

ENTREGA EJERCICIO 2: Nifi + ELK

Usando nifi+ELK debéis presentar una solución que muestre sobre un mapa la disposición de delitos presentes en esta

api: <https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9>

Entregables:

- Enlace a código Github:
https://github.com/MireyaSC/EntregasEDEM/tree/main/Entrega2_Nifi_ELK
- Captura de pantalla de vuestra interfaz Kibana

Opcional:

- Existen otras fuentes de datos que proporcionan también llamadas al 311 de otras zonas, opcionalmente podéis concatenar más datasets y pintarlos de manera conjunta. <https://catalog.data.gov/dataset/311-data-in-development>



PASOS SEGUIDOS PARA LA REALIZACIÓN DEL EJERCIO

Primero: Generar un fichero docker-compose.yml con Nifi, Elasticsearch y Kibana:

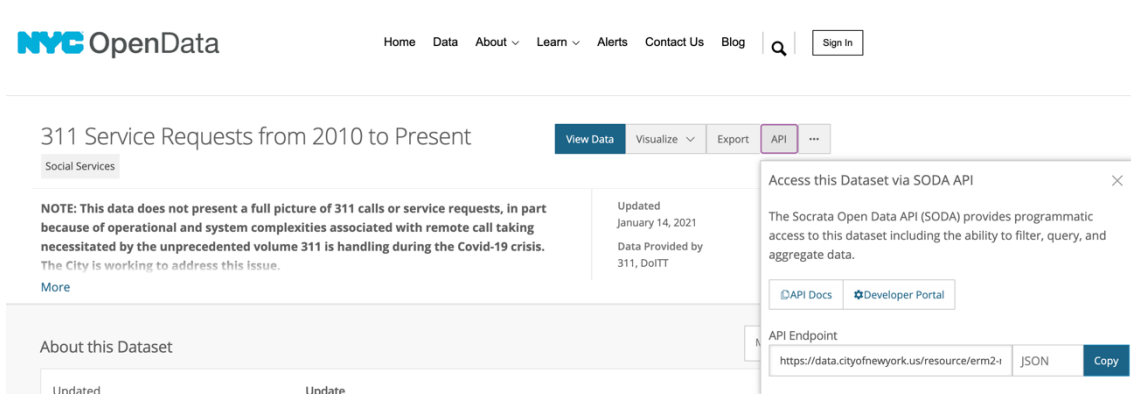
https://github.com/MireyaSC/EntregasEDEM/blob/main/Entrega2_Nifi_ELK/Cod e/docker-compose.yml

Segundo: hacemos un flujo en Nifi configurando los procesadores y las conexiones entre ellos:

Procesador 1: Elegimos un procesador del tipo InvokeHTTP para poder ingestar datos con el API:

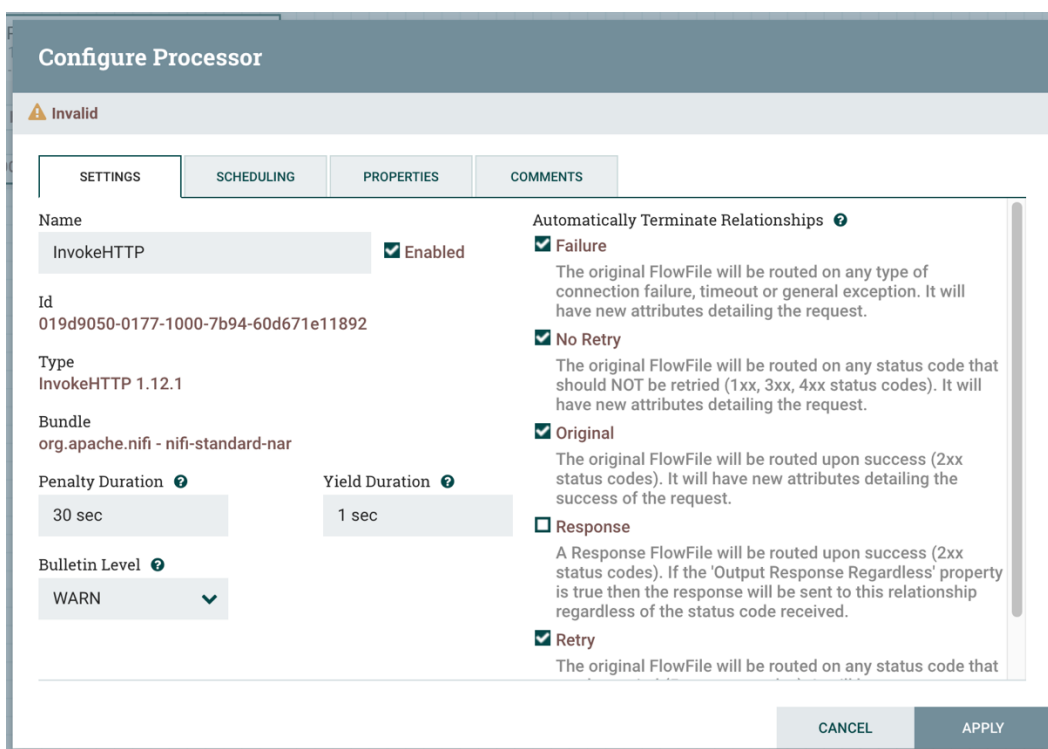
		
 InvokeHTTP InvokeHTTP 1.12.1 org.apache.nifi - nifi-standard-nar		
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Entramos en la configuración del procesador y necesitamos la “Remote URL”. Para conseguir este valor, vamos a la web <https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9> y hacemos click en API. Así obtenemos la Remote URL <https://data.cityofnewyork.us/resource/erm2-nwe9.json>.



The screenshot shows the NYC OpenData website. The main heading is "311 Service Requests from 2010 to Present". Below it, there's a note: "NOTE: This data does not present a full picture of 311 calls or service requests, in part because of operational and system complexities associated with remote call taking necessitated by the unprecedented volume 311 is handling during the Covid-19 crisis. The City is working to address this issue." There are tabs for "View Data", "Visualize", "Export", and "API". A modal window titled "Access this Dataset via SODA API" is open, showing the "API Endpoint" as <https://data.cityofnewyork.us/resource/erm2-nwe9.json> in JSON format, with a "Copy" button.

La ponemos en la configuración de nuestro procesador:



The screenshot shows the "Configure Processor" dialog for the "InvokeHTTP" processor. The "SETTINGS" tab is active. The "Name" is "InvokeHTTP" and it is "Enabled". The "Id" is "019d9050-0177-1000-7b94-60d671e11892". The "Type" is "InvokeHTTP 1.12.1" and the "Bundle" is "org.apache.nifi - nifi-standard-nar". The "Penalty Duration" is "30 sec" and the "Yield Duration" is "1 sec". The "Bulletin Level" is set to "WARN". On the right, under "Automatically Terminate Relationships", the options "Failure", "No Retry", "Original", and "Retry" are all checked, while "Response" is unchecked. The "Failure" and "No Retry" options have descriptive text. At the bottom, there are "CANCEL" and "APPLY" buttons.

Configure Processor

Invalid

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS


Required field

Property	Value
HTTP Method	GET
Remote URL	https://data.cityofnewyork.us/resource/erm2-nwe9.json
SSL Context Service	No value set
Connection Timeout	5 secs
Read Timeout	15 secs
Idle Timeout	5 mins
Max Idle Connections	5
Include Date Header	True
Follow Redirects	True
Attributes to Send	No value set
Useragent	No value set
Basic Authentication Username	No value set
Basic Authentication Password	No value set

CANCEL

APPLY

Procesador 2: Elegimos un SplitJson para poder separar el string de datos en documentos separados:



SplitJson

SplitJson 1.12.1

org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Entramos en la configuración de nuestro procesador y la completamos:

Configure Processor

Invalid

SETTINGS SCHEDULING PROPERTIES COMMENTS

Name
SplitJson ☒ Enabled

Id
05336f00-0177-1000-8471-4b18bd1a1c7f

Type
Split.Json 1.12.1

Bundle
org.apache.nifi - nifi-standard-nar

Penalty Duration Yield Duration

Bulletin Level

Automatically Terminate Relationships ☒ failure
If a FlowFile fails processing for any reason (for example, the FlowFile is not valid JSON or the specified path does not exist), it will be routed to this relationship

☒ original
The original FlowFile that was split into segments. If the FlowFile fails processing, nothing will be sent to this relationship

☐ split
All segments of the original FlowFile will be routed to this relationship

CANCEL APPLY

Configure Processor

Invalid



SETTINGS SCHEDULING PROPERTIES COMMENTS

Required field

Property	Value
JsonPath Expression	\$.*
Null Value Representation	the string 'null'

CANCEL APPLY

Procesador 3: Elegimos un procesador PutElasticsearchHttp para poder introducir los datos en Elasticsearch por http:

<div></div> <div><div></div><div><div>PutElasticsearchHttp</div><div>PutElasticsearchHttp 1.12.1</div><div>org.apache.nifi - nifi-elasticsearch-nar</div></div></div>		
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Entramos en la configuración y la completamos:

Configure Processor

Stopped

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Name

PutElasticsearchHttp

☒ Enabled

Id

053bf17b-0177-1000-dd93-27ff892ff367

Type

PutElasticsearchHttp 1.12.1

Bundle

org.apache.nifi - nifi-elasticsearch-nar

Penalty Duration

30 sec

Yield Duration

1 sec

Bulletin Level

WARN

Automatically Terminate Relationships

☒ failure

All FlowFiles that cannot be written to Elasticsearch are routed to this relationship

☒ retry

A FlowFile is routed to this relationship if the database cannot be updated but attempting the operation again may succeed

☒ success

All FlowFiles that are written to Elasticsearch are routed to this relationship

CANCEL

APPLY

Configure Processor

Invalid

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Required field

Property	Value
Elasticsearch URL	http://elasticsearch:9200
SSL Context Service	No value set
Character Set	UTF-8
Username	No value set
Password	No value set
Connection Timeout	5 secs
Response Timeout	15 secs
Proxy Configuration Service	No value set
Proxy Host	No value set
Proxy Port	No value set
Proxy Username	No value set
Proxy Password	No value set

CANCEL

APPLY

Configure Processor

Stopped

SETTINGS SCHEDULING **PROPERTIES** COMMENTS

Required field +

Property	Value
Password	No value set
Connection Timeout	5 secs
Response Timeout	15 secs
Proxy Configuration Service	No value set
Proxy Host	No value set
Proxy Port	No value set
Proxy Username	No value set
Proxy Password	No value set
Identifier Attribute	No value set
Index	edem
Type	_doc
Batch Size	100
Index Operation	index

CANCEL APPLY

Unimos los procesadores 1 y 2 y configuramos la conexión:

Configure Connection

DETAILS **SETTINGS**

From Processor	To Processor
InvokeHTTP	SplitJson
InvokeHTTP	SplitJson
Within Group	Within Group
NiFi Flow	NiFi Flow
For Relationships	
<input type="checkbox"/> Failure	
<input type="checkbox"/> No Retry	
<input type="checkbox"/> Original	
<input checked="" type="checkbox"/> Response	
<input type="checkbox"/> Retry	

CANCEL APPLY

Unimos los procesadores 2 y 3 y configuramos la conexión:

Configure Connection

DETAILS

SETTINGS

From Processor

SplitJson

SplitJson

To Processor

PutElasticsearchHttp

PutElasticsearchHttp

Within Group

NiFi Flow

Within Group

NiFi Flow

For Relationships

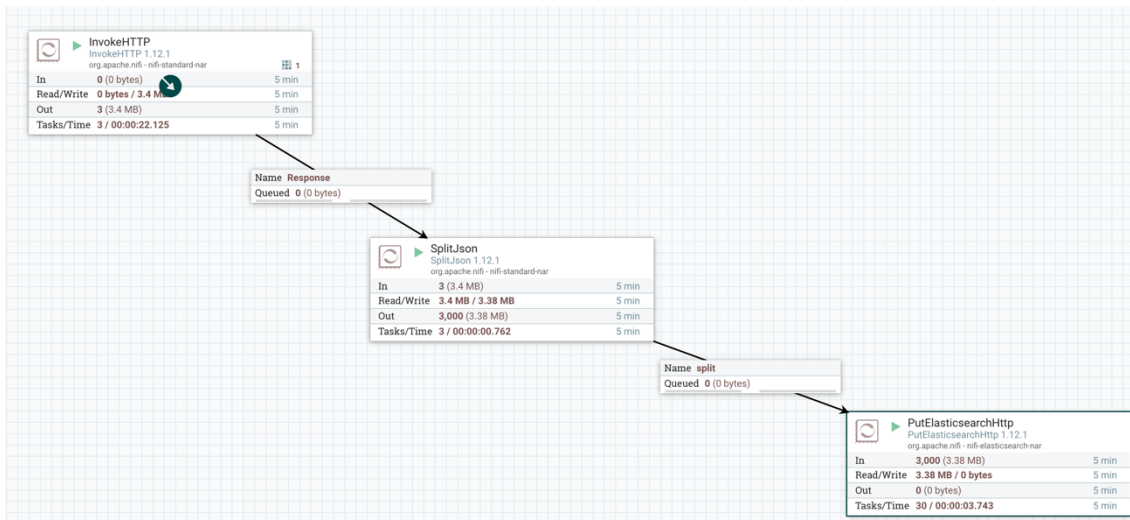
☐ failure

☐ original

☒ split

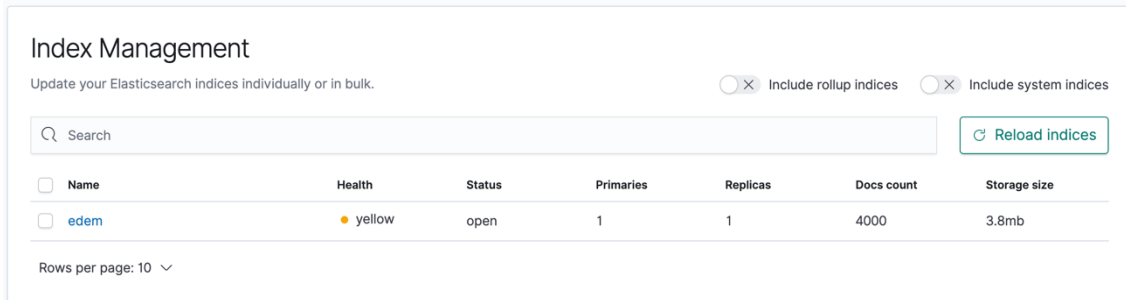
CANCEL

APPLY



Tercero: Vamos a Kibana accediendo a <http://localhost:5601/>.

Accediendo al gestor de índices, vemos que nos ha creado el índice "edem" como hemos puesto en el procesador de Nifi PutElasticsearchHttp:



The screenshot shows the 'Index Management' page in Kibana. At the top, there's a search bar and two toggle switches for 'Include rollout indices' and 'Include system indices'. Below the search bar is a table of indices. The table has columns for Name, Health, Status, Primaries, Replicas, Docs count, and Storage size. One index, 'edem', is listed with a 'yellow' health status, 'open' status, 1 primary, 1 replica, 4000 docs, and 3.8mb storage size. A 'Reload indices' button is on the right. At the bottom, it says 'Rows per page: 10'.

Name	Health	Status	Primaries	Replicas	Docs count	Storage size
edem	yellow	open	1	1	4000	3.8mb

Tenemos que crear un nuevo índice desde la consola para poder indicar que latitude y longitude deben ser entendidos como un geopoint. Le llamaremos edem_index:

```
GET edem
PUT edem_index
{
  "mappings": {
    "properties": {
      "location": {
        "type": "geo_point"
      }
    }
  }
}
POST _reindex
{
  "source": {
    "index": "edem"
  },
  "dest": {
    "index": "edem_index"
  },
  "script": {
    "source": "ctx._source.location=['lat':ctx._source.latitude, 'lon':ctx._source.longitude];"
  }
}
```


Cramos un nuevo tipo de patrón utilizando en la búsqueda nuestro edem_index:

Index patterns [?](#)

Search...

Pattern ↑

apm-*

Default

edem_*

Rows per page: 10 ▼

Create index pattern

Vamos a visualizar nuestros datos y añadirmos el bucket:

