerts to the email addresses specified under _mailaddress. Accepts comma (,) separated list of email addresses.
_mailaddresscritical	SQL DM for MySQL will send critical alerts to the email addresses specified under _mailaddresscritical.
_mailaddresswarning	SQL DM for MySQL will send warning alerts to the email addresses specified under _mailaddresswarning
_mailaddressother	SQL DM for MySQL will send other alerts such as MySQL server restart and MySQL configuration changes to the email addresses specified under _mailaddressother.
_writetosyslog	Seting to yes will route the alert notifications to syslog. **Permitted values:** yes/no; true,false; 1/0
_slackalerts	Setting to "yes" will route alerts/notifications to the Slack room. **Permitted values:** yes/no
_slackrulename	Specify the Slack Rule to be used.

Property name	Description
_pagerdutyalerts	Setting to "yes" will route alerts/notifications to the PagerDuty. **Permitted values:** yes/no
_pagerdutyrulename	Specify the PagerDuty rule to be used.
_alertableinterval	Number of times to wait before sending the alerts. **Default:** "1"
_notifyserverconfigchange	SQL DM for MySQL will send an alert whenever there is a change in MySQL configuration. **Permitted values:** yes/no; true,false; 1/0 **Default:** "yes"
_notifyserverrestart	SQL DM for MySQL will send an alert whenever the server restarts. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
_briefemail	Enable or disable detailed email notification. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
_notifystable	Notify stable alerts when monitor goes into alert-able state and then becomes stable. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
_notifytillstable	Keeps notifying the user until the counter becomes stable. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
_reminderinterval	Defines the number of data collections after which the user receives the notification until the counter becomes stable. **Default:** "5"

Data collection settings

Property name	Description
`_datacollection`	Enable data collection for server. SQL DM for MySQL will not collect data for this registration if data collection is disabled. **Permitted values:** yes/no; true,false; 1/0 **Default:** "yes"
`_datacollectioninterval`	The interval for the data collections in seconds. **Default:** "300" seconds (5 minutes)
`_dataretentiontime`	Data purging interval for the server. Timeframe should be specified in seconds. **Default:** "604800" (7 days)

Replication settings

Property name	Description
`_replicationslave`	Consider a server as slave server. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_autoregisterslaves`	Register all slave servers. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"

Galera Settings

Property name	Description
`_autoregistergaleranodes`	Auto-register all galera nodes. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"

Error log monitoring

Property name	Description
`_enableerrorlog`	To enable error log monitoring. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_errorlogreadmode`	Mode to read the error log file. **Permitted values:** sftp, local, rds **Default:** "local"
`_errorlogpath`	MySQL error log path. **Default:** "/var/log/mysql/server-err.log"
`_dbidentifier`	A unique name to identify your RDS/Aurora instance.
`_instanceregion`	The region in which your instance is hosted, e.g: us-east-1.
`_accesskey`	A 20 character long key ID, is generated from AWS Mangement Console.
`_secretkey`	A 40 character long key ID, is generated from AWS Management Console.

Slow query log and General query log settings

Property name	Description
`_logreadmode`	Mode to read the General and Slow query log files. **Permitted values:** sftp, local, rds **Default:** "local"
`_querylogdestination`	Read logs stored either from FILE or TABLE. **Permitted values:** file, table **Default:** "file"
`_enableslowquery`	To enable slow query log monitoring. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_slowquerylogpath`	Path for slow log. **Default:** /var/log/mysql/server-slow.log

Property name	Description
`_enablegeneralquery`	To enable general query log monitoring. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_generalquerylogpath`	Path for general log. **Default:** /var/log/mysql/server-general.log
`_dbidentifier`	A unique name to identify your RDS/Aurora instance.
`_instanceregion`	The region in which your instance is hosted, e.g: us-east-1.
`_accesskey`	A 20 character long key ID, is generated from AWS Management Console.
`_secretkey`	A 40 character long key ID, is generated from AWS Management Console.

Audit log settings

Property name	Description
`_auditlogreadmode`	Mode to read Audit log. **Permitted values:** sftp, local, rds **Default:** "local"
`_enableauditlog`	To enable audit log monitoring. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
`_auditlogpath`	Path for audit log. **Default:** /var/lib/mysql/server_audit.log
`_dbidentifier`	A unique name to identify your RDS/Aurora instance.
`_instanceregion`	The region in which your instance is hosted, e.g: us-east-1.
`_accesskey`	A 20 character long key ID, is generated from AWS Management Console.

Property name	Description
_secretkey	A 40 character long key ID, is generated from AWS Management Console.

Sniffer settings

Property name	Description
_enablesniffer	Enable or disable sniffer analysis. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
_sniffermode	Specifying a way to populate sniffer data for analysis. **Permitted values:** processlist, performanceschema, proxy **Default:** "processlist"
_monitorlongrunningqueries	Monitor only long running queries, executing beyond the specified time. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
_longrunningquerytime	The time which qualifies a query as long running query (in seconds). **Default:** "10"
_ignorequeriesbyuser	A filter to ignore queries by a user.
_longrunningqueryaction	The action to be performed for long running queries. **Permitted values:** notify, kill, notifyandkill **Default:** notify
_monitorlockedqueries	Monitors for queries in the locked state. **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"
_snifferproxyhost	MySQL proxy host.
_snifferproxyport	MySQL proxy port.
_sniffinginterval	This interval specifies how frequently SQL DM for MySQL should "sniff" MySQL server (in seconds). **Default:** "1"

Property name	Description
`_snifferpurginginterval`	Specifying a timeframe to purge the data collected (in seconds). **Default:** "259200" (3 days)
`_snifferfilteruser`	A filter to sniff queries only by specified users.
`_snifferfilterhost`	A filter to sniff queries only by specified hosts.
`_sniffquerystartingwith`	To sniff queries starting with the string.
`_snifferlongquerymail`	Setting to "yes" will enable SQL DM for MySQL to send mail alerts for long running queries. **Permitted values:** yes/no
`_snifferlongquerymailto`	SQL DM for MySQL will send long running query alert to the email addresses specified under _snifferlongquerymailto
`_snifferlongquerytrap`	Setting to "yes" will enable SQL DM for MySQL to send SNMP traps for the long running queries. **Permitted values:** yes/no
`_snifferlongquerywritetosyslog`	Setting to "yes" will enable SQL DM for MySQL to write the alerts for the long running queries to the syslog of the host machine of SQL DM for MySQL. **Permitted values:** yes/no
`_snifferlongqueryslack`	Setting to "yes" will enable SQL DM for MySQL to route the alerts to the Slack room for the long running queries. **Permitted values:** yes/no
`_snifferlongqueryslackrule`	Specify the Slack rule to be used for sending the Long running query alerts.
`_snifferlongquerypagerduty`	Setting to "yes" will enable SQL DM for MySQL to route the alerts to PagerDuty for the long running queries. **Permitted values:** yes/no
`_snifferlongquerypagerdutyrule`	Specify the PagerDuty rule to be used for sending the Long running query alerts.

Deadlock monitoring settings

Property name	Description
`_enabledeadlockmonitoring`	Enable or disable InnoDB deadlock monitoring. Turning this ON will help in tracing deadlocks reported by "SHOW INNODB STATUS". **Permitted values:** yes/no; true,false; 1/0 **Default:** "no"

Manage Monitors settings

Property name	Description
`_disabledmonitorgroups`	This specifies the monitor groups that are to be disabled for the server. Comma-separated Group IDs may be supplied. To know the Group IDs of the various monitor groups, you may hover over the Monitor Groups in Customize -> Manage Monitor Groups. For example, if you want to disable, Binary Log (Group ID: 17), Replication (Group ID : 19) and MySQL Cluster(Group ID: 27), the parameters will be 17,19,27 **Default:** 7,14,22

Real-Time Mode Setting

Property name	Description
`_realtimemode`	For setting up Real-Time monitoring mode. **Permitted values:** processlist/performanceschema **Default:** "processlist"

Connection Settings

Property name	Description
`_mysqlconnecttimeout`	Specify MySQL connection timeout. **Default:** "30" seconds
`_sshconnecttimeout`	Specify SSH tunnel connection timeout. **Default:** "30" seconds
`_sshsystemconnecttimeout`	Specify SSH connection timeout. **Default:** "30" seconds

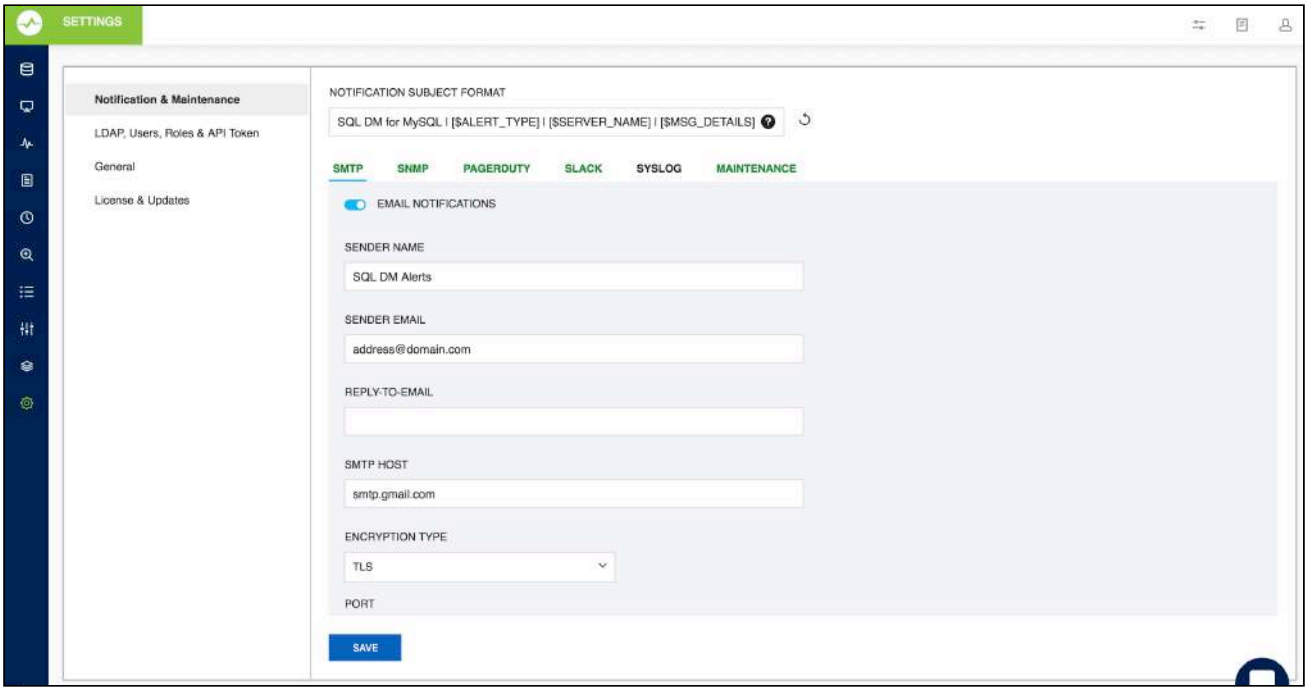
Settings

Use the Settings page to configure the following options in SQL DM for MySQL:

- Notification & Maintenance[37]
- LDAP, Users, Roles, & API Token[38]
- General[39]
- License & Updates[40]
- SQL DM for MySQL Log[41]

Notification & Maintenance

The options here offer you to configure multiple notification channels. To route the alerts generated in SQL DM for MySQL, you have to enable and configure the required channels.



Notification Subject Format

This section provides details to customize the Subject format of the alert notifications that you receive from SQL DM for MySQL. The default format is: **SDM for MySQL | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS]**

[$ALERT_TYPE]: This describes the severity of the alert - the values can be Critical, Warning, Stable, and INFO. The Alert type is set up based on the type of the alert for which the email was generated. The alert type is set based on

37 http://wiki.idera.com/x/uAEGBg
38 http://wiki.idera.com/x/uQEGBg
39 http://wiki.idera.com/x/ugEGBg
40 http://wiki.idera.com/x/uwEGBg
41 http://wiki.idera.com/x/vAEGBg

the following conditions:

- **Critical -** SQL DM for MySQL sends critical alerts to the registered emails in the following scenarios:
    a. When Monitor value crosses the set Critical threshold limit.
    b. When the Long running query action is “Notify And Kill".
    c. Low disk space alert.
    E.g.: SDM for MySQL | Critical | Production | Excessive Privileges: Number Of Users Having Globa…
    SQL DM for MySQL | Critical | SQLDM-Host | Low Disk Space.
- **Warning -** Warning alerts are sent when it meets the following conditions:
    a. When Monitor value crosses the set Warning threshold limit.
    b. When the Long running action is “Notify”
    E.g.: SQL DM for MySQL | Warning | 5.7.18 128 | MySQL Logs: General log - Enabled? - Yes.
- **Stable -** When a Monitor value had crossed the set Critical/Warning threshold limits and is back to the recommended values, user is notified when the alerting monitor is now stable. The Alert Type of this notication is Stable.
E.g.: SQL DM for MySQL | Stable | Staging | Current Connections: Currently running threads - 1.
- **Info -** Alert type is Info when the mysql server restarts or there is a configuration change. This alert isgenerated when the below conditions are enabled when the server was registered.
    a. Notify when server restarts.
    b. Notify when server config changed.
    E.g.: SQL DM for MySQL | Info | Testing | Server configuration change detected.

[$SERVER_NAME]: Name of the server for which the alert is generated. For Low disk space mails, the server_name is SQLDM-Host.
[$MSG_DETAILS]: This specifies the details of the Alert.
For example, “Linux: CPU usage - 74.4%” OR “Long Running Query: QUERY” OR “Server configuration change detected" Or "Server restarted".
If there are more than one monitor alert in an email then this will list down the number of events in the email, i.e, “n Events” where ‘n’ is the “total no of events”.
E.g.: SQL DM for MySQL | Critical | Production | 2 Events.

You can use the ‘Set Default’ option(the refresh button) to revert back to the default format.

Mail (SMTP)

Use this action to specify the SMTP server address and the "from" address of the e-mail sender.

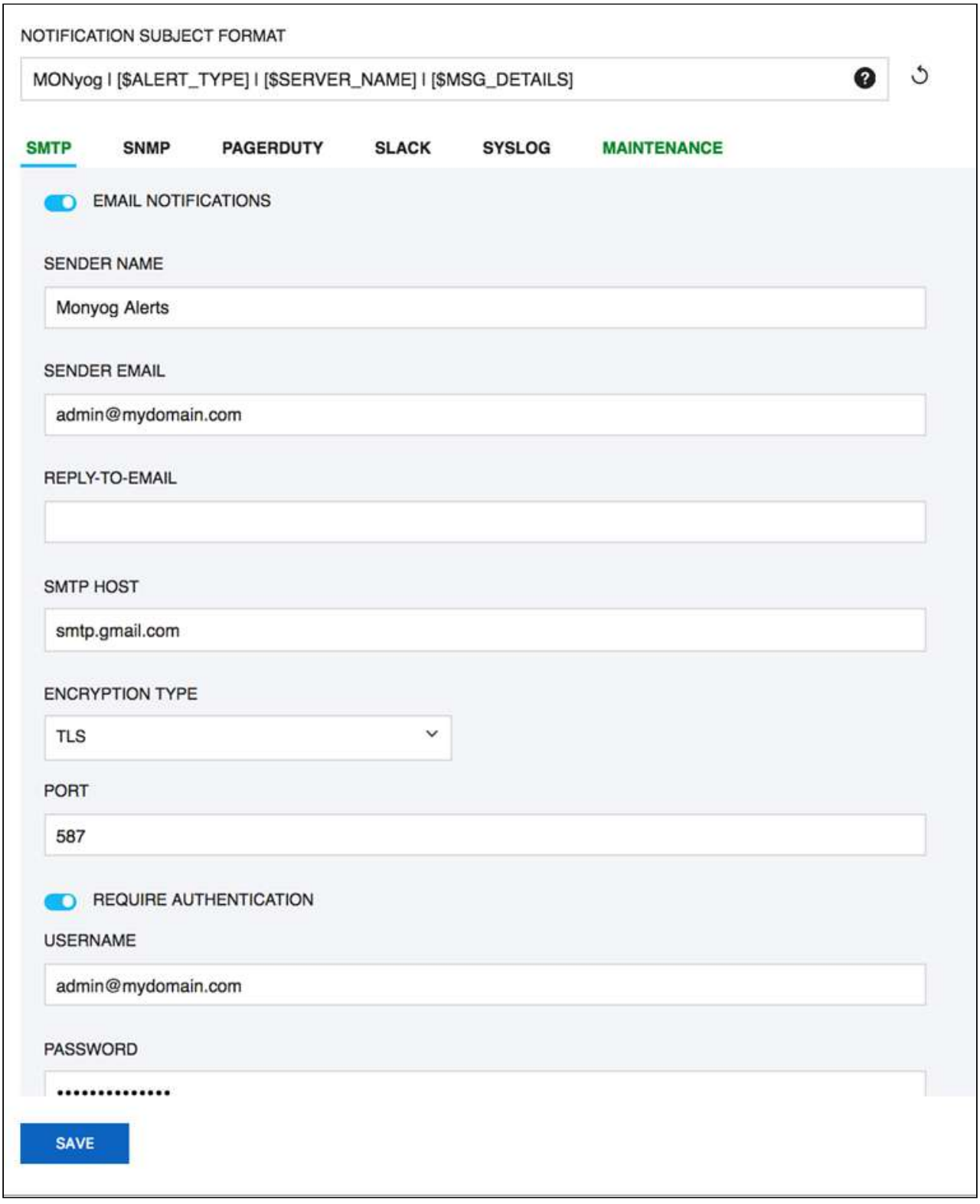
1. Click **Settings**, **Notification & Maintenance**, and select **SMTP**.The SMTP Server Settings window opens. Alternatively, you can click **Configure mail settings** found on the Notifications Settings when registering/ editing the details of a server.
2. **From:** Type the from name and e-mail address with the name and address in the fields Your Name and Your Email respectively. This field must be in standard *name@domain.com* address format unless your relay server is set up to accept default domain from addresses. Most problems involving configuration of the email alert are due primarily to an invalid email address format specified in the FROM/TO fields.
3. **Reply-to email:** Enter email address of the recipient (or recipients) of the email message. This field must be in standard *name@domain.com* address format unless your relay server is prepared to accept the default domain from addresses.

