



**UNIVERSIDAD  
MODELO**

# Administración de base de datos

Autor: Jorge Said Serrano Soto

## Indice

|  |          |
|--|----------|
| <b>1. Objetivo</b>   | <b>2</b> |
| <b>2. Alcance</b>  | <b>2</b> |
| <b>3. Proceso</b>  | <b>2</b> |
| 3.1 Contenido del archivo docker-compose.yml   | 2        |
| 3.2 Explicación de cada linea del archivo  | 2        |
| 3.2.1 version:   | 2        |
| 3.2.2 services:  | 3        |
| 3.2.3 db:  | 3        |
| 3.2.4 image:   | 3        |
| 3.2.5 ports:   | 3        |
| 3.2.6 enviroments:   | 3        |
| 3.2.7 volumes:   | 4        |
| 3.3 Proceso para la configuración del gestor   | 4        |
| 3.4 configuración para auditoría y registro de logs  | 4        |
| <b>4 Diferencias entre un gestor de base de datos tradicional y uno con docker y contenedores.</b> | <b>5</b> |
| <b>5. Conclusiones</b>   | <b>5</b> |

## 1. Objetivo

El Objetivo del siguiente documento es explicar el uso y razón de contenedores para la creación de gestores de bases de datos, al igual que la demostración de su uso teniendo en cuenta la comparación de estos con los gestores de bases de datos tradicionales, de la misma manera se añade el objetivo de ilustrar al lector sobre los componentes de un contenedor, tanto el documento con terminación .yaml así como el contenido del mismo y los detalles tanto internos como externos.

## 2. Alcance

El alcance de este conocimiento es demasiado amplio, debido a que en la era moderna del software el desarrollo en la nube es la tecnología más eficiente para proyectos de alto alcance, tanto para la administración de una serie de proyectos en el apartado de la base de datos, así como sus apartados en el área operacional y de seguridad, en base al uso de un mejor modelo de trabajo, esto con el fin de realizar proyectos que logren tener una mejor estructura tecnológica que si se realizaran con el modelo de trabajo tradicional que incluía el uso de un servicio en un servidor el cual hasta llega a ser mucho más costoso, siendo que las grandes corporaciones como netflix o amazon plantean su uso dentro de su arquitectura, es bastante notable el gran alcance que tiene este tipo de conocimiento a fin de la realización de proyectos profesionales.

## 3. Proceso

### ***3.1 Contenido del archivo docker-compose.yaml***

**Anexo 1.0** Ejemplo de documento docker-compose.yaml

### ***3.2 Explicación de cada línea del archivo***

#### *3.2.1 version:*

En esta línea de código es en donde se especifica la versión del docker-compose engine el cual va a ser utilizado para la ejecución del mismo,

siendo que por ahora la versión más recomendada en la ejecución de un contenedor es la versión 3.

### *3.2.2 services:*

En esta línea de código es en donde se van a introducir los servicios ejecutados en el docker-compose, siendo un ejemplo utilizado para una base de datos el servicio db el cual es en donde se van a insertar las imágenes a utilizar en post de preparar un contenedor con un servicio de base de datos.

### *3.2.3 db:*

En el apartado anterior se mencionó que este es uno de los servicios que se van a utilizar para plantear el contenedor que se va a utilizar, a fin de poder introducir los diferentes parámetros como lo son la imagen en específico, los puertos y las variables de entorno.

### *3.2.4 image:*

Este apartado es primordial, pues aquí es en donde se establece la imagen de uso, la cual será en el caso del ejemplo específico del **Anexo 1.0** será mariadb, siendo que esto dará al docker compose su funcionalidad principal, al menos en el apartado de db.

### *3.2.5 ports:*

Es bien conocido que en las bases de datos relacionales se utilizan puertos específicos, por default el puerto específico es el 3306, pero también se tiene la opción de cambiarlo, la cual es altamente recomendada.

