

Instituto Tecnológico Superior de Lerdo



Materia

Tópicos de Bases de datos

Profesor

Francisco Eduardo Rodríguez Campos

Manual de Instalación Firebird Docker en Ubuntu.

Nombre del Alumno

Guerrero Artiño Marcos Adrian

No. De Control

17231726

Fecha de Entrega

30/10/2020

Introducción.

Este manual tiene como objetivo mostrar paso a paso como se instala el gestor de Base de datos llamado Firebird empezando con la instalación de su imagen y su implementación, para posteriormente realizar una conexión local en localhost y después una conexión remota.

La tecnología Docker nos permite crear contenedores con Servicios o aplicaciones, lo cual en este manual se creará el contenedor necesario para Mongo de forma sencilla y entendible y al final de procederá a conectar remotamente con nuestro servidor.

Acceso protocolo SSH a consola de Ubuntu.

En nuestro caso, descargamos la consola llamada CMDER para conectarme remotamente a la consola de mi servidor, que en mi caso es Ubuntu.

Como primer paso, debo saber mi dirección IP de mi servidor.

```
yodameme@yodameme-VirtualBox:~$ hostname -I  
192.168.100.18 172.17.0.1 2806:2f0:3160:40:f10  
7e63:c831:2951:1a69
```

Para realizar la conexión, en CMDER colocamos el siguiente comando.

```
C:\Users\marco\Desktop\Consola  
λ ssh yodameme@192.168.100.18
```

Seguidamente nos pedirá la contraseña del usuario y nos preguntará si estamos seguros de realizar la conexión. Finalmente nos conectaremos a la consola de nuestro servidor Linux desde Windows.

```
λ Cmder  
yodameme@yodameme-VirtualBox:~$ |
```

Verificación Docker.

Como primer paso debemos verificar que Docker esté instalado con el siguiente comando:

Sudo service docker status

```
Cmder
yodameme@yodameme-VirtualBox:~$ sudo service docker status
[sudo] password for yodameme:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2020-10-21 22:51:03 CDT; 15min ago
     Triggers: ● docker.socket
```

Una vez checado que tenemos activo Docker en el sistema, verificamos las imágenes que Docker tiene instalada para verificar que no tengamos una imagen previa de MongoDB.

Sudo Docker images

```
yodameme@yodameme-VirtualBox:~$ sudo docker images
[sudo] password for yodameme:
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mysql/mysql-server   latest             8a3a24ad33be       3 months ago       366MB
hello-world          latest             bf756fb1ae65       9 months ago       13.3kB
yodameme@yodameme-VirtualBox:~$
```

Instalación de Imagen Firebird.

No existe una imagen oficial de FireBird, pero existen imágenes que se adaptan y se han creado para este gestor. Para descargar la imagen colocamos el siguiente comando:

Sudo docker pull mladenp87/firebird-1.5.6-ss

```
yodameme@yodameme-VirtualBox:~$ sudo docker pull mladenp87/firebird-1.5.6-ss
```

Verificamos que la imagen de Firebird se haya descargado satisfactoriamente con el siguiente comando:

Sudo docker images

```
yodameme@yodameme-VirtualBox:~$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
postgres            latest             c96f8b6bc0d9       2 weeks ago       314MB
dpag/pgadmin4       latest             e7a7e02ba7af       2 weeks ago       224MB
mongoclient/mongoclient latest          16ff4e68d176       3 weeks ago       1.18GB
mongo               4.2               0a2f1fdf242c       4 weeks ago       387MB
mongo               latest             ba0c2ff8d362       4 weeks ago       492MB
mysql/mysql-server   latest             8a3a24ad33be       3 months ago       366MB
hello-world          latest             bf756fb1ae65       10 months ago     13.3kB
mladenp87/firebird-1.5.6-ss latest          ed5d3eb6683d       2 years ago       210MB
yodameme@yodameme-VirtualBox:~$
```

Implementación de instancia FireBird como container.

