PERCONA

www.percona.com

Percona Monitoring and Management Documentation

Release 2.1.0

Nov 11, 2019

CONTENTS

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

PMM Concepts

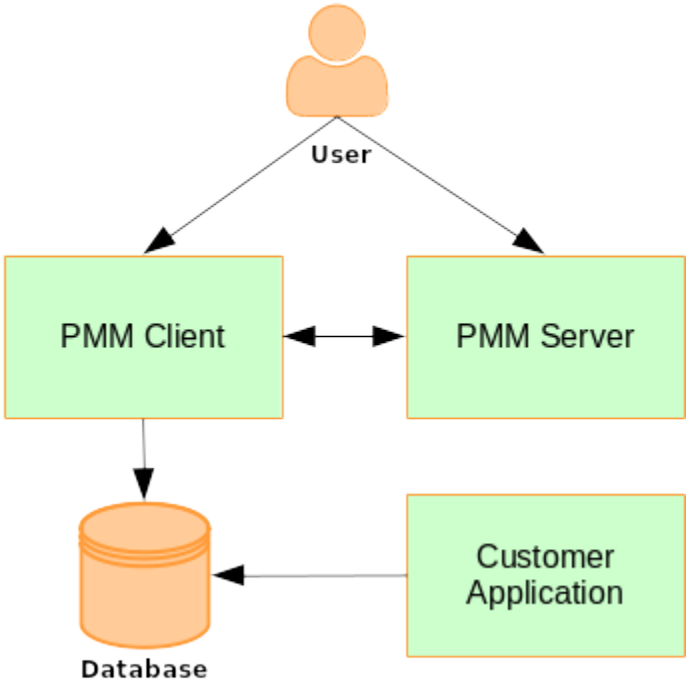
CHAPTER

ONE

CLIENT/SERVER ARCHITECTURE - AN OVERVIEW

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

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



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

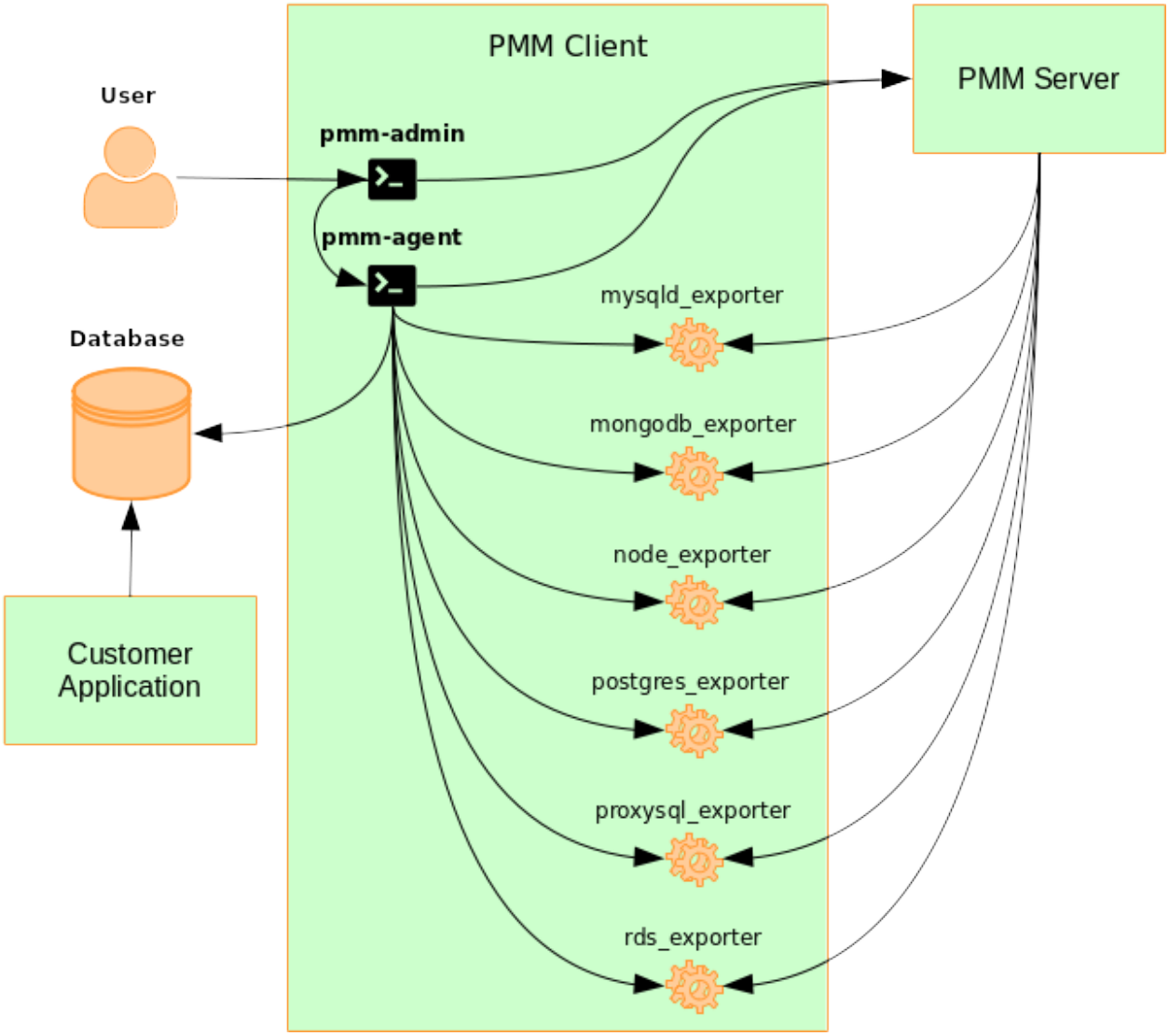
- *PMM Client*
- *PMM Server*

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

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

The following sections illustrates how PMM is currently structured.

1.1 PMM Client



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

The PMM Client package consist of the following:

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

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

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

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

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

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

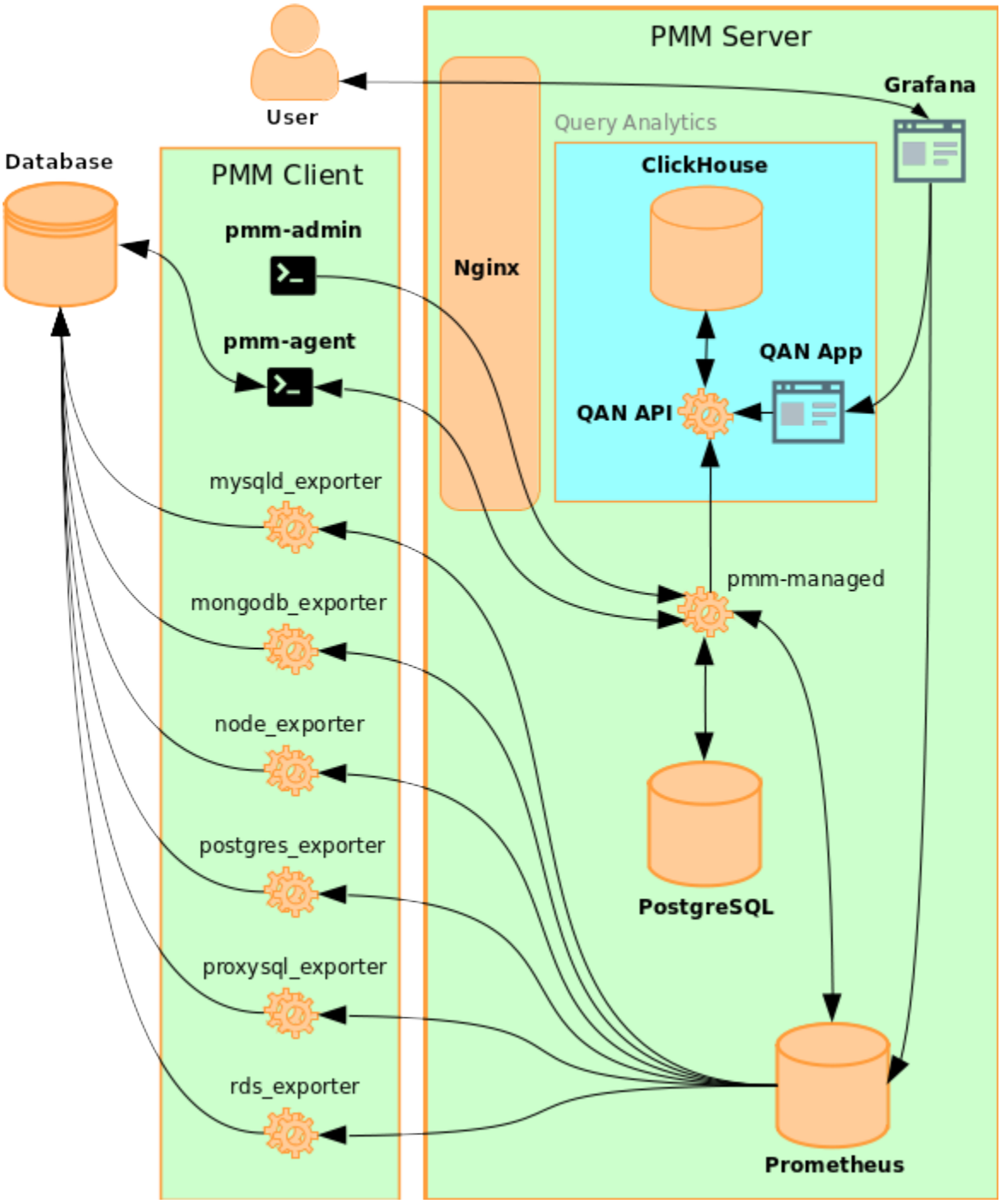
See also:

How to install PMM Client *Installing Clients*

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

List of available exporter options pmm.list.exporter

1.2 PMM Server



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

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

For more information, see Installing PMM Server.

PMM Server includes the following tools:

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

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

CHAPTER

TWO

SERVICES

2.1 MySQL requirements

PMM supports all commonly used variants of *MySQL*, including |percona-server|, |mariadb|, and |amazon-rds|. To prevent data loss and performance issues, *PMM* does not automatically change *MySQL* configuration. However, there are certain recommended settings that help maximize monitoring efficiency. These recommendations depend on the variant and version of *MySQL* you are using, and mostly apply to very high loads.

PMM can collect query data either from the |slow-query-log| or from |performance-schema|. The |slow-query-log| provides maximum details, but can impact performance on heavily loaded systems. On |percona-server| the query sampling feature may reduce the performance impact.

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

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

You can add configuration examples provided 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 |abbr.qan|.

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

2.1.1 Creating a MySQL User Account to Be Used with PMM

When adding a *MySQL* instance to monitoring, you can specify the *MySQL* server superuser account credentials. However, monitoring with the superuser account is not secure. It's better to 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 Service Monitoring*

For example can 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 --username root --password root
```

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

```
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';
```

For more information, run as root |pmm-admin.add| |opt.mysql| |opt.help|.

2.2 Configuring MongoDB for Monitoring in *PMM Query Analytics*

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

Supported versions of MongoDB

QAN supports MongoDB version 3.2 or higher.

2.2.1 Setting Up the Required Permissions

For MongoDB monitoring services to be able work in QAN, you need to set up the **mongodb_exporter** user. This user should be assigned the *clusterMonitor* 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.

See also:

Adding a `mongodb:metrics` monitoring service pmm-admin.add.mongodb-metrics

2.2.2 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

Note that you need to specify a path to an existing directory that stores database files with the `--dpbath`. 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 pmm.qan-mongodb.conf.profiling.command_line.enable. Note that the configuration file is in the YAML format. In this format the indentation of your lines is important as it defines levels of nesting.

Restart the *mongod* service to enable the settings.

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

Related Information

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

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

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

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

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

---

2.3 PostgreSQL requirements

2.3.1 Supported versions of PostgreSQL

PMM follows postgresql.org EOL policy, and thus supports monitoring *PostgreSQL* version 9.4 and up. Older versions may work, but will not be supported. For additional assistance, visit Percona PMM Forums.

Part II

Installing PMM Server

CHAPTER

THREE

RUNNING PMM SERVER VIA DOCKER

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

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

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

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

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

3.1 Setting Up a Docker Container for PMM Server

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

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

3.1.1 Pulling the PMM Server Docker Image

To pull the latest version from Docker Hub:

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

This step is not required if you are running PMM Server for the first time. However, it ensures that if there is an older version of the image tagged with `2.1.0` available locally, it will be replaced by the actual latest version.

3.1.2 Creating the pmm-data Container

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

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

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

The previous command does the following:

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

Important: 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.

3.1.3 Creating and Launching the PMM Server Container

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

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

This command does the following:

- The **`docker run`** command runs a new container based on the `percona/pmm-server:2` image.
- The `-p` option maps the 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:2` is the name and version tag of the image to derive the container from.

3.1.4 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 2.0 with disabled update button in the web interface would look as follows:

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

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

See also:

Updating PMM Updating PMM

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

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

3.2 Updating PMM Server Using Docker

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

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

```
$ docker ps
CONTAINER ID   IMAGE                         COMMAND                  CREATED       STATUS␣
↪             PORTS                                 NAMES
480696cd4187   percona/pmm-server:1.4.0   "/opt/entrypoint.sh"   4 weeks ago   Up␣
↪About an hour   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.