### *3.2.6 enviroments:*

Este apartado del código se utiliza para la declaración de las variables de entorno, siendo que en este ejemplo se utilizan variables como el usuario root, la contraseña del admin\_db y los usuarios normales en la base de datos.

### 3.2.7 volumes:

Aquí es en donde se especifica el guardado de los logs los cuales se utilizan para el registro de inicios de sesión fallidos, esto con el fin de poder verificar posibles ataques de fuerza bruta en la base de datos, de la misma manera se guarda el apartado de la administración de conexiones las cuales deben de ser configuradas para una conexión pública en caso de usarse para producción.

### 3.3 Proceso para la configuración del gestor

El primer paso es la búsqueda de la imagen que utilizaremos para el docker-compose la cual es en este ejemplo maria db **Anexo 1.1**, luego entonces se empezara a redactar el archivo docker-compose.yml, el cual tendrá los parámetros que se observan en el **Anexo 1.0**.

Se debe de configurar el archivo my.cnf para su conexión pública a dispositivos externos al equipo que ejecuta el gestor, este archivo se encuentra en el interior del contenedor el cual cuenta con una consola de el sistema operativo linux Ubuntu, siendo que se va a ejecutar el comando mostrado en el **Anexo 1.3**, a fin de entrar en la máquina del contenedor en donde se va a ubicar el archivo 50-server.cnf el cual va a tener la configuración que tenemos que modificar la cual se va a visualizar en el apartado de bin-adress cambiando el 127.0.0.1 a 0.0.0.0 con el fin de permitir el uso de conexiones externas al contenedor **Anexo 1.4** y **Anexo 1.5** para finalizar vamos a salir de la máquina de linux Ubuntu y ejecutar el penúltimo paso.

Para ese entonces se va ejecutar una ventana de consola y se debe de ubicar en la carpeta en donde se encuentra el archivo docker-compose.yml, para esto y verificar que funciona se ejecutará el comando docker-compose up -d el cual nos va lanzar un mensaje en verde de donde si es que el proceso se llevó a cabo de manera correcta **Anexo 1.2**.

Por último se utiliza una aplicación para conectarse al gestor de base de datos **Anexo 1.6**.

### 3.4 configuración para auditoría y registro de logs

Se toma en cuenta la existencia de un resguardo de los logs en la máquina del contenedor aquí es en donde inicialmente se tiene un archivo con los volúmenes

parecidos al **Anexo 1.7** pero se debe de configurar en base al uso de los logs que se encuentran en la máquina a fin de verificarlos según se vayan haciendo las conexiones erróneas esto se realiza yendo a la carpeta ya creada como se puede ver en el **Anexo 1.8** y viendo el archivo de conexiones erróneas en el archivo error.log **Anexo 1.9**.

#### **4 Diferencias entre un gestor de base de datos tradicional y uno con docker y contenedores.**

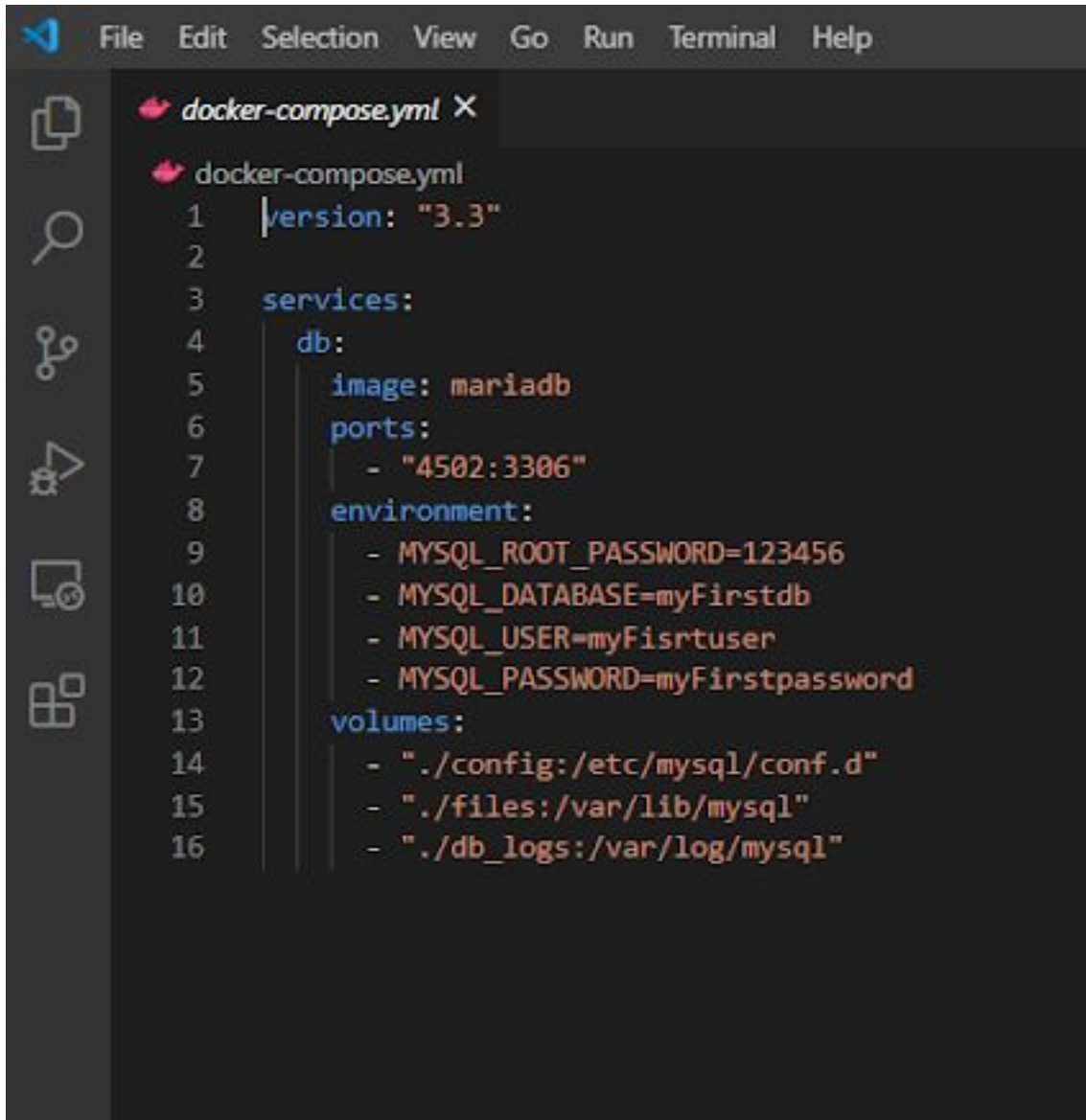
Se toma en cuenta que el gestor de base de datos tradicional como primera diferencia se destaca la creación de una sola conexión de base de datos, lo cual limita de manera muy extensa el testeo inicial de una cantidad de bases de datos en un proyecto o proyectos que así lo requieran, otra diferencia notable es la instalación de un gestor de base de datos, el cual con el uso de docker y contenedores es nula debido a que solo se tiene que levantar un servicio dockerizado y automatizado listo para que uno solo se conecte, por último se deben de mencionar la facilidad de los respaldos en docker ya que estos una vez levantado el servicio en producción en un contenedor que se ejecuta en linux los respaldos pueden ser automatizados, al igual que la consulta del archivo error.log **Anexo 1.9**, el cual puede ser consultado para la prevención de conexiones no autorizadas.

#### **5. Conclusiones**

Debo mencionar que personalmente el uso de estas herramientas planteadas en la nube son realmente maravillosas, siendo que la tecnología es muy cambiante, docker con el tiempo que lleva se ha adaptado a los procesos de desarrollo profesionales con una muy alta estima, siendo que también logre como ingeniero visualizar al menos a una escala pequeña el potencial que tiene esta herramienta para el desarrollo de futuros proyectos a fin de poder mejorar mis buenas prácticas en el desarrollo en la nube, debido a que docker tiene una estructura infalible para su uso en el apartado de las buenas prácticas.

