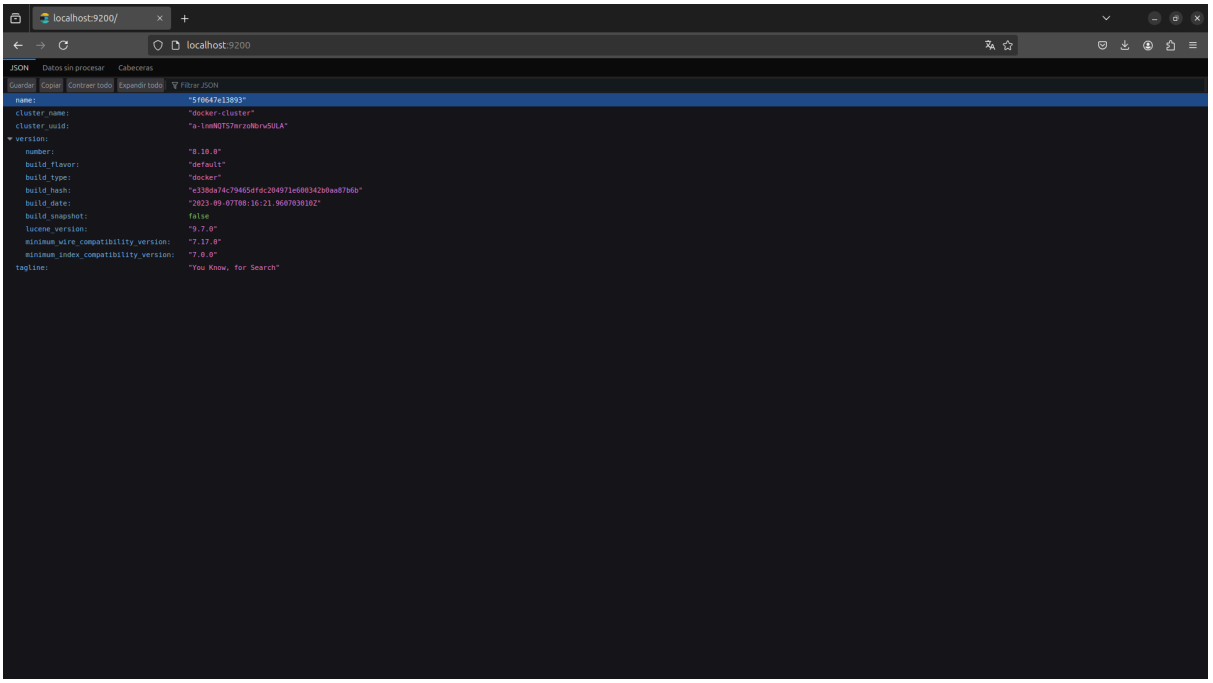
Parte 1: Configuración de Elasticsearch

1.Iniciar Elasticsearch con Docker:

Para iniciar el contenedor de Elasticsearch he utilizado el siguiente docker-compose.yml para iniciar el servicio:

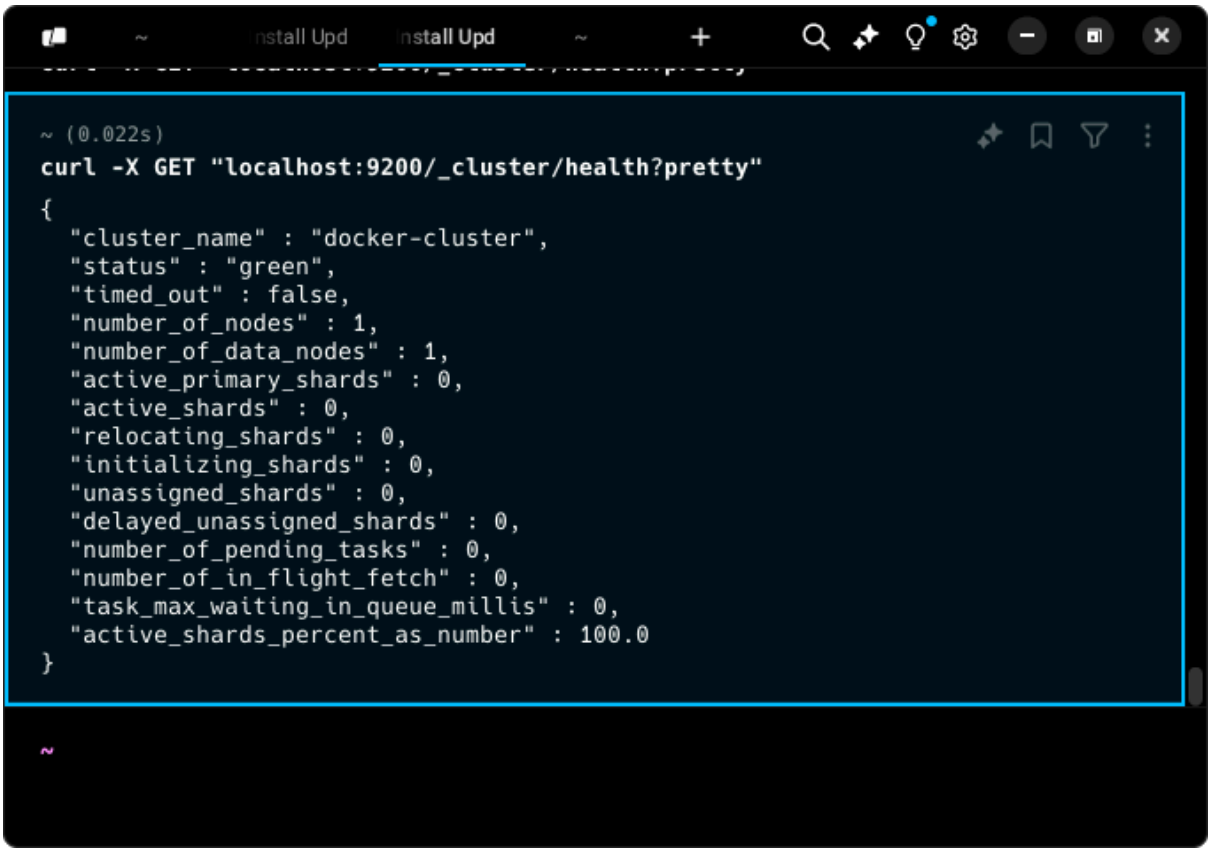
```
services:
 elastic_search:
   image: docker.elastic.co/elasticsearch/elasticsearch:8.10.0
   container_name: elastic_search
   environment:
     - discovery.type=single-node
     - xpack.security.enabled=false
     - ES_JAVA_OPTS=-Xms2g -Xmx2g # Hay que limitar la memoria de la
JVM
   ports:
     - 9200:9200
     - 9300:9300
   networks:
     - elastic_search_network
   volumes:
     - elastic_search_data:/usr/share/elasticsearch/data

volumes:
 elastic_search_data:
networks:
 elastic_search_network:
   driver: bridge
```

Con el comando docker compose up -d lo iniciamos en segundo plano,esperamos un par de minutos para acceder a él y en el navegador copiamos el siguiente link http://localhost:9200 ,para no tener problemas con autenticación y seguridad estableceremos los complementos de seguridad,monitoreo y alertas de xpack que contiene elasticsearch a false.



2. Explorando el estado del clúster:

Insertando el siguiente comando para saber el estado del cluster: curl -X GET "localhost:9200/_cluster/health?pretty" ,podemos comprobamos que el status es green por lo tanto saludable.