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

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

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

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

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

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

3.2.2 Pulling a new Docker Image

Docker images for all versions of PMM are available from percona/pmm-server 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:2
```

3.2.3 Creating a new Docker container based on the new image

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

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

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

The **`docker run`** command refers to the pulled image as the last parameter. If you used a specific version number when running **`docker pull`** (see *Pulling the PMM Server Docker Image*) replace `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 4
↪minutes    192.168.100.1:80->80/tcp, 443/tcp    pmm-server
```

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

3.2.4 Removing the backup container

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

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

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

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

Restoring the previous version

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

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

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

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

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

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

See also:

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

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

Updating the PMM Server and the PMM Client deploy-pmm.updating section.

3.3 Backing Up PMM Data from the Docker Container

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

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

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

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

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

2. Create the essential sub directory:

```
$ mkdir srv
```

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

1. Stop the docker container:

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

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

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

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

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

See also:

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

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

3.4 Restoring the Backed Up Information to the PMM Data Container

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

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

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

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

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

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

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

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

4. Create a new `pmm-data` container

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

Important: The last step creates a new `pmm-data` container based on the `percona/pmm-server:2` image. If you do not intend to use the `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:2 chown -R␣
↪pmm:pmm /opt/prometheus/data /opt/consul-data
$ docker run --rm --volumes-from pmm-data -it percona/pmm-server:2 chown -R␣
↪grafana:grafana /var/lib/grafana
$ docker run --rm --volumes-from pmm-data -it percona/pmm-server:2 chown -R␣
↪mysql:mysql /var/lib/mysql
```

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

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

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

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

See also:

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

Updating PMM Updating PMM

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

CHAPTER

FOUR

VERIFYING PMM SERVER

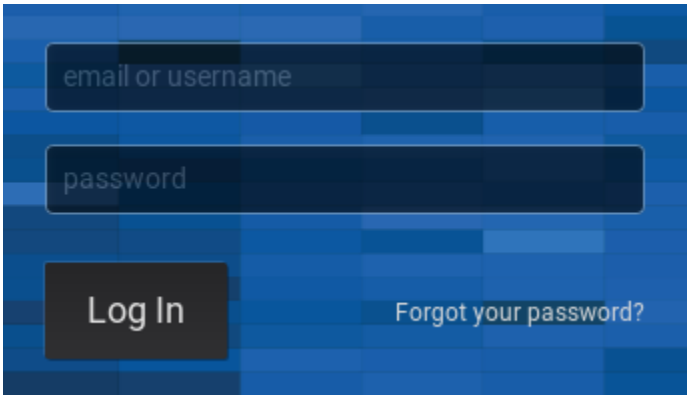
In your browser, go to the server by its IP address. If you run your server as a virtual appliance or by using an Amazon machine image, you will need to setup the user name, password and your public key if you intend to connect to the server by using ssh. This step is not needed if you run PMM Server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

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

Table 4.1: Accessing the Components of the Web Interface

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

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



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

Note: You will use the same credentials at connecting your PMM Client to PMM Server.

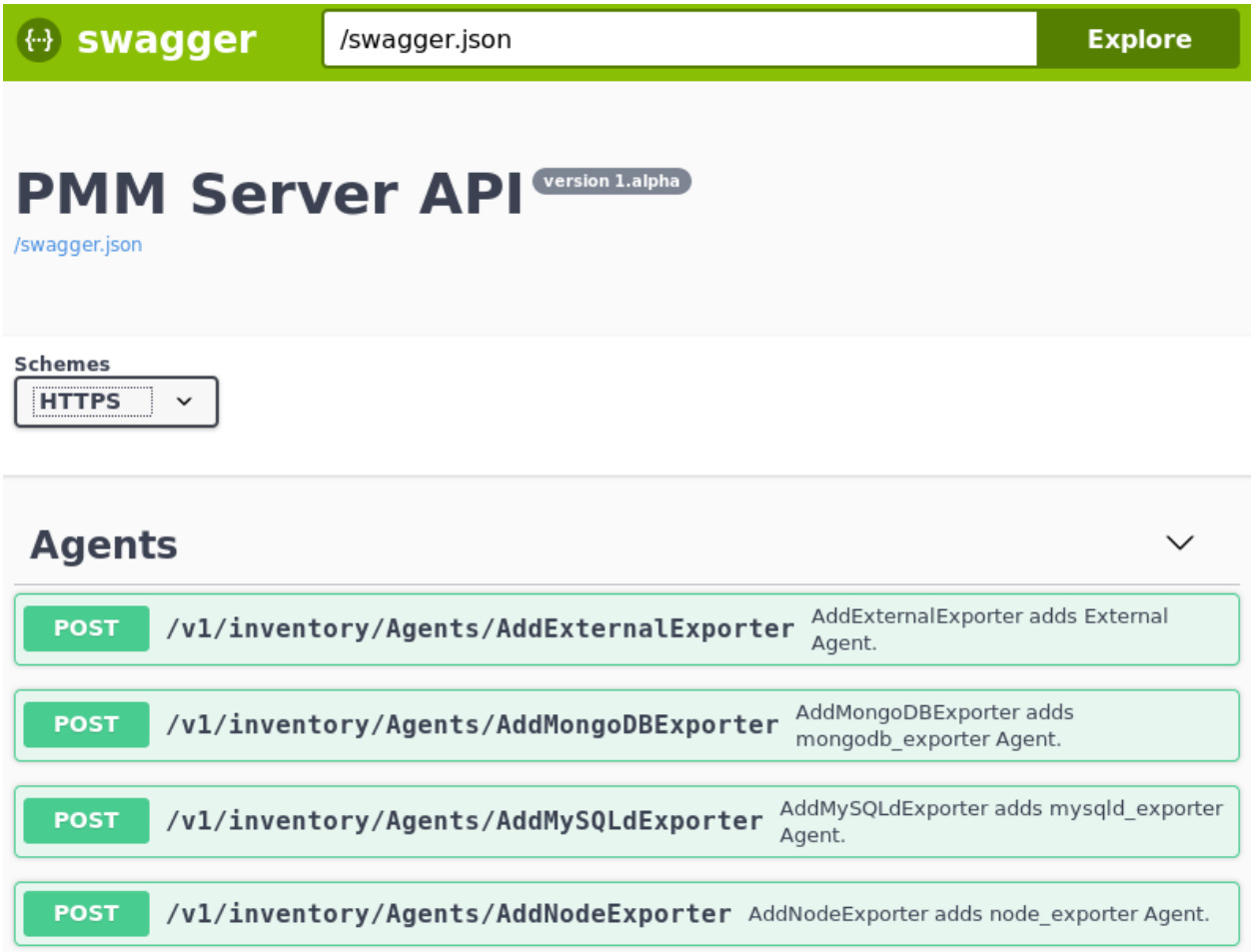
CHAPTER

FIVE

CONFIGURING PMM SERVER

5.1 Exploring PMM API

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



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

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

- A **Node** represents a bare metal server, virtual machine or Docker container. It can also be of more specific type:

Curl

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

Request URL

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

Server response

Code	Details
200	Response body

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

one example is Amazon RDS Node. Node runs zero or more Services and Agents. It also has zero or more Agents providing insights for it.

- A **Service** represents something useful running on the Node: Amazon Aurora MySQL, MySQL, MongoDB, Prometheus, etc. It runs on zero (Amazon Aurora Serverless), single (MySQL), or several (Percona XtraDB Cluster) Nodes. It also has zero or more Agents providing insights for it.
- An **Agent** represents something that runs on the Node which is not useful itself but instead provides insights (metrics, query performance data, etc) about Nodes and/or Services. Always runs on the single Node (except External Exporters), provides insights for zero or more Services and Nodes.

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

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

Part III

Installing PMM Client

CHAPTER

SIX

INSTALLING CLIENTS

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

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

```
$ ping 192.168.100.1
```

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

Supported platforms

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

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

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

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

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

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

Storage requirements

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

CHAPTER

SEVEN

INSTALLING DEB PACKAGES USING APT-GET

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

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

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

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

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

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

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

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

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

   See percona-release official documentation for details.

2. Install the `pmm2-client` package:

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

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

```
pmm-admin config --server-insecure-tls --server-address=<IP Address>:443
```

You should see the following:

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

CHAPTER

EIGHT

INSTALLING RPM PACKAGES USING YUM

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

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

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

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

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

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

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

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

   See percona-release official documentation for details.

2. Install the `pmm2-client` package:

   ```
   yum install pmm2-client
   ```

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

   ```
   pmm-admin config --server-insecure-tls --server-address=<IP Address>:443
   ```

   You should see the following:

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

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

Part IV

Using PMM Client

CHAPTER

NINE

CONFIGURING PMM CLIENT WITH PMM-ADMIN CONFIG

9.1 Connecting PMM Clients to the PMM Server

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

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

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

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

```
$ pmm-admin config --server-insecure-tls --server-url=https://admin:admin@192.168.100.
↪1:443
```

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

```
# pmm-admin config --server-insecure-tls --server-url=https://admin:admin@192.168.100.
↪1:443
Checking local pmm-agent status...
pmm-agent is running.
Registering pmm-agent on PMM Server...
Registered.
Configuration file /usr/local/percona/pmm-agent.yaml updated.
Reloading pmm-agent configuration...
Configuration reloaded.
Checking local pmm-agent status...
pmm-agent is running.
```

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

CHAPTER

TEN

ADDING MYSQL SERVICE MONITORING

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

USAGE

```
pmm-admin add mysql --query-source=slowlog --username=pmm --password=pmm 127.0.0.
↪1:3306
```

where username and password are credentials for the monitored MySQL access, which will be used locally on the database host. Additionally, a service name can be appended to the command line parameters, otherwise it will be generated automatically as `<node>-mysql`.

The output of this command may look as follows:

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

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

```
pmm-admin add mysql --username=pmm --password=pmm --query-source=perfschema 127.0.0.
↪1:3306
```

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

CHAPTER

ELEVEN

ADDING MONGODB SERVICE MONITORING

Before adding MongoDB should be prepared for the monitoring, which involves creating the user, and setting the profiling level.

When done, add monitoring as follows:

```
pmm-admin add mongodb  --username=pmm  --password=pmm 127.0.0.1:27017
```

where username and password are credentials for the monitored MongoDB access, which will be used locally on the database host. Additionally, a service name can be appended to the command line parameters, otherwise it will be generated automatically as `<node>-mongodb`.

The output of this command may look as follows:

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

$ pmm-admin add mongodb –use-profiler –username=pmm –password=pmm

–cluster='MongoDBCluster1' –replication-set='MongoDBReplSet2' –environment='Production' –custom-labels='az=sfo2' 127.0.0.1:27017 mongodb1

where username and password are credentials for the monitored MongoDB access, * –use-profiler - enable query capture * –username - MongoDB username * –password - MongoDB Password * –cluster - MongoDBCluster1 * –replication-set - MongoDBReplSet1 * –environment - Production, Staging, Development * –custom-labels - arbitrary key=value pairs which will be used locally on the database host.

You can then check your MySQL and MongoDB dashboards and Query Analytics in order to view your server's performance information.

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 pmm.qan-mongodb.conf

CHAPTER

TWELVE

ADDING A PROXYSQL HOST

12.1 Adding ProxySQL metrics service

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

USAGE

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

where username and password are credentials for the monitored MongoDB access, which will be used locally on the database host. Additionally, a service name can be appended to the command line parameters, otherwise it will be generated automatically as `<node>-proxysql`.

The output of this command may look as follows:

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

CHAPTER

THIRTEEN

ADDING LINUX METRICS

13.1 Adding general system metrics service

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

CHAPTER

FOURTEEN

POSTGRESQL

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

14.1 Adding PostgreSQL extension for queries monitoring

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

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

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

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

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

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

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

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

14.2 Adding PostgreSQL queries and metrics monitoring

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

```
pmm-admin add postgresql --username=pmm --password=pmm 127.0.0.1:5432
```

where username and password parameters should contain actual PostgreSQL user credentials (for more information about `pmm-admin add`, see pmm-admin.add). Additionally, a service name can be appended to the command line parameters, otherwise it will be generated automatically as `<node>-postgresql`.

The output of this command may look as follows:

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

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

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();
```

14.3 Setting up the required user permissions and authentication

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

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

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

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

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

CHAPTER

FIFTEEN

REMOVING MONITORING SERVICES WITH PMM-ADMIN REMOVE

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

USAGE

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

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

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

SERVICES

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

EXAMPLES

- Removing MySQL service named "mysql-sl":

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

- To remove *MongoDB* service named "mongo":

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

- To remove *PostgreSQL* service named "postgres":

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

- To remove *ProxySQL* service named "ubuntu-proxysql":

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

For more information, run **pmm-admin remove** –help.

