TRABAJO DE INVESTIGACIÓN N° 02

SIEM, maduración y valor agregado

Continuando con el Trabajo de Investigación anterior, se plantea la siguiente actividad:

01) Deberán proponer e implementar al menos 3 (tres) acciones concretas que den mayor madurez al SIEM Graylog ya montado. Ejemplos de medidas de maduración:
- Configuración de NTP para que el horario de Graylog sea el correcto.
- Cambio de contraseña por defecto de Admin.
- Creación de cuenta separada para operar Graylog y así no utilizar Admin.
- Implementación de un certificado digital para la web de Graylog.
- Etc.

02) Deberán integrar a Graylog las siguientes plataformas para que dejen sus logs:
a. 1 Ubuntu Linux.
b. 1 Windows Server 2012 con un SQL Server instalado y reportando logs de bases de datos.
c. 1 MySQL instalado en Linux enviando logs al SIEM vía Syslog (podrá ser instalado en el Ubuntu Linux del inciso a).

Para cada tipo de log deberán documentar paso a paso, copiando los comandos lanzados y pantallas impresas que vayan mostrando cómo fueron configurando. Elaborar 1 documento para cada uno de los 4 incisos precedentes (Entregables A, B, C, D).

03) Deberán seleccionar y configurar al menos 2 (dos) alertas para que les lleguen por mail. Queda a discreción de cada grupo qué tipo de eventos son relevantes para configurar alertas. Sugerencia: habitualmente, es un control importante el inicio de sesión de los usuarios privilegiados en las plataformas o el uso de comandos críticos en bases de datos (Entregable E).

Por favor, cualquier supuesto que hagan, generar un apartado al final del documento. Ante cualquier inconveniente, no duden en contactarnos.

Esp. Ing. Carlos I. Tapia.
ctapia@iua.edu.ar

MCs. Ing. Eduardo Casanovas
ecasanovas@iua.edu.ar

Joaquin Cernik, Eduardo Friesen, Alan Marin

TRABAJO PRACTICO DE INVESTIGACION 2

SIEM, maduración y valor agregado

Introducción

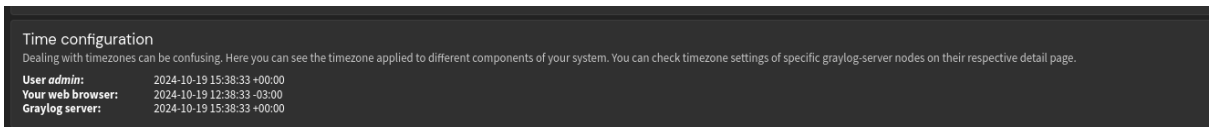
Partiendo desde el ultimo trabajo realizado donde ya tenemos nuestro docker levantado:

```
joaquincernik@dell:~$ sudo docker ps
[sudo] password for joaquincernik:
CONTAINER ID   IMAGE                COMMAND                  CREATED       STATUS                    PORTS
                                    NAMES
1b8fc4a4e7c7   graylog/graylog:5.0   "/usr/bin/tini -- wa…"   6 weeks ago   Up 27 minutes (unhealthy)   0.0.0.0:1514->1514/tcp, :::1514->1514/tcp, 0.0
.0.0:9000->9000/tcp, 0.0.0.0:1514->1514/udp, :::9000->9000/tcp, :::1514->1514/udp, 0.0.0.0:12201->12201/tcp, 0.0.0.0:12201->12201/udp, :::12201->12201
/tcp, :::12201->12201/udp   pruebadocker_graylog_1
```

1. Incrementar el nivel de madurez de nuestro graylog

Configurar mediante NTP correctamente el horario (https://stackoverflow.com/questions/24551592/how-to-make-sure-dockers-time-syncs-with-that-of-the-host)

Como vemos aqui, el horario desplegado por graylog no es el correcto:



Lo que realizamos para solucionar este inconveniente es agregar al docker-compose.yml en la parte de graylog la siguiente línea

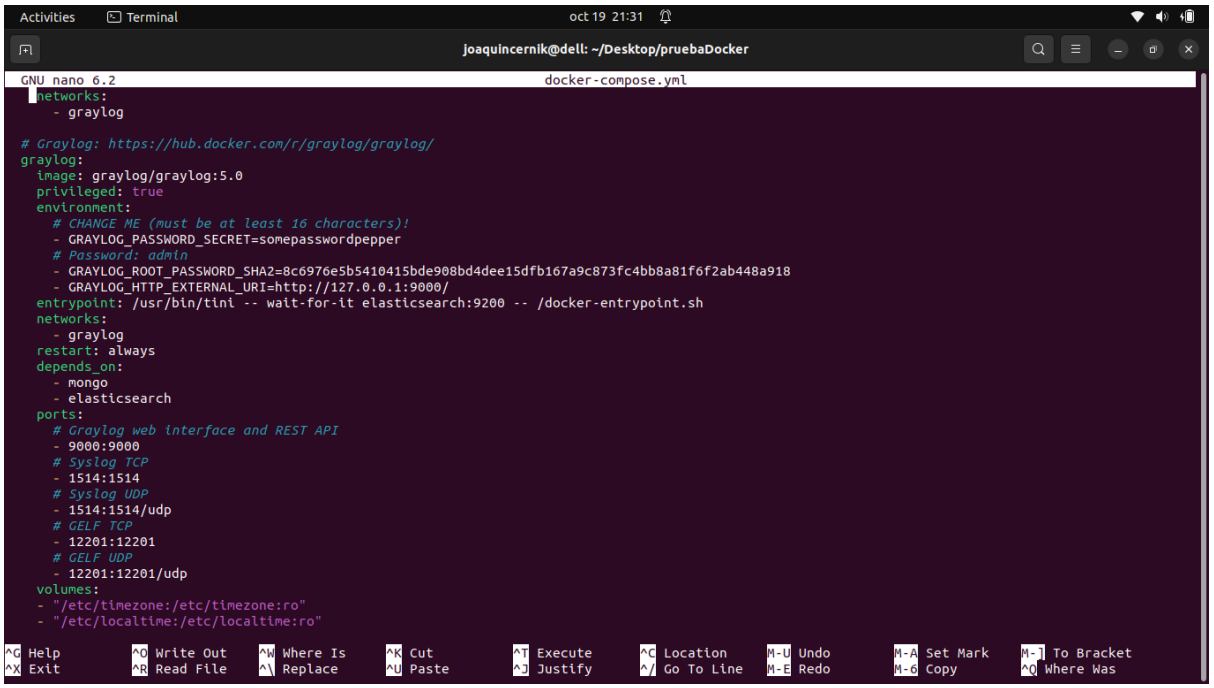
volumes:

- "/etc/timezone:/etc/timezone:ro"

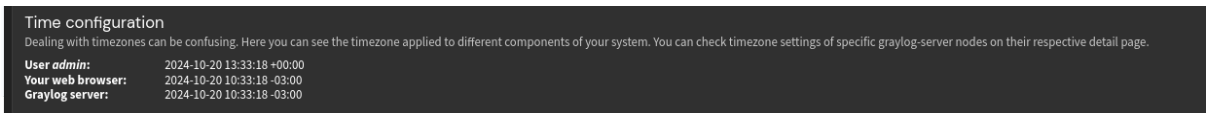
- "/etc/localtime:/etc/localtime:ro"

Esto nos permite crear nuestro contenedor con el horario de nuestra máquina local.

Se adjunta como queda el archivo docker-compose



Ahora la interfaz nos muestra los siguientes horarios:



Cambiar la contraseña por defecto:

Al iniciar graylog, nos ofrece por defecto un usuario **admin** y su contraseña **admin.** Claramente esto es algo que tenemos que cambiar, debido a que deja nuestro SIEM totalmente vulnerable ante cualquier atacante.

Lo que tendremos que realizar es cambiar esta contraseña. Lo haremos modificando nuevamente el docker-compose.

Debido a que graylog no almacena contraseñas en texto plano tendremos que usar un hash para crearla. Usaremos un hash SHA-256, en la terminal ejecutamos lo siguiente:

```
joaquincernik@dell:~/Desktop/pruebaDocker$ echo -n "ADMINseguro01010" | sha256sum
5dbe73df2d50d60ebf9d2779670cfdff6b8c54fb957e7a43b545447634a171fe  -
```

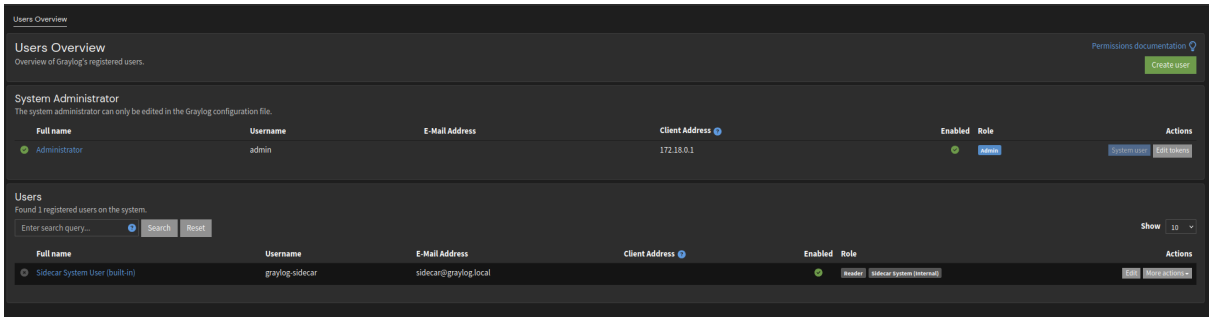
"*ADMINseguro01010*" será nuestra contraseña y utilizaremos lo que nos devolvió la consola para ingresarlo en el archivo composer:

```
# Graylog: https://hub.docker.com/r/graylog/graylog/
graylog:
  image: graylog/graylog:5.0
  privileged: true
  environment:
    # CHANGE ME (must be at least 16 characters)!
    - GRAYLOG_PASSWORD_SECRET=ADMINseguro01010
    # Password: admin
    - GRAYLOG_ROOT_PASSWORD_SHA2=5dbe73df2d50d60ebf9d2779670cfdff6b8c54fb957e7a43b545447634a171fe
    - GRAYLOG_HTTP_EXTERNAL_URI=http://127.0.0.1:9000/
  entrypoint: /usr/bin/tini -- wait-for-it elasticsearch:9200 -- /docker-entrypoint.sh
  networks:
    - graylog
  restart: always
  depends_on:
    - mongo
    - elasticsearch
```

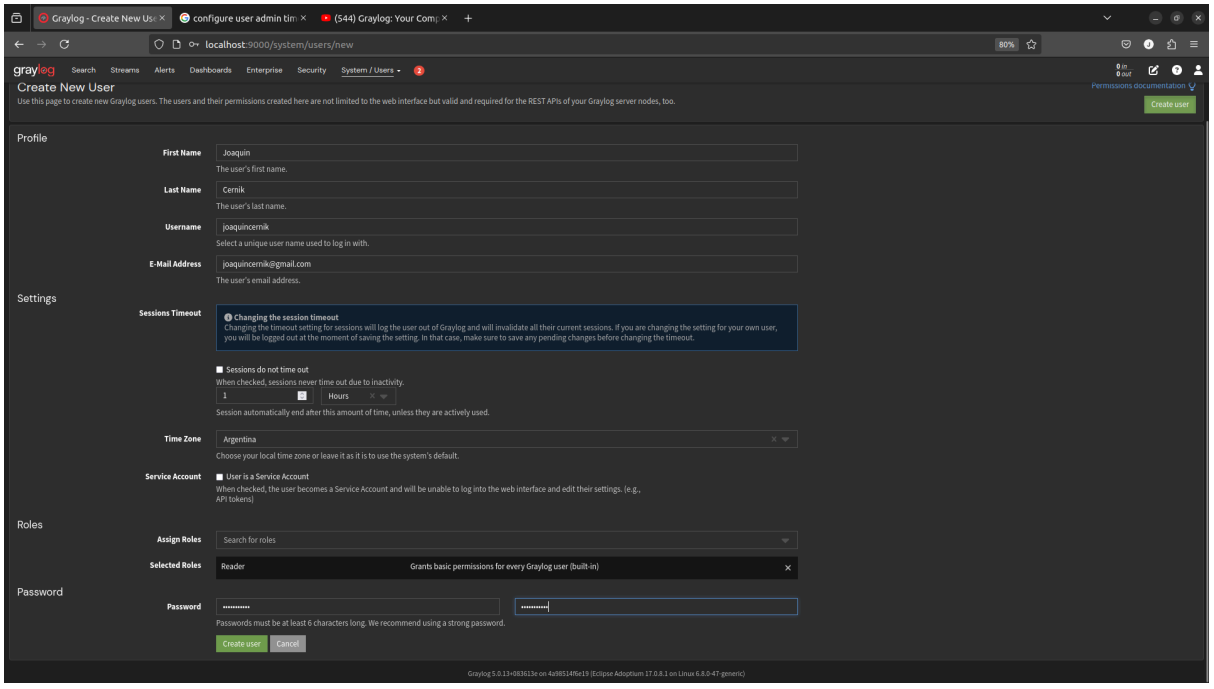
Realizamos *docker-compose down* y luego *docker-compose up -d* para reiniciar y ver los cambios

Creación de cuenta separada para no usar admin:

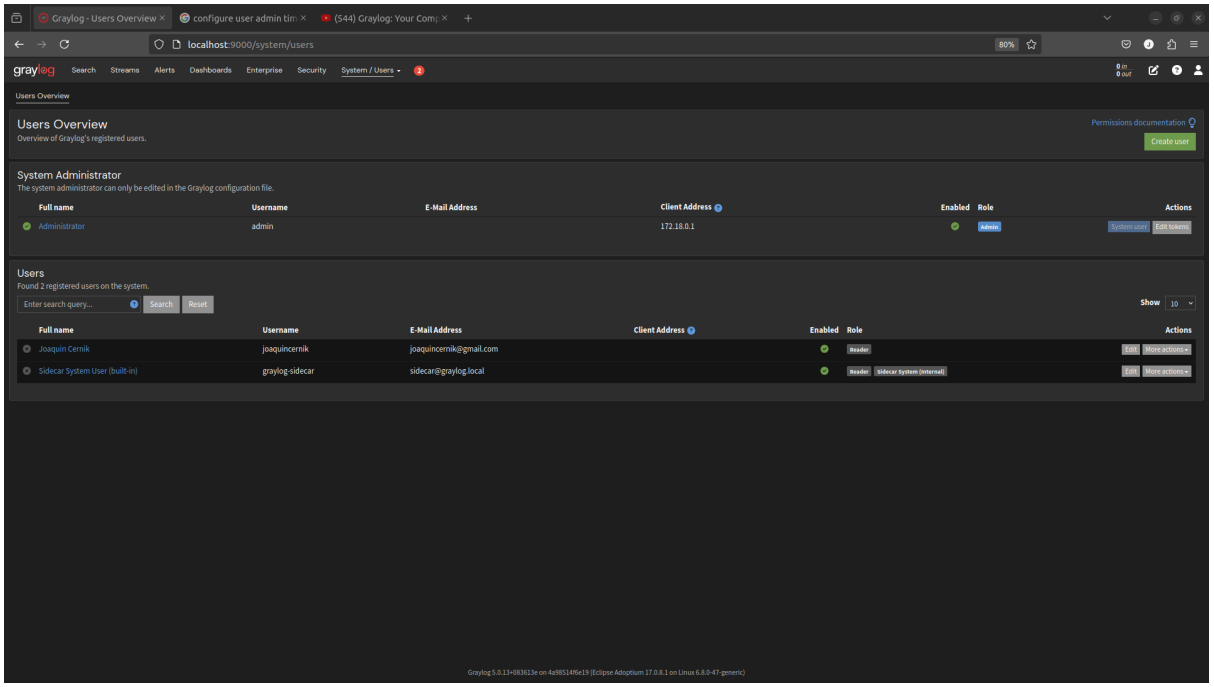
Vamos a la interfaz de graylog y vamos al dropdown System/Users y elegimos Users and terms



Damos click en el botón *Create User* y rellenamos los datos:



Vemos que nuestro usuario ha sido correctamente creado:



2. Gestionar logs de diferentes máquinas virtuales

- **Logs en nuestra maquina local**
  (https://archivedocs.graylog.org/en/latest/pages/sending_data.html)

Primero antes de iniciar con el tratado de las máquinas virtuales, dado que llevan una complejidad superior, creemos necesario chequear si graylog está funcionando correctamente enviando logs desde nuestra máquina local.

Para esto utilizaremos *syslog udp* para enviar los logs, debido a que que es un estándar ampliamente utilizado y es altamente compatible con graylog

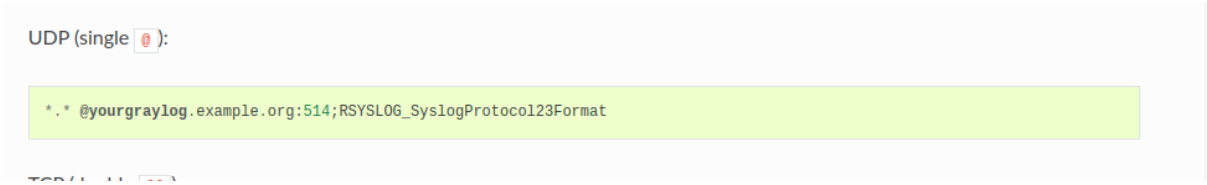
Para que graylog comience a ver nuestros logs desde nuestro ubuntu local debemos primero instalar rsyslog en nuestra terminal:

```
joaquincernik@dell:~/Desktop/pruebaDocker$ sudo apt-get install rsyslog
[sudo] password for joaquincernik:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
rsyslog is already the newest version (8.2112.0-2ubuntu2.2).
rsyslog set to manually installed.
The following packages were automatically installed and are no longer required:
  libc-ares2 libnode72 php8.1-curl php8.1-mbstring php8.1-mysql php8.1-xml
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Luego debemos especificarle en el archivo config de rsyslog a donde queremos enviar los logs accediendo a este mediante el siguiente comando:

```
joaquincernik@dell:~/Desktop/pruebaDocker$ sudo nano /etc/rsyslog.conf
```

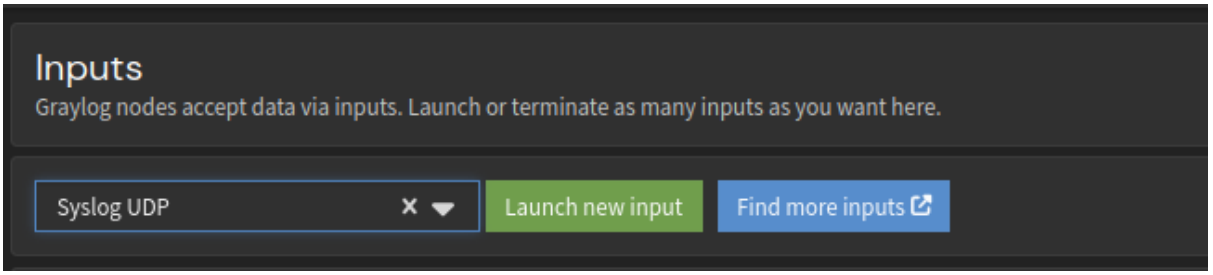
Ingresamos la siguiente línea



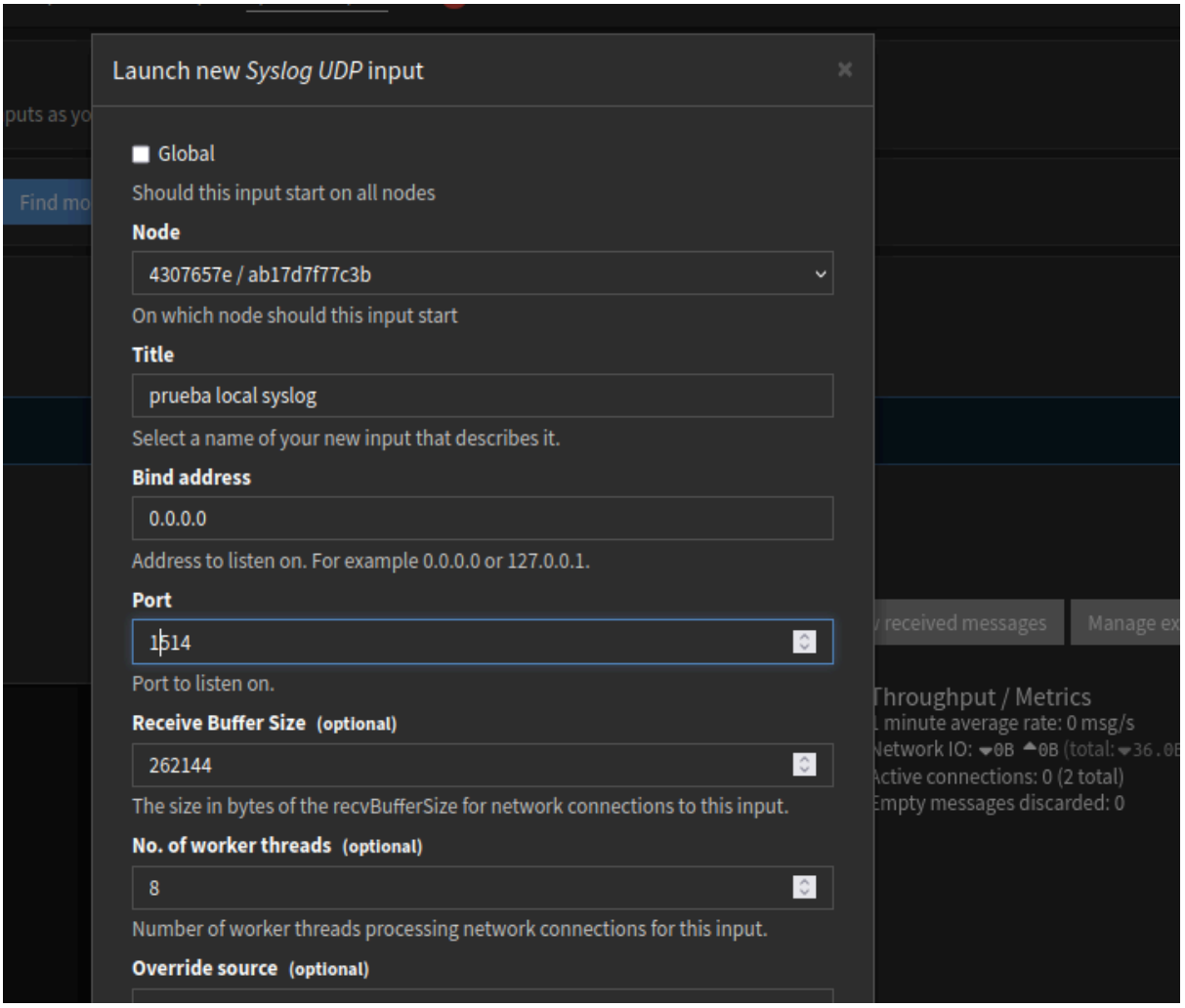
Donde reemplazamos **@yourgraylog.example.org:514** por ***@localhost:1514***

****aclaración**** ***-> Si bien por defecto se recomienda usar el puerto 514 se optó por usar el 1514 debido a simplicidad y no tener que estar modificando el firewall para que se puedan ver los puertos inferiores a 1024***

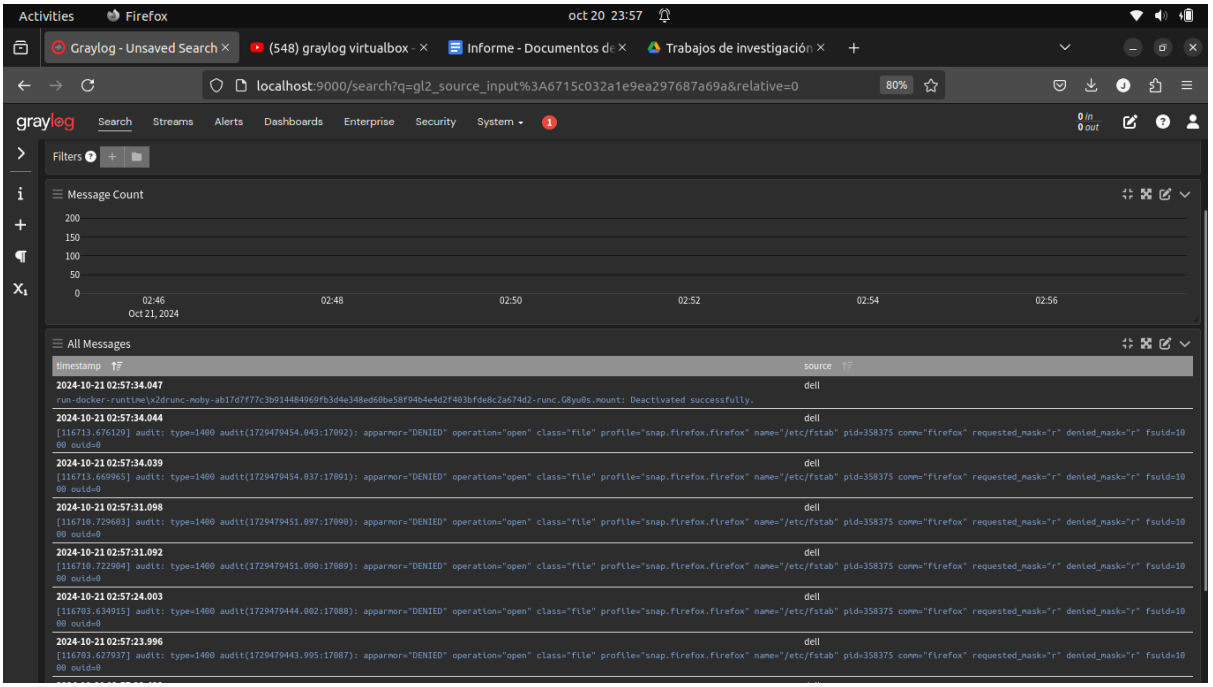
Ahora nos vamos a graylog a *input* y creamos el siguiente:



Dentro seleccionamos lo siguiente:



Con esto ya podemos ver los logs exitosamente 👍



Instalacion de virtual box

Procedemos a instalar virtual box mediante el siguiente link
https://www.virtualbox.org/wiki/Linux_Downloads

Pruebas con mysql local

Antes de pasar a probar la detección de logs de mysql en la máquina virtual de linux, deseamos realizar una prueba local para ver que todo funcione correctamente

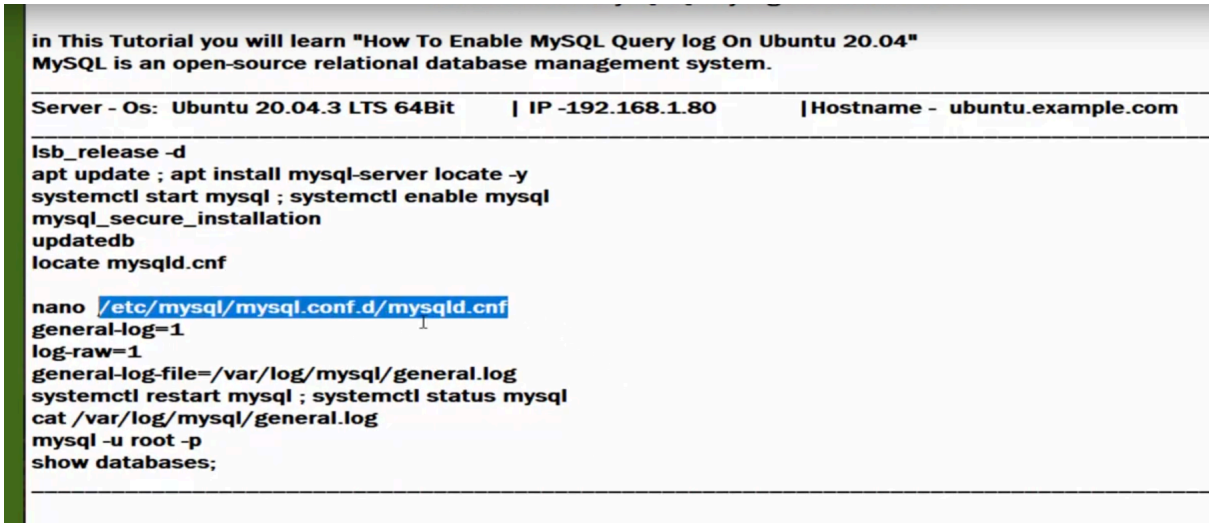
La consigna nos plantea reportar estos logs mediante syslog, por lo tanto debemos resolver esa problemática primero: **Como hacer para que mysql reporte los logs en syslog?**

Aprovechando que tenemos mysql ya descargado en nuestra maquina local, podemos empezar en la investigacion de como resolver esta problemática

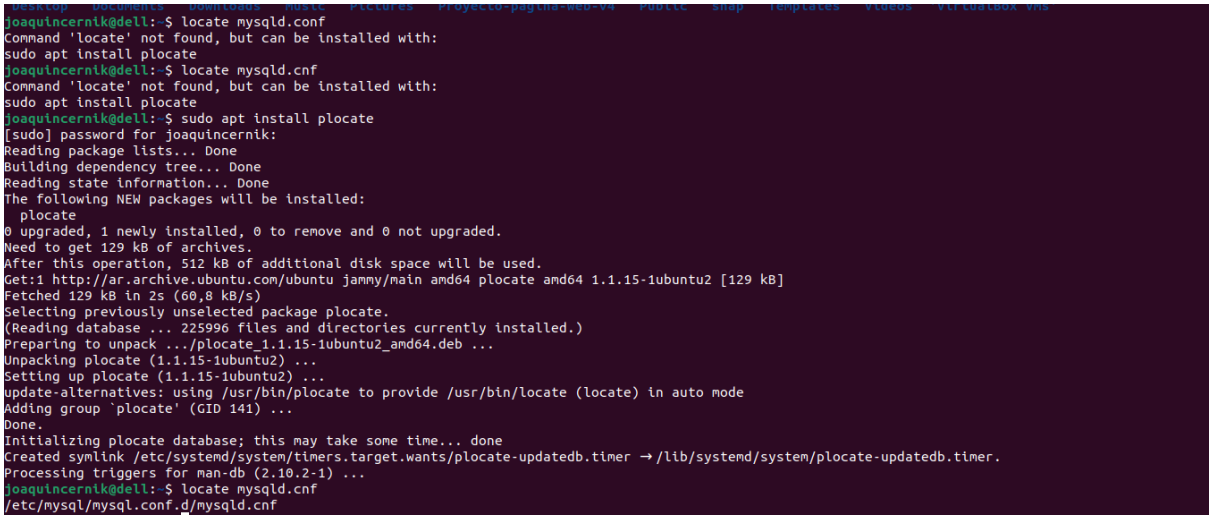
Reportar los logs de mysql

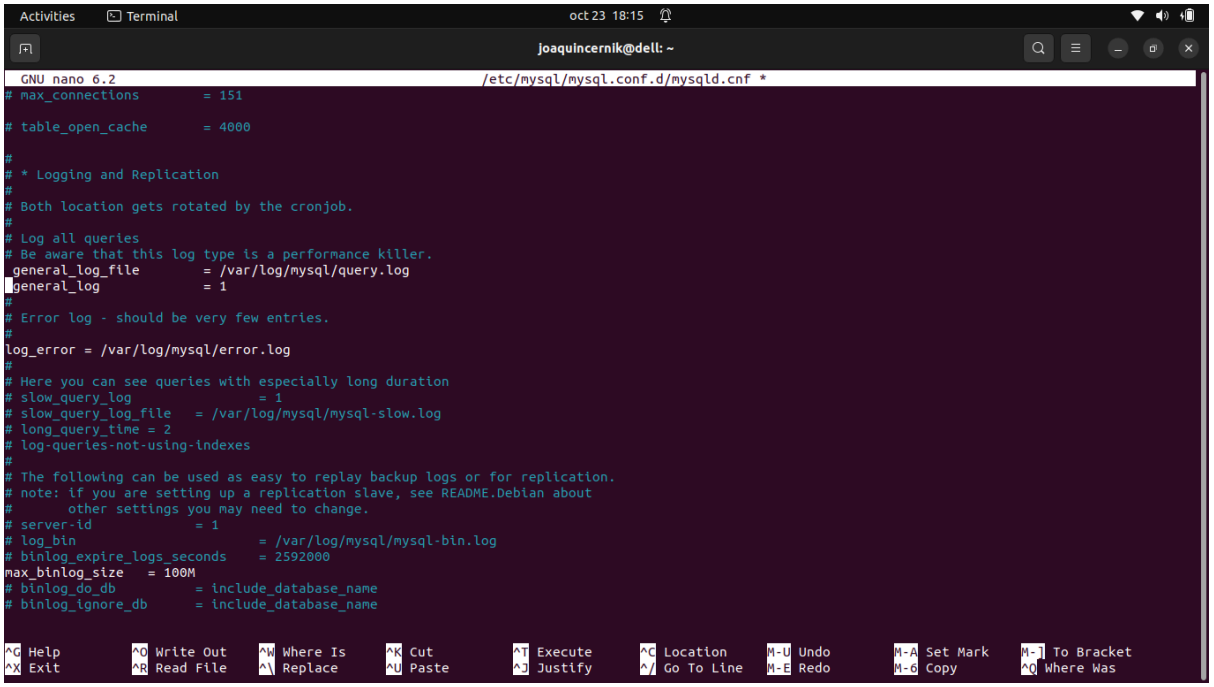
Antes de meternos con syslog, primero debemos saber como hacer para ver los logs de mysql, debido a que no es algo que realizamos comúnmente cuando interactuamos con una base de datos.

Se siguió los siguientes pasos:



Según lo investigado, se debe realizar unas configuraciones antes en el archivo msqld.cnf, por lo tanto tratemos de buscarlo mediante el siguiente comando





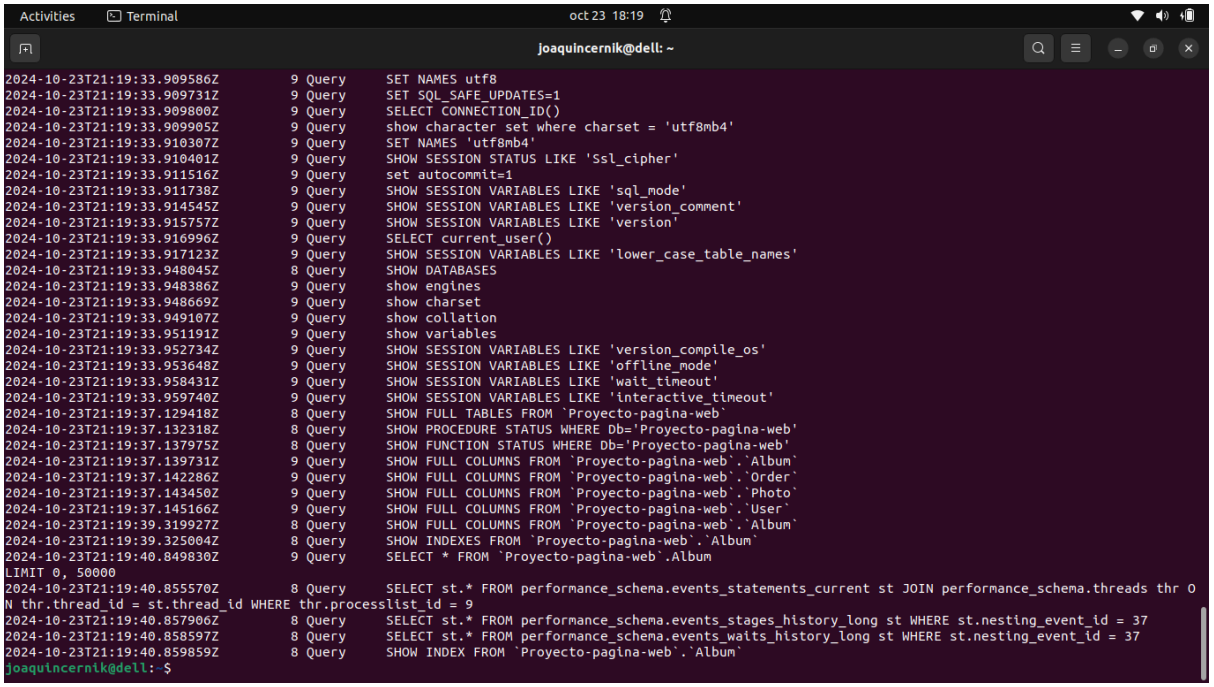
Reiniciamos:

```
joaquincernik@dell:~$ sudo systemctl status  mysql
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2024-10-23 18:17:23 -03; 7s ago
    Process: 29255 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCES
   Main PID: 29264 (mysqld)
     Status: "Server is operational"
      Tasks: 38 (limit: 13984)
     Memory: 411.6M
        CPU: 525ms
     CGroup: /system.slice/mysql.service
             └─29264 /usr/sbin/mysqld

oct 23 18:17:23 dell systemd[1]: Starting MySQL Community Server...
oct 23 18:17:23 dell systemd[1]: Started MySQL Community Server.
```

Vemos el log file antes de que tenga que ser sobrescrito

```
joaquincernik@dell:~$ cat /var/log/mysql/query.log
cat: /var/log/mysql/query.log: Permission denied
joaquincernik@dell:~$ sudo cat /var/log/mysql/query.log
/usr/sbin/mysqld, Version: 8.0.39-0ubuntu0.22.04.1 ((Ubuntu)). started with:
Tcp port: 3306  Unix socket: /var/run/mysqld/mysqld.sock
Time                 Id Command    Argument
2024-10-23T21:17:23.853323Z         0 Execute   CREATE TABLE performance_schema.innodb_redo_log_files(
`FILE_ID` BIGINT NOT NULL COMMENT 'Id of the file.',
`FILE_NAME` VARCHAR(2000) NOT NULL COMMENT 'Path to the file.',
`START_LSN` BIGINT NOT NULL COMMENT 'LSN of the first block in the file.',
`END_LSN` BIGINT NOT NULL COMMENT 'LSN after the last block in the file.',
`SIZE_IN_BYTES` BIGINT NOT NULL COMMENT 'Size of the file (in bytes).',
`IS_FULL` TINYINT NOT NULL COMMENT '1 iff file has no free space inside.',
`CONSUMER_LEVEL` INT NOT NULL COMMENT 'All redo log consumers registered on smaller levels than this v
)engine = 'performance_schema'
```

Después de listar una tabla previamente creada por ejemplo:



Vemos que reporta los logs correctamente

Una vez que sabemos que los logs están siendo correctamente reportados a */var/log/mysql/query.log* podemos ver como hacer para que syslog los “atrape” y envíe a graylog

Lograr que syslog vea los logs de mysql (https://www.freecodecamp.org/news/what-is-syslog-handbook/#heading-redirecting-logs-from-existing-log-files)

Configuramos para que syslog pueda ver el archivo de los logs :

```
module(load="imfile" PollingInterval="10") # Load the imfile module

# For info logs
input(type="imfile"
      File="/var/log/mysql/query.log"
      Tag="logsmysql"
      Severity="info"
      Facility="local0")
```

El parámetro **`facility`** define el origen o la categoría del mensaje de log. En otras palabras, indica qué parte del sistema generó el mensaje. Algunas de las facilities comunes son:

- **`kern`**: Mensajes del núcleo del sistema.
- **`user`**: Mensajes de programas de usuario.
- **`mail`**: Mensajes del sistema de correo.
- **`daemon`**: Mensajes de los demonios del sistema.
- **`auth`**: Mensajes relacionados con la autenticación.
- **`syslog`**: Mensajes del propio sistema de logs.
- **`local0`** **a** **`local7`**: Facilities reservadas para el uso del usuario, permitiendo categorizar mensajes específicos de aplicaciones o servicios personalizados.

Por ejemplo, al establecer `Facility="local0"`, estás indicando que los mensajes que se registren provienen de una fuente de usuario definida (local).

El parámetro **`severity`** indica el nivel de severidad o prioridad del mensaje de log. Esto ayuda a filtrar los mensajes según su importancia. Los niveles de severidad, en orden de menor a mayor gravedad, son:

- **`debug`**: Información detallada para diagnóstico.
- **`info`**: Información general sobre el funcionamiento del sistema.
- **`notice`**: Condiciones normales pero significativas.
- **`warning`**: Advertencias sobre condiciones que no son necesariamente errores, pero que podrían causar problemas.
- **`err`**: Errores que no son fatales.
- **`crit`**: Errores críticos, que requieren atención inmediata.
- **`alert`**: Condiciones que requieren atención urgente.
- **`emerg`**: Mensajes de emergencia, que indican que el sistema es inoperante.

Ahora modifiquemos el archivo donde mysql envia los logs realizando lo siguiente :

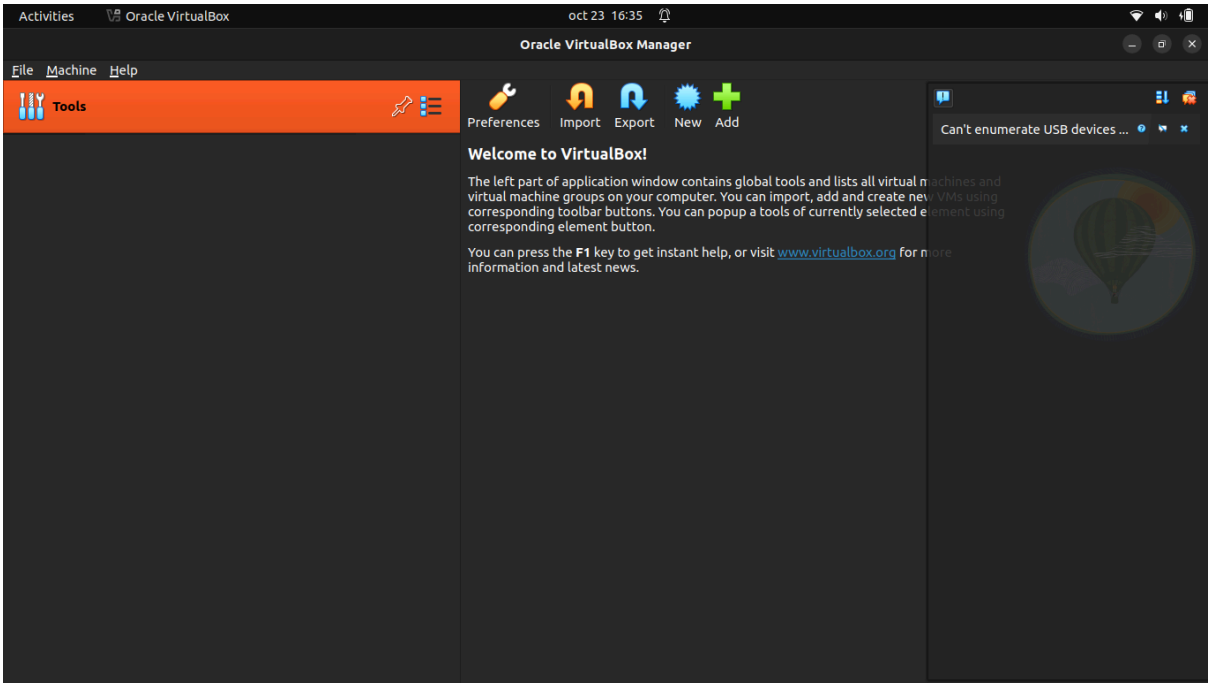
```
joaquincernik@dell:/etc/rsyslog.d$ ls -l /var/log/mysql/query.log
-rw-r----- 1 mysql mysql 14140 oct 23 18:52 /var/log/mysql/query.log
joaquincernik@dell:/etc/rsyslog.d$ sudo chown syslog:mysql /var/log/mysql/query.log
joaquincernik@dell:/etc/rsyslog.d$ sudo chmod 644 /var/log/mysql/query.log
joaquincernik@dell:/etc/rsyslog.d$ sudo systemctl restart rsyslog
```

Ahora ya podemos verificar que graylog esta viendo estos logs correctamente:

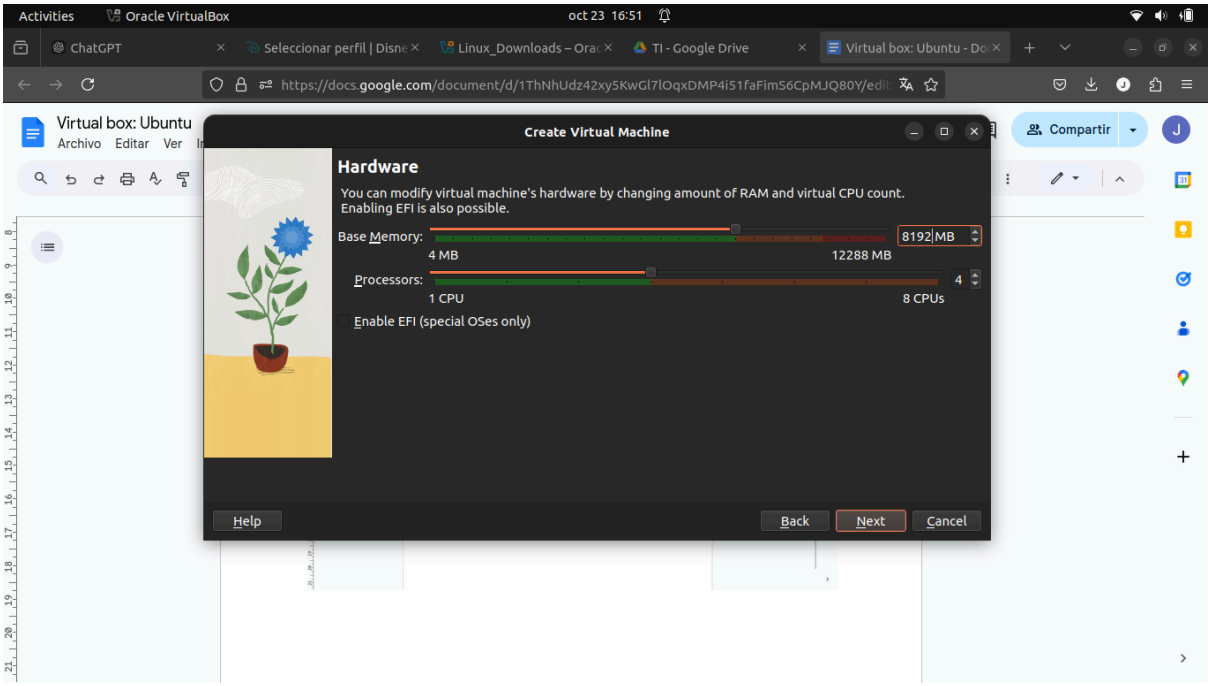
Ya estamos listos para hacer lo mismo en la máquina virtual! 👍

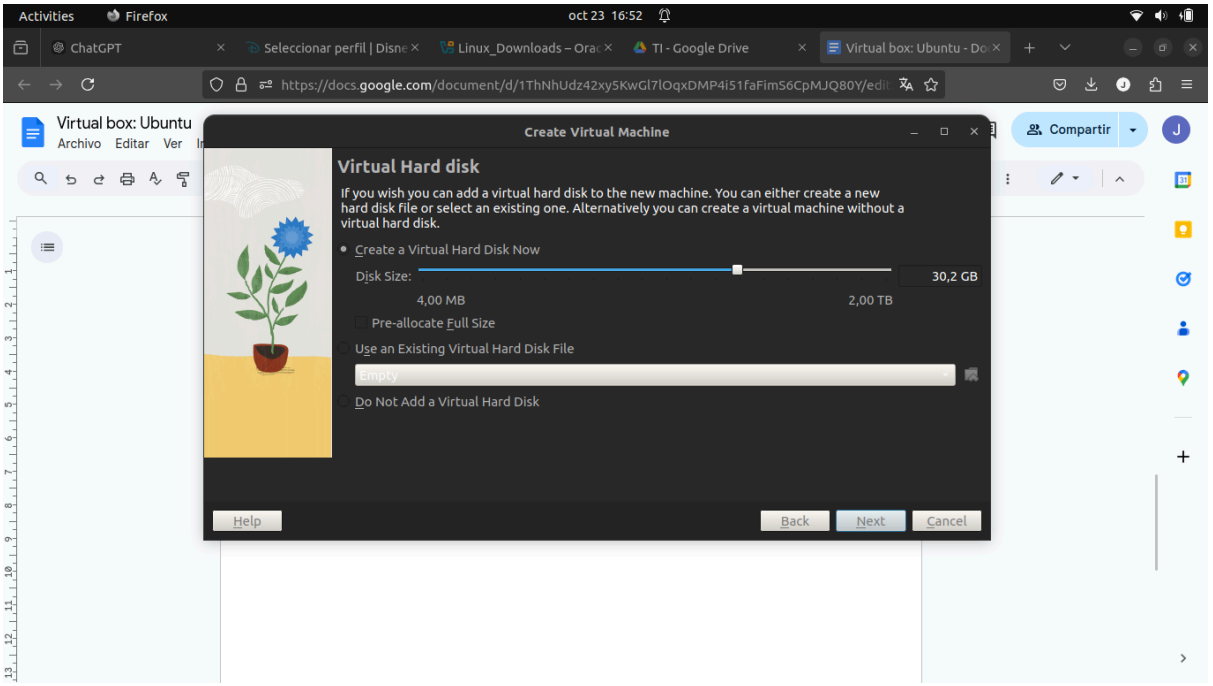
VIRTUAL BOX: UBUNTU LINUX *(entregable B)*

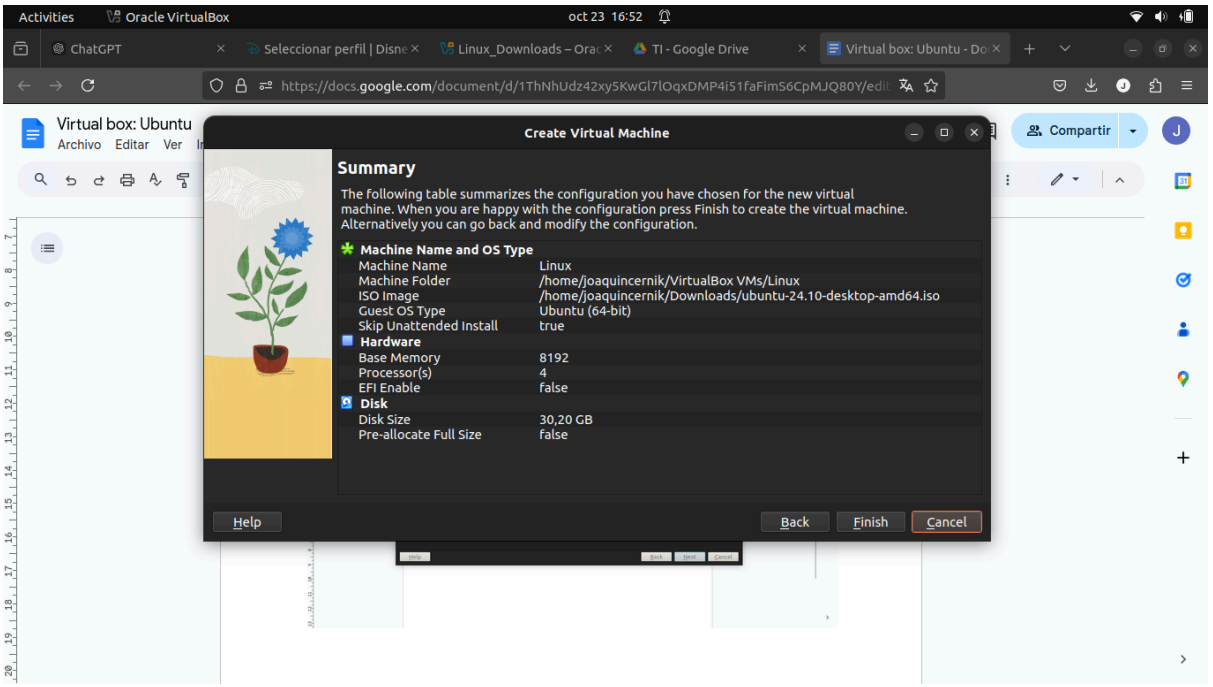
Una vez instalado virtual box estamos en condiciones de crear nuestra maquina virtual ubuntu



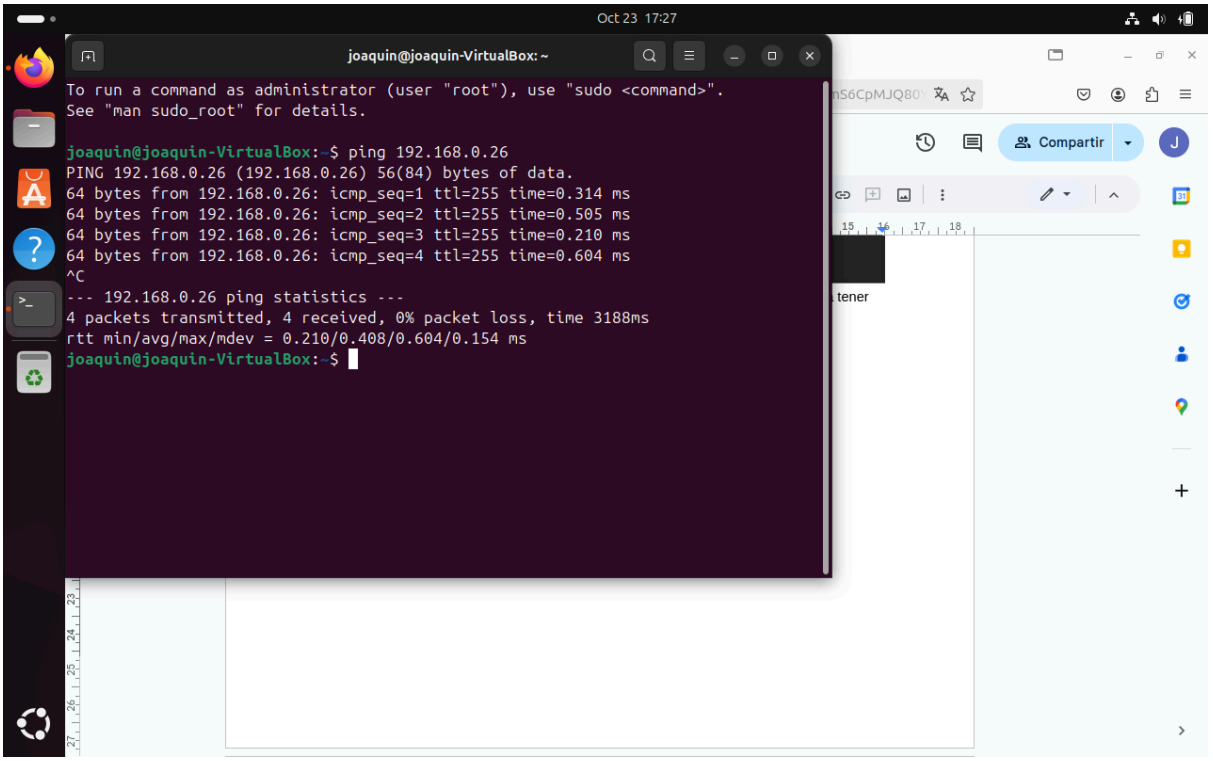
Vamos a New y seleccionamos como imagen la del ubuntu 24.10 en este caso. Para tener esta imagen tenemos que ir a la documentación oficial y descargarla desde ahi







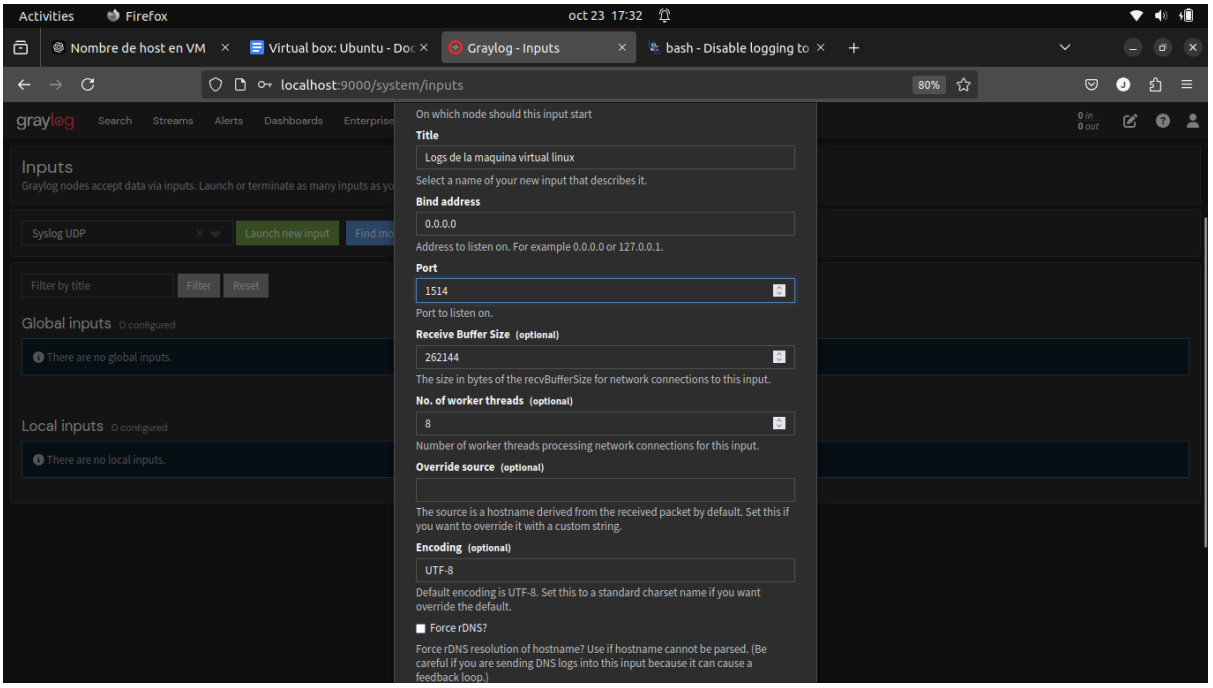
Una vez instalado chequeamos si existe una conexión correcta entre la maquina virtual y la máquina local haciendo *ping <ip de la maquina local>*



Vemos que todo salió correctamente entonces procedemos.

Creación del input en graylog

Tal como lo hicimos en la prueba con la máquina local, ahora tendremos que hacer lo mismo con los logs de la máquina virtual

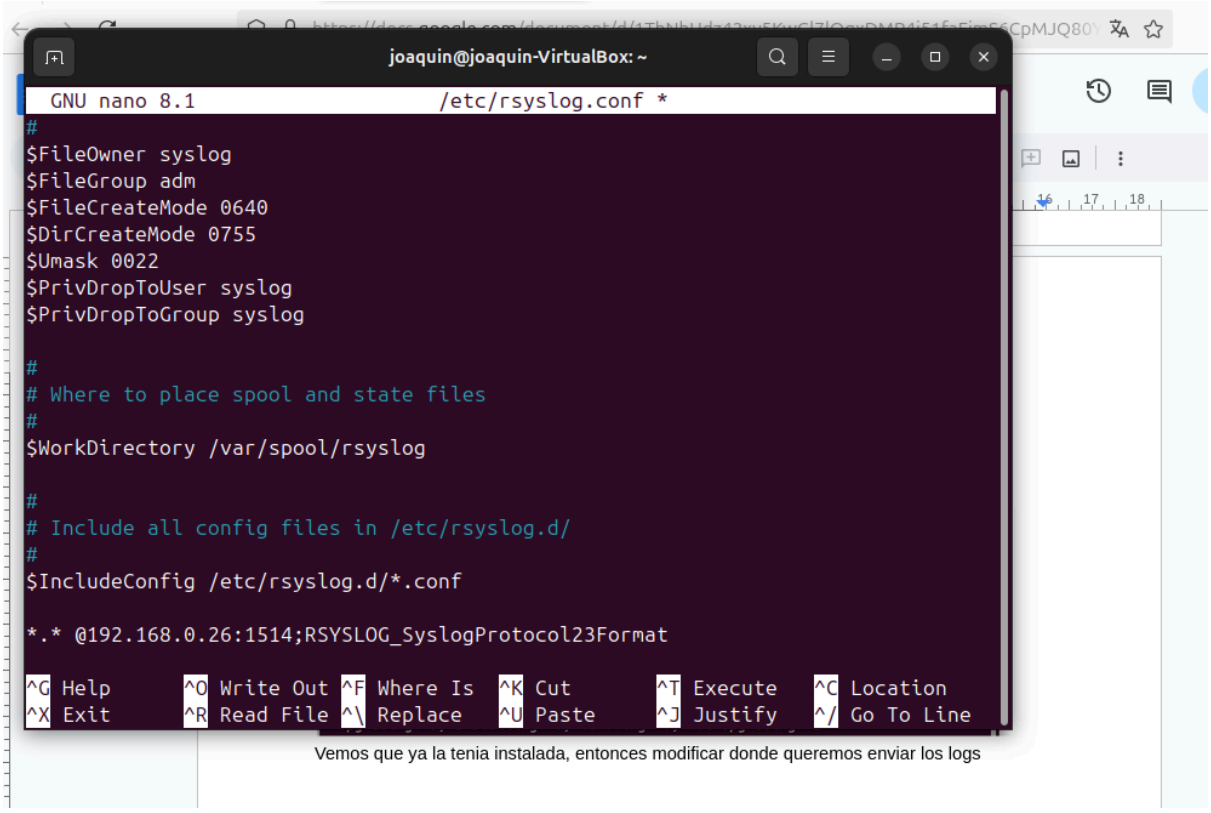


Se elige el puerto 1514 por la misma razón que la que la elegimos en el local

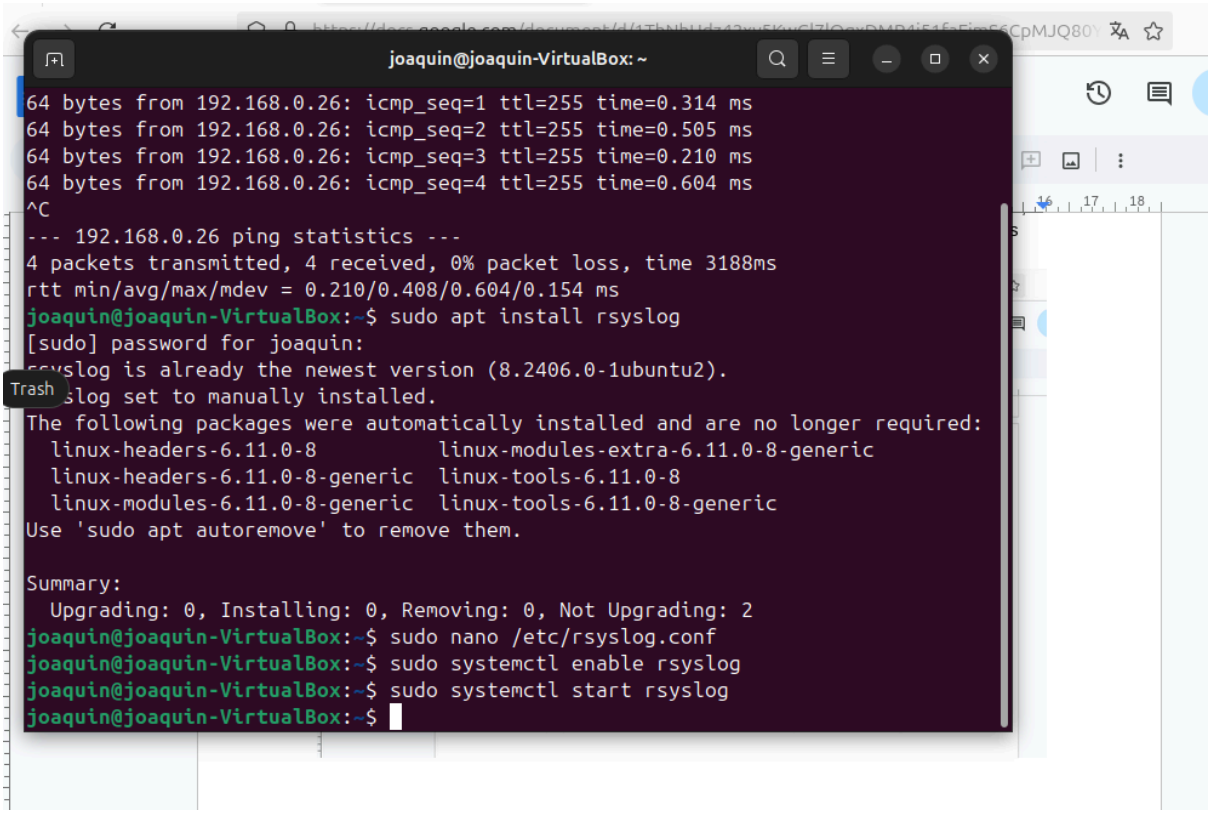
Instalacion de syslog en la máquina virtual

```
joaquin@joaquin-VirtualBox:~$ sudo apt install rsyslog
[sudo] password for joaquin:
rsyslog is already the newest version (8.2406.0-1ubuntu2).
rsyslog set to manually installed.
The following packages were automatically installed and are no longer required:
  linux-headers-6.11.0-8          linux-modules-extra-6.11.0-8-generic
  linux-headers-6.11.0-8-generic  linux-tools-6.11.0-8
  linux-modules-6.11.0-8-generic  linux-tools-6.11.0-8-generic
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 2
```

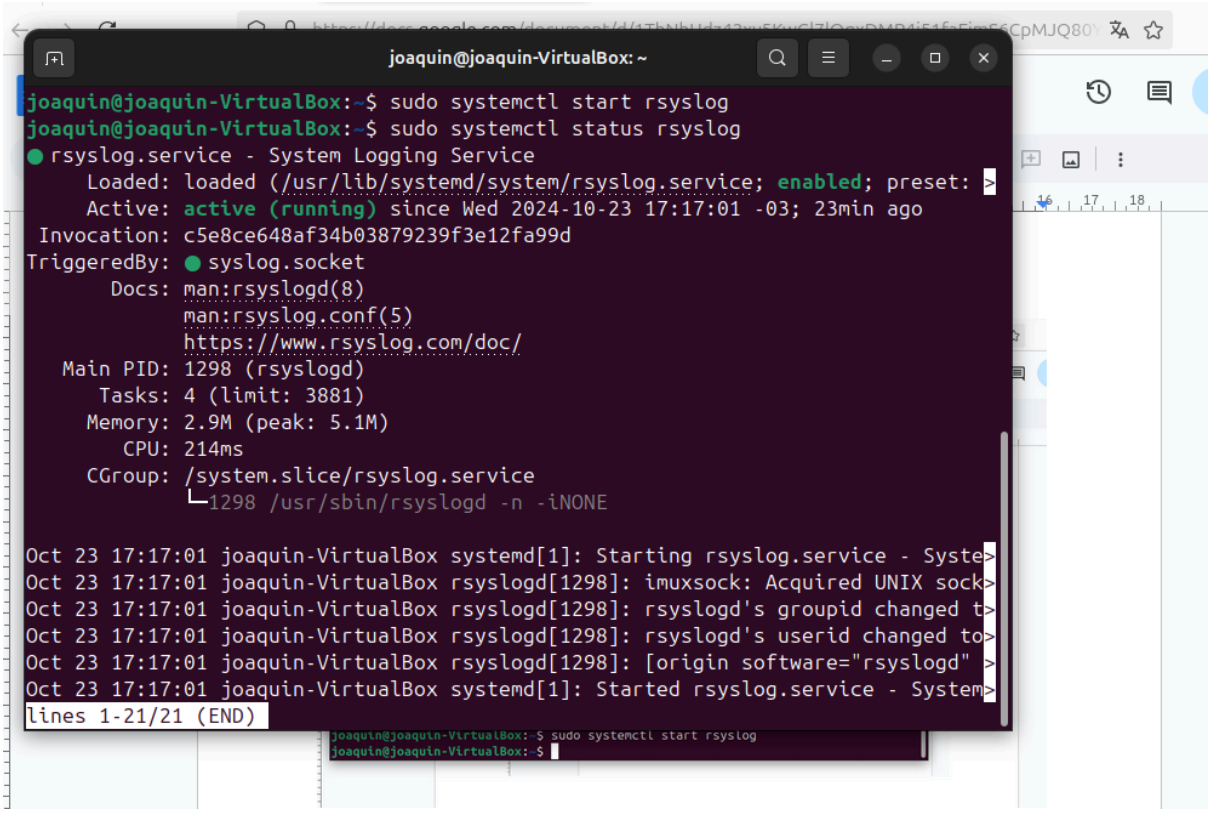
Vemos que ya la tenia instalada, entonces modificar donde queremos enviar los logs. Se inserta lo siguiente en el rsyslog.conf, donde a diferencia de cuando configuramos el local, debemos especificar la ip de nuestra maquina donde graylog esta escuchando, la cual es 192.168.0.26



Lo activamos :

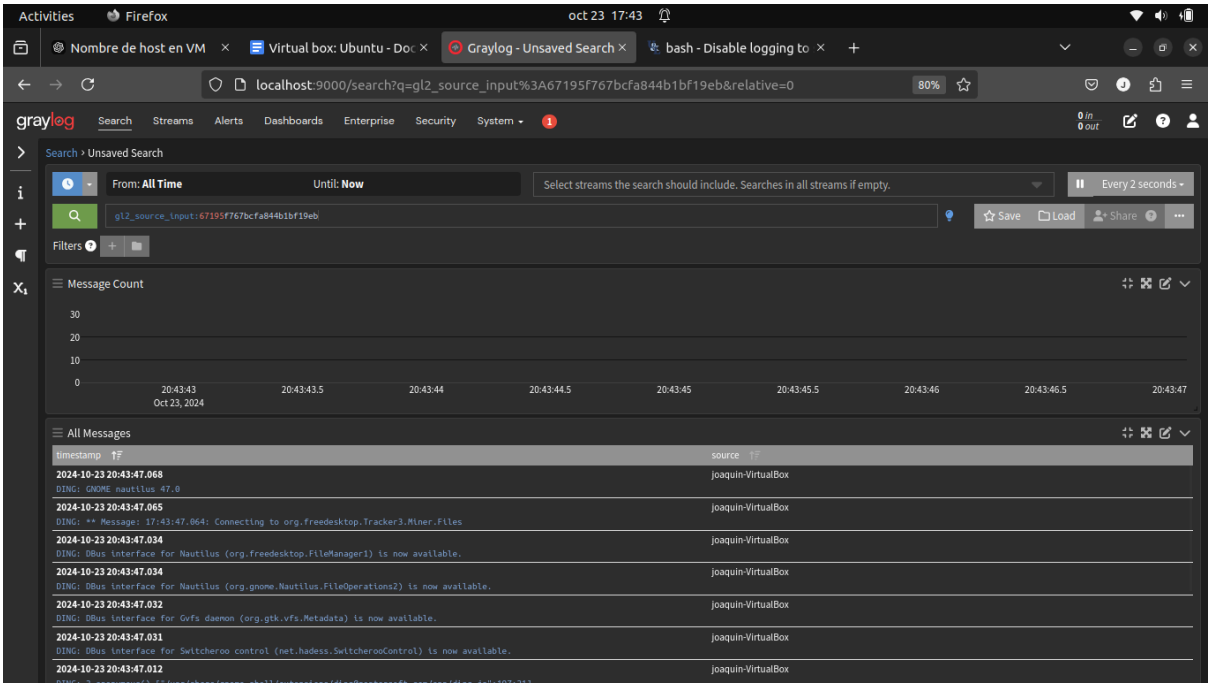


Y chequeamos si está corriendo:



Vemos que está corriendo correctamente

Ahora vamos a la interfaz de graylog y podemos ver cómo están funcionando correctamente el envío de logs :

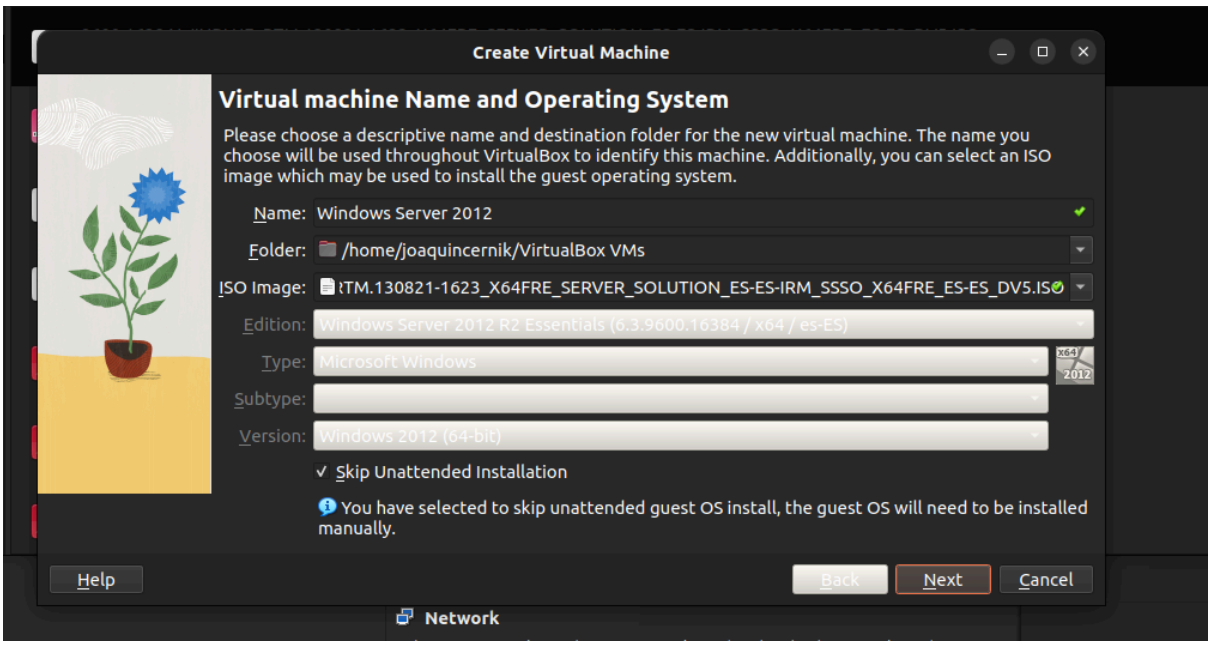


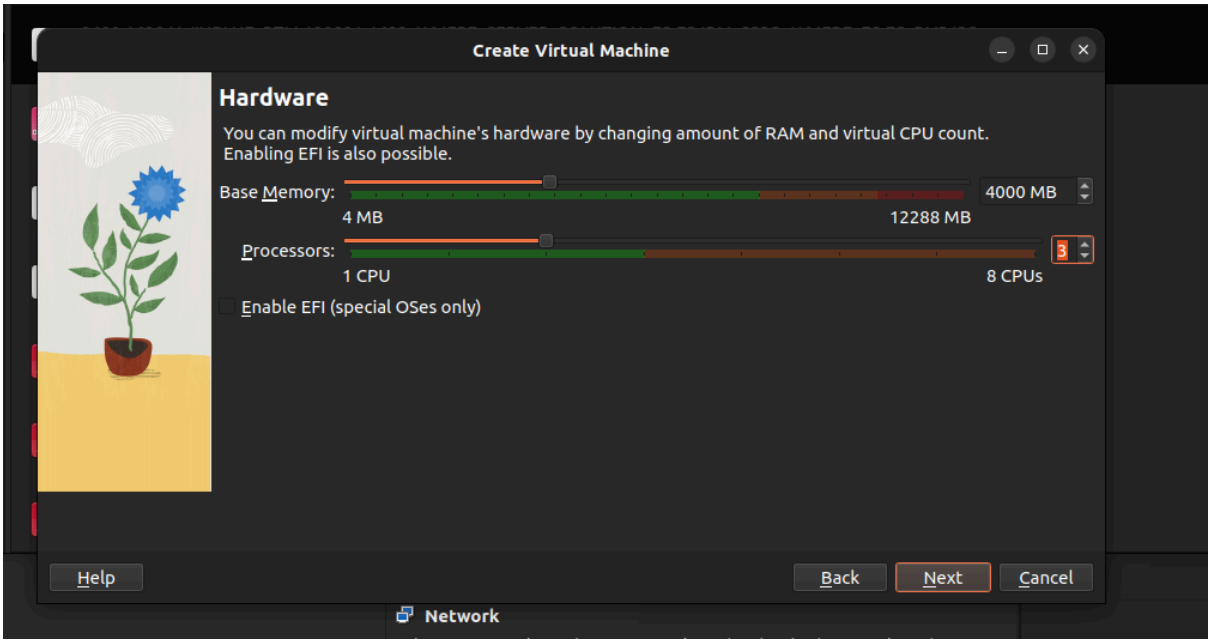
Un detalle importante a considerar fue que en medio de la configuración de syslog fue necesario cambiar los ajustes de la máquina virtual de NAT a bridge, ya que no llegaban los logs, y gracias a este cambio ahora llegan correctamente

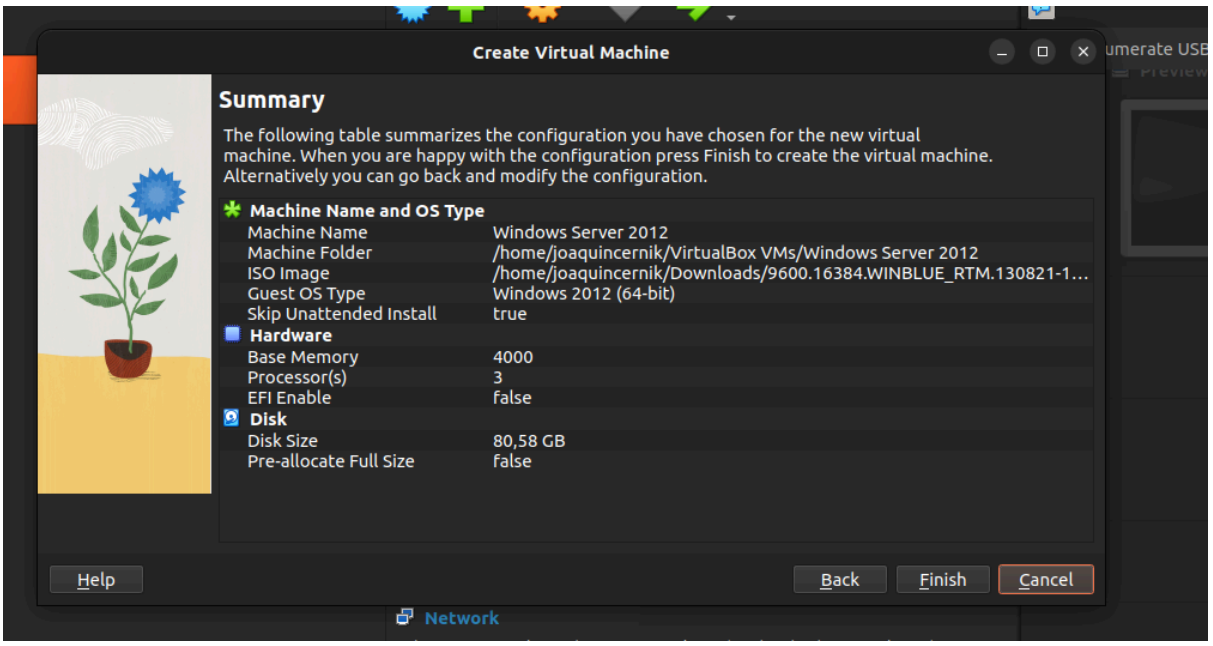
\Virtual box: Windows Server 2012 *(entregable C)*

Windows Server 2012 en Virtual Box:

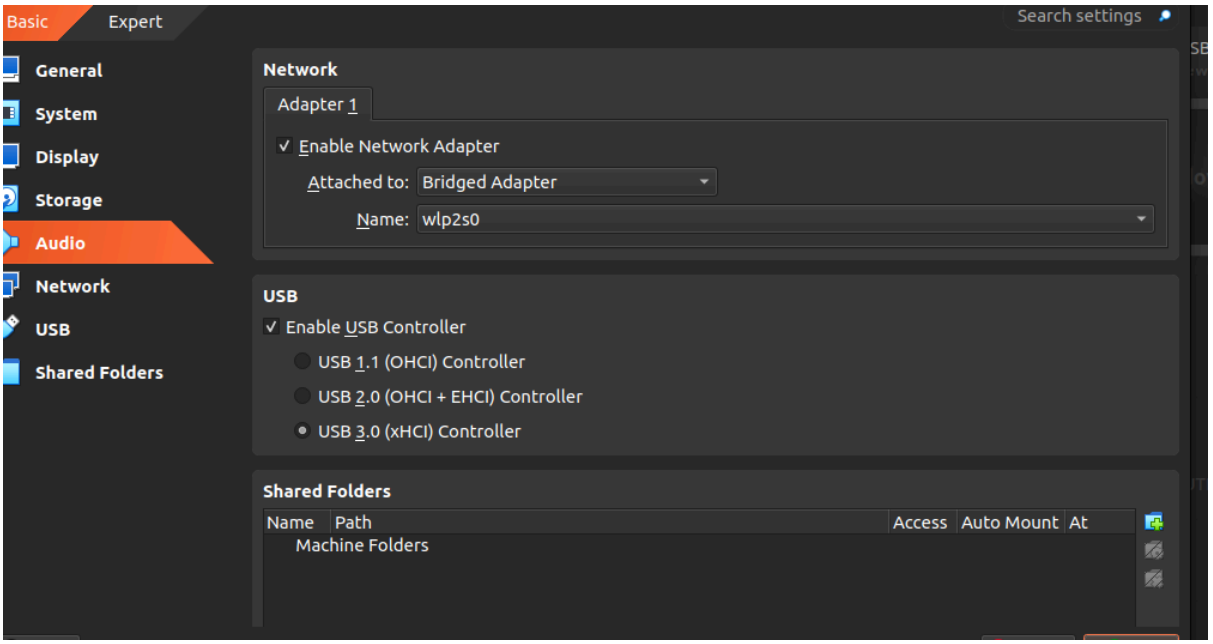
Descargamos la imagen ISO en
https://www.microsoft.com/es-es/evalcenter/download-windows-server-2012-r2-essentials


Create Virtual Machine
Virtual machine Name and Operating System
Please choose a descriptive name and destination folder for the new virtual machine. The name you choose will be used throughout VirtualBox to identify this machine. Additionally, you can select an ISO image which may be used to install the guest operating system.
Name: Windows Server 2012
Folder: /home/joaquincernik/VirtualBox VMs
ISO Image: tTM.130821-1623_X64FRE_SERVER_SOLUTION_ES-ES-IRM_SSSO_X64FRE_ES-ES_DV5.ISO
Edition: Windows Server 2012 R2 Essentials (6.3.9600.16384 / x64 / es-ES)
Type: Microsoft Windows
Subtype:
Version: Windows 2012 (64-bit)
Skip Unattended Installation
You have selected to skip unattended guest OS install, the guest OS will need to be installed manually.
Help
Back
Next
Cancel
Network


Create Virtual Machine
Hardware
You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count.
Enabling EFI is also possible.
Base Memory:
4000 MB
4 MB
12288 MB
Processors:
3
1 CPU
8 CPUs
Enable EFI (special OSes only)
Help
Back
Next
Cancel
Network



Conectamos a la red interna:



Se sigue la siguiente guia https://www.youtube.com/watch?v=i-JOzmZC2xA

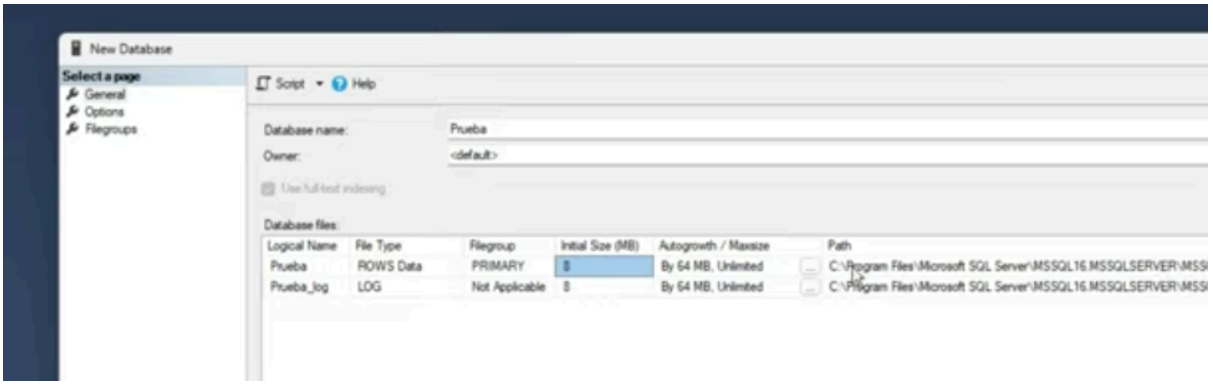
Una vez instalado procedemos a instalar SQL Server siguiendo los pasos de esta guia https://www.youtube.com/watch?v=teqc5RbRAu4

Logs en SQL server:

Ahora que pudimos instalar correctamente el sql server en nuestro servidor de la máquina virtual procedemos a investigar como funciona sql server.

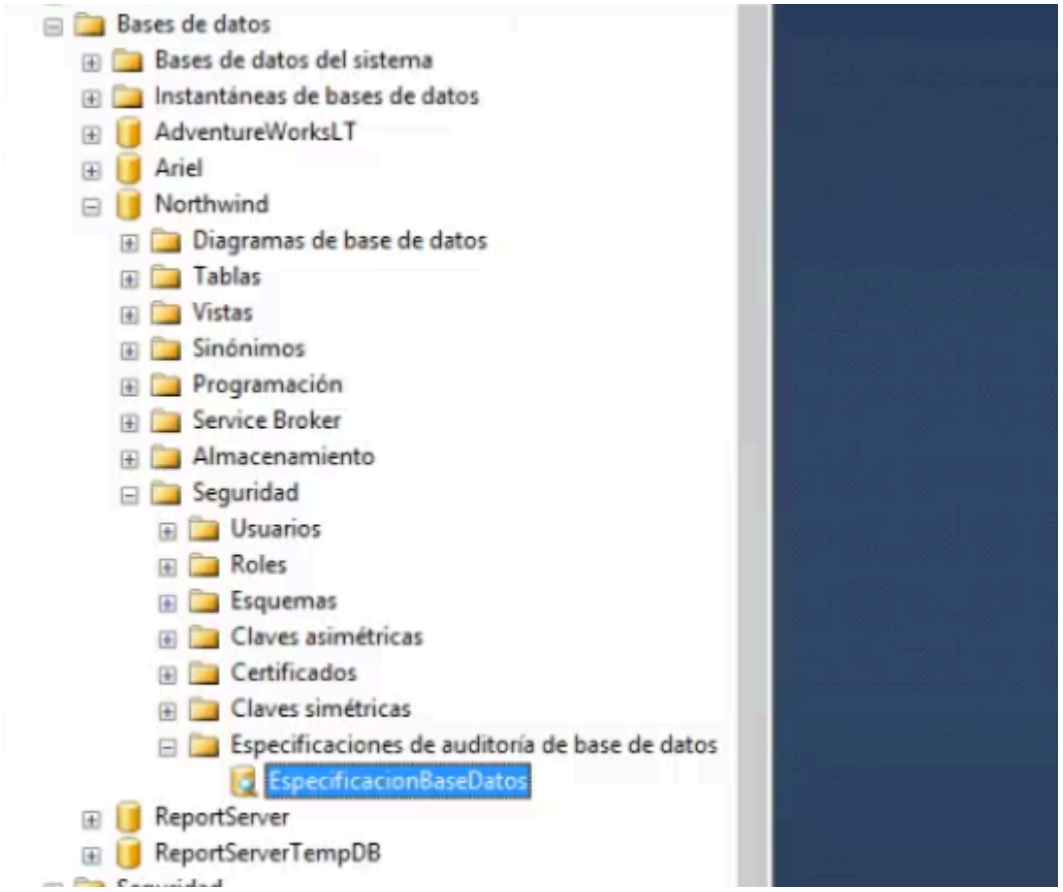
Debido a que funciona de manera bastante diferente a lo que hemos estado acosumbrados fue necesario ver una serie de videos de esta lista:

Al crear una base de datos vemos que junto con ella se crea el archivo de logs:

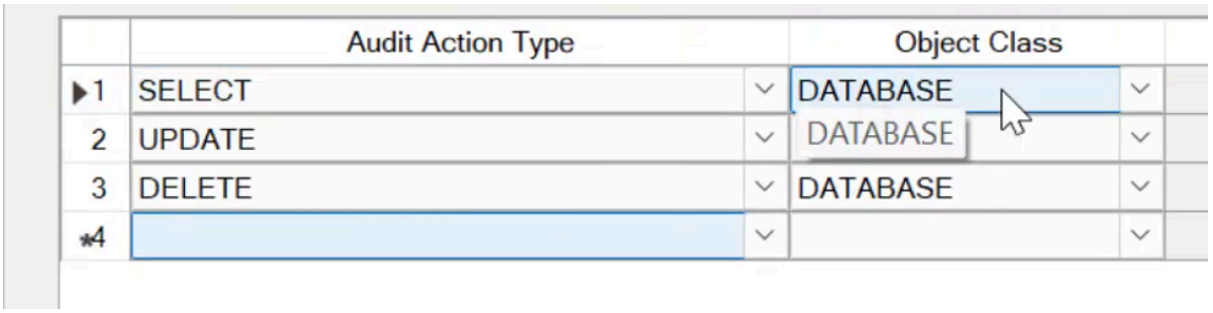


Para esto nos guiamos con el siguiente video
https://www.youtube.com/watch?v=BTzhPkK9Ego

Una vez creada la auditoria ya tendríamos lo siguiente:



Declaramos que tipo de especificaciones seguirá la auditoria:



Es importante aclarar que en el parte de nueva auditoría, es fundamental seleccionar la opción de registros de aplicación debido a que esta será la que le estará informando y enviando los logs a NX Log

NX Log:

NX Log es altamente usada para reportar los logs en WIndows a Graylog, debido a su alta compatibilidad
syso
Para esto se utilizo la siguiente guia:
https://www.youtube.com/watch?v=a3LbQow7i4Q&t=302s

Para transferir los archivos a la virtual machine se realizo el uso de las carpetas compartidas
Se descargo sysmon aqui: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon
Se utilizo el siguiente archivo de configuracion:
https://github.com/lawrencesystems/graylog/blob/master/nxlog.conf

```
58
59 # Sends Eevent in GELF format to Graylog
60  <Output out>
61      Module      om_udp
62      Host        192.168.0.26
63      Port        12201
64      OutputType  GELF
65  </Output>
```

Le agregamos la ip de la máquina local

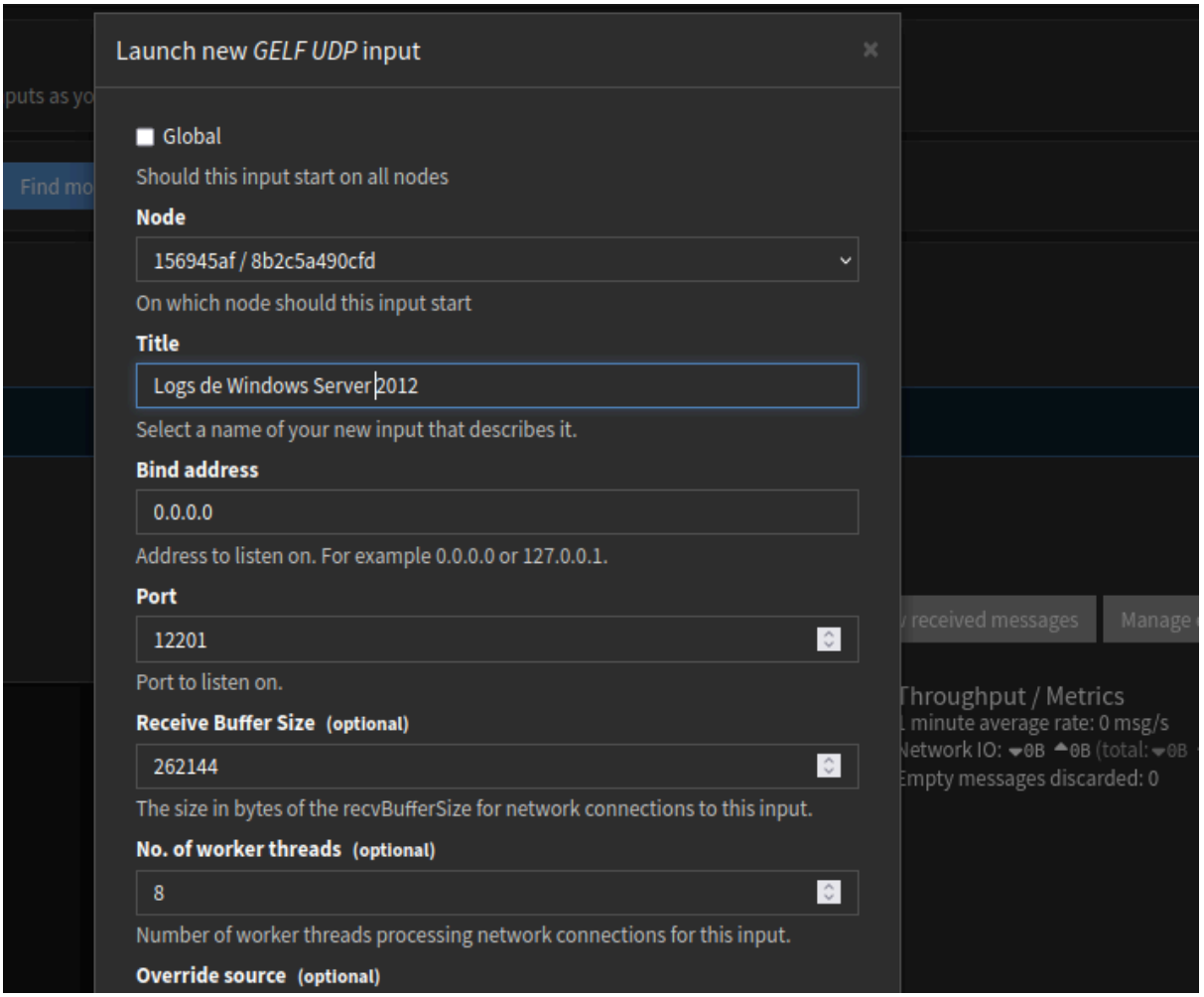
Capturar los logs que vienen de NX Log:

Una vez instalado vamos a nuestra interfaz de graylog y agregamos un input.
Antes de esto tenemos que saber que lo capturaremos mediante GELF, por lo tanto chequeamos previamente, cuál es el puerto que tenemos especificado en el docker-compose para este formato:

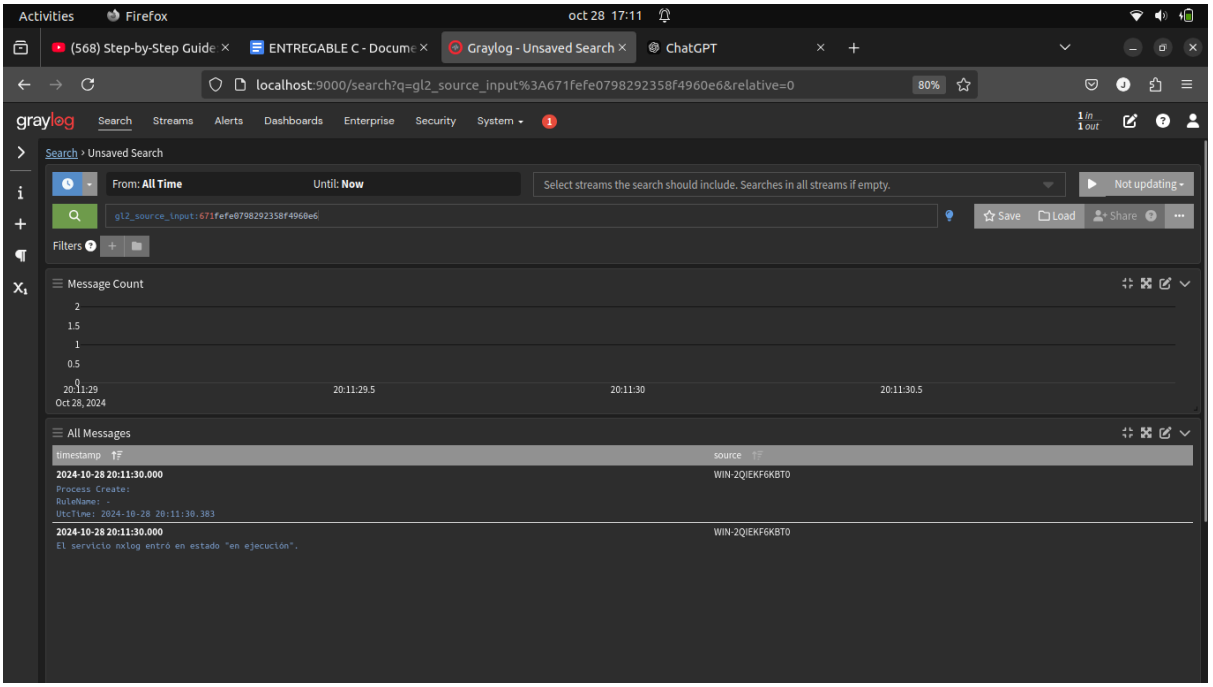
```
ports:
  # Graylog web interface and REST API
  - 9000:9000
  # Syslog TCP
  - 1514:1514
  # Syslog UDP
  - 1514:1514/udp
  # GELF TCP
  - 12201:12201
  # GELF UDP
  - 12201:12201/udp
  # SMTP Email
  - 587:587
  # SMTP gmail
  - 465:465
```

Usaremos el 12201

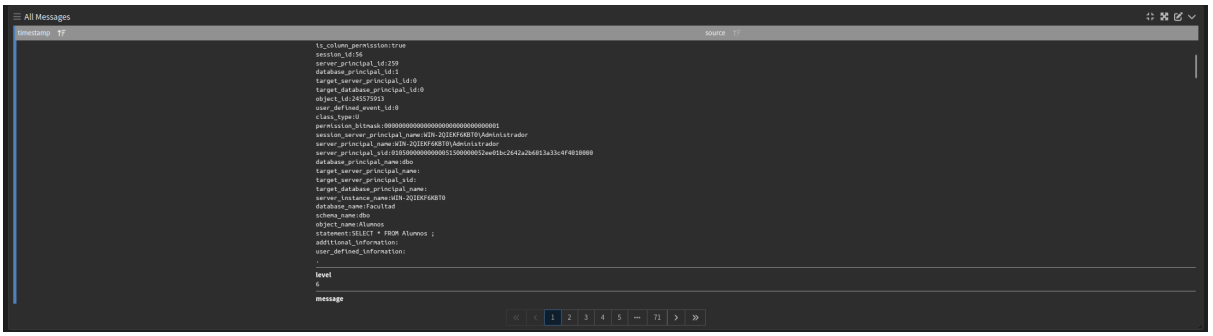
Creamos el siguiente input:



Vemos que efectivamente estan llegando los logs de Windows Server!



Y ya que nos aseguramos de que hayamos elegido la opción de registros de aplicación en la nueva auditoría probamos haciendo un select * de una tabla llamada ALumnos:



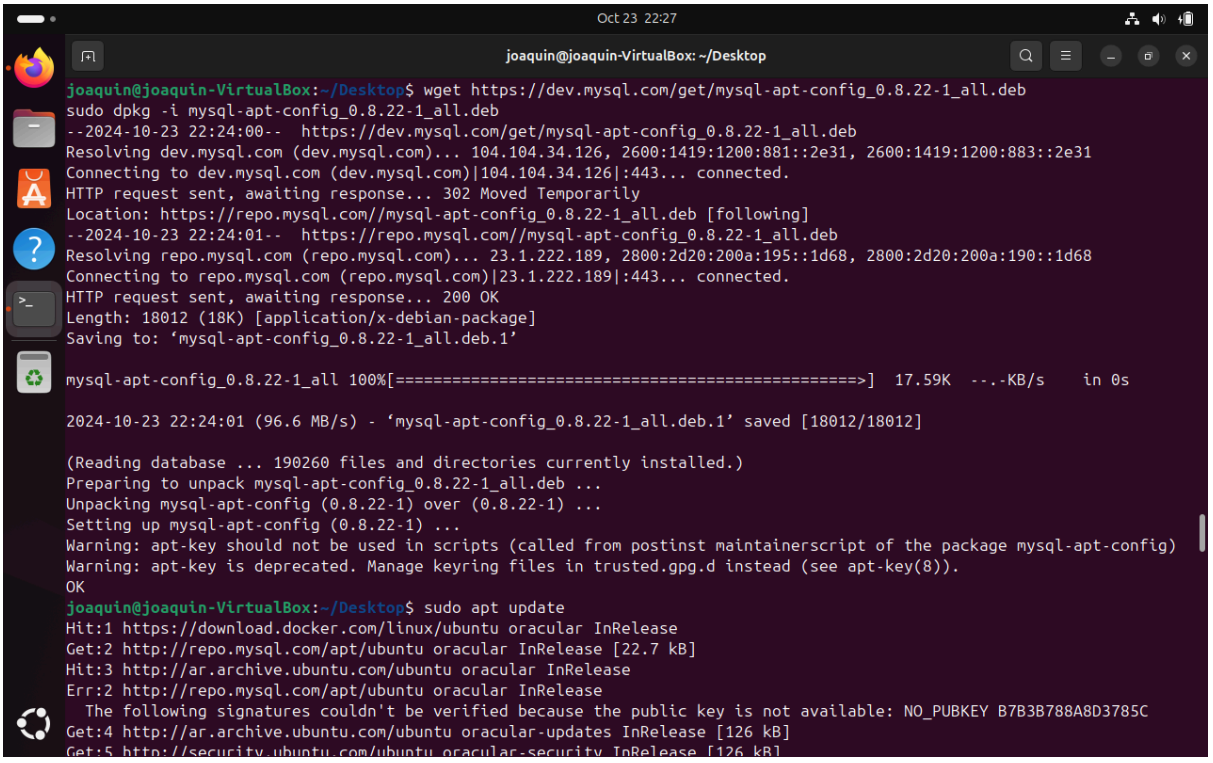
Vemos en el campo statement que estos logs llegan correctamente 👍

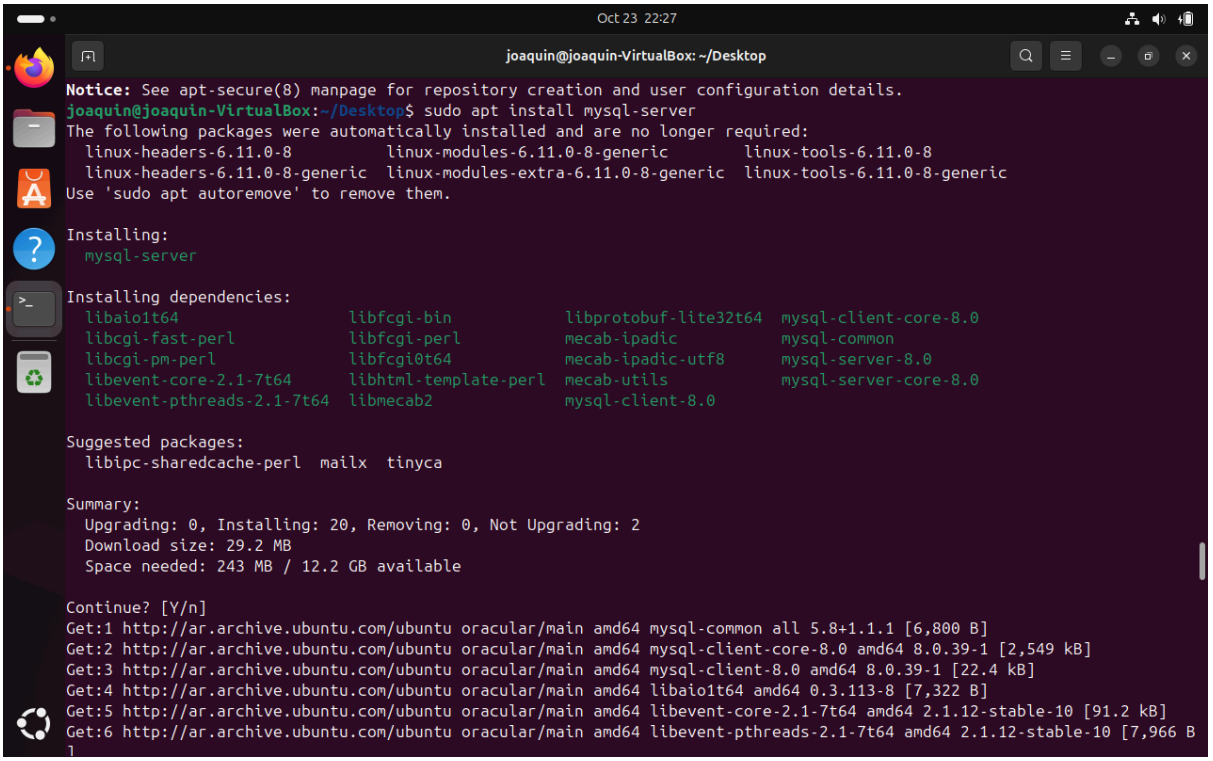
VIRTUAL BOX: MySQL en UBUNTU *(entregable D)*

Una vez realizada la prueba en la máquina local realizamos el mismo procedimiento en esta máquina virtual:

Configuración de MySQL:

Se opto por instalar mysql-server en la máquina virtual en lugar de contenedores por complicidad para la comunicación entre los contenedores de docker y graylog

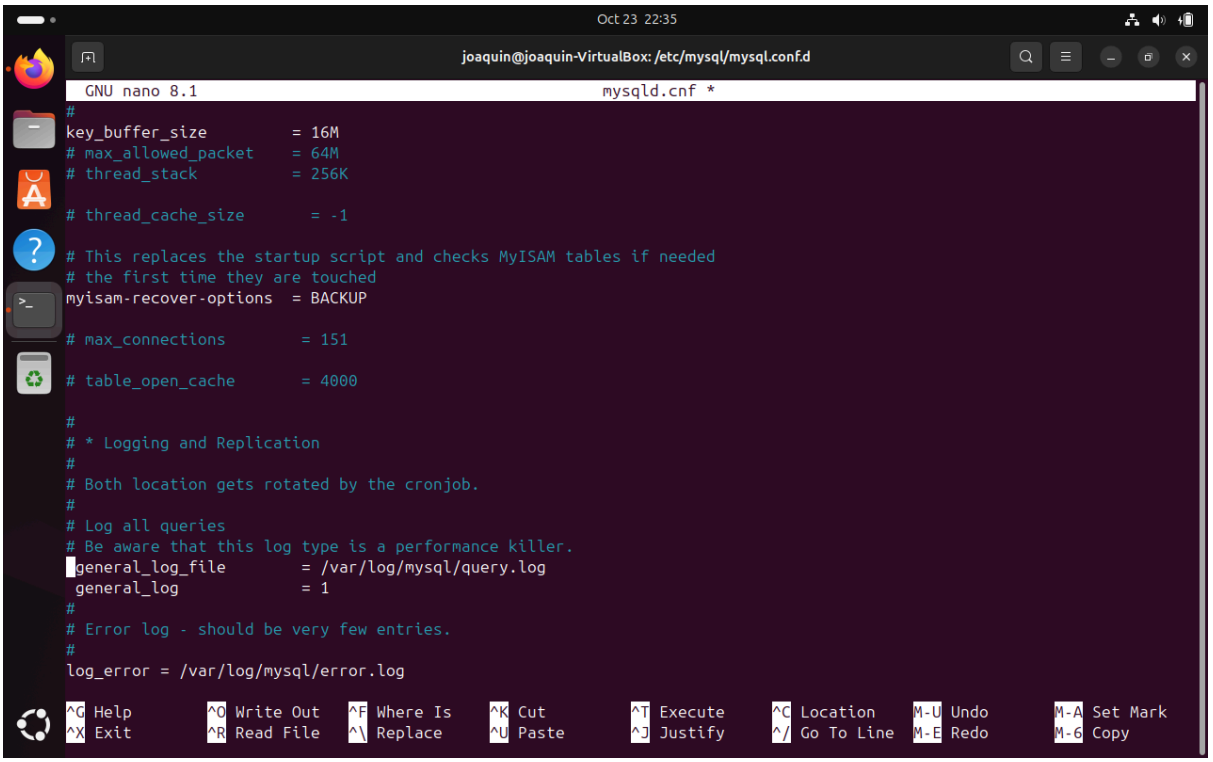




Vemos que ya funciona :