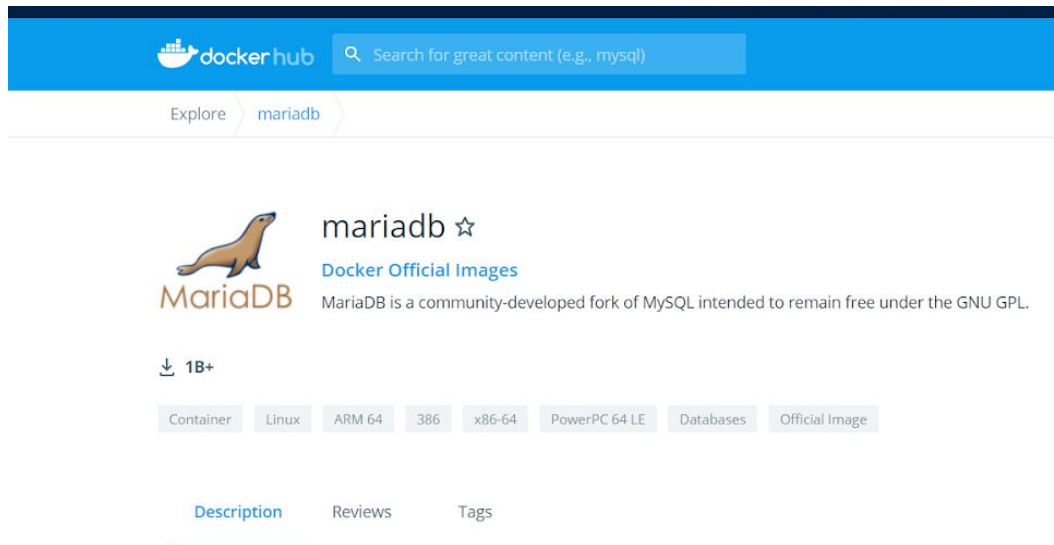
Por último debo de mencionar el hecho de que este tipo de aprendizajes serán de alta utilidad cuando los utilice de manera profesional pues me permitirán hacer que los proyectos que desarrolle tengan estándares de alta calidad.

## Anexos


A screenshot of a code editor window with a dark theme. The title bar at the top shows a menu with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. Below the menu, the editor displays a file named 'docker-compose.yml' with a red heart icon and a close button. The code is a YAML configuration for a Docker service. It starts with 'version: "3.3"' on line 1. Line 3 has 'services:'. Line 4 has 'db:'. Line 5 has 'image: mariadb'. Line 6 has 'ports:'. Line 7 has a list item '- "4502:3306"'. Line 8 has 'environment:'. Lines 9-12 have a list of environment variables: '- MYSQL\_ROOT\_PASSWORD=123456', '- MYSQL\_DATABASE=myFirstdb', '- MYSQL\_USER=myFisrtuser', and '- MYSQL\_PASSWORD=myFirstpassword'. Line 13 has 'volumes:'. Lines 14-16 have a list of volume mappings: '- ./config:/etc/mysql/conf.d', '- ./files:/var/lib/mysql', and '- ./db\_logs:/var/log/mysql'. The left sidebar of the editor contains icons for file explorer, search, source control, and other development tools.

```
1  version: "3.3"
2
3  services:
4    db:
5      image: mariadb
6      ports:
7        - "4502:3306"
8      environment:
9        - MYSQL_ROOT_PASSWORD=123456
10       - MYSQL_DATABASE=myFirstdb
11       - MYSQL_USER=myFisrtuser
12       - MYSQL_PASSWORD=myFirstpassword
13     volumes:
14       - ./config:/etc/mysql/conf.d"
15       - ./files:/var/lib/mysql"
16       - ./db_logs:/var/log/mysql"
```

