



PERCONA

www.percona.com

Percona Monitoring and Management Documentation

Release 2.5.0

Percona LLC and/or its affiliates 2009-2020

Apr 14, 2020

CONTENTS

I	PMM Concepts	3
1	Client/Server Architecture - an Overview	5
1.1	PMM Client	6
1.2	PMM Server	8
2	Services	11
2.1	MySQL requirements	11
2.1.1	Creating a MySQL User Account to Be Used with PMM	12
2.2	MongoDB requirements	12
2.2.1	Configuring MongoDB for Monitoring in <i>PMM Query Analytics</i>	12
2.2.2	Setting Up the Required Permissions	13
2.2.3	Enabling Profiling	13
2.3	PostgreSQL requirements	14
2.3.1	Supported versions of PostgreSQL	14
II	Installing PMM Server	15
3	Running PMM Server via Docker	17
3.1	Setting Up a Docker Container for PMM Server	17
3.1.1	Pulling the PMM Server Docker Image	17
3.1.2	Creating the pmm-data Container	18
3.1.3	Creating and Launching the PMM Server Container	18
3.1.4	Installing and using specific PMM Server version	19
3.2	Updating PMM Server Using Docker	19
3.2.1	Creating a backup version of the current pmm-server Docker container	20
3.2.2	Pulling a new Docker Image	20
3.2.3	Creating a new Docker container based on the new image	20
3.2.4	Removing the backup container	21
3.3	Backing Up PMM Data from the Docker Container	22
3.4	Restoring the Backed Up Information to the PMM Data Container	22
4	Running PMM Server Using AWS Marketplace	25
4.1	Setting Up a PMM Instance Using the website GUI	28
4.1.1	Setting up a VPC and an EC2 Instance Type	28
4.1.2	Limiting Access to the instance: security group and a key pair	28
4.1.3	Applying settings	29
4.1.4	Adjusting instance settings in the EC2 Console	29
4.1.5	Running the instance	29

5	PMM Server as a Virtual Appliance	39
5.1	Supported Platforms for Running the PMM Server Virtual Appliance	39
5.2	Identifying PMM Server IP Address	40
5.3	Accessing PMM Server	40
5.4	Accessing the Virtual Machine	41
5.5	Next Steps	42
6	Verifying PMM Server	43
7	Configuring PMM Server	45
7.1	PMM Settings Page	45
7.1.1	Settings	45
7.1.2	Telemetry	46
7.1.3	SSH Key Details	47
7.1.4	AlertManager integration	47
7.1.5	Diagnostics	48
7.2	Exploring PMM API	48
III	Installing PMM Client	53
8	Installing Clients	55
9	Installing DEB packages using apt-get	57
10	Installing RPM packages using yum	59
IV	Using PMM Client	61
11	Configuring PMM Client with pmm-admin config	63
11.1	Connecting PMM Clients to the PMM Server	63
12	Adding MySQL Service Monitoring	65
13	Adding MongoDB Service Monitoring	67
14	Adding a ProxySQL host	69
14.1	Adding ProxySQL metrics service	69
15	Adding Linux metrics	71
15.1	Adding general system metrics service	71
16	PostgreSQL	73
16.1	Adding PostgreSQL extension for queries monitoring	73
16.2	Adding PostgreSQL queries and metrics monitoring	73
16.3	Setting up the required user permissions and authentication	74
17	Removing monitoring services with pmm-admin remove	75
V	Service configuration for best results	77
18	MySQL	79
18.1	Slow Log Settings	79
18.2	Configuring Performance Schema	79

18.3	MySQL InnoDB Metrics	81
18.4	Percona Server specific settings	81
18.4.1	MySQL User Statistics (userstat)	81
18.4.2	Query Response Time Plugin	82
18.4.3	log_slow_rate_limit	82
18.4.4	log_slow_verbosity	82
18.4.5	slow_query_log_use_global_control	83
18.5	Configuring MySQL 8.0 for PMM	83
19	MongoDB	85
19.1	Passing SSL parameters to the mongodb monitoring service	85
VI	Configuring AWS RDS or Aurora	87
20	Required AWS settings	89
20.1	Creating an IAM user with permission to access Amazon RDS DB instances	90
20.2	Creating a policy	90
20.3	Creating an IAM user	91
20.4	Creating an access key for an IAM user	93
20.5	Attaching a policy to an IAM user	93
20.6	Setting up the Amazon RDS DB Instance	93
21	Adding an Amazon RDS MySQL, Aurora MySQL, or Remote Instance	95
VII	Using PMM Metrics Monitor	99
22	Understanding Dashboards	101
22.1	Opening a Dashboard	101
22.2	Viewing More Information about a Graph	101
23	Navigating across Dashboards	103
23.1	Zooming in on a single metric	104
VIII	Using PMM Query Analytics	107
24	Introduction	109
25	Navigating to Query Analytics	111
26	Understanding Top 10	113
26.1	Query Detail Section	113
27	Filtering Queries	115
27.1	Totals of the Query Summary	117
27.2	Queries in the Query Summary Table	117
27.3	Viewing a Specific Value of a Metric	117
28	MongoDB specific	119
28.1	Query Analytics for MongoDB	119

IX Metrics Monitor Dashboards

121

29 Insight	125
29.1 Home Dashboard	125
29.1.1 General Information	125
29.1.2 Shared and Recently Used Dashboards	125
29.1.3 Statistics	125
29.1.4 Environment Overview	125
29.2 <i>PMM System Summary</i> Dashboard	126
29.3 <i>Advanced Data Exploration</i> Dashboard	126
29.3.1 View actual metric values (Gauge)	127
29.3.2 View actual metric values (Counters)	127
29.3.3 View actual metric values (Counters)	127
29.4 Cross Server Graphs	128
29.4.1 Load Average	128
29.4.2 MySQL Queries	128
29.4.3 MySQL Traffic	128
29.5 Summary Dashboard	128
29.5.1 CPU Usage	129
29.5.2 Processes	129
29.5.3 Network Traffic	130
29.5.4 I/O Activity	130
29.5.5 Disk Latency	130
29.5.6 MySQL Queries	130
29.5.7 InnoDB Row Operations	130
29.5.8 Top MySQL Commands	130
29.5.9 Top MySQL Handlers	131
29.6 Trends Dashboard	131
29.6.1 CPU Usage	131
29.6.2 I/O Read Activity	131
29.6.3 I/O Write Activity	132
29.6.4 MySQL Questions	132
29.6.5 InnoDB Rows Read	132
29.6.6 InnoDB Rows Changed	132
29.7 <i>Network Overview</i> Dashboard	132
29.7.1 Last Hour Statistic	133
29.7.2 Network Traffic	133
29.7.3 Network Traffic Details	133
29.7.4 Network Netstat TCP	133
29.7.5 Network Netstat UDP	133
29.7.6 ICMP	134
29.8 Inventory Dashboard	135
30 OS Dashboards	137
30.1 CPU Utilization Details (Cores)	137
30.1.1 Overall CPU Utilization	137
30.1.2 Current CPU Core Utilization	138
30.1.3 All Cores - Total	138
30.2 Disk space	139
30.2.1 Mountpoint Usage	139
30.2.2 Mountpoint	139
30.3 <i>System Overview</i> Dashboard	139
30.4 <i>Compare System Parameters</i> Dashboard	140
30.5 <i>NUMA Overview</i> Dashboard	140

30.5.1	Memory Usage	141
30.5.2	Free Memory Percent	141
30.5.3	NUMA Memory Usage Types	141
30.5.4	NUMA Allocation Hits	141
30.5.5	NUMA Allocation Missed	141
30.5.6	Anonymous Memory	141
30.5.7	NUMA File (PageCache)	141
30.5.8	Shared Memory	142
30.5.9	HugePages Statistics	142
30.5.10	Local Processes	142
30.5.11	Remote Processes	142
30.5.12	Slab Memory	142
31	Prometheus Dashboards	143
31.1	<i>Prometheus</i> Dashboard	143
31.1.1	Prometheus overview	143
31.1.2	Resources	143
31.1.3	Storage (TSDB)	143
31.1.4	Scraping	143
31.1.5	Queries	144
31.1.6	Network	144
31.1.7	Time Series Information	144
31.1.8	System Level Metrics	144
31.1.9	PMM Server Logs	144
31.2	Prometheus Exporter Status	144
31.3	Prometheus Exporters Overview	145
31.3.1	Prometheus Exporters Summary	145
31.3.2	Prometheus Exporters Resource Usage by Host	146
31.3.3	Prometheus Exporters Resource Usage by Type	146
31.3.4	List of Hosts	146
32	MySQL Dashboards	147
32.1	MySQL Amazon Aurora Metrics	147
32.1.1	Amazon Aurora Transaction Commits	147
32.1.2	Amazon Aurora Load	147
32.1.3	Aurora Memory Used	147
32.1.4	Amazon Aurora Statement Latency	148
32.1.5	Amazon Aurora Special Command Counters	148
32.1.6	Amazon Aurora Problems	148
32.2	MySQL <i>Command/Handler Counters Compare</i> Dashboard	148
32.3	MySQL <i>InnoDB Compression</i> Dashboard	149
32.3.1	Compression level and failure rate threshold	149
32.3.2	Statistics of Compression Operations	150
32.3.3	CPU Core Usage	150
32.3.4	Buffer Pool Total	150
32.3.5	Buffer Pool All	150
32.4	MySQL InnoDB Metrics	150
32.4.1	InnoDB Checkpoint Age	151
32.4.2	InnoDB Transactions	151
32.4.3	InnoDB Row Operations	151
32.4.4	InnoDB Row Lock Time	151
32.4.5	InnoDB I/O	152
32.4.6	InnoDB Log File Usage Hourly	152
32.4.7	InnoDB Deadlocks	152

32.4.8	InnoDB Condition Pushdown	153
32.4.9	Other Metrics	153
32.5	MySQL InnoDB Metrics (Advanced) Dashboard	153
32.5.1	Change Buffer Performance	154
32.5.2	InnoDB Log Buffer Performance	154
32.5.3	InnoDB Page Splits	154
32.5.4	InnoDB Page Reorgs	155
32.5.5	InnoDB Purge Performance	155
32.5.6	InnoDB Locking	155
32.5.7	InnoDB Main Thread Utilization	155
32.5.8	InnoDB Transactions Information	155
32.5.9	InnoDB Undo Space Usage	155
32.5.10	InnoDB Activity	156
32.5.11	InnoDB Contention - OS Waits	156
32.5.12	InnoDB Contention - Spin Rounds	156
32.5.13	InnoDB Group Commit Batch Size	156
32.5.14	InnoDB Purge Throttling	156
32.5.15	InnoDB AHI Usage	156
32.5.16	InnoDB AHI Maintenance	157
32.5.17	InnoDB Online DDL	157
32.5.18	InnoDB Defragmentation	157
32.6	MySQL MyISAM Aria Metrics Dashboard	157
32.6.1	Aria Storage Engine	157
32.6.2	Aria Pagecache Reads/Writes	158
32.6.3	Aria Pagecache Blocks	158
32.6.4	Aria Transactions Log Syncs	158
32.7	MySQL MyRocks Metrics Dashboard	158
32.8	MySQL Overview	159
32.8.1	MySQL Uptime	160
32.8.2	Current QPS	160
32.8.3	InnoDB Buffer Pool Size	160
32.8.4	Buffer Pool Size % of Total RAM	160
32.8.5	MySQL Connections	161
32.8.6	MySQL Active Threads	161
32.8.7	MySQL Questions	161
32.8.8	MySQL Thread Cache	161
32.8.9	MySQL Select Types	162
32.8.10	MySQL Sorts	162
32.8.11	MySQL Slow Queries	162
32.8.12	Aborted Connections	162
32.8.13	Table Locks	162
32.8.14	MySQL Network Traffic	163
32.8.15	MySQL Network Usage Hourly	163
32.8.16	MySQL Internal Memory Overview	163
32.8.17	Top Command Counters and Top Command Counters Hourly	164
32.8.18	MySQL Handlers	164
32.8.19	MySQL Query Cache Memory and MySQL Query Cache Activity	164
32.8.20	MySQL Open Tables, MySQL Table Open Cache Status, and MySQL Table Definition Cache	164
32.9	MySQL Performance Schema Dashboard	165
32.10	Performance Schema Wait Event Analysis Dashboard	165
32.11	MySQL Query Response Time	166
32.11.1	Average Query Response Time	166
32.11.2	Query Response Time Distribution	166
32.11.3	Average Query Response Time (Read/Write Split)	166

32.11.4	Read Query Response Time Distribution	167
32.11.5	Write Query Response Time Distribution	167
32.12	MySQL Replication	167
32.12.1	IO Thread Running	168
32.12.2	SQL Thread Running	168
32.12.3	Replication Error No	169
32.12.4	Read only	169
32.12.5	MySQL Replication Delay	170
32.12.6	Binlog Size	170
32.12.7	Binlog Data Written Hourly	171
32.12.8	Binlog Count	171
32.12.9	Binlogs Created Hourly	171
32.12.10	Relay Log Space	171
32.12.11	Relay Log Written Hourly	172
32.13	MySQL Table Statistics	172
32.13.1	Largest Tables	172
32.13.2	Pie	172
32.13.3	Table Activity	172
32.13.4	Rows read	173
32.13.5	Rows Changed	173
32.13.6	Auto Increment Usage	173
32.14	MySQL User Statistics	173
33	MongoDB Dashboards	175
33.1	MongoDB Cluster Summary	175
33.2	MongoDB inMemory Dashboard	175
33.3	MongoDB MMAPv1 Dashboard	176
33.4	MongoDB Overview	176
33.4.1	Command Operations	176
33.4.2	Connections	176
33.4.3	Cursors	177
33.4.4	Document Operations	177
33.4.5	Queued Operations	177
33.4.6	getLastError Write Time, getLastError Write Operations	177
33.4.7	Asserts	177
33.4.8	Memory Faults	177
33.5	MongoDB ReplSet	177
33.5.1	Replication Operations	178
33.5.2	ReplSet State	178
33.5.3	ReplSet Members	178
33.5.4	ReplSet Last Election	178
33.5.5	ReplSet Lag	178
33.5.6	Storage Engine	179
33.5.7	Oplog Insert Time	179
33.5.8	Oplog Recovery Window	179
33.5.9	Replication Lag	179
33.5.10	Elections	179
33.5.11	Member State Uptime	179
33.5.12	Max Heartbeat Time	179
33.5.13	Max Member Ping Time	180
33.6	MongoDB RocksDB Dashboard	180
33.7	MongoDB WiredTiger Dashboard	180
34	PostgreSQL Dashboards	183

34.1	<i>PostgreSQL Overview</i>	183
34.1.1	Connected	183
34.1.2	Version	184
34.1.3	Shared Buffers	184
34.1.4	Disk-Page Buffers	184
34.1.5	Memory Size for each Sort	184
34.1.6	Disk Cache Size	184
34.1.7	Autovacuum	185
34.1.8	PostgreSQL Connections	185
34.1.9	PostgreSQL Tuples	185
34.1.10	PostgreSQL Transactions	185
34.1.11	Temp Files	185
34.1.12	Conflicts and Locks	186
34.1.13	Buffers and Blocks Operations	186
34.1.14	Canceled Queries	186
34.1.15	Cache Hit Ratio	186
34.1.16	Checkpoint Stats	186
34.1.17	PostgreSQL Settings	187
34.1.18	System Summary	187
35	HA Dashboards	189
35.1	PXC/Galera Cluster Overview Dashboard	189
X	Contacting and Contributing	191
XI	Terminology Reference	195
36	Data retention	197
37	Data Source Name	199
38	DSN	201
39	Grand Total Time	203
40	%GTT	205
41	External Monitoring Service	207
42	Metrics	209
43	Metrics Monitor (MM)	211
44	Monitoring service	213
45	PMM	215
46	pmm-admin	217
47	PMM Client	219
48	PMM Docker Image	221
49	PMM Home Page	223

50 PMM Server	225
51 PMM Server Version	227
52 PMM user permissions for AWS	229
53 PMM Version	231
54 QAN	233
55 Query Analytics (QAN)	235
56 Query Load	237
57 Query Metrics Summary Table	239
58 Query Metrics Table	241
59 Query Summary Table	243
60 Quick ranges	245
61 Selected Time or Date Range	247
62 Telemetry	249
63 Version	251
XII Percona Monitoring and Management Release Notes	253
64 <i>Percona Monitoring and Management 2.5.0</i>	255
64.1 New Features	255
64.2 Improvements	255
64.3 Bugs Fixed	255
65 <i>Percona Monitoring and Management 2.4.0</i>	257
65.1 New Features	257
65.2 Improvements	257
65.3 Bugs Fixed	258
66 <i>Percona Monitoring and Management 2.3.0</i>	259
66.1 Improvements and new features	259
66.2 Fixed bugs	259
67 <i>Percona Monitoring and Management 2.2.2</i>	261
67.1 Improvements and new features	261
67.2 Fixed bugs	261
68 <i>Percona Monitoring and Management 2.2.1</i>	263
68.1 Improvements and new features	263
68.2 Fixed bugs	263
69 <i>Percona Monitoring and Management 2.2.0</i>	265
69.1 Improvements and new features	265
69.2 Fixed bugs	266

70 Percona Monitoring and Management 2.1.0	269
70.1 Improvements and new features	269
70.2 Fixed bugs	269
71 Percona Monitoring and Management 2.0.1	271
71.1 Improvements	271
71.2 Fixed bugs	271
72 Percona Monitoring and Management 2.0.0	273
XIII Frequently Asked Questions	275
73 How can I contact the developers?	279
74 What are the minimum system requirements for PMM?	281
75 How to control data retention for PMM?	283
76 How often are nginx logs in PMM Server rotated?	285
77 What privileges are required to monitor a MySQL instance?	287
78 Can I monitor multiple service instances?	289
79 Can I rename instances?	291
80 Can I add an AWS RDS MySQL or Aurora MySQL instance from a non-default AWS partition?	293
81 How to troubleshoot communication issues between PMM Client and PMM Server?	295
82 What resolution is used for metrics?	297
83 How to set up Alerting in PMM?	299
84 How to use a custom Prometheus configuration file inside of a PMM Server?	301

Percona Monitoring and Management (PMM) is an open-source platform for managing and monitoring MySQL, PostgreSQL, MongoDB, and ProxySQL performance. It is developed by Percona in collaboration with experts in the field of managed database services, support and consulting.

PMM is a free and open-source solution that you can run in your own environment for maximum security and reliability. It provides thorough time-based analysis for MySQL, PostgreSQL and MongoDB servers to ensure that your data works as efficiently as possible.

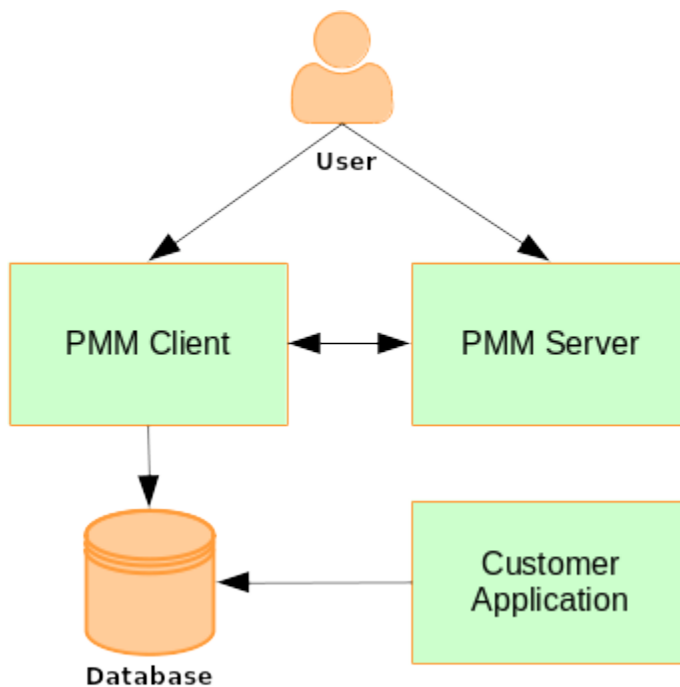
Part I

PMM Concepts

CLIENT/SERVER ARCHITECTURE - AN OVERVIEW

The PMM platform is based on a client-server model that enables scalability. It includes the following modules:

- *PMM Client* installed on every database host that you want to monitor. It collects server metrics, general system metrics, and Query Analytics data for a complete performance overview.
- *PMM Server* is the central part of PMM that aggregates collected data and presents it in the form of tables, dashboards, and graphs in a web interface.



The modules are packaged for easy installation and usage. It is assumed that the user should not need to understand what are the exact tools that make up each module and how they interact. However, if you want to leverage the full potential of PMM, the internal structure is important.

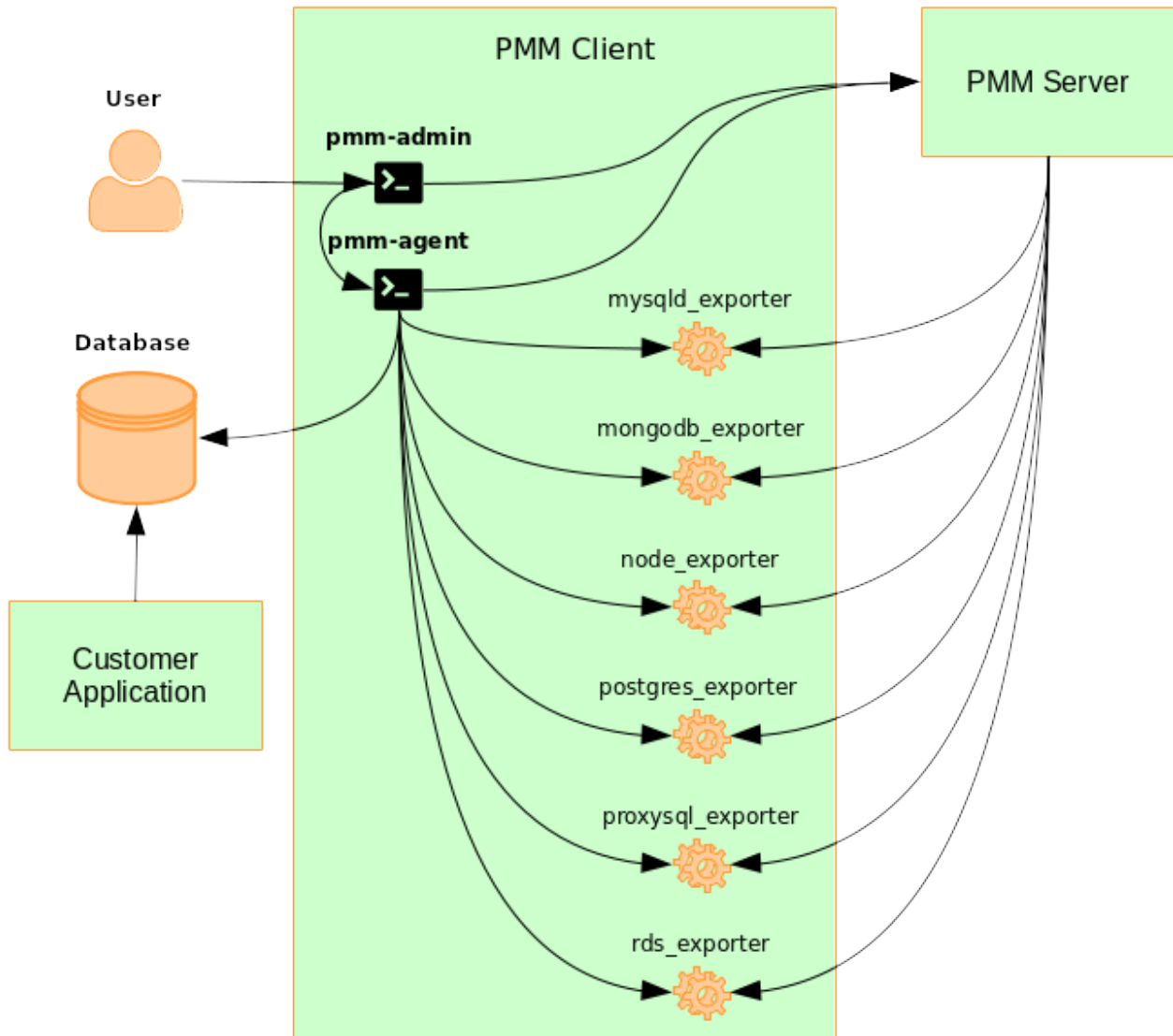
- *PMM Client*
 - *PMM Server*

PMM is a collection of tools designed to seamlessly work together. Some are developed by Percona and some are third-party open-source tools.

Note: The overall client-server model is not likely to change, but the set of tools that make up each component may evolve with the product.

The following sections illustrates how PMM is currently structured.

1.1 PMM Client



Each PMM Client collects various data about general system and database performance, and sends this data to the corresponding PMM Server.

The PMM Client package consist of the following:

- **pmm-admin** is a command-line tool for managing PMM Client, for example, adding and removing database instances that you want to monitor. For more information, see `pmm-admin`.
- **pmm-agent** is a client-side component a minimal command-line interface, which is a central entry point in charge for bringing the client functionality: it carries on client's authentication, gets the client configuration

stored on the PMM Server, manages exporters and other agents.

- **node_exporter** is a Prometheus exporter that collects general system metrics.
- **mysqld_exporter** is a Prometheus exporter that collects MySQL server metrics.
- **mongodb_exporter** is a Prometheus exporter that collects MongoDB server metrics.
- **postgres_exporter** is a Prometheus exporter that collects PostgreSQL performance metrics.
- **proxysql_exporter** is a Prometheus exporter that collects ProxySQL performance metrics.

To make data transfer from PMM Client to PMM Server secure, all exporters are able to use SSL/TLS encrypted connections, and their communication with the PMM server is protected by the HTTP basic authentication.

Note: Credentials used in communication between the exporters and the PMM Server are the following ones:

- login is “pmm”
 - password is equal to Agent ID, which can be seen e.g. on the Inventory Dashboard.
-

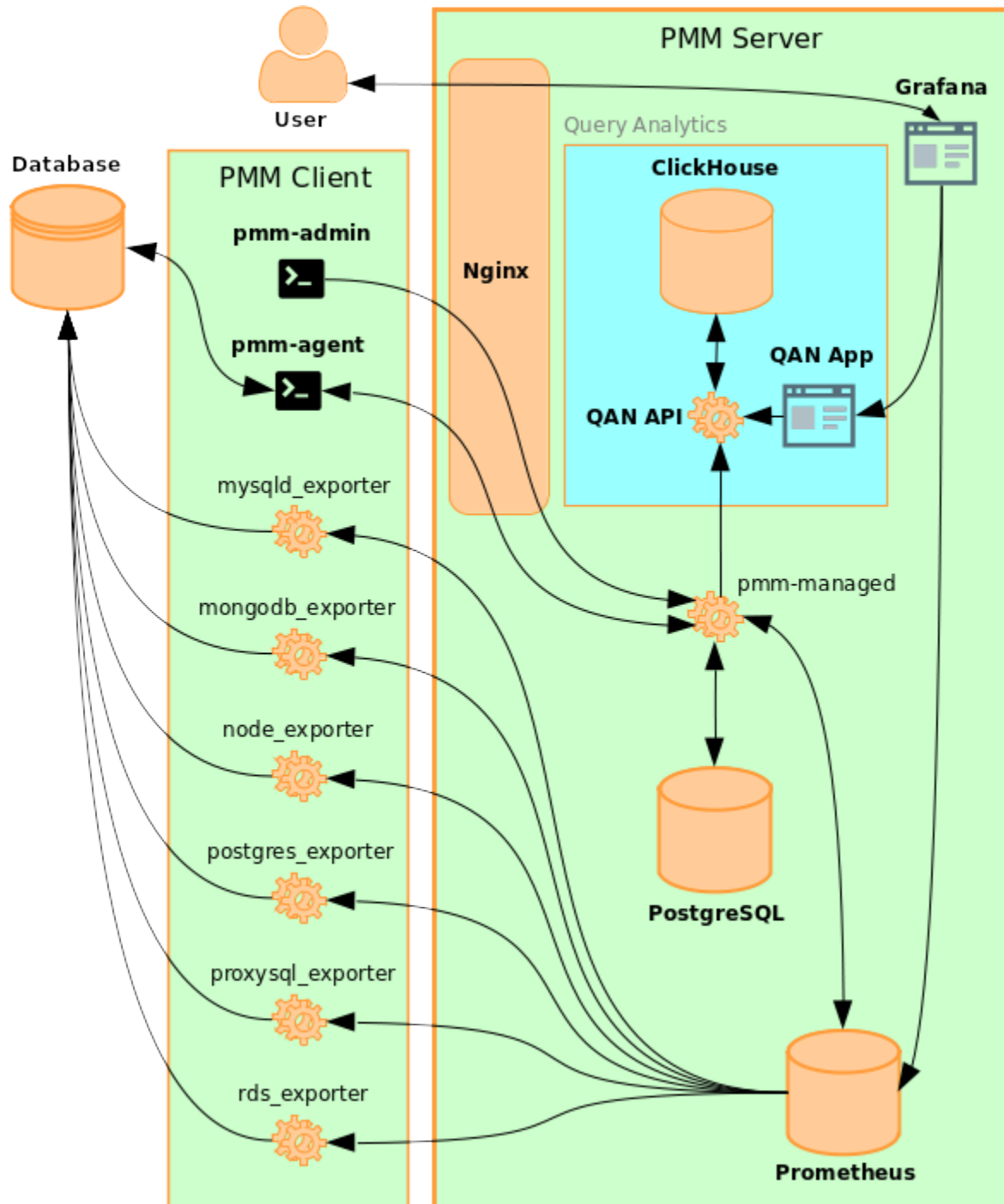
See also:

How to install PMM Client *Installing Clients*

How to pass exporter specific options when adding a monitoring service `pmm.pmm-admin.monitoring-service.pass-parameter`

List of available exporter options `pmm.list.exporter`

1.2 PMM Server



PMM Server runs on the machine that will be your central monitoring host. It is distributed as an appliance via the following:

- Docker image that you can use to run a container
- OVA (Open Virtual Appliance) that you can run in VirtualBox or another hypervisor
- AMI (Amazon Machine Image) that you can run via Amazon Web Services

For more information, see [Installing PMM Server](#).

PMM Server includes the following tools:

- Query Analytics enables you to analyze MySQL query performance over periods of time. In addition to the client-side QAN agent, it includes the following:
 - QAN API is the backend for storing and accessing query data collected by the QAN agent running on a *PMM Client*.
 - QAN Web App is a web application for visualizing collected Query Analytics data.
- Metrics Monitor provides a historical view of metrics that are critical to a MySQL or MongoDB server instance. It includes the following:
 - Prometheus is a third-party time-series database that connects to exporters running on a *PMM Client* and aggregates metrics collected by the exporters. For more information, see [Prometheus Docs](#).
 - ClickHouse is a third-party column-oriented database that facilitates the Query Analytics functionality. For more information, see [ClickHouse Docs](#).
 - Grafana is a third-party dashboard and graph builder for visualizing data aggregated by Prometheus in an intuitive web interface. For more information, see [Grafana Docs](#).
 - * Percona Dashboards is a set of dashboards for Grafana developed by Percona.

All tools can be accessed from the PMM Server web interface (landing page). For more information, see [using](#).

2.1 MySQL requirements

PMM supports all commonly used variants of MySQL, including Percona Server, MariaDB, and Amazon RDS. To prevent data loss and performance issues, PMM does not automatically change MySQL configuration. However, there are certain recommended settings that help maximize monitoring efficiency. These recommendations depend on the variant and version of MySQL you are using, and mostly apply to very high loads.

PMM can collect query data either from the *slow query log* or from *Performance Schema*. The *slow query log* provides maximum details, but can impact performance on heavily loaded systems. On Percona Server the query sampling feature may reduce the performance impact.

Performance Schema is generally better for recent versions of other MySQL variants. For older MySQL variants, which have neither sampling, nor *Performance Schema*, configure logging only slow queries.

Note: MySQL with too many tables can lead to PMM Server overload due to the streaming of too much time series data. It can also lead to too many queries from `mysqld_exporter` causing extra load on MySQL. Therefore PMM Server disables most consuming `mysqld_exporter` collectors automatically if there are more than 1000 tables.

You can add configuration examples provided below to `my.cnf` and restart the server or change variables dynamically using the following syntax:

```
SET GLOBAL <var_name>=<var_value>
```

The following sample configurations can be used depending on the variant and version of MySQL:

- If you are running Percona Server (or XtraDB Cluster), configure the *slow query log* to capture all queries and enable sampling. This will provide the most amount of information with the lowest overhead.

```
log_output=file
slow_query_log=ON
long_query_time=0
log_slow_rate_limit=100
log_slow_rate_type=query
log_slow_verbosity=full
log_slow_admin_statements=ON
log_slow_slave_statements=ON
slow_query_log_always_write_time=1
slow_query_log_use_global_control=all
innodb_monitor_enable=all
userstat=1
```

- If you are running MySQL 5.6+ or MariaDB 10.0+, configure *Configuring Performance Schema*.

```
innodb_monitor_enable=all
performance_schema=ON
```

- If you are running MySQL 5.5 or MariaDB 5.5, configure logging only slow queries to avoid high performance overhead.

Note: This may affect the quality of monitoring data gathered by QAN (Query Analytics).

```
log_output=file
slow_query_log=ON
long_query_time=0
log_slow_admin_statements=ON
log_slow_slave_statements=ON
```

2.1.1 Creating a MySQL User Account to Be Used with PMM

When adding a MySQL instance to monitoring, you can specify the MySQL server superuser account credentials. However, monitoring with the superuser account is not advised. It's better to create a user with only the necessary privileges for collecting data.

As an example, the user `pmm` can be created manually with the necessary privileges and pass its credentials when adding the instance.

To enable complete MySQL instance monitoring, a command similar to the following is recommended:

```
$ sudo pmm-admin add mysql --username pmm --password <password>
```

Of course this user should have necessary privileges for collecting data. If the `pmm` user already exists, you can grant the required privileges as follows:

```
CREATE USER 'pmm'@'localhost' IDENTIFIED BY 'pass' WITH MAX_USER_CONNECTIONS 10;
GRANT SELECT, PROCESS, SUPER, REPLICATION CLIENT, RELOAD ON *.* TO 'pmm'@'localhost';
GRANT SELECT ON performance_schema.* TO 'pmm'@'localhost';
```

See also:

Adding MySQL Service Monitoring - Using the `pmm-admin add` command to add a monitoring service

For more information, run: `pmm-admin add mysql --help`

2.2 MongoDB requirements

2.2.1 Configuring MongoDB for Monitoring in PMM Query Analytics

In QAN, you can monitor MongoDB metrics and MongoDB queries. Run the `pmm-admin add` command to use these monitoring services (for more information, see *Adding MongoDB Service Monitoring*).

Supported versions of MongoDB

QAN supports MongoDB version 3.2 or higher.

2.2.2 Setting Up the Required Permissions

For MongoDB monitoring services to be able work in QAN, you need to set up the `mongodb_exporter` user. This user should be assigned the `clusterMonitor` and `readAnyDatabase` roles for the `admin` database.

The following is an example you can run in the MongoDB shell, to add the `mongodb_exporter` user and assign the appropriate roles:

2.2.3 Enabling Profiling

For MongoDB to work correctly with QAN, you need to enable profiling in your `mongod` configuration. When started without profiling enabled, QAN displays the following warning:

Note: A warning message is displayed when profiling is not enabled

It is required that profiling of the monitored MongoDB databases be enabled, however profiling is not enabled by default because it may reduce the performance of your MongoDB server.

Enabling Profiling on Command Line

You can enable profiling from command line when you start the `mongod` server. This command is useful if you start `mongod` manually.

Run this command as root or by using the `sudo` command

Note that you need to specify a path to an existing directory that stores database files with the `--dpath`. When the `--profile` option is set to `2`, `mongod` collects the profiling data for all operations. To decrease the load, you may consider setting this option to `1` so that the profiling data are only collected for slow operations.

The `--slowms` option sets the minimum time for a slow operation. In the given example, any operation which takes longer than `200` milliseconds is a slow operation.

The `--rateLimit` option, which is available if you use PSMDB instead of MongoDB, refers to the number of queries that the MongoDB profiler collects. The lower the rate limit, the less impact on the performance. However, the accuracy of the collected information decreases as well.

See also:

`--rateLimit` in [PSMDB documentation](#)

Enabling Profiling in the Configuration File

If you run `mongod` as a service, you need to use the configuration file which by default is `/etc/mongod.conf`.

In this file, you need to locate the `operationProfiling`: section and add the following settings:

```
operationProfiling:
  slowOpThresholdMs: 200
  mode: slowOp
  rateLimit: 100
```

These settings affect `mongod` in the same way as the command line options described in section `pmm.qan-mongod.conf.profilng.command_line.enable`. Note that the configuration file is in the [YAML](#) format. In this format the indentation of your lines is important as it defines levels of nesting.

Restart the `mongod` service to enable the settings.

Run this command as root or by using the `sudo` command

Related Information

MongoDB Documentation: Enabling Profiling <https://docs.mongodb.com/manual/tutorial/manage-the-database-profiler/>

MongoDB Documentation: Profiling Mode <https://docs.mongodb.com/manual/reference/configuration-options/#operationProfiling.mode>

MongoDB Documentation: SlowOpThresholdMs option <https://docs.mongodb.com/manual/reference/configuration-options/#operationProfiling.slowOpThresholdMs>

MongoDB Documentation: Profiler Overhead (from MongoDB documentation) <https://docs.mongodb.com/manual/tutorial/manage-the-database-profiler/#profiler-overhead>

Documentation for Percona Server for MongoDB: Profiling Rate Limit <https://www.percona.com/doc/percona-server-for-mongodb/LATEST/rate-limit.html>

2.3 PostgreSQL requirements

2.3.1 Supported versions of PostgreSQL

PMM follows [postgresql.org](https://www.postgresql.org/about/news/eol-policy) EOL policy, and thus supports monitoring *PostgreSQL* version 9.4 and up. Older versions may work, but will not be supported. For additional assistance, visit [Percona PMM Forums](#).

Part II

Installing PMM Server

RUNNING PMM SERVER VIA DOCKER

Docker images of PMM Server are stored at the [percona/pmm-server](#) public repository. The host must be able to run Docker 1.12.6 or later, and have network access.

PMM needs roughly 1GB of storage for each monitored database node with data retention set to one week. Minimum memory is 2 GB for one monitored database node, but it is not linear when you add more nodes. For example, data from 20 nodes should be easily handled with 16 GB.

Make sure that the firewall and routing rules of the host do not constrain the Docker container. For more information, see *How to troubleshoot communication issues between PMM Client and PMM Server?*.

For more information about using Docker, see the [Docker Docs](#).

Important: By default, *retention* is set to 30 days for Metrics Monitor. Also consider *disabling table statistics*, which can greatly decrease Prometheus database size.

3.1 Setting Up a Docker Container for PMM Server

- *Pulling the PMM Server Docker Image*
- *Creating the pmm-data Container*
- *Creating and Launching the PMM Server Container*
- *Installing and using specific PMM Server version*

A Docker image is a collection of preinstalled software which enables running a selected version of PMM Server on your computer. A Docker image is not run directly. You use it to create a Docker container for your PMM Server. When launched, the Docker container gives access to the whole functionality of PMM.

The setup begins with pulling the required Docker image. Then, you proceed by creating a special container for persistent PMM data. The last step is creating and launching the PMM Server container.

3.1.1 Pulling the PMM Server Docker Image

To pull the latest version from Docker Hub:

```
$ docker pull percona/pmm-server:2
```

This step is not required if you are running PMM Server for the first time. However, it ensures that if there is an older version of the image tagged with 2.5.0 available locally, it will be replaced by the actual latest version.

3.1.2 Creating the pmm-data Container

To create a container for persistent PMM data, run the following command:

```
$ docker create \
  -v /srv \
  --name pmm-data \
  percona/pmm-server:2 /bin/true
```

Note: This container does not run, it simply exists to make sure you retain all PMM data when you upgrade to a newer PMM Server image. Do not remove or re-create this container, unless you intend to wipe out all PMM data and start over.

The previous command does the following:

- The **docker create** command instructs the Docker daemon to create a container from an image.
- The **-v** options initialize data volumes for the container.
- The **--name** option assigns a custom name for the container that you can use to reference the container within a Docker network. In this case: `pmm-data`.
- `percona/pmm-server:2` is the name and version tag of the image to derive the container from.
- `/bin/true` is the command that the container runs.

Important: PMM Server expects that the data volume initialized with the **-v** option will be `/srv`. Using any other value will result in data loss in an upgrade.

3.1.3 Creating and Launching the PMM Server Container

To create and launch PMM Server in one command, use **docker run**:

```
$ docker run -d \
  -p 80:80 \
  -p 443:443 \
  --volumes-from pmm-data \
  --name pmm-server \
  --restart always \
  percona/pmm-server:2
```

This command does the following:

- The **docker run** command runs a new container based on the `percona/pmm-server:2` image.
- The **-p** option maps the host port to the server port inside the docker container for accessing the PMM Server web UI in the format of `-p <hostPort>:<containerPort>`. For example, if port **80** is not available on your host, you can map the landing page to port 8080 using `-p 8080:80`, the same for port **443**: `-p 8443:443`.
- The **--volumes-from** option mounts volumes from the `pmm-data` container created previously (see *Creating the pmm-data Container*).

- The `--name` option assigns a custom name to the container that you can use to reference the container within the Docker network. In this case: `pmm-server`.
- The `--restart` option defines the container's restart policy. Setting it to `always` ensures that the Docker daemon will start the container on startup and restart it if the container exits.
- `percona/pmm-server:2` is the name and version tag of the image to derive the container from.

3.1.4 Installing and using specific PMM Server version

To install a specific PMM Server version instead of the latest one, just put desired version number after the colon. Also in this scenario it may be useful to [prevent updating PMM Server via the web interface](#) with the `DISABLE_UPDATES` docker option.