Part V

Service configuration for best results

CHAPTER

SIXTEEN

MYSQL

16.1 Slow Log Settings

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

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

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

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

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

16.2 Configuring Performance Schema

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

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

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 dashboard-mysql-performance-schema dashboard. To enable full instrumentation, set the option `--performance_schema_instrument` to `'%=on'` when starting the MySQL server.

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

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

See also:

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

---

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

1. Open the *PMM Query Analytics* dashboard.
2. Click the *Settings* button.
3. Open the *Settings* section.
4. Select `Performance Schema` in the *Collect from* drop-down list.
5. Click *Apply* to save changes.

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

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

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

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

16.3 MySQL InnoDB Metrics

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

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

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

See also:

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

16.4 Percona Server specific settings

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

16.4.1 MySQL User Statistics (userstat)

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

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

See also:

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

MySQL Documentation Setting variables

16.4.2 Query Response Time Plugin

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

To enable collection of query response time:

1. Install the `QUERY_RESPONSE_TIME` plugins:

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

2. Set the global varible `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

16.4.3 log_slow_rate_limit

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

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

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

See also:

MySQL Documentation Setting variables

16.4.4 log_slow_verbosity

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

See also:

MySQL Documentation Setting variables

16.4.5 slow_query_log_use_global_control

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

See also:

MySQL Documentation Setting variables

16.5 Configuring MySQL 8.0 for PMM

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

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

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

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

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

Altering the MySQL User

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

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

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

Changing the global MySQL Configuration

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

See also:

Creating a MySQL User for PMM privileges

More information about adding the MySQL query analytics monitoring service pmm-admin.add-mysql-queries

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

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

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

CHAPTER

SEVENTEEN

MONGODB

17.1 Passing SSL parameters to the mongodb monitoring service

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

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

Listing 17.1: *Passing an SSL/TLS parameter to* **mongod** *to enable a TLS connection.*

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

Table 17.1: Supported SSL/TLS Parameters

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

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

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

Part VI

Using PMM Metrics Monitor

CHAPTER

EIGHTEEN

UNDERSTANDING DASHBOARDS

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

18.1 Opening a Dashboard

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

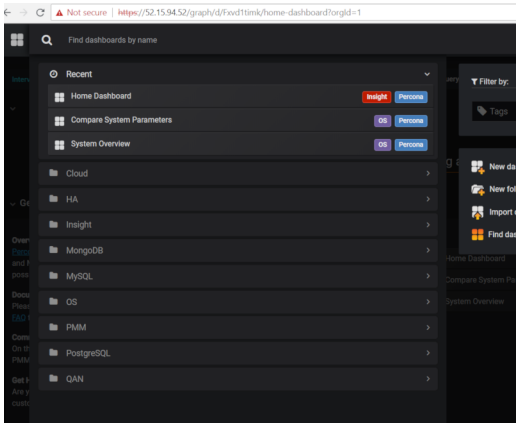


Fig. 18.1: With *Dashboard Dropdown*, search the alphabetical list for any dashboard.

18.2 Viewing More Information about a Graph

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

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

See also:

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

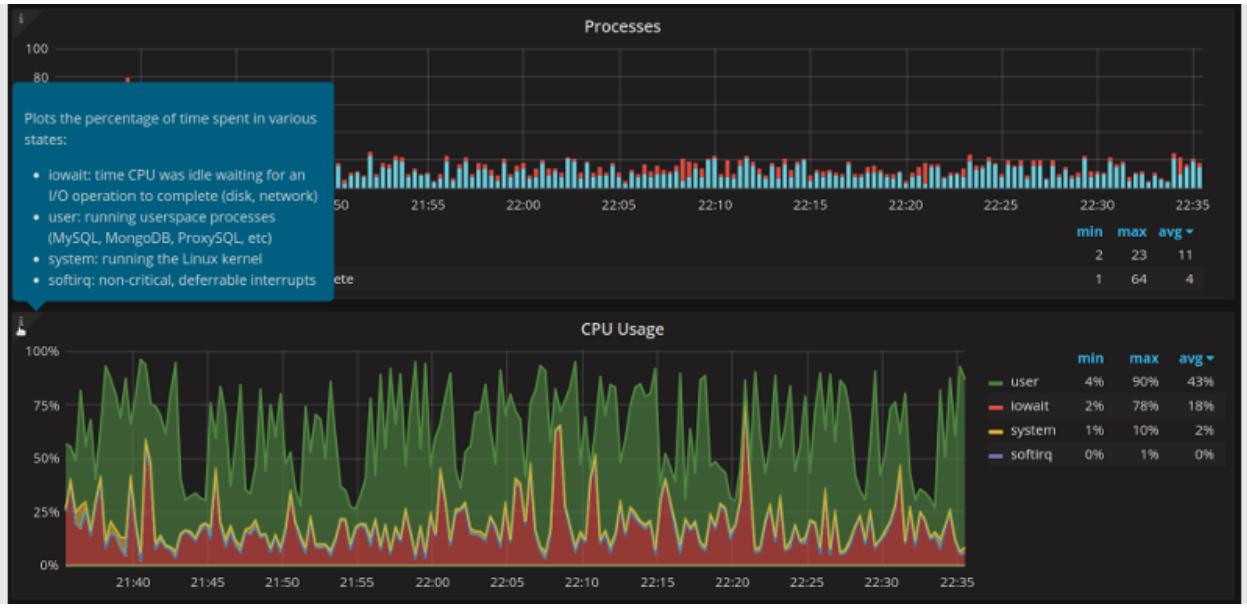


Fig. 18.2: Graph descriptions provide more information about a graph without claiming any space in the interface.

CHAPTER

NINETEEN

NAVIGATING ACROSS DASHBOARDS

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

Group	Dashboards for monitoring . . .
PMM Query Analytics	QAN component (see *Introduction*)
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

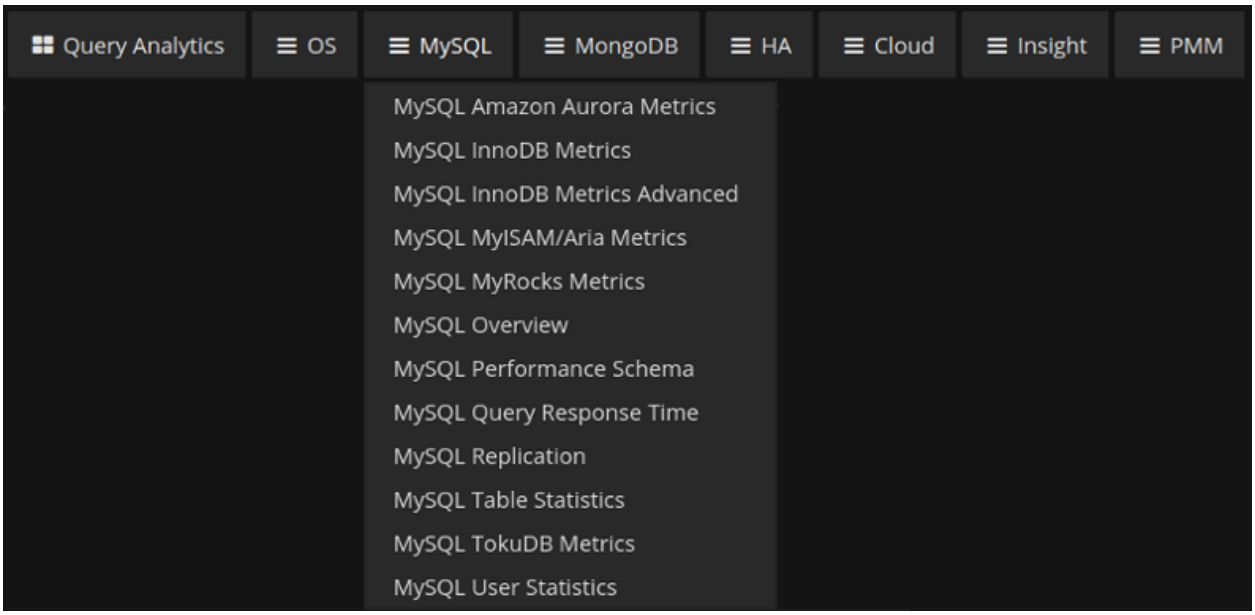


Fig. 19.1: MySQL group selected in the navigation menu

19.1 Zooming in on a single metric

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

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

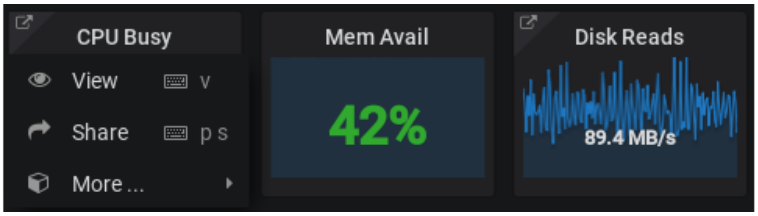
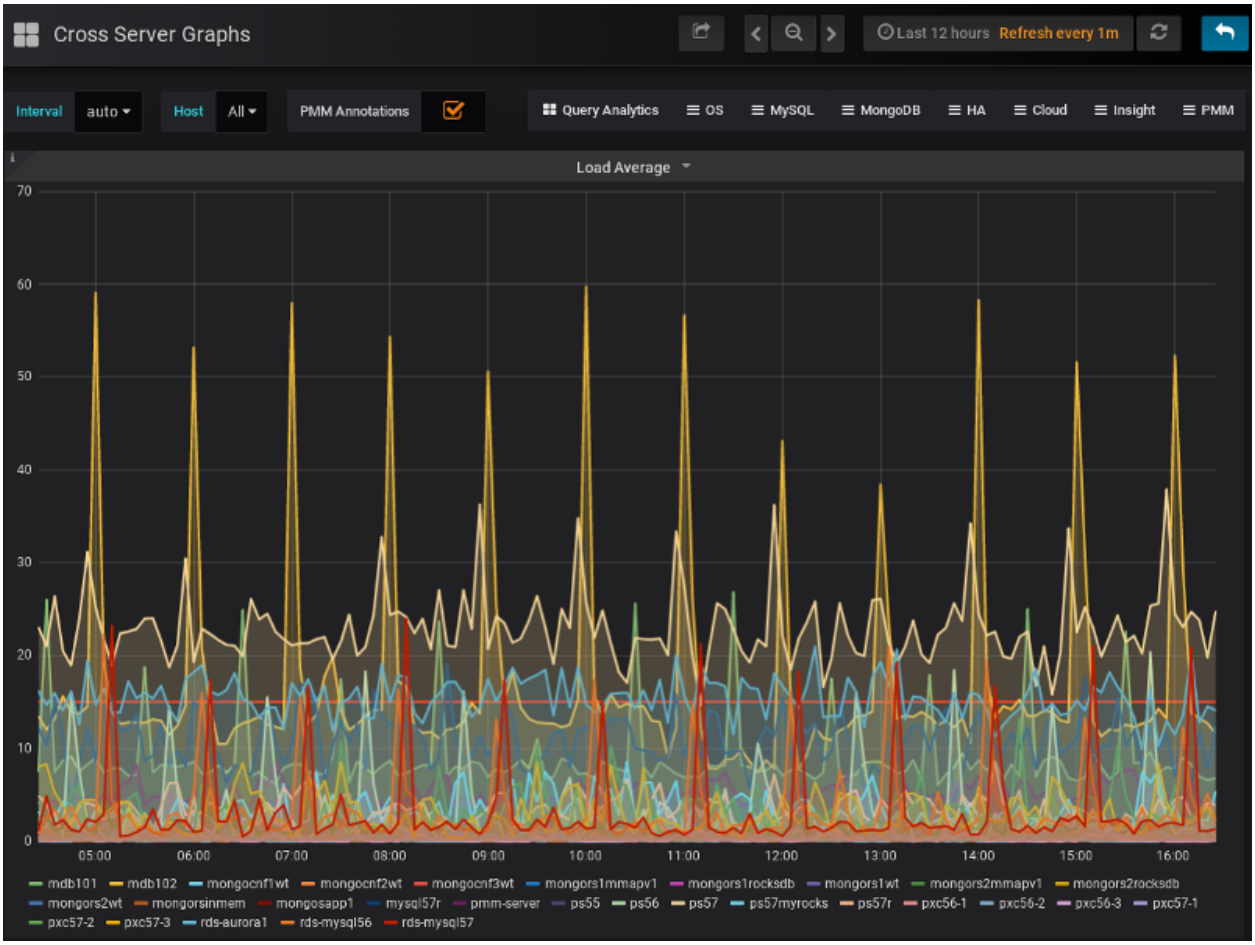


Fig. 19.2: The context menu of a metric

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



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

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

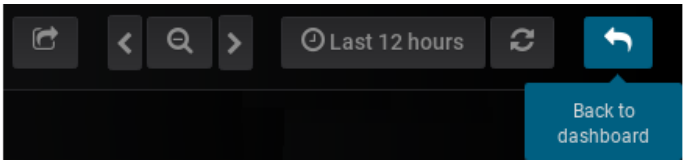


Fig. 19.3: The *Back to dashboard* button returns to the dashboard; this button appears when you are zooming in on one metric.