Ya con la imagen disponible de FireBird, necesitamos implementarla. Para esto ejecutamos el contenedor con el siguiente comando:

```
sudo docker run -d --name firebird -p 3051:3051 mladenp87/firebird-1.5.6-ss
```

```
yodameme@yodameme-VirtualBox:~$ sudo docker run -d --name firebird -p 3051:3051 mladenp87/firebird-1.5.6-ss
902420b66107ff34dd3b3c96f8324973637aec9606a24edeb11ac8b0bf105a44
```

Para ver nuestro contenedor utilizamos el siguiente comando:

```
Sudo docker ps -a
```

```
yodameme@yodameme-VirtualBox:~$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
902420b66107	mladenp87/firebird-1.5.6-ss	"/opt/firebird/run.sh"	About a minute ago	Up About a minute	3050/tcp, 0.0.0.0:3051->3051/tcp

Conexión a container y Shell de Firebird.

Como primer paso, debemos conectarnos al container que acabamos de generar con el nombre mongocontainer con el comando **exec** de docker:

```
Sudo docker exec -i -t firebird /bin/bash
```

```
yodameme@yodameme-VirtualBox:~$ sudo docker exec -i -t firebird /bin/bash
root@902420b66107:/# |
```

- Root es el usuario root para entrar al container.
- 902420b66107 es el ID de mi container.

Para poder crear una base de datos, necesitamos ingresar al Shell de mongo, para esto solo escribimos **isql** en la terminal.

```
yodameme@yodameme-VirtualBox:~$ sudo docker exec -i -t firebird /bin/bash
root@902420b66107:/# isql|
```

Una vez que entramos, nos aparece el Shell de mongo, indicando primero la versión de mongo.

```
root@902420b66107:/# isql
Use CONNECT or CREATE DATABASE to specify a database
SQL> |
```

Ya podemos ingresar comandos.

Creación de base de datos de prueba.

Una vez que ya hemos entrado al Shell de Firebird, procedemos a crear una base de datos que se llame "baseitsl", para esto ingresamos el comando:

Créate database "/var/lib/baseitsl.fdb" user 'SYSDBA' password 'masterkey';

```
SQL> create database "/var/lib/baseitsl.fdb" user 'SYSDBA' password 'masterkey';
```

Verificamos que se haya creado la base de datos con el comando:

Show database;

```
SQL> show database;
Database: /var/lib/baseitsl.fdb
Owner: SYSDBA
PAGE_SIZE 4096
Number of DB pages allocated = 146
Sweep interval = 20000
Forced Writes are OFF
Transaction - oldest = 1
Transaction - oldest active = 2
Transaction - oldest snapshot = 2
Transaction - Next = 5
Default Character set: NONE
SQL> |
```

Ahora, procedemos a conectarnos a la base de datos, para esto utilizamos el comando:

connect "/var/lib/baseitsl.fdb" user 'SYSDBA' password 'masterkey';

```
SQL> connect "/var/lib/baseitsl.fdb" user 'SYSDBA' password 'masterkey';
Commit current transaction (y/n)?y
Committing.
Database: "/var/lib/baseitsl.fdb", User: SYSDBA
SQL> |
```

Ya dentro, creamos una tabla llamada alumnos con el siguiente comando:

create table alumnos (id int not null, nombre varchar(20), apellido varchar(20), edad int, carrera varchar(15), primary key(id));

```
Database: "/var/lib/baseitsl.fdb", User: SYSDBA
SQL> create table alumnos (id int not null, nombre varchar(20), apellido varchar(20), edad int, carrera varchar(15), primary key(id));
```

Para mostrar esa tabla, ingresamos el comando:

Show tables;

```
SQL> show tables;
ALUMNOS
SQL> |
```

Configuración remota Firebird.

Para crear la conexión remota con nuestra base de datos es necesario realizar una serie de pasos de configuración.

Lo primero es ingresar al bash del container de Firebird con el comando:

Sudo docker exec -i -t firebird /bin/bash