To add multiple recipients, specify multiple recipients by simply separating the email addresses with a comma ",". Example: *me@here.com, you@there.com, somebody@somewhere.com.*

4. Type the **SMTP** server address. You can enter the host name, including domain, or the TCP/IP address. Example Gmail SMTP address is "smtp.gmail.com".
5. **Encryption:** Select the type of Encryption - SSL/TLS mail encryption which is now supported for mail alerts.
6. **Port:** This field signifies the TCP port on which host/IP addresss should connect in order to deliver the message. By default, this field is set to 25 (SMTP). However, some internal SMTP servers may be setup on non-standard ports which will require that this field be changed to match the listening port of the mail exchange. For example Gmail listens to port 465 for SSL encryption and 587 for TLS encryption.
7. **Username and Password**: When sending mail through an authenticated SMTP server, you can fill in the User and Password fields appropriately. If you are not using a secure SMTP server, an error 5xx unrecognized command may be returned from the server if you enter anything into one of these fields.

Most problems involving configuration of the Email alert are due primarily to an invalid email address format specified in the FROM/TO fields. Please refer Registering servers[42] for more information.

42 http://wiki.idera.com/x/dgEGBg

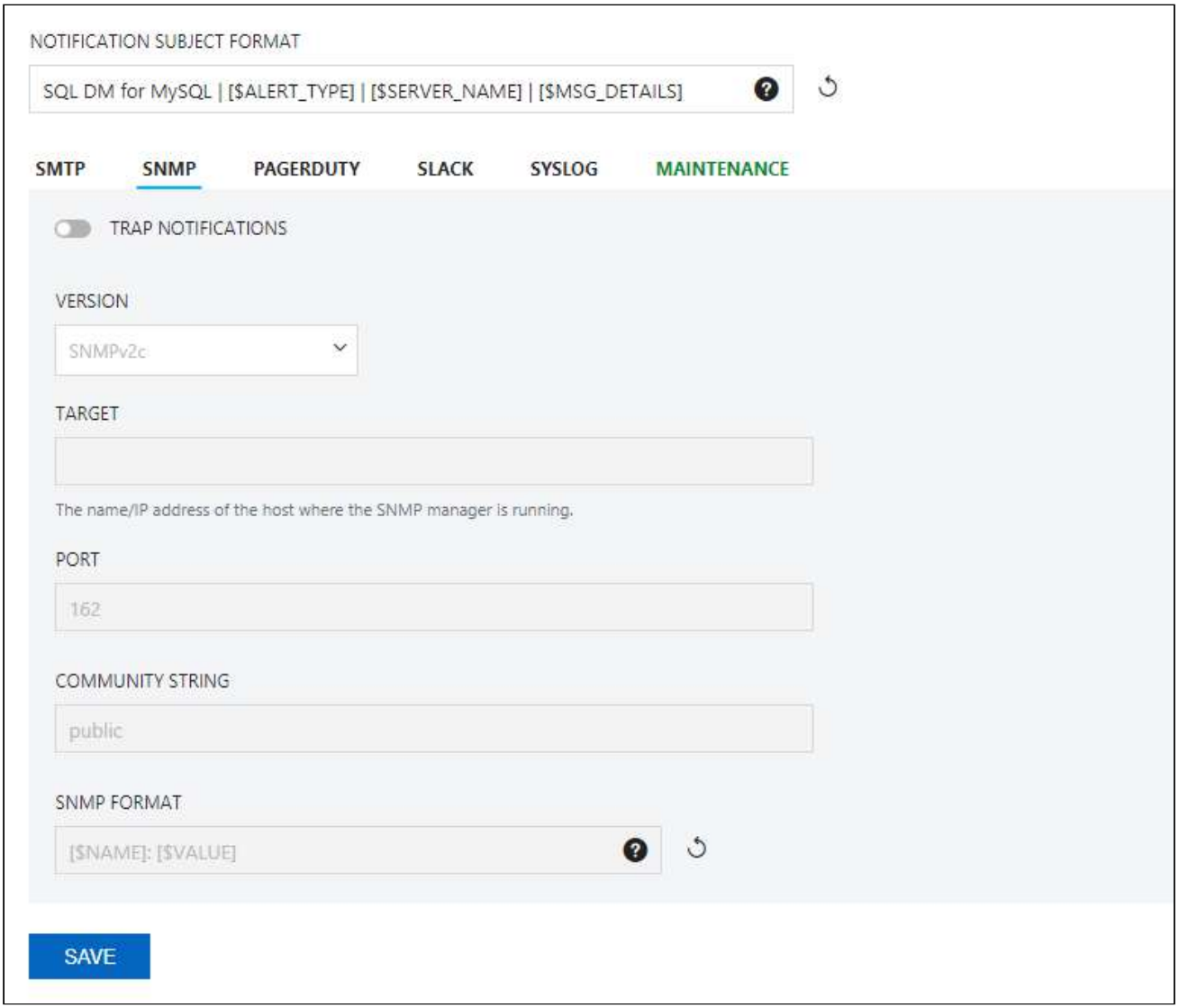

NOTIFICATION SUBJECT FORMAT
MONyog | [$ALERT_TYPE] | [$SERVER_NAME] | [$MSG_DETAILS]
SMTP
SNMP
PAGERDUTY
SLACK
SYSLOG
MAINTENANCE
EMAIL NOTIFICATIONS
SENDER NAME
Monyog Alerts
SENDER EMAIL
admin@mydomain.com
REPLY-TO-EMAIL
SMTP HOST
smtp.gmail.com
ENCRYPTION TYPE
TLS
PORT
587
REQUIRE AUTHENTICATION
USERNAME
admin@mydomain.com
PASSWORD
SAVE

SNMP

Use this action to specify the SNMP settings and options available for use with SQL DM for MySQL.

Click **Settings -> Notification & Maintenance -> SNMP** to navigate to the SNMP settings page.

- **Version:** SQL DM for MySQL supports 2 types of SNMP versions: SNMPv1 and SNMPV2c. You can select the trap type from this drop down.
- **Target:** Type in the name or IP address of the host to which you want SQL DM for MySQL to direct traps.
- **Port:** Enter the destination port here; by default, it is 162. This is the port where your SNMP Manager (or Client) listens for traps.
- **Community string:** The read/write community string helps classify your SNMP operations. By default, SQL DM for MySQL sets it to Public, but you can change to whatever suits your need.
- **SNMP Format:** Specify SNMP trap format e.g. [$NAME]:[$VALUE].
- **Enable status traps:** Status traps are sent when SQL DM for MySQL is starting up to indicate just that. If you want to be informed of when SQL DM for MySQL is starting up, select **Yes**.
- **Use the remote MySQL Server host IP as the SNMP trap agent address for Monitor traps**: Check this option if you want the sender IP of the traps sent by SQL DM for MySQL to be that of the host where the monitored MySQL server is running, instead of the host where SQL DM for MySQL is running. This only affects the Monitor traps sent by SQL DM for MySQL.
- Clicking the **Send Test Trap,** results in sending a status trap to the target and port specified, containing a string that indicates that this is a test trap.
- To have your SNMP client decode the arcane digits identifying a trap, you need to load **SQL DM for MySQL's Management Information Base (MIB)**s into your SNMP client. The MIB file is available in the installation directory of SQL DM for MySQL. You can also download the file from the link provided in SQL DM for MySQL browser interface.

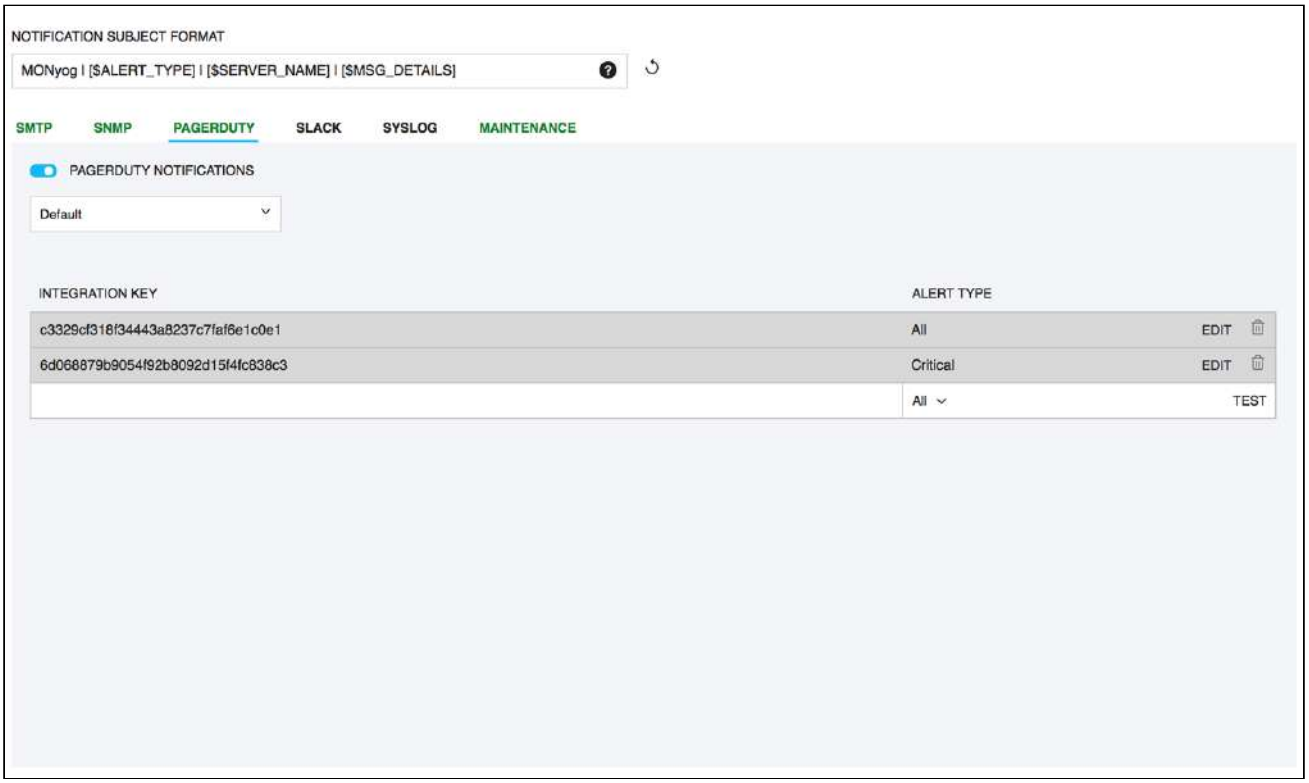


Note more information on types of Traps:

- **Monitor Traps:** Traps sent when one of the Monitors reaches an alert condition. Just like the notification emails, it contains details about the faulting Monitor. Monitor traps are sent when the Send notifications over SNMP option is enabled, and the corresponding counters have mail alert enabled through the MOM.
- **Status Traps**: Sent when SQL DM for MySQL starts up (if status traps are enabled) indicating just that, i.e. "SQL DM is starting up". Also, clicking **Send Test Trap** sends a status trap with the string, "This is a test trap! If you see this message, you have correctly configured SNMP for SQL DM."

PagerDuty

Click **Settings -> Notification & Maintenance -> PAGERDUTY** to navigate to the PagerDuty settings page.



PagerDuty integration allows you to route alerts to your PagerDuty service.

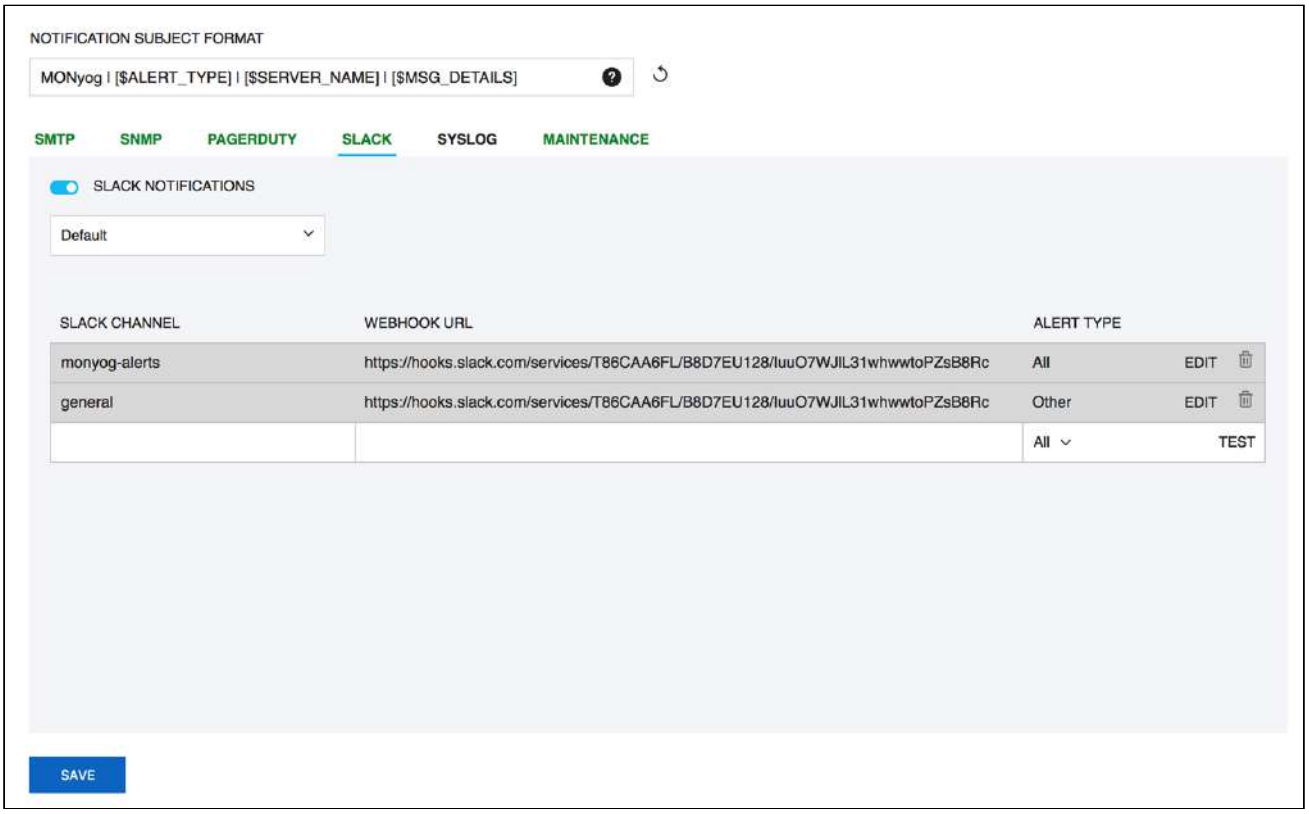
Use the integration key of the service that you want SQL DM for MySQL alerts to be sent to and select the type of alerts to be sent out. Click **Test** to verify whether the integration is configured successfully.

Also, you can create multiple routing rules based on the requirement for different servers registered with SQL DM for MySQL. To create a new rule select **CREATE NEW RULE** option in the drop down.

> ⚠ Make sure that the integration key that you add has the integration type as **API**.

Slack

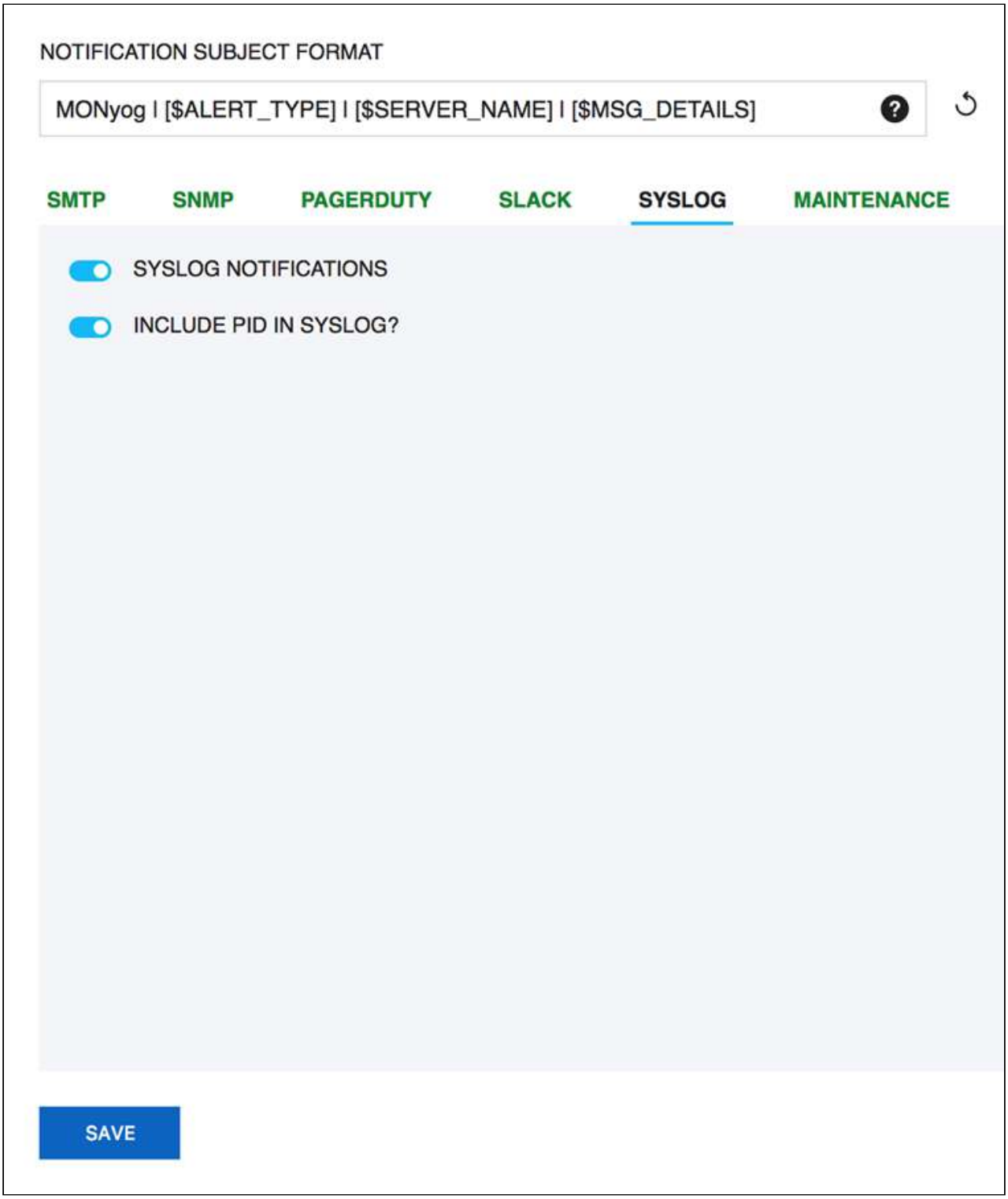
Click **Settings -> Notification & Maintenance -> SLACK** to navigate to the SLACK settings page.