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

Part VII

Using PMM Query Analytics

CHAPTER

TWENTY

INTRODUCTION

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

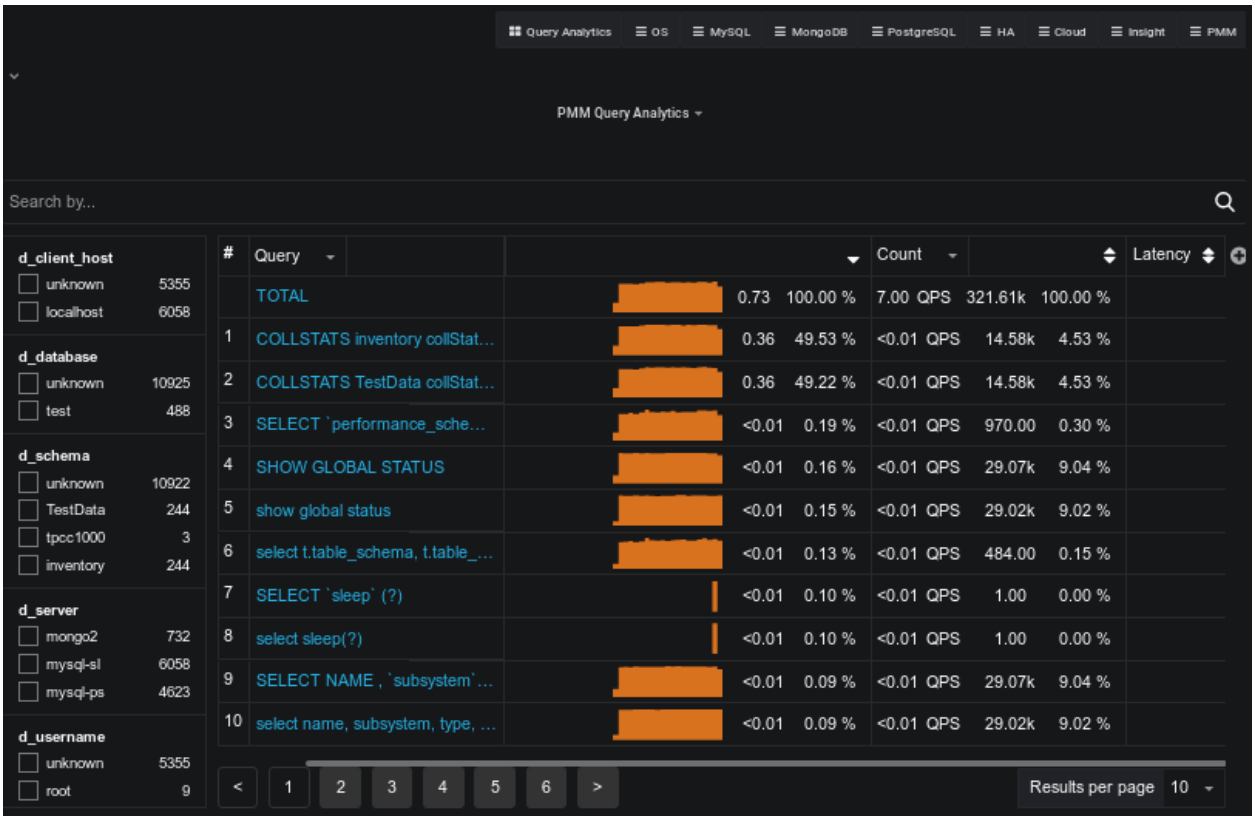


Fig. 20.1: QAN helps analyze database queries over periods of time and find performance problems.

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

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

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

CHAPTER

TWENTYONE

NAVIGATING TO QUERY ANALYTICS

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

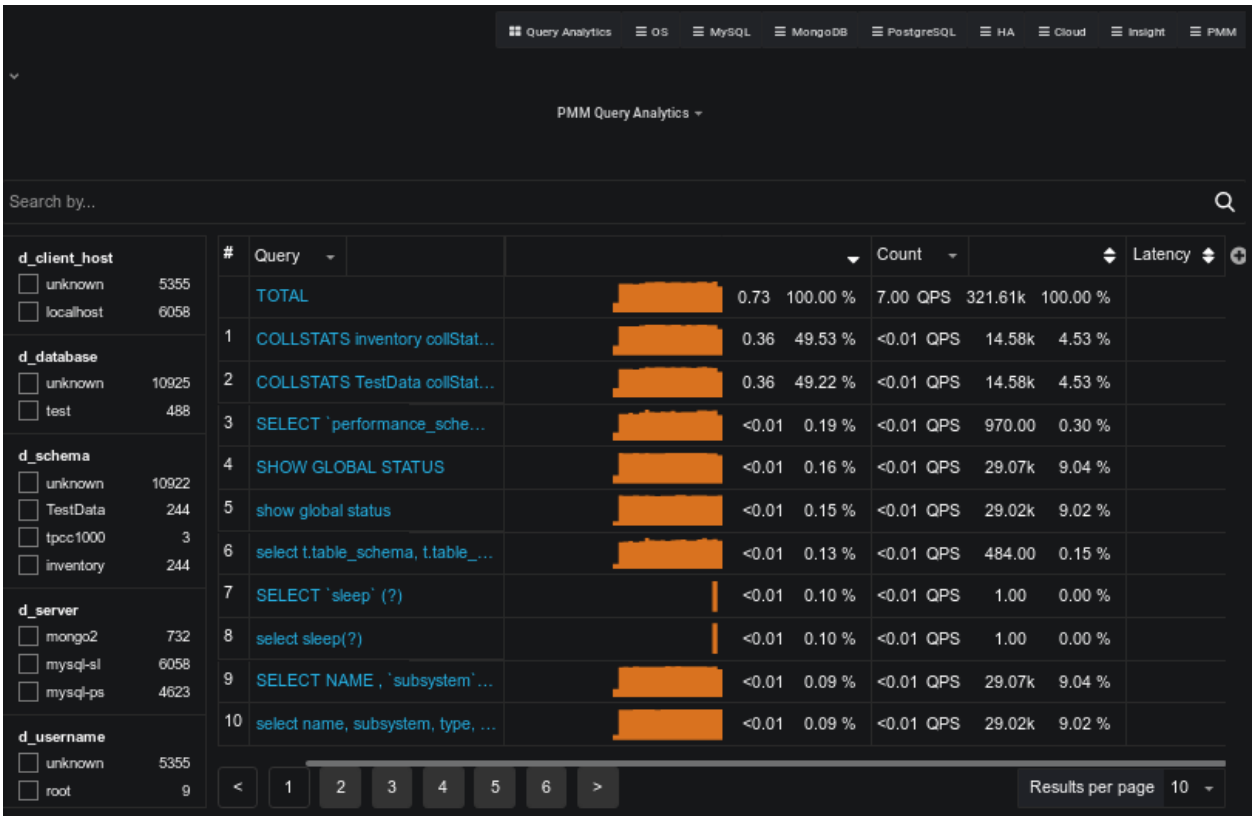


Fig. 21.1: The query summary table.

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

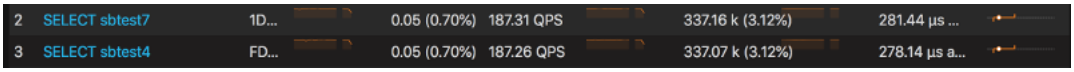


Fig. 21.2: Showing intervals for which data are unavailable yet.

CHAPTER

TWENTYTWO

UNDERSTANDING TOP 10

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

Fig. 22.1: The query summary table shows the monitored queries from the selected database.

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

22.1 Query Detail Section

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

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

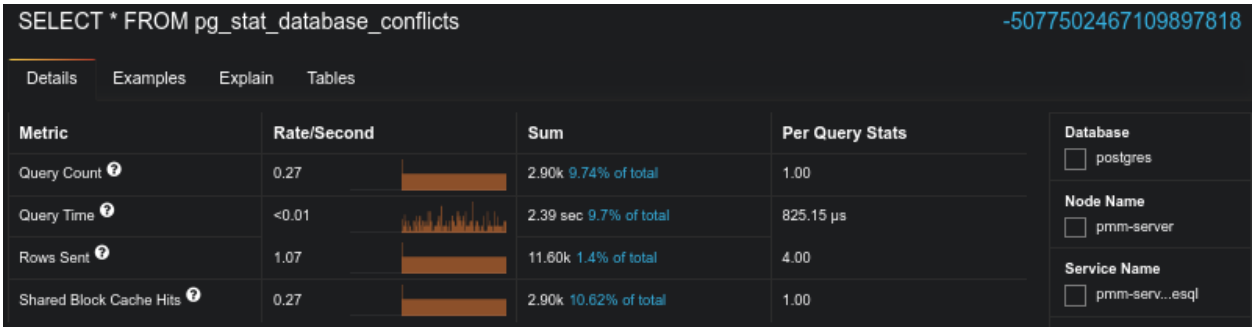


Fig. 22.2: The Query Detail section shows the SQL statement for the selected query.

