



Compliance Challenge

MANUAL DE USO

Agustin Zeppa

Tabla de contenido

Consideraciones y explicación del sistema realizado..... 3

Instalación 4

Prueba..... 5

Consideraciones y Explicación del sistema realizado

Desarrollado en: Python y MYSQL

El sistema realizado consta de dos aplicaciones: un agente que extrae información del servidor en donde ejecuta, y una API (Application programming interface), que recibe esa información por parte del agente, y la persiste en una base de datos.

Dado que la comunicación entre ambas aplicaciones debe ser vía internet, se decidió deployear la API en la nube. La misma se encuentra en los servidores de PYTHONANYWHERE, junto a una base de datos MYSQL.

La base de datos ya se encuentra creada con todas sus tablas y procedimientos en la nube. No hay necesidad de crearla. Para ver como se hizo este proceso, se comparte el script de creación. (SCRIPT_DB.SQL)

Diseño de la Base de Datos

Ver *DER MELI.pdf* antes de seguir.

La base de datos contiene cinco store procedures para facilitar la inserción y verificación de datos.

- **Insert_os_data:** inserta los datos del sistema operativo en la base. Previamente verifica que la información que va a persistir no esté ya presente.
- **Insert_processor_data:** inserta la información del procesador instalado en los servidores en la base. Previamente verifica que la información que va a persistir no esté ya presente.
- **Insert_servers_data:** inserta los datos del servidor en la base. Dado que la tabla de servidores tiene dos foreign keys, primero selecciona el ID del sistema operativo y del procesador asociado al mismo. Previamente a insertar verifica que el servidor que va a persistir no exista ya en la base. Una vez finalizado esto, selecciona el ID del servidor (autogenerado por la base de datos) y lo inserta junto a la fecha de ejecución en la tabla de executions.
- **Insert_processes_data:** inserta los datos de los procesos ejecutando en el servidor en la base. Previamente verifica que la información que va a persistir no esté ya presente. Por cada proceso que se inserta, se selecciona el ID de ejecución, y el ID (índice autogenerado por la base) del proceso para poder actualizar la tabla de procesos por ejecución.
Aclaración: se utiliza un índice como PK ya que podría darse la situación de que existan 2 procesos distintos, provenientes de servidores diferentes, con un mismo numero PID.
- **Insert_users_data:** inserta los datos de los usuarios conectado al servidor en la base. Previamente verifica que la información del usuario que va a persistir no exista ya en la base. Posteriormente, selecciona el ID del usuario autogenerado y el ID de ejecución, y se actualiza la tabla de usuarios por ejecución.

Nota: tanto los usuarios como los procesos, están conectados a la tabla de ejecuciones mediante relaciones mucho a muchos. Y a su vez, la tabla de ejecuciones se relaciona con los servidores. De esta forma puedo ver los usuarios y procesos corriendo en un servidor, en un determinado momento, y compararlo contra otras ejecuciones.

Las tablas de usuarios y procesos, listan la información, independientemente de donde provenga. Luego se establecen las relaciones mediante las otras tablas.

Existe una situación extraña en donde podrían llegar 2 procesos con mismo PID y nombre de distintos servidores. Se podría considerar que ambos son procesos distintos, ya que vienen de orígenes diferentes, pero dado que el objetivo del sistema es listar los procesos (con nombre y PID) por servidores, se decidió que ante esta situación, el proceso se almacene una sola vez, y establezca su relación mediante la tabla de procesos por ejecución. Como solo me interesa listar el PID, y el nombre del proceso, no se inserta un duplicado del mismo.

Instalación

Dado que la API y la base de datos se encuentran desplegadas en la nube, la instalación de las librerías utilizadas para dicha aplicación y la creación de la base de datos ya fue llevada a cabo.

Por otro lado, para ejecutar el agente, si hace falta instalar ciertas librerías. A continuación se detalla el paso a paso.

Ejecutar en el CMD del sistema, o consola de Linux, dentro del directorio del proyecto:

(Usar PIP o PIP3 dependiendo la versión de python)

- **Pip3 install py-cpuinfo**
- **Pip3 install psutil**
- **Pip3 install requests**
- **Pip3 install sockets**

Se comparte un archivo “requirements.txt” para ver todas las librerías asociadas al programa agente por si se genera un error de imports.