You can Integrate Slack to SQL DM for MySQL to receive the alerts and notifications from SQL DM for MySQL to your Slack channel.
Enter the name of the Channel that you want SQL DM for MySQL alerts to be sent to in the SLACK CHANNEL field, enter the Incoming Webhook in the WEBHOOK URL field and Select the desired Alert Type. To ensure that the rule is configured successfully, try out the rule by clicking **Test**.

SYSLOG

Click **Settings -> Notification & Maintenance -> SYSLOG** to navigate to the SYSLOG settings page.



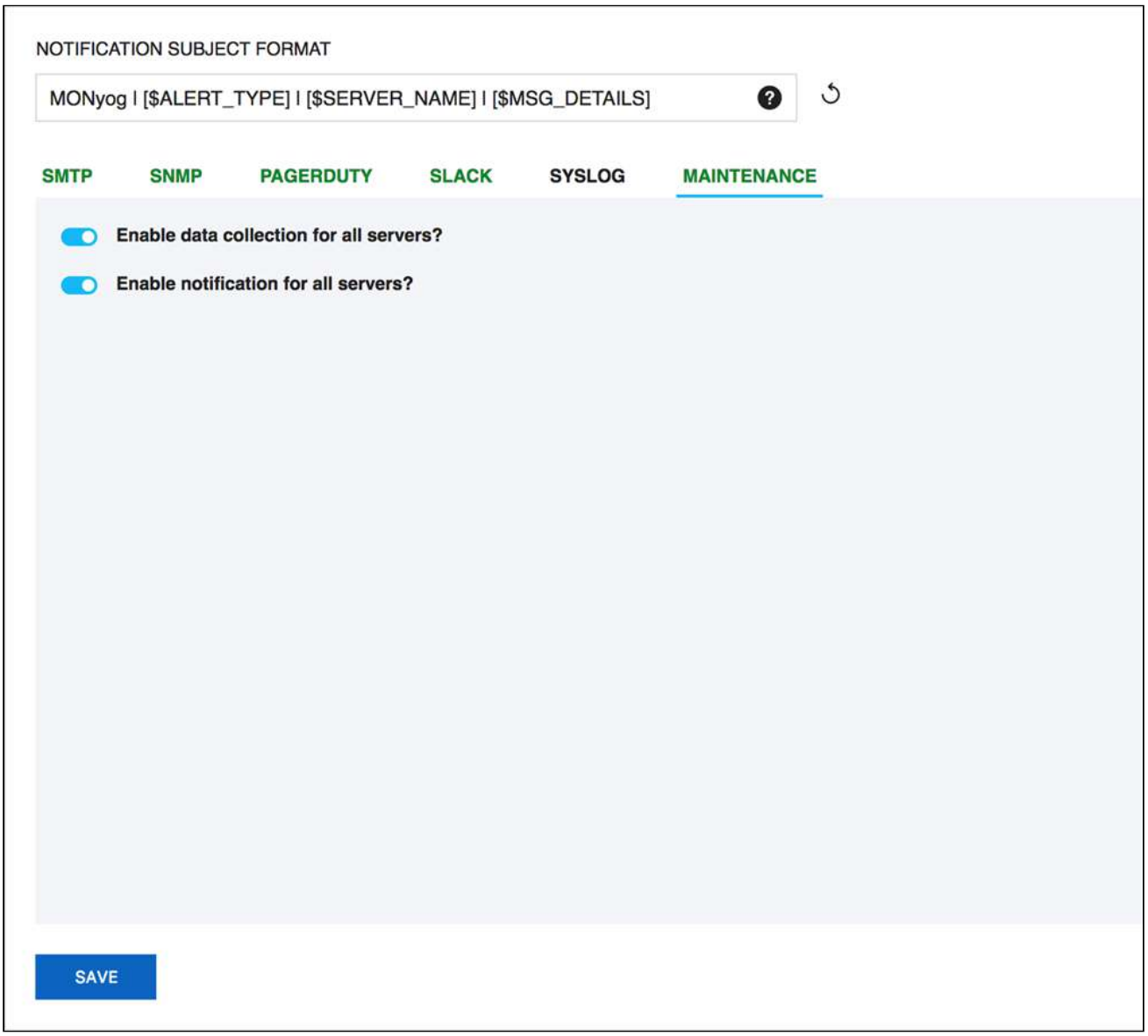
On enabling this option SQL DM for MySQL can write out the alerts and notifications to the Syslog (of the machine where SQL DM for MySQL is installed).

⚠ This is applicable only for Linux installations.

Maintenance

Using this Maintenance option, you can enable/disable data collection from and/or alerting about all servers.

You can enable/disable data collection option and alert for all the registered server using **Settings -> Notification & Maintenance -> Maintenance.**



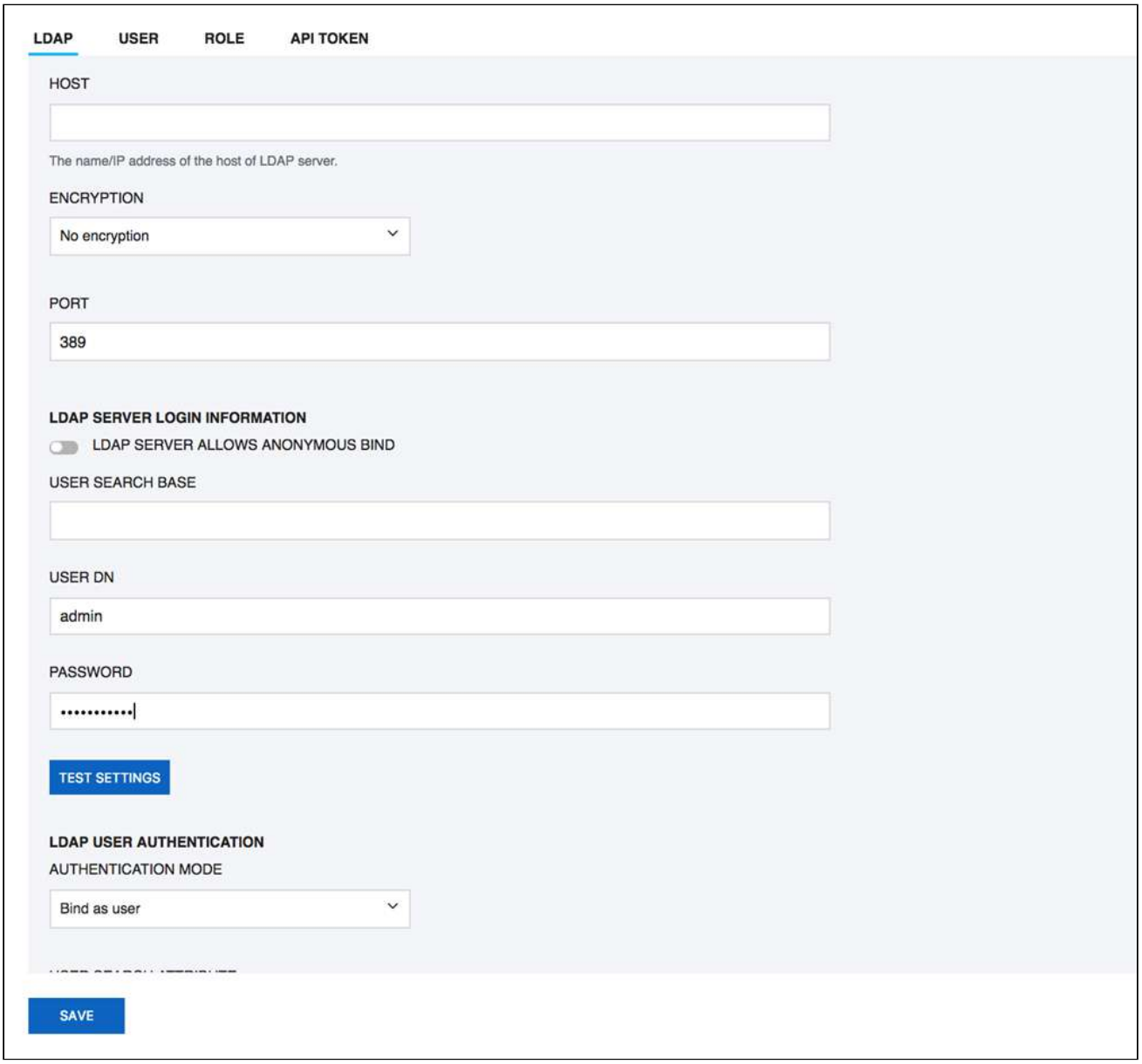
LDAP, Users, Roles, & API Token

This page allows user to add LDAP authentication, create users, create different roles, and also to generate a new API token.

LDAP Settings

When logging into SQL DM for MySQL from the browser interface you may use authentication provided by LDAP server (including the Microsoft/Windows LDAP 'dialect' known as 'Active Directory'). In this case users need not to know SQL DM for MySQL authentication details directly, but only how to authenticate to the LDAP server. To use LDAP authentication for SQL DM for MySQL, specify settings as below:

1. Click **Settings**
2. Select **LDAP, Users, Roles & API Token,** the LDAP setting page opens, displaying the following options:
    - **Host:** Enter the hostname, IP address or URI (Uniform resource identifier) of your LDAP directory server.
    - **Encryption:** Select the type of encryption required for communication with the LDAP directory server. Supported encryption methods are None, StartTLS, and SSL(ldaps).
    - **CA CERTIFICATE:** If your encryption mode is StartTLS and SSL(Ldaps), then paste the content of your digital certificate issued by CA.
    - **Port:** Type in the port your LDAP directory server uses.
    - **LDAP server allows anonymous binds:** Select this option if your LDAP directory server allows anonymous binds to the server.
    - **User DN:** Enter the distinguished name of the entry to bind to the LDAP directory server.
    - **Password:** Enter the password of the User DN specified for binding the user to LDAP directory server.
    - **Test Settings:** Click **Test Settings** to use the mentioned **User DN**, **Password**, and binds with the LDAP directory server.
    - **Authentication mode:** Select the type of authentication mode to use for authenticating the user with the LDAP directory server. **Bind as User** binds user to LDAP directory with the password provided at login in SQL DM for MySQL interface. Authentication via Comparison is done by comparing the user credentials provided at login with the LDAP directory.
    - **User search base:** Type in the User search base filter for the object class you want to filter your users for authentication.
    - **User search attribute:** Enter the attribute name that contains the user name.
    - **Search entire subtree**: This option controls the search for objects specified in user search base. Selecting this option searchs the entire subtree of **User search base**.



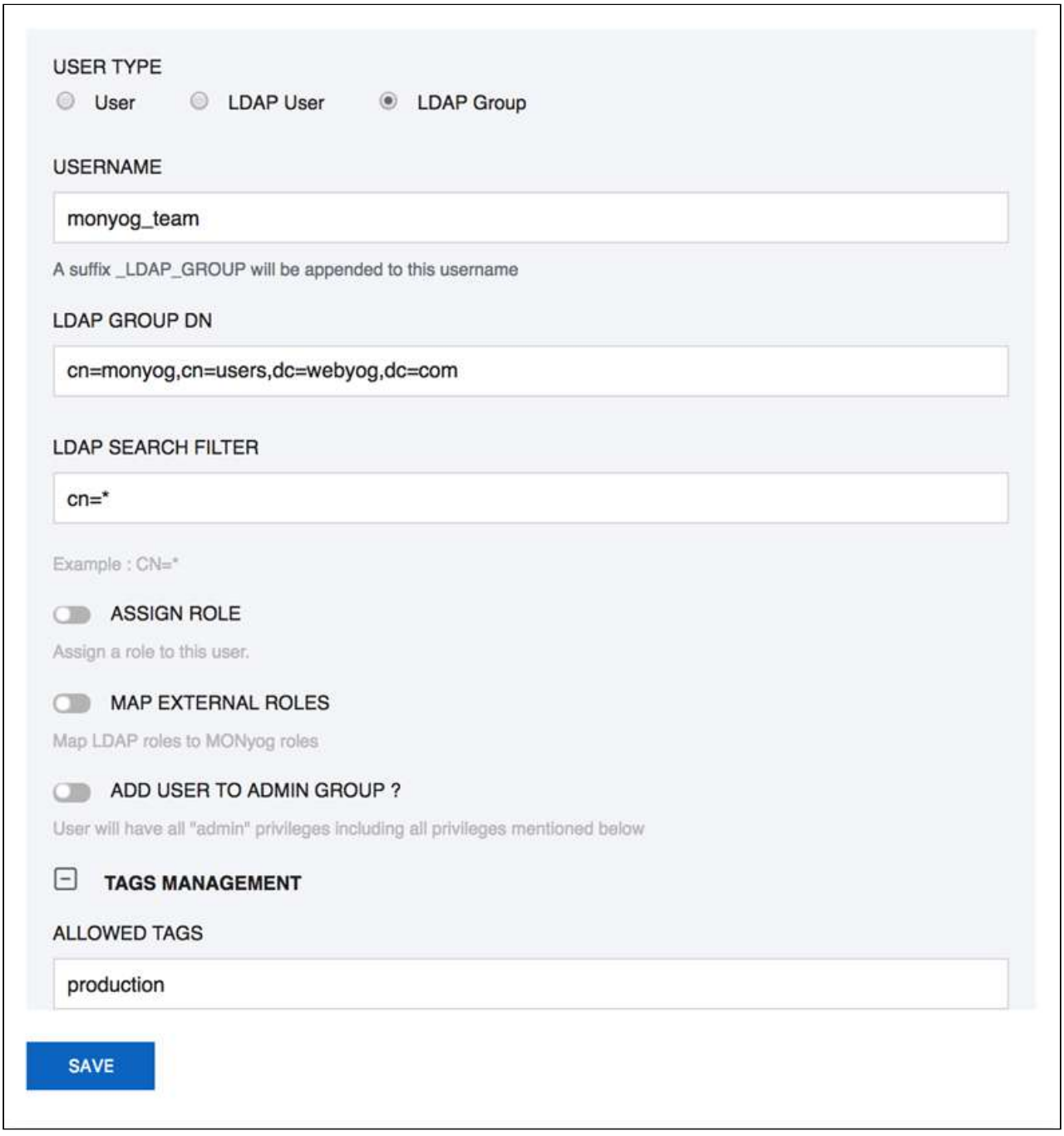
User Management

Using this option User Management, allows you to create, edit, and delete users.

How To Create User?

1. Click **Settings,** and select **LDAP, Users, Roles & API Token**. The window opens where you can create and delete users.
2. To create a new user under the USER tab, click **Add user**, add username, and password in the appropriate fields.
3. To add LDAP group, select **LDAP Group** from the options and specify Username, LDAP group DN, and LDAP search filter.
4. **Assign Role:** Select this option to assign SQL DM for MySQL role.

5. **External Roles:** Use this option to Map LDAP roles to SQL DM for MySQL roles.
6. **Add user to Admin group:** You can refer Managing multiple users(see page 194) for further more information.
7. **Action management:** Use this option to give different privileges like server edit, kill query, etc.
8. **T ags management:** You can give the list of allowed/disallowed tags to the user.
9. **Tab management**: An Admin user can create other users with restrictions to individual Custom Dashboards and permissions to create New Dashboards.



Managing multiple users

You can manage access to your servers and settings based on your needs using User Management. This feature is useful in creating users who will have limited access to the particular servers - which helps in preventing accidentally killing queries, executing FLUSH STATUS on your MySQL servers, or changing your server settings without your knowledge.

Admin

The SQL DM for MySQL admin user can now create other users having access to a subset of available servers only. Also, the Admin is the only allowed to create, delete server, and user registrations.

The Admin can create users with restrictions to individual Custom Dashboards and permissions to create New Dashboards.