Anexo 1.0 Ejemplo de documento docker-compose.yml




## Anexo 1.1 ejemplo de la imagen oficial mariadb

 Símbolo del sistema

```
C:\Users\biosh\Desktop\Universidad\quinto parcial\Administración de base de datos\Examen\docker-exec>docker-compose up -d
Creating network "docker-exec_default" with the default driver
Creating docker-exec_db_1 ... done

C:\Users\biosh\Desktop\Universidad\quinto parcial\Administración de base de datos\Examen\docker-exec>
```

## Anexo 1.2 ejecución de comando docker-compose up -d

 Símbolo del sistema

```
C:\Users\biosh\Desktop\Universidad\quinto parcial\Administración de base de datos\Examen\docker-exec>docker exec -it docker-exec_db_1 /bin/bash
```

## Anexo 1.3 visualización de los adentros del contenedor en base a encontrar el archivo my.cnf



```

root@f3bfeaff3c88: /etc/mysql/mariadb.conf.d
root@f3bfeaff3c88: /# cd etc/mysql
root@f3bfeaff3c88: /etc/mysql# ls
conf.d  debian-start  debian.cnf  mariadb.cnf  mariadb.conf.d  my.cnf
root@f3bfeaff3c88: /etc/mysql# cd mariadb.conf.d
root@f3bfeaff3c88: /etc/mysql/mariadb.conf.d# ls
50-client.cnf  50-mysql-clients.cnf  50-mysqld_safe.cnf  50-server.cnf  60-galera.cnf
root@f3bfeaff3c88: /etc/mysql/mariadb.conf.d# nano 50-server.cnf

