Foglight for MySQL 6.0.0

Cartridge Guide

If you have any questions regarding your potential use of this material, contact:

Quest Software Inc.

Attn: LEGAL Dept

4 Polaris Way

Aliso Viejo, CA 92656

Refer to our Web site (https://www.quest.com) for regional and international office information.

Patents

Quest Software is proud of our advanced technology. Patents and pending patents may apply to this product. For the most current information about applicable patents for this product, please visit our website at https://www.quest.com/legal .

Trademarks

Quest, the Quest logo, and Join the Innovation are trademarks and registered trademarks of Quest Software Inc. For a complete list of Quest marks, visit https://www.quest.com/legal/trademark-information.aspx. All other trademarks and registered trademarks are property of their respective owners.

Legend

CAUTION: A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

IMPORTANT, **NOTE**, **TIP**, **MOBILE**, or **VIDEO:** An information icon indicates supporting information.

Foglight for MySQL Cartridge Guide
Updated – June 2021
Version – 6.0.0

Contents

Foglight for MySQL Introduction

With over 65,000 downloads per day and six million installations worldwide, MySQL has become the most widely deployed open source database solution and is second most widely deployed database solution period! Originally released in 1995, adoption has steadily grown and with the acquisition by Oracle, MySQL now sets the standard as a robust low-cost alternative to high-cost enterprise data stores.

Early adopters included Internet pioneers such as Google, eBay, Craigslist and Yahoo, but the list of customers has steadily grown and now includes financial industry giants as Dun & Bradstreet, JP Morgan Chase, and telecom giants Nokia as well as pharmaceutical powerhouse Eli Lilly. Whether you are building and supporting commercial websites, distributing enterprise applications or engineering advanced communications networks, the technologies used to run your organization must be readily adaptable for you to remain competitive.

Foglight for MySQL Overview

Based on Quest Software's leading application performance management solution; Foglight for MySQL provides combines world-class monitoring and alerting with operational best-practices designed to ensure the performance and availability of your MySQL databases. Utilizing standard API's, the cartridge provides integration to MariaDB and MySQL Versions 5.0 and above. The robust nature of the Agent ensures its ability to collect granular performance data and display that information through intuitive, easy-to-use dashboards. Based on vast experience building and deploying monitoring solutions, Foglight for MySQL ensures the performance and availability of this critical component. The solution leverages best practices for collecting, storing and representing data as well as detecting environmental abnormalities.

True Enterprise Information Correlation

At the core of the solution is Quest Software's Foglight. Foglight's rich architecture combines a central repository, rules and notification engine, data visualization platform to provide an application performance management that is second to none. This platform permits MySQL data to be combined with data from other enterprise domains to create a true end-to-end view of your critical applications and services. MySQL data collection intervals are pre-defined but are user adjustable for each functional area of the database. Once data has been collected, it is stored in the Foglight repository for a period defined by the administrator. The repository centralizes data

collection and facilitates data visualization and trend analysis. Combining MySQL data with other metrics collected by Foglight produces a solution that provides the operator with unprecedented visibility into applications and business services.

Dashboards

Data is displayed through database specific dashboards. Foglight for MySQL provides a series of specific dashboards for MySQL databases and tables. These dashboards are easily complemented and extended utilizing Foglight's powerful drag-and-drop dashboarding capabilities. Operators can easily save dashboards as 'bookmarks' and return to them as needed. Dashboards and/or Reports can be created based on any data stored in the central repository and is not limited to data from a specific database such as MySQL.

Diagnostics is made possible through drill-downs from included dashboards. Each dashboard was designed to provide at-a-glance health state for domains. Additional diagnostic information is made available through drill-downs providing the critical information necessary to troubleshoot complex issues. Since most enterprises rely on a complex technology stack to ensure business continuity, Foglight provides the ability to correlate, view, alert and report on technology from most standard enterprise applications.

Rules Engine

Alerting is often the first indication that a problem exists or may exist if preventative measures are not taken in a timely manner. A notable strength of the solution lies is its ability to leverage both MySQL Error Log alerts and the Foglight rules engine. Error Log messages are propagated to Foglight where they are assigned a severity and can be acknowledged or cleared. Foglight provides the ability for notifications to be sent to administrative and support personnel. The Foglight rules engine permits the operator to easily construct or modify any rule based on any metric using standard operators and variables. A robust set of rules is included with the solution and is defined in this document.

Statement Digests

Statement digest monitoring extends the rich functionality of the MySQL Cartridge, providing the ability for Foglight to collect and analyze information from your MySQL server's performance_schema for versions 5.6.5 and greater, or the Slow Query Log for lesser versions. Queries are normalized and aggregated in order to provide meaningful results on the performance of unique query structures without exposing non-relevant or protected string values.

This functionality allows users to quickly identify slow running and problematic queries and helps administrators better understand the efficiency of these queries in both development and production environments with a minimal amount of overhead. Further information provides an accurate picture of how queries perform when the server is under a realistic workload with detailed metrics on performance throughout the query's execution, along with an explain plan available on

request. Coupled with Foglight's historical data collection and powerful rules engine, this ensures you will be alerted on all negative performance trends or spikes in query performance.

About Monitoring Extensions

During the installation process you can choose to install and configure one or more of the monitoring extensions. The monitoring extensions provide a more in-depth analysis of the monitored database and the environment it is running on, creating a whole and unified status.

SQL Performance Investigator Extension

SQL Performance Investigator (PI) allows you to rapidly identify bottlenecks, anomalies, and application trends by focusing on top resource consumers and providing multi-dimensional SQL domain drill-downs. SQL PI allows you to:

- Monitor real-time MySQL Database performance at a glance
- Gather and diagnose historical views
- Identify and anticipate performance issues

NOTE: SQL Performance Investigator requires a license. If you are using a trial version and would like to request pricing, contact https://www.quest.com/register/107452/.

NOTE: SQL PI requires a repository database that is installed automatically on the Agent Manager.

Foglight for MySQL Requirements

Foglight for MySQL is compatible with MySQL 5.5+ and equivalent versions for drop-in replacements like MariaDB and managed database services like AWS Aurora/RDS and Microsoft Azure. However, older versions of the server may not provide some monitoring data that Foglight for MySQL presents for later versions. Most notably, statement digest data is not present earlier than 5.6.5.

The agent may be run on a FglAM that is either local or remote to the MySQL server. More information on configuring the MySQL server for monitoring can be found in the MySQL Server Pre-Configuration section of this document.

NOTE: Quest expects Foglight to support any distribution or fork of MySQL, but cannot commit to fully testing all of them. We encourage customers to report any issues that they run into, and we will update our Support knowledge base with any limitations that we uncover.

Requirements for MySQL PI

In order for PI to collect data, the MySQL **Performance Schema** must be enabled with the required instrumentation. Refer to Performance Schema Startup Configuration in the MySQL documentation for more information.

NOTE: MySQL Performance Schema is enabled by default since MySQL 5.6.6

MySQL PI Data is collected based on MySQL **consumers** and **instruments**.

Refer to Configuring MySQL for Performance Investigation for more information.

Installing and Configuring Agents

Installation of Foglight for MySQL is covered in the following sections and should be performed in order:

- MySQL Server Pre-Configuration
- Cartridge Installation
- Creating and Configuring Agents

MySQL Server Pre-Configuration

In order to allow full monitoring of the MySQL Server, the agent will require a user with sufficient privilege to execute system queries. Additional steps may also be required to enable a SSL connection or monitoring the slow query log if desired. These are covered in the following sections:

- MySQL Agent User Permissions
- Configuring an Encrypted Connection
- Configuring the MySQL Slow Query Log
- Configuring MySQL for Performance Investigation

MySQL Agent User Permissions

The Foglight MySQL agent requires certain minimum privileges for the MySQL Database User to be able to monitor a MySQL database.

Create a MySQL Database User by logging into the MySQL server and granting the privileges identified below.

User privileges required for the MySQL agent on the host machine:

- SELECT
- REPLICATION CLIENT (if monitoring replication)
- PROCESS
- SUPER (for MySQL versions below 5.1.24)

If Administration is enabled, the Admin user provided in the agent properties will be required to have privileges necessary to run operations or request explain plans for the functions that you want to have available.

If monitoring a replication slave server with the agent, that database user must have:

- SELECT
- REPLICATION CLIENT

Example 1:

```
CREATE USER '<user>'@'<localhost or DB HostName or IP>' IDENTIFIED
BY '<password>';

GRANT SELECT, REPLICATION CLIENT, PROCESS ON *.* TO
'<user>'@'<localhost or DB HostName or IP>';

FLUSH PRIVILEGES;
```

e.g.

```
CREATE USER 'MySQLuser'@'localhost' IDENTIFIED BY 'MySQLpassword';

GRANT SELECT, REPLICATION CLIENT, PROCESS ON *.* TO
'MySQLuser'@'localhost';

FLUSH PRIVILEGES;
```

or

Example 2:

```
GRANT SELECT, REPLICATION CLIENT, PROCESS ON *.* TO
'<user>'@'<localhost or DB HostName or IP>' IDENTIFIED BY
'<password>';

FLUSH PRIVILEGES;
```

e.g.

```
GRANT SELECT, REPLICATION CLIENT, PROCESS ON *.* TO
'MySQLuser'@'localhost' IDENTIFIED BY 'MySQLpassword';

FLUSH PRIVILEGES;
```

Permissions for SQL Server Based PI Repository Database

If you are using SQL Server Based PI Repository®, ensure that these permissions are set.

Table 1: Permissions for Creating a New PI Repository Database

Instance Level		Database Level	
CREATE ANY DATABASE	Granted for: Creating SQL PI Database	db_owner	Granted for: Insert/delete PI records

NOTE: The CREATE ANY DATABASE permission can be removed after the SQL PI database is created. The SQL PI user should be the database owner of the new created SQL PI database.

Table 2: Permissions for Using an Existing PI Database

Instance Level		Database Level	
VIEW ANY DATABASE	Granted for: Selecting Existing Database	db_owner	Granted for: Insert/delete PI records

Configuring an Encrypted Connection

The below instructions cover common steps used to configure an encrypted connection from the MySQL Agent client. For full information on secure connections and server-side configuration, refer to the Using Secure Connections section of the MySQL documentation for your version.

In order to use SSL, your MySQL server must be built with OpenSSL or yaSSL. To check whether SSL is enabled, run this query:

```
SHOW VARIABLES LIKE 'have_ssl';
```

If the query returns YES, your server can use SSL. If it returns DISABLED, the server must be started with the SSL options listed in the above mentioned section. SSL and RSA certificates and keys must also be generated in order to use SSL. Information on generating those can be found here.

The client requires a client certificate and certificate authority (CA) certificate. These are named client-cert.pem and ca.pem respectively if generated by the MySQL server and should be located in the data directory. First, the client certificate needs to be converted into DER format. This can be performed by downloading OpenSSL and running the following command:

```
openssl x509 -outform DER -in client-cert.pem -out client.cert
```

Then, the certificates must be imported into the FglAM keystore. You can use the bundled keytool, which will be located in the Foglight Agent Manager\jre\1.8.0.72\jre\bin directory, or the equivalent on your system, with these commands:

```
keytool.exe -import -file ca.pem -keystore truststore -alias
mysqlServerCACert
```

```
keytool.exe -import -file client.cert -keystore keystore -alias
mysqlClientCertificate
```

If you have not changed the password for the keystore, the default password will be “changeit”. Next, edit the baseline.jvmargs.config file in the Foglight Agent Manager\state\default\config directory and add the following parameters with file paths and passwords appropriate for your system. Escape any quotes with a ‘\’.

```
vmparameter.0 = "-Djavax.net.ssl.keyStore=\"C:/Program Files/Common
Files/Dell/Foglight Agent Manager/jre/1.8.0.72/jre/bin/keystore\"";
```

```
vmparameter.1 = "-Djavax.net.ssl.keyStorePassword=changeit";
```

```
vmparameter.2 = "-Djavax.net.ssl.trustStore=\"C:/Program
Files/Common Files/Dell/Foglight Agent
Manager/jre/1.8.0.72/jre/bin/truststore\"";
```

```
vmparameter.3 = "-Djavax.net.ssl.trustStorePassword=changeit";
```

Then, restart the FglAM and continue with the agent configuration, setting the **Use SSL** option in the **Agent Properties** to true.

Configuring the MySQL Slow Query Log

For MySQL servers version 5.6.5 or greater, it is recommended that you monitor statement digests through the performance_schema, as it collects more information. However, some limited information can be gathered through the slow query log.

Configuration of the Slow Query Monitor first requires that slow query logging is enabled. This can be accomplished in one of three ways.

1. Change the server runtime parameters while the server is running.

2 Provide certain command-line options when starting the server.

3 Edit the MySQL configuration file to enable and configure the log.

We recommend using the third option, which will ensure that the MySQL Server is always started with the correct parameters. The first option can be used to avoid doing an initial restart of the server in order to enable the slow query log. As of version 5.1.29, the following options may be used:

Option	Samle Value	Required	Note
slow_query_log	1	Yes, defaults to 0	1 enables the log, 0 disables it
slow_query_log_file	C:\Program Files\MySQL\slow_query.log	Yes, defaults to hostname-slow.log	Any path name is acceptable
long_query_time	0.5	Yes, defaults to 10	Units are in seconds, can also use 0 or microseconds
log_short_format	FALSE	Yes, defaults to FALSE	Must be set to FALSE
log_slow_admin_statements	OFF	No, defaults to OFF	Logs admin statements like ANALYZE, OPTIMIZE, ALTER TABLE, etc.
log_queries_not_using_indexes	ON	No, defaults to OFF	Logs queries expected to retrieve all rows
log_slow_slave_statements	OFF	No, defaults to OFF	Logs statements on a replication slave server

See section 5.2.5 of the online MySQL documentation for more information on the slow query log specific to your version. Some of these properties may become deprecated in future versions or not exist in versions before 5.1.12.

Additionally, it is important to understand that the server does not write queries handled by the query cache to the slow query log, nor does it write queries that would not benefit from the presence of an index as the table has either zero or one row. Also of note is that the server does not automatically rotate the slow query log. If you wish to rotate your log file, you may simply delete or rename the current log file manually or with a utility program and the server will create a new log

file itself. It is recommended you do this at a period of low activity so as to not cause the agent to miss reading important data.

Configuring MySQL for Performance Investigation

In order for PI to collect data, the MySQL **Performance Schema** must be enabled with the required instrumentation.

Enabling consumers

At a minimum, Quest recommends enabling the following consumers at startup:

- **global_instrumentation** – enabled by default
- **thread_instrumentation** – enabled by default
- **events_statements_current** – enabled by default
- **events_statements_history** – already enabled by default (10 events)
- **statements_digest** – enabled by default
- **events_waits_current**
- **events_statements_long** – (1000 events)

To enable the above consumers at startup – see Performance Schema Startup Configuration in the MySQL documentation.

Additionally, evaluate the following settings based on your data collection needs and desired performance. While leaving these setting enabled will collect the most data, you may achieve better performance by disabling unnecessary collections.

Performance schema consumers

- performance-schema-consumer-events-waits-current=ON
- performance-schema-consumer-events-waits-history=ON
- performance-schema-consumer-statements-digest=ON
- performance-schema-consumer-thread-instrumentation=ON
- performance-schema-consumer-events-statements-current=ON
- performance-schema-consumer-events-statements-history=ON
- performance-schema-consumer-statements-digest=ON
- performance-schema-consumer-events-statements-history-long=ON

Enabling Instruments

At a minimum, Quest recommends setting the following instruments:

- **Memory/*** – enabled by default
- **statement/*** – enabled by default
- **wait/io**
- **wait/io/file** – enabled by default
- **wait/io/table** – enabled by default
- **wait/io/socket**
- **wait/synch**

Additionally, evaluate the following settings based on your data collection needs and desired performance. While leaving these setting enabled will collect the most data, you may achieve better performance by disabling unnecessary collections.

Performance schema instruments

- performance_schema = ON
- performance_schema_instrument = 'statement/%=on'
- #performance_schema_instrument = 'stage/%=on'
- #performance_schema_instrument = 'memory/sql/%=on'
- performance_schema_instrument = 'wait/%=on'

Configuring other server variables

In addition to instruments and consumers, the sections below contain additional variable configurations that can improve monitoring results.