Parte 2: Indexación de datos

1. Crear un índice:

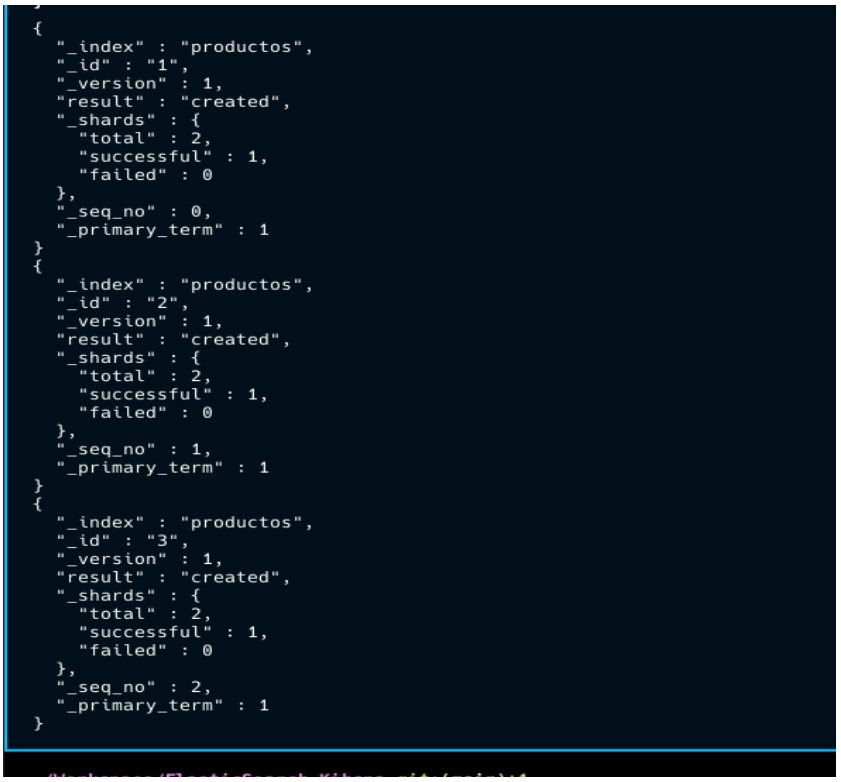
Con el siguiente comando curl -X PUT "localhost:9200/productos?pretty" crearemos un índice llamado productos con el siguiente resultado:

2. Indexar documentos:

Agregamos con el siguiente comando documentos(productos) a este índice que acabamos de crear:

```
# Documento 1
curl -X POST "localhost:9200/productos/_doc/1?pretty" \
-H "Content-Type: application/json" \
-d '{
  "nombre": "Libro de Elasticsearch",
  "precio": 25.99,
  "categoria": "libros"
}'

# Documento 2
curl -X POST "localhost:9200/productos/_doc/2?pretty" \
-H "Content-Type: application/json" \
-d '{
  "nombre": "Portátil",
  "precio": 999.99,
  "categoria": "tecnología"
}'

# Documento 3
curl -X POST "localhost:9200/productos/_doc/3?pretty" \
-H "Content-Type: application/json" \
-d '{
  "nombre": "Auriculares",
  "precio": 49.99,
  "categoria": "tecnología"
}'
```

Con el siguiente resultado:



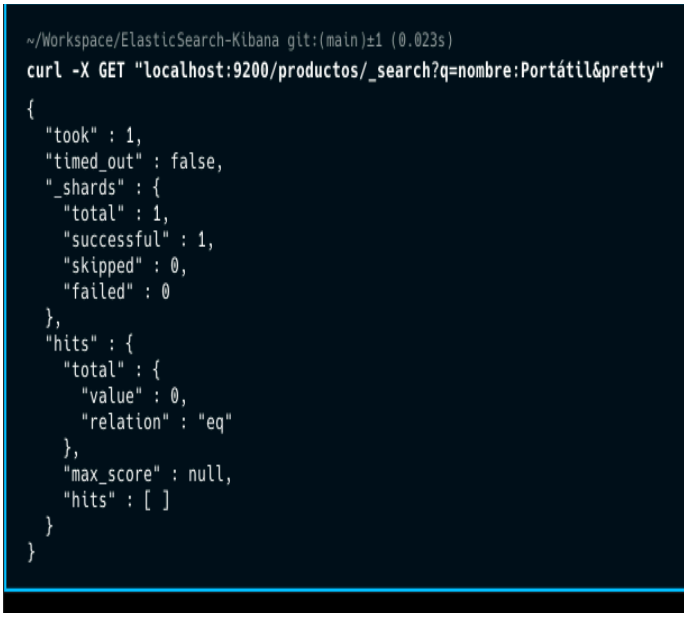
```
{
  "_index" : "productos",
  "_id" : "1",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
{
  "_index" : "productos",
  "_id" : "2",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 1,
  "_primary_term" : 1
}
{
  "_index" : "productos",
  "_id" : "3",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 2,
  "_primary_term" : 1
}
```