Non-Admin

Following restrictions applies to non-admin users:

- Cannot register a new server.
- Cannot delete a registered server.
- Cannot change tags of a server.
- Can edit a server only if "Server Edit" permission is granted.
- Can kill a query from the 'Processlist' page only if 'Kill Query' permission is granted.
- Can execute 'FLUSH STATUS' from the Monitors page only if 'FLUSH STATUS' permission is granted.
- If no 'Allowed tags' are specified, normal users will have access to servers with no tags only.
- If the same tag is specified in 'Allowed tags' as well as 'Disallowed tags', then the user will not have access to servers with that tag.
- Cannot change user settings (except own password).
- Cannot change Preferences.

Permissions

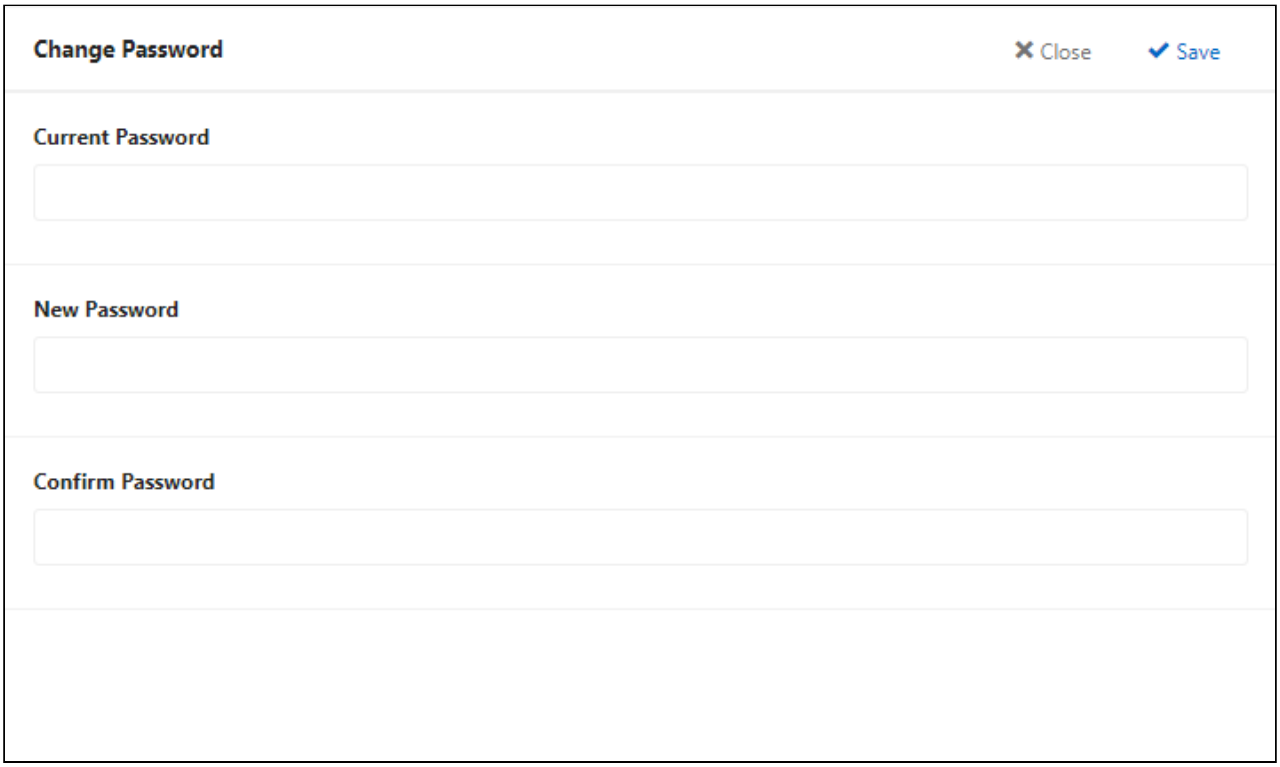
A user can be granted a combination of the following permissions:

- **Server Edit:** Allows the user to edit the settings of servers accessible to him/her.
- **Kill Query:** Allows the user to kill queries through the 'Processlist' page on servers.
- **FLUSH STATUS:** Allows the user to execute the FLUSH STATUS command on servers.
- **View Literals in Queries:** Allows the users to view literals in the Query Analyzer page.
- **Open/Close alert:** Allows the users to open/close alerts through the "Monitors" and "Events" pages.

Change Password

Users can change their password by using the Change Password option under the User Profile on the SQL DM for MySQL interface. Click the **User Profile -> Change password**.

Enter your old password in the first field. Enter your new password in the second field, and confirm the new password exactly the same way in the third field and save it.

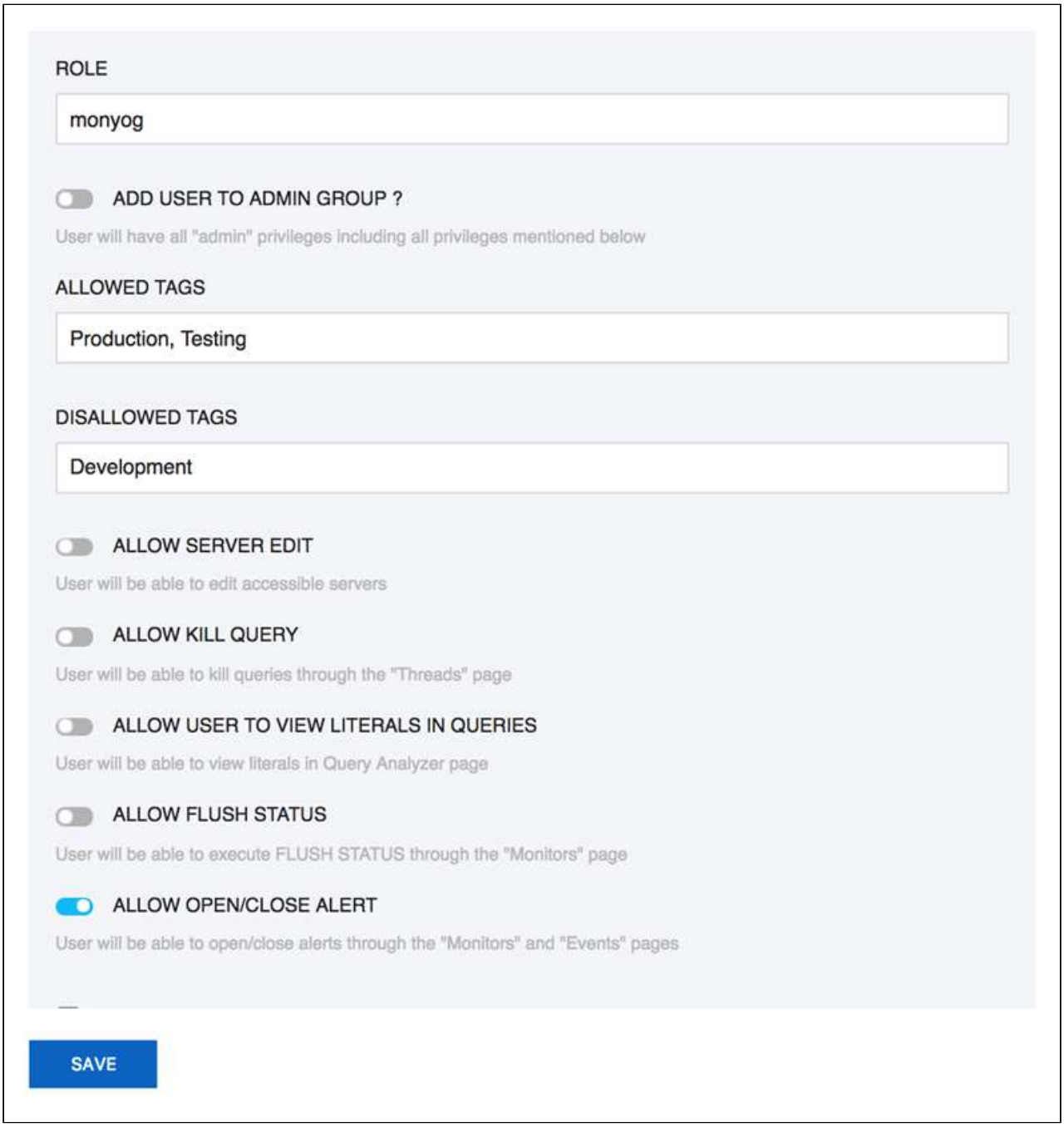


Role Manager

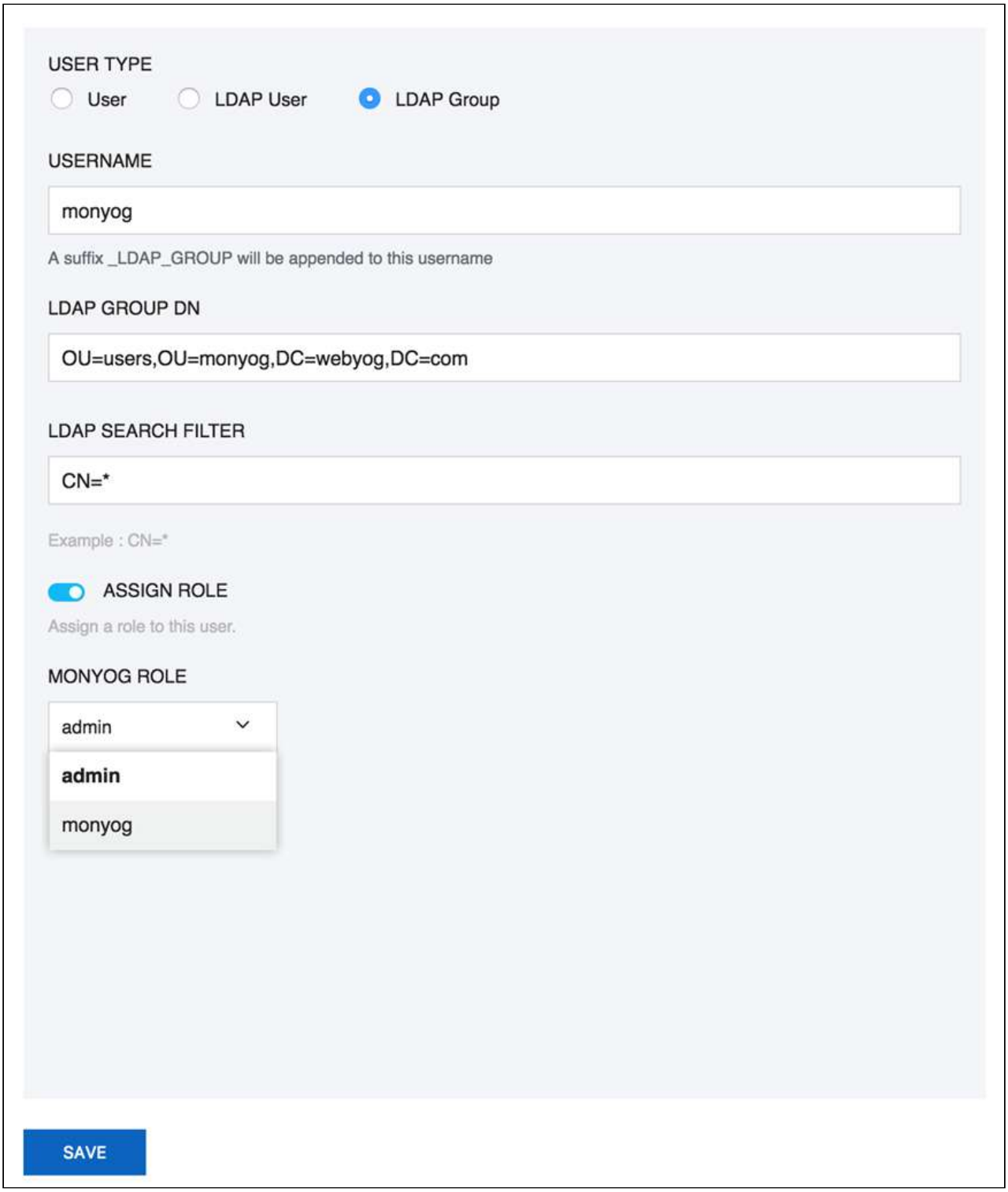
The Role Manager feature allows to create roles in SQL DM for MySQL, which can be then mapped to any users like external LDAP/AD users or the local users created in SQL DM for MySQL. The roles created can then be given different privileges like Allow server edit, Allow kill query, etc. along with the option to restrict access to selected tabs in SQL DM for MySQL.

Creating and Assigning Roles

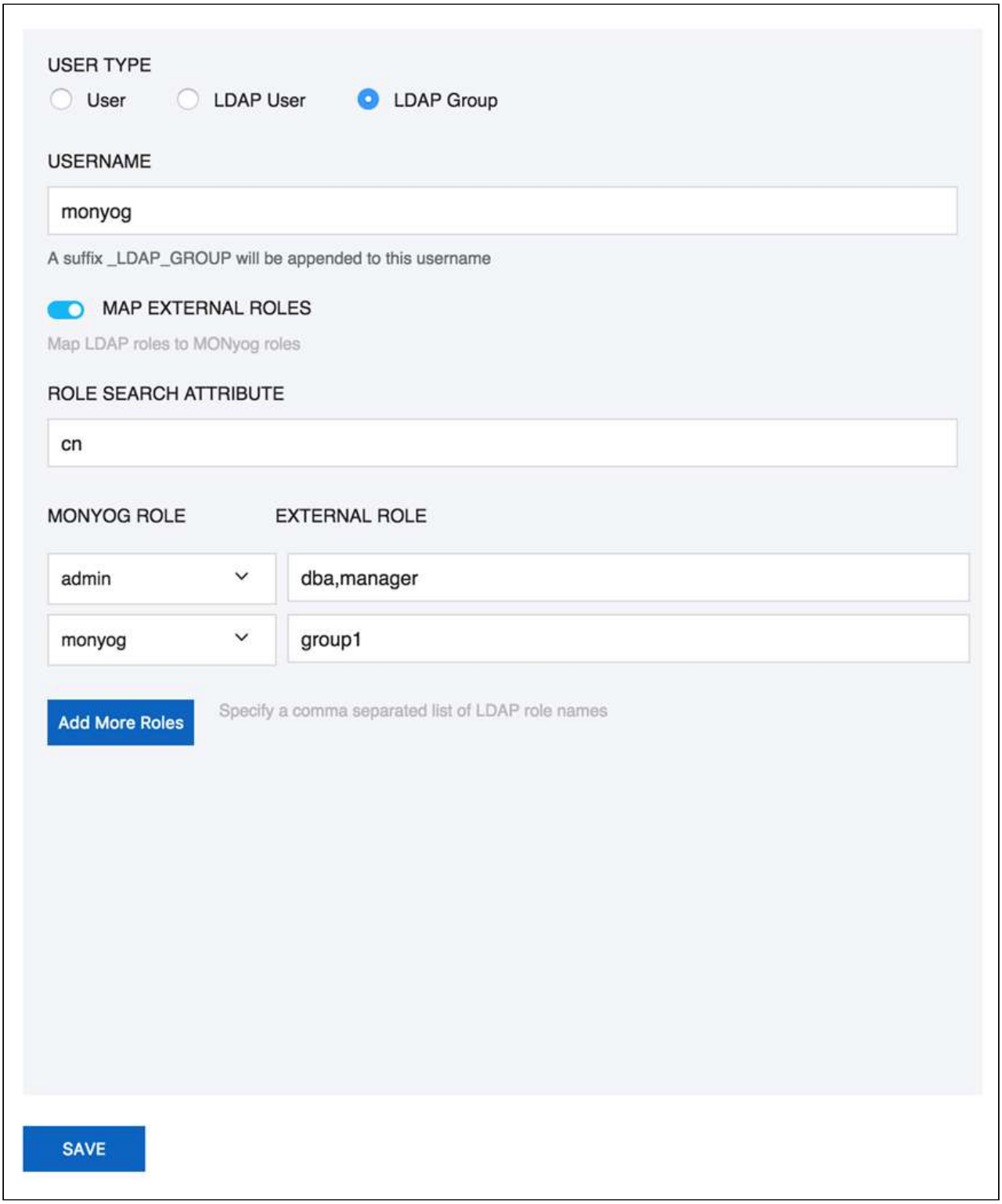
To create a Role in SQL DM for MySQL, go to **Settings**, select **LDAP, Users, Roles & API Token**, and press **Add role.**



Go to **Settings,** select **LDAP, Users, Roles & API Token** to create, edit a user, and assign the created role(s). Select the **Assign Role** option in the Create, Edit user pop up page, and select a role to assign from the drop down menu.