- **To see waits and locks - default is 10**

  performance_schema_events_waits_history_size = 100

- **To see digest and text information (especially in MySQL5*) - default is 10**

  performance_schema_events_statements_history_size = 1000

- **On 5.6 default is 200 - On 5.7+8.0 the default is 10000**

  performance_schema_events_statements_history_long_size=2000

Additional default values

The values below can affect data collection in your environment. Consider changing the default values if your actual data collection does not match your requirements.

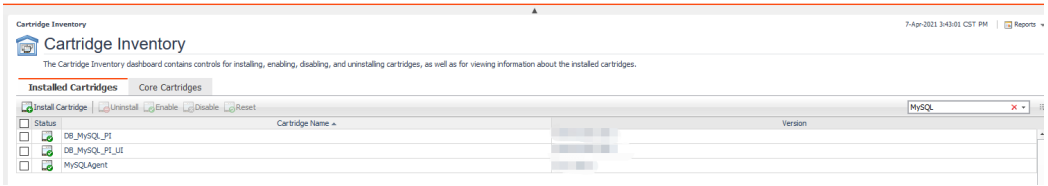
The default value is in **bold** in the entries below.

- #performance_schema_events_waits_history_long_size=**10000**
- #max_digest_length=**1024**
- #performance_schema_max_sql_text_length=**1024**
- #performance_schema_digests_size=**10000**

  On MySQL 5.6 the default is 200 - On 5.7, 8.0 the default is 10000
- #performance_schema_max_digest_length=**1024**

Cartridge Installation

1. Open the Foglight Web Console.
2. From the navigation pane, select: **Administration**.
3. Click **Licenses** in the **Administer Server** section.
4. Install the appropriate license key to run the cartridge.
5. Return to the **Administration** dashboard.
6. Click **Cartridges** in the **Administer Server** section.
   - To install the Foglight for MySQL Cartridge:
     a. Load the *MySQLAgent-6_x.car* file by clicking **Install Cartridge**. Leave **Enable on Install** selected when installing the cartridge.
     b. Once the installation is completed on the Foglight Management Server, the MySQL Agent Cartridge will appear in this list below as an installed cartridge.
   - To install the MySQL PI Cartridge:
     a. If you want to enable SQL Performance Investigator, ensure the **DB_MySQL_PI** cartridge is installed.
     b. Load the *DB_MySQL_PI-6_x.car* file by clicking **Install Cartridge**. Leave the **Enable on Install** check box checked when installing the cartridge.
     c. Once the installation is completed on the Foglight Management Server, the **DB_MySQL_PI** and **DB_MySQL_PI_UI** Cartridges will appear in this list below as installed cartridges.



Creating and Configuring Agents

Agents can be created in one of two ways:

- Using the Agent Installer Wizard
- Using the Agent Status Dashboard

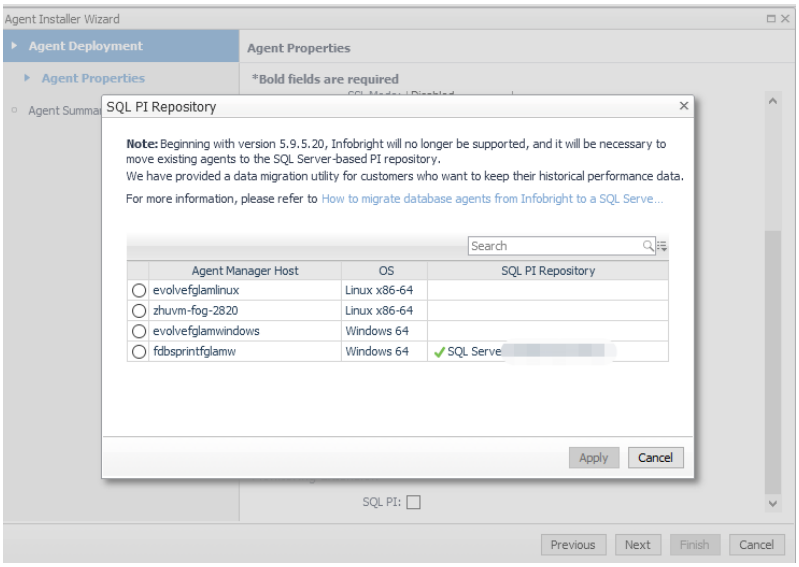
The Agent Installer Wizard simplifies the agent creation and configuration process and can be accessed from the Databases dashboard. For advanced configuration or modification of agent properties post-creation, use the Agent Status dashboard.

Using the Agent Installer Wizard

Foglight for MySQL provides a graphic, intuitive method for creating and configuring agents, which can be used instead of Foglight's default method for creating agents and editing their properties using the Agent Status dashboard. Foglight for MySQL allows running a wizard that provides a common entry point for adding database instances and then configuring these instances for monitoring.

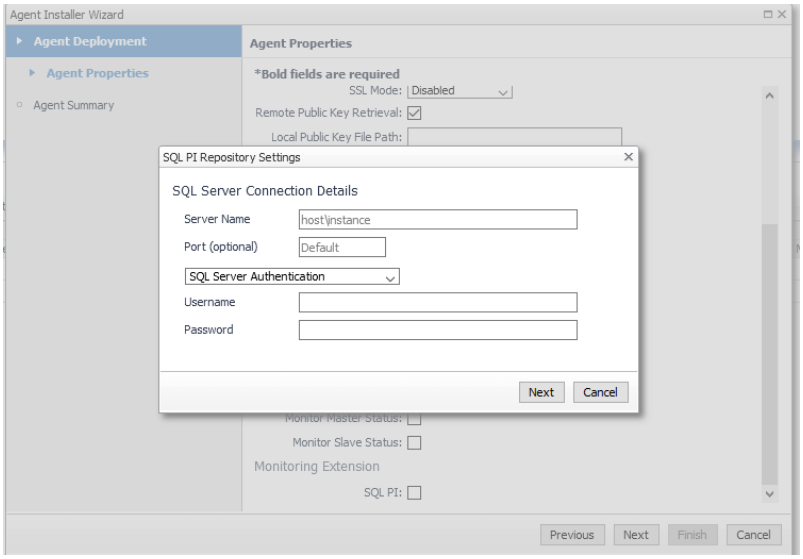
To run the instance installation wizard:

1. On the navigation panel, click **Homes > Databases**.
2. Click the **MySQL** box in the Databases View, and then click **Monitor**.
3. The **Agent Installer Wizard** dialog box appears.
4. On the **Agent Deployment** page, complete the dialog using the below as a guide:
    a. **Agent Name** – Provide a name for the agent that will be created. This is not canonical and should be representative of the database instance that this agent will monitor.
    b. **Agent Manager** - Choose the agent manager on which the agent should run. Considerations for this may include physical or virtual locality to the monitored instance, allocated resources, or grouping with other agents of the same type or monitored environment. If the agent package has not been deployed to this Agent Manager yet, it will be installed when the first agent of this type is created.
5. The **Agent Properties** dialog requires a basic set of parameters for connecting to and monitoring the database instance. A full explanation of these properties is available in the Agent Properties section of this document.
6. [Optional] To enable SQL Performance Investigator, select **SQL PI** in the **Monitoring Extension** section.
    a. The SQL PI Repository dialog is displayed. Select the Agent Manager on which the SQL PI repository should be installed. Then click **Apply**.



b If the SQL PI repository existed on the selected Agent Manager, no more action is required for the configuration.

c If no SQL PI repository exists on the selected Agent Manager, the **SQL PI Repository Settings** dialog is displayed. Enter the SQL PI Repository connection details.



d If the given SQL PI repository user does not have sufficient privileges, the **Insufficient Privileges** dialog will be displayed. Enter an admin user and password, and then click **Grant Privileges**.

7 The **Agent Summary** dialog displays a review of the configuration that will be created and an option allowing the agent to be activated after creation. If the configuration looks good, click **Finish** to start the process.

8 When the process completes, a results screen will appear showing the results of agent creation. If the agent was not created, follow the instructions on the results screen. If successful, the database instance should appear in the Databases table within a few minutes.

NOTE: If the agent was created successfully but data is not appearing, go to the **Dashboards > Administration > Agents > Agent Status** page and click the icon in the **Log File** column for the agent you created. In most cases, the reason for the failure will be obvious. You can also refer to the Foglight for MySQL Installation and Troubleshooting document for common errors and solutions. If the solution requires reconfiguring the agent properties, follow steps 3-7 of the **Using the Agent Status Dashboard** section.

Using the Agent Status Dashboard

The Agent Status page can be used to create new agents and configure and manage existing agents. To access the page from the navigation pane, select: Dashboards > Administration > Agents > Agent Status.

Use the following steps to create a new agent instance:

1. If the MySQL agent package has never been deployed to the FglAM that will be used to host the agent, this must be done before an agent has been created. You can use the **Deploy Agent Package** button on the **Agent Status** or **Agent Managers** page to perform this.
2. Click **Create Agent** and follow the instructions for the cards:
    a. **Host Selector** - Choose the Agent Manager on which the agent should run. Considerations for this may include physical or virtual locality to the monitored instance, allocated resources, or grouping with other agents of the same type or monitored environment.
    b. **Agent Type and Instance Name** – Select the MySQLAgent type. Then, select the Specify Name radio button and provide a name for the agent that will be created. This is not canonical and should be representative of the database instance that this agent will monitor.
    c. **Summary** – Click Finish.
3. Once the agent has been created, click the checkbox next to the MySQL agent.
4. Click the **Edit Properties** button.
5. Select **Modify the default properties** for this agent.
6. Edit the agent properties for the MySQL agent instance:
    - Connection Parameters (mandatory)
    - Administration (optional)
    - Replication (optional)
    - Statements (optional)
    - Collection Intervals (optional)

- Slow Query Log Monitoring (optional)
- Additional Options (optional)

7 Click **Activate**.

To modify the properties for an existing agent, skip to step 3 and **Deactivate**, then **Reactivate** the agents after changing the configuration.

Agent Properties

When an agent connects to the Foglight Management Server, it is provided a set of properties that is then used to configure its correct running state.

Default Agent properties are installed with the Foglight Cartridge for MySQL. However, the user will typically edit the default agent properties. Agent properties may apply only to a specific agent instance, or may be applicable across multiple agents.

For more information about working with agent properties, see the Foglight Administration and Configuration Guide.

To modify the agent properties for a new agent instance:

1 Open Foglight.

2 From the **Navigation** pane, select: **Dashboards > Administration > Agents > Agent Status**. The **Agent Status** screen appears.

3 Deploy the MySQL agent package.

4 Create a MySQL agent instance.

5 Click on a MySQL agent type row. The selected row is highlighted with a yellow background.

6 Click the **Edit Properties** button.

7 Select **Modify the default properties** for this agent.

8 Edit the agent properties for the MySQL agent instance:

- Setting Connection Parameters (mandatory)
- Setting Slave Connection Parameters (optional)
- Setting Collection Intervals (optional)

9 Click **Activate**.

Connection Parameters (mandatory)

- **Database Host** – Host where MySQL server is running. Default is “localhost”. (e.g.<hostname> or <IP address>)

- **Database Port** – Port the MySQL database is running. Default is 3306.
- **Database Name** – Name of a valid MySQL database that the user is allowed to connect to.

  NOTE: This is for connection purposes only. All non-system databases will still be monitored. The 'mysql' database is usually an acceptable entry.
- **Database User** – User that can connect to the MySQL server being monitored.
- **Database Password** – Password of the user that can connect to the MySQL database being monitored.
- **SSLMode** – Sets the level of security to use for encrypting a connection to the MySQL server. The following values are allowed:
  - **DISABLED** - Establish unencrypted connections;
  - **PREFERRED** - (default) Establish encrypted connections if the server enabled them, otherwise fall back to unencrypted connections;
  - **REQUIRED** - Establish secure connections if the server enabled them, fail otherwise;
  - **VERIFY_CA** - Like **REQUIRED** but additionally verify the server TLS certificate against the configured Certificate Authority (CA) certificates;
  - **VERIFY_IDENTITY** - Like **VERIFY_CA**, but additionally verify that the server certificate matches the host to which the connection is attempted. For more information on enabling SSL, refer to the Configuring an Encrypted Connection section.
- **Allow Public Key Remote Retrieval** – Allows the client to retrieve the public key from the server in order to secure the connection. This must be enabled if an SSL connection is not configured and the public key is not available locally.
- **Local Public Key File Path** – The file path to the public key for the MySQL server that the agent is attempting to connect with. The public key must be locally accessible by the agent, which is hosted on the FglAM machine. This can be used instead of allowing public key retrieval, preventing possible MITM interception.
- **Use Cleartext** – Enables the cleartext authentication method that sends the unhashed password to the MySQL server. It is recommended that SSL be enabled if this is done.

Administration (optional)

- **Enable Administration** – Set to **true** in order to use the MySQL Administration Panel for this agent. See the **MySQL Administration Panel** for more information. Default is **false**.
- **Enable Explain Plans** – Set to **true** in order to enable Explain Plan requests from the **Administration Panel** and **Statement Digests** pages. Default is **false**.
- **Admin User** – User that will be employed only to perform administrative actions on the monitored MySQL server. This user must have the privileges to perform the requested actions or the action will fail.

- **Admin Password** – Password of the user that will be employed to perform administrative actions on the server.

Replication (optional)

- **Monitor Master Status?** – Set to true to monitor the Master replication status on the MySQL server.
- **Monitor Slave Status?** – Set to true to monitor the Slave replication status on the MySQL server.
- **Monitor Replication Slave?** – Set to true to monitor the Slave replication status (and only this) on the replication slave server, of which the parameters below pertain to.
- **Database Host** – Host where MySQL Replication Slave Server is running.
- **Database Port** – Port the Slave Connection is running on.
- **Database Name** – Name of MySQL Replication Server Database to be monitored.
- **Database User** – User that can connect to the MySQL Replication Server database being monitored.
- **Database Password** – Password of the user that can connect to the MySQL Replication Server database being monitored.
- **Use SSL** – Requires the connection to the MySQL server to use SSL/TLS. For more information on enabling SSL, refer to the Configuring an Encrypted Connection section.
- **Allow Public Key Remote Retrieval** – Allows the client to retrieve the public key from the server in order to secure the connection. This must be enabled if an SSL connection is not configured and the public key is not available locally.
- **Local Public Key File Path** – The file path to the public key for the MySQL server that the agent is attempting to connect with. The public key must be locally accessible by the agent, which is hosted on the FglAM machine. This can be used instead of allowing public key retrieval, preventing possible MITM interception.
- **Use Cleartext** – Enables the cleartext authentication method that sends the unhashed password to the MySQL server. It is recommended that SSL be enabled if this is done.

Statements (optional)

- **Statements** – If the MySQL server version is 5.6.5+ and the performance_schema engine is enabled, the agent will collect statement digest information at this interval. Default set to 300 seconds
- **# of Top Statements** – Number of statements that will be collected, ordered by the Sort By property. Default is 1000.
- **Sort By** – Ordering parameter for statement collection limit. Default is Total Executions.

Collection Intervals (optional)

The Collection Interval fields in the agent properties are used to set the sample frequencies. Default collection frequencies (in seconds) have already been set for each table below.

- **Blocked Transactions** – Default set to 60 seconds.
- **Buffer Pool (i.e. Innodb_Buffer_Pool)** – Default set to 300 seconds.
- **Configuration Monitoring** - Default set to 300 seconds.
- **Connection_Status** – Default set to 60 seconds.
- **Database Information** – Default set to 300 seconds.
- **Database Stats** – Default set to 300 seconds.
- **Failed Logins** - If the MySQL server version is 5.6.0+ and the performance_schema engine is enabled, the agent will collect information on failed login attempts at this interval, shown on the Connections dashboard. Default set to 300 seconds.
- **Galera** – Default set to 60 seconds.
- **Handler** – Default set to 300 seconds.
- **Index Structure** - Default set to 3600 seconds.
- **Index/Table-level Compression** – Default set to 1800 seconds.
- **InnoDB Compression** – Default set to 300 seconds.
- **InnoDB Engine (i.e. Innodb_Storage_Engine)** – Default set to 300 seconds.
- **Joins** – Default set to 300 seconds.
- **Key Buffer** – Default set to 300 seconds.
- **Network Interface** – Default set to 300 seconds.
- **Query Cache** – Default set to 300 seconds.
- **Replication** – Default set to 300 seconds.
- **Sort Buffer** – Default set to 300 seconds.
- **Tables** – Default set to 1800 seconds.
- **Table Locks** – Default set to 60 seconds.
- **Thread Pool** – Default set to 300 seconds.
- **Top Sessions** – Default set to 300 seconds.
- **Transaction Log (i.e. Innodb_Transaction_Log)** – Default set to 1800 seconds.
- **Users** – Default set to 3600 seconds.