3. Consultar documentos indexados:

Ahora hacemos la búsqueda de estos 3 documentos añadidos recientemente con la siguiente petición curl -X GET "localhost:9200/productos/_search?pretty" ,con el siguiente resultado:

```
~/Workspace/ElasticSearch-Kibana git:(main)±1 (0.083s)
curl -X GET "localhost:9200/productos/_search?pretty"
{
  "took" : 55,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 3,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "productos",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "nombre" : "Libro de Elasticsearch",
          "precio" : 25.99,
          "categoria" : "libros"
        }
      },
      {
        "_index" : "productos",
        "_id" : "2",
        "_score" : 1.0,
        "_source" : {
          "nombre" : "Portátil",
          "precio" : 999.99,
          "categoria" : "tecnología"
        }
      },
      {
        "_index" : "productos",
        "_id" : "3",
        "_score" : 1.0,
        "_source" : {
          "nombre" : "Auriculares",
          "precio" : 49.99,
          "categoria" : "tecnología"
        }
      }
    ]
  }
}
```

4. Buscar por nombre:
A continuación vamos a hacer una búsqueda con un parámetro en este caso del nombre de producto con la siguiente petición
curl -X GET "localhost:9200/productos/_search?q=nombre:Portátil&pretty"



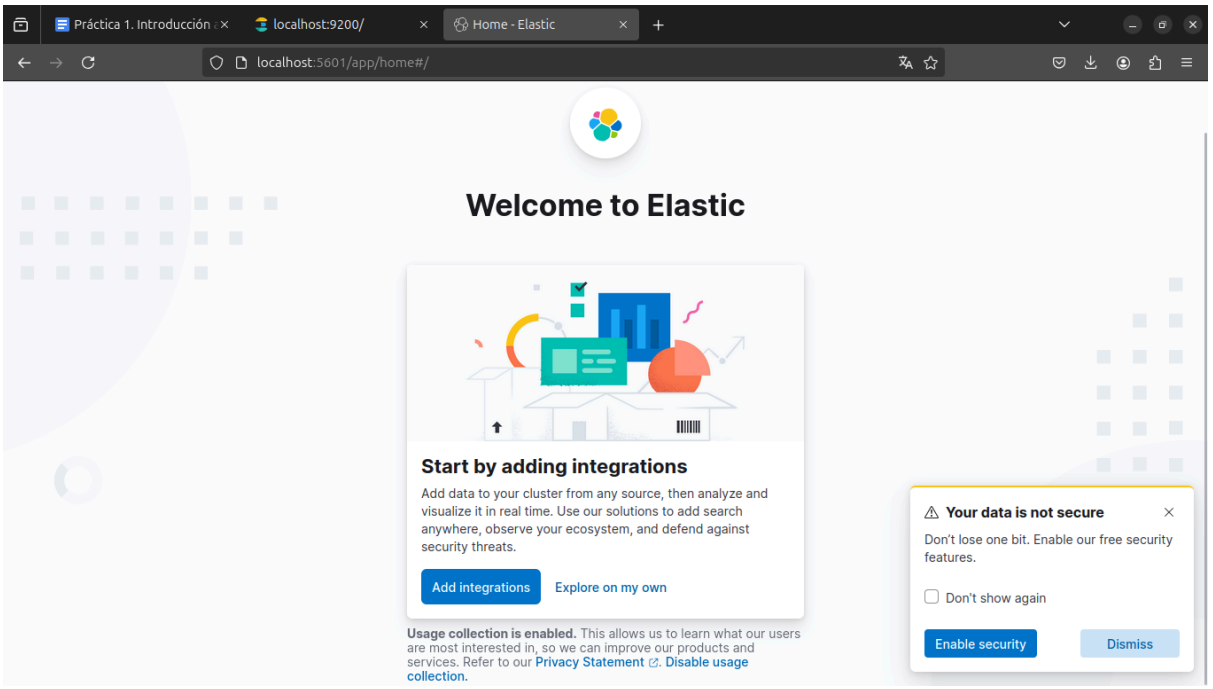
Parte 3: Uso de Kibana para visualización

1. Iniciar Kibana con Docker:
Como hicimos con Elasticsearch,ahora hacemos con este servicio y lo incluimos dentro de la red existente de docker para que esté en contacto con Elasticsearch,paramos los servicios en ejecución con docker compose stop ,y de nuevo hacemos el docker compose up -d ,es importante que en el environment pongamos el host de elastic search al que se tiene que conectar,con esto último nos quedaría el siguiente docker-compose.yml final:

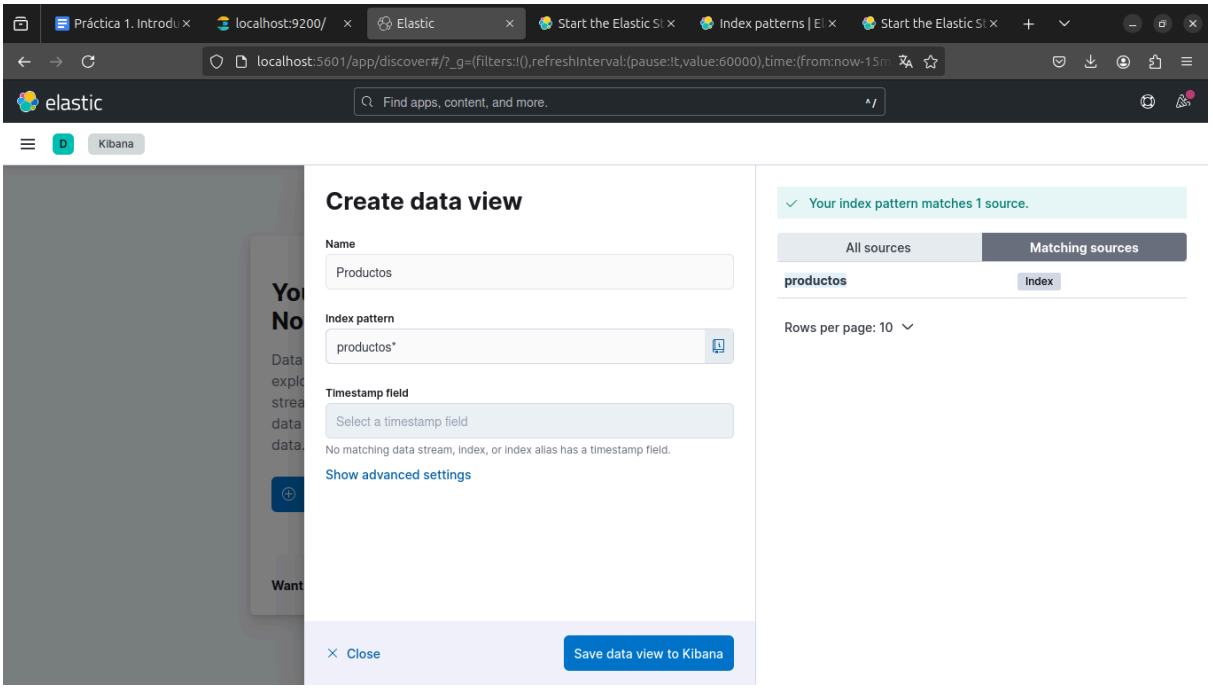
```
services:
 elastic_search:
   image: docker.elastic.co/elasticsearch/elasticsearch:8.10.0
   container_name: elastic_search
   environment:
     - discovery.type=single-node
     - xpack.security.enabled=false
     - ES_JAVA_OPTS=-Xms2g -Xmx2g # Limitar la memoria de la JVM
   ports:
     - 9200:9200
     - 9300:9300
   networks:
     - elastic_search_network
   volumes:
     - elastic_search_data:/usr/share/elasticsearch/data
 kibana:
   image: docker.elastic.co/kibana/kibana:8.10.1
```

```
    container_name: kibana
    environment:
      - ELASTICSEARCH_HOSTS=http://elastic_search:9200
    ports:
      - 5601:5601
    networks:
      - elastic_search_network
volumes:
  elastic_search_data:
networks:
  elastic_search_network:
    driver: bridge
```