```
All done!
joaquin@joaquin-VirtualBox:~/Desktop$ sudo systemctl start mysql
sudo systemctl enable mysql
Synchronizing state of mysql.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable mysql
joaquin@joaquin-VirtualBox:~/Desktop$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.39-1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

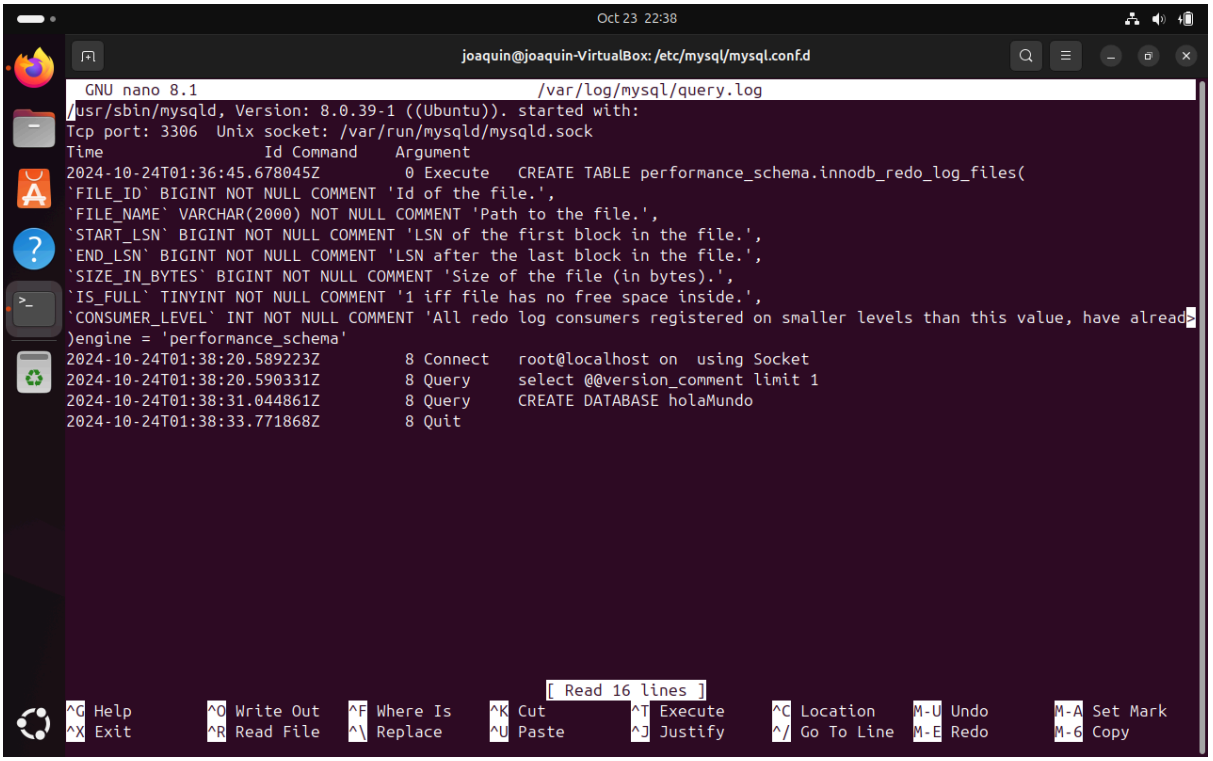
Descomentamos las mismas cosas en el archivo mysqld.conf



Le damos permisos especiales a este archivo

```
joaquin@joaquin-VirtualBox:/etc/mysql/mysql.conf.d$ sudo chmod 644 /var/log/mysql/query.log
joaquin@joaquin-VirtualBox:/etc/mysql/mysql.conf.d$
```

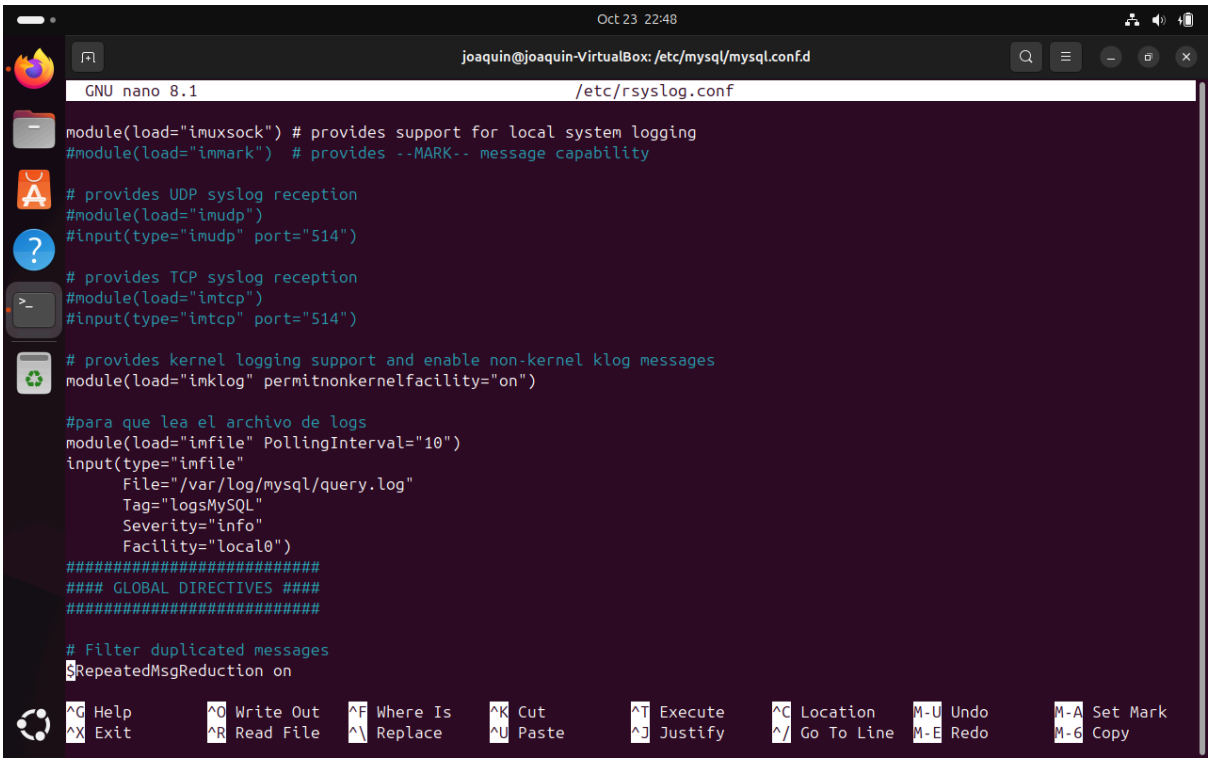
Nos aseguramos que esté reportando los logs correctamente :



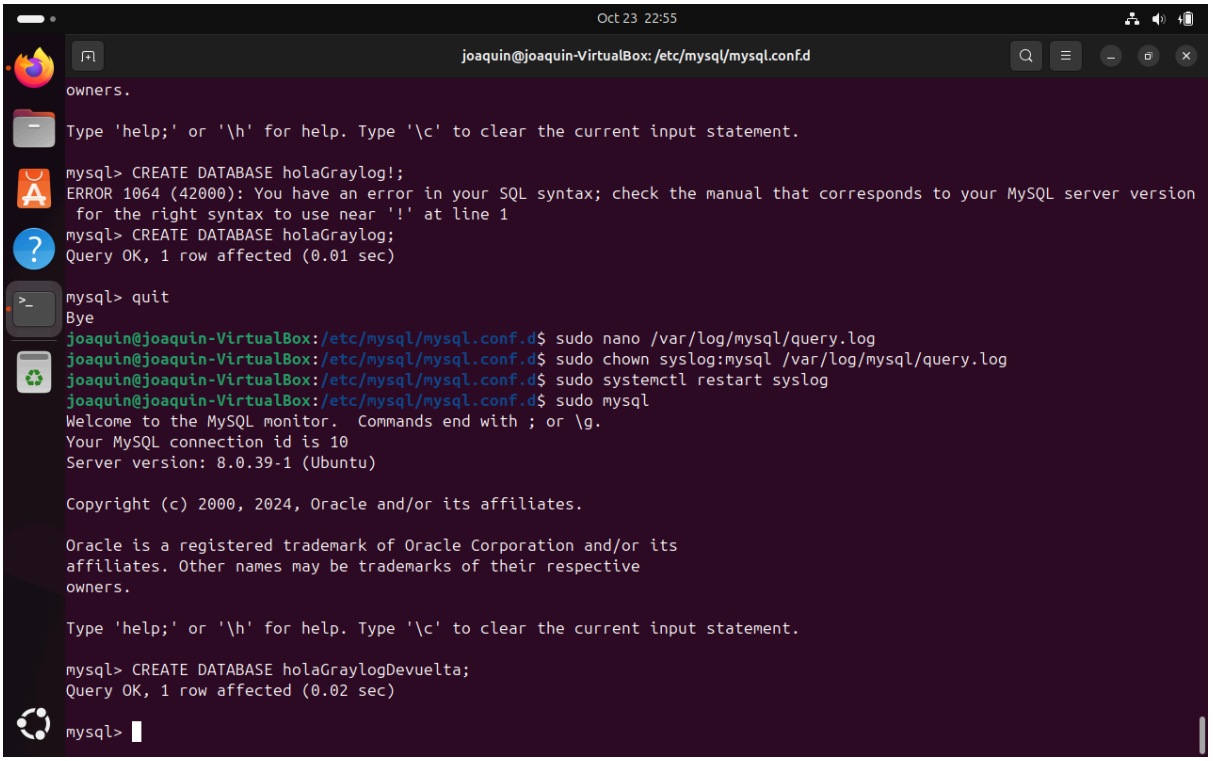
Todo OK asi que continuamos 👍

Configuración de syslog:

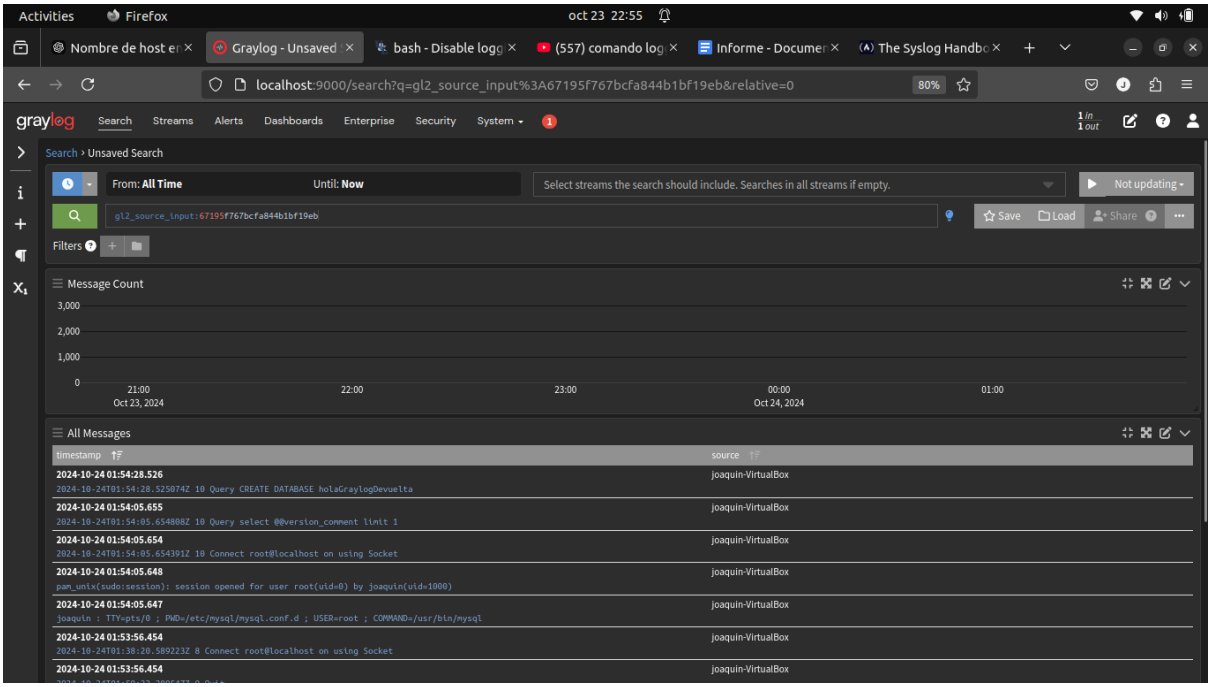
Ahora configuramos syslog para que atienda a este archivo de logs



Ahora retocamos un par de cosas que quedaron y probamos devuelta



Vemos que efectivamente llegan los logs a graylog! 🎉



ALERTAS A PARTIR DE LOGS

(entregable E)

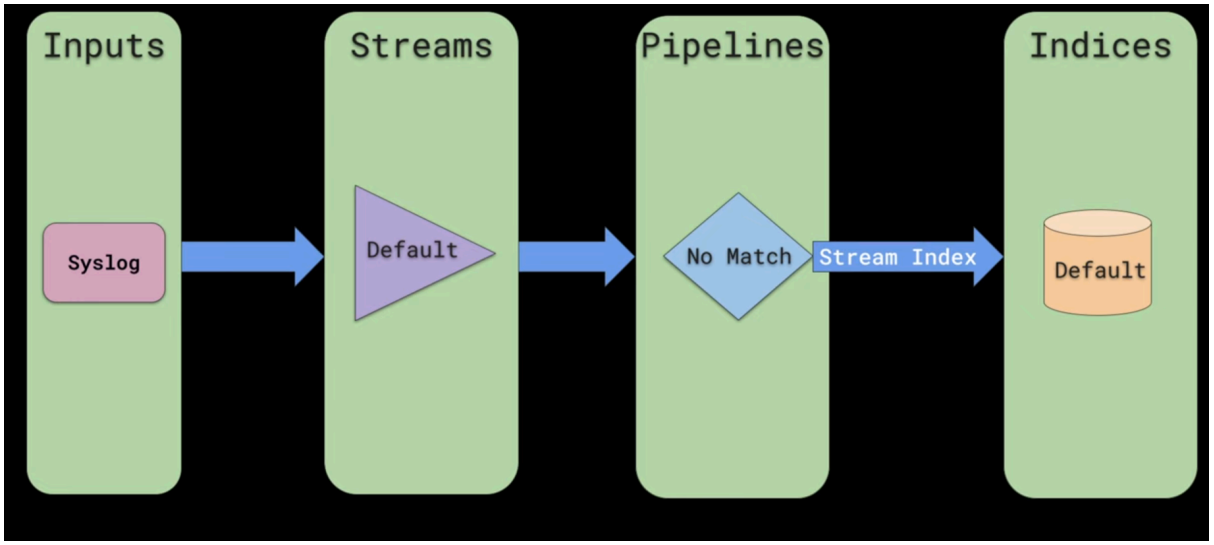
Se presento un error de permisos con syslog a las carpetas de logs, se lo soluciono realizando lo siguiente 👍

```
joaquin@joaquin-VirtualBox:~$ sudo usermod -aG mysql syslog
```

La consigna nos pide enviar un mail ante situaciones la cuales deben ser atendidas por los administradores.

En nuestro caso consideramos que es importante ser alertado cuando un usuario **borre** o realice un **update** de alguna tabla de la base de datos o bien la database entera

Captura de logs mediante streams:



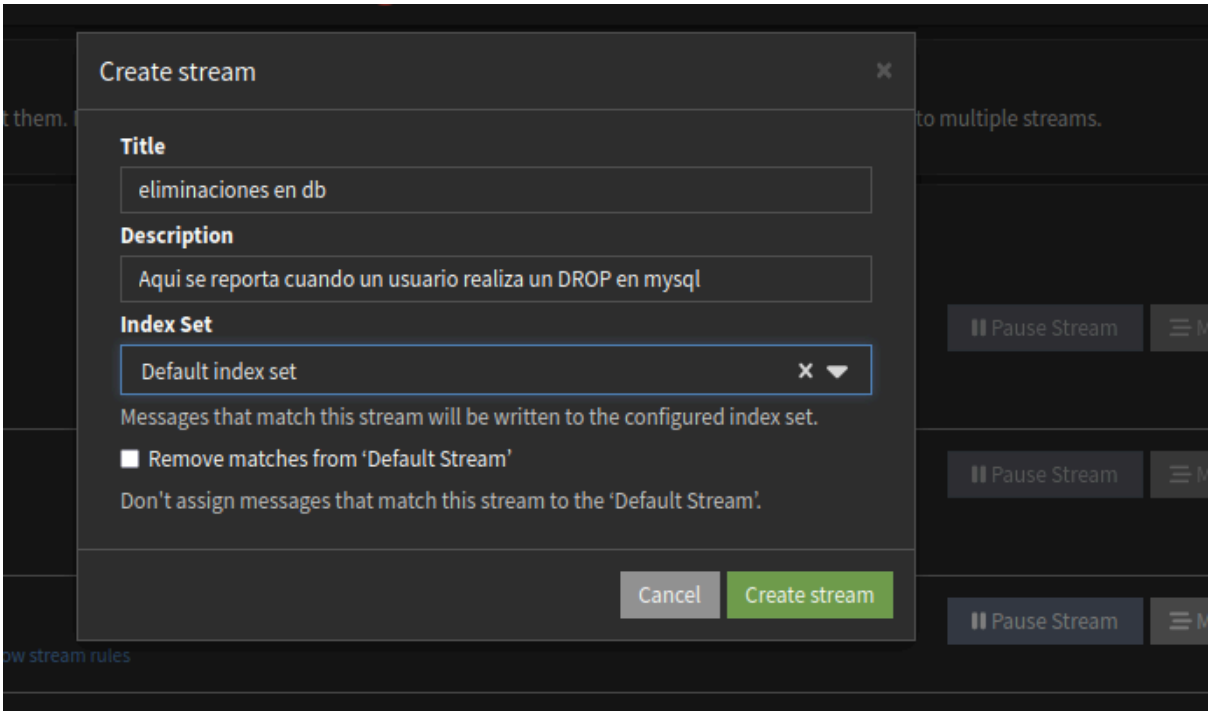
Una vez que nos aseguramos de que los inputs esten capturando correctamente los logs de mysql, debemos hacer algo con ellos. Tenemos que buscar alguna manera de **filtrarlos**

Para esto graylog tiene una herramienta llamada **streams,** la cual podemos establecerle reglas y cuando este encuentra alguna que match la guarde aquí.