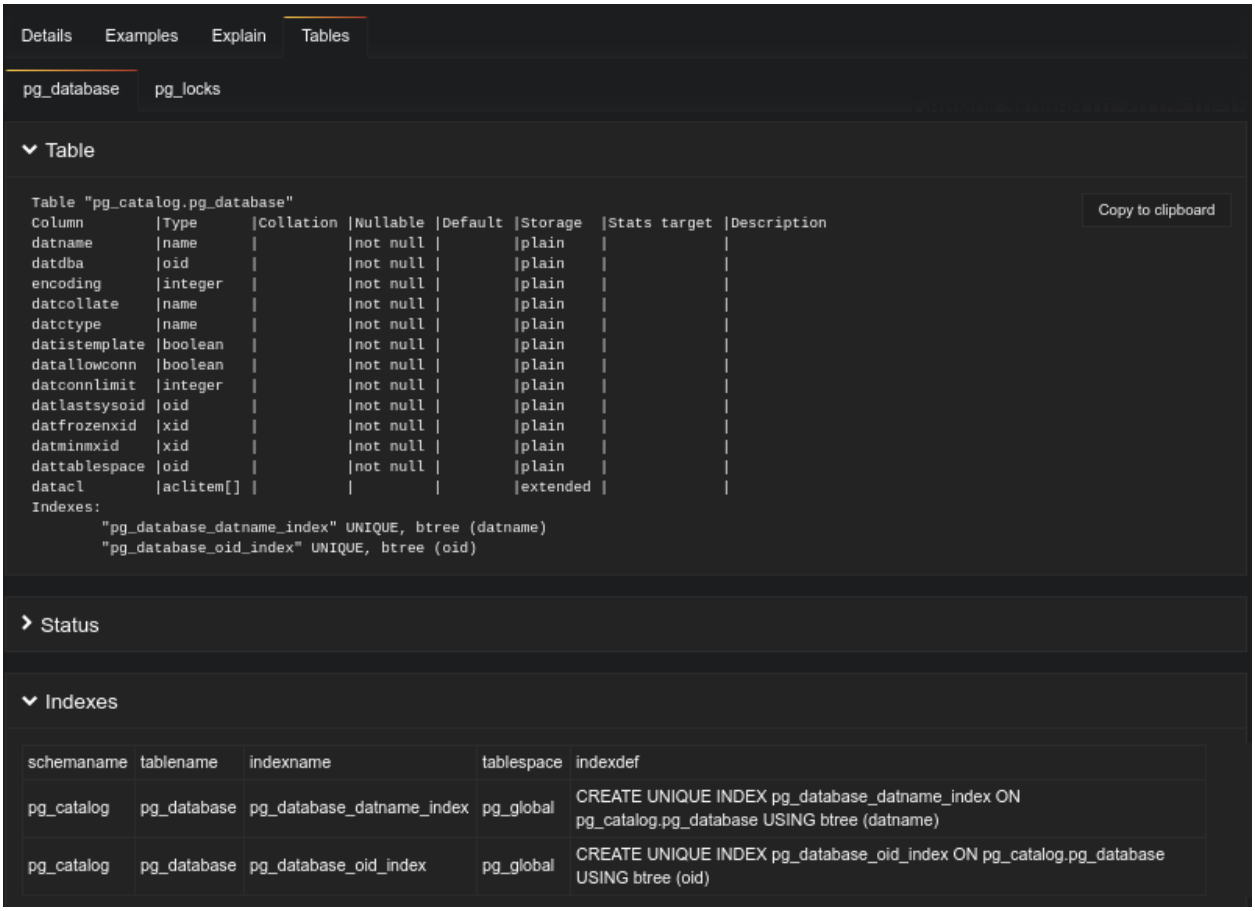


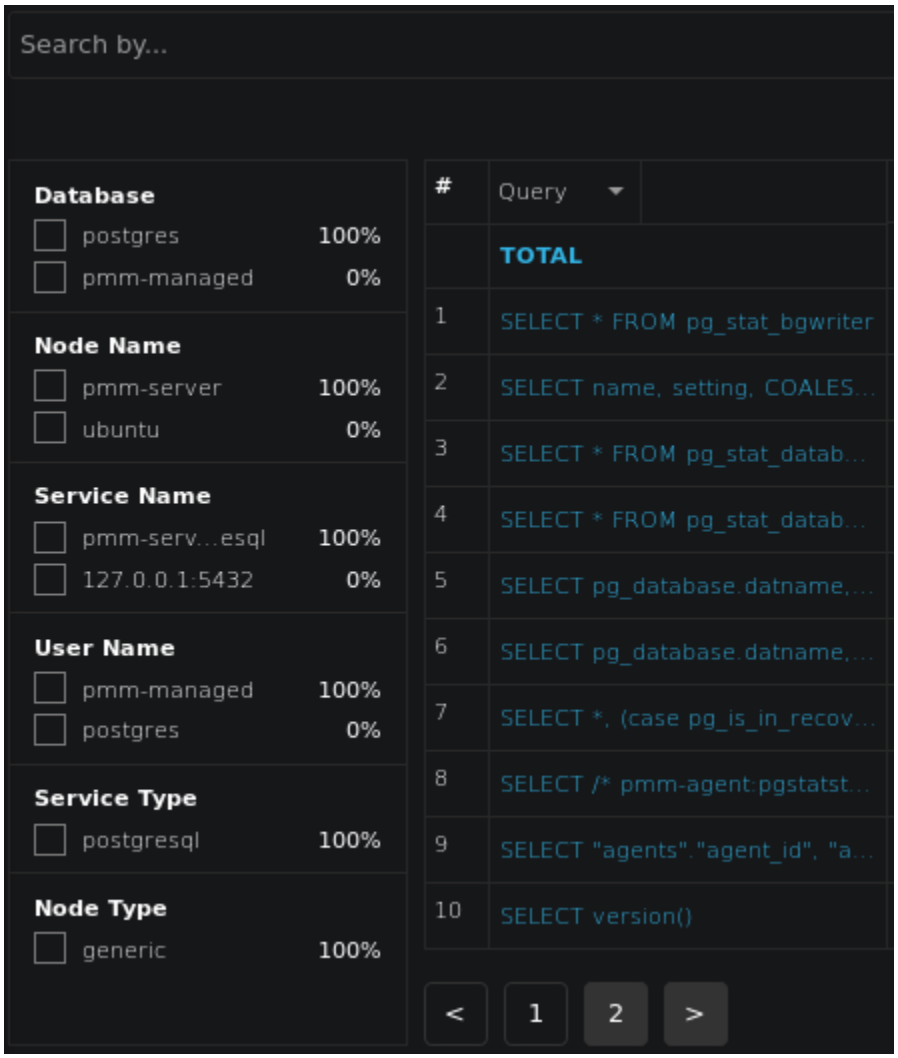
Fig. 22.3: Tables and indexes details the selected query.

CHAPTER

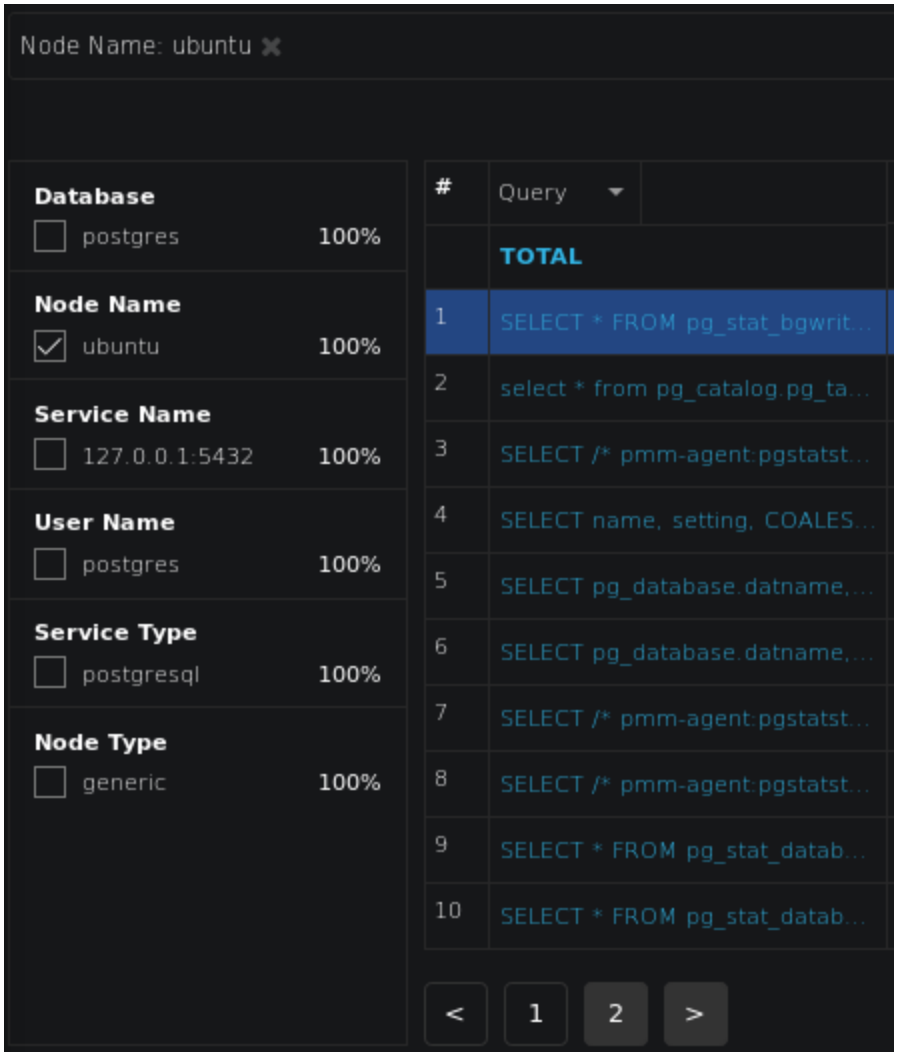
TWENTYTHREE

FILTERING QUERIES

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



Following example shows how to filter just the queries that are executed on the *Ubuntu* node (note that the filtering panel reflects only Labels available within the set of currently applied filters):


Node Name: ubuntu
Database
postgres 100%
Node Name
ubuntu 100%
Service Name
127.0.0.1:5432 100%
User Name
postgres 100%
Service Type
postgresql 100%
Node Type
generic 100%
Query
TOTAL
1 SELECT * FROM pg_stat_bgwrit...
2 select * from pg_catalog.pg_ta...
3 SELECT /* pmm-agent:pgstatst...
4 SELECT name, setting, COALES...
5 SELECT pg_database.datname,...
6 SELECT pg_database.datname,...
7 SELECT /* pmm-agent:pgstatst...
8 SELECT /* pmm-agent:pgstatst...
9 SELECT * FROM pg_stat_datab...
10 SELECT * FROM pg_stat_datab...
< 1 2 >

Selecting Time or Date Range

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

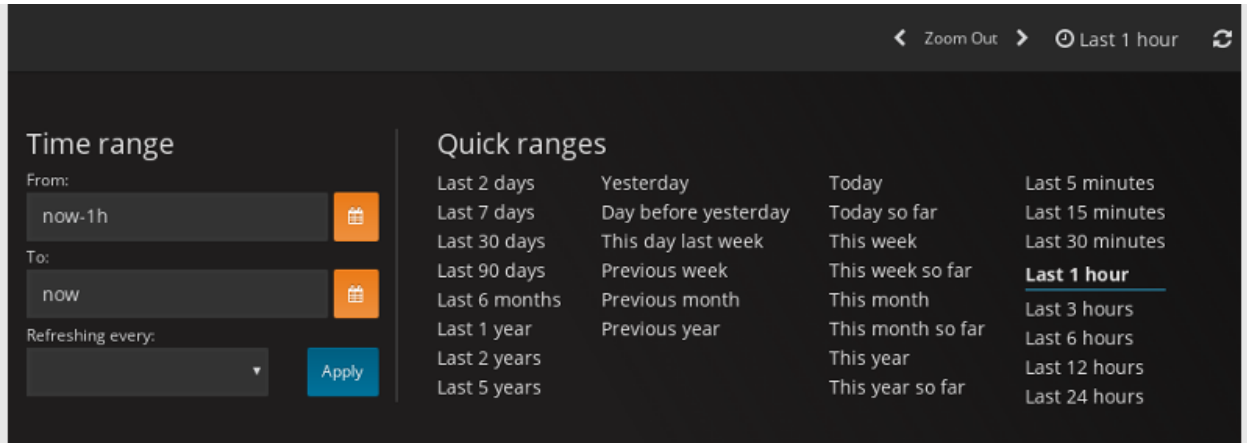


Fig. 23.1: QAN displays query metrics for the time period or date range that you specify.

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

23.1 Totals of the Query Summary

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

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

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

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

23.2 Queries in the Query Summary Table

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

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

23.3 Viewing a Specific Value of a Metric

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

Fig. 23.2: *Hover the cursor to see a value at the point.*

CHAPTER

TWENTYFOUR

MONGODB SPECIFIC

24.1 Query Analytics for MongoDB

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

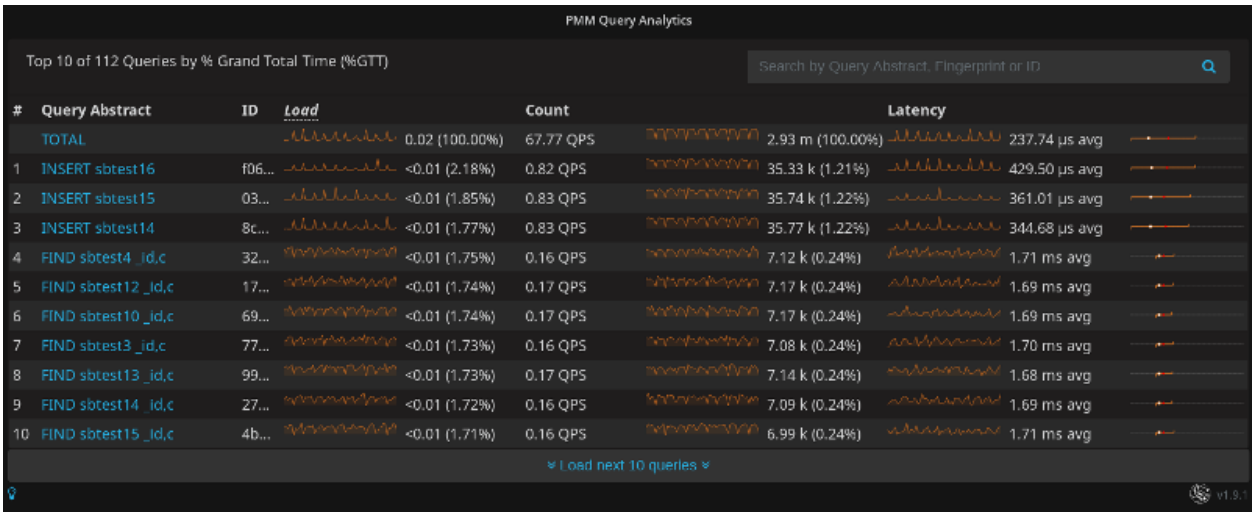


Fig. 24.1: A list of queries from a MongoDB host

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

See also:

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

Fig. 24.2: Analyze MongoDB queries using the same tools as relational database management systems.