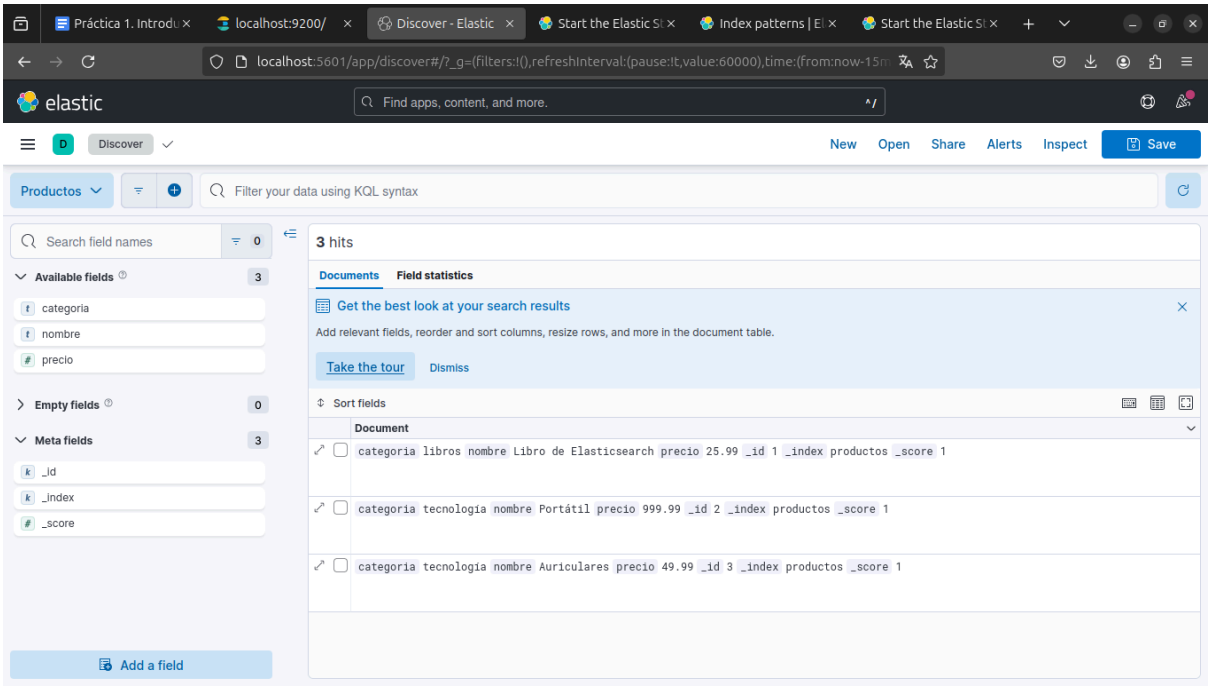
Por lo tanto con el siguiente enlace http://localhost:5601 podremos acceder a elastic search de manera visual a través de kibana.



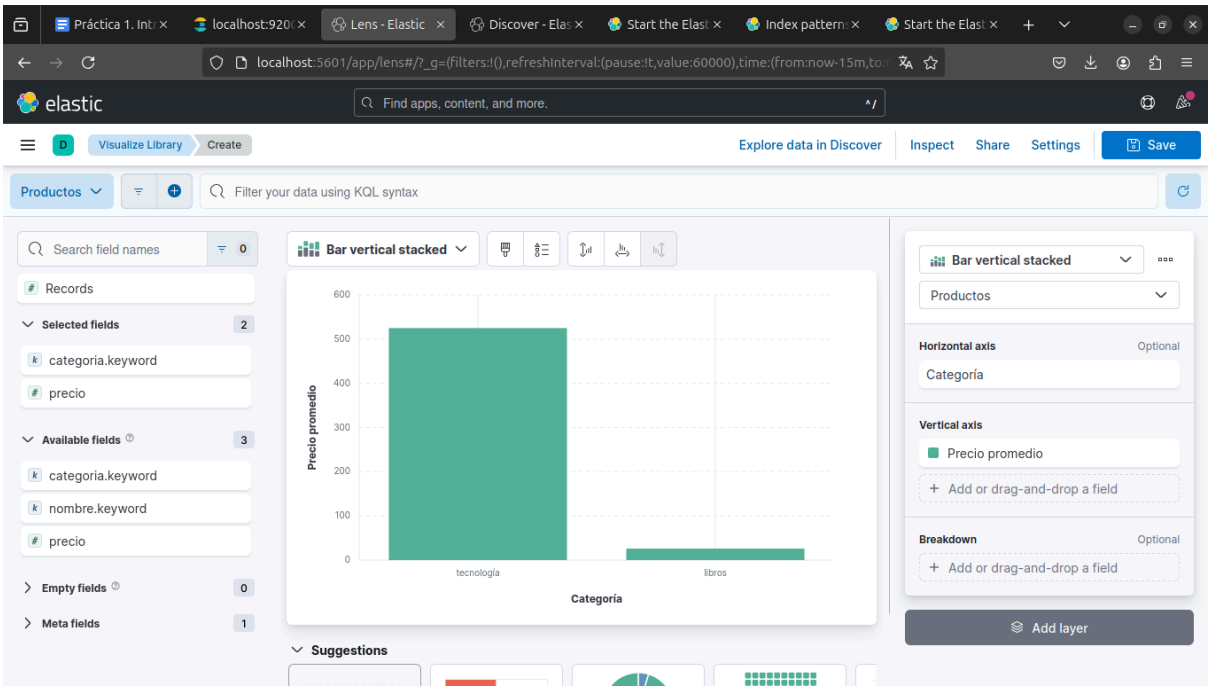
3 y 4.Creamos un data view para poder visualizar el contenido de nuestros documentos creados anteriormente a la misma vez que creamos el index pattern:



Y a continuación podremos tener nuestros documentos de manera visual:



5. Visualización simple
En el eje Y creamos el precio medio con la función disponible de Average y en el eje X seleccionamos el field categoría y nos saldrá la siguiente gráfica:



Parte 4: Agregaciones y consultas avanzadas

1. Realizar agregaciones en Elasticsearch:
Precio promedio de todos los productos en la categoría “tecnología”:

```
curl -X GET "localhost:9200/productos/_search?pretty" \
-H 'Content-Type: application/json' \
-d '{
  "size": 0,
  "aggs": {
        "precio_promedio_tecnologia": {
        "avg": {
        "field": "precio"
        }
        }
  },
  "query": {
        "match": {
        "categoria": "tecnología"
        }
  }
}'
```

Nos saldrá el siguiente resultado:

```
~/Workspace/ElasticSearch-Kibana git:(main)±1 (0.023s)
curl -X GET "localhost:9200/productos/_search?pretty" \
-H 'Content-Type: application/json' \
-d '{
  "size": 0,
  "aggs": {
    "precio_promedio_tecnologia": {
      "avg": {
        "field": "precio"
      }
    }
  },
  "query": {
    "match": {
      "categoria": "tecnología"
    }
  }
}'
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 2,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "precio_promedio_tecnologia" : {
      "value" : 524.9899959564209
    }
  }
}
```

2. Buscar productos por rango de precios y ordenado ascendentemente:

```
curl -X GET "localhost:9200/productos/_search?pretty" \
-H 'Content-Type: application/json' \
-d '{
  "query": {
          "range": {
          "precio": {
          "gte": 20,
          "lte": 100
          }
          }
  },
  "sort": [
          { "precio": "asc" }
  ]
}'
{
  "took" : 6,
  "timed_out" : false,
  "_shards" : {
          "total" : 1,
          "successful" : 1,
          "skipped" : 0,
          "failed" : 0
```

```
},
"hits" : {
        "total" : {
        "value" : 4,
        "relation" : "eq"
        },
        "max_score" : null,
        "hits" : [
        {
        "_index" : "productos",
        "_id" : "1AgTK5MBFiW2Z6KMmHog",
        "_score" : null,
        "_source" : {
        "nombre" : "Libro de Elasticsearch",
        "precio" : 25.99,
        "categoria" : "libros"
        },
        "sort" : [
        25.99
        ]
        },
        {
        "_index" : "productos",
        "_id" : "1",
        "_score" : null,
        "_source" : {
        "nombre" : "Libro de Elasticsearch",
        "precio" : 25.99,
        "categoria" : "libros"
        },
        "sort" : [
        25.99
        ]
        },
        {
        "_index" : "productos",
        "_id" : "3",
        "_score" : null,
        "_source" : {
        "nombre" : "Auriculares",
        "precio" : 49.99,
        "categoria" : "tecnología"
        },
        "sort" : [
        49.99
        ]
        }
        ]
```

```
  }
}
```

Parte 5: Importar un Documento JSON al Índice en Elasticsearch

1.Para poder importar este archivo JSON ,se necesita transformar el json a la estructura que espera Elasticsearch ya que cada documento necesita una línea de metadata seguida de una línea de datos,por lo tanto este documento JSON:

```
[
   {
   "id": 4,
   "nombre": "Laptop",
   "precio": 750,
   "categoria": "Electrónica"
   },
   {
   "id": 5,
   "nombre": "Teléfono",
   "precio": 300,
   "categoria": "Electrónica"
   },
   {
   "id": 6,
   "nombre": "Silla",
   "precio": 100,
   "categoria": "Muebles"
   }
 ]
```

