M2P-DB-MONITORIING - 3.109.229.121 / 172.17.0.1

Infra account

AWS-infra key

NOTE: Need Elastic IP

NOTE: Install Docker on the desired FS

NOTE: Open Ports between server and client

Install Percona Monitoring and Management

----------------------------------------

(1) Requirements: Docker

yum install docker

service docker status

service docker start

service docker status

(2) Create data volume:

sudo docker create -v /datadb/docker --name pmm-data percona/pmm-server:2 /bin/true

[root@ip-172-150-0-56 docker]# sudo docker create -v /datadb/docker --name pmm-data percona/pmm-server:2 /bin/true

Unable to find image 'percona/pmm-server:2' locally

2: Pulling from percona/pmm-server

2d473b07cdd5: Pull complete

cb234533d813: Pull complete

Digest: sha256:ff0bb20cba0dbfcc8929dbbba0558bb01acc933ec593717727707dce083441b4

Status: Downloaded newer image for percona/pmm-server:2

25eecceb5f12de159613d6ddaf5aab873901b4a528eca8fc8b432d94cade34d0

docker ps -a

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

[root@ip-172-150-0-56 docker]# sudo docker run -d -p 80:80 -p 443:443 --volumes-from pmm-data --name pmm-server --restart always percona/pmm-server:2

031c3f65266d718c151d5a8211b049c8ecbea8a687e019d1ba6c0fd6fd6f18ed

[root@ip-172-150-0-56 docker]#

WEBLOGIN

http://3.109.229.121

Provide credentials

==================================================================================

PMM CLIENT – MYSQL

PMM

Master server – client-server

MASTER - DB Monitoring

3.109.229.121/172.150.0.56

CLIENT - CHN

13.71.61.189 /175.1.11.8

Check if exists

----------------

rpm -qa | grep -i pmm

(Optional) REMOVE OLD

--------------------

yum remove pmm2-client-2.21.0-6.el8.x86\_64

step 1: INSTALL

-------------------

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

step 2: Enable Percona Server 8.0 repository

----------------------------

percona-release setup ps80 -y

step 3: Install pmm2-client

----------------------------

sudo yum install pmm2-client -y

step 4:

port 443 to be opened for DB server which is to be monitored

telnet 3.109.229.121 443

telnet 172.150.0.56 443

step 5

port 3306 to be opened in DB server from pmm server

telnet 175.1.11.8 3306

telnet 13.71.61.189 3306

Step 6

--------------------

pmm-admin config --server-insecure-tls --force --server-url= https://dbamon:Pmmdb@123@ 172.150.0.56

Step 7: Log in to MySQL

-----------------------

mysql -u m2p331 -p -v

select user,host from mysql.user where user='pmm';

CREATE USER 'pmm'@'%' IDENTIFIED BY 'Open@123';

GRANT SELECT, CREATE TEMPORARY TABLES, REPLICATION CLIENT ON \*.\* TO 'pmm'@'%' ;

GRANT SELECT, PROCESS, SUPER, REPLICATION CLIENT, RELOAD ON \*.\* TO 'pmm'@'%';

show grants for pmm;

flush privileges;

show global variables like 'innodb\_monitor%';

set global innodb\_monitor\_enable=All;

+--------------------------+-------+

| Variable\_name | Value |

+--------------------------+-------+

| innodb\_monitor\_disable | |

| innodb\_monitor\_enable | all |

| innodb\_monitor\_reset | |

| innodb\_monitor\_reset\_all | |

+--------------------------+-------+

step 7 :

--------------------

pmm-admin add mysql --query-source=perfschema --username=pmm --password=Open@123 --service-name=LQ\_PG --host=172.192.10.225 --port=3201

MySQL Service added.

Table statistics collection disabled (the limit is 1000, the actual table count is 34172).

