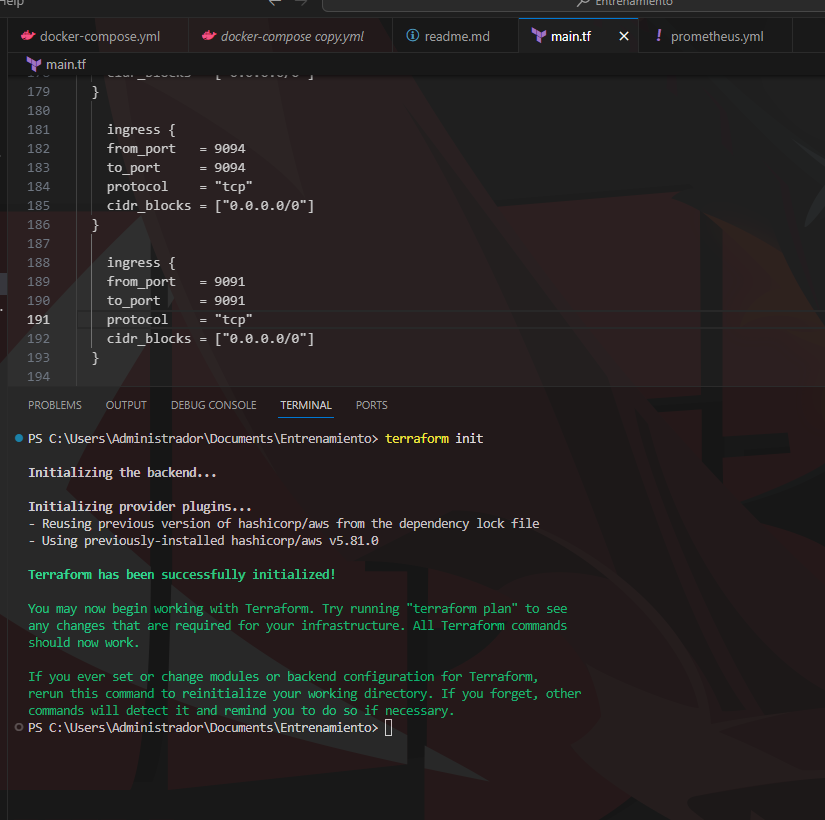
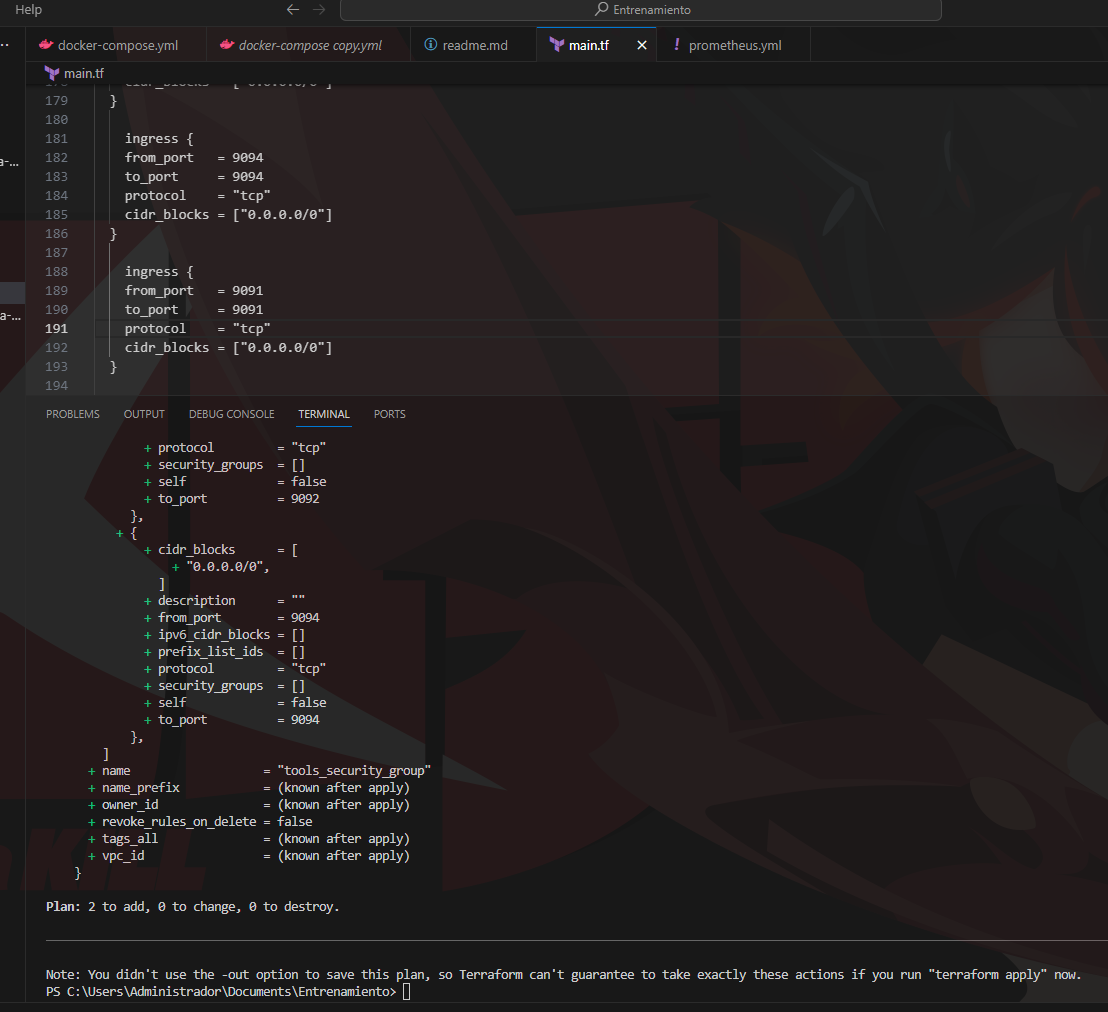
1. Prerrequisitos del sistema: Para Windows descargar terraform y realizar la instalación de variable de entorno, Cuenta de aws, crear un pair key .pem y llamarlo con el nombre dockerinstance.pem, descargar visual code o IDE de su preferencia, descarga MobaXterm
2. Descargar git:
3. Ejecutar los siguiente comandos:

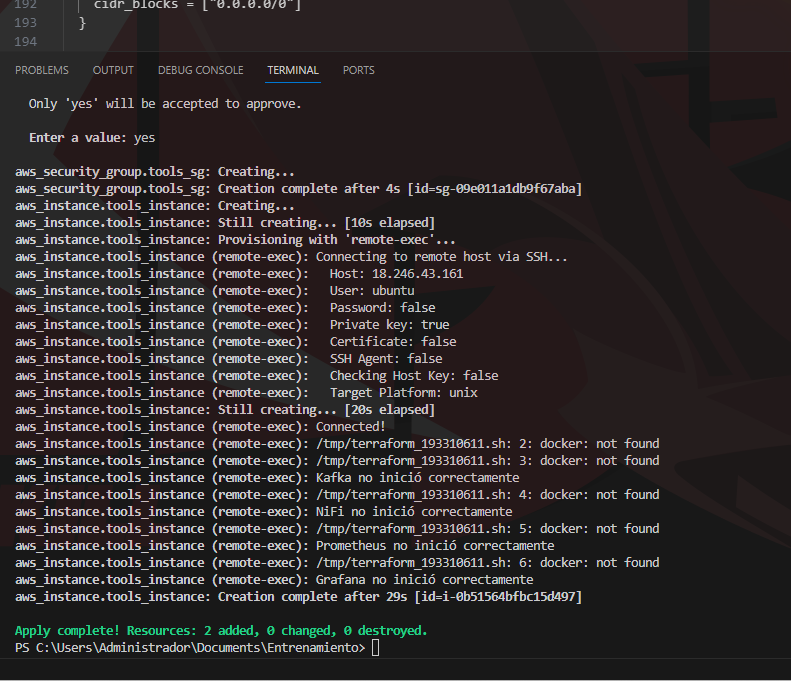
terraform init:



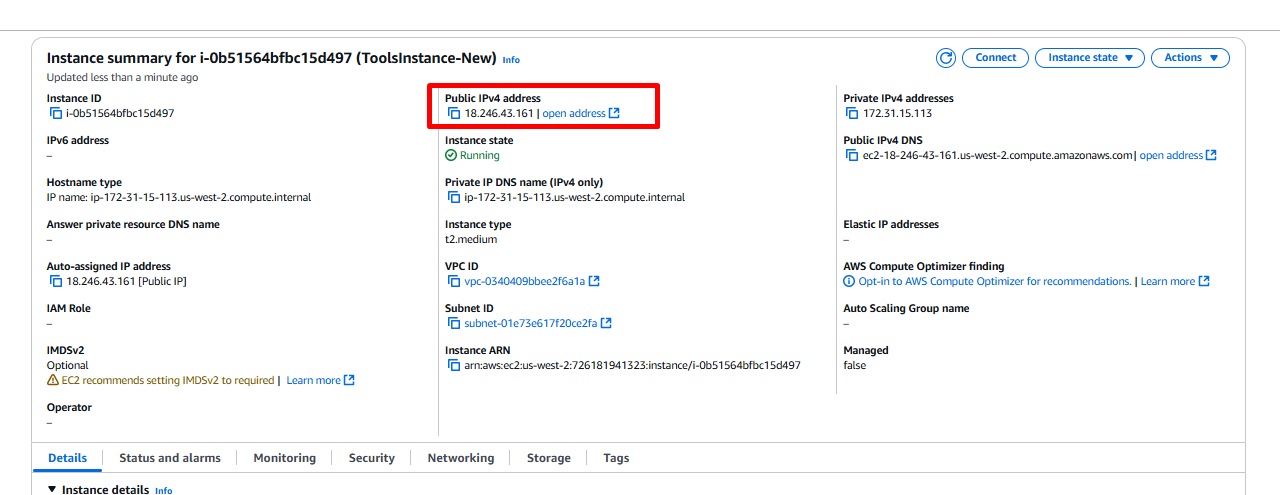
terraform plan



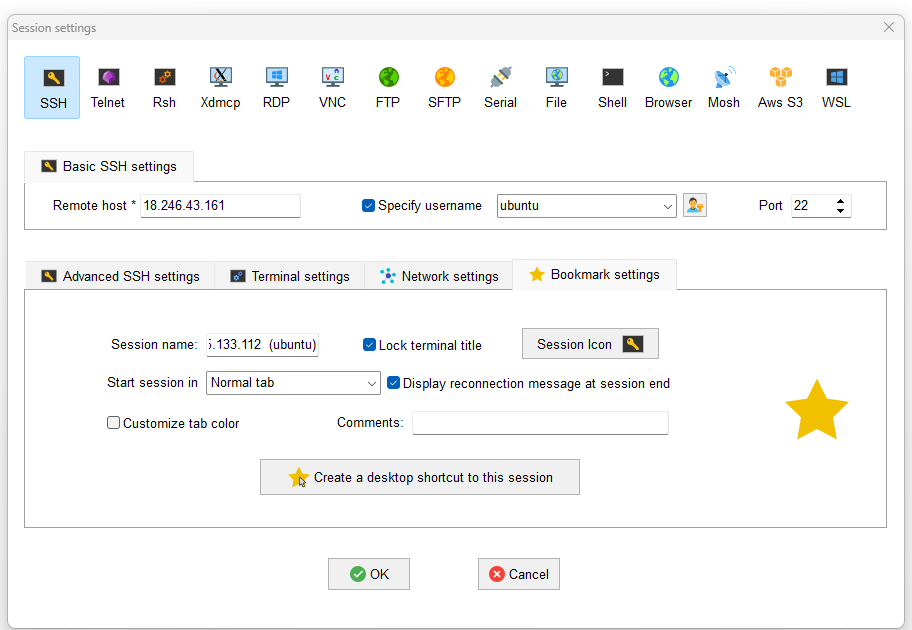
terraform apply



1. Validar la dirección publica la cual se creo



1. Configure MobaXterm



1. Ejecute el terminar estos permisos y cierre y abra sesión

permisos

cd tools

sudo chmod u+w docker-compose.yml

sudo chmod 777 docker-compose.yml

sudo chown ubuntu:ubuntu docker-compose.yml

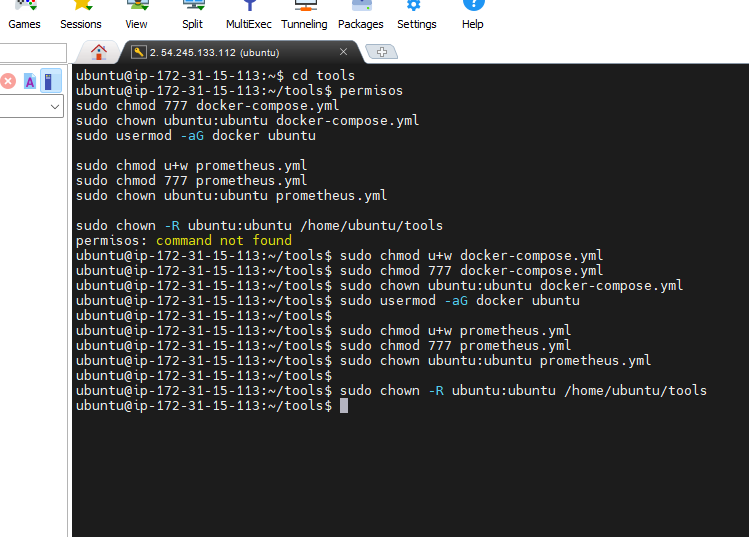
sudo usermod -aG docker ubuntu

sudo chmod u+w prometheus.yml

sudo chmod 777 prometheus.yml

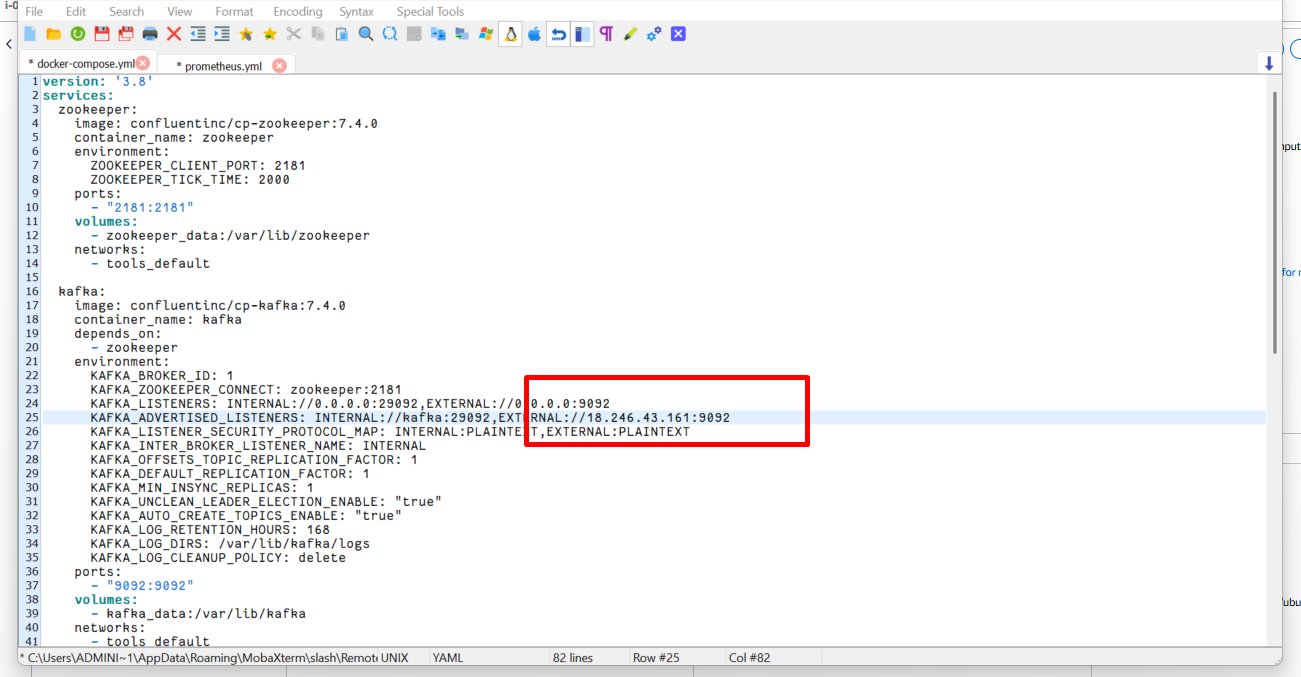
sudo chown ubuntu:ubuntu prometheus.yml

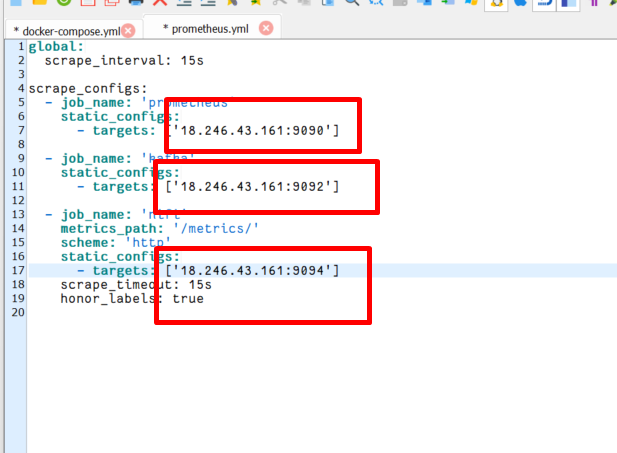
sudo chown -R ubuntu:ubuntu /home/ubuntu/tools



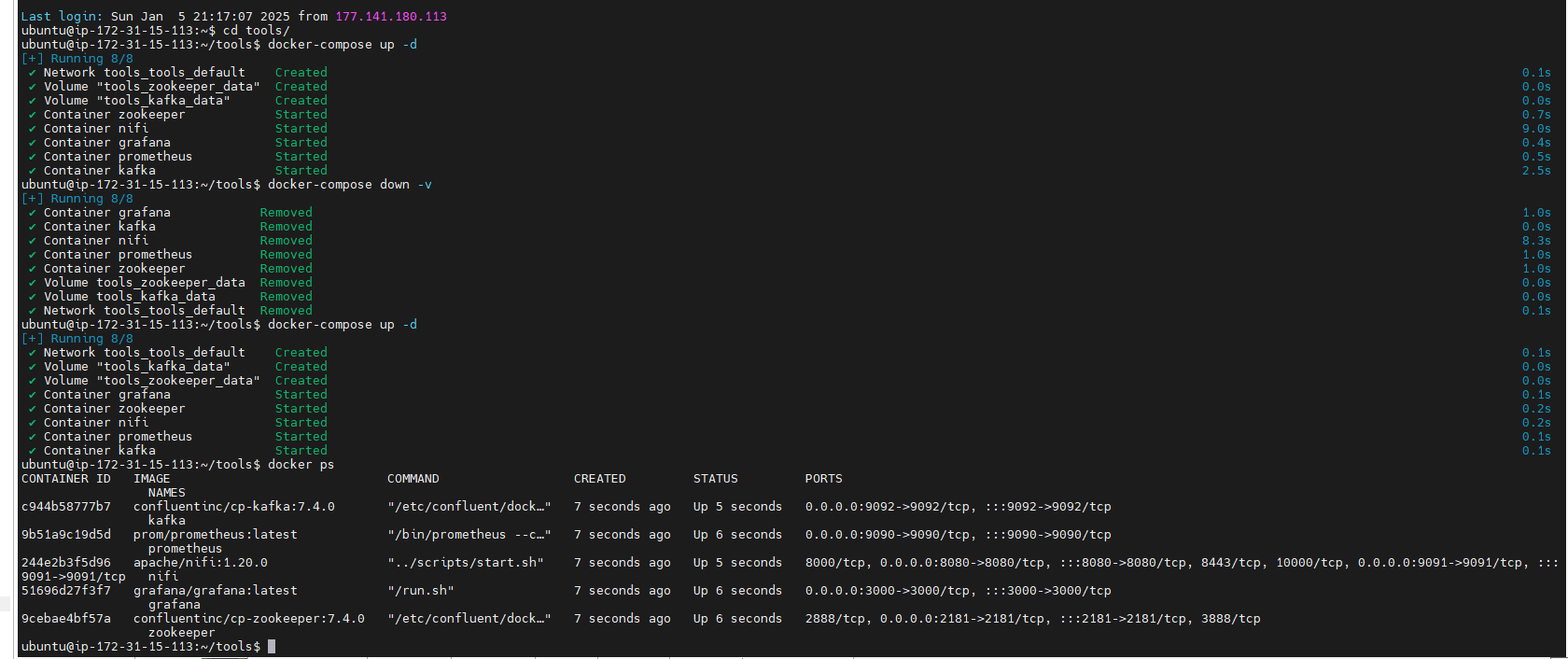
Cambie lós valores el contenido que esta en la carpeta de tools de los archivos yml de prometheus.yml y docker-compose.yml por los que están en git

Valide siempre que tenga la ip publica que genero en aws





1. Dentro la carpeta tools ejecutar el comando docker-compose up –d



Valide con el comando docker ps o docker ps –a que lo contenedores si estén funcionando

1. Vamos a crear el proxy inverso con el siguiente comando proxy inverso

sudo apt update

sudo apt install nginx

1. Modifica el proxy con el siguiente comando sudo nano /etc/nginx/sites-available/nifi\_proxy

server {

listen 9094;

location /metrics/ {

proxy\_pass http://127.0.0.1:9091/metrics/; # Apunta al puerto correcto

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

# Forzar el tipo de contenido

add\_header Content-Type "text/plain; version=0.0.4";

}

# Evitar errores de favicon

location = /favicon.ico {

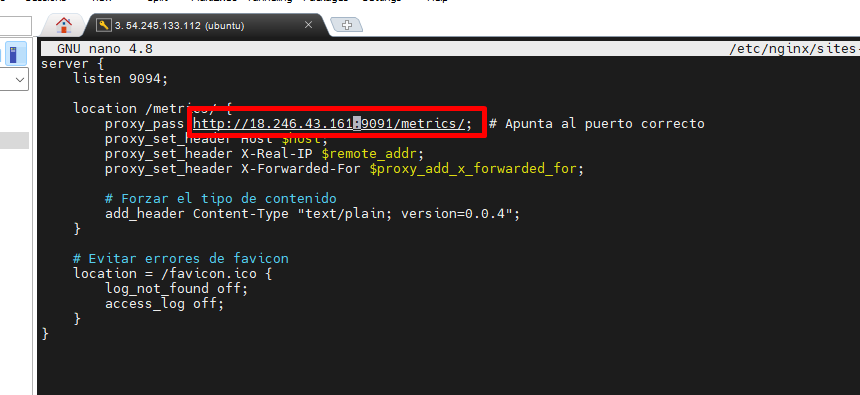
log\_not\_found off;

access\_log off;

}

}

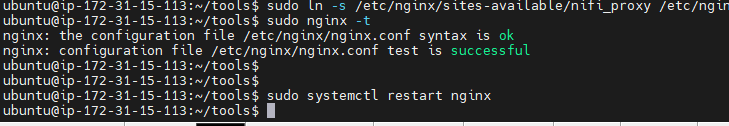
Vale que quede la ip publica que genero aws



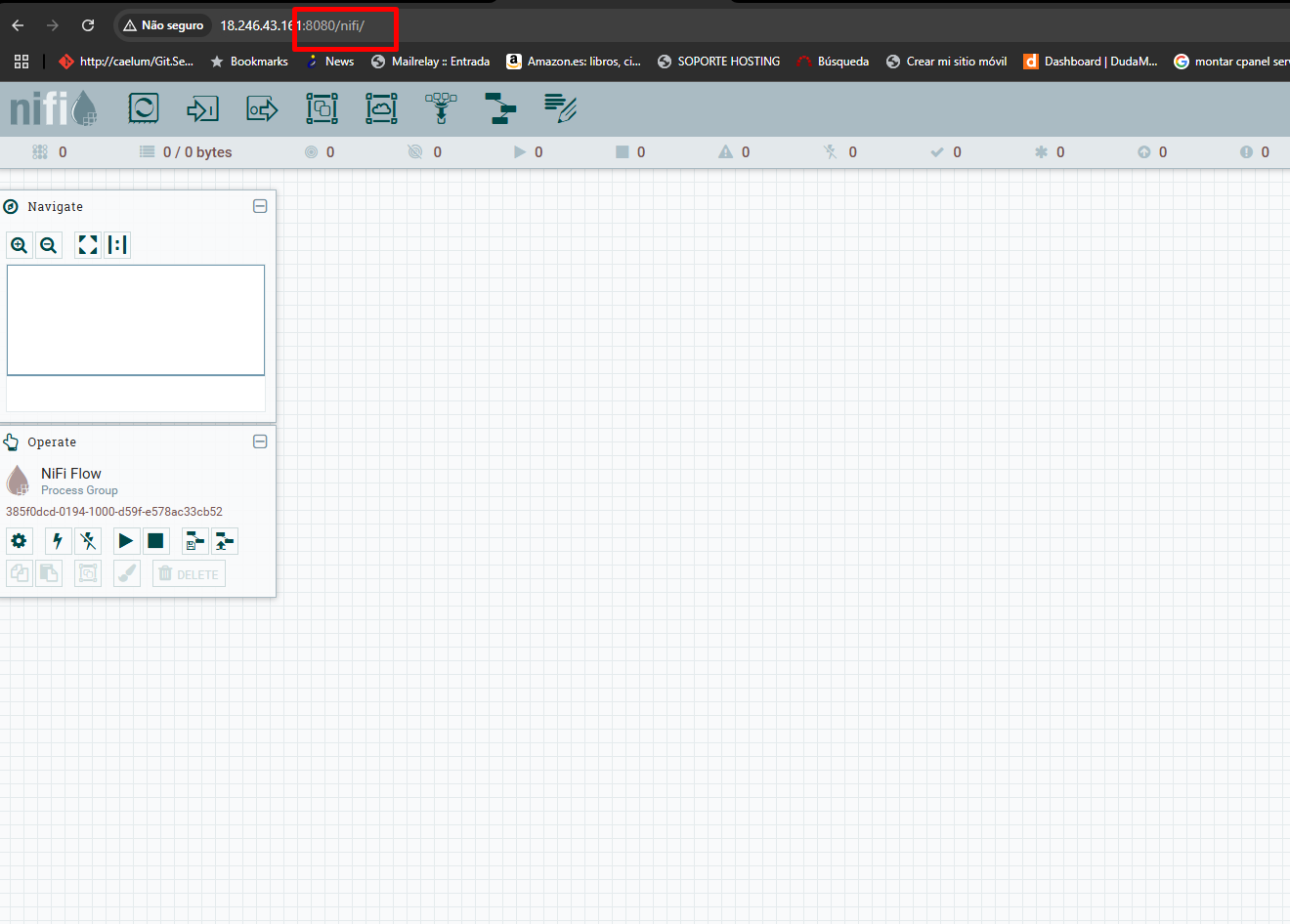
1. Ejecute el siguiente comando para comience a funcionar el proxy reverso sudo ln -s /etc/nginx/sites-available/nifi\_proxy /etc/nginx/sites-enabled/
2. Seguido ejecute los siguiente comandos

sudo nginx -t

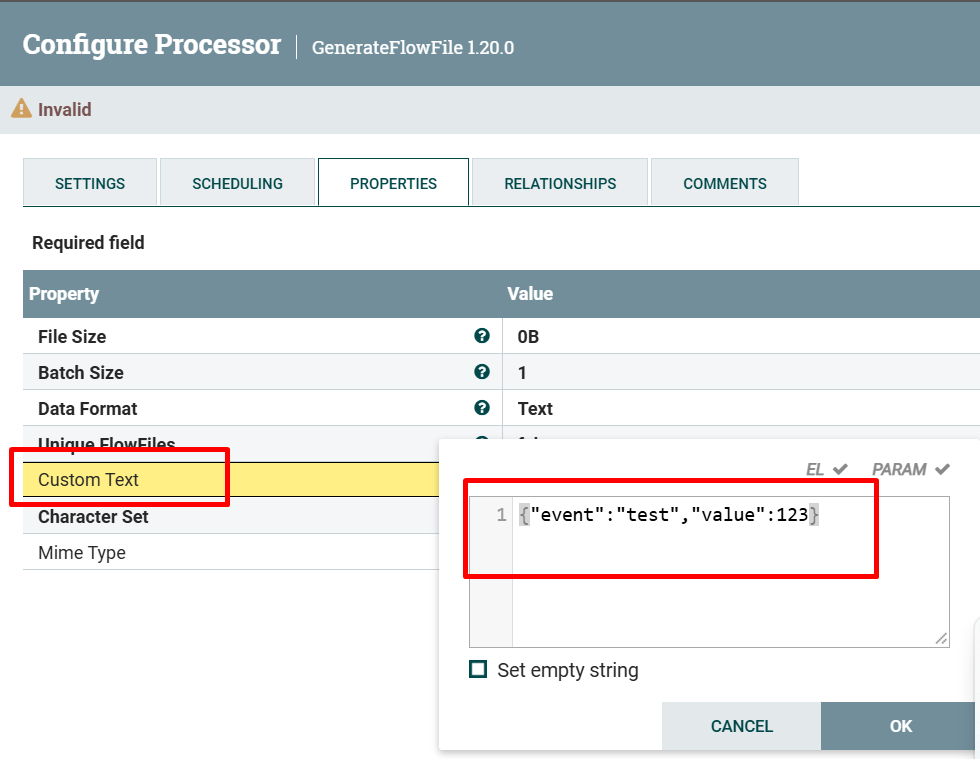
sudo systemctl restart nginx



1. Ingresa a nifi por el puerto 8080

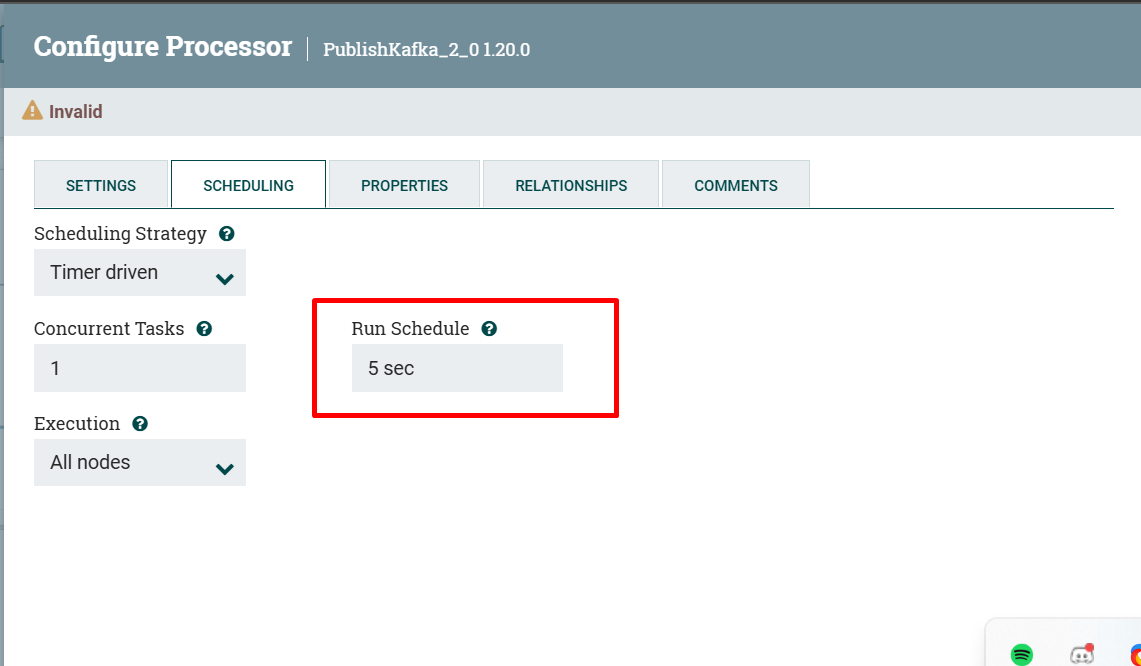


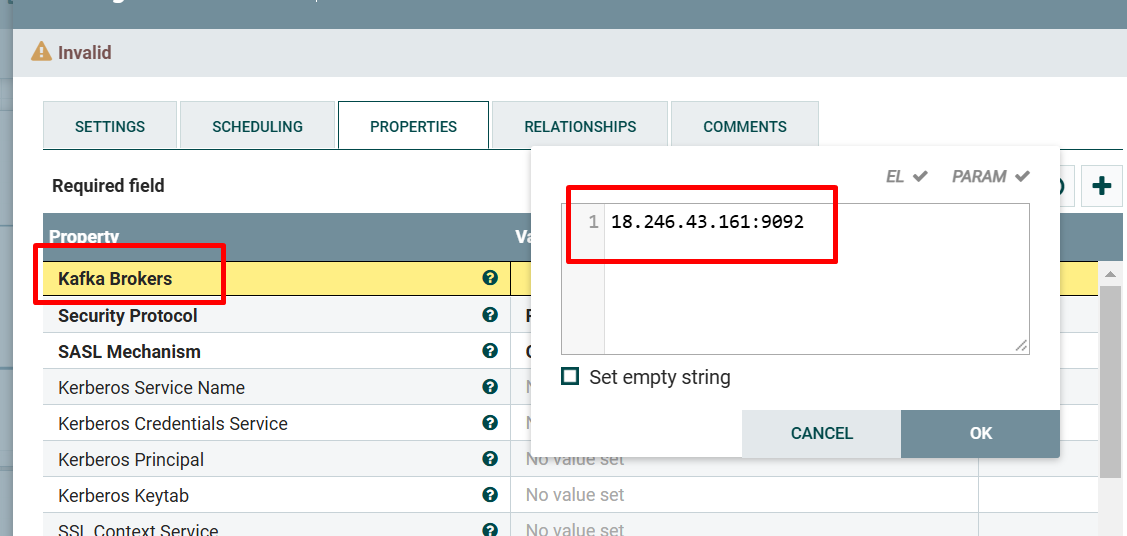
13 crear los siguiente procesos

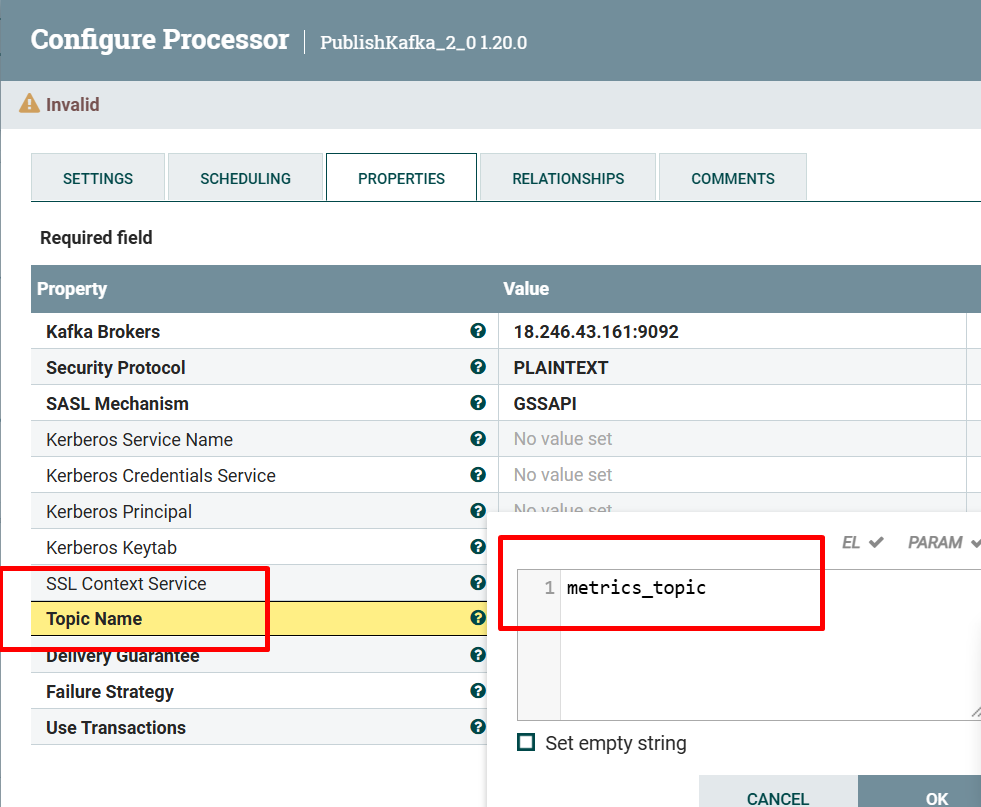


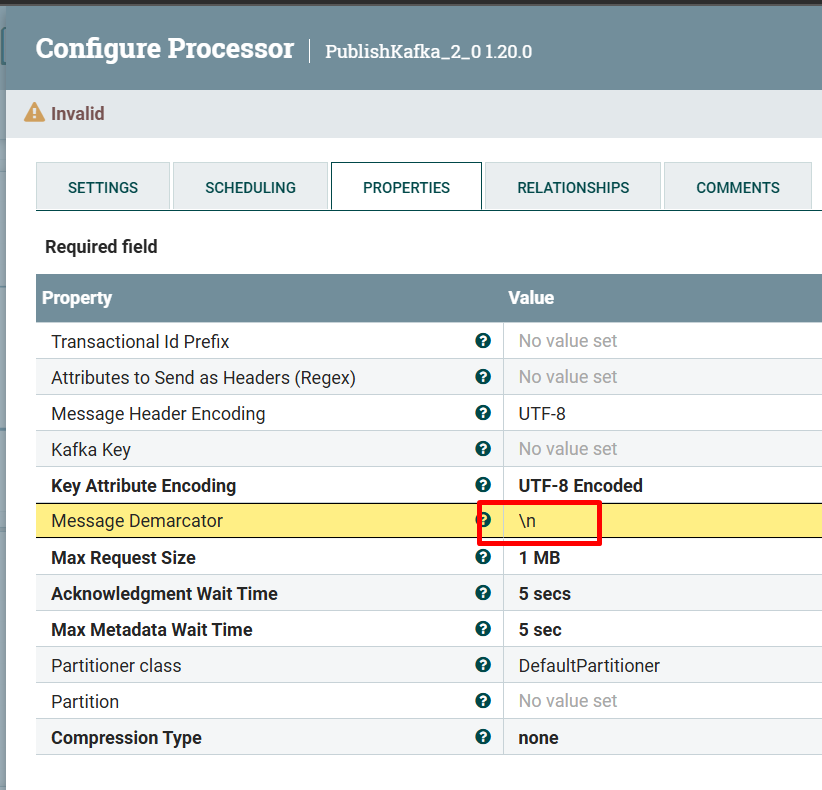
Custom Text: {"event":"test","value":123}

1. Modificar el tiempo

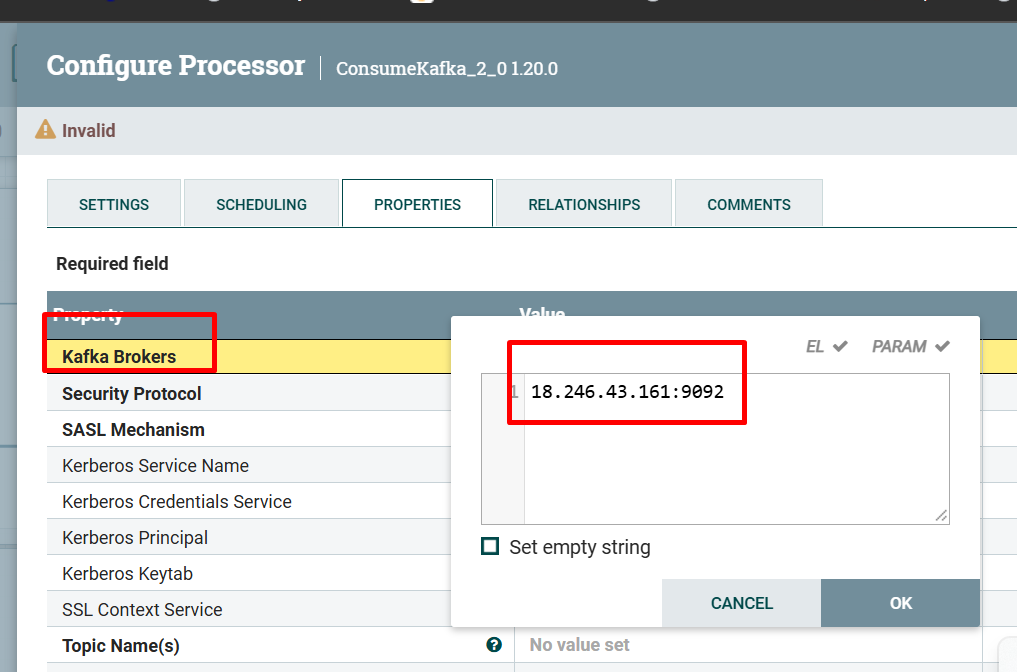


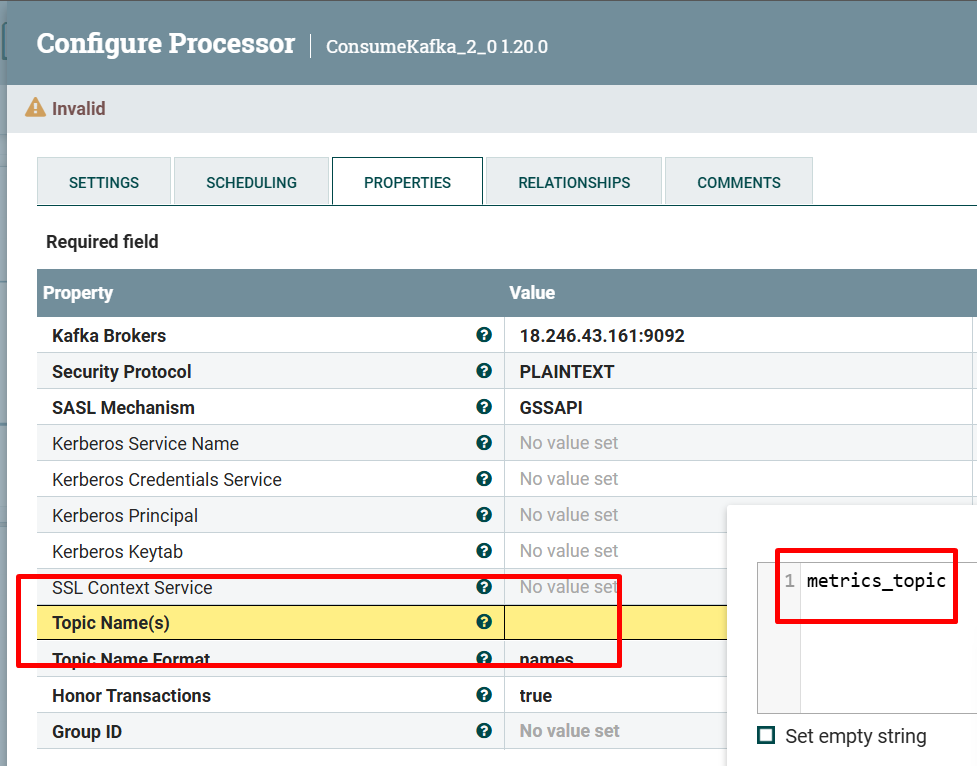


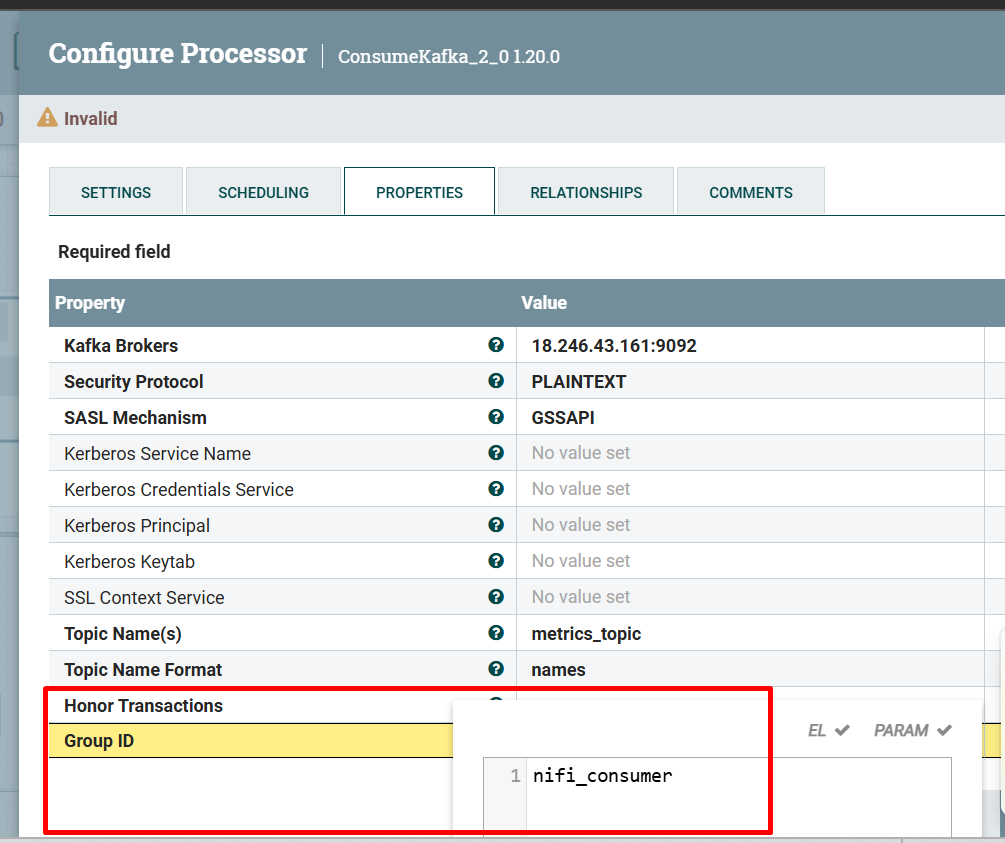




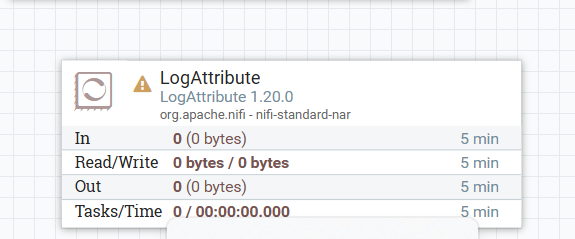
1. Configuracion de proceso consumerKafka\_2\_0



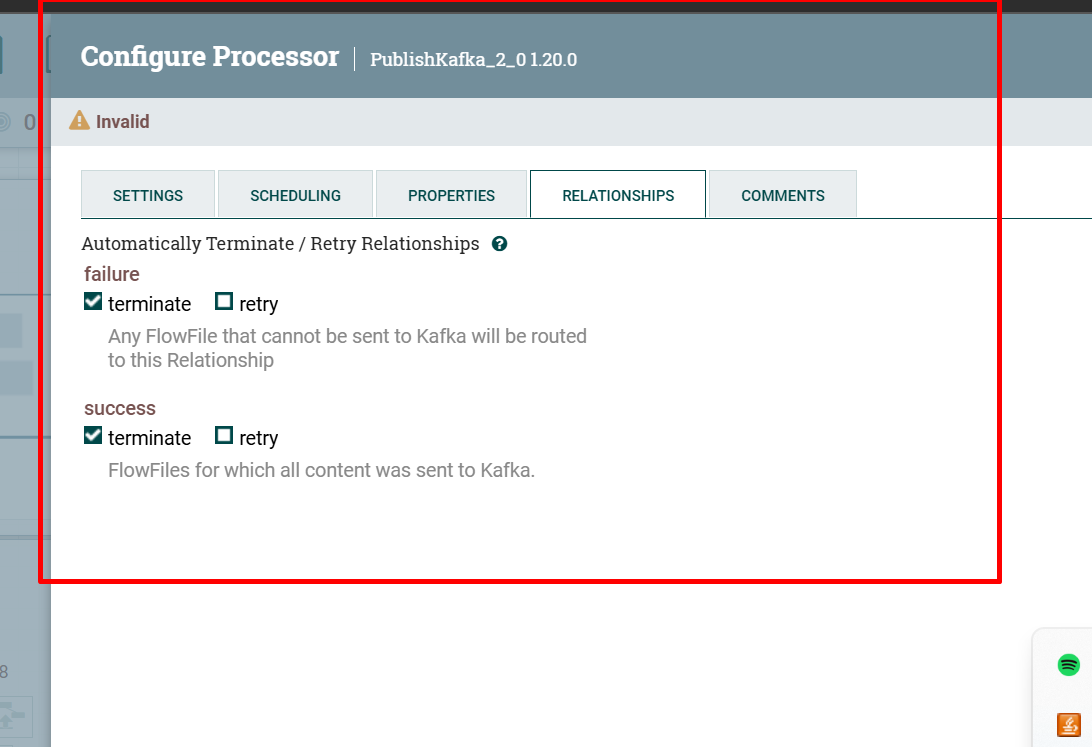




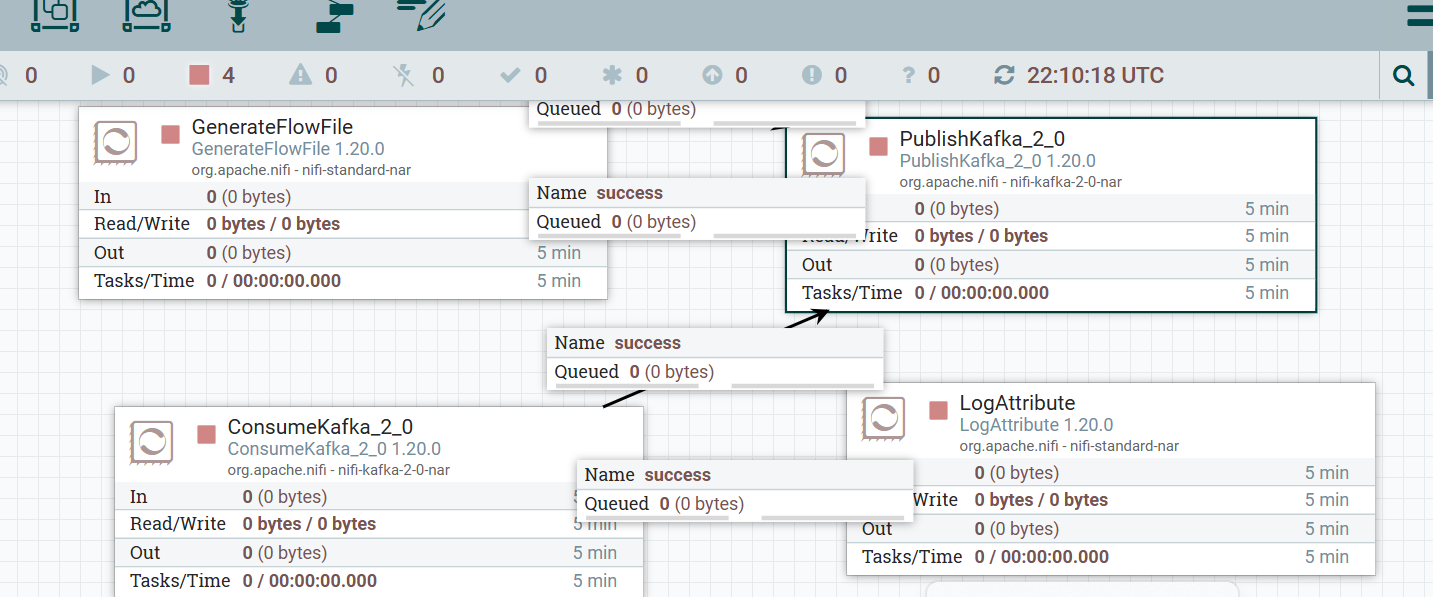
1. Este proceso se deja por defecto



1. Conecte lo flujos y para quitar las advertencias solo ingresar a relationships y colocar en procesos terminate

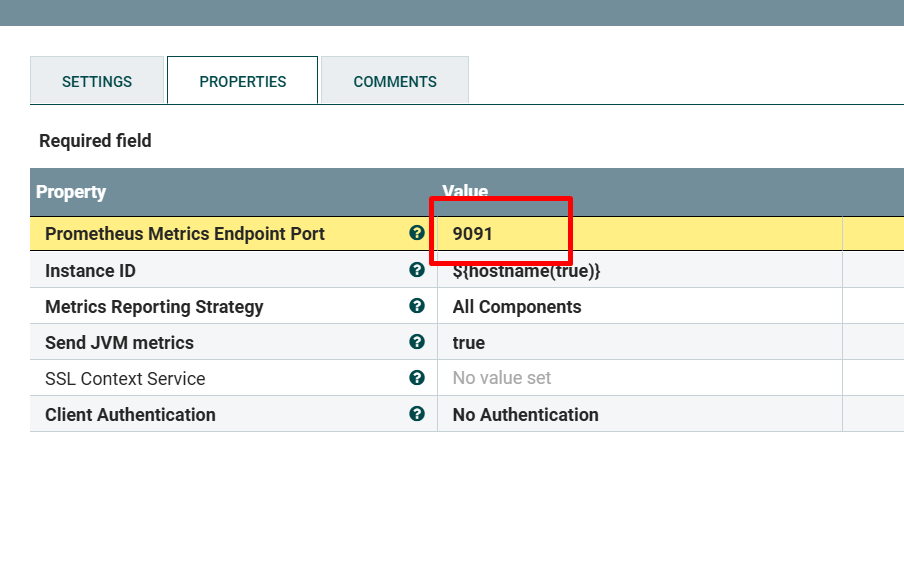


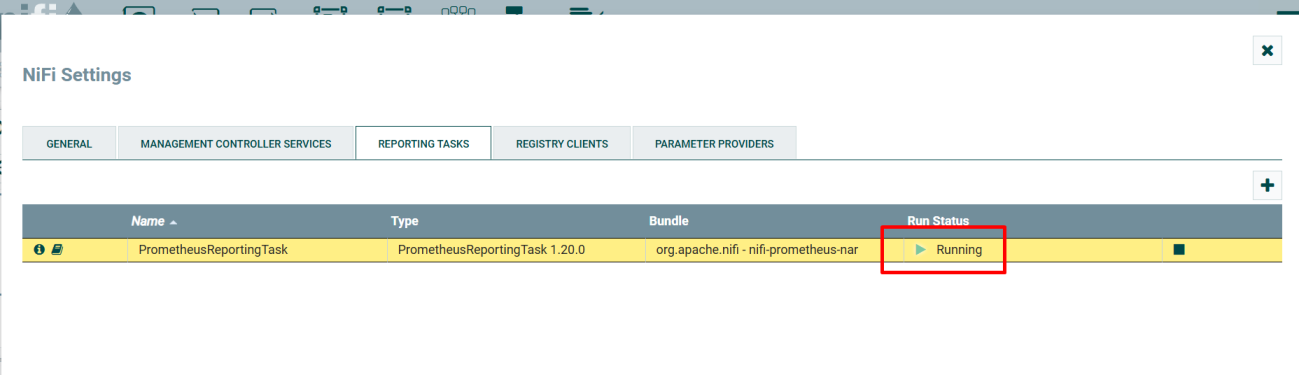
1. Todos ya deben estar conectados y en rosado listo para ejecutarse



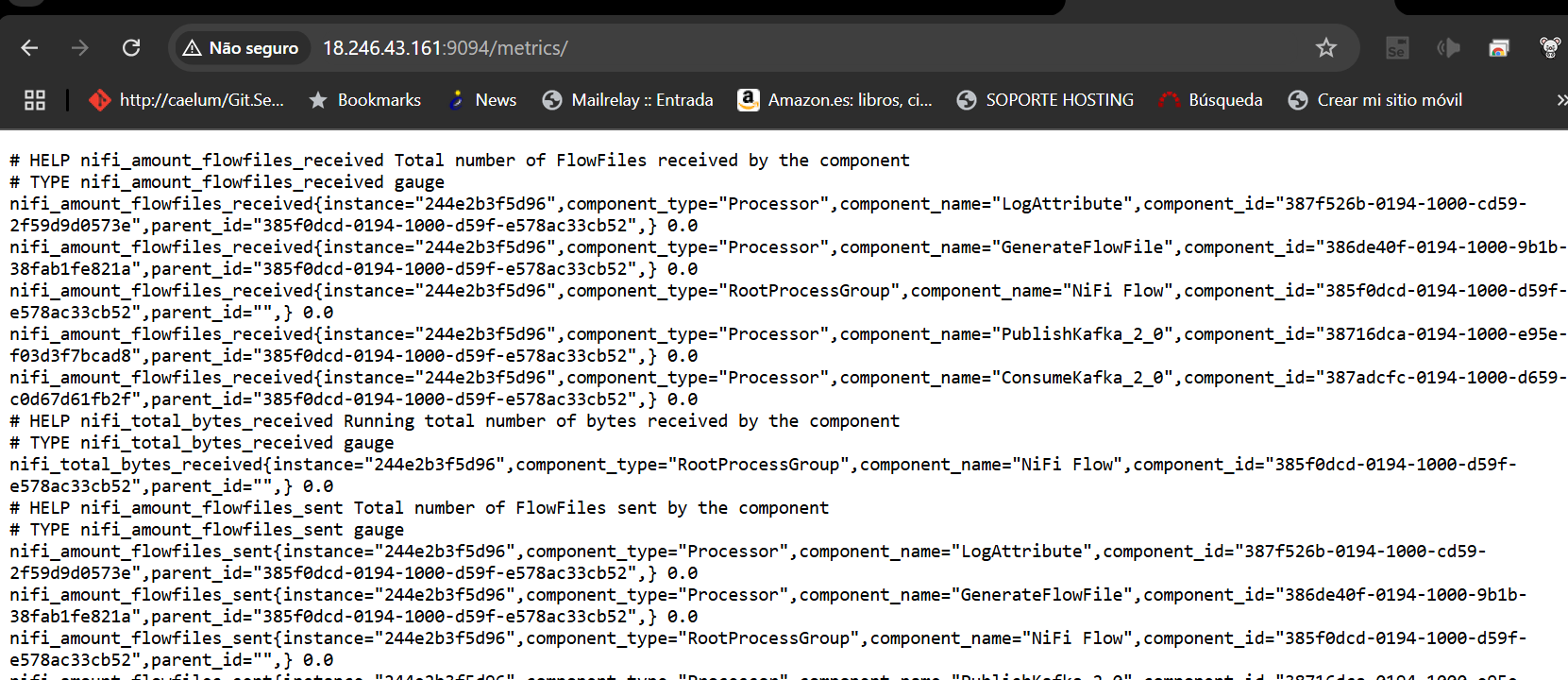
1. Configurar el reporte prometheus



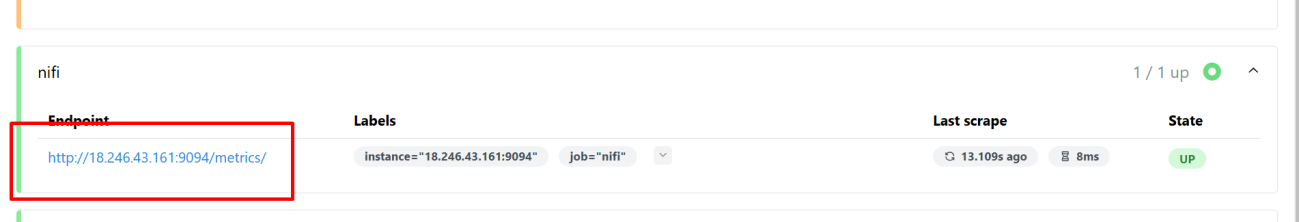




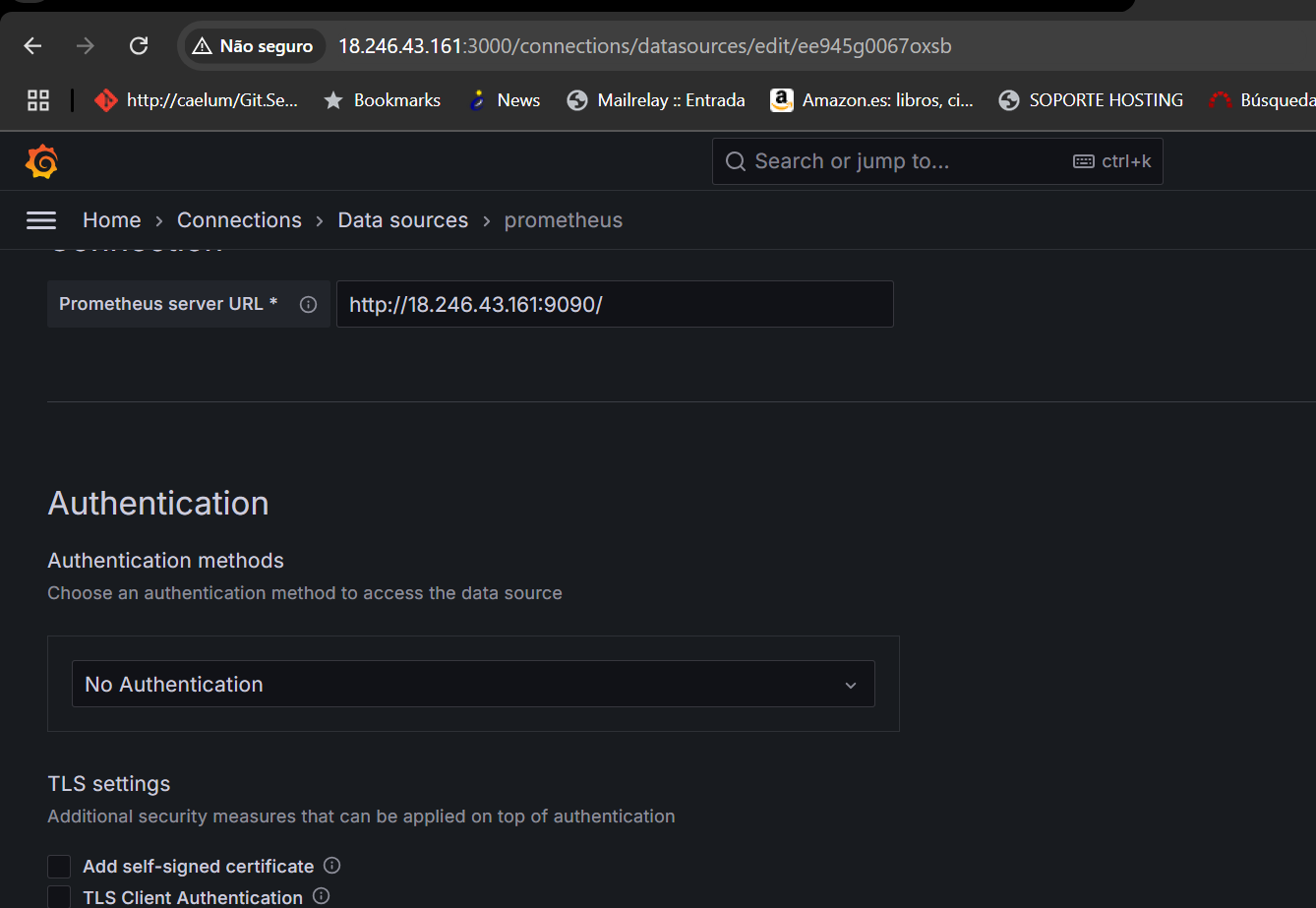
1. Probar por url que el servicio este arriba desde el proxy reverso puerto 9094



Validar que prometheus este en UP para el puerto 9094



1. Entrar al grafanna puerto 3000 y empezar a configurar las conexiones con prometheus y graficas de nifi

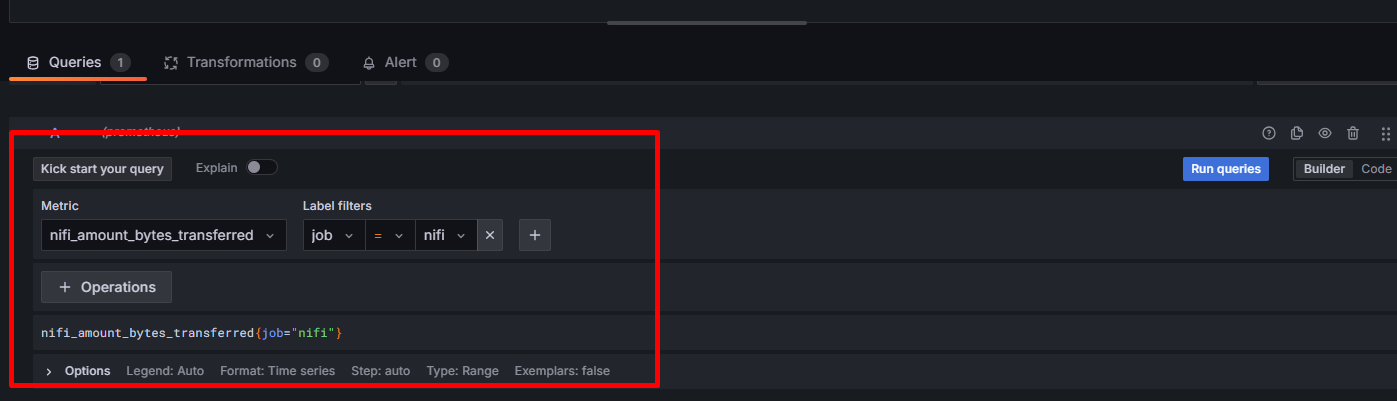


1. Configura un nuevo dashboard y graficar las siguiente varibles

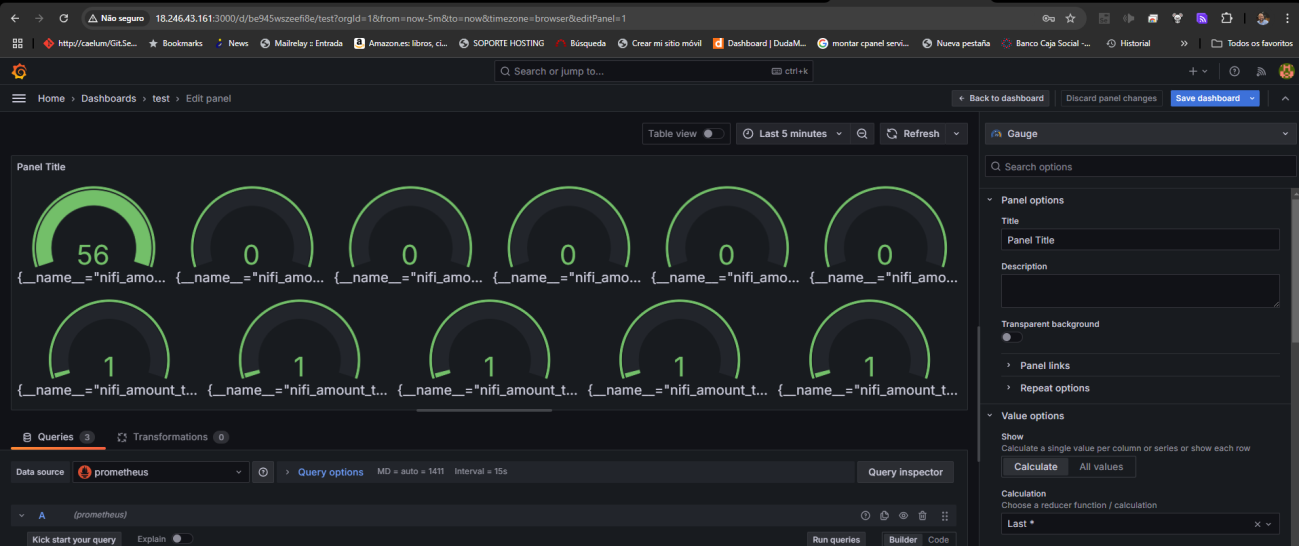
nifi\_amount\_flowfiles\_transferred → Total de FlowFiles transferidos.

nifi\_amount\_bytes\_sent → Bytes enviados.

nifi\_amount\_threads\_active → Hilos activos.



1. Por ultimo el grafico que de mas sentido





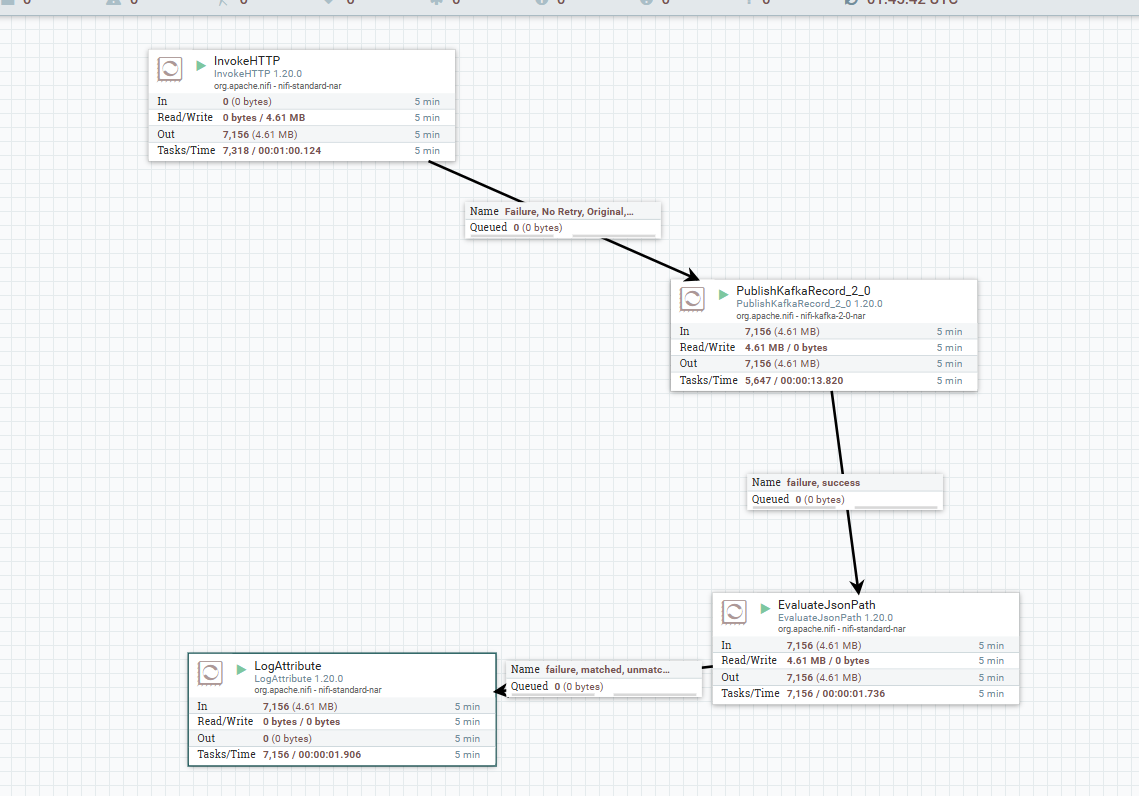
23 si quiere valiar los datos que se están autogenerado se puede realizar desde la línea de comandos

docker exec -it kafka kafka-console-consumer --bootstrap-server 35.93.3.72:9092 --topic metrics\_topic --from-beginning

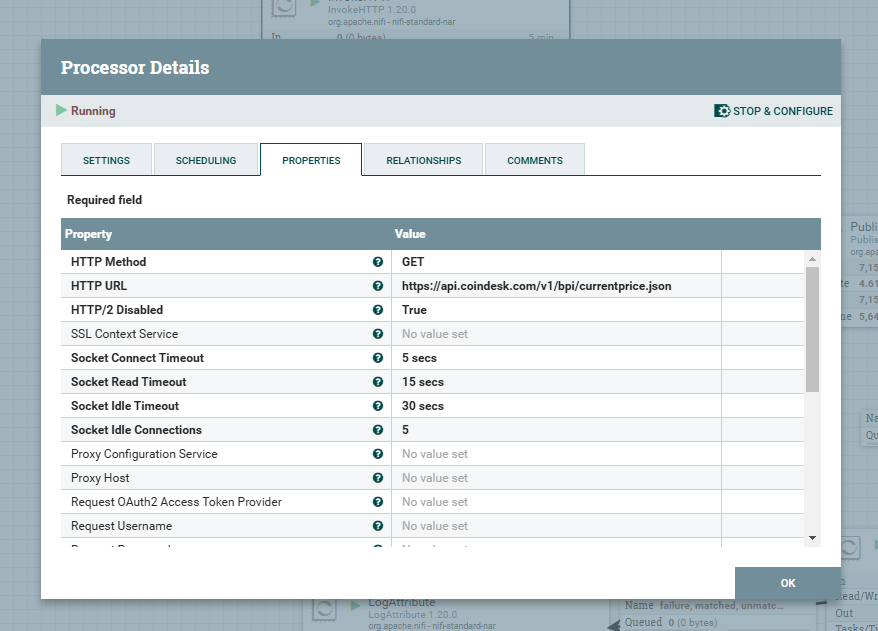
Ejercio numero 2

**Coindesk API (Precio de Bitcoin)**

* **Descripción:** API pública para obtener información sobre el precio actual de Bitcoin.