Part VIII

Percona Monitoring and Management Release Notes

CHAPTER

TWENTYFIVE

PERCONA MONITORING AND MANAGEMENT 2.1.0

Date November 11, 2019

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

For install instructions, see deploy-pmm.

Note: PMM 2 is designed to be used as a new installation — please don't try to upgrade your existing PMM 1 environment.

25.1 Improvements and new features

- PMM-4063: Update QAN filter panel to show only labels available for selection under currently applied filters
- PMM-815: Latency Detail graph added to the MongoDB Instance Summary dashboard
- PMM-4768: Disable heavy-load collectors automatically when there are too many tables
- PMM-4821: Use color gradient in filled graphs on all dashboards
- PMM-4733: Add more log and config files to the downloadable `logs.zip` archive
- PMM-4672: Use integer percentage values in QAN filter panel
- PMM-4857: Update tooltips for all MongoDB dashboards
- PMM-4616: Rename column in the Query Details section in QAN from Total to Sum
- PMM-4770: Use Go 1.12.10
- PMM-4780: Update Grafana to version 6.4.1
- PMM-4918: Update Grafana plugins to newer versions, including the `clickhouse-datasource` plugin

25.2 Fixed bugs

- PMM-4935: Wrong instance name displayed on the MySQL Instance Summary dashboard due to the incorrect string crop
- PMM-4916: Wrong values are shown when changing the time range for the Node Summary Dashboard in case of remote instances

- PMM-4895 and PMM-4814: The update process reports completion before it is actually done and therefore some dashboards, etc. may not be updated
- PMM-4876: PMM Server access credentials are shown by the `pmm-admin status` command instead of hiding them for security reasons
- PMM-4875: PostgreSQL error log gets flooded with warnings when `pg_stat_statements` extension is not installed in the database used by PMM Server or when PostgreSQL user is unable to connect to it
- PMM-4852: Node name has an incorrect value if the Home dashboard opened after QAN
- PMM-4847: Drilldowns from the Environment Overview dashboard doesn't show data for the pre-selected host
- PMM-4841 and PMM-4845: `pg_stat_statement` QAN Agent leaks database connections
- PMM-4831: Clean-up representation of selectors names on MySQL-related dashboards for a better consistency
- PMM-4824: Incorrectly calculated singlestat values on MySQL Instances Overview dashboard
- PMM-4819: In case of the only one monitored host, its uptime is shown as a smaller value than the all hosts uptime due to the inaccurate rounding
- PMM-4816: Set equal thresholds to avoid confusing singlestat color differences on a Home dashboard
- PMM-4718: Labels are not fully displayed in the filter panel of the Query Details section in QAN
- PMM-4545: Long queries are not fully visible in the Query Examples section in QAN

Help us improve our software quality by reporting any Percona Monitoring and Management bugs you encounter using our bug tracking system.

CHAPTER

TWENTYSIX

PERCONA MONITORING AND MANAGEMENT 2.0.1

Date October 9, 2019

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

For install instructions, see deploy-pmm.

Note: PMM 2 is designed to be used as a new installation — please don't try to upgrade your existing PMM 1 environment.

26.1 Improvements

- PMM-4779: Securely share dashboards with Percona
- PMM-4735: Keep one old slowlog file after rotation
- PMM-4724: Alt+click on check updates button enables force-update
- PMM-4444: Return "what's new" URL with the information extracted from the pmm-update package changelog

26.2 Fixed bugs

- PMM-4758: Remove Inventory rows from dashboards
- PMM-4757: `qan_mysql_perfschema_agent` failed querying `events_statements_summary_by_digest` due to data types conversion
- PMM-4755: Fixed a typo in the InnoDB AHI Miss Ratio formula
- PMM-4749: Navigation from Dashboards to QAN when some Node or Service was selected now applies filtering by them in QAN
- PMM-4742: General information links were updated to go to PMM 2 related pages
- PMM-4739: Remove request instances list
- PMM-4734: A fix was made for the collecting `node_name` formula at MySQL Replication Summary dashboard
- PMM-4729: Fixes were made for formulas on MySQL Instances Overview

- PMM-4726: Links to services in MongoDB singlestats didn't show Node name
- PMM-4720: `machine_id` could contain trailing `\n`
- PMM-4640: It was not possible to add MongoDB remotely if password contained a `#` symbol

Help us improve our software quality by reporting any Percona Monitoring and Management bugs you encounter using our bug tracking system.

CHAPTER

TWENTYSEVEN

PERCONA MONITORING AND MANAGEMENT 2.0.0-RC3

Date September 16, 2019

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

For install instructions, see deploy-pmm.

Note: This release is not recommended for production environments. PMM 2 is designed to be used as a new installation — please don't try to upgrade your existing PMM 1 environment.

27.1 Improvements

- PMM-3786: API clean-ups
- PMM-4686: Add a Service Name linked to a MySQL Summary on the Details page, and a Node row
- PMM-4689: Add TLS support for the Remote UI
- PMM-4255: Replace –use-perfschema and –use-slowlog options with –query-source
- PMM-4635: Add the Total Number of Rows in the Pager
- PMM-4688: put dashboards with new tags in folders
- PMM-4699: Always enable MySQL Table Statistics (for 2.0)

27.2 Fixed bugs

- PMM-4693: Invalid enum values for Agent type
- PMM-3989: Hide password on PMM Inventory page and don't return them by API
- PMM-3181: MySQL instance which is configured with SSL cannot be added remotely from UI

Help us improve our software quality by reporting any Percona Monitoring and Management bugs you encounter using our bug tracking system.

CHAPTER

TWENTYEIGHT

PERCONA MONITORING AND MANAGEMENT 2.0.0-RC2

Date September 13, 2019

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

For install instructions, see deploy-pmm.

Note: This release is not recommended for production environments. PMM 2 is designed to be used as a new installation — please don't try to upgrade your existing PMM 1 environment.

28.1 Improvements

- PMM-4447: Use /v1/Updates/Perform API to update PMM Server
- PMM-3480: Pass all QAN filtering parameters in URL to alow copy-paste
- PMM-3882: Improved pmm-agent logging on CentOS 6
- PMM-4362: Hide pt-summary and pt-mysql-summary actions until correspondent features are implemented in 2.x
- PMM-4459: Add flag –disable-queryexamples for QAN Agents for MySQL
- PMM-4531: Add "Show top 5" and "Show all" for the QAN filter section
- PMM-4612: Use Query Analytics name instead of QAN to be shown among dashboard folders
- PMM-4614: QAN Filter section clean-up
- PMM-4624: skip Row Affected metric in SELECT queries with PS Slow Log
- PMM-4638: PMM Dashboards clean-up in the main menu
- PMM-4642: Dashboards improvements and fixes after beta7
- PMM-4648: MySQL slowlog implementation
- PMM-4663: TLS support for all Agents and Actions
- PMM-4671: Name for Node and Service for pmm-server

28.2 Fixed bugs

- PMM-4652: Disk Read singlestat on the Home dashboard has incorrect link
- PMM-4675: Minor problems with singlestats on MySQL Instance Summary dashboard
- PMM-4678: There is no Details section for the Total row
- PMM-4680: exporter_build_info metrics are missed for all exporters except node_exporter

Help us improve our software quality by reporting any Percona Monitoring and Management bugs you encounter using our bug tracking system.

CHAPTER

TWENTYNINE

PERCONA MONITORING AND MANAGEMENT 2.0.0-RC1

Date September 11, 2019

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

For install instructions, see deploy-pmm.

Note: This release is not recommended for production environments. PMM 2 is designed to be used as a new installation — please don't try to upgrade your existing PMM 1 environment.

29.1 Improvements

- PMM-4564: Replace license for PMM2 Client
- PMM-3294: Update TLS cipher suite & other parameters
- PMM-3670: Cleanup pmm-update's playbook
- PMM-4042: Better pmm-admin help output
- PMM-4498: Don't display "/sec" for Time Metrics on sparkline
- PMM-4534: Support SSL/TLS for postgres_exporter
- PMM-4536: Support SSL/TLS for PostgreSQL connection checker
- PMM-4622: new Dashboards naming and URLs

29.2 Fixed bugs

- PMM-4647: Empty charts/singlestats on the CPU Utilization dashboard
- PMM-3704: Remove old landing page
- PMM-4357: Remove systemd remnants
- PMM-4649: Error in Scraped Target by Job Unassigned
- PMM-4666: pmm-update cleanup
- PMM-4667: Correct typo in Disk Details dashboard name

Help us improve our software quality by reporting any Percona Monitoring and Management bugs you encounter using our bug tracking system.

Part IX

Metrics Monitor Dashboards

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

CHAPTER

THIRTY

INSIGHT

30.1 Home Dashboard

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

See also:

Overview of PMM using

30.1.1 General Information

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

30.1.2 Shared and Recently Used Dashboards

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

30.1.3 Statistics

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

30.1.4 Environment Overview

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

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

- DB Uptime

30.2 *PMM System Summary* Dashboard

The *PMM System Summary* dashboard shows detailed infromation 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.

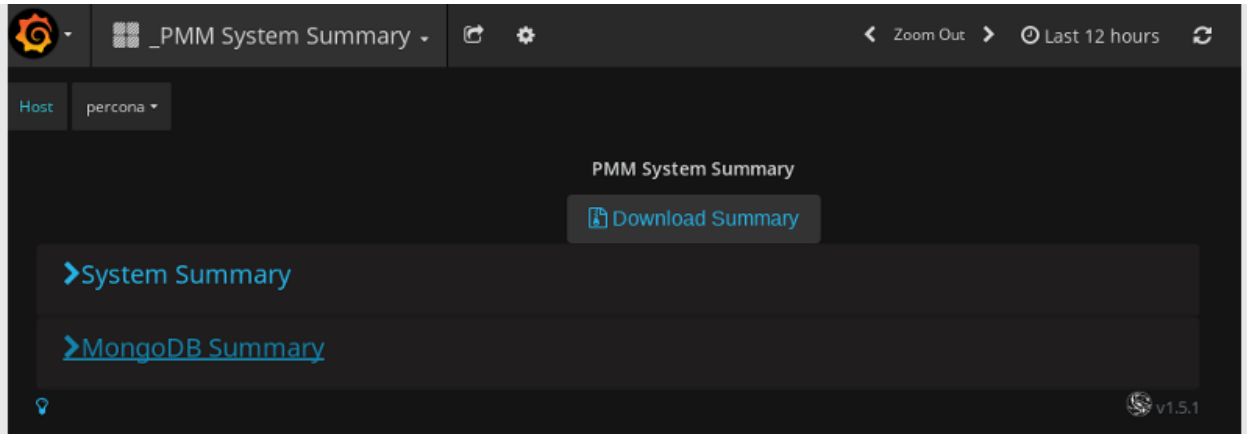


Fig. 30.1: Accessing information about the system and database

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

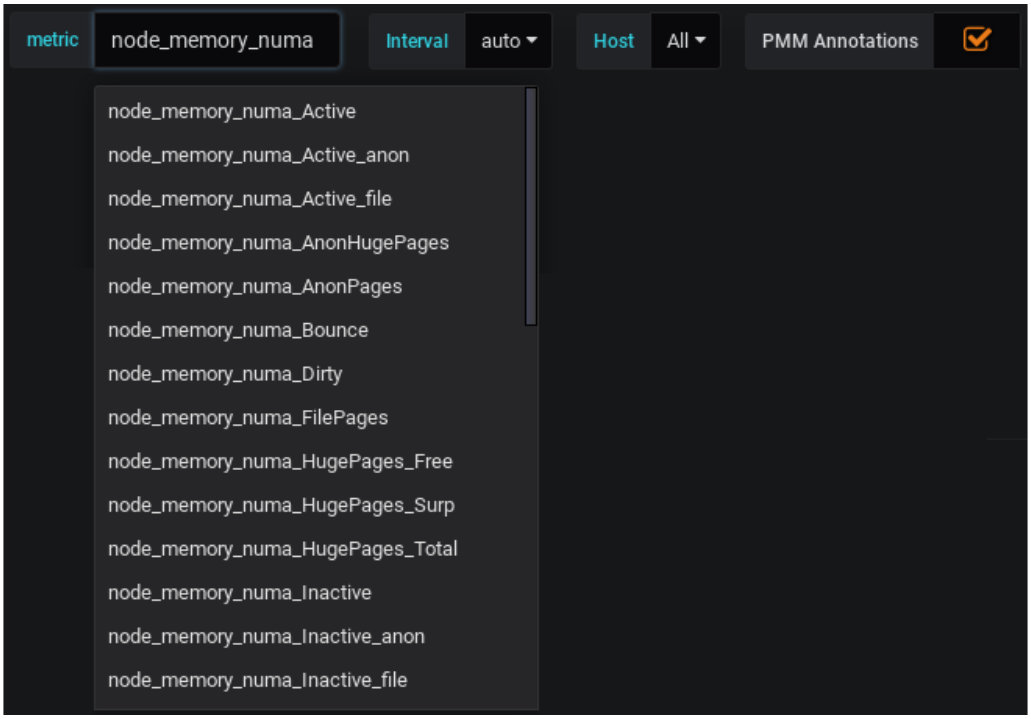
30.3 *Advanced Data Exploration* Dashboard

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

Added NUMA related metrics

New in version 1.13.0.

