PERCONA

www.percona.com

Percona Monitoring and Management Documentation

Release 1.1.3

April 21, 2017

Percona Monitoring and Management (PMM) is an open-source platform for managing and monitoring MySQL and MongoDB 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 and MongoDB servers to ensure that your data works as efficiently as possible.

CHAPTER

ONE

BASICS

1.1 Percona Monitoring and Management Architecture

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

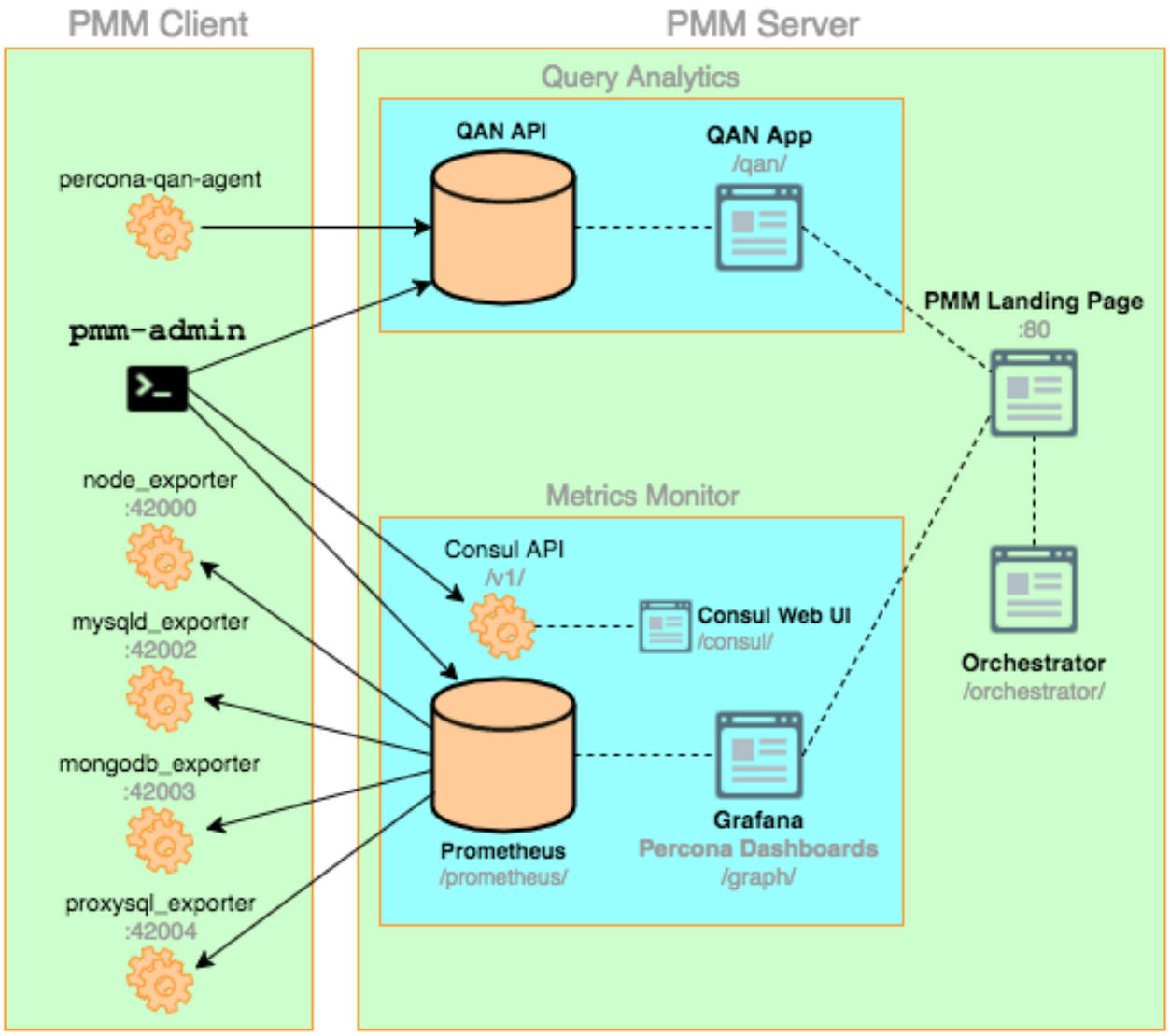
- *PMM Client* is installed on every database host that you want to monitor. It collects server metrics, general system metrics, and query analytics data for a complete performance overview. Collected data is sent to *PMM Server*.
- *PMM Server* is the central part of PMM that aggregates collected data 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, internal structure is important.

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 diagram illustrates how PMM is currently structured:



1.1.1 PMM Client

PMM Client packages are available for most popular Linux distributions:

- DEB for Debian-based distributions (including Ubuntu and others)
- RPM for Red Hat Enterprise Linux derivatives (including CentOS, Oracle Linux, Amazon Linux, and others)

There are also generic tarball binaries that can be used on any Linux system.

For more information, see *Installing PMM Client*.

PMM Client packages 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 *Managing PMM Client*.
- `percona-qan-agent` is a service that manages the Query Analytics (QAN) agent as it collects query performance data. It also connects with QAN API in *PMM Server* and sends over collected data.

- `node_exporter` is a Prometheus exporter that collects general system metrics. For more information, see https://github.com/prometheus/node_exporter.
- `mysqld_exporter` is a Prometheus exporter that collects MySQL server metrics. For more information, see https://github.com/percona/mysqld_exporter.
- `mongodb_exporter` is a Prometheus exporter that collects MongoDB server metrics. For more information, see https://github.com/percona/mongodb_exporter.
- `proxysql_exporter` is a Prometheus exporter that collects ProxySQL performance metrics. For more information, see https://github.com/percona/proxysql_exporter.

1.1.2 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
- Open Virtual Appliance (OVA) that you can run in VirtualBox or another hypervisor
- Amazon Machine Image (AMI) that you can run via Amazon Web Services (AWS)

For more information, see *Running PMM Server*.

PMM Server consists of the following tools:

- **Query Analytics** (QAN) 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 `percona-qan-agent` running on a *PMM Client*.
  - **QAN Web App** is a web application for visualizing collected Query Analytics data.
- **Metrics Monitor** (MM) 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 colleted metrics. For more information, see Prometheus Docs [1].
    - **Consul** provides an API that a *PMM Client* can use to remotely list, add, and remove hosts for Prometheus. It also stores monitoring metadata. For more information, see Consul Docs [2].

      Warning: Although the Consul web UI is accessible, do not make any changes to the configuration.
  - **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 [3].
    - **Percona Dashboards** is a set of dashboards for *Grafana* developed by Percona.
- **Orchestrator** is a MySQL replication topology management and visualization tool. For more information, see: Orchestrator Manual [4].

All tools can be accessed from the *PMM Server* web interface (landing page). For more information, see *Using the Percona Monitoring and Management Platform*.

[1] https://prometheus.io/docs/introduction/overview/
[2] https://www.consul.io/docs/
[3] http://docs.grafana.org/
[4] https://github.com/outbrain/orchestrator/wiki/Orchestrator-Manual

1.1.3 Deployment Scenarios

PMM is designed to be scalable for various environments. Depending on the size and complexity of your infrastructure, you can deploy it in several ways.

Simple Scenario

If you have just one MySQL or MongoDB server, you can install and run both modules (*PMM Client* and *PMM Server*) on this one database host.

Typical Scenario

It is more typical to have several MySQL and MongoDB server instances distributed over different hosts. In this case, you can run *PMM Server* on a dedicated monitoring host, and install *PMM Client* on every database host that you want to monitor. Data from hosts will be aggregated on the PMM Server.

References

1.2 Deploying Percona Monitoring and Management

The following procedure describes how to properly deploy PMM:

1. *Run PMM Server* on the host that will be used to access collected data, view time-based graphs, and carry out performance analysis.

   The following options are available:

   - *Run PMM Server using Docker*
   - *Run PMM Server using VirtualBox*
   - *Run PMM Server using Amazon Machine Image (AMI)*

2. *Install PMM Client* on every MySQL and MongoDB instance that you want to monitor.

   Percona provides *PMM Client* packages for automatic installation from software repositories on the most popular Linux distributions:

   - *Install PMM Client on Debian or Ubuntu*
   - *Install PMM Client on Red Hat or CentOS*

3. *Connect PMM Client to PMM Server*
4. *Start data collection*

1.2.1 Removing and Upgrading

For information about removing and upgrading PMM, see:

- *Removing PMM Server*
- *Upgrading PMM Server*
- *Removing PMM Client*
- *Upgrading PMM Client*

Running PMM Server

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 it on a host that you will use to access this data.

There are several options available to run *PMM Server*:

- *Run PMM Server using Docker*
- *Run PMM Server using VirtualBox*
- *Run PMM Server using Amazon Machine Image (AMI)*

Verifying PMM Server

When you run *PMM Server*, you should be able to access the PMM web interface using the IP address of the host where the container is running. For example, if it is running on 192.168.100.1 with default port 80, you should be able to access the following:

Component	URL
PMM landing page	`http://192.168.100.1`
Query Analytics (QAN web app)	`http://192.168.100.1/qan/`
Metrics Monitor (Grafana)	`http://192.168.100.1/graph/` User name: `admin` Password: `admin`
Orchestrator	`http://192.168.100.1/orchestrator`

Running PMM Server Using Docker Docker images of *PMM Server* are hosted publicly at https://hub.docker.com/r/percona/pmm-server/. If you want to run *PMM Server* from a Docker image, the host must be able to run Docker containers and have network access.

For more information about using Docker, see the Docker Docs.

Note: Make sure that you are using the latest version of Docker. The ones provided via `apt` and `yum` may be outdated and cause errors.

Note: When using the `pmm-server` image, use a specific version tag instead of the `latest` tag. The current stable version is `1.1.3`.

Note: By default, Docker will pull the image from DockerHub if it is not available locally.