```

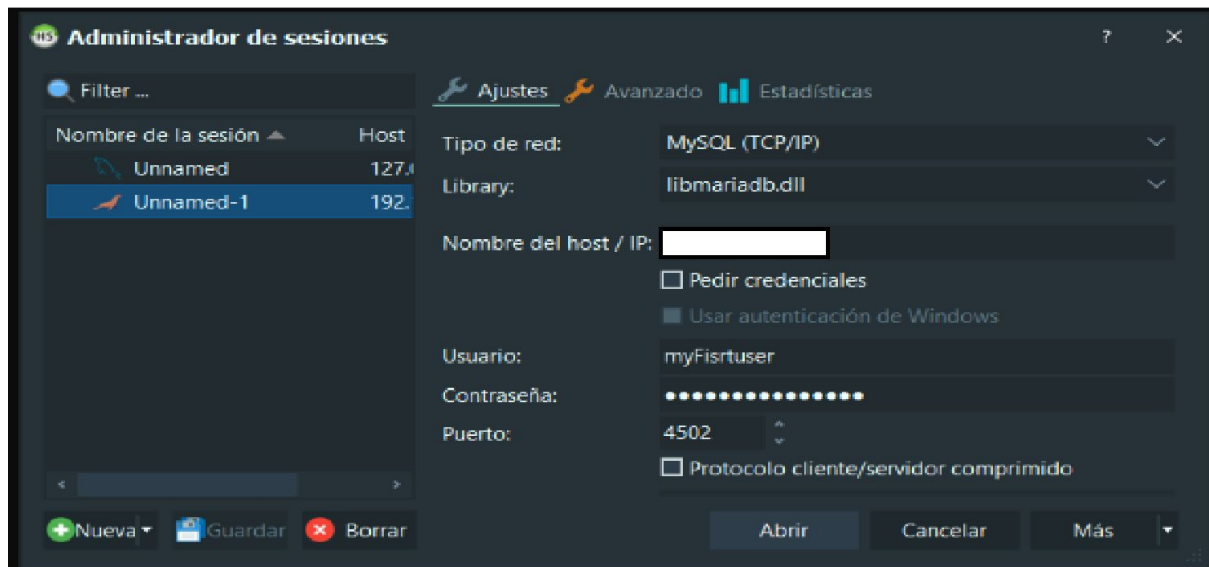
#### Anexo 1.4 archivo 50-server.cnf con contenido de conexiones importantes

```

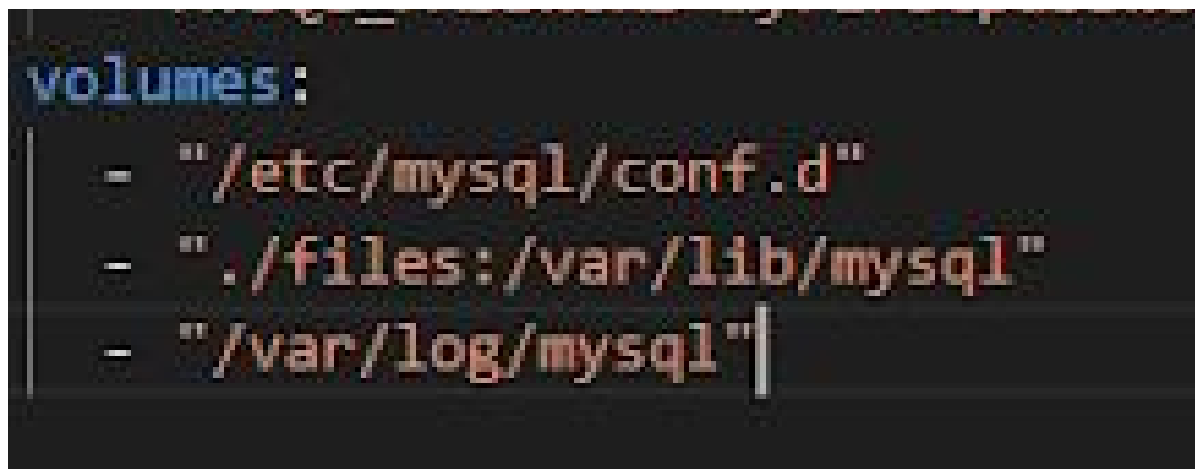
root@f3bfeaff3c88: /etc/mysql/mariadb.conf.d
GNU nano 4.8
#
# These groups are read by MariaDB server.
# Use it for options that only the server (but not clients) should see
#
# this is read by the standalone daemon and embedded servers
[server]
#
# this is only for the mysqld standalone daemon
[mysqld]
#
# * Basic Settings
#
#user                    = mysql
pid-file                 = /run/mysqld/mysqld.pid
basedir                  = /usr
datadir                  = /var/lib/mysql
tmpdir                   = /tmp
lc-messages-dir          = /usr/share/mysql
lc-messages               = en_US
skip-external-locking
#
# Broken reverse DNS slows down connections considerably and name resolve is
# safe to skip if there are no "host by domain name" access grants
#skip-name-resolve
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address             = 127.0.0.1
#
# * Fine Tuning
#
#key_buffer_size          = 128M
#max_allowed_packet       = 1G
#thread_stack             = 192K
#thread_cache_size        = 8
# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
#myisam_recover_options    = BACKUP
#max_connections           = 100
#table_cache               = 64
#

```

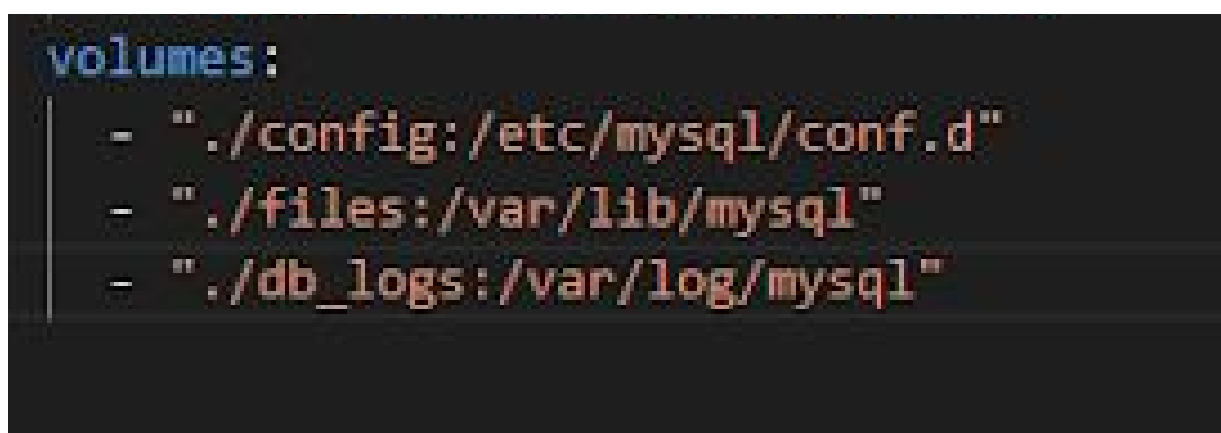
#### Anexo 1.5 interior de archivo 50-server.cnf



