



PERCONA

www.percona.com

Percona Monitoring and Management Documentation

Release 1.17.0

Percona LLC and/or its affiliates 2009-2018

Nov 20, 2018

CONTENTS

I Basics	3
II Advanced	59
III Reference	177

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.

Part I

Basics

OVERVIEW OF PERCONA MONITORING AND MANAGEMENT ARCHITECTURE

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

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

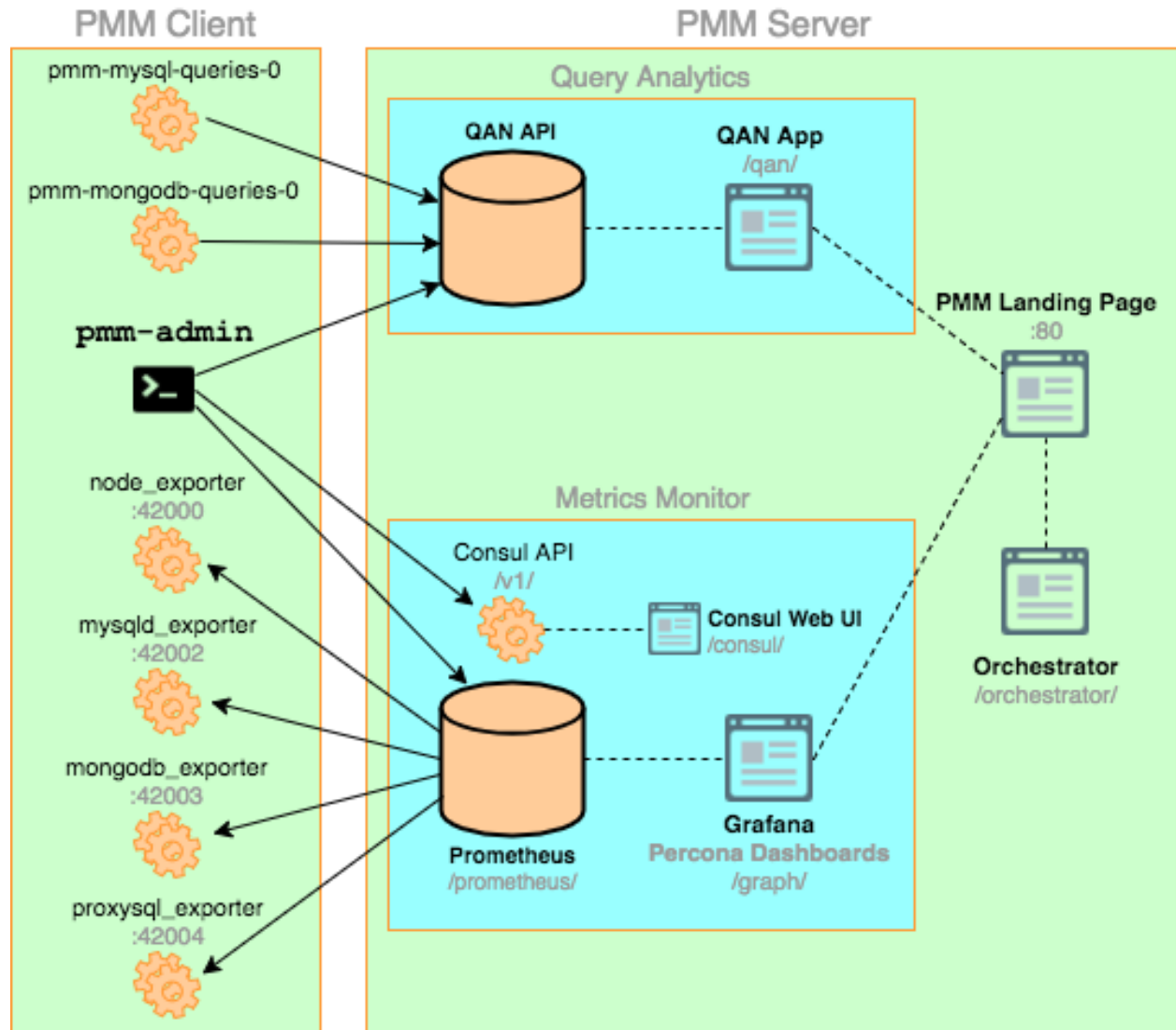
The modules are packaged for easy installation and usage. It is assumed that the user should not need to understand what are the exact tools that make up each module and how they interact. However, if you want to leverage the full potential of PMM, the internal structure is important.

- *PMM Client*
 - *PMM Server*
 - *Orchestrator*

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:



PMM Client

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

The PMM Client package consist of the following:

- `pmm-admin` is a command-line tool for managing PMM Client, for example, adding and removing database instances that you want to monitor. For more information, see *Managing PMM Client*.
- `pmm-mysql-queries-0` is a service that manages the QAN agent as it collects query performance data from MySQL and sends it to the QAN API on *PMM Server*.
- `pmm-mongodb-queries-0` is a service that manages the QAN agent as it collects query performance data from MongoDB and sends it to QAN API on *PMM Server*.
- `node_exporter` is a Prometheus exporter that collects general system metrics.
- `mysqld_exporter` is a Prometheus exporter that collects MySQL server metrics.

- `mongodb_exporter` is a Prometheus exporter that collects MongoDB server metrics.
- `proxysql_exporter` is a Prometheus exporter that collects ProxySQL performance metrics.

See also:

How to install PMM Client [Installing Clients](#)

How to pass exporter specific options when adding a monitoring service [Passing options to the exporter](#)

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

PMM Server

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

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

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

PMM Server includes the following tools:

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

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](#).
 - * 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](#).

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

See also:

Default ports [Ports](#) in [Terminology Reference](#)

Enabling orchestrator *Orchestrator* in *Terminology Reference*

Orchestrator

Orchestrator is a MySQL replication topology management and visualization tool. If it is enabled, you can access it using the `/orchestrator` URL after PMM Server address. Alternatively, you can click the *MySQL Replication Topology Manager* button on the 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_client_password';
```

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 `ORCHESTRATOR_PASSWORD` and `ORCHESTRATOR_USER` options.

```
$ docker run ... -e ORCHESTRATOR_ENABLED=true ORCHESTRATOR_USER=name -e ORCHESTRATOR_  
->PASSWORD=pass ... percona/pmm-server:latest
```

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

Note: Orchestrator is not enabled by default starting with PMM 1.3.0

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.

In version 1.3.0 and later, Orchestrator is not enabled by default. To enable it, see *Additional options* in the *Running PMM Server via Docker* section.

DEPLOYING *PERCONA MONITORING AND MANAGEMENT*

PMM (Percona Monitoring and Management) is designed to be scalable for various environments. If you have just one MySQL or MongoDB server, you can install and run both PMM server and PMM clients on one database host.

It is more typical to have several MySQL and MongoDB server instances distributed over different hosts. In this case, you need to install the PMM client package on each database host that you want to monitor. In this scenario, the PMM server is set up on a dedicated monitoring host.

In this chapter

- [Installing PMM Server](#)
- [Installing Clients](#)
- [Connecting PMM Clients to the PMM Server](#)
- [Collecting Data from PMM Clients on PMM Server](#)
- [Obtaining Diagnostics Data for Support](#)
- [Updating](#)
- [Uninstalling PMM Components](#)

Installing PMM Server

To install and set up the PMM Server, use one of the following options:

Running PMM Server via Docker

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

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

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

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

Important: By default, *retention* is set to 30 days for Metrics Monitor and to 8 days for *PMM Query Analytics*. Also consider *disabling table statistics*, which can greatly decrease Prometheus database size.

Setting Up a Docker Container for PMM Server

- *Simplified Automatic Installation*
- *Manual Installation*
 - Pulling the PMM Server Docker Image
 - Creating the pmm-data Container
 - Creating and Launching the PMM Server Container
 - Installing and using specific docker version
 - Additional options

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

PMM Server deployment via Docker can be done either automatically with a special installation script, or manually. Both variants are covered in the following sections.

Simplified Automatic Installation

Automatic installation of the PMM Server is done with a special script, `get-pmm.sh`. This script should be downloaded and executed, for example with the following two commands:

```
curl -fsSL https://raw.githubusercontent.com/percona/pmm/master/get-pmm.sh -o get-  
pmm.sh  
sh get-pmm.sh
```

This script should be executed under root user, or under non-root user either with rights to run docker containers, or with privileges to run `sudo` command.

Being executed, `get-pmm.sh` performs following actions:

- checking if Docker is installed, and trying to install it if not,
- running Docker if necessary,
- downloading the PMM Server image,
- generating the necessary `pmm-data` container,
- configuring and starting the PMM Server container.

Manual Installation

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

Pulling the PMM Server Docker Image

To pull the latest version from Docker Hub:

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

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

Creating the 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:latest /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:latest` is the name and version tag of the image to derive the container from.
- `/bin/true` is the command that the container runs.

Important: Make sure that the data volumes that you initialize with the `-v` option match those given in the example. PMM Server expects that those directories are bind mounted exactly as demonstrated.

Creating and Launching the PMM Server Container

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

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

This command does the following:

- The **docker run** command runs a new container based on the `percona/pmm-server:latest` image.
- The `-d` option starts the container in the background (detached mode).
- 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 `-v` option mounts volumes from the `pmm-data` container (see *Creating the pmm-data Container*).
- The `--name` option assigns a custom name to the container that you can use to reference the container within the Docker network. In this case: `pmm-server`.
- The `--restart` option defines the container's restart policy. Setting it to `always` ensures that the Docker daemon will start the container on startup and restart it if the container exits.
- `percona/pmm-server:latest` is the name and version tag of the image to derive the container from.

Installing and using specific docker version

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

For example, installing version 1.14.1 with disabled update button in the web interface would look as follows:

```
$ docker create \
  -v /opt/prometheus/data \
  -v /opt/consul-data \
  -v /var/lib/mysql \
  -v /var/lib/grafana \
  --name pmm-data \
  percona/pmm-server:1.14.1 /bin/true

$ docker run -d \
  -p 80:80 \
  --volumes-from pmm-data \
  --name pmm-server \
  -e DISABLE_UPDATES=true \
  --restart always \
  percona/pmm-server:1.14.1
```

Additional options

When running the PMM Server, you may pass additional parameters to the **docker run** subcommand. All options that appear after the `-e` option are the additional parameters that modify the way how PMM Server operates.

The section *PMM Server Additional Options* lists all supported additional options.

See also:

Default ports *Ports in Terminology Reference*

Updating PMM *Updating PMM*

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

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

Updating PMM Server Using Docker

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

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

```
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                                CREATED        STATUS
↳ About an hour ago    480696cd4187   percona/pmm-server:1.4.0             "/opt/entrypoint.sh"   4 weeks ago   Up
↳ About an hour ago    192.168.100.1:80->80/tcp, 443/tcp    pmm-server
```

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

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

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

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

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

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

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

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

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

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

Pulling a new Docker Image

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

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

This example shows how to pull a specific version:

```
$ docker pull percona/pmm-server:1.5.0
```

This example shows how to pull the `latest` version:

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

Creating a new Docker container based on the new image

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

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

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

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

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

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

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

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

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

Removing the backup container

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

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

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

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

Restoring the previous version

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

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

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

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

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

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

See also:

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

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

Updating the PMM Server and the PMM Client *Updating* section.

Backing Up PMM Data from the Docker Container

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

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

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

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

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

2. Create the essential sub directories:

```
$ mkdir -p opt/prometheus
$ mkdir -p var/lib
```

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

1. Stop the docker container:

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

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

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

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

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

See also:

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

Updating PMM Server run via Docker *Updating PMM Server Using Docker*

Restoring the Backed Up Information to the PMM Data Container

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

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

1. Stop the running pmm-server container.

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

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

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

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

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

4. Create a new pmm-data container

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

Important: The last step creates a new pmm-data container based on the percona/pmm-server:latest image. If you do not intend to use the latest tag, specify the exact version instead, such as **1.5.0**. You can find all available versions of pmm-server images at [percona/pmm-server](#).

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

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

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

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

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

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

3. Apply correct ownership to pmm-data files:

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

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

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

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

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

See also:

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

Updating PMM *Updating PMM*

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

PMM Server as a Virtual Appliance

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

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

In this chapter

- Supported Platforms for Running the PMM Server Virtual Appliance
- Identifying PMM Server IP Address

- [Accessing PMM Server](#)
- [Accessing the Virtual Machine](#)
- [Next Steps](#)

Supported Platforms for Running the PMM Server Virtual Appliance

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

The virtual machine used for the appliance runs CentOS 7.

Warning: The appliance must run in a network with DHCP, which will automatically assign an IP address for it. To assign a static IP manually, you need to acquire the root access as described in [How to set the root password when PMM Server is installed as a virtual appliance](#). Then, see the documentation for the operating system for further instructions: [Configuring network interfaces in CentOS](#)

Instructions for setting up the virtual machine for different platforms

VirtualBox Using 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-1.6.0.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 (without the console) mode.

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.

```
# Import image
VBoxManage import pmm-server-|VERSION NUMBER|.ova

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

# Log console output into file
VBoxManage modifyvm 'PMM Server [VERSION NUMBER]' --uart1 0x3F8 4 --uartmodel file /
↪ tmp/pmm-server-console.log

# Start instance
VBoxManage startvm --type headless 'PMM Server [VERSION NUMBER]'

# Wait for 1 minute and get IP address from the log
```

```
sleep 60
grep cloud-init /tmp/pmm-server-console.log
```

In this script, [VERSION NUMBER] is the placeholder of the version of PMM Server that you are installing. By convention **OVA** files start with *pmm-server-* followed by the full version number such as 1.17.0.

To use this script, make sure to replace this placeholder with the the name of the image that you have downloaded from the [Download Percona Monitoring and Management](#) site. This script also assumes that you have changed the working directory (using the `cd` command, for example) to the directory which contains the downloaded image file.

See also:

Using PMM Server as a virtual appliance *PMM Server as a Virtual Appliance*

Setting the root password *How to set the root password when PMM Server is installed as a virtual appliance*

Downloading the latest development version

VirtualBox Using the GUI

The following procedure describes how to run the PMM Server appliance using the graphical user interface of VirtualBox:

1. Download the OVA. The latest version is available at the [Download Percona Monitoring and Management](#) site.
2. Import the appliance. For this, open the *File* menu and click *Import Appliance* and specify the path to the OVA and click *Continue*. Then, 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*, so do not set the network adapter to NAT.

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 and set the root password (required on the first login).

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

See also:

Using PMM Server as a virtual appliance *PMM Server as a Virtual Appliance*

Setting the root password *How to set the root password when PMM Server is installed as a virtual appliance*

VMware Workstation Player

The following procedure describes how to run the *PMM Server* appliance using VMware Workstation Player:

1. Download the OVA. The latest version is available at the [Download Percona Monitoring and Management](#) site.
2. Import the appliance.
 - (a) Open the *File* menu and click *Open*.

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

Note: You may get an error indicating that import failed. Simply click *Retry* and import should succeed.

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

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

4. Start the PMM Server appliance and set the root password (required on the first login)

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

5. Set the root password as described in the section

See also:

Using PMM Server as a virtual appliance [PMM Server as a Virtual Appliance](#)

Setting the root password [How to set the root password when PMM Server is installed as a virtual appliance](#)

Identifying PMM Server IP Address

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

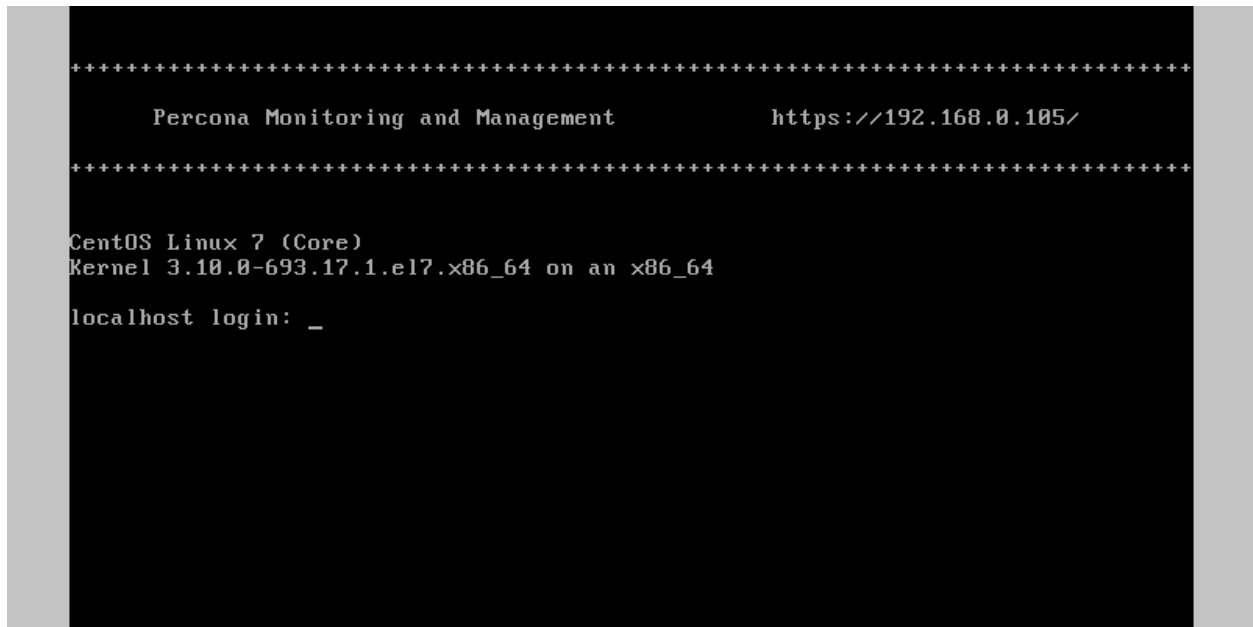
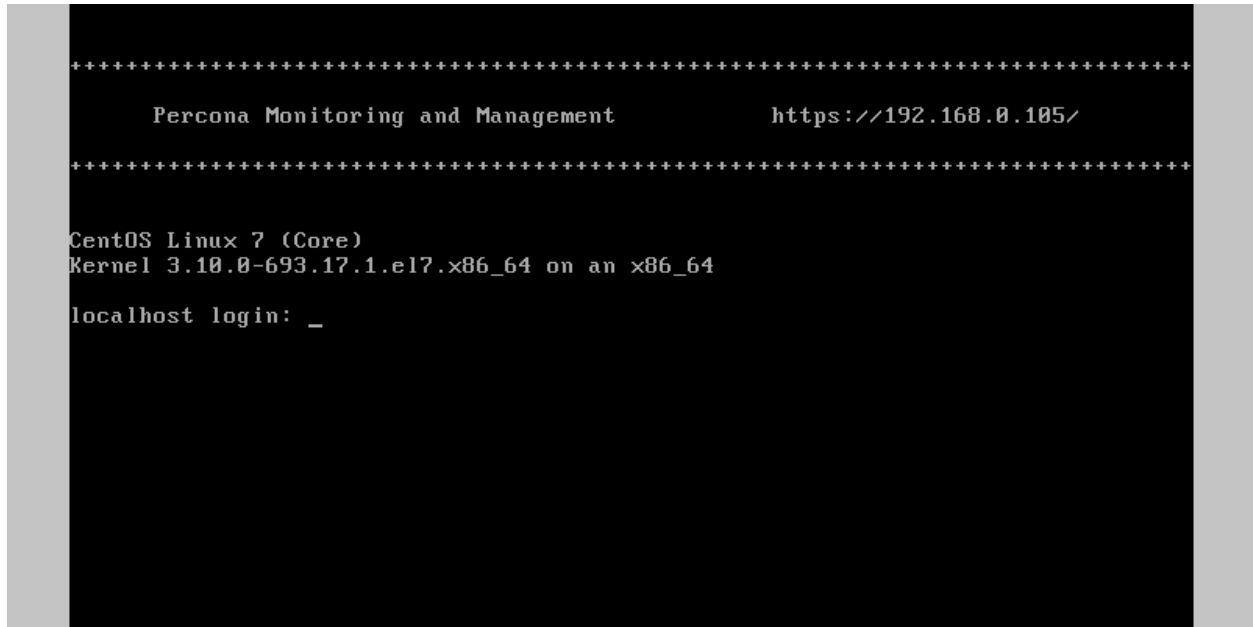


Fig. 2.1: The IP address appears above the login prompt.

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

Accessing PMM Server

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



```

*****
Percona Monitoring and Management          https://192.168.0.105/
*****

CentOS Linux 7 (Core)
Kernel 3.10.0-693.17.1.el7.x86_64 on an x86_64

localhost login: _

```

Fig. 2.2: The URL to open in the browser appears at the top of the terminal when running the virtual machine.

If you run PMM Server in your browser for the first time, you are requested to supply the user and a new password. Optionally, you may also provide your SSH public key.

Click *Submit* and enter your user name and password in the dialog window that pops up. The PMM Server is now ready and the home page opens.

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

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

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

```
$ pmm-admin config --username= --password= --server=1.2.3.4
```

Accessing the Virtual Machine

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

1. Open the URL for accessing PMM in a web browser.
The URL is provided either in the console window or in the appliance log.
2. Submit your **public key** in the PMM web interface.

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

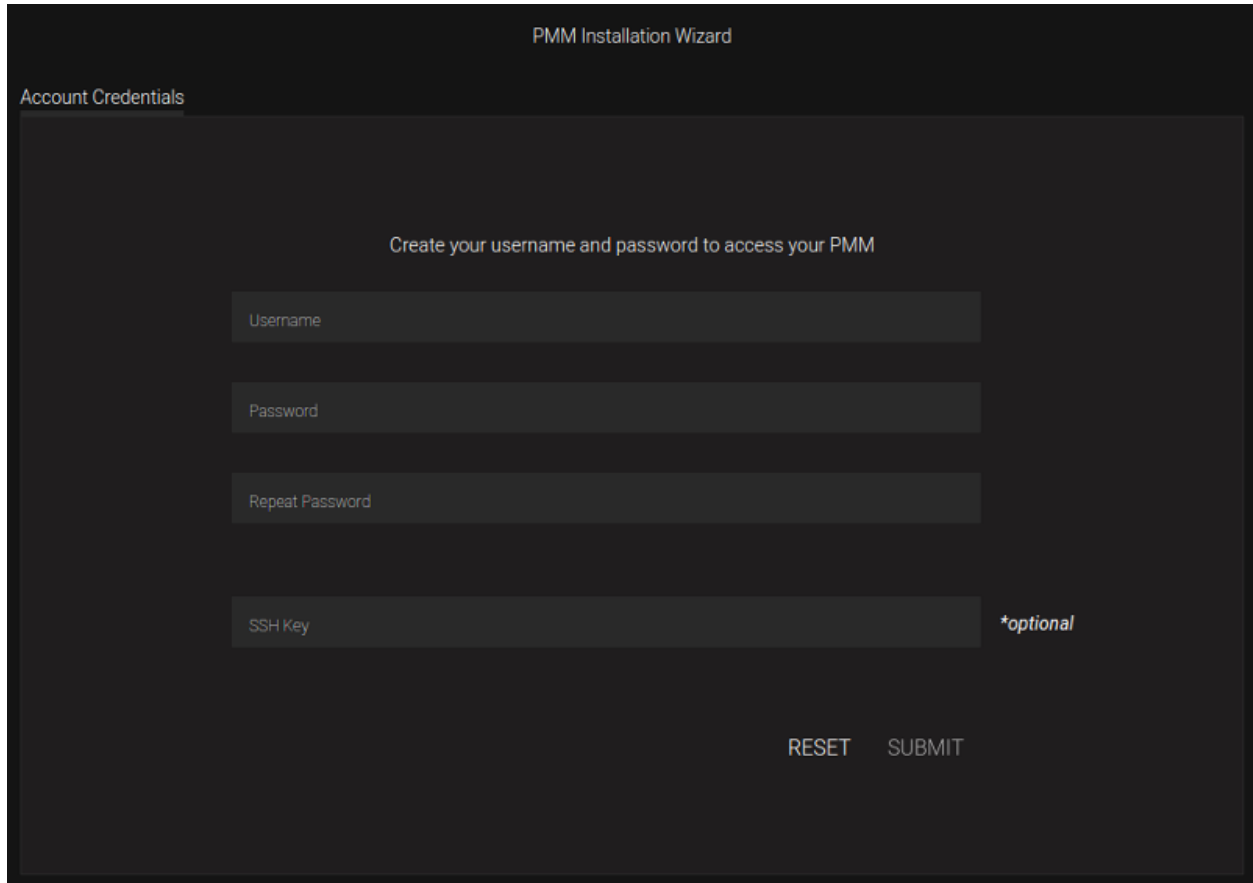


Fig. 2.3: Set the user and password to access the PMM Server web interface.

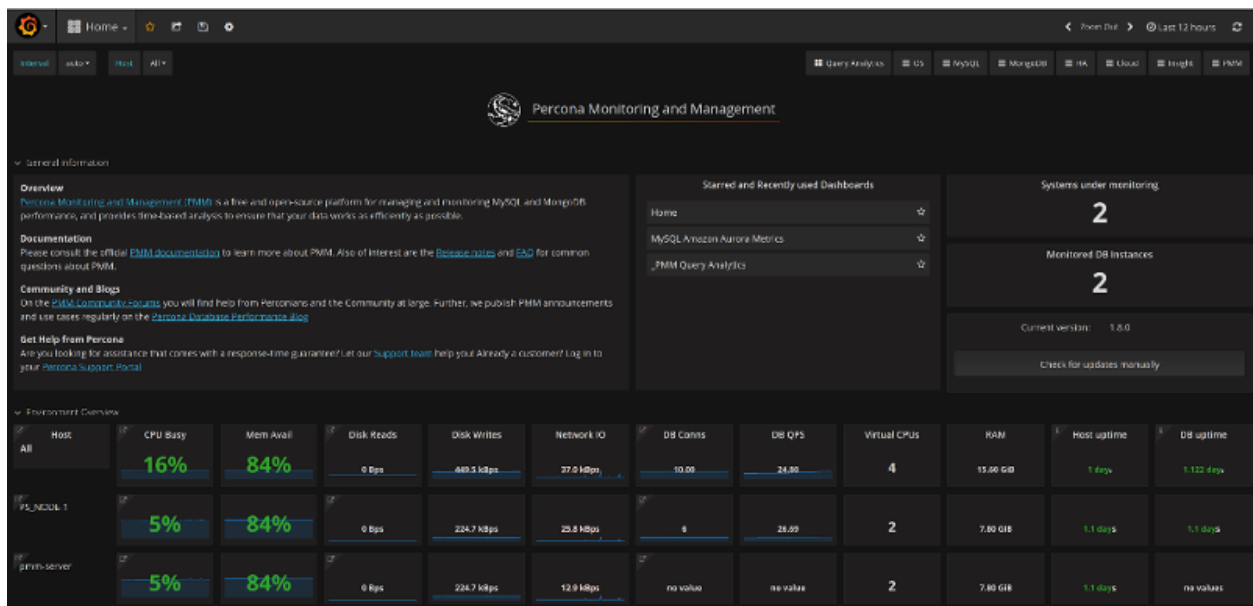


Fig. 2.4: PMM Server home page

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

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 AWS Marketplace

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

Percona Monitoring and Management Server

Sold by: [Percona](#)

pmm-server

Linux/Unix ☆☆☆☆☆ (0)

[Continue to Subscribe](#)

[Save to List](#)

Typical Total Price
\$0.100/hr

Total pricing per instance for services hosted on m4.large in US East (N. Virginia). [View Details](#)

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

Product Overview

Percona Monitoring and Management (PMM) is a free and open-source platform for managing and monitoring MySQL and MongoDB performance. You can run PMM in your own environment for maximum security and reliability. It provides thorough time-based analysis for MySQL, MariaDB and MongoDB servers to ensure that your data works as efficiently as possible.

Version	
Sold by	Percona
Video	See Product Video
Categories	Monitoring
Operating System	Linux/Unix, CentOS 7
Fulfillment Methods	Amazon Machine Image

Highlights

- Query Analytics (QAN) enables you to analyze MySQL query performance over periods of time
- Metrics Monitor (MM) provides a historical view of metrics that are critical to a MySQL or MongoDB server instance
- Orchestrator is a MySQL replication topology management and visualization tool

Fig. 2.5: The home page of PMM in AWS Marketplace.

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

The *Pricing Information* section allows to select your region and choose an instance type in the table that shows the pricing for the software and infrastructure hosted in the region you have selected (the recommended EC2 instance type is preselected for you). Note that actual choice will be done later, and this table serves the information purposes, to plan costs.

Note: Disk space consumed by PMM Server depends on the number of hosts under monitoring. Each environment

Pricing Information

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

Estimating your costs

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

Region: US East (N. Virginia)

Fulfillment Option: 64-bit Amazon Machine Image (AMI)

Software Pricing Details

Percona Monitoring and Management Server \$0 /hr savings on EC2 instance

Infrastructure Pricing Details

Estimated Infrastructure Cost: \$0.046 EC2/hr >

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

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

EC2 instance type	Software	EC2/hr	Total
t2.micro	\$0	\$0.012	\$0.012
t2.small	\$0	\$0.023	\$0.023
t2.medium	\$0	\$0.046	\$0.046
t2.large	\$0	\$0.093	\$0.093
t2.xlarge	\$0	\$0.186	\$0.186
t2.2xlarge	\$0	\$0.371	\$0.371

Fig. 2.6: As soon as you select your region, you can choose the EC2 instance in it and see its price. PMM comes for no cost, you may only need to pay for the infrastructure provided by Amazon.

will be unique, however consider modeling your data consumption based on PMM Demo <<https://pmmdemo.percona.com/>>_ web site, which consumes ~230MB/host/day, or ~6.9GB/host

at the default 30 day retention period. See [this blog post](#) for more details.

Clicking the *Continue to Subscribe* button will proceed to the terms and conditions page. Clicking *Continue to Configuration* there will bring a new page to start setting up your instance.

Launch this software

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

Configuration Details

Fulfillment Option: 64-bit Amazon Machine Image (AMI)
Percona Monitoring and Management Server
running on m4.large

Software Version

Region: US East (N. Virginia)

Choose Action

Launch from Website Choose this action to launch from this website

EC2 Instance Type

m4.large

Memory: 8 GiB
CPU: 6.5 EC2 Compute Units (2 Virtual cores with 3.25 Units each)
Storage: EBS storage only
Network Performance: Moderate

Fig. 2.7: Percona Monitoring and Management on AWS Marketplace - launch options.

Available launch options in the drop-down menu include *Launch from Website* and *Launch through EC2*. The first one is a quick way to make your instance ready. For more control, use the Manual Launch through EC2 option.

In this chapter

- Setting Up a PMM Instance Using the website GUI
- Running PMM Server Using Amazon Machine Images

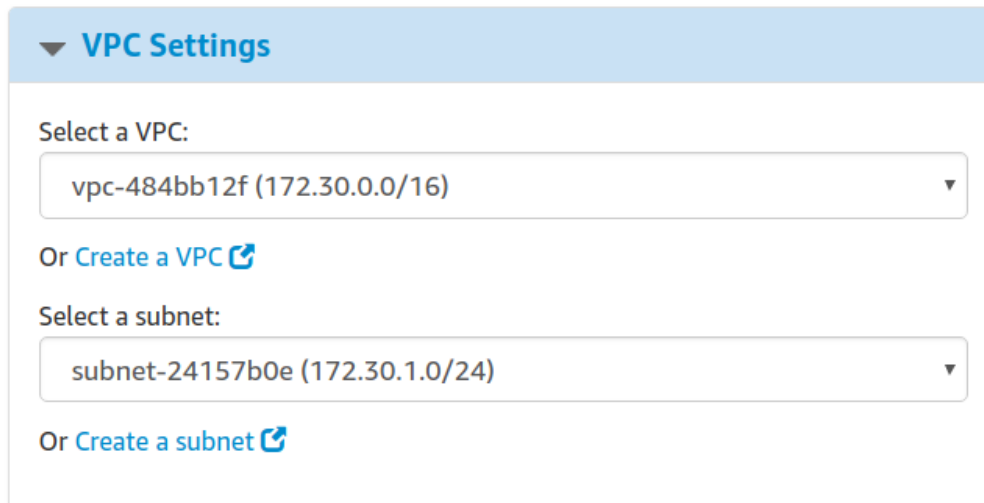
Setting Up a PMM Instance Using the website GUI

Choose *Launch from Website* option, your region, and the EC2 instance type on the launch options page. On the previous screenshot, we use the US East (N. Virginia) region and the *EC2 Instance Type* named `m4.large`. To reduce cost, you need to choose the region closest to your location.

When all choices are done, click the *Continue to Launch* button to proceed.

Setting up a VPC and an EC2 Instance Type

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



▼ VPC Settings

Select a VPC:

vpc-484bb12f (172.30.0.0/16) ▼

Or [Create a VPC](#)

Select a subnet:

subnet-24157b0e (172.30.1.0/24) ▼

Or [Create a subnet](#)

Fig. 2.8: Select VPC in the VPC Settings section.

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

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

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

See also:

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

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

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

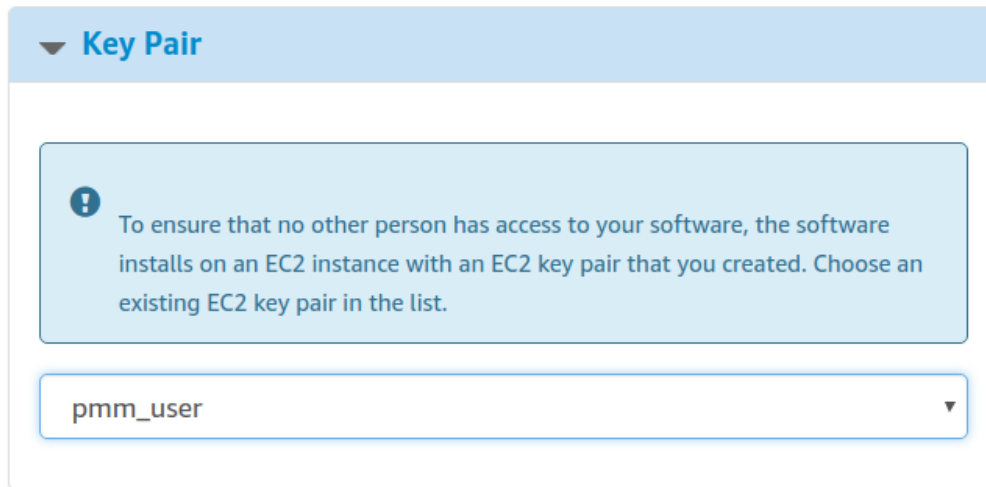


Fig. 2.9: Select an already existing key pair (use the EC2 console to create one if necessary)

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

See also:

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

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

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

Applying settings

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

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

Adjusting instance settings in the EC2 Console

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

▼ **Security Group**

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

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

Create new based on seller settings ▼

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

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

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

Fig. 2.10: Select a security group which manages firewall settings.

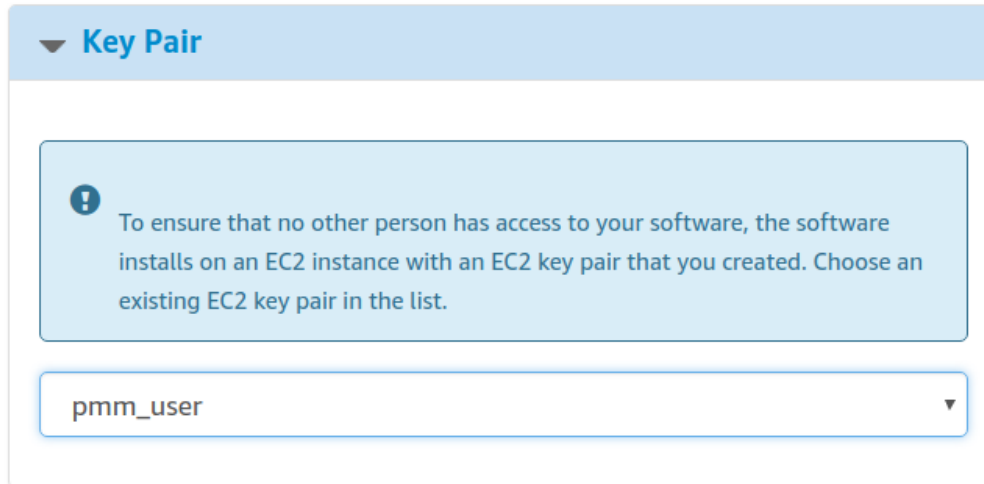


Fig. 2.11: Your instance settings are summarized in a special area. Click the Launch with 1 click button to continue.

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

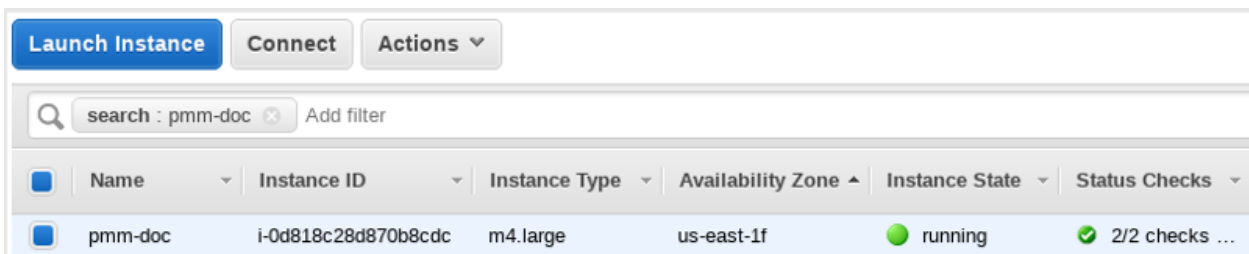


Fig. 2.12: The newly created instance selected.

Running the instance

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

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

See also:

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

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

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



Fig. 2.13: The public IP address of the instance

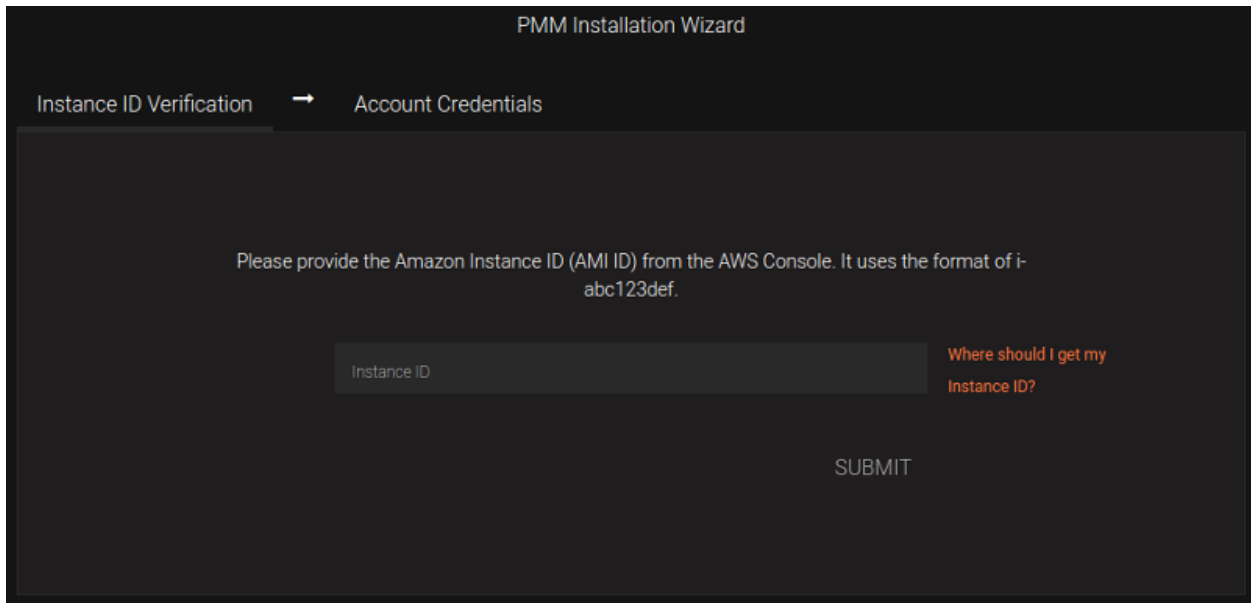


Fig. 2.14: Entering the instance ID when installing PMM Server

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



Fig. 2.15: Hover the cursor over the instance ID for the Copy button to appear.

Paste the instance in the *Instance ID* field of the *Percona Monitoring and Management* welcome page and click *Submit*. Click *Submit* and enter your user name and password in the dialog window that pops up. The PMM Server is now ready and the home page opens.

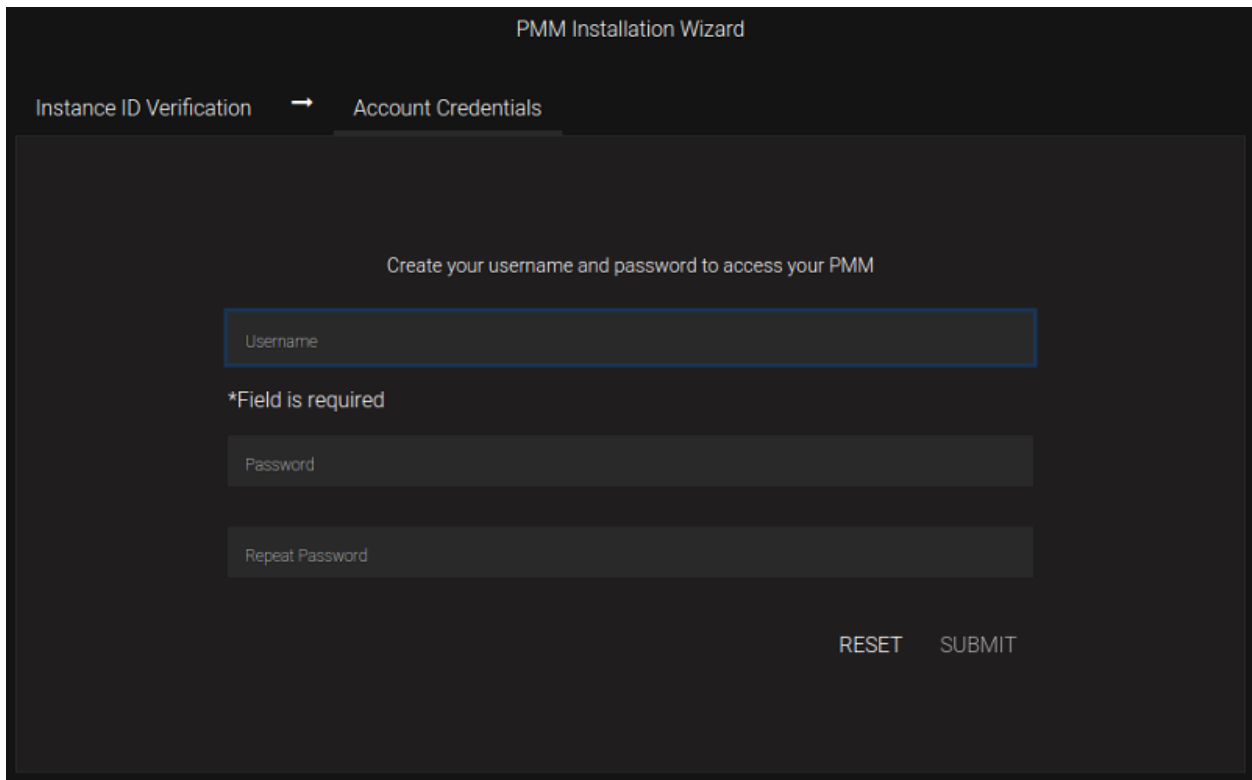


Fig. 2.16: Create credentials for your instance.

Click *Submit* and enter your user name and password in the dialog window that pops up. The PMM Server is now ready and the home page opens.

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

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

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

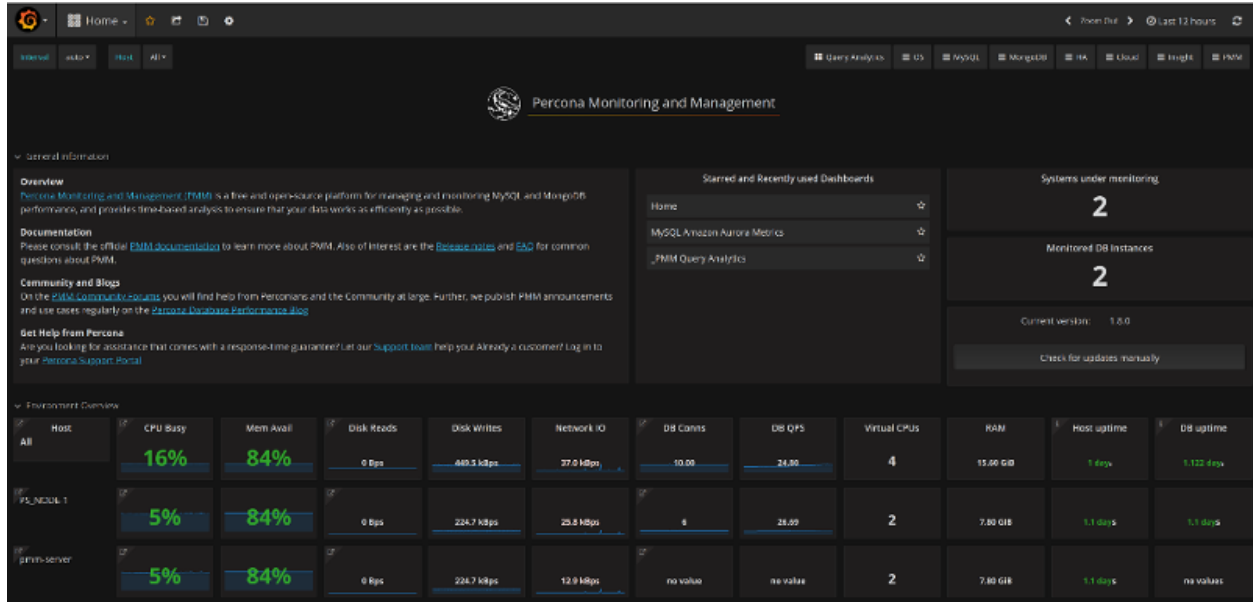


Fig. 2.17: PMM Server home page

```
$ pmm-admin config --username= --password= --server=1.2.3.4
```

Note: Accessing the instance by using an SSH client.

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

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

See also:

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

How to connect a PMM Client to the PMM Server? [Connecting PMM Clients to the PMM Server](#)

Resizing the EBS Volume

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

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

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

More information in AWS documentation

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

Amazon EC2 Security Groups for Linux Instances <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html>

Connecting to Your Linux Instance Using SSH (use `admin` as the user name) <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html>

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	String	AMI ID
Asia Pacific (Tokyo)	ap-northeast-1	ami-081b63c019880c9d6
Asia Pacific (Seoul)	ap-northeast-2	ami-0299fd006902387fb
Asia Pacific (Mumbai)	ap-south-1	ami-0ae6d88793bfd1c25
Asia Pacific (Singapore)	ap-southeast-1	ami-0b75d0daff534821b
Asia Pacific (Sydney)	ap-southeast-2	ami-066240e0e538a2d5b
Canada (Central)	ca-central-1	ami-043636fe79bd57dcf
EU (Frankfurt)	eu-central-1	ami-0f11e38faf1e13b95
EU (Ireland)	eu-west-1	ami-01144c4acc8b6e1f2
EU (London)	eu-west-2	ami-0f83e56754f334842
EU (Paris)	eu-west-3	ami-0c969dd2dc2bd59da
South America (São Paulo)	sa-east-1	ami-0e4a9988906825cdf
US East (N. Virginia)	us-east-1	ami-0c0b294064bea85d0
US East (Ohio)	us-east-2	ami-0125f42891b1de6e0
US West (N. California)	us-west-1	ami-0284d9d12f7fe0959
US West (Oregon)	us-west-2	ami-036f591241f108627

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-30ad0f4d \  
  --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
```

Important: On each computer where PMM Client is installed, the *following ports* must be open. These are default ports that you can change when adding the respective monitoring service with the `pmm-admin add` command.

See also:

Improving security *Security Features in Percona Monitoring and Management*

Verifying PMM Server

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

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

Accessing the Components of the Web Interface

Component	URL
<i>PMM Home Page</i>	<code>http://192.168.100.1</code>
<i>Metrics Monitor (MM)</i>	<code>http://192.168.100.1/graph/</code> User name: admin Password: admin
Orchestrator	<code>http://192.168.100.1/orchestrator</code>

You can also check if PMM Server is available requesting the `/ping` URL as in the following example:

```
$ curl http://192.168.100.1/ping
{'version': '1.8.0'}
```

Installing Clients

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

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

```
$ ping 192.168.100.1
```

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

Supported platforms

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

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

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

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

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

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

Storage requirements

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

Connecting PMM Clients to the PMM Server

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

To connect a PMM Client, enter the IP address of the PMM Server as the value of the `--server` parameter to the `pmm-admin config` command.