Step 1. Create a PMM Data Container To create a container for persistent PMM data, run the following command:

```
$ docker create \
   -v /opt/prometheus/data \
   -v /opt/consul-data \
   -v /var/lib/mysql \
   -v /var/lib/grafana \
   --name pmm-data \
   percona/pmm-server:1.1.3 /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:1.1.3` is the name and version tag of the image to derive the container from.
- `/bin/true` is the command that the container runs.

Step 2. Create and Run the PMM Server Container To run *PMM Server*, use the following command:

```
$ docker run -d \
   -p 80:80 \
   --volumes-from pmm-data \
   --name pmm-server \
   --restart always \
   percona/pmm-server:1.1.3
```

The previous command does the following:

- The `docker run` command instructs the `docker` daemon to run a container from an image.
- The `-d` option starts the container in detached mode (that is, in the background).
- The `-p` option maps the port for accessing the *PMM Server* web UI. For example, if port 80 is not available, you can map the landing page to port 8080 using `-p 8080:80`.
- The `--volumes-from` option mounts volumes from the `pmm-data` container (see *Step 1. Create a PMM Data 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-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:1.1.3` is the name and version tag of the image to derive the container from.

Next Steps *Verify that PMM Server is running* by connecting to the PMM web interface using the IP address of the host running the container, then *install PMM Client* on all database hosts that you want to monitor.

Running PMM Server Using VirtualBox Images Percona provides an *Open Virtual Appliance* (OVA) of *PMM Server*, which you can run in most popular hypervisors. The following procedure describes how to run the appliance in VirtualBox:

1. Download the OVA.

   The latest version is available at https://www.percona.com/redir/downloads/TESTING/pmm/.

2. Import the appliance.

   (a) Open the **File** menu and click **Import Appliance**.

(b) Specify the path to the OVA and click **Continue**.

(c) Select **Reinitialize the MAC address of all network cards** and click **Import**.

3. Configure network settings to make the appliance accessible from other hosts in your network.

   Note: All database hosts must be in the same network as *PMM Server*.

   If you are running the appliance on a host with properly configured network settings, select **Bridged Adapter** in the **Network** section of the appliance settings.

4. Start the *PMM Server* appliance.

   If it was assigned an IP address on the network, the URL for accessing PMM will be printed in the console window.

Running on the Command Line Instead of using the VirtualBox GUI, you can do everything on the command line. Use the `VBoxManage` command to import, configure, and start the appliance.

The following script imports the *PMM Server* appliance from `PMM-Server-2017-01-24.ova` and configures it to bridge the *en0* adapter from the host. Then the script routes console output from the appliance to `/tmp/pmm-server-console.log`. This is done because the script then starts the appliance in headless mode (that is, without the console). To get the IP address for accessing PMM, the script waits for 1 minute until the appliance boots up and returns the lines with the IP address from the log file.

```
# Import image
VBoxManage import PMM-Server-2017-01-24.ova

# Modify NIC settings if needed
VBoxManage list bridgedifs
VBoxManage modifyvm 'PMM Server [2017-01-24]' --nic1 bridged --bridgeadapter1 'en0: Wi-Fi (AirPort)'

# Log console output into file
VBoxManage modifyvm 'PMM Server [2017-01-24]' --uart1 0x3F8 4 --uartmode1 file /tmp/pmm-server-consol

# Start instance
VBoxManage startvm --type headless 'PMM Server [2017-01-24]'

# Wait for 1 minute and get IP address from the log
sleep 60
grep cloud-init /tmp/pmm-server-console.log
```

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.

Running PMM Server Using Amazon Machine Images Percona provides public Amazon Machine Images (AMI) with *PMM Server* in all regions where Amazon Web Services (AWS) is available. You can launch an instance using the web console for the corresponding image:

Region	AMI ID
US East (N. Virginia)	ami-fdea77eb
US East (Ohio)	ami-ef7d5a8a
US West (N. California)	ami-b70d29d7
US West (Oregon)	ami-df4bd7bf
Canada (Central)	ami-afc07ccb
EU (Ireland)	ami-761b1d10
EU (Frankfurt)	ami-8826fbe7
EU (London)	ami-fff1e59b
Asia Pacific (Singapore)	ami-fd8d359e
Asia Pacific (Sydney)	ami-5fb5bc3c
Asia Pacific (Seoul)	ami-4e26f420
Asia Pacific (Tokyo)	ami-b3b990d4
Asia Pacific (Mumbai)	ami-bd186ad2
South America (São Paulo)	ami-dd4d20b1

Running from Command Line

1. Launch the *PMM Server* instance using the `run-instances` command for the corresponding region and image. For example:

```
aws ec2 run-instances \
  --image-id ami-9a0acb8c \
  --security-group-ids sg-3b6e5e46 \
  --instance-type t2.micro \
  --subnet-id subnet-4765a930 \
  --region us-east-1 \
  --key-name SSH-KEYNAME
```

Note: Providing the public SSH key is optional. Specify it if you want SSH access to *PMM Server*.

2. Set a name for the instance using the `create-tags` command. For example:

```
aws ec2 create-tags  \
  --resources i-XXXX-INSTANCE-ID-XXXX \
  --region us-east-1 \
  --tags Key=Name,Value=OWNER_NAME-pmm
```

3. Get the IP address for accessing *PMM Server* from console output using the `get-console-output` command. For example:

```
aws ec2 get-console-output \
  --instance-id i-XXXX-INSTANCE-ID-XXXX \
  --region us-east-1 \
  --output text \
  | grep cloud-init
```

Next Steps *Verify that PMM Server is running* by connecting to the PMM web interface using the IP address from the console output, then *install PMM Client* on all database hosts that you want to monitor.

Removing PMM Server Before you stop and remove *PMM Server*, make sure that the related *PMM Clients* are not collecting any data by removing all monitored instances as described in *Removing monitoring services*.

- If you are *running a Docker container*:

1. Stop and remove the `pmm-server` container:

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

2. If you also want to discard all collected data, remove the `pmm-data` container:

   ```
   $ docker rm pmm-data
   ```

- If you are *running an image in VirtualBox*, stop the *PMM Server* appliance.

  Remove the appliance if necessary.

- If you are *running an Amazon Machine Image*, terminate the instance using the `terminate-instances` command. For example:

  ```
  $ aws ec2 terminate-instances --instance-ids i-XXXX-INSTANCE-ID-XXXX
  ```

Upgrading PMM Server When a new version of PMM becomes available, *remove PMM Server* and *run the new version*.

For example, if you are *running a Docker container*:

```
$ docker stop pmm-server
$ docker rm pmm-server
$ docker run -d \
   -p 80:80 \
   --volumes-from pmm-data \
   --name pmm-server \
   --restart always \
   percona/pmm-server:1.1.3
```

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

Installing PMM Client

PMM Client is a package of agents and exporters installed on a MySQL or MongoDB host that you want to monitor. The components collect various data about general system and database performance, and send this data to corresponding *PMM Server* components.

Before installing the *PMM Client* package on a database host, make sure that your *PMM Server* host is accessible. For example, you can `ping 192.168.100.1` or whatever IP address *PMM Server* is running on.

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`).

The minimum requirements for Query Analytics (QAN) are:

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

Note: 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.

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

- *Install PMM Client on Debian or Ubuntu*

- *Install PMM Client on Red Hat or CentOS*

If you are not able to install from Percona's software repositories or running some other Linux distribution, try *Installing PMM Client Manually*.

Installing PMM Client on Debian or Ubuntu

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 7 (wheezy)
- Debian 8 (jessie)
- Ubuntu 12.04 LTS (Precise Pangolin)
- Ubuntu 14.04 LTS (Trusty Tahr)
- Ubuntu 16.04 LTS (Xenial Xerus)
- Ubuntu 16.10 (Yakkety Yak)

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

To install *PMM Client*:

1. If your system does not already have Percona's `apt` repository configured, fetch the repository package:

   ```
   wget https://repo.percona.com/apt/percona-release_0.1-4.$(lsb_release -sc)_all.deb
   ```

2. Install the repository package:

   ```
   sudo dpkg -i percona-release_0.1-4.$(lsb_release -sc)_all.deb
   ```

3. Update the local `apt` cache:

   ```
   sudo apt-get update
   ```

4. Install the `pmm-client` package:

   ```
   sudo apt-get install pmm-client
   ```

Testing and Experimental Repositories Percona offers pre-release builds from the testing repo, and early-stage development builds from the experimental repo. To enable them, add either `testing` or `experimental` at the end of the Percona repository definition in your repository file (by default, `/etc/apt/sources.list.d/percona-release.list`).

For example, if you are running Debian 8 ("jessie") and want to install the latest testing builds, the definitions should look like this:

```
deb http://repo.percona.com/apt jessie main testing
deb-src http://repo.percona.com/apt jessie main testing
```

If you are running Ubuntu 14.04 LTS (Trusty Tahr) and want to install the latest experimental builds, the definitions should look like this:

```
deb http://repo.percona.com/apt trusty main experimental
deb-src http://repo.percona.com/apt trusty main experimental
```

Next Steps After you install *PMM Client*, *connect it to PMM Server*.

Installing PMM Client on Red Hat and CentOS

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 *PMM Client*:

1. If your system does not already have Percona's `yum` repository configured, run the following command:

   ```
   sudo yum install http://www.percona.com/downloads/percona-release/redhat/0.1-4/percona-release-0
   ```

2. Install the `pmm-client` package:

   ```
   sudo yum install pmm-client
   ```

Testing and Experimental Repositories Percona offers pre-release builds from the testing repo, and early-stage development builds from the experimental repo. You can enable either one in the Percona repository configuration file `/etc/yum.repos.d/percona-release.repo`. There are three sections in this file, for configuring corresponding repositories:

- stable release
- testing
- experimental

The latter two repositories are disabled by default.

If you want to install the latest testing builds, set `enabled=1` for the following entries:

```
[percona-testing-$basearch]
[percona-testing-noarch]
```

If you want to install the latest experimental builds, set `enabled=1` for the following entries:

```
[percona-experimental-$basearch]
[percona-experimental-noarch]
```

Next Steps After you install *PMM Client*, *connect it to PMM Server*.

Installing PMM Client Manually

Note: It is recommended to install all Percona software from repositories for officially supported Linux distributions:

- *Install PMM Client on Debian or Ubuntu*
- *Install PMM Client on Red Hat or CentOS*

However, Percona provides downloadable *PMM Client* packages at https://www.percona.com/downloads/pmm-client. These include:

- DEB and RPM packages for all supported Linux distributions, which you can install using corresponding tools (for example, `dpkg` and `rpm`).
- Generic tarball that you can extract on any Linux distribution and run the included `install` script.
- Source code tarball if you want to build *PMM Client* from source.

Removing PMM Client

1. Remove all monitored instances as described in *Removing monitoring services*.
2. Change into the directory with the extracted *PMM Client* tarball and run:

```
$ sudo ./uninstall
```

Note:

- If you installed using RPM packages:

```
$ rpm -e pmm-client
```

- If you installed using YUM:

```
$ yum remove pmm-client
```

- If you installed using DEB packages:

```
$ dpkg -r pmm-client
```

- If you installed using APT:

```
$ apt-get remove pmm-client
```

Upgrading PMM Client

When a newer version of *PMM Client* becomes available, you can update from the Percona software repositories:

- For Debian or Ubuntu:

```
$ sudo apt-get update && sudo apt-get install pmm-client
```

- For RedHat or CentOS:

```
$ yum update pmm-client
```

If installed *PMM Client* *manually*:

1. *Remove PMM Client*.
2. Download and install the *PMM Client* package as described *here*.

Connecting PMM Client to PMM Server

After you *install PMM Client*, it does not automatically connect to PMM Server.

To connect the client to PMM Server, specify the IP address using the `pmm-admin config --server` command. For example, if *PMM Server* is running on `192.168.100.1`, and you installed *PMM Client* on a machine with IP `192.168.200.1`:

```
$ sudo pmm-admin config --server 192.168.100.1
OK, PMM server is alive.