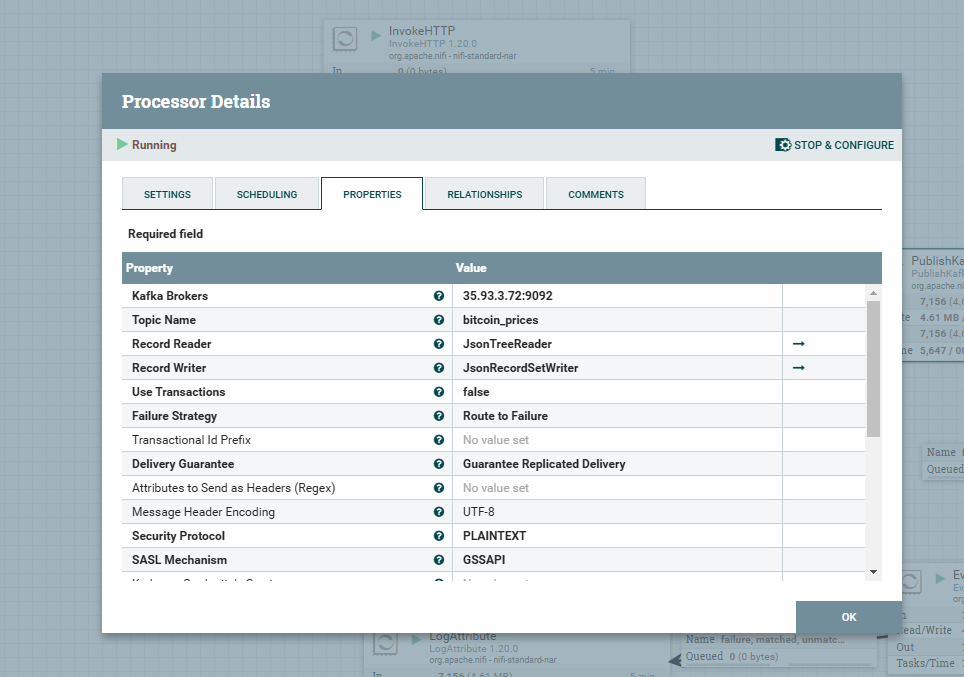


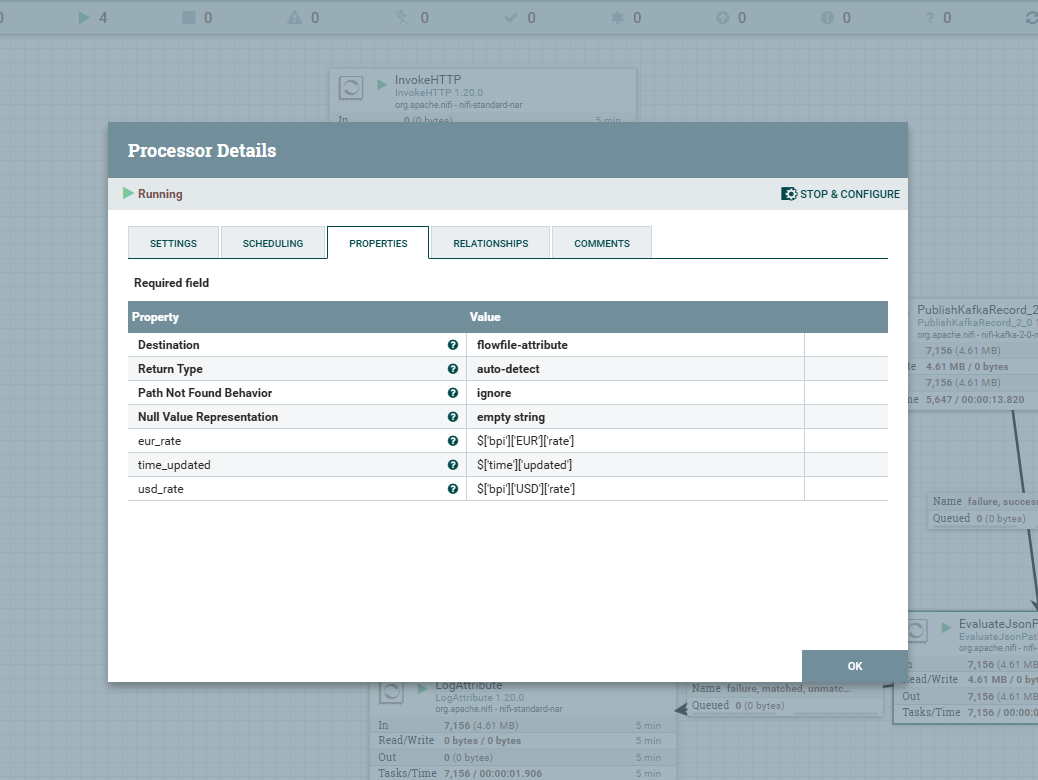
https://api.coindesk.com/v1/bpi/currentprice.json



bitcoin\_prices

use transactions: false

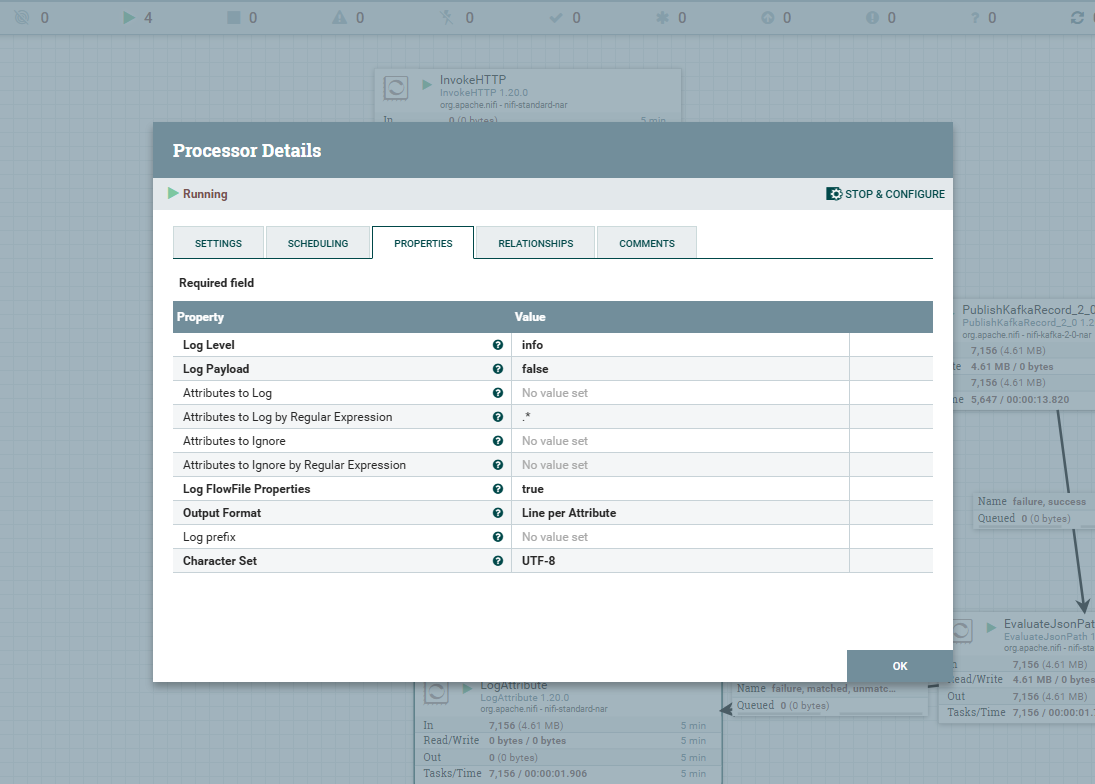


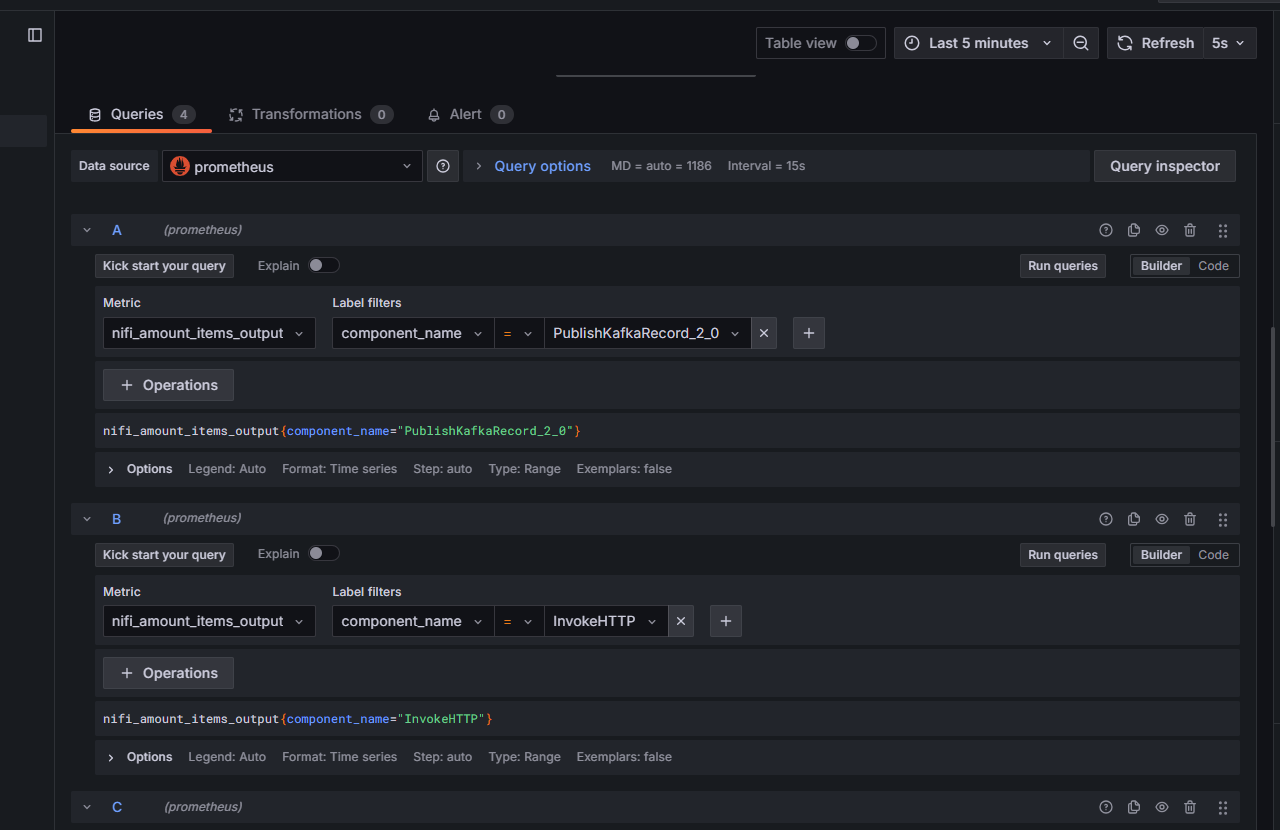


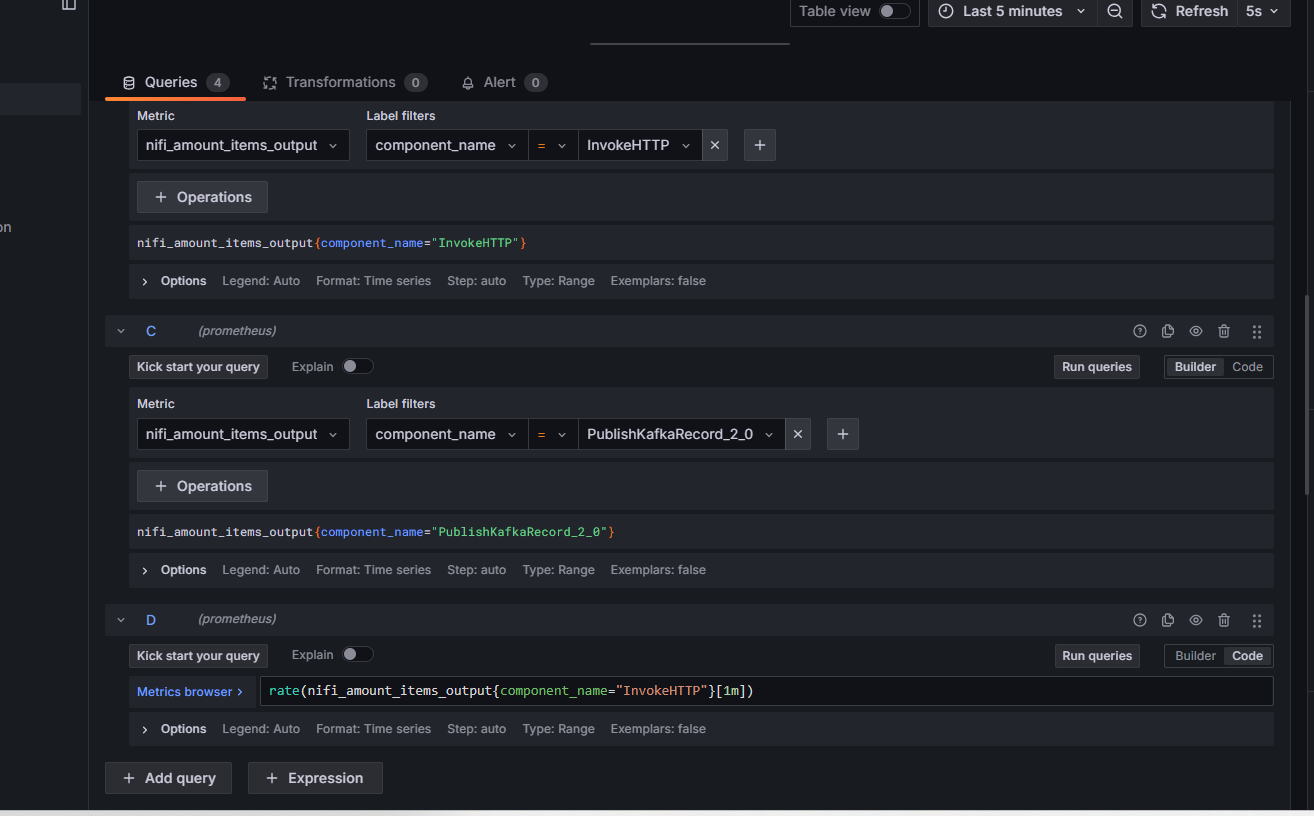
eur\_rate: $['bpi']['EUR']['rate']