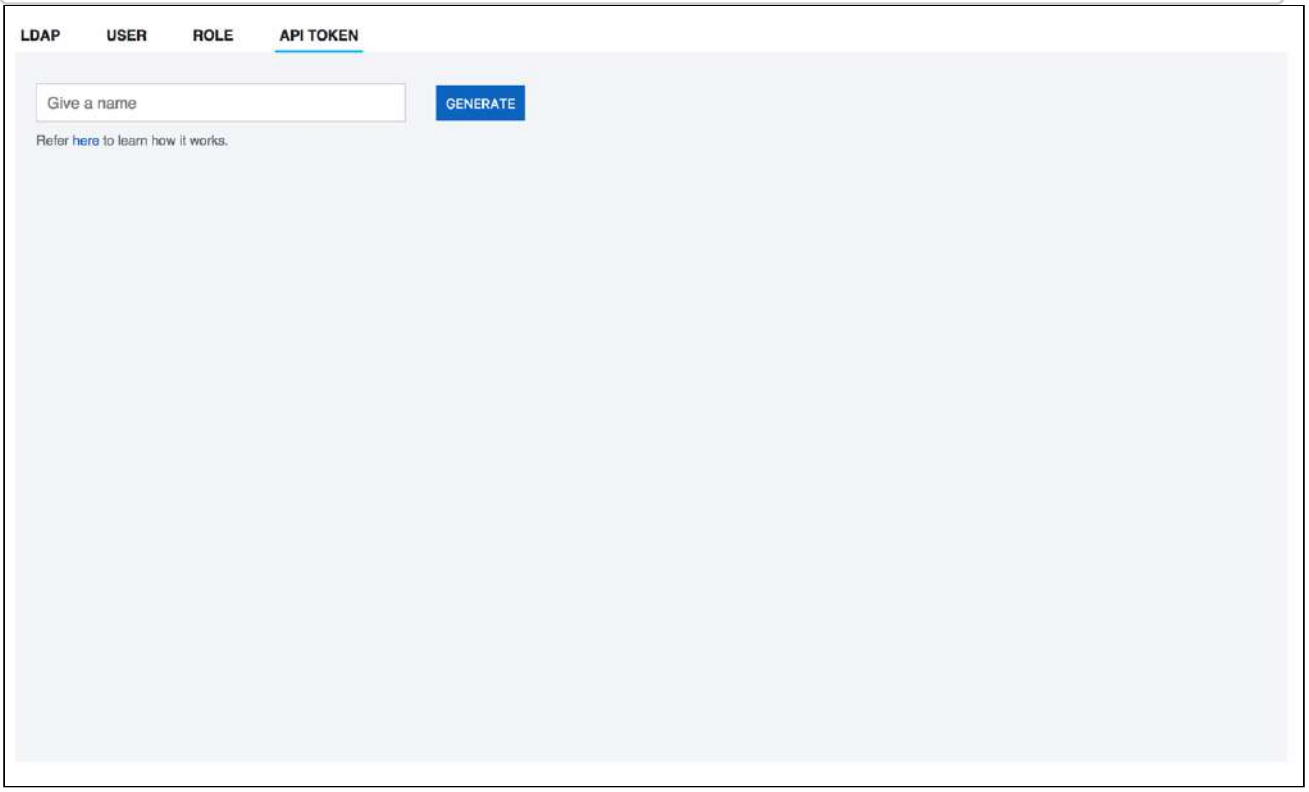
You can Map the LDAP group to the SQL DM for MySQL role created from the Create, Edit user pop up page, and by selecting the option **Map External Roles**. You can specify the comma separated LDAP group names and select the corresponding SQL DM for MySQL role from the drop-down menu.


USER TYPE
User
LDAP User
LDAP Group
USERNAME
monyog
A suffix _LDAP_GROUP will be appended to this username
MAP EXTERNAL ROLES
Map LDAP roles to MONyog roles
ROLE SEARCH ATTRIBUTE
cn
MONYOG ROLE
EXTERNAL ROLE
admin
dba,manager
monyog
group1
Add More Roles
Specify a comma separated list of LDAP role names
SAVE

API Token Manager

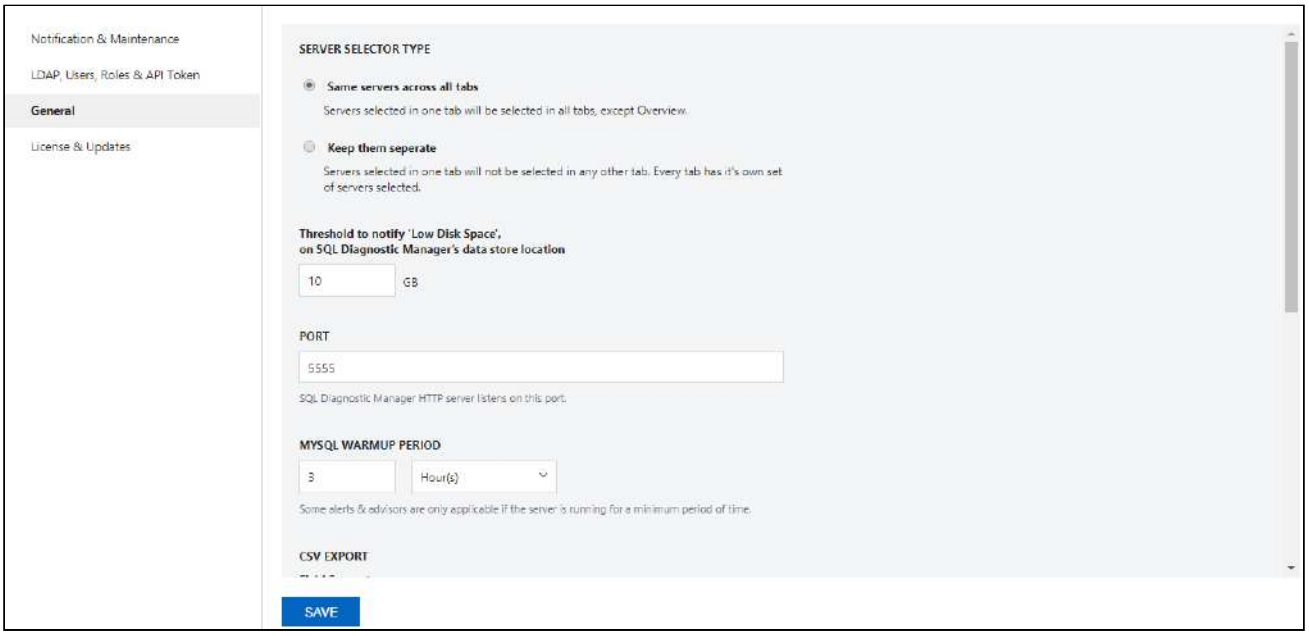
This gives an option to generate token in SQL DM for MySQL and use it in API as an alternative to user and password. This feature is available only for Admin users in SQL DM for MySQL. Admin can create multiple tokens for different purposes; like revoke or delete it from inside SQL DM for MySQL whenever required. This also helps to not share the password with anyone else as well as save it from getting logged in some logs. After clicking **GENERATE NEW TOKEN**, the user should give a name which is associated with the generated token. The generated token can be used in SQL DM for MySQL API in the following way:

```
curl -H "X-MONYOG-TOKEN: 1234567890abcedfghijklmnopqrstuvwxyz"
      "http://192.168.1.1:5555/?_object=MONyogAPI&_action=DataCollection
      &_value=enable&_server=Production001"
```



General

These options provide you to set threshold for the Disk space available to SQL DM for MySQL, choose the server selector behavior in SQL DM for MySQL, and change the port for SQL DM for MySQL GUI. You can also change the settings for MySQL warmup time, exporting CSV files, maximum query length fetched by SQL DM for MySQL, and edit the colors of the charts displayed in SQL DM for MySQL.



SERVER SELECTOR TYPE

There are two options to switch between the behavior of Server Selector:

- **Same servers across all tabs:** If you enable this option, a server selected on one page makes it the default selected server selected across all pages in SQL DM for MySQL (excluding the Overview page). The Overview page has its own separate server selector applicable only for the page.
- **Keep them separate:** This is the default selected option. If you enable this option, a server selected on one page makes it selected only for that page, i.e, each of the pages in SQL DM for MySQL has its own server selector.

SQL DM for MySQL Disk Monitoring

Low disk space is a critical factor for any application. It can cause the application to perform poorly and make it difficult or unable to install the software upgrades.

SQL DM for MySQL sends SNMP trap, email alerts to all the email-IDs if the disk space where SQL DM for MySQL is installed and is running out. User can free up some disk space before the system goes out of space and shuts down. Thus maintaining high-availability of SQL DM for MySQL.

How to monitor disk space with SQL DM for MySQL?

To monitor the disk space, you need to set the threshold for the free disk space. SQL DM for MySQL sends emails to IDs mentioned under the notification settings of all servers. Alerts are also sent to email IDs provided in the SMTP settings.

The threshold limit set denotes is for the disk space available in the Data directory of SQL DM for MySQL. In case the installation directory and data directory are stored in different locations, SQL DM for MySQL alerts if the space available to the Installation directory gets below 500MB.

SQL DM for MySQL monitors the disk space on a 5-minute interval in stable conditions and 1-minute interval when the disk space goes below the threshold value.

The alert notifications are sent every 6 Hours if the disk space remains below the threshold value.

PORT

Using the **Change port** option, you can change the SQL DM for MySQL webserver port. By using this option, you can define on which port the SQL DM for MySQL (web) server listens. This port must be specified when connecting to SQL DM for MySQL from a browser.

Users can change the port from **General > PORT**. Once the port is changed, the user should restart the SQL DM for MySQL service for it to take effect.

MySQL Warmup Period

Using the Change MySQL Warmup Time option, you can change the warm-up time for MySQL servers. Some alerts and advisors are only applicable if the server is running for a minimum period of time.

1. Users can change the MySQL Warmup Time time by using the user settings screen, choose **General > MySQL Warmup Period**.
2. The drop-down can be used to display options like seconds, minutes, hours, days, etc.

CSV Export

Use this option to define the field delimiter for CSV export.

Users can change the CSV export settings by using **General > CSV EXPORT**. You can define the field delimiter for exporting as CSV, and the default is set to comma " , " but you can specify any other delimiters if you need.

Maximum Query Length

Settings -> Maximum Query Length defines the maximum query length displayed. Default is 10000 characters and max Value is 64000 characters.

The maximum query length setting here applies to queries fetched in processlist mode. In Performance Schema mode, the maximum query length is defined by the MySQL global variable performance_schema_max_digest_length. The default value is 1024 bytes and the maximum value allowed is 1048576.

> ⚠ This variable was added in MySQL 5.7.8. In MySQL 5.7.6 and 5.7.7, use max_digest_length instead. Before 5.7.6, the value cannot be changed.

Chart Color

Use this option to set chart colors of your choice.

License & Updates

This options helps you to review your License Details and enable check for updates automatically.

LICENSE

In the License Manager option you can enter your license details Registration name and License key after purchasing a license. Your license key controls how many servers are able to register in SQL DM for MySQL. If you are using a TRIAL you can register up to 512 servers, but after TRIAL expiration, a license key is required and only the number of MySQL servers that you have license for will be available. So be careful to enter a license key for a proper license before TRIAL expiration. When logging on to a unregistered SQL DM for MySQL instance running in TRIAL period a similar license manager is displayed before you have access to the logon screen. As long as the TRIAL period has not expired you can skip the registration and continue to use SQL DM for MySQL on TRIAL terms.

Enter your License Key by using the **Settings -> License & Updates**. Enter the Registration Name and the key in the appropriate fields.

UPDATES

Using the Update Manager option, you can check for updates manually by clicking the **Check for Updates** button and also you can select 'Yes' so that the SQL DM for MySQL will automatically check for updates once per day.

To receive a notification on SQL DM for MySQL updates automatically, set up in **Settings -> License & Updates**, and enable **Check for updates automatically?** under UPDATES.

SQL DM for MySQL Log

The SQL DM for MySQL Log has detailed records of all sorts of server errors and messages. This can help you tracking down any problems with SQL DM for MySQL.

The SQL DM for MySQL log records these types of events :

- MySQL server errors received.
- MySQL client/API errors.
- SMTP errors.
- SSH server and client errors.

To read the log just click **Settings -> Show Log** tab in the SQL DM for MySQL interface. The log is a plain text file MONyog.log which is stored in MONyog folder:

- Location in Windows 2008 : `C:\ProgramData\Webyog\Monyog\`
- Location in Windows Vista/7: `{System\_drive}:\ProgramData\Webyog\MONyog)`

Changing SQL DM for MySQL log path

The MONyog.log path can be changed to a desired directory from the MONyog.ini file. You just need to add the parameter “MONyogLogPath” in the MONyog.ini file and give the path to the MONyog.log file, for e.g: MONyogLogPath=/home/ubuntu/MONyogLog/

> ⚠ Please stop MONyog(SQL DM for MySQL) service before editing the MONyog.ini file.

Enabling log rotation in SQL DM for MySQL

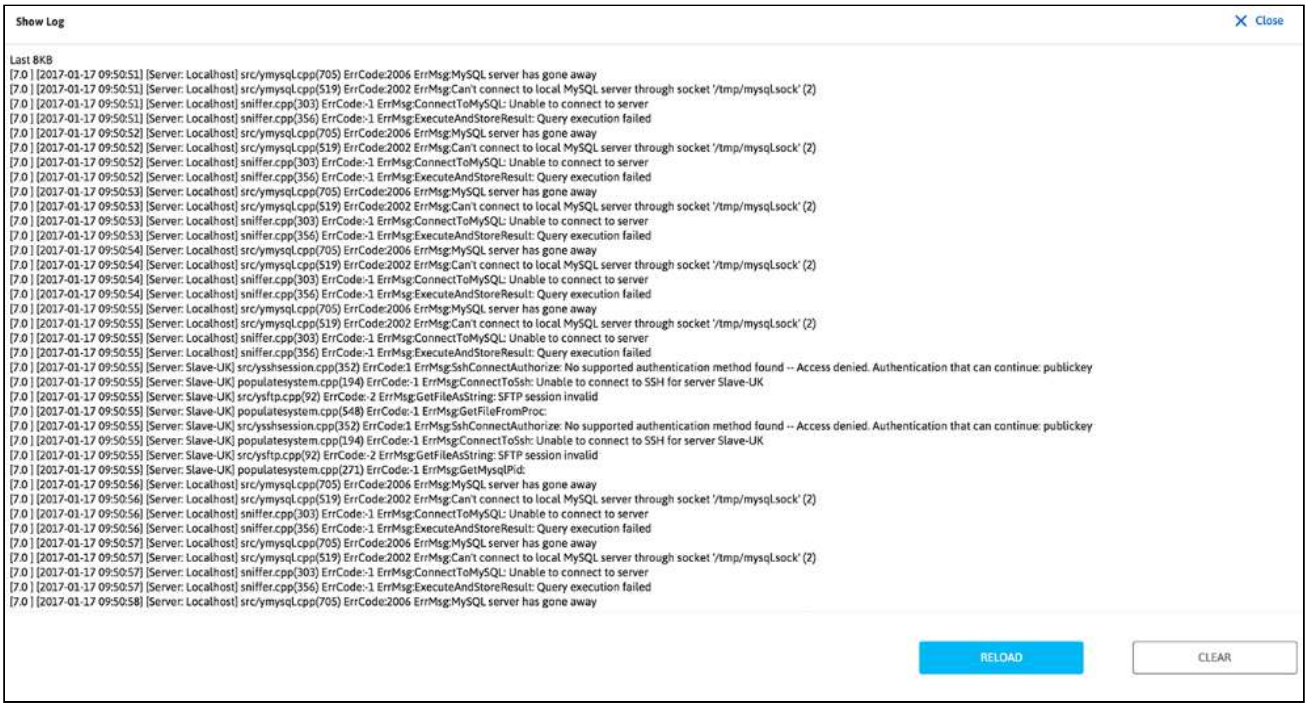
SQL DM for MySQL Linux builds ship with log rotation script to truncate MONyog.log. MONyog-logrotate is available in the installation directory and should be copied to /etc/logrotate.d/ directory.

You can edit and configure the log retention time. By default, the log rotation happens every 7 days.

> ⚠ Log rotation is not available for Windows.

Sample Log

The SQL DM for MySQL log has detailed records of all sorts of server errors and messages. Below, the sample log:



How helpful it is?

A user does not have to keep his attention fixed to the SQL DM for MySQL interface all the time.

It basically logs the critical places and network interfaces problems faced by SQL DM for MySQL. Like, if for some reason Mail Alert has not been send, or MySQL server is down, etc.

Troubleshooting

Troubleshooting

Having trouble? See if we can help you out with your issue:

I am not able to connect to MySQL

The error message: `"Error No. 2003: Can't connect to MySQL server on 'localhost' (or some other host)"`

Simply means that connection is not possible for one of the following (or similar) reasons:

- There is no MySQL server running at the specified host.
- Connection to the MySQL server is not allowed using TCP/IP. Check the 'skip-networking' setting in the MySQL configuration file (my.ini on Windows, my.cnf on Unix/Linux). It shall be commented out like '#skip-networking'. If it is not commented out, then do it and restart the MySQL server for the change to take effect. SQL DM for MySQL needs to connect using TCP/IP.
- When trying to connect to a MySQL server at an ISP this error message often indicates that direct connection to MySQL has been blocked. You must then use SSH-tunneling to connect.
- Some networking issue prevents connection. It could be a network misconfiguration or a firewall issue. We have experienced sometimes that some firewalls (ZoneAlarm in particular) is blocking TCP/IP connections even if it claims to be disabled. Most often it will help to uninstall and reinstall the firewall.

I am not able to view the SQL DM for MySQL home page

Simply means that connection is not possible for one of the following (or similar) reasons:

- Wrong details (port for instance)
- Firewall

I can connect to MySQL on the host but not able to retrieve OS data

SQL DM for MySQL can retrieve OS data from Linux operations systems if SSH shell access is possible or configured to that system. Note that SQL DM for MySQL can do this no matter on what Operating System SQL DM for MySQL itself is installed.

In the Opera browser the red and yellow indicators in 'Monitors' interface do not update correctly.

You need to always configure Opera to check if graphics was updated. **Opera Tools > Preferences > Advanced > History**. Opera alone has this option. This is a performance optimization that may be OK with largely static webpages that do not work properly with SQL DM for MySQL webpages. Set Check Images to Always.

SQL DM for MySQL History/Trend graphics display partly outside visible screen area

On some systems that use a wide-screen monitor, earlier versions of the Opera browser (< 8.5) is not obeying the javascript command to open a new window for the display of a History/TRENDs graph. It opened in an ordinary tab instead. That resulted in the lower part of the graph to be invisible with relatively low screen resolutions (including the popular widescreen resolution 900*1440 (very common with laptops designed for Windows Vista)). You should upgrade Opera.