PMM Server      | 192.168.100.1
Client Name     | ubuntu-amd64
Client Address  | 192.168.200.1
```

Note: If you changed the default port 80 when running PMM Server, specify it after the server's IP address. For example:

```
$ sudo pmm-admin config --server 192.168.100.1:8080
```

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

Next Steps

When the client is connected to PMM Server, you can *start collecting data* from the database instance.

Starting Data Collection

After you *connect the client to PMM Server*, enable data collection from the database instance using the `pmm-admin add` command.

For more information about `pmm-admin`, see *Managing PMM Client*.

To enable general system metrics, MySQL metrics, and query analytics, run:

```
sudo pmm-admin add mysql
```

To enable general system metrics and MongoDB metrics, run:

```
sudo pmm-admin add mongodb
```

To enable ProxySQL performance metrics, run:

```
sudo pmm-admin add proxysql:metrics
```

To see what is being monitored, run:

```
$ sudo pmm-admin list
```

For example, if you enable general OS and MongoDB metrics monitoring, output should be similar to the following:

```
$ sudo pmm-admin list
pmm-admin 1.1.0

PMM Server      | 192.168.100.1
Client Name     | ubuntu-amd64
Client Address  | 192.168.200.1
```

```
Service manager | linux-systemd

---------------- ------------ ------------ -------- --------------- --------
METRIC SERVICE   NAME         CLIENT PORT  RUNNING  DATA SOURCE     OPTIONS
---------------- ------------ ------------ -------- --------------- --------
linux:metrics    ubuntu-amd64  42000        YES      -
mongodb:metrics  ubuntu-amd64  42003        YES      localhost:27017
```

For more information about adding instances, run `pmm-admin add --help`.

Next Steps

After you set up data collection, you can *install PMM Client* on another database instance, *connect it to PMM Server*, and enable data collection in a similar way.

1.3 Using the Percona Monitoring and Management Platform

You can access the PMM web interface using the IP address of the host where *PMM Server* is running. For example, if *PMM Server* is running on a host with IP 192.168.100.1, access the following address with your web browser: `http://192.168.100.1`.

The landing page has links to corresponding PMM tools:

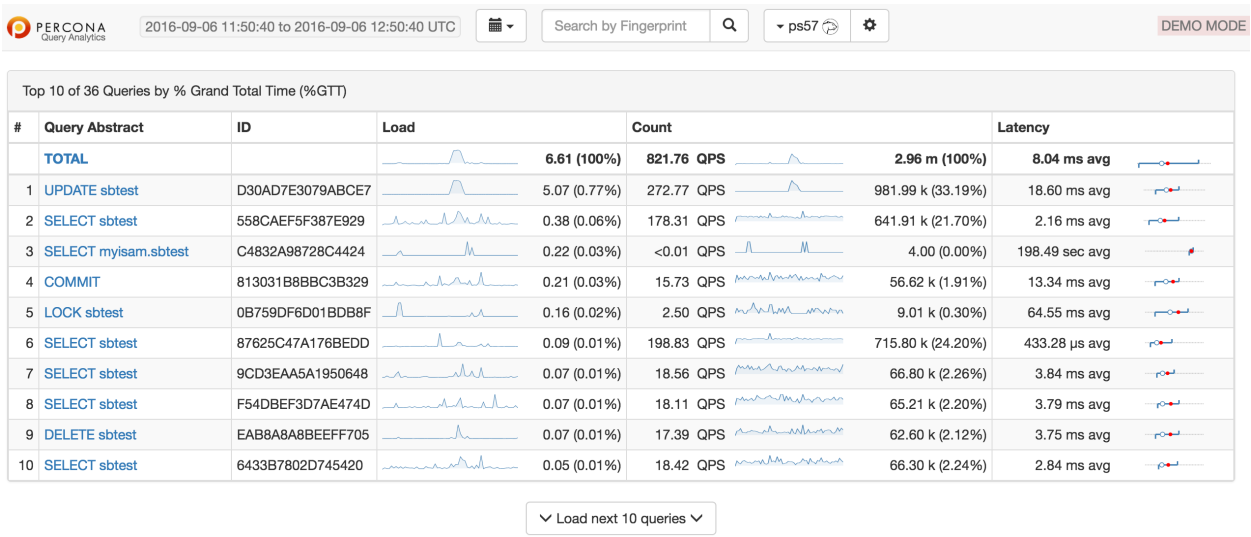
- Query Analytics
- Metrics Monitor
- Orchestrator

These tools provide comprehensive insight into the performance of a MySQL host.

1.3.1 Query Analytics

The *Query Analytics* tool enables database administrators and application developers to analyze MySQL queries over periods of time and find performance problems. Query Analytics 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 following image shows the *Query Analytics* app.



The summary table contains top 10 queries ranked by **%GTT** (percent of grand total time), which is the percentage of time that the MySQL server spent executing a specific query, compared to the total time it spent executing all queries during the selected period of time.

You can select the period of time at the top, by selecting a predefined interval (last hour, 3 hours, 6 hours, 12 hours, last day, or 5 days), or select a specific inteval using the calendar icon.

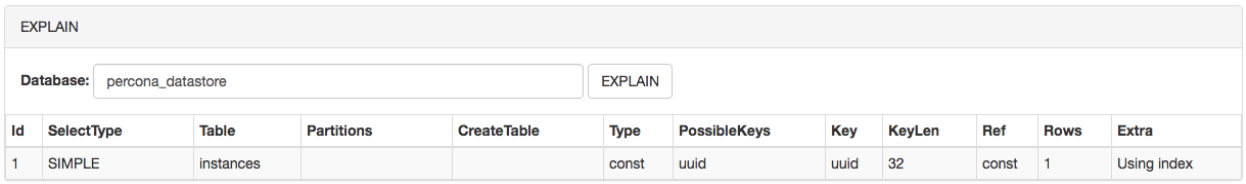
If you have multiple MySQL hosts with *PMM Client* installed, you can switch between those hosts using the drop-down list at the top.

To configure the QAN agent running on a MySQL host with *PMM Client*, click the gear icon at the top.

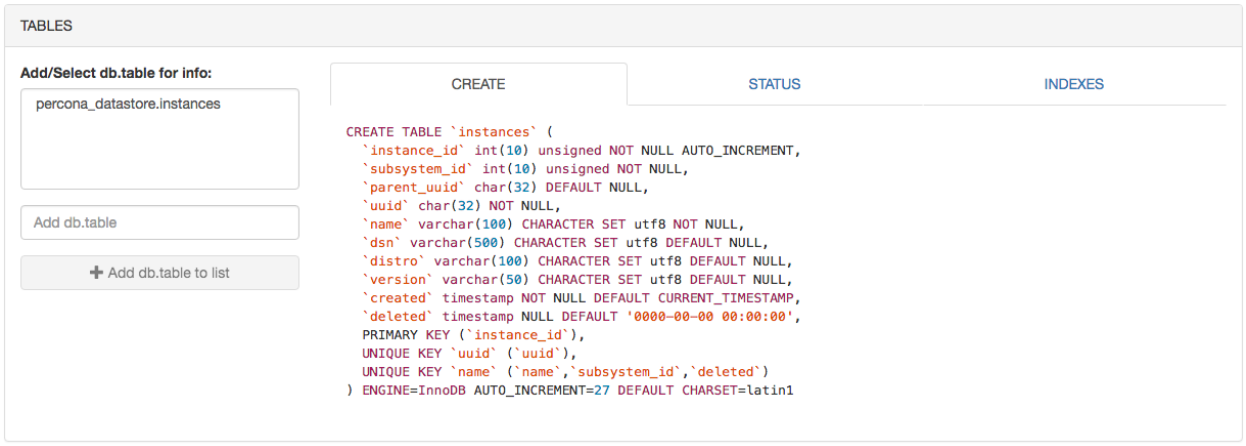
Query Details

You can get details for a query if you click it in the summary table. The details contain all metrics specific to that particular query, such as, bytes sent, lock time, rows sent, and so on. You can see when the query was first and last seen, get an example of the query, as well as its fingerprint.

The details section enables you to run `EXPLAIN` on the selected query directly from the PMM web interface (simply specify the database).



At the bottom, you can run Table Info for the selected query. This enables you to get `SHOW CREATE TABLE`, `SHOW INDEX`, and `SHOW TABLE STATUS` for each table used by the query directly from the PMM web interface.



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 select to parse query data from the Performance Schema. 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 running Percona Server, a *properly configured slow query log* will provide the most amount of information with the lowest overhead. Otherwise, using *Performance Schema* will likely provide better results.

To use Performance Schema:

1. Make sure that the `performance_schema` variable is set to `ON`:

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

   If not, add the the following lines to `my.cnf` and restart MySQL:

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

   Note: 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.

   ---

2. Configure QAN agent to collect data from Performance Schema:

   If the instance is already running:

   (a) In the Query Analytics web UI, click the gear button at the top.

   (b) Under **Query Analytics**, select **Performance Schema** in the **Collect from** drop-down list.

   (c) Click **Apply** to save changes.

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

```
sudo pmm-admin add mysql --user root --password root --create-user --query-source perfschema
```

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

1.3.2 Metrics Monitor

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.

When you open *Metrics Monitor* for the first time, it loads the **Cross Server Graphs** dashboard. The credentials used to sign in to Grafana depend on the options that you specified when *starting PMM Server*:

- If you did not specify either `SERVER_USER` or `SERVER_PASSWORD`, you will be signed in anonymously. You can change to a different existing Grafana user.
- If you specified both `SERVER_USER` and `SERVER_PASSWORD`, then these credentials will be used to sign in to Grafana.
- If you specified only `SERVER_PASSWORD`, a single user (`pmm`) will be used to sign in to all components (including QAN, Prometheus, Grafana, etc.). You will not be able to change to a different Grafana user.
- If you specified only `SERVER_USER`, this parameter will be ignored.

Warning: Do not include the `#` or `:` symbols in `SERVER_USER`.

To access the dashboards, provide default user credentials:

- User: `admin`
- Password: `admin`

On the Home screen, select a dashboard from the list of available Percona Dashboards. For example, the following image shows the **MySQL Overview** dashboard:

1.3.3 Orchestrator

Note: Orchestrator was included into PMM for experimental purposes. It is a standalone tool, not integrated with PMM other than that you can access it from the landing page.

Orchestrator is a MySQL replication topology management and visualization tool. You can access it using the `/orchestrator` URL after *PMM Server* address. Alternatively, you can click the **MySQL Replication Topology Manager** button on the main *PMM Server* landing page.

To use it, create a MySQL user for Orchestrator on all managed instances:

```
GRANT SUPER, PROCESS, REPLICATION SLAVE, RELOAD ON *.* TO 'orc_client_user'@'%' IDENTIFIED BY 'orc_cl
```

Note: The credentials in the previous example are default. If you use a different user name or password, you have to pass them when *running PMM Server* using the following options:

```
-e ORCHESTRATOR_USER=name -e ORCHESTRATOR_PASSWORD=pass
```

Then you can use the **Discover** page in the Orchestrator web interface to add the instances to the topology.

CHAPTER

TWO

ADVANCED

2.1 Managing PMM Client

Use the `pmm-admin` tool to manage *PMM Client*.

Note: The `pmm-admin` tool requires root access (you should either be logged in as a user with root privileges or be able to run commands with `sudo`).

Use the `--help` option to view the built-in help. For example, you can view all available commands and options by running the following:

```
sudo pmm-admin --help
```


2.1.1 Adding monitoring services

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

For complete MySQL instance monitoring:

```
sudo pmm-admin add mysql
```

The previous command adds the following services:

- `linux:metrics`
- `mysql:metrics`
- `mysql:queries`

For complete MongoDB instance monitoring:

```
sudo pmm-admin add mongodb
```

The previous command adds the following services:

- `linux:metrics`
- `mongodb:metrics`

linux:metrics

To enable general system metrics monitoring:

```
sudo pmm-admin add linux:metrics
```

This creates the `pmm-linux-metrics-42000` service that collects local system metrics for this particular OS instance.

Note: It should be able to detect the local PMM Client name, but you can also specify it explicitely as an argument.

For more information, run `sudo pmm-admin add linux:metrics --help`

mysql:queries

To enable MySQL query analytics:

```
sudo pmm-admin add mysql:queries
```

This creates the `pmm-mysql-queries-0` service that is able to collect QAN data for multiple remote MySQL server instances.

The `pmm-admin` tool will attempt to automatically detect the local MySQL instance and MySQL superuser credentials. You can use options to provide this information for `pmm-admin` if it is not able to auto-detect. You can also specify the `--create-user` option to create a dedicated `pmm` user on the MySQL host that you want to monitor. This user will be given all the necessary privileges for monitoring, and is recommended over using the MySQL superuser.

For example, to set up remote monitoring of QAN data on a MySQL server located at 192.168.200.2, use a command similar to the following:

```
sudo pmm-admin add mysql:queries --user root --password root --host 192.168.200.2 --create-user
```

QAN can use either the slow query log or Performance Schema as the source. By default, it chooses the slow query log for a local MySQL instance and Performance Schema otherwise. For more information about the differences, see *Performance Schema*.

You can explicitely set the query source when adding a QAN instance using the `--query-source` option.

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

mysql:metrics

To enable MySQL metrics monitoring:

```
sudo pmm-admin add mysql:metrics
```

This creates the `pmm-mysql-metrics-42002` service that collects MySQL instance metrics.

The `pmm-admin` tool will attempt to automatically detect the local MySQL instance and MySQL superuser credentials. You can use options to provide this information for `pmm-admin` if it is not able to auto-detect. You can also

specify the `--create-user` option to create a dedicated `pmm` user on the MySQL host that you want to monitor. This user will be given all the necessary privileges for monitoring, and is recommended over using the MySQL superuser.

For example, to set up remote monitoring of MySQL metrics on a server located at 192.168.200.3, use a command similar to the following:

```
sudo pmm-admin add mysql:metrics --user root --password root --host 192.168.200.3 --create-user
```

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

mongodb:metrics

To enable MongoDB metrics monitoring:

```
sudo pmm-admin add mongodb:metrics
```

This creates the `pmm-mongodb-metrics-42003` service that collects local MongoDB metrics for this particular MongoDB instance.

Note: It should be able to detect the local PMM Client name, but you can also specify it explicitely as an argument.

You can use options to specify the MongoDB replica set, cluster name, and node type. For example:

```
sudo pmm-admin add mongodb --replset repl1 --cluster cluster1 --nodetype mongod
```

For more information, run `sudo pmm-admin add mongodb:metrics --help`

proxysql:metrics

To enable ProxySQL performance metrics monitoring:

```
sudo pmm-admin add proxysql:metrics
```

This creates the `pmm-proxysql-metrics-42004` service that collects local ProxySQL performance metrics.

Note: It should be able to detect the local PMM Client name, but you can also specify it explicitely as an argument.

For more information, run `sudo pmm-admin add proxysql:metrics --help`

2.1.2 Removing monitoring services

Use the `pmm-admin rm` command to remove monitoring services. Specify the instance's type and name. You can see the names of instances by running `sudo pmm-admin list`.

For example, to remove a MySQL instance designated by `ubuntu-amd4` from monitoring, run the following:

```
sudo pmm-admin rm mysql ubuntu-amd64
```

For more information, run `sudo pmm-admin rm --help`.

2.1.3 Listing monitored instances

To see what is being monitored, run the following:

```
sudo pmm-admin list
```

The output provides the following info:

- Version of `pmm-admin`
- *PMM Server* host address, and local host name and address (this can be configured using `pmm-admin config`)
- System manager that `pmm-admin` uses to manage PMM services
- A table that lists all services currently managed by `pmm-admin`, with basic information about each service

For example, if you enable general OS and MongoDB metrics monitoring, output should be similar to the following:

```
$ sudo pmm-admin list
pmm-admin 1.1.3

PMM Server      | 192.168.100.1
Client Name     | ubuntu-amd64
Client Address  | 192.168.200.1
Service manager | linux-systemd

-------------- ------------- ------------ -------- ---------------- --------
METRIC SERVICE  NAME          CLIENT PORT  RUNNING  DATA SOURCE      OPTIONS
-------------- ------------- ------------ -------- ---------------- --------
linux:metrics   ubuntu-amd64  42000        YES      -
mongodb:metrics ubuntu-amd64  42003        YES      localhost:27017
```

2.1.4 Configuring PMM Client

Use the `pmm-admin config` command to configure how `pmm-admin` communicates with *PMM Server*.

The following options are available:

--client-address string Client host address (detected automatically)

--client-name string Client host name (set to the current host name)

--server string PMM Server host address

--server-insecure-ssl Enable insecure SSL (self-signed certificate)

--server-password string HTTP password configured on PMM Server

--server-ssl Enable SSL to communicate with PMM Server

--server-user string HTTP user configured on PMM Server (default "pmm")

For more information, run `sudo pmm-admin config --help`

2.1.5 Getting information about PMM Client

Use the `pmm-admin info` command to display basic info about `pmm-admin`. The output is also displayed before the table with services when you run `pmm-admin list`.

The following example shows the output if both *PMM Server* and *PMM Client* are on the same host named `ubuntu-amd64`, which uses `systemd` to manage services.

```
$ sudo pmm-admin info
pmm-admin 1.1.3

PMM Server      | 192.168.100.6
Client Name     | ubuntu-amd64
Client Address  | 192.168.200.1
Service manager | linux-systemd
```

This can be configured using `pmm-admin config`.

For more information, run `sudo pmm-admin info --help`.

2.1.6 Checking network connectivity

Use the `pmm-admin check-network` command to run tests that verify connectivity between *PMM Client* and *PMM Server*. The tests are performed both ways, with results separated accordingly:

- Client > Server

  Pings Consul API, Query Analytics API, and Prometheus API to make sure they are alive and reachable.

  Performs a connection performance test to see the latency from *PMM Client* to *PMM Server*.

- Server > Client

  Checks the status of Prometheus endpoints and makes sure it can scrape metrics from corresponding exporters.

  Successful pings of *PMM Server* from *PMM Client* do not mean that Prometheus is able to scrape from exporters. If the output shows some endpoints in problem state, make sure that the corresponding service is running (see `pmm-admin list`). If the services that correspond to problematic endpoints are running, make sure that the firewall settings on *PMM Client* allow incoming connections for corresponding ports.

The `pmm-admin check-network` command has one option (`--no-emoji`), which replaces emojis with words in the status.

The following example shows output without emojis:

```
$ sudo pmm-admin check-network --no-emoji
PMM Network Status

Server | 192.168.100.1
Client | 192.168.200.1

* Client > Server
--------------- -------------
SERVICE         CONNECTIVITY
--------------- -------------
Consul API      OK
QAN API         OK
Prometheus API  OK

Connection duration | 166.689µs
Request duration    | 364.527µs
Full round trip     | 531.216µs

* Server > Client
---------------- ------------- ---------------------- -------------
METRIC SERVICE   NAME          PROMETHEUS ENDPOINT    REMOTE STATE
---------------- ------------- ---------------------- -------------
linux:metrics    ubuntu-amd64  192.168.200.1:42000    OK
```

```
mysql:metrics    ubuntu-amd64  192.168.200.1:42002    OK
mongodb:metrics  ubuntu-amd64  192.168.200.1:42003    PROBLEM
```

For more information, run `sudo pmm-admin check-network --help`.

2.1.7 Pinging PMM Server

Use the `pmm-admin ping` command to ping *PMM Server*. If the ping is successful, it returns `OK`.

For more information, run `sudo pmm-admin ping --help`.

2.1.8 Starting and stopping metric services

Services that you add using `pmm-admin add` can be started and stopped manually using `pmm-admin start` and `pmm-admin stop`.

For example, to start the `mongodb:metrics` service on host `ubuntu-amd64`:

```
sudo pmm-admin start mongodb:metrics ubuntu-amd64
```

To stop the `linux:metrics` service on host `centos-amd64`:

```
sudo pmm-admin stop linux:metrics centos-amd64
```

To stop all services managed by this `pmm-admin`:

```
sudo pmm-admin stop --all
```

For more information, run `sudo pmm-admin start --help` or `sudo pmm-admin stop --help`.

2.2 Using PMM with Amazon RDS

It is possible to use PMM for monitoring Amazon RDS (just like any remote MySQL instance).

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

Note: If latency is higher than 1 second, you should change the minimum resolution by setting the `METRICS_RESOLUTION` environment variable when *creating and running the PMM Server container*. For more information, see *What resolution is used for metrics?*.

Query analytics requires *Performance Schema* as the query source. Enable the `performance_schema` option under **Parameter Groups** on RDS (you will probably need to create a new **Parameter Group** and set it to the database instance).

When adding a monitoring instance for 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 RDS instance that you want to monitor:

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

If you have RDS with MySQL version prior to 5.7, *REPLICATION CLIENT* privilege is not available there and has to be excluded from the above statement.

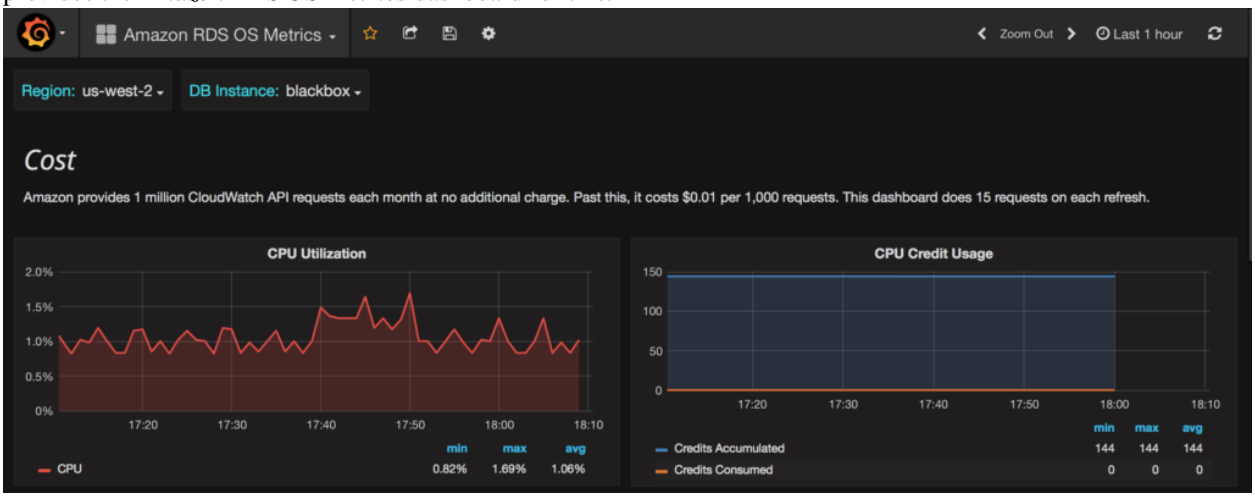
The following example shows how to enable QAN and MySQL metrics monitoring on Amazon RDS:

```
$ sudo pmm-admin add mysql:metrics --host rds-mysql57.vb81uqbc7tbe.us-west-2.rds.amazonaws.com --user
$ sudo pmm-admin add mysql:queries --host rds-mysql57.vb81uqbc7tbe.us-west-2.rds.amazonaws.com --user
```

Note: General system metrics cannot be monitored remotely, because `node_exporter` requires access to the local file system. This means that the `linux:metrics` service cannot be used to monitor Amazon RDS instances or any remote MySQL instance.

2.2.1 Monitoring Amazon RDS OS Metrics

You can use CloudWatch as the data source in Grafana to monitor OS metrics for Amazon RDS instances. PMM provides the *Amazon RDS OS Metrics* dashboard for this.



To set up OS metrics monitoring for Amazon RDS in PMM via CloudWatch:

1. Create an IAM user on the AWS panel for accessing CloudWatch data, and attach the managed policy `CloudWatchReadOnlyAccess` to it.
2. Create a credentials file on the host running PMM Server with the following contents:

   ```
   [default]
   aws_access_key_id = <your_access_key_id>
   aws_secret_access_key = <your_secret_access_key>
   ```

3. Start the `pmm-server` container with an additional `-v` flag that specifies the location of the file with the IAM user credentials and mounts it to `/usr/share/grafana/.aws/credentials` in the container. For example:

   ```
   $ docker run -d \
     -p 80:80 \
     --volumes-from pmm-data \
     -v /path/to/file/with/creds:/usr/share/grafana/.aws/credentials \
     --name pmm-server \
     --restart always \
     percona/pmm-server:1.0.6
   ```

The *Amazon RDS OS Metrics* dashboard uses 60 second resolution and shows the average value for each data point. An exception is the *CPU Credit Usage* graph, which has a 5 minute average and interval length. All data is fetched in real time and not stored anywhere.

This dashboard can be used with any Amazon RDS database engine, including MySQL, Aurora, etc.

Note: Amazon provides one million CloudWatch API requests per month at no additional cost. Past this, it costs $0.01 per 1,000 requests. The pre-defined dashboard performs 15 requests on each refresh and an extra two on initial loading.

For more information, see Amazon CloudWatch Pricing.

2.3 Configuring MySQL for Percona Monitoring and Management

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 will 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*. Using the slow query log to capture all queries provides maximum details, but can impact performance on heavily loaded systems unless it is used with the query sampling feature available only in Percona Server. 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.

You can add configuration examples provided in this guide 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 *Percona 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 *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 Query Analytics.

```
log_output=file
slow_query_log=ON
long_query_time=0.01
log_slow_admin_statements=ON
log_slow_slave_statements=ON
```

2.3.1 Configuring the Slow Query Log in Percona Server

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*. That is, queries with execution time above a certain threshold, which 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.

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.

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.

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`.

2.3.2 Configuring Performance Schema

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 the better alternative.

As of MySQL 5.6 (including MariaDB 10.0+ and Percona Server 5.6+), Performance Schema is enabled by default with no additional configuration required.

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)
```

For more information about using Performance Schema in PMM, see *Performance Schema*.

2.3.3 Settings for Dashboards

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

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.

MySQL InnoDB Metrics

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

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

MySQL User Statistics

User statistics is a feature available in 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`.

MySQL Performance Schema

With MySQL version 5.6 or later, Performance Schema instrumentation is enabled by default. If certain instruments are not enabled, you will not see the corresponding graphs in the *Performance Schema* dashboard. To enable full instrumentation, set the `--performance_schema_instrument` option to `'%=on'` at startup:

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

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

MySQL Query Response Time

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.

Note: This feature causes very high overhead, especially on systems processing more than 10 000 queries per second. Use it only temporarily when troubleshooting problems.

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';
   ```

   For more information, see this guide

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

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

2.4 Security Features in Percona Monitoring and Management

You can protect PMM from unauthorized access using the following security features:

- HTTP password protection adds authentication when accessing the *PMM Server* web interface
- SSL encryption secures traffic between *PMM Client* and *PMM Server*

2.4.1 Enabling Password Protection

You can set the password for accessing the *PMM Server* web interface by passing the `SERVER_PASSWORD` environment variable when *creating and running the PMM Server container*. To set the environment variable, use the `-e` option. For example, to set the password to `pass1234`:

```
-e SERVER_PASSWORD=pass1234
```

By default, the user name is `pmm`. You can change it by passing the `SERVER_USER` variable.

For example:

```
$ docker run -d -p 80:80 \
    --volumes-from pmm-data \
    --name pmm-server \
    -e SERVER_USER=jsmith \
    -e SERVER_PASSWORD=pass1234 \
```

```
    --restart always \
    percona/pmm-server:1.1.3
```

PMM Client uses the same credentials to communicate with *PMM Server*. If you set the user name and password as described, specify them when *Connecting PMM Client to PMM Server*:

```
$ pmm-admin config --server 192.168.100.1 --server-user jsmith --server-password pass1234
```

2.4.2 Enabling SSL Encryption

You can encrypt traffic between *PMM Client* and *PMM Server* using SSL certificates.

1. Buy or generate SSL certificate files for PMM.

   For example, you can generate necessary self-signed certificate files into the `/etc/pmm-certs` directory using the following commands:

   ```
   # openssl dhparam -out /etc/pmm-cert/dhparam.pem 4096
   # openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/pmm-certs/server.key -out /et
   Generating a 2048 bit RSA private key
   ..................................+++
   ....+++
   writing new private key to '/etc/pmm-certs/server.key'
   -----
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [XX]:US
   State or Province Name (full name) []:North Carolina
   Locality Name (eg, city) [Default City]:Raleigh
   Organization Name (eg, company) [Default Company Ltd]:Percona
   Organizational Unit Name (eg, section) []:PMM
   Common Name (eg, your name or your server's hostname) []:centos7.vm
   Email Address []:jsmith@example.com
   ```

   Note: The `dhparam.pem` file is not required. It can take a lot of time to generate, so you can skip it.

   Note: The `server.key` and `server.crt` files must be named exactly as shown. Files with other names will be ignored.

2. Mount the directory with the certificate files into `/etc/nginx/ssl` when *running the PMM Server container*:

   ```
   $ docker run -d -p 443:443 \
       --volumes-from pmm-data \
       --name pmm-server \
       -v /etc/pmm-certs:/etc/nginx/ssl \
       --restart always \
       percona/pmm-server:1.1.3
   ```

   Note: Note that the container should expose port 443 instead of 80 to enable SSL encryption.

3. Enable SSL when *Connecting PMM Client to PMM Server*. If you purchased the certificate from a certificate authority (CA):

   ```
   $ pmm-admin config --server 192.168.100.1 --server-ssl
   ```

   If you generated a self-signed certificate:

   ```
   $ pmm-admin config --server 192.168.100.1 --server-insecure-ssl
   ```

2.4.3 Combining Security Features

You can enable both HTTP password protection and SSL encryption by combining the corresponding options.

The following example shows how you might *run the PMM Server container*:

```
$ docker run -d -p 443:443 \
    --volumes-from pmm-data \
    --name pmm-server \
    -e SERVER_USER=jsmith \
    -e SERVER_PASSWORD=pass1234 \
    -v /etc/pmm-certs:/etc/nginx/ssl \
    --restart always \
    percona/pmm-server:1.1.3
```

The following example shows how you might *connect to PMM Server*:

```
$ pmm-admin config --server 192.168.100.1 --server-user jsmith --server-password pass1234 --server-ir
```

To see which security features are enabled, run either `pmm-admin ping`, `pmm-admin config`, `pmm-admin info`, or `pmm-admin list` and look at the server address field. For example:

```
[root@centos7 pmm-client]# pmm-admin ping
OK, PMM server is alive.