```
yodameme@yodameme-VirtualBox:~$ sudo docker exec -i -t firebird /bin/bash
root@902420b66107:/#
```

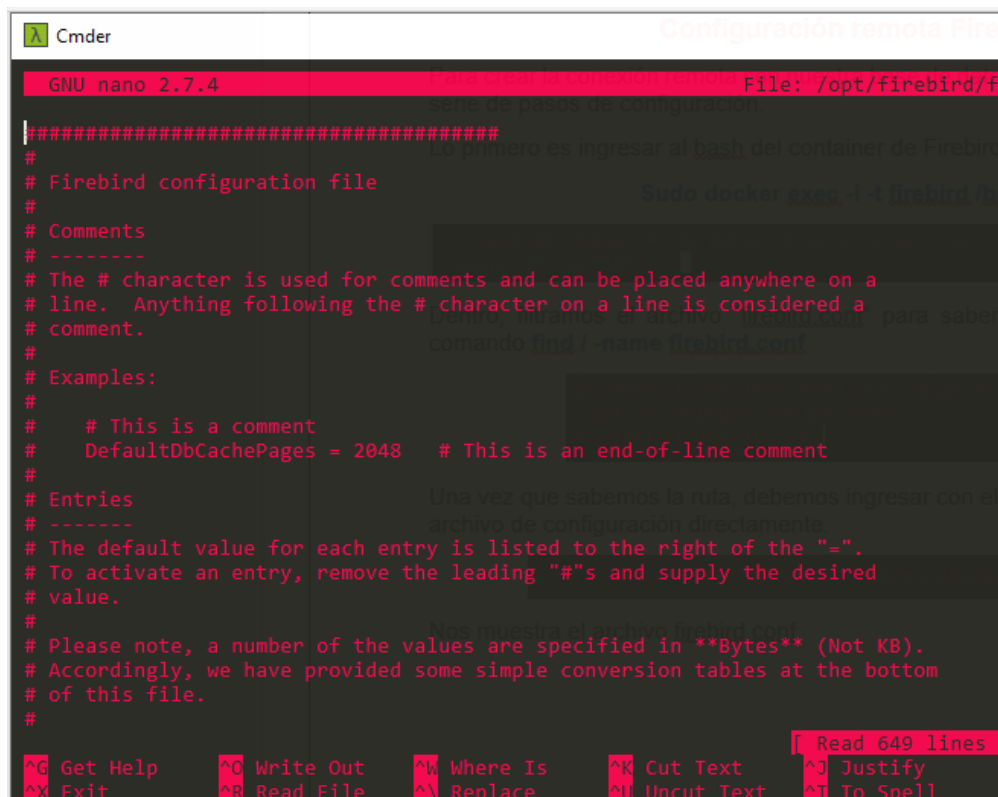
Dentro, filtramos el archivo "firebird.conf" para saber su ruta de acceso con el comando **find / -name firebird.conf**

```
root@902420b66107:/# find / -name firebird.conf
/opt/firebird/firebird.conf
root@902420b66107:/#
```

Una vez que sabemos la ruta, debemos ingresar con el comando nano para abrir el archivo de configuración directamente.

```
root@902420b66107:/# nano /opt/firebird/firebird.conf
```

Nos muestra el archivo firebird.conf.



The screenshot shows the nano text editor interface. The title bar at the top says "Cmdr" and "Configuración remota Fire". The editor window shows the first part of the firebird.conf file. The status bar at the bottom indicates "Read 649 lines".

```
GNU nano 2.7.4 File: /opt/firebird/f
#####
#
# Firebird configuration file
#
# Comments
# -----
# The # character is used for comments and can be placed anywhere on a
# line. Anything following the # character on a line is considered a
# comment.
#
# Examples:
#
#   # This is a comment
#   DefaultDbCachePages = 2048   # This is an end-of-line comment
#
# Entries
# -----
# The default value for each entry is listed to the right of the "=".
# To activate an entry, remove the leading "#s and supply the desired
# value.
#
# Please note, a number of the values are specified in **Bytes** (Not KB).
# Accordingly, we have provided some simple conversion tables at the bottom
# of this file.
#
```

Read 649 lines

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell

Cambiamos las siguientes líneas.

De #DatabaseAccess = Full a DatabaseAccess = Full

```
#  
DatabaseAccess = Full
```

De #ExternalFileAccess = None a ExternalFileAccess = Full

```
#  
ExternalFileAccess = Full
```

De #RemoteBindAddress = a RemoteBindAddress = 0.0.0.0

```
#  
RemoteBindAddress = 0.0.0.0
```

De #RemoteFileOpenAbility = 0 a RemoteFileOpenAbility = 1

```
# type: boolean  
#  
RemoteFileOpenAbility = 1
```

Una vez hemos cambiado eso, guardamos el archivo con Ctrl+O y presionamos Enter.

Salimos del bash del container con Exit.