Getting mysql.sock error

This error comes up because, if you give 'localhost' as the host, mysql client library trying to use the unix domain socket (file based) instead of the TCP one. And, every mysql client library has one path to the Unix domain socket.

- In SQL DM for MySQL's mysql client library it is: '/var/lib/mysql/mysql.sock'
- In "phpMyAdmin", it is: '/var/run/mysqld/mysqld.sock'

So, if one software works the other breaks. To solve this problem review the following:

- You can create a symbolic link to your original '/var/run/mysqld/mysqld.sock' in '/var/lib/mysql/mysql.sock' '/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'

The command to create this is:

```
$ ln -s
/var/run/mysqld/mysqld.sock /var/lib/mysql/mysql.sock
```

This creates a symbolic link '/var/lib/mysql/mysql.sock' to '/var/run/mysqld/mysqld.sock'. So both applications work.

- You can keep '/var/run/mysqld/mysqld.sock' as it is. And, you can force SQL DM for MySQL to use TCP based connection specifying "127.0.0.1" as the host instead of 'localhost'.

Tunneling works for me but I cannot get system counters

I can connect to my FTP server, but could not use that same user to get system counters. Even, I can use the same user in SQL DM for MySQL to tunnel to MySQL server. To collect system counters, the SSH user should have access to a shell. This is specified usually in '/etc/passwd' file. But to tunnel to MySQL server the SSH user does not need a shell access. You can check the shell access which is given in /etc/passwd file.

For FTP users generally shell access will be blocked. A typical FTP user has an entry something like this in the '/etc/passwd' file:

```
john:x:10009:10001:Jonathan:/var/www/vhosts/yourdomain.com:/bin/false
```

A colon (:) is used to separate the fields. All the fields are explained below:

- **john**: The login name.
- **x**: An x indicates that the encrypted password for this user is kept in /etc/shadow file
- **10009**: The unique User ID for this user.
- **10001**: The primary Group ID for this user.
- **Jonathan**: This is the User ID info. Generally comments, full name of the user, etc.
- **/var/www/vhosts/yourdomain.com**: The home directory for this FTP user.
- **/bin/false**: This field indicates the shell access given to the user. And you can see that no shell access has been given to 'john'. It is /bin/false.

This user 'john' can not use a shell. That is the reason 'john' can not be used for collecting system counters. But user 'john' can be used for tunneling to MySQL server.

A typical user entry with shell access has an entry in /etc/passwd something like:

```
jenny:x:10002:10003:Jenniffer:/home/jenny:/bin/bash
```

Key based authentication does not work with SQL DM for MySQL

Please note that for key based authentication SQL DM for MySQL supports only OpenSSH specified standard key format for public and private keys. SQL DM for MySQL does not work with the keys generated by other SSH related products. This list includes but not limited to key pairs generated by:

- Puttygen
- SecureCRT

MONyog-bin not found on MONyog START command

There may be two reasons for this error:

1. Path is invalid. When using the .gz-compressed build for Linux you start the MONyog(SQL DM for MySQL) service like "{path to} MONyog START". If you are executing from the MONyog folder yourself you need to write "./MONyog START". "./" means 'current folder' and on most Linux 'current folder' is not in PATH environmental variable, so you have to specify it.
2. You are trying to run a 64-bit build on a 32-bit platform or vice-versa. The MONyog-bin file is not recognized by the OS as a valid binary. You should use the 32-bit build on 32-bit OS's and the 64-bit build on 64-bit OS's (however support for 32 bit binaries may be configured on 64 bit Linux's, but often it is not the case with DEBIAN based Linux distros - including (k/x)Ubuntu's).

'Test Path' button in Query Analyzer settings keeps throwing 'File Path Invalid' error

Problem: I have a log in a shared folder. I am able to access that log myself, but press **test path** button in Query Analyzer settings keeps throwing 'File Path Invalid' error.

Solution: By default SQL DM for MySQL installs on Windows with the privileges of the local system account. It does not automatically give you access to shared folders located on other systems. In the Windows service manager (Control Panel -> Administrative Tools -> Services) locate the MONyog(SQL DM for MySQL) service and select

Properties from the context (right-click) menu. In the Log On tab, select an account that has sufficient privileges to access the shared folder on the remote system.

Authentication problem with key authentication

To resolve an error like the following:

"Failed to connect to SFTP: Error: offering public key failed, access denied, authentications that can continue: ...." or similar, check the following workaround:

1. Ensure that whatever the public key content you pasted in authorized_keys is same 'byte by byte' that you paste in SQL DM for MySQL public key field.
2. Ensure that you are trying with the same username. This is a common mistake to add keys for one user authorized_keys and trying to connect with another user.
3. The authorized_keys file should have permission 600. That is, read/write permission only to the owner.
4. Finally, ensure that the keys are standard OpenSSH keys not any proprietary or application-specific format.

When I fetched Details from MySQL, Log file Path is not showing?

Log file path could be fetched if the MySQL server you are registering is greater than 5.1.6. If your server is lesser than this, then all fields would be read only and you have entered the slow query log/general query log path yourself.

CSV file is not displaying correct results for "Query Execution Time" columns

Say for example, in the Query Analyzer Page, the Total column value is 2:16:5.282 but in CSV it is displayed as 16:5.282.

This is a formatting problem in Excel instead if you open the export in Notepad or any other editor this problem will not persist. To view in Excel follow these steps:

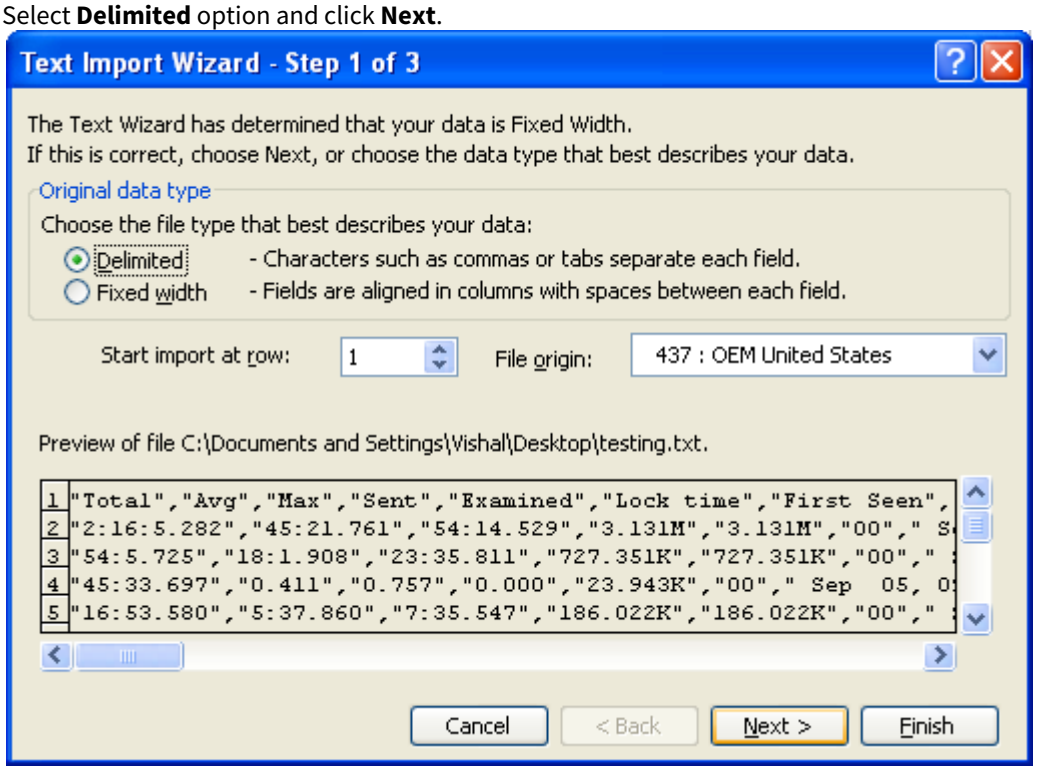
- Save the .csv file on disk.
- Import the .csv file in Excel as explained below. (Note: Set the Data Format for columns containing time values to Text).

How to import a text file into Excel?

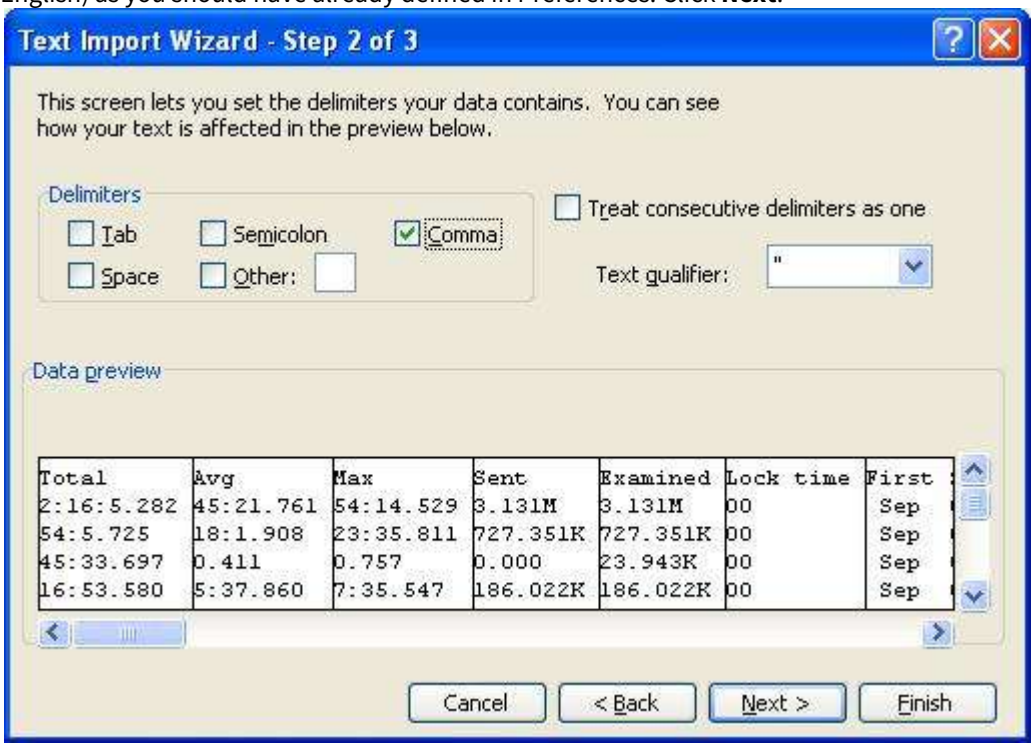
After you have started Excel (this FAQ uses Excel 2003), follow these steps:

1. On the Data menu, point to **Import External Data**, and then click **Import Data**.
2. In the Files of type dialog, click **Text Files**.
3. In the Look in list, locate and double-click the text file you want to import.

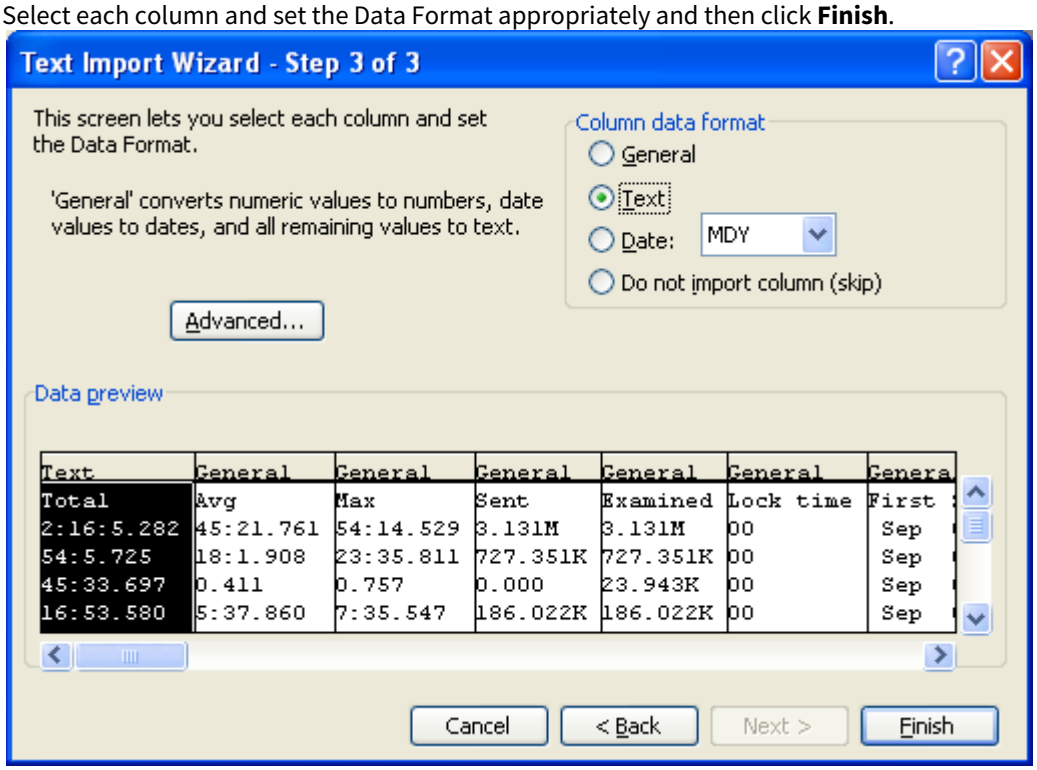
4. Select **Delimited** option and click **Next**.



5. Set Delimiter to your locale-specific delimiter (COMMA " , " for English and SEMICOLON " ; " for most non-English) as you should have already defined in Preferences. Click **Next**.

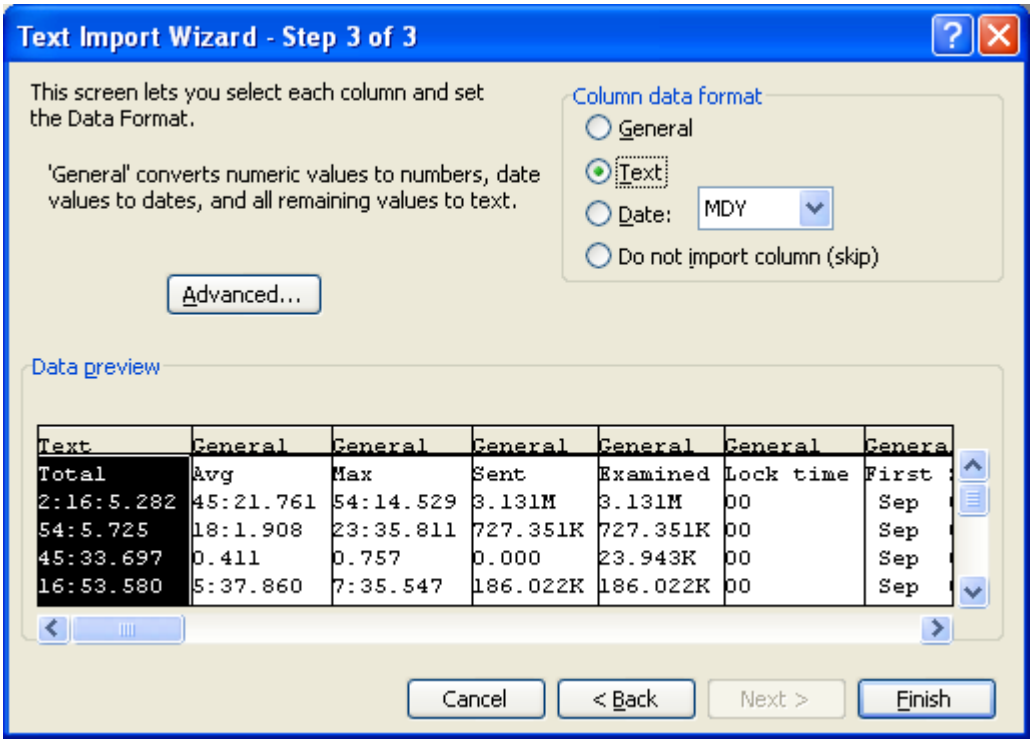


6. Select each column and set the Data Format appropriately and then click **Finish**.



In the Import Data dialog, either one of the following options should be performed:

- To return the data to the location you selected, click **Existing worksheet**, and then click **OK**.
- To return the data to a new worksheet, click **New worksheet**, and then click **OK**. Microsoft Excel adds a new worksheet to your workbook and automatically puts the external data range in the upper-left corner of the new worksheet.


Text Import Wizard - Step 3 of 3
This screen lets you select each column and set the Data Format.
'General' converts numeric values to numbers, date values to dates, and all remaining values to text.
Advanced...
Column data format
General
Text
Date:
MDY
Do not import column (skip)
Data preview
Text
General
General
General
General
General
Genera
Total
Avg
Max
Sent
Examined
Lock time
First
2:16:5.282
45:21.761
54:14.529
3.131M
3.131M
00
Sep
54:5.725
18:1.908
23:35.811
727.351K
727.351K
00
Sep
45:33.697
0.411
0.757
0.000
23.943K
00
Sep
16:53.580
5:37.860
7:35.547
186.022K
186.022K
00
Sep
Cancel
< Back
Next >
Finish

FAQ

Table of content:

1. SQL DM for MySQL is licensed per server. Does that mean SQL DM for MySQL servers or MySQL servers?

It means MySQL servers. You may install as many instances of SQL DM for MySQL as you like as long as the total number of MySQL servers monitored does not exceed your license.

TOP(see page 211)

2. Do I need to install SQL DM for MySQL on the same host as MySQL?

MySQL servers, SQL DM for MySQL server(s) and Clients (browser) can be installed independently everywhere where a TCP connection (like an Internet/Intranet connection) is available. Available TCP connections is all that is required. Regarding installing SQL DM for MySQL on the same host as MySQL then SQL DM for MySQL Connects to/ Monitors MySQL running on any platform.