- **Group Replication/InnoDB Cluster Monitoring**
- **Enable Monitoring** – Enable this collection by setting to True.
- **Collection Interval (sec)** – Frequency of collection sample, 120 seconds by default.
- **Member Host Aliases** – This property list can be used to alter the host names of other members in the InnoDB cluster collected from this monitored server for display purposes and/or to reconcile host names if a different name is collected from other monitored member servers.

Slow Query Log Monitoring (optional)

- **Monitor Slow Query Log** - Enable this collection by setting to True.
- **File Access** – If the MySQL Agent is running locally on the same machine as the MySQL server it is monitoring, you may select Local. The other option, Remote_SSH, allows an SSH connection to a remote server in order to retrieve the slow query log.
- **Collection Period** – How often collection occurs, in seconds. It is recommended that this setting be at least 30-60 minutes or more. Currently, unique queries can only be aggregated within their own collection period, so a lengthier collection period will provide a more accurate analysis of a unique query without having to navigate through too many data collections.
- **Line Limit** – Restricts the amount of lines read from the slow query log during each collection. Unless a truly exceptional amount of data is being generated to the slow query log (in which case the long_query_time parameter should probably be increased), this parameter should remain at 0. A value of -1 will cause the agent to read all previous data during its first collection rather than marking the current file position and reading from that point during its next collection.
- **File Path** – File path to slow query log directory.
- **File Name** – File name of slow query log. Optionally, for Local collection only, a regex string may be used and the most recently modified file matching the regex string name will be used.

If using remote file access, the following values must be provided

- **Remote Hostname or IP** – A hostname or IP address valid from the MySQL Agent's location.
- **SSH User** – An SSH user with access to the slow query log's location and file permissions to perform a read action.
- **SSH Password** – The password for the provided SSH user.

Additional Options (optional)

- **Enable Dynamic Memory Allocation** – Allows the agent to request more memory from the FglAM when monitoring a server with over 10,000 tables. Default is false.

- **Agent Host Name** – If a hostname alias is being used for monitoring purposes that is different than the one provided by the MySQL server, provide that here. If using the Infrastructure cartridge for OS monitoring, this name should match any alias provided for OS monitoring in order to link the data correctly.
- **Server Time Zone Override** – Allows the client to override the server time zone, usually for the purpose of correcting a java-incompatible time zone id. A list of acceptable time zone IDs can be found here.
- **Client Time Zone Override** – Allows the client to override the time zone used for time-based performance data. A list of acceptable time zone IDs can be found here.
- **Exclude DB Table/Index Collections** – Table and index collections for individual databases can be disabled here. This will decrease resource usage by the agent and used space in the FMS repository. By default, the list is populated with system databases.
- **Skip Startup Connection Test** – When the agent is first activated, it performs a connection test to ensure that a connection can be made to the MySQL server. If it cannot, it is assumed there is a persistent failure due to incorrect configuration or other issues which need to be resolved and the agent will show as failed until fixed and restarted. In cases like a restart when the agent and the MySQL server are located on the same machine and you expect the agent to activate before the server is accessible, you may want to disable this. If you wish to skip the connection test, set to true.
- **Minimum TRX Block Time (sec)** – This is the minimum wait time for a transaction to be considered deadlocked and thus valid for inclusion in the Blocked Transactions collection. The default time is 15 seconds.
- **Include Views in Table Collection** – Option to include Views in the table collection. Set to false by default.
- **Use Alternate MariaDB JDBC Driver** – Changes driver used by agent from MySQL/J connector to MariaDB driver. This driver can be used for MariaDB or MySQL and is useful as a workaround for a current bug with the MySQL/J connector with MySQL servers located on RHEL servers.
- **Enable MariaDB trustServerCertificate** – Enables a driver option available with the MariaDB driver that causes the client to trust the identity of the server it is connecting to without requiring a valid certificate. This can be enabled to work around certificate issues with little risk since no sensitive data will be sent over this connection to the target server.

Upgrading the Agent

1. Go to **Dashboards > Administration > Cartridges > Cartridge Inventory** and click **Install Cartridge**.
2. Locate the *.car* file on your system and install it with auto-enable selected. If you get a message that a bundled cartridge is of an older version than the one currently enabled on your FMS and will not be enabled, ignore it and continue.

3 Once the cartridge is installed and enabled, go to **Dashboards > Administration > Agents > Agent Managers**. Agent Managers that can be upgraded with newer agent packages will show **yes** in the **Upgradable | Agents column**. Select all Agent Managers you wish to upgrade and click **Upgrade**.

> **NOTE:** If an Agent Manager is not upgradable, check that the Agent Manager version is compatible with the newer agent version. If it is not, the Agent Manager will need to be upgraded first.

Removing Monitored Databases

1 Go to the **Databases** dashboard.

2 Select the databases you wish to remove.

3 Click **Settings**, then click **OK**.

> **NOTE:** : Doing this will remove the monitoring agents as well as the historical data already collected. If you wish to delete only the agents, you can do that on the Administration > Agents > Agent Status page. Because the Databases dashboard only shows databases which are being actively monitored, you will only be able to view these databases by going through the MySQL > MySQL Global View dashboard.

3

Administration

Opening the Databases Administration Dashboard

You can edit agent settings for one or more MySQL instances on the **Databases > Administration** dashboard.

NOTE: If you attempt to select instances of more than one type of database, such as a MySQL database and an Oracle database, an error message is displayed.

To open the Databases Administration dashboard:

1. In the navigation panel, under **Homes**, click **Databases**.
2. Select the check boxes beside one or more MySQL instances.
3. Click **Settings** and then **Administration**. The Administration dashboard opens, containing settings for all the selected agents. Settings are broken down into categories, which are organized under a MySQL tree.

Reviewing the Administration Settings

The Databases Administration dashboard allows settings options for collecting, storing, and displaying data, which apply to all the currently selected agents. Click a category of settings on the left (for example: Connection Details) to open a view containing related settings on the right.

To view the full list of selected agents, click **Selected Agents** at the upper right corner of the screen. To change the list of agents to which the metrics will apply, exit the Databases Administration dashboard, select the requested agents and re-open the view.

Administering SQL Performance Investigator

The SQL Performance Investigator view in the **Administration** dashboard allows you to enable and disable SQL PI monitoring for selected agents.

Customizing Alarms for Foglight for MySQL Rules

Many Foglight for MySQL multiple-severity rules trigger alarms. To improve your monitoring experience, you can customize when alarms are triggered and whether they are reported. You can also set up email notifications.

Introducing the Alarms View

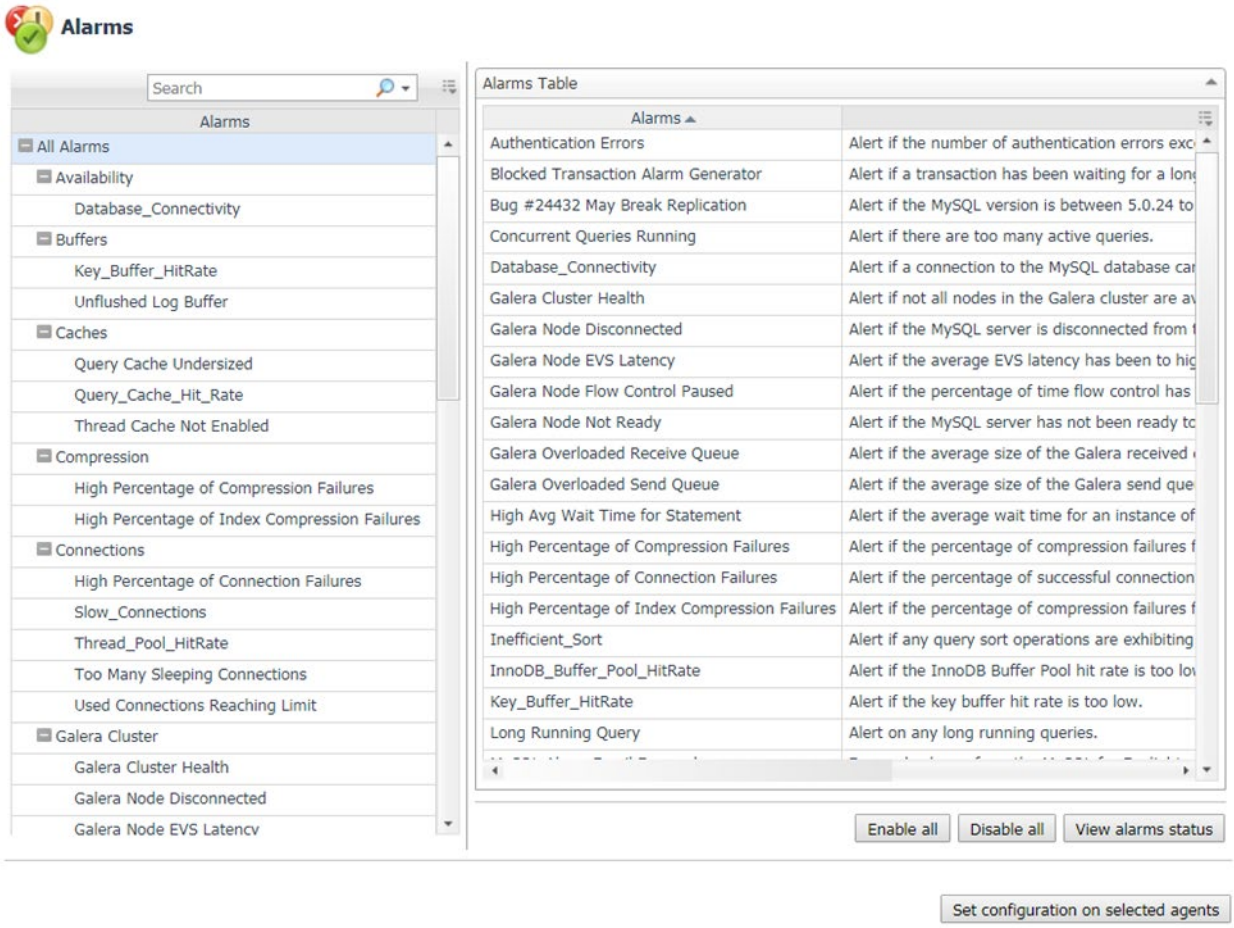
The **Alarms** view enables you to modify global settings and agent-specific settings for alarms.

To open the Alarms view:

1. Open the **Administration** dashboard as described in Opening the Databases Administration Dashboard.
2. Select the agents you wish to modify and do one of the following steps:
   a. Click Settings and open the Administration dashboard, then click Alarms.
   b. Click Configure Alarm.
3. From the **Alarms** view, you can complete the following tasks:
   - Modifying Alarm Settings
   - Reviewing Rule Definitions
   - Cloning Agent Settings

Modifying Alarm Settings

You can customize how the alarms generated by the default rules are triggered and displayed in the Alarm view. Changes to alarm settings will apply to all selected agents, though thresholds can be customized by individual agent.



The Alarms list controls the contents displayed to the right and the tasks that are available.

- **All Alarms** – Displays all rules with configured alarms and indicates whether alarms are enabled. In this view, you can enable or disable alarms for all the rules at once. You can also set email notifications and define mail server settings.
- **Category of rules** – Displays a set of related rules with configured alarms. In this view, you can enable or disable alarms and also set email notifications for the category of rules.
- **Rule name** – Displays the alarm status for the selected rule. If the rule has multiple severity levels, displays the threshold configured for each severity level. In this view, you can enable or disable the alarm, edit the alarm text, and edit severity levels and their thresholds. You can also set email notifications for the alarm.

You can complete the following tasks:

- Enabling or disabling alarms for selected agents
- Modifying alarm threshold values
- Editing the text of the alarm message

Your changes are saved separately and applied over the default rules. This protects you from software upgrades that may change the underlying default rules.

Enabling or disabling alarms for selected agents

You can override the global alarm sensitivity level setting for the selected agents. You can enable or disable alarms for all rules, a category of rules, or an individual rule.

To see descriptions of the rules, follow the steps described in Reviewing Rule Definitions.

To enable or disable alarms:

1. Navigate to the **Alarms** view.
2. Decide on the scope for the change: all alarms, a category of rules, or a selected rule.
3. Complete the steps for the selected scope:

Scope	Procedure
All alarms	Click All Alarms. In the Alarms Settings tab, click either Enable all or Disable all.
Category of rules	Click a category. Click either Enable all or Disable all.
Selected rule	Click the rule. In the Alarms Settings tab, click the link that displays the alarm status. Select Enabled or Disabled from the list and click Set.

Modifying alarm threshold values

You can and should modify the thresholds associated with alarms to better suit your environment. If you find that alarms are firing for conditions that you consider to be acceptable, you can change the threshold values that trigger the alarm. You can also enable or disable severity levels to better suit your environment.

When a rule has severity levels, a **Threshold** section appears in the **Alarm Settings** tab showing the severity levels and bounds by agent. The threshold values correspond to the lower bounds shown in this table. Many rules do not have severity levels and thresholds.

When editing thresholds, ensure that the new values make sense in context with the other threshold values. For most metrics, threshold values are set so that Warning < Critical < Fatal. However, in metrics where normal performance has a higher value, such as DBSS - Buffer Cache Hit Rate, the threshold values are reversed: Warning > Critical > Fatal.

To change severity levels and thresholds:

1. Navigate to the **Alarms** view.
2. Click the multiple-severity rule that you want to edit.
3. Click the **Alarms Settings** tab.

4 In the **Threshold** section, review the defined severity levels and existing threshold bounds for all target agents.

5 Modify the severity levels for one or more agents by following one of the following procedures:

Task	Procedure
Edit severity levels and set threshold values for all agents.	Click Enhance alarm. Select the check boxes for the severity levels you want enabled and set the threshold values. Click Set.
Change the threshold values for one agent.	Click Edit beside the agent name. Set the new threshold values and click Set.
Copy the changes made to one agent's threshold values to all other agents.	Click Edit beside the agent name that has the values you want to copy. Select Set for all agents in table and click Set.

Editing the text of the alarm message

For individual rules, you can change the message displayed when an alarm fires. You cannot add or remove the variables used in the message. This is a global setting that affects all agents.

To change the alarm message:

1 In the **Alarms** view, click the **Settings** tab.

2 Select a rule.

3 Click the **Alarm Settings** tab.

4 Click **Enhance alarm**. A **Customize** <rule> dialog box opens.

5 In the Message box, edit the message text. To restore the default message, click **Reset message**.

6 Click **Set**.

Reviewing Rule Definitions

If you want to review the conditions of a rule, open the rule in the Rule Management dashboard.

IMPORTANT: Avoid editing rules in the Rule Management dashboard unless you are creating your own rules or copies. These rules may be modified during regular software updates and your edits will be lost.

You can create user-defined rules from the Rule Management dashboard. If you want to modify a rule, we recommend copying the rule and creating a user-defined rule. User-defined rules need to

be managed from the Rule Management dashboard; these rules are not displayed in the Alarms view of the Databases Administration dashboard. For help creating rules, open the online help from the Rule Management dashboard.

To open the Rule Management dashboard:

1. On the navigation panel, under **Homes**, click **Administration**.
2. In the Administration dashboard, click **Rules**.
3. Type **MySQL** in the **Search** field to see the list of predefined rules for MySQL databases. The MySQL rules are displayed. From here, you can review threshold values, alarm counts, and descriptions.
4. To see the full rule definition, click a rule and then click **View**, then **Edit**.
5. In the **Rule Detail** dialog box, click **Rule Editor**.
6. When you are done with your review, click **Rule Management** in the breadcrumbs to return to the dialog box.
7. Click **Cancel** to avoid changing the rule unintentionally.

Cloning Agent Settings

You may want an agent to have the same settings as another agent. For example, if you add new agents, you may want them to use the same settings as an existing agent. In this case, you can clone the settings from one agent to other agents. This process does not link the agents; in the future if you update the source agent, you also need to update the target agents.

This procedure walks you through selecting the source agent from the Databases dashboard. However, you can also open the Administration dashboard with multiple agents selected. In this case, you select the source agent in Clone Alarm-related Settings to Other Agents dialog box.

To clone alarm-related settings:

1. On the **Databases** dashboard, select the check box for the agent with the settings you want to clone.
2. Click **Settings**, then **Administration**.
3. In the **Administration** dashboard, click **Alarms**.
4. Click **Set configuration on selected agents**. The **Clone rule settings across agents** dialog box opens.
5. In the **Select the source agent** drop-down list, you should see the agent you selected.
6. In the **Select the target agents table**, select the check boxes for agents that should inherit settings from the source agent.
7. Click **Apply**.
8. When prompted for confirmation, click **Yes**.

Configuring Email Notifications

We recommend that you set email notifications for the alarms you are most interested in tracking closely. For example, you may want to be notified by email of any Critical or Fatal situation. Or you may want to be informed whenever a key metric is no longer operating within acceptable boundaries.

You can set up email notifications that are generated when an alarm fires and/or on a defined schedule, as described in the following topics:

- Configuring an email server
- Defining Default Email settings
- Enabling or disabling email notifications
- Defining email notifications, recipients, and messages
- Defining variables to contain email recipients

Configuring an email server

You need to define the global mail server variables (connection details) to be used for sending email notifications.

The setting of the email should be configured in **Foglight Administration > Email configuration**.

Defining Default Email settings

You can define a default email address to be used by every new agent created in the future, by selecting the Default email button when configuring email notification.

The Email addresses entered are applied to all monitored agents not only for the agents that were selected to enter the Alarm administration.

Enabling or disabling email notifications

You can enable or disable email notifications for all alarms, a category of alarms, or a selected rule. Email notifications are sent only if all the following conditions are met:

- The alarm email notification setting is enabled for the affected rule.
- The alarm is triggered by changes in the monitored environment.
- Alarm notification is enabled at the triggered severity level. See Defining email notifications, recipients, and messages.

To enable or disable email notifications:

1 In the **Alarms** view, click the **Settings** tab.

2 Decide on the scope for the change: all alarms, a category of rules, or a selected rule.

3 Complete the steps for the selected scope:

- All alarms - Click All Alarms. Click the Define Email Settings button. Select either Enabled or Disabled from the Alarms notification status list. Click Set.
- **Category of rules** - Click a category. Click the **Define Email Settings** button. Select either **Enabled** or **Disabled** from the **Alarms notification status** list. Click **Set**.
- **Selected rule** - Click a rule. In the **Alarms Settings** tab, click the **Define Email Settings** tab. Click the link that displays the alarm notification status. Select **Enabled** or **Disabled** and click **Set**.

Defining email notifications, recipients, and messages

You control who receives email messages, the subject line, and some text in the body of the email. The body of the email always contains information about the alarm. This information is not editable. You can also control whether an email is sent based on severity levels. You can set different distribution lists for different rules and different severity levels, or set the same notification policy for all rules.

To configure email notifications:

1 In the **Alarms** view, click the **Settings** tab.

2 Decide on the scope for the change: all alarms, a category of rules, or a selected rule.

3 Complete the steps for the selected scope:

- **All alarms** - Click All Alarms. Click the Define Email Settings button.
- **Category of rules** - Click a category. Click the Define Email Settings button.
- **Selected rule** - Click a rule. Click the Email Notification Settings tab.

4 If you selected **All Alarms** or a category, in the **Email Notification Settings** dialog box, do one of the following:

- To change the severity levels that warrant an email notification, from the Messages will be enabled for severities box, select the desired levels of severity.
- To configure the same email recipients and message for all severity levels, click **Configure mail recipients for all Severities** and then click **All severities**.
- To configure different email recipients and messages for each of the severity levels, click **Configure mail recipients for the following options** and then click a severity level.

5 In the **Message Settings** dialog box, configure the email recipients and message.

NOTE: You can use registry variables in place of email addresses. Type the variable name between two hash (#) symbols, for example: #EmailTeamName#. For more information, see Defining variables to contain email recipients.

- **To** — Type the addresses of the people who need to take action when this alarm triggers.
- **CC** — Type the addresses of the people who want to be notified when the alarm triggers.
- **Subject** — Optional. Edit the text of the subject line to better suit your environment. Avoid editing the variables, which are identified with the @ symbol.
- **Body Prefix** — Optional. Add text that should appear above the alarm information in the body of the email.

6 Click **Set** to save the message configuration and close the dialog box.

7 If the **Edit Notification Settings** dialog box is open, click **Set**.

Defining variables to contain email recipients

You can create registry variables that contain one or more email addresses and use these registry variables when defining email notifications. This procedure describes how to create a registry value.

To create a registry variable:

1 On the navigation panel, under **Dashboards**, click **Administration > Rules & Notifications > Manage Registry Variables**.

2 Click **Add**. The **New Registry Variable Wizard** opens.

3 Select the registry variable type **String** and click **Next**.

4 In the **Name** field, enter a name, for example: EmailTeamName

5 Click **Next**.

6 Select **Static Value**.

7 In the **Enter desired value** box, enter one or more email addresses (separated by commas).

8 Click **Finish**.

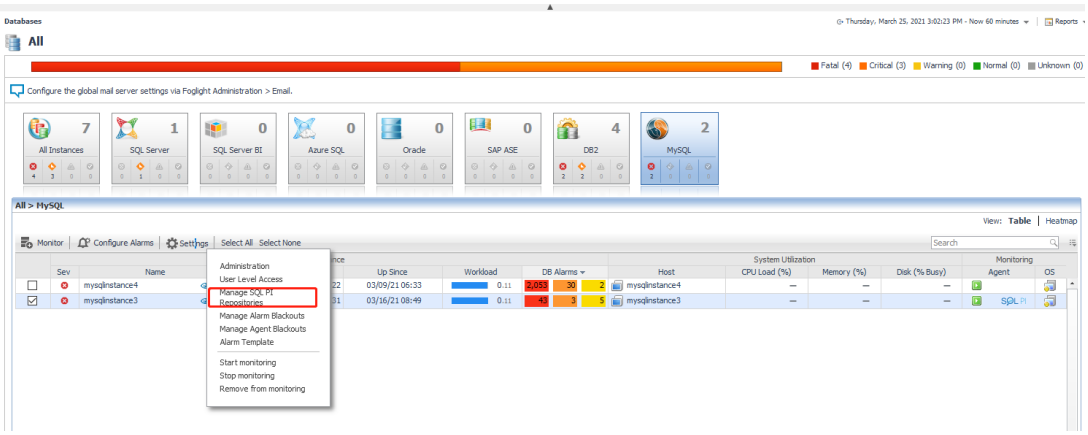
Managing SQL PI Repositories

You can view SQL PI repositories and change SQL PI repository credentials in the Manage SQL PI Repositories dashboard.

To open the Manage SQL PI Repositories dashboard:

1 In the navigation panel, under **Homes**, click **Databases**.

2 Click **Settings** and then click **Manage SQL PI Repositories**.



3 To change SQL PI Repository credentials:

   a Select the radio button of a repository row in the **SQL PI Repositories** table.

   b Click **Set Credentials**.

4 On the **Set SQL PI Repository Credentials** dialog make any required changes to **Authentication**, **Username**, and **Password** as necessary. Click **Set**.

5 The **Confirm Editing Credentials** dialog appears. Click **Yes**.

   a If the given SQL PI repository user does not have sufficient privileges. The **Insufficient Privileges** dialog appears.

   b Enter an admin user and password, and then click **Grant Privileges**.

4

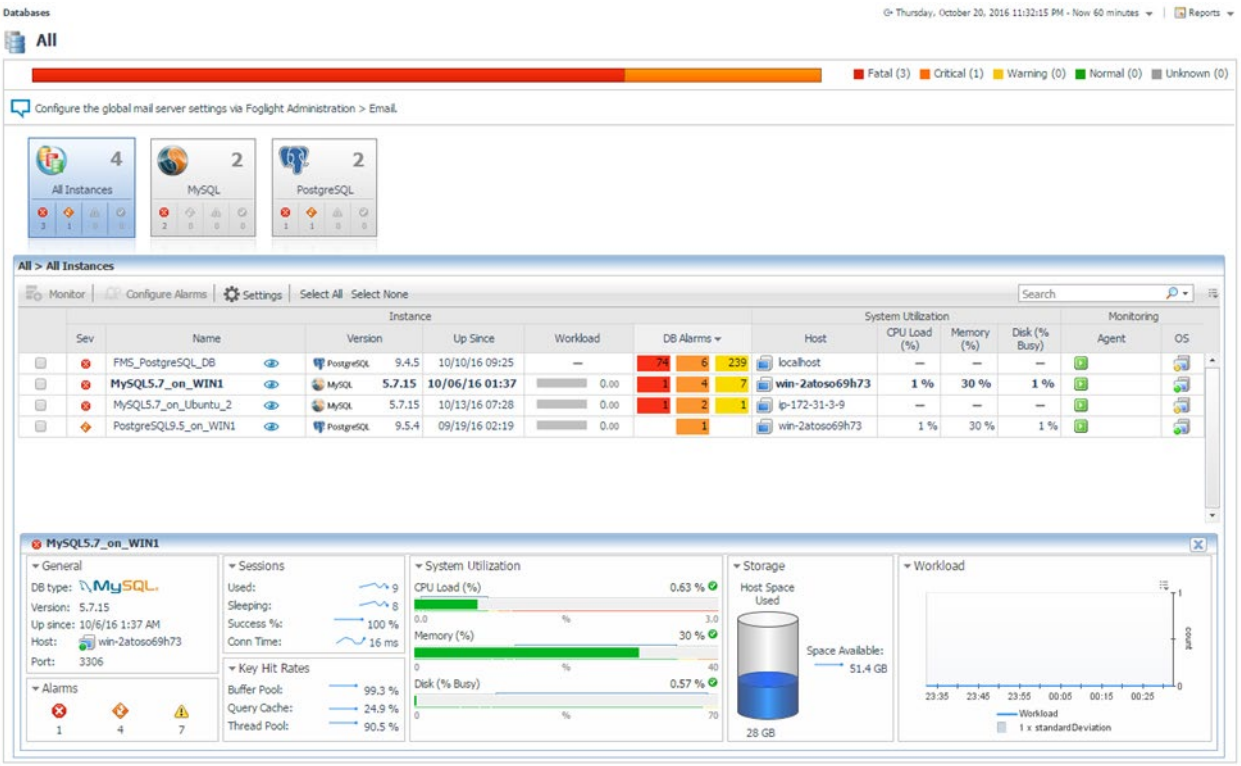
MySQL Dashboards

The installation of the MySQL Cartridge includes the MySQL Dashboards. The MySQL Cartridge offers several principal dashboards:

- Databases
- Global View (deprecated)
- Galera Clusters
- Server Overview
- Server Metrics
- Tables
- Connections
- Statements
- Configuration
- Administration Panel
- Replication Data

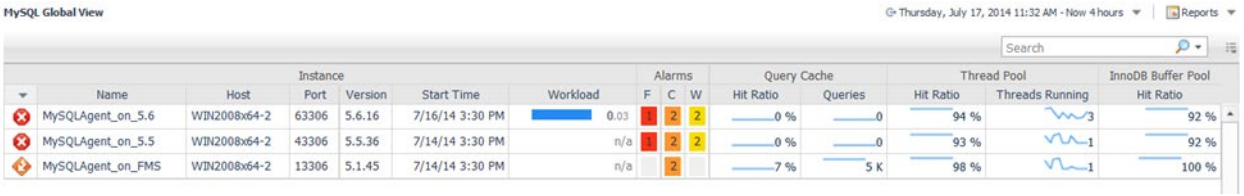
Drill downs from these dashboards are available to expose more granular data. It should be noted that there is a great deal of additional information that is collected that is not displayed in these primary dashboards. Should an operator wish to have that information displayed, Foglight allows for the creation of simple drag and drop dashboards.

Databases



Foglight for MySQL is now incorporated into the Databases dashboard along with any other monitored database types in your environment. Like other products, the list of databases can be filtered by type and severity level and includes basic information, alarms, and host utilization metrics. Clicking on the eye icon will bring up the Quick View with more key information. Clicking the name will drill down to the MySQL Server View.

Global View Dashboard (deprecated)



The MySQL Global View dashboard displays the following metrics, in row format, for each MySQL agent instance deployed:

- **Instance Group**
  - Health Indicator – Shows the overall health of the deployed instance.

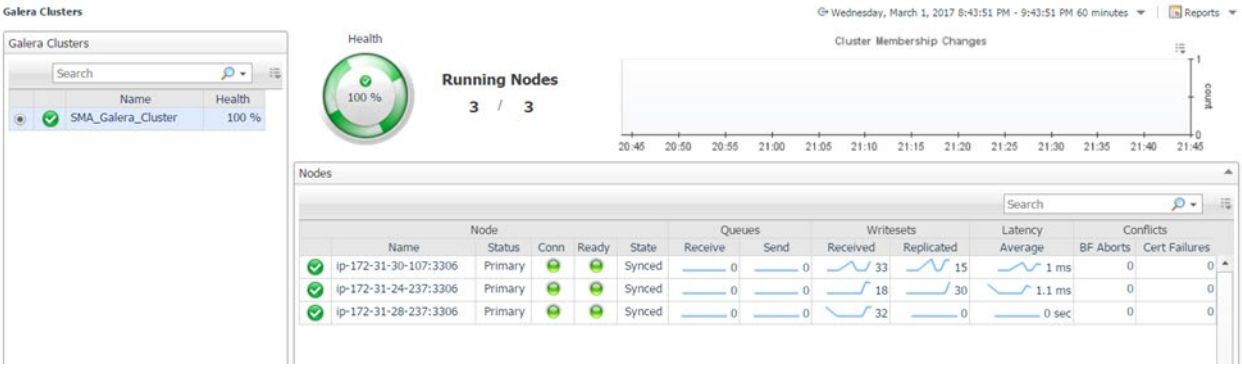
- Name – The “Instance Name” of the MySQL agent. This is the name entered when the MySQL agent instance was created.
- Host – Host Name of the MySQL agent instance.
- MySQL Version – The version of the MySQL database being monitored.

- **Alarms Group**
  - Fatal – Provides a count of the “Fatal” alerts for this agent.
  - Critical – Provides a count of the “Critical” alerts for this agent.
  - Warning – Provides a count of the “Warning” alerts for this agent.
- **Query Cache Group**
  - Hit Ratio – Provides a historical trend and a current value of the Query Cache Hit Ratio for the selected time range.
  - Queries – Provides a historical trend and a current value of the number of SQL statements that are in cache for the selected time range.
- **Thread Pool Group**
  - Hit Ratio – Provides a historical trend and a current value of the ratio of the number of threads that were used from the pool to the total number of connections made for the selected time range.
  - Threads Running – Provides a historical trend and a current value of the number of active (non-sleeping) threads for the selected time range.
- **InnoDB Buffer Pool**
  - Hit Ratio – Provides a historical trend and a current value of the InnoDB BufferPool Hit Ratio for the selected time range.

To get additional information for some of the metrics, the MySQL Global View dashboard provides a dwell capability (by hovering over the metric) and drill down (clicking on the metric) functionality built-in.

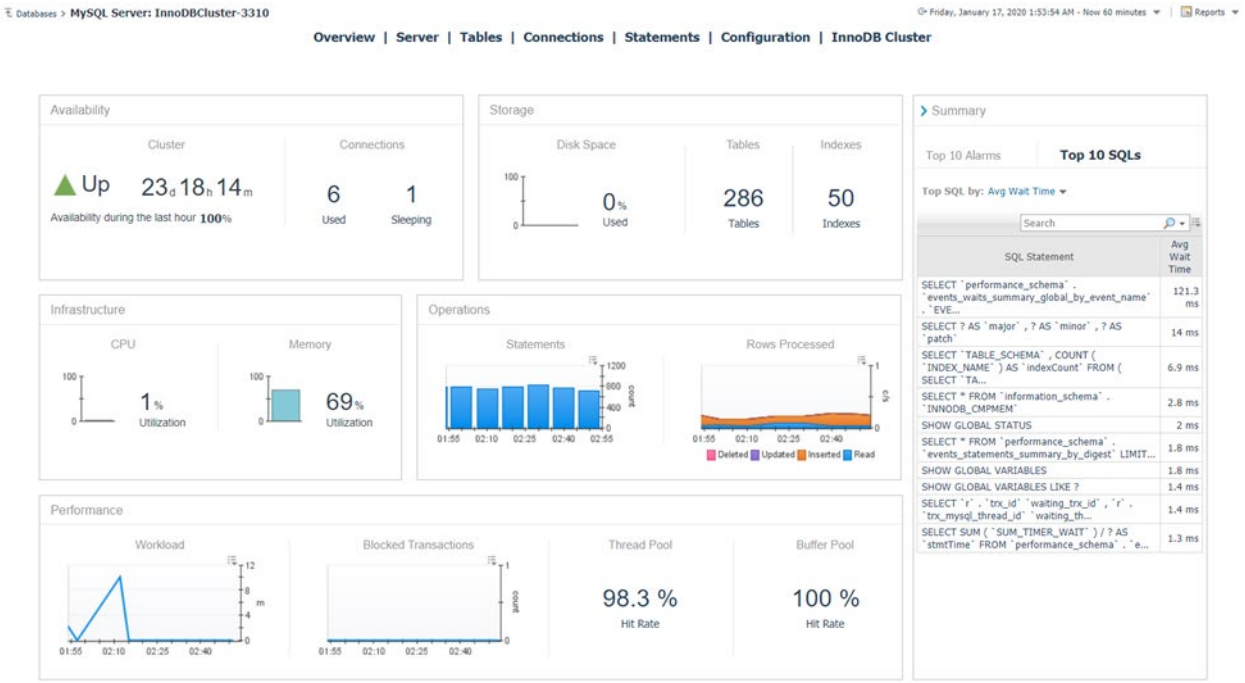
The dwell (hovering over the metric) functionality provides a graphical trend for the selected metric over a specified time range

Galera Clusters

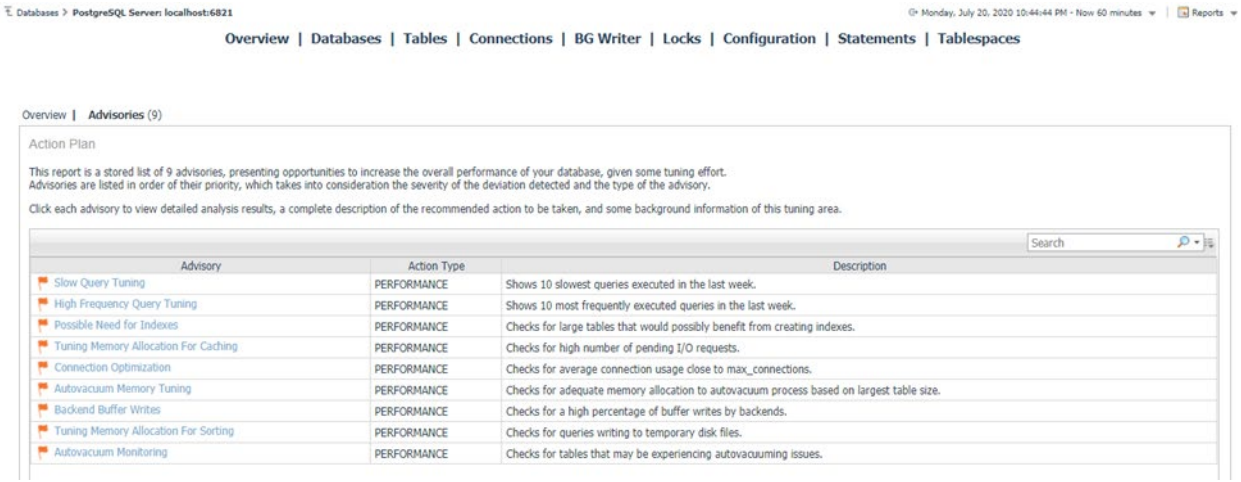


The Galera Clusters dashboard displays all monitored Galera clusters in your environment. The table on the left lists the Galera clusters and current state and health - the percentage of monitored Galera nodes connected to the cluster. When a cluster is selected, the right panel shows a cluster summary, including a health spinner with a time plot popup, the number of running/monitored nodes, cluster membership changes for the selected time window, and a list of nodes. The nodes table displays each monitored node in the cluster with current states, queue sizes, writeset counts, latency, and conflicts. Clicking on the node name drills down to the Galera Node page.

Server Overview



This page displays an overview of the MySQL server with availability, performance, storage metrics and the top 10 alarms and SQL statements. The right side of the page features an expandable summary box that can display the Top 10 SQL executed in the current time range by one of several metrics or the Top 10 Alarms currently active for this server. The SQL statements can be ordered by the metrics in the dropdown box.

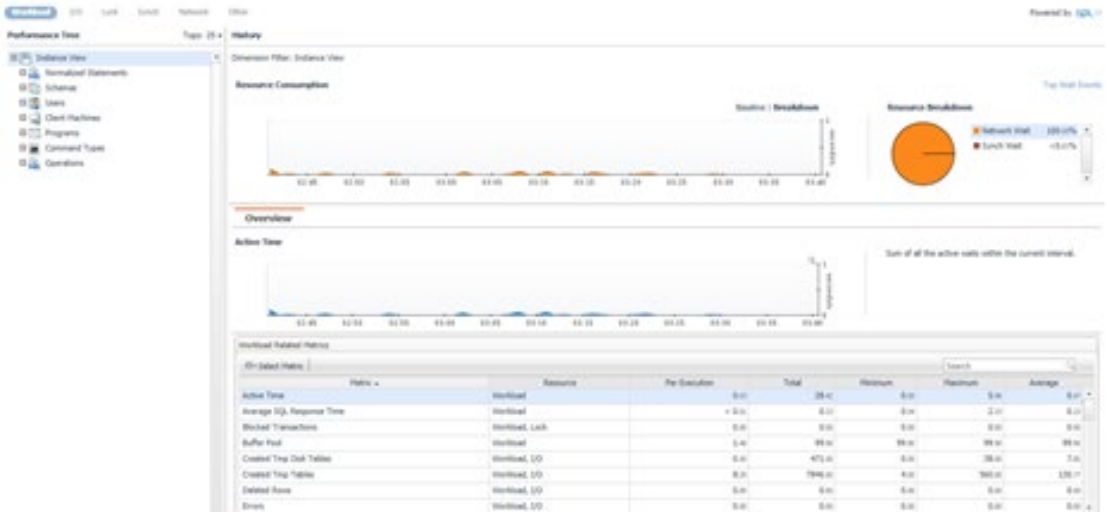


A link at the top left shows the number of active Advisories and allows switching between the Overview and Advisories pages. Clicking on any of the listed Advisories will launch a popup of the full Advisory, which contains information and advice on improving the performance, security, or other aspects of the monitored server.

SQL Performance Investigator

NOTE: This dashboard will not be available if SQL Performance Investigator is not enabled.

The SQL PI view provides the ability to perform a more in-depth analysis and investigation of the database activity and resource consumption.



Performance Tree

The performance tree provides iterative (up to three levels) access to any of the key dimensions associated with MySQL Database activity, based on the OLAP multidimensional model and a database view of the database activity. Domain nodes offer a hierarchical view of all types of MySQL Database activity characteristics.

Selecting a dimension from the tree determines what subset of activity is displayed. Iterative drill-down into domains of interest provides increasingly refined focus and diagnosis.

For example, to begin the investigation by first identifying the most active User, follow the steps described below:

1. Select the **Users** node to display the most active database users in the selected time range. That is the database users who consumed the highest amount of the selected resource.
2. Select the first user to focus the entire window on that user's activity.
3. Identify the most demanding SQL statement that this specific user has executed, by expanding the user node and then selecting the Normalized statement dimension node. This displays the most active Normalized statements executed by this user.
4. Select a specific Normalized statement to focus the entire window on the selected statement's activity.
5. Select Client Machines under the selected Normalized Statement, to view the computers on which the statement was run.

In a similar manner, such iterative drill-downs can be carried out into any MySQL Database dimension of interest, to gain a complete understanding of the causes of its behavior.

The default Azure SQL Database dimensions are as follows:

- **Normalized Statements** – The executed SQL queries.
- **Schemas** – The default database for the statement
- **Users**-- MySQL Database login names used for logging in to MySQL Database.
- **Client Machines** — The machines on which the client executable (connected to MySQL Database) is running.
- **Programs** — names of the programs that connected to MySQL Database and executed the SQL statements.
- **Command Types** — Executed SQL command type (for example, INSERT and SELECT).
- **Operations** – The type of operation performed, such as `lock, read,` or `write.`

Viewing Historical Metrics

The History section view is divided into two sections that are correlated to each other:

- **Resource consumption charts** — This section displays data in five different charts:
  - **Workload chart** — Displays the database resource activity over the selected time frame by emphasizing the resources by colors.
    - **Baseline chart** — Displays the database workload compared to the baseline over time.
    - **Breakdown chart** — Activity of the database by second.
  - **Resource Breakdown pie chart** — Displays the resource breakdown usage by % of the total database activity.
  - **Top Wait Events pop up** — Displays details of the wait events that the database is waiting on during the selected time range.
- **Overview section-** Displays a graphical representation of the metrics highlighted in the Workload related Metrics table below.
  - **Workload Related Metrics** - A table that displays a variety of resource consumption metrics which can give an in-depth of the database activity, each resource holds its default metrics.

Selecting each dimension in the performance tree together with a specific resource effects the data displayed for each level.

- For example, by selecting the Lock resource the Database view dimension will present only locks related data, the Normalized Statements dimension will present only the statements that were experiencing locks and DB users the were experiencing locks and so on through all the dimensions and resources.

Server Metrics

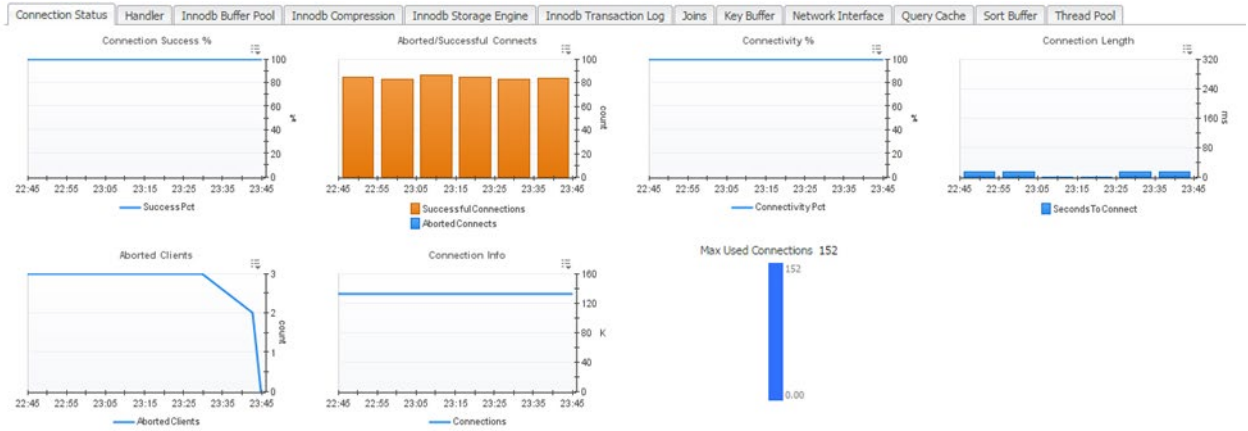
The Server Metrics Dashboard is accessible through either the navigation panel or by clicking on the Sorts, Query Cache, Thread Pool, Buffer Pool, or Storage Engine table data in the MySQL Server Dashboard. When accessing the Server Metrics dashboard from the MySQL Server dashboard the selected tab will default to the same category as the table that was clicked.

Server Metrics Health Overview

Located at the top of the Server Metrics page, overview boxes show the current health state and all active alarms for each component of the agent's data collection. Clicking or dwelling over the health

icon or outstanding alarms will bring up additional information related to the state of that component.

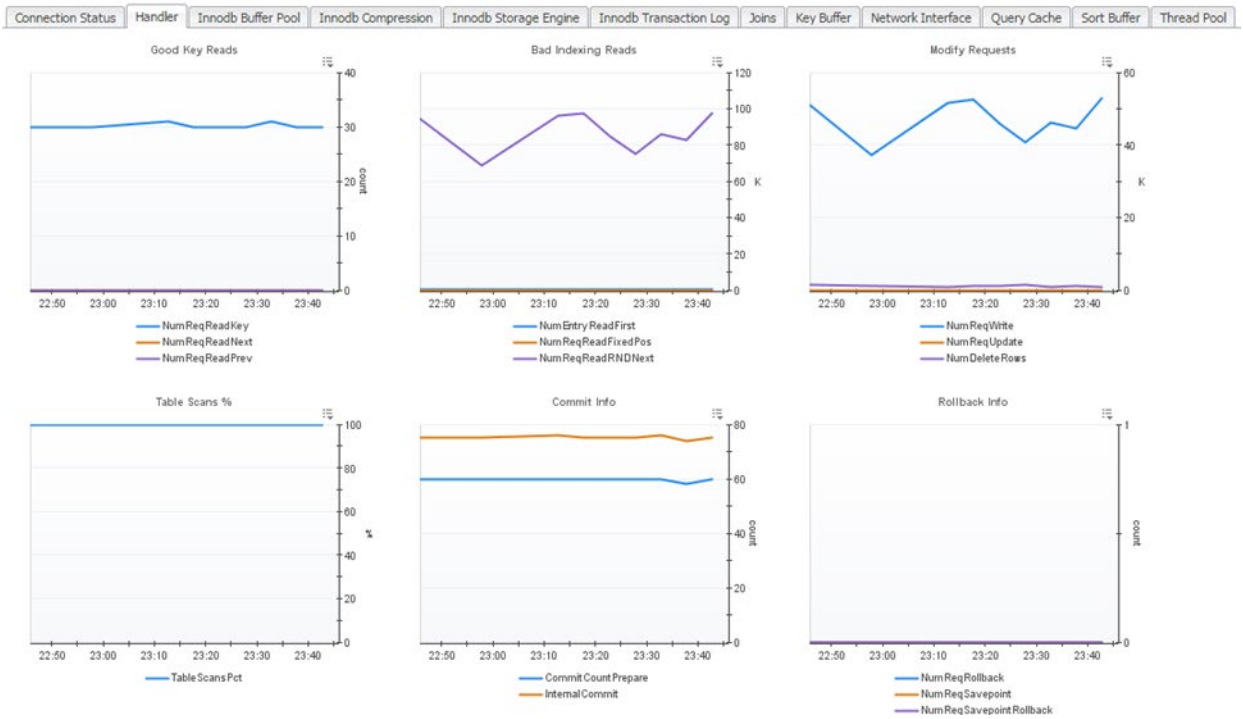
Connection Status



Charts and Metrics (left to right):

- **Max Used Connections** - The maximum number of connections that have been in use simultaneously since the server started.
- **Connection Info** - The number of connection attempts (successful or not) to the MySQL server.
- **Connectivity %** - The probability of making a successful new connection.
- **Connection Length** – The number of seconds MySQL took to establish a new connection.

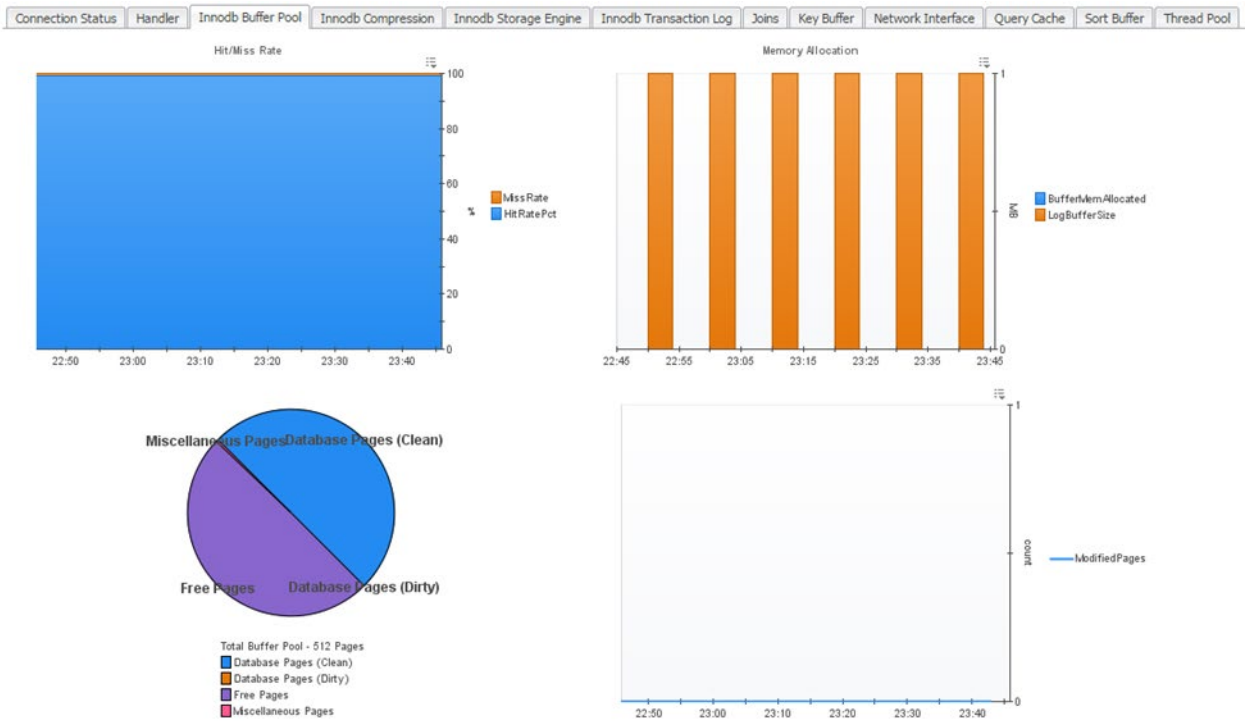
Handler



Charts and Metrics (left to right, top to bottom):

- **Good Key Reads** – Number of reads employing table keys, usually indicative of good indexing.
- **Bad Indexing Reads** – Number of reads which typically indicate bad indexing.
- **Modify Requests** – Requests related to modifying table data through writing, updating, or deleting rows.
- **Commit Info** – MySQL metrics related to transaction commits.
- **Rollback Info** - MySQL metrics related to transaction rollbacks.

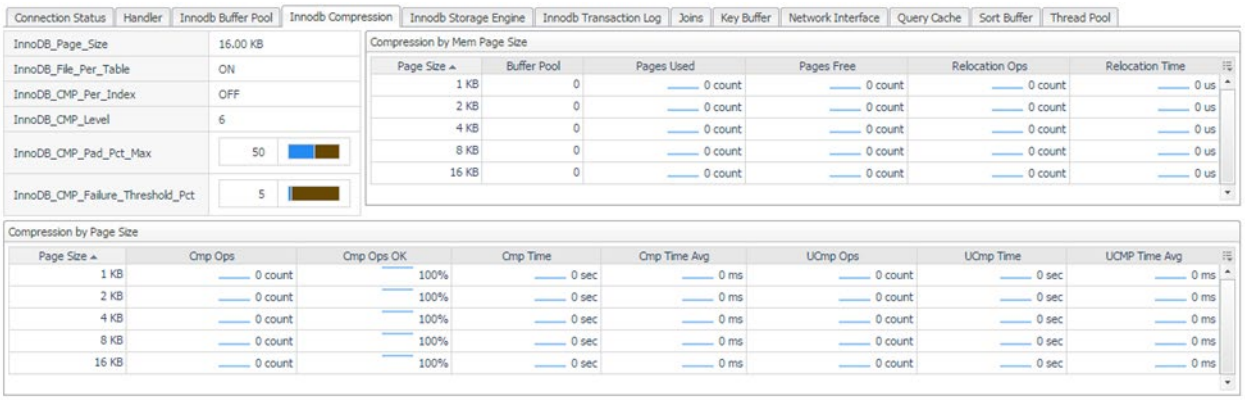
Innodb Buffer Pool



Charts and Metrics (left to right, top to bottom):

- **Hit/Miss Rate** – Hit and Miss rate percentages for InnoDB Buffer Pool.
- **Memory Allocation** – Amount of memory allocated to the InnoDB Buffer Pool and Log Buffer Size.
- **Buffer Pool Pages** – Breakdown of Buffer Pages by type.
- **Modified Pages** – Number of Buffer Pages being modified.

Innodb Compression



Charts and Metrics (left to right, top to bottom):

- **Properties** - InnoDB compression-related variables
- **Compression by Mem Page Size** – Compression metrics grouped by memory page size.
- **Compression by Page Size** – Compression metrics grouped by page size.

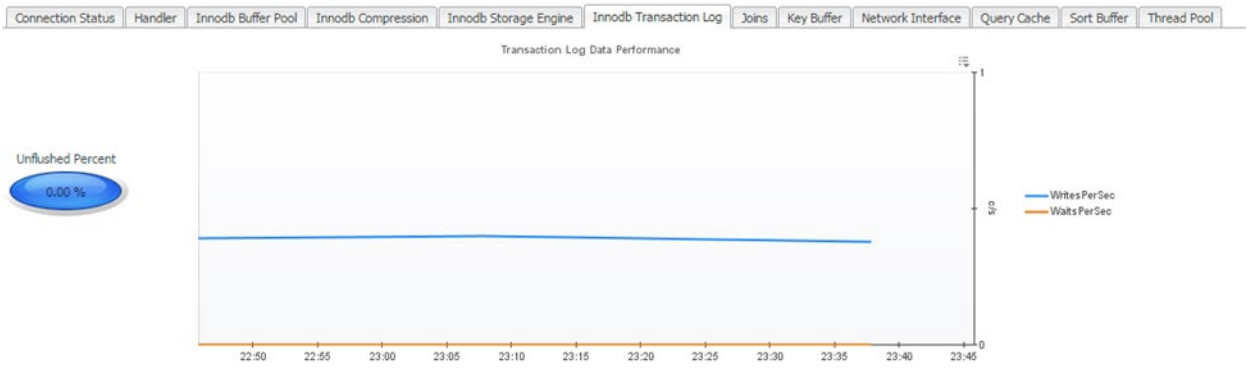
Innodb Storage Engine

Charts and Metrics (left to right, top to bottom):

- **Queries in Queue** – The number of queries waiting to enter InnoDB.
- **Queries inside InnoDB** – The number of queries active inside InnoDB.
- **Rows Affected by Query Type** – The amount of rows per second which are being affected, shown according to the type of query affecting them.
- **Data Reads/Writes** – The number of physical data reads and writes per second.
- **Insert Buffer Async I/O** – The number of insert buffer asynchronous I/O reads and writes per second.
- **Pending Normal Async I/O** – The number of pending normal asynchronous I/O reads and writes per second.
- **Mutex Info** – Information relating to InnoDB waiting for a mutex.
- **RW Locks** – Information relating to InnoDB waiting for a read/write lock to be released.
- **OS File I/O** – The rate of Operating System reads, writes, and fsync operations per second.

- **Transactions Rate** – The InnoDB transaction rate graphed over time.
- **Transaction Purge Lag** – The transaction purge lag graphed over time.

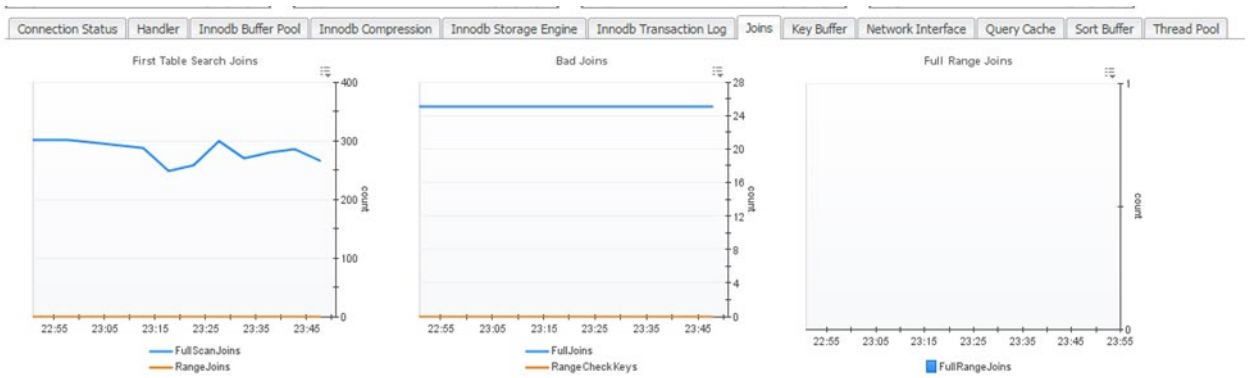
InnoDB Transaction Log



Charts and Metrics (left to right):

- **Unflushed Percent** – The percent of InnoDB log buffer that has yet to be flushed to disk.
- **Transaction Log Data Performance** – The rate of writes and waits for the Transaction Log to be flushed to disk.

Joins



Charts and Metrics (left to right):

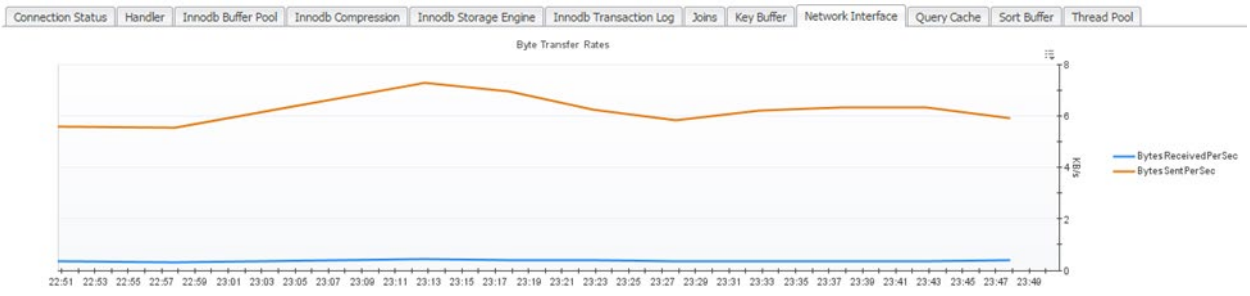
- **First Table Search Joins** – The number of joins operating on the first table.
- **Bad Joins** – Inefficient joins which do not use keys or indexes.
- **Full Range Joins** – The number of joins that used a range search on a reference table.

Key Buffer

Charts and Metrics (left to right, top to bottom):

- **Key Reads** – The number of reads or read requests for a key block.
- **Key Writes** – The number of write or write requests for a key block.
- **Key Hit Rate** – The rate of physical reads from a disk to requests to read from the cache.
- **Key Blocks** – A status breakdown of key blocks in the key caches.

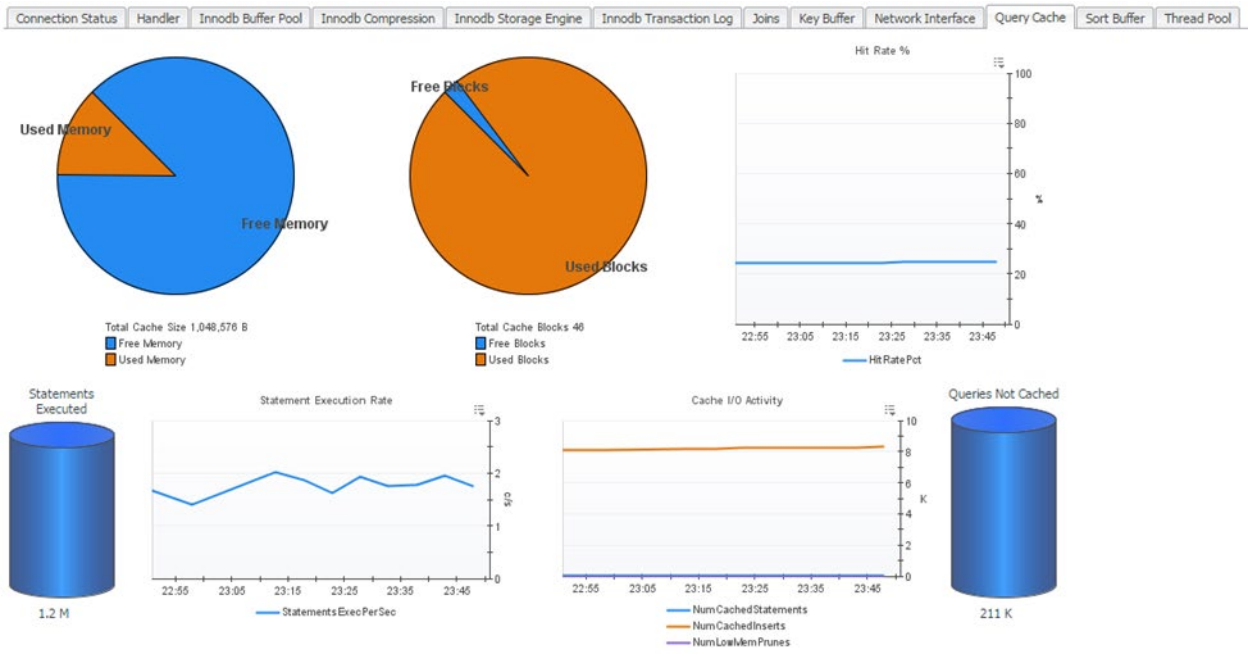
Network Interface



Charts and Metrics (left to right):

- **Byte Transfer Rates** - The number of bytes received and sent per second from and to the network

Query Cache



Charts and Metrics (left to right, top to bottom):

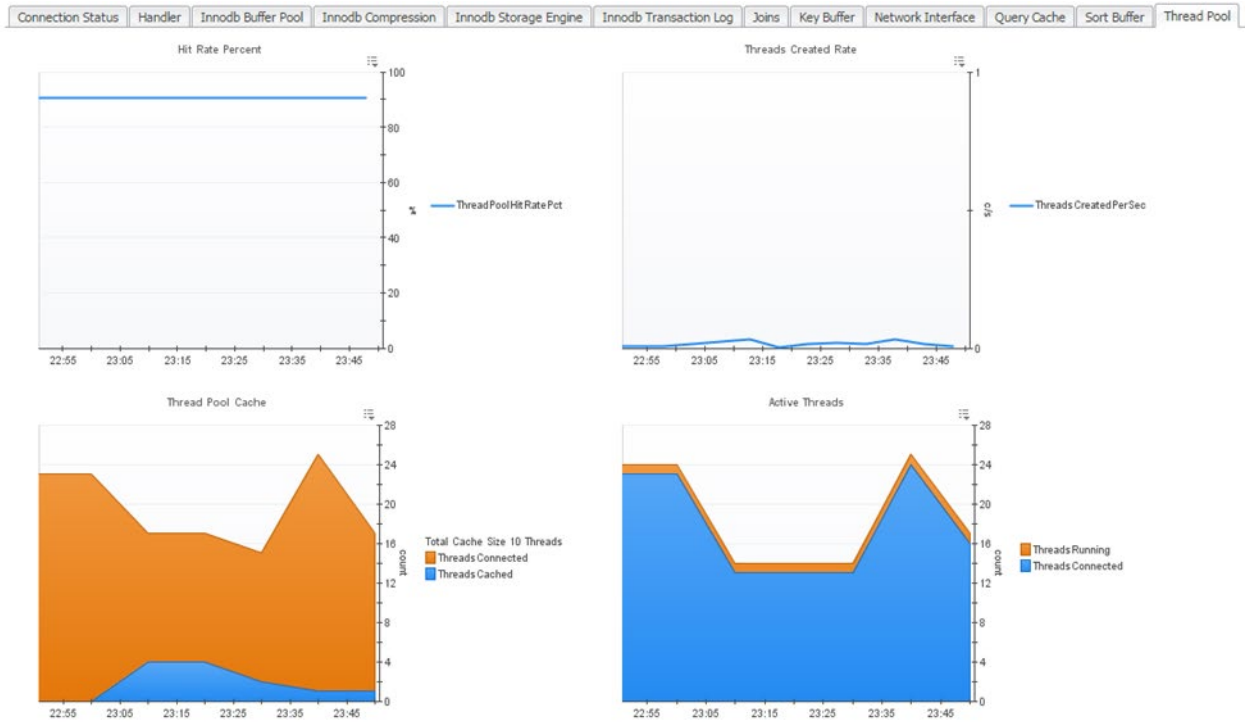
- **Cache Memory Usage** – A breakdown of the total cache size into free and used memory.
- **Cache Memory Blocks** – A breakdown of the total cache blocks into free and used blocks.
- **Hit Rate %** - The percent of SQL queries used from the Query Cache.
- **Statements Executed** – The number of SQL statements currently being executed.
- **Statement Execution Rate** – The number of SQL statements per second that are currently being executed.
- **Cache I/O Activity** – The number of statements being added to the cache, existing in the cache, and being dropped from the cache.
- **Queries Not Cached** – The number of non-cached queries.

Sort Buffer

Charts and Metrics (left to right):

- **Sort Buffer Size** – The buffer size that each thread needs to allocate to do a sort.
- **Row Sort Rate** – The number of rows being sorted per second.
- **Sorting Rates** – The sorting rates per second broken down by type of sort.

Thread Pool

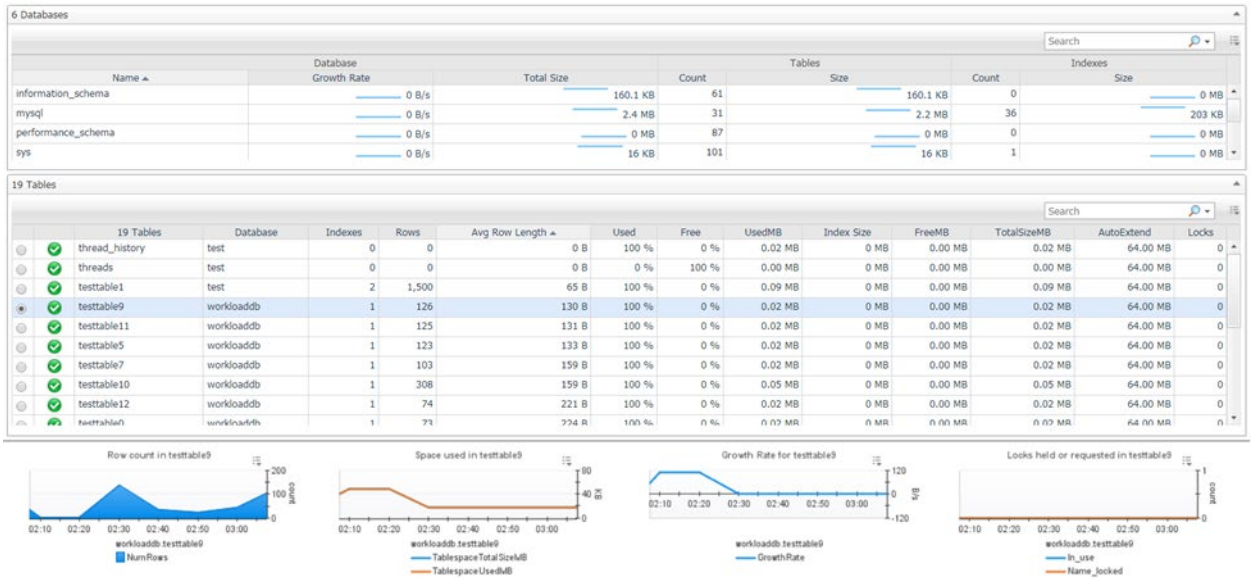


Charts and Metrics (left to right, top to bottom):

- **Hit Rate Percent** – The number of threads that were used from the pool in comparison to the total number of connections made.

- **Threads Created Rate** – The number of new threads created per second.
- **Thread Pool Cache** – The number of threads in the Thread Pool and currently open connections.
- **Active Threads** – The number of currently open connections and active threads.

Tables



The Tables dashboard provides pertinent table information for the MySQL instance being monitored.

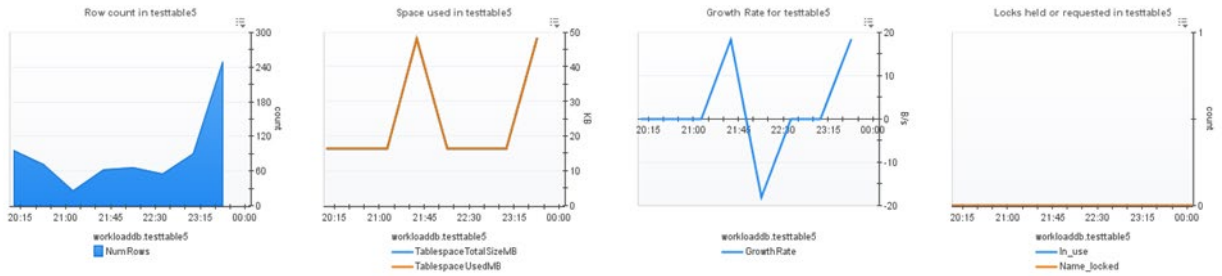
The Databases table shows aggregated metrics for tables in that database, including growth rate and table and index counts and sizes.

The Tables table displays the following metrics, in row format, for each MySQL agent instance deployed:

- **Selector button** – used to select a specific table for the graphs below.
- **Health Indicator** – Indicates the overall health of the deployed instance.
- **<Count> Tables** – provides the name and a total count of tables in the MySQL instance being monitored
- **Indexes** – The number of indexes in the table
- **Rows** – The number of rows in the table
- **Avg Row Length** – Average size of rows in the table
- **Used** – Percentage of used space in the table

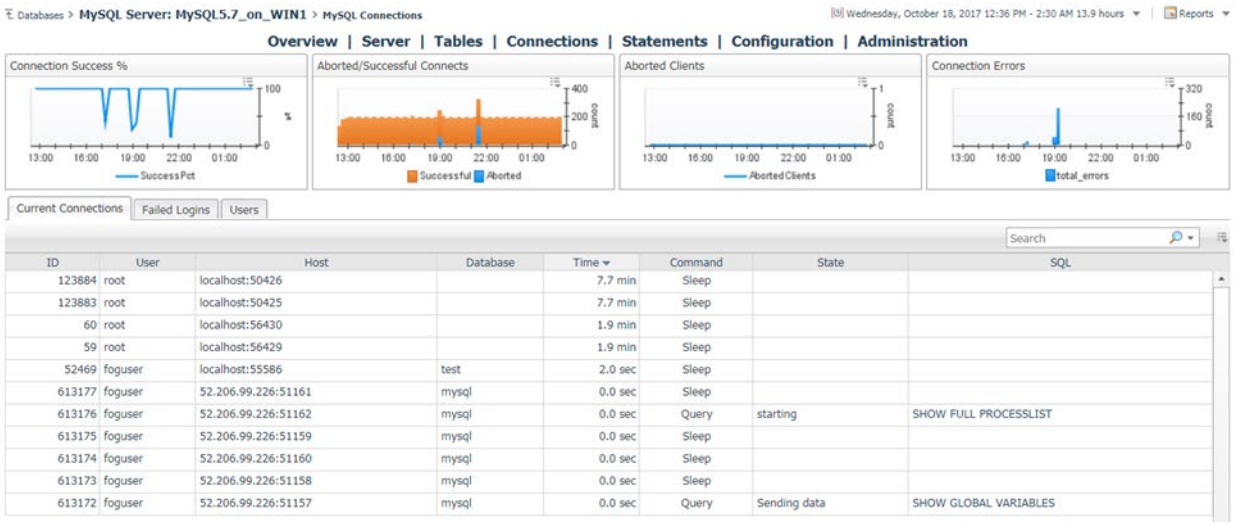
- **Free** – Percentage of free space in the table
- **UsedMB** – Megabytes of used space in the table
- **FreeMB** – Megabytes of free space in the table
- **TotalSizeMB** – Megabytes of total size in the table
- **AutoExtend** – The increment size for extending the size of an auto-extending table when it becomes full.

The graphs below provide the following information for the selected table.



- **Row Count** – Graphs the number of rows in the tablespace for the selected time range
- **Space Used** – Graphs the Table's Used Space vs. Total Size for the selected time range.
- **Growth Rate** – Shows growth/shrinkage of selected table size.
- **Locks Held or Requested** – Shows locks held or requested for selected table/

Connections



ID	User	Host	Database	Time	Command	State	SQL
123884	root	localhost:50426		7.7 min	Sleep		
123883	root	localhost:50425		7.7 min	Sleep		
60	root	localhost:56430		1.9 min	Sleep		
59	root	localhost:56429		1.9 min	Sleep		
52469	foguser	localhost:55586	test	2.0 sec	Sleep		
613177	foguser	52.206.99.226:51161	mysql	0.0 sec	Sleep		
613176	foguser	52.206.99.226:51162	mysql	0.0 sec	Query	starting	SHOW FULL PROCESSLIST
613175	foguser	52.206.99.226:51159	mysql	0.0 sec	Sleep		
613174	foguser	52.206.99.226:51160	mysql	0.0 sec	Sleep		
613173	foguser	52.206.99.226:51158	mysql	0.0 sec	Sleep		
613172	foguser	52.206.99.226:51157	mysql	0.0 sec	Query	Sending data	SHOW GLOBAL VARIABLES

The Connections page shows graphs of some key connection metrics and three tabs with more information:

- **Current Connections** - The MySQL process list at the time of the last sample
- **Failed Logins** – This section shows all hosts that have made failed login attempts in the selected time range, along with error counts for all connection error types.
- **Users** - A list of MySQL users and privileges, along with password status, current connections, and total connections in the selected time range.

Statements

The MySQL Statement Digests dashboard is available for MySQL 5.6.5+ servers with the performance_schema engine enabled. Statement digests are normalized statements which have the same operation plan. Even if values in the statement differ, the server is performing the same operations and can therefore be aggregated for the purposes of analysis. The agent collects these from the server along with raw and calculated statistical data and provides them to the FMS.

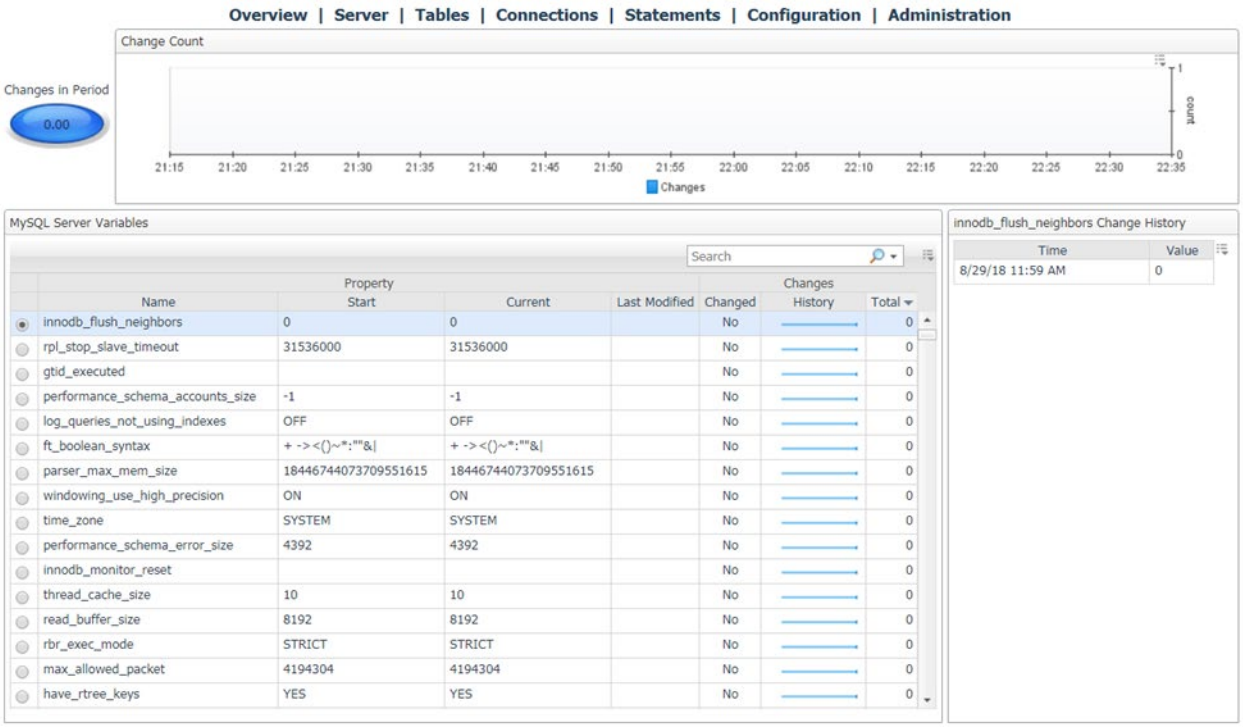
The main time plot displays the top X statements by one of several metrics that can be selected using the provided dropdown. Below that are several graphs which can be correlated with statement performance to determine impact on the system.

On the bottom half of the page is a table listing all statement digests gathered from the server. The number of statement digests which the server maintains varies by version and is set with performance_schema_digests_size. If the number of digest types exceeds that, excess statements will be put into a single digest, also collected. All statistics shown in the table are period values for the selected time range with the exception of First/Last seen and Min/Max Wait Time. By default, an advanced filter is set for the table to exclude statements performed on the system databases mysql, performance_schema, and information_schema. This can be disabled by clicking the ‘x’ in the filter text box. Selecting a row of the table will update the bottom portion with graphed metrics related to the selected digest.

Page actions available in the right-hand panel include the MySQL Agent Selector and an action to Delete Old Statements. When the MySQL server is restarted or the performance_schema’s statement digests table is truncated, digest statistics will be reset. If a previously collected statement is executed on the server again, it will be merged with the historical data preserved by Foglight. If it is never executed again, it will become a dead object. The Delete Old Statements action is a

convenience action to delete these dead topology objects and remove them and their history from the FMS.

Configuration



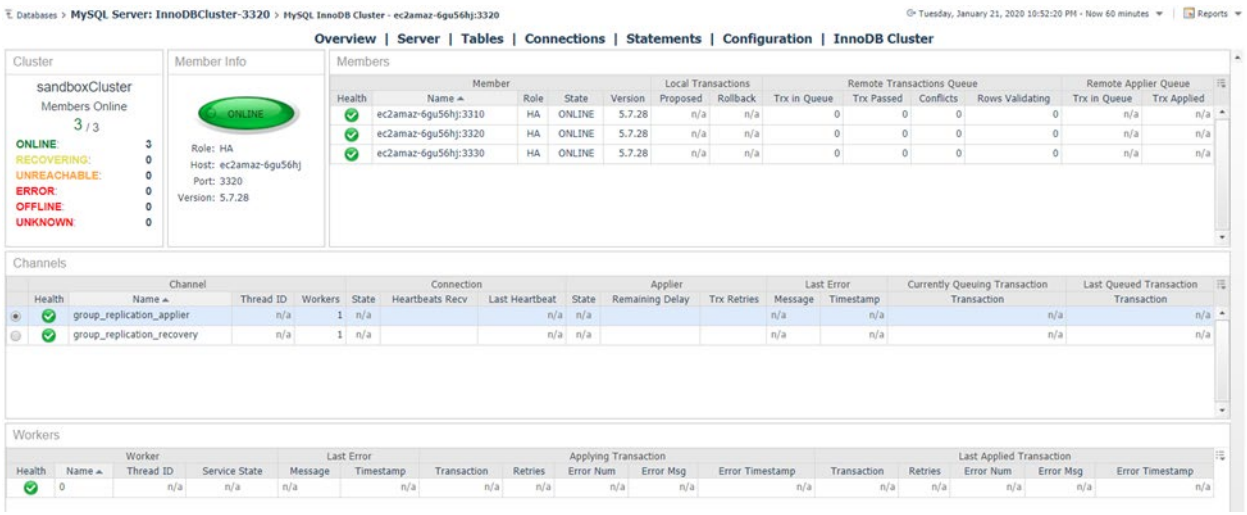
The configuration page tracks the MySQL server configuration. The metric indicator and change count graph shows numbers of changes in the selected time range. The table displays every server variable, showing the name, value at beginning and end of the selected time range, the date of the last modification since server monitoring began, and a history of changes for that variable.

By selecting a variable, you can view a history of changes to that variable in the panel on the left for as long as the data is retained.

Galera Node

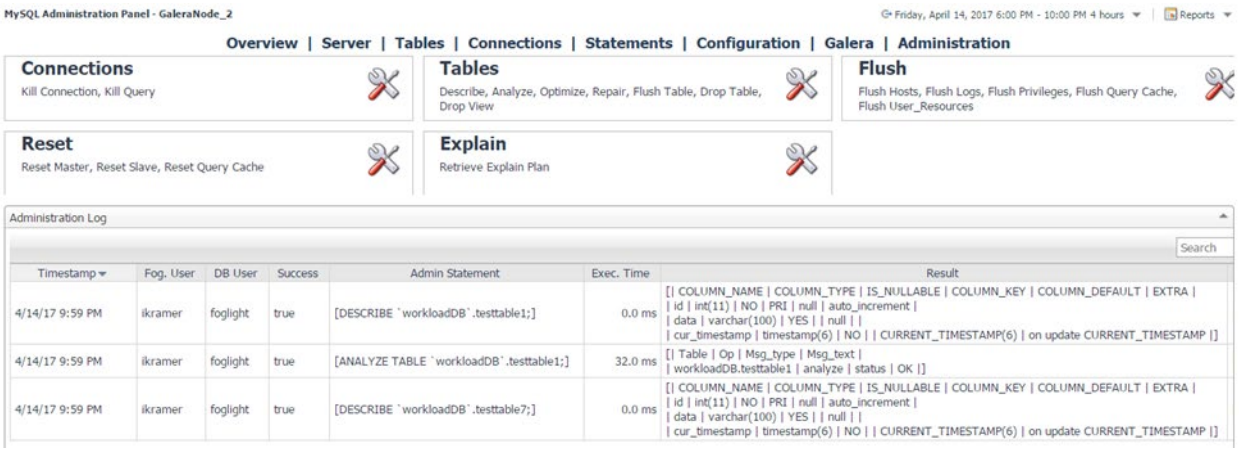
The Galera Node page shows Galera-related information for a MySQL server configured as a Galera node. This page shows the current status of the Galera nodes as well as relevant server variables and informational and performance metrics for writesets, flow control, latency, transactions, and replicated data size. At the top right is a summary for the Galera cluster which the node is a part of, showing the number of collected / monitored nodes for the cluster and listing each node in the cluster and their current states. By clicking the name of a different node, the page will update to feature information on the selected node.

InnoDB Cluster



The InnoDB Cluster page is available for MySQL servers that are part of an InnoDB cluster. The page displays a summary of the cluster and list of members along with information on channels, coordinators, and workers for the monitored MySQL server. If a MySQL server is monitored, it can be switched to by clicking the name of the server in the Member list. InnoDB Cluster monitoring is supported for versions 5.7+, however some data is unavailable for versions before 8.0. InnoDB cluster monitoring must be enabled in the Agent Properties.

Administration Panel



The MySQL Administration Panel features server operations which can be performed through the console by permitted users. The actions currently available represent the initial offering and later versions will incorporate more possibilities. An action is performed by clicking the icon in the relevant category, which will bring up a menu dialog with options and confirmation of the requested

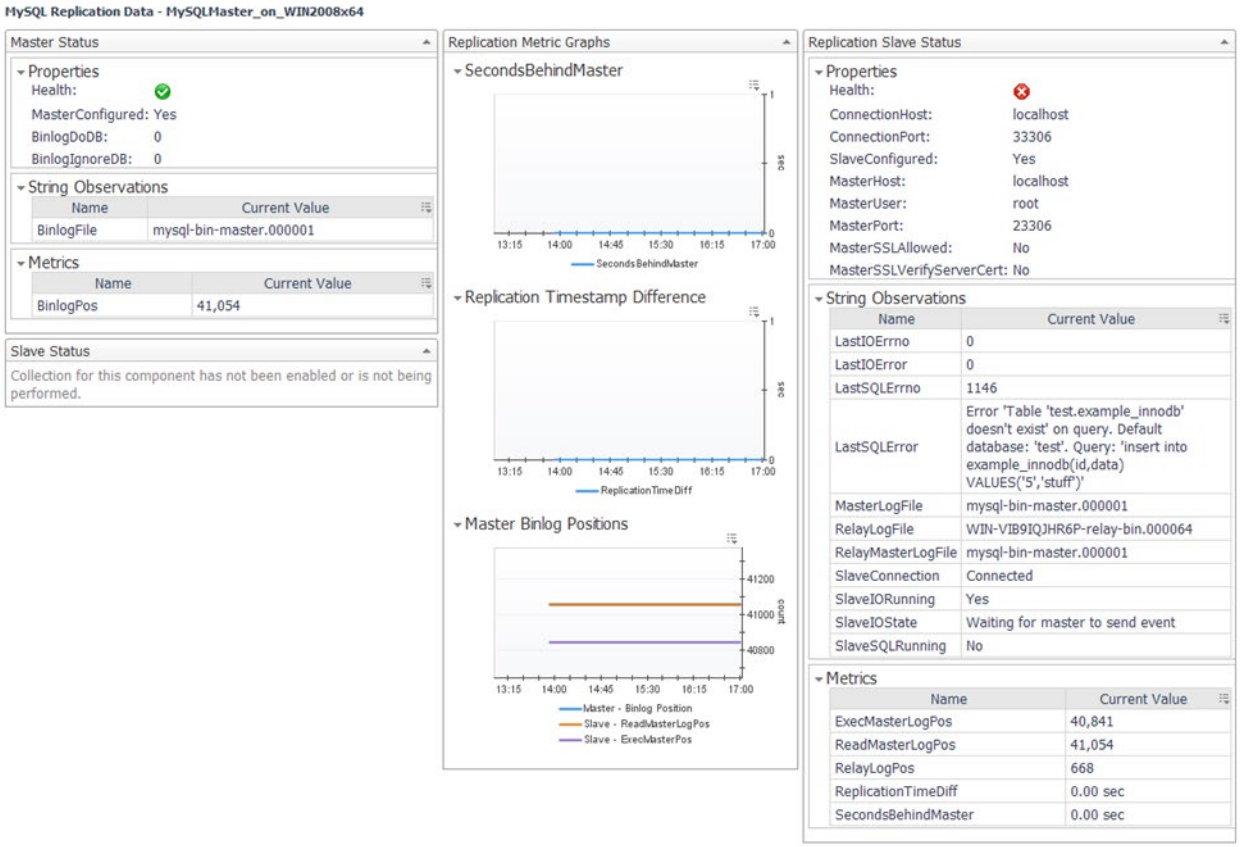
action. Actions can only be performed on one server at a time. The active server can be switched using the MySQL Agent Selector in the right-hand pane. Only agents where administration is enabled will be shown.

The Administration Log below records actions performed in the specified time range for auditing purposes. Information includes a timestamp, the Foglight user who performed the action, the database user employed, the actual statement text, execution time, result (if any) and any resulting SQLExceptions or warnings from the statement.

Prerequisites to use Administrative Actions

1. In the Agent Properties of the agent monitoring the MySQL server, Enable Administration must be set to true.
2. Also in the Agent Properties, a DB user and password must be provided. This user must have the necessary privileges to perform actions taken through the Administration Panel or the action will fail. This user will be solely employed to perform actions on request, not for data collection or other purposes.
3. A Foglight user must have the MySQL Administrator role granted to them in the Administration>Users & Security dashboard in order to access and use the Administration Panel.

Replication Data



This page shows complete data on MySQL Replication collected by a single MySQL agent and will change depending on which collections have been enabled. This page is primarily meant to serve as a demonstration of data collected by the agent, but can be used for simple Master-Slave replication configurations. More complex configurations can use elements of this page in another dashboard.

Slow Queries (deprecated)

The Slow Queries dashboard has been deprecated. For MySQL versions 5.6.5+, the Statements dashboard features more data gathered from the Performance Schema database.

The Slow¬ Queries dashboard displays collected queries and their aggregated metrics as well as several derived metrics to show slow query activity overall. At the top of the page, 3 time plots graph these derived metrics. The first graph shows the number of unique queries and the number of total queries found during each collection period. The second graph shows the average and maximum execution time for all collected queries across time. The third graph shows the average and maximum lock times.

Below that, a dropdown list shows all available data snapshots that took place within the scoped time range of the page. By manipulating the Zonar or the calendar feature at the top right, you can change the data snapshots available in the dropdown as well as the scoped time range of the graphs. In the table displaying the unique queries for that snapshot, the following data is shown, in order from left to right:

- **Time Range (Earliest)** – The first appearance of this unique query in the current collection.
- **Time Range (Latest)** – The last appearance of this unique query in the current collection.
- **Users** – A list of the user(s) who have executed this query.
- **SQL** – A generalized version of the SQL string for this unique query.
- **Count** – The number of instances of this query.
- **Query Time (Avg)** – The average execution time for this query.
- **Query Time (Max)** – The maximum execution time for this query.
- **Lock Time (Avg)** – The average lock time for this query.
- **Lock Time (Max)** – The maximum lock time for this query.

- **Rows Examined Time (Avg)** – The average number of rows examined for this query.
- **Rows Examined (Max)** – The maximum number of rows examined for this query.
- **Rows Sent (Avg)** – The average number of rows sent for this query.
- **Rows Sent (Max)** – The maximum number of rows sent for this query.

In addition to these columns, minimum values for these unique query metrics are available, though hidden. To make these columns visible or hide others, use the customizer at the top-right of the table. You can also use the search bar to filter the list of available queries.

5

Rules

Rule	Description
Authentication Errors	Alert if the number of authentication errors exceed a defined threshold as a percentage of total connection attempts.
Blocked Transaction Alarm Generator	Alert if a transaction has been waiting for a long time to complete.
Concurrent Queries Running	Alert if there are too many active queries.
Database Connectivity	Alert if a connection to the MySQL database cannot be established.
Galera Cluster Health	Alert if not all nodes in the Galera cluster are available.
Galera Node Disconnected	Alert if the MySQL server is disconnected from the Galera cluster.
Galera Node EVS Latency	Alert if the average EVS latency has been too high.
Galera Node Flow Control Paused	Alert if the percentage of time flow control has been paused on the MySQL server has been too high recently.

Rule	Description
Galera Node Not Ready	Alert if the MySQL server has not been ready to accept queries recently.
Galera Overloaded Receive Queue	Alert if the average size of the Galera received queue on the MySQL server has been too high.
Galera Overloaded Send Queue	Alert if the average size of the Galera send queue on the MySQL server has been too high.
High Avg Wait Time for Statement	Alert if the average wait time for an instance of a statement is high compared to its previous average.
High Percentage of Compression Failures	Alert if the percentage of compression failures for a given page size is too high.
High Percentage of Connection Failures	Alert if the percentage of successful connections to the MySQL Server is too low.
High Percentage of Index Compression Failures	Alert if the percentage of compression failures for an index is too high.
Inefficient Sort	Alert if any query sort operations are exhibiting inefficient behavior.
InnoDB Buffer Pool Hit Rate	Alert if the InnoDB Buffer Pool hit rate is too low.
InnoDB Cluster Health	Alerts when at least one node is down with increasing severity when cluster stability is in jeopardy.
Key Buffer Hit Rate	Alert if the key buffer hit rate is too low.

Rule	Description
Long Running Query	Alert on any long running queries.
MySQL Alarm Email Forwarder	Forwards alarms from the MySQL for Foglight cartridge to email when they fire.
MySQL Cleared Alarm Email Forwarder	Forwards alarms from the MySQL for Foglight cartridge to email when they clear.
MySQL-EmailNotification_Rule_General	Sends email notifications for configured alarm severities.
MySQL Server Running Out of Disk Space	Alert if the database server's host will run out of disk space soon given the current database growth rate.
Non-Authentication Errors	Alert if non-authentication errors exceed a defined threshold as a percentage of total connection attempts.
Query Cache Hit Rate	Alert if the query cache hit rate is too low.
Query Cache Undersized	Alert if there are any query cache low memory prunes.
Query Waiting For Table Lock	Alert on queries waiting for a long time on a table lock.
Replication Server Times Out of Sync	Alert if the clocks on the replication servers monitored by the MySQL server are out of sync.
Replication Slave Behind Master	Alert if a replication slave is falling behind the master.

Rule	Description
Replication Slave Connection Unavailable	Alert if a replication slave is unreachable by the Foglight Agent.
Replication Slave I/O in Failed State	Alert if a replication server is either reconnecting or waiting to reconnect after a disconnection event.
Replication Slave I/O Thread Not Running	Alert if the Slave IO Thread for a Replication Slave server is either not running or not connected to a replication master.
Replication Slave SQL Thread Not Running	Alert if the Slave SQL Thread for a Replication Slave server is not running
Slave Behind Master	Alert if a slave is falling behind the master.
Slave I/O in Failed State	Alert if the SlaveIOState for the MySQL server is either reconnecting or waiting to reconnect after a disconnection event.
Slave I/O Thread Not Running	Alert if the Slave IO Thread for the MySQL server is either not running or not connected to a replication master.
Slave SQL Thread Not Running	Alert if the Slave SQL Thread for a Slave server is not running.
Slow Connections	Alert if it is taking a long time to connect.
Slow Query Average Execution Time	Alert if the average execution time for queries written to the slow query log is too long.
Slow Query Max Execution Time	Alert if the maximum execution time for queries written to the slow query log is too long.

Rule	Description
Table Scans Excessive	Alert if the MySQL server does not appear to be using indexes efficiently.
Tablespace Utilization	Alert if the amount of tablespace left in an InnoDB database that does not have auto extend enabled is low.
Thread Cache Not Enabled	Alert if the Thread Cache is not enabled.
Thread Pool Hit Rate	Alert if the Thread Pool hit rate is too low.
Too Many Sleeping Connections	Alert if the MySQL server has too many sleeping connections.
Transaction Purge Lag	Alert if the Transaction Purge Lag is too high.
Unflushed Log Buffer	Alert if percentage of entries in the log buffer waiting to be flushed to disk is too high.
Used Connections Reaching Limit	Alert if the number of connections currently in use for a MySQL server is too high as a percentage of the maximum connections allowed.

6

Reports

Report	Description
Executive Summary	Executive summary of MySQL instance, including workload, availability, and performance metrics.
Failed Logins	Shows failed login attempts to MySQL server in selected timeRange and generated error counts. Galera Cluster Summary
Galera Cluster Summary	Galera cluster health and membership over the selected timeRange as well as key performance metrics from monitored nodes in the cluster.
Health Check Report	Displays various aspects of the specified instance health.
InnoDB Cluster Report	Shows information on a MySQL InnoDB Cluster.
MySQL Server Configuration	Displays current configuration properties for a MySQL Server with change history in specified time range.
MySQL Server Configuration Comparison	Compares the configuration properties of a group of MySQL Servers against a single server used as a template. This report counts number of variances per property and also per server and uses the latest values collected for the server.
Storage Report	Shows information on MySQL server storage capacity, growth rate, etc.
Top MySQL Servers by Connections	Displays top MySQL Servers and related information in selected service, ordered by successful connections, aborted connects, success percentage, or aborted clients.
Top MySQL Tables	Displays top tables and related information in a MySQL Server, ordered by total size, row count, used space percentage, or growth rate.
Top Statement Digests	Displays top statement digests and related information for a MySQL Server, ordered by count, average wait time, sum wait time, max wait time, average lock time, warnings, or errors. performance_schema and mysql DB statements are filtered by default.

Users Report	Shows information on MySQL users, connections, and privileges.