```
root@902420b66107:/# exit  
exit  
yodameme@yodameme-VirtualBox:~$
```


Configuración de puertos Ubuntu.

En Ubuntu, debemos habilitar el puerto asignado a firebird que en este caso fue el 3051, para esto ingresamos el siguiente comando:

Sudo ufw allow 3051/tcp

```
yodameme@yodameme-VirtualBox:~$ sudo ufw allow 3051/tcp
Rule added
Rule added (v6)
```

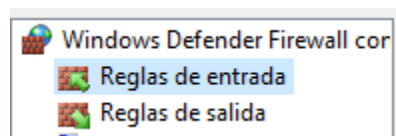
Luego verificamos con el comando **sudo ufw status**.

```
Rule added (v6)
yodameme@yodameme-VirtualBox:~$ sudo ufw status
Status: active

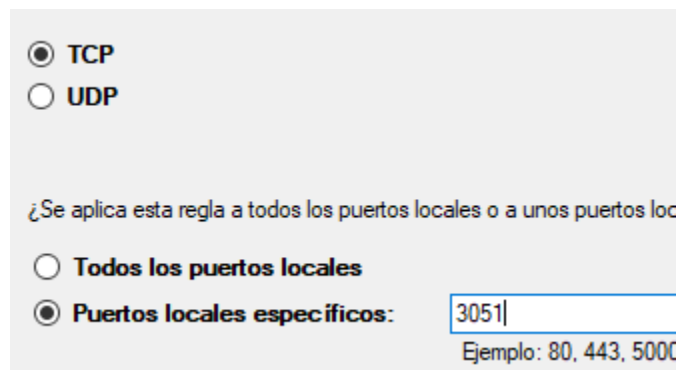
To Action From
--
3050/tcp ALLOW Anywhere
22/tcp ALLOW Anywhere
3389/tcp ALLOW Anywhere
3051/tcp ALLOW Anywhere
3050/tcp (v6) ALLOW Anywhere (v6)
22/tcp (v6) ALLOW Anywhere (v6)
3389/tcp (v6) ALLOW Anywhere (v6)
3051/tcp (v6) ALLOW Anywhere (v6)
```

Configuración de puertos Windows.

En Windows habilitamos las reglas de entrada y de salida del Firewall de Windows al puerto 3051.

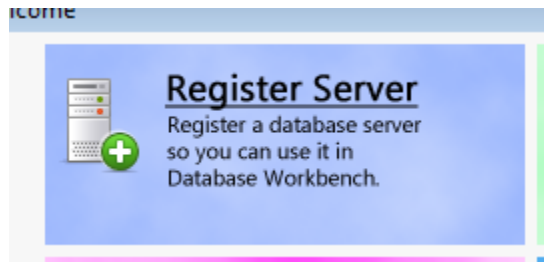


El proceso es el mismo a las documentaciones anteriores.



Crear conexión con un GUI de mongo de forma Remota.

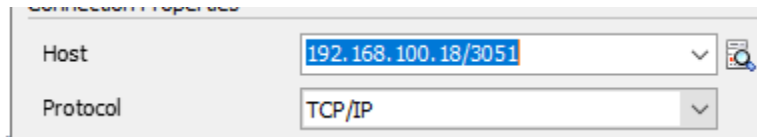
Como primer paso, abrimos Database Workbench y damos clic en Register Server.



Seleccionamos Firebird.



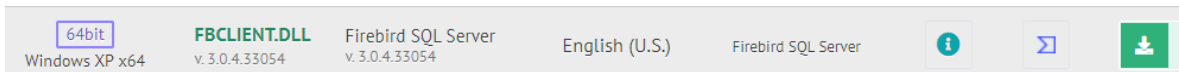
Agregamos un Alias para la conexión y agregamos la IP del servidor, en este caso 192.168.1.70 de Ubuntu y el puerto 3050 del socket de comunicación.



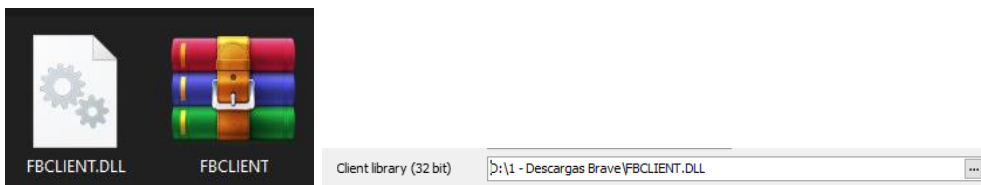
En cliente Library debemos colocar un archivo que se descarga en el siguiente link:

- <https://es.azdill.net/files/fbclient-dll>

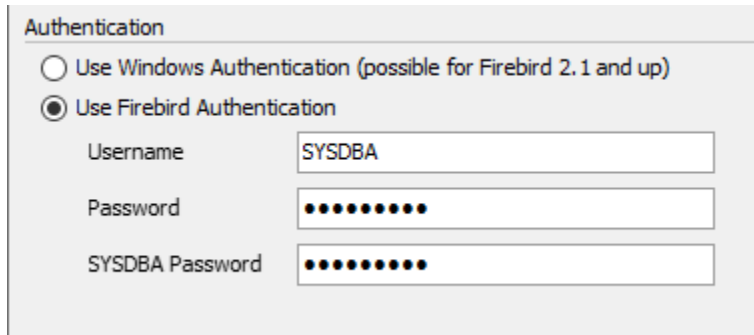