En caso de que haya un error: ejecutar **pip install -r requirements.txt**

Sin embargo, dado que en ciertas ocasiones python no puede descargar todas las librerías del archivo requirements: si no se produce el error, realizar la instalación manualmente como se especifico anteriormente.

Prueba

Dado que Pythonanywhere no permite el acceso a la bases de datos localmente sin tener una cuenta paga en los servidores:

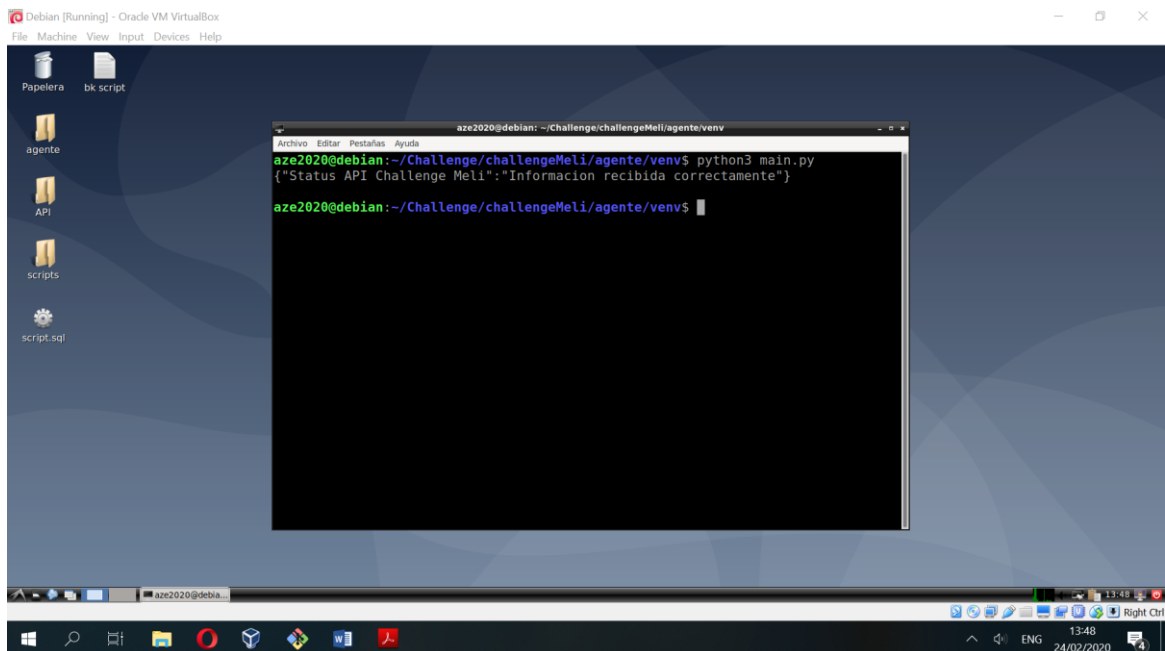
<https://help.pythonanywhere.com/pages/AccessingMySQLFromOutsidePythonAnywhere/>

Para poder acceder a la base de datos se compartirá un link mediante el cual se accederá a una consola y se ejecutaran algunos scripts para corroborar el buen funcionamiento de la base. **El link llegara al mail de la persona que ejecute la prueba.**

Ejecución del agente:

1. Abrir una consola e ingresar al directorio: challengeMeli/agente/venv
2. Ejecutar el archivo main.py (python main.py o python3 main.py)

La ejecución demora unos segundos y muestra un mensaje como el siguiente:

A screenshot of a terminal window titled 'Debian [Running] - Oracle VM VirtualBox'. The terminal shows the user 'aze2020@debian' in the directory '~/Challenge/challengeMeli/agente/venv'. The command 'python3 main.py' has been executed, resulting in the output: '{"Status API Challenge Meli": "Informacion recibida correctamente"}'. The terminal window is open on a desktop environment with icons for 'Papelera', 'bk script', 'agente', 'API', 'scripts', and 'script.sql'. The system tray at the bottom shows the time as 13:48 on 24/02/2020.