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

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

For example, if your PMM Server is running on `192.168.100.1`, and you have installed PMM Client on a machine with IP `192.168.200.1`, run the following in the terminal of your client. Run the following commands as root or by using the `sudo` command:

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

PMM Server      | 192.168.100.1
Client Name     | ubuntu-amd641
Client Address  | 192.168.200.1
```

If you change the default port **80** when *running PMM Server*, specify the new port number after the IP address of PMM Server. For example:

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

Important: On each computer where PMM Client is installed, the *following ports* must be open. These are default ports that you can change when adding the respective monitoring service with the `pmm-admin add` command.

See also:

Improving security *Security Features in Percona Monitoring and Management*

See also:

What other options can I pass to `pmm-admin config`? Run `pmm-admin config --help`

Collecting Data from PMM Clients on PMM Server

To start collecting data on each PMM Client connected to a PMM server, run the `pmm-admin add` command along with the name of the selected monitoring service.

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

Enable general system metrics, MySQL metrics, MySQL query analytics:

```
$ pmm-admin add mysql
```

Enable general system metrics, MongoDB metrics, and MongoDB query analytics:

```
$ pmm-admin add mongodb
```

Enable ProxySQL performance metrics:

```
$ pmm-admin add proxysql:metrics [NAME] [OPTIONS]
```

To see what is being monitored, run `pmm-admin list`. For example, if you enable general OS and MongoDB metrics monitoring, the output should be similar to the following:

```
$ pmm-admin list
```

```
...
```

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

SERVICE TYPE	NAME	LOCAL PORT	RUNNING	DATA SOURCE	OPTIONS
linux:metrics	mongo-main	42000	YES	-	
mongodb:metrics	mongo-main	42003	YES	localhost:27017	

See also:

What other monitoring services can I add using the `pmm-admin add` command? Run `pmm-admin add --help` in your terminal

Obtaining Diagnostics Data for Support

PMM Server is able to generate a set of files for enhanced diagnostics, which can be examined and/or shared with Percona Support to solve an issue faster.

Collected data are provided by the `logs.zip` service, and cover the following subjects:

- Prometheus targets
- Consul nodes, QAN API instances
- Amazon RDS and Aurora instances
- Version
- Server configuration
- Percona Toolkit commands

You can retrieve collected data from your PMM Server in a single zip archive using this URL:

```
https://<address-of-your-pmm-server>/managed/logs.zip
```

Updating

When changing to a new version of PMM, you update the PMM Server and each PMM Client separately.

Updating the PMM Server

The updating procedure of your PMM Server depends on the option that you selected for installing it.

If you are running PMM Server as a *virtual appliance* or using an *Amazon Machine Image*, use the *Check for Updates Manually* button on the Home dashboard (see *PMM Home Page*).

See also:

How to update PMM Server installed using Docker? *Updating PMM Server Using Docker*

Updating a PMM Client

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

Debian or Ubuntu

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

Red Hat or CentOS

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

If you installed your PMM client manually, *remove it* and then *download and install a newer version*.

Systems under monitoring

2

Monitored DB Instances

2

Current version: 1.8.0

Check for updates manually

Uninstalling PMM Components

Each PMM Client and the PMM Server are removed separately. First, remove all monitored services by using the `pmm-admin remove` command (see *Removing monitoring services*). Then you can remove each PMM Client and the PMM Server.

Removing the PMM Client

Remove all monitored instances as described in *Removing monitoring services*. Then, uninstall the `pmm-admin` package. The exact procedure of removing the PMM Client depends on the method of installation.

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

Using YUM

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

Using APT

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

Manually installed RPM package

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

Manually installed DEB package

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

Using the generic PMM Client tarball. `cd` into the directory where you extracted the tarball contents. Then, run the `uninstall` script:

```
$ ./uninstall
```

Removing the PMM Server

If you run your PMM Server using Docker, stop the container as follows:

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

To discard all collected data (if you do not plan to use PMM Server in the future), remove the `pmm-data` container:

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

If you run your PMM Server using a virtual appliance, just stop and remove it.

To terminate the PMM Server running from an Amazon machine image, run the following command in your terminal:

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

See also:

PMM Building Blocks *Overview of Percona Monitoring and Management Architecture*

About using the `pmm-admin add` command *Adding monitoring services*

TOOLS OF PMM

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`.

See also:

Installing PMM Server *Installing PMM Server*

The PMM home page that opens provides an overview of the environment that you have set up to monitor by using the `pmm-admin` tool.

From the PMM home page, you can access specific monitoring tools, or dashboards. Each dashboard features a collection of metrics. These are graphs of a certain type that represent one specific aspect showing how metric values change over time.

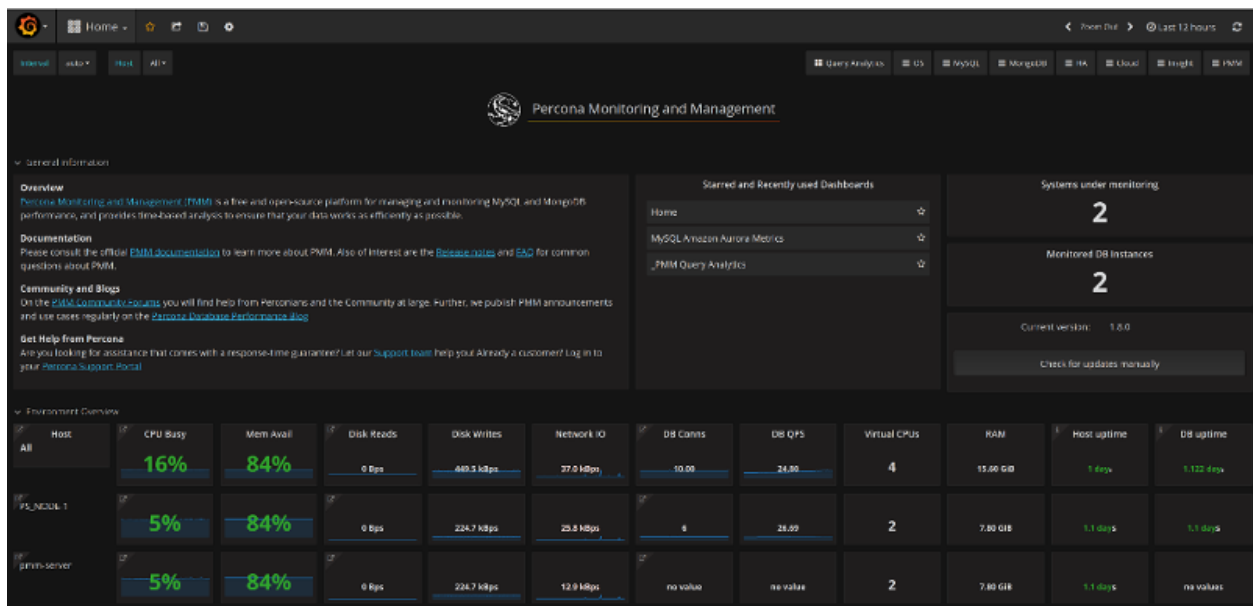


Fig. 3.1: The home page is an overview of your system

By default the PMM home page lists most recently used dashboards and helpful links to the information that may be useful to understand PMM better.

The PMM home page lists all hosts that you have set up for monitoring as well as the essential details about their performance such as CPU load, disk performance, or network activity.

More about PMM Components

PMM Query Analytics

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

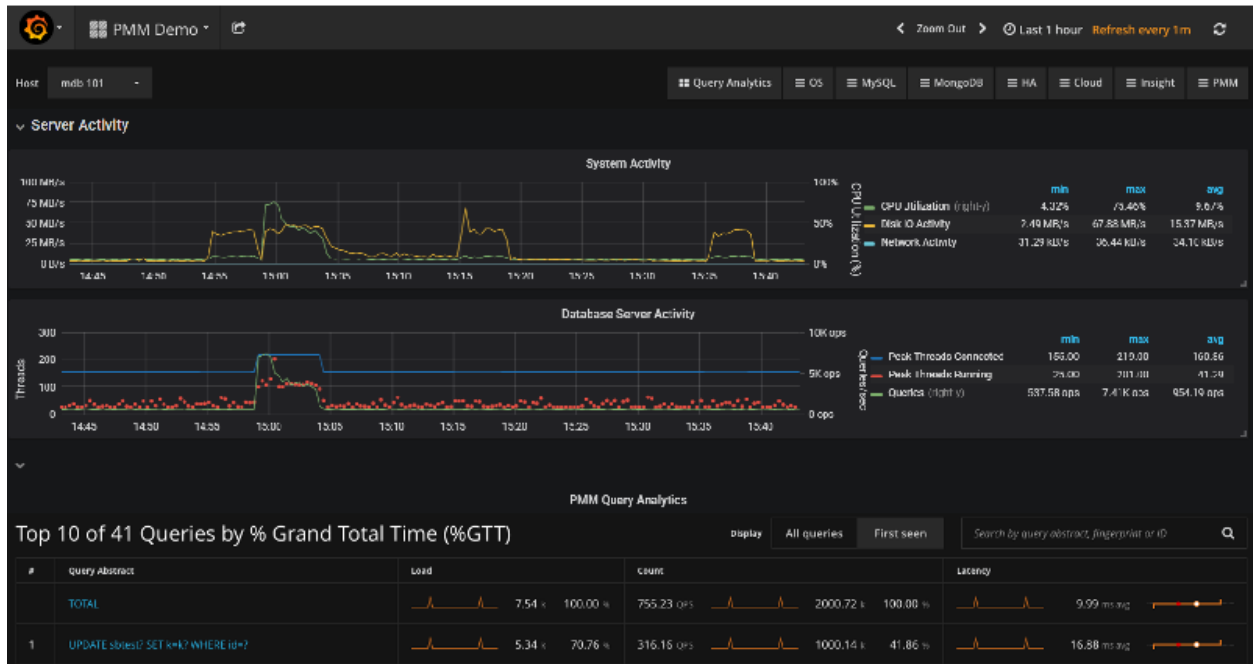


Fig. 3.2: QAN helps analyze database queries over periods of time and find performance problems.

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

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

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

- Opening QAN from the PMM Home Page
- Filtering Queries
 - Totals of the Query Summary
 - Queries in the Query Summary Table
 - Viewing a Specific Value of a Metric
- Zooming into a Query

- Query Section
- Explain Section
- Table Info Section
- Configuring QAN
 - Settings Tab
 - Status Tab
 - Log Tab
- QAN for MongoDB

See also:

A historical view of metrics that are critical to a database server [Metrics Monitor](#)

Overview of PMM Components [Tools of PMM](#)

Opening QAN from the PMM Home Page

To start working with QAN, open the list of dashboards on the PMM home page. Then, proceed to the *Host* field at the top of the page and select a host where the PMM Client is installed from the list of database instances.

The QAN dashboard will show a summary of the selected host and database activity metrics followed by the list of queries in a summary table. By default, QAN shows the top *ten* queries ranked by *%GTT* (Grand total time). Each query displays three essential metrics: *Load*, *Count*, and *Latency*. Also queries which are newly seen within the time range selected are highlighted with a blue color for quicker identification.

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

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

Fig. 3.3: Showing intervals for which data are unavailable yet.

To view more queries, click the *Load next 10 queries* button below the query summary table.

Filtering Queries

If you need to limit the list of available queries to only those that you are interested in, use the *Query Filter* field located above the query summary table.

In the *Query Filter* field, you can enter a query ID, query abstract, or query fingerprint. The ID is a unique signature of a query and looks like a long hexadecimal number. Note that each query in the summary table displays its ID in the *ID* column.

The query fingerprint is a simplified form of a query: all specific values are replaced with placeholders. You may enter only a fragment of the fingerprint to view all queries that contain that fragment in their fingerprints.

The query abstract is the portion of the query fingerprint which contains the type of the query, such as *SELECT* or *FIND*, and the attributes from the projection (a set of requested columns in case of MySQL database, for example).

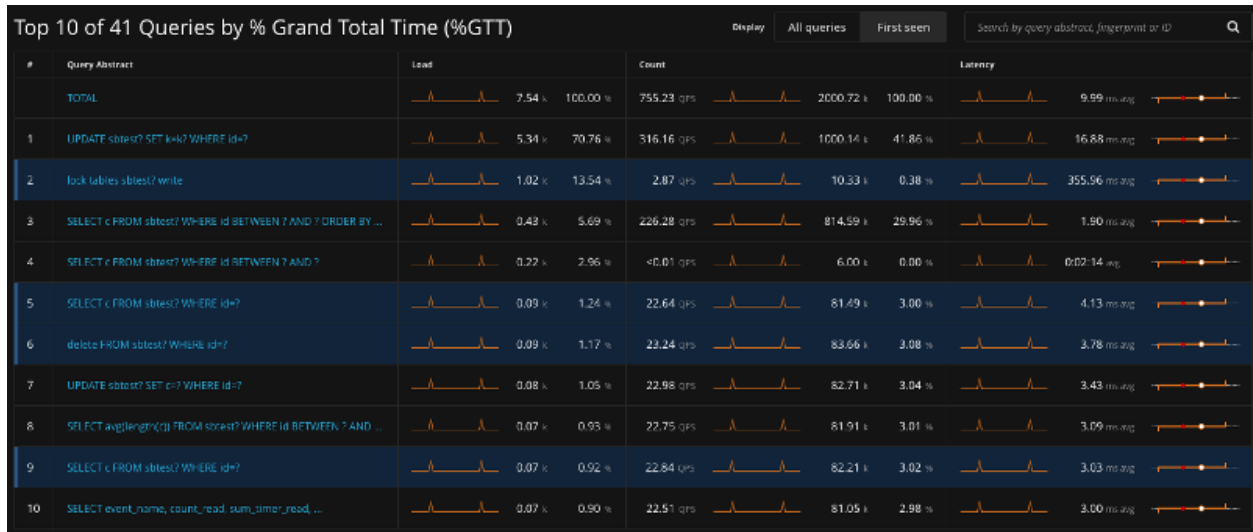


Fig. 3.4: The query summary table shows the monitored queries from the selected database.

When you apply a filter, the query summary table changes to display only the queries which match your criterion. Note that the TOTAL row which runs above the list of queries in the summary table does not change its values. These are always calculated based on all queries run within the selected time or date range.



Fig. 3.5: A list of queries

Selecting Time or Date Range

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

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

Totals of the Query Summary

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

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

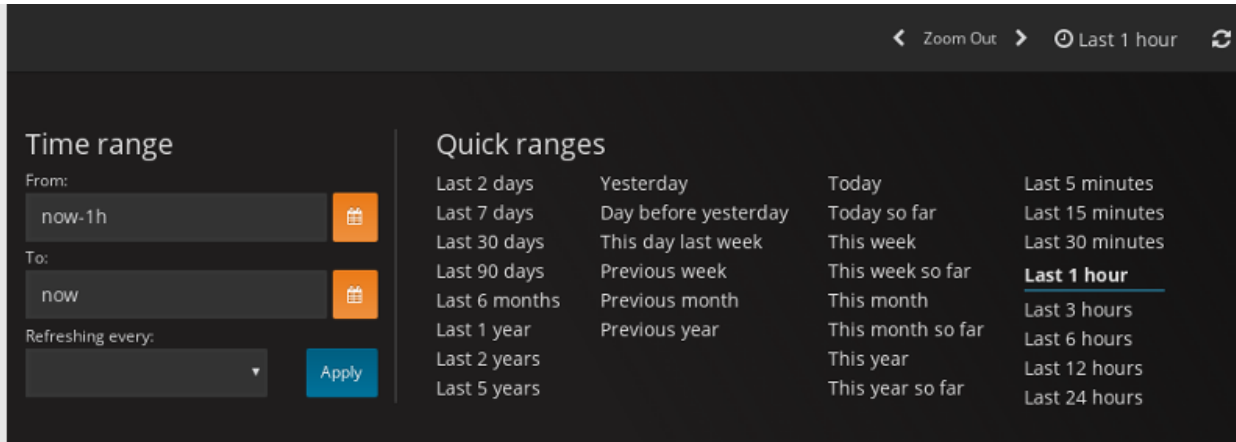


Fig. 3.6: QAN displays query metrics for the time period or date range that you specify.



Fig. 3.7: The totals appear at the top of the query summary table.

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

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

Queries in the Query Summary Table

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

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

Viewing a Specific Value of a Metric

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

Zooming into a Query

Click one of the queries to zoom it in. QAN displays detailed information about the query in the *query metrics summary table* below the *query summary table*. The detailed information includes the query type specific metrics. It also contains details about the database and tables which are used in the query.

Query Section

In addition to the metrics in the *query metrics summary table*, QAN displays more information about the query itself. The *Query* section contains the *fingerprint* and an example of the query.



Fig. 3.8: Hover the cursor to see a value at the point.

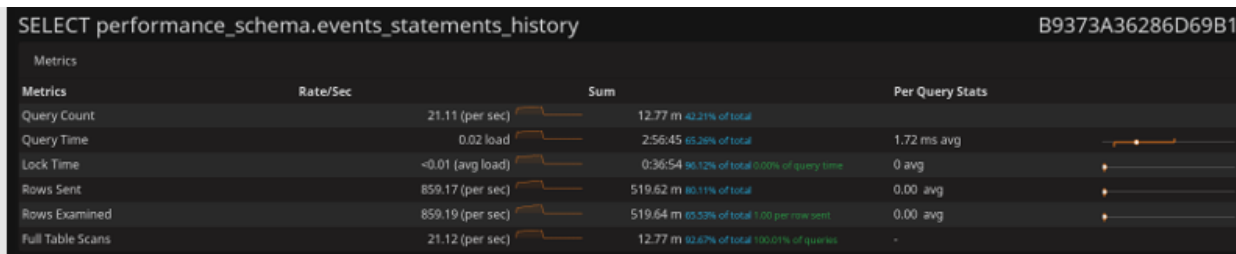


Fig. 3.9: Select a query from the query summary table to open its metrics.

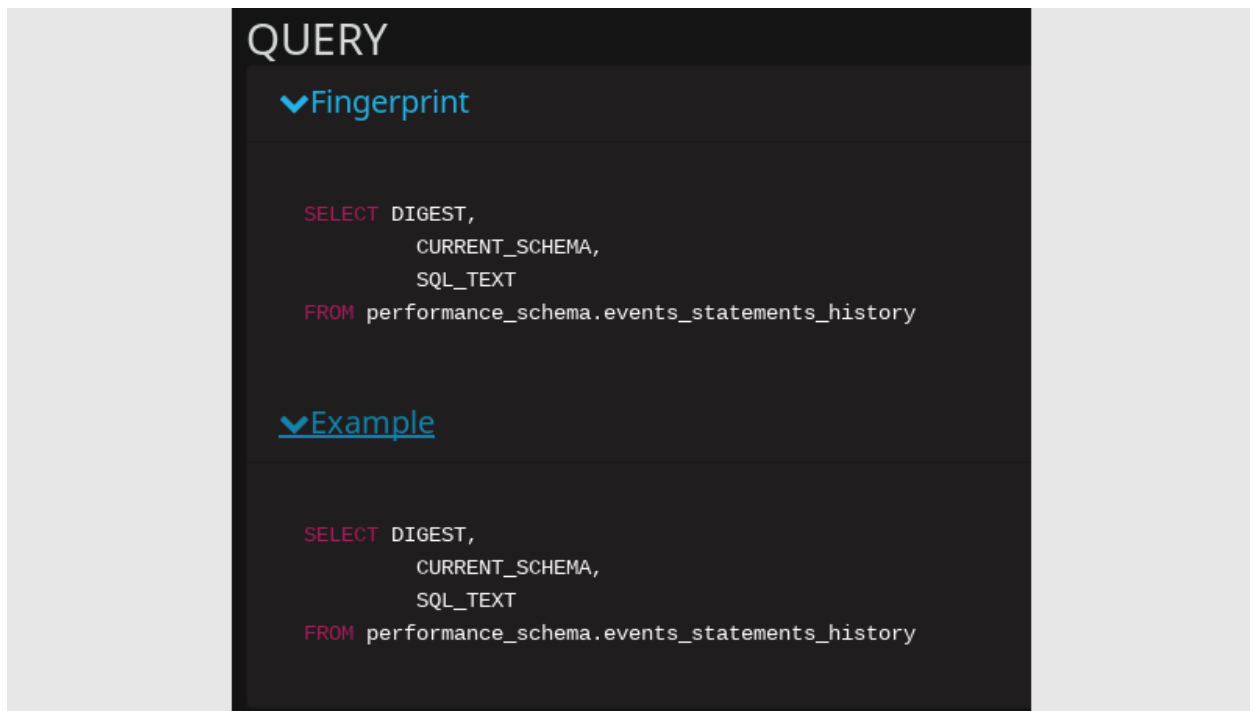


Fig. 3.10: The Query section shows the SQL statement for the selected query.

Explain Section

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

Id	SelectType	Table	Partitions	CreateTable	Type	PossibleKeys	Key	KeyLen	Ref	Rows	Extra
1	SIMPLE	events, statements, history			ALL					2560	

The output appears in three forms: classic, JSON and visual. The classic form presents the attributes of the `EXPLAIN` command as columns of a table. The JSON format presents the output of `EXPLAIN` as a JSON document. To help you better understand how the query has been optimized, the visual form shows how the query accesses the tables it includes. The output of the visual form is identical to that of `pt-visual-explain`.

Note: The *Copy to clipboard* button available in Explain, Fingerprint, Example, and Tables sections is useful to save the output and pass it to external tools, such as `pt-visual-explain`.

Note that the `EXPLAIN` command only works with the following statements:

- `SELECT`
- `DELETE`
- `INSERT`
- `REPLACE`
- `UPDATE`

If you are viewing the details of a query of another type, the *Explain* section will not contain any data.

Related information

`pt-visual-explain`: a tool to show the query plan based on the output of the `EXPLAIN` command
<https://www.percona.com/doc/percona-toolkit/LATEST/pt-visual-explain.html>

Table Info Section

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.

EXPLAIN sbtest EXPLAIN

▼ CLASSIC

Id	SelectType	Table	Partitions	CreateTable	CreateTable	PossibleKeys	Key	KeyLen	Ref	Rows	Extra
1	SIMPLE	sbtest1			const	PRIMARY	PRIMARY	4	const	1	

▼ JSON

```

- {
  "query_block": - {
    "select_id": 1,
    "cost_info": - {
      "query_cost": "19.50"
    },
    "table": - {
      "table_name": "tables",
      "access_type": "ALL",
      "key": "TABLE_SCHEMA",
      "open_full_table": true,
      "scanned_databases": "1",
      "used_columns": "[ ... ]",
      "attached_condition": "(`information_schema`.`tables`.`TABLE_SCHEMA` = 'sys')"
    }
  }
}
    
```

Expand All Copy to clipboard

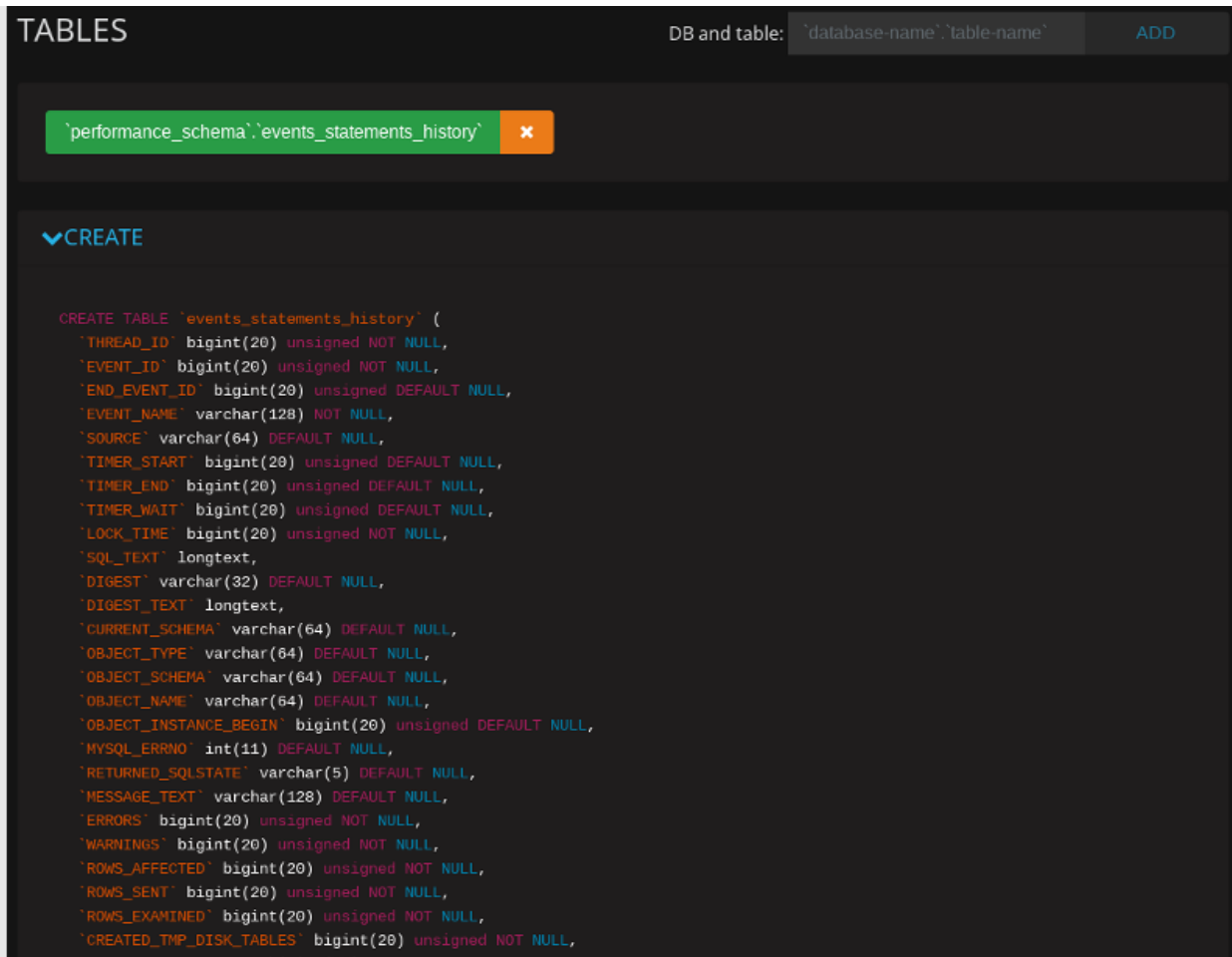
▼ VISUAL

```

Bookmark lookup
+- Table
| table      sbtest1
| possible_keys PRIMARY
+- Constant index lookup
key         sbtest1->PRIMARY
possible_keys PRIMARY
key_len     4
ref         const
rows       1
    
```

Copy to clipboard

Fig. 3.11: The three output formats of the EXPLAIN command.



Configuring QAN

All *PMM Query Analytics* settings are available from the *Query Analytics Settings* dashboard. To open this dashboard, use the PMM menu group.

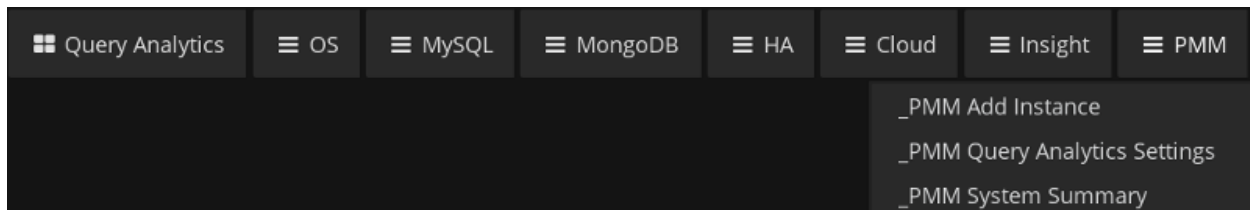


Fig. 3.12: The PMM group in the Metrics Monitor navigation menu

Settings Tab

The *Settings* tab displays the essential configuration settings of the database server selected from the *Databases* list. From this tab you can see which *DSN* is being used as well as the *database server version*.

This tab contains several settings which influence how the monitored data are collected. Note that these settings cannot be changed directly in QAN. You need to set the appropriate options by using the tools from the database server itself.

You can, however, select where the database server metrics are collected from, such as *slow log*, or *Performance Schema*. For this, change the value of the *Collect from* field accordingly.

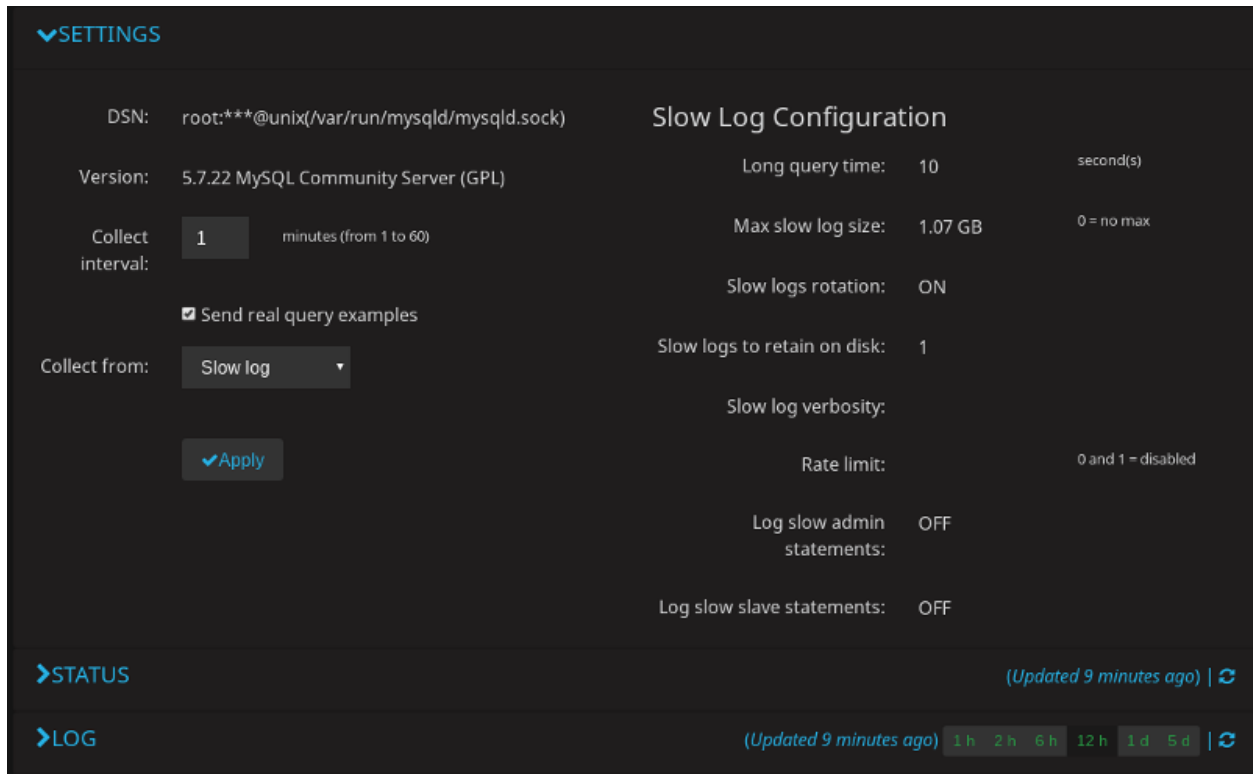


Fig. 3.13: The *Settings* tab to view the essential settings of the selected database server.

When you choose to collect MySQL data from *slow log*, a group of read only values becomes available. Note that these settings cannot be set in PMM directly. These are essential parameters of MySQL that affect the operation of *slow log*. If you need to change these settings refer to the appropriate sections of MySQL documentation.

See also:

Percona Server Documentation: [Slow Query Log Rotation and Expiration](#)

MySQL Documentation: [The Slow Query Log](#)

Status Tab

The *Status* tab contains detailed information about the current status of the monitored database server. QAN collects this information from the database server directly. For example, in case of a MySQL server, the `SHOW STATUS` command is used.

Log Tab

The *Log* tab contains the latest version of the monitored log, such as *slow log*. At the top of this tab, you may notice when exactly the snapshot was taken.

QAN for MongoDB

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

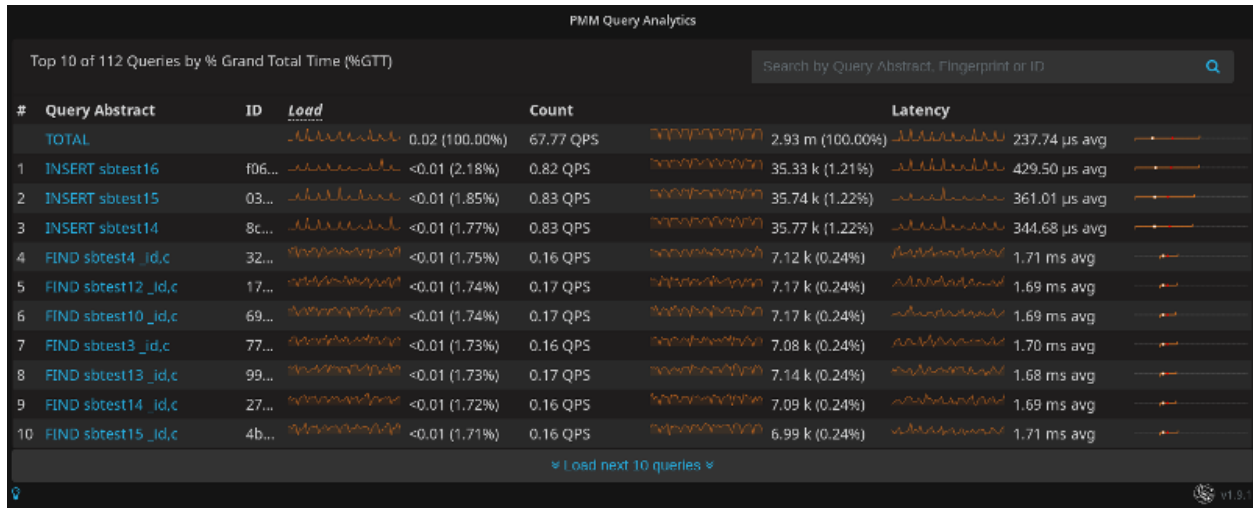


Fig. 3.14: A list of queries from a MongoDB host

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

See also:

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

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.

In this section

- [Signing in](#)
- [Opening a Dashboard](#)
- [Viewing More Information about a Graph](#)
- [Zooming in on a single metric](#)
- [Marking Important Events with Annotations](#)



Fig. 3.15: Analyze MongoDB queries using the same tools as relational database management systems.

- [Creating Snapshots of a Dashboard](#)

See also:

[How to analyze database queries over periods of time and find performance problems](#) *PMM Query Analytics*

Signing in

The credentials used to sign in to Grafana depend on the options that you specified when *starting PMM Server*:

- If you do not specify either `SERVER_USER` or `SERVER_PASSWORD`, you will log in as an anonymous user. You can change to a different existing Grafana user.
- If you specify both `--SERVER_USER` and `--SERVER_PASSWORD`, then these credentials will be used to sign in to Grafana.
- If you specify 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 specify only `--SERVER_USER`, this parameter will be ignored.

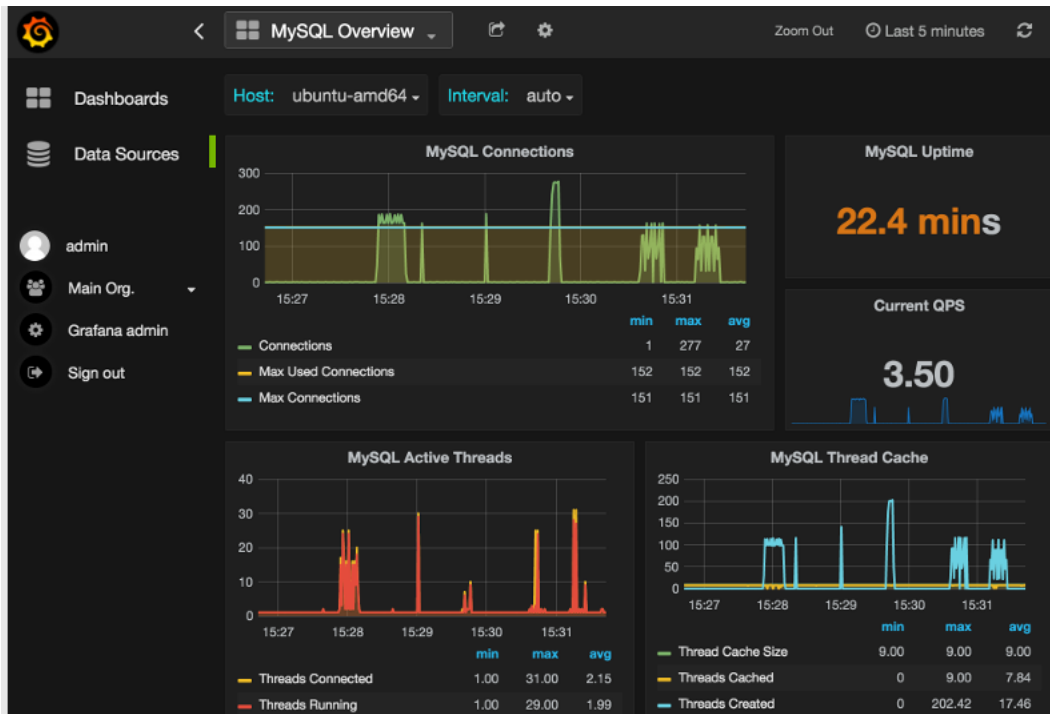
Warning: The value of the `--SERVER_USER` parameter may not contain the `#` or `:` symbols.

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:



Opening a Dashboard

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

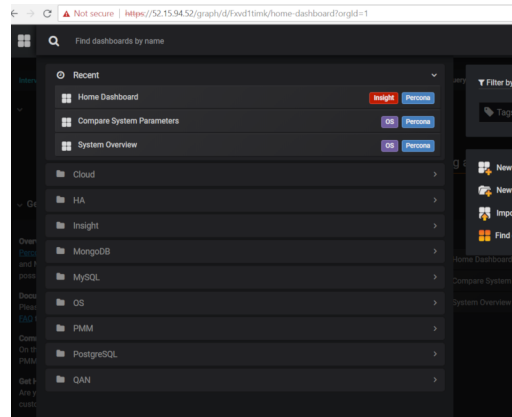


Fig. 3.16: With *Dashboard Dropdown*, search the alphabetical list for any dashboard.

You can also use a navigation menu which groups dashboards by application. Click the required group and then select

the dashboard that matches your choice.

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

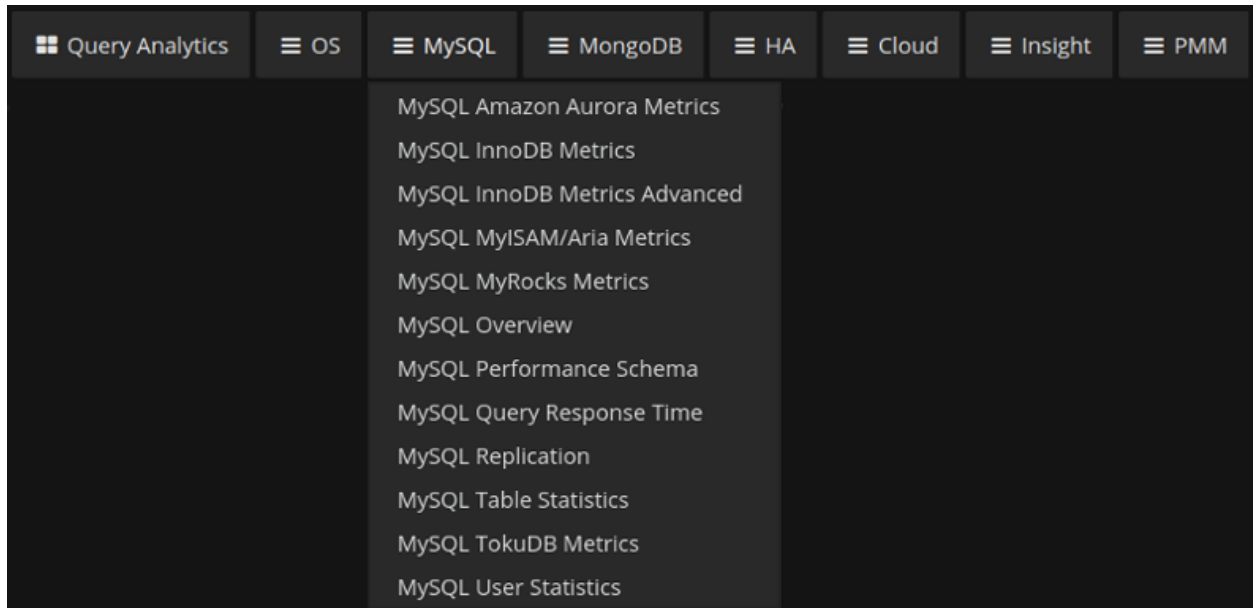


Fig. 3.17: MySQL group selected in the navigation menu

See also:

Percona support for high availability <https://www.percona.com/services/support/mysql-ha-cluster-support>

List of Metrics Monitor dashboards See section *Metrics Monitor Dashboards*

Viewing More Information about a Graph

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

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

Zooming in on a single metric

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

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

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

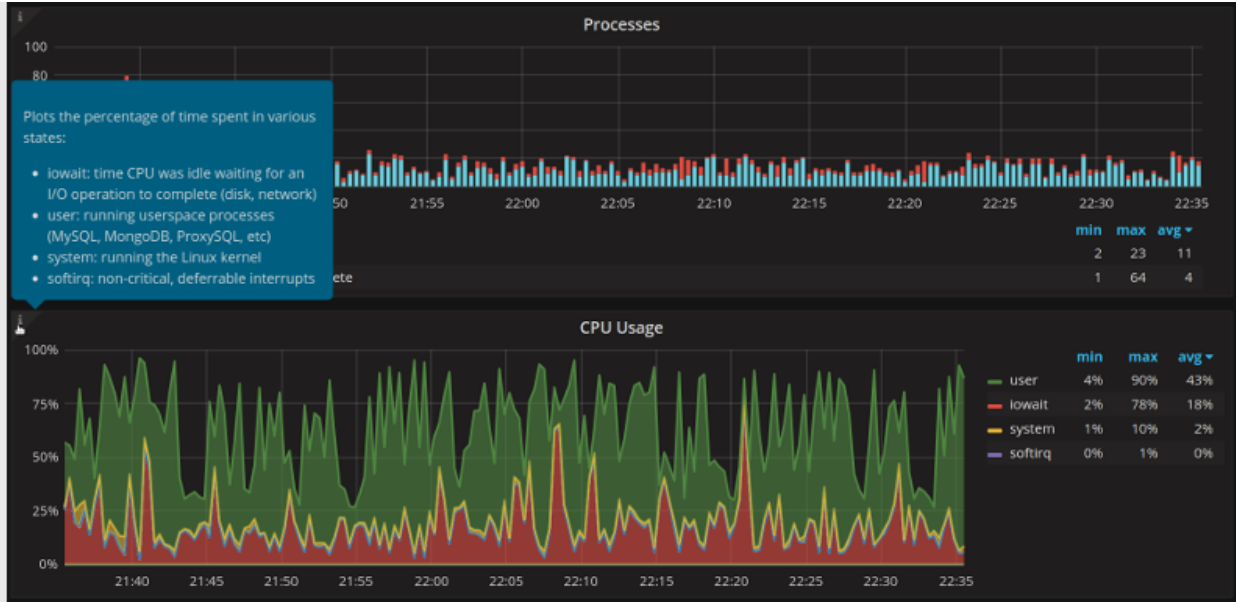
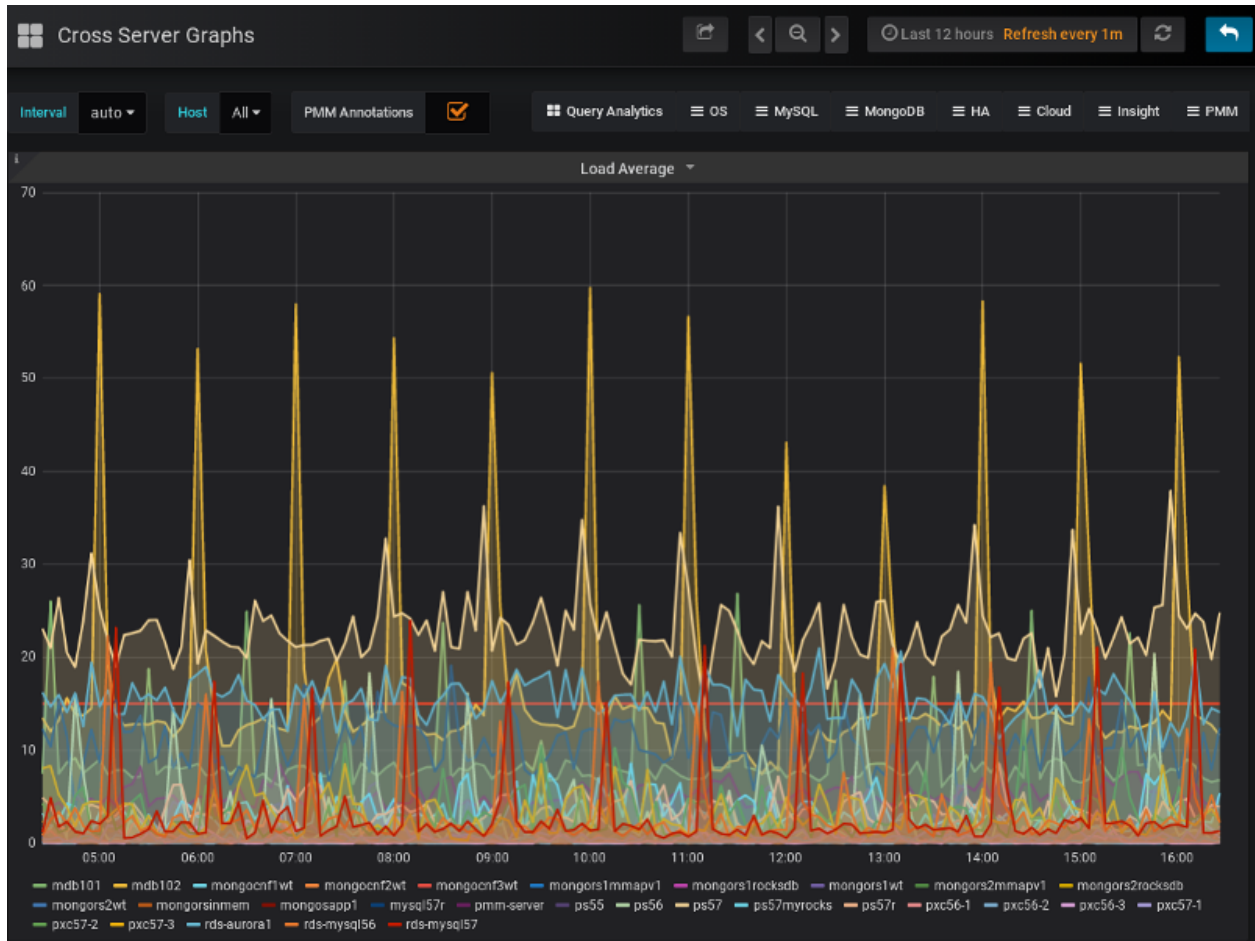


Fig. 3.18: Graph descriptions provide more information about a graph without claiming any space in the interface.



Fig. 3.19: The context menu of a metric



Note: If you are zooming in on a metric which is represented as a single number on its dashboard, you cannot change the level of detail by selecting a range on the graph itself.

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

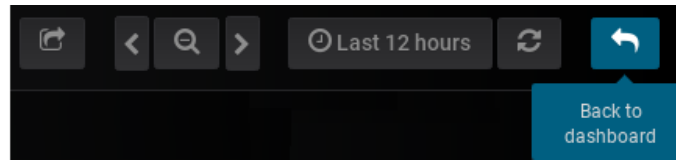


Fig. 3.20: The *Back to dashboard* button returns to the dashboard; this button appears when you are zooming in on one metric.

See also:

More information about the time range selector [Selecting time or date range](#)

Marking Important Events with Annotations

Some events in your application may impact your database. Annotations visualize these events on each dashboard of PMM Server.

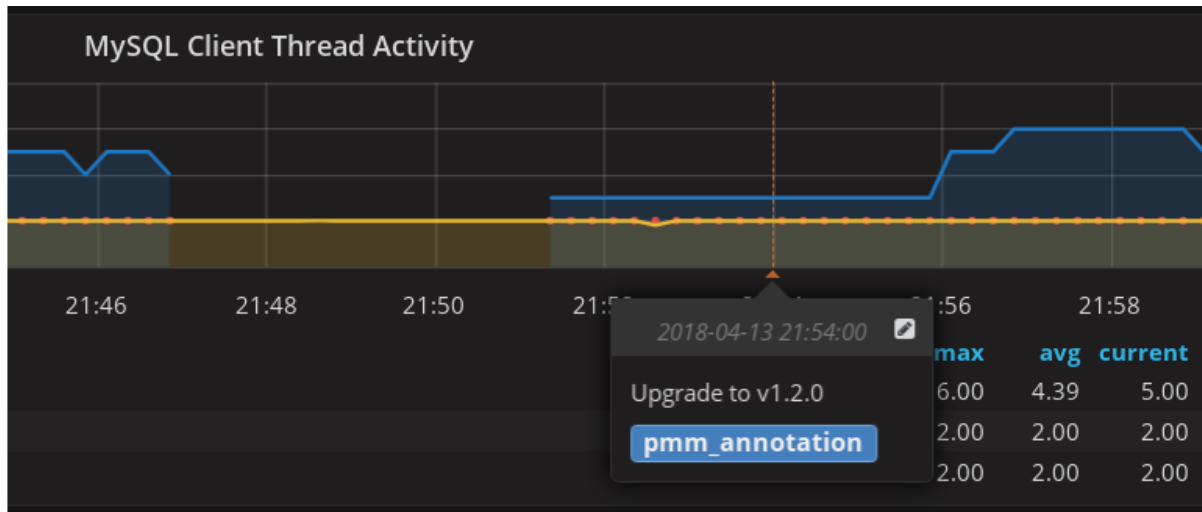


Fig. 3.21: An annotation appears as a vertical line which crosses a graph at a specific point. Its text explains which event occurred at that time.

To create a new annotation, run `pmm-admin annotate` command on PMM Client passing it text which explains what event the new annotation should represent. Use the `--tags` option to supply one or more tags separated by a comma.

You may toggle displaying annotations on metric graphs by using the *PMM Annotations* checkbox.

See also:

Adding annotations [Adding annotations](#)

Grafana Documentation:

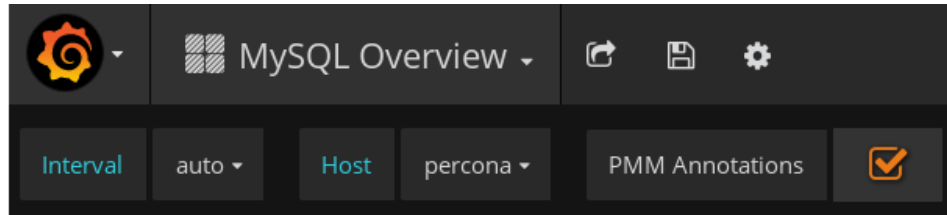


Fig. 3.22: Remove the checkmark from the *PMM Annotations* checkbox to hide annotations from all dashboards.

- [Annotations](#)
- [Using annotations in queries](#)

Creating Snapshots of a Dashboard

A snapshot is a way to securely share your dashboard with Percona. When created, we strip sensitive data like queries (metrics, template variables, and annotations) along with panel links. The shared dashboard will only be available for viewing by Percona engineers. The content on the dashboard will assist Percona engineers in troubleshooting your case.

You can safely leave the defaults set as they are, but for further information:

Snapshot name The name Percona will see when viewing your dashboard.

Expire How long before snapshot should expire, configure lower if required. Percona automatically purges shared dashboards after 90 days.

Timeout (seconds) Duration the dashboard will take to load before the snapshot is generated.

First, open the dashboard that you would like to share. Click the *Share* button at the top of the page and select the *Snapshot* command. Finally, click the *Share with Percona* button.

What to do next

After clicking *Share with Percona*, wait for the dashboard to be generated, and you will be provided a unique URL that then needs to be communicated to Percona via the ticket.

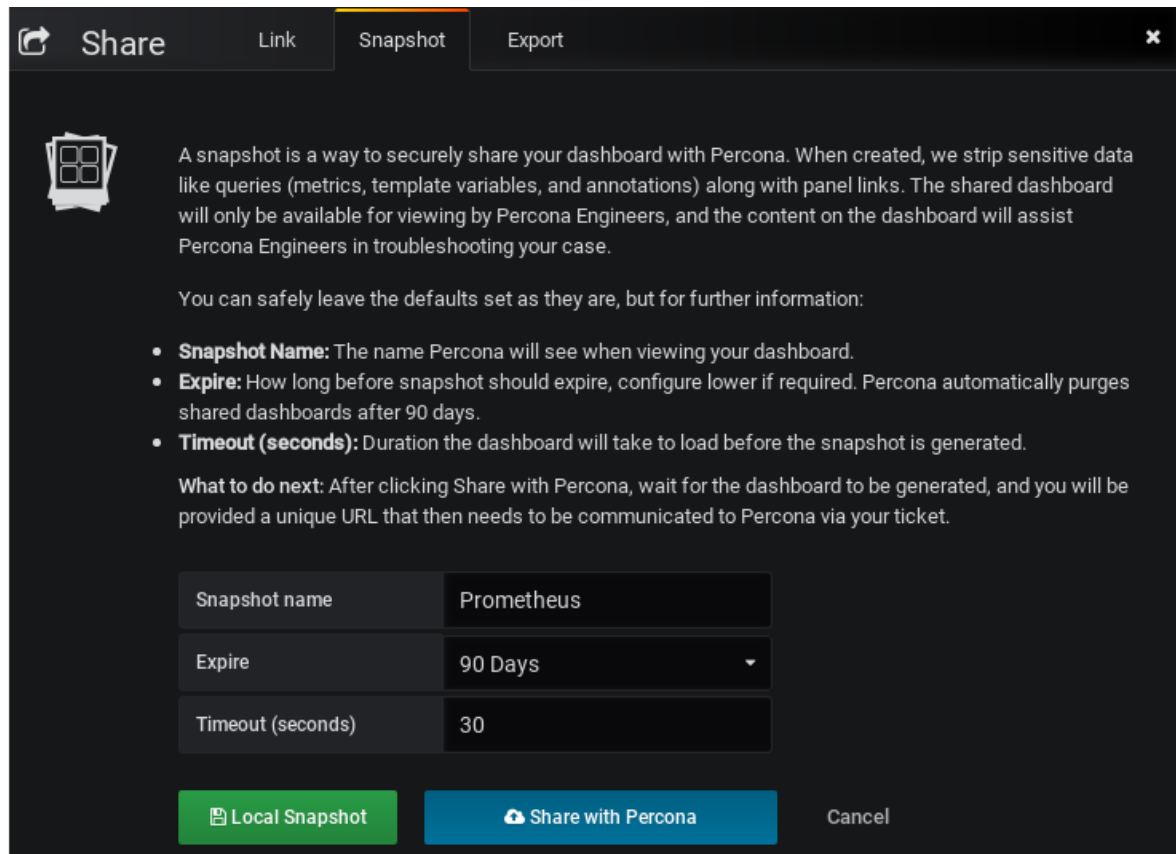


Fig. 3.23: The *Snapshot* tab in the *Share* dialog window.

Part II

Advanced

MANAGING PMM CLIENT

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

In this chapter

- Adding monitoring services
 - Adding external monitoring services
 - Passing options to the exporter
 - Passing SSL parameters to the mongodb monitoring service
 - Adding general system metrics service
 - Extending metrics with textfile collector
 - Adding MySQL query analytics service
 - Adding MySQL metrics service
 - Adding MongoDB query analytics service
 - Adding MongoDB metrics service
 - Adding ProxySQL metrics service
- Adding annotations
- Checking network connectivity
- Obtaining Diagnostics Data for Support
- Configuring PMM Client
- Getting help for any command
- Getting information about PMM Client
- Listing monitoring services
- Pinging PMM Server
- Purging metrics data
- Removing monitoring services
- Removing orphaned services
- Restarting monitoring services
- Getting passwords used by PMM Client

- Starting monitoring services
- Stopping monitoring services
- Cleaning Up Before Uninstall
- Monitoring Service Aliases

USAGE

```
pmm-admin [OPTIONS] [COMMAND]
```

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`).

To view all available commands and options, run `pmm-admin` without any commands or options:

```
$ sudo pmm-admin
```

OPTIONS

The following options can be used with any command:

- `-c, --config-file` Specify the location of PMM configuration file (default `/usr/local/percona/pmm-client/pmm.yml`).
- `-h, --help` Print help for any command and exit.
- `-v, --version` Print version of PMM Client.
- `--verbose` Print verbose output.

COMMANDS

`pmm-admin add` Add a monitoring service.

Adding annotations Add an annotation

`pmm-admin check-network` Check network connection between PMM Client and PMM Server.

`pmm-admin config` Configure how PMM Client communicates with PMM Server.

`pmm-admin help` Print help for any command and exit.

`pmm-admin info` Print information about PMM Client.

`pmm-admin list` List all monitoring services added for this PMM Client.

`pmm-admin ping` Check if PMM Server is alive.

`pmm-admin purge` Purge metrics data on PMM Server.

`pmm-admin remove, pmm-admin rm` Remove monitoring services.

`pmm-admin repair` Remove orphaned services.

`pmm-admin restart` Restart monitoring services.

`pmm-admin show-passwords` Print passwords used by PMM Client (stored in the configuration file).

`pmm-admin start` Start monitoring service.

`pmm-admin stop` Stop monitoring service.

`pmm-admin uninstall` Clean up PMM Client before uninstalling it.

Adding monitoring services

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

USAGE

```
$ pmm-admin add [OPTIONS] [SERVICE]
```

When you add a monitoring service `pmm-admin` automatically creates and sets up a service in the operating system. You can tweak the `systemd` configuration file and change its behavior.

For example, you may need to disable the HTTPS protocol for the Prometheus exporter associated with the given service. To accomplish this task, you need to remove all SSL related options.

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

1. Open the `systemd` unit file associated with the monitoring service that you need to change, such as `pmm-mysql-metrics-42002.service`.

```
$ cd /etc/systemd/system
$ cat pmm-mysql-metrics-42002.service
```

2. Remove the SSL related configuration options (key, cert) from the `systemd` unit file or `init.d` startup script. sample.systemd highlights the SSL related options in the `systemd` unit file.

The following code demonstrates how you can remove the options using the `sed` command. (If you need more information about how `sed` works, see the documentation of your system).