Following docker tags are currently available to represent PMM Server versions:

- `:latest` currently means the latest release of the PMM 1.X
- `:2` is the latest released version of PMM 2
- `:2.X` can be used to refer any minor released version, excluding patch releases
- `:2.X.Y` tag means specific patch release of PMM

For example, installing the latest 2.x version with disabled update button in the web interface would look as follows:

```
$ docker create \
  -v /srv \
  --name pmm-data \
  percona/pmm-server:2 /bin/true

$ docker run -d \
  -p 80:80 \
  -p 443:443 \
  --volumes-from pmm-data \
  --name pmm-server \
  -e DISABLE_UPDATES=true \
  --restart always \
  percona/pmm-server:2
```

See also:

Updating PMM *Updating PMM*

Backing Up the PMM Server Docker container *Backing Up PMM Data from the Docker Container*

Restoring pmm-data *Restoring the Backed Up Information to the PMM Data Container*

3.2 Updating PMM Server Using Docker

To check the version of PMM Server, run `docker ps` on the host.

Run the following commands as root or by using the `sudo` command

```
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                                  CREATED        STATUS        PORTS
4bdcc8463e64  percona/pmm-server:2.2.0           "/opt/entrypoint.sh"                   2 weeks ago   Up About an hour   0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
NAME          pmm-server
```

The version number is visible in the *Image* column. For a Docker container created from the image tagged 2, the *Image* column contains 2 and not the specific version number of PMM Server.

The information about the currently installed version of PMM Server is available from the `/srv/update/main.yml` file. You may extract the version number by using the `docker exec` command:

```
$ docker exec -it pmm-server head -1 /srv/update/main.yml
# v1.5.3
```

To check if there exists a newer version of PMM Server, visit [percona/pmm-server](https://percona.com/pmm-server).

3.2.1 Creating a backup version of the current pmm-server Docker container

You need to create a backup version of the current `pmm-server` container if the update procedure does not complete successfully or if you decide not to upgrade your PMM Server after trying the new version.

The `docker stop` command stops the currently running `pmm-server` container:

```
$ docker stop pmm-server
```

The following command simply renames the current `pmm-server` container to avoid name conflicts during the update procedure:

```
$ docker rename pmm-server pmm-server-backup
```

3.2.2 Pulling a new Docker Image

Docker images for all versions of PMM are available from [percona/pmm-server](https://percona.com/pmm-server) Docker repository.

When pulling a newer Docker image, you may either use a specific version number or the 2 image which always matches the highest version number.

This example shows how to pull a specific version:

This example shows how to pull the latest PMM 2 version:

```
$ docker pull percona/pmm-server:2
```

3.2.3 Creating a new Docker container based on the new image

After you have pulled a new version of PMM from the Docker repository, you can use `docker run` to create a `pmm-server` container using the new image.

```
$ docker run -d \
  -p 80:80 \
  -p 443:443 \
  --volumes-from pmm-data \
  --name pmm-server \
  --restart always \
  percona/pmm-server:2
```

Important: The `pmm-server` container must be stopped before attempting `docker run`.

The `docker run` command refers to the pulled image as the last parameter. If you used a specific version number when running `docker pull` (see *Pulling the PMM Server Docker Image*) replace 2 accordingly.

Note that this command also refers to `pmm-data` as the value of `--volumes-from` option. This way, your new version will continue to use the existing data.

Warning: Do not remove the `pmm-data` container when updating, if you want to keep all collected data.

Check if the new container is running using `docker ps`.

```
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                                CREATED        STATUS        PORTS                    NAMES
480696cd4187   percona/pmm-server:1.5.0           "/opt/entrypoint.sh"                 4 minutes ago Up 4 minutes   192.168.100.1:80->80/tcp, 443/tcp    pmm-server
```

Then, make sure that the PMM version has been updated (see *PMM Version*) by checking the PMM Server web interface.

3.2.4 Removing the backup container

After you have tried the features of the new version, you may decide to continue using it. The backup container that you have stored (*Creating a backup version of the current pmm-server Docker container*) is no longer needed in this case.

To remove this backup container, you need the `docker rm` command:

```
$ docker rm pmm-server-backup
```

As the parameter to `docker rm`, supply the tag name of your backup container.

Restoring the previous version

If, for whatever reason, you decide to keep using the old version, you just need to stop and remove the new `pmm-server` container.

```
$ docker stop pmm-server && docker rm pmm-server
```

Now, rename the `pmm-server-backup` to `pmm-server` (see *Creating a backup version of the current pmm-server Docker container*) and start it.

```
$ docker start pmm-server
```

Warning: Do not use the `docker run` command to start the container. The `docker run` command creates and then runs a new container.

To start a new container use the `docker start` command.

See also:

Setting up a Docker container *Setting Up a Docker Container for PMM Server*

Backing Up the PMM Server Docker container *Backing Up PMM Data from the Docker Container*

Updating the PMM Server and the PMM Client [deploy-pmm.updating](#) section.

3.3 Backing Up PMM Data from the Docker Container

When PMM Server is run via Docker, its data are stored in the `pmm-data` container. To avoid data loss, you can extract the data and store outside of the container.

This example demonstrates how to back up PMM data on the computer where the Docker container is run and then how to restore them.

To back up the information from `pmm-data`, you need to create a local directory with essential sub folders and then run Docker commands to copy PMM related files into it.

1. Create a backup directory and make it the current working directory. In this example, we use *pmm-data-backup* as the directory name.

```
$ mkdir pmm-data-backup; cd pmm-data-backup
```

2. Create the essential sub directory:

```
$ mkdir srv
```

Run the following commands as root or by using the **sudo** command

1. Stop the docker container:

```
$ docker stop pmm-server
```

2. Copy data from the `pmm-data` container:

```
$ docker cp pmm-data:/srv ./
```

Now, your PMM data are backed up and you can start PMM Server again:

```
$ docker start pmm-server
```

See also:

Restoring `pmm-data` [Restoring the Backed Up Information to the PMM Data Container](#)

Updating PMM Server run via Docker [Updating PMM Server Using Docker](#)

3.4 Restoring the Backed Up Information to the PMM Data Container

If you have a backup copy of your `pmm-data` container, you can restore it into a Docker container. Start with renaming the existing PMM containers to prevent data loss, create a new `pmm-data` container, and finally copy the backed up information into the `pmm-data` container.

Run the following commands as root or by using the **sudo** command

1. Stop the running `pmm-server` container.

```
$ docker stop pmm-server
```

2. Rename the `pmm-server` container to `pmm-server-backup`.

```
$ docker rename pmm-server pmm-server-backup
```

3. Rename the pmm-data to pmm-data-backup

```
$ docker rename pmm-data pmm-data-backup
```

4. Create a new pmm-data container

```
$ docker create \
  -v /srv \
  --name pmm-data \
  percona/pmm-server:2 /bin/true
```

Important: The last step creates a new pmm-data container based on the percona/pmm-server:2 image. If you do not intend to use the 2 tag, specify the exact version instead, such as **2.2.1**. You can find all available versions of pmm-server images at [percona/pmm-server](https://percona.com/pmm-server).

Assuming that you have a backup copy of your pmm-data, created according to the procedure described in the:ref:pmm.server.docker-backing-up section, restore your data as follows:

1. Change the working directory to the directory that contains your pmm-data backup files.

```
$ cd ~/pmm-data-backup
```

Note: This example assumes that the backup directory is found in your home directory.

2. Copy data from your backup directory to the pmm-data container.

```
$ docker cp opt/prometheus/data pmm-data:/opt/prometheus/
$ docker cp opt/consul-data pmm-data:/opt/
$ docker cp var/lib/mysql pmm-data:/var/lib/
$ docker cp var/lib/grafana pmm-data:/var/lib/
```

3. Apply correct ownership to pmm-data files:

```
$ docker run --rm --volumes-from pmm-data -it percona/pmm-server:2 chown -R_
↳pmm:pmm /opt/prometheus/data /opt/consul-data
$ docker run --rm --volumes-from pmm-data -it percona/pmm-server:2 chown -R_
↳grafana:grafana /var/lib/grafana
$ docker run --rm --volumes-from pmm-data -it percona/pmm-server:2 chown -R_
↳mysql:mysql /var/lib/mysql
```

4. Run (create and launch) a new pmm-server container:

```
$ docker run -d \
  -p 80:80 \
  -p 443:443 \
  --volumes-from pmm-data \
  --name pmm-server \
  --restart always \
  percona/pmm-server:2
```

To make sure that the new server is available run the **pmm-admin check-network** command from the computer where PMM Client is installed. Run this command as root or by using the **sudo** command.

```
$ pmm-admin check-network
```

See also:

Setting up PMM Server via Docker *Setting Up a Docker Container for PMM Server*

Updating PMM *Updating PMM*

Backing Up the PMM Server Docker container *Backing Up PMM Data from the Docker Container*

RUNNING PMM SERVER USING AWS MARKETPLACE

You can run an instance of PMM Server hosted at AWS Marketplace. This method replaces the outdated method where you would have to accessing an AMI (Amazon Machine Image) by using its ID, different for each region.

Percona Monitoring and Management Server
By: [Percona](#)
pmm-server
Linux/Unix ☆☆☆☆☆ [0 AWS reviews](#)
Free Tier

[Continue to Subscribe](#)
[Save to List](#)

Typical Total Price
\$0.10/hr
Total pricing per instance for services hosted on m4.large in US East (N. Virginia). [View Details](#)

[Overview](#) [Pricing](#) [Usage](#) [Support](#) [Reviews](#)

Product Overview

Percona Monitoring and Management (PMM) is a single pane of glass to help manage complex database environments in public, private or on-premises environments. Designed to help DBAs and developers gain deep insight into their applications and databases, PMM is used by thousands of organizations around the globe to manage complex database environments. PMM is an award-winning database monitoring tool built by Percona, the database performance and scalability experts, using best-of-breed tools.

Version	
By	Percona
Video	See Product Video
Categories	Monitoring
Operating System	Linux/Unix, CentOS 7
Delivery Methods	Amazon Machine Image

Highlights

- Keep your revenue engine up and running. With PMM, you can keep your databases running smoothly and continuously, with consistent end-user experience for applications. Easily find, fix, and prevent scaling issues, bottlenecks, and potential outages.
- Spend less time managing complex environments. Enable developers and DBAs to be able to view and monitor complex environments with multi-databases, multiple technologies, and multiple providers.
- Speed up development. PMM creates a common language between DBAs, developers, and sysadmins to help speed development and release cycles. With PMM, high-quality releases will not negatively impact performance, scale, or security.

Fig. 4.1: The home page of PMM in AWS Marketplace.

Assuming that you have an AWS (Amazon Web Services) account, locate *Percona Monitoring and Management Server* in AWS Marketplace.

The *Pricing Information* section allows to select your region and choose an instance type in the table that shows the

pricing for the software and infrastructure hosted in the region you have selected (the recommended EC2 instance type is preselected for you). Note that actual choice will be done later, and this table serves the information purposes, to plan costs.

Pricing Information

Use this tool to estimate the software and infrastructure costs based on your configuration choices. Your usage and costs might be different from this estimate. They will be reflected on your monthly AWS billing reports.

Estimating your costs

Choose your region and fulfillment option to see the pricing details. Then, modify the estimated price by choosing different instance types.

Region

US East (N. Virginia) ▾

Fulfillment Option

64-bit (x86) Amazon Machine Image (AMI) ▾

Software Pricing Details

Percona Monitoring and Management Server **\$0 /hr** >

running on t2.medium

Infrastructure Pricing Details

Estimated Infrastructure Cost **\$0.046 EC2/hr** >

Free Tier EC2 charges for Micro instances are free for up to 750 hours a month if you qualify for the [AWS Free Tier](#).

The table shows current software and infrastructure pricing for services hosted in **US East (N. Virginia)**. Additional taxes or fees may apply.

Percona Monitoring and Management Server			
EC2 Instance type	Software/hr	EC2/hr	Total/hr
<input type="radio"/> t2.micro	\$0	\$0.012	\$0.012
<input type="radio"/> t2.small	\$0	\$0.023	\$0.023
<input checked="" type="radio"/> t2.medium	\$0	\$0.046	\$0.046
<input type="radio"/> t2.large	\$0	\$0.093	\$0.093
<input type="radio"/> t2.xlarge	\$0	\$0.186	\$0.186
<input type="radio"/> t2.2xlarge	\$0	\$0.371	\$0.371

Fig. 4.2: As soon as you select your region, you can choose the EC2 instance in it and see its price. PMM comes for no cost, you may only need to pay for the infrastructure provided by Amazon.

Note: Disk space consumed by PMM Server depends on the number of hosts under monitoring. Each environment will be unique, however consider modeling your data consumption based on [PMM Demo](#) web site, which consumes ~230MB/host/day, or ~6.9GB/host at the default 30 day retention period. See [this blog post](#) for more details.

- Clicking the *Continue to Subscribe* button will proceed to the terms and conditions page.
- Clicking *Continue to Configuration* there will bring a new page to start setting up your instance. You will be able to re-check the PMM Server version and the region. When done, continue to the launch options by clicking *Continue to Launch*.

Select the previously chosen instance type from the *EC2 Instance Type* drop-down menu. Also chose the launch option. Available launch options in the *Chose Action* drop-down menu include *Launch from Website* and *Launch through EC2*. The first one is a quick way to make your instance ready. For more control, use the Manual Launch through EC2 option.

[< Product Detail](#) [Subscribe](#) [Configure](#) [Launch](#)

Launch this software

Review your configuration and choose how you wish to launch the software.

Configuration Details

Fulfillment Option	64-bit (x86) Amazon Machine Image (AMI) Percona Monitoring and Management Server <i>running on m4.large</i>
Software Version	
Region	US East (N. Virginia)

[Usage Instructions](#)

Choose Action

Choose this action to launch from this website

EC2 Instance Type

Memory: 4 GiB
CPU: 2 virtual cores
Storage: EBS storage only
Network Performance: Low to Moderate

Fig. 4.3: Percona Monitoring and Management on AWS Marketplace - launch options.

4.1 Setting Up a PMM Instance Using the website GUI

Choose *Launch from Website* option, your region, and the EC2 instance type on the launch options page. On the previous screenshot, we use the US East (N. Virginia) region and the *EC2 Instance Type* named `t2.medium`. To reduce cost, you need to choose the region closest to your location.

4.1.1 Setting up a VPC and an EC2 Instance Type

In this demonstration, we use the VPC (virtual private cloud) named `vpc-484bb12f`. The exact name of VPC may be different from the example discussed here.

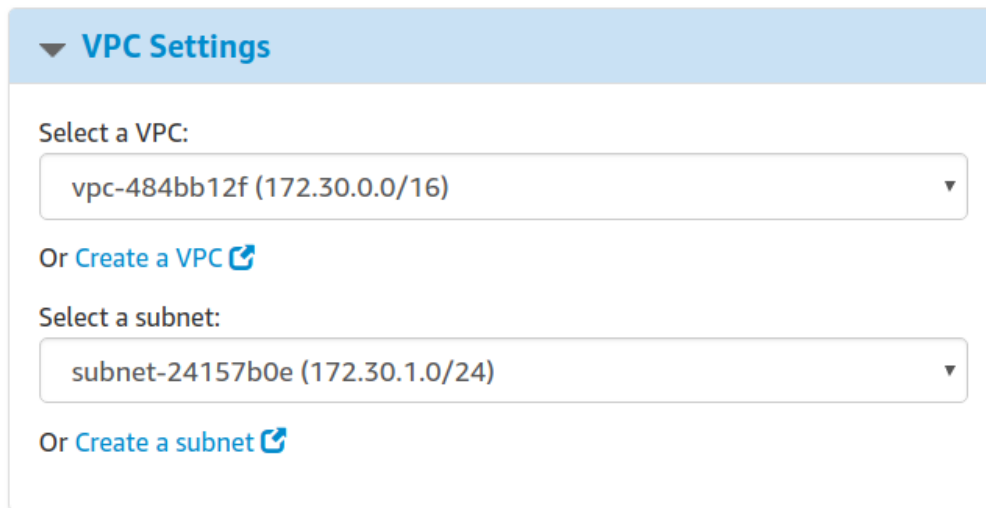


Fig. 4.4: Select VPC in the VPC Settings section.

Instead of a VPC (virtual private cloud) you may choose the `EC2 Classic (no VPC)` option and use a public cloud.

Selecting a subnet, you effectively choose an availability zone in the selected region. We recommend that you choose the availability zone where your RDS is located.

Note that the cost estimation is automatically updated based on your choice.

See also:

AWS Documentation: Availability zones <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html>

4.1.2 Limiting Access to the instance: security group and a key pair

In the *Security Group* section, which acts like a firewall, you may use the preselected option `Create new based on seller settings` to create a security group with recommended settings. In the *Key Pair* select an already set up EC2 key pair to limit access to your instance.

Important: It is important that the security group allow communication via the the following ports: `22`, `80`, and `443`. PMM should also be able to access port `3306` on the RDS that uses the instance.

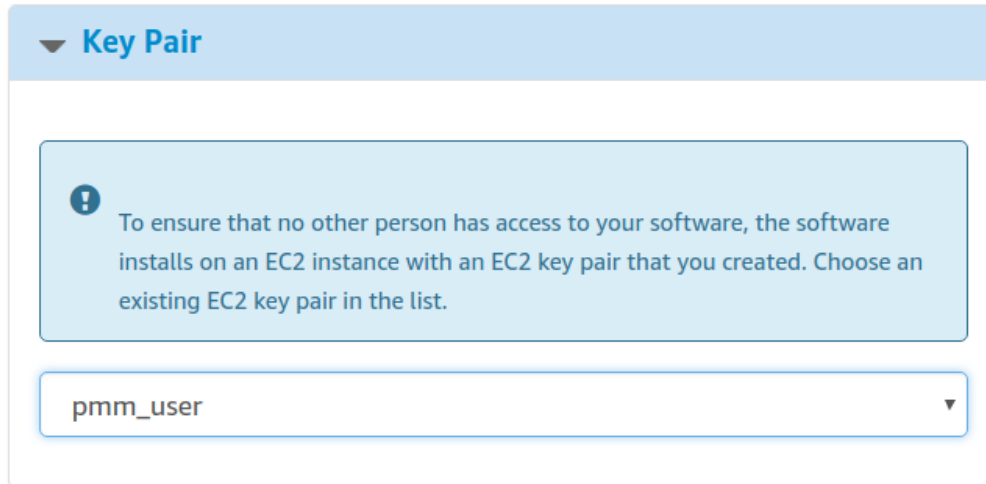


Fig. 4.5: Select an already existing key pair (use the EC2 console to create one if necessary)

See also:

Amazon Documentation: Security groups <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html>

Amazon Documentation: Key pairs <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html>

Amazon Documentation: Importing your own public key to Amazon EC2 <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#how-to-generate-your-own-key-and-import-it-to-aws>

4.1.3 Applying settings

Scroll up to the top of the page to view your settings. Then, click the *Launch with 1 click* button to continue and adjust your settings in the **EC2 console**.

Note: The *Launch with 1 click* button may alternatively be titled as *Accept Software Terms & Launch with 1-Click*.

4.1.4 Adjusting instance settings in the EC2 Console

Your clicking the *Launch with 1 click* button, deploys your instance. To continue setting up your instance, run the **EC2 console**. It is available as a link at the top of the page that opens after you click the *Launch with 1 click* button.

Your instance appears in the **EC2 console** in a table that lists all instances available to you. When a new instance is only created, it has no name. Make sure that you give it a name to distinguish from other instances managed via the **EC2 console**.

4.1.5 Running the instance

After you add your new instance it will take some time to initialize it. When the AWS console reports that the instance is now in a running state, you may continue with configuration of PMM Server.

▼ Security Group

A security group acts as a firewall that controls the traffic allowed to reach one or more instances. Learn more about [Security Groups](#).

You can create a new security group based on seller-recommended settings or choose one of your existing groups.

Create new based on seller settings ▼

! A new security group will be generated by AWS Marketplace. It is based on recommended settings for Percona Monitoring and Management Server provided by Percona.

Connection Method	Protocol	Port Range	Source (IP or Group)
SSH	tcp	22 - 22	Anywhere ▼ 0.0.0.0/0
HTTP	tcp	80 - 80	Anywhere ▼ 0.0.0.0/0
HTTPS	tcp	443 - 443	Anywhere ▼ 0.0.0.0/0

! Rules with source of 0.0.0.0/0 allows all IP addresses to access your instance. We recommend limiting access to only known IP addresses.

Fig. 4.6: Select a security group which manages firewall settings.

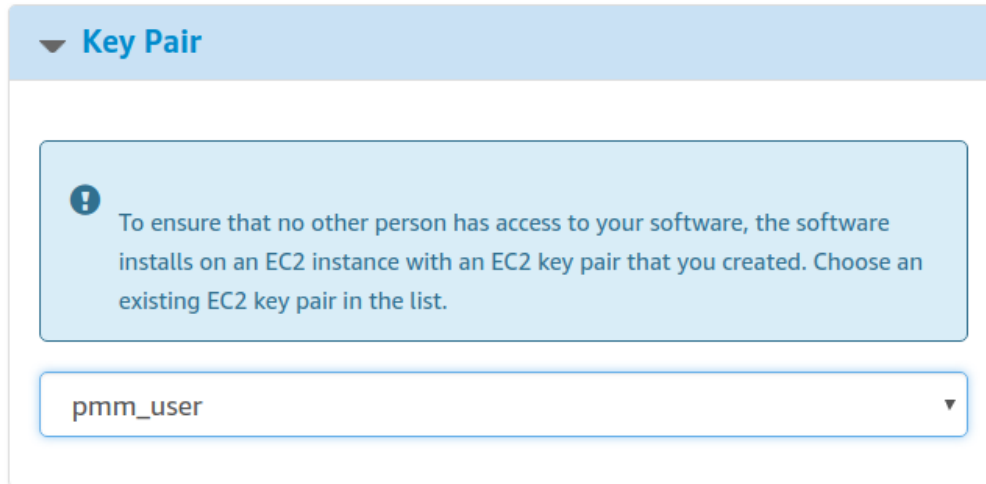


Fig. 4.7: Your instance settings are summarized in a special area. Click the Launch with 1 click button to continue.

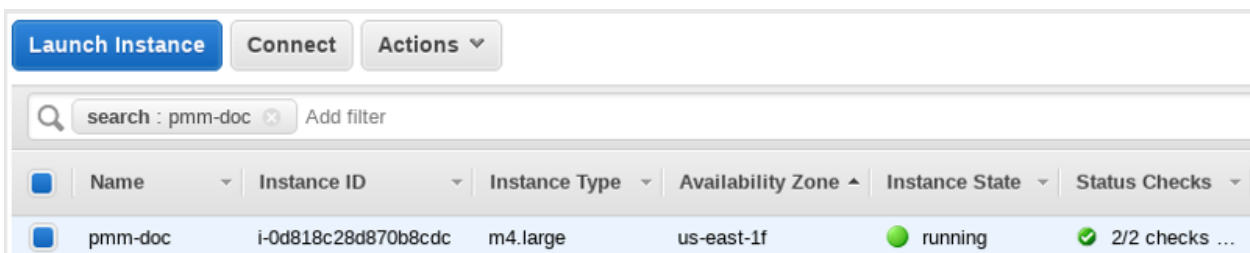


Fig. 4.8: The newly created instance selected.

Note: When started the next time after rebooting, your instance may acquire another IP address. You may choose to set up an elastic IP to avoid this problem.

See also:

Amazon Documentation: Elastic IP Addresses <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html>

With your instance selected, open its IP address in a web browser. The IP address appears in the *IPv4 Public IP* column or as value of the *Public IP* field at the top of the *Properties* panel.



Fig. 4.9: The public IP address of the instance

To run the instance, copy and paste its public IP address to the location bar of your browser. In the *Percona Monitoring and Management* welcome page that opens, enter the instance ID.

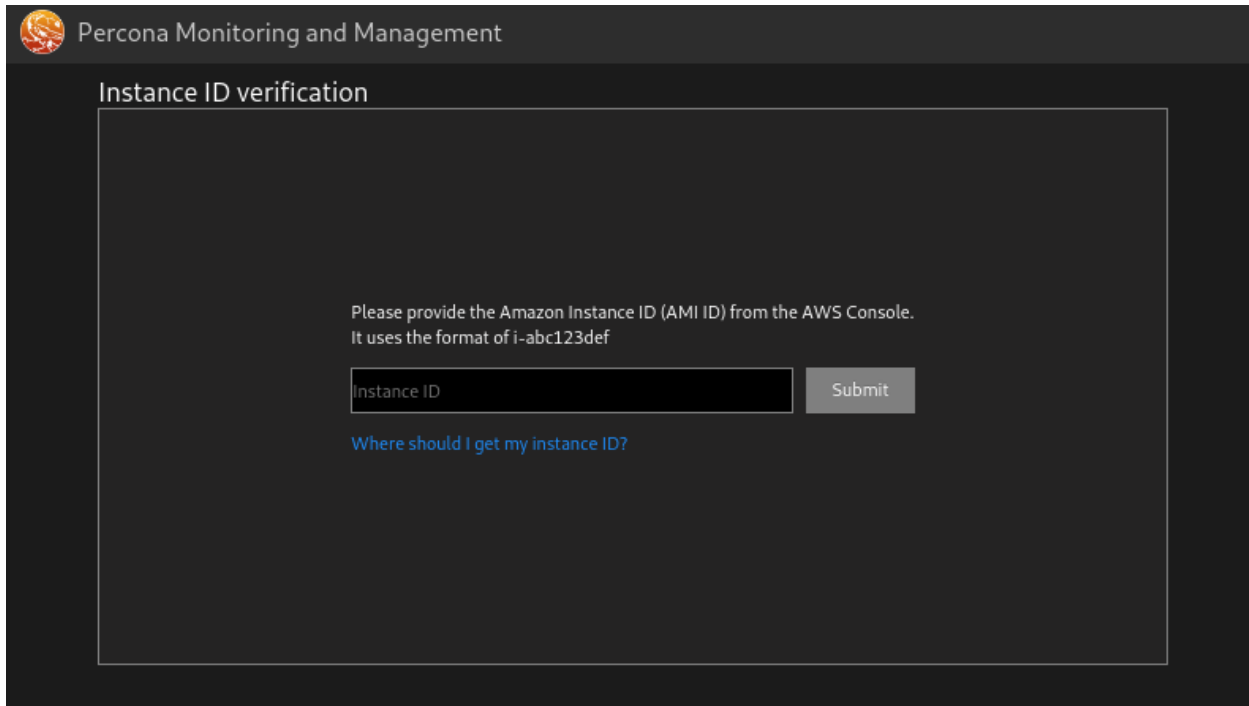


Fig. 4.10: Entering the instance ID when installing PMM Server

You can copy the instance ID from the *Properties* panel of your instance, select the *Description* tab back in the **EC2 console**. Click the *Copy* button next to the *Instance ID* field. This button appears as soon as you hover the cursor of your mouse over the ID.

Paste the instance in the *Instance ID* field of the *Percona Monitoring and Management* welcome page and click *Submit*.



Fig. 4.11: Hover the cursor over the instance ID for the Copy button to appear.

PMM Server provides user access control, and therefore you will need user credentials to access it:

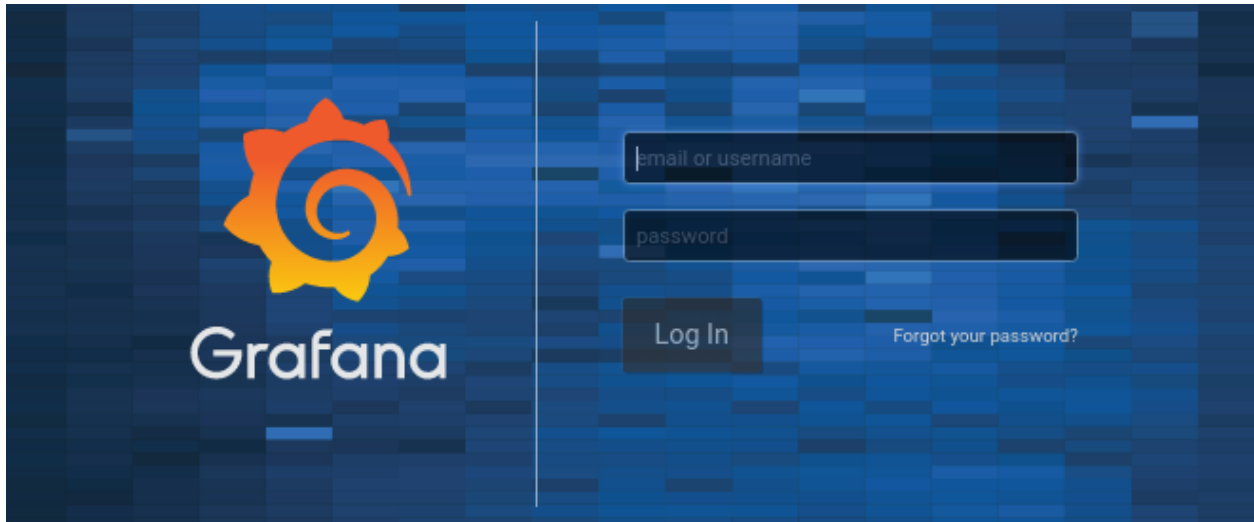


Fig. 4.12: Create credentials for your instance.

The default user name is `admin`, and the default password is `admin` also. You will be proposed to change the default password at login if you didn't it.

The PMM Server is now ready and the home page opens.

You are creating a username and password that will be used for two purposes:

1. authentication as a user to PMM - this will be the credentials you need in order to log in to PMM.
2. authentication between PMM Server and PMM Clients - you will re-use these credentials when configuring `pmm-client` for the first time on a server, for example:

Run this command as root or by using the `sudo` command

```
pmm-admin config --server-insecure-tls --server-url=https://admin:admin@<IP_
↪Address>:443
```

Note: Accessing the instance by using an SSH client.

For instructions about how to access your instances by using an SSH client, see [Connecting to Your Linux Instance Using SSH](#)

Make sure to replace the user name `ec2-user` used in this document with `admin`.

See also:

How to verify that the PMM Server is running properly? [Verifying PMM Server](#)

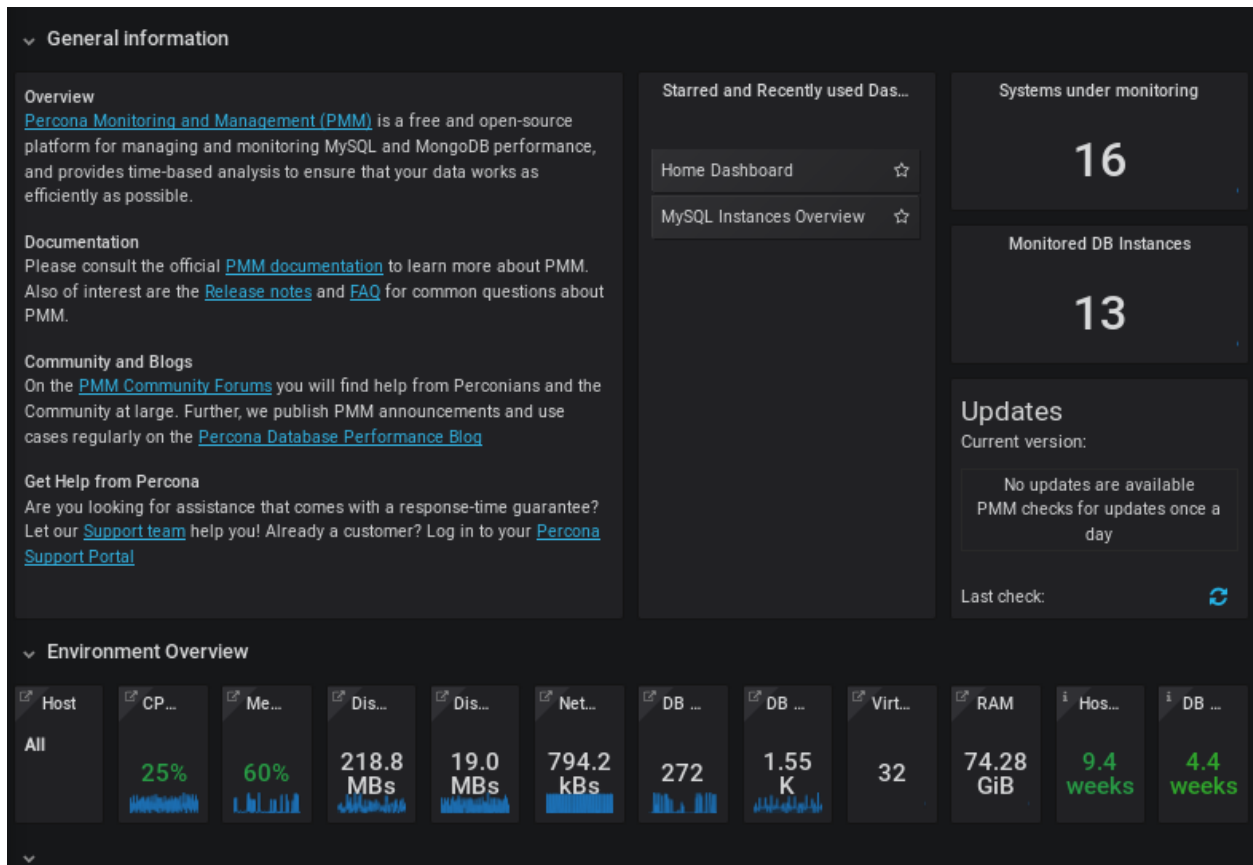


Fig. 4.13: PMM Server home page

How to connect a PMM Client to the PMM Server? `deploy-pmm.client_server.connecting`

Resizing the EBS Volume

Your AWS instance comes with a predefined size which can become a limitation. To make more disk space available to your instance, you need to increase the size of the EBS volume as needed and then your instance will reconfigure itself to use the new size.

The procedure of resizing EBS volumes is described in the Amazon documentation: [Modifying the Size, IOPS, or Type of an EBS Volume on Linux](#).

After the EBS volume is updated, PMM Server instance will autodetect changes in approximately 5 minutes or less and will reconfigure itself for the updated conditions.

Upgrading PMM Server on AWS

Upgrading EC2 instance class

Upgrading to a larger EC2 instance class is supported by PMM provided you follow the instructions from the [AWS manual](#). The PMM AMI image uses a distinct EBS volume for the PMM data volume which permits independent resize of the EC2 instance without impacting the EBS volume.

Expanding the PMM Data EBS Volume

The PMM data volume is mounted as an XFS formatted volume on top of an LVM volume. There are two ways to increase this volume size:

1. Add a new disk via EC2 console or API, and expand the LVM volume to include the new disk volume.
2. Expand existing EBS volume and grow the LVM volume.

Expand existing EBS volume

To expand the existing EBS volume in order to increase capacity, the following steps should be followed.

1. Expand the disk from AWS Console/CLI to the desired capacity.
2. Login to the PMM EC2 instance and verify that the disk capacity has increased. For example, if you have expanded disk from 16G to 32G, `dmesg` output should look like below:

```
[ 535.994494] xvdb: detected capacity change from 17179869184 to 34359738368
```

3. You can check information about volume groups and logical volumes with the `vgs` and `lvs` commands:

```
[root@ip-10-1-2-70 ~]# vgs
VG      #PV #LV #SN Attr   VSize  VFree
DataVG  1   2   0 wz--n- <16.00g  0

[root@ip-10-1-2-70 ~]# lvs
LV      VG      Attr      LSize   Pool Origin Data%  Meta% Move Log Cpy%Sync
↔Convert
DataLV  DataVG Vwi-aotz-- <12.80g ThinPool          1.74
ThinPool DataVG twi-aotz-- 15.96g 1.39 1.29
```

- Now we can use the `lsblk` command to see that our disk size has been identified by the kernel correctly, but LVM2 is not yet aware of the new size. We can use `pvresize` to make sure the PV device reflects the new size. Once `pvresize` is executed, we can see that the VG has the new free space available.

```
[root@ip-10-1-2-70 ~]# lsblk | grep xvdb
xvdb                202:16 0 32G 0 disk

[root@ip-10-1-2-70 ~]# pvscan
PV /dev/xvdb   VG DataVG   lvm2 [<16.00 GiB / 0   free]
Total: 1 [<16.00 GiB] / in use: 1 [<16.00 GiB] / in no VG: 0 [0   ]

[root@ip-10-1-2-70 ~]# pvresize /dev/xvdb
Physical volume "/dev/xvdb" changed
1 physical volume(s) resized / 0 physical volume(s) not resized

[root@ip-10-1-2-70 ~]# pvs
PV          VG      Fmt  Attr  PSize   PFree
/dev/xvdb  DataVG  lvm2 a--  <32.00g 16.00g
```

- We then extend our logical volume. Since the PMM image uses thin provisioning, we need to extend both the pool and the volume:

```
[root@ip-10-1-2-70 ~]# lvs
LV          VG      Attr          LSize   Pool   Origin Data%  Meta% Move Log Cpy%Sync_
↔Convert
DataLV     DataVG  Vwi-aotz--  <12.80g ThinPool          1.77
ThinPool   DataVG  twi-aotz--   15.96g          1.42  1.32

[root@ip-10-1-2-70 ~]# lvextend /dev/mapper/DataVG-ThinPool -l 100%VG
Size of logical volume DataVG/ThinPool_tdata changed from 16.00 GiB (4096
↔extents) to 31.96 GiB (8183 extents).
Logical volume DataVG/ThinPool_tdata successfully resized.

[root@ip-10-1-2-70 ~]# lvs
LV          VG      Attr          LSize   Pool   Origin Data%  Meta% Move Log Cpy%Sync_
↔Convert
DataLV     DataVG  Vwi-aotz--  <12.80g ThinPool          1.77
ThinPool   DataVG  twi-aotz--   31.96g          0.71  1.71
```

- Once the pool and volumes have been extended, we need to now extend the thin volume to consume the newly available space. In this example we've grown available space to almost 32GB, and already consumed 12GB, so we're extending an additional 19GB:

```
[root@ip-10-1-2-70 ~]# lvs
LV          VG      Attr          LSize   Pool   Origin Data%  Meta% Move Log Cpy%Sync_
↔Convert
DataLV     DataVG  Vwi-aotz--  <12.80g ThinPool          1.77
ThinPool   DataVG  twi-aotz--   31.96g          0.71  1.71

[root@ip-10-1-2-70 ~]# lvextend /dev/mapper/DataVG-DataLV -L +19G
Size of logical volume DataVG/DataLV changed from <12.80 GiB (3276 extents) to
↔<31.80 GiB (8140 extents).
Logical volume DataVG/DataLV successfully resized.

[root@ip-10-1-2-70 ~]# lvs
LV          VG      Attr          LSize   Pool   Origin Data%  Meta% Move Log Cpy%Sync_
↔Convert
DataLV     DataVG  Vwi-aotz--  <31.80g ThinPool          0.71
```



```
ThinPool DataVG twi-aotz-- 31.96g 0.71 1.71
```

7. We then expand the XFS filesystem to reflect the new size using `xfs_growfs`, and confirm the filesystem is accurate using the `df` command.

```
[root@ip-10-1-2-70 ~]# df -h /srv
Filesystem                Size Used Avail Use% Mounted on
/dev/mapper/DataVG-DataLV 13G 249M 13G  2% /srv

[root@ip-10-1-2-70 ~]# xfs_growfs /srv
meta-data=/dev/mapper/DataVG-DataLV isize=512    agcount=103, agsize=32752 blks
        =                               sectsz=512   attr=2, projid32bit=1
        =                               crc=1      finobt=0 spinodes=0
data      =                               bsize=4096 blocks=3354624, imaxpct=25
        =                               sunit=16   swidth=16 blks
naming    =version 2                       bsize=4096 ascii-ci=0 ftype=1
log       =internal                       bsize=4096 blocks=768, version=2
        =                               sectsz=512 sunit=16 blks, lazy-count=1
realtime  =none                             extsz=4096 blocks=0, rtextents=0
data blocks changed from 3354624 to 8335360

[root@ip-10-1-2-70 ~]# df -h /srv
Filesystem                Size Used Avail Use% Mounted on
/dev/mapper/DataVG-DataLV 32G 254M 32G  1% /srv
```


PMM SERVER AS A VIRTUAL APPLIANCE

Percona provides a *virtual appliance* for running PMM Server in a virtual machine. It is distributed as an *Open Virtual Appliance (OVA)* package, which is a **tar** archive with necessary files that follow the *Open Virtualization Format (OVF)*. OVF is supported by most popular virtualization platforms, including:

- VMware - ESXi 6.5
- Red Hat Virtualization
- VirtualBox
- XenServer
- Microsoft System Center Virtual Machine Manager

In this chapter

- *Supported Platforms for Running the PMM Server Virtual Appliance*
- *Identifying PMM Server IP Address*
- *Accessing PMM Server*
- *Accessing the Virtual Machine*
- *Next Steps*

5.1 Supported Platforms for Running the PMM Server Virtual Appliance

The virtual appliance is ideal for running PMM Server on an enterprise virtualization platform of your choice. This page explains how to run the appliance in VirtualBox and VMware Workstation Player, which is a good choice to experiment with PMM at a smaller scale on a local machine. Similar procedure should work for other platforms (including enterprise deployments on VMware ESXi, for example), but additional steps may be required.

The virtual machine used for the appliance runs CentOS 7.

Warning: The appliance must run in a network with DHCP, which will automatically assign an IP address for it. To assign a static IP manually, you need to acquire the root access as described in `pmm.deploying.server.virtual-appliance.root-password.setting`. Then, see the documentation for the operating system for further instructions: [Configuring network interfaces in CentOS](#)

Instructions for setting up the virtual machine for different platforms

5.2 Identifying PMM Server IP Address

When run PMM Server as virtual appliance, The IP address of your PMM Server appears at the top of the screen above the login prompt. Use this address to access the web interface of PMM Server.

```
[ 1.029514] sd 0:0:0:0: [sda] Incomplete mode parameter data
[ 1.029834] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 1.030718] sd 0:0:1:0: [sdb] Incomplete mode parameter data
[ 1.031033] sd 0:0:1:0: [sdb] Assuming drive cache: write through

*****
Percona Monitoring and Management          https://192.168.1.4/
*****

CentOS Linux 7 (Core)
Kernel 3.10.0-1062.9.1.el7.x86_64 on an x86_64

localhost login: ^I$
```

Fig. 5.1: The IP address appears above the login prompt.