--------------

show global variables like 'performance\_schema%';

PMM CLIENT – MONGO

PMM

Master server – client-server

MASTER - DB Monitoring

3.109.229.121 / 172.150.0.56

CLIENT - CHN

13.233.138.108/ 172.160.0.229

##########################################

Check if exists

----------------

rpm -qa | grep -i pmm

(Optional) REMOVE OLD

--------------------

yum remove pmm2-client-2.21.0-6.el8.x86\_64

step 1:

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

step 2:

percona-release setup ps80

step 3:

sudo yum install pmm2-client -y

step 4:

port 443 to be opened for db server which is to be monitored

telnet 172.150.0.56 443

telnet 3.109.229.121 443

step 5

port 27017 to be opened in DB server from pmm server

netstat -lamp | grep ':7777'

telnet 13.233.138.108 27017

Step 6

pmm-admin config --server-insecure-tls --force --server-url= https://dbamon:Pmmdb@123@172.150.0.56

Step 6.1 - Set Profiling

mongo

use admin

db.getSiblingDB("admin").createRole({

role: "explainRole",

privileges: [{

resource: {

db: "",

collection: ""

},

actions: [

"listIndexes",

"listCollections",

"dbStats",

"dbHash",

"collStats",

"find"

]

}],

roles:[]

})

db.getSiblingDB("admin").createUser({

user: "pmm",

pwd: "Open@123",

roles: [

{ role: "explainRole", db: "admin" },

{ role: "clusterMonitor", db: "admin" },

{ role: "read", db: "local" }

]

})

----------------------------------------------------------------------------------------------

Step 6.2 - Config Level

operationProfiling:

mode: all

slowOpThresholdMs: 200

----------------------------------------------------------------------------------------------

(Optional)

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

mongod --dbpath=/dbdata01/mongo --profile 2 --slowms 200 --rateLimit 100

mongod --dbpath=/dbdata01/mongo --profile 2 --slowms 200

----------------------------------------------------------------------------------------------

Step 7 : Session Level

use admin

db.setProfilingLevel(2)

use admin

db.setProfilingLevel(2)

{ "was" : 2, "slowms" : 200, "sampleRate" : 1, "ok" : 1 }

----------------------------------------------------------------------------------------------

step 8

pmm-admin add mongodb --username=pmm --password=Open@123 --query-source=profiler --service-name=GROWW\_PROD\_1 --host= 172.120.5.35 --port=22000

PMM CLIENT – POSTGRESQL

--Step 1 :

Create a Postgres user that can be used for monitoring. You could choose any username; pmm in the following command is just an example.

$ psql -c "CREATE USER pmm WITH ENCRYPTED PASSWORD 'Open@123'"

--Step 2 :

Grant pg\_monitor role to the pmm.

$ psql -c "GRANT pg\_monitor to pmm"

--Step 3 :

If you are not using localhost, but using the IP address of the PostgreSQL server while enabling monitoring in the next steps, you should ensure to add appropriate entries to enable connections from the IP and the pmm in the pg\_hba.conf file.

$ echo "host all pmm 192.168.80.20/32 md5" >> $PGDATA/pg\_hba.conf

$ psql -c "select pg\_reload\_conf()"

1

2

$ echo "host all pmm 192.168.80.20/32 md5" >> $PGDATA/pg\_hba.conf

$ psql -c "select pg\_reload\_conf()"

In the above step, replace 192.168.80.20 with the appropriate PostgreSQL Server’s IP address.

--Step 4 :

Validate whether you can connect as pmm to the Postgres database from the Postgres server itself.

# psql -h 192.168.80.20 -p 5432 -U pmm -d postgres

Password for user pmm:

# psql -h 192.168.80.20 -p 5432 -U pmm -d postgres

Password for user pmm:

postgres=>

Enabling PostgreSQL Monitoring with and without QAN (Query Analytics)

Using PMM, we can monitor several metrics in PostgreSQL such as database connections, locks, checkpoint stats, transactions, temp usage, etc. However, you could additionally enable Query Analytics to look at the query performance and understand the queries that need some tuning. Let us see how we can simply enable PostgreSQL monitoring with and without QAN.

==================

With QAN

With PMM2, there is an additional step needed to enable QAN. You should create a database with the same name as the monitoring user ( pmm here). And then, you should create the extension: pg\_stat\_statements in that database. This behaviour is going to change on the next release so that you can avoid creating the database.