```
$ sed -e -i.backup 's/-web.ssl[^\ ]+[\^>]*//g' pmm-mysql-metrics-42002.service
```

3. Reload `systemd`:

```
$ systemctl daemon-reload
```

4. Restart the monitoring service by using `pmm-admin restart`:

```
$ pmm-admin restart mysql:metrics
```

OPTIONS

The following option can be used with the `pmm-admin add` command:

`--dev-enable` Enable experimental features.

`--disable-ssl` Disable (otherwise enabled) SSL for the connection between PMM Client and PMM Server. Turning off SSL encryption for the data acquired from some objects of monitoring allows to decrease the overhead for a PMM Server connected with a lot of nodes.

`--service-port`

Specify the *service port*.

You can also use *global options that apply to any other command*.

SERVICES

Specify a *monitoring service alias*, along with any relevant additional arguments.

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

- Adding external monitoring services
- Passing options to the exporter
- Passing SSL parameters to the mongodb monitoring service
- Adding general system metrics service
- Extending metrics with textfile collector
 - *Example - collecting docker container information*
- Adding MySQL query analytics service
- Adding MySQL metrics service
- Adding MongoDB query analytics service
- Adding MongoDB metrics service
- Adding ProxySQL metrics service

Adding external monitoring services

The `pmm-admin add` command is also used to add external *monitoring services*. This command adds an external monitoring service assuming that the underlying Prometheus exporter is already set up and accessible. The default scrape timeout is 10 seconds, and the interval equals to 1 minute.

To add an external monitoring service use the `external:service` monitoring service followed by the port number, name of a Prometheus job. These options are required. To specify the port number the `--service-port` option.

```
$ pmm-admin add external:service --service-port=9187 postgresql
pmm-admin 1.12.0
PMM Server      | 127.0.0.1:80
Client Name     | percona
Client Address  | 172.17.0.1
Service Manager | linux-systemd
...
Job name      Scrape interval Scrape timeout Metrics path Scheme Target
↪Labels      Health
postgresql  10s             1m           /metrics     http   172.17.0.1:9187
↪instance="percona"
```

By default, the `pmm-admin add` command automatically creates the name of the host to be displayed in the *Host* field of the *Advanced Data Exploration* dashboard where the metrics of the newly added external monitoring service will be displayed. This name matches the name of the host where `pmm-admin` is installed. You may choose another display name when adding the `external:service` monitoring service giving it explicitly after the Prometheus exporter name.

You may also use the `external:metrics` monitoring service. When using this option, you refer to the exporter by using a URL and a port number. The following example adds an external monitoring service which monitors a PostgreSQL instance at 192.168.200.1, port 9187. After the command completes, the `pmm-admin list` command shows the newly added external exporter at the bottom of the command's output:

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

```
$ pmm-admin add external:metrics postgresql 192.168.200.1:9187
```

```
PMM Server      | 192.168.100.1
Client Name     | percona
Client Address  | 192.168.200.1
Service Manager | linux-systemd
```

```
-----
```

SERVICE TYPE	NAME	LOCAL PORT	RUNNING	DATA SOURCE	OPTIONS
linux:metrics	percona	42000	YES		-

```
-----
```

Name	Scrape interval	Scrape timeout	Metrics path	Scheme	Instances
postgres	10s	1m	/metrics	http	192.168.200.1:9187

See also:

View all added monitoring services See [Listing monitoring services](#)

Use the external monitoring service to add PostgreSQL running on an Amazon RDS instance See [use-case.external-monitoring-service.postgresql.rds](#)

Passing options to the exporter

`pmm-admin add` sends all options which follow `--` (two consecutive dashes delimited by whitespace) to the Prometheus exporter that the given monitoring services uses. Each exporter has its own set of options.

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

Listing 4.1: Passing `--collect.perf_schema.eventsstatements` to the `mysql:metrics` monitoring service

```
$ pmm-admin add mysql:metrics -- --collect.perf_schema.eventsstatements
```

Listing 4.2: Passing `--collect.perf_schema.eventswaits=false` to the `mysql:metrics` monitoring service

```
$ pmm-admin add mysql:metrics -- --collect.perf_schema.eventswaits=false
```

The section `pmm.list.exporter` contains all option grouped by exporters.

Passing SSL parameters to the mongodb monitoring service

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

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

Listing 4.3: *Passing an SSL/TLS parameter to `mongod` to enable a TLS connection.*

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

Table 4.1: Supported SSL/TLS Parameters

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

Note: PMM does not support passing SSL/TLS related parameters to `mongodb:queries`.

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

Adding general system metrics service

Use the `linux:metrics` alias to enable general system metrics monitoring.

USAGE

```
$ pmm-admin add linux:metrics [NAME] [OPTIONS]
```

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 explicitly as an argument.

OPTIONS

The following option can be used with the `linux:metrics` alias:

--force Force to add another general system metrics service with a different name for testing purposes.

You can also use *global options that apply to any other command*, as well as *options that apply to adding services in general*.

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

See also:

Default ports *Ports in Terminology Reference*

Extending metrics with textfile collector

New in version 1.16.0.

While PMM provides an excellent solution for system monitoring, sometimes you may have the need for a metric that's not present in the list of `node_exporter` metrics out of the box. There is a simple method to extend the list of available metrics without modifying the `node_exporter` code. It is based on the textfile collector.

Starting from version 1.16.0, this collector is enabled for the `linux:metrics` in PMM Client by default.

The default directory for reading text files with the metrics is `/usr/local/percona/pmm-client/textfile-collector`, and the exporter reads files from it with the `.prom` extension. By default it contains an example file `example.prom` which has commented contents and can be used as a template.

You are responsible for running a cronjob or other regular process to generate the metric series data and write it to this directory.

Example - collecting docker container information

This example will show you how to collect the number of running and stopped docker containers on a host. It uses a crontab task, set with the following lines in the cron configuration file (e.g. in `/etc/crontab`):

```
*/* * * * *      root    echo -n "" > /tmp/docker_all.prom; /usr/bin/docker ps -a | sed
↳ -n '1!p'| /usr/bin/wc -l | sed -ne 's/^/node_docker_containers_total /p' >> /usr/
↳ local/percona/pmm-client/docker_all.prom;
*/1 * * * *      root    echo -n "" > /tmp/docker_running.prom; /usr/bin/docker ps |
↳ sed -n '1!p'| /usr/bin/wc -l | sed -ne 's/^/node_docker_containers_running_total /p
↳ ' >> /usr/local/percona/pmm-client/docker_running.prom;
```

The result of the commands is placed into the `docker_all.prom` and `docker_running.prom` files and read by exporter.

The first command executed by cron is rather simple: the destination text file is cleared by executing `echo -n ""`, then a list of running and closed containers is generated with `docker ps -a`, and finally `sed` and `wc` tools are used to count the number of containers in this list and to form the output file which looks like follows:

```
node_docker_containers_total 2
```

The second command is similar, but it counts only running containers.

Adding MySQL query analytics service

Use the `mysql:queries` alias to enable MySQL query analytics.

USAGE

```
pmm-admin add mysql:queries [NAME] [OPTIONS]
```

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

The `pmm-admin add` command is able to detect the local PMM Client name, but you can also specify it explicitly as an argument.

Important: If you connect MySQL Server version 8.0, make sure it is started with the `default_authentication_plugin` set to the value `mysql_native_password`.

You may alter your PMM user and pass the authentication plugin as a parameter:

```
mysql> ALTER USER pmm@'localhost' IDENTIFIED WITH mysql_native_password BY '$eCR8Tp@s
↳ $w*RD';
```

See also:

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

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

OPTIONS

The following options can be used with the `mysql:queries` alias:

--create-user Create a dedicated MySQL user for PMM Client (named `pmm`).

--create-user-maxconn Specify maximum connections for the dedicated MySQL user (default is 10).

--create-user-password Specify password for the dedicated MySQL user.

--defaults-file Specify path to `my.cnf`.

--disable-queryexamples Disable collection of query examples.

--slow-log-rotation

Do not manage *slow log* files by using PMM. Set this option to *false* if you intend to manage *slow log* files by using a third party tool. The default value is *true*

See also:

Example of disabling the slow log rotation feature and using a third party tool [use-case.slow-log-rotation](#)

Related Information

Percona Database Performance Blog: Rotating MySQL Slow Logs Safely <https://www.percona.com/blog/2013/04/18/rotating-mysql-slow-logs-safely/>

Percona Database Performance Blog: Log Rotate and the (Deleted) MySQL Log File Mystery <https://www.percona.com/blog/2014/11/12/log-rotate-and-the-deleted-mysql-log-file-mystery/>

--force Force to create or update the dedicated MySQL user.

--host Specify the MySQL host name.

--password Specify the password for MySQL user with admin privileges.

--port Specify the MySQL instance port.

--query-source Specify the source of data:

- `auto`: Select automatically (default).
- `slowlog`: Use the slow query log.
- `perfschema`: Use Performance Schema.

--retain-slow-logs Specify the maximum number of files of the *slow log* to keep automatically. The default value is 1 file.

--socket Specify the MySQL instance socket file.

--user Specify the name of MySQL user with admin privileges.

You can also use *global options that apply to any other command*, as well as *options that apply to adding services in general*.

See also:

Default ports *Ports in Terminology Reference*

DETAILED DESCRIPTION

When adding the MySQL query analytics service, 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, if it cannot be detected automatically.

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

See also:

More information about MySQL users with PMM *Creating a MySQL User Account to Be Used with PMM*

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:

```
$ 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 *Configuring Performance Schema*.

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

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

See also:

How to set up MySQL for monitoring? *Configuring MySQL for Best Results*

Adding MySQL metrics service

Use the `mysql:metrics` alias to enable MySQL metrics monitoring.

USAGE

```
$ pmm-admin add mysql:metrics [NAME] [OPTIONS]
```

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

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

OPTIONS

The following options can be used with the `mysql:metrics` alias:

- create-user** Create a dedicated MySQL user for PMM Client (named `pmm`).
- create-user-maxconn** Specify maximum connections for the dedicated MySQL user (default is 10).
- create-user-password** Specify password for the dedicated MySQL user.
- defaults-file** Specify the path to `my.cnf`.
- disable-binlogstats** Disable collection of binary log statistics.
- disable-processlist** Disable collection of process state metrics.
- disable-tablestats** Disable collection of table statistics.
- disable-tablestats-limit** Specify the maximum number of tables for which collection of table statistics is enabled (by default, the limit is 1 000 tables).
- disable-userstats** Disable collection of user statistics.
- force** Force to create or update the dedicated MySQL user.
- host** Specify the MySQL host name.
- password** Specify the password for MySQL user with admin privileges.
- port** Specify the MySQL instance port.
- socket** Specify the MySQL instance socket file.
- user** Specify the name of MySQL user with admin privileges.

You can also use *global options that apply to any other command*, as well as *options that apply to adding services in general*.

See also:

Default ports *Ports in Terminology Reference*

More information about PMM Query Analytics *PMM Query Analytics*

DETAILED DESCRIPTION

When adding the MySQL metrics monitoring service, the `pmm-admin` tool attempts to automatically detect the local MySQL instance and MySQL superuser credentials. You can use options to provide this information, if it cannot be detected automatically.

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:

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

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

See also:

How to set up MySQL for monitoring? [Configuring MySQL for Best Results](#)

Adding MongoDB query analytics service

Use the `mongodb:queries` alias to enable MongoDB query analytics.

USAGE

```
pmm-admin add mongodb:queries [NAME] [OPTIONS]
```

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

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

OPTIONS

The following options can be used with the `mongodb:queries` alias:

--uri Specify the MongoDB instance URI with the following format:

```
[mongodb://][user:pass@]host[:port][/database][?options]
```

By default, it is `localhost:27017`.

Important: In cases when the password contains special symbols like the *at* (`@`) symbol, the host might not be detected correctly. Make sure that you insert the password with special characters replaced with their escape sequences. The simplest way is to use the `encodeURIComponent` JavaScript function.

For this, open the web console of your browser (usually found under *Development tools*) and evaluate the following expression, passing the password that you intend to use:

```
> encodeURIComponent('$ecRet_pas$w@rd')
"%24ecRet_pas%24w%40rd"
```

Related Information

MDN Web Docs: encodeURIComponent https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/encodeURIComponent

You can also use *global options that apply to any other command*, as well as *options that apply to adding services in general*.

Note: PMM does not support passing SSL/TLS related parameters to `mongodb:queries`.

For more information, run `pmm-admin add mongodb:queries --help`.

See also:

Default ports *Ports in Terminology Reference*

Adding MongoDB metrics service

Use the `mongodb:metrics` alias to enable MongoDB metrics monitoring.

USAGE

```
$ pmm-admin add mongodb:metrics [NAME] [OPTIONS]
```

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 explicitly as an argument.

OPTIONS

The following options can be used with the `mongodb:metrics` alias:

--cluster Specify the MongoDB cluster name.

--uri Specify the MongoDB instance URI with the following format:

```
[mongodb://][user:pass@]host[:port][/database][?options]
```

By default, it is `localhost:27017`.

You can also use *global options that apply to any other command*, as well as *options that apply to adding services in general*.

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

Monitoring a cluster

When using PMM to monitor a cluster, you should enable monitoring for each instance by using the **pmm-admin add** command. This includes each member of replica sets in shards, mongos, and all configuration servers. Make sure that for each instance you supply the cluster name via the `--cluster` option and provide its URI via the `--uri` option.

Run this command as root or by using the **sudo** command. This examples uses `127.0.0.1` as a URL.

```
$ pmm-admin add mongodb:metrics \
--uri mongodb://127.0.0.1:<port>/admin <instance name> \
--cluster <cluster name>
```

See also:

Default ports *Ports in Terminology Reference*

Essential MongoDB configuration *Configuring MongoDB for Monitoring in PMM Query Analytics*

Adding ProxySQL metrics service

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

USAGE

```
$ pmm-admin add proxysql:metrics [NAME] [OPTIONS]
```

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 explicitly as an argument.

OPTIONS

The following option can be used with the `proxysql:metrics` alias:

--dsn Specify the ProxySQL connection DSN. By default, it is `stats:stats@tcp(localhost:6032)/`.

You can also use *global options that apply to any other command*, as well as *options that apply to adding services in general*.

For more information, run **pmm-admin add proxysql:metrics --help**.

See also:

Default ports *Ports in Terminology Reference*

Adding annotations

Use the **pmm-admin annotate** command to set notifications about important application events and display them on all dashboards. By using annotations, you can conveniently analyze the impact of application events on your database.

USAGE

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

```
$ pmm-admin annotate "Upgrade to v1.2" --tags "UX Improvement,v1.2"
```

OPTIONS

The **pmm-admin annotate** supports the following options:

`--tags`

Specify one or more tags applicable to the annotation that you are creating. Enclose your tags in quotes and separate individual tags by a comma, such as “tag 1,tag 2”.

You can also use *global options that apply to any other command*.

Checking network connectivity

Use the **pmm-admin check-network** command to run tests that verify connectivity between PMM Client and PMM Server.

USAGE

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

```
pmm-admin check-network [OPTIONS]
```

OPTIONS

The **pmm-admin check-network** command does not have its own options, but you can use *global options that apply to any other command*

DETAILED DESCRIPTION

Connection tests are performed both ways, with results separated accordingly:

- Client --> Server

Pings Consul API, *PMM 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.

- Client <-- Server

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 firewall settings on the PMM Client host allow incoming connections for corresponding ports.

OUTPUT EXAMPLE

```

$ pmm-admin check-network
PMM Network Status

Server Address | 192.168.100.1
Client Address | 192.168.200.1

* System Time
NTP Server (0.pool.ntp.org)      | 2017-05-03 12:05:38 -0400 EDT
PMM Server                       | 2017-05-03 16:05:38 +0000 GMT
PMM Client                       | 2017-05-03 12:05:38 -0400 EDT
PMM Server Time Drift            | OK
PMM Client Time Drift           | OK
PMM Client to PMM Server Time Drift | OK

* Connection: Client --> Server
-----
SERVER SERVICE      STATUS
-----
Consul API          OK
Prometheus API      OK
Query Analytics API OK

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

* Connection: Client <-- Server
-----
SERVICE TYPE      NAME          REMOTE ENDPOINT      STATUS  HTTPS/TLS  PASSWORD
-----
linux:metrics      mongo-main    192.168.200.1:42000  OK      YES        -
mongodb:metrics    mongo-main    192.168.200.1:42003  PROBLEM YES        -

```

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

Obtaining Diagnostics Data for Support

PMM Client is able to generate a set of files for enhanced diagnostics, which is designed to be shared with Percona Support to solve an issue faster. This feature fetches logs, network, and the Percona Toolkit output. To perform data collection by PMM Client, execute:

```
pmm-admin summary
```

The output will be a tarball you can examine and/or attach to your Support ticket in the Percona's [issue tracking system](#). The single file will look like this:

```
summary__2018_10_10_16_20_00.tar.gz
```

Configuring PMM Client

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

USAGE

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

```
pmm-admin config [OPTIONS]
```

OPTIONS

The following options can be used with the **pmm-admin config** command:

- bind-address** Specify the bind address, which is also the local (private) address mapped from client address via NAT or port forwarding. By default, it is set to the client address.
- client-address** Specify the client address, which is also the remote (public) address for this system. By default, it is automatically detected via request to server.
- client-name** Specify the client name. By default, it is set to the host name.
- force** Force to set the client name on initial setup after uninstall with unreachable server.
- server** Specify the address of the PMM Server host. If necessary, you can also specify the port after colon, for example:

```
pmm-admin config --server 192.168.100.6:8080
```

By default, port 80 is used with SSL disabled, and port 443 when SSL is enabled.

- server-insecure-ssl** Enable insecure SSL (self-signed certificate).
- SERVER_PASSWORD** Specify the HTTP password configured on PMM Server.
- server-ssl** Enable SSL encryption for connection to PMM Server.
- SERVER_USER** Specify the HTTP user configured on PMM Server (default is `pmm`).

You can also use *global options that apply to any other command*.

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

Getting help for any command

Use the **pmm-admin help** command to print help for any command.

USAGE

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

```
$ pmm-admin help [COMMAND]
```

This will print help information and exit. The actual command is not run and options are ignored.

Note: You can also use the global `-h` or `--help` option after any command to get the same help information.

COMMANDS

You can print help information for any *command* or *service alias*.

Getting information about PMM Client

Use the `pmm-admin info` command to print basic information about PMM Client.

USAGE

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

```
pmm-admin info [OPTIONS]
```

OPTIONS

The `pmm-admin info` command does not have its own options, but you can use *global options that apply to any other command*

OUTPUT

The output provides the following information:

- 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
- Go version and runtime information

For example:

```
$ pmm-admin info
PMM Server      | 192.168.100.1
Client Name     | ubuntu-amd64
Client Address  | 192.168.200.1
Service manager | linux-systemd

Go Version      | 1.8
Runtime Info    | linux/amd64
```

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

Listing monitoring services

Use the `pmm-admin list` command to list all enabled services with details.

USAGE

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

```
pmm-admin list [OPTIONS]
```

OPTIONS

The **pmm-admin list** command supports *global options that apply to any other command* and also provides a machine friendly JSON output.

--json list the enabled services as a JSON document. The information provided in the standard tabular form is captured as keys and values. The general information about the computer where PMM Client is installed is given as top level elements:

- Version
- ServerAddress
- ServerSecurity
- ClientName
- ClientAddress
- ClientBindAddress
- Platform

Note that you can quickly determine if there are any errors by inspecting the `Err` top level element in the JSON output. Similarly, the `ExternalErr` element reports errors in external services.

The `Services` top level element contains a list of documents which represent enabled monitoring services. Each attribute in a document maps to the column in the tabular output.

The `ExternalServices` element contains a list of documents which represent enabled external monitoring services. Each attribute in a document maps to the column in the tabular output.

OUTPUT

The output provides the following information:

- 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:

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

```
$ pmm-admin list
...
PMM Server      | 192.168.100.1
Client Name     | ubuntu-amd64
Client Address  | 192.168.200.1
Service manager | linux-systemd
```

SERVICE TYPE	NAME	LOCAL PORT	RUNNING	DATA SOURCE	OPTIONS
linux:metrics	mongo-main	42000	YES	-	
mongodb:metrics	mongo-main	42003	YES	localhost:27017	

Pinging PMM Server

Use the `pmm-admin ping` command to verify connectivity with PMM Server.

USAGE

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

```
pmm-admin ping [OPTIONS]
```

If the ping is successful, it returns OK.

```
$ 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
```

OPTIONS

The `pmm-admin ping` command does not have its own options, but you can use *global options that apply to any other command*.

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

Purging metrics data

Use the `pmm-admin purge` command to purge metrics data associated with a service on PMM Server. This is usually required after you *remove a service* and do not want its metrics data to show up on graphs.

USAGE

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

```
pmm-admin purge [SERVICE [NAME]] [OPTIONS]
```

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

SERVICES

Specify a *monitoring service alias*. To see which services are enabled, run `pmm-admin list`.

OPTIONS

The `pmm-admin purge` command does not have its own options, but you can use *global options that apply to any other command*

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

Removing monitoring services

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

USAGE

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

```
pmm-admin rm [OPTIONS] [SERVICE]
```

When you remove a service, collected data remains in Metrics Monitor on PMM Server. To remove the collected data, use the `pmm-admin purge` command.

OPTIONS

The following option can be used with the `pmm-admin rm` command:

`--all` Remove all monitoring services.

You can also use *global options that apply to any other command*.

SERVICES

Specify a *monitoring service alias*. To see which services are enabled, run `pmm-admin list`.

EXAMPLES

- To remove all services enabled for this PMM Client:

```
$ pmm-admin rm --all
```

- To remove all services related to MySQL:

```
$ pmm-admin rm mysql
```

- To remove only `mongodb:metrics` service:

```
$ pmm-admin rm mongodb:metrics
```

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

Removing orphaned services

Use the `pmm-admin repair` command to remove information about orphaned services from PMM Server. This can happen if you removed services locally while PMM Server was not available (disconnected or shut down), for example, using the `pmm-admin uninstall` command.

USAGE

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

```
$ pmm-admin repair [OPTIONS]
```

OPTIONS

The `pmm-admin repair` command does not have its own options, but you can use *global options that apply to any other command*.

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

Restarting monitoring services

Use the `pmm-admin restart` command to restart services managed by this PMM Client. This is the same as running `pmm-admin stop` and `pmm-admin start`.

USAGE

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

```
pmm-admin restart [SERVICE [NAME]] [OPTIONS]
```

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

OPTIONS

The following option can be used with the `pmm-admin restart` command:

`--all` Restart all monitoring services.

You can also use *global options that apply to any other command*.

SERVICES

Specify a *monitoring service alias* that you want to restart. To see which services are available, run `pmm-admin list`.

EXAMPLES

- To restart all available services for this PMM Client:

```
# pmm-admin restart --all
```

- To restart all services related to MySQL:

```
$ pmm-admin restart mysql
```

- To restart only the `mongodb:metrics` service:

```
$ pmm-admin restart mongodb:metrics
```

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

Getting passwords used by PMM Client

Use the `pmm-admin show-passwords` command to print credentials stored in the configuration file (by default: `/usr/local/percona/pmm-client/pmm.yml`).

USAGE

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

```
pmm-admin show-passwords [OPTIONS]
```

OPTIONS

The `pmm-admin show-passwords` command does not have its own options, but you can use *global options that apply to any other command*

OUTPUT

This command prints HTTP authentication credentials and the password for the `pmm` user that is created on the MySQL instance if you specify the `--create-user` option when *adding a service*.

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

```
$ pmm-admin show-passwords
HTTP basic authentication
User      | aname
Password | secr3tPASS
```

```
MySQL new user creation
Password | g,3i-QR50tQJi9M1y19-
```

For more information, run `pmm-admin show-passwords --help`.

Starting monitoring services

Use the `pmm-admin start` command to start services managed by this PMM Client.

USAGE

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

```
pmm-admin start [SERVICE [NAME]] [OPTIONS]
```

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

OPTIONS

The following option can be used with the `pmm-admin start` command:

`--all` Start all monitoring services.

You can also use *global options that apply to any other command*.

SERVICES

Specify a *monitoring service alias* that you want to start. To see which services are available, run `pmm-admin list`.

EXAMPLES

- To start all available services for this PMM Client:

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

- To start all services related to MySQL:

```
$ pmm-admin start mysql
```

- To start only the `mongodb:metrics` service:

```
$ pmm-admin start mongodb:metrics
```

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

Stopping monitoring services

Use the `pmm-admin stop` command to stop services managed by this PMM Client.

USAGE

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

```
pmm-admin stop [SERVICE [NAME]] [OPTIONS]
```

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

OPTIONS

The following option can be used with the `pmm-admin stop` command:

`--all` Stop all monitoring services.

You can also use *global options that apply to any other command*.

SERVICES

Specify a *monitoring service alias* that you want to stop. To see which services are available, run `pmm-admin list`.

EXAMPLES

- To stop all available services for this PMM Client:

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

- To stop all services related to MySQL:

```
$ pmm-admin stop mysql
```

- To stop only the `mongodb:metrics` service:

```
$ pmm-admin stop mongodb:metrics
```

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

Cleaning Up Before Uninstall

Use the `pmm-admin uninstall` command to remove all services even if PMM Server is not available. To uninstall PMM correctly, you first need to remove all services, then uninstall PMM Client, and then stop and remove PMM Server. However, if PMM Server is not available (disconnected or shut down), `pmm-admin rm` will not work. In this case, you can use `pmm-admin uninstall` to force the removal of monitoring services enabled for PMM Client.

Note: Information about services will remain in PMM Server, and it will not let you add those services again. To remove information about orphaned services from PMM Server, once it is back up and available to PMM Client, use the `pmm-admin repair` command.

USAGE

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

```
pmm-admin uninstall [OPTIONS]
```

OPTIONS

The `pmm-admin uninstall` command does not have its own options, but you can use *global options that apply to any other command*.

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

Monitoring Service Aliases

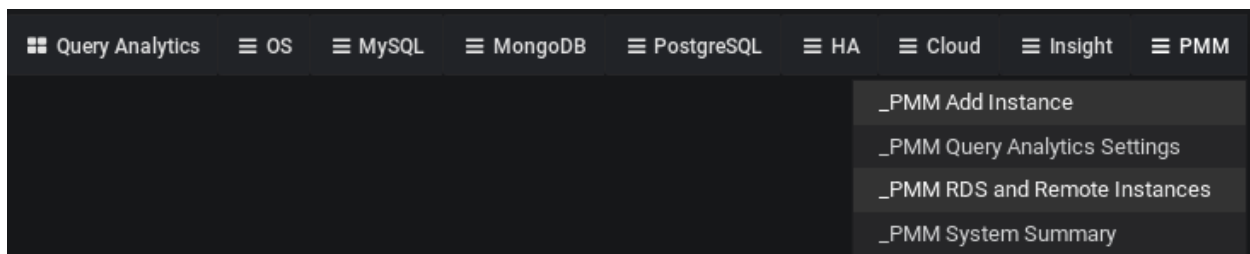
The following aliases are used to designate PMM services that you want to *add*, *remove*, *restart*, *start*, or *stop*:

Alias	Services
<code>linux:metrics</code>	General system metrics monitoring service
<code>mysql:metrics</code>	MySQL metrics monitoring service
<code>mysql:queries</code>	MySQL query analytics service
<code>mongodb:metrics</code>	MongoDB metrics monitoring service
<code>mongodb:queries</code>	MongoDB query analytics service
<code>proxysql:metrics</code>	ProxySQL metrics monitoring service
<code>mysql</code>	Complete MySQL instance monitoring: <ul style="list-style-type: none"> • <code>linux:metrics</code> • <code>mysql:metrics</code> • <code>mysql:queries</code>
<code>mongodb</code>	Complete MongoDB instance monitoring: <ul style="list-style-type: none"> • <code>linux:metrics</code> • <code>mongodb:metrics</code> • <code>mongodb:queries</code>

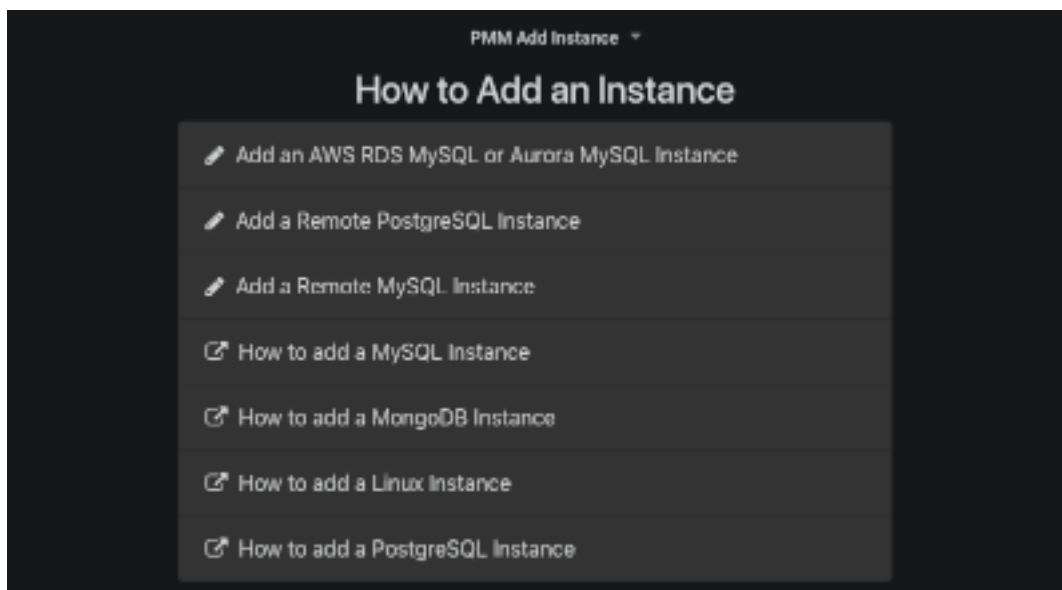
ADDING A MYSQL OR POSTGRESQL REMOTE DB INSTANCE TO PMM

There is a quick method for users to add DBaaS instances to PMM without having to hook into the Cloud Provider's API, and with no need to have PMM Client installed or any exporters running on the monitored node. The drawback of this approach is that you will not have visibility of host-level metrics (CPU, memory, and disk activity will not be captured nor displayed in PMM).

Both methods can be accessed in the Metrics Monitor navigation menu by selecting the *PMM Add Instance* item in a *PMM Dropdown* group:

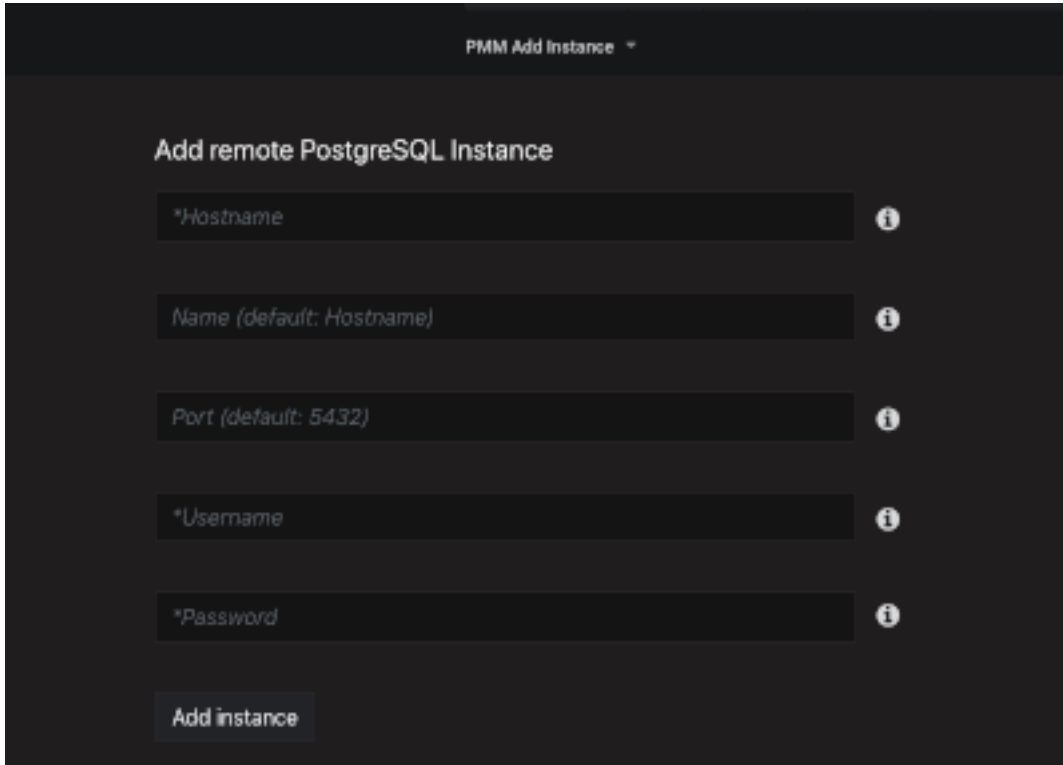


Two database servers are currently supported by this method: PostgreSQL and MySQL.



Adding a Remote PostgreSQL Instance

To add a remote PostgreSQL DB instance, you will need to fill in three fields: Hostname, Username, and Password, and optionally override the default Port and Name fields:



The screenshot shows a dark-themed web interface titled "PMM Add Instance". Below the title is the heading "Add remote PostgreSQL Instance". There are five input fields, each with an information icon (i) to its right: "*Hostname", "Name (default: Hostname)", "Port (default: 5432)", "*Username", and "*Password". At the bottom of the form is a button labeled "Add instance".

Adding a Remote MySQL Instance

To add a remote MySQL DB instance, you will need to fill in three fields: Hostname, Username, and Password, and optionally override the default Port and Name fields:

Viewing Remote MySQL and PostgreSQL Instances

Amazon RDS and remote instances can be seen in the RDS and Remote Instances list, which can be accessed in the Metrics Monitor navigation menu by selecting the *PMM RDS and Remote Instances* item from the *PMM Dropdown* menu:

Remote ones have *remote* keyword as a Region:

PMM Add Instance ▾

Add remote MySQL Instance

*Hostname ⓘ

Name (default: Hostname) ⓘ

(default: 3306) ⓘ

*Username ⓘ

*Password ⓘ

Add instance

Query Analytics OS MySQL MongoDB PostgreSQL HA Cloud Insight PMM

- _PMM Add Instance
- _PMM Query Analytics Settings
- _PMM RDS and Remote Instances**
- _PMM System Summary

RDS and remote instances

Name	Endpoint	Region ↕	Engine ↕	Remove
Hostname	172.19.0.1:5432	remote	PostgreSQL	🗑️

🔍 1.17.0

ADDING AN AMAZON RDS DB INSTANCE TO PMM

New in version 1.5.0.

The *PMM Add Instance* is now a preferred method of adding an Amazon RDS DB instance to PMM. This method supports Amazon RDS DB instances that use Amazon Aurora, MySQL, or MariaDB engines.

- Essential AWS settings for monitoring Amazon RDS DB instances in PMM

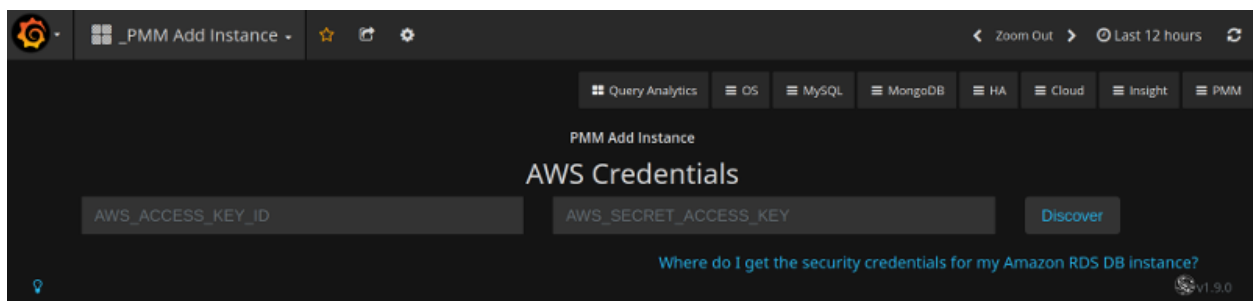


Fig. 6.1: Enter the access key ID and the secret access key of your IAM user to view Amazon RDS DB instances.

1. Open the PMM web interface and select the *PMM Add Instance* dashboard.
2. Select the *Add an AWS RDS MySQL or Aurora MySQL Instance* option in the dashboard.
3. Enter the access key ID and the secret access key of your IAM user.
4. Click the *Discover* button for PMM to retrieve the available Amazon RDS instances.

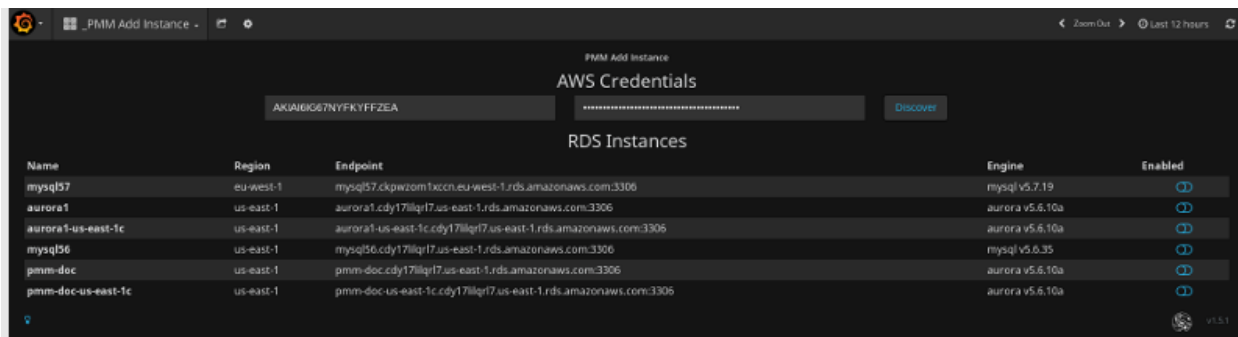


Fig. 6.2: PMM displays the available Amazon RDS instances

For each instance that you would like to monitor, select the *Enabled* button and enter the user name and password. Click *Connect*. You can now monitor your instances in the *Amazon RDS / Aurora MySQL Metrics*.

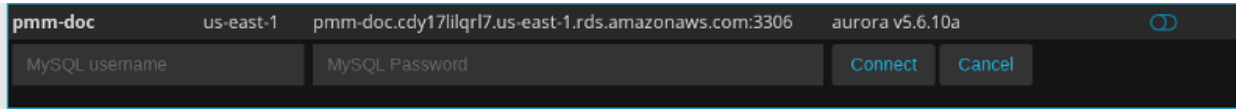


Fig. 6.3: Enter the DB user name and password to connect to the selected RDS or Amazon Aurora instance.

See also:

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

Essential AWS settings for monitoring Amazon RDS DB instances in PMM

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

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

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

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

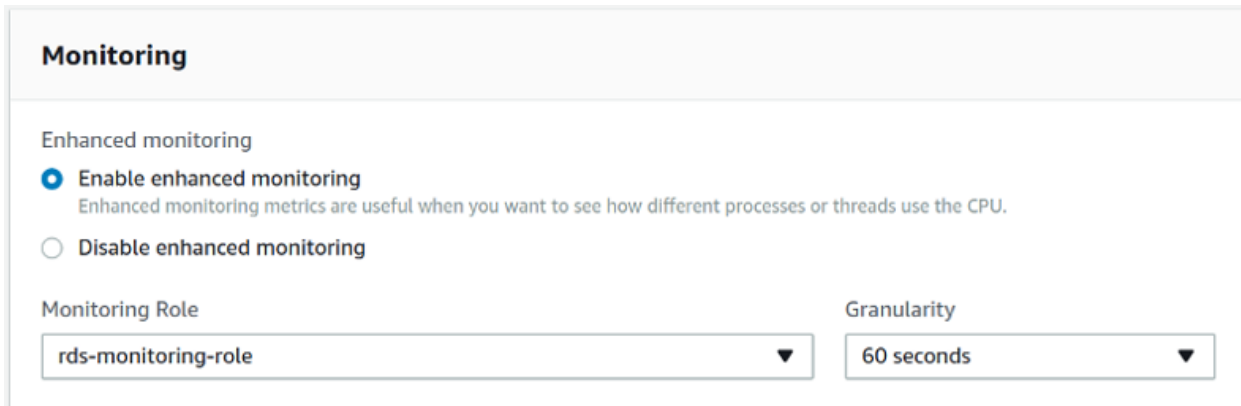


Fig. 6.4: Set the *Enable Enhanced Monitoring* option in the settings of your Amazon RDS DB instance.

See also:

Amazon RDS Documentation:

- [Modifying an Amazon RDS DB Instance](#)
- [More information about enhanced monitoring](#)
- [Setting Up](#)

- Getting started
- Creating a MySQL DB Instance
- Availability zones
- What privileges are automatically granted to the master user of an Amazon RDS DB instance?

Which ports should be open? See *Ports* in glossary

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

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

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

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

See also:

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

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

Creating a policy

A policy defines how AWS services can be accessed. Once defined it can be associated with an existing user or group.

To define a new policy use the IAM page at AWS.

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

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

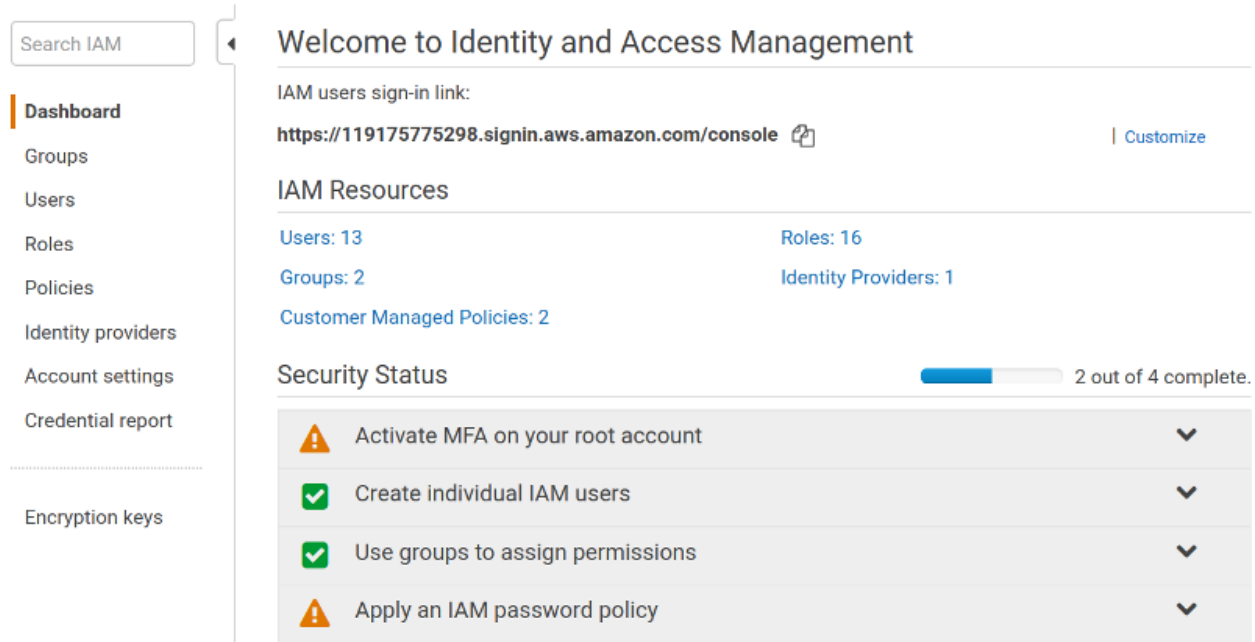


Fig. 6.5: The IAM page at AWS

```

    {
      "Sid": "Stmt1508410723001",
      "Effect": "Allow",
      "Action": [ "logs:DescribeLogStreams",
                  "logs:GetLogEvents",
                  "logs:FilterLogEvents" ],
      "Resource": [ "arn:aws:logs:*:*:log-
↪group:RDSOSMetrics:*" ] }
    ]
  }

```

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

See also:

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

Creating an IAM user

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

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

See also:

AWS Documentation:

Create policy

1 2

Review policy

Name*

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

Description

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

Summary

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

* Required Cancel Previous **Create policy**

Fig. 6.6: A new policy is ready to be created.

Search IAM

Add user **Delete user** ↻ ⚙️ ⓘ

<input type="checkbox"/>	User name	Groups	Access key age	Password age	Last activity	MFA	Access k
--------------------------	-----------	--------	----------------	--------------	---------------	-----	----------

Dashboard

Groups

Users

Roles

Policies

Identity providers

Account settings

Credential report

Encryption keys

Fig. 6.7: Navigate to *Users* on the IAM console

- Creating IAM users
- IAM roles

Creating an access key for an IAM user

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

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

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

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

See also:

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

Attaching a policy to an IAM user

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

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

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

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

See also:

Creating an IAM policy for PMM *Creating a policy*

Setting up the Amazon RDS DB Instance

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

See also:

Add permissions to pmm_user

1 2

Grant permissions

Use IAM policies to grant permissions. You can assign an existing policy or create a new one.

 Add user to group

 Copy permissions from existing user

 Attach existing policies directly

Attach one or more existing policies directly to the users or create a new policy. [Learn more](#)

Policy name	Type	Attachments	Description
<input checked="" type="checkbox"/> AmazonRDSforPMMPolicy	Customer managed	0	

Fig. 6.8: To attach, find the policy on the list and place a check mark to select it

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

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

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

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

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

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

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

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

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

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

See also:

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

CONFIGURING MYSQL FOR BEST RESULTS

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

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

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

- [Creating a MySQL User Account to Be Used with PMM](#)
- [Configuring the slow query log in Percona Server](#)
- [Configuring Performance Schema](#)
- [Configuring MySQL 8.0 for PMM](#)
- [Settings for Dashboards](#)
- [executing Custom 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 XtraDB Cluster), configure the *slow query log* to capture all queries and enable sampling. This will provide the most amount of information with the lowest overhead.

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

```
innodb_monitor_enable=all
userstat=1
```

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

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

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

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

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

Creating a MySQL User Account to Be Used with PMM

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

See also:

Using the `pmm-admin add` command to add a monitoring service *Adding MySQL metrics service*

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 'pass' WITH MAX_USER_CONNECTIONS 10;
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 as root `pmm-admin add mysql --help`.

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*. These are the queries the execution time of which is above a certain threshold. The threshold is defined by the `long_query_time` variable.

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

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

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`.

Configuring Performance Schema

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

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

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

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

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

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

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

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

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

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

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

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

See also:

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

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

1. Open the *PMM Query Analytics* dashboard.
2. Click the *Settings* button.
3. Open the *Settings* section.
4. Select *Performance Schema* in the *Collect from* drop-down list.
5. Click *Apply* to save changes.

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

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

```
pmm-admin add mysql --user root --password root --create-user --query-source_
↳perfschema
```

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

Configuring MySQL 8.0 for PMM

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

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

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

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

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

Altering the MySQL User

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

```
mysql> ALTER USER pmm@'localhost' IDENTIFIED WITH mysql_native_password BY '$eCR8Tp@s
↪$w*rD';
```

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

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

Changing the global MySQL Configuration

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

```
[mysqld]
default_authentication_plugin=mysql_native_password
```

See also:

Creating a MySQL User for PMM *What privileges are required to monitor a MySQL instance?*

More information about adding the MySQL query analytics monitoring service *Adding MySQL query analytics service*

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

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

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

Settings for Dashboards

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

See also:

More information about PMM dashboards [Metrics Monitor](#)

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 they all are enabled all the time. To enable all InnoDB metrics, set the global variable `innodb_monitor_enable` to `all`:

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

See also:

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

MySQL User Statistics

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

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

See also:

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

MySQL Documentation [Setting variables](#)

Percona Server Query Response Time Distribution

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

To enable collection of query response time:

1. Install the `QUERY_RESPONSE_TIME` plugins:

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

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

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

Related Information: Percona Server Documentation

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

executing Custom Queries

Starting from the version 1.15.0, PMM provides user the ability to take a SQL `SELECT` statement and turn the result set into metric series in PMM. The queries are executed at the `LOW RESOLUTION` level, which by default is every 60 seconds. A key advantage is that you can extend PMM to profile metrics unique to your environment (see users table example below), or to introduce support for a table that isn't part of PMM yet. This feature is on by default and only requires that you edit the configuration file and use valid YAML syntax. The default configuration file location is `/usr/local/percona/pmm-client/queries-mysqld.yml`.

Example - Application users table

We're going to take a users table of upvotes and downvotes and turn this into two metric series, with a set of labels. Labels can also store a value. You can filter against labels.

Browsing metrics series using Advanced Data Exploration Dashboard

Lets look at the output so we understand the goal - take data from a MySQL table and store in PMM, then display as a metric series. Using the Advanced Data Exploration Dashboard you can review your metric series.

MySQL table

Lets assume you have the following users table that includes true/false, string, and integer types.

```
SELECT * FROM `users`
+----+-----+-----+-----+-----+-----+-----+-----+
↪+-----+-----+-----+-----+-----+-----+-----+-----+
| id | app | user_type      | last_name | first_name | logged_in | active_subscription_
↪| banned | upvotes | downvotes |
+----+-----+-----+-----+-----+-----+-----+-----+
↪+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | app2 | unprivileged | Marley   | Bob       | 1       | 1
↪| 0 | 100 | 25 |
| 2 | app3 | moderator    | Young   | Neil     | 1       | 1
↪| 1 | 150 | 10 |
| 3 | app4 | unprivileged | OConnor | Sinead   | 1       | 1
↪| 0 | 25 | 50 |
| 4 | app1 | unprivileged | Yorke   | Thom     | 0       | 1
↪| 0 | 100 | 100 |
```

	5		app5		admin		Buckley		Jeff		1		1
↔		0		175		0							
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+													
↔	+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+												

Explaining the YAML syntax

We'll go through a simple example and mention what's required for each line. The metric series is constructed based on the first line and appends the column name to form metric series. Therefore the number of metric series per table will be the count of columns that are of type GAUGE or COUNTER. This metric series will be called `appl_users_metrics_downvotes`:

```

appl_users_metrics:                                ## leading section of your metric_
↔series.
  query: "SELECT * FROM appl.users"                 ## Your query. Don't forget the_
↔schema name.
  metrics:                                          ## Required line to start the_
↔list of metric items
    - downvotes:                                   ## Name of the column returned by_
↔the query. Will be appended to the metric series.
      usage: "COUNTER"                             ## Column value type. COUNTER_
↔will make this a metric series.
      description: "Number of upvotes"             ## Helpful description of the_
↔column.

```

Full queries-mysqld.yml example

Each column in the SELECT is named in this example, but that isn't required, you can use a SELECT * as well. Notice the format of `schema.table` for the query is included.

```

---
appl_users_metrics:
  query: "SELECT app,first_name,last_name,logged_in,active_subscription,banned,
↔upvotes,downvotes FROM appl.users"
  metrics:
    - app:
      usage: "LABEL"
      description: "Name of the Application"
    - user_type:
      usage: "LABEL"
      description: "User's privilege level within the Application"
    - first_name:
      usage: "LABEL"
      description: "User's First Name"
    - last_name:
      usage: "LABEL"
      description: "User's Last Name"
    - logged_in:
      usage: "LABEL"
      description: "User's logged in or out status"
    - active_subscription:
      usage: "LABEL"
      description: "Whether User has an active subscription or not"
    - banned:
      usage: "LABEL"

```

```
    description: "Whether user is banned or not"
  - upvotes:
    usage: "COUNTER"
    description: "Count of upvotes the User has earned. Upvotes once granted,
↳cannot be revoked, so the number can only increase."
    - downvotes:
    usage: "GAUGE"
    description: "Count of downvotes the User has earned. Downvotes can be,
↳revoked so the number can increase as well as decrease."
  ...
```

This custom query description should be placed in a YAML file (`queries-mysqld.yml` by default) on the corresponding server with MySQL.

In order to modify the location of the queries file, for example if you have multiple `mysqld` instances per server, you need to explicitly identify to the PMM Server MySQL with the `pmm-admin add` command after the double dash:

```
pmm-admin add mysql:metrics ... -- --queries-file-name=/usr/local/percona/pmm-client/
↳query.yml
```


CONFIGURING POSTGRESQL FOR MONITORING

Monitoring PostgreSQL metrics with the `postgres_exporter` is enabled by `pmm-admin add postgresql` command. The `postgresql` alias will set up `postgresql:metrics` and also `linux:metrics` on a host (for more information, see [Adding monitoring services](#)).

`pmm-admin` supports passing PostgreSQL connection information via following flags:

Flag	Description
<code>--host</code>	PostgreSQL host
<code>--password</code>	PostgreSQL password
<code>--port</code>	PostgreSQL port
<code>--user</code>	PostgreSQL user

An example command line would look like this:

```
pmm-admin add postgresql --host=localhost --password='secret' --port=5432 --user=pmm_
↪user
```

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

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

Supported versions of PostgreSQL

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

Setting Up the Required Permissions

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

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

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

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

CONFIGURING MONGODB FOR MONITORING IN *PMM QUERY ANALYTICS*

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

Supported versions of MongoDB

QAN supports MongoDB version 3.2 or higher.

- *Setting Up the Essential Permissions*
- *Enabling Profiling*

Setting Up the Essential Permissions

For `mongodb:metrics` and `mongodb:queries` monitoring services to be able work in QAN, you need to set up the `mongodb_exporter` user. This user should be assigned the `clusterMonitor` role for the `admin` database and the `read` role for the `local` database.

The following example that you can run in the MongoDB shell, adds the `mongodb_exporter` user and assigns the appropriate roles.

```
db.getSiblingDB("admin").createUser({
  user: "mongodb_exporter",
  pwd: "s3cR#tpa$$worD",
  roles: [
    { role: "clusterMonitor", db: "admin" },
    { role: "read", db: "local" }
  ]
})
```

Then, you need to pass the user name and password in the value of the `--uri` option when adding the `mongodb:metrics` monitoring service in the `pmm-admin add` command:

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

See also:

Adding a `mongodb:metrics` monitoring service *Adding MongoDB metrics service*

Enabling Profiling

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

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

It is required that profiling of the monitored MongoDB databases be enabled.

Note that profiling is not enabled by default because it may reduce the performance of your MongoDB server.