PMM Server uses DHCP for security reasons, and thus you need to check the PMM Server console in order to identify the address. If you require configuration of a static IP address, see [Configuring network interfaces in CentOS](#)

5.3 Accessing PMM Server

To run the PMM Server, start the virtual machine and open in your browser the URL that appears at the top of the terminal when you are logging in to the virtual machine.

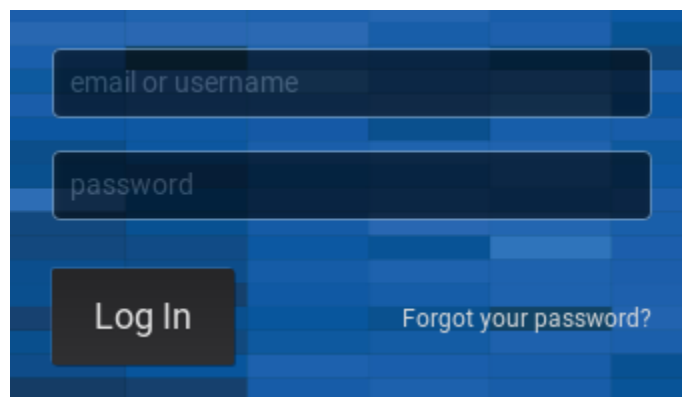


Fig. 5.2: Enter the user login and password to access the PMM Server web interface.

If you run PMM Server in your browser for the first time, you are requested to supply the user login and password. The default PMM Server credentials are:

- **username:** admin
- **password:** admin

After login you will be proposed to change this default password. Enter the new password twice and click *Save*. The PMM Server is now ready and the home page opens.

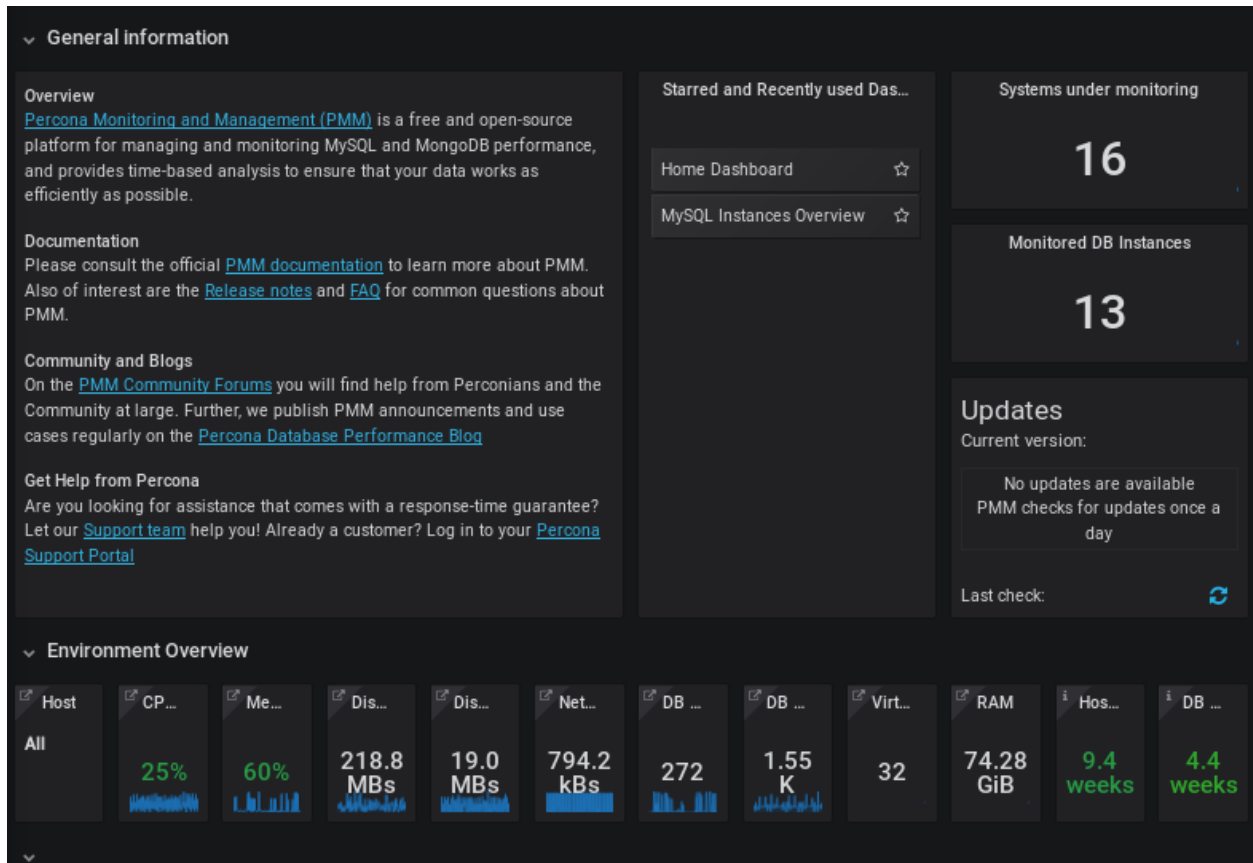


Fig. 5.3: PMM Server home page

You are creating a username and password that will be used for two purposes:

1. authentication as a user to PMM - this will be the credentials you need in order to log in to PMM.
2. authentication between PMM Server and PMM Clients - you will re-use these credentials as a part of the server URL when configuring PMM Client for the first time on a server:

Run this command as root or by using the **sudo** command

```
pmm-admin config --server-insecure-tls --server-url=https://admin:admin@<IP_
↵Address>:443
```

5.4 Accessing the Virtual Machine

To access the VM with the *PMM Server* appliance via SSH, you will need to provide your public key:

1. Open the URL for accessing PMM in a web browser.

The URL is provided either in the console window or in the appliance log.

2. Select the *PMM Settings* dashboard in the main menu.

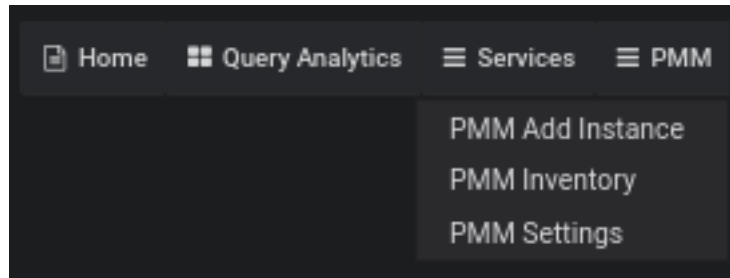


Fig. 5.4: Choosing the *PMM Settings* menu entry

3. Submit your **public key** in the *SSH Key Details* section and click the *Apply SSH Key* button.

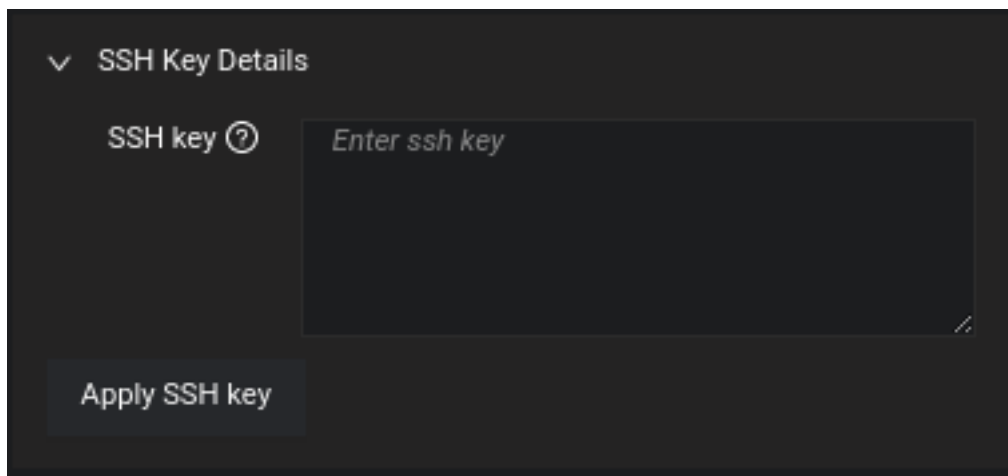


Fig. 5.5: Submitting the public key on the *Settings dashboard*

After that you can use `ssh` to log in as the `admin` user. For example, if *PMM Server* is running at `192.168.100.1` and your **private key** is `~/.ssh/pmm-admin.key`, use the following command:

```
ssh admin@192.168.100.1 -i ~/.ssh/pmm-admin.key
```

5.5 Next Steps

Verify that PMM Server is running by connecting to the PMM web interface using the IP address assigned to the virtual appliance, then *install PMM Client* on all database hosts that you want to monitor.

VERIFYING PMM SERVER

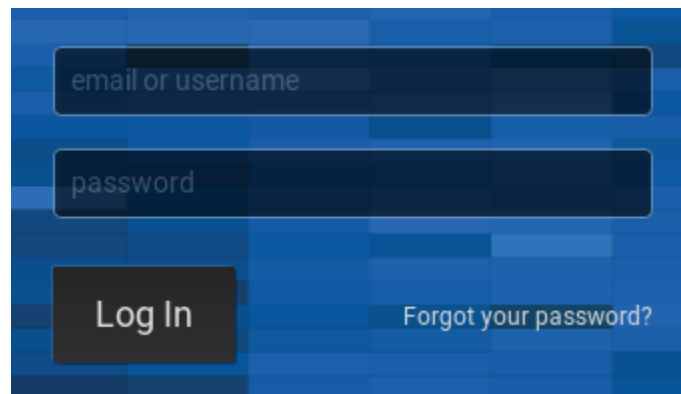
In your browser, go to the server by its IP address. If you run your server as a virtual appliance or by using an Amazon machine image, you will need to setup the user name, password and your public key if you intend to connect to the server by using ssh. This step is not needed if you run PMM Server using Docker.

In the given example, you would need to direct your browser to *http://192.168.100.1*. Since you have not added any monitoring services yet, the site will show only data related to the PMM Server internal services.

Table 6.1: Accessing the Components of the Web Interface

URL	Component
http://192.168.100.1	PMM Home Page
http://192.168.100.1/graph/	Metrics Monitor (MM)
http://192.168.100.1/swagger/	PMM API browser

PMM Server provides user access control, and therefore you will need user credentials to access it:



The default user name is `admin`, and the default password is `admin` also. You will be proposed to change the default password at login if you didn't it.

Note: You will use the same credentials at [connecting](#) your PMM Client to PMM Server.

CONFIGURING PMM SERVER

7.1 PMM Settings Page

PMM Settings is a special page dedicated to configuring a number of PMM options. It can be accessed through the main menu:

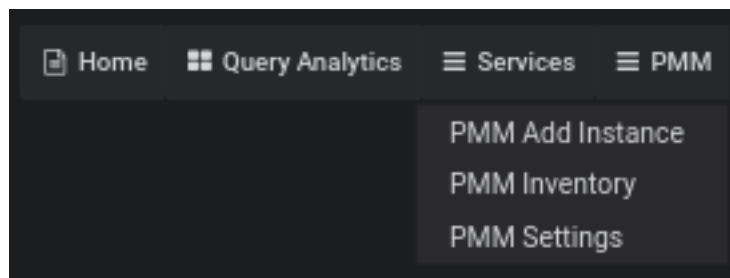


Fig. 7.1: Choosing the *PMM Settings* menu entry

PMM Settings page consists of the following sections, which allow to configure different aspects of the PMM Server:

- *Settings*
- *Telemetry*
- *SSH Key Details*
- *AlertManager integration*
- *Diagnostics*

7.1.1 Settings

Settings section allows you to change *metrics resolution*, *data retention*, as well as configure *telemetry* and automatic checking for updates:

Don't forget to click the *Apply changes* button to make changed options work.

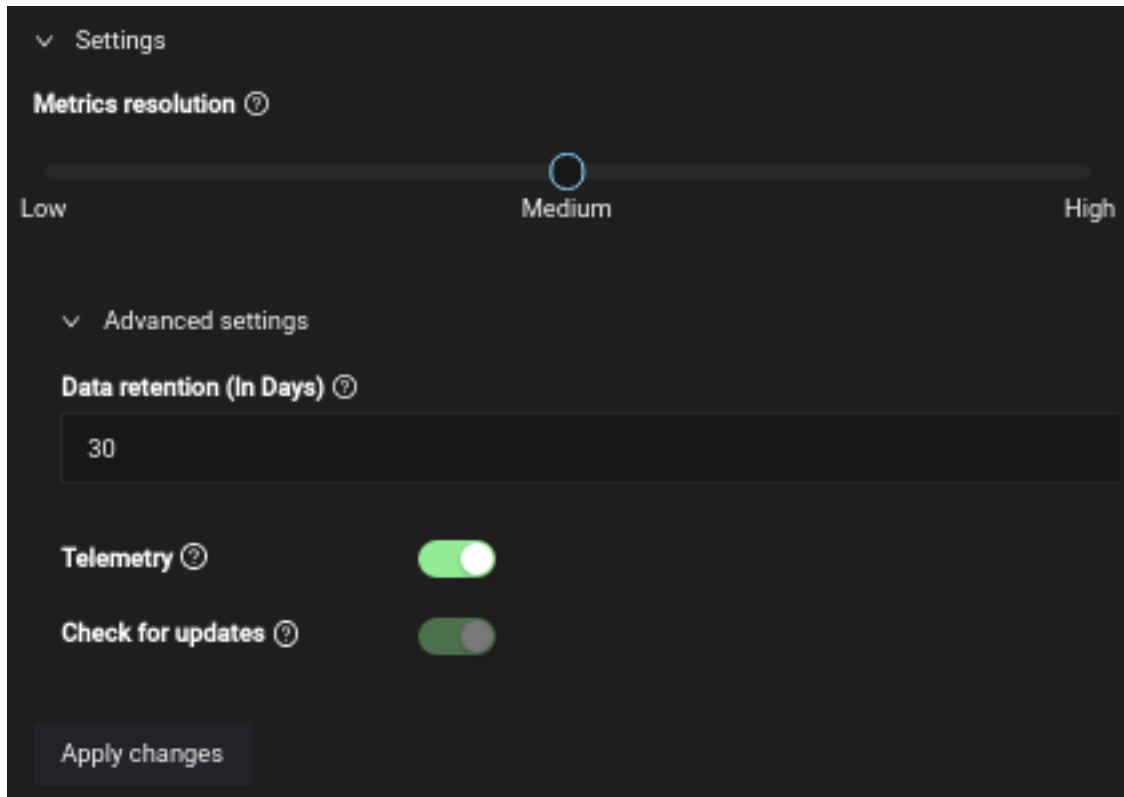


Fig. 7.2: Settings options

7.1.2 Telemetry

The **Telemetry** switch enables gathering and sending basic anonymous data to Percona, which helps us to determine where to focus the development

and what is the uptake of the various versions of PMM. Specifically, gathering this information helps determine if we need to release patches to legacy versions beyond support, determining when supporting a particular version is no longer necessary, and even understanding how the frequency of release encourages or deters adoption.

Currently, only the following information is gathered:

- PMM Version,
- Installation Method (Docker, AMI, OVF),
- the Uptime.

We do not gather anything that would make the system identifiable, but the following two things are to be mentioned:

1. The Country Code is evaluated from the submitting IP address before it is discarded.
2. We do create an “instance ID” - a random string generated using UUID v4. This instance ID is generated to distinguish new instances from existing ones, for figuring out instance upgrades.

Note: The first telemetry reporting of a new PMM Server instance is delayed by 24 hours to allow sufficient time to disable the service for those that do not wish to share any information.

There is a landing page for this service, available at check.percona.com, which clearly explains what this service is, what it's collecting, and how you can turn it off.

Note: The [Grafana internal reporting feature](#) is currently **not** managed by PMM. If you want to turn it, you need to go inside the PMM Server container and [change configuration](#) after each update.

Note: Beside using *PMM Settings* page, you can also disable Telemetry with the `-e DISABLE_TELEMETRY=1` option in your docker run statement for the PMM Server.

7.1.3 SSH Key Details

This section allows you to upload your public SSH key which can be used to access the PMM Server via SSH (e.g. the PMM Server deployed as a virtual appliance).

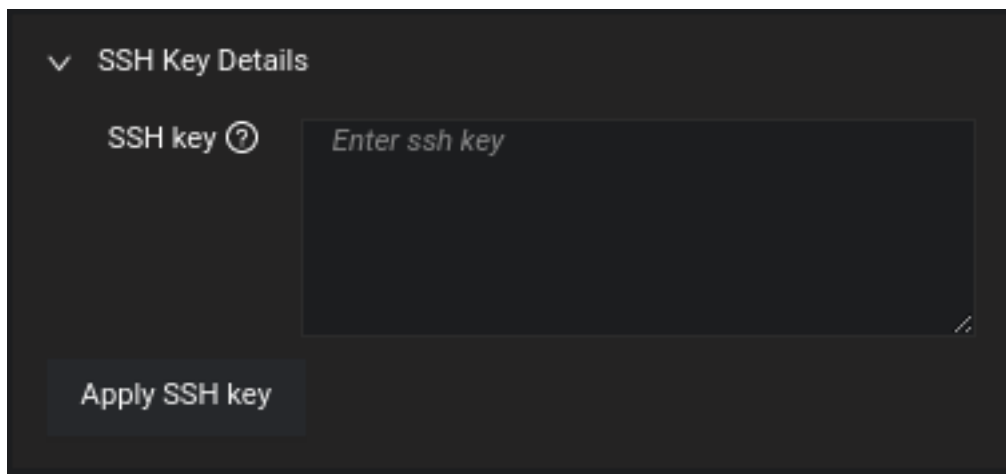


Fig. 7.3: Submitting the public key

Submit your **public key** in the *SSH Key* field and click the *Apply SSH Key* button.

7.1.4 AlertManager integration

This section allows you to configure [integration of Prometheus with an external Alertmanager](#).

- The **Alertmanager URL** field should contain the URL of the Alertmanager which would serve your PMM alerts.
- The **Alertmanager rules** field is used to specify alerting rules in the YAML configuration format.

Fill both fields and click the *Apply Alertmanager settings* button to proceed.

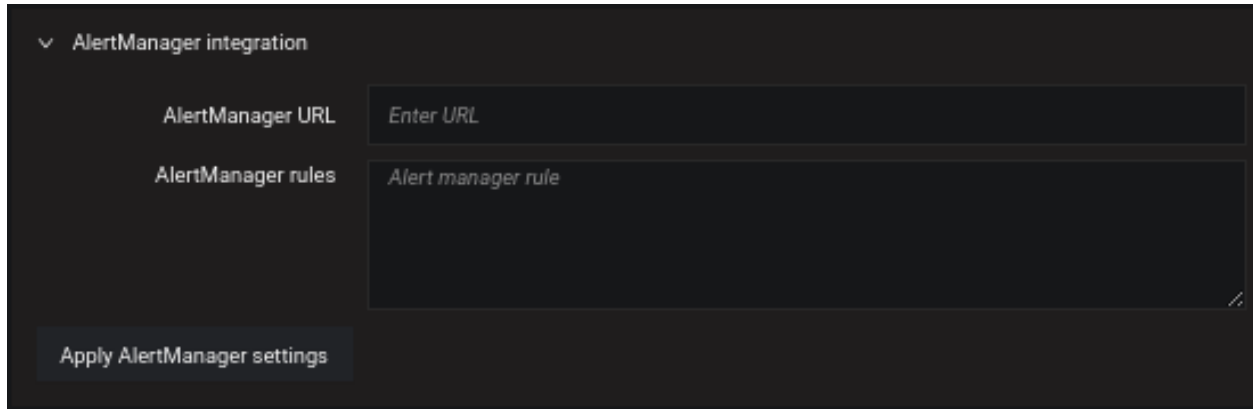


Fig. 7.4: Configuring the Alertmanager integration

7.1.5 Diagnostics

PMM can generate a set of diagnostics data which can be examined and/or shared with Percona Support in case of some issue to solve it faster. You can [get collected logs from PMM Server](#) by clicking the **Download PMM Server Logs** button.

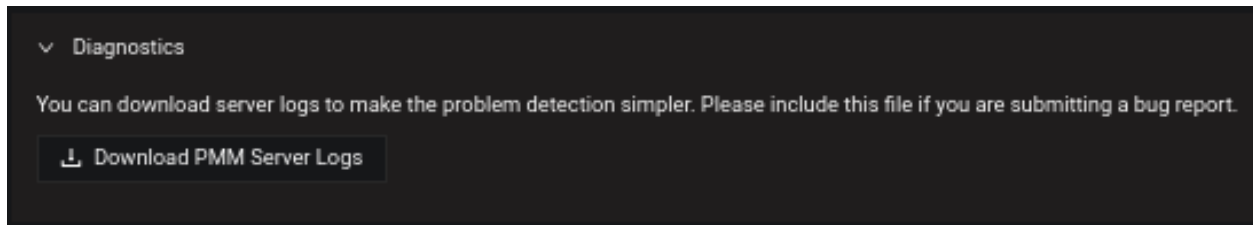


Fig. 7.5: Downloading the PMM Server logs


7.2 Exploring PMM API

PMM Server allows user to visually interact with the API's resources reflecting all objects which PMM "knows" about. Browsing the API can be done using [Swagger UI](#), accessible at the `/swagger` endpoint URL:

Clicking an objects allows to examine objects and execute requests to them:

Objects which can be found while exploring are nodes, services, or agents.

- A **Node** represents a bare metal server, virtual machine or Docker container. It can also be of more specific type: one example is Amazon RDS Node. Node runs zero or more Services and Agents. It also has zero or more Agents providing insights for it.
- A **Service** represents something useful running on the Node: Amazon Aurora MySQL, MySQL, MongoDB, Prometheus, etc. It runs on zero (Amazon Aurora Serverless), single (MySQL), or several (Percona XtraDB Cluster) Nodes. It also has zero or more Agents providing insights for it.
- An **Agent** represents something that runs on the Node which is not useful itself but instead provides insights (metrics, query performance data, etc) about Nodes and/or Services. Always runs on the single Node (except External Exporters), provides insights for zero or more Services and Nodes.

 **swagger** **Explore**

PMM Server API version 1.alpha

[/swagger.json](#)

Schemes

▾

Agents ▾

POST	/v1/inventory/Agents/AddExternalExporter	AddExternalExporter adds External Agent.
POST	/v1/inventory/Agents/AddMongoDBExporter	AddMongoDBExporter adds mongodb_exporter Agent.
POST	/v1/inventory/Agents/AddMySQLdExporter	AddMySQLdExporter adds mysqld_exporter Agent.
POST	/v1/inventory/Agents/AddNodeExporter	AddNodeExporter adds node_exporter Agent.

Curl

```
curl -X POST "http://157.230.168.157/v1/inventory/Nodes/List" -H "accept: application/json" -H "Content-Type: applica
```

Request URL

```
http://157.230.168.157/v1/inventory/Nodes/List
```

Server response

Code

Details

200

Response body

```
{
  "generic": [
    {
      "node_id": "/node_id/22a23494-191b-4583-9b6b-c737d2b03216",
      "node_name": "pmm2-alpha1",
      "machine_id": "0946980476dd1b85c97156b85cbdfdbf\n",
      "distro": "linux",
      "address": "157.230.168.157"
    },
    {
      "node_id": "pmm-server",
      "node_name": "PMM Server",
      "distro": "Linux"
    }
  ]
}
```

Nodes, Services, and Agents have **Types** which define specific properties they have, and the specific logic they implement.

Nodes and Services are external by nature – we do not manage them (create, destroy), but merely maintain a list of them (add to inventory, remove from inventory) in pmm-managed. Most Agents, on the other hand, are started and stopped by pmm-agent. The only exception is the External Exporter Type which is started externally.

Part III

Installing PMM Client

INSTALLING CLIENTS

PMM Client is a package of agents and exporters installed on a database host that you want to monitor. Before installing the PMM Client package on each database host that you intend to monitor, make sure that your PMM Server host is accessible.

For example, you can run the `ping` command passing the IP address of the computer that PMM Server is running on. For example:

```
$ ping 192.168.100.1
```

You will need to have root access on the database host where you will be installing PMM Client (either logged in as a user with root privileges or be able to run commands with `sudo`).

Supported platforms

PMM Client should run on any modern Linux 64-bit distribution, however Percona provides PMM Client packages for automatic installation from software repositories only on the most popular Linux distributions:

- *DEB packages for Debian based distributions such as Ubuntu*
- *RPM packages for Red Hat based distributions such as CentOS*

It is recommended that you install your PMM (Percona Monitoring and Management) client by using the software repository for your system. If this option does not work for you, Percona provides downloadable PMM Client packages from the [Download Percona Monitoring and Management](#) page.

In addition to DEB and RPM packages, this site also offers:

- Generic tarballs that you can extract and run the included `install` script.
- Source code tarball to build your PMM client from source.

Warning: You should not install agents on database servers that have the same host name, because host names are used by PMM Server to identify collected data.

Storage requirements

Minimum **100 MB** of storage is required for installing the PMM Client package. With a good constant connection to PMM Server, additional storage is not required. However, the client needs to store any collected data that it is not able to send over immediately, so additional storage may be required if connection is unstable or throughput is too low.

INSTALLING DEB PACKAGES USING APT-GET

If you are running a DEB-based Linux distribution, use the **apt** package manager to install PMM Client from the official Percona software repository.

Percona provides `.deb` packages for 64-bit versions of the following distributions:

- Debian 8 (jessie)
- Debian 9 (stretch)
- Ubuntu 14.04 LTS (Trusty Tahr)
- Ubuntu 16.04 LTS (Xenial Xerus)
- Ubuntu 16.10 (Yakkety Yak)
- Ubuntu 17.10 (Artful Aardvark)
- Ubuntu 18.04 (Bionic Beaver)

Note: PMM Client should work on other DEB-based distributions, but it is tested only on the platforms listed above.

To install the PMM Client package, complete the following procedure. Run the following commands as root or by using the **sudo** command:

1. Configure Percona repositories using the [percona-release](#) tool. First you'll need to download and install the official `percona-release` package from Percona:

```
wget https://repo.percona.com/apt/percona-release_latest.generic_all.deb
sudo dpkg -i percona-release_latest.generic_all.deb
```

Note: If you have previously enabled the experimental or testing Percona repository, don't forget to disable them and enable the release component of the original repository as follows:

```
sudo percona-release disable all
sudo percona-release enable original release
```

See [percona-release official documentation](#) for details.

2. Install the `pmm2-client` package:

```
sudo apt-get update
sudo apt-get install pmm2-client
```

3. Once PMM Client is installed, run the `pmm-admin config` command with your PMM Server IP address to register your Node within the Server:

```
pmm-admin config --server-insecure-tls --server-url=https://admin:admin@<IP_
↵Address>:443
```

You should see the following:

```
Checking local pmm-agent status...
pmm-agent is running.
Registering pmm-agent on PMM Server...
Registered.
Configuration file /usr/local/percona/pmm-agent.yaml updated.
Reloading pmm-agent configuration...
Configuration reloaded.
```

INSTALLING RPM PACKAGES USING YUM

If you are running an RPM-based Linux distribution, use the **yum** package manager to install PMM Client from the official Percona software repository.

Percona provides `.rpm` packages for 64-bit versions of Red Hat Enterprise Linux 6 (Santiago) and 7 (Maipo), including its derivatives that claim full binary compatibility, such as, CentOS, Oracle Linux, Amazon Linux AMI, and so on.

Note: PMM Client should work on other RPM-based distributions, but it is tested only on RHEL and CentOS versions 6 and 7.

To install the PMM Client package, complete the following procedure. Run the following commands as root or by using the **sudo** command:

1. Configure Percona repositories using the `percona-release` tool. First you'll need to download and install the official `percona-release` package from Percona:

```
sudo yum install https://repo.percona.com/yum/percona-release-latest.noarch.rpm
```

Note: If you have previously enabled the experimental or testing Percona repository, don't forget to disable them and enable the release component of the original repository as follows:

```
sudo percona-release disable all
sudo percona-release enable original release
```

See [percona-release official documentation](#) for details.

2. Install the `pmm2-client` package:

```
yum install pmm2-client
```

3. Once PMM Client is installed, run the `pmm-admin config` command with your PMM Server IP address to register your Node within the Server:

```
pmm-admin config --server-insecure-tls --server-url=https://admin:admin@IP_
↪Address>:443
```

You should see the following:

```
Checking local pmm-agent status...
pmm-agent is running.
Registering pmm-agent on PMM Server...
```

```
Registered.  
Configuration file /usr/local/percona/pmm-agent.yaml updated.  
Reloading pmm-agent configuration...  
Configuration reloaded.
```


Part IV

Using PMM Client

CONFIGURING PMM CLIENT WITH PMM-ADMIN CONFIG

11.1 Connecting PMM Clients to the PMM Server

With your server and clients set up, you must configure each PMM Client and specify which PMM Server it should send its data to.

To connect a PMM Client, enter the IP address of the PMM Server as the value of the `--server-url` parameter to the `pmm-admin config` command, and allow using self-signed certificates with `--server-insecure-tls`.

Note: The `--server-url` argument should include `https://` prefix and PMM Server credentials, which are `admin/admin` by default, if not changed at first PMM Server GUI access.

Run this command as root or by using the `sudo` command

```
$ pmm-admin config --server-insecure-tls --server-url=https://admin:admin@192.168.100.1:443
```

For example, if your PMM Server is running on `192.168.100.1`, you have installed PMM Client on a machine with IP `192.168.200.1`, and didn't change default PMM Server credentials, run the following in the terminal of your client. Run the following commands as root or by using the `sudo` command:

```
# pmm-admin config --server-insecure-tls --server-url=https://admin:admin@192.168.100.1:443
Checking local pmm-agent status...
pmm-agent is running.
Registering pmm-agent on PMM Server...
Registered.
Configuration file /usr/local/percona/pmm-agent.yaml updated.
Reloading pmm-agent configuration...
Configuration reloaded.
Checking local pmm-agent status...
pmm-agent is running.
```

If you change the default port `443` when running PMM Server, specify the new port number after the IP address of PMM Server.

Note: By default `pmm-admin config` refuses to add client if it already exists in the PMM Server inventory database. If you need to re-add an already existing client (e.g. after full reinstall, hostname changes, etc.), you can run `pmm-admin config` with the additional `--force` option. This will remove an existing node with the same name, if any, and all its dependent services.

ADDING MYSQL SERVICE MONITORING

You then add MySQL services (Metrics and Query Analytics) with the following command:

USAGE

```
pmm-admin add mysql --query-source=slowlog --username=pmm --password=pmm
```

where username and password are credentials for the monitored MySQL access, which will be used locally on the database host. Additionally, two positional arguments can be appended to the command line flags: a service name to be used by PMM, and a service address. If not specified, they are substituted automatically as `<node>-mysql` and `127.0.0.1:3306`.

The command line and the output of this command may look as follows:

```
# pmm-admin add mysql --query-source=slowlog --username=pmm --password=pmm sl-mysql_
↪127.0.0.1:3306
MySQL Service added.
Service ID   : /service_id/a89191d4-7d75-44a9-b37f-a528e2c4550f
Service name: sl-mysql
```

Note: There are two possible sources for query metrics provided by MySQL to get data for the Query Analytics: the [Slow Log](#) and the [Performance Schema](#). The `--query-source` option can be used to specify it, either as `slowlog` (it is also used by default if nothing specified) or as `perfschema`:

```
pmm-admin add mysql --username=pmm --password=pmm --query-source=perfschema ps-mysql_
↪127.0.0.1:3306
```

Beside positional arguments shown above you can specify service name and service address with the following flags: `--service-name`, `--host` (the hostname or IP address of the service), and `--port` (the port number of the service). If both flag and positional argument are present, flag gains higher priority. Here is the previous example modified to use these flags:

```
pmm-admin add mysql --username=pmm --password=pmm --service-name=ps-mysql --host=127.
↪0.0.1 --port=3306
```

Note: It is also possible to add MySQL instance using UNIX socket with use of a special `--socket` flag followed with the path to a socket without username, password and network type:

```
pmm-admin add mysql --socket=/var/path/to/mysql/socket
```

After adding the service you can view MySQL metrics or examine the added node on the new PMM Inventory Dashboard.

ADDING MONGODB SERVICE MONITORING

Before adding MongoDB should be [prepared for the monitoring](#), which involves creating the user, and setting the profiling level.

When done, add monitoring as follows:

```
pmm-admin add mongodb --username=pmm --password=pmm
```

where username and password are credentials for the monitored MongoDB access, which will be used locally on the database host. Additionally, two positional arguments can be appended to the command line flags: a service name to be used by PMM, and a service address. If not specified, they are substituted automatically as `<node>-mongodb` and `127.0.0.1:27017`.

The command line and the output of this command may look as follows:

```
# pmm-admin add mongodb --username=pmm --password=pmm mongo 127.0.0.1:27017
MongoDB Service added.
Service ID : /service_id/f1af8a88-5a95-4bf1-a646-0101f8a20791
Service name: mongo
```

Beside positional arguments shown above you can specify service name and service address with the following flags: `--service-name`, `--host` (the hostname or IP address of the service), and `--port` (the port number of the service). If both flag and positional argument are present, flag gains higher priority. Here is the previous example modified to use these flags:

```
pmm-admin add mongodb --username=pmm --password=pmm --service-name=mongo --host=127.0.
↪0.1 --port=27017
```


ADDING A PROXYSQL HOST

14.1 Adding ProxySQL metrics service

Use the `proxysql` alias to enable ProxySQL performance metrics monitoring.

USAGE

```
pmm-admin add proxysql --username=admin --password=admin
```

where `username` and `password` are credentials for the monitored MongoDB access, which will be used locally on the database host. Additionally, two positional arguments can be appended to the command line flags: a service name to be used by PMM, and a service address. If not specified, they are substituted automatically as `<node>-proxysql` and `127.0.0.1:3306`.

The output of this command may look as follows:

```
# pmm-admin add proxysql --username=admin --password=admin
ProxySQL Service added.
Service ID   : /service_id/f69df379-6584-4db5-a896-f35ae8c97573
Service name: ubuntu-proxysql
```

Beside positional arguments shown above you can specify service name and service address with the following flags: `--service-name`, `--host` (the hostname or IP address of the service), and `--port` (the port number of the service). If both flag and positional argument are present, flag gains higher priority. Here is the previous example modified to use these flags:

```
pmm-admin add proxysql --username=pmm --password=pmm --service-name=my-new-proxysql --
↪host=127.0.0.1 --port=3306
```


ADDING LINUX METRICS

15.1 Adding general system metrics service

PMM2 collects Linux metrics automatically starting from the moment when you have configured your node for monitoring with `pmm-admin config`.

POSTGRESQL

PMM provides both metrics and queries monitoring for PostgreSQL. Queries monitoring needs additional `pg_stat_statements` extension to be installed and enabled.

16.1 Adding PostgreSQL extension for queries monitoring

The needed extension is `pg_stat_statements`. It is included in the official PostgreSQL contrib package, so you have to install this package first with your Linux distribution package manager. Particularly, on Debian-based systems it is done as follows:

```
sudo apt-get install postgresql-contrib
```

Now add/edit the following three lines in your `postgres.conf` file:

```
shared_preload_libraries = 'pg_stat_statements'  
track_activity_query_size = 2048  
pg_stat_statements.track = all
```

Besides making the appropriate module to be loaded, these edits will increase the maximum size of the query strings PostgreSQL records and will allow it to track all statements including nested ones. When the editing is over, restart PostgreSQL.

Finally, the following statement should be executed in the PostgreSQL shell to install the extension:

```
CREATE EXTENSION pg_stat_statements SCHEMA public;
```

Note: `CREATE EXTENSION` statement should be run in the `postgres` database.

16.2 Adding PostgreSQL queries and metrics monitoring

You can add PostgreSQL metrics and queries monitoring with the following command:

```
pmm-admin add postgresql --username=pmm --password=pmm
```

where `username` and `password` parameters should contain actual PostgreSQL user credentials (for more information about `pmm-admin add`, see `pmm-admin.add`). Additionally, two positional arguments can be appended to the command line flags: a service name to be used by PMM, and a service address. If not specified, they are substituted automatically as `<node>-postgresql` and `127.0.0.1:5432`.

The command line and the output of this command may look as follows:

```
# pmm-admin add postgresql --username=pmm --password=pmm postgres 127.0.0.1:5432
PostgreSQL Service added.
Service ID   : /service_id/28f1d93a-5c16-467f-841b-8c014bf81ca6
Service name: postgres
```

As a result, you should be able to see data in PostgreSQL Overview dashboard, and also Query Analytics should contain PostgreSQL queries, if the needed extension was installed and configured correctly.

Beside positional arguments shown above you can specify service name and service address with the following flags: `--service-name`, `--host` (the hostname or IP address of the service), and `--port` (the port number of the service). If both flag and positional argument are present, flag gains higher priority. Here is the previous example modified to use these flags:

```
pmm-admin add postgresql --username=pmm --password=pmm --service-name=postgres --
↪host=127.0.0.1 --port=270175432
```

Note: Capturing read and write time statistics is possible only if `track_io_timing` setting is enabled. This can be done either in configuration file or with the following query executed on the running system:

```
ALTER SYSTEM SET track_io_timing=ON;
SELECT pg_reload_conf();
```

16.3 Setting up the required user permissions and authentication

Percona recommends that a PostgreSQL user be configured for `SUPERUSER` level access, in order to gather the maximum amount of data with a minimum amount of complexity. This can be done with the following command for the standalone PostgreSQL installation:

```
CREATE USER pmm_user WITH SUPERUSER ENCRYPTED PASSWORD 'secret';
```

Note: In case of monitoring a PostgreSQL database running on an Amazon RDS instance, the command should look as follows:

```
CREATE USER pmm_user WITH rds_superuser ENCRYPTED PASSWORD 'secret';
```

Note: Specified PostgreSQL user should have enabled local password authentication to enable access for PMM. This can be set in the `pg_hba.conf` configuration file changing `ident` to `md5` for the correspondent user. Also, this user should be able to connect to the `postgres` database which we have installed the extension into.

REMOVING MONITORING SERVICES WITH PMM-ADMIN REMOVE

Use the `pmm-admin remove` command to remove monitoring services.

USAGE

Run this command as root or by using the `sudo` command

```
pmm-admin remove [OPTIONS] [SERVICE-TYPE] [SERVICE-NAME]
```

When you remove a service, collected data remains in Metrics Monitor on PMM Server.

SERVICES

Service type can be *mysql*, *mongodb*, *postgresql* or *proxysql*, and service name is a monitoring service alias. To see which services are enabled, run `pmm-admin list`.

EXAMPLES

- Removing MySQL service named “mysql-sl”:

```
# pmm-admin remove mysql mysql-sl  
Service removed.
```

- To remove *MongoDB* service named “mongo”:

```
# pmm-admin remove mongodb mongo  
Service removed.
```

- To remove *PostgreSQL* service named “postgres”:

```
# pmm-admin remove postgresql postgres  
Service removed.
```

- To remove *ProxySQL* service named “ubuntu-proxysql”:

```
# pmm-admin remove proxysql ubuntu-proxysql  
Service removed.
```

For more information, run `pmm-admin remove -help`.

Part V

Service configuration for best results

18.1 Slow Log Settings

If you are running Percona Server, a properly configured slow query log will provide the most amount of information with the lowest overhead. In other cases, use *Performance Schema* if it is supported.

By definition, the slow query log is supposed to capture only *slow queries*. These are the queries the execution time of which is above a certain threshold. The threshold is defined by the `long_query_time` variable.

In heavily loaded applications, frequent fast queries can actually have a much bigger impact on performance than rare slow queries. To ensure comprehensive analysis of your query traffic, set the `long_query_time` to `0` so that all queries are captured.

However, capturing all queries can consume I/O bandwidth and cause the *slow query log* file to quickly grow very large. To limit the amount of queries captured by the *slow query log*, use the *query sampling* feature available in Percona Server.

A possible problem with query sampling is that rare slow queries might not get captured at all. To avoid this, use the `slow_query_log_always_write_time` variable to specify which queries should ignore sampling. That is, queries with longer execution time will always be captured by the slow query log.

18.2 Configuring Performance Schema

The default source of query data for PMM is the *slow query log*. It is available in MySQL 5.1 and later versions. Starting from MySQL 5.6 (including Percona Server 5.6 and later), you can choose to parse query data from the *Performance Schema* instead of *slow query log*. Starting from MySQL 5.6.6, *Performance Schema* is enabled by default.

Performance Schema is not as data-rich as the *slow query log*, but it has all the critical data and is generally faster to parse. If you are not running Percona Server (which supports sampling for the slow query log), then *Performance Schema* is a better alternative.

Note: Use of the performance schema is off by default in MariaDB 10.x.

To use *Performance Schema*, set the `performance_schema` variable to ON:

```
mysql> SHOW VARIABLES LIKE 'performance_schema';
+-----+-----+
| Variable_name | Value |
+-----+-----+
```

```
| performance_schema | ON |
+-----+-----+
```

If this variable is not set to **ON**, add the the following lines to the MySQL configuration file `my.cnf` and restart MySQL:

```
[mysql]
performance_schema=ON
```

If you are running a custom Performance Schema configuration, make sure that the `statements_digest` consumer is enabled:

```
mysql> select * from setup_consumers;
+-----+-----+
| NAME                                | ENABLED |
+-----+-----+
| events_stages_current               | NO      |
| events_stages_history               | NO      |
| events_stages_history_long          | NO      |
| events_statements_current           | YES     |
| events_statements_history           | YES     |
| events_statements_history_long      | NO      |
| events_transactions_current         | NO      |
| events_transactions_history         | NO      |
| events_transactions_history_long    | NO      |
| events_waits_current                | NO      |
| events_waits_history                | NO      |
| events_waits_history_long           | NO      |
| global_instrumentation              | YES     |
| thread_instrumentation              | YES     |
| statements_digest                   | YES     |
+-----+-----+
15 rows in set (0.00 sec)
```

Important: *Performance Schema* instrumentation is enabled by default in MySQL 5.6.6 and later versions. It is not available at all in MySQL versions prior to 5.6.

If certain instruments are not enabled, you will not see the corresponding graphs in the *MySQL Performance Schema Dashboard* dashboard. To enable full instrumentation, set the option `--performance_schema_instrument` to `'%=on'` when starting the MySQL server.

```
$ mysqld --performance-schema-instrument='%=on'
```

This option can cause additional overhead and should be used with care.

See also:

MySQL Documentation: `--performance_schema_instrument` option https://dev.mysql.com/doc/refman/5.7/en/performance-schema-options.html#option_mysqld_performance-schema-instrument

If the instance is already running, configure the QAN agent to collect data from *Performance Schema*:

1. Open the *PMM Query Analytics* dashboard.
2. Click the *Settings* button.
3. Open the *Settings* section.

4. Select `Performance Schema` in the *Collect from* drop-down list.
5. Click *Apply* to save changes.

If you are adding a new monitoring instance with the `pmm-admin` tool, use the `--query-source` *perfschema* option:

Run this command as root or by using the `sudo` command

```
pmm-admin add mysql --username=pmm --password=pmpassword --query-source='perfschema'
↪ps-mysql 127.0.0.1:3306
```

For more information, run `pmm-admin add mysql --help`.

18.3 MySQL InnoDB Metrics

Collecting metrics and statistics for graphs increases overhead. You can keep collecting and graphing low-overhead metrics all the time, and enable high-overhead metrics only when troubleshooting problems.

InnoDB metrics provide detailed insight about InnoDB operation. Although you can select to capture only specific counters, their overhead is low even when they all are enabled all the time. To enable all InnoDB metrics, set the global variable `innodb_monitor_enable` to `all`:

```
mysql> SET GLOBAL innodb_monitor_enable=all
```

See also:

MySQL Documentation: `innodb_monitor_enable` variable https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html#sysvar_innodb_monitor_enable

18.4 Percona Server specific settings

Not all dashboards in Metrics Monitor are available by default for all MySQL variants and configurations: Oracle's MySQL, Percona Server, or MariaDB. Some graphs require Percona Server, and specialized plugins, or additional configuration.

18.4.1 MySQL User Statistics (`userstat`)

User statistics is a feature of Percona Server and MariaDB. It provides information about user activity, individual table and index access. In some cases, collecting user statistics can lead to high overhead, so use this feature sparingly.

To enable user statistics, set the `userstat` variable to `1`.

See also:

Percona Server Documentation: `userstat` https://www.percona.com/doc/percona-server/5.7/diagnostics/user_stats.html#userstat

MySQL Documentation [Setting variables](#)

18.4.2 Query Response Time Plugin

Query response time distribution is a feature available in Percona Server. It provides information about changes in query response time for different groups of queries, often allowing to spot performance problems before they lead to serious issues.

To enable collection of query response time:

1. Install the `QUERY_RESPONSE_TIME` plugins:

```
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME_AUDIT SONAME 'query_response_time.so';
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME SONAME 'query_response_time.so';
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME_READ SONAME 'query_response_time.so';
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME_WRITE SONAME 'query_response_time.so';
```

2. Set the global variable `query_response_time_stats` to ON:

```
mysql> SET GLOBAL query_response_time_stats=ON;
```

Related Information: Percona Server Documentation

- `query_response_time_stats`: https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#query_response_time_stats
 - Response time distribution: https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#installing-the-plugins
-

18.4.3 log_slow_rate_limit

The `log_slow_rate_limit` variable defines the fraction of queries captured by the *slow query log*. A good rule of thumb is to have approximately 100 queries logged per second. For example, if your Percona Server instance processes 10_000 queries per second, you should set `log_slow_rate_limit` to 100 and capture every 100th query for the *slow query log*.

Note: When using query sampling, set `log_slow_rate_type` to `query` so that it applies to queries, rather than sessions.

It is also a good idea to set `log_slow_verbosity` to `full` so that maximum amount of information about each captured query is stored in the slow query log.

See also:

[MySQL Documentation](#) [Setting variables](#)

18.4.4 log_slow_verbosity

`log_slow_verbosity` variable specifies how much information to include in the slow query log. It is a good idea to set `log_slow_verbosity` to `full` so that maximum amount of information about each captured query is stored.

See also:

[MySQL Documentation](#) [Setting variables](#)

18.4.5 slow_query_log_use_global_control

By default, slow query log settings apply only to new sessions. If you want to configure the slow query log during runtime and apply these settings to existing connections, set the `slow_query_log_use_global_control` variable to `all`.

See also:

MySQL Documentation [Setting variables](#)

18.5 Configuring MySQL 8.0 for PMM

MySQL 8 (in version 8.0.4) changes the way clients are authenticated by default. The `default_authentication_plugin` parameter is set to `caching_sha2_password`. This change of the default value implies that MySQL drivers must support the SHA-256 authentication. Also, the communication channel with MySQL 8 must be encrypted when using `caching_sha2_password`.

The MySQL driver used with PMM does not yet support the SHA-256 authentication.

With currently supported versions of MySQL, PMM requires that a dedicated MySQL user be set up. This MySQL user should be authenticated using the `mysql_native_password` plugin. Although MySQL is configured to support SSL clients, connections to MySQL Server are not encrypted.

There are two workarounds to be able to add MySQL Server version 8.0.4 or higher as a monitoring service to PMM:

1. Alter the MySQL user that you plan to use with PMM
2. Change the global MySQL configuration

Altering the MySQL User

Provided you have already created the MySQL user that you plan to use with PMM, alter this user as follows:

Then, pass this user to `pmm-admin add` as the value of the `--username` parameter.

This is a preferred approach as it only weakens the security of one user.

Changing the global MySQL Configuration

A less secure approach is to set `default_authentication_plugin` to the value `mysql_native_password` before adding it as a monitoring service. Then, restart your MySQL Server to apply this change.

See also:

Creating a MySQL User for PMM [privileges](#)

More information about adding the MySQL query analytics monitoring service [pmm-admin.add-mysql-queries](#)

MySQL Server Blog: MySQL 8.0.4 [New Default Authentication Plugin][`caching_sha2_password`] https://mysqlserverteam.com/mysql-8-0-4-new-default-authentication-plugin-caching_sha2_password/

MySQL Documentation: Authentication Plugins <https://dev.mysql.com/doc/refman/8.0/en/authentication-plugins.html>

MySQL Documentation: Native Pluggable Authentication <https://dev.mysql.com/doc/refman/8.0/en/native-pluggable-authentication.html>

19.1 Passing SSL parameters to the mongodb monitoring service

SSL/TLS related parameters are passed to an SSL enabled MongoDB server as monitoring service parameters along with the `pmm-admin add` command when adding the MongoDB monitoring service.

Run this command as root or by using the `sudo` command

Listing 19.1: *Passing an SSL/TLS parameter to `mongod` to enable a TLS connection.*

```
$ pmm-admin add mongodb -- --mongodb.tls
```

Table 19.1: Supported SSL/TLS Parameters

Parameter	Description
<code>--mongodb.tls</code>	Enable a TLS connection with mongo server
<code>--mongodb.tls-ca</code> <i>string</i>	A path to a PEM file that contains the CAs that are trusted for server connections. <i>If provided:</i> MongoDB servers connecting to should present a certificate signed by one of these CAs. <i>If not provided:</i> System default CAs are used.
<code>--mongodb.tls-cert</code> <i>string</i>	A path to a PEM file that contains the certificate and, optionally, the private key in the PEM format. This should include the whole certificate chain. <i>If provided:</i> The connection will be opened via TLS to the MongoDB server.
<code>--mongodb.tls-disable-hostname-validation</code>	Do hostname validation for the server connection.
<code>--mongodb.tls-private-key</code> <i>string</i>	A path to a PEM file that contains the private key (if not contained in the <code>mongodb.tls-cert</code> file).

```
$ mongod --dbpath=DATABASEDIR --profile 2 --slowms 200 --rateLimit 100
```


Part VI

Configuring AWS RDS or Aurora

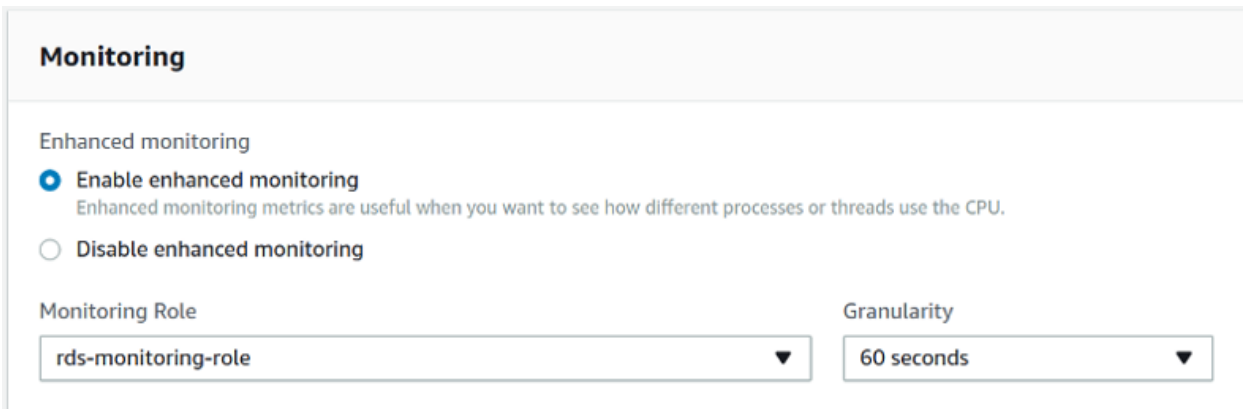
REQUIRED AWS SETTINGS

It is possible to use PMM for monitoring Amazon RDS (just like any remote MySQL instance). In this case, the PMM Client is not installed on the host where the database server is deployed. By using the PMM web interface, you connect to the Amazon RDS DB instance. You only need to provide the IAM user access key (or assign an IAM role) and PMM discovers the Amazon RDS DB instances available for monitoring.

First of all, ensure that there is the minimal latency between PMM Server and the Amazon RDS instance.

Network connectivity can become an issue for Prometheus to scrape metrics with 1 second resolution. We strongly suggest that you run PMM Server on AWS (Amazon Web Services) in the same availability zone as Amazon RDS instances.

It is crucial that *enhanced monitoring* be enabled for the Amazon RDS DB instances you intend to monitor.



The screenshot shows the 'Monitoring' configuration page for an Amazon RDS DB instance. Under the 'Enhanced monitoring' section, the 'Enable enhanced monitoring' radio button is selected. A note below it states: 'Enhanced monitoring metrics are useful when you want to see how different processes or threads use the CPU.' The 'Disable enhanced monitoring' radio button is unselected. Below this, there are two dropdown menus: 'Monitoring Role' is set to 'rds-monitoring-role' and 'Granularity' is set to '60 seconds'.

Fig. 20.1: Set the *Enable Enhanced Monitoring* option in the settings of your Amazon RDS DB instance.

See also:

Amazon RDS Documentation:

- [Modifying an Amazon RDS DB Instance](#)
- [More information about enhanced monitoring](#)
- [Setting Up](#)
- [Getting started](#)
- [Creating a MySQL DB Instance](#)
- [Availability zones](#)
- [What privileges are automatically granted to the master user of an Amazon RDS DB instance?](#)

Which ports should be open? See Ports in glossary

- *Creating an IAM user with permission to access Amazon RDS DB instances*
- *Creating a policy*
- *Creating an IAM user*
- *Creating an access key for an IAM user*
- *Attaching a policy to an IAM user*
- *Setting up the Amazon RDS DB Instance*

20.1 Creating an IAM user with permission to access Amazon RDS DB instances

It is recommended that you use an IAM user account to access Amazon RDS DB instances instead of using your AWS account. This measure improves security as the permissions of an IAM user account can be limited so that this account only grants access to your Amazon RDS DB instances. On the other hand, you use your AWS account to access all AWS services.

The procedure for creating IAM user accounts is well described in the Amazon RDS documentation. This section only goes through the essential steps and points out the steps required for using Amazon RDS with Percona Monitoring and Management.

See also:

Amazon RDS Documentation: Creating an IAM user https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SettingUp.html#CHAP_SettingUp.IAM

The first step is to define a policy which will hold all the necessary permissions. Then, you need to associate this policy with the IAM user or group. In this section, we will create a new user for this purpose.

20.2 Creating a policy

A policy defines how AWS services can be accessed. Once defined it can be associated with an existing user or group. To define a new policy use the IAM page at AWS.

1. Select the *Policies* option on the navigation panel and click the *Create policy* button.
2. On the *Create policy* page, select the JSON tab and replace the existing contents with the following JSON document.

```
{ "Version": "2012-10-17",
  "Statement": [{ "Sid": "Stmt1508404837000",
                  "Effect": "Allow",
                  "Action": [ "rds:DescribeDBInstances",
                             "cloudwatch:GetMetricStatistics",
                             "cloudwatch:ListMetrics" ],
                  "Resource": [ "*" ] },
                { "Sid": "Stmt1508410723001",
                  "Effect": "Allow",
                  "Action": [ "logs:DescribeLogStreams",
```

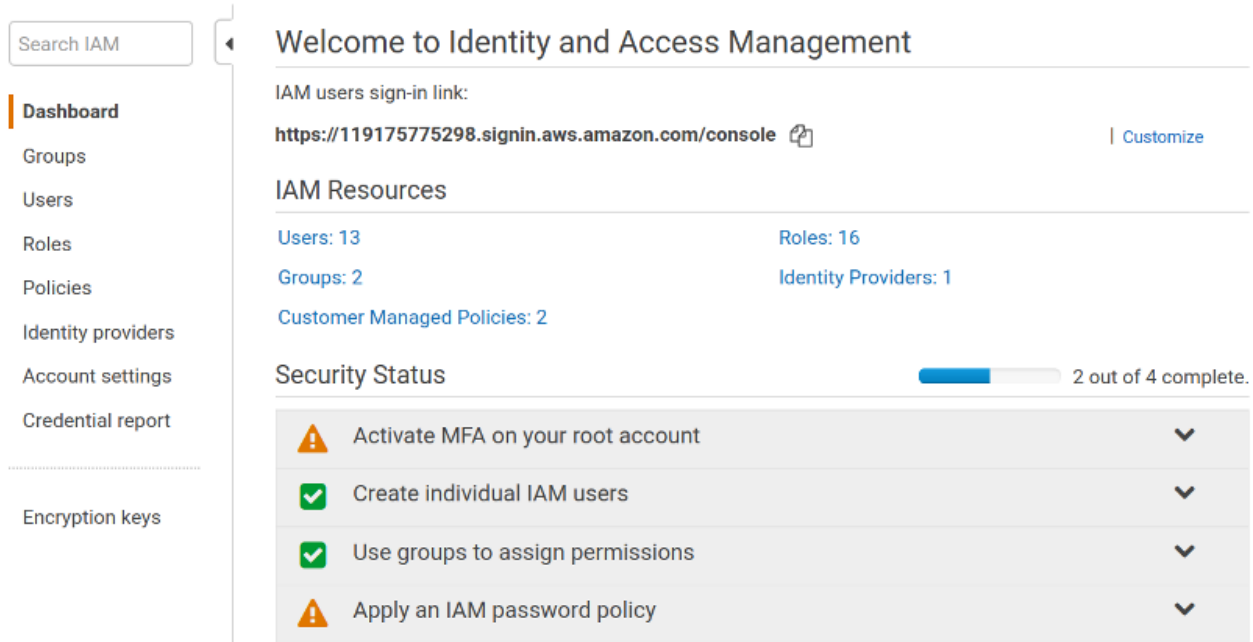


Fig. 20.2: The IAM page at AWS

```

    "logs:GetLogEvents",
    "logs:FilterLogEvents" ],
    "Resource": [ "arn:aws:logs:*:*:log-
->group:RDSOSMetrics:*" ] }
  ]
}

```

3. Click *Review policy* and set a name to your policy, such as *AmazonRDSforPMMPolicy*. Then, click the *Create policy* button.

See also:

AWS Documentation: Creating IAM policies https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html

20.3 Creating an IAM user

Policies are attached to existing IAM users or groups. To create a new IAM user, select *Users* on the Identity and Access Management page at AWS. Then click *Add user* and complete the following steps:

1. On the *Add user* page, set the user name and select the *Programmatic access* option under *Select AWS access type*. Set a custom password and then proceed to permissions by clicking the *Permissions* button.
2. On the *Set permissions* page, add the new user to one or more groups if necessary. Then, click *Review*.
3. On the *Add user* page, click *Create user*.

See also:**AWS Documentation:**

- [Creating IAM users](#)

Create policy

1 2

Review policy

Name*

Use alphanumeric and '+*,@-_' characters. Maximum 128 characters.

Description

Maximum 1000 characters. Use alphanumeric and '+*,@-_' characters.

Summary

Service ▾	Access level	Resource	Request condition
Allow (3 of 133 services) Show remaining 130			
CloudWatch	Full: List Limited: Read	All resources	None
CloudWatch Logs	Limited: List, Read	arn:aws:logs:*:*:log-group:RDSOSMetrics:*	None
RDS	Limited: List	All resources	None

* Required
[Cancel](#) [Previous](#) [Create policy](#)

Fig. 20.3: A new policy is ready to be created.

Add user
Delete user

↻ ⚙️ 🔒

User name ▾	Groups	Access key age	Password age	Last activity	MFA	Access k
-------------	--------	----------------	--------------	---------------	-----	----------

- Dashboard
- Groups
- Users
- Roles
- Policies
- Identity providers
- Account settings
- Credential report

- Encryption keys

Fig. 20.4: Navigate to *Users* on the IAM console

- IAM roles

20.4 Creating an access key for an IAM user

In order to be able to discover an Amazon RDS DB instance in PMM, you either need to use the access key and secret access key of an existing IAM user or an IAM role. To create an access key for use with PMM, open the IAM console and click *Users* on the navigation pane. Then, select your IAM user.

To create the access key, open the *Security credentials* tab and click the *Create access key* button. The system automatically generates a new access key ID and a secret access key that you can provide on the *PMM Add Instance* dashboard to have your Amazon RDS DB instances discovered.

Important: You may use an IAM role instead of IAM user provided your Amazon RDS DB instances are associated with the same AWS account as PMM.

In case, the PMM Server and Amazon RDS DB instance were created by using the same AWS account, you do not need create the access key ID and secret access key manually. PMM retrieves this information automatically and attempts to discover your Amazon RDS DB instances.

See also:

AWS Documentation: Managing access keys of IAM users https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html

20.5 Attaching a policy to an IAM user

The last step before you are ready to create an Amazon RDS DB instance is to attach the policy with the required permissions to the IAM user.

First, make sure that the Identity and Access Management page is open and open *Users*. Then, locate and open the IAM user that you plan to use with Amazon RDS DB instances. Complete the following steps, to apply the policy:

1. On the *Permissions* tab, click the *Add permissions* button.
2. On the *Add permissions* page, click *Attach existing policies directly*.
3. Using the *Filter*, locate the policy with the required permissions (such as *AmazonRDSforPMMPolicy*).
4. Select a checkbox next to the name of the policy and click *Review*.
5. The selected policy appears on the *Permissions summary* page. Click *Add permissions*.

The *AmazonRDSforPMMPolicy* is now added to your IAM user.

See also:

Creating an IAM policy for PMM *Creating a policy*

20.6 Setting up the Amazon RDS DB Instance

Query Analytics requires *Configuring Performance Schema* as the query source, because the slow query log is stored on the AWS side, and QAN agent is not able to read it. Enable the `performance_schema` option under `Parameter Groups` in Amazon RDS.

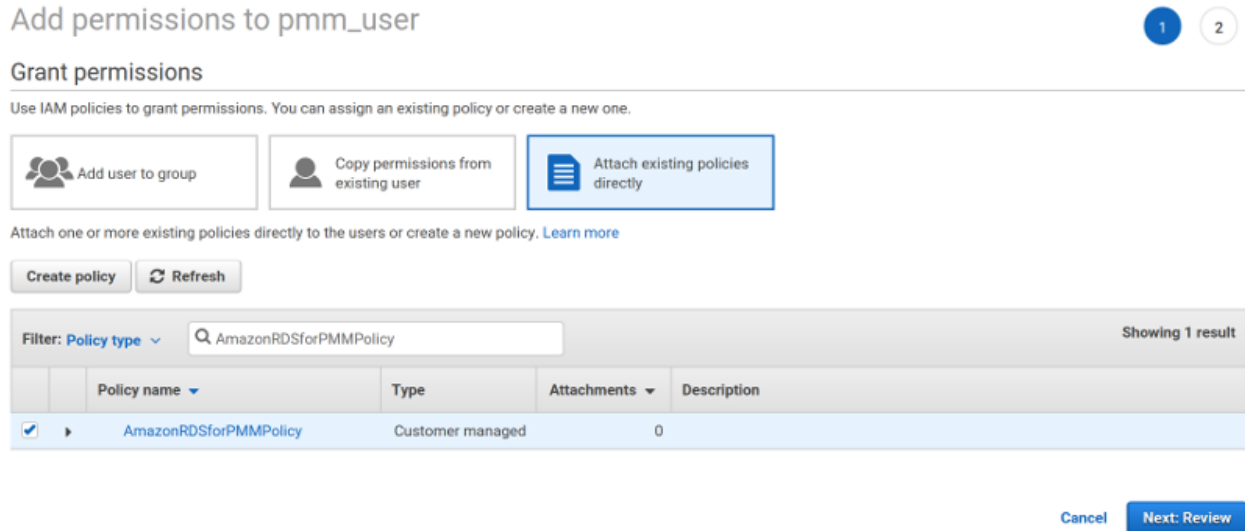


Fig. 20.5: To attach, find the policy on the list and place a check mark to select it

Warning: Enabling Performance Schema on T2 instances is not recommended because it can easily run the T2 instance out of memory.

See also:

More information about the performance schema See *Configuring Performance Schema*.

AWS Documentation: Parameter groups https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithParamGroups.html

When adding a monitoring instance for Amazon RDS, specify a unique name to distinguish it from the local MySQL instance. If you do not specify a name, it will use the client's host name.

Create the `pmm` user with the following privileges on the Amazon RDS instance that you want to monitor:

```
GRANT SELECT, PROCESS, REPLICATION CLIENT ON *.* TO 'pmm'@'%' IDENTIFIED BY 'pass'
↳ WITH MAX_USER_CONNECTIONS 10;
GRANT SELECT, UPDATE, DELETE, DROP ON performance_schema.* TO 'pmm'@'%';
```

If you have Amazon RDS with a MySQL version prior to 5.5, `REPLICATION CLIENT` privilege is not available there and has to be excluded from the above statement.

Note: General system metrics are monitored by using the `rds_exporter` Prometheus exporter which replaces `node_exporter`. `rds_exporter` gives access to Amazon Cloudwatch metrics.

`node_exporter`, used in versions of PMM prior to 1.8.0, was not able to monitor general system metrics remotely.

See also:

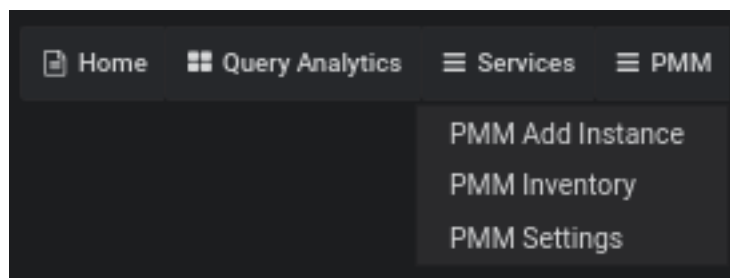
AWS Documentation: Connecting to a DB instance (MySQL engine) https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ConnectToInstance.html

ADDING AN AMAZON RDS MYSQL, AURORA MYSQL, OR REMOTE INSTANCE

The *PMM Add Instance* is now a preferred method of adding an Amazon RDS database instance to PMM. This method supports Amazon RDS database instances that use Amazon Aurora, MySQL, or MariaDB engines, as well as any remote PostgreSQL, ProxySQL, MySQL and MongoDB instances.

Following steps are needed to add an Amazon RDS database instance to PMM:

1. Open the PMM web interface and select the *PMM Add Instance* dashboard.



Choosing the *PMM Add instance* menu entry

2. Select the *Add an AWS RDS MySQL or Aurora MySQL Instance* option in the dashboard.
3. Enter the access key ID and the secret access key of your IAM user.

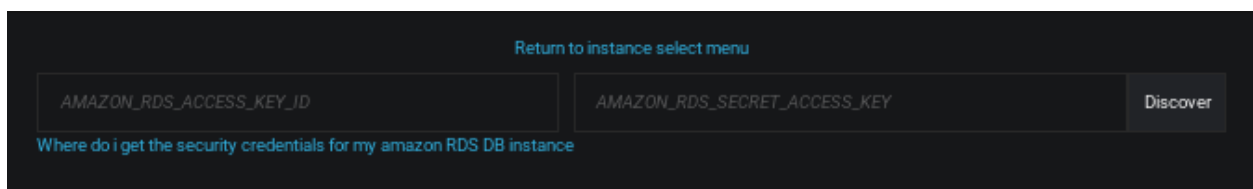
A screenshot of the PMM 'Add Instance' form. At the top, there is a link that says 'Return to instance select menu'. Below this, there are two input fields: 'AMAZON_RDS_ACCESS_KEY_ID' and 'AMAZON_RDS_SECRET_ACCESS_KEY'. To the right of these fields is a 'Discover' button. Below the input fields, there is a blue link that says 'Where do I get the security credentials for my Amazon RDS DB instance'.

Fig. 21.1: Enter the access key ID and the secret access key of your IAM user

4. Click the *Discover* button for PMM to retrieve the available Amazon RDS instances.

For the instance that you would like to monitor, select the *Start monitoring* button.

5. You will see a new page with the number of fields. The list is divided into the following groups: *Main details*, *RDS database*, *Labels*, and *Additional options*. Some already known data, such as already entered *AWS access key*, are filled in automatically, and some fields are optional.

The *Main details* section allows to specify the DNS hostname of your instance, service name to use within PMM, the port your service is listening on, the database user name and password.

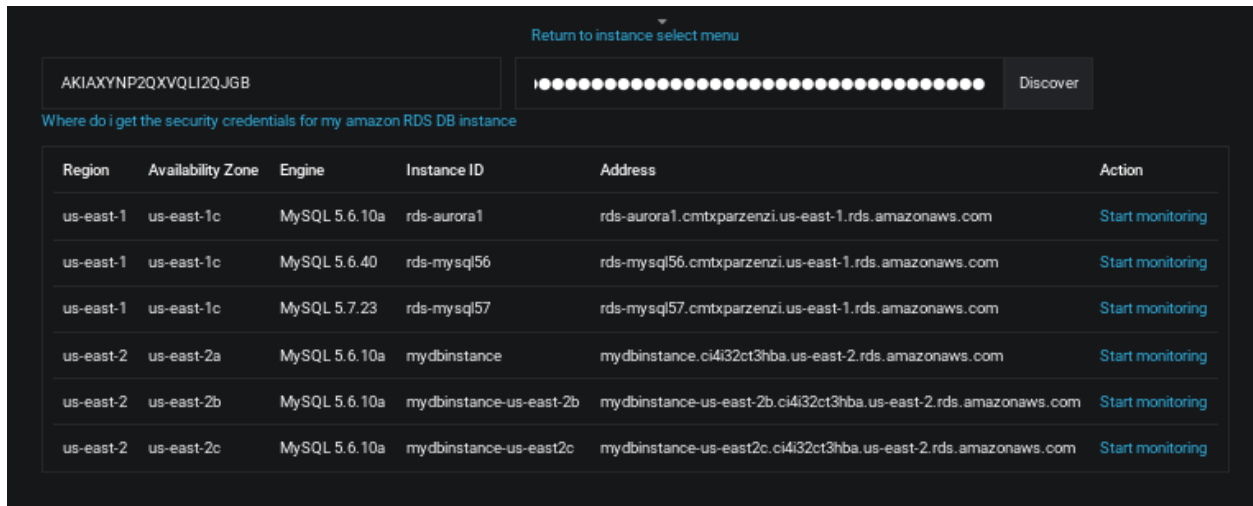


Fig. 21.2: PMM displays the available Amazon RDS instances

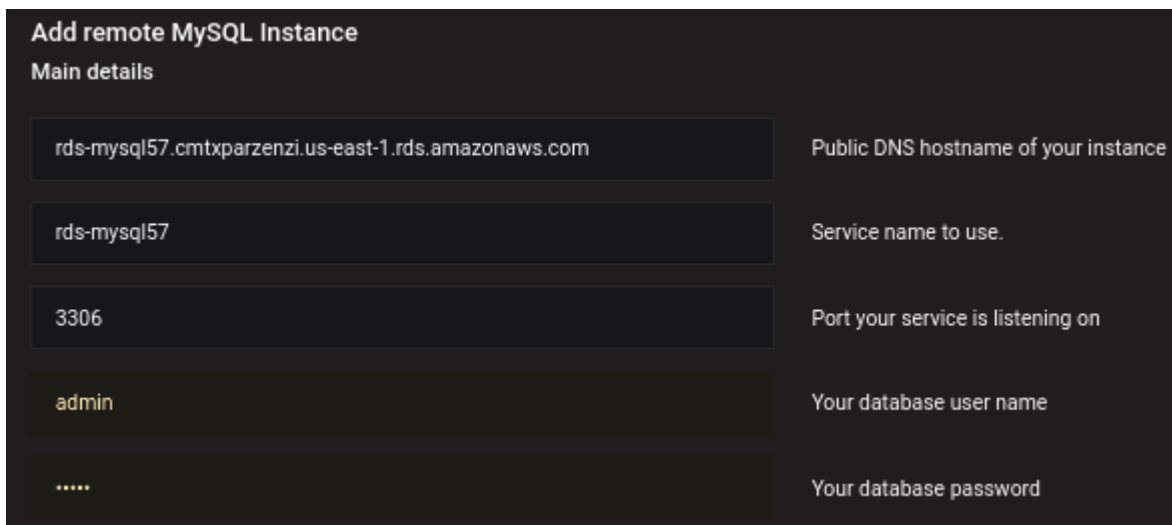


Fig. 21.3: Configuring the selected RDS or Amazon Aurora instance: the *Main details* section

Labels

Environment

us-east-1 Region

us-east-1c Availability Zone

Replication set

Cluster

Custom labels

Format:

key1:value1

key2:value2

Fig. 21.4: Configuring the selected RDS or Amazon Aurora instance: the *Labels* section

The *Labels* section allows specifying labels for the environment, the AWS region and availability zone to be used, the Replication set and Cluster names and also it allows to set the list of custom labels in a key:value format.

Additional options

Skip connection check

Use TLS for database connections

Skip TLS certificate and hostname validation

Use performance schema

Disable Basic Metrics

Disable Enhanced Metrics

Fig. 21.5: Configuring the selected RDS or Amazon Aurora instance: the *Additional options* section for the remote MySQL database

The *Additional options* section contains specific flags which allow to tune the RDS monitoring. They can allow you to skip connection check, to use TLS for the database connection, not to validate the TLS certificate and the hostname, as well as to disable basic and/or enhanced metrics collection for the RDS instance to reduce costs.

We should allow users to disable basic and/or enhanced metrics when RDS instance is added.

Also this section contains a database-specific flag, which would allow Query Analytics for the selected remote database:

- when adding some remote MySQL, AWS RDS MySQL or Aurora MySQL instance, you will be able to choose using performance schema for the database monitoring

- when adding a PostgreSQL instance, you will be able to activate using `pg_stat_statements` extension
- when adding a MongoDB instance, you will be able to choose using QAN MongoDB profiler

Finally press the *Add service* button to start monitoring your instance.

See also:

AWS Documentation: Managing access keys of IAM users https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html

Part VII

Using PMM Metrics Monitor

UNDERSTANDING DASHBOARDS

The Metrics Monitor tool provides a historical view of metrics that are critical to a database server. Time-based graphs are separated into dashboards by themes: some are related to MySQL or MongoDB, others provide general system metrics.

22.1 Opening a Dashboard

The default PMM installation provides more than thirty dashboards. To make it easier to reach a specific dashboard, the system offers two tools. The *Dashboard Dropdown* is a button in the header of any PMM page. It lists all dashboards, organized into folders. Right sub-panel allows to rearrange things, creating new folders and dragging dashboards into them. Also a text box on the top allows to search the required dashboard by typing.

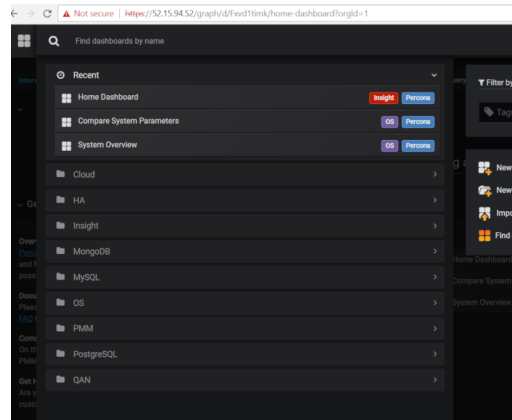


Fig. 22.1: With *Dashboard Dropdown*, search the alphabetical list for any dashboard.

22.2 Viewing More Information about a Graph

Each graph has a descriptions to display more information about the monitored data without cluttering the interface.

These are on-demand descriptions in the tooltip format that you can find by hovering the mouse pointer over the *More Information* icon at the top left corner of a graph. When you move the mouse pointer away from the *More Information* button the description disappears.

See also:

More information about the time range selector *Selecting time or date range*



Fig. 22.2: Graph descriptions provide more information about a graph without claiming any space in the interface.

NAVIGATING ACROSS DASHBOARDS

Beside the *Dashboard Dropdown* button you can also Navigate across Dashboards with the navigation menu which groups dashboards by application. Click the required group and then select the dashboard that matches your choice.

Group	Dashboards for monitoring ...
<i>PMM Query Analytics</i>	QAN component (see <i>Introduction</i>)
OS	The operating system status
MySQL	MySQL and Amazon Aurora
MongoDB	State of MongoDB hosts
HA	High availability
Cloud	Amazon RDS and Amazon Aurora
Insight	Summary, cross-server and Prometheus
PMM	Server settings

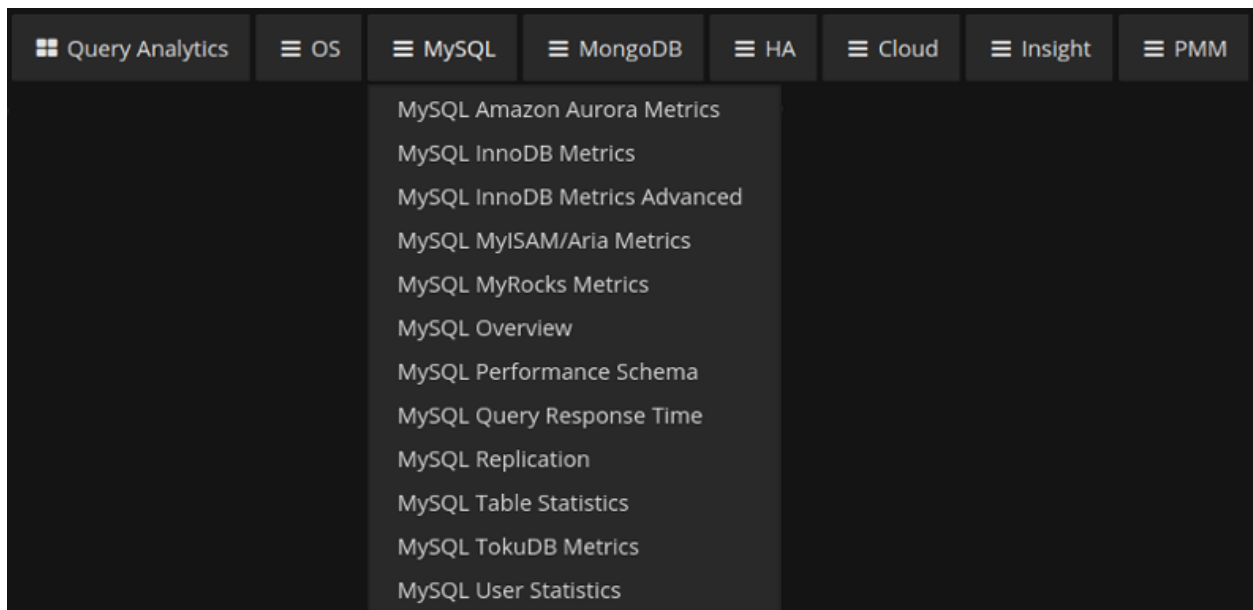


Fig. 23.1: MySQL group selected in the navigation menu

23.1 Zooming in on a single metric

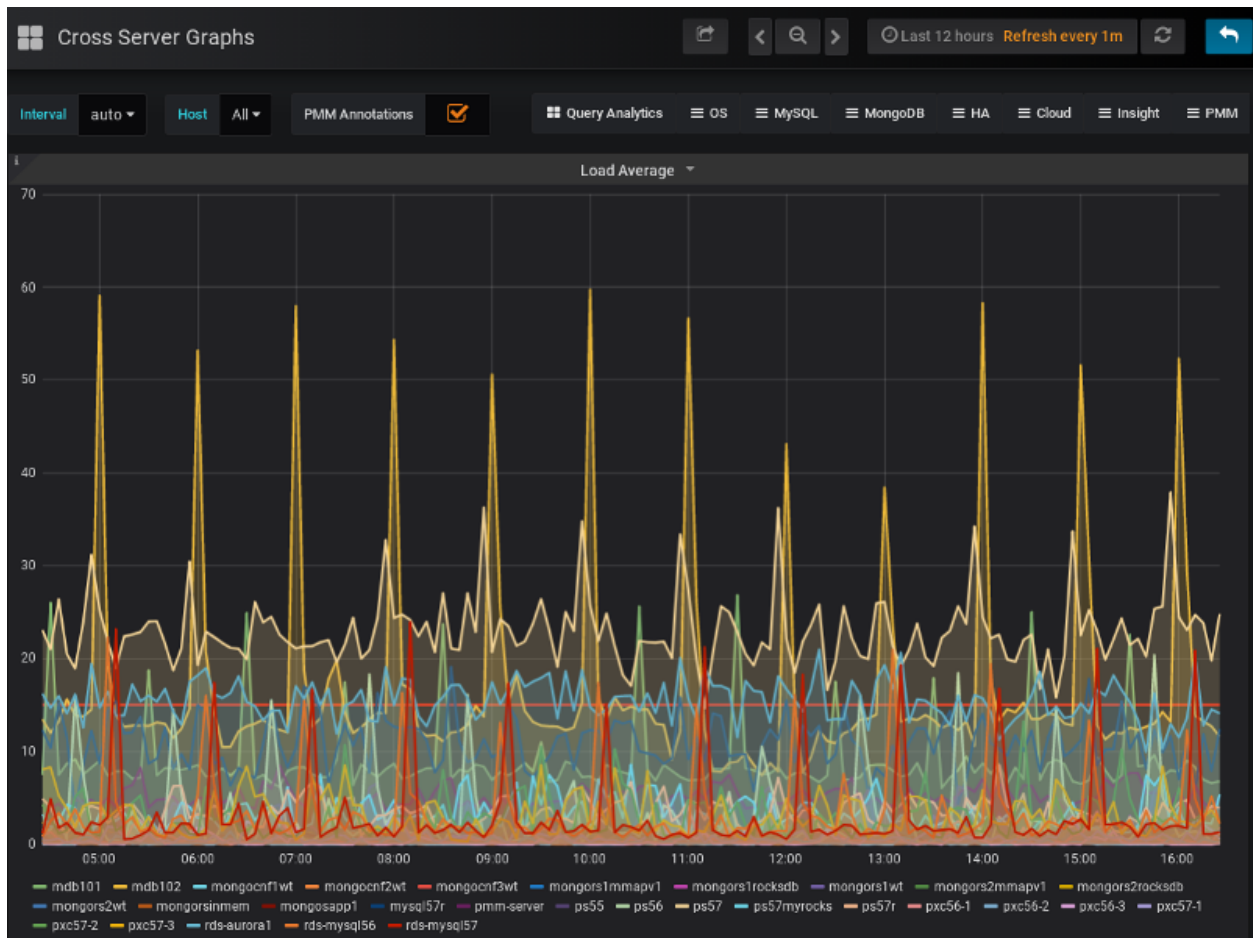
On dashboards with multiple metrics, it is hard to see how the value of a single metric changes over time. Use the context menu to zoom in on the selected metric so that it temporarily occupies the whole dashboard space.

Click the title of the metric that you are interested in and select the *View* option from the context menu that opens.



Fig. 23.2: The context menu of a metric

The selected metric opens to occupy the whole dashboard space. You may now set another time range using the time and date range selector at the top of the Metrics Monitor page and analyze the metric data further.



To return to the dashboard, click the *Back to dashboard* button next to the time range selector.

Navigation menu allows you to navigate between dashboards while maintaining the same host under observation and/or the same selected time range, so that for example you can start on *MySQL Overview* looking at host serverA,

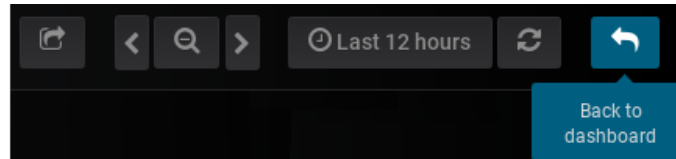


Fig. 23.3: The *Back to dashboard* button returns to the dashboard; this button appears when you are zooming in on one metric.

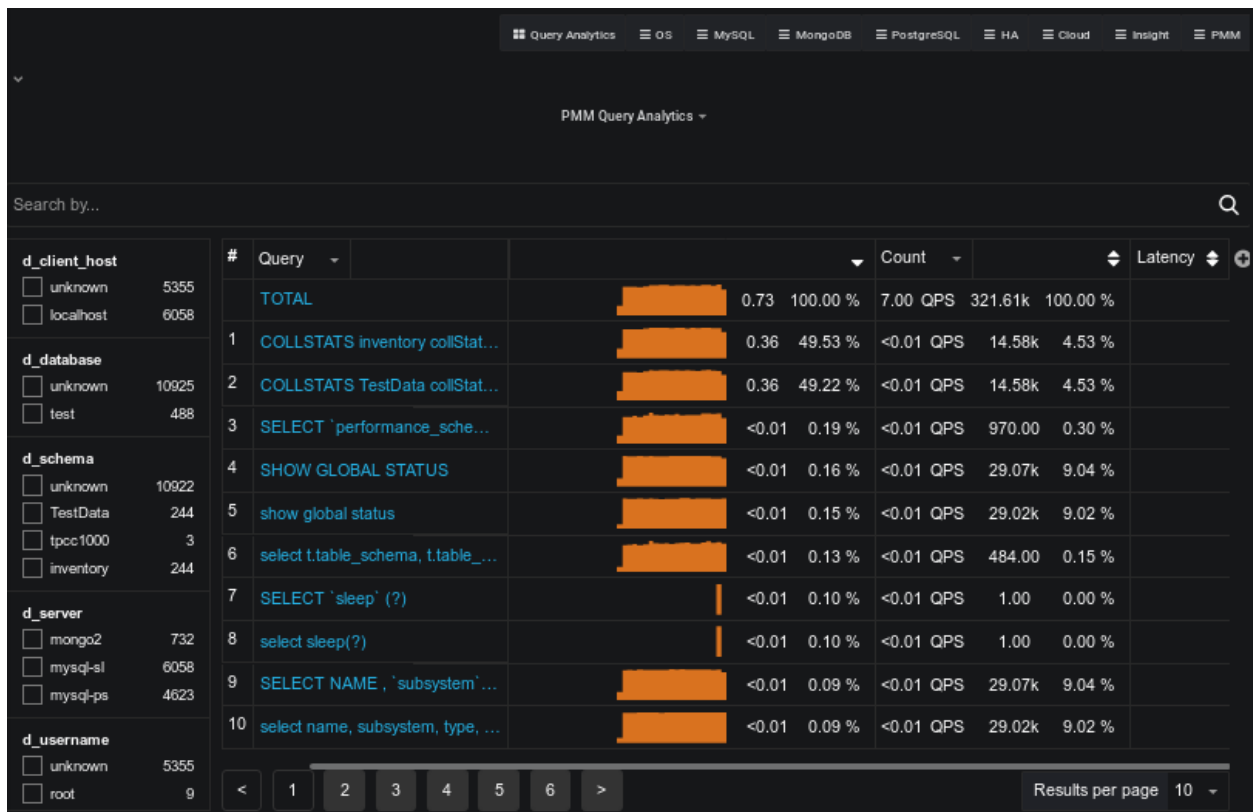
switch to MySQL InnoDB Advanced dashboard and continue looking at serverA, thus saving you a few clicks in the interface.

Part VIII

Using PMM Query Analytics

INTRODUCTION

The QAN is a special dashboard which enables database administrators and application developers to analyze database queries over periods of time and find performance problems. QAN helps you optimize database performance by making sure that queries are executed as expected and within the shortest time possible. In case of problems, you can see which queries may be the cause and get detailed metrics for them.



The screenshot shows the PMM Query Analytics dashboard. At the top, there are navigation tabs for Query Analytics, OS, MySQL, MongoDB, PostgreSQL, HA, Cloud, Insight, and PMM. Below the navigation is a search bar labeled "Search by...". The main content is a table with the following columns: #, Query, Count, and Latency. The table is filtered by various criteria on the left side, including d_client_host, d_database, d_schema, d_server, and d_username. The table contains 10 rows of data, including a total row and several specific queries.

#	Query	Count	Latency
TOTAL		7.00 QPS 321.61k 100.00 %	
1	COLLSTATS inventory collStat...	<0.01 QPS 14.58k 4.53 %	
2	COLLSTATS TestData collStat...	<0.01 QPS 14.58k 4.53 %	
3	SELECT `performance_sche...	<0.01 QPS 970.00 0.30 %	
4	SHOW GLOBAL STATUS	<0.01 QPS 29.07k 9.04 %	
5	show global status	<0.01 QPS 29.02k 9.02 %	
6	select t.table_schema, t.table_...	<0.01 QPS 484.00 0.15 %	
7	SELECT `sleep` (?)	<0.01 QPS 1.00 0.00 %	
8	select sleep(?)	<0.01 QPS 1.00 0.00 %	
9	SELECT NAME , `subsystem` ...	<0.01 QPS 29.07k 9.04 %	
10	select name, subsystem, type, ...	<0.01 QPS 29.02k 9.02 %	

Fig. 24.1: QAN helps analyze database queries over periods of time and find performance problems.

Important: *PMM Query Analytics* supports MySQL and MongoDB. The minimum requirements for MySQL are:

- MySQL 5.1 or later (if using the slow query log)
 - MySQL 5.6.9 or later (if using Performance Schema)
-

QAN displays its metrics in both visual and numeric form: the performance related characteristics appear as plotted graphics with summaries.

NAVIGATING TO QUERY ANALYTICS

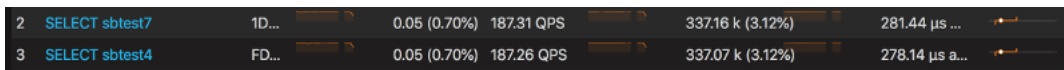
To start working with QAN, choose the *Query analytics*, which is the very left item of the system menu on the top. The QAN dashboard will show up several panels: a search panel, followed by a filter panel on the left, and a panel with the list of queries in a summary table. The columns on this panel are highly customizable, and by default, it displays *Query* column, followed by few essential metrics, such as *Load*, *Count*, and *Latency*.

The screenshot shows the PMM Query Analytics dashboard. At the top, there is a navigation bar with options: Query Analytics, OS, MySQL, MongoDB, PostgreSQL, HA, Cloud, Insight, and PMM. Below this is a search bar labeled "Search by...". The main content is a table with the following columns: #, Query, Count, and Latency. The table is filtered by several criteria: d_client_host (unknown, localhost), d_database (unknown, test), d_schema (unknown, TestData, tpcc1000, inventory), d_server (mongo2, mysql-sl, mysql-ps), and d_username (unknown, root). The table shows 10 rows of query data, including a "TOTAL" row and various queries like "COLLSTATS inventory collStat...", "SELECT `performance_sche...", "SHOW GLOBAL STATUS", "show global status", "select t.table_schema, t.table_...", "SELECT `sleep` (?)", "select sleep(?)", "SELECT NAME , `subsystem`...", and "select name, subsystem, type, ...". The table also includes a pagination bar at the bottom with page numbers 1 through 6 and a "Results per page" dropdown set to 10.

#	Query	Count	Latency
TOTAL		7.00 QPS 321.61k 100.00 %	
1	COLLSTATS inventory collStat...	<0.01 QPS 14.58k 4.53 %	
2	COLLSTATS TestData collStat...	<0.01 QPS 14.58k 4.53 %	
3	SELECT `performance_sche...	<0.01 QPS 970.00 0.30 %	
4	SHOW GLOBAL STATUS	<0.01 QPS 29.07k 9.04 %	
5	show global status	<0.01 QPS 29.02k 9.02 %	
6	select t.table_schema, t.table_...	<0.01 QPS 484.00 0.15 %	
7	SELECT `sleep` (?)	<0.01 QPS 1.00 0.00 %	
8	select sleep(?)	<0.01 QPS 1.00 0.00 %	
9	SELECT NAME , `subsystem`...	<0.01 QPS 29.07k 9.04 %	
10	select name, subsystem, type, ...	<0.01 QPS 29.02k 9.02 %	

Fig. 25.1: The query summary table.

Also it worth to mention that QAN data come in with typical 1-2 min delay, though it is possible to be delayed more because of specific network condition and state of the monitored object. In such situations QAN reports “no data” situation, using sparkline to and showing a gap in place of the time interval, for which data are not available yet.



The screenshot shows a table with two rows of query execution data. The first row (ID 2) shows a query 'SELECT sbtest7' with a duration of 0.05 (0.70%), 187.31 QPS, 337.16 k (3.12%), and 281.44 μs. The second row (ID 3) shows a query 'SELECT sbtest4' with a duration of 0.05 (0.70%), 187.26 QPS, 337.07 k (3.12%), and 278.14 μs. In both rows, the columns for '1D...' and 'FD...' are empty, indicating that data for these intervals is unavailable.

2	SELECT sbtest7	1D...	0.05 (0.70%)	187.31 QPS	337.16 k (3.12%)	281.44 μs ...
3	SELECT sbtest4	FD...	0.05 (0.70%)	187.26 QPS	337.07 k (3.12%)	278.14 μs a...

Fig. 25.2: Showing intervals for which data are unavailable yet.

UNDERSTANDING TOP 10

By default, QAN shows the top *ten* queries. You can sort queries by any column - just click the small arrow to the right of the column name. Also you can add a column for each additional field which is exposed by the data source by clicking the + sign on the right edge of the header and typing or selecting from the available list of fields.

#	Query	Load	Query Count
	TOTAL	<0.01 load 100 %	2.50 QPS 26.95k 100 %
1	SELECT * FROM pg_stat_bgwriter	<0.01 load 36.66 %	0.30 QPS 3.26k 12.1 %
2	SELECT * FROM pg_stat_datab...	<0.01 load 25.65 %	0.30 QPS 3.26k 12.1 %
3	SELECT name, setting, COALES...	<0.01 load 21.38 %	0.30 QPS 3.26k 12.1 %
4	SELECT * FROM pg_stat_datab...	<0.01 load 9.9 %	0.30 QPS 3.26k 12.1 %
5	SELECT pg_database.datname, ...	<0.01 load 2.67 %	0.30 QPS 3.26k 12.1 %
6	SELECT pg_database.datname, ...	<0.01 load 2.42 %	0.30 QPS 3.26k 12.1 %
7	SELECT *, (case pg_is_in_recov...	<0.01 load 0.82 %	0.30 QPS 3.26k 12.1 %
8	SELECT /* pmm-agent:pgstatst...	<0.01 load 0.29 %	0.01 QPS 136.00 0.5 %
9	SELECT version()	<0.01 load 0.12 %	0.30 QPS 3.26k 12.1 %
10	SELECT "agents"."agent_id", "a...	<0.01 load 0.02 %	0.01 QPS 135.00 0.5 %

< 1 2 3 > 24 distinct queries Results per page 10

Fig. 26.1: The query summary table shows the monitored queries from the selected database.

To view more queries, use buttons below the query summary table.

26.1 Query Detail Section

In addition to the metrics in the *query metrics summary table*, QAN displays more information about the query itself below the table.

Tables tab for the selected query seen in the same section contains the table definition and indexes for each table referenced by the query:



Fig. 26.2: The Query Detail section shows the SQL statement for the selected query.

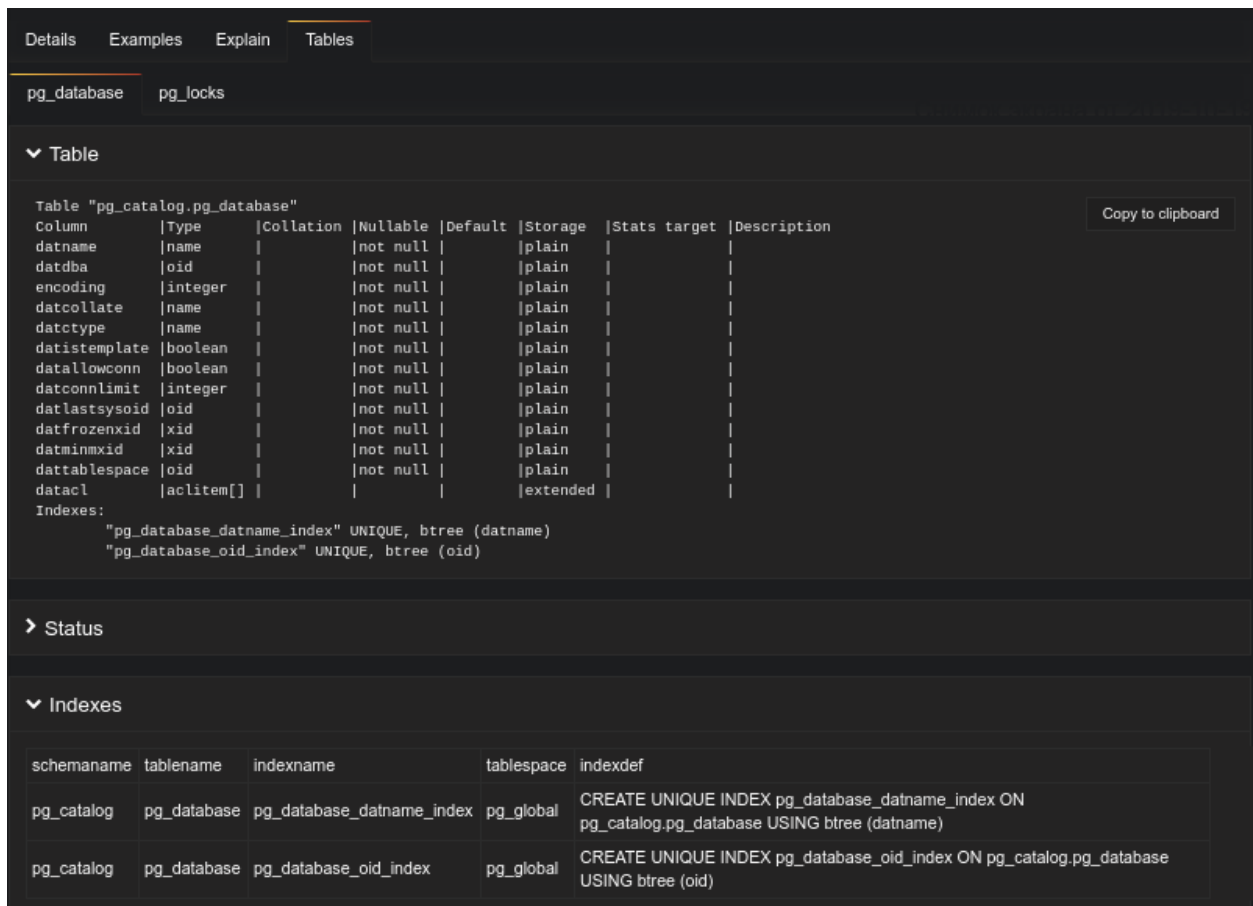
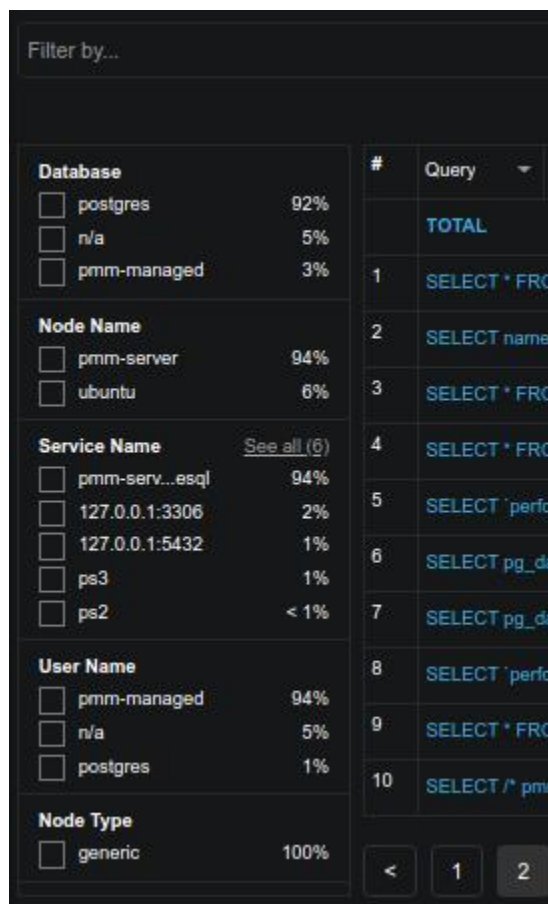


Fig. 26.3: Tables and indexes details the selected query.

FILTERING QUERIES

If you need to limit the list of available queries to only those that you are interested in, use the filtering panel on the left, or use the filter by bar to set filters using *key:value* syntax.



Following example shows how to filter just the queries that are executed on the *Ubuntu* node. Note that Labels unavailable within the set of currently applied filters are shown as gray/disabled on the filtering panel, and the percentage values are dynamically recalculated to reflect the current selection.

By default Filter panel shows top 5 Labels in each section. The additional “See all” link with the total number of Labels appears if their amount exceeds five. Clicking this link toggles it into “Show top 5” and reveals all Labels available in the section.

Node Name: ubuntu ✕

Database	#	Query
<input type="checkbox"/> n/a 80%		
<input type="checkbox"/> postgres 20%		
<input type="checkbox"/> pmm-managed		
Node Name		
<input type="checkbox"/> pmm-server 75%	1	TOTAL
<input checked="" type="checkbox"/> ubuntu 25%	2	SELECT 'perfor
Service Name See all (9)		
<input type="checkbox"/> 127.0.0.1:3306 21%	3	SELECT * FRO
<input type="checkbox"/> ps2 20%	4	SELECT * FRO
<input type="checkbox"/> ps3 20%	5	SELECT /* pmm
<input type="checkbox"/> ps4 19%	6	SELECT name.
<input type="checkbox"/> 127.0.0.1:5432 8%	7	SELECT pg_da
User Name		
<input type="checkbox"/> n/a 80%	8	SELECT /* pmm
<input type="checkbox"/> postgres 20%	9	SELECT pg_da
<input type="checkbox"/> pmm-managed	10	SELECT * FRO
Node Type		
<input type="checkbox"/> generic 100%		

< 1 2

Selecting Time or Date Range

The query metrics that appear in QAN are computed based on a time period or a range of dates. The default value is *the last hour*. To set another range use the *range selection tool* located at the top of the QAN page.

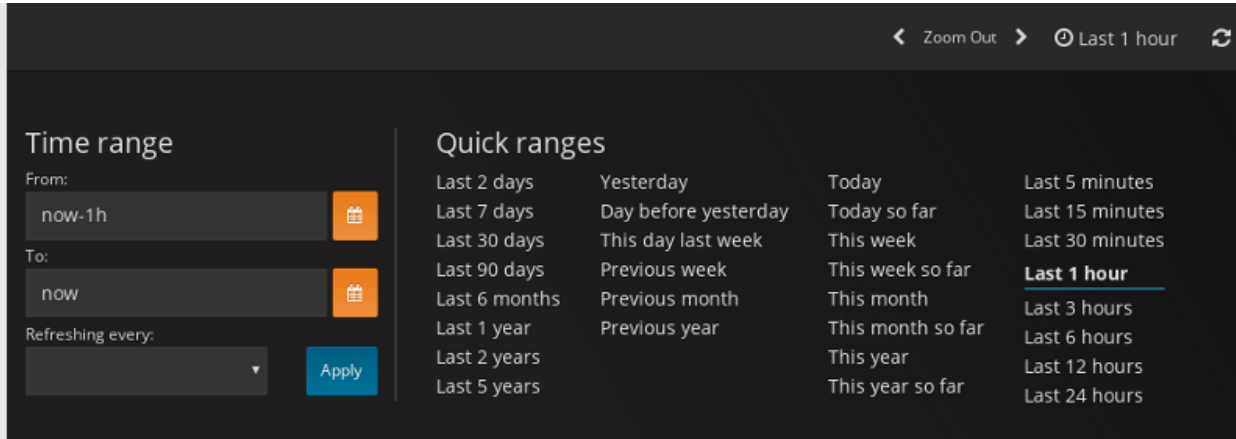


Fig. 27.1: QAN displays query metrics for the time period or date range that you specify.

The tool consists of two parts. The *Quick ranges* offers frequently used time ranges.. The date picker sets a range of dates.

27.1 Totals of the Query Summary

The first line of the query summary contains the totals of the *load*, *count*, and *latency* for all queries that were run on the selected database server during the time period that you've specified.

The *load* is the amount of time that the database server spent during the selected time or date range running all queries.

The *count* is the average number of requests to the server during the specified time or date range.

The *latency* is the average amount of time that it took the database server to retrieve and return the data.

27.2 Queries in the Query Summary Table

Each row in the query summary contains information about a single query. Each column is query attribute. The *Query* attribute informs the type of query, such as INSERT, or UPDATE, and the queried tables, or collections. The *ID* attribute is a unique hexadecimal number associated with the given query.

The *Load*, *Count*, and *Latency* attributes refer to the essential metrics of each query. Their values are plotted graphics and summary values in the numeric form. The summary values have two parts. The average value of the metric and its percentage with respect to the corresponding total value at the top of the query summary table.

27.3 Viewing a Specific Value of a Metric

If you hover the cursor over one of the metrics in a query, you can see a concrete value at the point where your cursor is located. Move the cursor along the plotted line to watch how the value is changing.



Fig. 27.2: Hover the cursor to see a value at the point.

MONGODB SPECIFIC

28.1 Query Analytics for MongoDB

MongoDB is conceptually different from relational database management systems, such as MySQL or MariaDB. Relational database management systems store data in tables that represent single entities. In order to represent complex objects you may need to link records from multiple tables. MongoDB, on the other hand, uses the concept of a document where all essential information pertaining to a complex object is stored together.

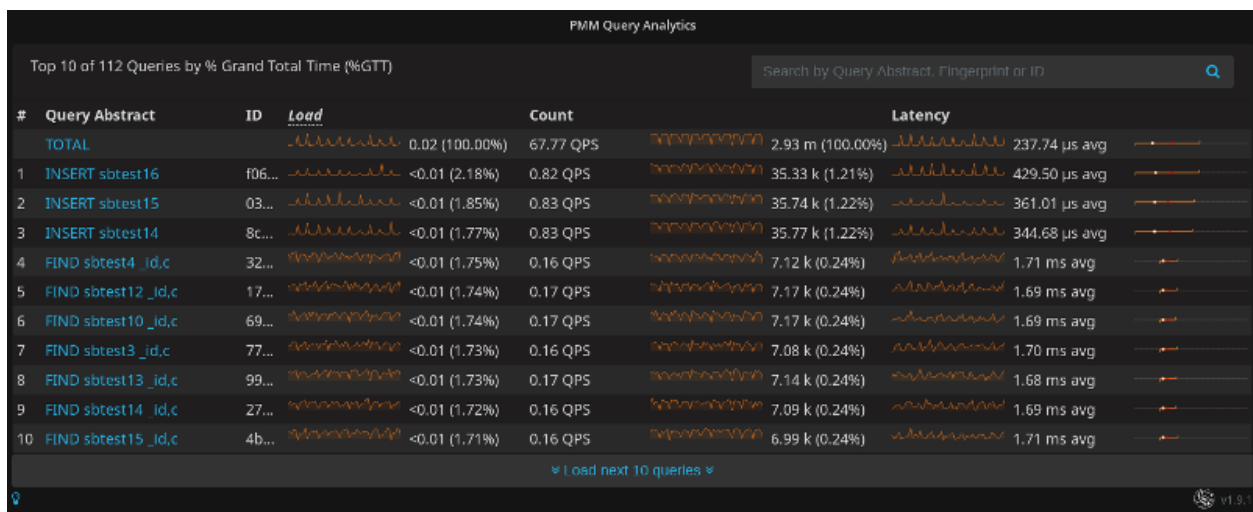


Fig. 28.1: A list of queries from a MongoDB host

QAN supports monitoring MongoDB queries. Although MongoDB is not a relational database management system, you analyze its databases and collections in the same interface using the same tools. By using the familiar and intuitive interface of QAN you can analyze the efficiency of your application reading and writing data in the collections of your MongoDB databases.

See also:

What MongoDB versions are supported by QAN? *See more information about how to configure MongoDB*

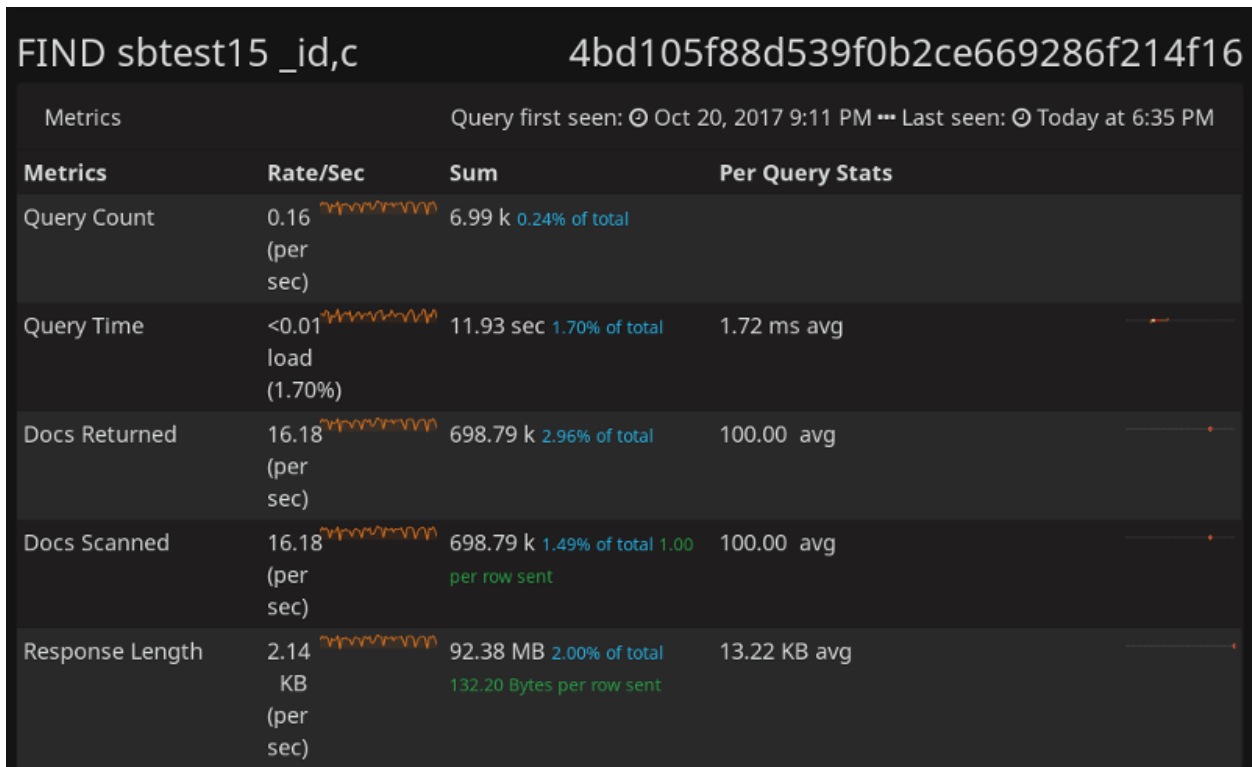


Fig. 28.2: Analyze MongoDB queries using the same tools as relational database management systems.

Part IX

Metrics Monitor Dashboards

This section contains a reference of dashboards available in Metrics Monitor.

29.1 Home Dashboard

The *Home* dashboard is a high level overview of your environment.

See also:

Overview of PMM using

29.1.1 General Information

This section contains links to online resources, such as PMM documentation, releases notes, and blogs.

29.1.2 Shared and Recently Used Dashboards

This section is automatically updated to show the most recent dashboards that you worked with. It also contains the dashboards that you have bookmarked.

29.1.3 Statistics

This section shows the total number of hosts added to PMM and the total number of database instanced being monitored. This section also current the version number. Use the *Check for Updates Manually* button to see if you are using the most recent version of PMM.

29.1.4 Environment Overview

This section lists all added hosts along with essential information about their performance. For each host, you can find the current values of the following metrics:

- CPU Busy
- Memory Available
- Disk Reads
- Disk Writes
- Network IO
- DB Connections
- DB QPS
- Virtual CPUs
- RAM
- Host Uptime

- DB Uptime

29.2 PMM System Summary Dashboard

The *PMM System Summary* dashboard shows detailed information about the selected host (the value of the *Host* field) and the database server deployed on this host.

The *System Summary* section contains details about the platform while the *Database Summary* offers detailed statistics about the database server.

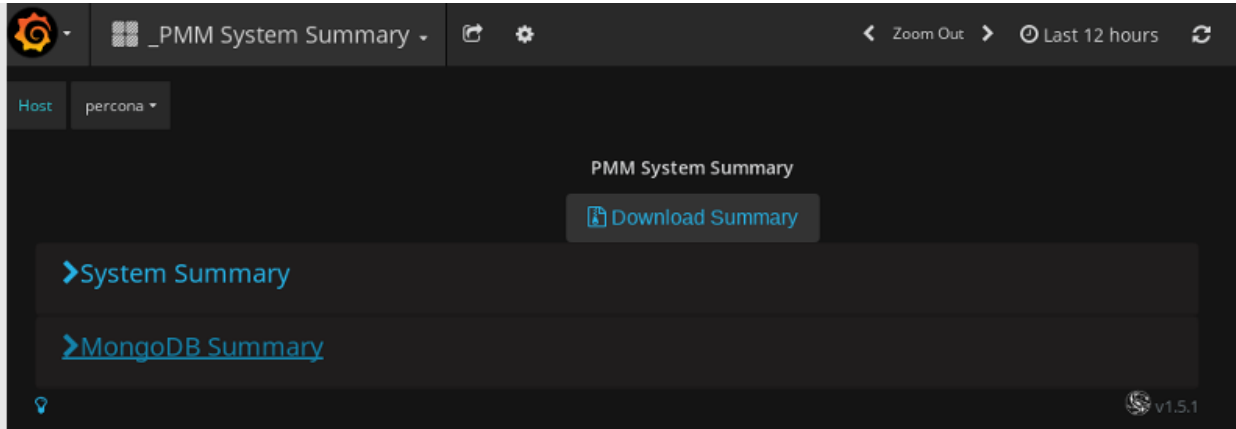


Fig. 29.1: Accessing information about the system and database

You can download the current values on this dashboard locally if you click the *Download Summary* button.

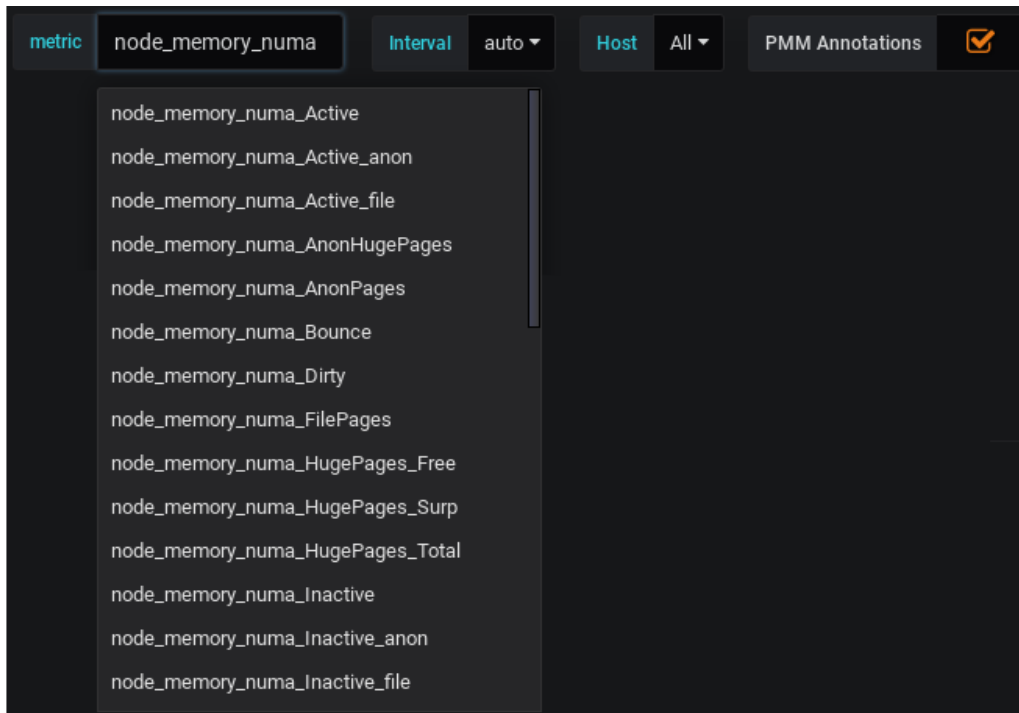
29.3 Advanced Data Exploration Dashboard

The *Advanced Data Exploration* dashboard provides detailed information about the progress of a single Prometheus metric across one or more hosts.

Added NUMA related metrics

New in version 1.13.0.

The *Advanced Data Exploration* dashboard supports metrics related to NUMA. The names of all these metrics start with *node_memory_numa*.



- *View actual metric values (Gauge)*
- *View actual metric values (Counters)*
- *View actual metric values (Counters)*

29.3.1 View actual metric values (Gauge)

In this section, the values of the selected metric may increase or decrease over time (similar to temperature or memory usage).

29.3.2 View actual metric values (Counters)

In this section, the values of the selected metric are accumulated over time (useful to count the number of served requests, for example).

29.3.3 View actual metric values (Counters)

This section presents the values of the selected metric in the tabular form.

See also:

Prometheus Documentation: Metric types https://prometheus.io/docs/concepts/metric_types/

29.4 Cross Server Graphs

- *Load Average*
- *MySQL Queries*
- *MySQL Traffic*

29.4.1 Load Average

This metric is the average number of processes that are either in a runnable or uninterruptable state. A process in a runnable state is either using the CPU or waiting to use the CPU. A process in uninterruptable state is waiting for some I/O access, eg waiting for disk.

This metric is best used for trends. If you notice the load average rising, it may be due to inefficient queries. In that case, you may further analyze your queries in QAN.

View all metrics of *Cross Server Graphs*

See also:

Description of *load average* in the man page of the `uptime` command in Debian <https://manpages.debian.org/stretch/procps/uptime.1.en.html>

29.4.2 MySQL Queries

This metric is based on the queries reported by the MySQL command `SHOW STATUS`. It shows the average number of statements executed by the server. This variable includes statements executed within stored programs, unlike the `Questions` variable. It does not count `COM_PING` or `COM_STATISTICS` commands.

View all metrics of *Cross Server Graphs*

See also:

MySQL Server Status Variables: Queries https://dev.mysql.com/doc/refman/5.6/en/server-status-variables.html#statvar_Queries

29.4.3 MySQL Traffic

This metric shows the network traffic used by the MySQL process.

View all metrics of *Cross Server Graphs*

29.5 Summary Dashboard

- *CPU Usage*
- *Processes*
- *Network Traffic*

- *I/O Activity*
- *Disk Latency*
- *MySQL Queries*
- *InnoDB Row Operations*
- *Top MySQL Commands*
- *Top MySQL Handlers*

29.5.1 CPU Usage

The CPU Usage graph shows how much of the overall CPU time is used by the server. It has 4 components: system, user, iowait and softirq.

System The proportion of time the CPU spent inside the Linux kernel for operations like context switching, memory allocation and queue handling.

User The time spent in the user space. Normally, most of the MySQL CPU time is in user space, a too high value may indicate an indexing issue.

Iowait The time the CPU spent waiting for disk IO requests to complete. A high value of iowait indicates a disk bound load.

Softirq The portion of time the CPU spent servicing software interrupts generated by the device drivers. A high value of *softirq* may indicate a poorly configured device. The network is generally the main source of high softirq values. Be aware the graph presents global values, while there may be a lot of unused CPU, a single core may be saturated. Look for any quantity saturating at 100/(cpu core count).

View all metrics of [Summary Dashboard](#)

See also:

Linux CPU Statistics

<http://blog.scoutapp.com/articles/2015/02/24/understanding-linuxs-cpu-stats>

29.5.2 Processes

The Processes metric shows how many processes/threads are either in the kernel run queue (runnable state) or in the blocked queue (waiting for I/O).

When the number of process in the runnable state is constantly higher than the number of CPU cores available, the load is CPU bound.

When the number of process blocked waiting for I/O is large, the load is disk bound.

The running average of the sum of these two quantities is the basis of the loadavg metric.

View all metrics of [Summary Dashboard](#)

See also:

More information about Vmstat <http://nonfunctionaltestingtools.blogspot.ca/2013/03/vmstat-output-explained.html>

29.5.3 Network Traffic

The Network Traffic graph shows the rate of data transferred over the network. Outbound is the data sent by the server while Inbound is the data received by the server.

Look for signs of saturation given the capacity of the network devices. If the outbound rate is consistently high and close to saturation and you have plenty of available CPU, you should consider activating the compression option on the MySQL clients and slaves.

View all metrics of *Summary Dashboard*

29.5.4 I/O Activity

The I/O Activity graph shows the rates of data read from (Page In) and written to (Page Out) the all the disks as collected from the vmstat bi and bo columns.

View all metrics of *Summary Dashboard*

29.5.5 Disk Latency

The Disk Latency graph shows the average time to complete read and write operations to the disks.

There is one data series per operation type (Read or Write) per disk mounted to the server.

High latency values, typically more than 15 ms, are an indication of a disk bound workload saturating the storage subsystem or, a faulty/degraded hardware.

View all metrics of *Summary Dashboard*

29.5.6 MySQL Queries

The MySQL Queries graph shows the rate of queries processed by MySQL. The rate of queries is a rough indication of the MySQL Server load.

View all metrics of *Summary Dashboard*

29.5.7 InnoDB Row Operations

The InnoDB Row Operations graph shows the rate of rows processed by InnoDB. It is a good indication of the MySQL Server load. A high value of Rows read, which can easily be above a million, is an indication of poor queries or deficient indexing.

The amounts of rows inserted, updated and deleted help appreciate the server write load.

View all metrics of *Summary Dashboard*

29.5.8 Top MySQL Commands

The Top MySQL Commands graph shows the rate of the various kind of SQL statements executed on the MySQL Server.

View all metrics of *Summary Dashboard*

29.5.9 Top MySQL Handlers

The Top MySQL Handlers graph shows the rate of the various low level storage engine handler calls. The most important ones to watch are *read_next* and *read_rnd_next*.

A high values for *read_rnd_next* is an indication there are table scans while a high value of *read_next* is an indication of index scans.

View all metrics of [Summary Dashboard](#)

29.6 Trends Dashboard

The *Trends* dashboard shows the essential statistics about the selected host. It also includes the essential statistics of MySQL, such as MySQL questions and InnoDB row reads and row changes.

Note: The MySQL statistics section is empty for hosts other than MySQL.

See also:

MySQL Documentation:

[Questions](#)

Metrics

- *CPU Usage*
- *I/O Read Activity*
- *I/O Write Activity*
- *MySQL Questions*
- *InnoDB Rows Read*
- *InnoDB Rows Changed*

29.6.1 CPU Usage

This metric shows the comparison of the percentage of the CPU usage for the current selected range, the previous day and the previous week. This graph is useful to demonstrate how the CPU usage has changed over time by visually overlaying time periods.

View all metrics of [Trends Dashboard](#)

29.6.2 I/O Read Activity

This metric shows the comparison of I/O Read Activity in terms of bytes read for the current selected range versus the previous day and the previous week for the same time range. This graph is useful to demonstrate how I/O Read Activity has changed over time by visually overlaying time periods.

View all metrics of [Trends Dashboard](#)

29.6.3 I/O Write Activity

Shows the comparison of I/O Write Activity in terms of byte written for the current selected range versus the previous day and the previous week for the same time range. This graph is useful to demonstrate how I/O Write Activity has changed over time by visually overlaying time periods.

View all metrics of *Trends Dashboard*

29.6.4 MySQL Questions

This metric shows the comparison of the MySQL Questions for the current selected range versus the previous day and the previous week for the same time range. This graph is useful to demonstrate how MySQL Questions has changed over time by visually overlaying time periods.

View all metrics of *Trends Dashboard*

29.6.5 InnoDB Rows Read

This metric shows the comparison of the InnoDB Rows Read for the current selected range versus the previous day and the previous week for the same time range. This graph is useful to demonstrate how InnoDB Rows Read has changed over time by visually overlaying time periods.

View all metrics of *Trends Dashboard*

29.6.6 InnoDB Rows Changed

This metric shows the comparison of InnoDB Rows Changed for the current selected range versus the previous day and the previous week for the same time range. This graph is useful to demonstrate how the InnoDB Rows Changed has fluctuated over time by visually overlaying time periods.

View all metrics of *Trends Dashboard*

29.7 Network Overview Dashboard

The information in the *Network Overview* dashboard is grouped into the following sections:

- *Last Hour Statistic*
- *Network Traffic*
- *Network Traffic Details*
- *Network Netstat TCP*
- *Network Netstat UDP*
- *ICMP*

29.7.1 Last Hour Statistic

This section reports the *inbound speed*, *outbound speed*, *traffic errors and drops*, and *retransmit rate*.

View all metrics of [Network Overview Dashboard](#)

29.7.2 Network Traffic

This section contains the *Network traffic* and *network utilization hourly* metrics.

View all metrics of [Network Overview Dashboard](#)

29.7.3 Network Traffic Details

This section offers the following metrics:

- Network traffic by packets
- Network traffic errors
- Network traffic drop
- Network traffic multicust

View all metrics of [Network Overview Dashboard](#)

29.7.4 Network Netstat TCP

This section offers the following metrics:

- Timeout value used for retransmitting
- Min TCP retransmission timeout
- Max TCP retransmission timeout
- Netstat: TCP
- TCP segments

View all metrics of [Network Overview Dashboard](#)

29.7.5 Network Netstat UDP

In this section, you can find the following metrics:

- Netstat: UDP
- UDP Lite

The graphs in the *UDP Lite* metric give statistics about:

InDatagrams Packets received

OutDatagrams Packets sent

InCsumErrors Datagrams with checksum errors

InErrors Datagrams that could not be delivered to an application

RcvbufErrors Datagrams for which not enough socket buffer memory to receive

SndbufErrors Datagrams for which not enough socket buffer memory to transmit

NoPorts Datagrams received on a port with no listener

View all metrics of [Network Overview Dashboard](#)

29.7.6 ICMP

This section has the following metrics:

- ICMP Errors
- Messages/Redirects
- Echos
- Timestamps/Mask Requests

ICMP Errors

InErrors Messages which the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, etc.)

OutErrors Messages which this entity did not send due to problems discovered within ICMP, such as a lack of buffers

InDestUnreachs Destination Unreachable messages received

OutDestUnreachs Destination Unreachable messages sent

InType3 Destination unreachable

OutType3 Destination unreachable

InCsumErrors Messages with ICMP checksum errors

InTimeExcds Time Exceeded messages received

Messages/Redirects

InMsgs Messages which the entity received. Note that this counter includes all those counted by icmpInErrors

InRedirects Redirect messages received

OutMsgs Messages which this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors

OutRedirects Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects

Echos

InEchoReps Echo Reply messages received

InEchos Echo (request) messages received

OutEchoReps Echo Reply messages sent

OutEchos Echo (request) messages sent

Timestamps/Mask Requests

InAddrMaskReps Address Mask Reply messages received

InAddrMasks Address Mask Request messages received

OutAddrMaskReps Address Mask Reply messages sent

OutAddrMasks Address Mask Request messages sent

InTimestampReps Timestamp Reply messages received

InTimestamps Timestamp Request messages received

OutTimestampReps Timestamp Reply messages sent

OutTimestamps Timestamp Request messages sent

View all metrics of [Network Overview Dashboard](#)

29.8 Inventory Dashboard

The *Inventory* dashboard is a high level overview of all objects PMM “knows” about.

It contains three tabs (*services*, *agents*, and *nodes*) with lists of the correspondent objects and details about them, so that users are better able to understand which objects are registered against PMM Server. These objects are composing a hierarchy with Node at the top, then Service and Agents assigned to a Node.

- **Nodes** – Where the service and agents will run. Assigned a `node_id`, associated with a `machine_id` (from `/etc/machine-id`). Few examples are bare metal, virtualized, container.
- **Services** – Individual service names and where they run, against which agents will be assigned. Each instance of a service gets a `service_id` value that is related to a `node_id`. Examples are MySQL, Amazon Aurora MySQL. This feature also allows to support multiple `mysqld` instances on a single node, with different service names, e.g. `mysql1-3306`, and `mysql1-3307`.
- **Agents** – Each binary (exporter, agent) running on a client will get an `agent_id` value.
 - `pmm-agent` one is the top of the tree, assigned to a `node_id`
 - `node_exporter` is assigned to `pmm-agent agent_id`
 - `mysqld_exporter` & QAN MySQL Perfschema are assigned to a `service_id`.

Examples are `pmm-agent`, `node_exporter`, `mysqld_exporter`, QAN MySQL Perfschema.

OS DASHBOARDS

30.1 CPU Utilization Details (Cores)

- *Overall CPU Utilization*
- *Current CPU Core Utilization*
- *All Cores - Total*

30.1.1 Overall CPU Utilization

The Overall CPU Utilization metric shows how much of the overall CPU time is used by the server. It has 4 components:

- system
- user
- iowait
- softirq

In addition, sampling of the Max utilization of a single core is shown.

System

This component the proportion of time the CPUs spent inside the Linux kernel for operations like context switching, memory allocation and queue handling.

User

This component is the time spent in the user space. Normally, most of the MySQL CPU time is in user space. A high value of user time indicates a CPU bound workload.

Iowait

This component is the time the CPU spent waiting for disk IO requests to complete. A high value of iowait indicates a disk bound load.

Softirq

This component is the portion of time the CPU spent servicing software interrupts generated by the device drivers. A high value of softirq may indicates a poorly configured device. The network devices are generally the main source of high softirq values.

Be aware that this metric presents global values: while there may be a lot of unused CPU, a single core may be saturated. Look at the Max Core Utilization to see if any core is reaching close to 100%.

30.1.2 Current CPU Core Utilization

The Current CPU Core Utilization metric shows the total utilization of each CPU core along with the average utilization of all CPU cores. Watch for any core close to 100% utilization and investigate the root cause.

View all metrics of *CPU Utilization Details (Cores)*

30.1.3 All Cores - Total

The All Cores - total graph shows the average distribution of all the CPU times. It has 5 components:

- system
- user
- iowait
- steal
- other

Components

System

This component is the proportion of time the CPUs spent inside the Linux kernel for operations like context switching, memory allocation and queue handling.

User

This component is the time spent in the user space. Normally, most of the MySQL CPU time is in user space. A high value of user time indicates a CPU bound workload.

Iowait

This component is the time the CPU spent waiting for disk IO requests to complete. A high value of iowait indicates a disk bound load. Steal is non zero only in a virtual environment.

Steal

When multiple virtual machines share the same physical host, some virtual machines may be allowed to use more of their share of CPU and that CPU time is accounted as Steal by the virtual machine from which the time is taken.

Other

This component is essentially the softirq time which is the portion of time the CPU spent servicing software interrupts generated by the device drivers. A high value of softirq may indicate a poorly configured device. The network devices are generally the main source of high softirq values.

Be aware that this metric presents global values: while there may be a lot of unused CPU, a single core may be saturated.

View all metrics of *CPU Utilization Details (Cores)*

30.2 Disk space

- *Mountpoint Usage*
- *Mountpoint*

30.2.1 Mountpoint Usage

This metric shows the percentage of disk space utilization for every mountpoint defined on the system. It is not good having some of the mountpoints close to 100% of space utilization, the risk is to have a *disk full* error that can block one of the services or even causing a crash of the entire system.

In case a mountpoint is close to 100%, consider to cancel unused files or to expand the space allocate to it.

View all metrics of *Disk space*

30.2.2 Mountpoint

This metric shows information about the disk space usage of the specified mountpoint.

Used Is the amount of space used

Free Is the amount if space not in use

The total disk space allocated to the mountpoint is the sum of *Used* and *Free* space.

It is not good having *Free* close to 0 B. The risk is to have a *disk full* error that can block one of the services or even causing a crash of the entire system.

In case *Free* is close to 0 B, consider to cancel unused files or to expand the space allocated to the mountpoint.

View all metrics of *Disk space*

30.3 System Overview Dashboard

The *System Overview* dashboard provides details about the efficiency of work of the following components. Each component is represented as a section in the *System Overview* dashboard.

- CPU
- Memory
- Disk
- Network

The *CPU* section offers the *CPU Usage*, *CPU Saturation and Max Core Usage*, *Interrupts and Context Switches*, and *Processes* metrics.

In the *Memory* section, you can find the *Memory Utilization*, *Virtual Memory Utilization*, *Swap Space*, and *Swap Activity* metrics.

The *Disk* section contains the *I/O Activity*, *Global File Descriptors Usage*, *Disk IO Latency*, and *Disk IO Load* metrics.

In the *Network* section, you can find the *Network Traffic*, *Network Utilization Hourly*, *Local Network Errors**, and *TCP Retransmission* metrics.

30.4 Compare System Parameters Dashboard

The *Compare System Parameters* dashboard allows to compare wide range of parameters of the servers monitored by PMM. Same type parameters are shown side by side for all the servers, grouped in the following sections:

- System Information
- CPU
- Memory
- Disk Partitions
- Disk Performance
- Network

The *System Information* section shows the *System Info* summary of each server, as well as *System Uptime*, *CPU Cores*, *RAM*, *Saturation Metrics*, and *Load Average* gauges.

The *CPU* section offers the *CPU Usage*, *Interrupts*, and *Context Switches* metrics.

In the *Memory* section, you can find the *Memory Usage*, *Swap Usage*, and *Swap Activity* metrics.

The *Disk Partitions* section encapsulates two metrics, *Mountpoint Usage* and *Free Space*.

The *Disk Performance* section contains the *I/O Activity*, *Disk Operations*, *Disk Bandwidth*, *Disk IO Utilization*, *Disk Latency*, and *Disk Load* metrics.

Finally, *Network* section shows *Network Traffic*, and *Network Utilization Hourly* metrics.

30.5 NUMA Overview Dashboard

For each node, this dashboard shows metrics related to Non-uniform memory access (NUMA).

- *Memory Usage*
- *Free Memory Percent*
- *NUMA Memory Usage Types*
- *NUMA Allocation Hits*
- *NUMA Allocation Missed*
- *Anonymous Memory*
- *NUMA File (PageCache)*
- *Shared Memory*
- *HugePages Statistics*
- *Local Processes*
- *Remote Processes*
- *Slab Memory*

30.5.1 Memory Usage

Remotes over time the total, used, and free memory.

View all metrics of [NUMA Overview Dashboard](#)

30.5.2 Free Memory Percent

Shows the free memory as the ratio to the total available memory.

View all metrics of [NUMA Overview Dashboard](#)

30.5.3 NUMA Memory Usage Types

Dirty Memory waiting to be written back to disk

Bounce Memory used for block device bounce buffers

Mapped Files which have been mmaped, such as libraries

KernelStack The memory the kernel stack uses. This is not reclaimable.

View all metrics of [NUMA Overview Dashboard](#)

30.5.4 NUMA Allocation Hits

Memory successfully allocated on this node as intended.

View all metrics of [NUMA Overview Dashboard](#)

30.5.5 NUMA Allocation Missed

Memory missed is allocated on a node despite the process preferring some different node.

Memory foreign is intended for a node, but actually allocated on some different node.

View all metrics of [NUMA Overview Dashboard](#)

30.5.6 Anonymous Memory

Active Anonymous memory that has been used more recently and usually not swapped out.

Inactive Anonymous memory that has not been used recently and can be swapped out.

View all metrics of [NUMA Overview Dashboard](#)

30.5.7 NUMA File (PageCache)

Active(file) Pagecache memory that has been used more recently and usually not reclaimed until needed.

Inactive(file) Pagecache memory that can be reclaimed without huge performance impact.

View all metrics of [NUMA Overview Dashboard](#)

30.5.8 Shared Memory

Shmem Total used shared memory (shared between several processes, thus including RAM disks, SYS-V-IPC and BSD like SHMEM)

View all metrics of [NUMA Overview Dashboard](#)

30.5.9 HugePages Statistics

Total Number of hugepages being allocated by the kernel (Defined with `vm.nr_hugepages`).

Free The number of hugepages not being allocated by a process

Surp The number of hugepages in the pool above the value in `vm.nr_hugepages`. The maximum number of surplus hugepages is controlled by `vm.nr_overcommit_hugepages`.

View all metrics of [NUMA Overview Dashboard](#)

30.5.10 Local Processes

Memory allocated on a node while a process was running on it.

View all metrics of [NUMA Overview Dashboard](#)

30.5.11 Remote Processes

Memory allocated on a node while a process was running on some other node.

View all metrics of [NUMA Overview Dashboard](#)

30.5.12 Slab Memory

Slab Allocation is a memory management mechanism intended for the efficient memory allocation of kernel objects.

SReclaimable The part of the Slab that might be reclaimed (such as caches).

SUnreclaim The part of the Slab that can't be reclaimed under memory pressure

View all metrics of [NUMA Overview Dashboard](#)

PROMETHEUS DASHBOARDS

31.1 *Prometheus* Dashboard

The *Prometheus* dashboard informs how Prometheus functions.

See also:

Overview of PMM using

All dashboards *Metrics Monitor Dashboards*

31.1.1 Prometheus overview

This section shows the most essential parameters of the system where Prometheus is running, such as CPU and memory usage, scrapes performed and the samples ingested in the head block.

31.1.2 Resources

This section provides details about the consumption of CPU and memory by the Prometheus process. This section contains the following metrics:

- Prometheus Process CPU Usage
- Prometheus Process Memory Usage

31.1.3 Storage (TSDB)

This section includes a collection of metrics related to the usage of storage. It includes the following metrics:

- Data blocks (Number of currently loaded data blocks)
- Total chunks in the head block
- Number of series in the head block
- Current retention period of the head block
- Activity with chunks in the head block
- Reload block data from disk

31.1.4 Scraping

This section contains metrics that help monitor the scraping process. This section contains the following metrics:

- Ingestion
- Prometheus Targets

- Scraped Target by Job
- Scrape Time by Job
- Scraped Target by Instance
- Scraped Time by Instance
- Scrapes by Target Frequency
- Scrape Frequency Versus Target
- Scraping Time Drift
- Prometheus Scrape Interval Variance
- Slowest Jobs
- Largest Samples Jobs

31.1.5 Queries

This section contains metrics that monitor Prometheus queries. This section contains the following metrics:

- Prometheus Queries
- Prometheus Query Execution
- Prometheus Query Execution Latency
- Prometheus Query Execution Load

31.1.6 Network

Metrics in this section help detect network problems.

- HTTP Requests by Handler
- HTTP Errors
- HTTP Avg Response time by Handler
- HTTP 99% Percentile Response time by Handler
- HTTP Response Average Size by Handler
- HTTP 99% Percentile Response Size

31.1.7 Time Series Information

This section shows the top 10 metrics by time series count and the top 10 hosts by time series count.

31.1.8 System Level Metrics

Metrics in this section give an overview of the essential system characteristics of PMM Server. This information is also available from the *System Overview* dashboard.

31.1.9 PMM Server Logs

This section contains a link to download the logs collected from your PMM Server and further analyze possible problems. The exported logs are requested when you submit a bug report.

31.2 Prometheus Exporter Status

The Prometheus Exporter Status dashboard reports the consumption of resources by the Prometheus exporters used by PMM. For each exporter, this dashboard reveals the following information:

- CPU usage
- Memory usage
- File descriptors used
- Exporter uptime

See also:

All PMM exporters `pmm.list.exporter`

Summary information about the usage of Prometheus exporters [Prometheus Exporters Overview](#)

31.3 Prometheus Exporters Overview

The Prometheus Exporters Overview dashboard provides the summary of how exporters are used across the selected hosts.

See also:

Percona Database Performance Blog

[Understand Your Prometheus Exporters with Percona Monitoring and Management \(PMM\)](#)

Prometheus documentation

[Exporters and integrations](#)

- [Prometheus Exporters Summary](#)
- [Prometheus Exporters Resource Usage by Host](#)
- [Prometheus Exporters Resource Usage by Type](#)
- [List of Hosts](#)

31.3.1 Prometheus Exporters Summary

This section provides a summary of how exporters are used across the selected hosts. It includes the average usage of CPU and memory as well as the number of hosts being monitored and the total number of running exporters.

Metrics in this section

- **Avg CPU Usage per Host** shows the average CPU usage in percent per host for all exporters.
- **Avg Memory Usage per Host** shows the Exporters average Memory usage per host.
- **Monitored Hosts** shows the number of monitored hosts that are running Exporters.
- **Exporters Running** shows the total number of Exporters running with this PMM Server instance.

Note: The CPU usage and memory usage do not include the additional CPU and memory usage required to produce metrics by the application or operating system.

31.3.2 Prometheus Exporters Resource Usage by Host

This section shows how resources, such as CPU and memory, are being used by the exporters for the selected hosts.

Metrics in this section

- **CPU Usage** plots the Exporters' CPU usage across each monitored host (by default, All hosts).
- **Memory Usage** plots the Exporters' Memory usage across each monitored host (by default, All hosts).

31.3.3 Prometheus Exporters Resource Usage by Type

This section shows how resources, such as CPU and memory, are being used by the exporters for host types: MySQL, MongoDB, ProxySQL, and the system.

Metrics in this section

- **CPU Cores Used** shows the Exporters' CPU Cores used for each type of Exporter.
- **Memory Usage** shows the Exporters' memory used for each type of Exporter.

31.3.4 List of Hosts

At the bottom, this dashboard shows details for each running host.

- **CPU Used** show the CPU usage as a percentage for all Exporters.
- **Mem Used** shows total Memory Used by Exporters.
- **Exporters Running** shows the number of Exporters running.
- **RAM** shows the total amount of RAM of the host.
- **Virtual CPUs** shows the total number of virtual CPUs on the host.

You can click the value of the *CPU Used*, *Memory Used*, or *Exporters Running* column to open the *Prometheus Exporter Status* for further analysis.

Related information: Prometheus Documentation

Exporters and integrations <https://prometheus.io/docs/instrumenting/exporters>

MYSQL DASHBOARDS

32.1 MySQL Amazon Aurora Metrics

This dashboard provides metrics for analyzing Amazon Aurora instances.

- *Amazon Aurora Transaction Commits*
- *Amazon Aurora Load*
- *Aurora Memory Used*
- *Amazon Aurora Statement Latency*
- *Amazon Aurora Special Command Counters*
- *Amazon Aurora Problems*

32.1.1 Amazon Aurora Transaction Commits

This graph shows number of commits which the Amazon Aurora engine performed as well as the average commit latency. Graph Latency does not always correlates with number of commits performed and can quite high in certain situations.

32.1.2 Amazon Aurora Load

This graph shows what statements contribute most load on the system as well as what load corresponds to Amazon Aurora transaction commit.

- Write Transaction Commit Load: Load in Average Active Sessions per second for COMMIT operations
- UPDATE load: load in Average Active Sessions per second for UPDATE queries
- SELECT load: load in Average Active Sessions per second for SELECT queries
- DELETE load: load in Average Active Sessions per second for DELETE queries
- INSERT load: load in Average Active Sessions per second for INSERT queries

32.1.3 Aurora Memory Used

This graph shows how much memory is used by Amazon Aurora lock manager as well as amount of memory used by Amazon Aurora to store Data Dictionary.

- Aurora Lock Manager Memory: the amount of memory used by the Lock Manager, the module responsible for handling row lock requests for concurrent transactions.
- Aurora Dictionary Memory: the amount of memory used by the Dictionary, the space that contains metadata used to keep track of database objects, such as tables and indexes.

32.1.4 Amazon Aurora Statement Latency

This graph shows average latency for most important types of statements. Latency spikes are often indicative of the instance overload.

- DDL Latency: Average time to execute DDL queries
- DELETE Latency: average time to execute DELETE queries
- UPDATE Latency: average time to execute UPDATE queries
- SELECT Latency: average time to execute SELECT queries
- INSERT Latency: average time to execute INSERT queries

32.1.5 Amazon Aurora Special Command Counters

Amazon Aurora MySQL allows a number of commands which are not available from standard MySQL. This graph shows usage of such commands. Regular `unit_test` calls can be seen in default Amazon Aurora install, the rest will depend on your workload.

show_volume_status The number of executions per second of the command **SHOW VOLUME STATUS**. The **SHOW VOLUME STATUS** query returns two server status variables: Disks and Nodes. These variables represent the total number of logical blocks of data and storage nodes, respectively, for the DB cluster volume.

awslambda The number of AWS Lambda calls per second. AWS Lambda is an event-driven, serverless computing platform provided by AWS. It is a compute service that runs codes in response to an event. You can run any kind of code from Aurora invoking Lambda from a stored procedure or a trigger.

alter_system The number of executions per second of the special query ALTER SYSTEM, that is a special query to simulate an instance crash, a disk failure, a disk congestion or a replica failure. It is a useful query for testing the system.

32.1.6 Amazon Aurora Problems

This metric shows different kinds of internal Amazon Aurora MySQL problems which should be zero in case of normal operation.

- Reserved mem Exceeded Incidents
- Missing History on Replica Incidents
- Thread deadlocks: number of deadlocks per second

32.2 MySQL Command/Handler Counters Compare Dashboard

This dashboard shows server status variables. On this dashboard, you may select multiple servers and compare their counters simultaneously.

Server status variables appear in two sections: *Commands* and *Handlers*. Choose one or more variables in the *Command* and *Handler* fields in the top menu to select the variables which will appear in the *COMMANDS* or *HANDLERS* section for each host. Your comparison may include from one up to three hosts.

By default or if no item is selected in the menu, PMM displays each command or handler respectively.

See also:

MySQL Documentation: Server Status Variables <https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html>

32.3 MySQL InnoDB Compression Dashboard

This dashboard helps you analyze the efficiency of InnoDB compression.

See also:

MySQL Documentation <https://dev.mysql.com/doc/refman/5.7/en/innodb-information-schema-compression-tables.html>

- *Compression level and failure rate threshold*
- *Statistics of Compression Operations*
- *CPU Core Usage*
- *Buffer Pool Total*
- *Buffer Pool All*

32.3.1 Compression level and failure rate threshold

InnoDB Compression Level

The level of zlib compression to use for InnoDB compressed tables and indexes.

InnoDB Compression Failure Threshold

The compression failure rate threshold for a table.

Compression Failure Rate Threshold

The maximum percentage that can be reserved as free space within each compressed page, allowing room to reorganize the data and modification log within the page when a compressed table or index is updated and the data might be recompressed.

Write Pages to the Redo Log

Specifies whether images of re-compressed pages are written to the redo log. Re-compression may occur when changes are made to compressed data.

32.3.2 Statistics of Compression Operations

This section contains the following metrics:

- Compress Attempts
- Uncompressed Attempts
- Compression Success Ratio

Compress Attempts

Number of compression operations attempted. Pages are compressed whenever an empty page is created or the space for the uncompressed modification log runs out.

Uncompressed Attempts

Number of uncompression operations performed. Compressed InnoDB pages are uncompressed whenever compression fails, or the first time a compressed page is accessed in the buffer pool and the uncompressed page does not exist.

32.3.3 CPU Core Usage

- CPU Core Usage for Compression
- CPU Core Usage for Uncompression

32.3.4 Buffer Pool Total

- Total Used Pages
- Total Free Pages

32.3.5 Buffer Pool All

- Used Pages (Buffer Pool 0)
- Pages Free (Buffer Pool 0)

32.4 MySQL InnoDB Metrics

This dashboard contains metrics that help analyze how the InnoDB engine performs.

- *InnoDB Checkpoint Age*
- *InnoDB Transactions*
- *InnoDB Row Operations*
- *InnoDB Row Lock Time*
- *InnoDB I/O*

- *InnoDB Log File Usage Hourly*
- *InnoDB Deadlocks*
- *InnoDB Condition Pushdown*
- *Other Metrics*

32.4.1 InnoDB Checkpoint Age

The maximum checkpoint age is determined by the total length of all transaction log files (*innodb_log_file_size*).

When the checkpoint age reaches the maximum checkpoint age, blocks are flushed synchronously. The rule of thumb is to keep one hour of traffic in those logs and let the checkpointing perform its work as smooth as possible. If you don't do this, InnoDB will do synchronous flushing at the worst possible time, i.e. when you are busiest.

View all metrics of *MySQL InnoDB Metrics*

32.4.2 InnoDB Transactions

InnoDB is an MVCC storage engine, which means you can start a transaction and continue to see a consistent snapshot even as the data changes. This is implemented by keeping old versions of rows as they are modified.

The *InnoDB History List* is the undo logs which are used to store these modifications. They are a fundamental part of the InnoDB transactional architecture.

If the history length is rising regularly, do not let open connections linger for a long period as this can affect the performance of InnoDB considerably. It is also a good idea to look for long running queries in QAN.

View all metrics of *MySQL InnoDB Metrics*

32.4.3 InnoDB Row Operations

This metric allows you to see which operations occur and the number of rows affected per operation. A metric like *Queries Per Second* will give you an idea of queries, but one query could effect millions of rows.

View all metrics of *MySQL InnoDB Metrics*

32.4.4 InnoDB Row Lock Time

When data is locked, then that means that another session cannot update that data until the lock is released (which unlocks the data and allows other users to update that data. Locks are usually released by either a **ROLLBACK** or **COMMIT** SQL statement.

InnoDB implements standard row-level locking where there are two types of locks, shared (S) locks and exclusive (X) locks.

A shared (S) lock permits the transaction that holds the lock to read a row. An exclusive (X) lock permits the transaction that holds the lock to update or delete a row. *Average Row Lock Wait Time* is the row lock wait time divided by the number of row locks.

Row Lock Waits indicates how many times a transaction waited on a row lock per second.

Row Lock Wait Load is a rolling 5 minute average of *Row Lock Waits*.

View all metrics of *MySQL InnoDB Metrics*

See also:

MySQL Server Documentation: Shared lock https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_shared_lock

MySQL Server Documentation: Exclusive lock https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_exclusive_lock

MySQL Server Documentation: InnoDB locking <https://dev.mysql.com/doc/refman/5.7/en/innodb-locking.html>

32.4.5 InnoDB I/O

This metric has the following series:

- *Data Writes* is the total number of InnoDB data writes.
- *Data Reads* is the total number of InnoDB data reads (OS file reads).
- *Log Writes* is the number of physical writes to the InnoDB redo log file.
- *Data Fsyncs* is the number of fsync() operations. The frequency of fsync() calls is influenced by the setting of the `innodb_flush_method` configuration option.

View all metrics of *MySQL InnoDB Metrics*

32.4.6 InnoDB Log File Usage Hourly

Along with the buffer pool size, `innodb_log_file_size` is the most important setting when we are working with InnoDB. This graph shows how much data was written to InnoDB's redo logs over each hour. When the InnoDB log files are full, InnoDB needs to flush the modified pages from memory to disk.

The rule of thumb is to keep one hour of traffic in those logs and let the checkpointing perform its work as smooth as possible. If you don't do this, InnoDB will do synchronous flushing at the worst possible time, ie when you are busiest.

This graph can help guide you in setting the correct `innodb_log_file_size`.

View all metrics of *MySQL InnoDB Metrics*

See also:

Percona Database Performance Blog: Calculating a good InnoDB log file size <https://www.percona.com/blog/2008/11/21/how-to-calculate-a-good-innodb-log-file-size/>

Percona Server Documentation: Improved InnoDB I/O scalability http://www.percona.com/doc/percona-server/5.5/scalability/innodb_io_55.html#innodb_log_file_size

32.4.7 InnoDB Deadlocks

A deadlock in MySQL happens when two or more transactions mutually hold and request for locks, creating a cycle of dependencies. In a transaction system, deadlocks are a fact of life and not completely avoidable. InnoDB automatically detects transaction deadlocks, rolls back a transaction immediately and returns an error.

View all metrics of *MySQL InnoDB Metrics*

See also:

Percona Database Performance Blog: Dealing with MySQL deadlocks <https://www.percona.com/blog/2014/10/28/how-to-deal-with-mysql-deadlocks/>

32.4.8 InnoDB Condition Pushdown

Index Condition Pushdown (ICP) is an optimization for the case where MySQL retrieves rows from a table using an index.

Without ICP, the storage engine traverses the index to locate rows in the base table and returns them to the MySQL server which evaluates the `WHERE` condition for the rows. With ICP enabled, and if parts of the `WHERE` condition can be evaluated by using only columns from the index, the MySQL server pushes this part of the `WHERE` condition down to the storage engine. The storage engine then evaluates the pushed index condition by using the index entry and only if this is satisfied is the row read from the table.

ICP can reduce the number of times the storage engine must access the base table and the number of times the MySQL server must access the storage engine.

View all metrics of *MySQL InnoDB Metrics*

See also:

MySQL Server Documentation: Index Condition Pushdown optimisation <https://dev.mysql.com/doc/refman/5.7/en/index-condition-pushdown-optimization.html>

Percona Database Performance Blog: ICP counters and how to interpret them https://www.percona.com/blog/2017/05/09/mariadb-handler_icp_-counters-what-they-are-and-how-to-use-them/

32.4.9 Other Metrics

- InnoDB Logging Performance
- InnoDB Buffer Pool Content
- InnoDB Buffer Pool Pages
- InnoDB Buffer Pool I/O
- InnoDB Buffer Pool Requests
- InnoDB Buffer Read-Ahead
- InnoDB Change Buffer
- InnoDB Change Buffer Activity

32.5 MySQL InnoDB Metrics (Advanced) Dashboard

The MySQL InnoDB Metrics (Advanced) dashboard contains metrics that provide detailed information about the performance of the InnoDB storage engine on the selected MySQL host. This dashboard contains the following metrics:

Important: If you do not see any metric, try running the following command in the MySQL client:

```
mysql > SET GLOBAL innodb_monitor_enable=all;
```

Metrics of MySQL InnoDB Metrics (Advanced) Dashboard

- *Change Buffer Performance*
- *InnoDB Log Buffer Performance*

- *InnoDB Page Splits*
- *InnoDB Page Reorgs*
- *InnoDB Purge Performance*
- *InnoDB Locking*
- *InnoDB Main Thread Utilization*
- *InnoDB Transactions Information*
- *InnoDB Undo Space Usage*
- *InnoDB Activity*
- *InnoDB Contention - OS Waits*
- *InnoDB Contention - Spin Rounds*
- *InnoDB Group Commit Batch Size*
- *InnoDB Purge Throttling*
- *InnoDB AHI Usage*
- *InnoDB AHI Maintenance*
- *InnoDB Online DDL*
- *InnoDB Defragmentation*

32.5.1 Change Buffer Performance

This metric shows the activity on the InnoDB change buffer. The InnoDB change buffer records updates to non-unique secondary keys when the destination page is not in the buffer pool. The updates are applied when the page is loaded in the buffer pool, prior to its use by a query. The merge ratio is the number of insert buffer changes done per page, the higher the ratio the better is the efficiency of the change buffer.

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

32.5.2 InnoDB Log Buffer Performance

The InnoDB Log Buffer Performance graph shows the efficiency of the InnoDB log buffer. The InnoDB log buffer size is defined by the `innodb_log_buffer_size` parameter and illustrated on the graph by the *Size* graph. *Used* is the amount of the buffer space that is used. If the *Used* graph is too high and gets close to *Size*, additional log writes will be required.

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

32.5.3 InnoDB Page Splits

The InnoDB Page Splits graph shows the InnoDB page maintenance activity related to splitting and merging pages. When an InnoDB page, other than the top most leaf page, has too much data to accept a row update or a row insert, it has to be split in two. Similarly, if an InnoDB page, after a row update or delete operation, ends up being less than half full, an attempt is made to merge the page with a neighbor page. If the resulting page size is larger than the InnoDB page size, the operation fails. If your workload causes a large number of page splits, try lowering the `innodb_fill_factor` variable (5.7+).

[View all metrics of MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

32.5.4 InnoDB Page Reorgs

The InnoDB Page Reorgs graph shows information about the page reorganization operations. When a page receives an update or an insert that affect the offset of other rows in the page, a reorganization is needed. If the reorganization process finds out there is not enough room in the page, the page will be split. Page reorganization can only fail for compressed pages.

[View all metrics of MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

32.5.5 InnoDB Purge Performance

The InnoDB Purge Performance graph shows metrics about the page purging process. The purge process removed the undo entries from the history list and cleanup the pages of the old versions of modified rows and effectively remove deleted rows.

[View all metrics of MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

32.5.6 InnoDB Locking

The InnoDB Locking graph shows the row level lock activity inside InnoDB.

[View all metrics of MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

32.5.7 InnoDB Main Thread Utilization

The InnoDB Main Thread Utilization graph shows the portion of time the InnoDB main thread spent at various task.

[View all metrics of MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

32.5.8 InnoDB Transactions Information

The InnoDB Transactions Information graph shows details about the recent transactions. Transaction IDs Assigned represents the total number of transactions initiated by InnoDB. RW Transaction Commits are the number of transactions not read-only. Insert-Update Transactions Commits are transactions on the Undo entries. Non Locking RO Transaction Commits are transactions commit from select statement in auto-commit mode or transactions explicitly started with “start transaction read only”.

[View all metrics of MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

32.5.9 InnoDB Undo Space Usage

The InnoDB Undo Space Usage graph shows the amount of space used by the Undo segment. If the amount of space grows too much, look for long running transactions holding read views opened in the InnoDB status.

[View all metrics of MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

32.5.10 InnoDB Activity

The InnoDB Activity graph shows a measure of the activity of the InnoDB threads.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

32.5.11 InnoDB Contention - OS Waits

The InnoDB Contention - OS Waits graph shows the number of time an OS wait operation was required while waiting to get the lock. This happens once the spin rounds are exhausted.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

32.5.12 InnoDB Contention - Spin Rounds

The InnoDB Contention - Spin Rounds metric shows the number of spin rounds executed in order to get a lock. A spin round is a fast retry to get the lock in a loop.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

32.5.13 InnoDB Group Commit Batch Size

The InnoDB Group Commit Batch Size metric shows how many bytes were written to the InnoDB log files per attempt to write. If many threads are committing at the same time, one of them will write the log entries of all the waiting threads and flush the file. Such process reduces the number of disk operations needed and enlarge the batch size.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

32.5.14 InnoDB Purge Throttling

The InnoDB Purge Throttling graph shows the evolution of the purge lag and the max purge lag currently set. Under heavy write load, the purge operation may start to lag behind and when the max purge lag is reached, a delay, proportional to the value defined by `innodb_max_purge_lag_delay` (in microseconds) is added to all update, insert and delete statements. This helps prevents flushing stalls.

https://dev.mysql.com/doc/refman/5.6/en/innodb-parameters.html#sysvar_innodb_max_purge_lag

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

32.5.15 InnoDB AHI Usage

The InnoDB AHI Usage graph shows the search operations on the InnoDB adaptive hash index and its efficiency. The adaptive hash index is a search hash designed to speed access to InnoDB pages in memory. If the Hit Ratio is small, the working data set is larger than the buffer pool, the AHI should likely be disabled.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

32.5.16 InnoDB AHI Maintenance

The InnoDB AHI Maintenance graph shows the maintenance operation of the InnoDB adaptive hash index. The adaptive hash index is a search hash to speed access to InnoDB pages in memory. A constant high number of rows/pages added and removed can be an indication of an ineffective AHI.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

32.5.17 InnoDB Online DDL

The InnoDB Online DDL graph shows the state of the online DDL (alter table) operations in InnoDB. The progress metric is estimate of the percentage of the rows processed by the online DDL.

Note: Currently available only on MariaDB Server

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

32.5.18 InnoDB Defragmentation

The InnoDB Defragmentation graph shows the status information related to the InnoDB online defragmentation feature of MariaDB for the optimize table command. To enable this feature, the variable innodb-defragment must be set to **1** in the configuration file.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

Note: Currently available only on MariaDB Server.

32.6 MySQL MyISAM Aria Metrics Dashboard

The MySQL MyISAM Aria Metrics dashboard describes the specific features of MariaDB MySQL server: [Aria storage engine](#), [Online DDL \(online alter table\)](#),

and [InnoDB defragmentation patch](#). This dashboard contains the following metrics:

- *Aria Storage Engine*
- *Aria Pagecache Reads/Writes*
- *Aria Pagecache Blocks*
- *Aria Transactions Log Syncs*

32.6.1 Aria Storage Engine

Aria storage is specific for MariaDB Server. Aria has most of the same variables that MyISAM has, but with an Aria prefix. If you use Aria instead of MyISAM, then you should make `key_buffer_size` smaller and `aria-pagecache-buffer-size` bigger.

32.6.2 Aria Pagecache Reads/Writes

This graph is similar to InnoDB buffer pool reads and writes. `aria-pagecache-buffer-size` is the main cache for aria storage engine. If you see high reads and writes (physical IO), i.e. reads is close to read requests or writes are close to write requests you may need to increase the `aria-pagecache-buffer-size` (you may need to decrease other buffers: `key_buffer_size`, `innodb_buffer_pool_size` etc)

32.6.3 Aria Pagecache Blocks

This graphs shows the utilization for the aria pagecache. This is similar to InnoDB buffer pool graph. If you see all blocks are used you may consider increasing `aria-pagecache-buffer-size` (you may need to decrease other buffers: `key_buffer_size`, `innodb_buffer_pool_size` etc)

32.6.4 Aria Transactions Log Syncs

This metric is similar to InnoDB log file syncs. If you see lots of log syncs and want to relax the durability settings you can change (in seconds) from 30 (default) to a higher number. It is good to look at the disk IO dashboard as well.

See also:

List of Aria system variables

<https://mariadb.com/kb/en/library/aria-system-variables/>

32.7 MySQL MyRocks Metrics Dashboard

The **MyRocks** storage engine developed by Facebook based on the RocksDB storage engine is applicable to systems which primarily interact with the database by writing data to it rather than reading from it. RocksDB also features a good level of compression, higher than that of the InnoDB storage engine, which makes it especially valuable when optimizing the usage of hard drives.

PMM collects statistics on the MyRocks storage engine for MySQL in the Metrics Monitor information for this dashboard comes from the *Information Schema* tables.

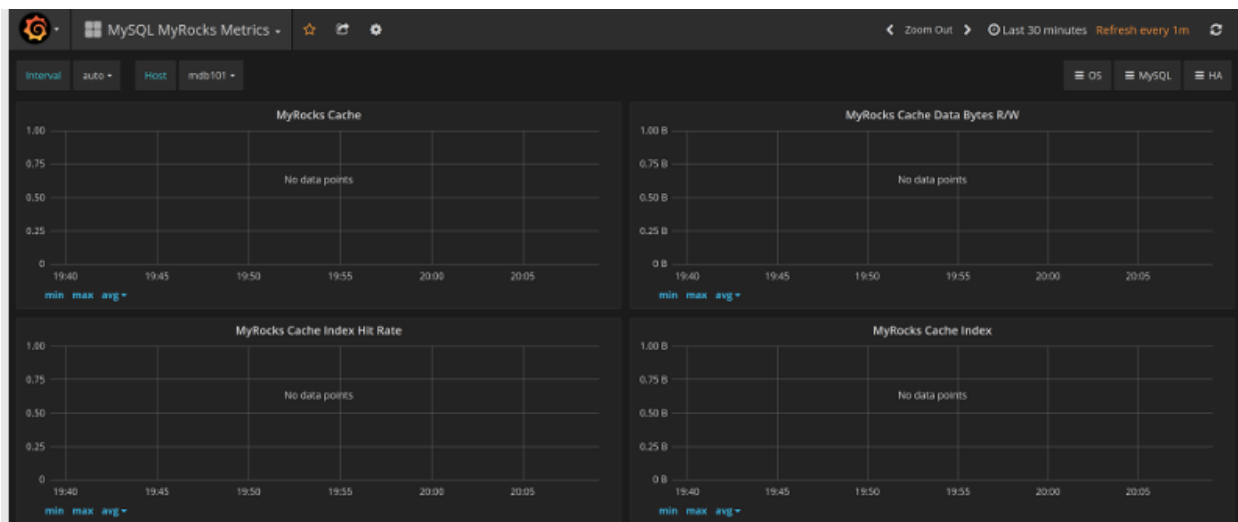


Fig. 32.1: The MySQL MyRocks metrics dashboard

See also:

Information schema <https://github.com/facebook/mysql-5.6/wiki/MyRocks-Information-Schema>

Metrics

- MyRocks cache
- MyRocks cache data bytes R/W
- MyRocks cache index hit rate
- MyRocks cache index
- MyRocks cache filter hit rate
- MyRocks cache filter
- MyRocks cache data bytes inserted
- MyRocks bloom filter
- MyRocks memtable
- MyRocks memtable size
- MyRocks number of keys
- MyRocks cache L0/L1
- MyRocks number of DB ops
- MyRocks R/W
- MyRocks bytes read by iterations
- MyRocks write ops
- MyRocks WAL
- MyRocks number reseek in iterations
- RocksDB row operations
- MyRocks file operations
- RocksDB stalls
- RocksDB stops/slowdowns

32.8 MySQL Overview

This dashboard provides basic information about MySQL hosts.

- *MySQL Uptime*
- *Current QPS*
- *InnoDB Buffer Pool Size*
- *Buffer Pool Size % of Total RAM*
- *MySQL Connections*
- *MySQL Active Threads*
- *MySQL Questions*
- *MySQL Thread Cache*
- *MySQL Select Types*
- *MySQL Sorts*
- *MySQL Slow Queries*

- *Aborted Connections*
- *Table Locks*
- *MySQL Network Traffic*
- *MySQL Network Usage Hourly*
- *MySQL Internal Memory Overview*
- *Top Command Counters and Top Command Counters Hourly*
- *MySQL Handlers*
- *MySQL Query Cache Memory and MySQL Query Cache Activity*
- *MySQL Open Tables, MySQL Table Open Cache Status, and MySQL Table Definition Cache*

32.8.1 MySQL Uptime

The amount of time since the MySQL server process was started.

View all metrics of *MySQL Overview*

32.8.2 Current QPS

Based on the queries reported by MySQL's **SHOW STATUS** command, this metric shows the number of queries executed by the server during the last second. This metric includes statements executed within stored programs.

This variable does not include the following commands:

- COM_PING
- COM_STATISTICS

View all metrics of *MySQL Overview*

See also:

MySQL Server Status Variables: Queries https://dev.mysql.com/doc/refman/5.6/en/server-status-variables.html#statvar_Queries

32.8.3 InnoDB Buffer Pool Size

Absolute value of the InnoDB buffer pool used for caching data and indexes in memory.

The goal is to keep the working set in memory. In most cases, this should be between 60%-90% of available memory on a dedicated database host, but depends on many factors.

View all metrics of *MySQL Overview*

32.8.4 Buffer Pool Size % of Total RAM

The ratio between InnoDB buffer pool size and total memory. In most cases, the InnoDB buffer pool should be between 60% and 90% of available memory on a dedicated database host, but it depends on many factors.

View all metrics of *MySQL Overview*

32.8.5 MySQL Connections

Max Connections The maximum permitted number of simultaneous client connections. This is the value of the `max_connections` variable.

Max Used Connections The maximum number of connections that have been in use simultaneously since the server was started.

Connections The number of connection attempts (successful or not) to the MySQL server.

View all metrics of *MySQL Overview*

See also:

MySQL Server status variables: max_connections https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html#sysvar_max_connections

32.8.6 MySQL Active Threads

Threads Connected The number of open connections.

Threads Running The number of threads not sleeping.

View all metrics of *MySQL Overview*

32.8.7 MySQL Questions

The number of queries sent to the server by clients, *excluding those executed within stored programs*.

This variable does not count the following commands:

View all metrics of *MySQL Overview*

- COM_PING
- COM_STATISTICS
- COM_STMT_PREPARE
- COM_STMT_CLOSE
- COM_STMT_RESET

View all metrics of *MySQL Overview*

32.8.8 MySQL Thread Cache

The `thread_cache_size` metric informs how many threads the server should cache to reuse. When a client disconnects, the client's threads are put in the cache if the cache is not full. It is autosized in MySQL 5.6.8 and above (capped to 100).

Requests for threads are satisfied by reusing threads taken from the cache if possible, and only when the cache is empty is a new thread created.

- `Threads_created`: The number of threads created to handle connections.
- `Threads_cached`: The number of threads in the thread cache.

View all metrics of *MySQL Overview*

See also:

MySQL Server status variables: `thread_cache_size` https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html#sysvar_thread_cache_size

32.8.9 MySQL Select Types

As with most relational databases, selecting based on indexes is more efficient than scanning the data of an entire table. Here, we see the counters for selects not done with indexes.

- *Select Scan* is how many queries caused full table scans, in which all the data in the table had to be read and either discarded or returned.
- *Select Range* is how many queries used a range scan, which means MySQL scanned all rows in a given range.
- *Select Full Join* is the number of joins that are not joined on an index, this is usually a huge performance hit.

View all metrics of *MySQL Overview*

32.8.10 MySQL Sorts

Due to a query's structure, order, or other requirements, MySQL sorts the rows before returning them. For example, if a table is ordered 1 to 10 but you want the results reversed, MySQL then has to sort the rows to return 10 to 1.

This graph also shows when sorts had to scan a whole table or a given range of a table in order to return the results and which could not have been sorted via an index.

View all metrics of *MySQL Overview*

32.8.11 MySQL Slow Queries

Slow queries are defined as queries being slower than the `long_query_time` setting. For example, if you have `long_query_time` set to **3**, all queries that take longer than **3** seconds to complete will show on this graph.

View all metrics of *MySQL Overview*

32.8.12 Aborted Connections

When a given host connects to MySQL and the connection is interrupted in the middle (for example due to bad credentials), MySQL keeps that info in a system table (since 5.6 this table is exposed in `performance_schema`).

If the amount of failed requests without a successful connection reaches the value of `max_connect_errors`, **mysqld** assumes that something is wrong and blocks the host from further connections.

To allow connections from that host again, you need to issue the **FLUSH HOSTS** statement.

View all metrics of *MySQL Overview*

32.8.13 Table Locks

MySQL takes a number of different locks for varying reasons. In this graph we see how many Table level locks MySQL has requested from the storage engine. In the case of InnoDB, many times the locks could actually be row locks as it only takes table level locks in a few specific cases.

It is most useful to compare *Locks Immediate* and *Locks Waited*. If *Locks Waited* is rising, it means you have lock contention. Otherwise, *Locks Immediate* rising and falling is normal activity.

View all metrics of *MySQL Overview*

32.8.14 MySQL Network Traffic

This metric shows how much network traffic is generated by MySQL. *Outbound* is network traffic sent from MySQL and *Inbound* is the network traffic that MySQL has received.

[View all metrics of MySQL Overview](#)

32.8.15 MySQL Network Usage Hourly

This metric shows how much network traffic is generated by MySQL per hour. You can use the bar graph to compare data sent by MySQL and data received by MySQL.

[View all metrics of MySQL Overview](#)

32.8.16 MySQL Internal Memory Overview

This metric shows the various uses of memory within MySQL.

System Memory

Total Memory for the system.

InnoDB Buffer Pool Data

InnoDB maintains a storage area called the buffer pool for caching data and indexes in memory. Knowing how the InnoDB buffer pool works, and taking advantage of it to keep frequently accessed data in memory, is an important aspect of MySQL tuning.

TokuDB Cache Size

Similar in function to the InnoDB Buffer Pool, TokuDB will allocate 50% of the installed RAM for its own cache. While this is optimal in most situations, there are cases where it may lead to memory over allocation. If the system tries to allocate more memory than is available, the machine will begin swapping and run much slower than normal.

Key Buffer Size

Index blocks for MyISAM tables are buffered and are shared by all threads. *key_buffer_size* is the size of the buffer used for index blocks. The key buffer is also known as the *key cache*.

Adaptive Hash Index Size

The InnoDB storage engine has a special feature called adaptive hash indexes. When InnoDB notices that some index values are being accessed very frequently, it builds a hash index for them in memory on top of B-Tree indexes. This allows for very fast hashed lookups.

Query Cache Size

The query cache stores the text of a **SELECT** statement together with the corresponding result that was sent to the client. The query cache has huge scalability problems in that only one thread can do an operation in the query cache at the same time. This serialization is true for **SELECT** and also for **INSERT**, **UPDATE**, and **DELETE**. This also means that the larger the *query_cache_size* is set to, the slower those operations become.

InnoDB Dictionary Size

The data dictionary is InnoDB internal catalog of tables. InnoDB stores the data dictionary on disk, and loads entries into memory while the server is running. This is somewhat analogous to table cache of MySQL, but instead of operating at the server level, it is internal to the InnoDB storage engine.

InnoDB Log Buffer Size

The MySQL InnoDB log buffer allows transactions to run without having to write the log to disk before the transactions commit. The size of this buffer is configured with the `innodb_log_buffer_size` variable.

[View all metrics of MySQL Overview](#)

32.8.17 Top Command Counters and Top Command Counters Hourly

See https://dev.mysql.com/doc/refman/5.7/en/server-status-variables.html#statvar_Com_xxx

[View all metrics of MySQL Overview](#)

32.8.18 MySQL Handlers

Handler statistics are internal statistics on how MySQL is selecting, updating, inserting, and modifying rows, tables, and indexes.

This is in fact the layer between the Storage Engine and MySQL.

- `read_rnd_next` is incremented when the server performs a full table scan and this is a counter you don't really want to see with a high value.
- `read_key` is incremented when a read is done with an index.
- `read_next` is incremented when the storage engine is asked to 'read the next index entry'. A high value means a lot of index scans are being done.

[View all metrics of MySQL Overview](#)

32.8.19 MySQL Query Cache Memory and MySQL Query Cache Activity

The query cache has huge scalability problems in that only one thread can do an operation in the query cache at the same time. This serialization is true not only for **SELECT**, but also for **INSERT**, **UPDATE**, and **DELETE**.

This also means that the larger the `query_cache_size` is set to, the slower those operations become. In concurrent environments, the MySQL Query Cache quickly becomes a contention point, decreasing performance. MariaDB and Amazon Aurora have done work to try and eliminate the query cache contention in their flavors of MySQL, while MySQL 8.0 has eliminated the query cache feature.

The recommended settings for most environments is to set:

```
query_cache_type=0
query_cache_size=0
```

Note: While you can dynamically change these values, to completely remove the contention point you have to restart the database.

[View all metrics of MySQL Overview](#)

32.8.20 MySQL Open Tables, MySQL Table Open Cache Status, and MySQL Table Definition Cache

The recommendation is to set the `table_open_cache_instances` to a loose correlation to virtual CPUs, keeping in mind that more instances means the cache is split more times. If you have a cache set to 500 but it has 10 instances, each cache will only have 50 cached.

The `table_definition_cache` and `table_open_cache` can be left as default as they are autosized MySQL 5.6 and above (do not set them to any value).

View all metrics of [MySQL Overview](#)

See also:

Configuring MySQL for PMM [pmm.conf-mysql.settings.dashboard](#)

MySQL Documentation: InnoDB buffer pool <https://dev.mysql.com/doc/refman/5.7/en/innodb-buffer-pool.html>

Percona Server Documentation: Running TokuDB in Production https://www.percona.com/doc/percona-server/LATEST/tokudb/tokudb_quickstart.html#considerations-to-run-tokudb-in-production

Blog post: Adaptive Hash Index in InnoDB <https://www.percona.com/blog/2016/04/12/is-adaptive-hash-index-in-innodb-right-for-my-workload/>

MySQL Server System Variables: key_buffer_size https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_key_buffer_size

MySQL Server System Variables: table_open_cache http://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html#sysvar_table_open_cache

32.9 MySQL Performance Schema Dashboard

The MySQL *Performance Schema* dashboard helps determine the efficiency of communicating with *Performance Schema*. This dashboard contains the following metrics:

- *Performance Schema* file IO (events)
- *Performance Schema* file IO (load)
- *Performance Schema* file IO (Bytes)
- *Performance Schema* waits (events)
- *Performance Schema* waits (load)
- Index access operations (load)
- Table access operations (load)
- *Performance Schema* SQL and external locks (events)
- *Performance Schema* SQL and external locks (seconds)

See also:

MySQL Documentation: *Performance Schema*

<https://dev.mysql.com/doc/refman/5.7/en/performance-schema.html>

32.10 Performance Schema Wait Event Analysis Dashboard

This dashboard helps to analyse *Performance Schema* wait events. It plots the following metrics for the chosen (one or more) wait events:

- *Count - Performance Schema Waits*
- *Load - Performance Schema Waits*
- *Avg Wait Time - Performance Schema Waits*

See also:

MySQL Documentation: *Performance Schema*

<https://dev.mysql.com/doc/refman/5.7/en/performance-schema.html>

32.11 MySQL Query Response Time

This dashboard provides information about query response time distribution.

- *Average Query Response Time*
- *Query Response Time Distribution*
- *Average Query Response Time (Read/Write Split)*
- *Read Query Response Time Distribution*
- *Write Query Response Time Distribution*

32.11.1 Average Query Response Time

The Average Query Response Time graph shows information collected using the Response Time Distribution plugin sourced from table `INFORMATION_SCHEMA.QUERY_RESPONSE_TIME`. It computes this value across all queries by taking the sum of seconds divided by the count of seconds.

View all metrics of *MySQL Query Response Time*

See also:

Percona Server Documentation: QUERY_RESPONSE_TIME table https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME

32.11.2 Query Response Time Distribution

Shows how many fast, neutral, and slow queries are executed per second.

Query response time counts (operations) are grouped into three buckets:

- 100ms - 1s
- 1s - 10s
- > 10s

View all metrics of *MySQL Query Response Time*

32.11.3 Average Query Response Time (Read/Write Split)

Available only in Percona Server for MySQL, this metric provides visibility of the split of READ vs WRITE query response time.

View all metrics of *MySQL Query Response Time*

See also:

Percona Server Documentation: Logging queries in separate READ and WRITE tables https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#logging-the-queries-in-separate-read-and-write-tables

Percona Server Documentation: QUERY_RESPONSE_TIME_READ https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME_READ

Percona Server Documentation: QUERY_RESPONSE_TIME_WRITE https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME_WRITE

32.11.4 Read Query Response Time Distribution

Available only in Percona Server for MySQL, illustrates READ query response time counts (operations) grouped into three buckets:

- 100ms - 1s
- 1s - 10s
- > 10s

View all metrics of *MySQL Query Response Time*

See also:

Percona Server Documentation: QUERY_RESPONSE_TIME_READ https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME_READ

32.11.5 Write Query Response Time Distribution

Available only in Percona Server for MySQL, illustrates WRITE query response time counts (operations) grouped into three buckets:

- 100ms - 1s
- 1s - 10s
- > 10s

View all metrics of *MySQL Query Response Time*

See also:

Configuring MySQL for PMM `pmm.conf-mysql.query-response-time`

Percona Server Documentation: QUERY_RESPONSE_TIME_WRITE https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME_WRITE

32.12 MySQL Replication

- *IO Thread Running*
- *SQL Thread Running*
- *Replication Error No*
- *Read only*
- *MySQL Replication Delay*
- *Binlog Size*

- *Binlog Data Written Hourly*
- *Binlog Count*
- *Binlogs Created Hourly*
- *Relay Log Space*
- *Relay Log Written Hourly*

32.12.1 IO Thread Running

This metric shows if the IO Thread is running or not. It only applies to a slave host.

SQL Thread is a process that runs on a slave host in the replication environment. It reads the events from the local relay log file and applies them to the slave server.

Depending on the format of the binary log it can read query statements in plain text and re-execute them or it can read raw data and apply them to the local host.

Possible values

Yes

The thread is running and is connected to a replication master

No

The thread is not running because it is not launched yet or because an error has occurred connecting to the master host

Connecting

The thread is running but is not connected to a replication master

No value

The host is not configured to be a replication slave

IO Thread Running is one of the parameters that the command **SHOW SLAVE STATUS** returns.

See also:

MySQL Documentation

- [Replication](#)
- [SHOW SLAVE STATUS Syntax](#)
- [IO Thread states](#)

View all metrics of *MySQL Replication*

32.12.2 SQL Thread Running

This metric shows if the SQL thread is running or not. It only applies to a slave host.

SQL Thread is a process that runs on a slave host in the replication environment. It reads the events from the local relay log file and applies them to the slave server.

Depending on the format of the binary log it can read query statements in plain text and re-execute them or it can read raw data and apply them to the local host.

Possible values

Yes

SQL Thread is running and is applying events from the realy log to the local slave host

No

SQL Thread is not running because it is not launched yet or because of an error occurred while applying an event to the local slave host

See also:

MySQL Documentation:

- [Replication](#)
- [SHOW SLAVE STATUS Syntax](#)
- [SQL Thread states](#)

View all metrics of *MySQL Replication*

32.12.3 Replication Error No

This metric shows the number of the last error in the SQL Thread encountered which caused replication to stop.

One of the more common errors is *Error: 1022 Duplicate Key Entry*. In such a case replication is attempting to update a row that already exists on the slave. The SQL Thread will stop replication in order to avoid data corruption.

See also:

MySQL Documentation:

[A complete list of error codes](#)

View all metrics of *MySQL Replication*

32.12.4 Read only

This metric indicates whether the host is configured to be in *Read Only* mode or not.

Possible values

Yes

The slave host permits no client updates except from users who have the SUPER privilege or the REPLICATION SLAVE privilege.

This kind of configuration is typically used for slave hosts in a replication environment to avoid a user can inadvertently or voluntarily modify data causing inconsistencies and stopping the replication process.

No

The slave host is not configured in *Read Only* mode.

See also:

MySQL Documentation:

[Replication](#)

View all metrics of *MySQL Replication*

32.12.5 MySQL Replication Delay

This metric shows the number of seconds the slave host is delayed in replication applying events compared to when the Master host applied them, denoted by the `Seconds_Behind_Master` value, and only applies to a slave host.

Since the replication process applies the data modifications on the slave asynchronously, it could happen that the slave replicates events after some time. The main reasons are:

- **Network round trip time** - high latency links will lead to non-zero replication lag values.
- **Single threaded nature of replication channels** - master servers have the advantage of applying changes in parallel, whereas slave ones are only able to apply changes in serial, thus limiting their throughput. In some cases Group Commit can help but is not always applicable.
- **High number of changed rows or computationally expensive SQL** - depending on the replication format (ROW vs STATEMENT), significant changes to the database through high volume of rows modified, or expensive CPU will all contribute to slave servers lagging behind the master.

Generally adding more CPU or Disk resources can alleviate replication lag issues, up to a point.

See also:

Related metrics:

- [Relay Log Space](#)

MySQL Documentation

- [SHOW SLAVE STATUS Syntax](#)
- [Improving replication performance](#)
- [Replication Slave Options and Variables](#)

View all metrics of *MySQL Replication*

32.12.6 Binlog Size

This metric shows the overall size of the binary log files, which can exist on both master and slave servers. The binary log (also known as the binlog) contains events that describe database changes: `CREATE TABLE`, `ALTER TABLE`, updates, inserts, deletes and other statements or database changes. The binlog is the file that is read by slaves via their IO Thread process in order to replicate database changes modification on the data and on the table structures. There can be more than one binlog file present depending on the binlog rotation policy adopted (for example using the configuration variables `max_binlog_size` and `expire_logs_days`).

There can be more binlog files depending on the rotation policy adopted (for example using the configuration variables `max_binlog_size` and `expire_logs_days`) or even because of server reboots.

When planning the disk space, take care of the overall dimension of binlog files and adopt a good rotation policy or think about having a separate mount point or disk to store the binlog data.

See also:

MySQL Documentation:

- [The binary log](#)
- [Configuring replication](#)

View all metrics of *MySQL Replication*

32.12.7 Binlog Data Written Hourly

This metric shows the amount of data written hourly to the binlog files during the last 24 hours. This metric can give you an idea of how big is your application in terms of data writes (creation, modification, deletion).

View all metrics of *MySQL Replication*

32.12.8 Binlog Count

This metric shows the overall count of binary log files, on both master and slave servers.

There can be more binlog files depending on the rotation policy adopted (for example using the configuration variables `max_binlog_size` and `expire_logs_days`) or even because of server reboots.

When planning the disk space, take care of the overall dimension of binlog files and adopt a good rotation policy or think about having a separate mount point or disk to store the binlog data.

See also:

MySQL Documentation:

- [The binary log](#)
- [Configuring replication](#)

View all metrics of *MySQL Replication*

32.12.9 Binlogs Created Hourly

This metric shows the number of binlog files created hourly during the last 24 hours.

There can be more binlog files depending on the rotation policy adopted (for example using the configuration variables `max_binlog_size` and `expire_logs_days`) or even because of server reboots.

When planning the disk space, take care of the overall dimension of binlog files and adopt a good rotation policy or think about having a separate mount point or disk to store the binlog data.

View all metrics of *MySQL Replication*

32.12.10 Relay Log Space

This metric shows the overall size of the relay log files. It only applies to a slave host.

The relay log consists of a set of numbered files containing the events to be executed on the slave host in order to replicate database changes.

The relay log has the same format as the binlog.

There can be multiple relay log files depending on the rotation policy adopted (using the configuration variable `max_relay_log_size`).

As soon as the SQL thread completes to execute all events in the relay log file, the file is deleted.

If this metric contains a high value, the variable `max_relay_log_file` is high too. Generally, this not a serious issue. If the value of this metric is constantly increased, the slave is delaying too much in applying the events.

Treat this metric in the same way as the *MySQL Replication Delay* metric.

See also:

MySQL Documentation:

- [The Slave Relay Log](#)

View all metrics of *MySQL Replication*

32.12.11 Relay Log Written Hourly

This metric shows the amount of data written hourly into relay log files during the last 24 hours.

View all metrics of *MySQL Replication*

32.13 MySQL Table Statistics

This dashboard presents various data related to MySQL tables.

- [Largest Tables](#)
- [Pie](#)
- [Table Activity](#)
- [Rows read](#)
- [Rows Changed](#)
- [Auto Increment Usage](#)

32.13.1 Largest Tables

Largest Tables by Row Count The estimated number of rows in the table from `information_schema.tables`.

Largest Tables by Size The size of the table components from `information_schema.tables`.

32.13.2 Pie

Total Database Size The total size of the database: as data + index size, so freeble one.

Most Fragmented Tables by Freeable Size The list of 5 most fragmented tables ordered by their freeable size

32.13.3 Table Activity

The next two graphs are available only for [Percona Server](#) and [MariaDB](#) and require `userstat` variable turned on.

32.13.4 Rows read

The number of rows read from the table, shown for the top 5 tables.

32.13.5 Rows Changed

The number of rows changed in the table, shown for the top 5 tables.

32.13.6 Auto Increment Usage

The current value of an `auto_increment` column from `information_schema`, shown for the top 10 tables.

32.14 *MySQL User Statistics*

This dashboard presents various data related to MySQL users.

Note: This dashboard requires Percona Server for MySQL 5.1+ or MariaDB 10.1/10.2 with XtraDB. Also `userstat` should be enabled, for example with the `SET GLOBAL userstat=1` statement. See [Configuring MySQL for Best Results](#) for further instructions.

Data is displayed for the 5 top users.

Top Users by Connections Created The number of times user's connections connected using SSL to the server.

Top Users by Traffic The number of bytes sent to the user's connections.

Top Users by Rows Fetched/Read The number of rows fetched by the user's connections.

Top Users by Rows Updated The number of rows updated by the user's connections.

Top Users by Busy Time The cumulative number of seconds there was activity on connections from the user.

Top Users by CPU Time The cumulative CPU time elapsed, in seconds, while servicing connections of the user.

MONGODB DASHBOARDS

33.1 MongoDB Cluster Summary

The MongoDB Cluster Summary dashboard shows the statistics on the selected MongoDB cluster. Namely, it reports the following information:

- Unsharded DBs
- Sharded DBs
- Sharded collections
- Shards
- Chunks
- Balancer enabled
- Chunks balanced
- Mongos operations
- Mongos connections
- Mongos cursors
- Chunk split events
- Change log events
- Operations per shard
- Chunks by shard
- Connections per shard
- Cursors per shard
- Replication lag by set
- Oplog range by set
- Shard elections
- Collection lock time

33.2 MongoDB inMemory Dashboard

The MongoDB inMemory dashboard shows statistics on the In-Memory storage engine for the selected MongoDB instances. This dashboard contains the following metrics:

- InMemory data size
- InMemory max data size
- InMemory available
- InMemory dirty pages
- InMemory transactions
- InMemory capacity
- InMemory sessions
- InMemory pages

- InMemory concurrency tickets
- Queued operations
- Document changes
- InMemory cache eviction
- Scanned and moved objects
- Page faults

33.3 MongoDB MMAPv1 Dashboard

The MongoDB MMAPv1 dashboard contains metrics that describe the performance of the MMAPv1 storage engine for MongoDB. This dashboard includes the following metrics:

- MMAPv1 lock wait ratio
- MMAPv1 write lock time
- Memory cached
- Memory available
- Document activity
- MMAPv1 lock ratios
- MMAPv1 lock wait time
- MMAPv1 page faults
- MMAPv1 journal write activity
- MMAPv1 journal commit activity
- MMAPv1 journaling time
- MMAPv1 journaling time - 99th percentile
- MMAPv1 background flushing time
- Queued operations
- Client operations
- Scanned and moved objects
- MMAPv1 memory usage
- MMAPv1 memory dirty pages

33.4 MongoDB Overview

This dashboard provides basic information about MongoDB instances.

33.4.1 Command Operations

Shows how many times a command is executed per second on average during the selected interval.

Look for peaks and drops and correlate them with other graphs.

View all metrics of [MongoDB Overview](#)

33.4.2 Connections

Keep in mind the hard limit on the maximum number of connections set by your distribution.

Anything over 5,000 should be a concern, because the application may not close connections correctly.

View all metrics of [MongoDB Overview](#)

33.4.3 Cursors

Helps identify why connections are increasing. Shows active cursors compared to cursors being automatically killed after 10 minutes due to an application not closing the connection.

View all metrics of *MongoDB Overview*

33.4.4 Document Operations

When used in combination with **Command Operations**, this graph can help identify *write amplification*. For example, when one `insert` or `update` command actually inserts or updates hundreds, thousands, or even millions of documents.

View all metrics of *MongoDB Overview*

33.4.5 Queued Operations

Any number of queued operations for long periods of time is an indication of possible issues. Find the cause and fix it before requests get stuck in the queue.

View all metrics of *MongoDB Overview*

33.4.6 `getLastError Write Time`, `getLastError Write Operations`

This is useful for write-heavy workloads to understand how long it takes to verify writes and how many concurrent writes are occurring.

View all metrics of *MongoDB Overview*

33.4.7 Asserts

Asserts are not important by themselves, but you can correlate spikes with other graphs.

View all metrics of *MongoDB Overview*

33.4.8 Memory Faults

Memory faults indicate that requests are processed from disk either because an index is missing or there is not enough memory for the data set. Consider increasing memory or sharding out.

View all metrics of *MongoDB Overview*

33.5 MongoDB ReplSet

This dashboard provides information about replica sets and their members.

- *Replication Operations*
- *ReplSet State*

- *ReplSet Members*
- *ReplSet Last Election*
- *ReplSet Lag*
- *Storage Engine*
- *Oplog Insert Time*
- *Oplog Recovery Window*
- *Replication Lag*
- *Elections*
- *Member State Uptime*
- *Max Heartbeat Time*
- *Max Member Ping Time*

33.5.1 Replication Operations

This metric provides an overview of database replication operations by type and makes it possible to analyze the load on the replica in more granular manner. These values only appear when the current host has replication enabled.

33.5.2 ReplSet State

This metric shows the role of the selected member instance (PRIMARY or SECONDARY).

View all metrics of *MongoDB ReplSet*

33.5.3 ReplSet Members

This metric the number of members in the replica set.

View all metrics of *MongoDB ReplSet*

33.5.4 ReplSet Last Election

This metric how long ago the last election occurred.

View all metrics of *MongoDB ReplSet*

33.5.5 ReplSet Lag

This metric shows the current replication lag for the selected member.

View all metrics of *MongoDB ReplSet*

33.5.6 Storage Engine

This metric shows the storage engine used on the instance

View all metrics of *MongoDB ReplSet*

33.5.7 Oplog Insert Time

This metric shows how long it takes to write to the oplog. Without it the write will not be successful.

This is more useful in mixed replica sets (where instances run different storage engines).

View all metrics of *MongoDB ReplSet*

33.5.8 Oplog Recovery Window

This metric shows the time range in the oplog and the oldest backed up operation.

For example, if you take backups every 24 hours, each one should contain at least 36 hours of backed up operations, giving you 12 hours of restore window.

View all metrics of *MongoDB ReplSet*

33.5.9 Replication Lag

This metric shows the delay between an operation occurring on the primary and that same operation getting applied on the selected member

View all metrics of *MongoDB ReplSet*

33.5.10 Elections

Elections happen when a primary becomes unavailable. Look at this graph over longer periods (weeks or months) to determine patterns and correlate elections with other events.

View all metrics of *MongoDB ReplSet*

33.5.11 Member State Uptime

This metric shows how long various members were in PRIMARY and SECONDARY roles.

View all metrics of *MongoDB ReplSet*

33.5.12 Max Heartbeat Time

This metric shows the heartbeat return times sent by the current member to other members in the replica set.

Long heartbeat times can indicate network issues or that the server is too busy.

View all metrics of *MongoDB ReplSet*

33.5.13 Max Member Ping Time

This metric can show a correlation with the replication lag value.

View all metrics of *MongoDB ReplSet*

33.6 MongoDB RocksDB Dashboard

The MongoDB RocksDB dashboard contains metrics that describe the performance of the RocksDB storage engine for the selected MongoDB host instance. This dashboard contains the following metrics:

- RocksDB Memtable used
- RocksDB block cache used
- Memory cached
- Document activity
- RocksDB cache usage
- RocksDB Memtable entries
- RocksDB block cache hit ratio
- RocksDB write activity
- RocksDB read activity
- RocksDB Level0 read latency
- RocksDB LevelN read average latency
- RocksDB LevelN 99th percentile read latency
- RocksDB LevelN maximum read latency
- RocksDB compaction time
- RocksDB compaction write amplification
- RocksDB compaction read rate
- RocksDB compaction write rate
- RocksDB compaction key rate
- RocksDB compaction threads
- RocksDB compaction level files
- RocksDB compaction level size
- RocksDB write ahead log rate
- RocksDB write ahead log sync size
- RocksDB flush rate
- RocksDB pending operations
- RocksDB stall time
- RocksDB stalls
- Client operations
- Queued operations
- Scanned and moved objects
- Page faults

33.7 MongoDB WiredTiger Dashboard

The MongoDB WiredTiger dashboard contains statistics on the WiredTiger storage engine for the selected MongoDB host. This dashboard contains the following statistics:

- WiredTiger cache usage
- WiredTiger max cache size
- Memory cached
- Memory available

- WiredTiger transactions
- WiredTiger cache activity
- WiredTiger block activity
- WiredTiger sessions
- WiredTiger concurrency tickets available
- Queued operations
- WiredTiger checkpoint time
- WiredTiger cache eviction
- WiredTiger cache capacity
- WiredTiger cache pages
- WiredTiger log operations
- WiredTiger log records
- Document changes
- Scanned and moved objects
- Page faults

POSTGRESQL DASHBOARDS

34.1 PostgreSQL Overview

This dashboard provides basic information about PostgreSQL hosts.

- *Connected*
- *Version*
- *Shared Buffers*
- *Disk-Page Buffers*
- *Memory Size for each Sort*
- *Disk Cache Size*
- *Autovacuum*
- *PostgreSQL Connections*
- *PostgreSQL Tuples*
- *PostgreSQL Transactions*
- *Temp Files*
- *Conflicts and Locks*
- *Buffers and Blocks Operations*
- *Canceled Queries*
- *Cache Hit Ratio*
- *Checkpoint Stats*
- *PostgreSQL Settings*
- *System Summary*

34.1.1 Connected

Reports whether PMM Server can connect to the PostgreSQL instance.

View all metrics of *PostgreSQL Overview*

34.1.2 Version

The version of the PostgreSQL instance.

View all metrics of *PostgreSQL Overview*

34.1.3 Shared Buffers

Defines the amount of memory the database server uses for shared memory buffers. Default is 128MB. Guidance on tuning is 25% of RAM, but generally doesn't exceed 40%.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: `shared_buffers` <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-SHARED-BUFFERS>

34.1.4 Disk-Page Buffers

The setting `wal_buffers` defines how much memory is used for caching the write-ahead log entries. Generally this value is small (3% of `shared_buffers` value), but it may need to be modified for heavily loaded servers.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: `wal_buffers` <https://www.postgresql.org/docs/current/static/runtime-config-wal.html#GUC-WAL-BUFFERS>

PostgreSQL Server status variables: `shared_buffers` <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-SHARED-BUFFERS>

34.1.5 Memory Size for each Sort

The parameter `work_mem` defines the amount of memory assigned for internal sort operations and hash tables before writing to temporary disk files. The default is 4MB.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: `work_mem` <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-WORK-MEM>

34.1.6 Disk Cache Size

PostgreSQL's `effective_cache_size` variable tunes how much RAM you expect to be available for disk caching. Generally adding `Linux free+cached` will give you a good idea. This value is used by the query planner whether plans will fit in memory, and when defined too low, can lead to some plans rejecting certain indexes.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: `effective_cache_size` <https://www.postgresql.org/docs/current/static/runtime-config-query.html#GUC-EFFECTIVE-CACHE-SIZE>

34.1.7 Autovacuum

Whether autovacuum process is enabled or not. Generally the solution is to vacuum more often, not less.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: autovacuum <https://www.postgresql.org/docs/current/static/routine-vacuuming.html#AUTOVACUUM>

34.1.8 PostgreSQL Connections

Max Connections The maximum number of client connections allowed. Change this value with care as there are some memory resources that are allocated on a per-client basis, so setting `max_connections` higher will generally increase overall PostgreSQL memory usage.

Connections The number of connection attempts (successful or not) to the PostgreSQL server.

Active Connections The number of open connections to the PostgreSQL server.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: max_connections <https://www.postgresql.org/docs/current/static/runtime-config-connection.html#GUC-MAX-CONNECTIONS>

34.1.9 PostgreSQL Tuples

Tuples The total number of rows processed by PostgreSQL server: fetched, returned, inserted, updated, and deleted.

Read Tuple Activity The number of rows read from the database: as returned so fetched ones.

Tuples Changed per 5min The number of rows changed in the last 5 minutes: inserted, updated, and deleted ones.

View all metrics of *PostgreSQL Overview*

34.1.10 PostgreSQL Transactions

Transactions The total number of transactions that have been either been committed or rolled back.

Duration of Transactions Maximum duration in seconds any active transaction has been running.

View all metrics of *PostgreSQL Overview*

34.1.11 Temp Files

Number of Temp Files The number of temporary files created by queries.

Size of Temp files The total amount of data written to temporary files by queries in bytes.

Note: All temporary files are taken into account by these two gauges, regardless of why the temporary file was created (e.g., sorting or hashing), and regardless of the `log_temp_files` setting.

View all metrics of *PostgreSQL Overview*

34.1.12 Conflicts and Locks

Conflicts/Deadlocks The number of queries canceled due to conflicts with recovery in the database (due to dropped tablespaces, lock timeouts, old snapshots, pinned buffers, or deadlocks).

Number of Locks The number of deadlocks detected by PostgreSQL.

View all metrics of *PostgreSQL Overview*

34.1.13 Buffers and Blocks Operations

Operations with Blocks The time spent reading and writing data file blocks by backends, in milliseconds.

Note: Capturing read and write time statistics is possible only if `track_io_timing` setting is enabled. This can be done either in configuration file or with the following query executed on the running system:

```
ALTER SYSTEM SET track_io_timing=ON;
SELECT pg_reload_conf();
```

Buffers The number of buffers allocated by PostgreSQL.

View all metrics of *PostgreSQL Overview*

34.1.14 Canceled Queries

The number of queries that have been canceled due to dropped tablespaces, lock timeouts, old snapshots, pinned buffers, and deadlocks.

Note: Data shown by this gauge are based on the `pg_stat_database_conflicts` view.

View all metrics of *PostgreSQL Overview*

34.1.15 Cache Hit Ratio

The number of times disk blocks were found already in the buffer cache, so that a read was not necessary.

Note: This only includes hits in the PostgreSQL buffer cache, not the operating system's file system cache.

View all metrics of *PostgreSQL Overview*

34.1.16 Checkpoint Stats

The total amount of time that has been spent in the portion of checkpoint processing where files are either written or synchronized to disk, in milliseconds.

View all metrics of *PostgreSQL Overview*

34.1.17 PostgreSQL Settings

The list of all settings of the PostgreSQL server.

View all metrics of *PostgreSQL Overview*

34.1.18 System Summary

This section contains the following system parameters of the PostgreSQL server: CPU Usage, CPU Saturation and Max Core Usage, Disk I/O Activity, and Network Traffic.

View all metrics of *PostgreSQL Overview*

See also:

Configuring PostgreSQL for Monitoring *PostgreSQL*

PostgreSQL Server status variables: wal_buffers <https://www.postgresql.org/docs/current/static/runtime-config-wal.html#GUC-WAL-BUFFERS>

PostgreSQL Server status variables: shared_buffers <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-SHARED-BUFFERS>

PostgreSQL Server status variables: work_mem <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-WORK-MEM>

PostgreSQL Server status variables: effective_cache_size <https://www.postgresql.org/docs/current/static/runtime-config-query.html#GUC-EFFECTIVE-CACHE-SIZE>

PostgreSQL Server status variables: autovacuum <https://www.postgresql.org/docs/current/static/routine-vacuuming.html#AUTOVACUUM>

PostgreSQL Server status variables: max_connections <https://www.postgresql.org/docs/current/static/runtime-config-connection.html#GUC-MAX-CONNECTIONS>

HA DASHBOARDS

35.1 PXC/Galera Cluster Overview Dashboard

- Flow Control Paused Time
- Flow Control Messages Sent
- Writeset Inbound Traffic
- Writeset Outbound Traffic
- Receive Queue
- Send Queue
- Transactions Received
- Transactions Replicated
- Average Incoming Transaction Size
- Average Replicated Transaction Size
- FC Trigger Low Limit
- FC Trigger High Limit
- Sequence Numbers of Transactions
- Average Galera Replication Latency
- Maximum Galera Replication Latency

Part X

Contacting and Contributing

Percona Monitoring and Management is an open source product. We provide ways for anyone to contact developers and experts directly, submit bug reports and feature requests, and contribute to source code directly.

Contacting Developers

Use the [community forum](#) to ask questions about using PMM. Developers and experts will try to help with problems that you experience.

Reporting Bugs

Use the [PMM project in JIRA](#) to report bugs and request features. Please register and search for similar issues before submitting a bug or feature request.

Contributing Source Code

Use the [GitHub repository](#) to explore source code and suggest contributions. You can fork and clone any Percona repositories, but to have your source code patches accepted please sign the Contributor License Agreement (CLA).

Part XI

Terminology Reference

DATA RETENTION

By default, Prometheus stores time-series data for 30 days, and QAN stores query data for 8 days. Depending on available disk space and your requirements, you may need to adjust data retention time. You can control data retention via the *Settings* dashboard.

DATA SOURCE NAME

A database server attribute found on the QAN page. It informs how PMM connects to the selected database.

CHAPTER
THIRTYEIGHT

DSN

See *Data Source Name*

GRAND TOTAL TIME

Grand Total Time.(percent of grand total time) is the percentage of time that the database server spent running a specific query, compared to the total time it spent running all queries during the selected period of time.

%GTT

See *Grand Total Time*

EXTERNAL MONITORING SERVICE

A monitoring service which is not provided by PMM directly. It is bound to a running Prometheus exporter. As soon as such a service is added, you can set up the *Metrics Monitor* to display its graphs.

METRICS

A series of data which are visualized in PMM.

METRICS MONITOR (MM)

Component of *PMM Server* that provides a historical view of metrics critical to a MySQL server instance.

MONITORING SERVICE

A special service which collects information from the database instance where *PMM Client* is installed.

To add a monitoring service, use the `pmm-admin add` command.

See also:

Passing parameters to a monitoring service `pmm.pmm-admin.monitoring-service.pass-parameter`

Percona Monitoring and Management

PMM-ADMIN

A program which changes the configuration of the *PMM Client*. See detailed documentation in the pmm-admin section.

PMM CLIENT

Collects MySQL server metrics, general system metrics, and query analytics data for a complete performance overview.

The collected data is sent to *PMM Server*.

For more information, see *Client/Server Architecture - an Overview*.

PMM DOCKER IMAGE

A docker image which enables installing the PMM Server by using **docker**.

See also:

Installing PMM Server using Docker `run-server-docker`

PMM HOME PAGE

The starting page of the PMM portal from which you can have an overview of your environment, open the tools of PMM, and browse to online resources.

On the PMM home page, you can also find the version number and a button to update your PMM Server (see *PMM Version*).

PMM SERVER

Aggregates data collected by *PMM Client* and presents it in the form of tables, dashboards, and graphs in a web interface.

PMM Server combines the backend API and storage for collected data with a frontend for viewing time-based graphs and performing thorough analysis of your MySQL and MongoDB hosts through a web interface.

Run PMM Server on a host that you will use to access this data.

See also:

PMM Architecture

[Client/Server Architecture - an Overview](#)

PMM SERVER VERSION

If *PMM Server* is installed via Docker, you can check the current PMM Server version by running **docker exec**:

Run this command as root or by using the **sudo** command

```
$ docker exec -it pmm-server head -1 /srv/update/main.yml  
# v1.5.3
```


PMM USER PERMISSIONS FOR AWS

When creating a [IAM user](#) for Amazon RDS DB instance that you intend to monitor in PMM, you need to set all required permissions properly. For this, you may copy the following JSON for your IAM user:

```
{ "Version": "2012-10-17",
  "Statement": [{ "Sid": "Stmt1508404837000",
                  "Effect": "Allow",
                  "Action": [ "rds:DescribeDBInstances",
                              "cloudwatch:GetMetricStatistics",
                              "cloudwatch:ListMetrics"],
                  "Resource": ["*"] },
                 { "Sid": "Stmt1508410723001",
                  "Effect": "Allow",
                  "Action": [ "logs:DescribeLogStreams",
                              "logs:GetLogEvents",
                              "logs:FilterLogEvents" ],
                  "Resource": [ "arn:aws:logs:*:*:log-
↪group:RDSOSMetrics:*" ] }
  ]
}
```

See also:

Creating an IAM user *Creating an IAM user*

PMM VERSION

The version of PMM appears at the bottom of the *PMM server home page*.

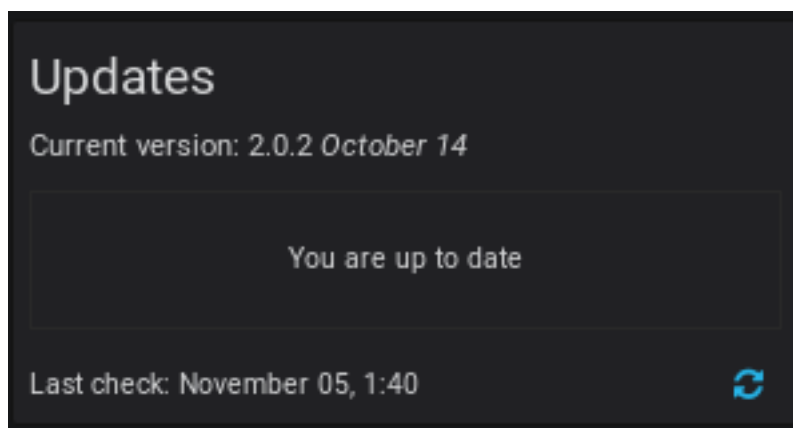


Fig. 53.1: To update your PMM Server, click the *Check for Updates Manually* button located next to the version number.

See also:

Checking the version of PMM Server

PMM Server Version

See *Query Analytics (QAN)*

QUERY ANALYTICS (QAN)

Component of *PMM Server* that enables you to analyze MySQL query performance over periods of time.

QUERY LOAD

The percentage of time that the MySQL server spent executing a specific query.

QUERY METRICS SUMMARY TABLE

An element of *Query Analytics (QAN)* which displays the available metrics for the selected query.

QUERY METRICS TABLE

A tool within QAN which lists metrics applicable to the query selected in the *query summary table*.

QUERY SUMMARY TABLE

A tool within QAN which lists the queries which were run on the selected database server during the *selected time or date range*.

QUICK RANGES

Predefined time periods which are used by QAN to collect metrics for queries. The following quick ranges are available:

- last hour
- last three hours
- last five hours
- last twelve hours
- last twenty four hours
- last five days

SELECTED TIME OR DATE RANGE

A predefined time period (see *Quick ranges*), such as 1 hour, or a range of dates that QAN uses to collect metrics.

TELEMETRY

Percona may collect some **anonymous** statistics about the machine where PMM is running.

Currently, only the following information is gathered:

- PMM Version,
- Installation Method (Docker, AMI, OVF),
- the Uptime,
- PMM Server unique ID.

You may find [here](#) more details about what and how information is gathered, and how to disable telemetry on the *Settings* dashboard, if needed.

VERSION

A database server attribute found on the QAN page. it informs the full version of the monitored database server, as well as the product name, revision and release number.

Part XII

Percona Monitoring and Management Release Notes

PERCONA MONITORING AND MANAGEMENT 2.5.0

Date April 14, 2020

Installation [Installing Percona Monitoring and Management](#)

PMM (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and PostgreSQL performance.

64.1 New Features

- [PMM-5042](#) and [PMM-5272](#): PMM can now connect to MySQL instances by specifying a UNIX socket. This can be done with a new `--socket` option of the `pmm-admin add mysql` command. (Note: Updates to both PMM Client and PMM Server were done to allow UNIX socket connections.)
- [PMM-4145](#): Amazon RDS instance metrics can now be independently enabled/disabled for Basic and/or Enhanced metrics.

64.2 Improvements

- [PMM-5581](#): PMM Server Grafana plugins can now be updated on the command line with the `grafana-cli` command-line utility.
- [PMM-5536](#): Three Grafana plugins were updated to the latest versions: `vertamedia-clickhouse-datasource` to 1.9.5, `grafana-polystat-panel` to 1.1.0, and `grafana-piechart-panel` to 1.4.0.
- [PMM-4252](#): The resolution of the PMM Server favicon image has been improved.