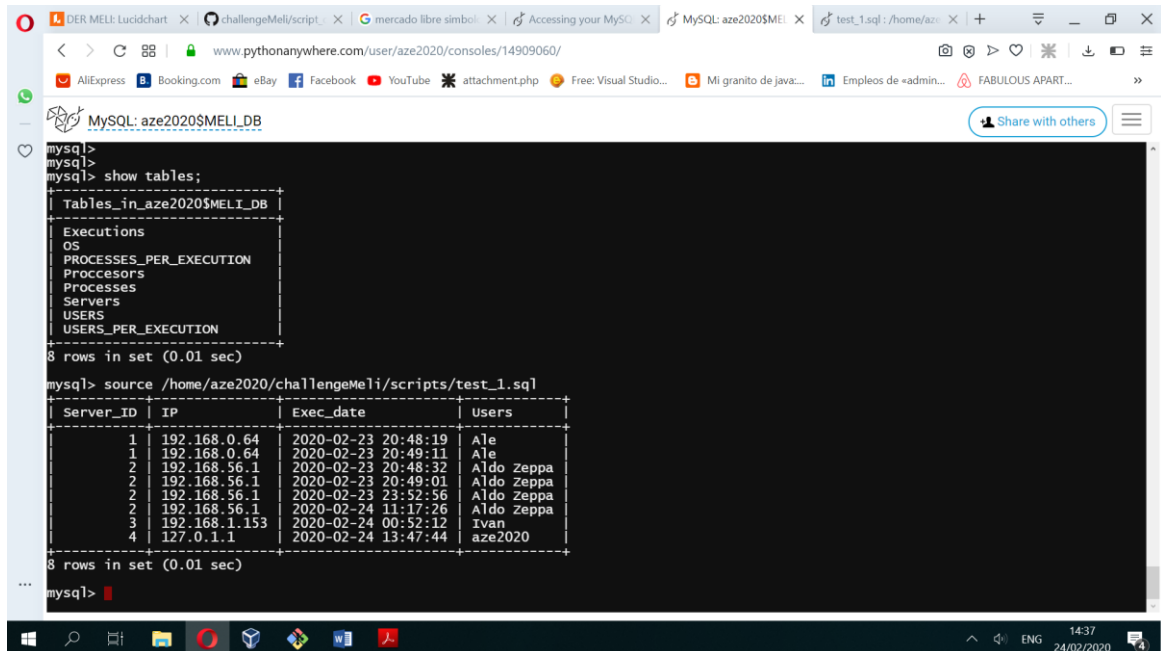
3. Para ver la información en la base de datos, ingresar al link compartido por mail y ejecutar los siguientes test (o hacer los SELECT que deseen).

NOTA: Los prints a continuación muestran múltiples ejecuciones que utilice de testing. Al momento de la prueba, la base se encontrara vacía.

Test 1: nombres de usuarios por servidor y fecha de ejecución.

Sentencia a ejecutar:

- `source /home/aze2020/challengeMeli/scripts/test_1.sql`



```
mysql> show tables;
+-----+
| Tables_in_aze2020$MELI_DB |
+-----+
| Executions                 |
| OS                         |
| PROCESSES_PER_EXECUTION   |
| Processors                 |
| Processes                  |
| Servers                    |
| USERS                      |
| USERS_PER_EXECUTION        |
+-----+
8 rows in set (0.01 sec)

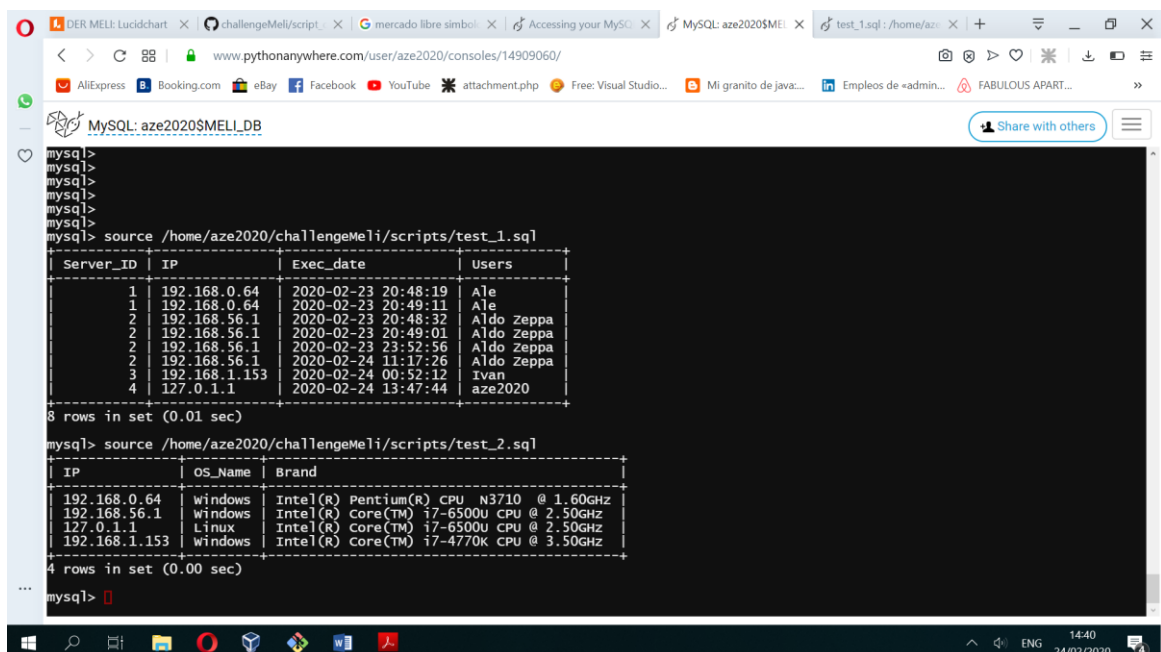
mysql> source /home/aze2020/challengeMeli/scripts/test_1.sql
+-----+
| Server_ID | IP           | Exec_date           | Users |
+-----+
| 1         | 192.168.0.64 | 2020-02-23 20:48:19 | Ale   |
| 1         | 192.168.0.64 | 2020-02-23 20:49:11 | Ale   |
| 2         | 192.168.56.1 | 2020-02-23 20:48:32 | Aldo Zeppa |
| 2         | 192.168.56.1 | 2020-02-23 20:49:01 | Aldo Zeppa |
| 2         | 192.168.56.1 | 2020-02-23 23:52:56 | Aldo Zeppa |
| 2         | 192.168.56.1 | 2020-02-24 11:17:26 | Aldo Zeppa |
| 3         | 192.168.1.153 | 2020-02-24 00:52:12 | Ivan  |
| 4         | 127.0.1.1     | 2020-02-24 13:47:44 | aze2020 |
+-----+
8 rows in set (0.01 sec)

mysql>
```

Test 2: Sistema operativo y procesador asociado a cada servidor

Sentencia a ejecutar:

- `source /home/aze2020/challengeMeli/scripts/test_2.sql`



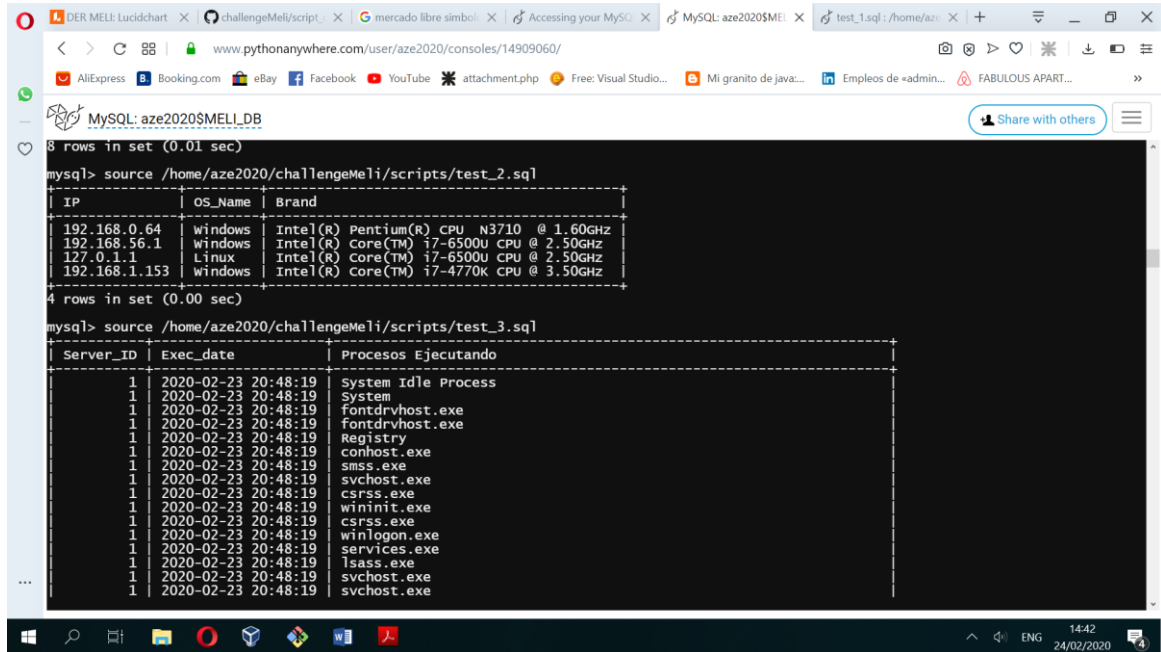
```
mysql> source /home/aze2020/challengeMeli/scripts/test_2.sql
+-----+
| IP           | OS_Name | Brand |
+-----+
| 192.168.0.64 | windows | Intel(R) Pentium(R) CPU N3710 @ 1.60GHz |
| 192.168.56.1 | windows | Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz |
| 127.0.1.1     | Linux   | Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz |
| 192.168.1.153 | windows | Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz |
+-----+
4 rows in set (0.00 sec)

mysql>
```

Test 3: procesos ejecutando por servidor y ejecución

Sentencia ejecutada:

- `source /home/aze2020/challengeMeli/scripts/test_3.sql`



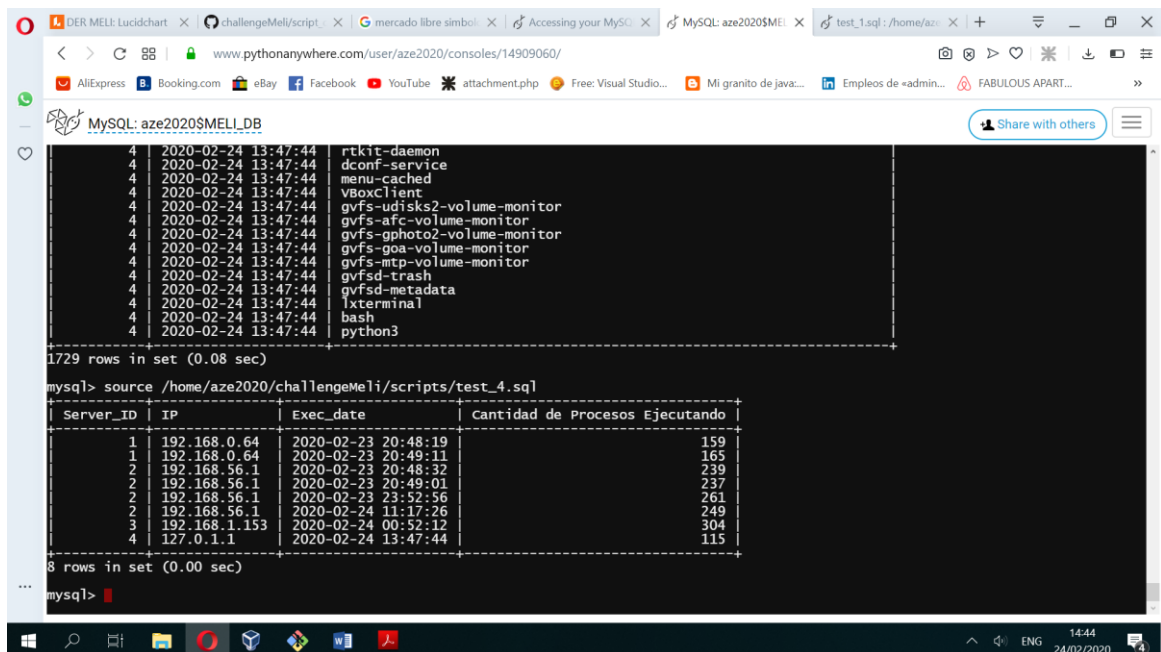
```
mysql> source /home/aze2020/challengeMeli/scripts/test_2.sql
+----+-----+-----+
| IP      | OS_Name | Brand |
+----+-----+-----+
| 192.168.0.64 | windows | Intel(R) Pentium(R) CPU N3710 @ 1.60GHz |
| 192.168.56.1 | windows | Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz |
| 127.0.1.1 | Linux | Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz |
| 192.168.1.153 | windows | Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz |
+----+-----+-----+
4 rows in set (0.00 sec)

mysql> source /home/aze2020/challengeMeli/scripts/test_3.sql
+----+-----+-----+
| Server_ID | Exec_date | Procesos Ejecutando |
+----+-----+-----+
| 1 | 2020-02-23 20:48:19 | System Idle Process |
| 1 | 2020-02-23 20:48:19 | System |
| 1 | 2020-02-23 20:48:19 | fontdrvhost.exe |
| 1 | 2020-02-23 20:48:19 | fontdrvhost.exe |
| 1 | 2020-02-23 20:48:19 | Registry |
| 1 | 2020-02-23 20:48:19 | conhost.exe |
| 1 | 2020-02-23 20:48:19 | smss.exe |
| 1 | 2020-02-23 20:48:19 | svchost.exe |
| 1 | 2020-02-23 20:48:19 | csrss.exe |
| 1 | 2020-02-23 20:48:19 | wininit.exe |
| 1 | 2020-02-23 20:48:19 | csrss.exe |
| 1 | 2020-02-23 20:48:19 | winlogon.exe |
| 1 | 2020-02-23 20:48:19 | services.exe |
| 1 | 2020-02-23 20:48:19 | lsass.exe |
| 1 | 2020-02-23 20:48:19 | svchost.exe |
| 1 | 2020-02-23 20:48:19 | svchost.exe |
+----+-----+-----+
```

Test 4: cantidad de procesos ejecutando por servidor en cada ejecución.

Sentencia ejecutada:

- `source /home/aze2020/challengeMeli/scripts/test_4.sql`



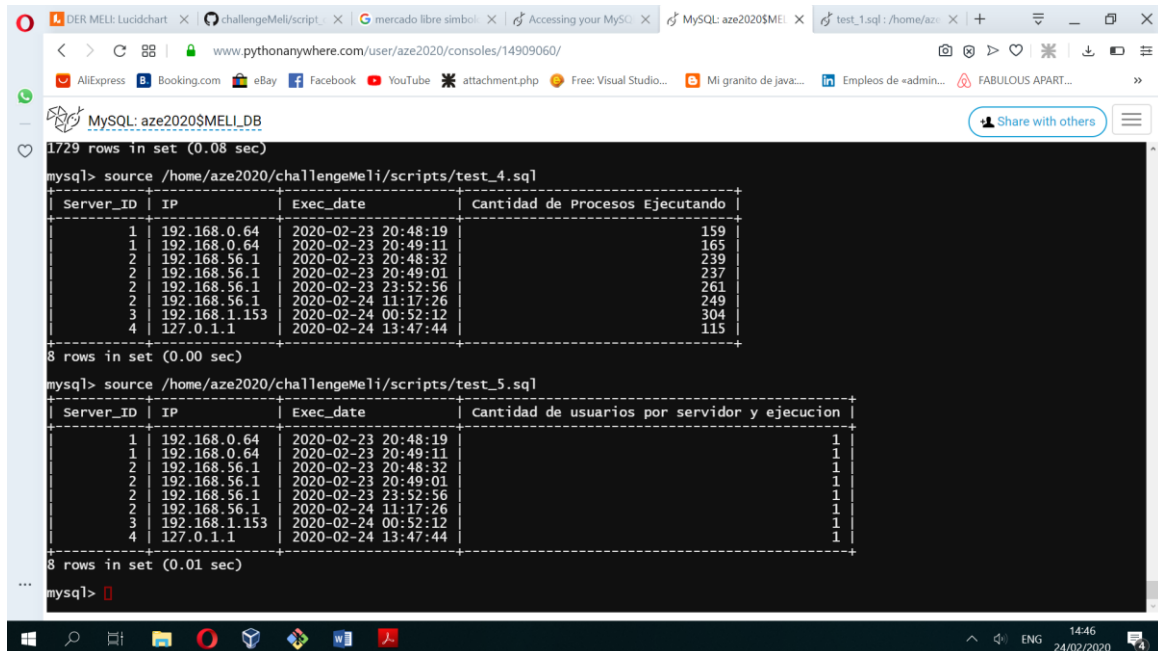
```
mysql> source /home/aze2020/challengeMeli/scripts/test_4.sql
+----+-----+-----+
| Server_ID | IP      | Exec_date | Cantidad de Procesos Ejecutando |
+----+-----+-----+
| 1 | 192.168.0.64 | 2020-02-23 20:48:19 | 159 |
| 1 | 192.168.0.64 | 2020-02-23 20:49:11 | 165 |
| 2 | 192.168.56.1 | 2020-02-23 20:48:32 | 239 |
| 2 | 192.168.56.1 | 2020-02-23 20:49:01 | 237 |
| 2 | 192.168.56.1 | 2020-02-23 23:52:56 | 261 |
| 2 | 192.168.56.1 | 2020-02-24 11:17:26 | 249 |
| 3 | 192.168.1.153 | 2020-02-24 00:52:12 | 304 |
| 4 | 127.0.1.1 | 2020-02-24 13:47:44 | 115 |
+----+-----+-----+
8 rows in set (0.00 sec)

mysql>
```

Test 5: Cantidad de usuarios por servidor y ejecución

Sentencia ejecutada:

- `source /home/aze2020/challengeMeli/scripts/test_5.sql`



```
mysql> source /home/aze2020/challengeMeli/scripts/test_4.sql
1729 rows in set (0.08 sec)

mysql> source /home/aze2020/challengeMeli/scripts/test_5.sql
8 rows in set (0.00 sec)

mysql> source /home/aze2020/challengeMeli/scripts/test_5.sql
8 rows in set (0.01 sec)

mysql>
```

Server_ID	IP	Exec_date	Cantidad de Procesos Ejecutando
1	192.168.0.64	2020-02-23 20:48:19	159
1	192.168.0.64	2020-02-23 20:49:11	165
2	192.168.56.1	2020-02-23 20:48:32	239
2	192.168.56.1	2020-02-23 20:49:01	237
2	192.168.56.1	2020-02-23 23:52:56	261
2	192.168.56.1	2020-02-24 11:17:26	249
3	192.168.1.153	2020-02-24 00:52:12	304
4	127.0.1.1	2020-02-24 13:47:44	115

Server_ID	IP	Exec_date	Cantidad de usuarios por servidor y ejecucion
1	192.168.0.64	2020-02-23 20:48:19	1
1	192.168.0.64	2020-02-23 20:49:11	1
2	192.168.56.1	2020-02-23 20:48:32	1
2	192.168.56.1	2020-02-23 20:49:01	1
2	192.168.56.1	2020-02-23 23:52:56	1
2	192.168.56.1	2020-02-24 11:17:26	1
3	192.168.1.153	2020-02-24 00:52:12	1
4	127.0.1.1	2020-02-24 13:47:44	1

Otras pruebas opcionales (las tablas son key sensitive):

- `SELECT * FROM OS;`
- `SELECT * FROM Procesos;`
- `SELECT * FROM Servers;`
- `SELECT * FROM Executions;`
- `SELECT * FROM Processes;`
- `SELECT * FROM PROCESSES_PER_EXECUTION;`
- `SELECT * FROM USERS;`
- `SELECT * FROM USERS_PER_EXECUTION;`