Proseguimos a abrir un input en particular de mysql y abrimos en detalle:

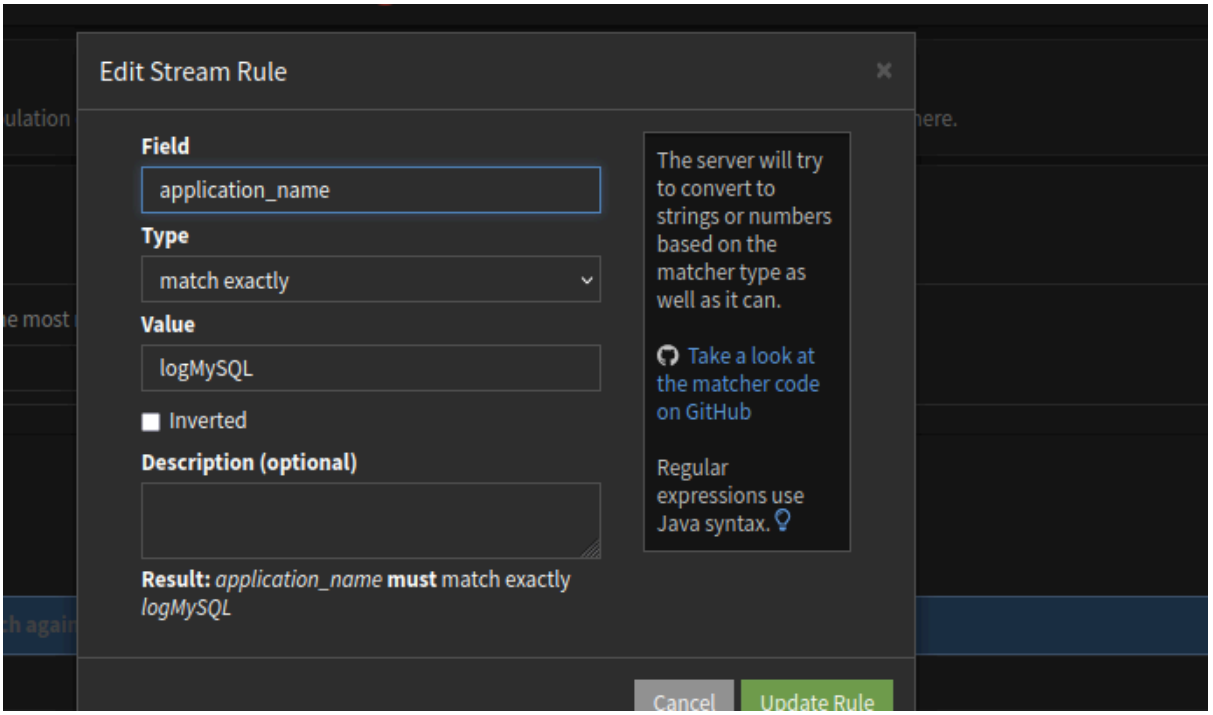
Vemos que nos interesan los siguientes campos:
application_name : aquí vemos que la aplicación o archivo que está enviando los logs se llama *logMySQL*
message : en este caso reporta lo que el usuario hizo en la consulta mysql

Ahora vamos a la pestaña **streams** y seleccionamos *create new stream*



Ingresamos un título determinado y si queremos un lenguaje, y finalmente un index set, el cual es optativo por si tenemos otro **index** que es donde elegimos guardar los logs que nos van llegando a graylog. En este caso debido a que es algo pequeño no resulta necesario crear otro index, por lo tanto usamos el default

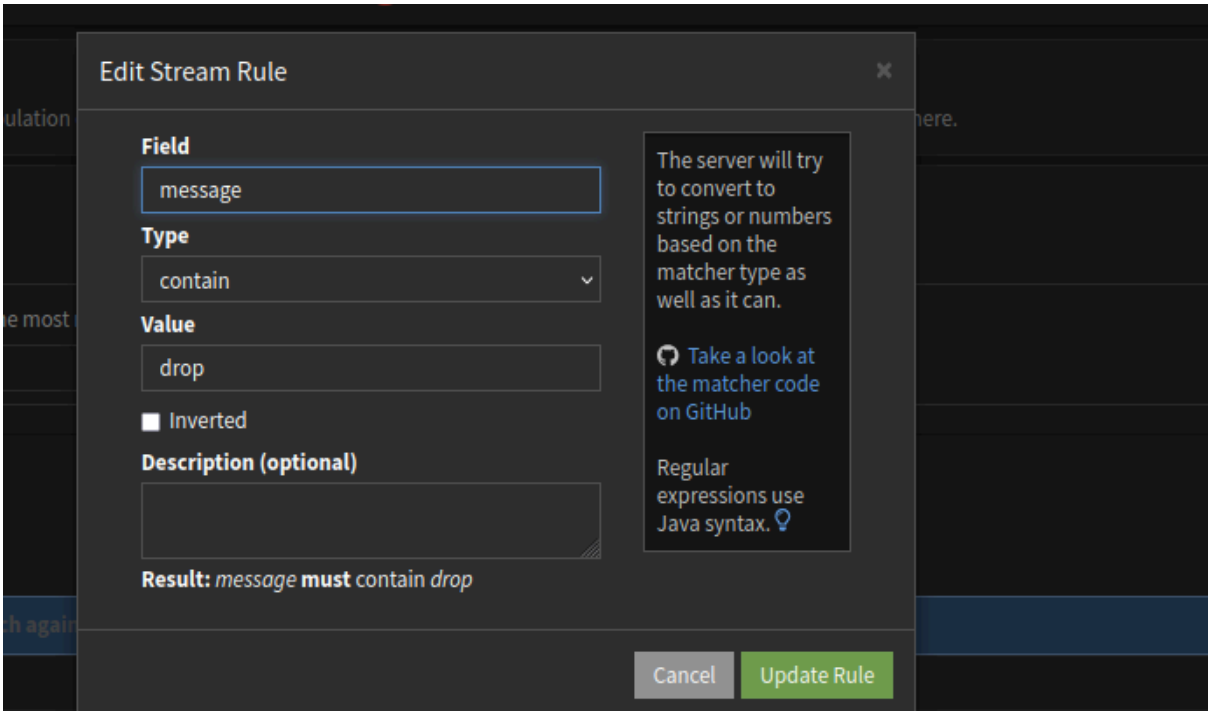
Le damos a *create stream* y luego lo seleccionamos y oprimimos *manage stream rules*
Aqui vamos a configurar que deseamos guardar en este stream en particular:



En este caso utilizaremos dos reglas las cuales se tienen que cumplir para que los logs vayan a este stream

Una es que **application_name** sea igual a *logMySQL,* que como ya vimos todos los logs de mysql van dirigidos aqui

Y el segundo es el siguiente :

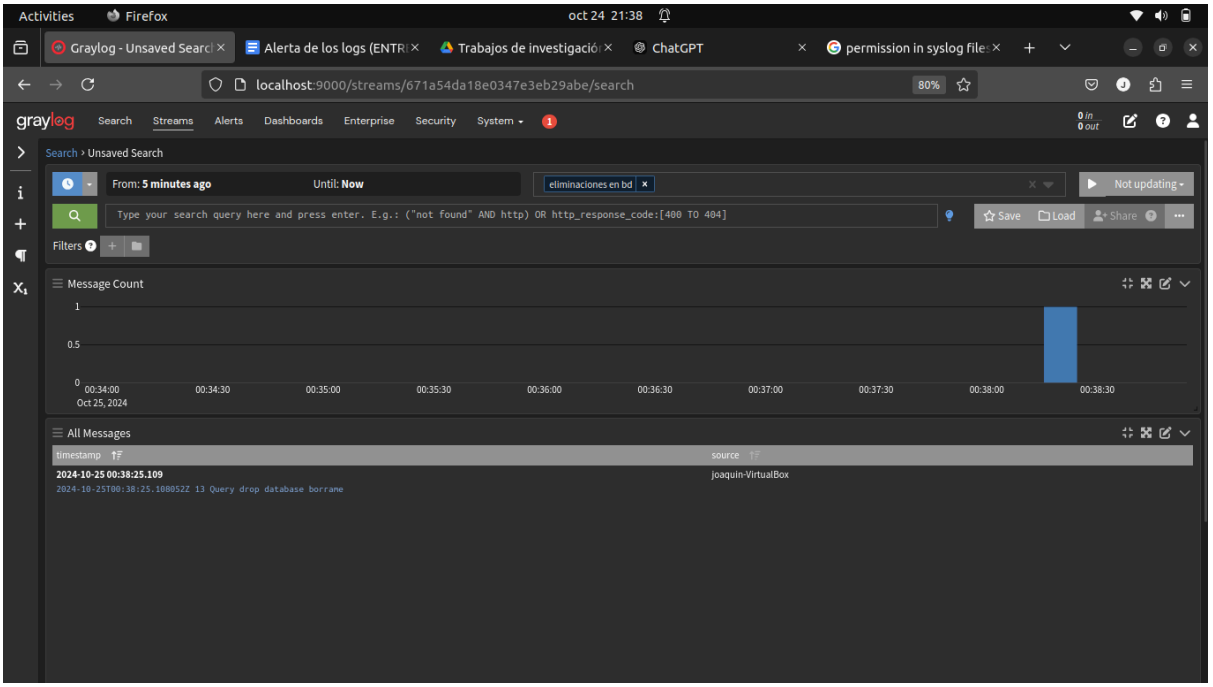


El mensaje reportado debe contener al menos la palabra **drop,** la cual se usa en mysql para borrar tablas y schemas

De este modo nos aseguramos de que si alguien borra algo de la base de datos se reportado aqui, hagamos la prueba

```
mysql> create database borrame ;
Query OK, 1 row affected (0.01 sec)

mysql> drop database borrame ;
Query OK, 0 rows affected (0.02 sec)

mysql>
```

Vamos a graylog y vemos lo siguiente :



Como podemos ver se muestra solamente el **drop** y no el create, por lo tanto lo hicimos correctamente

Alerta mediante mails en graylog:

Ahora que tenemos el stream definido vayamos a investigar el sistema de **alerts** que ofrece graylog.

Para esto debemos investigar un poco sobre que es **SMTP:**

SMTP (Simple Mail Transfer Protocol):

Es un protocolo estándar utilizado para enviar correos electrónicos a través de redes. Funciona como el "lenguaje" que los servidores de correo electrónico utilizan para comunicarse entre sí y transferir correos desde el cliente de correo (las alertas de graylog en este caso) hasta el servidor de correo del destinatario.

Cuando en la cotidianidad enviamos un correo via gmail por ejemplo, lo que esta ocurriendo es que gmail actúa como el cliente SMTP y se comunica con el servidor SMTP, enviando el mensaje que escribimos.

SMTP en Graylog:

Primero que nada, debemos preparar graylog para que esté habilitado para enviar estos mails via smtp

Debido a que estamos corriendo nuestro graylog mediante docker no va a ser posible editar el graylog.conf y agregar ahí nuestras configuraciones **transport**. Realizaremos estas configuraciones en nuestro docker-compose.yml en la parte de *environment,* la cual se encarga de configurar todas estas variables de entorno que nos interesan:

```
- GRAYLOG_HTTP_EXTERNAL_URI=http://127.0.0.1:9000/
#para el envio de mails
- GRAYLOG_TRANSPORT_EMAIL_ENABLED=true
- GRAYLOG_TRANSPORT_EMAIL_PROTOCOL=smtp
- GRAYLOG_TRANSPORT_EMAIL_HOSTNAME=smtp.gmail.com
- GRAYLOG_TRANSPORT_EMAIL_PORT=465
- GRAYLOG_TRANSPORT_EMAIL_USE_AUTH=true
- GRAYLOG_TRANSPORT_EMAIL_USE_TLS=false
- GRAYLOG_TRANSPORT_EMAIL_USE_SSL=true
- GRAYLOG_TRANSPORT_EMAIL_AUTH_USERNAME=joaquincernik@gmail.com
- GRAYLOG_TRANSPORT_EMAIL_AUTH_PASSWORD=[password dada por password para aplicaciones ]
- GRAYLOG_TRANSPORT_EMAIL_SUBJECT_PREFIX=[graylog]
- GRAYLOG_TRANSPORT_EMAIL_FROM_EMAIL=joaquincernik@gmail.com
- GRAYLOG_TRANSPORT_EMAIL_WEB_INTERFACE_URL=https://192.168.0.26:9000
```

Lo que se encontró en distintos videos de youtube y foros es la siguiente configuración que se agregaba en los archivos .conf de graylog:

Email transport

transport_email_enabled = true #This is important!
transport_email_hostname = smtp.gmail.com 406 #Take note of this address!
transport_email_port = 465 #Yours may vary
transport_email_use_auth = true
transport_email_use_tls = true
transport_email_use_ssl = true
transport_email_auth_username = specialaccount@yourdomain.tld #This is an account you've created for the purpose of sending automated emails
transport_email_auth_password = Y0urPassw0rd!
transport_email_subject_prefix = [graylog]
transport_email_from_email = graylog@example.com

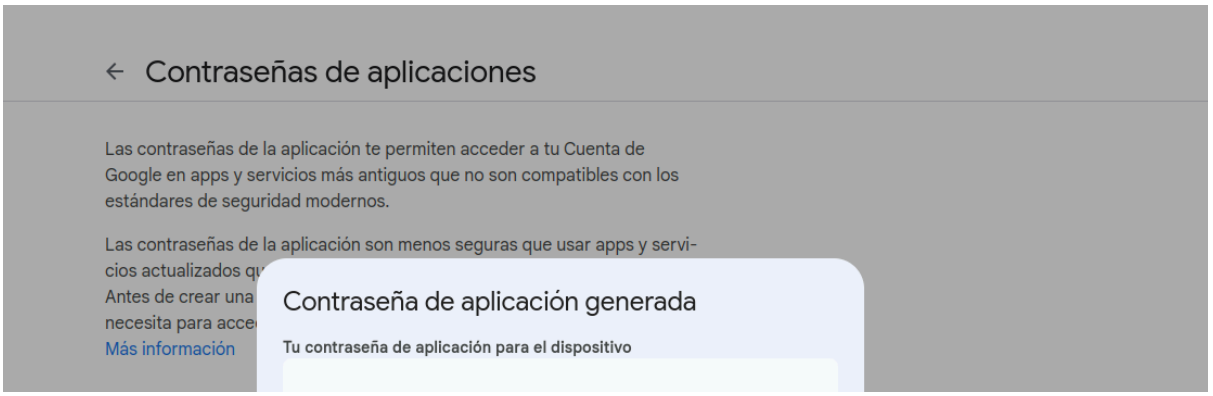
Pero como nosotros lo estamos configurando mediante docker, las variables de entorno deben ser especificadas con GRAYLOG como prefijo y todo el texto que va a la izquierda del = en mayuscula.

En nuestro caso ya que estamos usando gmail usaremos el puerto **465**

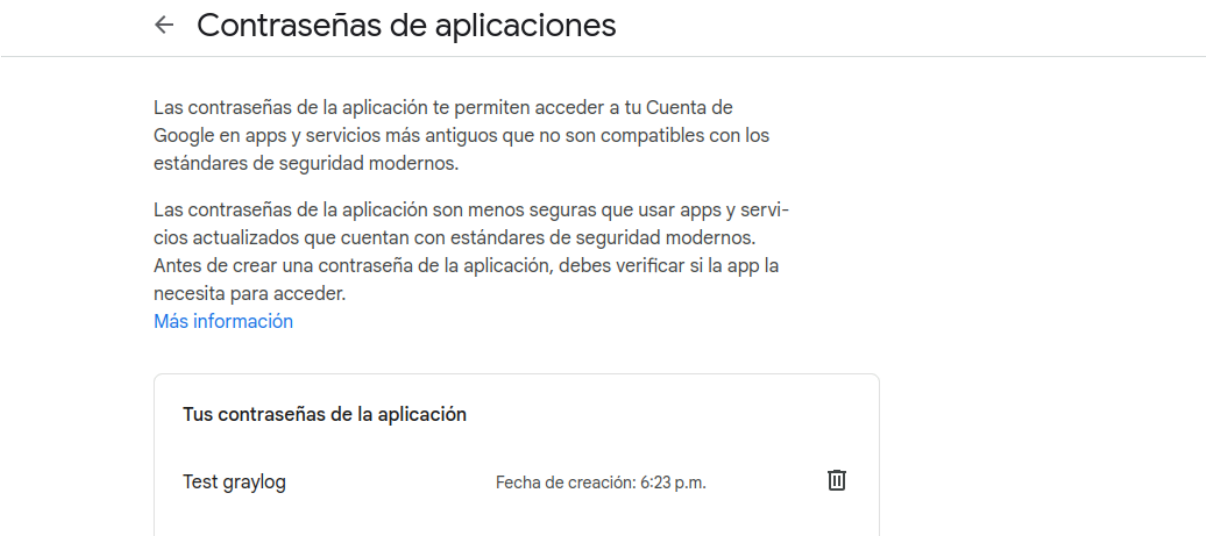
Configuración de gmail para habilitar los mensajes via smtp automatizados

Debido a que gmail ya no deja realizar el enviado de estos mensajes automáticamente con el usuario y contraseña corrientes, debemos setear una *password application,* la cual habilitara a graylog para enviar los mensajes

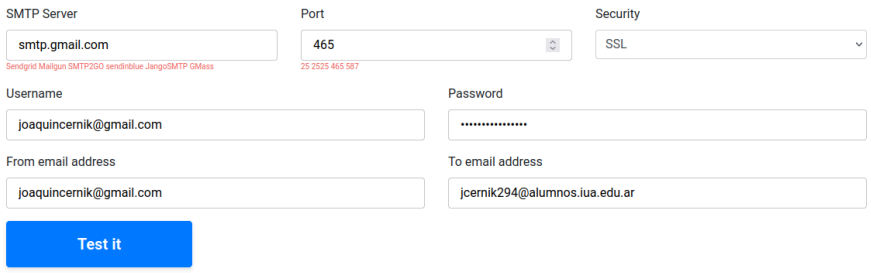
Vamos a nuestra cuenta y presionamos *manage account settings.* Una vez adentro buscamos la opcion de *application passwords,* donde elegimos un nombre para representar esta acción y nos dara una contraseña en especifico:



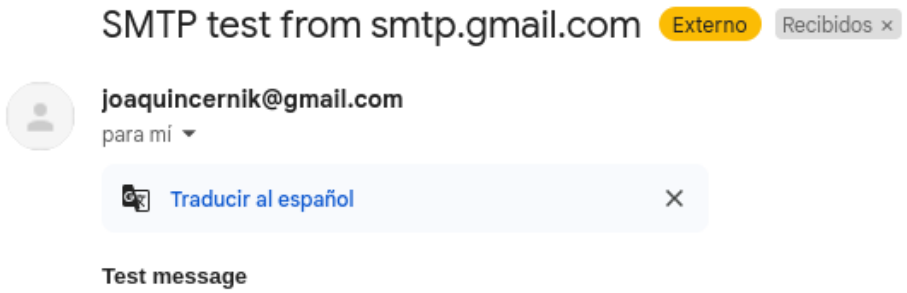
Una vez dada esta password, debemos copiarla rápidamente en algún lugar, ya que no va a poder ser accedida de vuelta.
La copiamos y la pegamos en la parte del docker-compose donde nos pide el AUTH_PASSWORD, en la imagen previa.



Ahora para asegurarnos de que este funcionando correctamente vamos a testear que se envien bien los mails en la siguiente pagina: https://www.gmass.co/smtp-test
Donde ingresamos los siguientes datos:



En password ingresamos la contraseña generada por google sin los espacios.



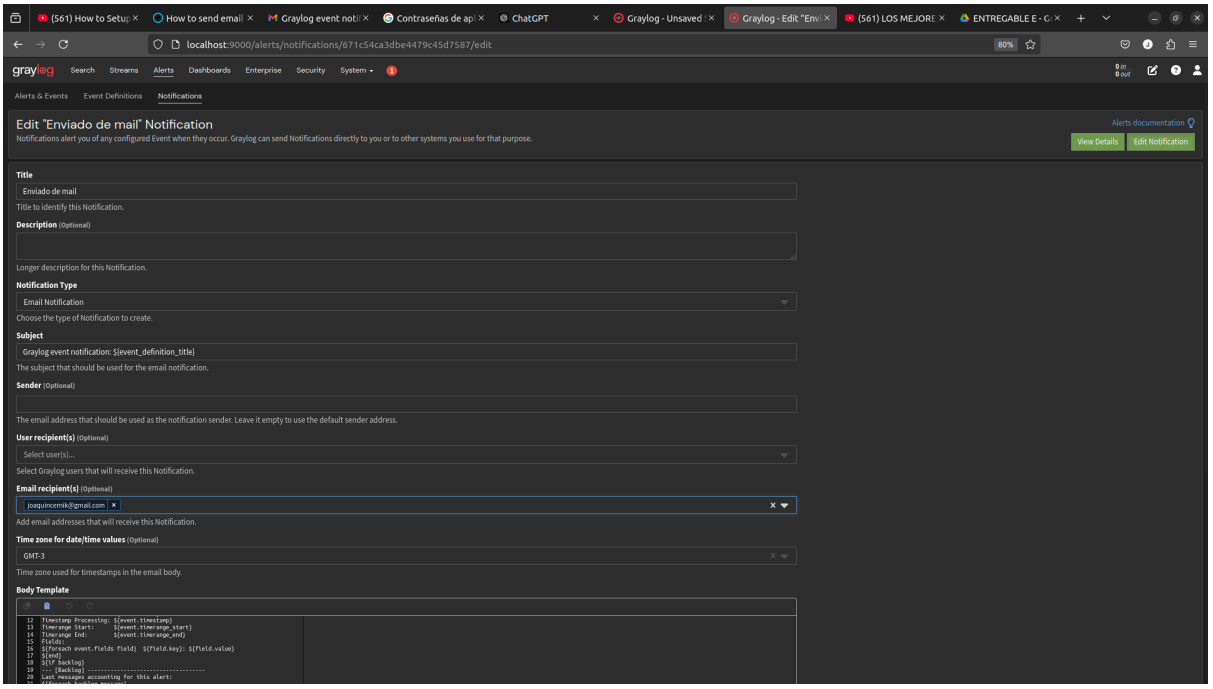
Vemos que llega correctamente 👍

Por lo tanto podemos seguir a la configuración de alertas en graylog

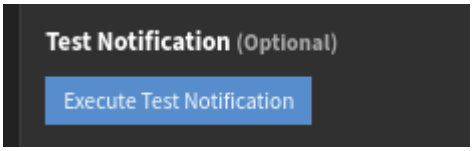
Creación de notification en graylog:

Vamos a *notifications* y seleccionamos *create notification:*

Una vez dentro lo configuramos de la siguiente manera:

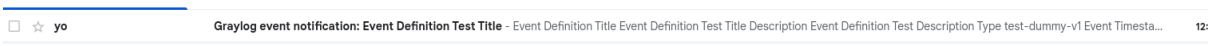


Le damos a test para que realice una prueba:



Y obtenemos:

Vamos al mail para corroborarlo y efectivamente se recibio el mail de prueba:

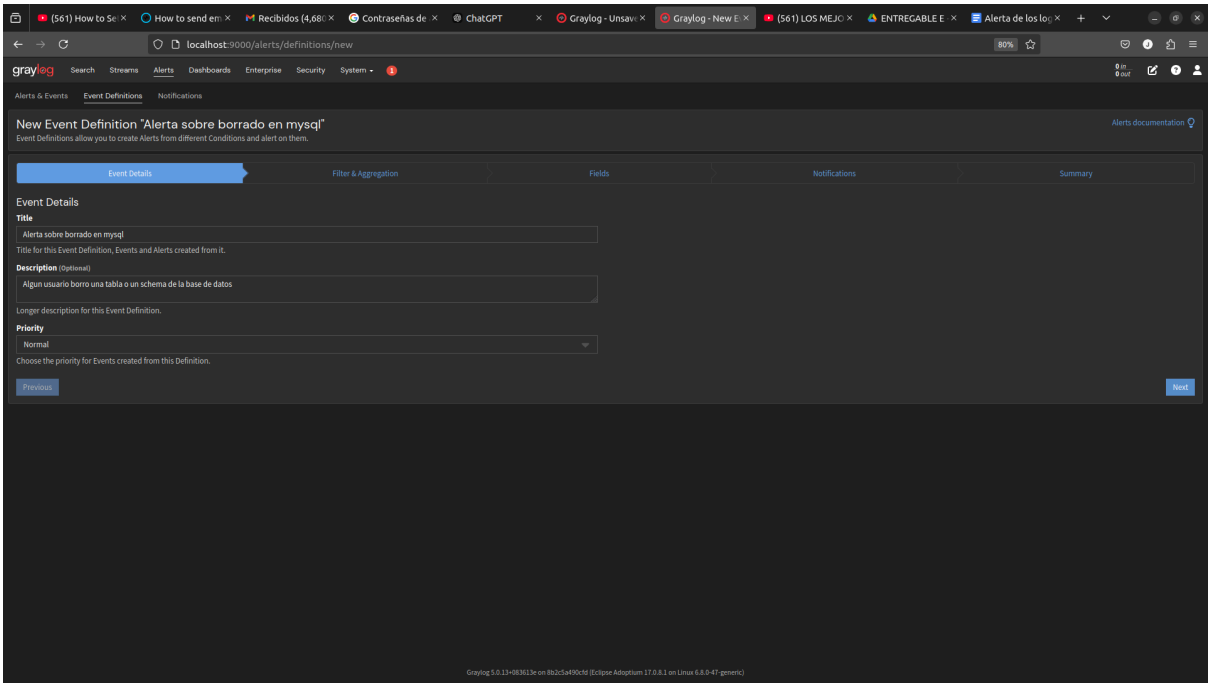


Creación del evento disparador de la alerta en graylog

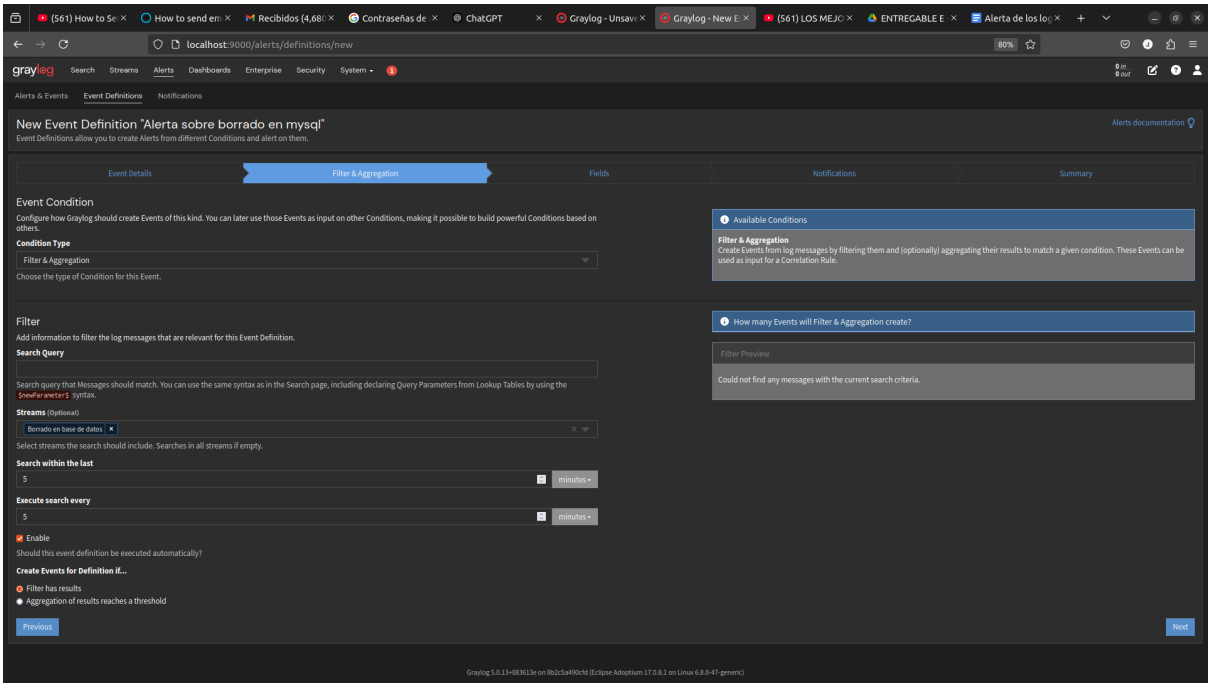
Ahora que tenemos configurada la notificación que se va a ejecutar cuando surja el evento que queremos que sea notificado tenemos que definir dicho evento

Es decir, tenemos que decirle a graylog cuando sera el momento que esté alerta se active