Anexo 1.6 conexión al gestor de la base de datos del contenedor



Anexo 1.7 ejemplo de los volúmenes sin configurar



Anexo 1.8 ejemplo de los volúmenes ya configurados

```

File Edit Selection View Go Run Terminal Help
error.log - Practica 1 - Visual Studio Code

EXPLORER
...
OPEN EDITORS
PRACTICA 1
  config
  db_logs
    error.log
    mysql-bin.000001
    mysql-bin.000002
    mysql-bin.index
  files
  .gitignore
  docker-compose.yml
  respaldos1.sql

db_logs > error.log
> 2020-10-01 2:40:00 0 [Note] InnoDB: Using x86_64 CRC32 instructions
6 2020-10-01 2:40:00 0 [Note] mysqld: O_TMPFILE is not supported on /tmp (disabling future attempts)
7 2020-10-01 2:40:00 0 [Note] InnoDB: Initializing buffer pool, total size = 134217728, chunk size = 134217728
8 2020-10-01 2:40:00 0 [Note] InnoDB: Completed initialization of buffer pool
9 2020-10-01 2:40:00 0 [Note] InnoDB: If the mysqld execution user is authorized, page cleaner thread priority can be changed. See the man page of setpriority().
10 2020-10-01 2:40:00 0 [Note] InnoDB: Starting crash recovery from checkpoint LSN=49780
11 2020-10-01 2:40:00 0 [Note] InnoDB: Last binlog file '/var/log/mysql/mysql-bin.000001', position 1089
12 2020-10-01 2:40:00 0 [Note] InnoDB: 128 rollback segments are active.
13 2020-10-01 2:40:00 0 [Note] InnoDB: Removed temporary tablespace data file: "ibtmp1"
14 2020-10-01 2:40:00 0 [Note] InnoDB: Creating shared tablespace for temporary tables
15 2020-10-01 2:40:00 0 [Note] InnoDB: Setting file './ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...
16 2020-10-01 2:40:00 0 [Note] InnoDB: File './ibtmp1' size is now 12 MB.
17 2020-10-01 2:40:00 0 [Note] InnoDB: 10.5.5 started; log sequence number 49792; transaction id 39
18 2020-10-01 2:40:00 0 [Note] Plugin 'FEEDBACK' is disabled.
19 2020-10-01 2:40:00 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
20 2020-10-01 2:40:00 0 [Note] InnoDB: Buffer pool(s) load completed at 201801 2:40:00
21 2020-10-01 2:40:00 0 [Note] Server socket created on IP: '0.0.0.0'.
22 2020-10-01 2:40:00 0 [Note] Reading of all Master_info entries succeeded
23 2020-10-01 2:40:00 0 [Note] Added new Master_info '' to hash table
24 2020-10-01 2:40:00 0 [Note] mysqld: ready for connections.
25 Version: '10.5.5-MariaDB-1:10.5.5+maria-focal-log' socket: '/run/mysqld/mysqld.sock' port: 3306 mariadb.org binary distribution
26 2020-10-01 2:40:19 4 [Warning] IP address '172.22.0.1' could not be resolved: Name or service not known
27

```

## Anexo 1.9 registro de las conexiones erróneas.