64.3 Bugs Fixed

- [PMM-5547](#): PMM dashboards were failing when presenting data from more than 100 monitored instances (error message `proxy error: context canceled`).
- [PMM-5624](#): Empty charts were being shown in some Node Temperature dashboards.
- [PMM-5637](#): The Data retention value in Settings was incorrectly showing the value as minutes instead of days.
- [PMM-5613](#): Sorting data by Query Time was not working properly in Query Analytics.
- [PMM-5554](#): Totals in charts were inconsistently plotted with different colors across charts.
- [PMM-4919](#): The force option (`--force`) in `pmm-admin config` was not always working.

- **PMM-5351**: The documentation on MongoDB user privileges has been corrected.

Help us improve our software quality by reporting any bugs you encounter using [our bug tracking system](#).

PERCONA MONITORING AND MANAGEMENT 2.4.0

Date March 18, 2020

Installation [Installing Percona Monitoring and Management](#)

PMM ([Percona Monitoring and Management](#)) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and PostgreSQL performance.

65.1 New Features

- [PMM-3387](#): Prometheus custom configuration is now supported by PMM Server. The feature is targeted at experienced users and is done by [adding the base configuration file into the PMM Server container](#) to be parsed and included into the managed Prometheus configuration.
- [PMM-5186](#): Including `-pprof` option in the `pmm-admin summary` command adds pprof debug profiles to the diagnostics data archive
- [PMM-5102](#): The new “Node Details” dashboard now displays data from the hardware monitoring sensors in `hwmon`. The [new dashboard](#) is based on the `hwmon` collector data from the `node_exporter`. Please note that data may be unavailable for some nodes because of the configuration or virtualization parameters

65.2 Improvements

- [PMM-4915](#): The Query Analytics dashboard now shows Time Metrics in the Profile Section as “AVG per query” instead of “AVG per second”
- [PMM-5470](#): Clickhouse query optimized for Query Analytics to improve its speed and reduce the load on the backend
- [PMM-5448](#): The default high and medium metrics resolutions were changed to 1-5-30 and 5-10-60 sec. To reduce the effect of this change on existing installations, systems having the “old” high resolution chosen on the PMM Settings page (5-5-60 sec.) will be automatically re-configured to the medium one during an upgrade. If the resolution was changed to some custom values via API, it will not be affected
- [PMM-5531](#): A healthcheck indicator was implemented for the PMM Server Docker image. It is based on the Docker [HEALTHCHECK](#). This feature can be leveraged as follows:

```
$ docker inspect -f {{.State.Health.Status}}
$ until [ "`docker inspect -f {{.State.Health.Status}} pmm-server`" == "healthy"
↔ ]; do sleep 1; done
```

- [PMM-5489](#): The “Total” line in all charts is now drawn with the same red color for better consistency

- **PMM-5461**: Memory graphs on the node-related dashboards were adjusted to have fixed colors that are more distinguishable from each other
- **PMM-5329**: Prometheus in PMM Server was updated to version 2.16.0. This update has brought several improvements. Among them are significantly reduced memory footprint of the loaded TSDB blocks, lower memory footprint for the compaction process (caused by the more balanced choice of what to buffer during compaction), and improved query performance for the queries that only touch the most recent 2h of data.
- **PMM-5210**: Data Retention is now specified in days instead of seconds on the PMM Settings page. Please note this is the UI-only change, so the actual data retention precision is not changed
- **PMM-5182**: The logs.zip archive available on the PMM Settings page now includes additional self-monitoring information in a separate “client” subfolder. This subfolder contains information collected on the PMM Server and is equivalent to the one collected on a node by the `pmm-admin summary` command.
- **PMM-5112**: The Inventory API List requests now can be filtered by the Node/Service/Agent type

65.3 Bugs Fixed

- **PMM-5178**: Query Detail Section of the Query Analytics dashboard didn't show tables definitions and indexes for the internal PostgreSQL database
- **PMM-5465**: MySQL Instance related dashboards had row names not always matching the actual contents. To fix this, elements were re-ordered and additional rows were added for better matching of the row name and the corresponding elements
- **PMM-5455**: Dashboards from the Insight menu were fixed to work correctly when the low resolution is set on the PMM Settings page
- **PMM-5446**: A number of the Compare Dashboards were fixed to work correctly when the low resolution is set on the PMM Settings page
- **PMM-5430**: MySQL Exporter section on the Prometheus Exporter Status dashboard now collapsed by default to be consistent with other database-related sections
- **PMM-5445**, **PMM-5439**, **PMM-5427**, **PMM-5426**, **PMM-5419**: Labels change (which occurs e.g. when the metrics resolution is changed on the PMM Settings page) was breaking dashboards
- **PMM-5347**: Selecting queries on the Query Analytics dashboard was generating errors in the browser console
- **PMM-5305**: Some applied filters on the Query Analytics dashboard were not preserved after changing the time range
- **PMM-5267**: The Refresh button was not working on the Query Analytics dashboard
- **PMM-5003**: `pmm-admin` list and status use different JSON naming for the same data
- **PMM-5526**: A typo was fixed in the Replication Dashboard description tooltip

Help us improve our software quality by reporting any bugs you encounter using [our bug tracking system](#).

PERCONA MONITORING AND MANAGEMENT 2.3.0

Date February 19, 2020

Percona Monitoring and Management (PMM) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and *PostgreSQL* performance.

For *PMM* install instructions, see *Installing PMM Server* and *Installing PMM Client*.

Note: PMM 2 is designed to be used as a new installation — your existing PMM 1 environment can't be upgraded to this version.

66.1 Improvements and new features

- **PMM-5064** and **PMM-5065:** Starting from this release, users will be able to integrate PMM with an external Alertmanager by specifying the Alertmanager URL and the Alert Rules to be executed inside the PMM server
- **PMM-4954:** Query Analytics dashboard now shows units both in the list of queries in a summary table and in the Details section to ease understanding of the presented data

Note: This feature is for advanced users only at this point

- **PMM-5179:** Relations between metrics are now specified in the Query Analytics Details section
- **PMM-5115:** The CPU frequency and temperature graphs were added to the CPU Utilization dashboard
- **PMM-5394:** A special treatment for the node-related dashboards was implemented for the situations when the data resolution change causes new metrics to be generated for existing nodes and services, to make graphs show continuous lines of the same colors

66.2 Fixed bugs

- **PMM-4620:** The high CPU usage by the pmm-agent process related to MongoDB Query Analytics was fixed
- **PMM-5377:** Singlestats showing percentage had sparklines scaled vertically along with the graph swing, which made it difficult to visually notice the difference between neighboring singlestats.
- **PMM-5204:** Changing resolution on the PMM settings page was breaking some singlestats on the Home and MySQL Overview dashboards

- [PMM-5251](#): Vertical scrollbars on the graph elements were not allowed to do a full scroll, making last rows of the legend unavailable for some graphs
- [PMM-5410](#): The “Available Downtime before SST Required” chart on the PXC/Galera Node Summary dashboard was not showing data because it was unable to use metrics available with different scraping intervals

PERCONA MONITORING AND MANAGEMENT 2.2.2

Date February 4, 2020

Percona Monitoring and Management (PMM) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and *PostgreSQL* performance.

For *PMM* install instructions, see *Installing PMM Server* and *Installing PMM Client*.

Note: PMM 2 is designed to be used as a new installation — your existing PMM 1 environment can't be upgraded to this version.

67.1 Improvements and new features

- **PMM-5321:** The optimization of the Query Analytics parser code for PostgreSQL queries allowed us to reduce the memory resources consumption by 1-5%, and the parsing time of an individual query by 30-40%
- **PMM-5184:** The `pmm-admin summary` command have gained a new `--skip-server` flag which makes it operating in a local-only mode, creating summary file without contacting the PMM Server

67.2 Fixed bugs

- **PMM-5340:** The Scraping Time Drift graph on the Prometheus dashboard was showing wrong values because the actual metrics resolution wasn't taken into account
- **PMM-5060:** Query Analytics Dashboard did not show the row with the last query of the first page, if the number of queries to display was 11

PERCONA MONITORING AND MANAGEMENT 2.2.1

Date January 23, 2020

Percona Monitoring and Management (PMM) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and *PostgreSQL* performance.

For *PMM* install instructions, see *Installing PMM Server* and *Installing PMM Client*.

Note: PMM 2 is designed to be used as a new installation — your existing PMM 1 environment can't be upgraded to this version.

PMM Server version 2.2.0 suffered an unauthenticated denial of service vulnerability (CVE-2020-7920). Any other PMM versions do not carry the same code logic, and are thus unaffected by this issue. **Users who have already deployed PMM Server 2.2.0 are advised to upgrade to version 2.2.1 which resolves this issue.**

68.1 Improvements and new features

- **PMM-5229:** The new RDS Exporter section added to the Prometheus Exporter Status dashboard shows singlestats and charts related to the `rds_exporter`
- **PMM-5228** and **PMM-5238:** The Prometheus dashboard and the Exporters Overview dashboard were updated to include the `rds_exporter` metrics in their charts, allowing better understanding of the impacts of monitoring RDS instances
- **PMM-4830:** The consistency of the applied filters between the Query Analytics and the Overview dashboards was implemented, and now filters selected in QAN will continue to be active after the switch to any of the Overview dashboards available in the Services menu
- **PMM-5235:** The DB uptime singlestats in node rows on the Home dashboard were changed to show minimal values instead of average ones to be consistent with the top row
- **PMM-5127:** The “Search by” bar on the Query Analytics dashboard was renamed to “Filter by” to make its purpose more clear
- **PMM-5131:** The Filter panel on the Query Analytics dashboard now shows the total number of available Labels within the “See all” link, which appears if the Filter panel section shows only top 5 of its Labels

68.2 Fixed bugs

- **PMM-5232:** The pmm-managed component of the PMM Server 2.2.0 is vulnerable to DoS attacks, that could be carried out by anyone who knows the PMM Server IP address (CVE-2020-7920). Versions other than 2.2.0

are not affected.

- **PMM-5226:** The handlebars package was updated to version 4.5.3 because of the Prototype Pollution vulnerability in it (CVE-2019-19919). Please note PMM versions were not affected by this vulnerability, as handlebars package is used as a build dependency only.
- **PMM-5206:** Switching to the Settings dashboard was breaking the visual style of some elements on the Home dashboard
- **PMM-5139:** The breadcrumb panel, which shows all dashboards visited within one session starting from the root, was unable to fully show breadcrumb longer than one line
- **PMM-5212:** The explanatory text was added to the Download PMM Server Logs button in the Diagnostic section of the PMM Settings dashboard, and a link to it was added to the Prometheus dashboard which was the previous place to download logs
- **PMM-5215:** The unneeded `mariadb-libs` package was removed from the PMM Server 2.2.0 OVF image, resulting in both faster updating with the `yum update` command and avoiding dependency conflict messages in the update logs
- **PMM-5216:** PMM Server Upgrade to 2.2.0 was showing Grafana Update Error page with the Refresh button which had to be clicked to start using the updated version
- **PMM-5211:** The “Where do I get the security credentials for my Amazon RDS DB instance” link in the Add AWS RDS MySQL or Aurora MySQL instance dialog was not targeted at the appropriate instruction
- **PMM-5217:** PMM2.x OVF Image memory size was increased from 1Gb to 4Gb with the additional 1Gb swap space because the previous amount was hardly housing the PMM Server, and it wasn’t enough in some cases like performing an upgrade
- **PMM-5271:** LVM logical volumes were wrongly resized on AWS deployment, resulting in “no space left on device” errors
- **PMM-5295:** Innodb Transaction Rollback Rate values on the MySQL InnoDB Details dashboard were calculated incorrectly
- **PMM-5270:** PXC/Galera Cluster Summary dashboard was showing empty Cluster drop-down list, making it impossible to choose the cluster name
- **PMM-4769:** The wrongly named “Timeout value used for retransmitting” `singlestat` on the Network Details dashboard was renamed to “The algorithm used to determine the timeout value” and updated to show the algorithm name instead of a digital code
- **PMM-5260:** Extensive resource consumption by `pmm-agent` took place in case of Query Analytics for PostgreSQL; it was fixed by a number of optimizations in the code, resulting in about 4 times smaller memory usage
- **PMM-5261:** CPU usage charts on all dashboards which contain them have undergone colors update to make `softIRQ` and `Steal` curves better differentiated
- **PMM-5244:** High memory consumption in the PMM Server with a large number of agents sending data simultaneously was fixed by improving bulk data insertion to the ClickHouse database

PERCONA MONITORING AND MANAGEMENT 2.2.0

Date December 24, 2019

PMM (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and *PostgreSQL* performance. You can run *PMM* in your own environment for maximum security and reliability. It provides thorough time-based analysis for *MySQL*, *MongoDB*, and *PostgreSQL* servers to ensure that your data works as efficiently as possible.

Main improvements in this release are:

- Alternative installation methods available for *PMM 1.x* are re-implemented for *PMM 2*: now *PMM Server* can be installed as a virtual appliance, or run using *AWS Marketplace*
- *AWS RDS* and remote instances monitoring re-added in this release include *AWS RDS MySQL / Aurora MySQL* instances, and remote *PostgreSQL*, *MySQL*, *MongoDB*, and *ProxySQL* ones
- The new *Settings* dashboard allows configuring *PMM Server* via the graphical interface

For *PMM* install instructions, see *Installing PMM Server* and *Installing PMM Client*.

Note: *PMM 2* is designed to be used as a new installation — your existing *PMM 1* environment can't be upgraded to this version.

69.1 Improvements and new features

- **PMM-4575:** The new *PMM Settings* dashboard allows users to configure various *PMM Server* options: setting metrics resolution and data retention, enabling or disabling send usage data statistics back to Percona and checking for updates; this dashboard is now the proper place to upload your public key for the *SSH* login and to download *PMM Server* logs for diagnostics
- **PMM-4907** and **PMM-4767:** The user's *AMI Instance ID* is now used to setup running *PMM Server* using *AWS Marketplace* as an additional verification on the user, based on the *Amazon Marketplace* rules
- **PMM-4950** and **PMM-3094:** Alternative *AWS partitions* are now supported when adding an *AWS RDS MySQL* or *Aurora MySQL Instance* to *PMM*
- **PMM-4976:** Home dashboard clean-up: “Systems under monitoring” and “Network IO” singlestats were refined to be based on the `host` variable; also avoiding using color as an indicator of state; “All” row elements were relinked to the “Nodes Overview” dashboard with regards to the selected host.
- **PMM-4800:** The `pmm-admin add mysql` command has been modified to make help text more descriptive: now when you enable tablestats you will get more detail on if they're enabled for your environment and where you stand with respect to the auto-disable limit

- [PMM-4969](#): Update Grafana to version 6.5.1
- [PMM-5053](#): A tooltip was added to the Head Block graph on the Prometheus dashboard
- [PMM-5068](#): Drill-down links were added to the Node Summary dashboard graphs
- [PMM-5050](#): Drill-down links were added to the graphs on all Services Compare dashboards
- [PMM-5037](#): Drill-down links were added to all graphs on the Services Overview dashboards
- [PMM-4988](#): Filtering in Query Analytics have undergone improvements to make group selection more intuitive: Labels unavailable under the current selection are shown as gray/disabled, and the percentage values are dynamically recalculated to reflect Labels available within the currently applied filters
- [PMM-4966](#): All passwords are now substituted with asterisk signs in the exporter logs for security reasons when not in debug mode
- [PMM-527](#): `node_exporter` is now providing hardware monitoring information such as CPU temperatures and fan statuses; while this information is being collected by PMM Server, it will not be shown until a dedicated dashboard is added in a future release
- [PMM-3198](#): Instead of showing All graphs for all services by default, MySQL Command/Handler Counters Compare dashboard now shows the pre-defined set of ten most informative ones, to reduce load on PMM Server at its first open

69.2 Fixed bugs

- [PMM-4978](#): The “Top MySQL Questions” singlestat on the MySQL Instances Overview dashboard was changed to show ops instead of percentage
- [PMM-4917](#): The “Systems under monitoring” and “Monitored DB Instances” singlestats on the Home dashboard now have a sparkline to make situation more clear with recently shut down nodes/instances
- [PMM-4979](#): Set decimal precision 2 for all the elements, including charts and singlestats, on all dashboards
- [PMM-4980](#): Fix “Load Average” singlestat on the Node Summary dashboard to show decimal value instead of percent
- [PMM-4981](#): Disable automatic color gradient in filled graphs on all dashboards
- [PMM-4941](#): Some charts were incorrectly showing empty fragments with high time resolution turned on
- [PMM-5022](#): Fix outdated drill-down links on the Prometheus Exporters Overview and Nodes Overview dashboards
- [PMM-5023](#): Make the All instances uptime singlestat on the Home dashboard to show Min values instead of Avg
- [PMM-5029](#): Option to upload dashboard snapshot to Percona was disappearing after upgrade to 2.1.x
- [PMM-4946](#): Rename singlestats on the Home dashboard for better clarity: “Systems under monitoring” to “Nodes under monitoring” and “Monitored DB Instances” to “Monitored DB Services”, and make the last one to count remote DB instances also
- [PMM-5015](#): Fix format of Disk Page Buffers singlestat on the Compare dashboard for PostgreSQL to have two digits precision for the consistency with other singlestats
- [PMM-5014](#): LVM logical volumes were wrongly sized on a new AWS deployment, resulting in “no space left on device” errors.

- **PMM-4804:** Incorrect parameters validation required both `service-name` and `service-id` parameters of the `pmm-admin remove` command to be presented, while the command itself demanded only one of them to identify the service.
- **PMM-3298:** Panic errors were present in the `rds_exporter` log after adding an RDS instance from the second AWS account
- **PMM-5089:** The `serialize-javascript` package was updated to version 2.1.1 because of the possibility of regular expressions cross-site scripting vulnerability in it (CVE-2019-16769). Please note PMM versions were not affected by this vulnerability, as `serialize-javascript` package is used as a build dependency only.
- **PMM-5149:** Disk Space `singlestat` was unable to show data for RDS instances because of not taking into account sources with unknown filesystem type

PERCONA MONITORING AND MANAGEMENT 2.1.0

Date November 11, 2019

PMM (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and *PostgreSQL* performance. You can run *PMM* in your own environment for maximum security and reliability. It provides thorough time-based analysis for *MySQL*, *MongoDB*, and *PostgreSQL* servers to ensure that your data works as efficiently as possible.

For install instructions, see `deploy-pmm`.

Note: *PMM* 2 is designed to be used as a new installation — please don't try to upgrade your existing *PMM* 1 environment.

70.1 Improvements and new features

- **PMM-4063:** Update QAN filter panel to show only labels available for selection under currently applied filters
- **PMM-815:** Latency Detail graph added to the MongoDB Instance Summary dashboard
- **PMM-4768:** Disable heavy-load collectors automatically when there are too many tables
- **PMM-4821:** Use color gradient in filled graphs on all dashboards
- **PMM-4733:** Add more log and config files to the downloadable `logs.zip` archive
- **PMM-4672:** Use integer percentage values in QAN filter panel
- **PMM-4857:** Update tooltips for all MongoDB dashboards
- **PMM-4616:** Rename column in the Query Details section in QAN from Total to Sum
- **PMM-4770:** Use Go 1.12.10
- **PMM-4780:** Update Grafana to version 6.4.1
- **PMM-4918:** Update Grafana plugins to newer versions, including the `clickhouse-datasource` plugin

70.2 Fixed bugs

- **PMM-4935:** Wrong instance name displayed on the MySQL Instance Summary dashboard due to the incorrect string crop
- **PMM-4916:** Wrong values are shown when changing the time range for the Node Summary Dashboard in case of remote instances

- [PMM-4895](#) and [PMM-4814](#): The update process reports completion before it is actually done and therefore some dashboards, etc. may not be updated
- [PMM-4876](#): PMM Server access credentials are shown by the `pmm-admin status` command instead of hiding them for security reasons
- [PMM-4875](#): PostgreSQL error log gets flooded with warnings when `pg_stat_statements` extension is not installed in the database used by PMM Server or when PostgreSQL user is unable to connect to it
- [PMM-4852](#): Node name has an incorrect value if the Home dashboard opened after QAN
- [PMM-4847](#): Drilldowns from the Environment Overview dashboard doesn't show data for the pre-selected host
- [PMM-4841](#) and [PMM-4845](#): `pg_stat_statement` QAN Agent leaks database connections
- [PMM-4831](#): Clean-up representation of selectors names on MySQL-related dashboards for a better consistency
- [PMM-4824](#): Incorrectly calculated singlestat values on MySQL Instances Overview dashboard
- [PMM-4819](#): In case of the only one monitored host, its uptime is shown as a smaller value than the all hosts uptime due to the inaccurate rounding
- [PMM-4816](#): Set equal thresholds to avoid confusing singlestat color differences on a Home dashboard
- [PMM-4718](#): Labels are not fully displayed in the filter panel of the Query Details section in QAN
- [PMM-4545](#): Long queries are not fully visible in the Query Examples section in QAN

Help us improve our software quality by reporting any Percona Monitoring and Management bugs you encounter using our [bug tracking system](#).

PERCONA MONITORING AND MANAGEMENT 2.0.1

Date October 9, 2019

PMM (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and *PostgreSQL* performance. You can run *PMM* in your own environment for maximum security and reliability. It provides thorough time-based analysis for *MySQL*, *MongoDB*, and *PostgreSQL* servers to ensure that your data works as efficiently as possible.

For install instructions, see `deploy-pmm`.

Note: *PMM* 2 is designed to be used as a new installation — please don't try to upgrade your existing *PMM* 1 environment.

71.1 Improvements

- **PMM-4779:** Securely share dashboards with Percona
- **PMM-4735:** Keep one old slowlog file after rotation
- **PMM-4724:** Alt+click on check updates button enables force-update
- **PMM-4444:** Return “what’s new” URL with the information extracted from the `pmm-update` package changelog

71.2 Fixed bugs

- **PMM-4758:** Remove Inventory rows from dashboards
- **PMM-4757:** `qan_mysql_perfschema_agent` failed querying `events_statements_summary_by_digest` due to data types conversion
- **PMM-4755:** Fixed a typo in the InnoDB AHI Miss Ratio formula
- **PMM-4749:** Navigation from Dashboards to QAN when some Node or Service was selected now applies filtering by them in QAN
- **PMM-4742:** General information links were updated to go to *PMM* 2 related pages
- **PMM-4739:** Remove request instances list
- **PMM-4734:** A fix was made for the collecting `node_name` formula at MySQL Replication Summary dashboard
- **PMM-4729:** Fixes were made for formulas on MySQL Instances Overview

- [PMM-4726](#): Links to services in MongoDB singlestats didn't show Node name
- [PMM-4720](#): `machine_id` could contain trailing `\n`
- [PMM-4640](#): It was not possible to add MongoDB remotely if password contained a # symbol

Help us improve our software quality by reporting any Percona Monitoring and Management bugs you encounter using our [bug tracking system](#).

PERCONA MONITORING AND MANAGEMENT 2.0.0

Date September 19, 2019

PMM (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and *PostgreSQL* performance. You can run *PMM* in your own environment for maximum security and reliability. It provides thorough time-based analysis for *MySQL*, *MongoDB*, and *PostgreSQL* servers to ensure that your data works as efficiently as possible.

For install instructions, see [deploy-pmm](#).

Note: *PMM 2* is designed to be used as a new installation — please don't try to upgrade your existing *PMM 1* environment.

The new *PMM2* introduces a number of enhancements and additional feature improvements, including:

- Detailed query analytics and filtering technologies which enable you to identify issues faster than ever before.
- A better user experience: Service-level dashboards give you immediate access to the data you need.
- The new addition of PostgreSQL query tuning.
- Enhanced security protocols to ensure your data is safe.
- Our new API allows you to extend and interact with third-party tools.

More details about new and improved features available within the release can be found in [the correspondent blog post](#).

Help us improve our software quality by reporting any Percona Monitoring and Management bugs you encounter using our [bug tracking system](#).

Part XIII

Frequently Asked Questions

- *How can I contact the developers?*
- *What are the minimum system requirements for PMM?*
- *How to control data retention for PMM?*
- *How often are nginx logs in PMM Server rotated?*
- *What privileges are required to monitor a MySQL instance?*
- *Can I monitor multiple service instances?*
- *Can I rename instances?*
- *Can I add an AWS RDS MySQL or Aurora MySQL instance from a non-default AWS partition?*
- *How to troubleshoot communication issues between PMM Client and PMM Server?*
- *What resolution is used for metrics?*
- *How to set up Alerting in PMM?*
- *How to use a custom Prometheus configuration file inside of a PMM Server?*

HOW CAN I CONTACT THE DEVELOPERS?

The best place to discuss PMM with developers and other community members is the [community forum](#).

If you would like to report a bug, use the [PMM project in JIRA](#).

WHAT ARE THE MINIMUM SYSTEM REQUIREMENTS FOR PMM?

PMM Server

Any system which can run Docker version 1.12.6 or later.

It needs roughly 1 GB of storage for each monitored database node with data retention set to one week.

Note: By default, *retention* is set to 30 days for Metrics Monitor and for Query Analytics. Also consider *disabling table statistics*, which can greatly decrease Prometheus database size.

Minimum memory is 2 GB for one monitored database node, but it is not linear when you add more nodes. For example, data from 20 nodes should be easily handled with 16 GB.

PMM Client

Any modern 64-bit Linux distribution. It is tested on the latest versions of Debian, Ubuntu, CentOS, and Red Hat Enterprise Linux.

Minimum 100 MB of storage is required for installing the PMM Client package. With good constant connection to PMM Server, additional storage is not required. However, the client needs to store any collected data that it is not able to send over immediately, so additional storage may be required if connection is unstable or throughput is too low.

HOW TO CONTROL DATA RETENTION FOR PMM?

By default, both Prometheus and QAN store time-series data for 30 days.

Depending on available disk space and your requirements, you may need to adjust data retention time.

You can control data retention by the following way.

1. Select the *PMM Settings* dashboard in the main menu.

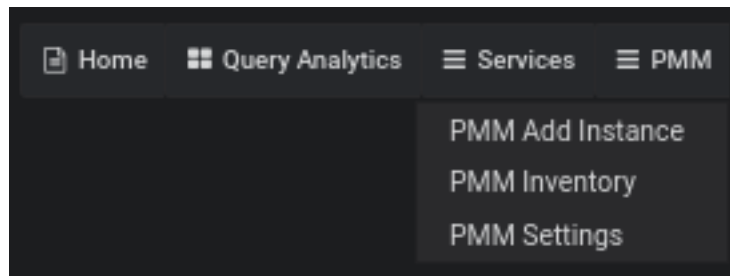


Fig. 75.1: Choosing the *PMM Settings* menu entry

2. In the *Settings* section, enter new data retention value in days.
3. Click the *Apply changes* button.

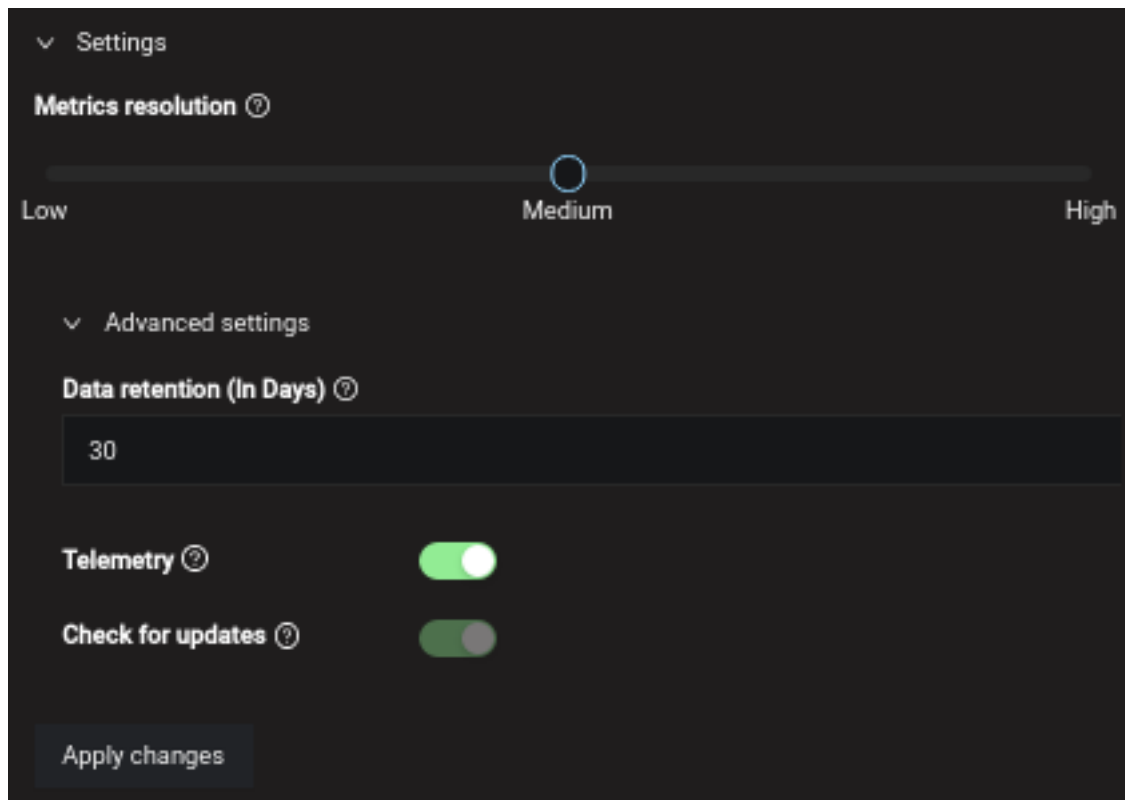


Fig. 75.2: Entering data retention on the *Settings dashboard*

HOW OFTEN ARE NGINX LOGS IN PMM SERVER ROTATED?

PMM Server runs `logrotate` to rotate nginx logs on a daily basis and keep up to 10 latest log files.

**WHAT PRIVILEGES ARE REQUIRED TO MONITOR A MYSQL
INSTANCE?**

See `pmm.conf-mysql.user-account.creating`.

CAN I MONITOR MULTIPLE SERVICE INSTANCES?

Yes, you can add multiple instances of MySQL or some other service to be monitored from one PMM Client. In this case, you will need to provide a distinct port and socket for each instance, and specify a unique name for each instance (by default, it uses the name of the PMM Client host).

For example, if you are adding complete MySQL monitoring for two local MySQL servers, the commands could look similar to the following:

```
$ sudo pmm-admin add mysql --username root --password root instance-01 127.0.0.1:3001
$ sudo pmm-admin add mysql --username root --password root instance-02 127.0.0.1:3002
```

For more information, run

```
$ pmm-admin add mysql --help
```


CAN I RENAME INSTANCES?

You can remove any monitoring instance as described in *Removing monitoring services with `pmm-admin remove`* and then add it back with a different name.

When you remove a monitoring service, previously collected data remains available in Grafana. However, the metrics are tied to the instance name. So if you add the same instance back with a different name, it will be considered a new instance with a new set of metrics. So if you are re-adding an instance and want to keep its previous data, add it with the same name.

CAN I ADD AN AWS RDS MYSQL OR AURORA MYSQL INSTANCE FROM A NON-DEFAULT AWS PARTITION?

By default the RDS discovery works with the default `aws` partition. But you can switch to special regions, like the GovCloud one, with the alternative AWS partitions (e.g. `aws-us-gov`) adding them to the *Settings* via the PMM Server API:

The screenshot shows a REST client interface for a POST request to `/v1/Settings/Change`. The request body is a JSON object with the following structure:

```
{
  "enable_telemetry": true,
  "disable_telemetry": true,
  "metrics_resolutions": {
    "hr": "string",
    "mr": "string",
    "lr": "string"
  },
  "data_retention": "string",
  "ssh_key": "string",
  "aws_partitions": [
    "string"
  ]
}
```

The parameter content type is set to `application/json`.

You can specify any of them instead of the `aws` default value, or use several of them, with the JSON Array syntax: `["aws", "aws-cn"]`.

HOW TO TROUBLESHOOT COMMUNICATION ISSUES BETWEEN PMM CLIENT AND PMM SERVER?

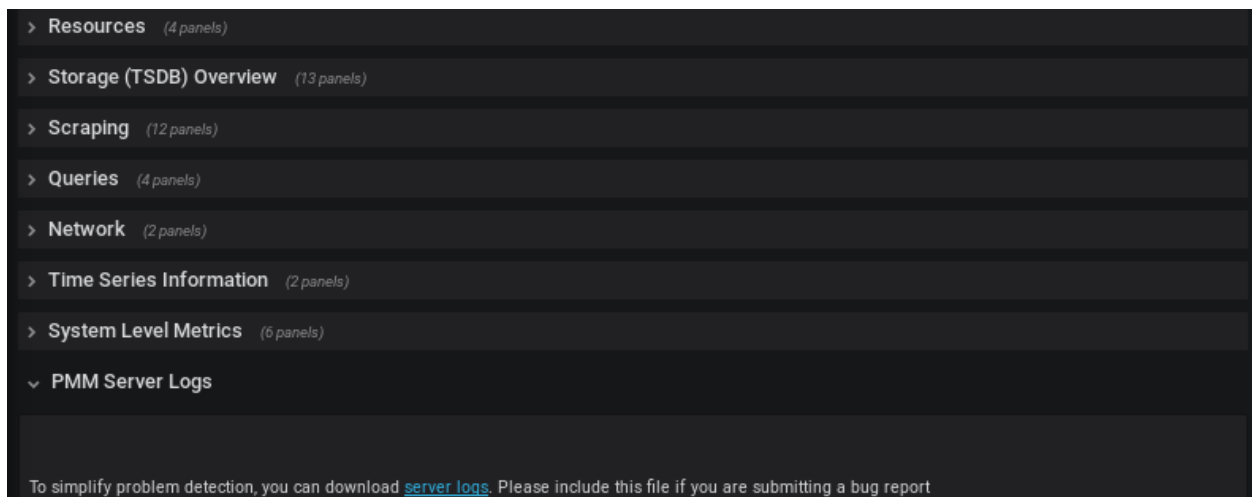
Broken network connectivity may be caused by rather wide set of reasons. Particularly, when *using Docker*, the container is constrained by the host-level routing and firewall rules. For example, your hosting provider might have default *iptables* rules on their hosts that block communication between PMM Server and PMM Client, resulting in *DOWN* targets in Prometheus. If this happens, check firewall and routing settings on the Docker host.

Also PMM is able to generate a set of diagnostics data which can be examined and/or shared with Percona Support to solve an issue faster. You can get collected logs from PMM Client using the `pmm-admin summary` command.

The logs archive obtained in this way includes PMM Client logs and also logs which were received from the PMM Server, stored separately in the `client` and `server` folders. The `server` folder also contains its own `client` subfolder with the self-monitoring client information collected on the PMM Server.

Note: Starting from PMM 2.4.0 there is an additional flag that allows to fetch `pprof` debug profiles and add them to the diagnostics data. To do it, run `pmm-admin summary --pprof`.

Obtaining logs from PMM Server can be done *by specifying the ‘‘https://<address-of-your-pmm-server>/logs.zip’’* URL, or by clicking the `server logs` link on the [Prometheus dashboard](#):



The logs archive obtained in this way includes diagnostics information gathered from the PMM Server, and the `client` subfolder with the self-monitoring client information collected on the PMM Server.

WHAT RESOLUTION IS USED FOR METRICS?

MySQL metrics are collected with different resolutions (5 seconds, 10 seconds, and 60 seconds by default). Linux and MongoDB metrics are collected with 1 second resolution.

In case of bad network connectivity between PMM Server and PMM Client or between PMM Client and the database server it is monitoring, scraping every second may not be possible when latency is higher than 1 second.

You can change the minimum resolution for metrics by the following way:

1. Select the *PMM Settings* dashboard in the main menu.

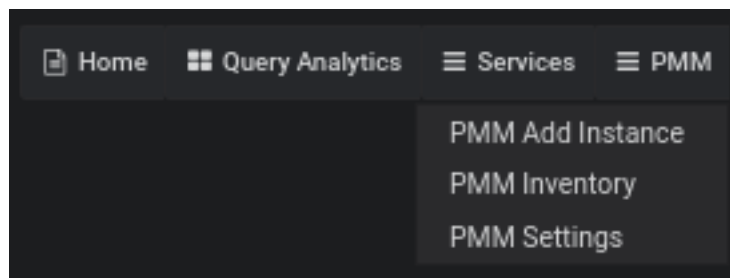


Fig. 82.1: Choosing the *PMM Settings* menu entry

2. In the *Settings* section, choose proper metrics resolution with the slider. The tooltip of the slider will show you actual resolution values.
3. Click the *Apply changes* button.

Note: Consider increasing minimum resolution when PMM Server and PMM Client are on different networks, or when *Adding an Amazon RDS MySQL, Aurora MySQL, or Remote Instance*.

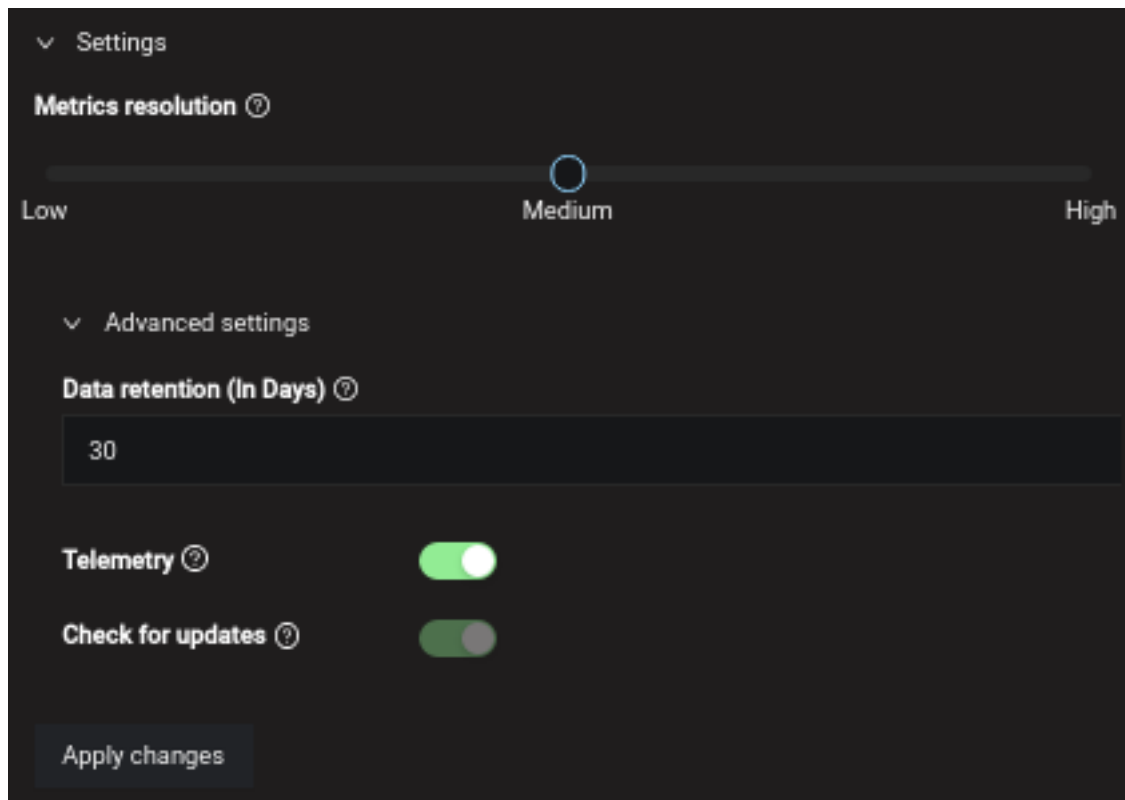


Fig. 82.2: Choosing metrics resolution on the *Settings dashboard*

HOW TO SET UP ALERTING IN PMM?

You can make PMM Server trigger alerts when your monitored service reaches some thresholds in two ways:

- using [Grafana Alerting feature](#),
- using external [Alertmanager](#) (a high-performance solution developed by the Prometheus project to handle alerts sent by Prometheus).

Both options can be considered advanced features and require knowledge of third-party documentation.

Either with Grafana Alerting or with Alertmanager you need to configure some alerting rule to define conditions under which the alert should be triggered, and the channel used to send the alert (e.g. email).

Grafana Alerts are already integrated into PMM Server and may be simpler to get set up, while Alertmanager allows the creation of more sophisticated alerting rules and can be easier to manage installations with a large number of hosts; this additional flexibility comes at the expense of simplicity and requires advanced knowledge of Alertmanager rules. Currently Percona cannot offer support for creating custom rules so you should already have a working Alertmanager instance prior to using this feature, however we are working hard to bring an integrated Alertmanager solution to make rule generation easy!

How to set up Alerting with Grafana

Alerting in Grafana allows attaching rules to your dashboard panels. Details about Grafana Alerting Engine and Rules can be found in the [official documentation](#). Setting it up and running within PMM Server is covered [by the following blog post](#).

How to integrate Alertmanager with PMM

PMM allows you to integrate Prometheus with an external Alertmanager. Configuration is done on the [PMM Settings dashboard](#). The Alertmanager section in it allows specifying the URL of the Alertmanager to serve your PMM alerts, as well as your [alerting rules in the YAML configuration format](#).

More details on the Alertmanager and its alerting rules can be found in the [official Alertmanager documentation](#), which also provides plain examples of the [alerting rules](#).

HOW TO USE A CUSTOM PROMETHEUS CONFIGURATION FILE INSIDE OF A PMM SERVER?

Normally PMM Server fully manages [Prometheus configuration file](#). Still, some users may want to be able to change generated configuration to add additional scrape jobs, configure remote storage, etc.

Starting from the version 2.4.0, when pmm-managed starts the Prometheus file generation process, it tries to load the `/srv/prometheus/prometheus.base.yml` file first, to use it as a base for the `prometheus.yml` if present and can be parsed.

Note: The `prometheus.yml` file can be regenerated by restarting the PMM Server container, or by the `SetSettings` [API call](#) with an empty body.

You can find more details about using a custom Prometheus configuration file with PMM [in a separate blog post](#).