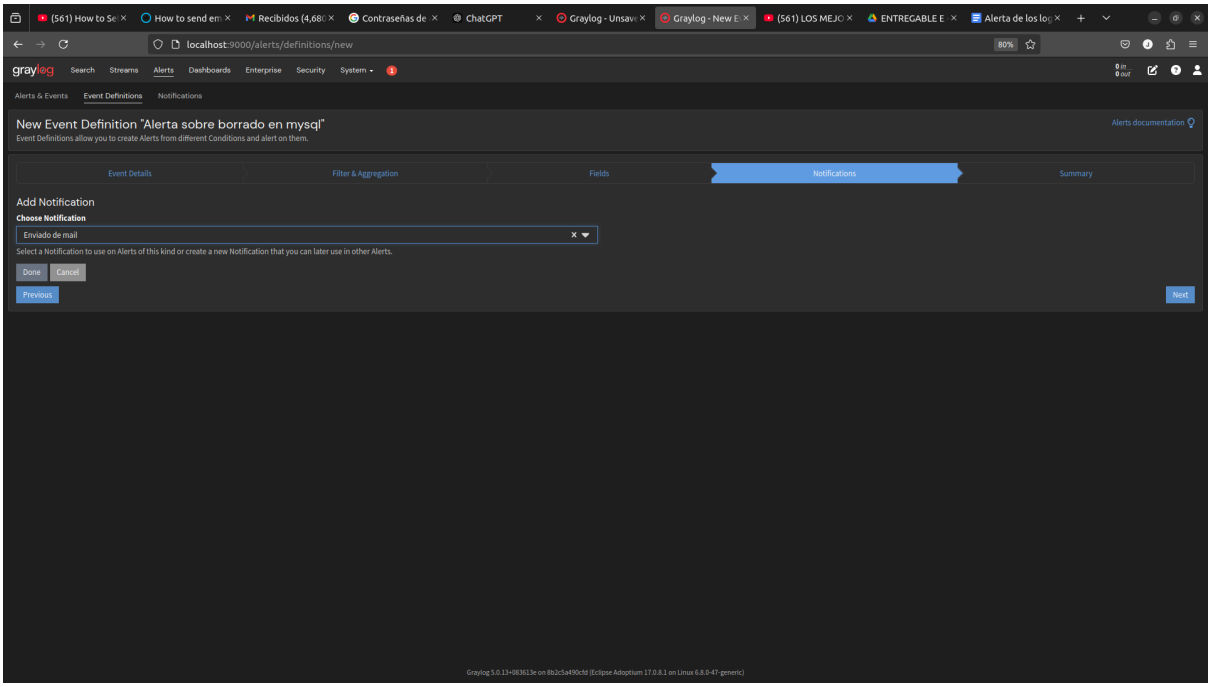
Lo que hacemos es ir a evento definition y creamos un evento



Una vez que rellenamos estos datos le damos a siguiente

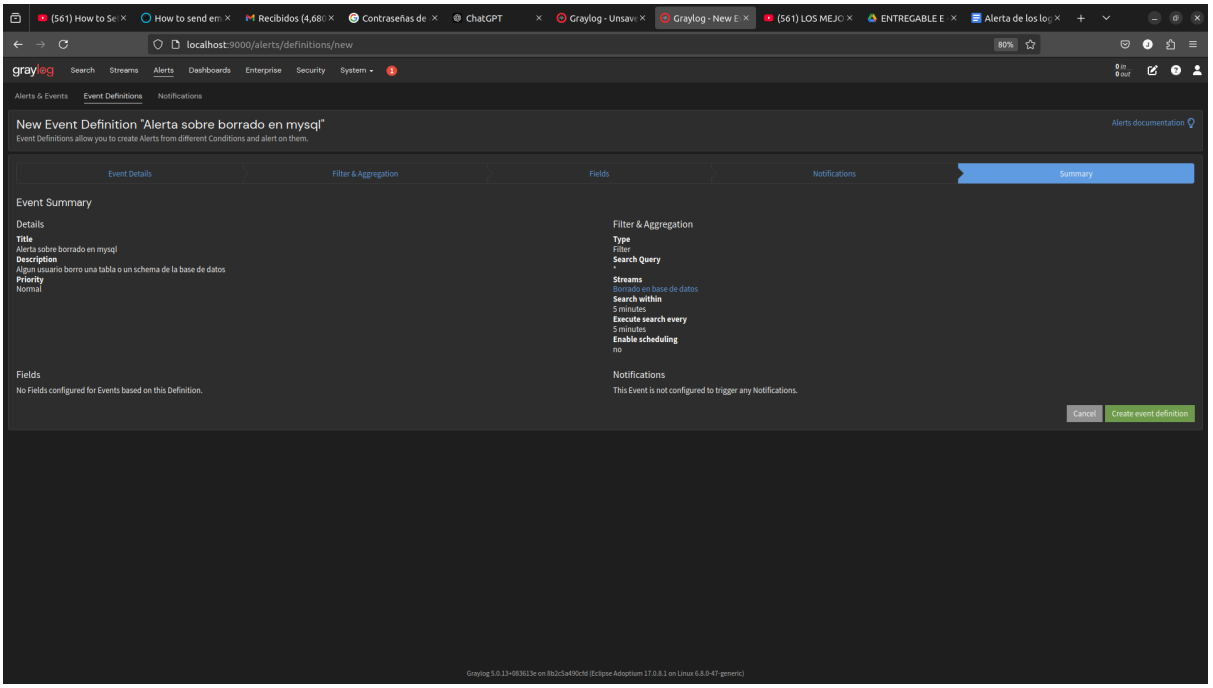


Aqui vamos a elegir nuestro stream, el cual es el que ya hemos configurado para que nos filtre los logs que nos interesa



Y finalmente en notificación agregamos la notificación de envío de mail

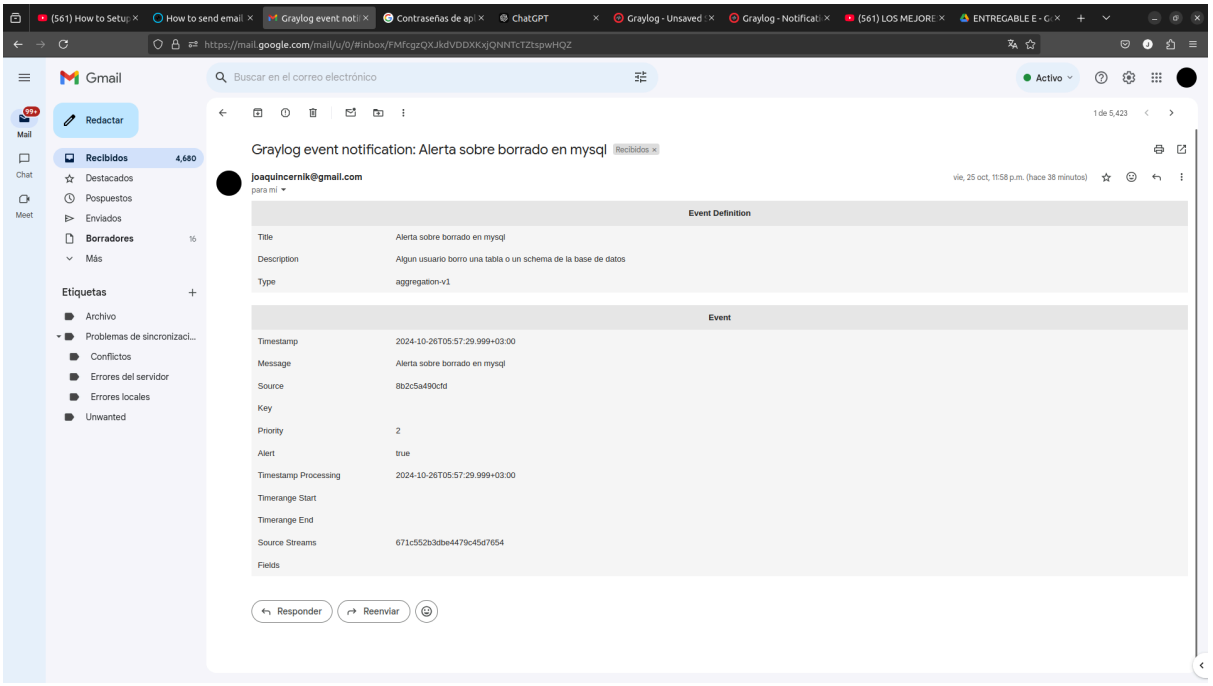
Nos queda un evento de la siguiente manera:



Ahora probamos borrando un schema por ejemplo:

```
mysql> drop database borrame ;
Query OK, 0 rows affected (0.02 sec)
```

Y vemos que nos llegó el siguiente mail:



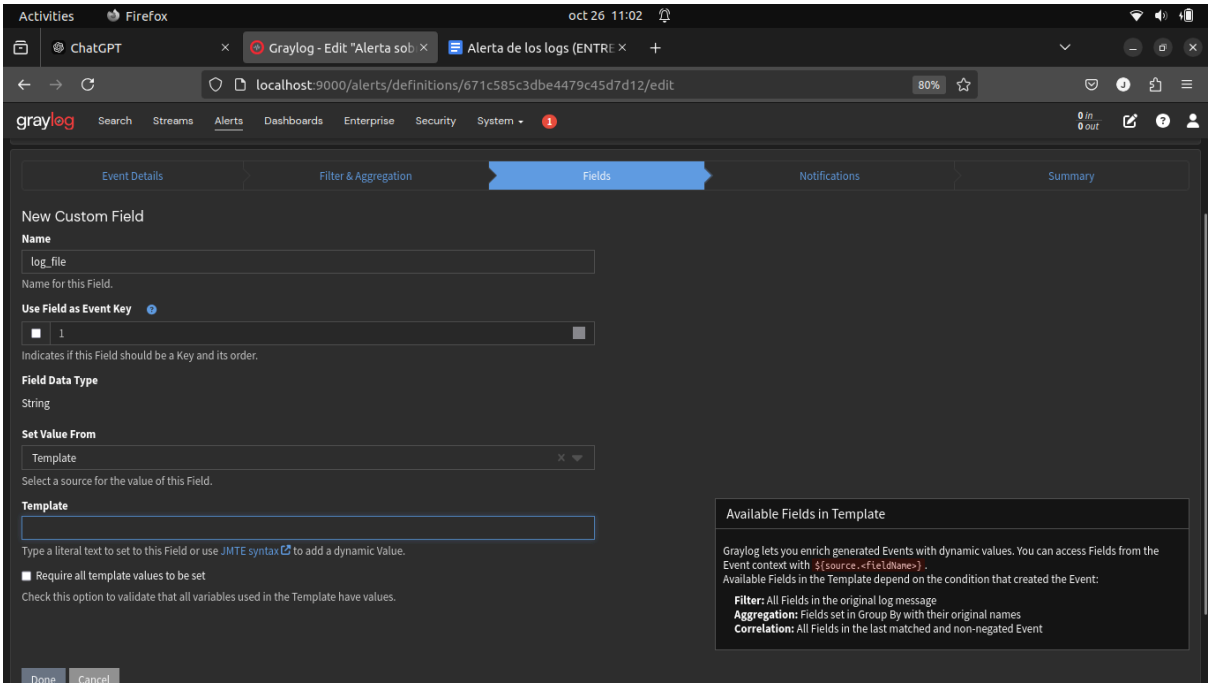
Como podemos ver se muestran pocos datos sobre la alerta, lo preferible es que podamos saber por lo menos que tabla o que schema ha sido borrado.

Por lo tanto ahora que vayamos a crear las alertas de UPDATE, debemos buscar esa precisión

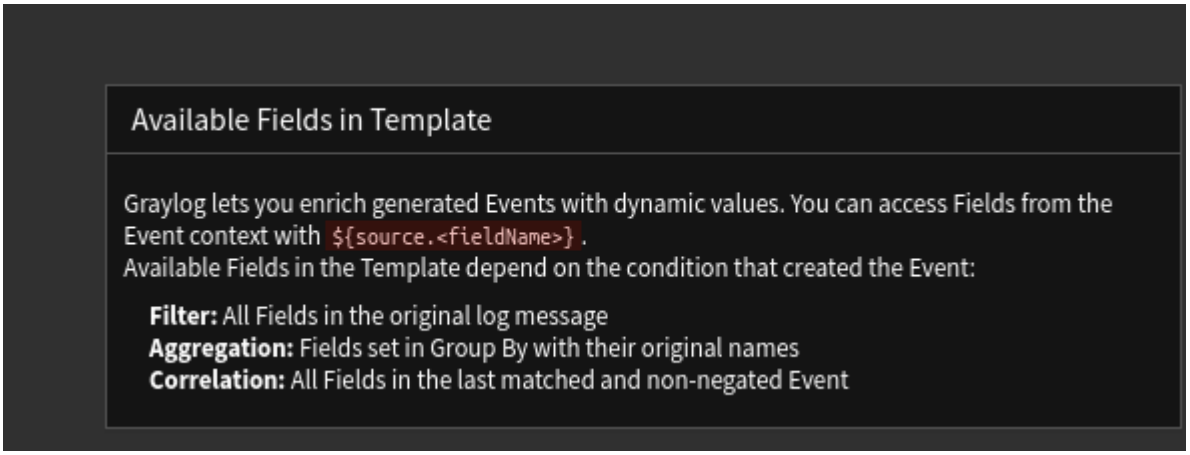
Empleo de **fields** en eventos

Para realizar lo siguiente, graylog utiliza **fields**, para mostrar en las notificaciones los campos que querramos de el log que nos llego

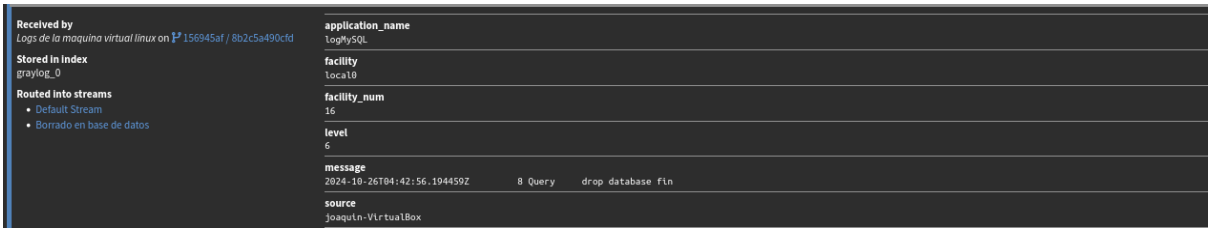
Por ejemplo, vamos a crear una field. Vamos a nuestro evento creado, le damos a la sección de field y creamos un evento



Aqui por ejemplo si deseo saber de que archivo vino el log correspondiente.
Prestamos atención a lo siguiente

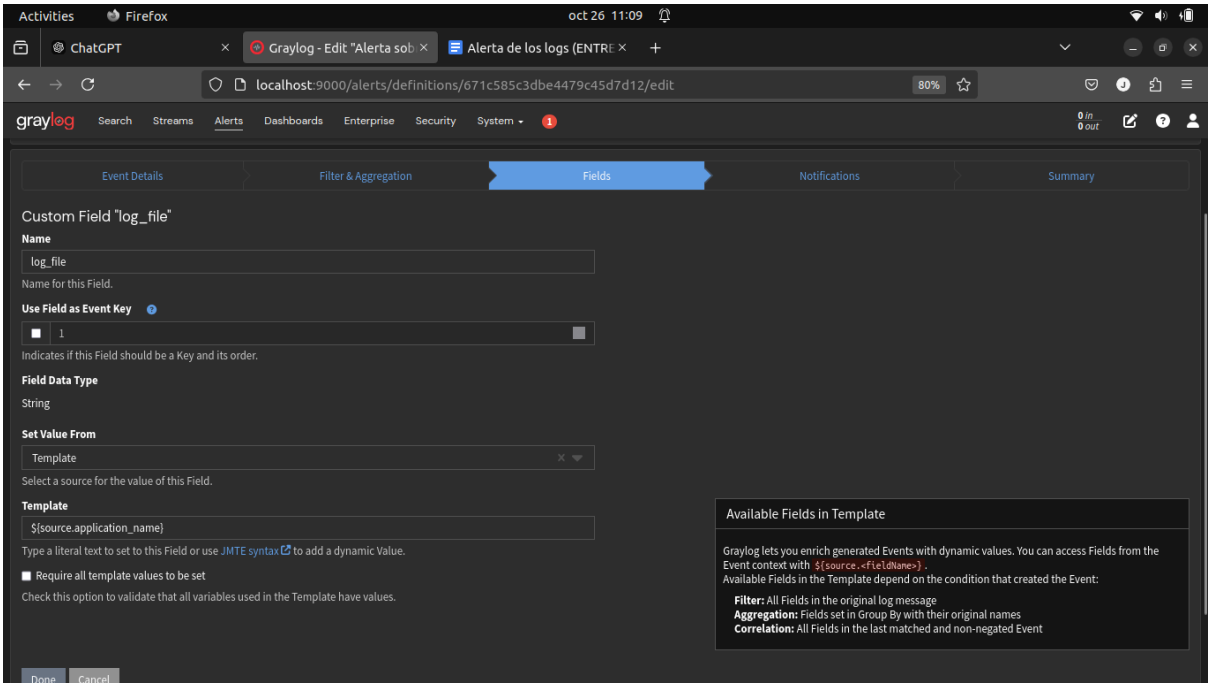


Esto nos dice que podemos acceder a los atributos del log realizando *${source.<fieldName>}*
Por lo tanto veamos que atributos contiene el log de MySQL:



Vemos que presenta estos atributos, donde resaltamos **application_name, source** y **message,** ya que son de nuestro interés

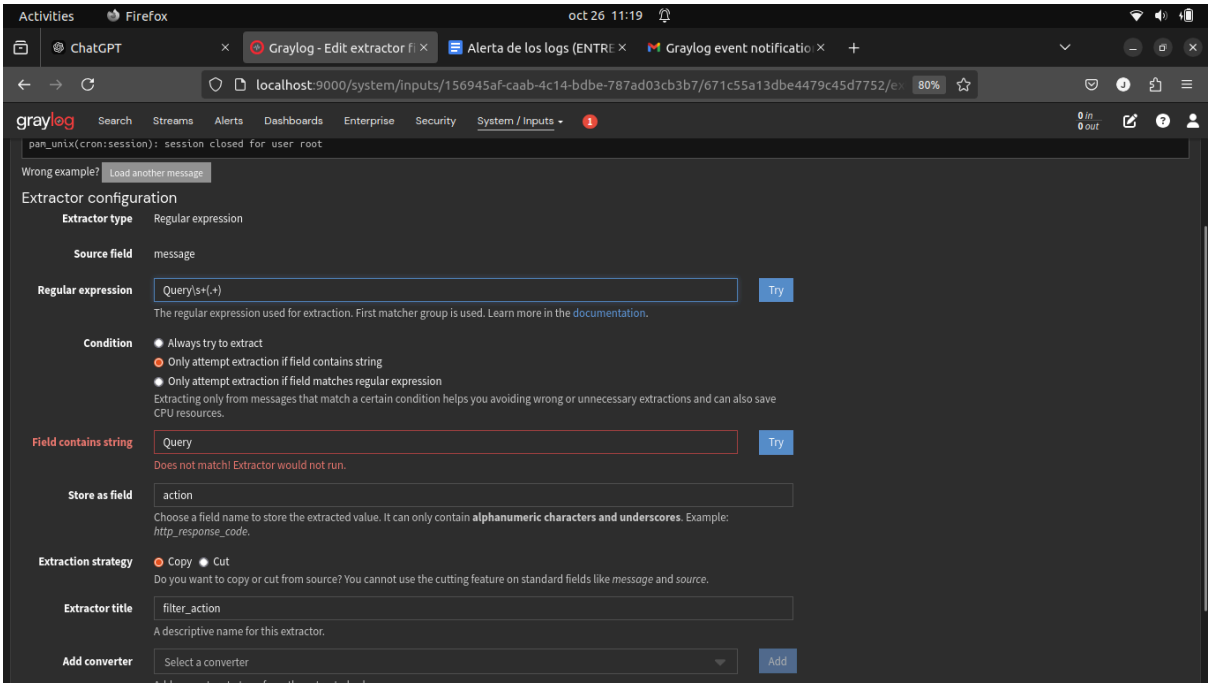
Por lo tanto en la field creada insertamos lo siguiente:



Extractors en graylog:

Eso estuvo bien, pero ahora se nos presenta el siguiente problema: cuando queremos imprimir el evento que el usuario realizó nos llega de la siguiente manera:

Vemos que solamente nos interesa la última parte, por lo tanto lo que hay que hacer es crear un **extractor** que se encargue de apartar en una field lo que a nosotros nos interesa. Por lo tanto vamos a input, y seleccionamos manage extractor. Una vez dentro creamos un extractor:



En este caso utilizaremos expresiones regulares para filtrar lo que necesitamos. Viendo el mensaje, nos damos cuenta que todo lo que siga después de Query nos es de utilidad, por lo tanto escribimos la expresión regular:

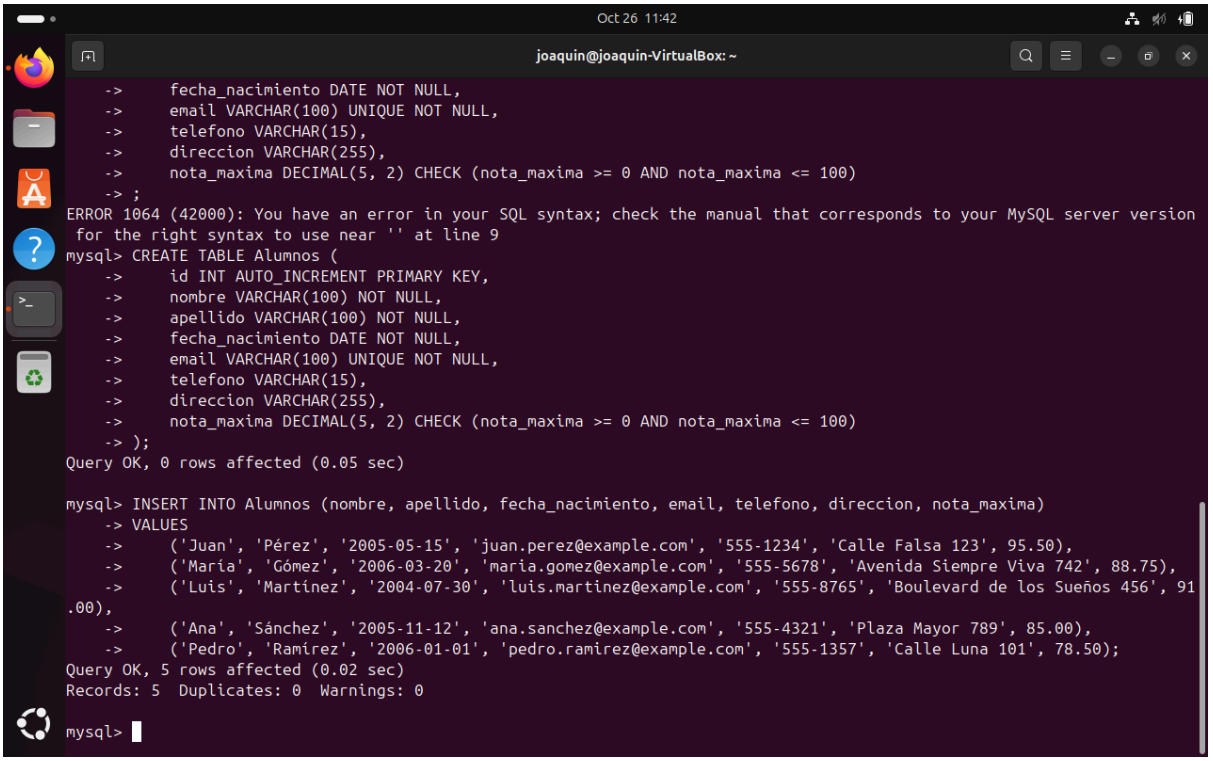
Query\s+(.+)

`Query`: busca literalmente la palabra "Query" al inicio del patrón.
`\s+`: `\s` representa cualquier espacio en blanco (como espacio, tabulación o salto de línea). El signo `+` indica que debe haber uno o más espacios en blanco después de "Query".
`(.+)`: el paréntesis crea un **grupo de captura** para los caracteres que siguen. El punto (`.`) representa cualquier carácter (excepto saltos de línea, a menos que se especifique un modo para incluirlos) y el signo `+` indica que puede haber uno o más caracteres de cualquier tipo. Este extractor actuara siempre que Query este dentro del mensaje

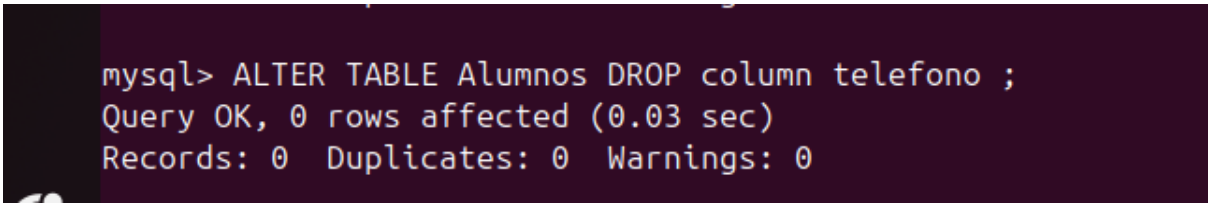
Hagamos una prueba y veamos como nos llega:

Funcionó de la manera esperada 👍

Hagamos una prueba creando una tabla llamada **alumnos** por ejemplo y veamos que pasa cuando alguien entra a nuestra base de datos y trata de borrar alguna columna



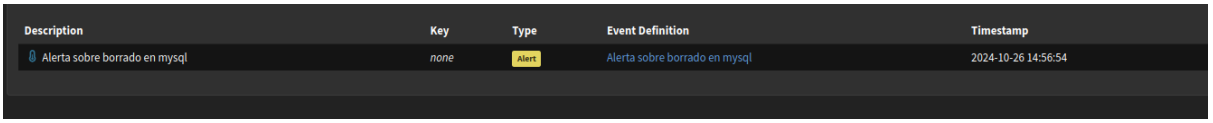
Supongamos que una persona ingresa a nuestra base de datos y borra con el mero fin de dañar nuestra organización borra la columna teléfonos 😣



Nos llega el siguiente mail:

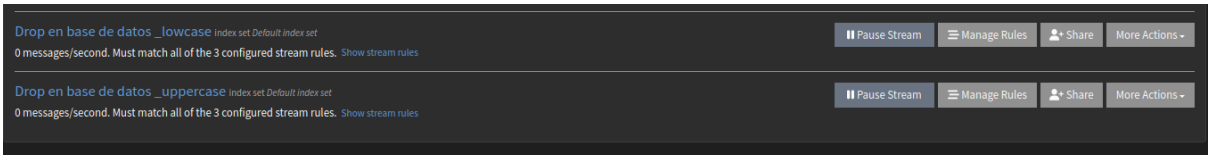
Type	aggregation-v1

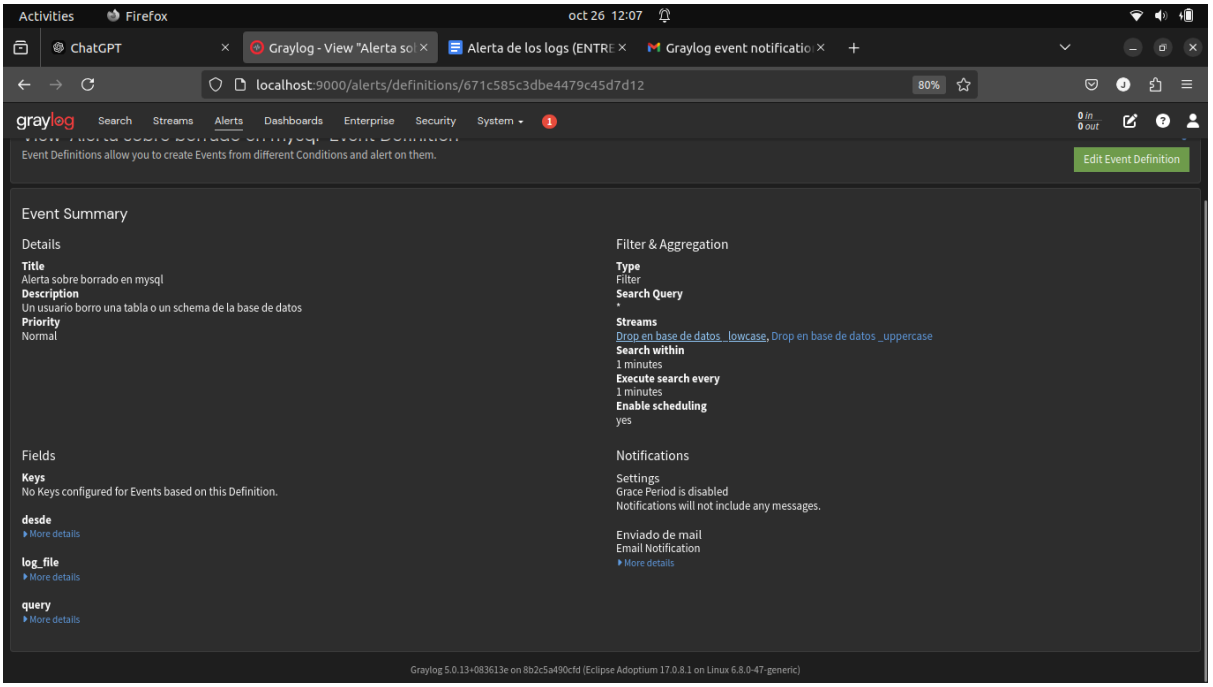
Event	
Timestamp	2024-10-26T17:56:54.657+03:00
Message	Alerta sobre borrado en mysql
Source	8b2c5a490cfd
Key	
Priority	2
Alert	true
Timestamp Processing	2024-10-26T17:56:54.657+03:00
Timerange Start	
Timerange End	
Source Streams	671d01f1721db279733121c7
Fields	▪ desde:joaquin-VirtualBox ▪ log_file:logMySQL ▪ query:ALTER TABLE Alumnos DROP column telefono



Ahora podemos saber al instante cuando se borra alguna columna, tabla o bien base de datos entera y podemos tomar acciones o medidas en base a lo ocurrido. Lo ideal sería saber el usuario que cometió el borrado, pero los logs de mysql por defecto no te tiran quien fue el ejecutante de la query y por razones de tiempo no se pudo implementar otra manera.