TOP(see page 211)

3. What operating system does SQL DM for MySQL require?

Currently we support Windows (do not support Windows 2003) and Linux operating systems. Those are the Operating Systems where SQL DM for MySQL itself needs to be installed in. The SQL DM for MySQL client functionalities only require an Internet browser and any platform (including platforms for handheld devices like mobile phone, PDA, tablet PC, etc.). There are also no restriction as regards the platforms the MySQL servers that SQL DM for MySQL connects to - it can be any. Additionally, SQL DM for MySQL is able to retrieve OS data from Linux Operation Systems.

TOP(see page 211)

4. How do I upgrade SQL DM for MySQL?

This is actually a license-related question and a technical question as well.

- **License:** SQL DM for MySQL ships with 1 year of free upgrades. After that you will be offered an upgrade with discount. Our website always tells the terms and conditions. Also our website has a Portal for registered users from where you can download free upgrades and purchase upgrades after the expiry of the free upgrade period.
- **Technical:** The automatic installers (the Windows version and the RPM build for Redhat type Linux) handles everything automatically. The gz-compressed build for other Linux's requires that you run execute a few installation scripts from a command shell. We constantly improve and simplify this. After extracting the tar.gz package, you get a file called README. Please refer to that file for details.
- **Need to monitor more servers:** You can upgrade anytime your SQL DM for MySQL installation from monitoring a certain number of servers to higher numbers by opening the License Manager and enter a new license key. If you need this please first contact us through our ticket system. We will consider the value of your existing license and compensate you (details depend on what license you have and what you need and how old your existing license is).

> **Note**
>
> Do not forget to backup whatever JavaScript you have edited, as they get overwritten when you upgrade. You can take a back-up of Counters.def and Udo.def located inside SQL DM for MySQL folder: MONyog for this. Alternatively, you can directly upgrade. After upgrade you get your JS changes as a conflict in SQL DM for MySQL UI, you can resolve those and keep your changes.

TOP(see page 211)

5. Why should I upgrade?

Every new release adds features, fixes bugs, improves performance, stability, the GUI interface etc. Why should you NOT upgrade? Refer to Version History for details.

TOP(see page 211)

6. I have installed SQL DM for MySQL. What now? How do I get the reports?

What you installed was the MONyog(SQL DM for MySQL) service. This service is basically a webserver program. You connect to the service with an Internet browser by specifying the host where it is installed and the port that was specified when installing.

TOP(see page 211)

7. How can SQL DM for MySQL 'know all what it does'?

SQL DM for MySQL queries the MySQL servers about almost anything the server 'knows' except for data stored on those servers. The MySQL servers themselves store and maintain records of server configuration, users, history and much more. SQL DM for MySQL retrieves this information, organizes it, calculates on it and reports.

TOP(see page 211)

8. Where does SQL DM for MySQL store data?

This depends on the Operating system.

- Windows 2008:

Data folder:

```
C:\ProgramData\Webyog\MONyog\Data
```

MONyog.ini + MONyog.log + preferences.config are in:

```
C:\ProgramData\Webyog\MONyog\
```

- Windows Vista:

Data folder:

```
Windows Vista:{System_drive}:\ProgramData\Webyog\MONyog\Data
```

MONyog.ini + MONyog.log + preferences.config are in:>

```
Windows Vista:{System_drive}:\ProgramData\Webyog\MONyog
```

- Linux RPM build:

Data folder: The connection configuration and collected data are kept here. You can find directories named like 0001, 0002, 0003, etc.

```
/usr/local/MONyog/data/
```

Installation file:

```
/usr/local/MONyog/MONyog.ini
```

Log file:

```
/usr/local/MONyog/MONyog.log
```

Configuration file:

```
/usr/local/MONyog/preferences.config
```

- Linux .gz archive:

If you have extracted SQL DM for MySQL package in a directory called MONyog the data stored is in:

Data folder: The connection configuration and collected data are kept here. You can find directories named like 0001, 0002, 0003, etc.

```
MONyog/data/
```

Installation file:

```
 MONyog/MONyog.ini
```

Log file:

```
MONyog/MONyog.log
```

Configuration file:

```
MONyog/preferences.config
```

The data folder specified above is the default settings only. You can store in any position on any mapped/mounted drive that is writable.

TOP(see page 211)

9. How does SQL DM for MySQL store its data?

Except for two plain text files: the MONyog.log file and a very small .ini file (that contains information about the port on which SQL DM for MySQL listens, The SQL DM for MySQL administrator password and the path to the data folder), everything is kept in high-performance database files (SQLite format).

TOP(see page 211)

10. Can I move a SQL DM for MySQL installation to another computer while keeping the data stored in SQL DM for MySQL database?

Yes. Just install SQL DM for MySQL on the 2nd machine. After install, stop the running MONyog(SQL DM for MySQL) service and copy the ..\MONyog\Data folder from the old installation. You may also copy the MONyog.log if you want. All the connection configuration and the data is located in 'data' directory. The error log is MONyog.log and settings are stored in 'MONyog.ini' and 'preferences.config'. If you have made any changes to monitors they are stored in 'Counters.def' and for CSO's in 'udo.def'. Copy all of them from your old installation onto new PC. After that start the service again.

TOP(see page 211)

11. Can SQL DM for MySQL be configured as a virtual host in my 'ordinary' Apache webserver?

Yes, at least with Apache this is possible. In your Apache configuration file (httpd.conf) add something like this (where 'ip1.ip2.ip3.ip4' is the IP address you reserve for SQL DM for MySQL).

```
<VirtualHost *:80>
      ServerName  monyog.mydomain.com
      ServerAlias http://monyog.mydomain.com
      Redirect permanent /  https://monyog.mydomain.com
  </VirtualHost>

  NameVirtualHost *:443
  <VirtualHost *:443>
      ServerName  monyog.mydomain.com
      ProxyPreserveHost On
      ProxyPass /  http://127.0.0.1:<MONyog-Port>/
      ProxyPassReverse /  http://127.0.0.1:<MONyog-Port>/
      SSLEngine On
      SSLCertificateFile <path-to-ssl-certificate.crt>
      SSLCertificateKeyFile <path-to-ssl-key.key>
  </VirtualHost>
```

And run the following command on the machine running Apache server:

```
/usr/sbin/setsebool httpd_can_network_connect=1
```

After changing the configuration, restart the Apache server.

TOP(see page 211)

12. How can I access SQL DM for MySQL pages proxying through other webservers?

We can also access SQL DM for MySQL using Apache proxy. You need to follow these simple steps to configure your Apache server to support proxy.

Here, we can setup Proxy in system A and we assume that SQL DM for MySQL is installed in system B, now you can access SQL DM for MySQL using, "http:///monyog/".

Configurations on system-A:

1. Please check whether you have libxml2 installed in your system.
2. Download mod_proxy_html.c from http://apache.webthing.com/(see page 211)
3. Now build mod_proxy_html with apxs, apxs -c -I/usr/include/libxml2 -i mod_proxy_html.c
4. You need to load the following modules, so add the following entries in [/etc/httpd/conf/httpd.conf].

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule headers_module modules/mod_headers.so
LoadModule deflate_module modules/mod_deflate.so
LoadFile /usr/lib/libxml2.so
LoadModule proxy_html_module modules/mod_proxy_html.so
```

5. Add the following configuration in your Apache configuration file [/etc/httpd/conf/httpd.conf]

ProxyRequests Off

```
<Proxy *>
            Order deny, allow
            Allow from all
            </Proxy>
            ProxyHTMLExtended On
            ProxyPass      /monyog/  http://<ip-system-B>:5555/
            ProxyHTMLURLMap http://<ip-system-B>:5555/ /monyog/
            <Location /monyog/>
                  ProxyPassReverse /
                  SetOutputFilter  proxy-html
                  ProxyHTMLURLMap  /      /monyog/
                  RequestHeader    unset  Accept-Encoding
            </Location>
```

TOP(see page 211)

13. How can I access SQL DM for MySQL pages proxying through nginx?

You can also access SQL DM for MySQL using nginx proxy. You need to follow these simple steps to configure your nginx server to support proxy. Here, we can setup Proxy in system A and we assume that SQL DM for MySQL is installed in system B, now you can access SQL DM for MySQL both over HTTP("http://") and HTTPS("https://").

Configuration of System A:

1. Install nginx on your system.
2. Create directories
   a. /var/log/nginx
   b. /var/www/cache

3. Configure nginx: Open nginx.conf found in /etc/nginx and add the following in the http section:

```
proxy_cache_path  /var/www/cache levels=1:2 keys_zone=my-cache:8m max_size=1000m
inactive=600m;
 proxy_temp_path /var/www/cache/tmp;
```

A sample nginx.conf would like the following:

```
user  nginx;
 worker_processes  1;
 error_log  /var/log/nginx/error.log debug;
 pid        /var/run/nginx.pid;
 events {
    worker_connections  1024;
 }
 http {
   include       /etc/nginx/mime.types;
   default_type  application/octet-stream;
   log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                     '$status $body_bytes_sent "$http_referer" '
                     '"$http_user_agent" "$http_x_forwarded_for"';
   access_log  /var/log/nginx/access.log  main;
   sendfile        on;
   #tcp_nopush     on;
   keepalive_timeout  0;
   proxy_cache_path  /var/www/cache levels=1:2 keys_zone=my-cache:8m max_size=1000m
inactive=600m;
   proxy_temp_path /var/www/cache/tmp;
   include /etc/nginx/conf.d/*.conf;
 }
```

4. Put your SSL certificates in /etc/nginx/conf/

5. Create a monyog.conf file inside /etc/nginx/conf.d/ and add the following:

```
server {
     server_name _;
     listen 80;
     location / {
            proxy_pass http://<ip-system-b>:5555;
            proxy_redirect     off;
            proxy_cache my-cache;
            proxy_cache_valid  200 302  0m;
            proxy_cache_valid  404      0m;
            proxy_set_header   Host             $host;
            proxy_set_header   X-Real-IP        $remote_addr;
            proxy_set_header   X-Forwarded-For  $proxy_add_x_forwarded_for;
            proxy_max_temp_file_size 0;
            client_max_body_size       10m;
            client_body_buffer_size    128k;
            proxy_connect_timeout      9000;
            proxy_send_timeout         9000;
            proxy_read_timeout         9000;
            proxy_buffer_size          4k;
            proxy_buffers              4 32k;
            proxy_busy_buffers_size    64k;
            proxy_temp_file_write_size 64k;
      }
}
server {
     server_name _;
     listen 443;
     ssl on;
     ssl_certificate /etc/nginx/conf/<certificate_name>.crt;
     ssl_certificate_key /etc/nginx/conf/<certificate_key>.key;
     location / {
            proxy_pass http://<ip-system-b>:5555;
            proxy_redirect off;
            proxy_cache my-cache;
            proxy_cache_valid 200 302 0m;
            proxy_cache_valid 404 0m;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_max_temp_file_size 0;
            client_max_body_size 10m;
            client_body_buffer_size 128k;
            proxy_connect_timeout 9000;
            proxy_send_timeout 9000;
            proxy_read_timeout 9000;
            proxy_buffer_size 4k;
            proxy_buffers 4 32k;
            proxy_busy_buffers_size 64k;
            proxy_temp_file_write_size 64k;
     }
```

```
}
```

TOP(see page 211)

14. Can I access SQL DM for MySQL pages using encrypted connection such as "https"?

Yes, you can access SQL DM for MySQL using "https", you may acquire a certificate from a certificate authority, such as Verisign or you may use the OpenSSL package to create your own certificate and configure your Apache webserver for "https".

Here are the steps you may follow to setup "https" in your Apache webserver.

1. Create a directory

   ```
   mkdir sslcert
   ```

   Now protect the directory,

   ```
   chmod 0700 sslcert
   ```

2. Create two sub-directories

   ```
   mkdir certs private
   ```

3. Create a database to keep track of each certificate

   ```
   echo '100001' >serial
   touch certindex.txt
   ```

4. Create a custom config file for OpenSSL to use similar to openssl.cnf in your /etc/pki/tls folder.

```
     dir = .
        [ ca ]
        default_ca = CA_default
        [ CA_default ]
        serial = $dir/serial
        database = $dir/certindex.txt
        new_certs_dir = $dir/certs
        certificate = $dir/cacert.pem
        private_key = $dir/private/cakey.pem
        default_days = 365
        default_md = md5
        preserve = no
        email_in_dn = no
        nameopt = default_ca
        certopt = default_ca
        policy = policy_match
        [ policy_match ]
        countryName = match
        stateOrProvinceName = match
        organizationName = match
        organizationalUnitName = optional
        commonName = supplied
        emailAddress = optional
        [ req ]
        default_bits = 1024 # Size of keys
        default_keyfile = key.pem # name of generated keys
        default_md = md5 # message digest algorithm
        string_mask = nombstr # permitted characters
        distinguished_name = req_distinguished_name
        req_extensions = v3_req
        [ req_distinguished_name ]
        0.organizationName = Organization Name (company)
        organizationalUnitName = Organizational Unit Name (department,
division)
        emailAddress = Email Address
        emailAddress_max = 40
        localityName = Locality Name (city, district)
        stateOrProvinceName = State or Province Name (full name)
        countryName = Country Name (2 letter code)
        countryName_min = 2
        countryName_max = 2
        commonName = Common Name (hostname, IP, or your name)
        commonName_max = 64
        0.organizationName_default = My Company
        localityName_default = My Town
        stateOrProvinceName_default = State or Providence
        countryName_default = US
        [ v3_ca ]
        basicConstraints = CA:TRUE
        subjectKeyIdentifier = hash
```

```
authorityKeyIdentifier = keyid:always,issuer:always
[ v3_req ]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
```

5. Create a root certificate. All other certificates you create will be based of this. Since this is not a commercial certificate software may complain when they use your certificates. You may give people the "public" certificate and your certificate works like the commercial ones when they import it. To create, while in the 'sslcert' directory type:

```
openssl req -new -x509 -extensions v3_ca
-keyout private/cakey.pem -out cacert.pem -days 365 -config ./openssl.cnf
```

You will be prompted for information and a password. Do not lose this password, make sure it is a secure one and back-up the two files that are created.
The two files that are created are cacert.pem, which is the one you can give to others for import in their browsers, and cakey.pem, which will be in the private directory.

6. Create a key and signing request

```
openssl req -new -nodes -out name-req.pem
-keyout private/name-key.pem -config ./openssl.cnf
```

You will be prompted for information. The critical part is the "Common Name". This must be the server's hostname, such as mail.your.domain or the IP address. If you want to cover all subdomains you can enter *.your.domain. Use the "Organizational Unit" to remind you what the certificate is for, such as "Web Server". This generates two files:

- name-req.pem - the request
- name-key.pem - the private key in the private directory

7. Sign the request. This generates the certificate:

```
openssl ca -out name-cert.pem -config
./openssl.cnf -infiles name-req.pem
```

You will be prompted for the password used when creating the root certificate. Two files are created:

- <number.pem> - a copy of it in the certs directory
- name-cert.pem - which is the certificate

8. Copy to the correct location For Apache 2.x on Red Hat using the default location, the directory is:
   a. For the name-key.pem:

```
cp
name-key.pem /etc/httpd/conf/ssl.key/
```

   b. For the certificate:

```
cp
name-cert.pem /etc/httpd/conf/ssl.crt/
```

9. Create a Virtual Host

```
<VirtualHost ip-system-A>:443> DocumentRoot /var/www/html
     ServerName myserver
     ErrorLog /etc/httpd/logs/ssl_error_log
     TransferLog /etc/httpd/logs/ssl_access_log
     SSLEngine On
     SSLCertificateFile /etc/httpd/conf/ssl.crt/name-cert.pem
     SSLCertificateKeyFile /etc/httpd/conf/ssl.key/name-key.pem
      </VirtualHost>
```

10. Configure proxy in Apache described in FAQ 13(see page 217) and restart Apache.
Edit the Hosts file [/etc/hosts]