Descargamos el de 64 bits.



Ahora en los 3 puntitos de Client Library, seleccionamos el .Zip descomprimido previamente descargado.



Por último, ingresamos la contraseña que hemos puesto al principio de la instalación de Firebird.

A dialog box titled "Authentication" with two radio button options. The first option is "Use Windows Authentication (possible for Firebird 2.1 and up)". The second option, "Use Firebird Authentication", is selected. Below the options are three text input fields: "Username" containing "SYSDBA", "Password" with masked characters, and "SYSDBA Password" also with masked characters.

Authentication

☐ Use Windows Authentication (possible for Firebird 2.1 and up)

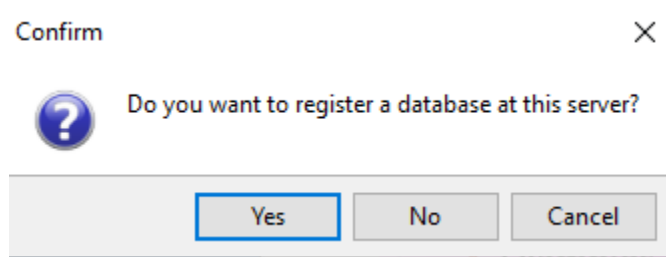
☒ Use Firebird Authentication

Username: SYSDBA

Password: [masked]

SYSDBA Password: [masked]

Damos clic en Ok y en la siguiente ventana damos que no.

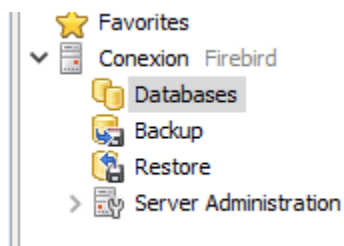
A small dialog box titled "Confirm" with a question mark icon. The text inside asks "Do you want to register a database at this server?". At the bottom are three buttons: "Yes", "No", and "Cancel". The "Yes" button is highlighted with a blue border.

Confirm

Do you want to register a database at this server?

Yes No Cancel

Se ha creado la conexión satisfactoriamente.



Conclusiones.

Docker es una tecnología que nos permite almacenar servicios o aplicaciones en contenedores alojados en un servidor de forma rápida y de sencillo uso. Aprendí a crear una imagen y después un contenedor. Docker Hub es una plataforma donde explican paso a paso la instalación de una imagen de un servicio y me sirvió para investigar e indagar más sobre esta nueva tecnología.