Ya que MySQL no es cap sensitive, fue necesario tomar las siguientes precauciones:





Se tuvo que hacer dos streams distintos, uno para mayúscula y otro para minuscula, y a su vez decirle al evento que esté atento a estos 2 streams

Alerta de que un usuario modifico alguna columna en la base de datos

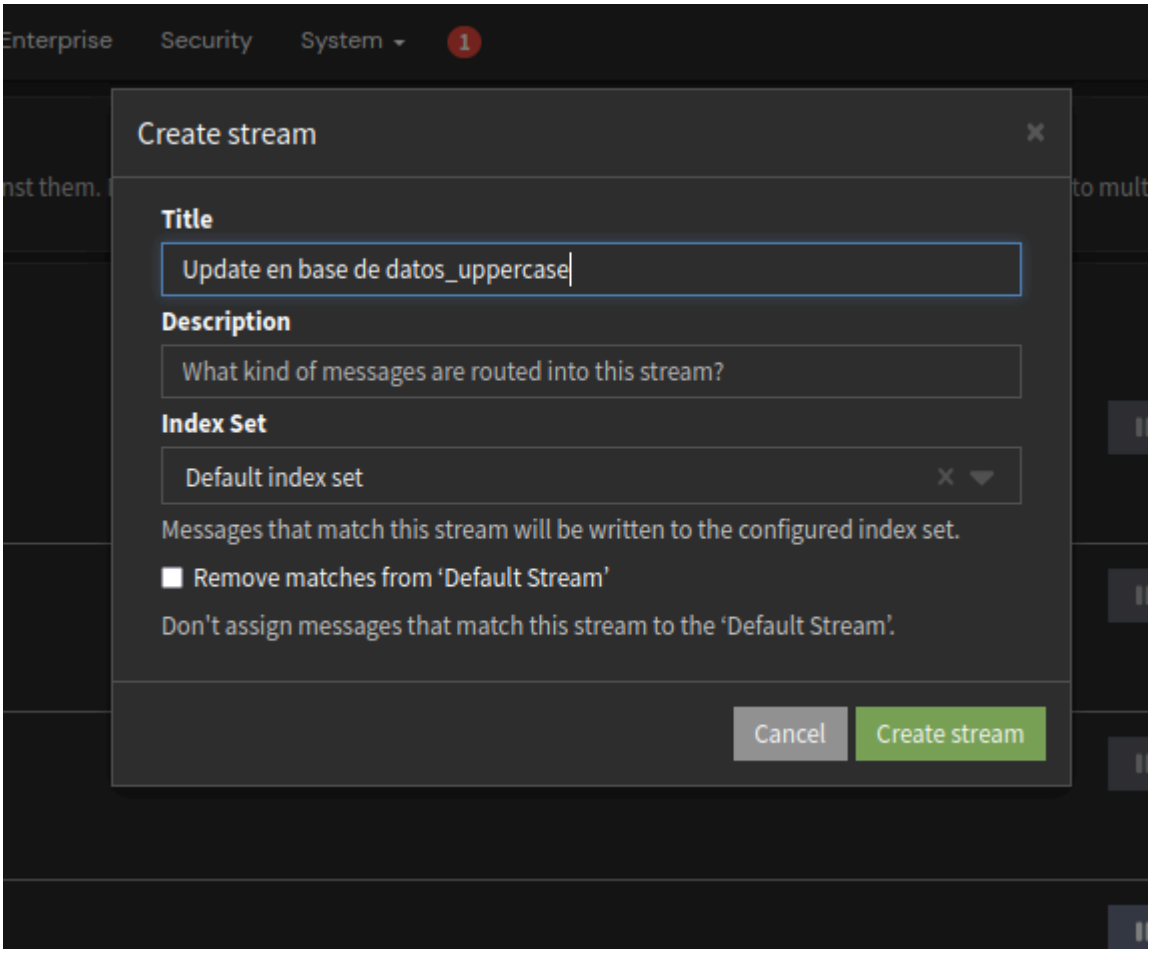
Las notificaciones sobre un posible atentado sobre la estructura de nuestra base de datos estuvo bien, pero que pasaría si algún alumno astuto se infiltró en nuestra base de datos y trata de cambiar su nota máxima 😐?
No podemos dejar que esto pase inadvertido, por lo tanto debemos ser alertados en el caso que esto ocurra!

Tenemos la siguiente tabla:

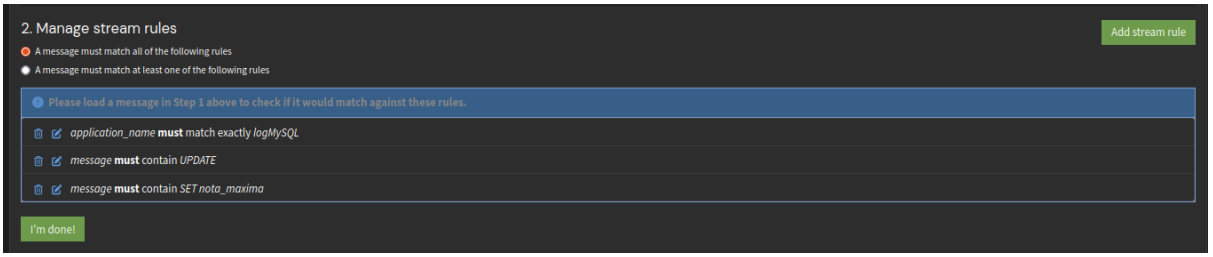
```
mysql> Select * from Alumnos ;
+----+--------+-----------+-----------------+--------------------------+-------------+
| id | nombre | apellido  | fecha_nacimiento | email                    | nota_maxima |
+----+--------+-----------+-----------------+--------------------------+-------------+
|  1 | Juan   | Pérez     | 2005-05-15       | juan.perez@example.com   |       45.00 |
|  2 | Maria  | Gómez     | 2006-03-20       | maria.gomez@example.com  |       88.00 |
|  3 | Luis   | Martínez  | 2004-07-30       | luis.martinez@example.com |       76.25 |
|  4 | Ana    | Sánchez   | 2005-11-12       | ana.sanchez@example.com  |       89.75 |
|  5 | Pedro  | Ramírez   | 2006-01-01       | pedro.ramirez@example.com |       83.00 |
+----+--------+-----------+-----------------+--------------------------+-------------+
5 rows in set (0.00 sec)
```

Supongamos la situación que Juan Perez realiza un intento desesperado por cambiar su nota y logra la manera de infiltrarse en la base de datos 🚫

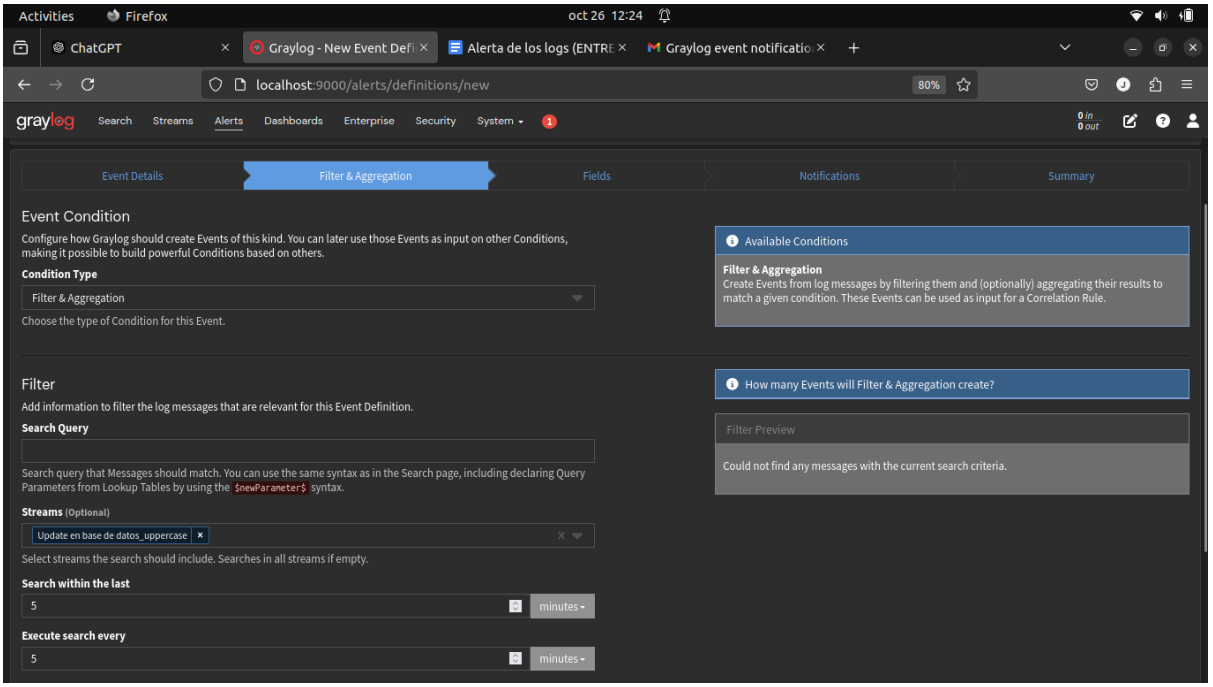
Creamos un stream:



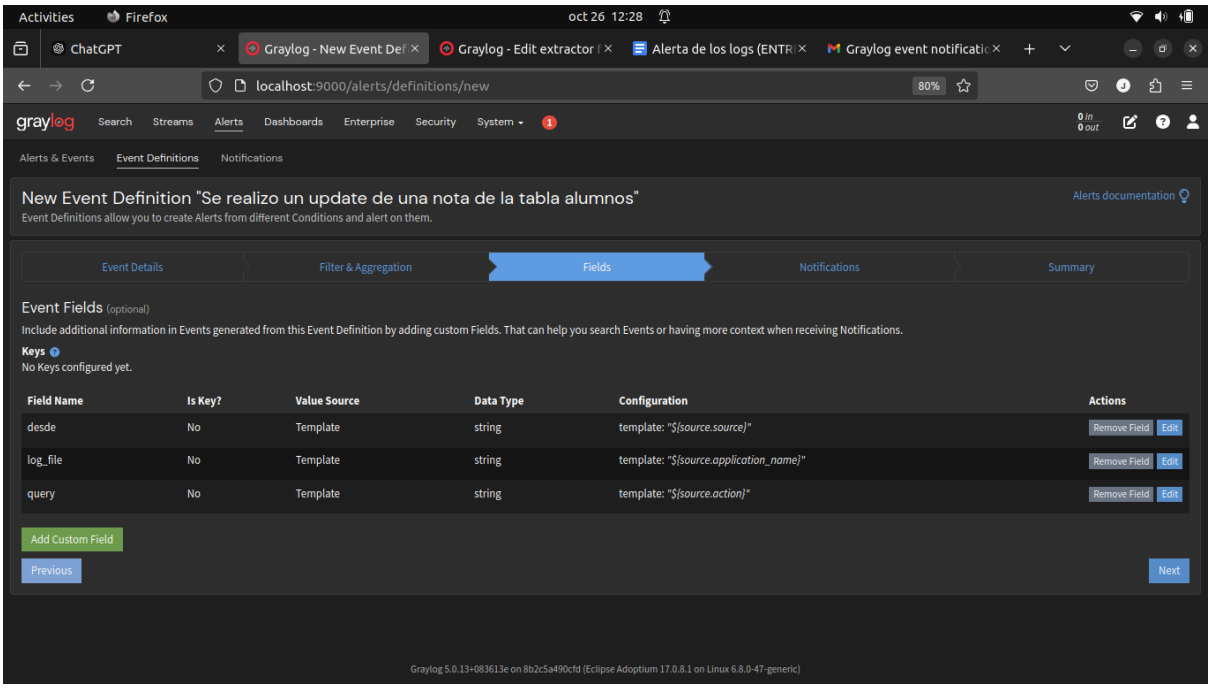
Definimos sus reglas:



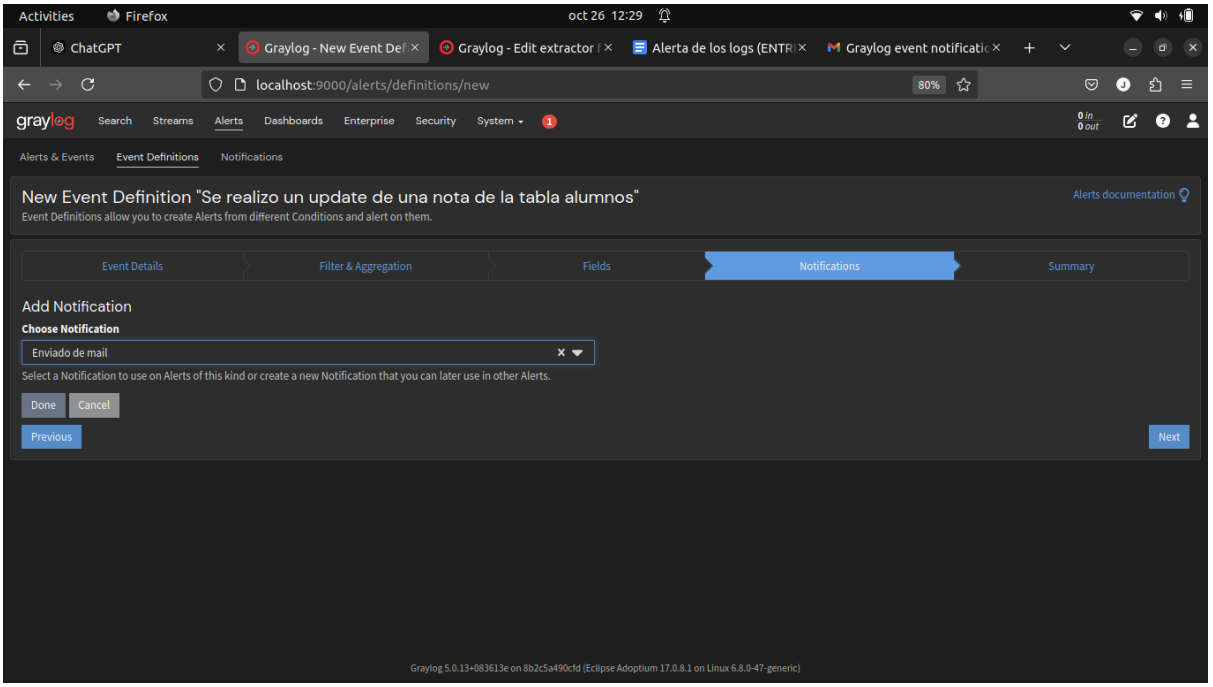
Creamos el evento:



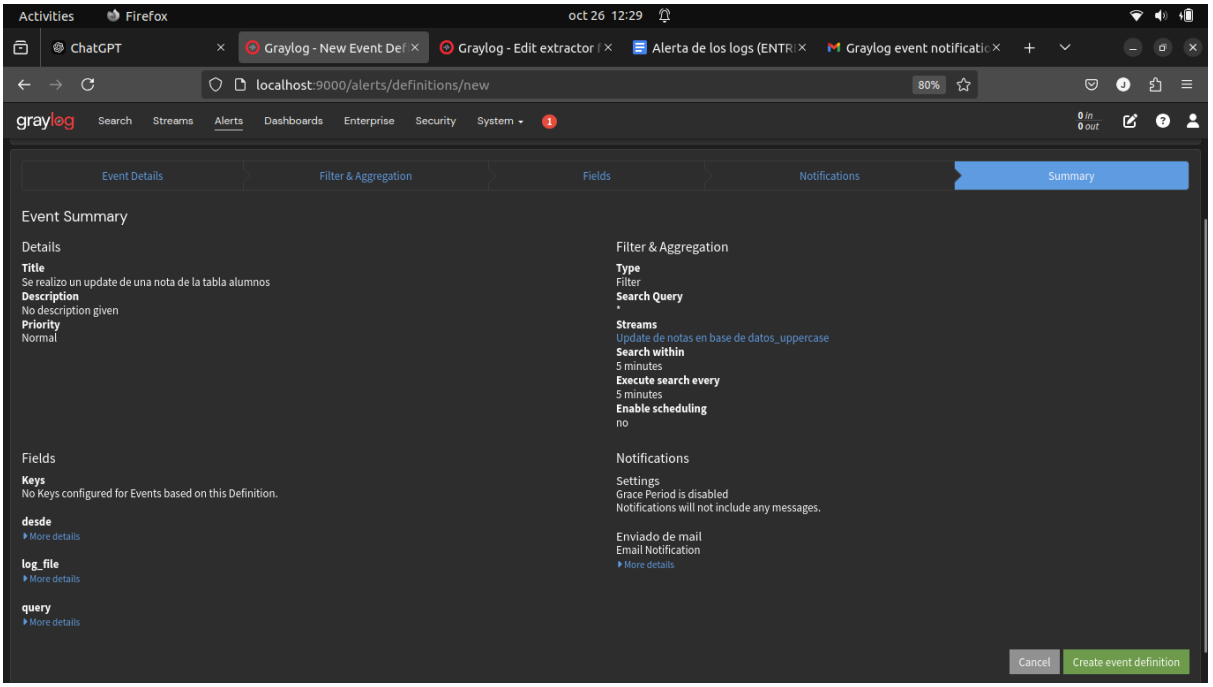
Creamos las fields:



Elegimos notificación mediante envío de mail que previamente creamos:

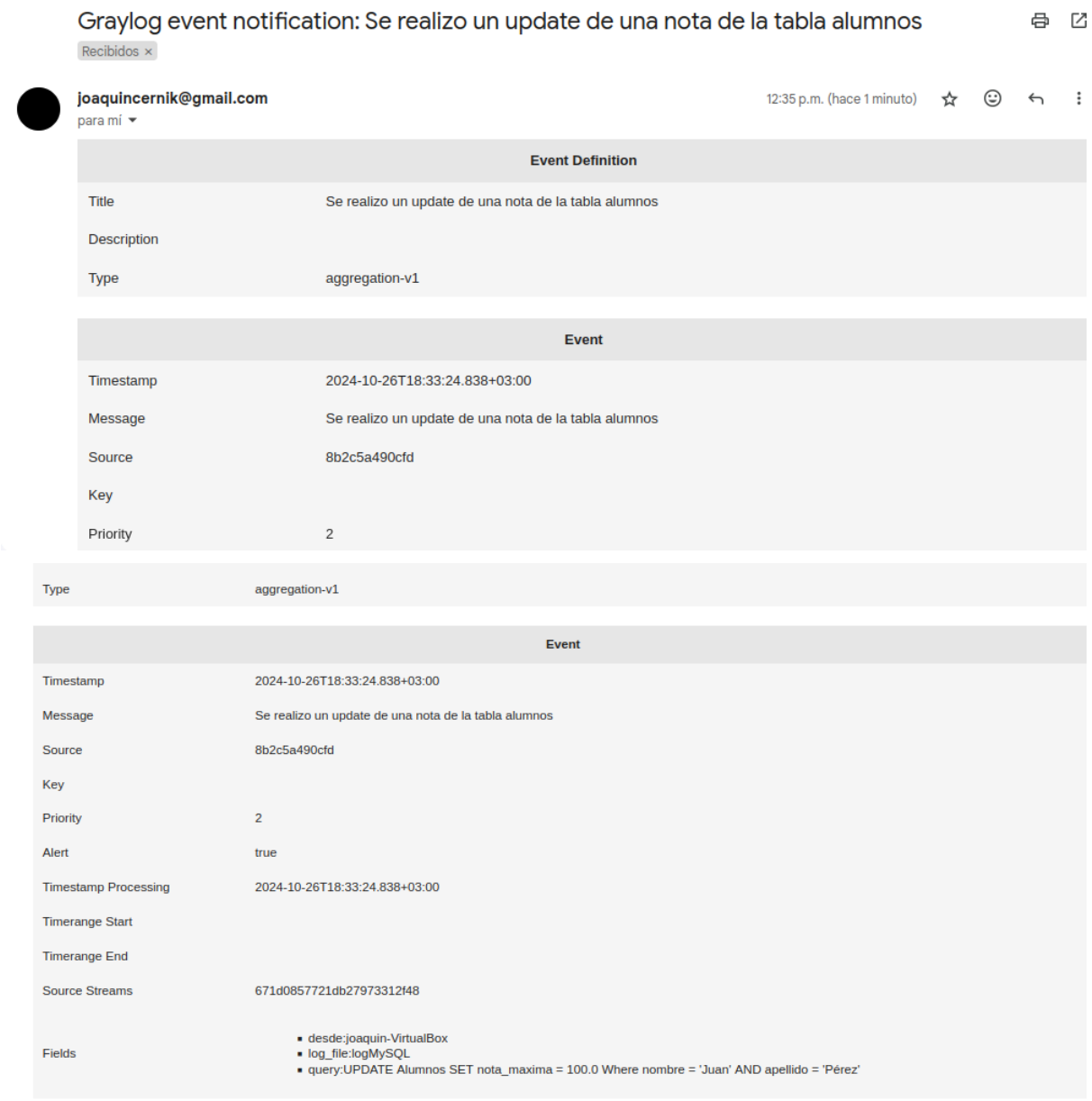


Nos quedo:



Ahora supongamos que somos Juan Perez:

```
mysql> UPDATE Alumnos SET nota_maxima = 100.0 Where nombre = 'Juan' AND apellido = 'Pérez' ;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```



Graylog event notification: Se realizo un update de una nota de la tabla alumnos

Recibidos ×

joaquincernik@gmail.com
para mí

12:35 p.m. (hace 1 minuto)

Event Definition	
Title	Se realizo un update de una nota de la tabla alumnos
Description	
Type	aggregation-v1

Event	
Timestamp	2024-10-26T18:33:24.838+03:00
Message	Se realizo un update de una nota de la tabla alumnos
Source	8b2c5a490cfd
Key	
Priority	2

Type	aggregation-v1

Event	
Timestamp	2024-10-26T18:33:24.838+03:00
Message	Se realizo un update de una nota de la tabla alumnos
Source	8b2c5a490cfd
Key	
Priority	2
Alert	true
Timestamp Processing	2024-10-26T18:33:24.838+03:00
Timerange Start	
Timerange End	
Source Streams	671d0857721db2797312f48
Fields	▪ desde:joaquin-VirtualBox ▪ log_file:logMySQL ▪ query:UPDATE Alumnos SET nota_maxima = 100.0 Where nombre = 'Juan' AND apellido = 'Pérez'

En este caso nos enteramos al instante de que algo puede estar andando mal con Juan 🧐!

Y con esto hemos concluido las dos alertas pedidas por la consigna.
Valga la aclaración que en la vida real se tendría que tomar mas precauciones como por ejemplo configurar que los logs de MySQL se reporten en una sola línea para trabajar correctamente, ya que debido que cuando se escribe una query en mysql usando \n, se los toma como diferentes logs, haciendo muy complicado la detección y manipulación de estos.
Y tambien tomar mucha precaución a la hora de realizar los streams y detectar cualquier acción debido a que lo que queremos capturar puede estar en mayúscula o en minúscula