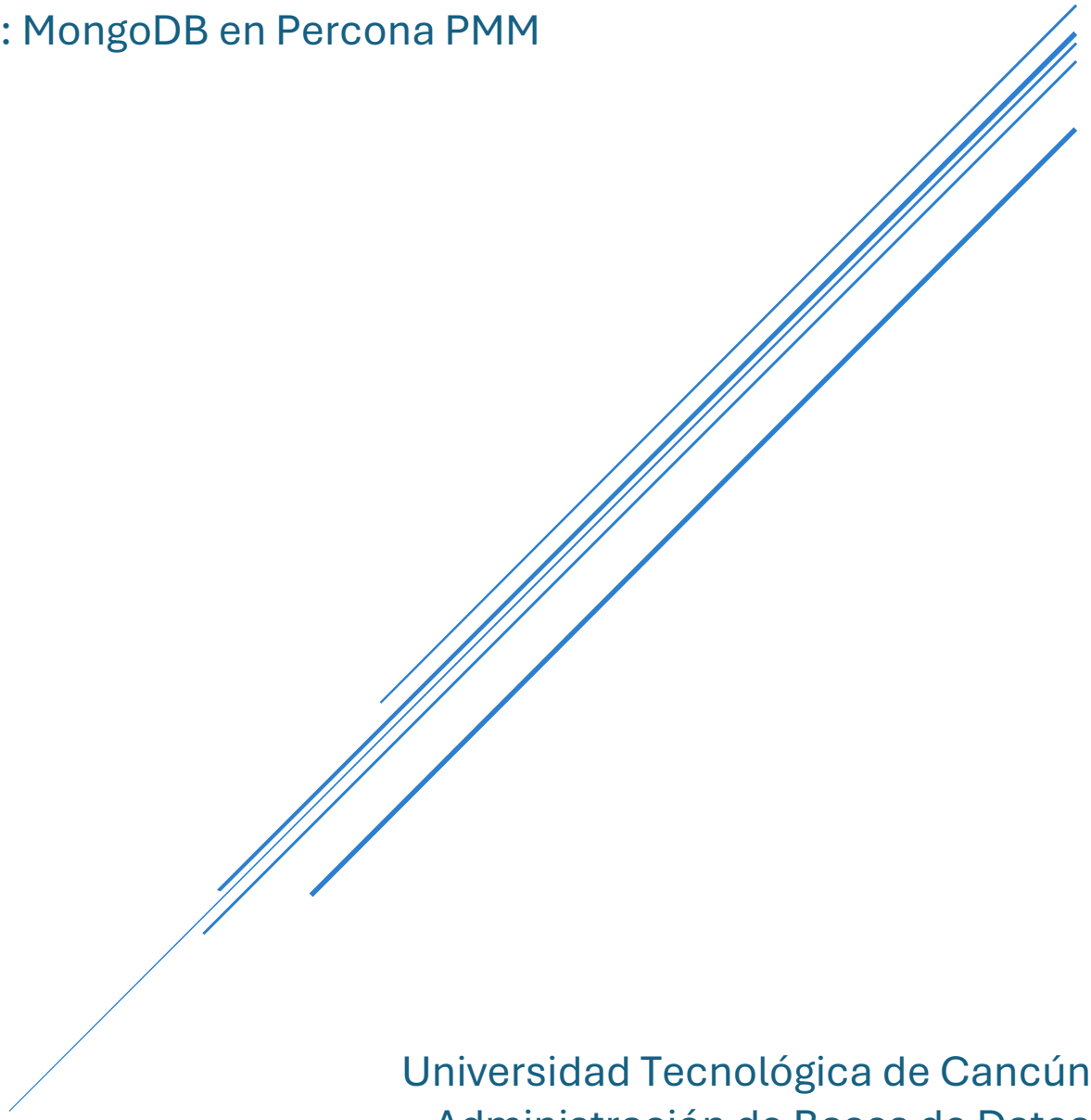
CONFIGURACIÓN FINAL: CONEXIÓN DE PERCONA SERVER FOR MONGODB CON PMM

AFU02: MongoDB en Percona PMM



Universidad Tecnológica de Cancún
Administración de Bases de Datos
Diaz Rosales Gerardo Antonio
IDYGS82

Contenido

Configuración Final: Docker + Percona PMM + Bases de Datos 2
Instalación de PMM Server 2
Creación del volumen 2
Ejecución del contenedor 2
Configuración de Percona Server para MySQL 5
Configuración de PMM Client 6
Adición de Servicio 7
Configuración alterna de PMM Client y conexión con Percona Server a través de la interfaz gráfica. 8
Percona Server y MongoDB 10
MongoDB: Configuración y Seguridad 12
Instalación de MongoDB Database Tools 13
Subida de respaldo a AWS S3 14

Configuración Final: Docker + Percona PMM + Bases de Datos.

Instalación de PMM Server

Dentro de nuestra aplicación de Docker Desktop, abrimos la consola y descargamos la imagen de Percona Server.

Comando de instalación:

```
docker pull percona/pmm-server:3
```

Usamos la versión 3, una versión más estable en lugar de usar la más resiente para evitar cambios no probados, esto con el propósito de garantizar la compatibilidad con PMM y evitar fallos por actualizaciones automáticas.

Creación del volumen

Los volúmenes de Docker persisten datos entre reinicios y actualización del contenedor

Comando:

```
docker volume create pmm-data
```

El **pmm-data** almacena las configuraciones de Grafana (paneles), métricas históricas (ClickHouse), usuarios y claves del dashboard, esto nos permite evitarnos el riesgo de perdida de todo el historial de monitoreo al reiniciar el contenedor

Ejecución del contenedor

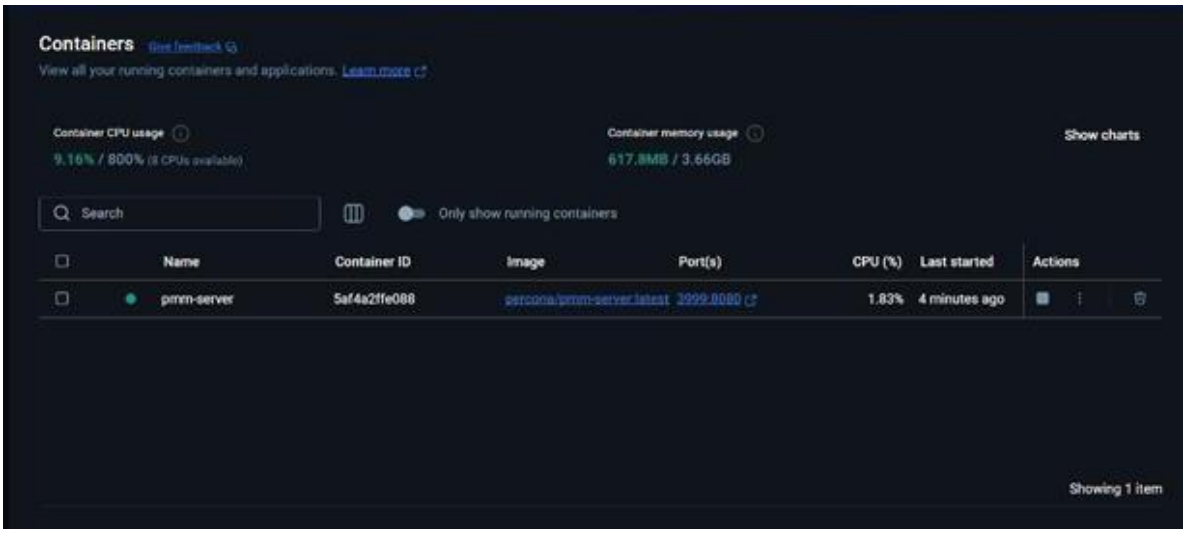
Para ejecutar Percona PMM Server, ejecutamos el siguiente comando en la consola o tefrminal:

```
docker run --detach --restart always --publish 3999:8443 -v pmm-data:/srv --name pmm-server percona/pmm-server:3
```

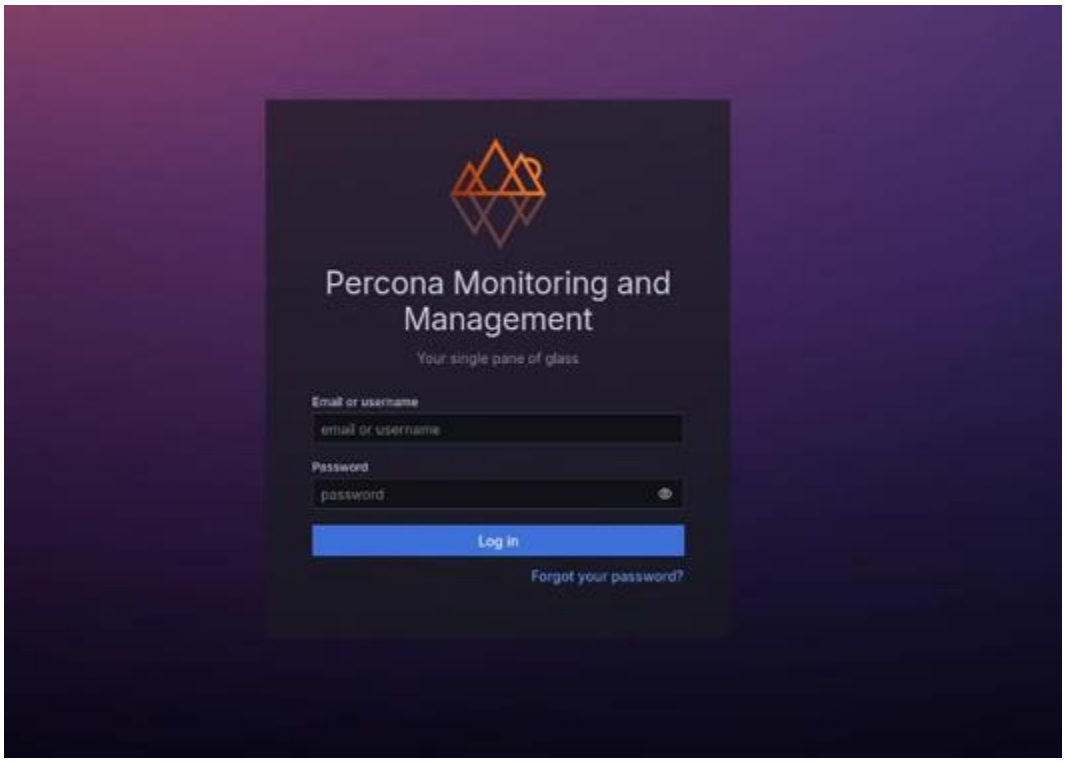
Donde:

Parámetro	Propósito	Descripción
--detach	Ejecuta en segundo plano	Permite seguir usando la terminal
--restart always	Reinicio automático	Previene caídas prolongadas tras reinicios del host
--publish 3999:8443	Mapeo de puertos	8443 (HTTPS interno) → 3999 (acceso externo)
-v pmm-data:/srv	Montaje de volumen	Vincula el volumen creado al directorio /srv del contenedor
--name pmm-server	Nombre del contenedor	Facilita gestión con docker start/stop pmm-server

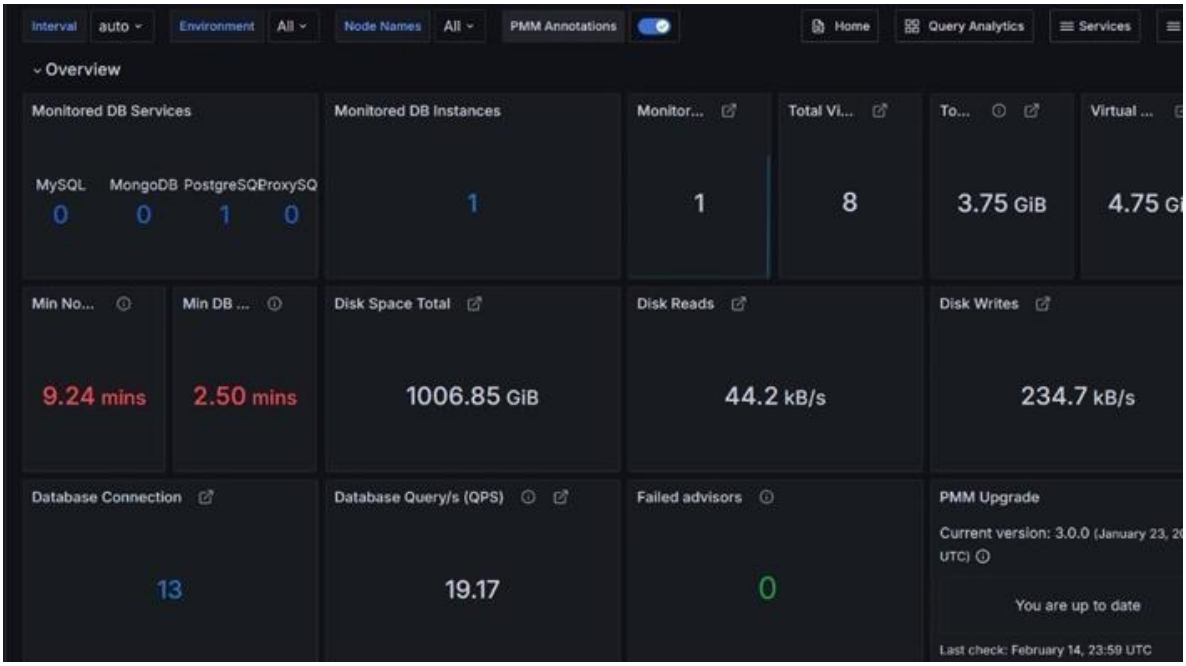
Posteriormente verificamos el estado del contenedor y vemos que se haya creado de forma correcta.



Luego de eso podremos acceder a la interfaz Web de Percona a través de la URL proporcionada por el puerto del contenedor.



Ingresamos nuestras credenciales de usuario y contraseña, siendo estas por defecto ambas **admin**. Después de eso, nos pedirá que cambiemos la contraseña ya ccedemos a la interfaz del Dashboard de Percona PMM Server.



Configuración de Percona Server para MySQL

Para acceder al cliente de MySQL a través de Docker desktop, tendremos que descargar la imagen y crear el contenedor.

Para la instalación de Percona Server con MySQL, ejecutamos el siguiente comando:

```
docker run -d --name ps-mysql -e MYSQL_ROOT_PASSWORD=root -p 3998:3306 percona/percona-server:latest
```

- **-e MYSQL_ROOT_PASSWORD=root**: Hay que tener en consideración que dentro de entornos de producción, deberíamos de usar una contraseña más compleja, pero en pruebas “root” simplificamos el acceso.

- **-p 3998:3306**: Mapea el puerto estándar de MySQL (3306) al 3998 en el host para evitar conflictos con otras instancias.

- **--restart unless-stopped**: Reinicia el contenedor de forma automática, excepto si se detiene manualmente. Es menos intrusivo que usar **always**.

Posteriormente podemos verificar la ejecución o estado del contenedor a través de los siguientes comandos:

```
$ docker ps
```

```
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS                    PORTS                                NAMES
e5b9802ffae1   percona/percona-server:latest   "/docker-entrypoint.…"   4 minutes ago    Up 4 minutes              33060/tcp, 0.0.0.0:3998->3306/tcp    ps-mysql
9d618cb5ab2e   percona/pmm-server:latest       "/opt/entrypoint.sh"     13 hours ago     Up 2 hours (healthy)      8443/tcp, 0.0.0.0:3999->8080/tcp     pmm-server
```

```
docker exec -it ps-mysql mysql -u root -p
```

docker exec: Ejecuta comandos dentro del contenedor de ejecución.

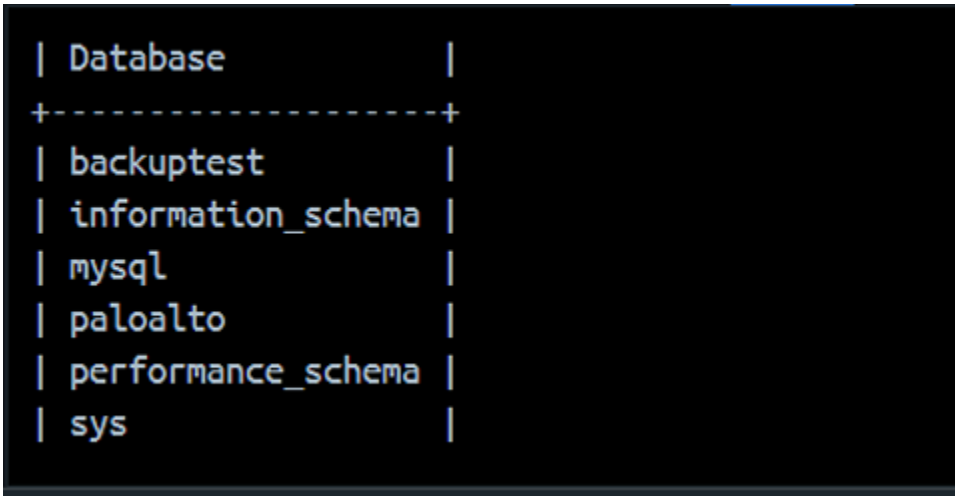
-it: Habilita la terminal interactiva (para ingresar la contraseña).

Una alternativa segura sería usar **MYSQL_PWD=root** para evitar exponer la contraseña en el historial

Para ponerlo aprueba, seleccionamos abror una nueva terminal a través del contenedor MySQL. Eejecutamos el cliente de MySQL a través de los comandos:

mysql -u root -p

Posteriormente, introducimos nuestras credenciales para acceder al cliente, donde ya podremos ejecutar MySQL de forma regular.



Configuración de PMM Client

Registro del Nodo:

```
docker exec pmm-client pmm-admin config --server-insecure-tls --server-url=http://admin:admin@172.17.0.2:8443 172.17.0.4 container ps-dk-container --force
```

- **--server-insecure-tls**: Permite conexiones HTTPS sin validar certificados SSL. Recomendado solo para entornos de prueba

- **172.17.0.4**: IP del **host Docker** (no del contenedor). Necesaria para que PMM Server identifique el nodo.

- **container**: Indica que el servicio monitoreado está en un contenedor (optimiza recolección de métricas).

- **--force**: Sobrescribe configuraciones previas (útil para pruebas).

Adición de Servicio

```
docker exec pmm-client pmm-admin add mysql --username=root --password=root --host=172.17.0.3 --port=3306
```

La IP **172.17.0.3** pertenece al contenedor de MySQL en la red de Docker **bridge** predeterminada.

Para obtenerla ejecutamos:

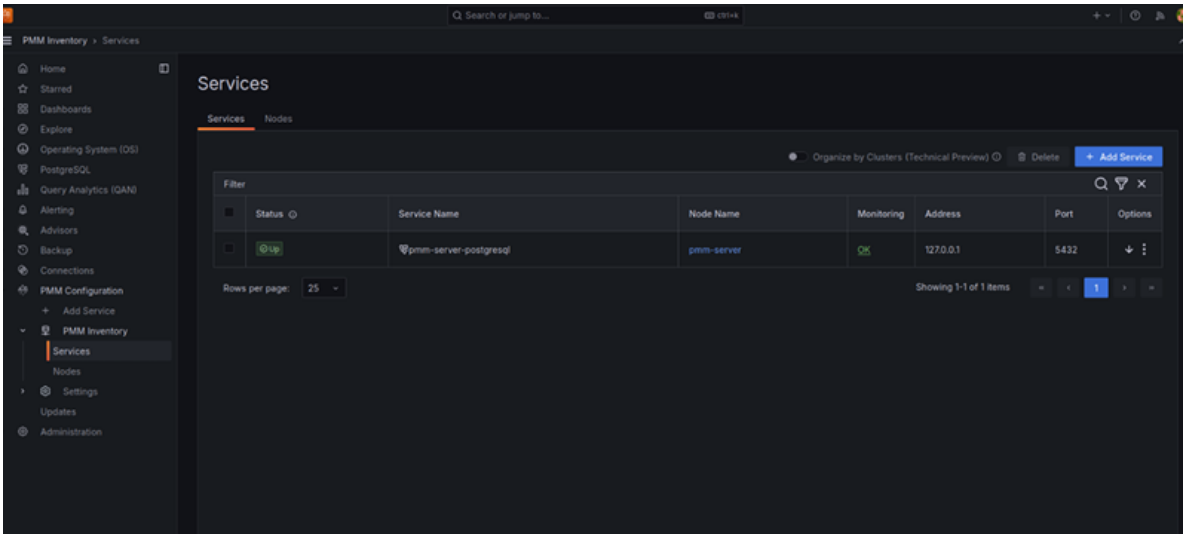
```
docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' ps-mysql
```

Configuración alterna de PMM Client y conexión con Percona Server a través de la interfaz gráfica.

A continuación, se detallará una configuración alterna de Percona PMM Server para agregar y configurar el monitoreo de un servidor MySQL.

Esta configuración es por si ya se tiene acceso al PMM Server y si se quieren realizar las configuraciones manualmente, y si se prefiere gestionar la configuración a través de una interfaz gráfica.

Ambas opciones son difeerentes en su implementación, pero hacen lo mismo, es decir, conectar MySQL a PMM para monitorear su rendimiento.

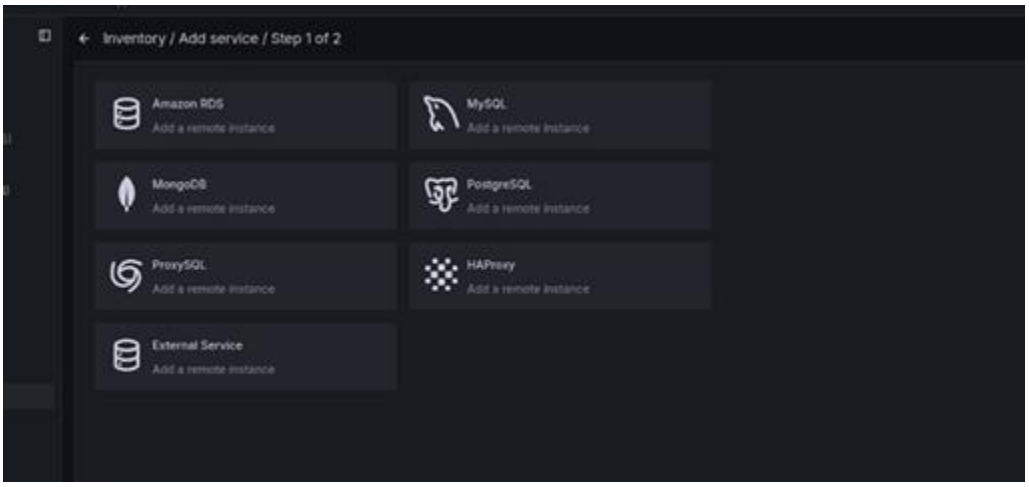


Dentro de la interfaz web de Percona Server, seguimos los siguientes pasos:

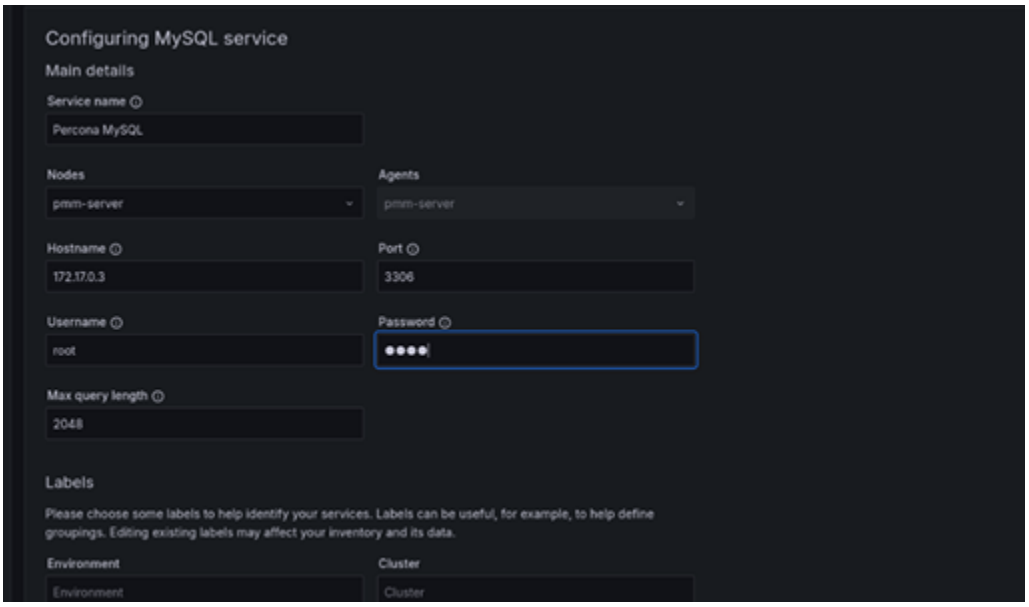
- Acceder a la sección **PMM Configuratión/PMM Inventory/Services**.

Agregamos un nuevo servicio para iniciar el proceso de agregar el servicio de la base de datos.

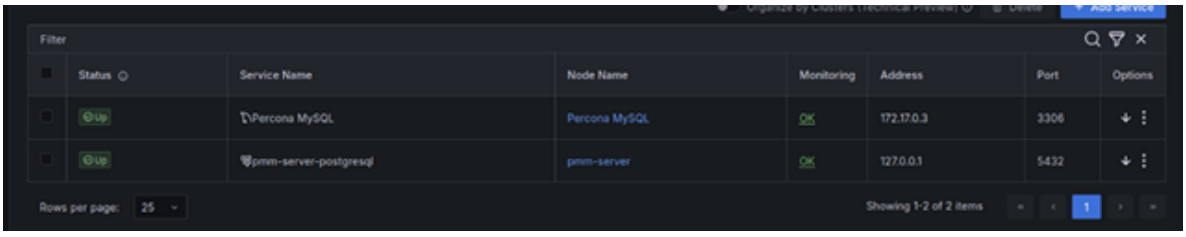
Se desplegará una lista de pçiones con disttiontos monitores de bases de datos disponibles. Seleccionamos MySQL para configurar el servicio.



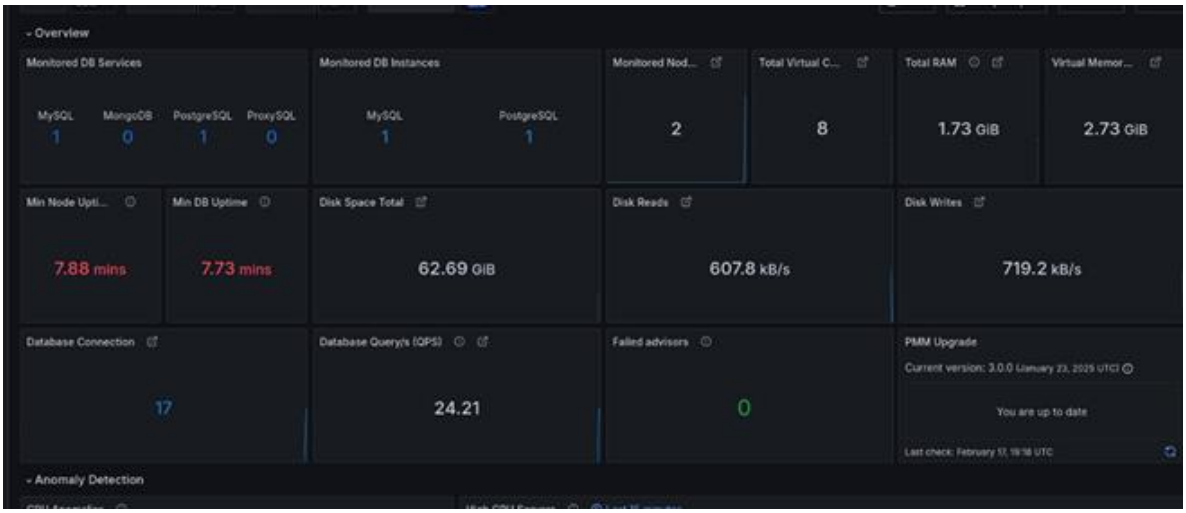
Al seleccionar MySQL, se llenarán los campos correspondientes:



Guardamos y verificamos la conexión. Una vez agregada, aparecerá en la lista:



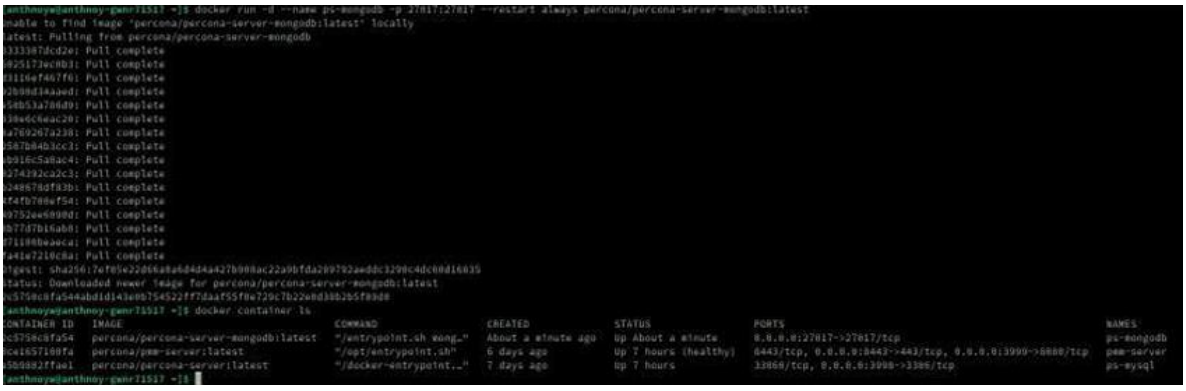
Finalmente, accedemos al dashboard y verificamos el servicio de MySQL.



Percona Server y MongoDB

Primero creamos el contenedor de Percona Server con MongoDB. Para ello ejecutamos el siguiente comando para iniciar la descarga:

$ docker run -d --name ps-mongodb -p 27017:27017 --restart always percona/percona-server- mongodb:latest



Verificamos el estado del contenedor mediante el uso del comando:

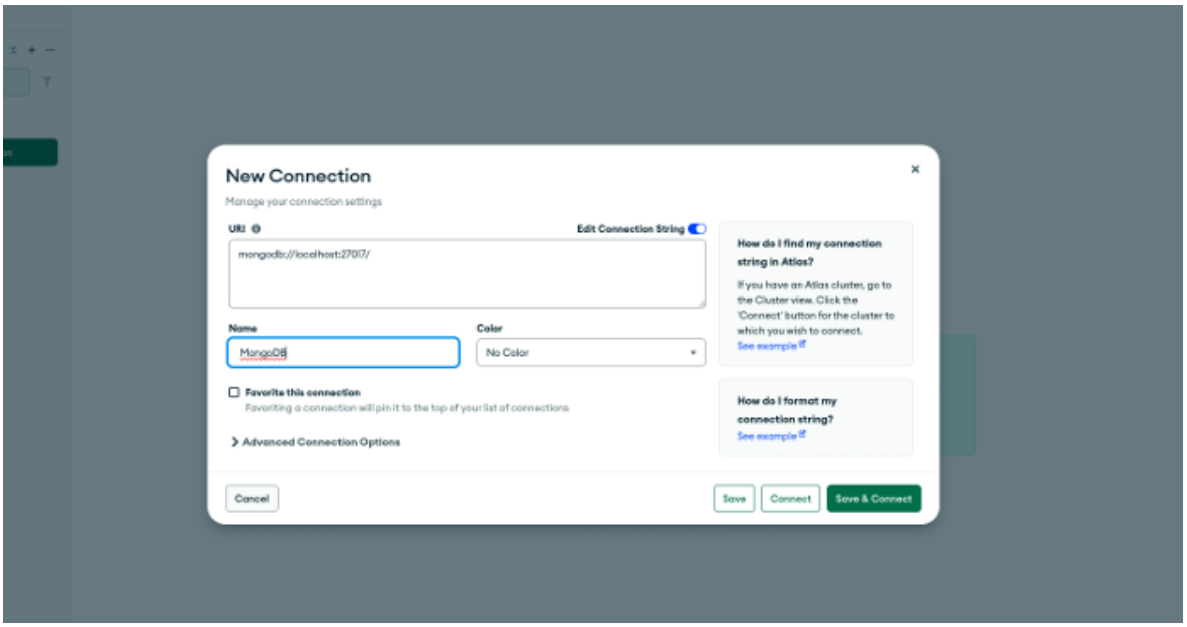
$ docker container ls

Dentro del contenedor ejecutamos el siguiente comando para acceder a los servicios de MongoDB:

$ mongosh

Realizamos la conexión de MongoDB desde MongoDB Compass, para ello verificamos que el puerto 27017 del contenedor estuviera expuesto correctamente y que el servidor MongoDB fuera accesible desde el sistema host.

Dentro de MongoDB Compass hacemos clic en agregar una nueva conexión con el puerto: **$ mongodb://localhost:27017**, nombramos nuestra conexión y guardamos.



Al conectar, se podrá acceder al servidor de MongoDB alojado en el contenedor de Docker y visualizar nuestras bases de datos disponibles, de esta manera confirmando que el servicio funciona correctamente.

MongoDB: Configuración y Seguridad

Creamos un usuario administrador:

```
db.createUser({
    user: "admin",
    pwd: "admin",
    roles: [{ role: "userAdminAnyDatabase", db: "admin" }]
})
```

Explicación de los Roles:

- **userAdminAnyDatabase**: Permite crear/editar usuarios en **cualquier** base de datos (equivalente a root en MySQL).

- **db: "admin":** La base de datos donde se almacenan los datos de autenticación (debe ser admin para roles globales).

Posteriormente reiniciamos MongoDB:

```
db.adminCommand({ shutdown: 1 })
```

Esto es necesario por lo siguiente:

- MongoDB **no aplica cambios de seguridad** en el cliente.
- El reinicio fuerza la carga de la nueva configuración de usuarios.

Instalación de MongoDB Database Tools

Para usar **mongodump** y **mongostoren** necesitamos instalar MongoDB Database Tools.

Descomprimimos nuestro archivo de descarga, instalamos y agregamos la ruta de nuestro archivo **bin** dentro del **PATH** del sistema.

Si hicimos la instalación de forma correcta podremos verificar la versión de mongodump en nuestra terminal.

```
mongodump version: 100.11.0
git version: b8a566a7f38fdcd9ba62256bee1880c999f2d4d7
Go version: go1.22.10
   os: windows
   arch: amd64
   compiler: gc
```

Para la creación de un respaldo podemos ejecutar el siguiente comando:

```
mongodump --host localhost --port 27017 --username admin --password CONTRASEÑA --out C:\backup-mongo
```

Donde:

- **--host y --port**: Dirección y puerto de MongoDB.
- **--username y --password**: Credenciales de autenticación.
- **--authenticationDatabase**: Base de datos usada para autenticarse.
- **--out**: Directorio donde se guardará el backup.

Al ejecutar el comando se nos creara una carpeta con el contenido de respaldo de nuestra base de datos dentro de la ruta especificada dentro del comando.

Archivo .bat: Creamos un archivo llamado backup_mongo.bat para realizar un respaldo de nuestra base de datos.

Ejemplo de archivo .bat:

```
@echo off
echo Creando respaldo de MongoDB...

set MONGO_PATH="C:\mongodb-database-tools-windows-x86_64-100.11.0\bin"
set BACKUP_PATH="C:\backup-mongo"
set DB_NAME="testDB"

"%MONGO_PATH%\mongodump.exe" --host localhost --port 27017 --username admin --password
Divergente2022 --authenticationDatabase admin --db %DB_NAME% --out %BACKUP_PATH%

echo Respaldo completado en %BACKUP_PATH%
pause
```

El respaldo se ejecutará cada vez que ejecutemos el archivo de respaldo.

```
mongodump version: 100.11.0
git version: b8a566a7f38fdcd9ba62256bee1880c999f2d4d7
Go version: go1.22.10
   os: windows
   arch: amd64
   compiler: gc
```

Subida de respaldo a AWS S3

Comando para realizar respaldos de MongoDB:

mongodump --host localhost --port 27017 --username admin --password CONTRASEÑA --authenticationDatabase admin --out C:\backup-mongo

Posteriormente comprimimos el archivo, en mi caso lo hice dentro de la ruta del escritorio para tener mayor facilidad a la hora de subirlo a la nube.

Posteriormente y para subir nuestro archivo de respaldo será importante descargar e instalar **AWS CLI** e ingresar nuestras credenciales de acceso.

Para configurar nuestras credenciales utilizamos el comando:

aws configure

Agregamos nuestras credenciales posteriormente podremos ejecutar el comando que nos permitirá subir nuestro respaldo, en mi caso ejecute el siguiente comando:

```
aws s3 cp "C:\Users\vampy\OneDrive\Desktop\Diaz_Gerardo.zip" s3://ut-admin-db-dygs82/
```

Donde deberemos establecer la ruta del archivo que queremos subir, y el bucket de subida.

Por último, podremos verificar la subida del archivo mediante el comando:

```
aws s3 ls s3://ut-admin-db-dygs82/
```