Enabling Profiling on Command Line

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

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

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

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

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

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

See also:

--rateLimit in PSMDB documentation <https://www.percona.com/doc/percona-server-for-mongodb/LATEST/rate-limit.html>

Enabling Profiling in the Configuration File

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

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

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

These settings affect `mongod` in the same way as the command line options described in section [Enabling Profiling on Command Line](#). Note that the configuration file is in the **YAML** format. In this format the indentation of your lines is important as it defines levels of nesting.

Restart the `mongod` service to enable the settings. Run this command as root or by using the `sudo` command

```
$ service mongod restart
```

Related Information

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

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

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

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

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

SECURITY FEATURES IN *PERCONA MONITORING AND MANAGEMENT*

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

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

In this chapter

- Enabling SSL Encryption
 - Valid certificates
 - Self-signed certificates
 - Enabling SSL when connecting PMM Client to PMM Server
- Enabling Password Protection
- Combining Security Features

Enabling SSL Encryption

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

Valid certificates

To use a valid SSL certificate, mount the directory with the certificate files to `/srv/nginx/` when *running the PMM Server container*.

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

The directory (`/etc/pmm-certs` in this example) that you intend to mount must contain the following files:

- `certificate.crt`
- `certificate.key`

- ca-certs.pem
- dhparam.pem

Note: To enable SSL encryption, The container publishes port *443* instead of *80*.

Alternatively, you can use **docker cp** to copy the files to an already existing `pmm-server` container.

```
$ docker cp certificate.crt pmm-server:/srv/nginx/certificate.crt
$ docker cp certificate.key pmm-server:/srv/nginx/certificate.key
$ docker cp ca-certs.pem pmm-server:/srv/nginx/ca-certs.pem
$ docker cp dhparam.pem pmm-server:/srv/nginx/dhparam.pem
```

This example assumes that you have changed to the directory that contains the certificate files.

Self-signed certificates

The PMM Server images (Docker, OVF, and AMI) already include self-signed certificates. To be able to use them in your Docker container, make sure to publish the container's port *443* to the host's port *443* when running the **docker run** command.

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

Enabling SSL when connecting PMM Client to PMM Server

Then, you need to enable SSL when *connecting a PMM Client to a 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
```

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.

By default, the user name is `pmm`. You can change it by passing the `SERVER_USER` environment variable. Note that the following example uses an insecure port `80` which is typically used for HTTP connections.

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


```
$ docker run -d -p 80:80 \
  --volumes-from pmm-data \
  --name pmm-server \
  -e SERVER_USER=jsmith \
  -e SERVER_PASSWORD=SomeR4ndom-Pa$$w0rd \
  --restart always \
  percona/pmm-server:latest
```

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 a PMM Client to a PMM Server*:

```
$ pmm-admin config --server 192.168.100.1 --server-user jsmith --server-password_
↳pass1234
```

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=SomeR4ndom-Pa$$w0rd \
  -v /etc/pmm-certs:/srv/nginx \
  --restart always \
  percona/pmm-server:latest
```

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-insecure-ssl
```

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:

```
$ 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
```


METRICS MONITOR DASHBOARDS

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

Insight

Home Dashboard

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

See also:

Overview of PMM *Tools of PMM*

General Information

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

Shared and Recently Used Dashboards

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

Statistics

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

Environment Overview

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

- CPU Busy
- Memory Available
- Disk Reads
- Disk Writes
- Network IO
- DB Connections

- DB QPS
- Virtual CPUs
- RAM
- Host Uptime
- DB Uptime

PMM System Summary Dashboard

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

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

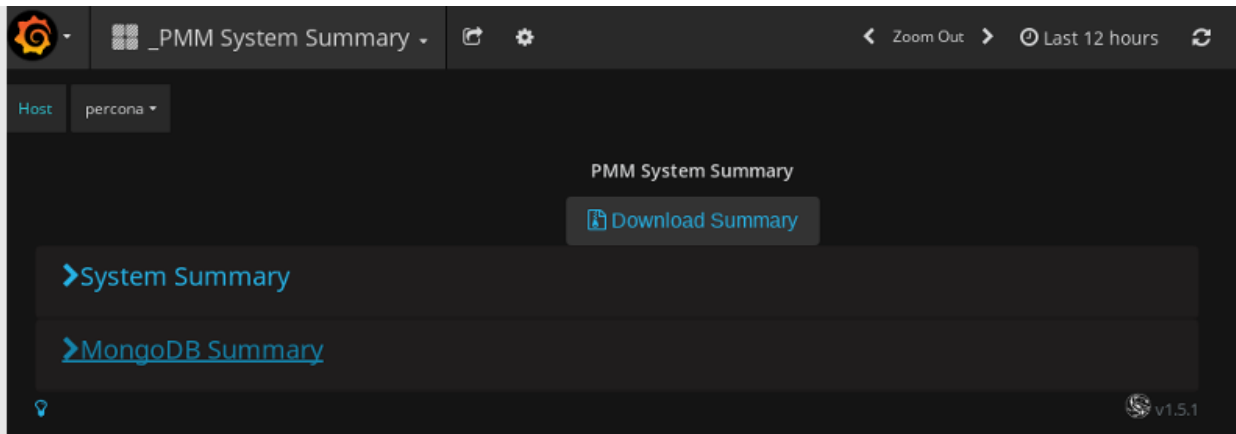


Fig. 11.1: Accessing information about the system and database

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

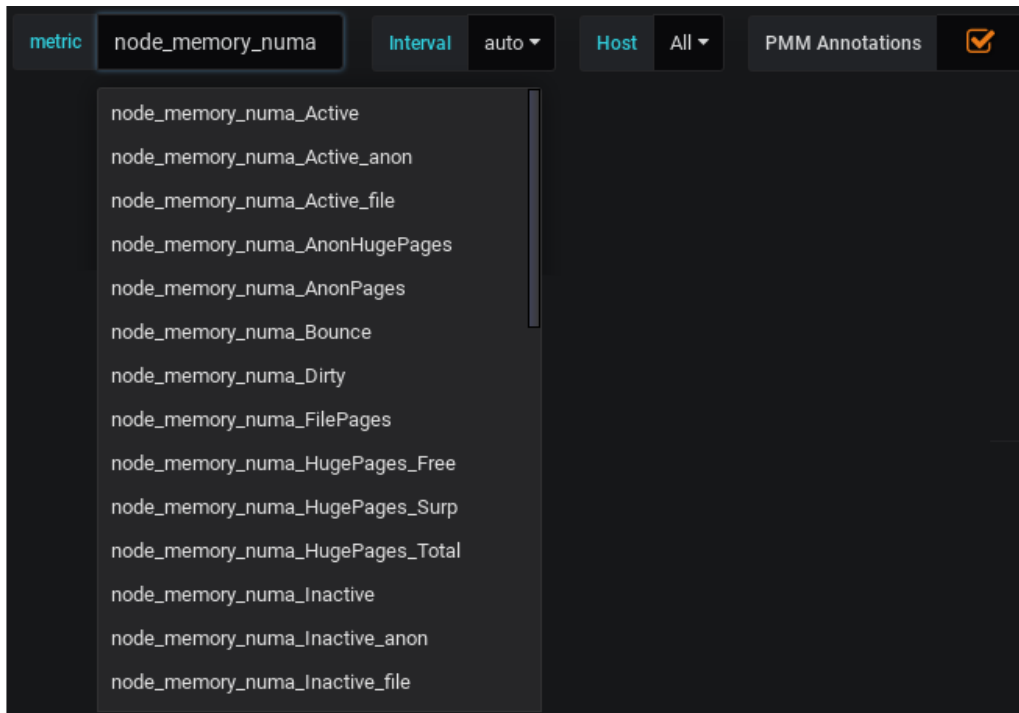
Advanced Data Exploration Dashboard

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

Added NUMA related metrics

New in version 1.13.0.

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



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

View actual metric values (Gauge)

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

View actual metric values (Counters)

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

View actual metric values (Counters)

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

See also:

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

Cross Server Graphs

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

Load Average

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

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

View all metrics of *Cross Server Graphs*

See also:

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

MySQL Queries

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

View all metrics of *Cross Server Graphs*

See also:

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

MySQL Traffic

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

View all metrics of *Cross Server Graphs*

Summary Dashboard

- *CPU Usage*
- *Processes*
- *Network Traffic*
- *I/O Activity*
- *Disk Latency*
- *MySQL Queries*

- [InnoDB Row Operations](#)
- [Top MySQL Commands](#)
- [Top MySQL Handlers](#)

CPU Usage

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

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

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

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

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

View all metrics of [Summary Dashboard](#)

See also:

Linux CPU Statistics

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

Processes

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

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

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

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

View all metrics of [Summary Dashboard](#)

See also:

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

Network Traffic

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

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

View all metrics of *Summary Dashboard*

I/O Activity

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

View all metrics of *Summary Dashboard*

Disk Latency

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

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

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

View all metrics of *Summary Dashboard*

MySQL Queries

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

View all metrics of *Summary Dashboard*

InnoDB Row Operations

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

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

View all metrics of *Summary Dashboard*

Top MySQL Commands

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

View all metrics of *Summary Dashboard*

Top MySQL Handlers

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

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

View all metrics of *Summary Dashboard*

Trends Dashboard

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

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

See also:

MySQL Documentation:

[Questions](#)

Metrics

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

CPU Usage

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

View all metrics of *Trends Dashboard*

I/O Read Activity

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

View all metrics of *Trends Dashboard*

I/O Write Activity

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

View all metrics of *Trends Dashboard*

MySQL Questions

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

View all metrics of *Trends Dashboard*

InnoDB Rows Read

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

View all metrics of *Trends Dashboard*

InnoDB Rows Changed

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

View all metrics of *Trends Dashboard*

Network Overview Dashboard

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

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

Last Hour Statistic

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

View all metrics of *Network Overview Dashboard*

Network Traffic

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

View all metrics of *Network Overview Dashboard*

Network Traffic Details

This section offers the following metrics:

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

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

Network Netstat TCP

This section offers the following metrics:

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

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

Network Netstat UDP

In this section, you can find the following metrics:

- Netstat: UDP
- UDP Lite

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

InDatagrams Packets received

OutDatagrams Packets sent

InCsumErrors Datagrams with checksum errors

InErrors Datagrams that could not be delivered to an application

RcvbufErrors Datagrams for which not enough socket buffer memory to receive

SndbufErrors Datagrams for which not enough socket buffer memory to transmit

NoPorts Datagrams received on a port with no listener

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

ICMP

This section has the following metrics:

- ICMP Errors
- Messages/Redirects

- Echos
- Timestamps/Mask Requests

ICMP Errors

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

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

InDestUnreachs Destination Unreachable messages received

OutDestUnreachs Destination Unreachable messages sent

InType3 Destination unreachable

OutType3 Destination unreachable

InCsumErrors Messages with ICMP checksum errors

InTimeExcds Time Exceeded messages received

Messages/Redirects

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

InRedirects Redirect messages received

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

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

Echos

InEchoReps Echo Reply messages received

InEchos Echo (request) messages received

OutEchoReps Echo Reply messages sent

OutEchos Echo (request) messages sent

Timestamps/Mask Requests

InAddrMaskReps Address Mask Reply messages received

InAddrMasks Address Mask Request messages received

OutAddrMaskReps Address Mask Reply messages sent

OutAddrMasks Address Mask Request messages sent

InTimestampReps Timestamp Reply messages received

InTimestamps Timestamp Request messages received

OutTimestampReps Timestamp Reply messages sent

OutTimestamps Timestamp Request messages sent

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

OS Dashboards

CPU Utilization Details (Cores)

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

Overall CPU Utilization

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

- system
- user
- iowait
- softirq

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

System

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

User

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

Iowait

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

Softirq

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

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

Current CPU Core Utilization

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

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

All Cores - Total

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

- system
- user
- iowait
- steal
- other

Components

System

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

User

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

Iowait

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

Steal

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

Other

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

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

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

Disk space

- [Mountpoint Usage](#)
- [Mountpoint](#)

Mountpoint Usage

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

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

View all metrics of [Disk space](#)

Mountpoint

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

Used Is the amount of space used

Free Is the amount of space not in use

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

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

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

View all metrics of [Disk space](#)

System Overview Dashboard

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

- CPU
- Memory
- Disk
- Network

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

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

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

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

Compare System Parameters Dashboard

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

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

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

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

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

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

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

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

NUMA Overview Dashboard

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

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

..note:

Users who already have [General system metrics service](#) monitored and would like to add NUMA metrics need to remove and re-add `linux:metrics` on the node:


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

Memory Usage

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

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

Free Memory Percent

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

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

NUMA Memory Usage Types

Dirty Memory waiting to be written back to disk

Bounce Memory used for block device bounce buffers

Mapped Files which have been mmaped, such as libraries

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

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

NUMA Allocation Hits

Memory successfully allocated on this node as intended.

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

NUMA Allocation Missed

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

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

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

Anonymous Memory

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

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

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

NUMA File (PageCache)

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

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

View all metrics of *NUMA Overview Dashboard*

Shared Memory

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

View all metrics of *NUMA Overview Dashboard*

HugePages Statistics

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

Free The number of hugepages not being allocated by a process

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

View all metrics of *NUMA Overview Dashboard*

Local Processes

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

View all metrics of *NUMA Overview Dashboard*

Remote Processes

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

View all metrics of *NUMA Overview Dashboard*

Slab Memory

Slab Allocation is a memory management mechanism intended for the efficient memory allocation of kernel objects.

SReclaimable The part of the Slab that might be reclaimed (such as caches).

SUnreclaim The part of the Slab that can't be reclaimed under memory pressure

View all metrics of *NUMA Overview Dashboard*

Prometheus Dashboards

Prometheus Dashboard

The *Prometheus* dashboard informs how Prometheus functions.

See also:

Overview of PMM *Tools of PMM*

All dashboards *Metrics Monitor Dashboards*

Prometheus overview

This section shows the most essential parameters of the system where Prometheus is running, such as CPU and memory usage, scrapes performed and the samples ingested in the head block.

Resources

This section provides details about the consumption of CPU and memory by the Prometheus process. This section contains the following metrics:

- Prometheus Process CPU Usage
- Prometheus Process Memory Usage

Storage (TSDB)

This section includes a collection of metrics related to the usage of storage. It includes the following metrics:

- Data blocks (Number of currently loaded data blocks)
- Total chunks in the head block
- Number of series in the head block
- Current retention period of the head block
- Activity with chunks in the head block
- Reload block data from disk

Scraping

This section contains metrics that help monitor the scraping process. This section contains the following metrics:

- Ingestion
- Prometheus Targets
- Scraped Target by Job
- Scrape Time by Job
- Scraped Target by Instance
- Scraped Time by Instance
- Scrapes by Target Frequency
- Scrape Frequency Versus Target
- Scraping Time Drift
- Prometheus Scrape Interval Variance
- Slowest Jobs
- Largest Samples Jobs

Queries

This section contains metrics that monitor Prometheus queries. This section contains the following metrics:

- Prometheus Queries
- Prometheus Query Execution
- Prometheus Query Execution Latency
- Prometheus Query Execution Load

Network

Metrics in this section help detect network problems.

- HTTP Requests by Handler
- HTTP Errors
- HTTP Avg Response time by Handler
- HTTP 99% Percentile Response time by Handler
- HTTP Response Average Size by Handler
- HTTP 99% Percentile Response Size

Time Series Information

This section shows the top 10 metrics by time series count and the top 10 hosts by time series count.

System Level Metrics

Metrics in this section give an overview of the essential system characteristics of PMM Server. This information is also available from the *System Overview* dashboard.

PMM Server Logs

This section contains a link to download the logs collected from your PMM Server and further analyze possible problems. The exported logs are requested when you submit a bug report.

Prometheus Exporter Status

The Prometheus Exporter Status dashboard reports the consumption of resources by the Prometheus exporters used by PMM. For each exporter, this dashboard reveals the following information:

- CPU usage
- Memory usage
- File descriptors used
- Exporter uptime

See also:

All PMM exporters `pmm.list.exporter`

Summary information about the usage of Prometheus exporters [Prometheus Exporters Overview](#)

Prometheus Exporters Overview

The Prometheus Exporters Overview dashboard provides the summary of how exporters are used across the selected hosts.

See also:

Percona Database Performance Blog

[Understand Your Prometheus Exporters with Percona Monitoring and Management \(PMM\)](#)

Prometheus documentation

Exporters and integrations

- *Prometheus Exporters Summary*
- *Prometheus Exporters Resource Usage by Host*
- *Prometheus Exporters Resource Usage by Type*
- *List of Hosts*

Prometheus Exporters Summary

This section provides a summary of how exporters are used across the selected hosts. It includes the average usage of CPU and memory as well as the number of hosts being monitored and the total number of running exporters.

Metrics in this section

- **Avg CPU Usage per Host** shows the average CPU usage in percent per host for all exporters.
- **Avg Memory Usage per Host** shows the Exporters average Memory usage per host.
- **Monitored Hosts** shows the number of monitored hosts that are running Exporters.
- **Exporters Running** shows the total number of Exporters running with this PMM Server instance.

Note: The CPU usage and memory usage do not include the additional CPU and memory usage required to produce metrics by the application or operating system.

Prometheus Exporters Resource Usage by Host

This section shows how resources, such as CPU and memory, are being used by the exporters for the selected hosts.

Metrics in this section

- **CPU Usage** plots the Exporters' CPU usage across each monitored host (by default, All hosts).
- **Memory Usage** plots the Exporters' Memory usage across each monitored host (by default, All hosts).

Prometheus Exporters Resource Usage by Type

This section shows how resources, such as CPU and memory, are being used by the exporters for host types: MySQL, MongoDB, ProxySQL, and the system.

Metrics in this section

- **CPU Cores Used** shows the Exporters' CPU Cores used for each type of Exporter.
- **Memory Usage** shows the Exporters' memory used for each type of Exporter.

List of Hosts

At the bottom, this dashboard shows details for each running host.

- **CPU Used** show the CPU usage as a percentage for all Exporters.
- **Mem Used** shows total Memory Used by Exporters.
- **Exporters Running** shows the number of Exporters running.
- **RAM** shows the total amount of RAM of the host.
- **Virtual CPUs** shows the total number of virtual CPUs on the host.

You can click the value of the *CPU Used*, *Memory Used*, or *Exporters Running* column to open the *Prometheus Exporter Status* for further analysis.

Related information: Prometheus Documentation

Exporters and integrations <https://prometheus.io/docs/instrumenting/exporters>

MySQL Dashboards

MySQL Amazon Aurora Metrics

This dashboard provides metrics for analyzing Amazon Aurora instances.

- Amazon Aurora Transaction Commits
- Amazon Aurora Load
- Aurora Memory Used
- Amazon Aurora Statement Latency
- Amazon Aurora Special Command Counters
- Amazon Aurora Problems

Amazon Aurora Transaction Commits

This graph shows number of commits which the Amazon Aurora engine performed as well as the average commit latency. Graph Latency does not always correlates with number of commits performed and can quite high in certain situations.

Amazon Aurora Load

This graph shows what statements contribute most load on the system as well as what load corresponds to Amazon Aurora transaction commit.

- Write Transaction Commit Load: Load in Average Active Sessions per second for COMMIT operations
- UPDATE load: load in Average Active Sessions per second for UPDATE queries
- SELECT load: load in Average Active Sessions per second for SELECT queries

- DELETE load: load in Average Active Sessions per second for DELETE queries
- INSERT load: load in Average Active Sessions per second for INSERT queries

Aurora Memory Used

This graph shows how much memory is used by Amazon Aurora lock manager as well as amount of memory used by Amazon Aurora to store Data Dictionary.

- Aurora Lock Manager Memory: the amount of memory used by the Lock Manager, the module responsible for handling row lock requests for concurrent transactions.
- Aurora Dictionary Memory: the amount of memory used by the Dictionary, the space that contains metadata used to keep track of database objects, such as tables and indexes.

Amazon Aurora Statement Latency

This graph shows average latency for most important types of statements. Latency spikes are often indicative of the instance overload.

- DDL Latency: Average time to execute DDL queries
- DELETE Latency: average time to execute DELETE queries
- UPDATE Latency: average time to execute UPDATE queries
- SELECT Latency: average time to execute SELECT queries
- INSERT Latency: average time to execute INSERT queries

Amazon Aurora Special Command Counters

Amazon Aurora MySQL allows a number of commands which are not available from standard MySQL. This graph shows usage of such commands. Regular `unit_test` calls can be seen in default Amazon Aurora install, the rest will depend on your workload.

show_volume_status The number of executions per second of the command `SHOW VOLUME STATUS`. The `SHOW VOLUME STATUS` query returns two server status variables: `Disks` and `Nodes`. These variables represent the total number of logical blocks of data and storage nodes, respectively, for the DB cluster volume.

awslambda The number of AWS Lambda calls per second. AWS Lambda is an event-driven, serverless computing platform provided by AWS. It is a compute service that runs codes in response to an event. You can run any kind of code from Aurora invoking Lambda from a stored procedure or a trigger.

alter_system The number of executions per second of the special query `ALTER SYSTEM`, that is a special query to simulate an instance crash, a disk failure, a disk congestion or a replica failure. It is a useful query for testing the system.

Amazon Aurora Problems

This metric shows different kinds of internal Amazon Aurora MySQL problems which should be zero in case of normal operation.

- Reserved mem Exceeded Incidents
- Missing History on Replica Incidents
- Thread deadlocks: number of deadlocks per second

MySQL Command/Handler Counters Compare Dashboard

This dashboard shows server status variables. On this dashboard, you may select multiple servers and compare their counters simultaneously.

Server status variables appear in two sections: *Commands* and *Handlers*. Choose one or more variables in the *Command* and *Handler* fields in the top menu to select the variables which will appear in the *COMMANDS* or *HANDLERS* section for each host. Your comparison may include from one up to three hosts.

By default or if no item is selected in the menu, PMM displays each command or handler respectively.

See also:

MySQL Documentation: Server Status Variables <https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html>

MySQL InnoDB Compression Dashboard

This dashboard helps you analyze the efficiency of InnoDB compression.

See also:

MySQL Documentation <https://dev.mysql.com/doc/refman/5.7/en/innodb-information-schema-compression-tables.html>

- *Compression level and failure rate threshold*
- *Statistics of Compression Operations*
- *CPU Core Usage*
- *Buffer Pool Total*
- *Buffer Pool All*

Compression level and failure rate threshold

InnoDB Compression Level

The level of zlib compression to use for InnoDB compressed tables and indexes.

InnoDB Compression Failure Threshold

The compression failure rate threshold for a table.

Compression Failure Rate Threshold

The maximum percentage that can be reserved as free space within each compressed page, allowing room to reorganize the data and modification log within the page when a compressed table or index is updated and the data might be recompressed.

Write Pages to the Redo Log

Specifies whether images of re-compressed pages are written to the redo log. Re-compression may occur when changes are made to compressed data.

Statistics of Compression Operations

This section contains the following metrics:

- Compress Attempts
- Uncompressed Attempts
- Compression Success Ratio

Compress Attempts

Number of compression operations attempted. Pages are compressed whenever an empty page is created or the space for the uncompressed modification log runs out.

Uncompressed Attempts

Number of uncompression operations performed. Compressed InnoDB pages are uncompressed whenever compression fails, or the first time a compressed page is accessed in the buffer pool and the uncompressed page does not exist.

CPU Core Usage

- CPU Core Usage for Compression
- CPU Core Usage for Uncompression

Buffer Pool Total

- Total Used Pages
- Total Free Pages

Buffer Pool All

- Used Pages (Buffer Pool 0)
- Pages Free (Buffer Pool 0)

MySQL InnoDB Metrics

This dashboard contains metrics that help analyze how the InnoDB engine performs.

- *InnoDB Checkpoint Age*
- *InnoDB Transactions*

- [InnoDB Row Operations](#)
- [InnoDB Row Lock Time](#)
- [InnoDB I/O](#)
- [InnoDB Log File Usage Hourly](#)
- [InnoDB Deadlocks](#)
- [InnoDB Condition Pushdown](#)
- [Other Metrics](#)

InnoDB Checkpoint Age

The maximum checkpoint age is determined by the total length of all transaction log files (*innodb_log_file_size*).

When the checkpoint age reaches the maximum checkpoint age, blocks are flushed synchronously. The rule of thumb is to keep one hour of traffic in those logs and let the checkpointing perform its work as smooth as possible. If you don't do this, InnoDB will do synchronous flushing at the worst possible time, i.e. when you are busiest.

View all metrics of [MySQL InnoDB Metrics](#)

InnoDB Transactions

InnoDB is an MVCC storage engine, which means you can start a transaction and continue to see a consistent snapshot even as the data changes. This is implemented by keeping old versions of rows as they are modified.

The *InnoDB History List* is the undo logs which are used to store these modifications. They are a fundamental part of the InnoDB transactional architecture.

If the history length is rising regularly, do not let open connections linger for a long period as this can affect the performance of InnoDB considerably. It is also a good idea to look for long running queries in QAN.

View all metrics of [MySQL InnoDB Metrics](#)

InnoDB Row Operations

This metric allows you to see which operations occur and the number of rows affected per operation. A metric like *Queries Per Second* will give you an idea of queries, but one query could effect millions of rows.

View all metrics of [MySQL InnoDB Metrics](#)

InnoDB Row Lock Time

When data is locked, then that means that another session cannot update that data until the lock is released (which unlocks the data and allows other users to update that data. Locks are usually released by either a ROLLBACK or COMMIT SQL statement.

InnoDB implements standard row-level locking where there are two types of locks, shared (S) locks and exclusive (X) locks.

A shared (S) lock permits the transaction that holds the lock to read a row. An exclusive (X) lock permits the transaction that holds the lock to update or delete a row. *Average Row Lock Wait Time* is the row lock wait time divided by the number of row locks.

Row Lock Waits indicates how many times a transaction waited on a row lock per second.

Row Lock Wait Load is a rolling 5 minute average of *Row Lock Waits*.

View all metrics of *MySQL InnoDB Metrics*

See also:

MySQL Server Documentation: Shared lock https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_shared_lock

MySQL Server Documentation: Exclusive lock https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_exclusive_lock

MySQL Server Documentation: InnoDB locking <https://dev.mysql.com/doc/refman/5.7/en/innodb-locking.html>

InnoDB I/O

This metric has the following series:

- *Data Writes* is the total number of InnoDB data writes.
- *Data Reads* is the total number of InnoDB data reads (OS file reads).
- *Log Writes* is the number of physical writes to the InnoDB redo log file.
- *Data Fsyncs* is the number of fsync() operations. The frequency of fsync() calls is influenced by the setting of the `innodb_flush_method` configuration option.

View all metrics of *MySQL InnoDB Metrics*

InnoDB Log File Usage Hourly

Along with the buffer pool size, *innodb_log_file_size* is the most important setting when we are working with InnoDB. This graph shows how much data was written to InnoDB's redo logs over each hour. When the InnoDB log files are full, InnoDB needs to flush the modified pages from memory to disk.

The rule of thumb is to keep one hour of traffic in those logs and let the checkpointing perform its work as smooth as possible. If you don't do this, InnoDB will do synchronous flushing at the worst possible time, ie when you are busiest.

This graph can help guide you in setting the correct *innodb_log_file_size*.

View all metrics of *MySQL InnoDB Metrics*

See also:

Percona Database Performance Blog: Calculating a good InnoDB log file size <https://www.percona.com/blog/2008/11/21/how-to-calculate-a-good-innodb-log-file-size/>

Percona Server Documentation: Improved InnoDB I/O scalability http://www.percona.com/doc/percona-server/5.5/scalability/innodb_io_55.html#innodb_log_file_size

InnoDB Deadlocks

A deadlock in MySQL happens when two or more transactions mutually hold and request for locks, creating a cycle of dependencies. In a transaction system, deadlocks are a fact of life and not completely avoidable. InnoDB automatically detects transaction deadlocks, rolls back a transaction immediately and returns an error.

View all metrics of *MySQL InnoDB Metrics*

See also:

Percona Database Performance Blog: Dealing with MySQL deadlocks <https://www.percona.com/blog/2014/10/28/how-to-deal-with-mysql-deadlocks/>

InnoDB Condition Pushdown

Index Condition Pushdown (ICP) is an optimization for the case where MySQL retrieves rows from a table using an index.

Without ICP, the storage engine traverses the index to locate rows in the base table and returns them to the MySQL server which evaluates the `WHERE` condition for the rows. With ICP enabled, and if parts of the `WHERE` condition can be evaluated by using only columns from the index, the MySQL server pushes this part of the `WHERE` condition down to the storage engine. The storage engine then evaluates the pushed index condition by using the index entry and only if this is satisfied is the row read from the table.

ICP can reduce the number of times the storage engine must access the base table and the number of times the MySQL server must access the storage engine.

View all metrics of *MySQL InnoDB Metrics*

See also:

MySQL Server Documentation: Index Condition Pushdown optimisation <https://dev.mysql.com/doc/refman/5.7/en/index-condition-pushdown-optimization.html>

Percona Database Performance Blog: ICP counters and how to interpret them https://www.percona.com/blog/2017/05/09/mariadb-handler_icp_-counters-what-they-are-and-how-to-use-them/

Other Metrics

- InnoDB Logging Performance
- InnoDB Buffer Pool Content
- InnoDB Buffer Pool Pages
- InnoDB Buffer Pool I/O
- InnoDB Buffer Pool Requests
- InnoDB Buffer Read-Ahead
- InnoDB Change Buffer
- InnoDB Change Buffer Activity

MySQL InnoDB Metrics (Advanced) Dashboard

The MySQL InnoDB Metrics (Advanced) dashboard contains metrics that provide detailed information about the performance of the InnoDB storage engine on the selected MySQL host. This dashboard contains the following metrics:

Important: If you do not see any metric, try running the following command in the MySQL client:

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

Metrics of MySQL InnoDB Metrics (Advanced) Dashboard

- *Change Buffer Performance*
- *InnoDB Log Buffer Performance*
- *InnoDB Page Splits*
- *InnoDB Page Reorgs*
- *InnoDB Purge Performance*
- *InnoDB Locking*
- *InnoDB Main Thread Utilization*
- *InnoDB Transactions Information*
- *InnoDB Undo Space Usage*
- *InnoDB Activity*
- *InnoDB Contention - OS Waits*
- *InnoDB Contention - Spin Rounds*
- *InnoDB Group Commit Batch Size*
- *InnoDB Purge Throttling*
- *InnoDB AHI Usage*
- *InnoDB AHI Maintenance*
- *InnoDB Online DDL*
- *InnoDB Defragmentation*

Change Buffer Performance

This metric shows the activity on the InnoDB change buffer. The InnoDB change buffer records updates to non-unique secondary keys when the destination page is not in the buffer pool. The updates are applied when the page is loaded in the buffer pool, prior to its use by a query. The merge ratio is the number of insert buffer changes done per page, the higher the ratio the better is the efficiency of the change buffer.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

InnoDB Log Buffer Performance

The InnoDB Log Buffer Performance graph shows the efficiency of the InnoDB log buffer. The InnoDB log buffer size is defined by the `innodb_log_buffer_size` parameter and illustrated on the graph by the *Size* graph. *Used* is the amount of the buffer space that is used. If the *Used* graph is too high and gets close to *Size*, additional log writes will be required.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

InnoDB Page Splits

The InnoDB Page Splits graph shows the InnoDB page maintenance activity related to splitting and merging pages. When an InnoDB page, other than the top most leaf page, has too much data to accept a row update or a row insert, it has to be split in two. Similarly, if an InnoDB page, after a row update or delete operation, ends up being less than half full, an attempt is made to merge the page with a neighbor page. If the resulting page size is larger than

the InnoDB page size, the operation fails. If your workload causes a large number of page splits, try lowering the `innodb_fill_factor` variable (5.7+).

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

InnoDB Page Reorgs

The InnoDB Page Reorgs graph shows information about the page reorganization operations. When a page receives an update or an insert that affect the offset of other rows in the page, a reorganization is needed. If the reorganization process finds out there is not enough room in the page, the page will be split. Page reorganization can only fail for compressed pages.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

InnoDB Purge Performance

The InnoDB Purge Performance graph shows metrics about the page purging process. The purge process removed the undo entries from the history list and cleanup the pages of the old versions of modified rows and effectively remove deleted rows.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

InnoDB Locking

The InnoDB Locking graph shows the row level lock activity inside InnoDB.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

InnoDB Main Thread Utilization

The InnoDB Main Thread Utilization graph shows the portion of time the InnoDB main thread spent at various task.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

InnoDB Transactions Information

The InnoDB Transactions Information graph shows details about the recent transactions. Transaction IDs Assigned represents the total number of transactions initiated by InnoDB. RW Transaction Commits are the number of transactions not read-only. Insert-Update Transactions Commits are transactions on the Undo entries. Non Locking RO Transaction Commits are transactions commit from select statement in auto-commit mode or transactions explicitly started with “start transaction read only”.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

InnoDB Undo Space Usage

The InnoDB Undo Space Usage graph shows the amount of space used by the Undo segment. If the amount of space grows too much, look for long running transactions holding read views opened in the InnoDB status.

View all metrics of *MySQL InnoDB Metrics (Advanced) Dashboard*

InnoDB Activity

The InnoDB Activity graph shows a measure of the activity of the InnoDB threads.

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

InnoDB Contention - OS Waits

The InnoDB Contention - OS Waits graph shows the number of time an OS wait operation was required while waiting to get the lock. This happens once the spin rounds are exhausted.

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

InnoDB Contention - Spin Rounds

The InnoDB Contention - Spin Rounds metric shows the number of spin rounds executed in order to get a lock. A spin round is a fast retry to get the lock in a loop.

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

InnoDB Group Commit Batch Size

The InnoDB Group Commit Batch Size metric shows how many bytes were written to the InnoDB log files per attempt to write. If many threads are committing at the same time, one of them will write the log entries of all the waiting threads and flush the file. Such process reduces the number of disk operations needed and enlarge the batch size.

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

InnoDB Purge Throttling

The InnoDB Purge Throttling graph shows the evolution of the purge lag and the max purge lag currently set. Under heavy write load, the purge operation may start to lag behind and when the max purge lag is reached, a delay, proportional to the value defined by `innodb_max_purge_lag_delay` (in microseconds) is added to all update, insert and delete statements. This helps prevent flushing stalls.

https://dev.mysql.com/doc/refman/5.6/en/innodb-parameters.html#sysvar_innodb_max_purge_lag

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

InnoDB AHI Usage

The InnoDB AHI Usage graph shows the search operations on the InnoDB adaptive hash index and its efficiency. The adaptive hash index is a search hash designed to speed access to InnoDB pages in memory. If the Hit Ratio is small, the working data set is larger than the buffer pool, the AHI should likely be disabled.

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

InnoDB AHI Maintenance

The InnoDB AHI Maintenance graph shows the maintenance operation of the InnoDB adaptive hash index. The adaptive hash index is a search hash to speed access to InnoDB pages in memory. A constant high number of rows/pages added and removed can be an indication of an ineffective AHI.

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

InnoDB Online DDL

The InnoDB Online DDL graph shows the state of the online DDL (alter table) operations in InnoDB. The progress metric is estimate of the percentage of the rows processed by the online DDL.

Note: Currently available only on MariaDB Server

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

InnoDB Defragmentation

The InnoDB Defragmentation graph shows the status information related to the InnoDB online defragmentation feature of MariaDB for the optimize table command. To enable this feature, the variable `innodb-defragment` must be set to `1` in the configuration file.

View all metrics of [MySQL InnoDB Metrics \(Advanced\) Dashboard](#)

Note: Currently available only on MariaDB Server.

MySQL MyISAM Aria Metrics Dashboard

The MySQL MyISAM Aria Metrics dashboard describes the specific features of MariaDB MySQL server: [Aria storage engine](#), [Online DDL \(online alter table\)](#),

and [InnoDB defragmentation patch](#). This dashboard contains the following metrics:

- [Aria Storage Engine](#)
- [Aria Pagecache Reads/Writes](#)
- [Aria Pagecache Blocks](#)
- [Aria Transactions Log Syncs](#)

Aria Storage Engine

Aria storage is specific for MariaDB Server. Aria has most of the same variables that MyISAM has, but with an Aria prefix. If you use Aria instead of MyISAM, then you should make `key_buffer_size` smaller and `aria-pagecache-buffer-size` bigger.

Aria Pagecache Reads/Writes

This graph is similar to InnoDB buffer pool reads and writes. `aria-pagecache-buffer-size` is the main cache for aria storage engine. If you see high reads and writes (physical IO), i.e. reads is close to read requests or writes are close to write requests you may need to increase the `aria-pagecache-buffer-size` (you may need to decrease other buffers: `key_buffer_size`, `innodb_buffer_pool_size` etc)

Aria Pagecache Blocks

This graphs shows the utilization for the aria pagecache. This is similar to InnoDB buffer pool graph. If you see all blocks are used you may consider increasing `aria-pagecache-buffer-size` (you may need to decrease other buffers: `key_buffer_size`, `innodb_buffer_pool_size` etc)

Aria Transactions Log Syncs

This metric is similar to InnoDB log file syncs. If you see lots of log syncs and want to relax the durability settings you can change (in seconds) from 30 (default) to a higher number. It is good to look at the disk IO dashboard as well.

See also:

List of Aria system variables

<https://mariadb.com/kb/en/library/aria-system-variables/>

MySQL MyRocks Metrics Dashboard

The **MyRocks** storage engine developed by Facebook based on the RocksDB storage engine is applicable to systems which primarily interact with the database by writing data to it rather than reading from it. RocksDB also features a good level of compression, higher than that of the InnoDB storage engine, which makes it especially valuable when optimizing the usage of hard drives.

PMM collects statistics on the MyRocks storage engine for MySQL in the Metrics Monitor information for this dashboard comes from the *Information Schema* tables.

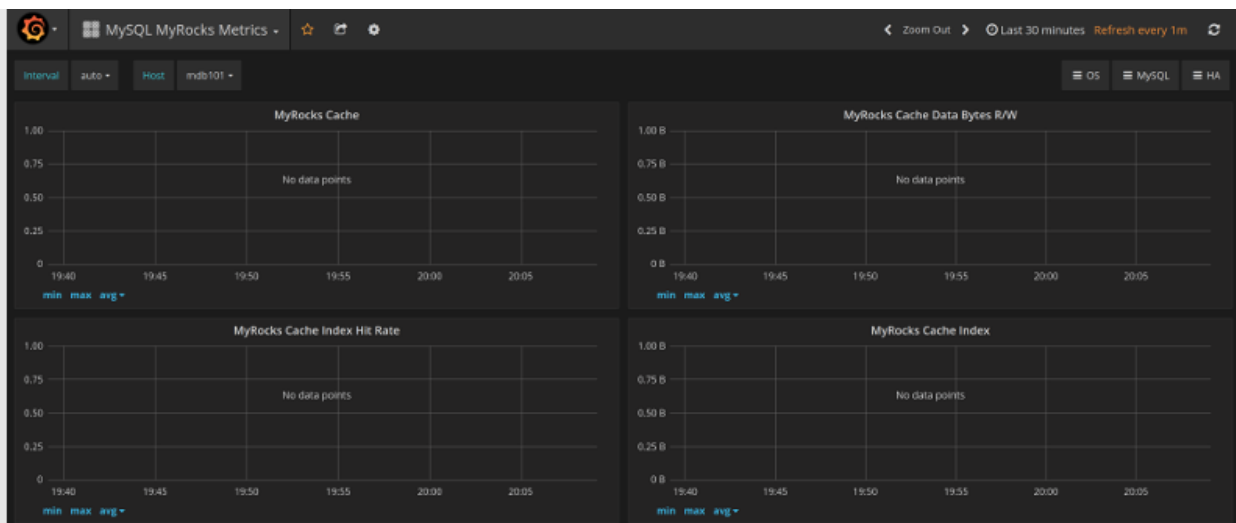


Fig. 11.2: The MySQL MyRocks metrics dashboard

See also:

Information schema <https://github.com/facebook/mysql-5.6/wiki/MyRocks-Information-Schema>

Metrics

- MyRocks cache

- MyRocks cache data bytes R/W
- MyRocks cache index hit rate
- MyRocks cache index
- MyRocks cache filter hit rate
- MyRocks cache filter
- MyRocks cache data bytes inserted
- MyRocks bloom filter
- MyRocks memtable
- MyRocks memtable size
- MyRocks number of keys
- MyRocks cache L0/L1
- MyRocks number of DB ops
- MyRocks R/W
- MyRocks bytes read by iterations
- MyRocks write ops
- MyRocks WAL
- MyRocks number reseek in iterations
- RocksDB row operations
- MyRocks file operations
- RocksDB stalls
- RocksDB stops/slowdowns

MySQL Overview

This dashboard provides basic information about MySQL hosts.

- MySQL Uptime
- Current QPS
- InnoDB Buffer Pool Size
- Buffer Pool Size % of Total RAM
- MySQL Connections
- MySQL Active Threads
- MySQL Questions
- MySQL Thread Cache
- MySQL Select Types
- MySQL Sorts
- MySQL Slow Queries
- Aborted Connections
- Table Locks
- MySQL Network Traffic
- MySQL Network Usage Hourly
- MySQL Internal Memory Overview
- Top Command Counters and Top Command Counters Hourly

- MySQL Handlers
- MySQL Query Cache Memory and MySQL Query Cache Activity
- MySQL Open Tables, MySQL Table Open Cache Status, and MySQL Table Definition Cache

MySQL Uptime

The amount of time since the MySQL server process was started.

View all metrics of *MySQL Overview*

Current QPS

Based on the queries reported by MySQL's `SHOW STATUS` command, this metric shows the number of queries executed by the server during the last second. This metric includes statements executed within stored programs.

This variable does not include the following commands:

- `COM_PING`
- `COM_STATISTICS`

View all metrics of *MySQL Overview*

See also:

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

InnoDB Buffer Pool Size

Absolute value of the InnoDB buffer pool used for caching data and indexes in memory.

The goal is to keep the working set in memory. In most cases, this should be between 60%-90% of available memory on a dedicated database host, but depends on many factors.

View all metrics of *MySQL Overview*

Buffer Pool Size % of Total RAM

The ratio between InnoDB buffer pool size and total memory. In most cases, the InnoDB buffer pool should be between 60% and 90% of available memory on a dedicated database host, but it depends on many factors.

View all metrics of *MySQL Overview*

MySQL Connections

Max Connections The maximum permitted number of simultaneous client connections. This is the value of the `max_connections` variable.

Max Used Connections The maximum number of connections that have been in use simultaneously since the server was started.

Connections The number of connection attempts (successful or not) to the MySQL server.

View all metrics of *MySQL Overview*

See also:

MySQL Server status variables: max_connections https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html#sysvar_max_connections

MySQL Active Threads

Threads Connected The number of open connections.

Threads Running The number of threads not sleeping.

View all metrics of *MySQL Overview*

MySQL Questions

The number of queries sent to the server by clients, *excluding those executed within stored programs*.

This variable does not count the following commands:

View all metrics of *MySQL Overview*

- COM_PING
- COM_STATISTICS
- COM_STMT_PREPARE
- COM_STMT_CLOSE
- COM_STMT_RESET

View all metrics of *MySQL Overview*

MySQL Thread Cache

The `thread_cache_size` metric informs how many threads the server should cache to reuse. When a client disconnects, the client's threads are put in the cache if the cache is not full. It is autosized in MySQL 5.6.8 and above (capped to 100).

Requests for threads are satisfied by reusing threads taken from the cache if possible, and only when the cache is empty is a new thread created.

- `Threads_created`: The number of threads created to handle connections.
- `Threads_cached`: The number of threads in the thread cache.

View all metrics of *MySQL Overview*

See also:

MySQL Server status variables: thread_cache_size https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html#sysvar_thread_cache_size

MySQL Select Types

As with most relational databases, selecting based on indexes is more efficient than scanning the data of an entire table. Here, we see the counters for selects not done with indexes.

- *Select Scan* is how many queries caused full table scans, in which all the data in the table had to be read and either discarded or returned.
- *Select Range* is how many queries used a range scan, which means MySQL scanned all rows in a given range.
- *Select Full Join* is the number of joins that are not joined on an index, this is usually a huge performance hit.

View all metrics of [MySQL Overview](#)

MySQL Sorts

Due to a query's structure, order, or other requirements, MySQL sorts the rows before returning them. For example, if a table is ordered 1 to 10 but you want the results reversed, MySQL then has to sort the rows to return 10 to 1.

This graph also shows when sorts had to scan a whole table or a given range of a table in order to return the results and which could not have been sorted via an index.

View all metrics of [MySQL Overview](#)

MySQL Slow Queries

Slow queries are defined as queries being slower than the `long_query_time` setting. For example, if you have `long_query_time` set to **3**, all queries that take longer than **3** seconds to complete will show on this graph.

View all metrics of [MySQL Overview](#)

Aborted Connections

When a given host connects to MySQL and the connection is interrupted in the middle (for example due to bad credentials), MySQL keeps that info in a system table (since 5.6 this table is exposed in `performance_schema`).

If the amount of failed requests without a successful connection reaches the value of `max_connect_errors`, **mysqld** assumes that something is wrong and blocks the host from further connections.

To allow connections from that host again, you need to issue the `FLUSH HOSTS` statement.

View all metrics of [MySQL Overview](#)

Table Locks

MySQL takes a number of different locks for varying reasons. In this graph we see how many Table level locks MySQL has requested from the storage engine. In the case of InnoDB, many times the locks could actually be row locks as it only takes table level locks in a few specific cases.

It is most useful to compare *Locks Immediate* and *Locks Waited*. If *Locks Waited* is rising, it means you have lock contention. Otherwise, *Locks Immediate* rising and falling is normal activity.

View all metrics of [MySQL Overview](#)

MySQL Network Traffic

This metric shows how much network traffic is generated by MySQL. *Outbound* is network traffic sent from MySQL and *Inbound* is the network traffic that MySQL has received.

View all metrics of [MySQL Overview](#)

MySQL Network Usage Hourly

This metric shows how much network traffic is generated by MySQL per hour. You can use the bar graph to compare data sent by MySQL and data received by MySQL.

View all metrics of [MySQL Overview](#)

MySQL Internal Memory Overview

This metric shows the various uses of memory within MySQL.

System Memory

Total Memory for the system.

InnoDB Buffer Pool Data

InnoDB maintains a storage area called the buffer pool for caching data and indexes in memory. Knowing how the InnoDB buffer pool works, and taking advantage of it to keep frequently accessed data in memory, is an important aspect of MySQL tuning.

TokuDB Cache Size

Similar in function to the InnoDB Buffer Pool, TokuDB will allocate 50% of the installed RAM for its own cache. While this is optimal in most situations, there are cases where it may lead to memory over allocation. If the system tries to allocate more memory than is available, the machine will begin swapping and run much slower than normal.

Key Buffer Size

Index blocks for MyISAM tables are buffered and are shared by all threads. *key_buffer_size* is the size of the buffer used for index blocks. The key buffer is also known as the *key cache*.

Adaptive Hash Index Size

The InnoDB storage engine has a special feature called adaptive hash indexes. When InnoDB notices that some index values are being accessed very frequently, it builds a hash index for them in memory on top of B-Tree indexes. This allows for very fast hashed lookups.

Query Cache Size

The query cache stores the text of a `SELECT` statement together with the corresponding result that was sent to the client. The query cache has huge scalability problems in that only one thread can do an operation in the query cache at the same time. This serialization is true for `SELECT` and also for `INSERT`, `UPDATE`, and `DELETE`. This also means that the larger the *query_cache_size* is set to, the slower those operations become.

InnoDB Dictionary Size

The data dictionary is InnoDB internal catalog of tables. InnoDB stores the data dictionary on disk, and loads entries into memory while the server is running. This is somewhat analogous to table cache of MySQL, but instead of operating at the server level, it is internal to the InnoDB storage engine.

InnoDB Log Buffer Size

The MySQL InnoDB log buffer allows transactions to run without having to write the log to disk before the transactions commit. The size of this buffer is configured with the `innodb_log_buffer_size` variable.

View all metrics of *MySQL Overview*

Top Command Counters and Top Command Counters Hourly

See https://dev.mysql.com/doc/refman/5.7/en/server-status-variables.html#statvar_Com_xxx

View all metrics of *MySQL Overview*

MySQL Handlers

Handler statistics are internal statistics on how MySQL is selecting, updating, inserting, and modifying rows, tables, and indexes.

This is in fact the layer between the Storage Engine and MySQL.

- `read_rnd_next` is incremented when the server performs a full table scan and this is a counter you don't really want to see with a high value.
- `read_key` is incremented when a read is done with an index.
- `read_next` is incremented when the storage engine is asked to 'read the next index entry'. A high value means a lot of index scans are being done.

View all metrics of *MySQL Overview*

MySQL Query Cache Memory and MySQL Query Cache Activity

The query cache has huge scalability problems in that only one thread can do an operation in the query cache at the same time. This serialization is true not only for `SELECT`, but also for `INSERT`, `UPDATE`, and `DELETE`.

This also means that the larger the `query_cache_size` is set to, the slower those operations become. In concurrent environments, the MySQL Query Cache quickly becomes a contention point, decreasing performance. MariaDB and Amazon Aurora have done work to try and eliminate the query cache contention in their flavors of MySQL, while MySQL 8.0 has eliminated the query cache feature.

The recommended settings for most environments is to set:

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

Note: While you can dynamically change these values, to completely remove the contention point you have to restart the database.

View all metrics of *MySQL Overview*

MySQL Open Tables, MySQL Table Open Cache Status, and MySQL Table Definition Cache

The recommendation is to set the `table_open_cache_instances` to a loose correlation to virtual CPUs, keeping in mind that more instances means the cache is split more times. If you have a cache set to 500 but it has 10 instances, each cache will only have 50 cached.

The `table_definition_cache` and `table_open_cache` can be left as default as they are autosized MySQL 5.6 and above (do not set them to any value).

View all metrics of *MySQL Overview*

See also:

Configuring MySQL for PMM *Settings for Dashboards*

MySQL Documentation: InnoDB buffer pool <https://dev.mysql.com/doc/refman/5.7/en/innodb-buffer-pool.html>

Percona Server Documentation: Running TokuDB in Production https://www.percona.com/doc/percona-server/LATEST/tokudb/tokudb_quickstart.html#considerations-to-run-tokudb-in-production

Blog post: Adaptive Hash Index in InnoDB <https://www.percona.com/blog/2016/04/12/is-adaptive-hash-index-in-innodb-right-for-my-workload/>

MySQL Server System Variables: `key_buffer_size` https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_key_buffer_size

MySQL Server System Variables: `table_open_cache` http://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html#sysvar_table_open_cache

MySQL Performance Schema Dashboard

The MySQL *Performance Schema* dashboard helps determine the efficiency of communicating with *Performance Schema*. This dashboard contains the following metrics:

- *Performance Schema* file IO (events)
- *Performance Schema* file IO (load)
- *Performance Schema* file IO (Bytes)
- *Performance Schema* waits (events)
- *Performance Schema* waits (load)
- Index access operations (load)
- Table access operations (load)
- *Performance Schema* SQL and external locks (events)
- *Performance Schema* SQL and external locks (seconds)

See also:

MySQL Documentation: *Performance Schema*

<https://dev.mysql.com/doc/refman/5.7/en/performance-schema.html>

Performance Schema Wait Event Analysis Dashboard

This dashboard helps to analyse *Performance Schema* wait events. It plots the following metrics for the chosen (one or more) wait events:

- *Count - Performance Schema Waits*
- *Load - Performance Schema Waits*
- *Avg Wait Time - Performance Schema Waits*

See also:

MySQL Documentation: *Performance Schema*

<https://dev.mysql.com/doc/refman/5.7/en/performance-schema.html>

MySQL Query Response Time

This dashboard provides information about query response time distribution.

- Average Query Response Time
- Query Response Time Distribution
- Average Query Response Time (Read/Write Split)
- Read Query Response Time Distribution
- Write Query Response Time Distribution

Average Query Response Time

The Average Query Response Time graph shows information collected using the Response Time Distribution plugin sourced from table `INFORMATION_SCHEMA.QUERY_RESPONSE_TIME`. It computes this value across all queries by taking the sum of seconds divided by the count of seconds.

View all metrics of *MySQL Query Response Time*

See also:

Percona Server Documentation: QUERY_RESPONSE_TIME table https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME

Query Response Time Distribution

Shows how many fast, neutral, and slow queries are executed per second.

Query response time counts (operations) are grouped into three buckets:

- 100ms - 1s
- 1s - 10s
- > 10s

View all metrics of *MySQL Query Response Time*

Average Query Response Time (Read/Write Split)

Available only in Percona Server for MySQL, this metric provides visibility of the split of READ vs WRITE query response time.

View all metrics of *MySQL Query Response Time*

See also:

Percona Server Documentation: Logging queries in separate READ and WRITE tables https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#logging-the-queries-in-separate-read-and-write-tables

Percona Server Documentation: QUERY_RESPONSE_TIME_READ https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME_READ

Percona Server Documentation: QUERY_RESPONSE_TIME_WRITE https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME_WRITE

Read Query Response Time Distribution

Available only in Percona Server for MySQL, illustrates READ query response time counts (operations) grouped into three buckets:

- 100ms - 1s
- 1s - 10s
- > 10s

View all metrics of *MySQL Query Response Time*

See also:

Percona Server Documentation: QUERY_RESPONSE_TIME_READ https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME_READ

Write Query Response Time Distribution

Available only in Percona Server for MySQL, illustrates WRITE query response time counts (operations) grouped into three buckets:

- 100ms - 1s
- 1s - 10s
- > 10s

View all metrics of *MySQL Query Response Time*

See also:

Configuring MySQL for PMM *Percona Server Query Response Time Distribution*

Percona Server Documentation: QUERY_RESPONSE_TIME_WRITE https://www.percona.com/doc/percona-server/5.7/diagnostics/response_time_distribution.html#QUERY_RESPONSE_TIME_WRITE

MySQL Replication

- *IO Thread Running*
- *SQL Thread Running*
- *Replication Error No*
- *Read only*
- *MySQL Replication Delay*
- *Binlog Size*
- *Binlog Data Written Hourly*
- *Binlog Count*
- *Binlogs Created Hourly*

- *Relay Log Space*
- *Relay Log Written Hourly*

IO Thread Running

This metric shows if the IO Thread is running or not. It only applies to a slave host.

SQL Thread is a process that runs on a slave host in the replication environment. It reads the events from the local relay log file and applies them to the slave server.

Depending on the format of the binary log it can read query statements in plain text and re-execute them or it can read raw data and apply them to the local host.

Possible values

Yes

The thread is running and is connected to a replication master

No

The thread is not running because it is not launched yet or because an error has occurred connecting to the master host

Connecting

The thread is running but is not connected to a replication master

No value

The host is not configured to be a replication slave

IO Thread Running is one of the parameters that the command `SHOW SLAVE STATUS` returns.

See also:

MySQL Documentation

- [Replication](#)
- [SHOW SLAVE STATUS Syntax](#)
- [IO Thread states](#)

View all metrics of *MySQL Replication*

SQL Thread Running

This metric shows if the SQL thread is running or not. It only applies to a slave host.

SQL Thread is a process that runs on a slave host in the replication environment. It reads the events from the local relay log file and applies them to the slave server.

Depending on the format of the binary log it can read query statements in plain text and re-execute them or it can read raw data and apply them to the local host.

Possible values

Yes

SQL Thread is running and is applying events from the realy log to the local slave host

No

SQL Thread is not running because it is not launched yet or because of an error occurred while applying an event to the local slave host

See also:

MySQL Documentation:

- [Replication](#)
- [SHOW SLAVE STATUS Syntax](#)
- [SQL Thread states](#)

View all metrics of *MySQL Replication*

Replication Error No

This metric shows the number of the last error in the SQL Thread encountered which caused replication to stop.

One of the more common errors is *Error: 1022 Duplicate Key Entry*. In such a case replication is attempting to update a row that already exists on the slave. The SQL Thread will stop replication in order to avoid data corruption.

See also:

MySQL Documentation:

[A complete list of error codes](#)

View all metrics of *MySQL Replication*

Read only

This metric indicates whether the host is configured to be in *Read Only* mode or not.

Possible values

Yes

The slave host permits no client updates except from users who have the SUPER privilege or the REPLICATION SLAVE privilege.

This kind of configuration is typically used for slave hosts in a replication environment to avoid a user can inadvertently or voluntarily modify data causing inconsistencies and stopping the replication process.

No

The slave host is not configured in *Read Only* mode.

See also:

MySQL Documentation:

[Replication](#)

[View all metrics of *MySQL Replication*](#)

MySQL Replication Delay

This metric shows the number of seconds the slave host is delayed in replication applying events compared to when the Master host applied them, denoted by the `Seconds_Behind_Master` value, and only applies to a slave host.

Since the replication process applies the data modifications on the slave asynchronously, it could happen that the slave replicates events after some time. The main reasons are:

- **Network round trip time** - high latency links will lead to non-zero replication lag values.
- **Single threaded nature of replication channels** - master servers have the advantage of applying changes in parallel, whereas slave ones are only able to apply changes in serial, thus limiting their throughput. In some cases Group Commit can help but is not always applicable.
- **High number of changed rows or computationally expensive SQL** - depending on the replication format (ROW vs STATEMENT), significant changes to the database through high volume of rows modified, or expensive CPU will all contribute to slave servers lagging behind the master.

Generally adding more CPU or Disk resources can alleviate replication lag issues, up to a point.

Ideally a value of 0 is desired, but be aware that `Seconds_Behind_Master` is an integer value and thus rounding is a factor. If you desire greater precision, consider the Percona Toolkit tool `pt-heartbeat`, as this graph will automatically take into account this tool and then show you greater resolution in the milliseconds.

See also:

Related metrics:

- [Relay Log Space](#)

MySQL Documentation

- [SHOW SLAVE STATUS Syntax](#)
- [Improving replication performance](#)
- [Replication Slave Options and Variables](#)

[View all metrics of *MySQL Replication*](#)

Binlog Size

This metric shows the overall size of the binary log files, which can exist on both master and slave servers. The binary log (also known as the binlog) contains events that describe database changes: `CREATE TABLE`, `ALTER TABLE`, updates, inserts, deletes and other statements or database changes. The binlog is the file that is read by slaves via their IO Thread process in order to replicate database changes modification on the data and on the table structures. There can be more than one binlog file present depending on the binlog rotation policy adopted (for example using the configuration variables `max_binlog_size` and `expire_logs_days`).

There can be more binlog files depending on the rotation policy adopted (for example using the configuration variables `max_binlog_size` and `expire_logs_days`) or even because of server reboots.

When planning the disk space, take care of the overall dimension of binlog files and adopt a good rotation policy or think about having a separate mount point or disk to store the binlog data.

See also:

MySQL Documentation:

- [The binary log](#)

- [Configuring replication](#)

View all metrics of *MySQL Replication*

Binlog Data Written Hourly

This metric shows the amount of data written hourly to the binlog files during the last 24 hours. This metric can give you an idea of how big is your application in terms of data writes (creation, modification, deletion).

View all metrics of *MySQL Replication*

Binlog Count

This metric shows the overall count of binary log files, on both master and slave servers.

There can be more binlog files depending on the rotation policy adopted (for example using the configuration variables `max_binlog_size` and `expire_logs_days`) or even because of server reboots.

When planning the disk space, take care of the overall dimension of binlog files and adopt a good rotation policy or think about having a separate mount point or disk to store the binlog data.

See also:

MySQL Documentation:

- [The binary log](#)
- [Configuring replication](#)

View all metrics of *MySQL Replication*

Binlogs Created Hourly

This metric shows the number of binlog files created hourly during the last 24 hours.

There can be more binlog files depending on the rotation policy adopted (for example using the configuration variables `max_binlog_size` and `expire_logs_days`) or even because of server reboots.

When planning the disk space, take care of the overall dimension of binlog files and adopt a good rotation policy or think about having a separate mount point or disk to store the binlog data.

View all metrics of *MySQL Replication*

Relay Log Space

This metric shows the overall size of the relay log files. It only applies to a slave host.

The relay log consists of a set of numbered files containing the events to be executed on the slave host in order to replicate database changes.

The relay log has the same format as the binlog.

There can be multiple relay log files depending on the rotation policy adopted (using the configuration variable `max_relay_log_size`).

As soon as the SQL thread completes to execute all events in the relay log file, the file is deleted.

If this metric contains a high value, the variable `max_relay_log_file` is high too. Generally, this not a serious issue. If the value of this metric is constantly increased, the slave is delaying too much in applying the events.

Treat this metric in the same way as the *MySQL Replication Delay* metric.

See also:

MySQL Documentation:

- [The Slave Relay Log](#)

View all metrics of *MySQL Replication*

Relay Log Written Hourly

This metric shows the amount of data written hourly into relay log files during the last 24 hours.

View all metrics of *MySQL Replication*

MySQL Table Statistics

This dashboard presents various data related to MySQL tables.

- *Largest Tables*
- *Pie*
- *Table Activity*
- *Rows read*
- *Rows Changed*
- *Auto Increment Usage*

Largest Tables

Largest Tables by Row Count The estimated number of rows in the table from `information_schema.tables`.

Largest Tables by Size The size of the table components from `information_schema.tables`.

Pie

Total Database Size The total size of the database: as data + index size, so freeble one.

Most Fragmented Tables by Freeable Size The list of 5 most fragmented tables ordered by their freeable size

Table Activity

The next two graphs are available only for [Percona Server](#) and [MariaDB](#) and require `userstat` variable turned on.

Rows read

The number of rows read from the table, shown for the top 5 tables.

Rows Changed

The number of rows changed in the table, shown for the top 5 tables.

Auto Increment Usage

The current value of an `auto_increment` column from `information_schema`, shown for the top 10 tables.

MySQL User Statistics

This dashboard presents various data related to MySQL users.

Note: This dashboard requires Percona Server for MySQL 5.1+ or MariaDB 10.1/10.2 with XtraDB. Also `userstat` should be enabled, for example with the `SET GLOBAL userstat=1` statement. See [Configuring MySQL for Best Results](#) for further instructions.

Data is displayed for the 5 top users.

Top Users by Connections Created The number of times user's connections connected using SSL to the server.

Top Users by Traffic The number of bytes sent to the user's connections.

Top Users by Rows Fetched/Read The number of rows fetched by the user's connections.

Top Users by Rows Updated The number of rows updated by the user's connections.

Top Users by Busy Time The cumulative number of seconds there was activity on connections from the user.

Top Users by CPU Time The cumulative CPU time elapsed, in seconds, while servicing connections of the user.

MongoDB Dashboards

MongoDB Cluster Summary

The MongoDB Cluster Summary dashboard shows the statistics on the selected MongoDB cluster. Namely, it reports the following information:

- Unsharded DBs
- Sharded DBs
- Sharded collections
- Shards
- Chunks
- Balancer enabled
- Chunks balanced
- Mongos operations
- Mongos connections
- Mongos cursors
- Chunk split events
- Change log events
- Operations per shard
- Chunks by shard
- Connections per shard
- Cursors per shard

- Replication lag by set
- Oplog range by set
- Shard elections
- Collection lock time

MongoDB inMemory Dashboard

The MongoDB inMemory dashboard shows statistics on the In-Memory storage engine for the selected MongoDB instances. This dashboard contains the following metrics:

- InMemory data size
- InMemory max data size
- InMemory available
- InMemory dirty pages
- InMemory transactions
- InMemory capacity
- InMemory sessions
- InMemory pages
- InMemory concurrency tickets
- Queued operations
- Document changes
- InMemory cache eviction
- Scanned and moved objects
- Page faults

MongoDB MMAPv1 Dashboard

The MongoDB MMAPv1 dashboard contains metrics that describe the performance of the MMAPv1 storage engine for MongoDB. This dashboard includes the following metrics:

- MMAPv1 lock wait ratio
- MMAPv1 write lock time
- Memory cached
- Memory available
- Document activity
- MMAPv1 lock ratios
- MMAPv1 lock wait time
- MMAPv1 page faults
- MMAPv1 journal write activity
- MMAPv1 journal commit activity
- MMAPv1 journaling time
- MMAPv1 journaling time - 99th percentile
- MMAPv1 background flushing time
- Queued operations
- Client operations
- Scanned and moved objects
- MMAPv1 memory usage
- MMAPv1 memory dirty pages

MongoDB Overview

This dashboard provides basic information about MongoDB instances.

Command Operations

Shows how many times a command is executed per second on average during the selected interval.

Look for peaks and drops and correlate them with other graphs.

View all metrics of [MongoDB Overview](#)

Connections

Keep in mind the hard limit on the maximum number of connections set by your distribution.

Anything over 5,000 should be a concern, because the application may not close connections correctly.

View all metrics of [MongoDB Overview](#)

Cursors

Helps identify why connections are increasing. Shows active cursors compared to cursors being automatically killed after 10 minutes due to an application not closing the connection.

View all metrics of [MongoDB Overview](#)

Document Operations

When used in combination with **Command Operations**, this graph can help identify *write amplification*. For example, when one `insert` or `update` command actually inserts or updates hundreds, thousands, or even millions of documents.

View all metrics of [MongoDB Overview](#)

Queued Operations

Any number of queued operations for long periods of time is an indication of possible issues. Find the cause and fix it before requests get stuck in the queue.

View all metrics of [MongoDB Overview](#)

getLastError Write Time, getLastError Write Operations

This is useful for write-heavy workloads to understand how long it takes to verify writes and how many concurrent writes are occurring.

View all metrics of [MongoDB Overview](#)

Asserts

Asserts are not important by themselves, but you can correlate spikes with other graphs.

View all metrics of [MongoDB Overview](#)

Memory Faults

Memory faults indicate that requests are processed from disk either because an index is missing or there is not enough memory for the data set. Consider increasing memory or sharding out.

View all metrics of *MongoDB Overview*

MongoDB ReplSet

This dashboard provides information about replica sets and their members.

- [Replication Operations](#)
- [ReplSet State](#)
- [ReplSet Members](#)
- [ReplSet Last Election](#)
- [ReplSet Lag](#)
- [Storage Engine](#)
- [Oplog Insert Time](#)
- [Oplog Recovery Window](#)
- [Replication Lag](#)
- [Elections](#)
- [Member State Uptime](#)
- [Max Heartbeat Time](#)
- [Max Member Ping Time](#)

Replication Operations

This metric provides an overview of database replication operations by type and makes it possible to analyze the load on the replica in more granular manner. These values only appear when the current host has replication enabled.

ReplSet State

This metric shows the role of the selected member instance (PRIMARY or SECONDARY).

View all metrics of *MongoDB ReplSet*

ReplSet Members

This metric the number of members in the replica set.

View all metrics of *MongoDB ReplSet*

ReplSet Last Election

This metric how long ago the last election occurred.

View all metrics of *MongoDB ReplSet*

ReplSet Lag

This metric shows the current replication lag for the selected member.

View all metrics of *MongoDB ReplSet*

Storage Engine

This metric shows the storage engine used on the instance

View all metrics of *MongoDB ReplSet*

Oplog Insert Time

This metric shows how long it takes to write to the oplog. Without it the write will not be successful.

This is more useful in mixed replica sets (where instances run different storage engines).

View all metrics of *MongoDB ReplSet*

Oplog Recovery Window

This metric shows the time range in the oplog and the oldest backed up operation.

For example, if you take backups every 24 hours, each one should contain at least 36 hours of backed up operations, giving you 12 hours of restore window.

View all metrics of *MongoDB ReplSet*

Replication Lag

This metric shows the delay between an operation occurring on the primary and that same operation getting applied on the selected member

View all metrics of *MongoDB ReplSet*

Elections

Elections happen when a primary becomes unavailable. Look at this graph over longer periods (weeks or months) to determine patterns and correlate elections with other events.

View all metrics of *MongoDB ReplSet*

Member State Uptime

This metric shows how long various members were in PRIMARY and SECONDARY roles.

View all metrics of *MongoDB ReplSet*

Max Heartbeat Time

This metric shows the heartbeat return times sent by the current member to other members in the replica set.

Long heartbeat times can indicate network issues or that the server is too busy.

View all metrics of *MongoDB ReplSet*

Max Member Ping Time

This metric can show a correlation with the replication lag value.

View all metrics of *MongoDB ReplSet*

MongoDB RocksDB Dashboard

The MongoDB RocksDB dashboard contains metrics that describe the performance of the RocksDB storage engine for the selected MongoDB host instance. This dashboard contains the following metrics:

- RocksDB Memtable used
- RocksDB block cache used
- Memory cached
- Document activity
- RocksDB cache usage
- RocksDB Memtable entries
- RocksDB block cache hit ratio
- RocksDB write activity
- RocksDB read activity
- RocksDB Level0 read latency
- RocksDB LevelN read average latency
- RocksDB LevelN 99th percentile read latency
- RocksDB LevelN maximum read latency
- RocksDB compaction time
- RocksDB compaction write amplification
- RocksDB compaction read rate
- RocksDB compaction write rate
- RocksDB compaction key rate
- RocksDB compaction threads
- RocksDB compaction level files
- RocksDB compaction level size
- RocksDB write ahead log rate
- RocksDB write ahead log sync size
- RocksDB flush rate
- RocksDB pending operations
- RocksDB stall time
- RocksDB stalls
- Client operations
- Queued operations
- Scanned and moved objects
- Page faults

MongoDB WiredTiger Dashboard

The MongoDB WiredTiger dashboard contains statistics on the WiredTiger storage engine for the selected MongoDB host. This dashboard contains the following statistics:

- WiredTiger cache usage
- WiredTiger max cache size
- Memory cached
- Memory available
- WiredTiger transactions
- WiredTiger cache activity
- WiredTiger block activity
- WiredTiger sessions
- WiredTiger concurrency tickets available
- Queued operations
- WiredTiger checkpoint time
- WiredTiger cache eviction
- WiredTiger cache capacity
- WiredTiger cache pages
- WiredTiger log operations
- WiredTiger log records
- Document changes
- Scanned and moved objects
- Page faults

PostgreSQL Dashboards

PostgreSQL Overview

This dashboard provides basic information about PostgreSQL hosts.

- Connected
- Version
- Shared Buffers
- Disk-Page Buffers
- *Memory Size for each Sort*
- Disk Cache Size
- Autovacuum
- PostgreSQL Connections
- PostgreSQL Tuples
- PostgreSQL Transactions
- Temp Files
- Conflicts and Locks
- Buffers and Blocks Operations

- Canceled Queries
- Cache Hit Ratio
- Checkpoint Stats
- PostgreSQL Settings
- System Summary

Connected

Reports whether PMM Server can connect to the PostgreSQL instance.

View all metrics of *PostgreSQL Overview*

Version

The version of the PostgreSQL instance.

View all metrics of *PostgreSQL Overview*

Shared Buffers

Defines the amount of memory the database server uses for shared memory buffers. Default is 128MB. Guidance on tuning is 25% of RAM, but generally doesn't exceed 40%.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: `shared_buffers` <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-SHARED-BUFFERS>

Disk-Page Buffers

The setting `wal_buffers` defines how much memory is used for caching the write-ahead log entries. Generally this value is small (3% of `shared_buffers` value), but it may need to be modified for heavily loaded servers.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: `wal_buffers` <https://www.postgresql.org/docs/current/static/runtime-config-wal.html#GUC-WAL-BUFFERS>

PostgreSQL Server status variables: `shared_buffers` <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-SHARED-BUFFERS>

Memory Size for each Sort

The parameter `work_mem` defines the amount of memory assigned for internal sort operations and hash tables before writing to temporary disk files. The default is 4MB.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: work_mem <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-WORK-MEM>

Disk Cache Size

PostgreSQL's `effective_cache_size` variable tunes how much RAM you expect to be available for disk caching. Generally adding Linux `free+cached` will give you a good idea. This value is used by the query planner whether plans will fit in memory, and when defined too low, can lead to some plans rejecting certain indexes.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: effective_cache_size <https://www.postgresql.org/docs/current/static/runtime-config-query.html#GUC-EFFECTIVE-CACHE-SIZE>

Autovacuum

Whether autovacuum process is enabled or not. Generally the solution is to vacuum more often, not less.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: autovacuum <https://www.postgresql.org/docs/current/static/routine-vacuuming.html#AUTOVACUUM>

PostgreSQL Connections

Max Connections The maximum number of client connections allowed. Change this value with care as there are some memory resources that are allocated on a per-client basis, so setting `max_connections` higher will generally increase overall PostgreSQL memory usage.

Connections The number of connection attempts (successful or not) to the PostgreSQL server.

Active Connections The number of open connections to the PostgreSQL server.

View all metrics of *PostgreSQL Overview*

See also:

PostgreSQL Server status variables: max_connections <https://www.postgresql.org/docs/current/static/runtime-config-connection.html#GUC-MAX-CONNECTIONS>

PostgreSQL Tuples

Tuples The total number of rows processed by PostgreSQL server: fetched, returned, inserted, updated, and deleted.

Read Tuple Activity The number of rows read from the database: as returned so fetched ones.

Tuples Changed per 5min The number of rows changed in the last 5 minutes: inserted, updated, and deleted ones.

View all metrics of *PostgreSQL Overview*

PostgreSQL Transactions

Transactions The total number of transactions that have been either committed or rolled back.

Duration of Transactions Maximum duration in seconds any active transaction has been running.

View all metrics of *PostgreSQL Overview*

Temp Files

Number of Temp Files The number of temporary files created by queries.

Size of Temp files The total amount of data written to temporary files by queries in bytes.

Note: All temporary files are taken into account by these two gauges, regardless of why the temporary file was created (e.g., sorting or hashing), and regardless of the `log_temp_files` setting.

View all metrics of *PostgreSQL Overview*

Conflicts and Locks

Conflicts/Deadlocks The number of queries canceled due to conflicts with recovery in the database (due to dropped tablespaces, lock timeouts, old snapshots, pinned buffers, or deadlocks).

Number of Locks The number of deadlocks detected by PostgreSQL.

View all metrics of *PostgreSQL Overview*

Buffers and Blocks Operations

Operations with Blocks The time spent reading and writing data file blocks by backends, in milliseconds.

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

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

Buffers The number of buffers allocated by PostgreSQL.

View all metrics of *PostgreSQL Overview*

Canceled Queries

The number of queries that have been canceled due to dropped tablespaces, lock timeouts, old snapshots, pinned buffers, and deadlocks.

Note: Data shown by this gauge are based on the `pg_stat_database_conflicts` view.

View all metrics of *PostgreSQL Overview*

Cache Hit Ratio

The number of times disk blocks were found already in the buffer cache, so that a read was not necessary.

Note: This only includes hits in the PostgreSQL buffer cache, not the operating system's file system cache.

View all metrics of *PostgreSQL Overview*

Checkpoint Stats

The total amount of time that has been spent in the portion of checkpoint processing where files are either written or synchronized to disk, in milliseconds.

View all metrics of *PostgreSQL Overview*

PostgreSQL Settings

The list of all settings of the PostgreSQL server.

View all metrics of *PostgreSQL Overview*

System Summary

This section contains the following system parameters of the PostgreSQL server: CPU Usage, CPU Saturation and Max Core Usage, Disk I/O Activity, and Network Traffic.

View all metrics of *PostgreSQL Overview*

See also:

Configuring PostgreSQL for Monitoring *Configuring PostgreSQL for Monitoring*

PostgreSQL Server status variables: wal_buffers <https://www.postgresql.org/docs/current/static/runtime-config-wal.html#GUC-WAL-BUFFERS>

PostgreSQL Server status variables: shared_buffers <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-SHARED-BUFFERS>

PostgreSQL Server status variables: work_mem <https://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-WORK-MEM>

PostgreSQL Server status variables: effective_cache_size <https://www.postgresql.org/docs/current/static/runtime-config-query.html#GUC-EFFECTIVE-CACHE-SIZE>

PostgreSQL Server status variables: autovacuum <https://www.postgresql.org/docs/current/static/routine-vacuuming.html#AUTOVACUUM>

PostgreSQL Server status variables: max_connections <https://www.postgresql.org/docs/current/static/runtime-config-connection.html#GUC-MAX-CONNECTIONS>

HA Dashboards

PXC/Galera Cluster Overview Dashboard

- Flow Control Paused Time

- Flow Control Messages Sent
- Writeseq Inbound Traffic
- Writeseq Outbound Traffic
- Receive Queue
- Send Queue
- Transactions Received
- Transactions Replicated
- Average Incoming Transaction Size
- Average Replicated Transaction Size
- FC Trigger Low Limit
- FC Trigger High Limit
- Sequence Numbers of Transactions
- Average Galera Replication Latency
- Maximum Galera Replication Latency

Part III

Reference

PERCONA MONITORING AND MANAGEMENT RELEASE NOTES

Percona Monitoring and Management 1.17.0

Date November 20, 2018

PMM (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and *PostgreSQL* performance. You can run *PMM* in your own environment for maximum security and reliability. It provides thorough time-based analysis for *MySQL*, *MongoDB*, and *PostgreSQL* servers to ensure that your data works as efficiently as possible.

In release 1.17.0 we made 6 improvements and fixed 11 bugs.

Although we patched a bug this release related to a Grafana CVE announcement, previous releases since 1.10 (April 2018) were not vulnerable, and we encourage you to see our [Grafana CVE blog post](#) for further details.

Dashboard Improvements

We updated five Dashboards with improved Tooltips - If you haven't seen this before, hover your mouse over the `i` icon in the top left of most graph elements, and you'll see a new box appear. This box provides a brief description of what the graph displays, along with links to related documentation resources so you can learn further. We hope you find the content useful!

The Dashboards we're updating are:

1. *MySQL Amazon Aurora Metrics*
2. *MySQL MyISAM/Aria Metrics*
3. *MySQL Replication*
4. *Prometheus Exporters Overview*
5. *Trends*

We hope you enjoy this release, and we welcome your feedback via the [Percona PMM Forums](#)!

New Features and Improvements

- [PMM-3272](#): Adhere to Amazon naming standards for RDS - should be Amazon RDS
- [PMM-2081](#): Implement Tooltips for Dashboard *Prometheus_Exporters_Overview* – credits to [Corrado Pandiani](#) for texts
- [PMM-1363](#): Implement Tooltips for Dashboard *MySQL_Amazon_Aurora_Metrics* – credits to [Corrado Pandiani](#) for texts

- [PMM-1359](#): Implement Tooltips for Dashboard Trends_Dashboard – credits to [Corrado Pandiani](#) for texts
- [PMM-1357](#): Implement Tooltips for Dashboard MySQL_Replication – credits to [Corrado Pandiani](#) for texts
- [PMM-1348](#): Implement Tooltips for Dashboard MySQL_MyISAM/Aria_Metrics – credits to [Alexander Rubin](#) for texts

Fixed Bugs

- [PMM-3257](#): Grafana Security patch for CVE-2018-19039
- [PMM-3252](#): Update button in 1.16 is not visible when a newer version exists
- [PMM-3209](#): Special symbols in username or password prevent addition of Remote Instances
- [PMM-2837](#): Image Rendering Does not work due to absent Phantom.JS binary
- [PMM-2428](#): Remove Host=All on dashboards where this variable does not apply
- [PMM-2294](#): No changes in zoomed out Cluster Size Graph if the node was absent for a short time
- [PMM-2289](#): SST Time Graph based on the wrong formula
- [PMM-2192](#): Memory leak in ProxySQL_Exporter when ProxySQL is down
- [PMM-2158](#): MongoDB “Query Efficiency - Document” arithmetic appears to be incorrectly calculated
- [PMM-1837](#): System Info shows duplicate hosts
- [PMM-1805](#): Available Downtime before SST Required doesn’t seem to be accurate - thanks to [Jaime Sicam](#) and [Yves Trudeau](#) for help

How to get PMM Server

PMM is available for installation using three methods:

- [Docker Hub](#) – `docker pull percona/pmm-server` – [Documentation](#)
- [AWS Marketplace](#) – [Documentation](#)
- [Open Virtualization Format \(OVF\)](#) – [Documentation](#)

Percona Monitoring and Management 1.16.0

Date November 1, 2018

PMM (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL*, *MongoDB*, and *PostgreSQL* performance. You can run *PMM* in your own environment for maximum security and reliability. It provides thorough time-based analysis for *MySQL* and *MongoDB* servers to ensure that your data works as efficiently as possible.

While much of the team is working on longer-term projects, we were able to provide the following features:

- *MySQL* and *PostgreSQL* support for all cloud DBaaS providers – Use PMM Server to gather Metrics and Queries from remote instances!
- Query Analytics + Metric Series – See Database activity alongside queries
- Collect local metrics using `node_exporter` + textfile collector

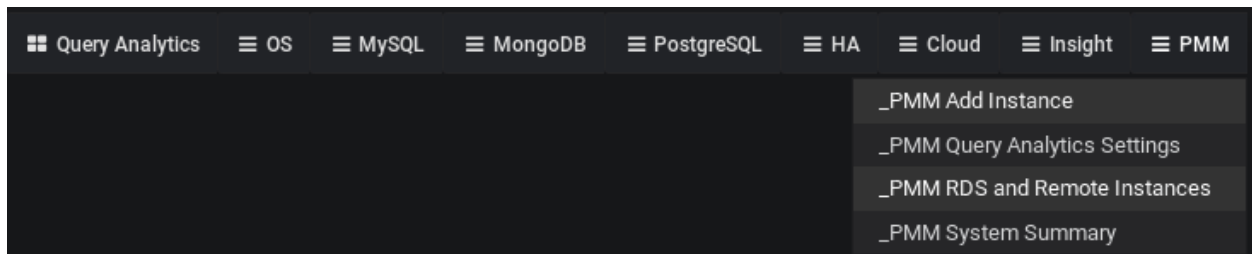
We addressed 11 new features and improvements, and fixed 21 bugs.

MySQL and PostgreSQL support for all cloud DBaaS providers

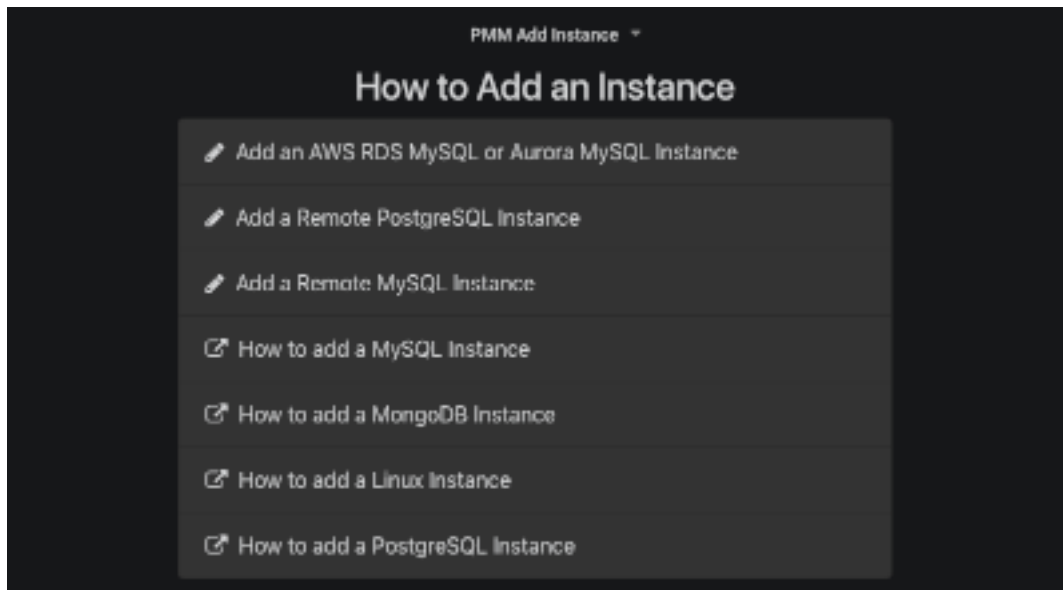
You're now able to connect *PMM Server* to your *MySQL* and *PostgreSQL* instances, whether they run in a cloud DBaaS environment, or you simply want Database metrics without the OS metrics. This can help you get up and running with *PMM* using minimal configuration and zero client installation, however be aware there are limitations – *there won't be any host-level dashboards populated for these nodes* since we don't attempt to connect to the provider's API nor are we granted access to the instance in order to deploy an exporter.

How to use

Using the *PMM Add Instance* screen, you can now add instances from any cloud provider (AWS RDS and Aurora, Google Cloud SQL for MySQL, Azure Database for MySQL) and benefit from the same dashboards that you are already accustomed to. You'll be able to collect Metrics and Queries from *MySQL*, and Metrics from *PostgreSQL*. You can add remote instances by selecting the *PMM Add Instance* item in a *PMM Dropdown* group of the system menu:



where you will then have the opportunity to add a Remote MySQL or Remote PostgreSQL instance:





You'll add the instance by supplying just the Hostname, database Username and Password (and optional Port and Name):


Also new as part of this release is the ability to display nodes you've added, on screen RDS and Remote Instances:


PMM Add Instance


Add remote MySQL Instance

*Hostname 

Name (default: Hostname) 


(default: 3306) 



*Username 

*Password 

Add instance

RDS and remote instances

Name	Endpoint	Region	Engine	Remove
Hostname	172.19.0.1:5432	remote	PostgreSQL	

  1.16.0

Server activity metrics in the PMM Query Analytics dashboard

The Query Analytics dashboard now shows a summary of the selected host and database activity metrics in addition to the top ten queries listed in a summary table. This brings a view of System Activity (CPU, Disk, and Network) and Database Server Activity (Connections, Queries per Second, and Threads Running) to help you better pinpoint query pileups and other bottlenecks:



Extending metrics with node_exporter textfile collector

While *PMM* provides an excellent solution for system monitoring, sometimes you may have the need for a metric that's not present in the list of *node_exporter* metrics out of the box. There is a simple method to extend the list of available metrics without modifying the *node_exporter* code. It is based on the textfile collector. We've enabled this collector as on by default, and is deployed as part of `linux:metrics` in *PMM Client*.

The default directory for reading text files with the metrics is `/usr/local/percona/pmm-client/textfile-collector`, and the exporter reads files from it with the `.prom` extension. By default it contains an example file `example.prom` which has commented contents and can be used as a template.

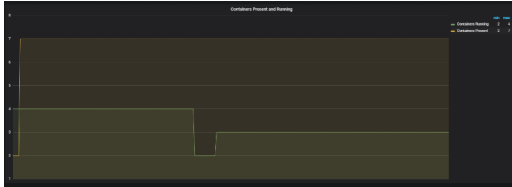
You are responsible for running a cronjob or other regular process to generate the metric series data and write it to this directory.

Example – collecting docker container information

This example will show you how to collect the number of running and stopped docker containers on a host. It uses a crontab task, set with the following lines in the cron configuration file (e.g. in `/etc/crontab`):

```
*/* * * * * root echo -n "" > /tmp/docker_all.prom; docker ps -a -q | wc -l |
↳xargs echo node_docker_containers_total >> /usr/local/percona/pmm-client/docker_all.
↳prom;
*/1 * * * * root echo -n "" > /tmp/docker_running.prom; docker ps | wc -l |
↳xargs echo node_docker_containers_running_total >> /usr/local/percona/pmm-client/
↳docker_running.prom;
```

The result of the commands is placed into the `docker_all.prom` and `docker_running.prom` files and read by exporter and will create two new metric series named `node_docker_containers_total` and `node_docker_containers_running_total`, which we'll then plot on a graph:



New Features and Improvements

- PMM-3195: Remove the light bulb
- PMM-3194: Change link for “Where do I get the security credentials for my Amazon RDS DB instance?”
- PMM-3189: Include Remote MySQL & PostgreSQL instance logs into PMM Server logs.zip system
- PMM-3166: Convert status integers to strings on ProxySQL Overview Dashboard – Thanks, [Iwo Panowicz](https://github.com/percona/grafana-dashboards/pull/239) for <https://github.com/percona/grafana-dashboards/pull/239>
- PMM-3133: Include Metric Series on Query Analytics Dashboard
- PMM-3078: Generate warning “how to troubleshoot postgresql:metrics” after failed `pmm-admin add postgresql` execution
- PMM-3061: Provide Ability to Monitor Remote MySQL and PostgreSQL Instances
- PMM-2888: Enable Textfile Collector by Default in `node_exporter`
- PMM-2880: Use consistent favicon (Percona logo) across all distribution methods
- PMM-2306: Configure EBS disk resize utility to run from crontab in PMM Server
- PMM-1358: Improve Tooltips on Disk Space Dashboard – thanks, [Corrado Pandiani](#) for texts

Fixed Bugs

- PMM-3202: Cannot add remote PostgreSQL to monitoring without specified dbname
- PMM-3186: Strange “Quick ranges” tag appears when you hover over documentation links on PMM Add Instance screen
- PMM-3182: Some sections for MongoDB are collapsed by default
- PMM-3171: Remote RDS instance cannot be deleted
- PMM-3159: Problem with enabling RDS instance
- PMM-3127: “Expand all” button affects JSON in all queries instead of the selected one
- PMM-3126: Last check displays locale format of the date
- PMM-3097: Update home dashboard to support PostgreSQL nodes in Environment Overview
- PMM-3091: `postgres_exporter` typo
- PMM-3090: TLS handshake error in PostgreSQL metric
- PMM-3088: It’s possible to downgrade PMM from Home dashboard
- PMM-3072: Copy to clipboard is not visible for JSON in case of long queries

- [PMM-3038](#): Error adding MySQL queries when options for `mysqld_exporters` are used
- [PMM-3028](#): Mark points are hidden if an annotation isn't added in advance
- [PMM-3027](#): Number of vCPUs for RDS is displayed incorrectly – report and proposal from [Janos Ruszo](#)
- [PMM-2762](#): Page refresh makes Search condition lost and shows all queries
- [PMM-2483](#): LVM in the PMM Server AMI is poorly configured/documented – reported by [Olivier Mignault](#) and lot of people involved. Special thanks to [Chris Schneider](#) for checking with fix options
- [PMM-2003](#): Delete all info related to external exporters on `pmm-admin` list output

How to get PMM Server

PMM is available for installation using three methods:

- [Docker Hub](#) – `docker pull percona/pmm-server` – [Documentation](#)
- [AWS Marketplace](#) – [Documentation](#)
- [Open Virtualization Format \(OVF\)](#) – [Documentation](#)

Percona Monitoring and Management 1.15.0

Date October 10, 2018

PMM ([Percona Monitoring and Management](#)) is a free and open-source platform for managing and monitoring *MySQL* and *MongoDB* performance. You can run *PMM* 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.

This release offers two new features for both the *MySQL* Community and Percona Customers:

- [MySQL Custom Queries](#) - Turn a `SELECT` into a dashboard!!
- [Server and Client logs](#) - Collect troubleshooting logs for Percona Support

We addressed 17 new features and improvements, and fixed 17 bugs.

MySQL Custom Queries

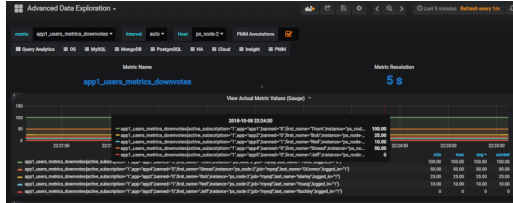
In 1.15 we are introducing the ability to take a `SQL SELECT` statement and turn the result set into metric series in *PMM*. The queries are executed at the `LOW RESOLUTION` level, which by default is every 60 seconds. A key advantage is that you can extend *PMM* to profile metrics unique to your environment (see `users` table example), or to introduce support for a table that isn't part of *PMM* yet. This feature is on by default and only requires that you edit the configuration file and use valid `YAML` syntax. The configuration file is in `/usr/local/percona/pmm-client/queries-mysqld.yml`.

Example - Application users table

We're going to take a fictional *MySQL* `users` table that also tracks the number of upvotes and downvotes, and we'll convert this into two metric series, with a set of seven labels, where each label can also store a value.

Browsing metrics series using Advanced Data Exploration Dashboard

Lets look at the output so we understand the goal - take data from a *MySQL* table and store in *PMM*, then display as a metric series. Using the Advanced Data Exploration Dashboard you can review your metric series. Exploring the metric series `appl_users_metrics_downvotes` we see the following:



MySQL table

Lets assume you have the following users table that includes true/false, string, and integer types.

```
SELECT * FROM `users`
+----+-----+-----+-----+-----+-----+-----+-----+
↪+----+-----+-----+-----+-----+-----+-----+-----+
| id | app | user_type | last_name | first_name | logged_in | active_subscription |
↪| banned | upvotes | downvotes |
+----+-----+-----+-----+-----+-----+-----+-----+
↪+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | app2 | unprivileged | Marley | Bob | 1 | 1 |
↪| 0 | 100 | 25 |
| 2 | app3 | moderator | Young | Neil | 1 | 1 |
↪| 1 | 150 | 10 |
| 3 | app4 | unprivileged | OConnor | Sinead | 1 | 1 |
↪| 0 | 25 | 50 |
| 4 | app1 | unprivileged | Yorke | Thom | 0 | 1 |
↪| 0 | 100 | 100 |
| 5 | app5 | admin | Buckley | Jeff | 1 | 1 |
↪| 0 | 175 | 0 |
+----+-----+-----+-----+-----+-----+-----+-----+
↪+----+-----+-----+-----+-----+-----+-----+-----+
```

Explaining the YAML syntax

We'll go through a simple example and mention what's required for each line. The metric series is constructed based on the first line and appends the column name to form metric series. Therefore the number of metric series per table will be the count of columns that are of type GAUGE or COUNTER. This metric series will be called `appl_users_metrics_downvotes`:

```
appl_users_metrics:                                     ## leading section of your metric_
↪series.
  query: "SELECT * FROM appl.users"                     ## Your query. Don't forget the_
↪schema name.
  metrics:                                             ## Required line to start the_
↪list of metric items
    - downvotes:                                       ## Name of the column returned by_
↪the query. Will be appended to the metric series.
      usage: "COUNTER"                                 ## Column value type. COUNTER_
↪will make this a metric series.
```

```
description: "Number of upvotes"           ## Helpful description of the_
↪column.
```

Full queries-mysqld.yml example

Each column in the SELECT is named in this example, but that isn't required, you can use a SELECT * as well. Notice the format of schema.table for the query is included.

```
---
appl_users_metrics:
  query: "SELECT app,first_name,last_name,logged_in,active_subscription,banned,
↪upvotes,downvotes FROM appl.users"
  metrics:
    - app:
      usage: "LABEL"
      description: "Name of the Application"
    - user_type:
      usage: "LABEL"
      description: "User's privilege level within the Application"
    - first_name:
      usage: "LABEL"
      description: "User's First Name"
    - last_name:
      usage: "LABEL"
      description: "User's Last Name"
    - logged_in:
      usage: "LABEL"
      description: "User's logged in or out status"
    - active_subscription:
      usage: "LABEL"
      description: "Whether User has an active subscription or not"
    - banned:
      usage: "LABEL"
      description: "Whether user is banned or not"
    - upvotes:
      usage: "COUNTER"
      description: "Count of upvotes the User has earned. Upvotes once granted_
↪cannot be revoked, so the number can only increase."
      - downvotes:
        usage: "GAUGE"
        description: "Count of downvotes the User has earned. Downvotes can be_
↪revoked so the number can increase as well as decrease."
    ...
```

We hope you enjoy this feature, and we welcome your feedback via the [Percona forums](#)!

Server and Client logs

We've enhanced the volume of data collected from both the Server and Client perspectives. Each service provides a set of files designed to be shared with Percona Support while you work on an issue.

Server

From the Server, we've improved the `logs.zip` service to include:

- Prometheus targets
- Consul nodes, QAN API instances
- Amazon RDS and Aurora instances
- Version
- Server configuration
- Percona Toolkit commands

You retrieve the link from your *PMM server* using this format:

<https://pmmdemo.percona.com/managed/logs.zip>

Client

On the Client side we've added a new action called `summary` which fetches logs, network, and *Percona Toolkit* output in order to share with Percona Support. To initiate a Client side collection, execute:

```
pmm-admin summary
```

The output will be a file you can use to attach to your Support ticket. The single file will look something like this:

```
summary__2018_10_10_16_20_00.tar.gz
```

New Features and Improvements

- **PMM-2913:** Provide ability to execute Custom Queries against MySQL - Credit to [wrouesnel](#) for the framework of this feature in [wrouesnel/postgres_exporter](#)!
- **PMM-2904:** Improve PMM Server Diagnostics for Support
- **PMM-2860:** Improve pmm-client Diagnostics for Support
- **PMM-1754:** Provide functionality to easily select query and copy it to clipboard in QAN
- **PMM-1855:** Add swap to AMI
- **PMM-3013:** Rename PXC Overview graph Sequence numbers of transactions to IST Progress
- **PMM-2726:** Abort data collection in Exporters based on Prometheus Timeout - MySQLd Exporter
- **PMM-3003:** PostgreSQL Overview Dashboard Tooltip fixes
- **PMM-2936:** Some improvements for Query Analytics Settings screen
- **PMM-3029:** PostgreSQL Dashboard Improvements

Fixed Bugs

- **PMM-2976:** Upgrading to PMM 1.14.x fails if dashboards from Grafana 4.x are present on an installation
- **PMM-2969:** rds_exporter becomes throttled by CloudWatch API
- **PMM-1443:** The credentials for a secured server are exposed without explicit request
- **PMM-3006:** Monitoring over 1000 instances is displayed imperfectly on the label
- **PMM-3011:** PMM's default MongoDB DSN is localhost, which is not resolved to IPv4 on modern systems

- [PMM-2211](#): Bad display when using old range in QAN
- [PMM-1664](#): Infinite loading with wrong queryID
- [PMM-2715](#): Since pmm-client-1.9.0, pmm-admin detects CentOS/RHEL 6 installations using linux-upstart as service manager and ignores SysV scripts
- [PMM-2839](#): Tablestats safety precaution does not work for RDS/Aurora instances
- [PMM-2845](#): pmm-admin purge causes client to panic
- [PMM-2968](#): pmm-admin list shows empty data source column for mysql:metrics
- [PMM-3043](#): Total Time percentage is incorrectly shown as a decimal fraction
- [PMM-3082](#): Prometheus Scrape Interval Variance chart doesn't display data

How to get PMM Server

PMM is available for installation using three methods:

- On [Docker Hub](#) – docker pull percona/pmm-server – [Documentation](#)
- [AWS Marketplace](#) – [Documentation](#)
- [Open Virtualization Format \(OVF\)](#) – [Documentation](#)

Percona Monitoring and Management 1.14.1

Date September 7, 2018

PMM ([Percona Monitoring and Management](#)) is a free and open-source platform for managing and monitoring *MySQL* and *MongoDB* performance. You can run *PMM* 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.

We're releasing hotfix 1.14.1 to address three issues found post-release of 1.14.0:

- [PMM-2963](#) - Upgrading to PMM 1.14.0 fails due to attempt to create already existing Dashboard
 - Our upgrade script incorrectly tried to create dashboards that already existed, and generating failure message:

```
A folder or dashboard in the general folder with the same name already exists
```
- [PMM-2958](#) - Grafana did not update to 5.1 when upgrading from versions older than 1.11
 - We identified a niche case where PMM installations that were upgraded from < 1.11 would fail to upgrade Grafana to correct release 5.1 (Users were left on Grafana 5.0)

Percona Monitoring and Management 1.14.0

Date September 5, 2018

PMM ([Percona Monitoring and Management](#)) is a free and open-source platform for managing and monitoring *MySQL* and *MongoDB* performance. You can run *PMM* 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.

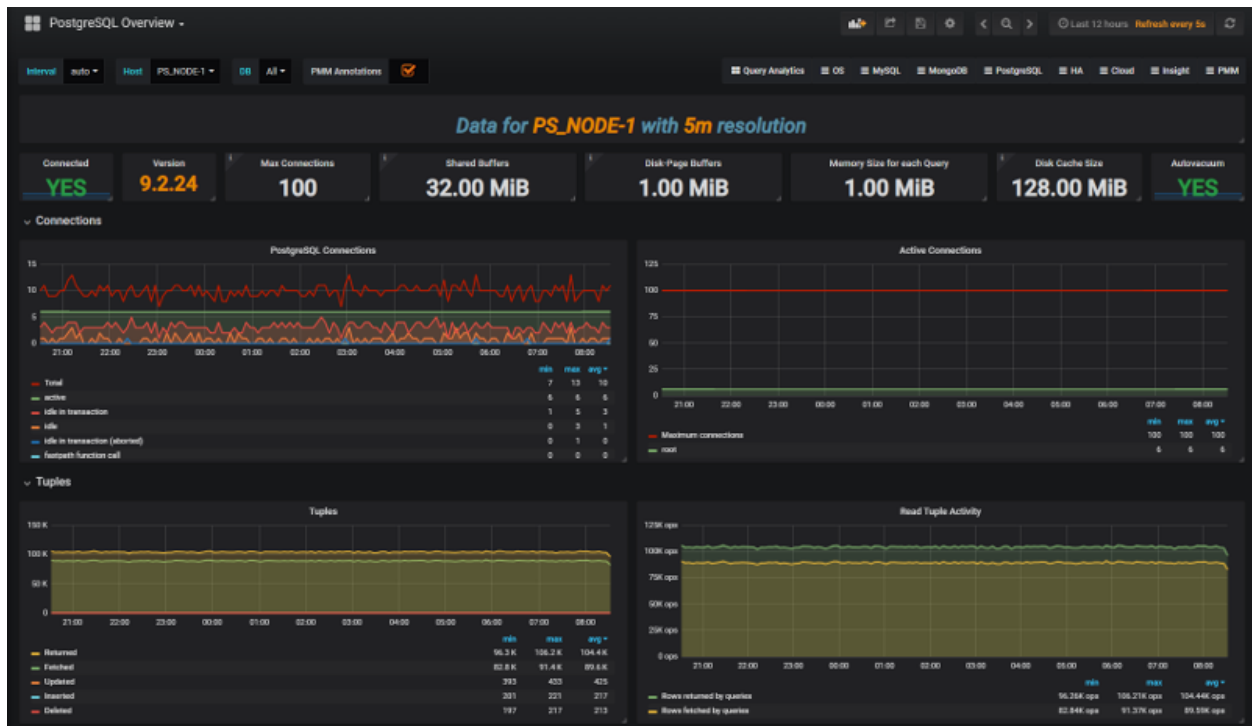
We've included a plethora of visual improvements in this release, including:

- PostgreSQL Metrics Collection - Visualize PostgreSQL performance!
- Identify New Queries in Query Analytics
- New Dashboard: Compare System Parameters
- New Dashboard: PERFORMANCE_SCHEMA Wait Events Analysis
- Dashboard Updates - Advanced Data Exploration, MyRocks, TokuDB, InnoDB Metrics
- Disable SSL between Prometheus and Exporters
- Dashboards grouped by Folder - We've organized the Dashboard drop-down to present a cleaner interface

We addressed 16 new features and improvements, and fixed 20 bugs.

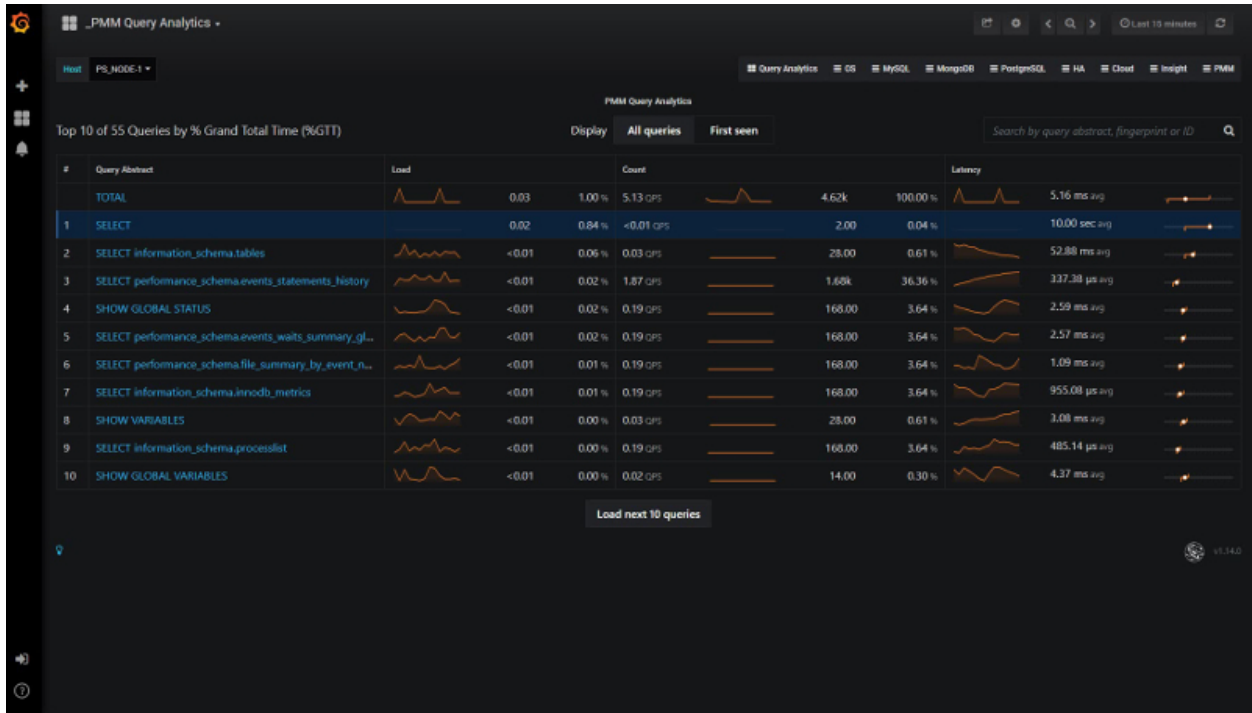
PostgreSQL Metrics Collection

The PMM team is very proud to bring you native support for PostgreSQL! We've shipped a new dashboard called PostgreSQL Overview, and we now provide the ability to add PostgreSQL instances as native, first-class citizens as part of PMM. This means you can add PostgreSQL + Linux monitoring capabilities through the standard `pmm-admin add postgresql` syntax, see our documentation links for more details!



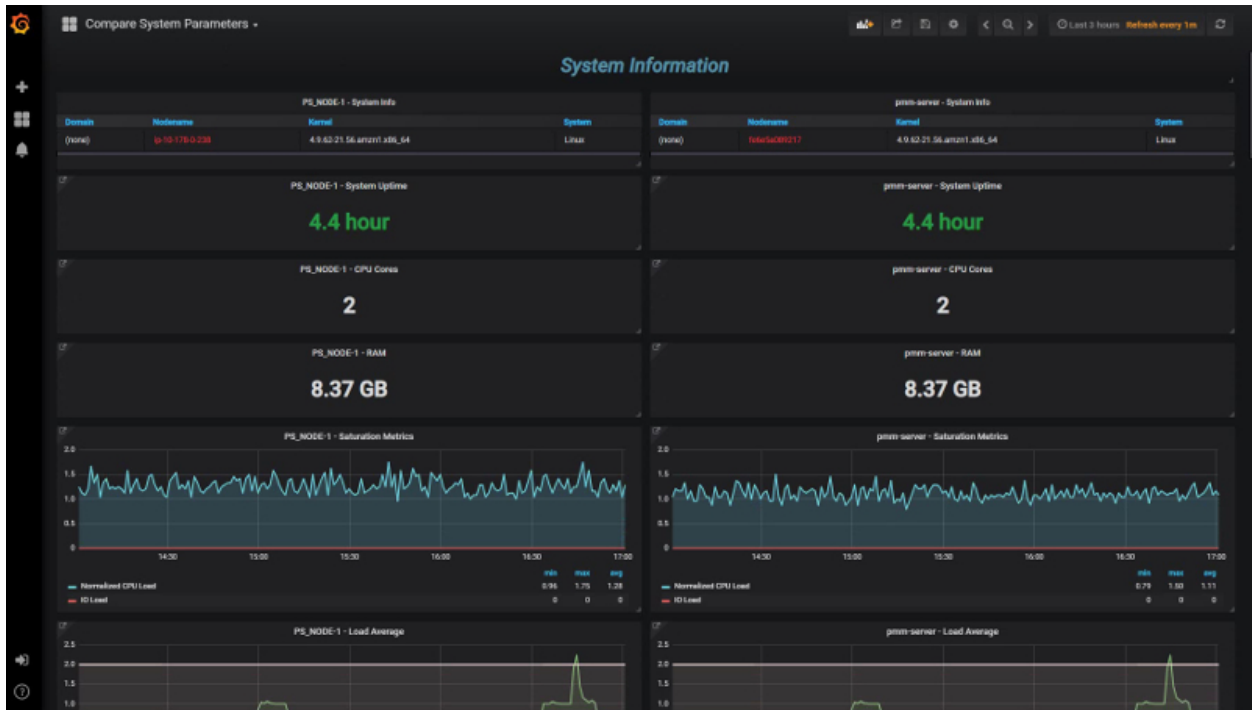
Identify New Queries in Query Analytics

A long-awaited feature is the ability to visually identify new queries that have appeared in Query Analytics - those queries who's first seen time is within the selected time range. New queries will be highlighted in a soft blue band for quick identification, and we've provided a button called First Seen which you can toggle to display only those newly seen queries. A common use case for this feature is potentially during code release / deployments, where you want to review which new queries have been deployed and to review their performance characteristics.



New Dashboard: Compare System Parameters

We've introduced a new dashboard to let you compare System Parameters across multiple servers so at a glance you can understand provisioning or configuration differences. This might be of help when comparing a pool of identical slaves or other logical groups of instances.



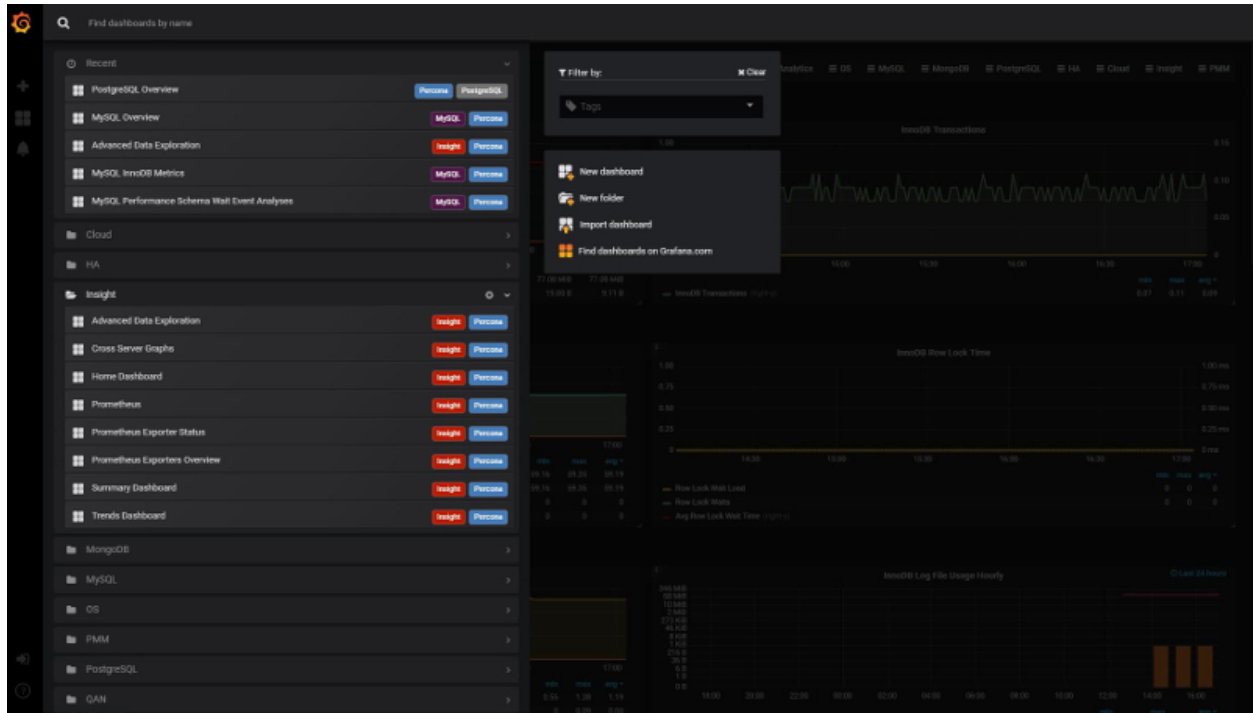
New Dashboard: PERFORMANCE_SCHEMA Wait Events Analysis

We've added a new dashboard that lets you drill down into great detail on one or several PERFORMANCE_SCHEMA wait event categories in order to visualize them over time.



Dashboards grouped by Folder

At long last we've addressed the sprawl of the long list of 30+ Dashboards, and grouped them into categories which match the pre-existing right-side navigation system. This should leave you with a more organized, less cluttered list of Dashboards.



Dashboard Updates - Advanced Data Exploration, MyRocks, TokuDB, InnoDB Metrics

We've improved four dashboards with minor but helpful improvements:

- Advanced Data Exploration dashboard with the addition of a graph element plotting the Metric Rates, which will help you understand the scraping efficiency of this metric series, or whether scrapes have failed / are failing.
- InnoDB Metrics to present the graph elements in two columns - previously we'd inconsistently use three columns or two columns, making it hard to visualize trends across graphs.
- MyRocks formulas were improved to be more precise
- TokuDB has many new graphs to expand our coverage of this Storage Engine

Disable SSL between PMM Server and Exporters

Lastly, we've delivered on a feature request from a Percona Customer to optionally disable SSL between PMM Server and Exporters, with the advantage that if you do not need encrypted traffic for your metric series, you can reduce the CPU overhead on PMM Server. We'd love to hear your feedback on this feature!

```
pmm-admin add mysql --disable-ssl ...
```

New Features & Improvements

- [PMM-1362](#): Update descriptions on MySQL InnoDB Metrics (Advanced) Dashboard - thanks to Yves Trudeau
- [PMM-2304](#): New Dashboard: Compare System Parameters
- [PMM-2331](#): Advanced Data Exploration: add graph for showing exporter scrapers over time intervals

- PMM-2356: Grouping dashboards in folders with Grafana5
- PMM-2472: Identify new queries in QAN
- PMM-2486: Allow the disabling of SSL by means of an option - thanks to Dongchan Sung
- PMM-2597: Improve MyRocks dashboard - thanks to Przemek Malkowski for the valuable ideas
- PMM-2704: PostgreSQL Metrics Collection
- PMM-2772: Display InnoDB Metrics dashboard using consistent two column view
- PMM-2775: Display PERFORMANCE_SCHEMA Wait Events Analysis
- PMM-2769: Display TokuDB Dashboard Improvements
- PMM-2797: MySQL Performance Schema - Filter HOSTS
- PMM-2798: Filter hosts on NUMA dashboard
- PMM-2833: Added granularity interval for scraping AWS API - thanks to Aleksandr Stepanov
- PMM-2846: Increase MySQL Max Connections in PMM Server

Fixed Bugs

- PMM-946: QAN sparklines drop to zero when data is not available
- PMM-1987: pt-archiver rule for agent_log is not correct - thanks to Yves Trudeau for providing a fix
- PMM-2013: Styling of QAN allows overlapping content
- PMM-2028: nginx shows “414 Request-URI Too Large” for 150 hosts - thanks to Nickolay Ihalainen for the bug report and fix
- PMM-2166: Add RDS instance page refresh will head to “Page Not Found” error
- PMM-2457: Improve External Exporter help documentation for duration interval
- PMM-2459: Cross-Graph Crosshair not enabled on the PXC/Galera Cluster
- PMM-2477: Frequent Access Denied prompts while using AWS Marketplace image
- PMM-2566: CPU busy graph shows incorrect values
- PMM-2763: Unknown version is available on Update widget
- PMM-2784: What’s new link on Update widget has wrong URL
- PMM-2793: Network Overview needs to be in OS menu, not insights
- PMM-2796: Overview NUMA Metrics dashboard should be renamed to NUMA Overview
- PMM-2801: Prometheus Exporters Overview - CPU metrics are strange
- PMM-2804: Prometheus Graph is empty with PMM 1.13
- PMM-2811: SQL to get Hosts in QAN - thanks to Forums member Fan
- PMM-2821: Clean local storage if status is “You are up to date” and use animation for refresh button
- PMM-2828: Weird Latency Graphs
- PMM-2841: Change memory defaults for Prometheus 1.8 and use additional environment variable
- PMM-2856: RDS/Aurora disk related graphs are empty
- PMM-2885: System Overview dashboard has incorrect values

Percona Monitoring and Management 1.13.0

Date August 1, 2018

The most significant feature in this release is Prometheus 2, however we also packed a lot of visual changes into release 1.13:

Prometheus 2 - Consumes less resources, and Dashboards load faster!

- **New Dashboard: Network Overview** - New dashboard for all things IPv4!
- **New Dashboard: NUMA Overview** - New Dashboard! Understand memory allocation across DIMMs
- **Snapshots and Updates Improvements** - Clearer instructions for snapshot sharing, add ability to disable update reporting
- **System Overview Dashboard improvements** - See high level summary, plus drill in on CPU, Memory, Disk, and Network
- **Improved SingleStat for percentages** - Trend line now reflects percentage value

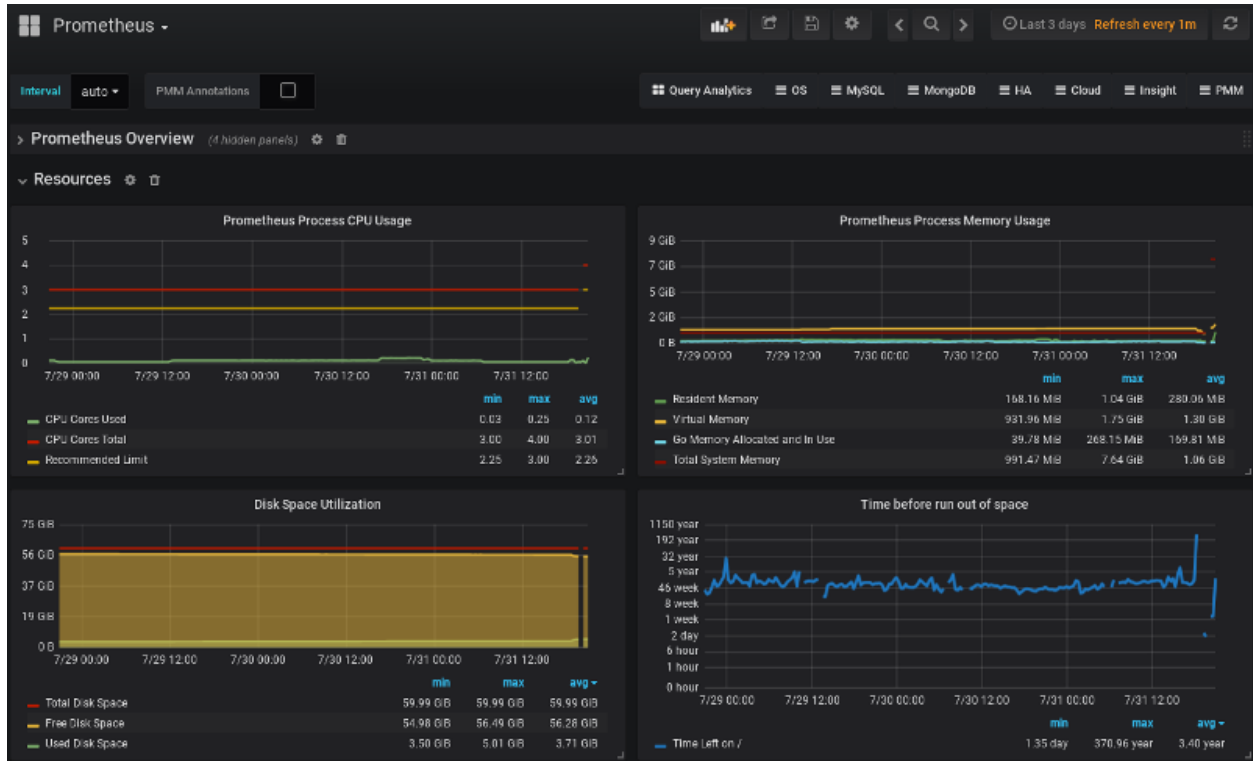
We addressed 13 new features and improvements, and fixed 13 bugs.

Prometheus 2

The long awaited Prometheus 2 release is here! By upgrading to PMM release 1.13, Percona's internal testing has shown you will achieve a 3x-10x reduction in CPU usage, which translates into PMM Server being able to handle more instances than you could in 1.12. You won't see any gaps in graphs since internally PMM Server will run two instances of Prometheus and leverage *remote_read* in order to provide consistent graphs!

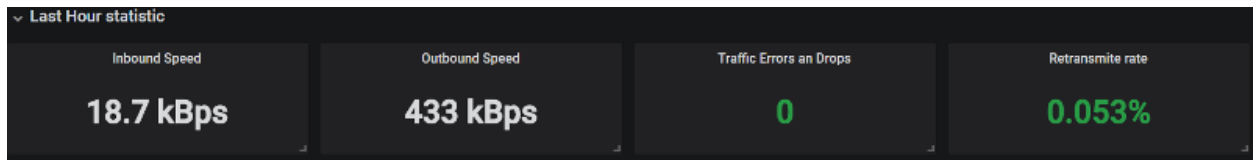
Our Engineering teams have worked very hard to make this upgrade as transparent as possible - hats off to them for their efforts!!

Lastly on Prometheus 2, we also included a new set of graphs to the Prometheus Dashboard to help you better understand when your PMM Server may run out of space. We hope you find this useful!

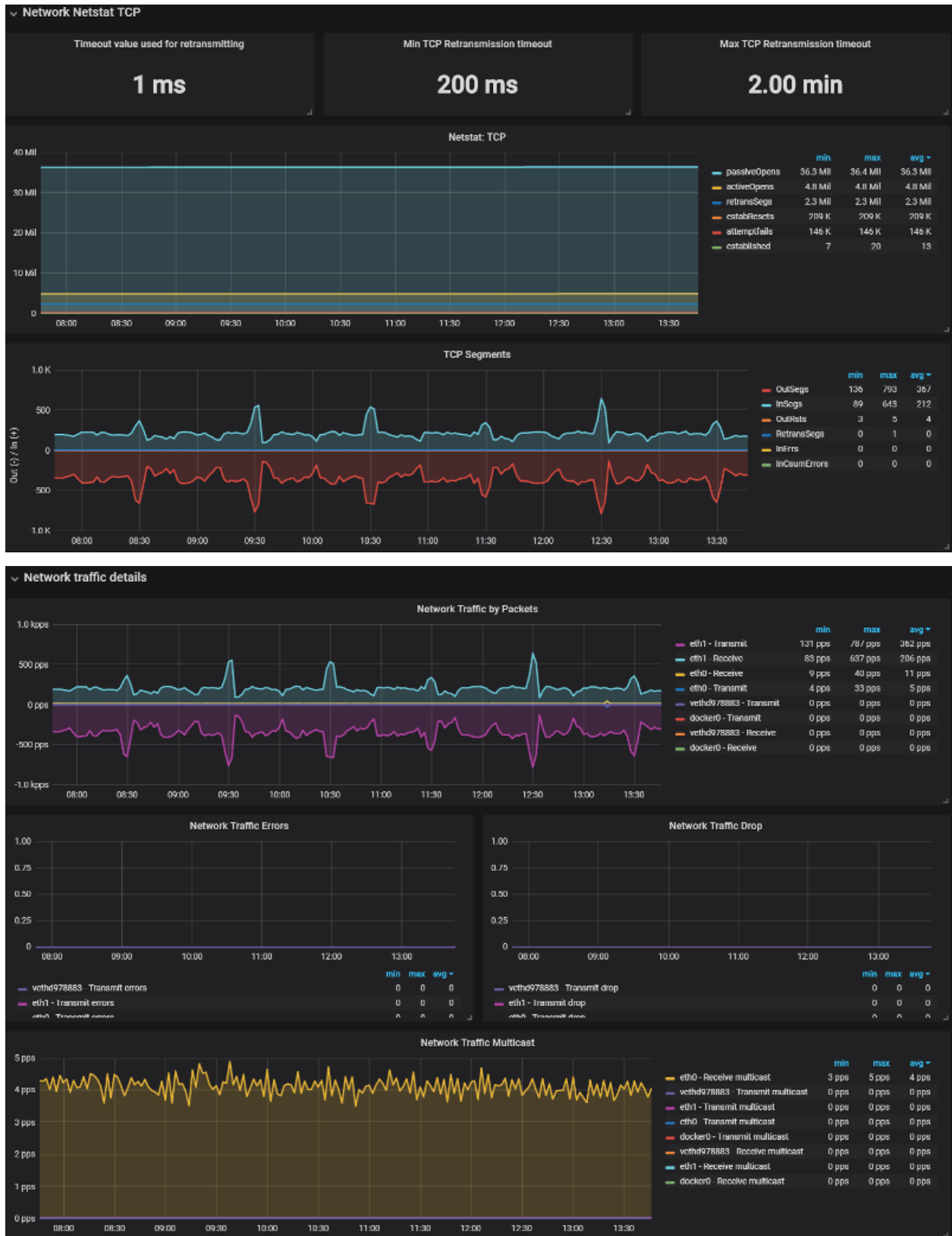


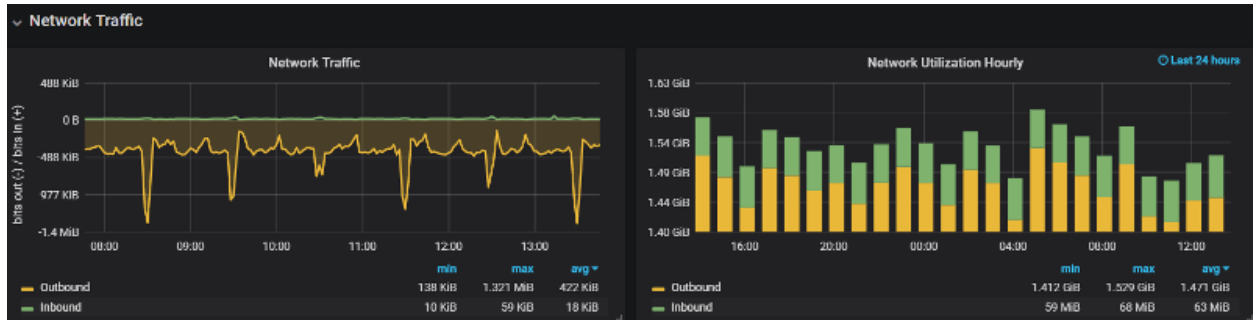
Network Overview Dashboard

We're introducing a new dashboard that focuses on all things Networking - we placed a Last Hour panel highlighting high-level network metrics, and then drill into Network Traffic + Details, then focus on TCP, UDP, and ICMP behavior.









Snapshots and Updates Improvements

Of most interest to current Percona Customers, we've clarified the instructions on how to take a snapshot of a Dashboard in order to highlight that you are securely sharing with Percona. We've also configured the sharing timeout to 30 seconds (up from 4 seconds) so that we more reliably share useful data to Percona Support Engineers, as shorter timeout led to incomplete graphs being shared.

The screenshot shows a 'Share' dialog box with tabs for 'Link', 'Snapshot', and 'Export'. The 'Snapshot' tab is active. It contains a description of snapshots, a list of configuration options, and a 'What to do next' section. At the bottom, there are three buttons: 'Local Snapshot', 'Share with Percona', and 'Cancel'.

A snapshot is a way to securely share your dashboard with Percona. When created, we strip sensitive data like queries (metrics, template variables, and annotations) along with panel links. The shared dashboard will only be available for viewing by Percona Engineers, and the content on the dashboard will assist Percona Engineers in troubleshooting your case.

You can safely leave the defaults set as they are, but for further information:

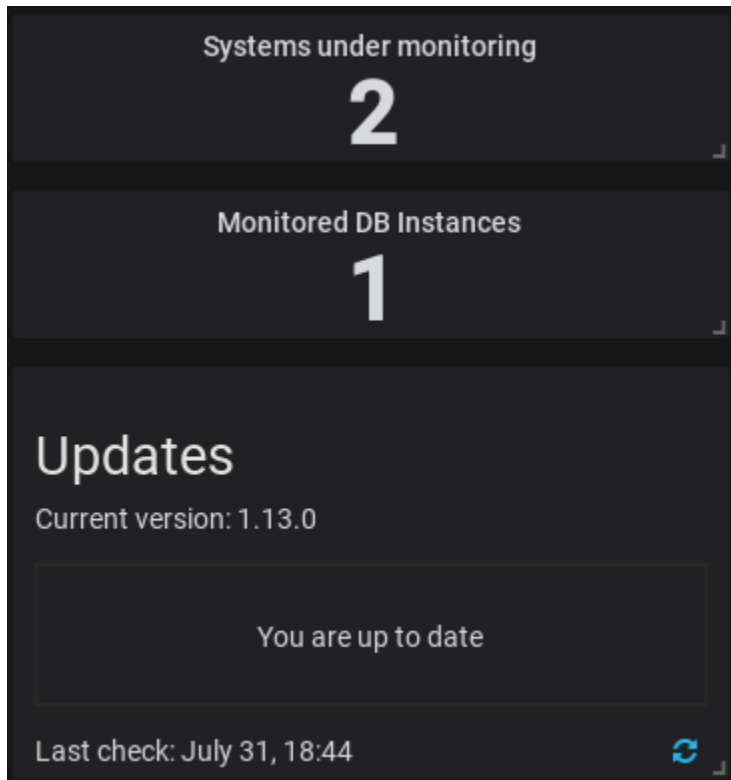
- **Snapshot Name:** The name Percona will see when viewing your dashboard.
- **Expire:** How long before snapshot should expire, configure lower if required. Percona automatically purges shared dashboards after 90 days.
- **Timeout (seconds):** Duration the dashboard will take to load before the snapshot is generated.

What to do next: After clicking Share with Percona, wait for the dashboard to be generated, and you will be provided a unique URL that then needs to be communicated to Percona via your ticket.

Snapshot name	Prometheus
Expire	90 Days
Timeout (seconds)	30

Local Snapshot Share with Percona Cancel

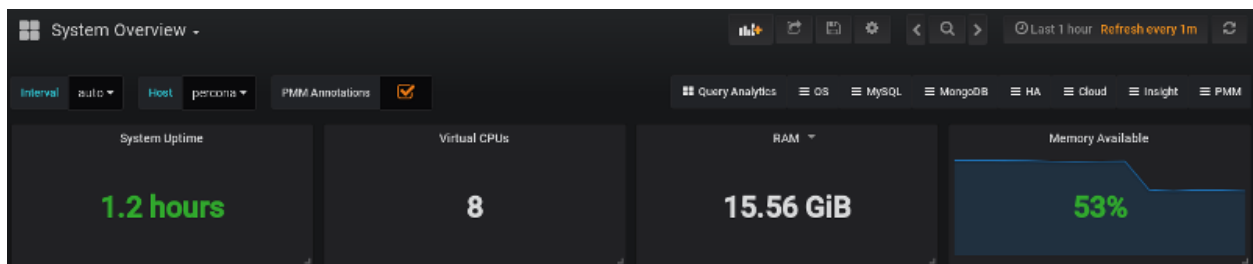
Packed into this feature is also a change to how we report installed version, latest version, and what's new information:

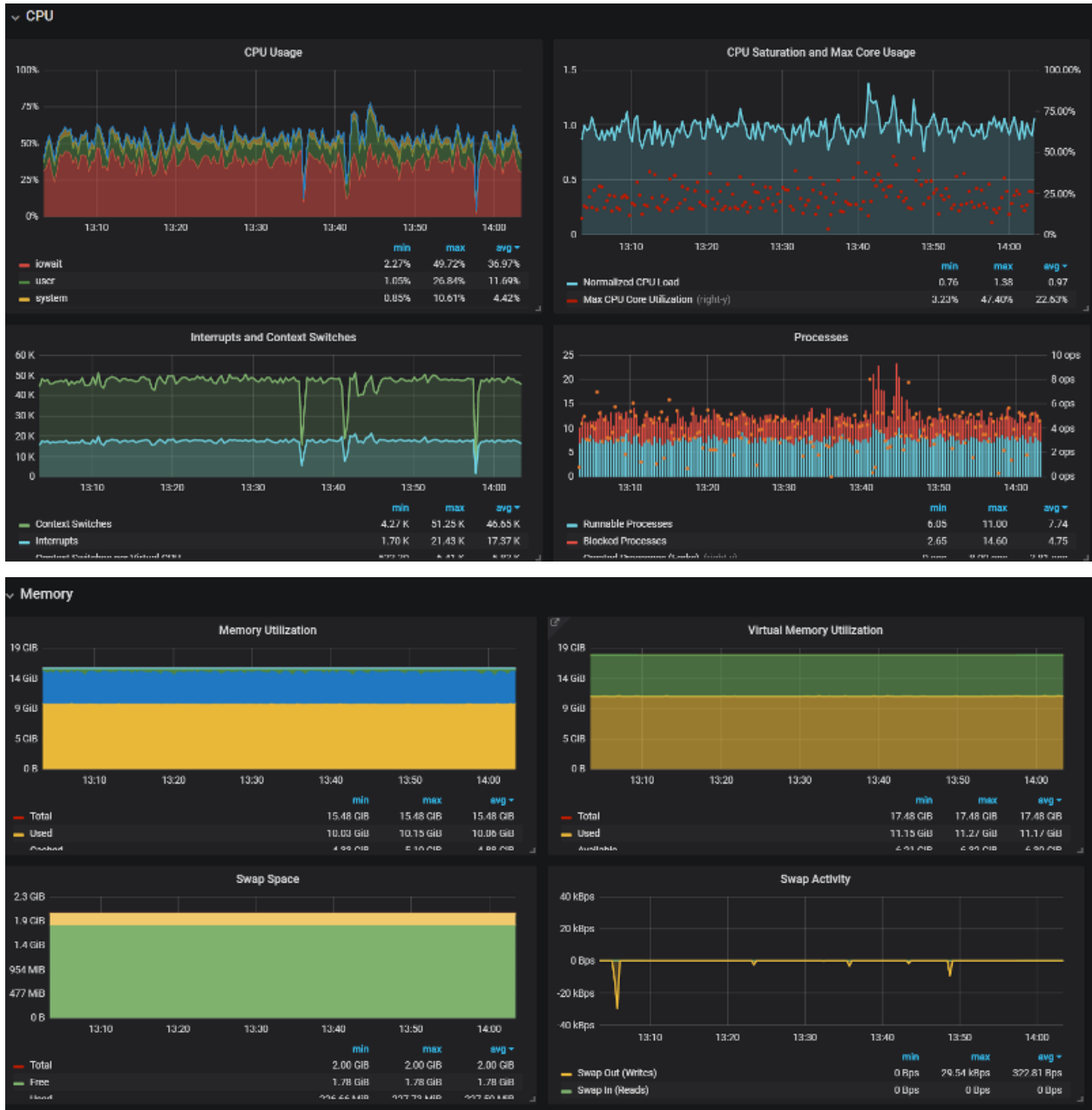


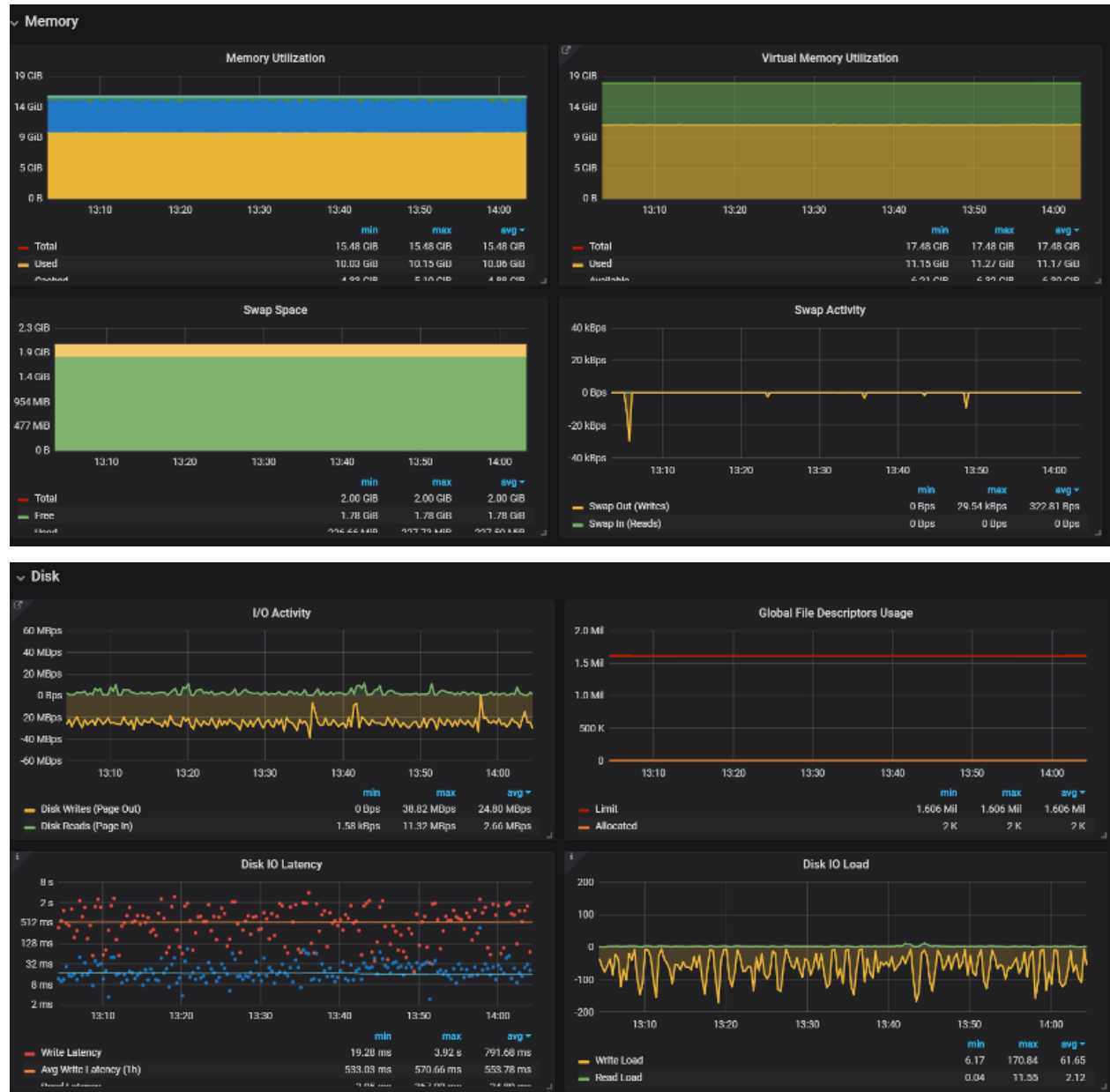
Lastly, we modified the behavior of the docker environment option `DISABLE_UPDATES` to remove the Update button. As a reminder, you can choose to disable update reporting for environments where you want tighter control over (i.e. lock down) who can initiate an update by launching the PMM docker container along with the environment variable as follows:

```
docker run ... -e DISABLE_UPDATES=TRUE
```

System Overview Dashboard Improvements We've updated our System Overview Dashboard to focus on the four criteria of CPU, Memory, Disk, and Network, while also presenting a single panel row of high level information (uptime, count of CPUs, load average, etc)

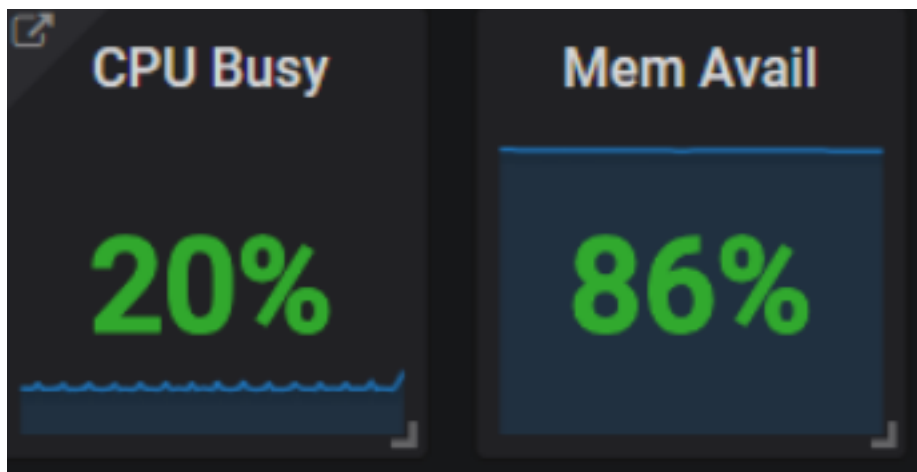








Our last feature we're introducing in 1.13 is a fix to [SingleStat](#) panels where the percentage value is reflected in the level of the trend line in the background. For example, if you have a stat panel at 20% and 86%, the line in the background should fill the respective amount of the box: Improved SingleStat for percentages.



Issues in this release

New Features & Improvements

- PMM-2225: Add new Dashboard: Network Overview
- PMM-2485: Improve Singlestat for percentage values to accurately display trend line
- PMM-2550: Update to Prometheus 2
- PMM-1667: New Dashboard: NUMA Overview
- PMM-1930: Reduce Durability for MySQL
- PMM-2291: Add Prometheus Disk Space Utilization Information
- PMM-2444: Increase space for legends

- PMM-2594: Upgrade to Percona Toolkit 3.0.10
- PMM-2610: Configure Snapshot Timeout Default Higher and Update Instructions
- PMM-2637: Check for Updates and Disable Updates Improvements
- PMM-2652: Fix “Unexpected error” on Home dashboard after upgrade
- PMM-2661: Data resolution on Dashboards became 15sec min instead of 1sec
- PMM-2663: System Overview Dashboard Improvements

Bug fixes

- PMM-1977: after upgrade pmm-client (1.6.1-1) can't start mysql:metrics - can't find .my.cnf
- PMM-2379: Invert colours for Memory Available graph
- PMM-2413: Charts on MySQL InnoDB metrics are not fully displayed
- PMM-2427: Information loss in CPU Graph with Grafana 5 upgrade
- PMM-2476: AWS PMM is broken on C5/M5 instances
- PMM-2576: Error in logs for MySQL 8 instance on CentOS
- PMM-2612: Wrong information in PMM Scrapes Task
- PMM-2639: mysql:metrics does not work on Ubuntu 18.04
- PMM-2643: Socket detection and MySQL 8
- PMM-2698: Misleading Graphs for Rare Events
- PMM-2701: MySQL 8 - InnoDB Checkpoint Age
- PMM-2722: Memory auto-configuration for Prometheus evaluates to minimum of 128MB in entrypoint.sh

Percona Monitoring and Management 1.12.0

Date June 27, 2018

In release 1.12, we invested our efforts in the following areas:

Visual Explain in Query Analytics – Gain insight into MySQL’s query optimizer for your queries
New Dashboard – InnoDB Compression Metrics – Evaluate effectiveness of InnoDB Compression
New Dashboard – MySQL Command/Handler Compare – Contrast MySQL instances side by side
Updated Grafana to 5.1 – Fixed scrolling issues
We addressed 10 new features and improvements, and fixed 13 bugs.

In release 1.12, we invested our efforts in the following areas:

- *Visual Explain in Query Analytics* – Gain insight into MySQL’s query optimizer for your queries
- *New Dashboard – InnoDB Compression Metrics* – Evaluate effectiveness of InnoDB Compression
- *New Dashboard – MySQL Command/Handler Compare* – Contrast MySQL instances side by side
- Updated Grafana to 5.1 – Fixed scrolling issues

We addressed 10 new features and improvements, and fixed 13 bugs.

Visual Explain in Query Analytics

We're working on substantial changes to Query Analytics and the first part to roll out is something that users of Percona Toolkit may recognize – we've introduced a new element called Visual Explain based on pt-visual-explain. This functionality transforms MySQL EXPLAIN output into a left-deep tree representation of a query plan, in order to mimic how the plan is represented inside MySQL. This is of primary benefit when investigating tables that are joined in some logical way so that you can understand in what order the loops are executed by the MySQL query optimizer. In this example we are demonstrating the output of a single table lookup vs two table join:

Single Table Lookup

```
SELECT DISTINCT c
FROM sbtest13
WHERE id
BETWEEN 49808
AND 49907
ORDER BY c
```

Two Tables via INNER JOIN

```
SELECT sbtest3.c
FROM sbtest1
INNER JOIN sbtest3
ON sbtest1.id = sbtest3.id
WHERE sbtest3.c = 'long-string';
```

The image shows two side-by-side screenshots of the MySQL EXPLAIN output, specifically the 'VISUAL' view. Both screenshots are titled 'EXPLAIN PS_NODE_1' and have a 'Copy to clipboard' button.

The left screenshot shows the 'VISUAL' view for a single table lookup query. The tree structure is as follows:

```

CLASSIC
JSON
VISUAL
  Filesort
  +- TEMPORARY
  |   table          temporary(sbtest13)
  +- Filter with WHERE
  |   +- Bookmark lookup
  |   |   +- Table
  |   |   |   table          sbtest13
  |   |   |   possible_keys  PRIMARY
  |   |   +- Index range scan
  |   |       key             sbtest13->PRIMARY
  |   |       possible_keys  PRIMARY
  |   |       key_len        4
  |   |       rows           100
  
```

The right screenshot shows the 'VISUAL' view for an inner join query. The tree structure is as follows:

```

CLASSIC
JSON
VISUAL
  JOIN
  +- Bookmark lookup
  |   +- Table
  |   |   table          sbtest3
  |   |   possible_keys  PRIMARY
  |   +- Constant index lookup
  |       key             sbtest3->PRIMARY
  |       possible_keys  PRIMARY
  |       key_len        4
  |       ref             const
  |       rows           1
  +- Constant index lookup
  |   key             sbtest1->PRIMARY
  |   possible_keys  PRIMARY
  |   key_len        4
  |   ref             const
  |   rows           1
  
```

InnoDB Compression Metrics Dashboard

A great feature of MySQL's InnoDB storage engine includes compression of data that is transparently handled by the database, saving you space on disk, while reducing the amount of I/O to disk as fewer disk blocks are required to store the same amount of data, thus allowing you to reduce your storage costs. We've deployed a new dashboard that helps you understand the most important characteristics of InnoDB's Compression. Here's a sample of visualizing Compression and Decompression attempts, alongside the overall Compression Success Ratio graph:



MySQL Command/Handler Compare Dashboard

We have introduced a new dashboard that lets you do side-by-side comparison of Command (Com_*) and Handler statistics. A common use case would be to compare servers that share a similar workload, for example across MySQL instances in a pool of replicated slaves. In this example I am comparing two servers under identical sysbench load, but exhibiting slightly different performance characteristics:



The number of servers you can select for comparison is unbounded, but depending on the screen resolution you might want to limit to 3 at a time for a 1080 screen size.

Issues in this release

New Features & Improvements

- PMM-2519: Display Visual Explain in Query Analytics
- PMM-2019: Add new Dashboard InnoDB Compression metrics
- PMM-2154: Add new Dashboard Compare Commands and Handler statistics
- PMM-2530: Add timeout flags to mongodb_exporter (thank you [unguiculus](#) for your contribution!)
- PMM-2569: Update the MySQL Golang driver for MySQL 8 compatibility
- PMM-2561: Update to Grafana 5.1.3
- PMM-2465: Improve pmm-admin debug output
- PMM-2520: Explain Missing Charts from MySQL Dashboards
- PMM-2119: Improve Query Analytics messaging when Host = All is passed
- PMM-1956: Implement connection checking in mongodb_exporter

Bug fixes

- PMM-1704: Unable to connect to AtlasDB MongoDB
- PMM-1950: pmm-admin (mongodb:metrics) doesn't work well with SSL secured mongodb server
- PMM-2134: rds_exporter exports memory in Kb with node_exporter labels which are in bytes
- PMM-2157: Cannot connect to MongoDB using URI style
- PMM-2175: Grafana singlestat doesn't use consistent colour when unit is of type Time
- PMM-2474: Data resolution on Dashboards became 15sec interval instead of 1sec
- PMM-2581: Improve Travis CI tests by addressing pmm-admin check-network Time Drift
- PMM-2582: Unable to scroll on “_PMM Add Instance” page when many RDS instances exist in an AWS account
- PMM-2596: Set fixed height for panel content in PMM Add Instances
- PMM-2600: InnoDB Checkpoint Age does not show data for MySQL
- PMM-2620: Fix balancerIsEnabled & balancerChunksBalanced values
- PMM-2634: pmm-admin cannot create user for MySQL 8
- PMM-2635: Improve error message while adding metrics beyond “exit status 1”

Known Issues

- PMM-2639: mysql:metrics does not work on Ubuntu 18.04

Percona Monitoring and Management 1.11.0

Date May, 23 2018

For more information about this release, see the [release announcement](#).

In PMM Release 1.11.0, we're delivering the following changes:

- Configurable MySQL Slow Log Rotation - Enable or Disable rotation, and specify how many files to keep on disk
- Predictable Graphs - We've updated our formulas to use aggregation functions over time for more reliable graphs
- MySQL Exporter Parsing of my.cnf - We've improved how we read my.cnf
- Annotation improvements - Passing multiple strings results in single annotation being written

The issues in the release includes 1 new features & improvements, and 9 bugs fixed.

MySQL Slow Log Rotation Improvements

We spent some time this release going over how we handle MySQL's Slow Log rotation logic. Query Analytics requires that slow logging be enabled (either to file, or to PERFORMANCE_SCHEMA) and what we found was that Users of Percona Server for MySQL overwhelmingly choose logging to a file in order to take advantage of `log_slow_verbosity` which provides [enhanced InnoDB Usage information](#). The challenge however with MySQL's Slow Log is that it is very verbose and thus the number one concern is disk space - PMM strives to Do No Harm and so doing MySQL Slow Log Rotation was a natural fit, but until this release we were very strict and hadn't enabled any configuration of these parameters.

Percona Server for MySQL Users have long known about [Slow Query Log Rotation](#) and Expiration, but until now had no way of using the in-built Percona Server feature while ensuring that PMM wasn't missing any queries from the Slow Log during file rotation. Or perhaps your use case is that you want to do Slow Log Rotation using logrotate or some other facility. Today with Release 1.11 this is now possible!

We have made two significant changes:

- You can now specify the number of Slow Log files to remain on disk, and let PMM handle deleting the oldest files first. **Default remains unchanged - 1 Slow Log to remain on disk.**
- Slow Log rotation can now be disabled, for example if you want to manage rotation using logrotate or Percona Server for MySQL Slow Query Log Rotation and Expiration. **Default remains unchanged - Slow Log Rotation is ON.**

You specify each of these two new controls when setting up the MySQL service. The following example specifies that 5 Slow Log files should remain on disk:

```
$ pmm-admin add mysql ... --retain-slow-logs=5
```

While the following example specifies that Slow Log rotation is to be disabled (flag value of false), with the assumption that you will perform your own Slow Log Rotation:

```
$ pmm-admin add mysql ... --slow-log-rotation=false
```

We do not currently support modifying option parameters for an existing service definition. This means you must remove, then re-add the service and including the new options.

We are including a logrotate script in this post to get you started, and it is designed to keep 30 copies of Slow Logs at 1GB each. Note that you will need to update the Slow Log location, and ensure a MySQL User Account with SUPER, RELOAD are used for this script to successfully execute.

Example logrotate

```

/var/mysql/mysql-slow.log {
    nocompress
    create 660 mysql mysql
    size 1G
    dateext
    missingok
    notifempty
    sharedscripts
    postrotate
        /bin/mysql -e 'SELECT @@global.long_query_time INTO @LQT_SAVE; SET_
↪GLOBAL long_query_time=2000; SELECT SLEEP(2); FLUSH SLOW LOGS; SELECT SLEEP(2); SET_
↪GLOBAL long_query_time=@LQT_SAVE;'
    endscript
    rotate 30
}

```

Predictable Graphs

We've updated the logic on 4 Dashboards to better handle predictability and also zooming to look at shorter time ranges. For example, refreshing PXC/Galera Graphs prior to 1.11 led to graphs spiking at different points during the metric series. We've reviewed each of these Graphs and their corresponding queries and added in `<aggregation>_over_time()` functions so that Graphs display a consistent view of the metric series. This also improves your ability to drill in on the Dashboards, so that no matter how short your time range, you will still observe the same spikes and troughs in your metric series. The four Dashboards affected by this improvement are:

- Home Dashboard
- PXC/Galera Graphs Dashboard
- MySQL Overview Dashboard
- MySQL InnoDB Metrics Dashboard
- MySQL Exporter Parsing of my.cnf

In earlier releases, the MySQL Exporter expected only key=value type flags and would ignore options without values (i.e. `disable-auto-rehash`), and could sometimes read the wrong section of the my.cnf file. We've updated the parsing engine to be more MySQL compatible.

Annotation Improvements

Annotations permit the display of an event on all dashboards in PMM. Users reported that passing more than one string to `pmm-admin annotate` would generate an error, so we updated the parsing logic to assume all strings passed during Annotation creation generates a single Annotation event. Previously you needed to enclose your strings in quotes so that it would be parsed as a single string.

Issues in this release

New Features & Improvements

- [PMM-2432](#): Configurable MySQL Slow Log File Rotation

Bug fixes

- PMM-1187: Graphs breaks at tight resolution
- PMM-2362: Explain is a part of query
- PMM-2399: RPM for PMM Server is missing some files
- PMM-2407: Menu items are not visible on PMM QAN dashboard
- PMM-2469: Parsing of a valid my.cnf can break the mysqld_exporter
- PMM-2479: PXC/Galera Cluster Overview dashboard: typo in metric names
- PMM-2484: PXC/Galera Graphs display unpredictable results each time they are refreshed
- PMM-2503: Wrong Innodb Adaptive Hash Index Statistics
- PMM-2513: QAN-agent always changes `max_slowlog_size` to 0
- PMM-2514: `pmm-admin annotate help` - fix typos
- PMM-2515: `pmm-admin annotate` - more than 1 annotation

How to get PMM

PMM is available for installation using three methods:

- On Docker Hub – `docker pull percona/pmm-server` <https://www.percona.com/doc/percona-monitoring-and-management/deploy/server/docker.html>
- AWS Marketplace – <https://www.percona.com/doc/percona-monitoring-and-management/deploy/server/ami.html>
- Open Virtualization Format (OVF) – <https://www.percona.com/doc/percona-monitoring-and-management/deploy/server/virtual-appliance.html>

Help us improve our software quality by reporting any bugs you encounter using our [bug tracking system](#).

Percona Monitoring and Management 1.10.0

Date April 20, 2018

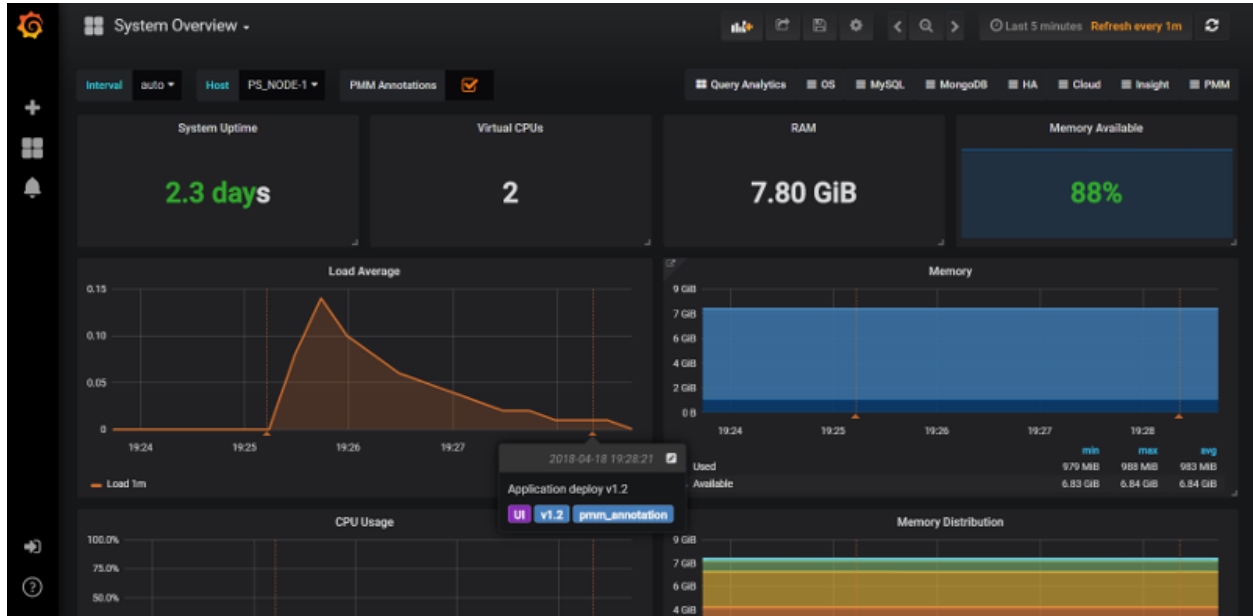
Percona announces the release of *Percona Monitoring and Management* 1.10.0. *PMM* (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL* and *MongoDB* performance. You can run *PMM* in your own environment for maximum security and reliability. It provides thorough time-based analysis for *MySQL* and *MongoDB* servers to ensure that your data works as efficiently as possible. We focused mainly on two features in 1.10.0, but there are also several notable improvements worth highlighting:

- *Annotations* - Record and display application events as Annotations using `pmm-admin annotate`
- *Grafana 5.0* - Improved visualization effects
- *Switching between Dashboards* - Now switching dashboards
- *New PXC Galera Replication Latency Graphs* - Added Galera Replication Latency graphs on PXC Overview dashboard with consistent colours

The *Issues in this release* includes 4 new features & improvements, and 8 bugs fixed.

Annotations

Application events are one of the contributors to changes in database performance characteristics, and in this release PMM now supports receiving events and displaying them as Annotations using the new command `pmm-admin annotate`. A recent Percona survey reveals that database and DevOps engineers highly value visibility into the application layer. By displaying application events on top of your PMM graphs, engineers can now correlate application events to database events (common cases: application deploys, outages, and upgrades) against Database and System level metric changes.



Usage

For example, you have an application deployment that just completed to version 1.2, and it affects the UI only, so you want to set tags for the version and interface impacted:

```
$ pmm-admin annotate "Application deploy v1.2" --tags "UI, v1.2"
```

Using the optional `--tags` option allows you to filter which Annotations are displayed on the dashboard via a toggle option. Read more about Annotations utilization in the [PMM documentation](#).

Grafana 5.0

We're extremely pleased to see Grafana ship 5.0 and we were fortunate enough to be at Grafanacon, including Percona's very own [@Dimitri Vanoverbeke](#) (Dim0) who presented What we Learned Integrating Grafana and Prometheus!

Included in Grafana 5.0 are a number of dramatic improvements, which in future PMM releases we plan to extend our usage of each feature, and the one we like best is the virtually unlimited way you can size and shape graphs. No longer are you bound by panel constraints to keep all objects at the same fixed height! This improvement indirectly addresses the visualization error in PMM Server where some graphs would appear to be on two lines and ended up wasting screen space.



Switching Between Dashboards

PMM now allows you to navigate between dashboards while maintaining the same host under observation, so that for example you can start on *MySQL Overview* looking at host serverA, switch to *MySQL InnoDB Advanced* dashboard and continue looking at serverA, thus saving you a few clicks in the interface.

New PXC Galera Replication Latency Graphs

We have added new PXC Replication Latency graphs on our *PXC Galera Cluster Overview* dashboard so that you can compare latency across all members in a cluster in one view.



Issues in this release

New Features and Improvements

- PMM-2293: Add the *Galera Replication Latency* graph to the *PXC/Galera Cluster overview* dashboard.
- PMM-2295: Improve colour selection on the *PXC/Galera Cluster Overview* dashboard
- PMM-2330: Application Annotations
- PMM-2332: Grafana 5 update

Bug fixes

- PMM-2311: Fix mis-alignment in Query Analytics Metrics table
- PMM-2341: Typo in text on password page of OVF

- PMM-2359: Trim leading and trailing whitespaces for all fields on AWS/OVF Installation wizard
- PMM-2360: Include a “What’s new?” link for Update widget
- PMM-2346: Arithmetic on InnoDB AHI Graphs are invalid
- PMM-2364: QPS are wrong in QAN
- PMM-2388: Query Analytics does not render fingerprint section in some cases
- PMM-2371: Pass host when switching between Dashboards

See also:

How to get PMM

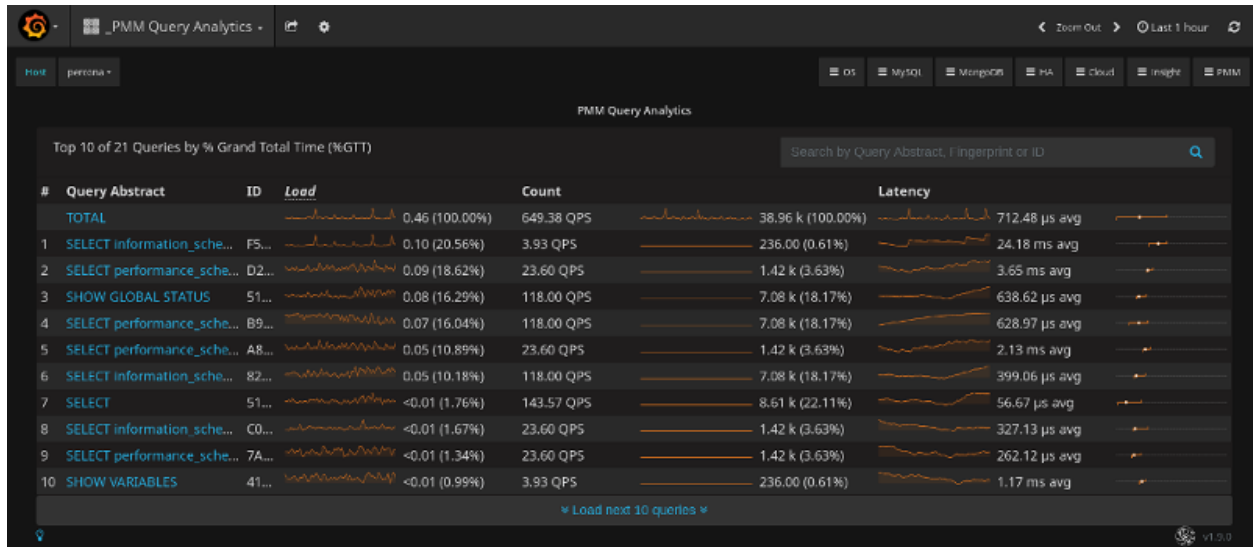
Deploying Percona Monitoring and Management

Percona Monitoring and Management 1.9.1

Date April 12, 2018

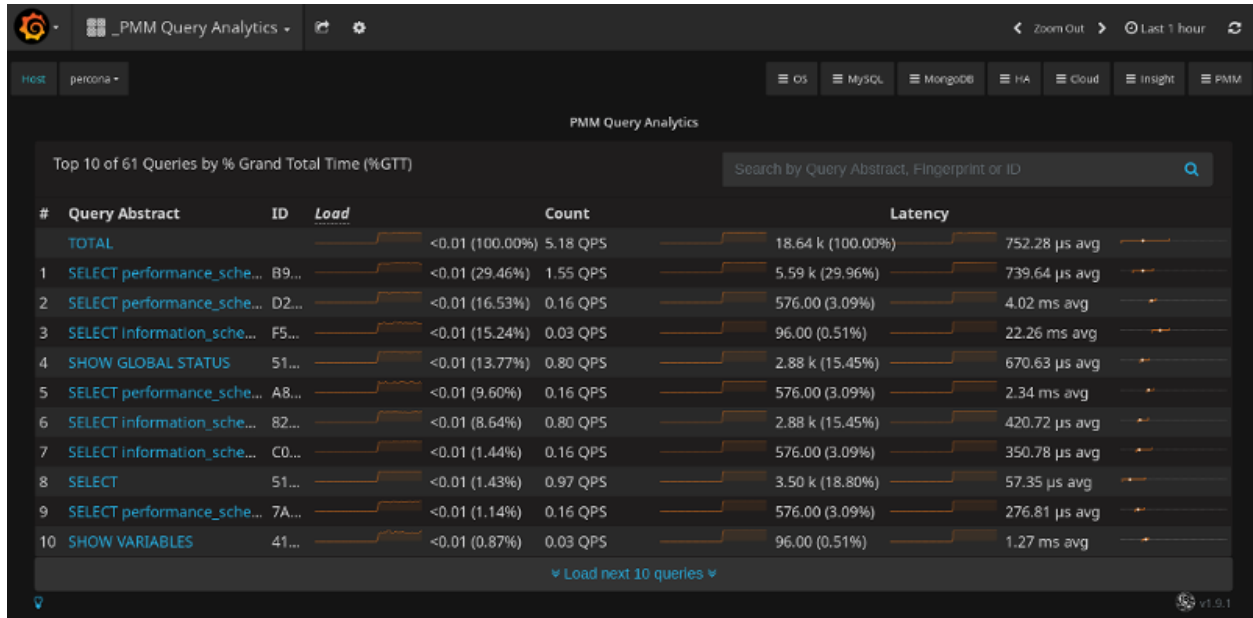
This release contains bug fixes only and supersedes *Percona Monitoring and Management 1.9.0*. This release effectively solves the problem in QAN when the *Count* column actually displayed the number of queries per minute, not per second, as the user would expect. The following screenshot demonstrates the problem. The value of the *Count* column for the **TOTAL** row is **649.38 QPS** (queries per second). The total number **38.96 k** (38960) is only sixty times greater than the reported value of QPS. Thus, queries were counted for each minute within the selected time range of Last 1 hour.

Query Analytics in PMM version 1.9.0.



The corrected version of QAN in PMM 1.9.1 shows that queries are now counted per second. The total number of queries is $60 * 60$ greater than the value of *QPS*, as should be expected for the chosen time range.

Query Analytics in PMM version 1.9.1.



Bug fixes

- PMM-2364: QPS are wrong in QAN

Percona Monitoring and Management 1.9.0

Date April 4, 2018

There are a number of significant updates in 1.9.0 that we hope you will like, some of the key highlights include:

- Faster loading of the index page: We have enabled performance optimizations using **gzip** and **HTTP2**.
- AWS improvements: We have added metrics from CloudWatch RDS to 6 dashboards, as well as changed our AWS add instance workflow, and made some changes to credentials handling.
- Percona Snapshot Server: If you are a Percona Customer you can now securely share your dashboards with Percona Engineers.
- Exporting PMM Server logs: Retrieve logs from PMM Server for troubleshooting using single button-click, avoiding the need to log in manually to the docker container.
- Low RAM support: We have reduced the memory requirement so PMM Server will run on systems with 512MB
- Dashboard improvements: We ave changed MongoDB instance identification for MongoDB graphs, and set maximum graph Y-axis on Prometheus Exporter Status dashboard

AWS Improvements

CloudWatch RDS metrics

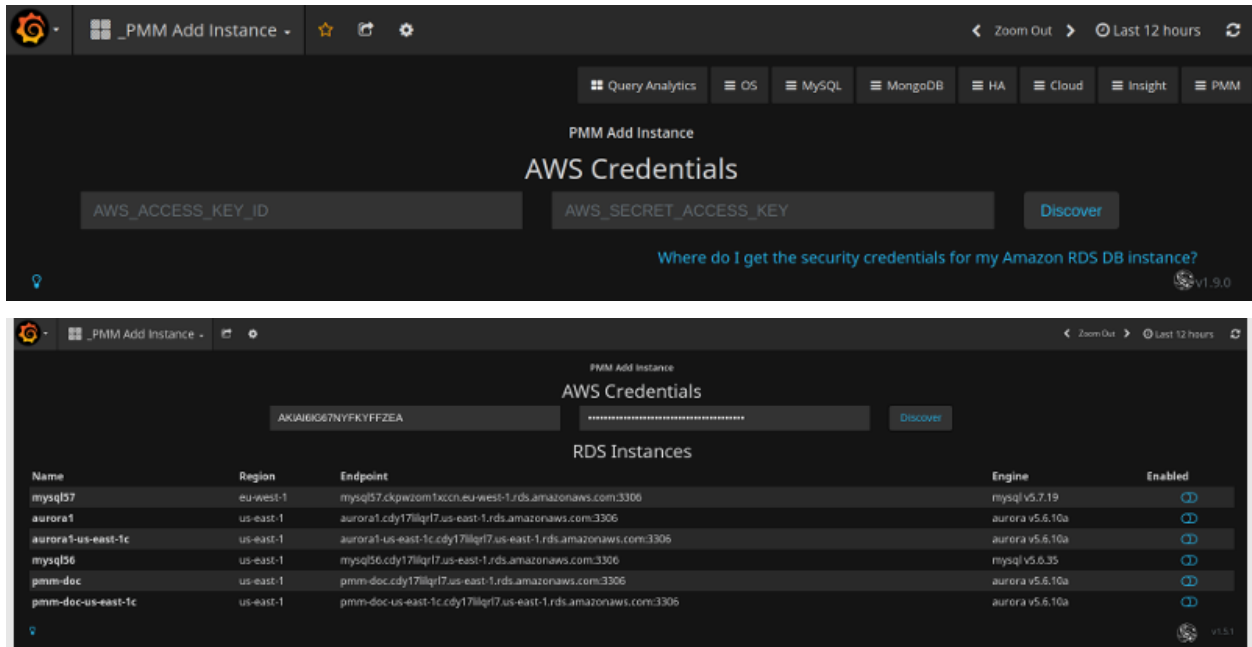
Since we are already consuming Amazon Cloudwatch metrics and persisting them in Prometheus, we have improved 6 node-specific dashboards to now display Amazon RDS node-level metrics:

- Cross_Server (Network Traffic)

- Disk Performance (Disk Latency)
- Home Dashboard (Network IO)
- MySQL Overview (Disk Latency, Network traffic)
- Summary Dashboard (Network Traffic)
- System Overview (Network Traffic)

AWS Add Instance changes

We have changed our AWS add instance interface and workflow to be more clear on information needed to add an Amazon Aurora MySQL or Amazon RDS MySQL instance. We have provided some clarity on how to locate your AWS credentials.



AWS Settings

We have improved our documentation to highlight connectivity best practices, and authentication options - IAM Role or IAM User Access Key.

Enabling Enhanced Monitoring

Monitoring

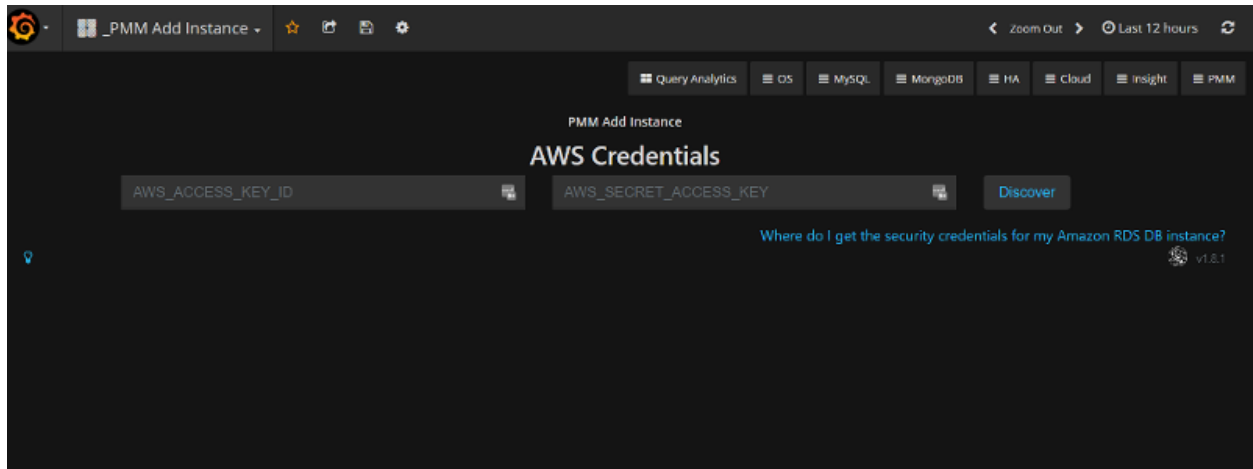
Enhanced monitoring

Enable enhanced monitoring
Enhanced monitoring metrics are useful when you want to see how different processes or threads use the CPU.

Disable enhanced monitoring

Monitoring Role Granularity

Credentials Screen



Low RAM Support

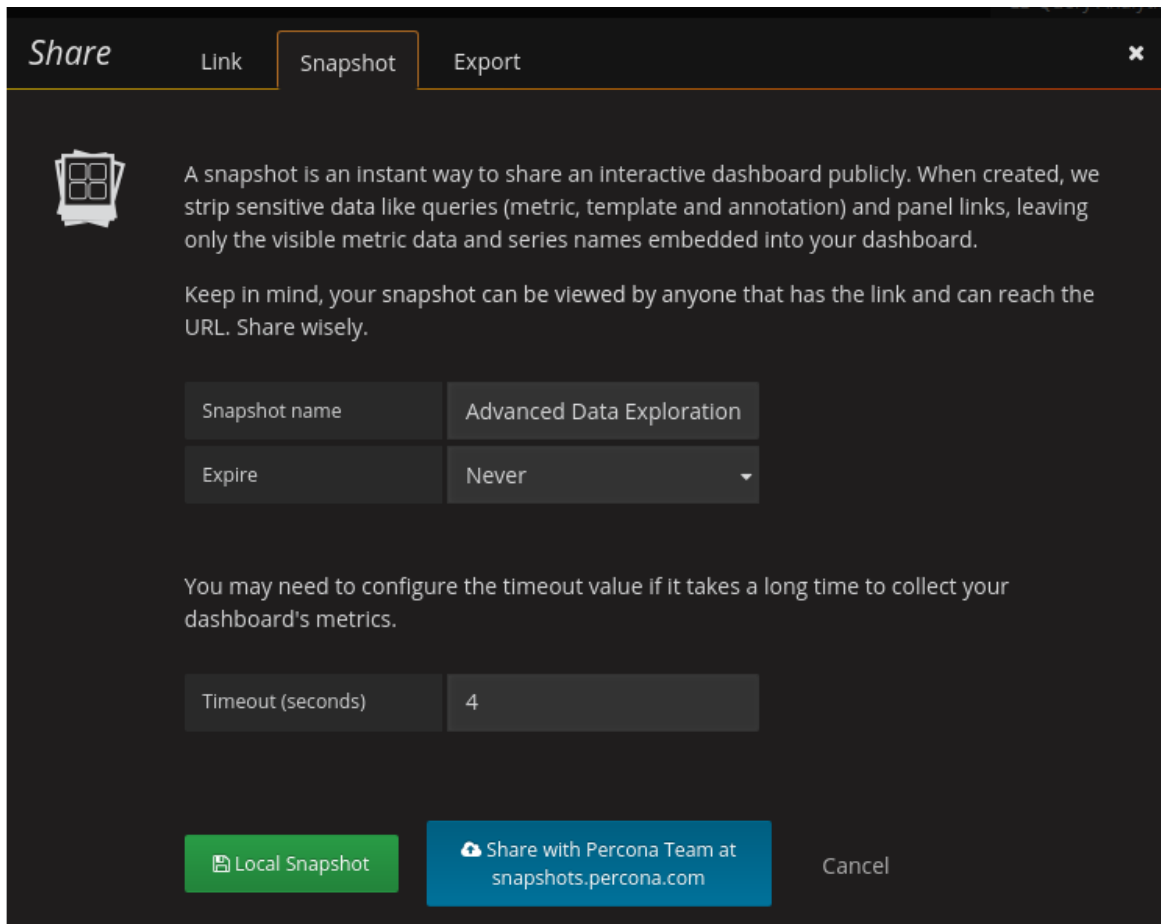
You can now run PMM Server on instances with memory as low as 512MB RAM, which means you can deploy to the free tier of many cloud providers if you want to experiment with PMM. Our memory calculation is now:

```
METRICS_MEMORY_MULTIPLIED=$(( ( ${MEMORY_AVAILABLE} - 256*1024*1024 ) / 100 * 40 ))
if [[ $METRICS_MEMORY_MULTIPLIED < $((128*1024*1024)) ]]; then
    METRICS_MEMORY_MULTIPLIED=$((128*1024*1024))
fi
```


Percona Snapshot Server

Snapshots are a way of sharing PMM dashboards via a link to individuals who do not normally have access to your PMM Server. If you are a Percona Customer you can now securely share your dashboards with Percona Engineers. We have replaced the button for sharing to the Grafana publicly hosted platform onto one administered by Percona. Your dashboard will be written to Percona snapshots and only Percona Engineers will be able to retrieve the data. We will

be expiring old snapshots automatically at 90 days, but when sharing you will have the option to configure a shorter retention period.



Share Link **Snapshot** Export ✕

 A snapshot is an instant way to share an interactive dashboard publicly. When created, we strip sensitive data like queries (metric, template and annotation) and panel links, leaving only the visible metric data and series names embedded into your dashboard.

Keep in mind, your snapshot can be viewed by anyone that has the link and can reach the URL. Share wisely.

Snapshot name	Advanced Data Exploration
Expire	Never ▾

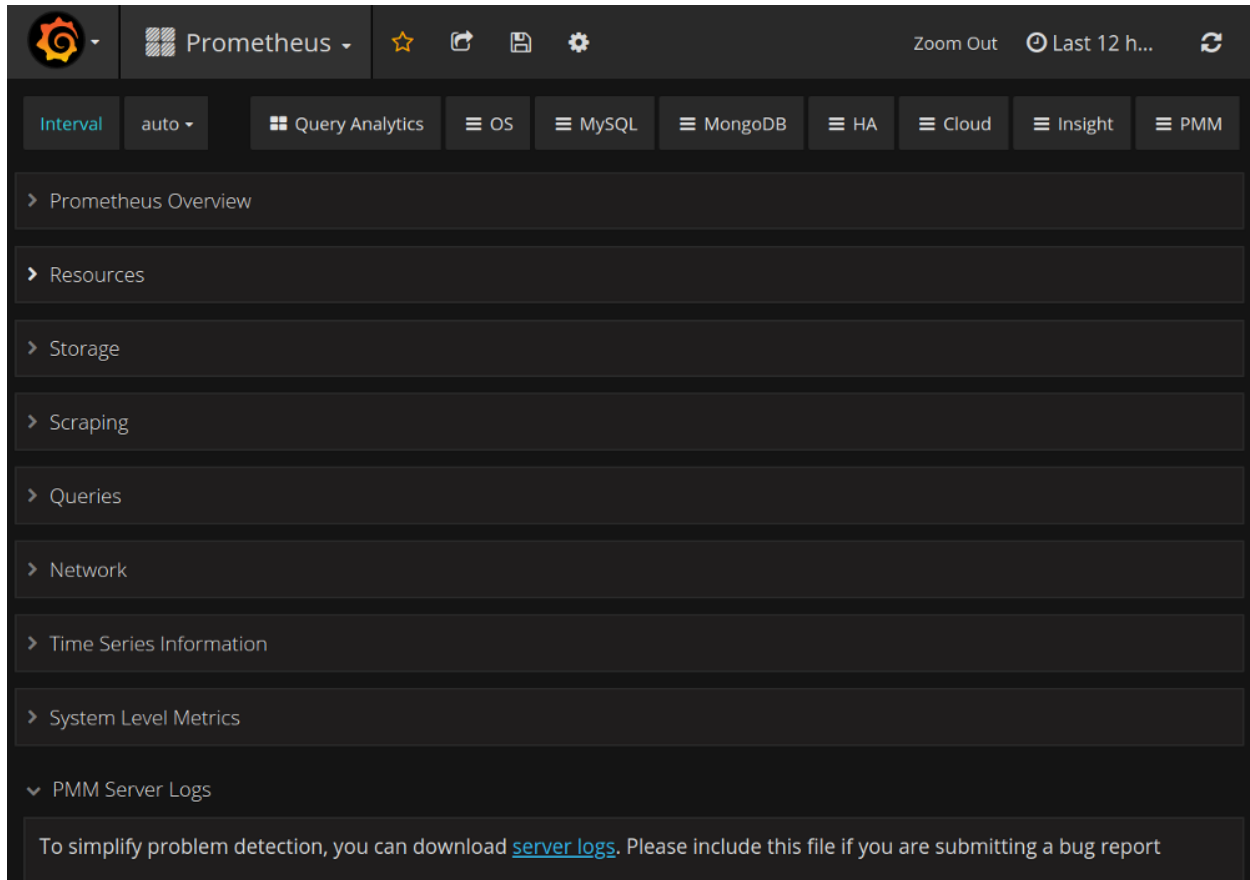
You may need to configure the timeout value if it takes a long time to collect your dashboard's metrics.

Timeout (seconds)	4
-------------------	---

Local Snapshot Share with Percona Team at snapshots.percona.com Cancel

Export of PMM Server Logs

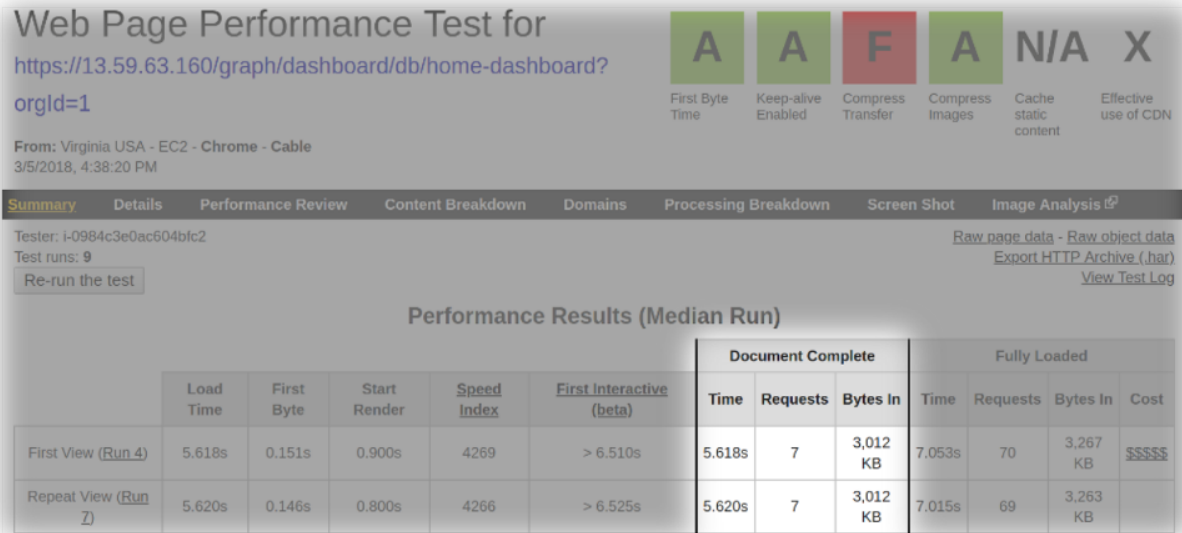
In this release, the logs from PMM Server can be exported using single button-click, avoiding the need to log in manually to the docker container. This simplifies the troubleshooting process of a PMM Server, and especially for Percona Customers, this feature will provide a more consistent data gathering task that you will perform on behalf of requests from Percona Engineers.



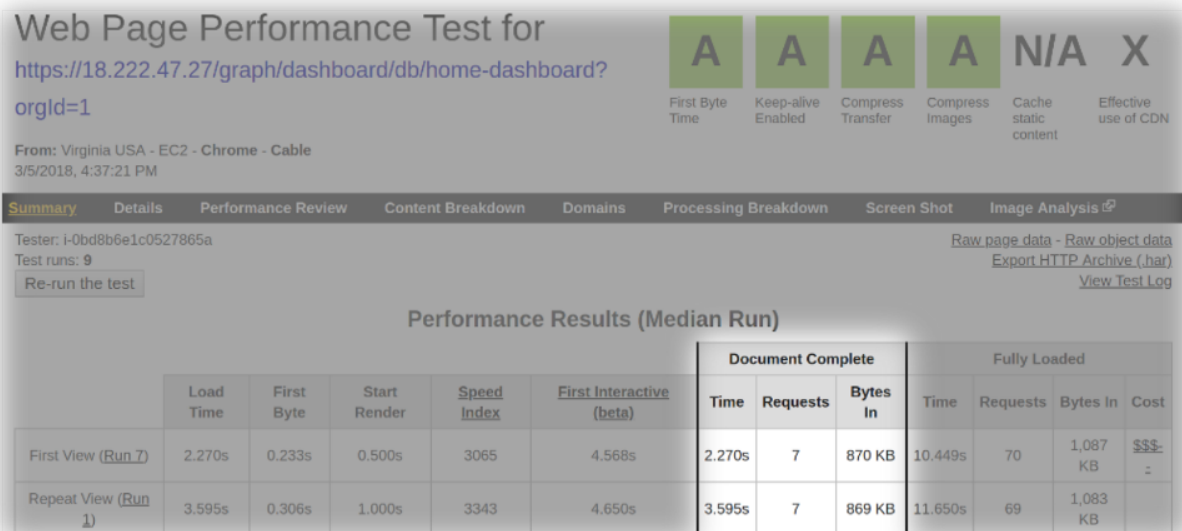
Faster Loading of the Index Page

In version 1.8.0, the index page was redesigned to reveal more useful information about the performance of your hosts as well as an immediate access to essential components of PMM, however the index page had to load much data dynamically resulting in a noticeably longer load time. In this release we enabled **gzip** and **HTTP2** to improve the load time of the index page. The following screenshots demonstrate the results of our tests on webpagetest.org where we reduce page load time by half. We will continue to look for opportunities to improve the performance of the index page and expect that when we upgrade to Prometheus 2 we will see another improvement.

The load time of the index page of PMM version 1.8.0



The load time of the index page of PMM version 1.9.0



Issues in this release

New Features

- PMM-781: Plot new PXC 5.7.17, 5.7.18 status variables on new graphs for PXC Galera, PXC Overview dashboards
- PMM-1274: Export PMM Server logs as zip file to the browser

- PMM-2058: Percona Snapshot Server

Improvements

- PMM-1587: Use `mongodb_up` variable for the MongoDB Overview dashboard to identify if a host is MongoDB.
- PMM-1788: AWS Credentials form changes
- PMM-1823: AWS Install wizard improvements
- PMM-2010: System dashboards update to be compatible with RDS nodes
- PMM-2118: Update grafana config for metric series that will not go above 1.0
- PMM-2215: PMM Web speed improvements
- PMM-2216: PMM can now be started on systems without memory limit capabilities in the kernel
- PMM-2217: PMM Server can now run in Docker with 512 Mb memory
- PMM-2252: Better handling of variables in the navigation menu

Bug fixes

- PMM-605: `pt-mysql-summary` requires additional configuration
- PMM-941: `ParseSocketFromNetstat` finds an incorrect socket
- PMM-948: Wrong load reported by QAN due to mis-alignment of time intervals
- PMM-1486: MySQL passwords containing the dollar sign (\$) were not processed properly.
- PMM-1905: In QAN, the Explain command could fail in some cases.
- PMM-2090: Minor formatting issues in QAN
- PMM-2214: Setting Send real query examples for Query Analytic OFF still shows the real query in example.
- PMM-2221: no Rate of Scrapes for MySQL & MySQL Errors
- PMM-2224: Exporter CPU Usage glitches
- PMM-2227: Auto Refresh for dashboards
- PMM-2243: Long host names in Grafana dashboards are not displayed correctly
- PMM-2257: PXC/galera cluster overview Flow control paused time has a percentage glitch
- PMM-2282: No data is displayed on dashboards for OVA images
- PMM-2296: The `mysql:metrics` service will not start on Ubuntu LTS 16.04

Percona Monitoring and Management 1.8.1

Date March 06, 2018

Percona announces the release of *Percona Monitoring and Management 1.8.1*. *PMM* (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL* and *MongoDB* performance. You can run *PMM* 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..

This release contains bug fixes only and supersedes *Percona Monitoring and Management* 1.8.0.

Improvements

- **PMM-2051:** The ProxySQL Overview dashboard enables selecting more than one host group in the *Hostgroup* field.
- **PMM-2163:** Dashboards based on the *rds_exporter* now use the *node_cpu_average* metric instead of *node_cpu*

Bug fixes

- **PMM-854:** In some cases, databases and tables could be detected incorrectly
- **PMM-1745:** For some queries, *Query Abstract* showed incorrect database name in QAN.
- **PMM-1928:** In some cases, *PMM Query Analytics* added a wrong schema
- **PMM-2014:** *PMM Query Analytics* could incorrectly include a schema from another server
- **PMM-2082:** The *PMM Query Analytics Settings* dashboard had minor user interface problems.
- **PMM-2122:** The time selector in *PMM Query Analytics* showed time in the local timezone while time values in the *Query Abstract*: were in the UTC format.
- **PMM-2127:** There was a typo in the QAN interface when there was no data
- **PMM-2129:** In some cases, QAN could show an incorrect fingerprint if the query contained no table.
- **PMM-2171:** The JSON section in *PMM Query Analytics* was displayed incorrectly
- **PMM-2172:** The *CPU Usage* metrics was not consistent in the *System Summary* dashboard
- **PMM-2173:** Summary values were inconsistent
- **PMM-2174:** Amazon Aurora nodes were not shown in the System Overview dashboard
- **PMM-2176:** Lengthy queries were not displayed correctly in *PMM Query Analytics*.
- **PMM-2177:** The *Incorrect Table name* error appeared on the first load of *PMM Query Analytics*.
- **PMM-2184:** When port forwarding was used with Docker, the permanent redirects would break

Percona Monitoring and Management 1.8.0

Date February 27, 2018

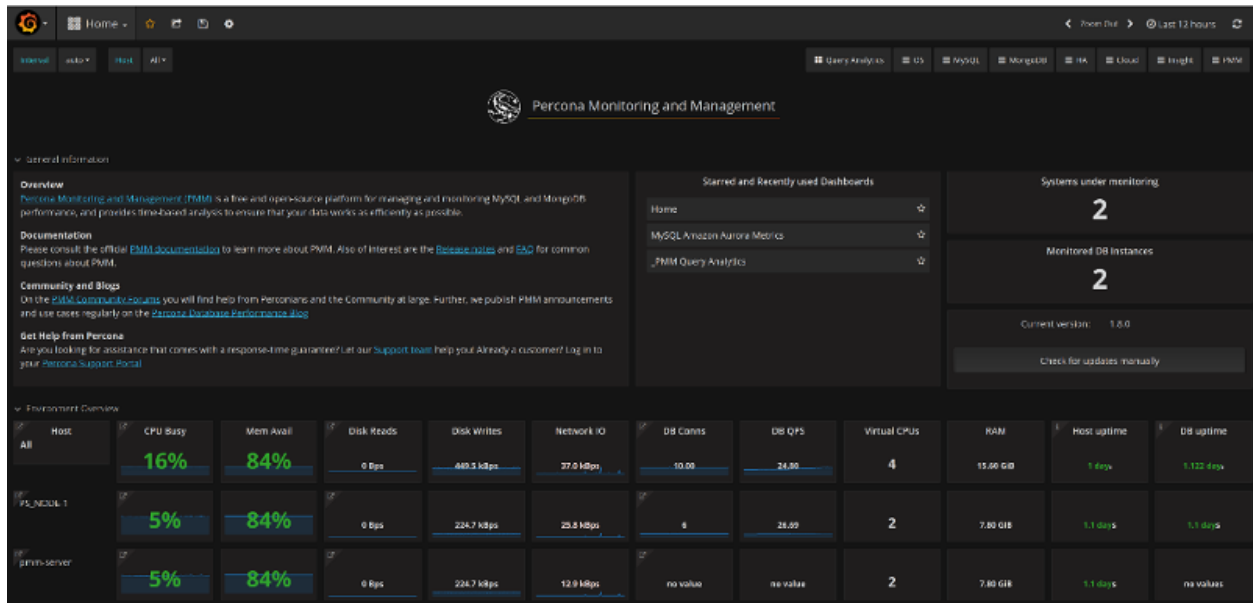
Percona announces the release of *Percona Monitoring and Management* 1.8.0. *PMM* (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL* and *MongoDB* performance. You can run *PMM* in your own environment for maximum security and reliability. It provides thorough time-based analysis for *MySQL* and *MongoDB* servers to ensure that your data works as efficiently as possible..

This release introduces many improvements in the user interface and optimizations of performance.

New landing page

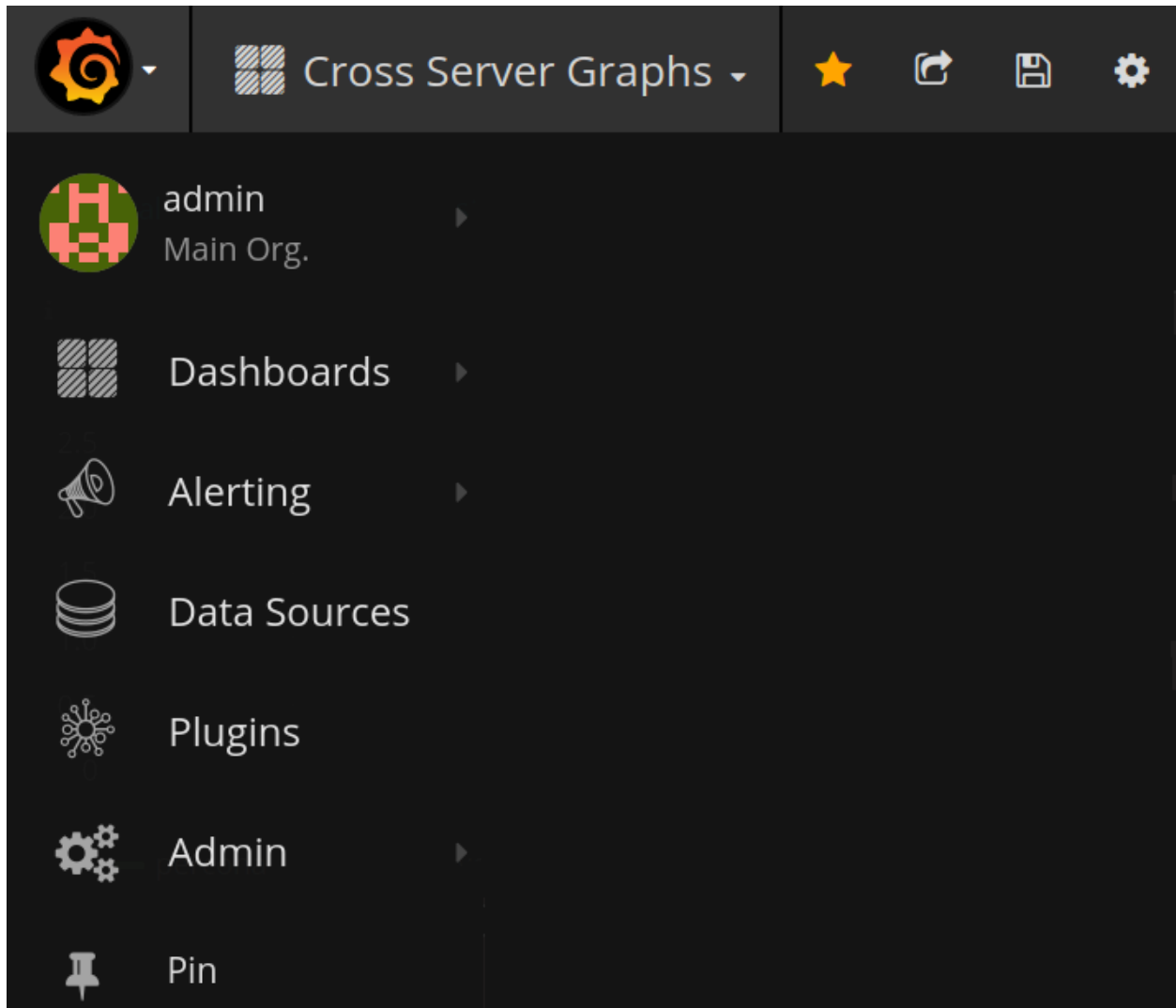
The home page has been removed in favor of the *Home* dashboard to make the look and feel of *PMM* web interface more consistent. The *Home* dashboard gives a general overview of your system and provides links to useful resources.

You can immediately go to a dashboard of your choice either by using the conventional *Dashboard Dropdown* at the top of the page or the new dashboard menu.



Improved dashboard menu layout

The new dashboard menu provides a convenient grouping of dashboards making the switching between dashboards easy: select a group, such as MySQL or OS, and choose from a short list in the menu that opens. The new dashboard menu is available from every dashboard in PMM.



Automatic Memory Configuration

This release also features improved memory usage on the host system in PMM Server. By default, PMM now automatically scales its use with system size rather than using static amount of memory, unless you explicitly set the maximum value (by using the `METRICS_MEMORY` in Docker, for example).

New Features

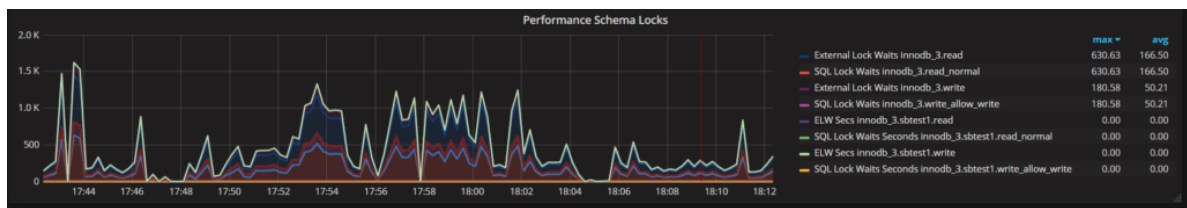
- **PMM-1145:** Move Percona dashboards to Grafana plugin. We have updated the Percona dashboards to use the Grafana plugin system in order to simplify upgrades.
- **PMM-1470:** Implement a custom Prometheus `rds_exporter` to collect Amazon Cloudwatch metrics. You can now explore Amazon Cloudwatch metrics from Prometheus, for example using the *Advanced Data Exploration* dashboard. In prior versions we were using Amazon Cloudwatch API calls directly.
- **PMM-2046:** Configure memory consumption automatically for PMM Server. In this release we improved memory usage in PMM Server. By default, PMM now automatically scales its use with system size rather than using a static amount of memory. You can override this behaviour (for example, if you need to share resources

with other containers) by using the `METRICS_MEMORY` option in Docker. You can use the old static size of memory by running:

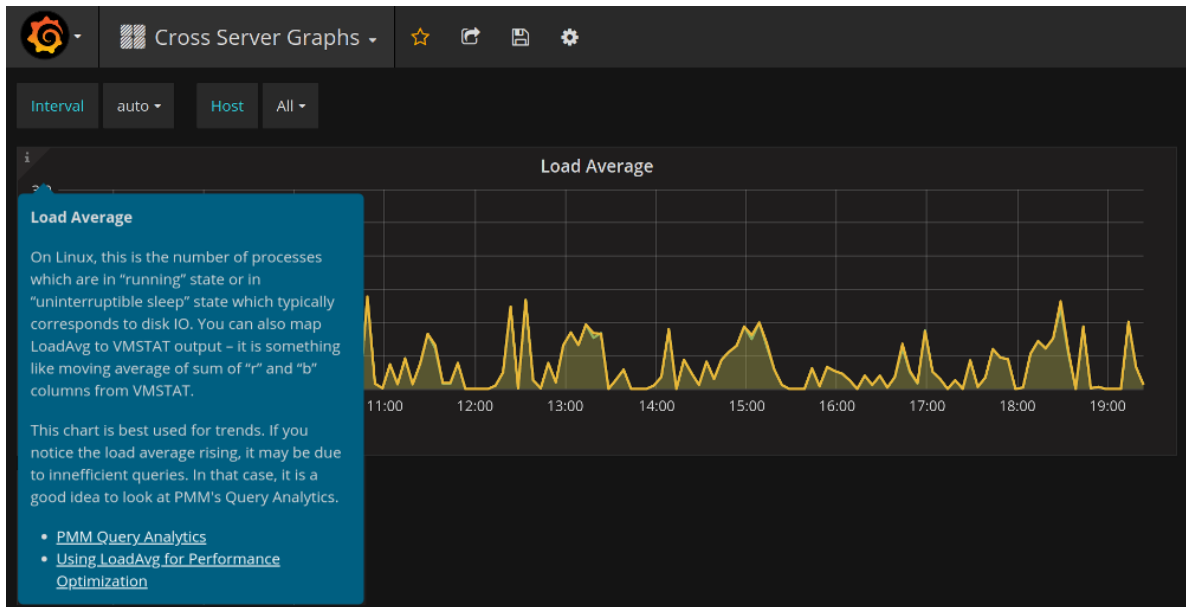
```
$ docker ... -e METRICS_MEMORY=786432
```

Improvements

- **PMM-1425:** Replace the Home page with the *Home* dashboard. The home page has been removed in favor of the *Home* dashboard to make the look and feel of PMM web interface more consistent. The *Home* dashboard gives a general overview of your system and provides links to useful resources. You can immediately go to a dashboard of your choice either by using the conventional *Dashboard Dropdown* at the top of the page or the new dashboard menu.
- **PMM-1738:** Add the *Dashboard* menu to make the discovery of dashboards easier.
- **PMM-1976:** Apply consistent rules for computing CPU metrics. We updated the query that generates CPU utilization for the *MySQL Overview*, *Summary* and *System Overview* dashboards as before we were being inconsistent in our determination of CPU.
- **PMM-2007:** Squash new Docker layers for a smaller total size We reduced the size of the Docker container from ~475 MB to ~350 MB
- **PMM-1711:** Add SQL and External Locks graphs to the *MySQL Performance Schema* dashboard. Added new graph to the *MySQL Performance Schema* dashboard to show locks from perspective of count of events and from time.



- **PMM-1763:** Update the forked Prometheus exporter `mysqld_exporter` to the [latest upstream version](#) to include recent bug fixes.
- **PMM-2004:** Add the `/ping` alias to `nginx` to enable a basic health check URL to the PMM Server API.
- **PMM-1365:** Provide descriptions of all metrics in the *Cross Server Graphs* dashboard. We have added metric descriptions in order to improve the understanding of the displayed metric series and to provide links to documentation and blog articles, beginning with the *Cross Server Graphs*.



- PMM-1343: Provide descriptions of all metrics in the *Summary* dashboard
- PMM-1364: Provide descriptions of all metrics in the *MySQL InnoDB Metrics* dashboard
- PMM-1994: Provide descriptions of all metrics in the *MySQL Amazon Aurora* dashboard
- :pmmbug:“
- PMM-1922: *pmm-server* as instance for all components on PMM Server
- PMM-2005: Upgrade Orchestrator to the latest stable release
- PMM-1957: Build Go 1.9.4. Updated Go to 1.9.4 which resolves a security problem in the go get command (For more information, see [Golang issue #23672](#))
- PMM-2002: In QAN, elements of the JSON output can be collapsed; the text in the *Create Table* block can be copied.
- PMM-2128: Improved formula for *Saturation Metrics* on the *System Overview* dashboard.

Bug fixes

- PMM-2124: The number of the new version is missing in the notification of the successful update of PMM
- PMM-1453: mysqld_exporter wrong mapping of innodb_buffer_pool_pages
- PMM-2029: Deadlock when adding external exporter
- PMM-1908: RDS MySQL nodes do not display Current QPS
- PMM-2100: rds_exporter crashed after running for several minutes
- PMM-1511: PXC cluster is not recognized correctly when MariaDB nodes are monitored
- PMM-1715: MongoDB QAN: Wrong Per Query Stats
- PMM-1892: Cannot detect tables in the Information Schema query
- PMM-1893: In QAN, the Tables element can be empty
- PMM-1941: Fingerprint storage bug

- **PMM-1933:** If some parts of collector got an error, whole collector is down
- **PMM-1934:** mysqld_exporter/collector/info_schema_auto_increment.go fails if there are same table names but with different cases
- **PMM-1951:** Regression in `pmm-admin add mysql:queries` only - not working
- **PMM-2142:** Wrong calculation of the CPU Busy parameter
- **PMM-2148:** rds_exporter node_cpu metric in percents but node_exporter values are in seconds

Percona Monitoring and Management 1.7.0

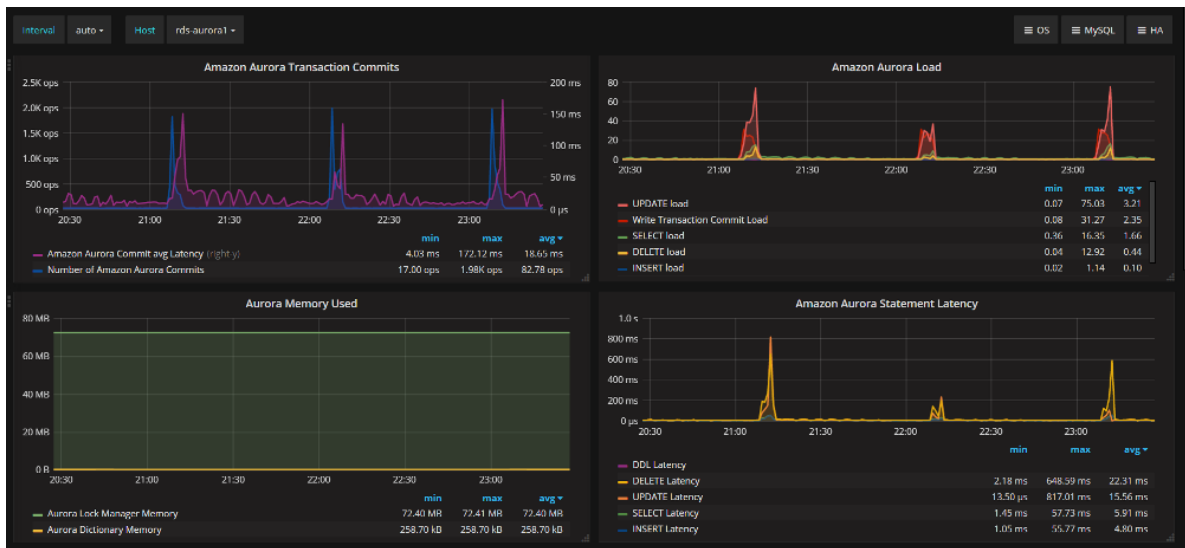
Date January 31, 2018

Percona announces the release of *Percona Monitoring and Management 1.7.0*. *PMM* (Percona Monitoring and Management) is a free and open-source platform for managing and monitoring *MySQL* and *MongoDB* performance. You can run *PMM* 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..

This release features improved support for external services, which enables a PMM Server to store and display metrics for any available Prometheus exporter. For example, you could deploy the `postgres_exporter` and use PMM external services feature to store PostgreSQL metrics in PMM. Immediately, you will see these new metrics in the *Advanced Data Exploration* dashboard. Then you could leverage many of the pre-developed PostgreSQL dashboards available on Grafana.com, and with a minimal amount of edits have a working PostgreSQL dashboard in PMM! Watch for an upcoming blog post to demonstrate a walk-through of this unlocked functionality.

New Features

- **PMM-1949:** New dashboard: *MySQL Amazon Aurora Metrics*.



Improvements

- **PMM-1712:** Improve external exporters to let you easily add data monitoring from an arbitrary Prometheus exporter you have running on your host.

- **PMM-1510:** Rename *swap in* and *swap out* labels to be more specific and help clearly see the direction of data flow for *Swap In* and *Swap Out*. The new labels are *Swap In (Reads)* and *Swap Out (Writes)* accordingly.
- **PMM-1966:** Remove Grafana from list of exporters on dashboard to eliminate confusion with existing Grafana in the list of exporters on the current version of the dashboard.
- **PMM-1974:** Add the *mongodb_up* in the Exporter Status dashboard. The new graph is added to maintain consistency of information about exporters. This is done based on new metrics implemented in **PMM-1586**.

Bug fixes

- **PMM-1967:** Inconsistent formulas in Prometheus dashboards.
- **PMM-1986:** Signing out with HTTP auth enabled leaves the browser *signed in*.

Percona Monitoring and Management 1.6.1

Date January 25, 2018

Percona announces the release of *Percona Monitoring and Management* 1.6.1. This release contains bug fixes only and supersedes *Percona Monitoring and Management* 1.6.0.

Bug fixes

- **PMM-1660:** QAN for MongoDB would not display data when authentication was enabled.
- **PMM-1822:** In Metrics Monitor, some tag names were incorrect in dashboards.
- **PMM-1832:** After upgrading to 1.5.2, it was not possible to disable an RDS instance on the *Add RDS instance* dashboard of Metrics Monitor.
- **PMM-1907:** In Metrics Monitor, the tooltip of the *Engine Uptime* metric referred to an incorrect unit of measure.
- **PMM-1964:** The same value of the *Exporter Uptime* metric could appear in different colors in different contexts.
- **PMM-1965:** In the *Prometheus Exporters Overview* dashboard of Metrics Monitor, the drill down links of some metrics could direct to a wrong host.

Percona Monitoring and Management 1.6.0

Date January 18, 2018

Percona announces the release of *Percona Monitoring and Management* 1.6.0. In this release, PMM Grafana metrics are made available in the *Advanced Data Exploration* dashboard. The integration with MyRocks has been improved and its data are now collected from *SHOW GLOBAL STATUS*.

The MongoDB exporter now features two new metrics: **mongodb_up** to inform if the MongoDB Server is running and **mongodb_scrape_errors_total** reporting the total number of errors when scraping MongoDB.

In this release, the performance of the `mongodb:metrics` monitoring service has been greatly improved.

PMM 1.6.0 also includes version 4.6.3 of Grafana which includes fixes to bugs in the alert list and in the alerting rules. More information is available in the

New Features

- **PMM-1773**: PMM Grafana specific metrics have been added to the *Advanced Data Exploration* dashboard.

Improvements

- **PMM-1485**: Updated MyRocks integration: MyRocks data is now collected entirely from **SHOW GLOBAL STATUS** and we have eliminated **SHOW ENGINE ROCKSDB STATUS** as a data source in **mysqld_exporter**.
- **PMM-1895**: Update Grafana to version 4.6.3:
 - Alert list: Now shows alert state changes even after adding manual annotations on dashboard #9951
 - Alerting: Fixes bug where rules evaluated as firing when all conditions was false and using OR operator. #9318
- **PMM-1586**: The **mongodb_exporter** exporter exposes two new metrics: **mongodb_up** informing if the MongoDB server is running and **mongodb_scrape_errors_total** informing the total number of times an error occurred when scraping MongoDB.
- **PMM-1764**: Various small **mongodb_exporter** improvements
- **PMM-1942**: Improved the consistency of using labels in all Prometheus related dashboards.
- **PMM-1936**: Updated the Prometheus dashboard in Metrics Monitor
- **PMM-1937**: Added the *CPU Utilization Details (Cores)* dashboard to Metrics Monitor.
- **PMM-1887**: Improved the help text for **pmm-admin** to provide more information about exporter options.
- **PMM-1939**: In Metrics Monitor, two new dashboards have been added to provide more information about Prometheus exporters. The *Prometheus Exporters Overview* dashboard provides a summary of how exporters utilize system resources and the *Prometheus Exporter Status* dashboard tracks the performance of each Prometheus exporter.

Bug fixes

- **PMM-1549**: Broken default auth db for `mongodb:queries`
- **PMM-1631**: In some cases percentage values were displayed incorrectly for MongoDB hosts.
- **PMM-1640**: RDS exporter: simplify configuration.
- **PMM-1760**: After the `mongodb:metrics` monitoring service was added, the usage of CPU considerably increased in QAN versions 1.4.1 through 1.5.3.

In our limited testing we observed the following results:

PMM Version	CPU Usage
1.5.0	95%
1.5.3	85%
1.6.0	1%

- **PMM-1815**: QAN could show data for a MySQL host when a MongoDB host was selected.
- **PMM-1888**: In QAN, query metrics were not loaded when the QAN page was refreshed.
- **PMM-1898**: In QAN, the *Per Query Stats* graph displayed incorrect values for MongoDB
- **PMM-1796**: In Metrics Monitor, The *Top Process States Hourly* graph from the the *MySQL Overview* dashboard showed incorrect data.

- **PMM-1777**: In QAN, the *Load* column could display incorrect data.
- **PMM-1744**: The error *Please provide AWS access credentials error* appeared although the provided credentials could be processed successfully.
- **PMM-1676**: In preparation for migration to Prometheus 2.0 we have updated the *System Overview* dashboard for compatibility.
- **PMM-1920**: Some standard MySQL metrics were missing from the **mysqld_exporter** Prometheus exporter.
- **PMM-1932**: The *Response Length* metric was not displayed for MongoDB hosts in QAN.

Percona Monitoring and Management 1.5.3

Date December 18, 2017

Percona announces the release of *Percona Monitoring and Management 1.5.3*. This release contains fixes to bugs found after *Percona Monitoring and Management 1.5.2* was released as well as some important fixes and improvements not related to the previous release.

Improvements

- **PMM-1874**: The read timeout of the proxy server (*/prometheus*) has been increased from the default of 60 seconds to avoid nginx gateway timeout error when loading data rich dashboards.
- **PMM-1863**: We improved our handling of temporary Grafana credentials

Bug fixes

- **PMM-1828**: On CentOS 6.9, **pmm-admin list** incorrectly reported that no monitoring services were running.
- **PMM-1842**: It was not possible to restart the `mysql:queries` monitoring service after PMM Client was upgraded from version 1.0.4.
- **PMM-1797**: It was not possible to update the CloudWatch data source credentials.
- **PMM-1829**: When the user clicked a link in the Query Abstract column, an outdated version of QAN would open.
- **PMM-1836**: PMM Server installed in a Docker container could not be started if the updating procedure had been temporarily interrupted.
- **PMM-1871**: In some cases, RDS instances could not be discovered.
- **PMM-1845**: Converted FLUSH SLOW LOGS to FLUSH NO_WRITE_TO_BINLOG SLOW LOGS so that GTID event isn't created
- **PMM-1816**: Fixed a rendering error in Firefox.

Percona Monitoring and Management 1.5.2

Date December 5, 2017

Percona announces the release of *Percona Monitoring and Management 1.5.2*. This release contains fixes to bugs found after *Percona Monitoring and Management 1.5.1* was released.

Bug fixes

- **PMM-1790:** QAN would display query metrics even for a host that was not configured for `mysql:queries` or `mongodb:queries`. We have fixed the behaviour to display an appropriate warning message when there are no query metrics for the selected host.
- **PMM-1826:** If PMM Server 1.5.0 is deployed via Docker, the *Update* button would not upgrade the instance to a later release.
- **PMM-1830:** If PMM Server 1.5.0 is deployed via AMI instance, the *Update* button would not upgrade the instance to a later release.

Percona Monitoring and Management 1.5.1

Date November 29, 2017

Percona announces the release of *Percona Monitoring and Management* 1.5.1. This release contains fixes to bugs found after *Percona Monitoring and Management* 1.5.0 was released.

Bug fixes

- **PMM-1771:** When upgrading PMM to 1.5.0 using Docker commands, *PMM System Summary*, *PMM Add Instance*, *PMM Query Analytics* dashboards were not available.
- **PMM-1761:** The *PMM Query Analytics* dashboard did not display the list of hosts correctly.
- **PMM-1769:** It was possible to add an Amazon RDS instance providing invalid credentials on the *PMM Add Instance* dashboard.

Other bug fixes: [PMM-1767](#), [PMM-1762](#)

Percona Monitoring and Management 1.5.0

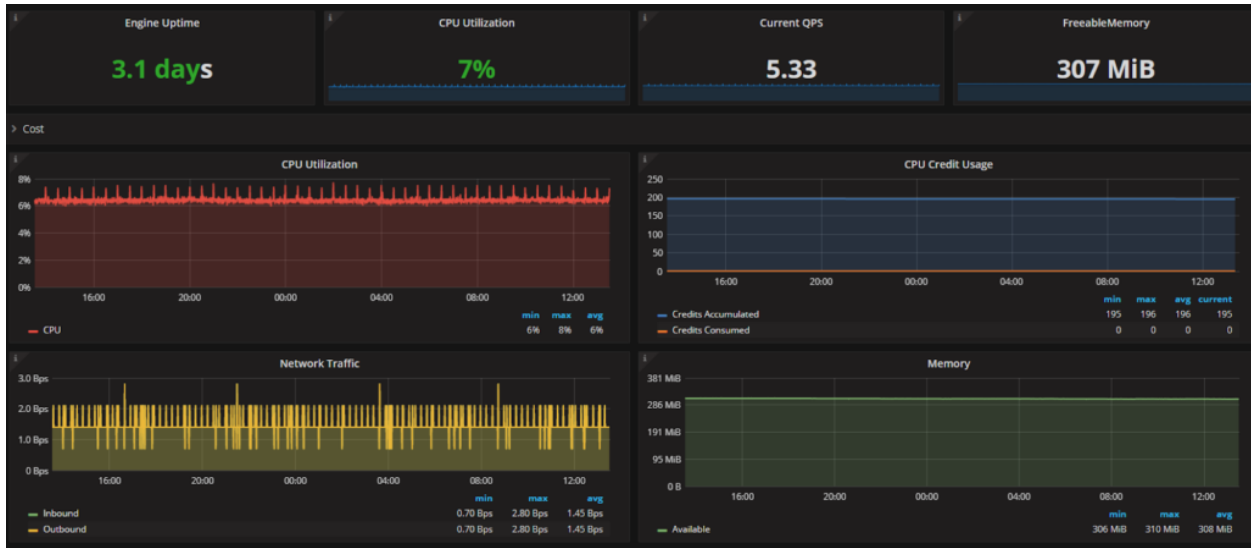
Date November 23, 2017

Percona announces the release of *Percona Monitoring and Management* 1.5.0, with a focus on the following features:

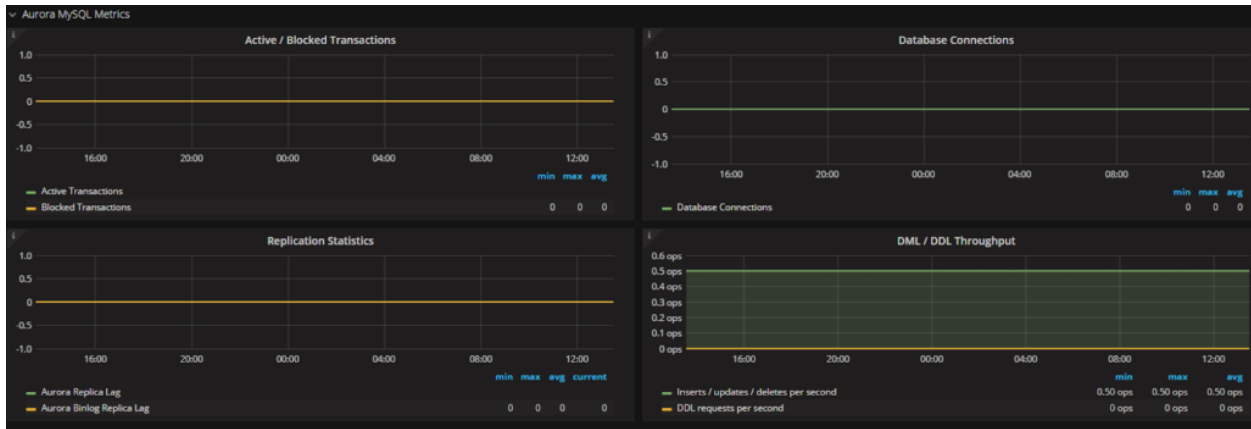
- Enhanced support for MySQL on Amazon RDS and Amazon Aurora – Dedicated Amazon Aurora dashboard offers maximum visibility into key database characteristics, eliminating the need for additional monitoring nodes. We renamed Amazon RDS OS Metrics to Amazon RDS / Amazon MySQL Metrics
- Simpler configuration – *Percona Monitoring and Management* now offers easier configuration of key Amazon RDS and Amazon Aurora settings via a web interface
- One-click data collection – One button retrieves vital information on server performance to assist with troubleshooting
- Improved interface – A simple, consistent user interface makes it faster and more fluid to switch between *PMM Query Analytics* and Metrics Monitor

Highlights from our new Amazon RDS / Aurora MySQL Metrics dashboard:

Amazon Aurora MySQL & RDS MySQL shared elements



Amazon Aurora MySQL unique elements



Amazon RDS for MySQL unique elements



QAN is now integrated into Metrics Monitor and appears as a separate dashboard known as PMM.



With this release, *Percona Monitoring and Management* introduces a new deployment option via AWS Marketplace. This is in addition to our distribution method of [Amazon Machine Images \(AMI\)](#).

In this release, Grafana and Prometheus have been upgraded. PMM now includes Grafana 4.6.1. One of the most prominent features that the upgraded Grafana offers is the support of annotations. You can mark a point or select a region in a graph and give it a meaningful description. For more information, see the [release highlights](#).

Prometheus version 1.8.2, shipped with this release, offers a number of bug fixes. For more information, see the [Prometheus change log](#).

New features

- **PMM-434:** PMM enables monitoring of Amazon RDS and Amazon Aurora metrics
- **PMM-1133:** *PMM Query Analytics* is available from Grafana as a dashboard
- **PMM-1470:** Cloudwatch metrics have been integrated into Prometheus
- **PMM-699:** AWS RDS and Amazon Aurora metrics were combined into one dashboard
- **PMM-722:** The MariaDB dashboard graph elements were distributed among other existing dashboards and MariaDB dashboard was removed; further the MyISAM dashboard was renamed to MyISAM/Aria Metrics
- **PMM-1258:** The `DISABLE_UPDATES` option enables preventing manual updates when PMM Server is run from a Docker container.
- **PMM-1500:** Added InnoDB Buffer Disk Reads to graph InnoDB Buffer Pool Requests to better understand missed InnoDB BP cache hits

Improvements

- **PMM-1577:** Prometheus was updated to version 1.8.2
- **PMM-1603:** Grafana was updated to version 4.6.1
- **PMM-1669:** The representation of numeric values in the Context Switches graph in the System Overview dashboard was changed to improve readability.
- **PMM-1575:** Templating rules were improved for the MyRocks and TokuDB dashboards so that only those instances with these storage engines are displayed.

Bug fixes

- **PMM-1082:** The CPU Usage graph on the Trends dashboard showed incorrect spikes
- **PMM-1549:** The authentication of the mongodb:queries monitoring service did not work properly when the name of the database to authenticate was not provided.
- **PMM-1673:** Fixed display issue with Microsoft Internet Explorer 11

Percona Monitoring and Management 1.4.1

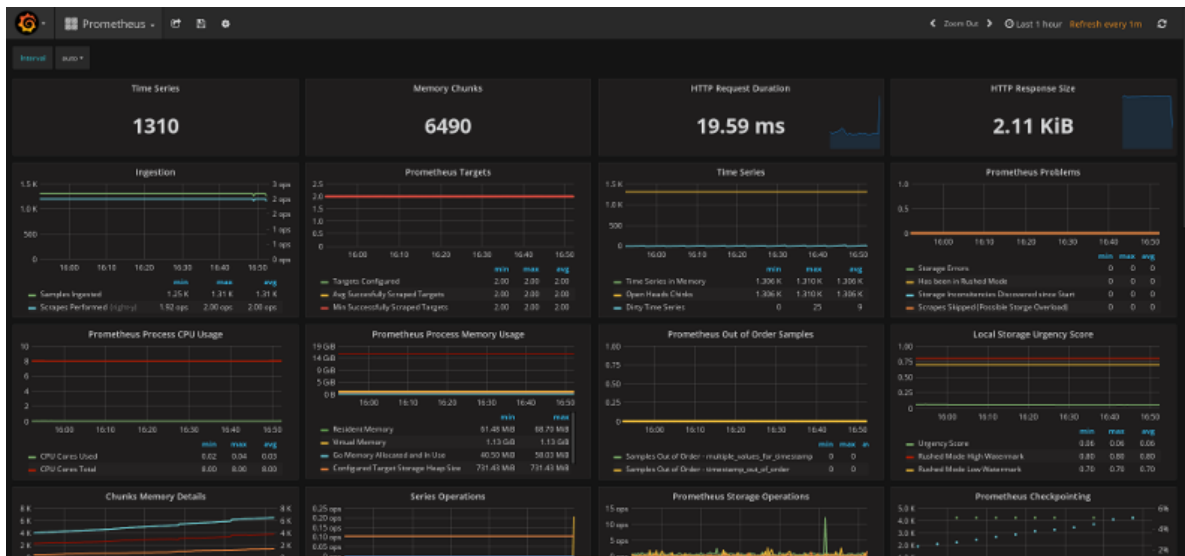
Date November 2, 2017

Percona announces the release of *Percona Monitoring and Management 1.4.1* on Thursday, November 2nd, 2017. This release contains fixes to bugs found after *Percona Monitoring and Management 1.4.0* was released. It also introduces two important improvements. The *btrfs* file system has been replaced with *XFS* in AMI and OVF images and the Prometheus dashboard has been enhanced to offer more information.

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

Improvements

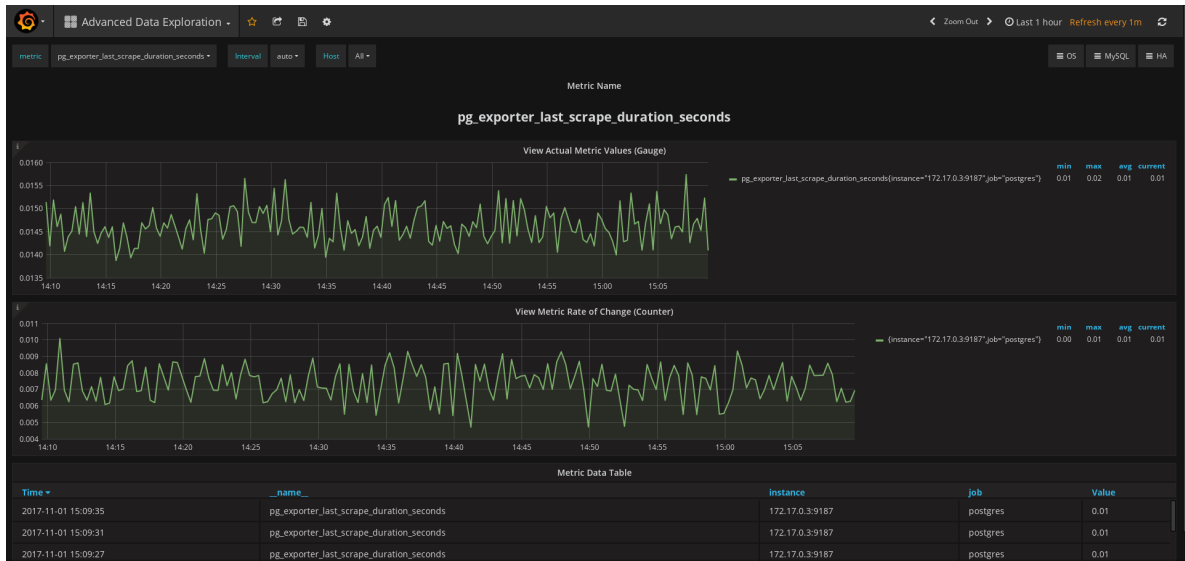
- **PMM-1567:** The *btrfs* file system has been replaced with *XFS* in *AMI and OVF images* to meet the requirements of AWS Marketplace.
- **PMM-1594:** In Metrics Monitor, the Prometheus dashboard has been updated to show more information about the running Prometheus jobs and help estimate their efficiency.



Bug fixes

- **PMM-1620:** In some cases, PMM could not be *upgraded* to version 1.4.0 by using the *Update* button on the landing page.
- **PMM-1633:** QAN would show error List of Tables is Empty for instances having been upgraded from earlier releases of PMM, due to incorrect values being stored in the database. This has been addressed to identify the incorrect values and replace with accurate schema and table information.

- **PMM-1634:** The *Advanced Data Exploration* dashboard did not always display data points from `external:metrics` monitoring services due to a too restrictive Grafana Template filter.



- **PMM-1636:** Special characters prevented the removal of `external:metrics` services using the `pmm-admin remove` command.

Percona Monitoring and Management 1.4.0

Date October 20, 2017

Percona announces the release of Percona Monitoring and Management 1.4.0.

This release introduces the support of external Prometheus exporters so that you can create dashboards in the Metrics monitor even for the monitoring services other than those provided with PMM client packages. To attach an existing external Prometheus exporter, run `pmm-admin add external:metrics NAME_OF_EXPORTER URL:PORT`.

Listing 12.1: Example of JSON output of the `pmm-admin list --json` command. It also contains an external monitoring service in the value of the `ExternalServices` element.

```
{
  "Version": "1.4.0",
  "ServerAddress": "127.0.0.1:80",
  "ServerSecurity": "",
  "ClientName": "percona",
  "ClientAddress": "172.17.0.1",
  "ClientBindAddress": "",
  "Platform": "linux-systemd",
  "Err": "",
  "Services": [
    {
      "Type": "linux:metrics",
      "Name": "percona",
      "Port": "42000",
      "Running": true,
      "DSN": "-",
      "Options": "",
      "SSL": "",
      "Password": ""
    }
  ],
  "ExternalErr": "",
  "ExternalServices": [
    {
      "JobName": "postgres",
      "ScrapeInterval": "1000000000",
      "ScrapeTimeout": "1000000000",
      "MetricsPath": "/metrics",
      "Scheme": "http",
      "StaticTargets": [
        "127.0.0.1:5432"
      ]
    }
  ]
}
```

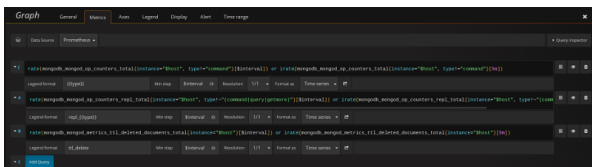
The list of attached monitoring services is now available not only in the tabular format but also as a JSON file to enable automatic verification of your configuration. To view the list of monitoring services in the JSON format run

```
pmm-admin list --json.
```

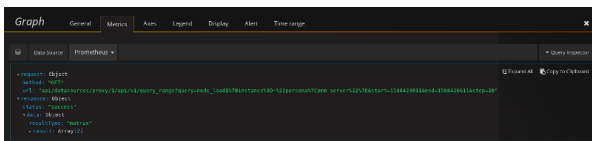
In this release, Prometheus and Grafana have been upgraded. Prometheus version 1.7.2, shipped with this release, offers a number of bug fixes that will contribute to its smooth operation inside PMM. For more information, see [the Prometheus change log](#).

Version 4.5.2 of Grafana, included in this release of PMM, offers a number of new tools that will facilitate data analysis in PMM:

- New `query editor` for Prometheus expressions which features syntax highlighting and autocompletion for metrics, functions and range vectors.



- `Query inspector` which provides detailed information about the query. The primary goal of graph inspector is to enable analyzing a graph which does not display data as expected.



The complete list of new features in **Grafana 4.5.0** is available from [What's New in Grafana v4.5](#).

For install and upgrade instructions, see [Deploying Percona Monitoring and Management](#).

New features

- PMM-1520: Prometheus upgraded to version 1.7.2.
- PMM-1521: Grafana upgraded to version 4.5.2.
- PMM-1091: The `pmm-admin list` produces a JSON document as output if the `--json` option is supplied.
- PMM-507: *External exporters are supported* with `pmm-admin`.
- PMM-1622: **docker** images of PMM Server are available for downloading as **tar** packages.

Bug fixes

- PMM-1172: In some cases, the TABLES section of a query in **QAN** could contain no data and display the *List of tables is empty* error. The `Query` and `Explain` sections had the relevant values.
- PMM-1519: A Prometheus instance could be forced to shut down if it contained too many targets (more than 50). When started the next time, Prometheus initiated a time consuming crash recovery routine which took long on large installations.

Percona Monitoring and Management 1.3.2

Date October 11, 2017

Percona announces the release of Percona Monitoring and Management 1.3.2. This release only contains bug fixes related to usability.

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

Bug fixes

- **PMM-1529**: When the user selected “Today”, “This week”, “This month”, or “This year” range in Metrics Monitor and clicked the *Query Analytics* button, the QAN page opened reporting no data for the selected range even if the data were available.
- **PMM-1528**: In some cases, the page not found error could appear instead of the **QAN** page after upgrading by using the Upgrade button.
- **PMM-1498**: In some cases, it was not possible to shutdown the virtual machine containing the PMM Server imported as an OVA image.

Other bug fixes in this release: [PMM-913](#), [PMM-1215](#), [PMM-1481](#), [PMM-1483](#), [PMM-1507](#)

Percona Monitoring and Management 1.3.1

Date September 29, 2017

Percona announces the release of Percona Monitoring and Management 1.3.1. This release only contains bug fixes related to usability.

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

Bug fixes

- **PMM-1271**: In QAN, when the user selected a database host with no queries, the query monitor could still show metrics.
- **PMM-1512**: When reached from **Grafana**, **QAN** would open its home page. Now, **QAN** opens and automatically selects the database host and time range active in **Grafana**.
- **PMM-1523**: User defined Prometheus memory settings were not honored, potentially causing performance issues in high load environments.

Other bug fixes in this release: [PMM-1452](#), [PMM-1515](#).

Percona Monitoring and Management 1.3.0

Date September 26, 2017

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

In release 1.3.0, Percona Monitoring and Management introduces a basic support for the MyRocks storage engine. There is a special dashboard in **Metrics Monitor** where the essential metrics of MyRocks are presented as separate graphs. And **Metrics Monitor** graphs now feature on-demand descriptions.

There are many improvements to QAN (Query Analytics) both in the user interface design and in its capabilities. In this release, **QAN** starts supporting all types of MongoDB queries.

Orchestrator is not enabled by default because leaving it in a non-configured state was confusing to users. *It is still possible to enable it.*

New Features

- **PMM-1290:** Basic support for the metrics of the MyRocks storage engine in MySQL via the `mysqld-exporter`.
- **PMM-1312:** Metrics Monitor now features a MyRocks dashboard.
- **PMM-1330:** Basic telemetry data are collected from PMM Servers.
- **PMM-1417:** A new dashboard titled *Advanced Data Exploration Dashboard* in Metrics Monitor enables exploring any data in Prometheus
- **PMM-1437:** `pmm-admin` allows passing parameters to exporters.
- **PMM-685:** The EXPLAIN command is now supported for MongoDB queries in **QAN**.

Improvements

- **PMM-1262:** The system checks for updates much faster
- **PMM-1015:** **QAN** should show all collections from a mongod instance. Make sure that profiling is enabled in MongoDB.
- **PMM-1057:** **QAN** supports all MongoDB query types.
- **PMM-1270:** In **Metrics Monitor**, the MariaDB dashboard host filter now displays only the hosts running MariaDB.
- **PMM-1287:** The `mongodb:queries` monitoring service is not considered to be experimental any more. The `dev-enable` option is no longer needed when you run the `pmm-admin add` command to add it.
- **PMM-1446:** In **Metrics Monitor**, the *MySQL Active Threads* graph displays data more accurately.
- **PMM-1455:** In **Metrics Monitor**, features improved descriptions of the InnoDB Transactions graph.
- **PMM-1476:** In **QAN**, the new interface is now used by default.
- **PMM-1479:** It is now possible to go to **QAN** directly from **Metrics Monitor**.
- **PMM-515:** **Orchestrator** is disabled by default. It is possible to enable it when running your docker container.

Bug fixes

- **PMM-1298:** In **QAN**, the query abstract could be empty for MySQL hosts for low ranking queries. This bug is fixed to contain *Low Ranking Queries* as the value of the query abstract.
- **PMM-1314:** The selected time range in **QAN** could be applied incorrectly. This problem is not observed in the new design of **QAN**.
- **PMM-1398:** The **Prometheus** server was not restarted after PMM was upgraded. This bug is now fixed.
- **PMM-1427:** The *CPU Usage/Load* graph in the *MySQL Overview* dashboard was displayed with slightly incorrect dimensions. This bug is now solved.
- **PMM-1439:** If the EXPLAIN command was not supported for the selected query, there could appear a JavaScript error.
- **PMM-1472:** In some cases, the monitoring of queries for MongoDB with replication could not be enabled.
- **PMM-943:** InnoDB AHI Usage Graph had incorrect naming and hit ratio computation.

Other bug fixes in this release: [PMM-1479](#)

Percona Monitoring and Management 1.2.2

Date August 23, 2017

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

This release contains bug fixes related to performance and introduces various improvements. It also contains an updated version of Grafana.

Changes in PMM Server

The following changes were introduced in *PMM Server 1.2.2*:

Bug fixes

- **PMM-927:** The error “Cannot read property ‘hasOwnProperty’ of undefined” was displayed on the QAN page for mongodb.
After enabling monitoring and generating data for mongodb, the PMM client showed the following error message on the QAN page: “Cannot read property ‘hasOwnProperty’ of undefined”. This bug is now fixed.
- **PMM-949:** Percona Server was not detected properly, the `log_slow_*` variables were not properly detected.
- **PMM-1081:** Performance Schema Monitor treated queries that didn’t show up in every snapshot as new queries reporting a wrong number of counts between snapshots.
- **PMM-1272:** mongodb: the query empty abstract. This bug is now fixed.
- **PMM-1277:** The QPS Graph had inappropriate prometheus query. This bug is now fixed.
- **PMM-1279:** The MongoDB summary did not work in QAN2 if mongodb authentication was activated. This bug is now fixed.
- **PMM-1284:** Dashboards pointed to QAN2 instead of QAN. This bug is now fixed.

Improvements

- **PMM-586:** The `wsrep_ews_repl_latency` parameter is now monitored in Grafana dashboards
- **PMM-624:** The Grafana User ID remains the same in the `pmm-server` docker image
- **PMM-1209:** OpenStack support is now enabled during the OVA image creation
- **PMM-1211:** It is now possible to configure a static IP for an OVA image

The root password can only be set from console. If the root password is not changed from the default, a warning message appears on the console requesting the user to change the root password on the root first login from console. Web/SSH users can neither use the root account password nor detect if the root password is set to the default value.

- **PMM-1221:** Grafana updated to version 4.4.3

Percona Monitoring and Management 1.2.1

Date August 16, 2017

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

This hotfix release improves memory consumption

Changes in PMM Server

The following changes were introduced in *PMM Server* 1.2.1:

Bug fixes

- **PMM-1280:** PMM server affected by nGinx [CVE-2017-7529](#)

An integer overflow exploit could result in a DOS (Denial of Service) for the affected nginx service with the `max_ranges` directive not set.

This problem is solved by setting the `set max_ranges` directive to 1 in the nGinx configuration.

Improvements

- **PMM-1232:** Update the default value of the `METRICS_MEMORY` configuration setting

Previous versions of *PMM Server* used a different value for the `METRICS_MEMORY` configuration setting which allowed Prometheus to use up to 768MB of memory.

PMM Server 1.2.0 used the `storage.local.target-heap-size` setting, its default value being 256MB. Unintentionally, this value reduced the amount of memory that Prometheus could use. As a result, the performance of Prometheus was affected.

To improve the performance of Prometheus, the default setting of `storage.local.target-heap-size` has been set to 768 MB.

Percona Monitoring and Management 1.2.0

Date July 14, 2017

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

Changes in PMM Server

The following changes were introduced in *PMM Server* 1.2.0:

Updated Components

- Consul 0.8.5
- Grafana 4.3.2
- Orchestrator 2.1.5
- Prometheus 1.7.1

New Features

- **PMM-737:** New graphs in **System Overview** dashboard:
 - **Memory Advanced Details**
 - **Saturation Metrics**
- **PMM-1090:** Added ESXi support for *PMM Server* virtual appliance.

UI Fixes

- **PMM-707:** Fixed QPS metric in **MySQL Overview** dashboard to always show *queries per second* regardless of the selected interval.
- **PMM-708:** Fixed tooltips for graphs that displayed incorrectly.
- **PMM-739, PMM-797:** Fixed *PMM Server* update feature on landing page.
- **PMM-823:** Fixed arrow padding for collapsible blocks in QAN.
- **PMM-887:** Disabled the **Add** button when no table is specified for showing query info in QAN.
- **PMM-888:** Disabled the **Apply** button in QAN settings when nothing is changed.
- **PMM-889:** Fixed the switch between UTC and local time zone in the QAN time range selector.
- **PMM-909:** Added message `No query example` when no example for query is available in QAN.
- **PMM-933:** Fixed empty tooltips for **Per Query Stats** column in the query details section of QAN.
- **PMM-937:** Removed the percentage of total query time in query details for the `TOTAL` entry in QAN (because it is 100% by definition).
- **PMM-951:** Fixed the **InnoDB Page Splits** graph formula in the **MySQL InnoDB Metrics Advanced** dashboard.
- **PMM-953:** Enabled stacking for graphs in **MySQL Performance Schema** dashboard.
- **PMM-954:** Renamed **Top Users by Connections** graph in **MySQL User Statistics** dashboard to **Top Users by Connections Created** and added the **Connections/sec** label to the Y-axis.
- **PMM-957:** Refined titles for **Client Connections** and **Client Questions** graphs in **ProxySQL Overview** dashboard to mention that they show metrics for all host groups (not only the selected one).
- **PMM-961:** Fixed the formula for **Client Connections** graph in **ProxySQL Overview** dashboard.
- **PMM-964:** Fixed the gaps for high zoom levels in **MySQL Connections** graph on the **MySQL Overview** dashboard.
- **PMM-976:** Fixed Orchestrator handling by `supervisorctl`.
- **PMM-1129:** Updated the **MySQL Replication** dashboard to support new `connection_name` label introduced in `mysqld_exporter` for multi-source replication monitoring.
- **PMM-1054:** Fixed typo in the tooltip for the **Settings** button in QAN.
- **PMM-1055:** Fixed link to Query Analytics from Metrics Monitor when running *PMM Server* as a virtual appliance.
- **PMM-1086:** Removed HTML code that showed up in the QAN time range selector.

Bug Fixes

- **PMM-547:** Added warning page to Query Analytics app when there are no PMM Clients running the QAN service.
- **PMM-799:** Fixed Orchestrator to show correct version.
- **PMM-1031:** Fixed initialization of **Query Profile** section in QAN that broke after upgrading Angular.
- **PMM-1087:** Fixed QAN package building.

Other Improvements

- **PMM-348:** Added daily log rotation for nginx.
- **PMM-968:** Added Prometheus build information.
- **PMM-969:** Updated the Prometheus memory usage settings to leverage new flag. For more information about setting memory consumption by PMM, see [FAQ](#).

Changes in PMM Client

The following changes were introduced in *PMM Client* 1.2.0:

New Features

- **PMM-1114:** Added *PMM Client* packages for Debian 9 (“stretch”).

Bug Fixes

- **PMM-481, PMM-1132:** Fixed fingerprinting for queries with multi-line comments.
- **PMM-623:** Fixed `mongodb_exporter` to display correct version.
- **PMM-927:** Fixed bug with empty metrics for MognoDB query analytics.
- **PMM-1126:** Fixed `promu build` for `node_exporter`.
- **PMM-1201:** Fixed `node_exporter` version.

Other Improvements

- **PMM-783:** Directed `mongodb_exporter` log messages to `stderr` and excluded many generic messages from the default `INFO` logging level.
- **PMM-756:** Merged upstream `node_exporter` version 0.14.0.

Several collectors were deprecated in this release:

- `gmond` - Out of scope.
- `megacli` - Requires forking, to be moved to `textfile` collection.
- `ntp` - Out of scope.

It also introduced the following breaking change:

- Collector errors are now a separate metric: `node_scrape_collector_success`, not a label on `node_exporter_scrape_duration_seconds`

- **PMM-1011:** Merged upstream `mysqld_exporter` version 0.10.0.

This release introduced the following breaking change:

- `mysql_slave_...` metrics now include an additional `connection_name` label to support MariaDB multi-source replication.

Percona Monitoring and Management 1.1.5

Date June 21, 2017

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

Changes in PMM Server

- **PMM-667:** Fixed the *Latency* graph in the *ProxySQL Overview* dashboard to plot microsecond values instead of milliseconds.
- **PMM-800:** Fixed the *InnoDB Page Splits* graph in the *MySQL InnoDB Metrics Advanced* dashboard to show correct page merge success ratio.
- **PMM-1007:** Added links to Query Analytics from *MySQL Overview* and *MongoDB Overview* dashboards. The links also pass selected host and time period values.

Note: These links currently open QAN2, which is still considered experimental.

Changes in PMM Client

- **PMM-931:** Fixed `pmm-admin` script when adding MongoDB metrics monitoring for secondary in replica set.

Percona Monitoring and Management 1.1.4-2

Date June 2, 2017

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

This minor release contains one security hot fix:

- Removed inactive build keys from SSH configuration.

Percona Monitoring and Management 1.1.4

Date May 29, 2017

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

This release includes experimental support for MongoDB in Query Analytics, including updated QAN interface.

Query Analytics for MongoDB

To enable MongoDB query analytics, use the `mongodb:queries` alias when *adding the service*. As an experimental feature, it also requires the `--dev-enable` option:

```
sudo pmm-admin add --dev-enable mongodb:queries
```

Note: Currently, it monitors only collections that are present when you enable MongoDB query analytics. Query data for collections that you add later is not gathered. This is a known issue and it will be fixed in the future.

Query Analytics Redesign

The QAN web interface was updated for better usability and functionality (including the new MongoDB query analytics data). The new UI is experimental and available by specifying `/qan2` after the URL of *PMM Server*.

Note: The button on the main landing page still points to the old QAN interface.

You can check out the new QAN web UI at <https://pmmdemo.percona.com/qan2>

Changes in PMM Server

- **PMM-724:** Added the *Index Condition Pushdown (ICP)* graph to the *MySQL InnoDB Metrics* dashboard.
- **PMM-734:** Fixed the *MySQL Active Threads* graph in the *MySQL Overview* dashboard.
- **PMM-807:** Fixed the *MySQL Connections* graph in the *MySQL Overview* dashboard.
- **PMM-850:** Updated the *MongoDB RocksDB* and *MongoDB WiredTiger* dashboards.
- **PMM-890:** and **PMM-891:** Removed the *InnoDB Deadlocks* and *Index Collection Pushdown* graphs from the *MariaDB* dashboard. Both graphs are available in the *MySQL InnoDB Metrics* dashboard.
- **PMM-928:** Added tooltips with descriptions for graphs in the *MySQL Query Response Time* dashboard. Similar tooltips will be gradually added to all graphs.

Changes in PMM Client

- **PMM-554:** Added options for `pmm-admin` to enable MongoDB cluster connections.
- **PMM-666** and **PMM-746:** Fixed `proxysql_exporter`.
- **PMM-801:** Improved *PMM Client* upgrade process to preserve credentials that are used by services.

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:
 - Olog Getmore Time
 - Olog Operations
 - Olog Processing Time
 - Olog Buffered Operations
 - Olog 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.

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.

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 as a virtual appliance*
- *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.

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.

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 upgrade 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.

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 upgrade 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.

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 upgrade 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 initial 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

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

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 *Creating the 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

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 *Creating the 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 PMM?*
- 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 information 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

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.

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 comprehensive 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

CONTACTING AND CONTRIBUTING

Percona Monitoring and Management is an open source product. We provide ways for anyone to contact developers and experts directly, submit bug reports and feature requests, and contribute to source code directly.

Contacting Developers

Use the [community forum](#) to ask questions about using PMM. Developers and experts will try to help with problems that you experience.

Reporting Bugs

Use the [PMM project in JIRA](#) to report bugs and request features. Please register and search for similar issues before submitting a bug or feature request.

Contributing Source Code

Use the [GitHub repository](#) to explore source code and suggest contributions. You can fork and clone any Percona repositories, but to have your source code patches accepted please sign the Contributor License Agreement (CLA).

FREQUENTLY ASKED QUESTIONS

- *How can I contact the developers?*
- *What are the minimum system requirements for PMM?*
- *How to control memory consumption for PMM? (relevant to versions lower than 1.13 of PMM)*
- *How to control data retention for PMM?*
- *Where are the services created by PMM Client?*
- *Where is DSN stored?*
- *Where are PMM Client log files located?*
- *How often are nginx logs in PMM Server rotated?*
- *What are common performance considerations?*
- *Can I stop all services at once?*
- *What privileges are required to monitor a MySQL instance?*
- *Can I monitor multiple MySQL instances?*
- *Can I rename instances?*
- *Can I use non-default ports for instances?*
- *How to troubleshoot communication issues between PMM Client and PMM Server?*
- *What resolution is used for metrics?*
- *Why do I get Failed ReadTopologyInstance error when adding MySQL host to Orchestrator?*
- *How to set the root password when PMM Server is installed as a virtual appliance*
- *How to install the experimental version of PMM Server?*

How can I contact the developers?

The best place to discuss PMM with developers and other community members is the [community forum](#).

If you would like to report a bug, use the [PMM project in JIRA](#).

What are the minimum system requirements for PMM?

PMM Server

Any system which can run Docker version 1.12.6 or later.

It needs roughly 1 GB of storage for each monitored database node with data retention set to one week.

Note: By default, *retention* is set to 30 days for Metrics Monitor and to 8 days for Query Analytics. Also consider *disabling table statistics*, which can greatly decrease Prometheus database size.

Minimum memory is 2 GB for one monitored database node, but it is not linear when you add more nodes. For example, data from 20 nodes should be easily handled with 16 GB.

PMM Client

Any modern 64-bit Linux distribution. It is tested on the latest versions of Debian, Ubuntu, CentOS, and Red Hat Enterprise Linux.

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

How to control memory consumption for PMM? (relevant to versions lower than 1.13 of PMM)

Prometheus 1.x.x, shipped with PMM up to version 1.13.0, used by default 768 MB of memory for storing the most recently used data chunks.

If you haven't upgraded to a version 1.13.0 or higher, you may require a higher limit, depending on the amount of data coming into Prometheus, to avoid throttling data ingestion, or to allow less memory consumption by Prometheus.

Important: Starting with version 1.13.0, PMM uses Prometheus 2. Due to optimizations in Prometheus, setting the metrics memory by passing the `METRICS_MEMORY` option is a deprecated practice.

For compatibility reasons PMM 1.13.0 and above is still supporting Prometheus 1.x, but with substantially decreased resources: now it uses only 15% of available memory and its connections amount limit is managed by the `MAX_CONNECTIONS` environment variable, set to 15 by default.

How to control data retention for PMM?

By default, Prometheus stores time-series data for 30 days, and QAN stores query data for 8 days.

Depending on available disk space and your requirements, you may need to adjust data retention time.

You can control data retention by passing the `METRICS_RETENTION` and `QUERIES_RETENTION` environment variables when *creating and running the PMM Server container*. To set environment variables, use the `-e` option. The value is passed as a combination of hours, minutes, and seconds. For example, the default value of 30 days for

`METRICS_RETENTION`> 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 for Prometheus to 8 days:

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

See also:

Metrics retention [METRICS_RETENTION](#)

Queries retention [QUERIES_RETENTION](#)

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 as root `pmm-admin info`.

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 password) is stored in Consul API for information purposes (used by the `pmm-admin list` command).

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 MySQL 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 monitoring services](#).

How often are nginx logs in PMM Server rotated?

PMM Server runs `logrotate` to rotate nginx logs on a daily basis and keep up to 10 latest log files.

What 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 as root `pmm-admin add mysql --help`.

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 monitoring 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 monitoring services*.

What privileges are required to monitor a MySQL instance?

See *Creating a MySQL User Account to Be Used with PMM*.

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` parameters, 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 `pmm-admin add mysql --help`.

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.

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 port.

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

How to troubleshoot communication issues between PMM Client and PMM Server?

There is a `pmm-admin check-network` command, which checks connectivity between PMM Client and PMM Server and presents the summary of this check in a human readable form.

Broken network connectivity may be caused by rather wide set of reasons. Particularly, when *using Docker*, the container is constrained by the host-level routing and firewall rules. For example, your hosting provider might have default *iptables* rules on their hosts that block communication between PMM Server and PMM Client, resulting in *DOWN* targets in Prometheus. If this happens, check firewall and routing settings on the Docker host.

Also PMM is able to generate a set of diagnostics data which can be examined and/or shared with Percona Support to solve an issue faster. See details on how to get collected logs [from PMM Server](#) and [from PMM Client](#).

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 *Adding an Amazon RDS DB instance to PMM*.

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*.

How to set the root password when PMM Server is installed as a virtual appliance

With your virtual appliance set up, you need to set the root password for your PMM Server. By default, the virtual machine is configured to enforce changing the default password upon the first login.

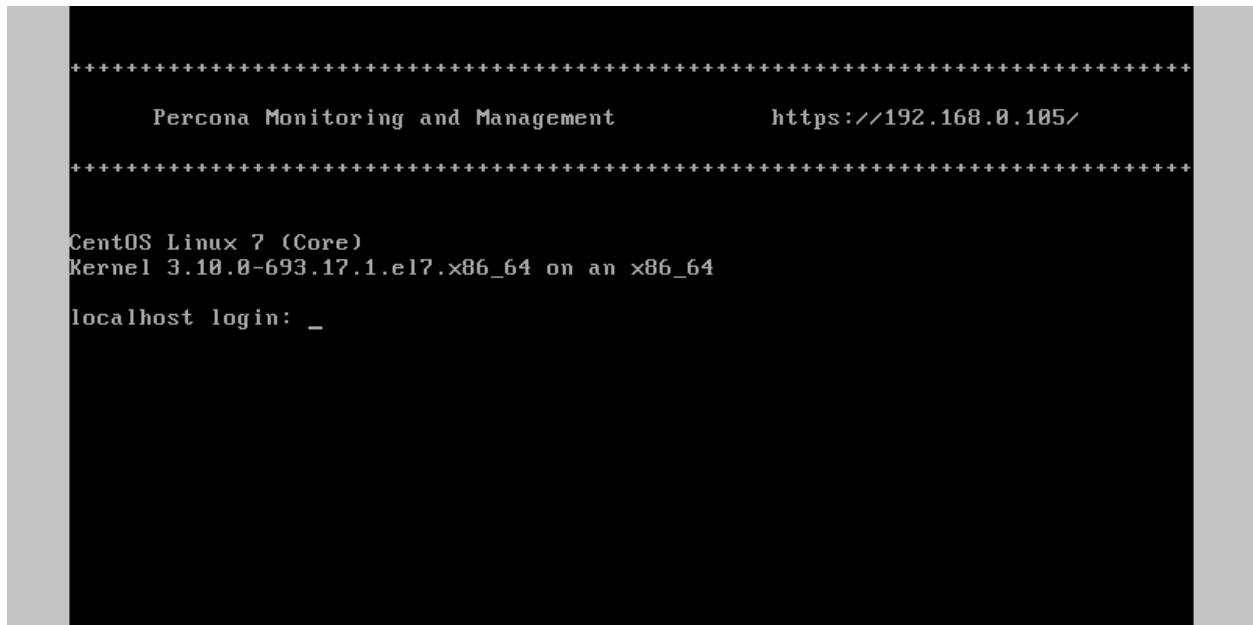


Fig. 14.1: Set the root password when logging in.

Run your virtual machine and when requested to log in, use the following credentials:

User root

Password percona

The system immediately requests that you change your password. Note that, for the sake of security, your password must not be trivial and pass at least the dictionary check. If you do not provide your password within sixty seconds you are automatically logged out. In this case use the default credentials to log in again.

```

Percona Monitoring and Management          https://192.168.0.105/
*****
CentOS Linux 7 (Core)
Kernel 3.10.0-693.17.1.el7.x86_64 on an x86_64

localhost login: root
Password:
You are required to change your password immediately (root enforced)
Changing password for root.
(current) UNIX password: login: timed out after 60 seconds

CentOS Linux 7 (Core)
Kernel 3.10.0-693.17.1.el7.x86_64 on an x86_64

localhost login: root
Password:
You are required to change your password immediately (root enforced)
Changing password for root.
(current) UNIX password:
New password:
Retype new password:
[root@localhost ~]# _

```

Fig. 14.2: Set a new password and have full access to your system

After the new password is set you control this system as a superuser and can make whatever changes required.

Important: You cannot access the root account if you access PMM Server using SSH or via the Web interface.

How to install the experimental version of PMM Server?

If you would like to experiment with the latest development version using Docker, you may use the `dev-latest` image. This version, however, is not intended to be used in a production environment.

```
$ docker pull perconalab/pmm-server:dev-latest
```

If you would like to experiment with the latest development version of PMM Server VirtualBox image, download the development version as follows:

```
$ wget "http://percona-vm.s3-website-us-east-1.amazonaws.com/PMM-Server-dev-latest.ova"
↪ "
```

Important: This is a development version which is not designed for a production environment.

See also:

Setting up PMM Server via Docker *setup procedure*

Setting up PMM Server via VirtualBox *VirtualBox Using the Command Line*

PMM Server Additional Options

This glossary contains the additional parameters that you may pass when starting PMM Server.

If you use Docker to run the server, use the `-e` flag followed by the parameter. Use this flag in front of each parameter that you pass to PMM Server.

Here, we pass more than one option to PMM Server along with the `docker run` command. Run this command as root or by using the `sudo` command.

```
$ docker run -d -p 80:80 \  
  --volumes-from pmm-data \  
  --name pmm-server \  
  -e SERVER_USER=jsmith \  
  -e SERVER_PASSWORD=SomeR4ndom-Pa$$w0rd \  
  --restart always \  
  percona/pmm-server:latest
```

List of PMM Server Parameters

DISABLE_TELEMETRY With *telemetry* enabled, your PMM Server sends some statistics to v.percona.com every 24 hours. This statistics includes the following details:

- PMM Server unique ID
- PMM version
- The name and version of the operating system
- MySQL version
- Perl version

If you do not want your PMM Server to send this information, disable telemetry when running your Docker container:

```
$ docker run ... -e DISABLE_TELEMETRY=true ... percona/pmm-server:latest
```

DISABLE_UPDATES To update your PMM from web interface you only need to click the *Update* on the home page. The `DISABLE_UPDATES` option is useful if updating is not desirable. Set it to `true` when running PMM in the Docker container.

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

```
$ docker run ... -e DISABLE_UPDATES=true ... percona/pmm-server:latest
```

The `DISABLE_UPDATES` option removes the *Update* button from the interface and prevents the system from being updated manually.

METRICS_MEMORY By default, Prometheus in PMM Server uses all available memory for storing the most recently used data chunks. Depending on the amount of data coming into Prometheus, you may require to allow less memory consumption if it is needed for other processes.

Important: Starting with version 1.13.0, PMM uses Prometheus 2. Due to optimizations in Prometheus, setting the metrics memory by passing the `METRICS_MEMORY` option is a deprecated practice.

If you are still using a version of PMM prior to 1.13 you might need to set the metrics memory by passing the `METRICS_MEMORY` environment variable along with the `docker run` command.

Run this command as root or by using the `sudo` command. The value must be passed in kilobytes. For example, to set the limit to 4 GB of memory run the following command:

```
$ docker run ... -e METRICS_MEMORY=4194304 ... percona/pmm-server:latest
```

See also:

Docker documentation: Controlling memory usage in a Docker container https://docs.docker.com/config/containers/resource_constraints/

METRICS_RESOLUTION This environment variable sets the minimum resolution for checking metrics. You should set it if the latency is higher than 1 second.

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

```
$ docker run ... -e METRICS_RESOLUTION=VALUE ... percona/pmm-server:latest
```

METRICS_RETENTION This option determines how long metrics are stored at *PMM Server*. The value is passed as a combination of hours, minutes, and seconds, such as **720h0m0s**. The minutes (a number followed by *m*) and seconds (a number followed by *s*) are optional.

To set the `METRICS_RETENTION` option to 8 days, set this option to *192h*.

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

```
$ docker run ... -e METRICS_RETENTION=192h ... percona/pmm-server:latest
```

See also:

Data retention in PMM [Data retention](#)

Queries retention [QUERIES_RETENTION](#)

ORCHESTRATOR_ENABLED This option enables Orchestrator (See *Orchestrator*). By default it is disabled. It is also disabled if this option contains **false**.

```
$ docker run ... -e ORCHESTRATOR_ENABLED=true ... percona/pmm-server:latest
```

See also:

Orchestrator [Orchestrator](#)

Orchestrator Credentials

- [ORCHESTRATOR_USER](#)

- *ORCHESTRATOR_PASSWORD*

ORCHESTRATOR_PASSWORD Pass this option, when running your *PMM Server* via Docker to set the orchestrator password.

This option has no effect if the *ORCHESTRATOR_ENABLED* option is set to **false**.

```
$ docker run ... -e ORCHESTRATOR_ENABLED=true ORCHESTRATOR_USER=name -e_
↳ORCHESTRATOR_PASSWORD=pass ... percona/pmm-server:latest
```

See also:

ORCHESTRATOR_ENABLED

ORCHESTRATOR_USER Pass this option, when running your *PMM Server* via Docker to set the orchestrator user. You only need this parameter (along with *ORCHESTRATOR_PASSWORD* if you have set up a custom Orchestrator user.

This option has no effect if the *ORCHESTRATOR_ENABLED* option is set to **false**.

```
$ docker run ... -e ORCHESTRATOR_ENABLED=true ORCHESTRATOR_USER=name -e_
↳ORCHESTRATOR_PASSWORD=pass ... percona/pmm-server:latest
```

QUERIES_RETENTION This option determines how many days queries are stored at *PMM Server*.

```
$ docker run ... -e QUERIES_RETENTION=30 ... percona/pmm-server:latest
```

See also:

Metrics retention *METRICS_RETENTION*

Data retention in PMM *Data retention*

SERVER_PASSWORD Set the password to access the PMM Server web interface.

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

```
$ docker run ... -e SERVER_PASSWORD=YOUR_PASSWORD ... percona/pmm-server:latest
```

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

SERVER_USER By default, the user name is `pmm`. Use this option to use another user name.

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

```
$ docker run ... -e SERVER_USER=USER_NAME ... percona/pmm-server:latest
```

Terminology Reference

%GTT See *Grand Total Time*

Data retention By default, Prometheus stores time-series data for 30 days, and *QAN* stores query data for 8 days.

Depending on available disk space and your requirements, you may need to adjust data retention time.

You can control data retention by passing the *METRICS_RETENTION* and *QUERIES_RETENTION* environment variables when *creating and running the PMM Server container*.

See also:

Metrics retention *METRICS_RETENTION*

Queries retention *QUERIES_RETENTION*

Data Source Name A database server attribute found on the *QAN* page. It informs how *PMM* connects to the selected database.

Default ports See *Ports*.

DSN See *Data Source Name*

External Monitoring Service A monitoring service which is not provided by *PMM* directly. It is bound to a running Prometheus exporter. As soon as such a service is added, you can set up the *Metrics Monitor* to display its graphs.

Grand Total Time Grand Total Time.(percent of grand total time) is the percentage of time that the database server spent running a specific query, compared to the total time it spent running all queries during the selected period of time.

Metrics A series of data which are visualized in PMM.

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

Monitoring service A special service which collects information from the database instance where *PMM Client* is installed.

To add a monitoring service, use the `pmm-admin add` command.

See also:

Passing parameters to a monitoring service *Passing options to the exporter*

Orchestrator The topology manager for MySQL. By default it is disabled for the *PMM Server*. To enable it, set the `ORCHESTRATOR_ENABLED`.

See also:

Docker container: Enabling orchestrator *ORCHESTRATOR_ENABLED*

PMM Percona Monitoring and Management

PMM annotation A feature of PMM Server which adds a special mark to all dashboards and signifies an important event in your application. Annotations are added on the PMM Client by using the `pmm-admin annotate` command.

See also:

Grafana Documentation: Annotations

<http://docs.grafana.org/reference/annotations/>

PMM Client Collects MySQL server metrics, general system metrics, and query analytics data for a complete performance overview.

The collected data is sent to *PMM Server*.

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

PMM Docker Image A docker image which enables installing the PMM Server by using `docker`.

See also:

Installing PMM Server using Docker *Running PMM Server via Docker*

PMM Home Page The starting page of the PMM portal from which you can have an overview of your environment, open the tools of PMM, and browse to online resources.

On the PMM home page, you can also find the version number and a button to update your PMM Server (see [PMM Version](#)).

PMM Server Aggregates data collected by [PMM Client](#) and presents it in the form of tables, dashboards, and graphs in a web interface.

PMM Server combines the backend API and storage for collected data with a frontend for viewing time-based graphs and performing thorough analysis of your MySQL and MongoDB hosts through a web interface.

Run PMM Server on a host that you will use to access this data.

See also:

PMM Architecture

Overview of Percona Monitoring and Management Architecture

PMM Server Version If [PMM Server](#) is installed via Docker, you can check the current PMM Server version by running `docker exec`:

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

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

PMM user permissions for AWS When creating a [IAM user](#) for Amazon RDS DB instance that you intend to monitor in PMM, you need to set all required permissions properly. For this, you may copy the following JSON for your IAM user:

```
{ "Version": "2012-10-17",
  "Statement": [{ "Sid": "Stmt1508404837000",
                  "Effect": "Allow",
                  "Action": [ "rds:DescribeDBInstances",
                              "cloudwatch:GetMetricStatistics",
                              "cloudwatch:ListMetrics" ],
                  "Resource": [ "*" ] },
                { "Sid": "Stmt1508410723001",
                  "Effect": "Allow",
                  "Action": [ "logs:DescribeLogStreams",
                              "logs:GetLogEvents",
                              "logs:FilterLogEvents" ],
                  "Resource": [ "arn:aws:logs:*:*:log-
↪group:RDSOSMetrics:*" ] }
    ]
}
```

See also:

Creating an IAM user [Creating an IAM user](#)

PMM Version The version of PMM appears at the bottom of the [PMM server home page](#).

See also:

Checking the version of PMM Server

PMM Server Version

pmm-admin A program which changes the configuration of the [PMM Client](#). See detailed documentation in the [Managing PMM Client](#) section.

Ports The following ports must be open to enable communication between the [PMM Server](#) and [PMM clients](#).

Systems under monitoring

2

Monitored DB Instances

2

Current version: 1.8.0

Check for updates manually

PMM Server should keep ports 80 or 443 ports open for computers where PMM Client is installed to access the PMM web interface.

42000 For PMM to collect general system metrics.

42001 This port is used by a service which collects query performance data and makes it available to QAN.

42002 For PMM to collect MySQL server metrics.

42003 For PMM to collect MongoDB server metrics.

42004 For PMM to collect ProxySQL server metrics.

See also:

Setting up a firewall on CentOS <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-using-firewalld-on-centos-7>

Setting up a firewall on Ubuntu <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-ubuntu-16-04>

QAN See *Query Analytics (QAN)*

Query Abstract Query pattern with placeholders. This term appears in *QAN* as an attribute of queries.

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

Query Fingerprint See *Query Abstract*

Query ID A *query fingerprint* which groups similar queries.

Query Load The percentage of time that the MySQL server spent executing a specific query.

Query Metrics Summary Table An element of *Query Analytics (QAN)* which displays the available metrics for the selected query.

Query Metrics Table A tool within *QAN* which lists metrics applicable to the query selected in the *query summary table*.

Query Summary Table A tool within *QAN* which lists the queries which were run on the selected database server during the *selected time or date range*.

Quick ranges Predefined time periods which are used by *QAN* to collect metrics for queries. The following quick ranges are available:

- last hour
- last three hours
- last five hours
- last twelve hours
- last twenty four hours
- last five days

Selected Time or Date Range A predefined time period (see *Quick ranges*), such as 1 hour, or a range of dates that *QAN* uses to collect metrics.

Telemetry Percona may collect some statistics about the machine where PMM is running.

This statistics includes the following information:

- PMM Server unique ID
- PMM version

- The name and version of the operating system, AMI or virtual appliance
- MySQL version
- Perl version

You may disable telemetry *by passing an additional parameter* to Docker.

```
$ docker run ... -e DISABLE_TELEMETRY=true ... percona/pmm-server:latest
```

Version A database server attribute found on the [QAN](#) page. It informs the full version of the monitored database server, as well as the product name, revision and release number.

Symbols

%GTT, [271](#)

D

Data retention, [271](#)

Data Source Name, [272](#)

Default ports, [272](#)

DISABLE_TELEMETRY, [269](#)

DISABLE_UPDATES, [269](#)

DSN, [272](#)

E

External Monitoring Service, [272](#)

G

Grand Total Time, [272](#)

M

Metrics, [272](#)

Metrics Monitor (MM), [272](#)

METRICS_MEMORY, [270](#)

METRICS_RESOLUTION, [270](#)

METRICS_RETENTION, [270](#)

Monitoring service, [272](#)

O

Orchestrator, [272](#)

ORCHESTRATOR_ENABLED, [270](#)

ORCHESTRATOR_PASSWORD, [271](#)

ORCHESTRATOR_USER, [271](#)

P

PMM, [272](#)

PMM annotation, [272](#)

PMM Client, [272](#)

PMM Docker Image, [272](#)

PMM Home Page, [272](#)

PMM Server, [273](#)

PMM Server Version, [273](#)

PMM user permissions for AWS, [273](#)

PMM Version, [273](#)

pmm-admin, [273](#)

Ports, [273](#)

Q

QAN, [275](#)

QUERIES_RETENTION, [271](#)

Query Abstract, [275](#)

Query Analytics (QAN), [275](#)

Query Fingerprint, [275](#)

Query ID, [275](#)

Query Load, [275](#)

Query Metrics Summary Table, [275](#)

Query Metrics Table, [275](#)

Query Summary Table, [275](#)

Quick ranges, [275](#)

S

Selected Time or Date Range, [275](#)

SERVER_PASSWORD, [271](#)

SERVER_USER, [271](#)

T

Telemetry, [275](#)

V

Version, [276](#)