```
<ip-system-A> myserver
```

TOP(see page 211)

15. What are the major differences between other major MySQL Monitoring Tool and SQL DM for MySQL?

They have similarities (in which they both differ from server-side scripts used for monitoring): they both make use of a HTTP service, a database and both use a web-browser for reporting. However, important differences are:

- SQL DM for MySQL needs no installation (of **'agents'**) on the server where the MySQL servers are running. Other does.
- SQL DM for MySQL **'has everything in itself'** - the webserver, the database, the MySQL client. It does not depend on the existence of other webservers, runtimes/Virtual Machines (like JAVA) and needs no separate database install. Other monitoring tools requires a full JDK (java), a TOMCAT server and a MySQL server instance for itself. Due to this simplified architecture install, configuration and first of all maintenance and upgrade is much simpler with SQL DM for MySQL. Download packages and disk storage required are much smaller with SQL DM for MySQL.

TOP(see page 211)

16. Can I trust the expertise of SQL DM for MySQL developers?

SQL DM for MySQL is developed by the Webyog Softworks that also created the most popular GUI for data management with the MySQL server - SQLyog Enterprise. We have more than 10 years of experience with designing MySQL related software. We have expanded our team with highly qualified developers ever since we started. We are devoted to constantly extending our knowledge and understanding of MySQL internals.

TOP(see page 211)

17. How does SQL DM for MySQL connect to MySQL?

SQL DM for MySQL uses the most proved and most efficient way of connecting: the native MariaDB Connector/C that is compiled into SQL DM for MySQL. Nothing else required: no separate client instance, no database abstraction layer (like ODBC/JDBC/ADO/.NET) and no webserver extensions (like PHP). Additionally, the connection can be

'wrapped' in a SSH tunnel. Also, SQL DM for MySQL implementation for this does not involve any other program (like Putty) running.

TOP(see page 211)

18. Windows warns after installation that SQL DM for MySQL may not have installed properly.

This can happen on some versions of Windows (Vista and higher) if you install a recent version of SQL DM for MySQL on top of an older version. The reason for this is that recent versions of Windows include a software called "Program Compatibility Assistant" (PCA) which tries to detect if an installer is running. It warns the user that the software might not have been correctly installed if the installer does not register a new uninstaller. The PCA is unable to detect changes made to an existing, registered uninstaller, which is what the new SQL DM for MySQL installers do. And thus, the warning is displayed. You can safely ignore this warning, but if it bothers you, you may just uninstall SQL DM for MySQL before upgrading to 3.5+. All collected data in the SQL DM for MySQL's data folder is still available after reinstall. However, you should:

- Backup the connections.data file before uninstalling
- Restore the old connections.data after the new install. After a restart, SQL DM for MySQL recognizes the connection settings in the old connections.data.

TOP(see page 211)

19. I would like to use SSH-tunnel, but my Windows server does not support it. Can that be fixed?

Yes, SSH support can be installed on Windows. You may install a complete Cygwin (Unix command line implementation for Windows). Alternatively, there are small packages available that support only a small subset of Cygwin (like SSH packages). Installation details depends on the exact Windows version.

TOP(see page 211)

20. SQL DM for MySQL throws an error when trying to connect to MySQL.

Please go through Error when trying to connect to MySQL(see page 211). The same applies to SQL DM for MySQL as the client code is exactly the same in both programs. Observe however that everything related to HTTP-tunneling with SQLyog is not relevant for SQL DM for MySQL.

TOP(see page 211)

21. Failed to connect to MySQL: Can't connect to local MySQL server through socket... What can i do about this?

Ensure that the host specified resolves to an IP-address. This error occurs with some Linux distributions (most important Debian) when specifying 'localhost'. The system maps this to a Unix SOCKET file. SQL DM for MySQL connects through TCP and not to SOCKET. Try the IP '127.0.0.1' instead.

TOP(see page 211)

22. SQL DM for MySQL is taking up too much of system resources with the PROCESSLIST-based sniffer.

You may have noticed that, while using the PROCESSLIST-based sniffer, SQL DM for MySQL increases the load on the CPU as well as the I/O subsystem of the system on which it is installed - even when the MySQL server is idle. Do not panic: it is normal. When using the PROCESSLIST-based sniffer, SQL DM for MySQL continually queries the MySQL server at the end of each time interval, which you can specify. It then retrieves the results and stores them in an internal sniffer database before displaying the results back to you. Now, if you set a short time interval, one that almost approaches 0, then SQL DM for MySQL can get stuck in an infinite loop! Consequently, the load on the CPU and I/O subsystem increases exponentially. We generally recommend an interval of not more than 0.1 seconds times the number of servers for which Processlist-based sniffers are enabled. However, if you are worried that you may miss out on some important queries running on the MySQL server, use the Performance Schem or MySQL Proxy mode. The LUA script supplied with SQL DM for MySQL should handle the task for MySQL proxy. For more information review MySQL Proxy(see page 211).

TOP(see page 211)

23. Why is display of queries truncated in Query Analyzer?

When using Query Analyzer feature, you may notice that sometimes queries displayed in output are incomplete. This may be due to one of the two known causes:

- As a security measure, SQL DM for MySQL extracts only the first 2000 characters of the query.
- MySQL does not record query delimiters in the General Log. Therefore, while analyzing the General Log, SQL DM for MySQL takes into consideration only the first line of the queries, and ignores the rest if they span over multiple lines.

TOP(see page 211)

24. The servers that I have registered do not display. What is wrong?

Check if the server is filtered based on a particular state, change your server filter to **All Servers** and now you can see your server between the servers if you had successfully registered it. You can also use the search bar next to the server filter to search for your server name or tag name to get to your server.

TOP(see page 211)

25. Now, anybody will be able to connect to my SQL DM for MySQL server and retrieve details about MySQL servers.

No, the SQL DM for MySQL authentication system will ensure that only those people that should have access have.

TOP(see page 211)

26. I have the same server registered twice. Metrics are reported different. Why?

For every registered server SQL DM for MySQL collects data independently. That is also the case when a server has been registered twice. Even if they were registered at the same time and even if the chosen sample interval is the same too, the connection and the server have some latency and data is not retrieved simultaneously. For that reason SQL DM for MySQL may retrieve and store slightly different values for each connection. This is most visible in the 'Delta' timeframe and least visible in the 'Current/all' timeframe. For GROUPING with 'History/Trends' the difference for each GROUP depends on the selected grouping interval. Due to laws of statistics the difference is less the longer the time interval (theoretically/statistically they converge more and more the closer time interval and/or the number of samples comes to infinity). Practically, you rarely need more than 20 samples in a GROUP for the difference to be negligible.

TOP(see page 211)

27. Does it affect the performance of a server if SQL DM for MySQL connects to it?

It does not affect on real 'live' servers. The queries sent by SQL DM for MySQL use almost no resources. We do not query data stored on disk and what we do query is stored in memory on the server. However if you are testing SQL DM for MySQL using a server instance that does almost nothing else and if you retrieve data at very short intervals the impact of SQL DM for MySQL may be slightly observable. The special Processlist feature (unique) may take a little more resources if there are lots of processes/client threads running. But SQL DM for MySQL only sends queries related to this when the corresponding SQL DM for MySQL client interface (the SQL DM for MySQL 'processlist' page) is open. Switching to another page or closing the browser stops sending the queries populating the SQL DM for MySQL processlist.

TOP(see page 211)

28. Is it possible to avoid that SQL DM for MySQL itself influences certain counters reported?

SQL DM for MySQL is a client. When it connects, the MySQL server starts a connection thread. And that connection is reported by SQL DM for MySQL. That cannot be avoided. The processlist feature has an option to 'filter out' SQL DM for MySQL connection - as well as other connections from other clients if you want - using a simple SELECT statement.

TOP(see page 211)

29. Can I customize SQL DM for MySQL counters?

Regarding customizing SQL DM for MySQL counters refer to Customization(see page 118). For scripting examples refer Customization Scripting Examples(see page 107).

TOP(see page 211)

30. I cannot sit watching a browser all the time - Can I get alerts if something goes wrong?

Yes, you can choose to get notifications (Email, SNMP traps, Slack, Pagerduty and Syslog) independently for every server and you can define your own warning levels and select what counters should raise an alert.

TOP(see page 211)

31. SQL DM for MySQL cannot identify if destination of the log file is on a "Mapped Network Drive". Why?

By default MONyog(SQL DM for MySQL) service runs under Local System Account. If you are having Slow query or General query log in a Mapped Network Drive, SQL DM for MySQL is not be able to reach it. If SQL DM for MySQL has to access the file present in a Mapped Network Drive, you have to convert the path into shared path (accessed with UNC notation: \system\share) and then follow these steps:

1. Click the **Start** menu, then click **Run** and then type,

   ```
   services.msc
   ```

2. After the Services window pops up with a list of all the services running in your system.
3. Search for Monyog and then right click --> **Properties**.
4. Click the **Log On** tab and then you can see that SQL DM for MySQL is using Local System Account.
5. You need to use **This account** option and then give the credentials that you use to log on to the system with Administrative privilege.
6. Save the settings, restart MONyog (SQL DM for MySQL) service.
7. After following the above steps try to access the file which is shared across network.

> **Note**
>
> The shared path should be accessed with UNC notation (\system\share). SQL DM for MySQL cannot identify if destination of the log file is on a Mapped Network Drive (this is a restriction with services on Windows and not with SQL DM for MySQL).

TOP(see page 211)

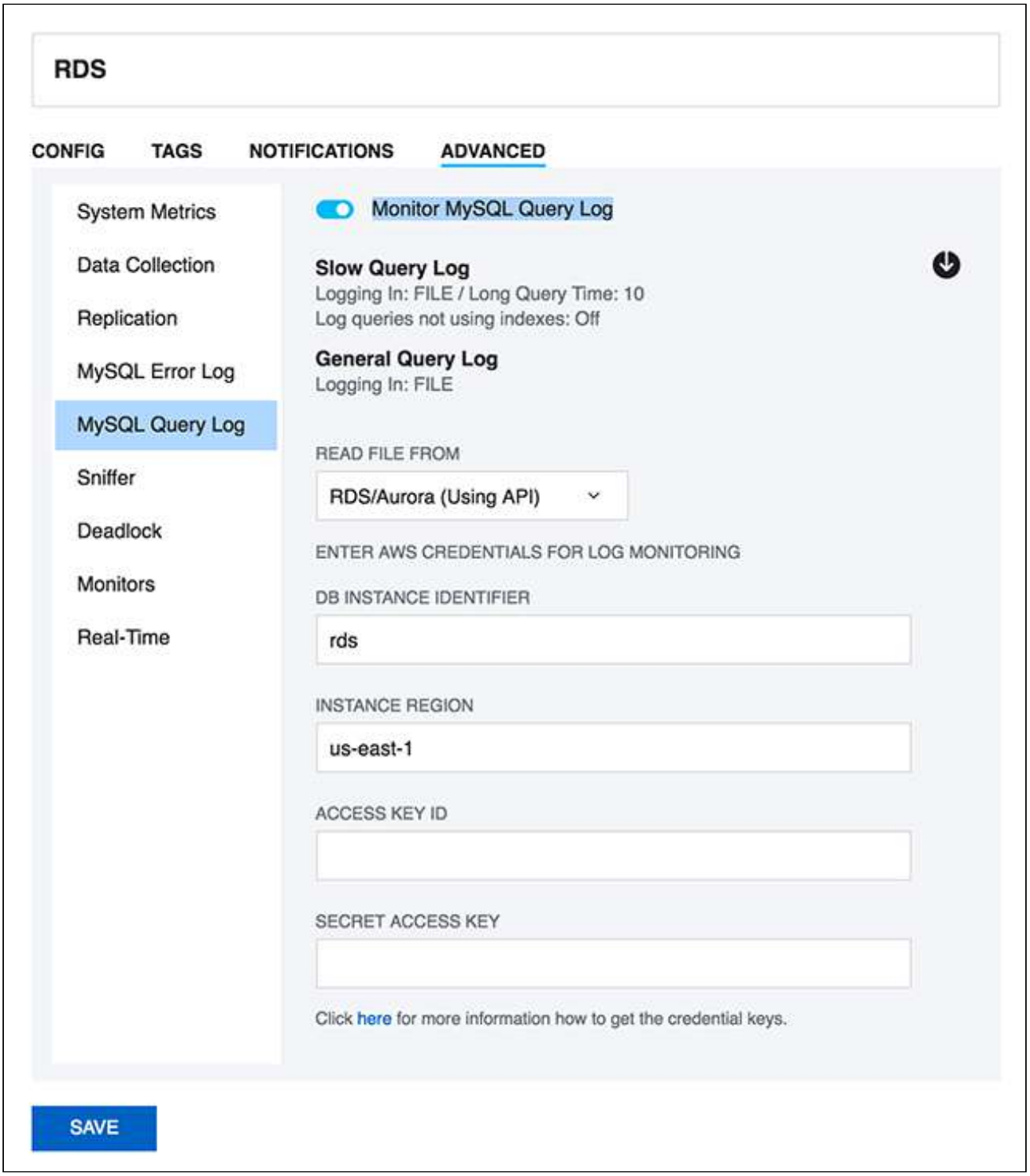
32. Failed to connect to MySQL: Unknown MySQL server host... What can I do about this?

You get this error if SQL DM for MySQL cannot resolve the hostname of a MySQL server. Ensure that other programs like ping, telnet, MySQL shell client are able to resolve the hostname to an IP-address. If yes, check "/etc/nsswitch.conf" of SQL DM for MySQL host. If the hosts section reads "files mdns4_minimal [NOTFOUND=return] dns mdns4", please change it to "files mdns4_minimal dns mdns4" or "files dns". This is introduced in some current Linux distribution. If other programs are not able to resolve the hostname, please check if host to IP resolution is properly defined inside "/etc/host" or in DNS server.

TOP(see page 211)

33. How can I monitor the queries from the file based RDS/Aurora Query logs?

SQL DM for MySQL can fetch the queries from the Slow Query log and General query log on Amazon RDS instance using the RDS REST APIs. SQL DM for MySQL requires the AWS access keys to fetch the file-based logs. Go to the **Edit Server->Advanced->MySQL Query log** and enable the option of "**Monitor MySQL Query Log**". Click the **Fetch logs**(down arrow) button and provide the AWS access key and secret access key to enable SQL DM for MySQL to monitor the log files.



TOP(see page 211)

34. What are future plans for SQL DM for MySQL?

SQL DM for MySQL is an important product for us. We plan to add new features as well as to 'refine' existing features. With the latest release we have completed what we originally planned for SQL DM for MySQL.

- It is now possible to get OS metrics from Amazon RDS/Aurora
- Added more notification channels (Slack, Pagerduty and Syslog) for SQL DM for MySQL alerts.

These features have all been requested by users. SQL DM for MySQL development has always been and continues to be very attentive to user requests. We update information here when plans for future developments have been decided.

TOP(see page 211)

35. How do I get help and report problems?

Four ways:

- Website Intercom https://www.idera.com/(see page 211)
- Post in our Forums http://community.idera.com/(see page 211)
- Create ticket: ideramysqlsupport@idera.com[43]
- SQL DM for MySQL UI intercom

TOP(see page 211)

36. Can I use the keys generated from PuTTY for SSH connection?

SQL DM for MySQL does not support the key generated from PuTTY for SSH connection. However, you can convert the private key generated from "PuTTY key generator" into an open SSH key, and then use this key in SQL DM for MySQL to connect to the server. Here are the steps to follow:

- Go to PuTTY key generator, and generate a public/private key on your local system (refer the screenshot). Click the **Generate** button to generate the keys.

43 mailto:ideramysqlsupport@idera.com

- Copy the public key generated under the "**Key**" space to the **authorized_keys** file, which is located in the **.ssh** directory on the remote host that you want to connect to.
- Go to **"Conversions"** in PuTTY key generator and click "**Export openssh key**" and save the new converted private key in a file.
- Now open the file containing the converted OpenSSH private key, copy this key and paste in the **"Private key"** field in SQL DM for MySQL (**Edit server -> SSH settings -> Private key**).

TOP(see page 211)

37. Steps to auto-start MONyog(SQL DM for MySQL) service with OS reboot in Ubuntu and Debian systems.

Users can make use of the "MONyog" script shipped with the Monyog(SQL DM for MySQL) package to auto-start MONyog(SQL DM for MySQL) service with OS reboot. The SQL DM for MySQL script is located at " /MONyog/bin/". Please follow the steps below:

- Copy the 'Monyog' to "/etc/init.d/" from " /MONyog/bin/"
- Open the 'Monyog' script located at "/etc/init.d/" and edit the variable "curdir" (line number 15) and set it to the path of bin. After editing, it should look like this: curdir="/home/Users/Downloads/MONyog/bin/"
- Make the script executable by 'chmod +x /etc/init.d/MONyog'
- Use debian utility update-rc.d to install the script: update- rc .d MONyog defaults

TOP(see page 211)

38. How to upgrade SQL DM for MySQL without losing your data or configuration?

The SQL DM for MySQL binaries are shipped in 3 packages: .tar, .rpm and .exe. The upgrade process is simple and depends on your package. Follow the steps below to upgrade to the latest version of SQL DM for MySQL:

For .rpm package :

```
rpm -Uvh <MONyog_package>.rpm
```

This command installs the latest build on top of your current installation.

For .tar package:

```
tar -xzvf <MONyog_package>.tar.gz
```

Please untar the package in the directory where the 'MONyog(SQL DM for MySQL)' package was untarred for the previous version to make sure that all your data and settings are intact.

For Windows (.exe) package:

Executing the file installs SQL DM for MySQL on top of the current installation.

All SQL DM for MySQL data and the configuration files are stored in a SQLite repository. In some of the SQL DM for MySQL GA releases, we modify/change the monitor definition in the SQLite files due to some bug or enhancements. In such cases, on upgrading, all the local changes made by the user in the previous version get replaced with the default shipped value and these local changes are shown as a conflict. You can see the conflicts as a notification on the top right-hand corner.

To resolve conflict

You can resolve these conflicts from the "**Settings -> Manage changes**" page, herein you get 2 options for all the listed conflicts: Use your changes/Discard your changes. 1- Use your changes, restores the local modifications which you had made in the previous version. 2- Discard your changes, replaces your changes with the default values.

TOP(see page 211)

39. How to maintain High Availability of SQL DM for MySQL?

Monit tool monitors the server process and can be used to maintain HA for SQL DM for MySQL.

Use the below commands to install monit:

apt:

```
sudo apt-get install monit
```

Yum:

```
sudo yum install monit
```

Once monit is installed, you can add programs and processes to the configuration file:

```
sudo nano /etc/monit/monitrc
```

Uncomment the below lines in the file to enable web interface. You can login to the web interface using the username ‘admin’ and password ‘monit’.

```
set httpd port 2812 and
use address localhost  # only accept connection from localhost
allow localhost        # allow localhost to connect to the server and
allow admin:monit      # require user 'admin' with password 'monit'
```

Also add the below lines in the configuration file to enable the monitoring of MONyog(SQL DM for MySQL) service:

Tar:

```
check process Monyog
    matching "MONyog"
    start program = "<MONyog extracted directory>/MONyog/bin/MONyog start"
    stop program = "<MONyog extracted directory>/MONyog/bin/MONyog stop"
```

Rpm:

```
check process Monyog
    matching "MONyog"
    start program = “/etc/init.d/MONyogd start”
    stop program = "/etc/init.d/MONyogd stop"
```

Once the above configuration changes are made, please use the below command to reload monit:

```
sudo monit reload
```

TOP(see page 211)