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



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

30.3.1 View actual metric values (Gauge)

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

30.3.2 View actual metric values (Counters)

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

30.3.3 View actual metric values (Counters)

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

See also:

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

30.4 Cross Server Graphs

30.4.1 Load Average

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

This metric is best used for trends. If you notice the load average rising, it may be due to innefficient 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

30.4.2 MySQL Queries

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

View all metrics of *Cross Server Graphs*

See also:

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

30.4.3 MySQL Traffic

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

View all metrics of *Cross Server Graphs*

30.5 Summary Dashboard

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

30.5.1 CPU Usage

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

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

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

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

Softirq The portion of time the CPU spent servicing software interrupts generated by the device drivers. A high value of *softirq* may indicates 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

30.5.2 Processes

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

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

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

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

View all metrics of *Summary Dashboard*

See also:

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

30.5.3 Network Traffic

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

Look for signs of saturation given the capacity of the network devices. If the outbound rate is coffffnstantly 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*

30.5.4 I/O Activity

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

View all metrics of *Summary Dashboard*

30.5.5 Disk Latency

The Disk Latency graph shows the average time to complete read an 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*

30.5.6 MySQL Queries

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

View all metrics of *Summary Dashboard*

30.5.7 InnoDB Row Operations

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

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

View all metrics of *Summary Dashboard*

30.5.8 Top MySQL Commands

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

View all metrics of *Summary Dashboard*

30.5.9 Top MySQL Handlers

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

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

View all metrics of *Summary Dashboard*

30.6 Trends Dashboard

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

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

See also:

MySQL Documentation:

Questions

Metrics

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

30.6.1 CPU Usage

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

View all metrics of *Trends Dashboard*

30.6.2 I/O Read Activity

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

View all metrics of *Trends Dashboard*

30.6.3 I/O Write Activity

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

View all metrics of *Trends Dashboard*

30.6.4 MySQL Questions

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

View all metrics of *Trends Dashboard*

30.6.5 InnoDB Rows Read

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

View all metrics of *Trends Dashboard*

30.6.6 InnoDB Rows Changed

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

View all metrics of *Trends Dashboard*

30.7 *Network Overview* Dashboard

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

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

30.7.1 Last Hour Statistic

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

View all metrics of *Network Overview Dashboard*

30.7.2 Network Traffic

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

View all metrics of *Network Overview Dashboard*

30.7.3 Network Traffic Details

This section offers the following metrics:

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

View all metrics of *Network Overview Dashboard*

30.7.4 Network Netstat TCP

This section offers the following metrics:

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

View all metrics of *Network Overview Dashboard*

30.7.5 Network Netstat UDP

In this section, you can find the following metrics:

- Netstat: UDP
- UDP Lite

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

InDatagrams Packets received

OutDatagrams Packets sent

InCsumErrors Datagrams with checksum errors

InErrors Datagrams that could not be delivered to an application

RcvbufErrors Datagrams for which not enough socket buffer memory to receive

SndbufErrors Datagrams for which not enough socket buffer memory to transmit

NoPorts Datagrams received on a port with no listener

View all metrics of *Network Overview Dashboard*

30.7.6 ICMP

This section has the following metrics:

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

ICMP Errors

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

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

InDestUnreachs Destination Unreachable messages received

OutDestUnreachs Destination Unreachable messages sent

InType3 Destination unreachable

OutType3 Destination unreachable

InCsumErrors Messages with ICMP checksum errors

InTimeExcds Time Exceeded messages received

Messages/Redirects

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

InRedirects Redirect messages received

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

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

Echos

InEchoReps Echo Reply messages received

InEchos Echo (request) messages received

OutEchoReps Echo Reply messages sent

OutEchos Echo (request) messages sent

Timestamps/Mask Requests

InAddrMaskReps Address Mask Reply messages received

InAddrMasks Address Mask Request messages received

OutAddrMaskReps Address Mask Reply messages sent

OutAddrMasks Address Mask Request messages sent

InTimestampReps Timestamp Reply messages received

InTimestamps Timestamp Request messages received

OutTimestampReps Timestamp Reply messages sent

OutTimestamps Timestamp Request messages sent

View all metrics of *Network Overview Dashboard*

30.8 Inventory Dashboard

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

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

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

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

CHAPTER

THIRTYONE

OS DASHBOARDS

31.1 CPU Utilization Details (Cores)

31.1.1 Overall CPU Utilization

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

- system
- user
- iowait
- softirq

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

System

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

User

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

Iowait

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

Softirq

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

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

31.1.2 Current CPU Core Utilization

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

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

31.1.3 All Cores - Total

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

- system
- user
- iowait
- steal
- other

Components

System

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

User

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

Iowait

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

Steal

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

Other

> This component is essentially the softirq time which is the portion of time the CPU spent servicing software interrupts generated by the device drivers. A high value of softirq may 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.

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

31.2 Disk space

31.2.1 Mountpoint Usage

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

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

View all metrics of *Disk space*

31.2.2 Mountpoint

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

Used Is the amount of space used

Free Is the amount if space not in use

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

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

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

View all metrics of *Disk space*

31.3 *System Overview* Dashboard

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

- CPU
- Memory
- Disk
- Network

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

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

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

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

31.4 *Compare System Parameters* Dashboard

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

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

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

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

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

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

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

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

31.5 *NUMA Overview* Dashboard

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

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

..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
```

31.5.1 Memory Usage

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

View all metrics of *NUMA Overview Dashboard*

31.5.2 Free Memory Percent

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

View all metrics of *NUMA Overview Dashboard*

31.5.3 NUMA Memory Usage Types

Dirty Memory waiting to be written back to disk

Bounce Memory used for block device bounce buffers

Mapped Files which have been mmaped, such as libraries

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

View all metrics of *NUMA Overview Dashboard*

31.5.4 NUMA Allocation Hits

Memory successfully allocated on this node as intended.

View all metrics of *NUMA Overview Dashboard*

31.5.5 NUMA Allocation Missed

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

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

View all metrics of *NUMA Overview Dashboard*

31.5.6 Anonymous Memory

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

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

View all metrics of *NUMA Overview Dashboard*

31.5.7 NUMA File (PageCache)

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

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

View all metrics of *NUMA Overview Dashboard*

31.5.8 Shared Memory

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

View all metrics of *NUMA Overview Dashboard*

31.5.9 HugePages Statistics

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

Free The number of hugepages not being allocated by a process

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

View all metrics of *NUMA Overview Dashboard*

31.5.10 Local Processes

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

View all metrics of *NUMA Overview Dashboard*

31.5.11 Remote Processes

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

View all metrics of *NUMA Overview Dashboard*

31.5.12 Slab Memory

Slab Allocation is a memory management mechanism intended for the efficient memory allocation of kernel objects.

SReclaimable The part of the Slab that might be reclaimed (such as caches).

SUnreclaim The part of the Slab that can't be reclaimed under memory pressure

View all metrics of *NUMA Overview Dashboard*

CHAPTER

THIRTYTWO

PROMETHEUS DASHBOARDS

32.1 *Prometheus* Dashboard

The *Prometheus* dashboard informs how Prometheus functions.

See also:

Overview of PMM using

All dashboards *Metrics Monitor Dashboards*

32.1.1 Prometheus overview

This section shows the most essential parameters of the system where Prometheus is running, such as CPU and memory usage, scrapes performed and the samples ingested in the head block.

32.1.2 Resources

This section provides details about the consumption of CPU and memory by the Prometheus process. This section contains the following metrics:

- Prometheus Process CPU Usage
- Prometheus Process Memory Usage

32.1.3 Storage (TSDB)

This section includes a collection of metrics related to the usage of storage. It includes the following metrics:

- Data blocks (Number of currently loaded data blocks)
- Total chunks in the head block
- Number of series in the head block
- Current retention period of the head block
- Activity with chunks in the head block
- Reload block data from disk

32.1.4 Scraping

This section contains metrics that help monitor the scraping process. This section contains the following metrics:

- Ingestion
- Prometheus Targets

- Scraped Target by Job
- Scrape Time by Job
- Scraped Target by Instance
- Scraped Time by Instance
- Scrapes by Target Frequency
- Scrape Frequency Versus Target
- Scraping Time Drift
- Prometheus Scrape Interval Variance
- Slowest Jobs
- Largest Samples Jobs

32.1.5 Queries

This section contains metrics that monitor Prometheus queries. This section contains the following metrics:

- Prometheus Queries
- Prometheus Query Execution
- Prometheus Query Execution Latency
- Prometheus Query Execution Load

32.1.6 Network

Metrics in this section help detect network problems.

- HTTP Requests by Handler
- HTTP Errors
- HTTP Avg Response time by Handler
- HTTP 99% Percentile Response time by Handler
- HTTP Response Average Size by Handler
- HTTP 99% Percentile Response Size

32.1.7 Time Series Information

This section shows the top 10 metrics by time series count and the top 10 hosts by time series count.

32.1.8 System Level Metrics

Metrics in this section give an overview of the essential system characteristics of PMM Server. This information is also available from the *System Overview* dashboard.

32.1.9 PMM Server Logs

This section contains a link to download the logs collected from your PMM Server and further analyze possible problems. The exported logs are requested when you submit a bug report.

32.2 Prometheus Exporter Status

The Prometheus Exporter Status dashboard reports the consumption of resources by the Prometheus exporters used by PMM. For each exporter, this dashboard reveals the following information:

- CPU usage
- Memory usage
- File descriptors used
- Exporter uptime

See also:

All PMM exporters pmm.list.exporter

Summary information about the usage of Prometheus exporters *Prometheus Exporters Overview*

32.3 Prometheus Exporters Overview

The Prometheus Exporters Overview dashboard provides the summary of how exporters are used across the selected hosts.

See also:

Percona Database Performance Blog

Understand Your Prometheus Exporters with Percona Monitoring and Management (PMM)

Prometheus documentation

Exporters and integrations

32.3.1 Prometheus Exporters Summary

This section provides a summary of how exporters are used across the selected hosts. It includes the average usage of CPU and memory as well as the number of hosts being monitored and the total number of running exporters.

Metrics in this section

- **Avg CPU Usage per Host** shows the average CPU usage in percent per host for all exporters.
- **Avg Memory Usage per Host** shows the Exporters average Memory usage per host.
- **Monitored Hosts** shows the number of monitored hosts that are running Exporters.
- **Exporters Running** shows the total number of Exporters running with this PMM Server instance.

Note: The CPU usage and memory usage do not include the additional CPU and memory usage required to produce metrics by the application or operating system.

32.3.2 Prometheus Exporters Resource Usage by Host

This section shows how resources, such as CPU and memory, are being used by the exporters for the selected hosts.

Metrics in this section

- **CPU Usage** plots the Exporters' CPU usage across each monitored host (by default, All hosts).
- **Memory Usage** plots the Exporters' Memory usage across each monitored host (by default, All hosts).

32.3.3 Prometheus Exporters Resource Usage by Type

This section shows how resources, such as CPU and memory, are being used by the exporters for host types: MySQL, MongoDB, ProxySQL, and the system.

Metrics in this section

- **CPU Cores Used** shows the Exporters' CPU Cores used for each type of Exporter.
- **Memory Usage** shows the Exporters' memory used for each type of Exporter.

32.3.4 List of Hosts

At the bottom, this dashboard shows details for each running host.

- **CPU Used** show the CPU usage as a percentage for all Exporters.
- **Mem Used** shows total Memory Used by Exporters.
- **Exporters Running** shows the number of Exporters running.
- **RAM** shows the total amount of RAM of the host.
- **Virtual CPUs** shows the total number of virtual CPUs on the host.

You can click the value of the *CPU Used*, *Memory Used*, or *Exporters Running* column to open the *Prometheus Exporter Status* for further analysis.

Related information: Prometheus Documentation

Exporters and integrations https://prometheus.io/docs/instrumenting/exporters

---

CHAPTER

THIRTYTHREE

MYSQL DASHBOARDS

CHAPTER

THIRTYFOUR

MONGODB DASHBOARDS

CHAPTER

THIRTYFIVE

POSTGRESQL DASHBOARDS

CHAPTER

THIRTYSIX

HA DASHBOARDS

36.1 PXC/Galera Cluster Overview Dashboard

- Flow Control Paused Time
- Flow Control Messages Sent
- Writeset Inbound Traffic
- Writeset Outbound Traffic
- Receive Queue
- Send Queue
- Transactions Received
- Transactions Replicated
- Average Incoming Transaction Size
- Average Replicated Transaction Size
- FC Trigger Low Limit
- FC Trigger High Limit
- Sequence Numbers of Transactions
- Average Galera Replication Latency
- Maximum Galera Replication Latency

Part X

Contacting and Contributing

Percona Monitoring and Management is an open source product. We provide ways for anyone to contact developers and experts directly, submit bug reports and feature requests, and contribute to source code directly.

Contacting Developers

Use the community forum to ask questions about using PMM. Developers and experts will try to help with problems that you experience.

Reporting Bugs

Use the PMM project in JIRA to report bugs and request features. Please register and search for similar issues before submitting a bug or feature request.

Contributing Source Code

Use the GitHub repository to explore source code and suggest contributions. You can fork and clone any Percona repositories, but to have your source code patches accepted please sign the Contributor License Agreement (CLA).

Part XI

Terminology Reference

CHAPTER

THIRTYSEVEN

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`

CHAPTER

THIRTYEIGHT

DATA SOURCE NAME

A database server attribute found on the QAN page. It informs how PMM connects to the selected database.

CHAPTER

THIRTYNINE

DSN

See *Data Source Name*

CHAPTER

FORTY

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.

CHAPTER

FORTYONE

%GTT

See *Grand Total Time*

CHAPTER

FORTYTWO

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 an service is added, you can set up the *Metrics Monitor* to display its graphs.

CHAPTER

FORTYTHREE

METRICS

A series of data which are visualized in PMM.

CHAPTER

FORTYFOUR

METRICS MONITOR (MM)

Component of *PMM Server* that provides a historical view of metrics critical to a MySQL server instance.

CHAPTER

FORTYFIVE

MONITORING SERVICE

A special service which collects information from the database instance where *PMM Client* is installed.

To add a monitoring service, use the **`pmm-admin add`** command.

See also:

Passing parameters to a monitoring service pmm.pmm-admin.monitoring-service.pass-parameter

CHAPTER

FORTYSIX

PMM

Percona Monitoring and Management

CHAPTER

FORTYSEVEN

PMM-ADMIN

A program which changes the configuration of the *PMM Client*. See detailed documentation in the pmm-admin section.

CHAPTER

FORTYEIGHT

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/

CHAPTER

FORTYNINE

PMM CLIENT

Collects MySQL server metrics, general system metrics, and query analytics data for a complete performance overview.

The collected data is sent to *PMM Server*.

For more information, see *Client/Server Architecture - an Overview*.

CHAPTER

FIFTY

PMM DOCKER IMAGE

A docker image which enables installing the PMM Server by using **`docker`**.

See also:

Installing PMM Server using Docker run-server-docker

CHAPTER

FIFTYONE

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*).

CHAPTER

FIFTYTWO

PMM SERVER

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

PMM Server combines the backend API and storage for collected data with a frontend for viewing time-based graphs and performing thorough analysis of your MySQL and MongoDB hosts through a web interface.

Run PMM Server on a host that you will use to access this data.

See also:

PMM Architecture

Client/Server Architecture - an Overview

CHAPTER

FIFTYTHREE

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
```