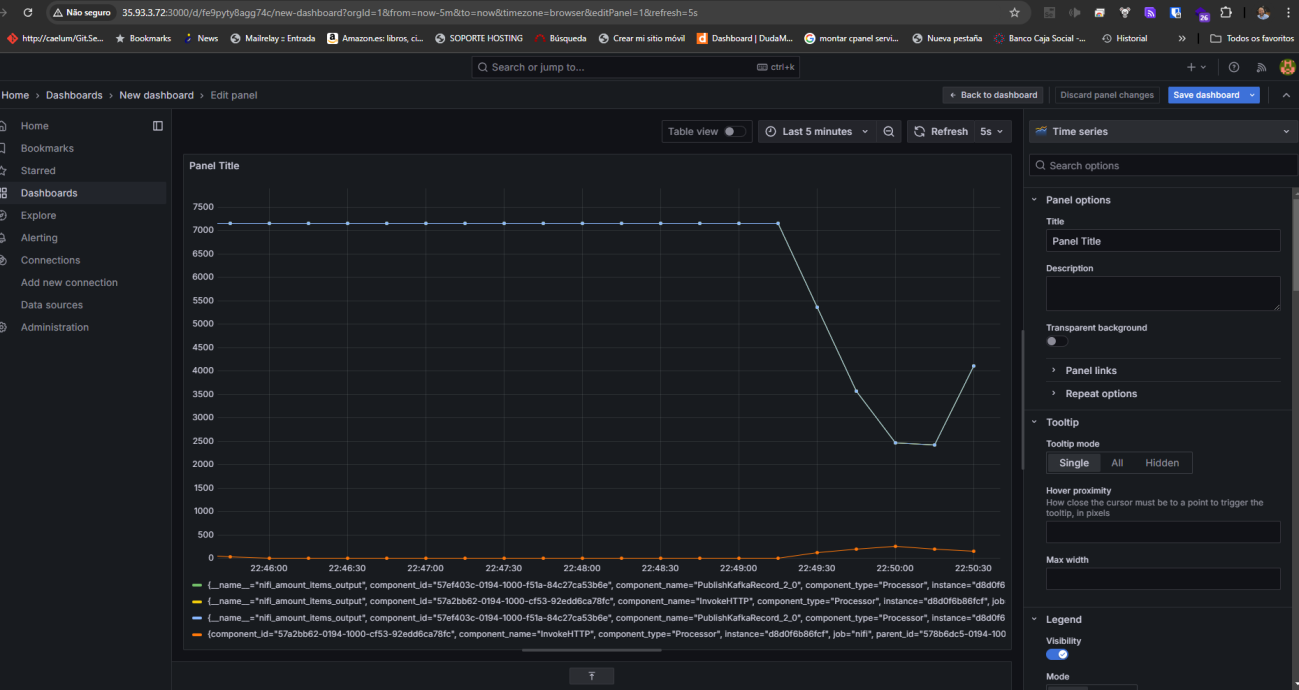
time\_updated: $['time']['updated']

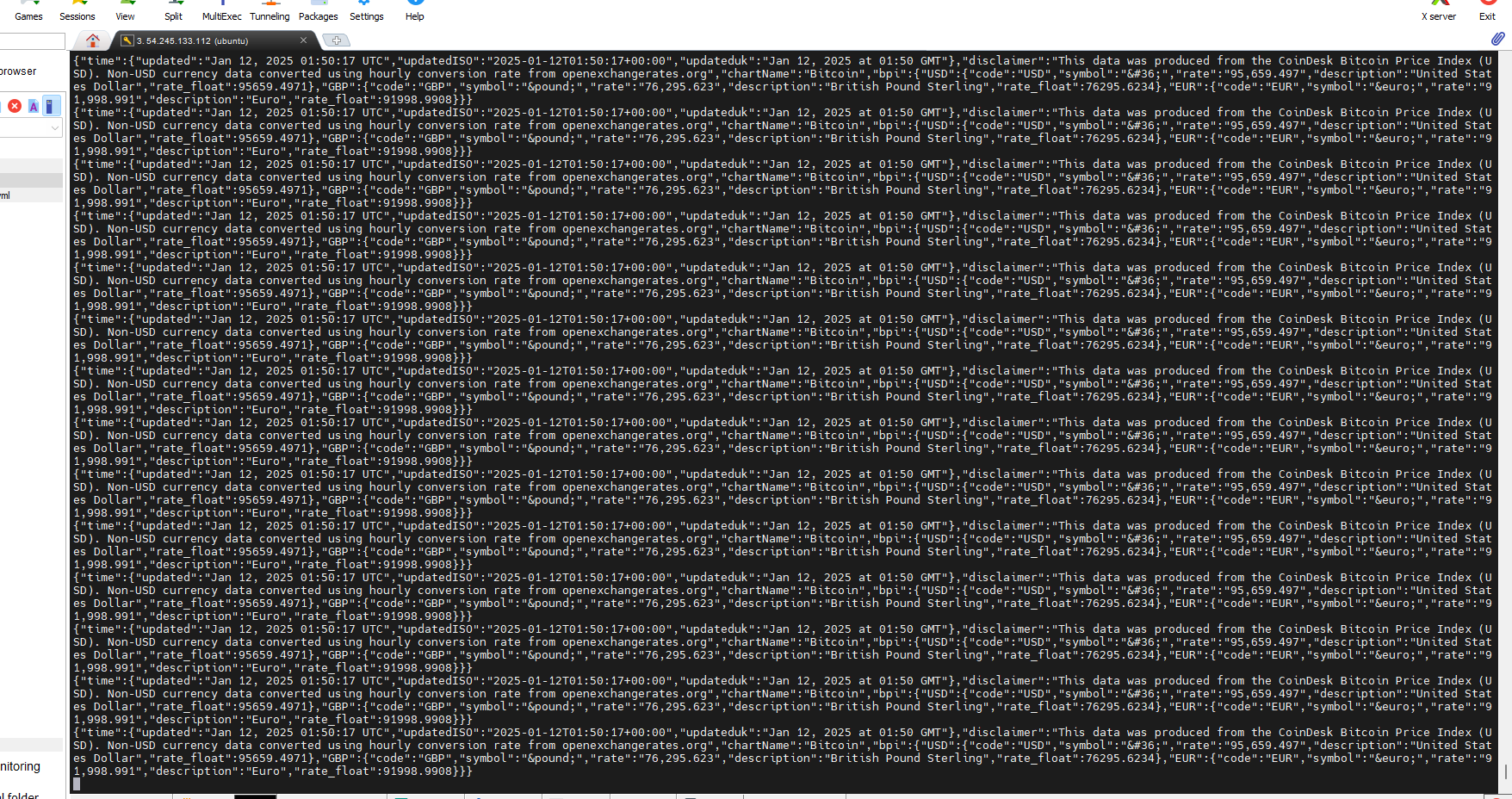
usd\_rate: $['bpi']['USD']['rate']





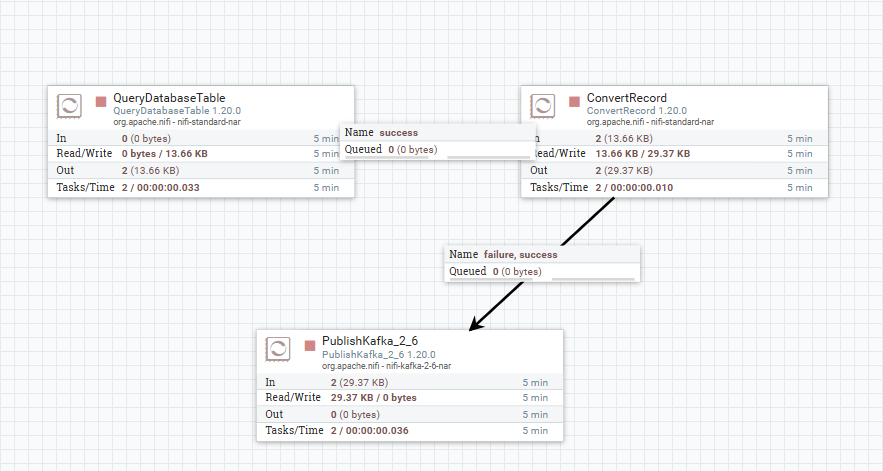






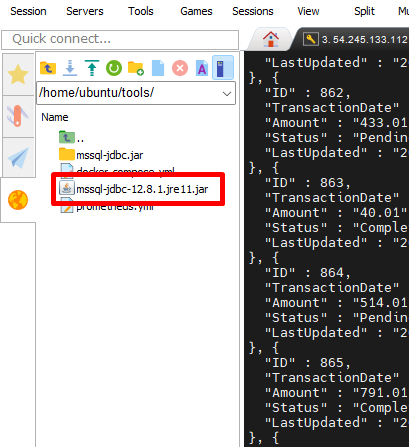
Extraer Datos de directamente de una base de datos

2.4 En el siguiente print el objetivo es crear los siguiente proccesor

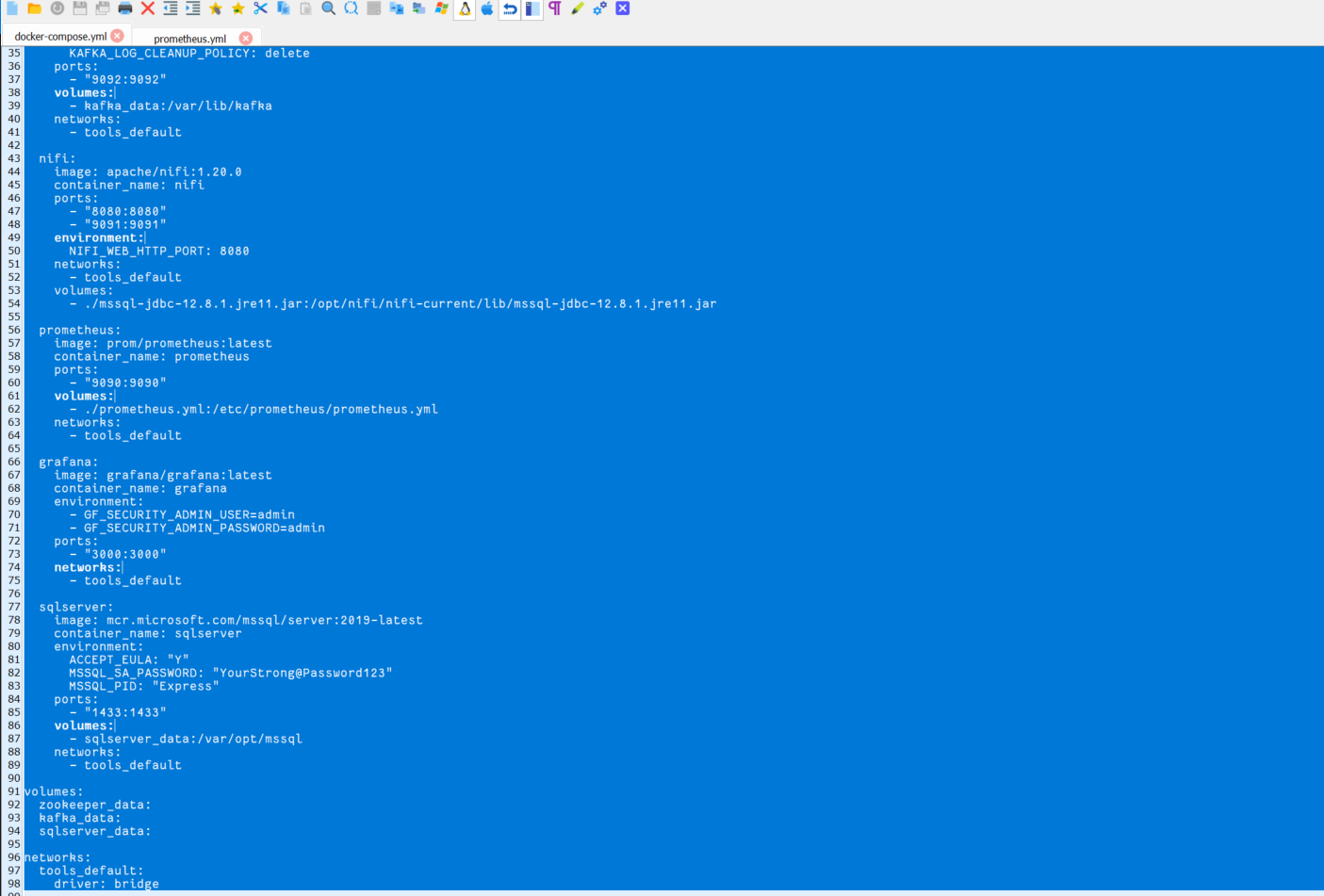


2.5 Antes de comenzar debemos configura el jdbc en nuestro ambiente y se realizara de la siguiente manera descarga el jdbc de la siguiente pagina: <https://learn.microsoft.com/es-es/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver16>

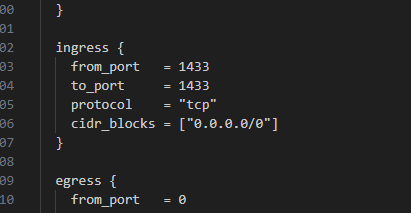
2.6 Copiamos la versión de java en la raíz de la carpeta tools



2.7 Eliminamos el ambiente actual si esta arriba si no solo actualizamos el docker-compose.yml con el nuevo archivo docker-compose\_bd.yml solo su contenido para poder instalar el sql server express en nuestro entorno



2.7 Valida si el puerto está habilitado en nuestro ambiente aws el 1433, igual en nuestro terraform antes de subir el ambiente podemos agregar la siguiente línea para que suba la configuración del puerto sin embargo estare actualizando el script



2.8 Utiliza cualquier cliente de preferencia de SQL Server y nos conectamos a la instancia y ejecutamos el siguiente query:

-- Crear la base de datos

**CREATE** **DATABASE** TestDB;

-- Usar la base de datos

**USE** TestDB;

-- Crear la tabla de ejemplo

**CREATE** **TABLE** Transactions (

ID **INT** **IDENTITY**(1,1) **PRIMARY** **KEY**,

TransactionDate **DATETIME** **NOT** **NULL**,

Amount **DECIMAL**(10, 2) **NOT** **NULL**,

Status **NVARCHAR**(50) **NOT** **NULL**,

LastUpdated **DATETIME** **DEFAULT** **GETDATE**()

);

-- Insertar datos iniciales

**INSERT** **INTO** Transactions (TransactionDate, Amount, Status)

**VALUES**

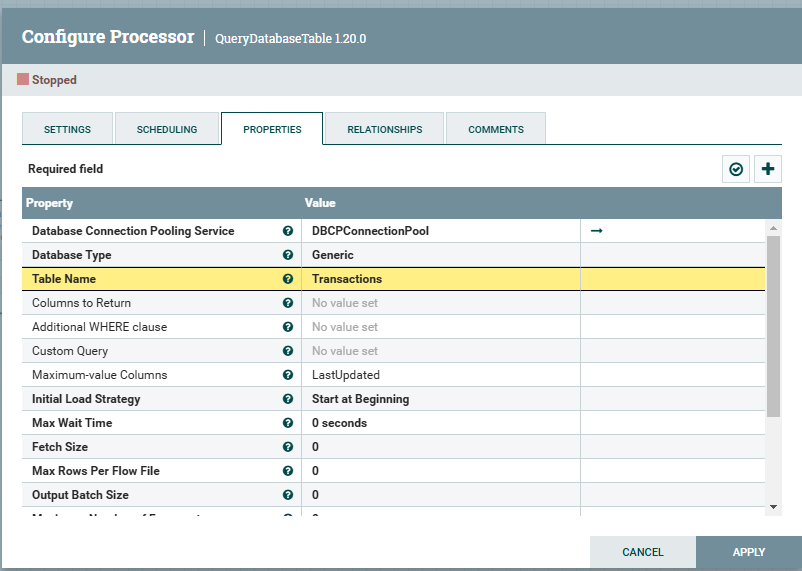
(**'2025-01-01 10:00:00'**, 100.50, **'Completed'**),

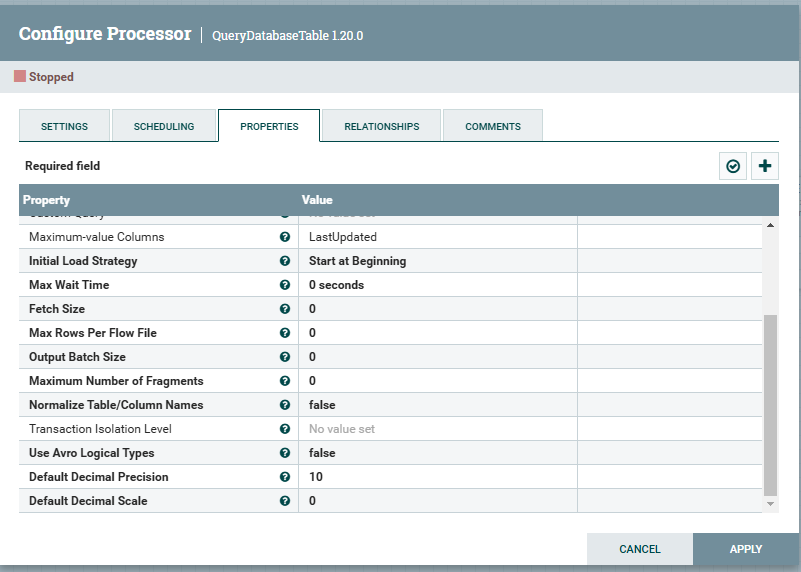
(**'2025-01-02 11:30:00'**, 200.00, **'Pending'**),

(**'2025-01-03 14:45:00'**, 150.75, **'Completed'**);

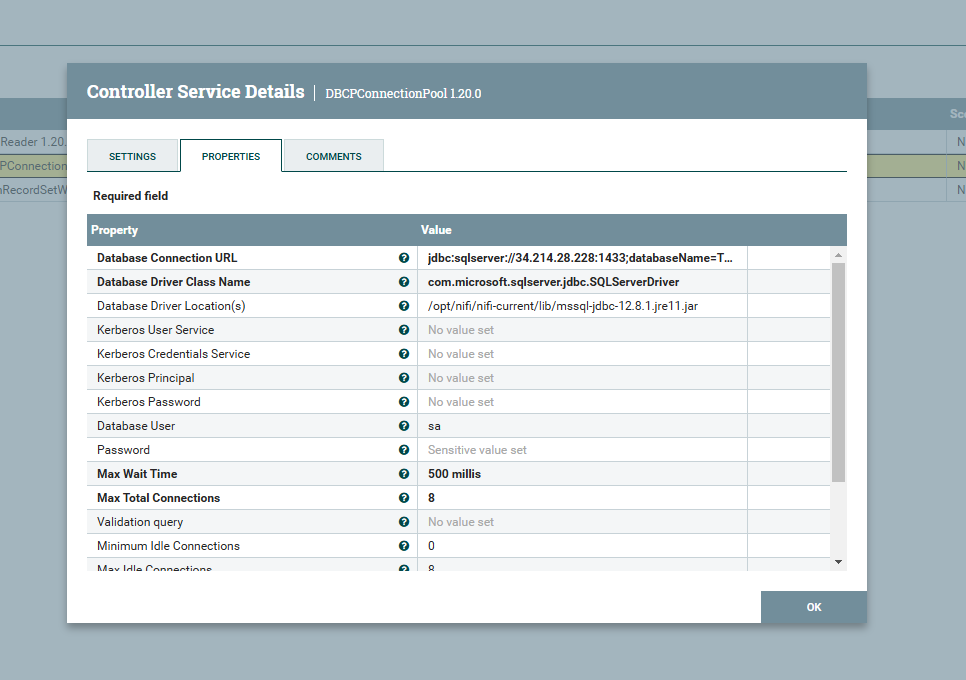
**select** \* **from** Transactions t

2.9 Configuramos lós siguientes processor QueryDatabseTable





Crea um servicio com lós siguiente datos:



jdbc:sqlserver://34.214.28.228:1433;databaseName=TestDB;encrypt=false

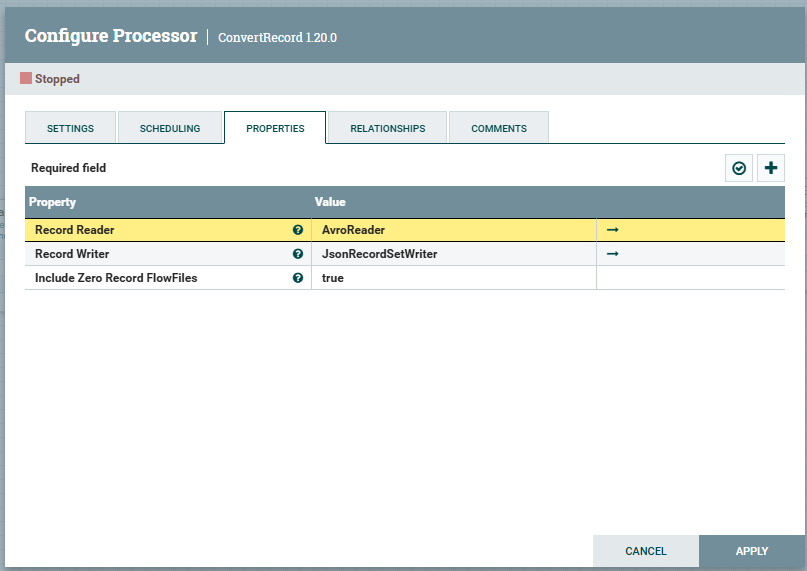
com.microsoft.sqlserver.jdbc.SQLServerDriver

/opt/nifi/nifi-current/lib/mssql-jdbc-12.8.1.jre11.jar

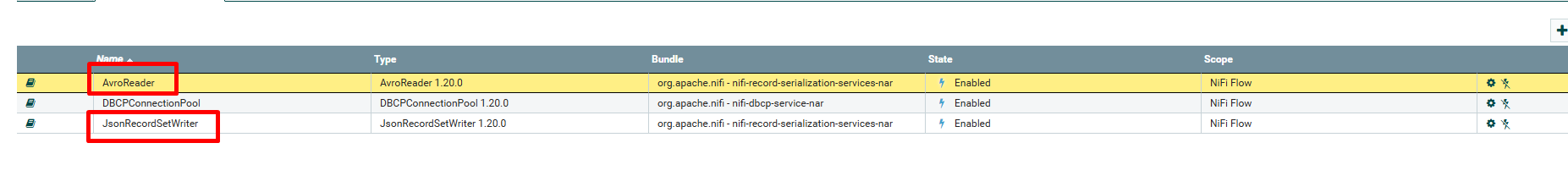
Database user: sa

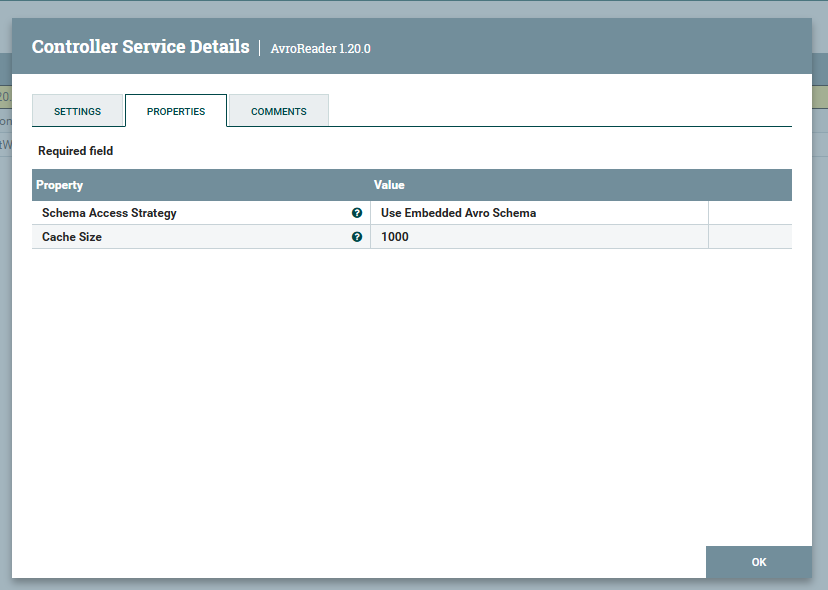
Password: YourStrong@Password123

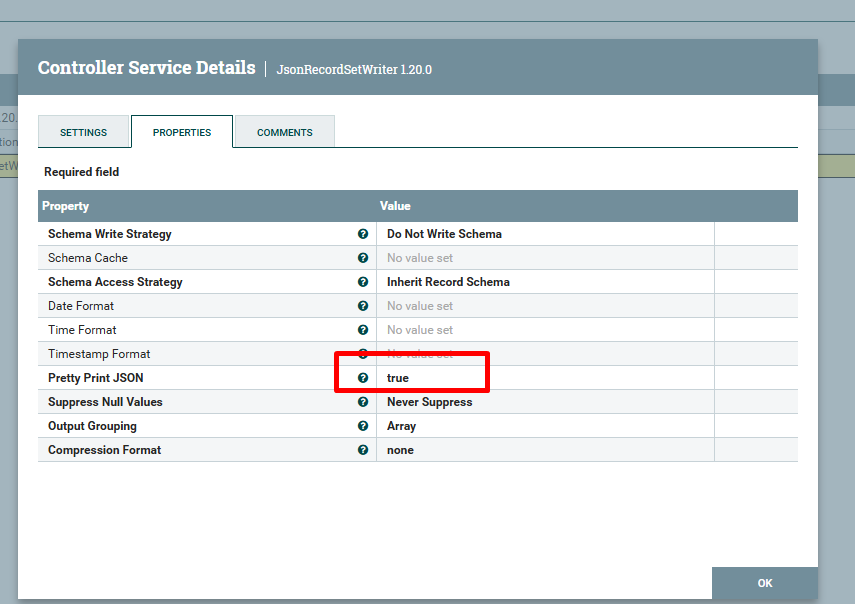
3.0 Configuramos ConvertRecord 1.20.0



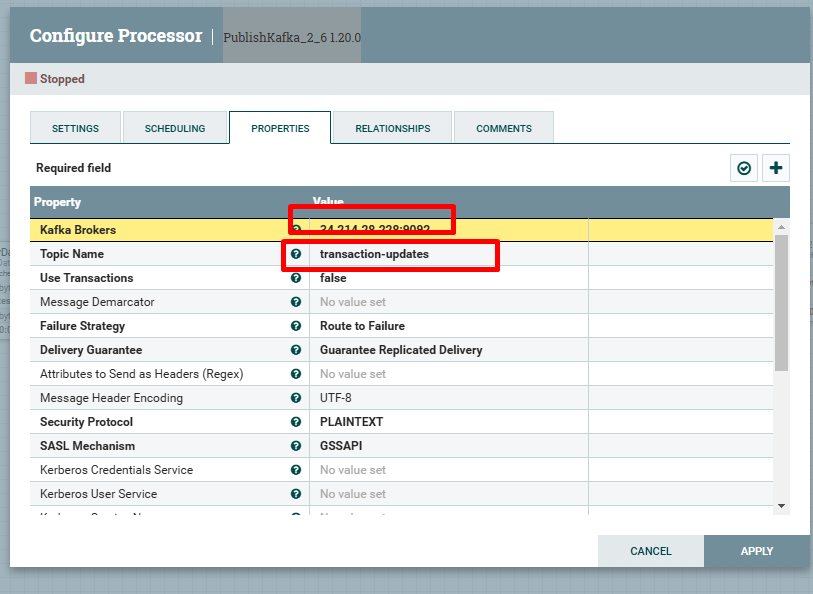
Configuramos ambos servicios





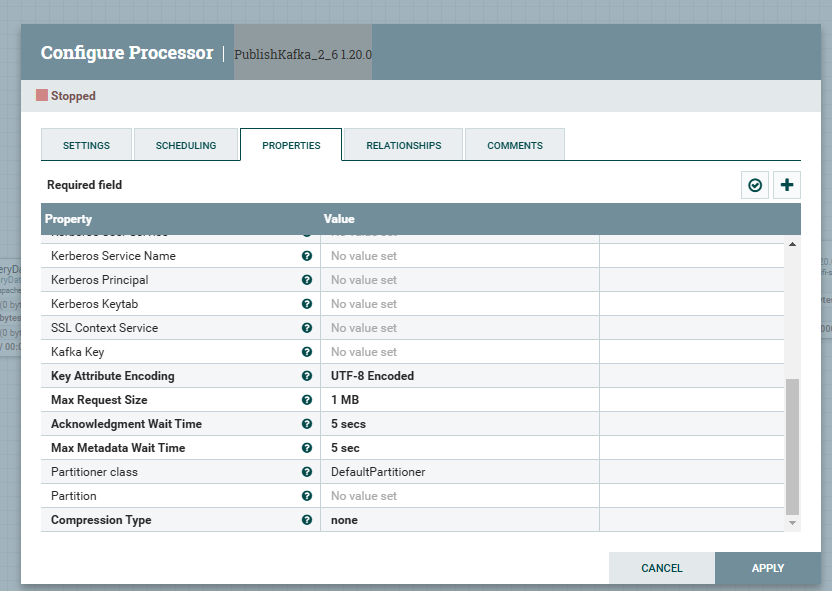


3.1 Configuramos PublishKafka\_2\_6 1.20.0



Topic Name: transaction-updates

Use transactions: False



3.2 Ejecutar el script en la base de datos:

**WHILE** 1=1

**BEGIN**

-- Actualizar el estado de las transacciones aleatoriamente

**UPDATE** Transactions

**SET** Status = **CASE**

**WHEN** Status = **'Completed'** **THEN** **'Pending'**

**WHEN** Status = **'Pending'** **THEN** **'Completed'**

**END**,

LastUpdated = **GETDATE**()

**WHERE** ID = **ABS**(CHECKSUM(NEWID()) % (**SELECT** **MAX**(ID) **FROM** Transactions)) + 1;

-- Insertar nuevas transacciones aleatorias

**INSERT** **INTO** Transactions (TransactionDate, Amount, Status)

**VALUES**

(**DATEADD**(**SECOND**, **ABS**(CHECKSUM(NEWID()) % 100), **GETDATE**()),

**CAST**(**ABS**(CHECKSUM(NEWID()) % 1000) + 0.01 **AS** **DECIMAL**(10, 2)),

**CASE** **WHEN** **ABS**(CHECKSUM(NEWID()) % 2) = 0 **THEN** **'Completed'** **ELSE** **'Pending'** **END**);

-- Espera 2 segundos antes de la siguiente actualización

**WAITFOR** DELAY **'00:00:02'**;

**END**;

3.3 Configurar las misma métricas de grafanna de los ejercicos anteriores