--Step 1 :

Create the database with the same name as the monitoring user. Create the extension: pg\_stat\_statements in the database.

psql -c "CREATE DATABASE pmm"

psql -h yappgresdb.cfzdwmybtrx8.ap-south-1.rds.amazonaws.com -p 5432 -Upostgres -c "CREATE DATABASE pmm"

psql -c -d pmm "CREATE EXTENSION pg\_stat\_statements"

psql -h yappgresdb.cfzdwmybtrx8.ap-south-1.rds.amazonaws.com -p 5432 -Upostgres -c -d pmm "CREATE EXTENSION pg\_stat\_statements"

psql -h yappgresdb.cfzdwmybtrx8.ap-south-1.rds.amazonaws.com -p 5432 -Upostgres pmm -c "CREATE EXTENSION pg\_stat\_statements"

--Step 2 :

If shared\_preload\_libraries has not been set to pg\_stat\_statements, we need to set it and restart PostgreSQL.

Shell

$ psql -c "ALTER SYSTEM SET shared\_preload\_libraries TO 'pg\_stat\_statements'"

$ pg\_ctl -D $PGDATA restart -mf

waiting for the server to shut down.... done

server stopped

...

...

done

server started

$ psql -c "ALTER SYSTEM SET shared\_preload\_libraries TO 'pg\_stat\_statements'"

$ pg\_ctl -D $PGDATA restart -mf

waiting for the server to shut down.... done

server stopped

...

...

done

server started

--Step 3 :

In the previous steps, we used the flag: --query-source=none to disable QAN. To enable QAN, you could just remove this flag and use pm-admin add PostgreSQL without the flag.

Shell

# pmm-admin add postgresql --username=pmm --password=secret postgres 192.168.80.20:5432

PostgreSQL Service added.

Service ID : /service\_id/24efa8b2-02c2-4a39-8543-d5fd54314f73

Service name: Postgres

# pmm-admin add postgresql --username=pmm --password=secret postgres 192.168.80.20:5432

PostgreSQL Service added.

Service ID : /service\_id/24efa8b2-02c2-4a39-8543-d5fd54314f73

Service name: Postgres

--Step 4 :

Once the above step is completed, you could validate the same again using the pmm-admin list. But this time, you should see an additional service: can-PostgreSQL-pgstatements-agent.

Shell

# pmm-admin list

Service type Service name Address and port Service ID

PostgreSQL postgres 192.168.80.20:5432 /service\_id/24efa8b2-02c2-4a39-8543-d5fd54314f73

Agent type Status Agent ID Service ID

pmm-agent connected /agent\_id/13fd2e0a-a01a-4ac2-909a-cae533eba72e

node\_exporter running /agent\_id/f6ba099c-b7ba-43dd-a3b3-f9d65394976d

postgres\_exporter running /agent\_id/7039f7c4-1431-4518-9cbd-880c679513fb /service\_id/24efa8b2-02c2-4a39-8543-d5fd54314f73

qan-postgresql-pgstatements-agent running /agent\_id/7f0c2a30-6710-4191-9373-fec179726422 /service\_id/24efa8b2-02c2-4a39-8543-d5fd54314f73

# pmm-admin list

Service type Service name Address and port Service ID

PostgreSQL postgres 192.168.80.20:5432 /service\_id/24efa8b2-02c2-4a39-8543-d5fd54314f73

Agent type Status Agent ID Service ID

pmm-agent connected /agent\_id/13fd2e0a-a01a-4ac2-909a-cae533eba72e

node\_exporter running /agent\_id/f6ba099c-b7ba-43dd-a3b3-f9d65394976d

postgres\_exporter running /agent\_id/7039f7c4-1431-4518-9cbd-880c679513fb /service\_id/24efa8b2-02c2-4a39-8543-d5fd54314f73

qan-postgresql-pgstatements-agent running /agent\_id/7f0c2a30-6710-4191-9373-fec179726422 /service\_id/24efa8b2-02c2-4a39-8543-d5fd54314f73

After this step, you can now see the Queries and their statistics captured on the Query Analytics Dashboard.

Meanwhile, have you tried Percona Distribution for PostgreSQL? It is a collection of finely-tested and implemented open source tools and extensions along with PostgreSQL 11, maintained by Percona. PMM works for both Community PostgreSQL and also the Percona Distribution for PostgreSQL. Please subscribe to our blog posts to learn more interesting features in PostgreSQL.