CHAPTER

FIFTYFOUR

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 pmm.amazon-rds.iam-user.creating

CHAPTER

FIFTYFIVE

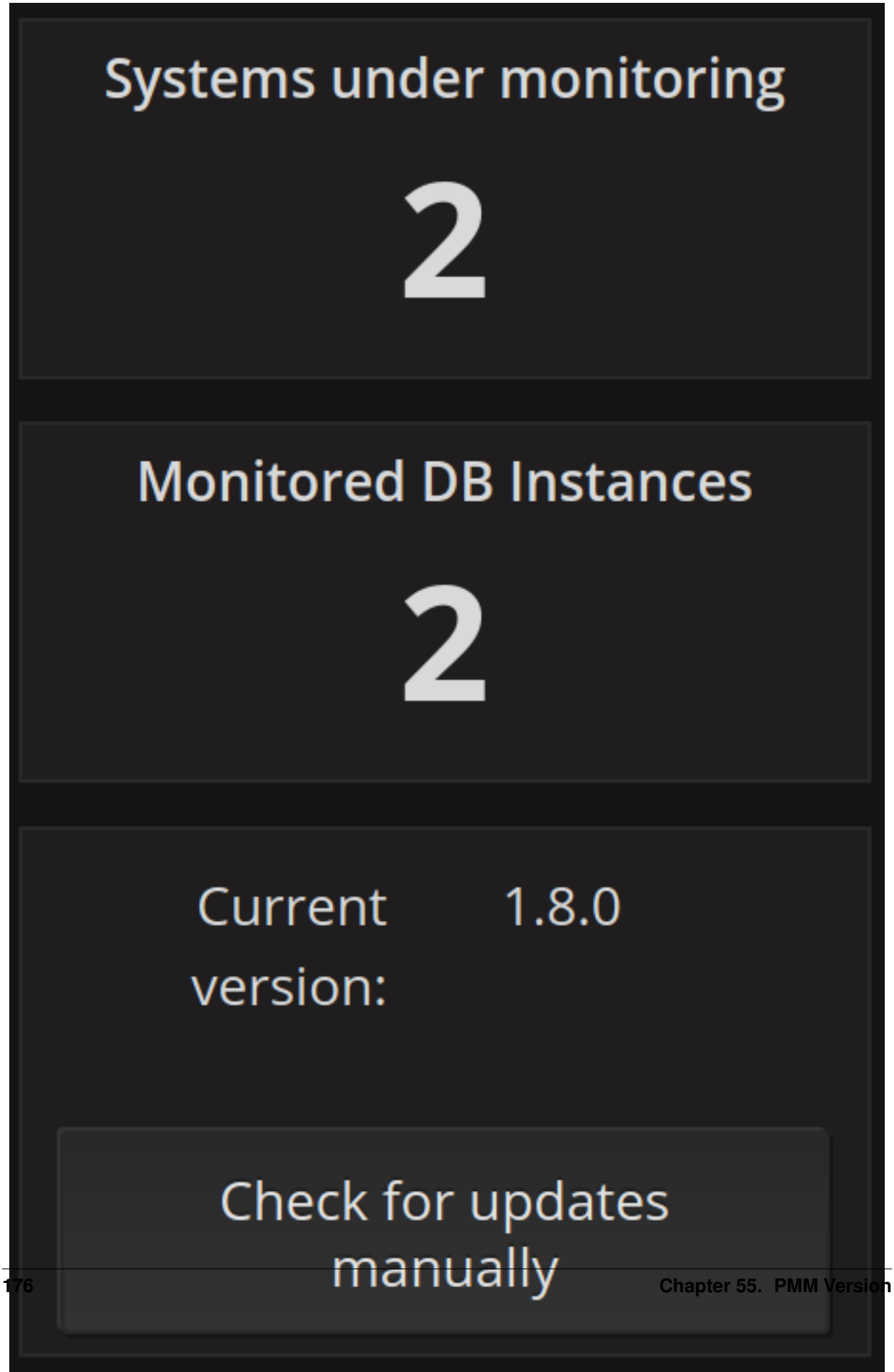
PMM VERSION

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


Systems under monitoring
2
Monitored DB Instances
2
Current version:
1.8.0
Check for updates manually

CHAPTER

FIFTYSIX

QAN

See *Query Analytics (QAN)*

CHAPTER

FIFTYSEVEN

QUERY ABSTRACT

Query pattern with placeholders. This term appears in *QAN* as an attribute of queries.

CHAPTER

FIFTYEIGHT

QUERY ANALYTICS (QAN)

Component of *PMM Server* that enables you to analyze MySQL query performance over periods of time.

CHAPTER

FIFTYNINE

QUERY FINGERPRINT

See *Query Abstract*

CHAPTER

SIXTY

QUERY ID

A *query fingerprint* which groups similar queries.

CHAPTER

SIXTYONE

QUERY LOAD

The percentage of time that the MySQL server spent executing a specific query.

CHAPTER

SIXTYTWO

QUERY METRICS SUMMARY TABLE

An element of *Query Analytics (QAN)* which displays the available metrics for the selected query.

CHAPTER

SIXTYTHREE

QUERY METRICS TABLE

A tool within QAN which lists metrics applicable to the query selected in the *query summary table*.

CHAPTER

SIXTYFOUR

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*.

CHAPTER

SIXTYFIVE

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

CHAPTER

SIXTYSIX

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 collects metrics.

CHAPTER

SIXTYSEVEN

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:2
```

CHAPTER

SIXTYEIGHT

VERSION

A database server attribute found on the QAN page. it informs the full version of the monitored database server, as well as the product name, revision and release number.

Part XII

Frequently Asked Questions

CHAPTER

SIXTYNINE

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.

CHAPTER

SEVENTY

WHAT ARE THE MINIMUM SYSTEM REQUIREMENTS FOR PMM?

PMM Server

Any system which can run Docker version 1.12.6 or later.

It needs roughly 1 GB of storage for each monitored database node with data retention set to one week.

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

Minimum memory is 2 GB for one monitored database node, but it is not linear when you add more nodes. For example, data from 20 nodes should be easily handled with 16 GB.

PMM Client

Any modern 64-bit Linux distribution. It is tested on the latest versions of Debian, Ubuntu, CentOS, and Red Hat Enterprise Linux.

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

CHAPTER

SEVENTYONE

HOW TO CONTROL DATA RETENTION FOR PMM?

By default, both Prometheus and QAN store time-series data for 30 days.

Depending on available disk space and your requirements, you may need to adjust data retention time.

You can control data retention by passing the `DATA_RETENTION` environment variable when *creating and running the PMM Server container*. To set environment variable, use the `-e` option. The value should be the number of hours, and requires h suffix. For example, the default value of 30 days for `METRICS_RETENTION>` is `720h`, but you can decrease the retention period for Prometheus to 8 days as follows:

```
-e DATA_RETENTION=192h
```

See also:

Metrics and queries retention `DATA_RETENTION`

CHAPTER

SEVENTYTWO

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.

CHAPTER

SEVENTYTHREE

WHAT PRIVILEGES ARE REQUIRED TO MONITOR A MYSQL INSTANCE?

See *Creating a MySQL User Account to Be Used with PMM*.

CHAPTER

SEVENTYFOUR

CAN I MONITOR MULTIPLE SERVICE INSTANCES?

Yes, you can add multiple instances of MySQL or some other service to be monitored from one PMM Client. In this case, you will need to provide a distinct port and socket for each instance, and specify a unique name for each instance (by default, it uses the name of the PMM Client host).

For example, if you are adding complete MySQL monitoring for two local MySQL servers, the commands could look similar to the following:

```
$ sudo pmm-admin add mysql --username root --password root instance-01 127.0.0.1:3001
$ sudo pmm-admin add mysql --username root --password root instance-02 127.0.0.1:3002
```

For more information, run

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

CHAPTER

SEVENTYFIVE

CAN I RENAME INSTANCES?

You can remove any monitoring instance as described in *Removing monitoring services with pmm-admin remove* and then add it back with a different name.

When you remove a monitoring service, previously collected data remains available in Grafana. However, the metrics are tied to the instance name. So if you add the same instance back with a different name, it will be considered a new instance with a new set of metrics. So if you are re-adding an instance and want to keep its previous data, add it with the same name.

CHAPTER

SEVENTYSIX

HOW TO TROUBLESHOOT COMMUNICATION ISSUES BETWEEN PMM CLIENT AND PMM SERVER?

Broken network connectivity may be caused by rather wide set of reasons. Particularly, when *using Docker*, the container is constrained by the host-level routing and firewall rules. For example, your hosting provider might have default *iptables* rules on their hosts that block communication between PMM Server and PMM Client, resulting in *DOWN* targets in Prometheus. If this happens, check firewall and routing settings on the Docker host.

Also PMM is able to generate a set of diagnostics data which can be examined and/or shared with Percona Support to solve an issue faster. You can get collected logs from PMM Client using the `pmm-admin summary` command. Obtaining logs from PMM Server can be done *by specifying the ''https://<address-of-your-pmm-server>/logs.zip'* URL, or by clicking the `server logs` link on the Prometheus dashboard: