

Prueba técnica: Data pipeline

Candidato: Martha Domínguez Orduño

Descripción: Desarrollar un pipeline de análisis de datos utilizando los datos abiertos de la Ciudad de México correspondientes a la ubicación de las unidades del metrobús durante la última hora para obtener un histórico de la posición en la que se encuentra cada unidad que pueda ser consultado mediante un API Rest filtrando por unidad o por alcaldía.

Arquitectura de la solución:



Stack tecnológico: Elastic Stack

ELK Stack (o Elastic Stack) es un conjunto de herramientas open source desarrolladas por Elasticsearch que permite recoger datos de cualquier tipo de fuente y en cualquier formato para realizar búsquedas, análisis y visualización de los datos en tiempo real.

Docker Compose

Con Compose puedes crear diferentes contenedores y al mismo tiempo, en cada contenedor, diferentes servicios, unirlos a un volumen común, iniciarlos y apagarlos, etc. Es un componente fundamental para poder construir aplicaciones y microservicios.

Prueba técnica: Data pipeline

Fuente

De la fuente obtenemos los datos crudos en formato json sobre las ubicaciones del Metrobús de la Ciudad de México, los datos lucen de la siguiente forma:

```
{
  "nhits": 207,
  "parameters": {
    "dataset": "prueba_fetchdata_metrobus",
    "timezone": "UTC",
    "rows": 10,
    "format": "json",
    "records": [
      {
        "datasetid": "prueba_fetchdata_metrobus",
        "recordid": "2772acalf8d450e3b9880905a79d093f395b007f",
        "fields": {
          "vehicle_id": "170",
          "date_updated": "2020-09-30 22:09:15",
          "position_longitude": "-99.18779754638672",
          "trip_schedule_relationship": 2,
          "position_speed": 16,
          "position_latitude": 19.317499160766,
          "trip_route_id": "367",
          "vehicle_label": "112",
          "position_odometer": 231,
          "vehicle_current_status": 2,
          "geographic_point": [19.317499160767, -99.1877975463871],
          "geometry": {
            "type": "Point",
            "coordinates": [
              -99.187797546387,
              19.3174991607671
            ]
          },
          "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
          "datasetid": "prueba_fetchdata_metrobus",
          "recordid": "8c34c423a11634dc8450557cf7bceff728f10084",
          "fields": {
            "vehicle_id": "177",
            "trip_start_date": "20200428",
            "date_updated": "2020-09-30 22:09:15",
            "position_longitude": "-99.17749786376953",
            "trip_schedule_relationship": 0,
            "position_speed": 13,
            "position_latitude": 19.292600631713867,
            "trip_route_id": "367",
            "vehicle_label": "119",
            "position_odometer": 0,
            "trip_id": "9732304",
            "vehicle_current_status": 1,
            "geographic_point": [19.292600631714, -99.177497863771],
            "geometry": {
              "type": "Point",
              "coordinates": [
                -99.17749786377,
                19.2926006317141
              ]
            },
            "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
            "datasetid": "prueba_fetchdata_metrobus",
            "recordid": "5891a34451e7329d26f5a2b69bb59900555e41a",
            "fields": {
              "vehicle_id": "1288",
              "date_updated": "2020-09-30 22:09:15",
              "position_longitude": "-99.04730224609375",
              "trip_schedule_relationship": 2,
              "position_speed": 0,
              "position_latitude": 19.39800047607422,
              "vehicle_label": "221",
              "position_odometer": 46,
              "vehicle_current_status": 2,
              "geographic_point": [19.398000476074, -99.047302246094],
              "geometry": {
                "type": "Point",
                "coordinates": [
                  -99.047302246094,
                  19.3980004760741
                ]
              },
              "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
              "datasetid": "prueba_fetchdata_metrobus",
              "recordid": "4752eb2ad8104e01d7ac560af195c93574600efb",
              "fields": {
                "vehicle_id": "375",
                "date_updated": "2020-09-30 22:09:15",
                "position_longitude": "-99.05950164794922",
                "trip_schedule_relationship": 2,
                "position_speed": 0,
                "position_latitude": 19.46489906311,
                "vehicle_label": "317",
                "position_odometer": 0,
                "vehicle_current_status": 1,
                "geographic_point": [19.46489906311, -99.059501647949],
                "geometry": {
                  "type": "Point",
                  "coordinates": [
                    -99.059501647949,
                    19.464899063111
                  ]
                },
                "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                "datasetid": "prueba_fetchdata_metrobus",
                "recordid": "eddbd530c14cd27710f469c6e59035af05302ebf",
                "fields": {
                  "vehicle_id": "1018",
                  "trip_start_date": "20200428",
                  "date_updated": "2020-09-30 22:09:15",
                  "position_longitude": "-99.12950134277344",
                  "trip_schedule_relationship": 0,
                  "position_speed": 9,
                  "position_latitude": 19.39780044555664,
                  "trip_route_id": "301",
                  "vehicle_label": "339",
                  "position_odometer": 492,
                  "trip_id": "9737162",
                  "vehicle_current_status": 2,
                  "geographic_point": [19.397800445557, -99.1295013427731],
                  "geometry": {
                    "type": "Point",
                    "coordinates": [
                      -99.129501342773,
                      19.3978004455571
                    ]
                  },
                  "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                  "datasetid": "prueba_fetchdata_metrobus",
                  "recordid": "bdc687fcbab66d0bc7d11183a64285e901b7ab5",
                  "fields": {
                    "vehicle_id": "398",
                    "trip_start_date": "20200428",
                    "date_updated": "2020-09-30 22:09:15",
                    "position_longitude": "-99.18679809570312",
                    "trip_schedule_relationship": 0,
                    "position_speed": 0,
                    "position_latitude": 19.40220069885254,
                    "trip_route_id": "301",
                    "vehicle_label": "340",
                    "position_odometer": 735,
                    "trip_id": "9736450",
                    "vehicle_current_status": 2,
                    "geographic_point": [19.402200698853, -99.1867980957031],
                    "geometry": {
                      "type": "Point",
                      "coordinates": [
                        -99.186798095703,
                        19.4022006988531
                      ]
                    },
                    "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                    "datasetid": "prueba_fetchdata_metrobus",
                    "recordid": "96ff0e95339e3c6501b375b20983ae189b3e97d",
                    "fields": {
                      "vehicle_id": "409",
                      "trip_start_date": "20200428",
                      "date_updated": "2020-09-30 22:09:15",
                      "position_longitude": "-99.17240142822266",
                      "trip_schedule_relationship": 0,
                      "position_speed": 12,
                      "position_latitude": 19.403900146484375,
                      "trip_route_id": "301",
                      "vehicle_label": "351",
                      "position_odometer": 131,
                      "trip_id": "9736261",
                      "vehicle_current_status": 2,
                      "geographic_point": [19.403900146484, -99.172401428223],
                      "geometry": {
                        "type": "Point",
                        "coordinates": [
                          -99.172401428223,
                          19.4039001464841
                        ]
                      },
                      "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                      "datasetid": "prueba_fetchdata_metrobus",
                      "recordid": "b3f4fc731c85781eaa94a5e0fd59a39cbe230c2",
                      "fields": {
                        "vehicle_id": "449",
                        "trip_start_date": "20200428",
                        "date_updated": "2020-09-30 22:09:15",
                        "position_longitude": "-99.17040252685547",
                        "trip_schedule_relationship": 0,
                        "position_speed": 5,
                        "position_latitude": 19.403499603271484,
                        "trip_route_id": "301",
                        "vehicle_label": "391",
                        "position_odometer": 0,
                        "trip_id": "9736737",
                        "vehicle_current_status": 1,
                        "geographic_point": [19.403499603271, -99.1704025268551],
                        "geometry": {
                          "type": "Point",
                          "coordinates": [
                            -99.170402526855,
                            19.4034996032711
                          ]
                        },
                        "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                        "datasetid": "prueba_fetchdata_metrobus",
                        "recordid": "551cfed918802a5a306aa91bdc3e37edfe82",
                        "fields": {
                          "vehicle_id": "1",
                          "trip_start_date": "20200428",
                          "date_updated": "2020-09-30 22:09:15",
                          "position_longitude": "-99.14900207519531",
                          "trip_schedule_relationship": 0,
                          "position_speed": 3,
                          "position_latitude": 19.427000045776367,
                          "trip_route_id": "1",
                          "vehicle_label": "401",
                          "position_odometer": 668,
                          "trip_id": "9685594",
                          "vehicle_current_status": 2,
                          "geographic_point": [19.427000045776, -99.149002075195],
                          "geometry": {
                            "type": "Point",
                            "coordinates": [
                              -99.149002075195,
                              19.4270000457761
                            ]
                          },
                          "record_timestamp": "2020-10-01T03:09:16.345000+00:00",
                          "datasetid": "prueba_fetchdata_metrobus",
                          "recordid": "eebf53918ac6f001db30f0315334f92d1ccfa6c",
                          "fields": {
                            "vehicle_id": "22",
                            "trip_start_date": "20200428",
                            "date_updated": "2020-09-30 22:09:15",
                            "position_longitude": "-99.15409851074219",
                            "trip_schedule_relationship": 0,
                            "position_speed": 1,
                            "position_latitude": 19.419599533081055,
                            "trip_route_id": "1",
                            "vehicle_label": "422",
                            "position_odometer": 560,
                            "trip_id": "9685769",
                            "vehicle_current_status": 2,
                            "geographic_point": [19.419599533081, -99.154098510742],
                            "geometry": {
                              "type": "Point",
                              "coordinates": [
                                -99.154098510742,
                                19.4195995330811
                              ]
                            },
                            "record_timestamp": "2020-10-01T03:09:16.345000+00:00"
                          ]
                        }
                      ]
                    }
                  }
                }
              ]
            }
          }
        }
      ]
    }
  }
}
```

https://datos.cdmx.gob.mx/api/records/1.0/search/?dataset=prueba_fetchdata_metrobus&q=

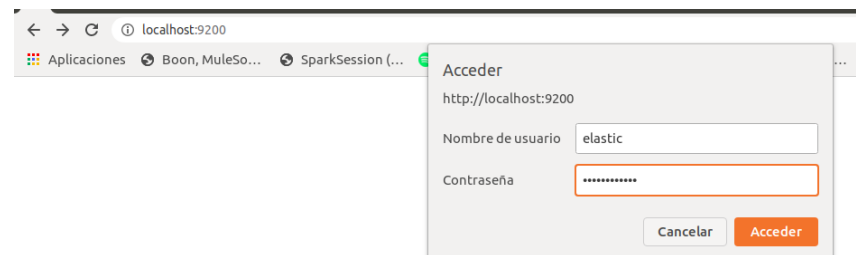
Mediante la herramienta docker-compose se levantaron los servicios del stack elastic con las siguientes configuraciones.

docker-compose.yml

Servicio elasticsearch: Se habilitó el puerto 9200, se habilitaron credenciales de acceso

```
martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x
version: '3.2'

services:
  elasticsearch:
    build:
      context: elasticsearch/
      args:
        ELK_VERSION: $ELK_VERSION
    volumes:
      - type: bind
        source: ./elasticsearch/config/elasticsearch.yml
        target: /usr/share/elasticsearch/config/elasticsearch.yml
        read_only: true
      - type: volume
        source: elasticsearch
        target: /usr/share/elasticsearch/data
    ports:
      - "9200:9200"
      - "9300:9300"
    environment:
      ES_JAVA_OPTS: "-Xmx256m -Xms256m"
      ELASTIC_PASSWORD: datapipeline
      # Use single node discovery in order to disable production mode and avoid bootstrap checks
      # see https://www.elastic.co/guide/en/elasticsearch/reference/current/bootstrap-checks.html
      discovery.type: single-node
    networks:
      - elk
```



Prueba técnica: Data pipeline

Servicio corriendo

```
localhost:9200
Aplicaciones Boon, MuleSo... SparkSession (... Spotify Or

{
  "name" : "a450112df9e2",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "k37cCzKZSgu9lZ5t7ln3GA",
  "version" : {
    "number" : "7.9.2",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "d34da0ea4a966c4e49417f2da2f244e3e97b4e6e",
    "build_date" : "2020-09-23T00:45:33.626720Z",
    "build_snapshot" : false,
    "lucene_version" : "8.6.2",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Servicio logstash: se habilitaron puertos de acceso - "5000:5000/tcp", - "5000:5000/udp" y "9600:9600", se especificó la red elk que se levanta con los servicios, es un servicio dependiente de elasticsearch

```
martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@ma
logstash:
build:
  context: logstash/
  args:
    ELK_VERSION: $ELK_VERSION
volumes:
  - type: bind
    source: ./logstash/config/logstash.yml
    target: /usr/share/logstash/config/logstash.yml
    read_only: true
  - type: bind
    source: ./logstash/pipeline
    target: /usr/share/logstash/pipeline
    read_only: true
ports:
  - "5000:5000/tcp"
  - "5000:5000/udp"
  - "9600:9600"
environment:
  LS_JAVA_OPTS: "-Xmx256m -Xms256m"
networks:
  - elk
depends_on:
  - elasticsearch
```

Servicio corriendo

```
localhost:9600
Aplicaciones Boon, MuleSo... SparkSession (... Online Scala C... ARCE Aprovisi... Portal de Emp... Slack Scala-Spark LL... Artifactory W... »

{"host": "5f05d6d40099", "version": "7.9.2", "http_address": "0.0.0.0:9600", "id": "bc7880be-ceb1-4285-b384-16082e91fb14", "name": "5f05d6d40099", "ephemeral_id": "5ddb9b7d-0630-476a-9147-6c0ba0128bbc", "status": "green", "snapshot": false, "pipeline": {"workers": 4, "batch_size": 125, "batch_delay": 50}, "monitoring": {"hosts": ["http://elasticsearch:9200"], "username": "elastic"}, "build_date": "2020-09-23T03:12:32Z", "build_sha": "0d3d0617adaa8c0c770469c9c32e18514b5d8f15", "build_snapshot": false}
```

Prueba técnica: Data pipeline

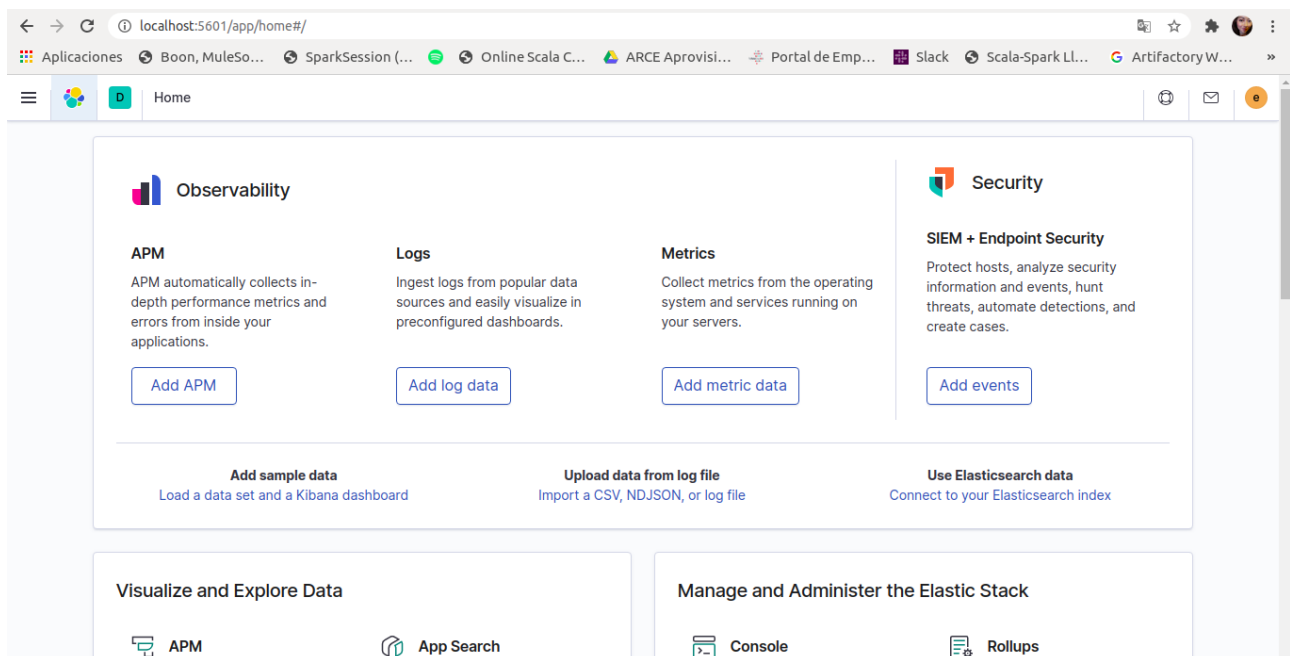
Servicio Kibana: se especificó la red elk que se levanta con los servicios, es un servicio dependiente de elasticsearch, se especificó el puerto de acceso 5601

```
kibana:
  build:
    context: kibana/
    args:
      ELK_VERSION: $ELK_VERSION
  volumes:
    - type: bind
      source: ./kibana/config/kibana.yml
      target: /usr/share/kibana/config/kibana.yml
      read_only: true
  ports:
    - "5601:5601"
  networks:
    - elk
  depends_on:
    - elasticsearch

networks:
  elk:
    driver: bridge

volumes:
  elasticsearch:
```

Servicio corriendo



Prueba técnica: Data pipeline

Configuraciones individuales:

elasticsearch.yml

```
martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x
--
## Default Elasticsearch configuration from Elasticsearch base image.
## https://github.com/elastic/elasticsearch/blob/master/distribution/docker/src/docker/config/elasticsearch.yml
#
cluster.name: "docker-cluster"
network.host: 0.0.0.0

## X-Pack settings
## see https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-xpack.html
#
xpack.license.self_generated.type: trial
xpack.security.enabled: true
xpack.monitoring.collection.enabled: true
--
```

kibana.yml

```
martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x
---
## Default Kibana configuration from Kibana base image.
## https://github.com/elastic/kibana/blob/master/src/dev/build/tasks/os_packages/docker_generator/templates/kibana.yml.template.js
#
server.name: kibana
server.host: 0.0.0.0
elasticsearch.hosts: [ "http://elasticsearch:9200" ]
monitoring.ui.container.elasticsearch.enabled: true

## X-Pack security credentials
#
elasticsearch.username: elastic
elasticsearch.password: datapipeline
--
```

logstash.yml

```
martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x
---
## Default Logstash configuration from Logstash base image.
## https://github.com/elastic/logstash/blob/master/docker/data/logstash/config/logstash-full.yml
#
http.host: "0.0.0.0"
xpack.monitoring.elasticsearch.hosts: [ "http://elasticsearch:9200" ]

## X-Pack security credentials
#
xpack.monitoring.enabled: true
xpack.monitoring.elasticsearch.username: elastic
xpack.monitoring.elasticsearch.password: datapipeline
--
```

Prueba técnica: Data pipeline

Ingesta

Logstash, uno de los productos principales del Elastic Stack, se usa para agregar y procesar datos y enviarlos a Elasticsearch. Logstash es una pipeline de procesamiento de datos open source y del lado del servidor que te permite ingestar datos de múltiples fuentes simultáneamente y enriquecerlos y transformarlos antes de que se indexen en Elasticsearch

Logstash se basa en el uso de plugins que soportan los distintos elementos de la herramienta mencionados: entradas, códecs, filtros y salidas.

Logstash dispone de un gran conjunto de plugins para la entrada de datos:

Beats: datos de Elastic Beats

Elasticsearch: resultados de consultas a Elasticsearch

Exec: salida de comandos

File: ficheros

http: datos recibidos por http o https

jdbc: datos de BBDD

pipe: datos de un comando pipe

tcp: datos recibidos por un socket tcp

Se creó un Pipeline de logstash para ingestar datos a elasticsearch como se muestra a continuación.

```
martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@mar...
input {
  http_poller {
    urls => {
      urlname => "https://datos.cdmx.gob.mx/api/records/1.0/search/?dataset=prueba_fetchdata_metrobus&q="
    }
    request_timeout => 60
    schedule => { every => "20s" }
    codec => "line"
  }
}

filter {
  json{
    source => "message"
    remove_field => ["[message]"]
  }
}

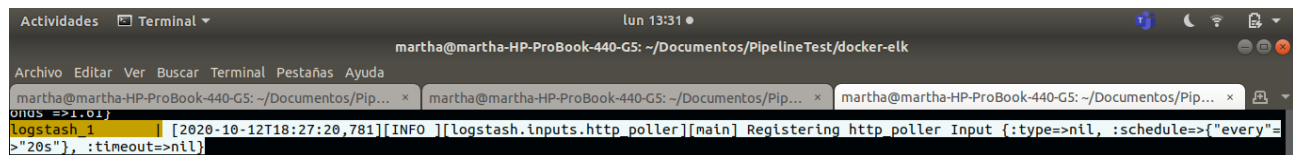
output {
  elasticsearch {
    index => "pipeline_base"
    hosts => "elasticsearch:9200"
    user => "elastic"
    password => "datapipeline"
    codec => "json"
  }
}
```

Dentro del flujo se tiene el **input**, donde se especifica con la ayuda del plugin http_poller la url de donde se ingestará la información, en este caso los datos abiertos de la Ciudad de México para la información del metrobus, se le dió un tiempo de respuesta de 60 segundos, y una consulta a la fuente de cada 20 segundos, se especificó mediante codec la representación de la información de entrada que en este caso será en line. La siguiente parte del flujo es **filter** donde se manipula el archivo con el plugin json, indicando que en la fuente “message” tenemos información sobre los records y posteriormente se elimina la etiqueta [message] para que no se almacene como un registro. La última parte del flujo **output** con la ayuda del plugin elasticsearch se especifica donde se almacenará la información con un nombre de índice, un nombre de host que en éste caso es del

Prueba técnica: Data pipeline

elasticsearch levantado para la prueba, se le información credenciales de acceso y la representación de la información de salida en formato json.

Una vez que se levanta el docker la ingesta se lleva a cabo creando así un índice que será almacenado en elasticsearch

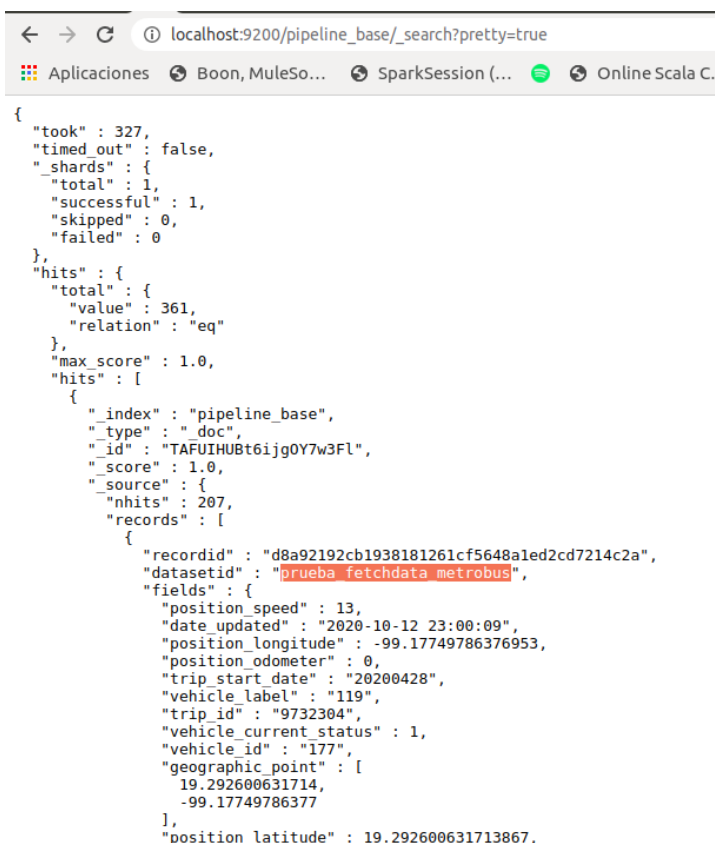


```
martha@martha-HP-ProBook-440-G5: ~/Documentos/PipelineTest/docker-elk
logstash_1 | [2020-10-12T18:27:20.781][INFO ][logstash.inputs.http_poller][main] Registering http_poller Input {:type=>nil, :schedule=>["every"=
>"20s"}, :timeout=>nil}
```

Almacenamiento

Para el almacenamiento de los datos se utiliza la herramienta elasticsearch la cual proporciona una API RESTful para la comunicación con las aplicaciones del cliente. Por lo tanto, las llamadas REST se usan para ingestar datos, realizar búsquedas y analíticas de datos, así como también administrar el cluster y sus índices. Internamente, todos los métodos descritos recurren a esta API para ingestar datos en Elasticsearch, en el clúster antes descrito se tienen tres nodos los cuales entre otras funcionalidades nos brindan el almacenamiento de los datos para posteriormente realizar el procesamiento de los datos con Kivana Dev Tools.

Información almacenada en elasticsearch

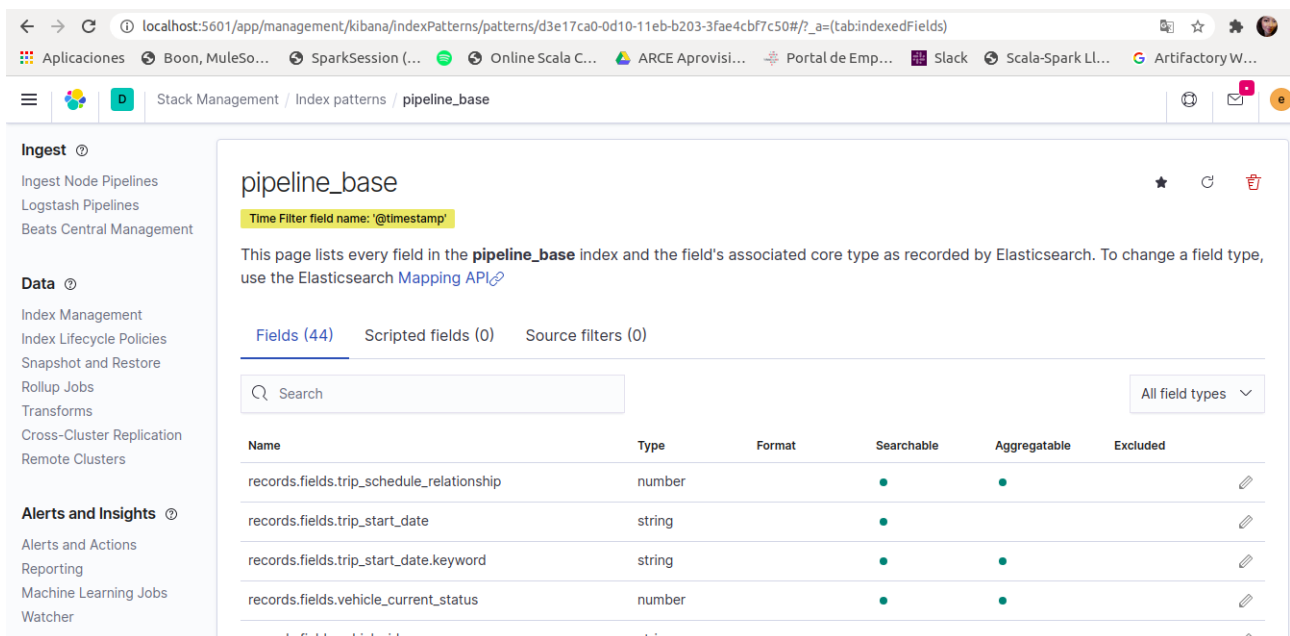
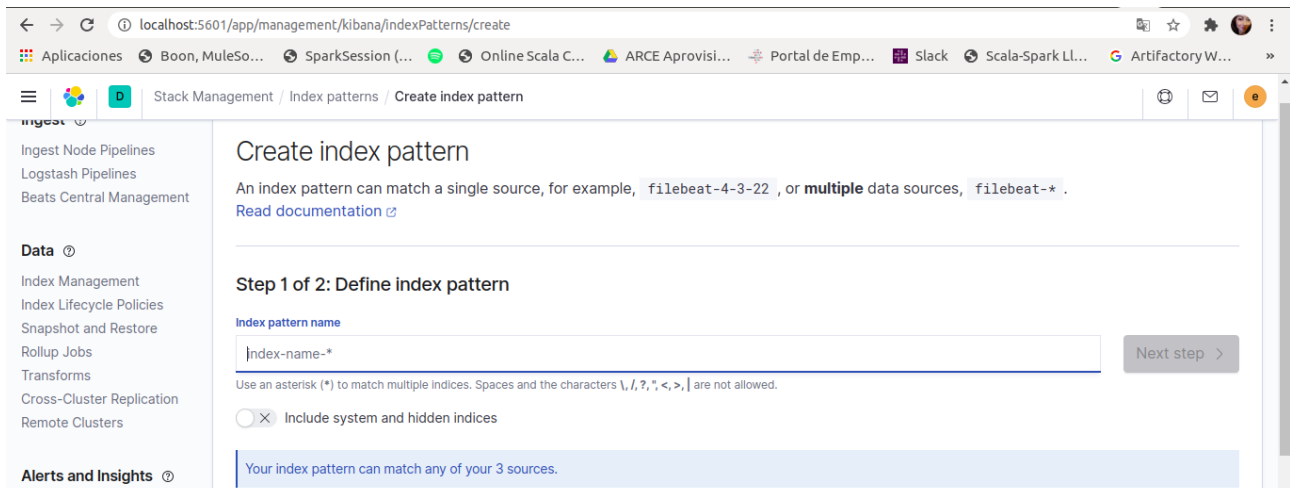


```
{
  "took" : 327,
  "timed_out" : false,
  "shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 361,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "pipeline_base",
        "_type" : "doc",
        "_id" : "TAFUIHUBt6ijg0Y7w3Ff",
        "_score" : 1.0,
        "_source" : {
          "nhits" : 207,
          "records" : [
            {
              "recordid" : "d8a92192cb1938181261cf5648aled2cd7214c2a",
              "datasetid" : "prueba fetchdata metrobús",
              "fields" : {
                "position_speed" : 13,
                "date_updated" : "2020-10-12 23:00:09",
                "position_longitude" : -99.17749786376953,
                "position_odometer" : 0,
                "trip_start_date" : "20200428",
                "vehicle_label" : "119",
                "trip_id" : "9732304",
                "vehicle_current_status" : 1,
                "vehicle_id" : "177",
                "geographic_point" : [
                  19.292600631714,
                  -99.17749786377
                ]
              },
              "position_latitude" : 19.292600631713867,
            }
          ]
        }
      }
    ]
  }
}
```


Prueba técnica: Data pipeline

Análisis y visualización

Para la visualización de la información de una forma gráfica se utiliza la herramienta Kibana donde se agregan los índices referentes a la información que se desea consultar, aquí se creó el índice nombrado desde el pipeline de logstash y que se almacenó en elasticsearch “pipeline_base”



Información sin graficar

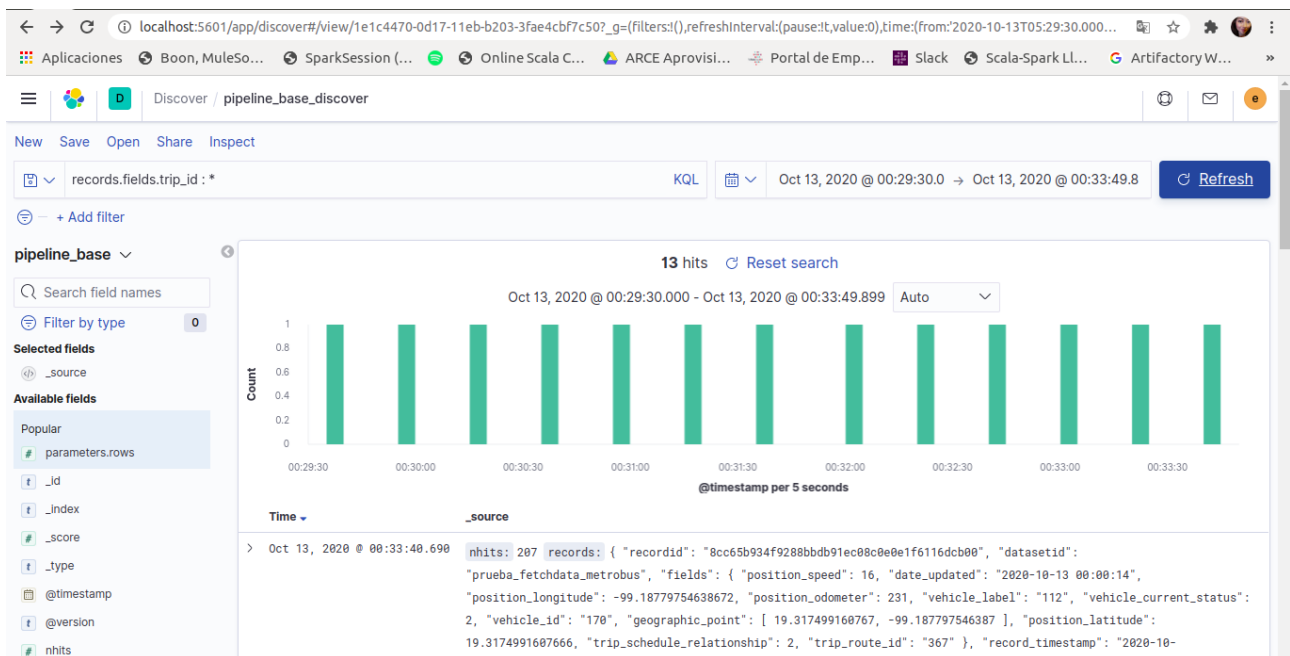
Prueba técnica: Data pipeline

```
1 GET pipeline_base/_search?pretty=true
2
```

```
1 {
2   "took": 261,
3   "timed_out": false,
4   "shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 51,
13      "relation": "eq"
14    },
15    "max_score": 1.0,
16    "hits": [
17      {
18        "_index": "pipeline_base",
19        "_type": "_doc",
20        "_id": "IwFNIHUBt6ijg0V7cmhs",
21        "_score": 1.0,
22        "_source": {
23          "nhits": 207,
24          "records": [
25            {
26              "recordid": "d8a92192cb1938181261cf5648a1ed2cd7214c2a",
27              "datasetid": "prueba_fetchdata_metrobus",
28              "fields": {
29                "position_speed": 13,
30                "date_updated": "2020-10-12 23:00:09",
31                "position_longitude": -99.17749786376953,
32                "position_odometer": 0,
33                "trip_start_date": "20200428",
34                "vehicle_label": "119",
35                "trin id": "9737384"

```

Información graficada



Para levantar los servicios se clona el repositorio con todas las configuraciones antes descritas

Prueba técnica: Data pipeline

<https://github.com/Marth4011/pipeline.git> → (Rama master)

Se levantan los servicios con el comando → sudo docker-compose up

(Se debe tener previamente instalado Docker)

Credenciales para acceder a los servicios:

username: elastic

password: datapipeline

Dentro de la carpeta existe un README general que describe las secciones de configuración de los servicios en docker a detalle, requerimientos, ligas a información importante sobre el stack elastic, etc.

```
martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x martha@martha-HP-ProBook-440-G5: ~/Documentos/Pip... x
# Elastic stack (ELK) on Docker

[!][Join the chat at https://gitter.im/deviantony/docker-elk](https://badges.gitter.im/Join%20Chat.svg)[!](https://gitter.im/deviantony/docker-elk?utm_source=badge&utm_medium=badge&utm_campaign=pr-badge&utm_content=badge)
[!][Elastic Stack version](https://img.shields.io/badge/ELK-7.9.2-blue.svg?style=flat)[!](https://github.com/deviantony/docker-elk/issues/539)
[!][Build Status](https://api.travis-ci.org/deviantony/docker-elk.svg?branch=master)[!](https://travis-ci.org/deviantony/docker-elk)

Run the latest version of the [Elastic stack][elk-stack] with Docker and Docker Compose.

It gives you the ability to analyze any data set by using the searching/aggregation capabilities of Elasticsearch and the visualization power of Kibana.

*:information source: The Docker images backing this stack include [Stack Features][stack-features] (formerly X-Pack) with [paid features][paid-features] enabled by default (see [How to disable paid features](#how-to-disable-paid-features) to disable them). **The [trial license][trial-license] is valid for 30 days**.*

Based on the official Docker images from Elastic:

* [Elasticsearch](https://github.com/elastic/elasticsearch/tree/master/distribution/docker)
* [Logstash](https://github.com/elastic/logstash/tree/master/docker)
* [Kibana](https://github.com/elastic/kibana/tree/master/src/dev/build/tasks/os_packages/docker_generator)

Other available stack variants:

* ['searchguard'](https://github.com/deviantony/docker-elk/tree/searchguard): Search Guard support

## Contents

1. [Requirements](#requirements)
   * [Host setup](#host-setup)
   * [SELinux](#selinux)
   * [Docker for Desktop](#docker-for-desktop)
     * [Windows](#windows)
     * [macOS](#macos)
2. [Usage](#usage)
```