PMM Server      | 192.168.100.1 (insecure SSL, password-protected)
Client Name     | centos7.vm
Client Address  | 192.168.200.1
```

2.5 Metrics Monitor Dashboards

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

2.5.1 MongoDB Dashboards

MongoDB Overview

Name	Importance	Description
Command Operations	INFO	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.
Connections	IMPORTANT	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.
Cursors	INFO	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.
Document Operations	INFO	When used in combination with **Command Operations**, this graph can help identify *write aplification*. For example, when one `insert` or `update` command actually inserts or updates hundreds, thousands, or even millions of documents.
Queued Operations	CRITICAL	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.
getLastError Write Time getLastError Write Operations	INFO	This is useful for write-heavy workloads to understand how long it takes to verify writes and how many concurrent writes are occurring.
Asserts	INFO	Asserts are not important by themselves, but you can correlate spikes with other graphs.
Memory Faults	CRITICAL	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.

MongoDB ReplSet

This dashboard provides information about replica sets and their members.

Name	Importance	Description
ReplSet State	INFO	Shows the role of the selected member instance (PRIMARY or SECONDARY)
ReplSet Members	INFO	Shows the number of members in the replica set
ReplSet Last Election	INFO	Shows how long ago the last election occurred
ReplSet Lag	INFO	Shows the current replication lag for the selected member
Storage Engine	INFO	Shows the storage engine used on the instance
Oplog Insert Time	INFO	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).
Oplog Recovery Window	CRITICAL	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.
Replication Lag	INFO	Shows the delay between an operation occurring on the primary and that same operation getting applied on the selected member
Elections	INFO	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.
Member State Uptime	INFO	Shows how long various members were in PRIMARY and SECONDARY roles
Max Heartbeat Time	IMPORTANT	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.
Max Member Ping Time	INFO	This can show a correlation with the replication lag value

CHAPTER

THREE

REFERENCE

3.1 Percona Monitoring and Management Release Notes

3.1.1 Percona Monitoring and Management 1.1.3

Date April 21, 2017

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/pmm-client/

For install instructions, see *Deploying Percona Monitoring and Management*.

New in PMM Server

- PMM-649: Added the *InnoDB Page Splits* and *InnoDB Page Reorgs* graphs to the *MySQL InnoDB Metrics Advanced* dashboard.
- Added the following graphs to the *MongoDB ReplSet* dashboard:
  - Oplog Getmore Time
  - Oplog Operations
  - Oplog Processing Time
  - Oplog Buffered Operations
  - Oplog Buffer Capacity
- Added descriptions for graphs in the following dashboards:
  - MongoDB Overview
  - MongoDB ReplSet
  - PMM Demo

New in PMM Client

- PMM-491: Improved `pmm-admin` error messages.
- PMM-523: Added the `--verbose` option for `pmm-admin add`.
- PMM-592: Added the `--force` option for `pmm-admin stop`.

- PMM-702: Added the `db.serverStatus().metrics.repl.executor` stats to `mongodb_exporter`. These new stats will be used for graphs in future releases.
- PMM-731: Added real time checks to `pmm-admin check-network` output.
- The following commands no longer require connection to *PMM Server*:
  - `pmm-admin start --all`
  - `pmm-admin stop --all`
  - `pmm-admin restart --all`
  - `pmm-admin show-passwords`

Note: If you want to start, stop, or restart a specific service, connection to *PMM Server* is still required.

3.1.2 Percona Monitoring and Management 1.1.2

Date April 3, 2017

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/pmm-client/

For install instructions, see *Deploying Percona Monitoring and Management*.

PMM Server

- Updated to latest versions:
  - Grafana 4.2
  - Consul 0.7.5
  - Prometheus 1.5.2
  - Orchestrator 2.0.3
- Migrated *PMM Server* to use CentOS 7 as base operating system.
- Changed the entrypoint so that supervisor is PID 1.
- Added the following dashboards:
  - MongoDB InMemory
  - MongoDB MMAPv1
  - MariaDB
- PMM-633: Set the following default values in `my.cnf`:

```
[mysqld]

# Default MySQL Settings
innodb_buffer_pool_size=128M
innodb_log_file_size=5M
innodb_flush_log_at_trx_commit=1
innodb_file_per_table=1
innodb_flush_method=O_DIRECT

# Disable Query Cache by default
```

```
query_cache_size=0
query_cache_type=0
```

- PMM-676: Added descriptions for graphs in Disk Performance and Galera dashboards.

PMM Client

- Fixed `pmm-admin remove --all` to clear all saved credentials.
- Several fixes to `mongodb_exporter` including PMM-629 and PMM-642.
- PMM-504: Added ability to change the name of a client with running services:

```
$ sudo pmm-admin config --client-name new_name --force
```

> **Warning:** Some Metrics Monitor data may be lost.

3.1.3 Percona Monitoring and Management 1.1.1

Date February 20, 2017

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/pmm-client/

For install instructions, see *Deploying Percona Monitoring and Management*.

This release introduces new ways for running *PMM Server*:

- *Run PMM Server using VirtualBox*
- *Run PMM Server using Amazon Machine Image (AMI)*

Note: These images are experimental and not recommended for production. It is best to *run PMM Server using Docker*.

There are no changes compared to previous *1.1.0 Beta* release, except small fixes for MongoDB metrics dashboards.

3.1.4 Percona Monitoring and Management 1.1.0 Beta

Date February 7, 2017

PMM Server https://hub.docker.com/r/perconalab/pmm-server/

PMM Client https://www.percona.com/downloads/TESTING/pmm/

Note: This beta release is highly experimental with features that are not ready for production. Do not upgrade to it from previous versions. Use it only in a test environment.

For install instructions, see *Deploying Percona Monitoring and Management*.

Changes

Introduced Amazon Machine Image (AMI) and VirtualBox images for PMM Server:

- OVA image for VirtualBox is available from the testing download area.
- Public Amazon Machine Image (AMI) is ami-9a0acb8c.

New in PMM Server:

- Grafana 4.1.1
- Prometheus 1.5.0
- Consul 0.7.3
- Updated the **MongoDB ReplSet** dashboard to show the storage engine used by the instance
- PMM-551: Fixed QAN changing query format when a time-based filter was applied to the digest

New in PMM Client:

- PMM-530: Fixed `pmm-admin` to support special characters in passwords
- Added displaying of original error message in `pmm-admin config` output

Known Issues:

- Several of the MongoDB metrics related to MongoRocks engine do not display correctly. This issue will be resolved in the GA production release.

3.1.5 Percona Monitoring and Management 1.0.7

Date December 12, 2016

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/pmm-client/

Upgrading

1. Stop and remove the `pmm-server` container:

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

2. Create the `pmm-server` container with the new version tag:

   ```
   docker run -d \
     -p 80:80 \
     --volumes-from pmm-data \
     --name pmm-server \
     --restart always \
     percona/pmm-server:1.0.7
   ```

3. *Install new PMM Client version* on all hosts that you are monitoring. If you previously installed using Percona repositories, you can upgade the package as follows:

   - For Debian-based distributions:

     ```
     sudo apt-get install --only-upgrade pmm-client
     ```

   - For Red Hat Enterprise Linux derivatives:

```
sudo yum update pmm-client
```

4. (Optional) *Remove* and *add* the services running on PMM clients.

   There are changes related to authentication and general security that will only be available after you re-add the services. For more information, see the changes mentioned below.

Changes

New in PMM Server:

- Grafana 4.0.2
- Prometheus 1.4.1
- Consul 0.7.1
- Orchestrator 2.0.1
- Enabled HTTPS/TLS and basic authentication support on Prometheus targets
- Fixed potential error with too many connections on Query Analytics API
- Added new widgets and graphs to *PXC/Galera Graphs* dashboard
- Fixed hostgroup filtering for *ProxySQL Overview* dashboard
- Various fixes to MongoDB dashboards

New in PMM Client:

- Added the `--bind-address` option to support running *PMM Server* and *PMM Client* on the different networks.

  By default, this is the address of *PMM Client*. When running PMM on different networks, set `--client-address` to remote (public) address and `--bind-address` to local (private) address.

  Note: This assumes you configure NAT and port forwarding between those addresses.

- Added the `show-passwords` command to display the current HTTP authentication credentials and password of the last created user on MySQL (this is useful for replication setups).
- Fixed slow log rotation for `mysql:queries` service with MySQL 5.1.
- Exposed PXC/Galera gcache size as a metric.
- Amended output of `systemv` service status if run ad-hoc (requires re-adding the services).
- Added automatic generation of self-signed SSL certificate to protect metric services with HTTPS/TLS by default (requires re-adding services, see `check-network` output).
- Enabled basic HTTP authentication for metric services when defined on *PMM Server* and configured on *PMM Client* to achieve client-side protection (requires re-adding services, see `check-network` output).
- Removed MongoDB connection string from being passed in command-line arguments and hidden password from the process list (requires re-adding the `mongodb:metrics` service).
- Removed network port listening by `mysql:queries` service (`percona-qan-agent` process) as there is no need for it.
- Replaced emojis with terminal colors for output of the `check-network` and `list` commands.

3.1.6 Percona Monitoring and Management 1.0.6

Date November 15, 2016

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/pmm-client/

Upgrading

1. Stop and remove the `pmm-server` container:

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

2. Create the `pmm-server` container with the new version tag:

   ```
   docker run -d \
     -p 80:80 \
     --volumes-from pmm-data \
     --name pmm-server \
     --restart always \
     percona/pmm-server:1.0.6
   ```

3. *Install new PMM Client version* on all hosts that you are monitoring. If you previously installed using Percona repositories, you can upgade the package as follows:

   - For Debian-based distributions:

     ```
     sudo apt-get install --only-upgrade pmm-client
     ```

   - For Red Hat Enterprise Linux derivatives:

     ```
     sudo yum update pmm-client
     ```

Changes

New in PMM Server:

- Prometheus 1.2.2
- Made external static files local for PMM home page
- Metrics Monitor improvements:
  - Added *Amazon RDS OS Metrics* dashboard and CloudWatch data source.
  - Added the PMM Server host to metrics monitoring.
  - Refactored MongoDB dashboards.
  - Added *File Descriptors* graph to **System Overview** dashboard.
  - Added *Mountpoint Usage* graph to **Disk Space** dashboard.
- Query Analytics improvements:
  - QAN data is now purged correctly.
  - QAN data retention is made configurable with `QUERIES_RETENTION` option. Default is 8 days.
- Various small fixes to Query Analytics.

New in PMM Client:

- Fixes for `mysql:queries` service using Performance Schema as query source:
  - Fixed crash when `DIGEST_TEXT` is `NULL`.
  - Removed iteration over all query digests on startup.
  - Added sending of query examples to QAN if available (depends on the workload).
- Added query source information for `mysql:queries` service in `pmm-admin list` output.
- Added `purge` command to purge metrics data on the server.
- Updated `mongodb_exporter` with RocksDB support and various fixes.
- Removed `--nodetype` and `--replset` flags for `mongodb:metrics`. The `--cluster` flag is now optional.

  It is recommended to re-add `mongodb:metrics` service and purge existing MongoDB metrics using the `purge` command.
- Enabled monitoring of file descriptors (requires re-adding `linux:metrics` service).
- Improved full uninstallation when PMM Server is unreachable.
- Added time drift check between server and client to *pmm-admin check-network* output.

3.1.7 Percona Monitoring and Management 1.0.5

Date October 14, 2016

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/pmm-client/

Upgrading

Note: All custom Grafana dashboards and settings in Metrics Monitor will be reset when you upgrade PMM Server. Back it up and restore after the upgrade. Starting from version 1.0.5, it is possible to preserve this data, but you will have to recreate the *data container* with `-v /var/lib/grafana`. If you choose to recreate the data container, all previously collected data will be lost.

---

1. Stop and remove the `pmm-server` container:

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

2. If you also want to recreate the `pmm-data` container with support for custom Grafana dashboards and settings:

   ```
   docker rm pmm-data
   ```

3. If you removed `pmm-data` container, create it with the new version tag:

   ```
   docker create \
     -v /opt/prometheus/data \
     -v /opt/consul-data \
     -v /var/lib/mysql \
     -v /var/lib/grafana \
     --name pmm-data \
     percona/pmm-server:1.0.5 /bin/true
   ```

4. Create the `pmm-server` container with the new version tag:

```
docker run -d \
  -p 80:80 \
  --volumes-from pmm-data \
  --name pmm-server \
  --restart always \
  percona/pmm-server:1.0.5
```

5. *Install new PMM Client version*. If you previously installed using Percona repositories, you can upgade the package as follows:

   - For Debian-based distributions:

     ```
     sudo apt-get install --only-upgrade pmm-client
     ```

   - For Red Hat Enterprise Linux derivatives:

     ```
     sudo yum update pmm-client
     ```

Changes

PMM Server changes:

- Prometheus 1.1.3
- Consul 0.7.0
- Added Orchestrator - a MySQL replication topology management and visualization tool. Available at `/orchestrator` URL.

  Note: Orchestrator was included into PMM for experimental purposes. It is a standalone tool, not integrated with PMM other than that you can access it from the landing page.

- Added ProxySQL metrics and dashboard
- Changed metric storage encoding to achieve less disk space usage by 50-70%.
- Grafana data is now stored in the *data container* to preserve your custom dashboards and settings.

  Note: To enable this, create the data container with `-v /var/lib/grafana`.

- MySQL Query Analytics data is now preserved when you remove and then add a `mysql:queries` instance with the same name using `pmm-admin`.
- Fixed rare issue when Nginx tries to use IPv6 for localhost connections.
- Improvements and fixes to Query Analytics.
- Various dashboard improvements.

PMM Client changes:

- Added check for orphaned local and remote services.
- Added `repair` command to remove orphaned services.
- Added `proxysql:metrics` service and `proxysql_exporter`.
- Amended `check-network` output.
- Disabled inital client configuration with a name that is already in use.

- Changed the threshold for automatically disabling table stats when adding `mysql:metrics` service to 1000 tables on the server. Table stats were previously automatically disabled only if there were over 10 000 tables. You can still manually disable table stats using `pmm-admin add mysql --disable-tablestats`. For more information, see *What are common performance considerations?*.
- Fixes for `mysql:queries` service:
  - Improved registration and detection of orphaned setup
  - PID file "" is no longer created on Amazon Linux (requires to re-add `mysql:queries` service)
  - Fixed support for MySQL using a timezone different than UTC
  - Corrected detection of slow log rotation and also perform its own rotation when used as a query source
  - `RELOAD` privilege is now required to flush the slow log

3.1.8 Percona Monitoring and Management 1.0.4

Date September 13, 2016

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/pmm-client/

This is the first General Availability (GA) release.

Upgrading

Note: This release introduces major changes and requires you to completely remove any previous versions of all PMM components. This means you will lose all previously collected data and start from scratch.

1. *Remove all PMM Clients*
2. *Remove PMM Server* (including the `pmm-data` container).

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

3. *Create the PMM data container*
4. *Create and run the PMM Server container*
5. *Install PMM Clients* on all your monitored hosts

Changes

PMM Server changes:

- Grafana 3.1.1
- Prometheus 1.0.2
- Added SSL and HTTP password protection support
- Removed the extra `alias` label for Prometheus
- Added `MongoDB RocksDB`, `PXC/Galera Cluster Overview` dashboards
- Introduced some visual amendments to the dashboards
- Added ability to save predefined dashboards in place

- Query Analytics App:
  - Added sparkline charts to metrics
  - Added search by query fingerprint
  - Various smaller fixes and improvements

PMM Client changes:

- Renamed services managed by `pmm-admin`:
  - `os` > `linux:metrics`
  - `mysql` > `mysql:metrics`
  - `queries` > `mysql:queries`
  - `mongodb` > `mongodb:metrics`
- Added group commands:
  - `pmm-admin add mysql` and `pmm-admin rm mysql`: add and remove `linux:metrics`, `mysql:metrics`, and `mysql:queries` services
  - `pmm-admin add mongodb` and `pmm-admin rm mongodb`: add and remove `linux:metrics` and `mongodb:metrics` services
- Added options to support SSL and HTTP password protection for *PMM Server*
- Added check whether the required binaries of exporters are installed.
- Changed behaviour of `--create-user` flag for adding MySQL instance:
  - Now `pmm-admin` employs a single *pmm* MySQL user, verifies if it exists, and stores the generated password in the configuration
  - Added checks whether MySQL is read-only or a replication slave
  - Stored credentials are automatically picked up by `pmm-admin` when valid
- Replaced standard `mysqld_exporter` with custom one (https://github.com/percona/mysqld_exporter). This enables `pmm-admin` to create a single `mysql:metrics` service instead of three per MySQL instance.
- Added check for MongoDB connectivity when adding `mongodb:metrics` instance.
- Removed the requirement to specify the name when removing a service (the client's name is used by default)
- Allowed to add more than one `linux:metrics` instance for testing purpose
- Added consistency checks to avoid duplicate services across clients
- Implemented automatic client address detection
- Improved installation process: the install script now just copies binaries. You need to use `pmm-admin config` to add *PMM Server* address.
- Now `pmm-admin` does not modify `linux:metrics` instance when adding `mongodb:metrics`
- Table stats are now disabled automatically if there are more than 10 000 tables

3.1.9 Percona Monitoring and Management Beta 1.0.3

Date August 5, 2016

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/TESTING/pmm/pmm-client.tar.gz

Upgrading

Note: This beta release introduces minor changes to both *PMM Client* and *PMM Server*.

If you are upgrading from version 1.0.2:

1. *Upgrade PMM Server*
2. *Upgrade PMM Client* on all monitored hosts

   Note: There is no need to stop monitoring instances and remove PMM Client. You can simply run the `install` script from the new client tarball, or manually copy `./bin/pmm-admin` from the new tarball to `/usr/local/percona/pmm-client/`.

If you are upgrading from an earlier version:

1. *Remove PMM Server*
2. *Remove all PMM Clients*
3. If you removed the `pmm-data` container, create it as described in *Step 1. Create a PMM Data Container*
4. *Create and run the PMM Server container*
5. *Install PMM Client* on all your monitored hosts

Changes

PMM Server changes:

- Fixed the math for query metrics in Query Analytics

PMM Client changes:

- Fixed password auto-detection for MySQL 5.7
- Fixed error when removing `os` and `mysql` instances using Upstart
- Fixed error when starting `percona-qan-agent` service (`queries` instance) under UNIX System V
- Added `--disable-userstats`, `--disable-binlogstats`, and `--disable-processlist` options for `pmm-admin add mysql`
- Renamed the `--disable-per-table-stats` option to `--disable-tablestats`
- Removed the `--disable-infoschema` option

3.1.10 Percona Monitoring and Management Beta 1.0.2

Date July 28, 2016

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/TESTING/pmm/pmm-client.tar.gz

Upgrading

Note: This beta release introduces major changes to *PMM Client* and simplifies Docker commands for *PMM Server*. If you want to preserve Metrics Monitor data, do not remove the `pmm-data` container. However, previous QAN data will be lost anyway.

1. *Remove PMM Server*
2. *Remove all PMM Clients*
3. If you removed the `pmm-data` container, create it as described in *Step 1. Create a PMM Data Container*
4. *Create and run the PMM Server container*
5. *Install PMM Client* on all your monitored hosts

Changes

New software used in PMM Server:

- Prometheus 1.0.1
- Grafana 3.1.0

Simplified interaction with PMM Server container:

- Eliminated port 9001

  Now the container uses only one configurable port (80 by default)

- Eliminated the `ADDRESS` variable

  The IP address of the host is now automatically detected

Redesigned the Query Analytics web app:

- Redesigned the metrics table
- Added the ability to show more than 10 queries
- Added sparkline charts
- Redesigned the instance settings page
- Redesigned the query profile table

Other changes related to PMM Server:

- Set the default metrics retention for Prometheus to 30 days. For more information, see *How to control data retention for Prometheus?*
- Improved MongoDB dashboards based on feedback from experts.

Improved PMM Client management:

The `pmm-admin` has been fully rewritten and is now much more powerful, with more commands, options, and a user-friendly CLI. For more inforamtion about using `pmm-admin`, see *Managing PMM Client*.

- Added the `--help` option to display built-in help for any command.
- Added the ability to set a custom name when adding an instance. By default, the local host name is used.
- Added the `--service-port` option to specify the port that you want the service to use when adding the corresponding instance. By default, it automatically assigns an available port starting from 42000. For more information, see *Can I use non-default ports for instances?*

- Added the `check-network` command to test bidirectional connection and latency between *PMM Client* and *PMM Server*.
- Added the `ping` command to ping *PMM Server* from *PMM Client*.
- Added the `start` and `stop` commands to manually start and stop services managed by `pmm-admin`.
- Added the following new options for `pmm-admin add mysql` to deal with performance issues:
  - `--disable-infoschema`: Disable all metrics from the `information_schema` tables.
  - `--disable-per-table-stats`: Disable per table metrics (for MySQL servers with a huge number of tables)

  For more information, see *What are common performance considerations?*
- When using the `--create-user` option to add a QAN or MySQL metrics monitoring instance, the password generated for the new user now conforms with MySQL 5.7 default password policy.

Other changes related to PMM Client:

- Eliminated intermediate `percona-prom-pm` process. All monitoring services are now created dynamically via the platform service manager (`systemd`, `upstart`, or `systemv`).
- Added the ability to monitor multiple instances of MySQL and MongoDB on the same node
- Cleaned up and improved the installation and uninstallation scripts

3.1.11 Percona Monitoring and Management Beta 1.0.1

Date June 10, 2016

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/TESTING/pmm/pmm-client.tar.gz

Upgrading

Note: This beta release introduces changes to the `pmm-data` container, which is used for storing collected data. To upgrade, you will need to remove and re-create this container, losing all your collected data.

---

1. Stop and remove the `pmm-server` container:

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

2. Remove the `pmm-data` container:

   ```
   $ docker rm pmm-data
   ```

3. Create the *PMM data container*.
4. Create and run the *PMM Server container*.
5. *Upgrade PMM Client* on all your monitored hosts.

New Features

- **Grafana 3.0**: PMM now includes the latest version of Grafana for visualizing collected metrics data.

- **MongoDB Metrics**: With the addition of `mongodb_exporter` for Prometheus and MongoDB dashboards for Grafana, you can now use PMM for monitoring MongoDB metrics.
- **Consul**: Instead of `prom-config-api`, PMM now uses Consul to provide an API service for communication between PMM Client and Prometheus.
- **Nginx**: PMM now uses Nginx, instead of a custom web server.
- **Server Summary**: Aggregated query metrics are now available in QAN.
- **MySQL InnoDB Metrics Advanced**: New dashboard for MySQL metrics.
- The web interface is now fully accessible via port 80.
  - `/qan/`: Query Analytics
  - `/graph/`: Metrics Monitor (Grafana)
  - `/prometheus/`: Prometheus web UI
  - `/consul/`: Consul web UI
  - `/v1/`: Consul API

  The only other port is 9001 used by QAN API.
- `pmm-admin` tool now includes the ability to add MongoDB instance and specify the port after the address of the PMM Server.

3.1.12 Percona Monitoring and Management Beta 1.0.0

This is the initial beta release of PMM.

Date April 17, 2016

PMM Server https://hub.docker.com/r/percona/pmm-server/

PMM Client https://www.percona.com/downloads/TESTING/pmm/pmm-client-1.0.0-x86_64.tar

Features of Query Analytics:

- Uses either the slow query log or Performance Schema
- Leverages features of Percona Server (slow log rate limiting, extra metrics, and more)
- Supports analysis for multiple MySQL hosts
- Provides query ranking for any time range
- Includes query details with example and fingerprint, real-time `EXPLAIN` and real-time Table Info

Features of Metrics Monitor:

- Supports monitoring of multiple MySQL hosts
- Combines general system metrics (CPU, memroy, disk usage, and so on) with compehensive MySQL metrics
- Includes Percona Dashboards - a set of dashboards created and tuned by MySQL experts
- Flexible graph resolution settings enable you to select almost any time range and period
- Zooming is synchronized across all graphs on a dashboard for granular analysis
- Logarithmic and linear scale
- Graphs automatically update when new data arrives

3.2 Contacting and Contributing

Use the community forum to ask questions about using PMM.

Use the PMM project in JIRA to report bugs.

Use the GitHub repository to explore source code and suggest contributions.

3.3 Frequently Asked Questions

3.3.1 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.

3.3.2 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.

  Minimum memory is 2 GB for one monitored database node, but it is not linear when you increase 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.

3.3.3 How to control memory consumption for Prometheus?

By default, Prometheus in PMM Server uses up to 256 MB of memory for storing the most recently used data chunks. Depending on the amount of data coming into Prometheus, you may require a higher limit to avoid throttling data ingestion, or allow less memory consumption if it is needed for other processes.

You can control the allowed memory consumption for Prometheus by passing the `METRICS_MEMORY` environment variable when *creating and running the PMM Server container*. To set the environment variable, use the `-e` option. The value must be passed in kilobytes. For example, to set the limit to 4 GB of memory:

```
-e METRICS_MEMORY=4194304
```

Note: The limit affects only memory reserved for data chunks. Actual RAM usage by Prometheus is higher. It is recommended to have at least three times more memory than the expected memory taken up by data chunks.

3.3.4 How to control data retention for Prometheus?

By default, Prometheus in PMM Server stores 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 time for Prometheus by passing the `METRICS_RETENTION` environment variable when *creating and running the PMM Server container*. To set the environment variable, use the `-e` option. The value is passed as a combination of hours, minutes, and seconds. For example, the default value of 30 days is `720h0m0s`. You probably do not need to be more precise than the number hours, so you can discard the minutes and seconds. For example, to decrease the retention period to 8 days:

```
-e METRICS_RETENTION=192h
```

3.3.5 Where are the services created by PMM Client?

When you add a monitoring instance using the `pmm-admin` tool, it creates a corresponding service. The name of the service has the following syntax: `pmm-<type>-<port>`

For example: `pmm-mysql-metrics-42002`.

The location of the services depends on the service manager:

Service manager	Service location
`systemd`	`/etc/systemd/system/`
`upstart`	`/etc/init/`
`systemv`	`/etc/init.d/`

To see which service manager is used on your system, run `sudo pmm-admin info`.

3.3.6 Where is DSN stored?

Every service created by `pmm-admin` when you add a monitoring instance gets a DSN from the credentials provided, auto-detected, or created (when adding the instance with the `--create-user` option).

For MySQL and MongoDB metrics instances (`mysql:metrics` and `mongodb:metrics` services), the DSN is stored with the corresponding service files. For more information, see *Where are the services created by PMM Client?*.

For QAN instances (`mysql:queries` service), the DSN is stored in local configuration files under `/usr/local/percona/qan-agent`.

Also, a sanitized copy of DSN (without the passowrd) is stored in Consul API for information purposes (used by the `pmm-admin list` command).

3.3.7 Where are PMM Client log files located?

Every service created by `pmm-admin` when you add a monitoring instance has a separate log file located in `/var/log/`. The file names have the following syntax: `pmm-<type>-<port>.log`.

For example, the log file for the QAN monitoring service is `/var/log/pmm-mysql-queries-0.log`.

You can view all available monitoring instance types and corresponding ports using the `pmm-admin list` command. For more information, see *Listing monitored instances*.

3.3.8 What are common performance considerations?

If a MySQL server has a lot of schemas or tables, it is recommended to disable per table metrics when adding the instance:

```
$ sudo pmm-admin add mysql --disable-tablestats
```

Note: Table statistics are disabled automatically if there are over 1 000 tables.

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

3.3.9 Can I stop all services at once?

Yes, you can use `pmm-admin` to start and stop either individual services that correspond to the added monitoring instances, or all of them at once.

To stop all services:

```
$ sudo pmm-admin stop --all
```

To start all services:

```
$ sudo pmm-admin start --all
```

For more information about starting and stopping services, see *Starting and stopping metric services*.

You can view all available monitoring instances and the states of the corresponding services using the `pmm-admin list` command. For more information, see *Listing monitored instances*.

3.3.10 What privileges are required to monitor a MySQL instance?

When adding MySQL instance to monitoring, you can specify the MySQL server superuser account credentials, which has all privileges. However, monitoring with the superuser account is not secure. If you also specify the `--create-user` option, it will create a user with only the necessary privileges for collecting data.

You can also set up the `pmm` user manually with 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 --user root --password root --create-user
```

The superuser credentials are required only to set up the pmm user with necessary privileges for collecting data. If you want to create this user yourself, the following privileges are required:

```
GRANT SELECT, PROCESS, SUPER, REPLICATION CLIENT, RELOAD ON *.* TO 'pmm'@' localhost' IDENTIFIED BY '
GRANT SELECT, UPDATE, DELETE, DROP ON performance_schema.* TO 'pmm'@' localhost';
```

If the pmm user already exists, simply pass its credential when you add the instance:

```
$ sudo pmm-admin add mysql --user pmm --password pass
```

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

3.3.11 Can I monitor multiple MySQL instances?

Yes, you can add multiple MySQL instances to be monitored from one *PMM Client*. In this case, you will need to provide a distinct port and socket for each instance using the `--port` and `--socket` variables, 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 --user root --password root --create-user --port 3001 instance-01
$ sudo pmm-admin add mysql --user root --password root --create-user --port 3002 instance-02
```

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

3.3.12 Can I rename instances?

You can remove any monitoring instance as described in *Removing monitoring services* 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.

3.3.13 Can I use non-default ports for instances?

When you add an instance with the `pmm-admin` tool, it creates a corresponding service that listens on a predefined client port:

General OS metrics	`linux:metrics`	42000
MySQL metrics	`mysql:metrics`	42002
MongoDB metrics	`mongodb:metrics`	42003
ProxySQL metrics	`proxysql:metrics`	42004

If a default port for the service is not available, `pmm-admin` automatically chooses a different one.

If you want to assign a different port, use the `--service-port` option when *adding instances*.

3.3.14 What resolution is used for metrics?

The `mysql:metrics` service collects metrics with different resolutions (1 second, 5 seconds, and 60 seconds)

The `linux:metrics` and `mongodb:metrics` services are set up to collect metrics 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 passing the `METRICS_RESOLUTION` environment variable when *creating and running the PMM Server container*. To set this environment variable, use the `-e` option. The values can be between `1s` (default) and `5s`. If you set a higher value, Prometheus will not start.

For example, to set the minimum resolution to 3 seconds:

```
-e METRICS_RESOLUTION=3s
```

Note: Consider increasing minimum resolution when *PMM Server* and *PMM Client* are on different networks, or when *Using PMM with Amazon RDS*.

3.3.15 Why do I get `Failed ReadTopologyInstance` error when adding MySQL host to Orchestrator?

You need to create Orchestrator's topology user on MySQL according to *this section*.

3.4 Glossary

Metrics Monitor (MM) Component of *PMM Server* that provides a historical view of metrics critical to a MySQL server instance.

PMM Client Collects MySQL server metrics, general system metrics, and query analytics data for a complete performance overview. Collected data is sent to *PMM Server*.

For more information, see *Percona Monitoring and Management Architecture*.

PMM Server Aggregates data collected by *PMM Client* and presents it in the form of tables, dashboards, and graphs in a web interface.

For more information, see *Percona Monitoring and Management Architecture*.

Query Analytics (QAN) Component of *PMM Server* that enables you to analyze MySQL query performance over periods of time.

M

P

Q