Se tiene que convertir en este formato :

```
{"index": { "_id": "4" } }
{ "id": 4, "nombre": "Laptop", "precio": 750, "categoria":
"Electrónica" }
{ "index": { "_id": "5" } }
{ "id": 5, "nombre": "Teléfono", "precio": 300, "categoria":
"Electrónica" }
{ "index": { "_id": "6" } }
{ "id": 6, "nombre": "Silla", "precio": 100, "categoria": "Muebles" }
```

Y nos saldrá el siguiente resultado :

6. Verifica que los datos se han importado correctamente:

Resultado de la query curl -X GET "localhost:9200/productos/_search?pretty" \
-H 'Content-Type: application/json' \
-d '{
  "query": {
        "match_all": {}
  }
}'

{
  "took" : 0,
  "timed_out" : false,
  "_shards" : {
        "total" : 1,
        "successful" : 1,
        "skipped" : 0,
        "failed" : 0
  },
  "hits" : {
        "total" : {
        "value" : 6,
        "relation" : "eq"
        },
        "max_score" : 1.0,
        "hits" : [
        {
        "_index" : "productos",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
        "id" : 1,
        "nombre" : "Laptop",
        "precio" : 750,
        "categoria" : "Electrónica"
        }
        },

```
{
"_index" : "productos",
"_id" : "2",
"_score" : 1.0,
"_source" : {
"id" : 2,
"nombre" : "Teléfono",
"precio" : 300,
"categoria" : "Electrónica"
}
},
{
"_index" : "productos",
"_id" : "3",
"_score" : 1.0,
"_source" : {
"id" : 3,
"nombre" : "Silla",
"precio" : 100,
"categoria" : "Muebles"
}
},
{
"_index" : "productos",
"_id" : "4",
"_score" : 1.0,
"_source" : {
"nombre" : "Libro de Elasticsearch",
"precio" : 25.99,
"categoria" : "libros"
}
},
{
"_index" : "productos",
"_id" : "5",
"_score" : 1.0,
"_source" : {
"nombre" : "Portátil",
"precio" : 999.99,
"categoria" : "tecnología"
}
},
{
"_index" : "productos",
"_id" : "6",
"_score" : 1.0,
"_source" : {
"nombre" : "Auriculares",
```

```
        "precio" : 49.99,
        "categoria" : "tecnología"
        }
        }
        ]
  }
}
```

7. Integración con python :

1.Instalación de entorno.
Para empezar tendrás que tener instalado Python en tu sistema operativo,después para la ejecución del cliente de python tendrás que ejecutar el siguiente comando para iniciar un nuevo entorno virtual para poder ejecutar las dependencias de elasticsearch de python.
python3 -m venv entorno

Después lo tendremos que activar e instalar la dependencia de elasticsearch:

source entorno/bin/activate
pip install elasticsearch

Para probar la conexión e inserción de datos ejecutaremos el script en la carpeta raíz con el siguiente comando:
python3 elastic_search.py

Y nos saldrán los siguientes Productos creados en un nuevo índice llamado productos_python y sus respectivas consultas:

```
(entorno) jesus@jesus-MS-7C94:~/Workspace/ElasticSearch-Kibana$ python3 elastic_search.py

Todos los productos:
ID: 1, Producto: {'nombre': 'Libro de Elasticsearch', 'precio': 25.99, 'categoria': 'libros'}
ID: 2, Producto: {'nombre': 'Portátil', 'precio': 999.99, 'categoria': 'tecnología'}
ID: 3, Producto: {'nombre': 'Auriculares', 'precio': 49.99, 'categoria': 'tecnología'}
ID: 4, Producto: {'id': 4, 'nombre': 'Laptop', 'precio': 750, 'categoria': 'Electrónica'}
ID: 5, Producto: {'id': 5, 'nombre': 'Teléfono', 'precio': 300, 'categoria': 'Electrónica'}
ID: 6, Producto: {'id': 6, 'nombre': 'Silla', 'precio': 100, 'categoria': 'Muebles'}

Productos de tecnología:
ID: 2, Producto: {'nombre': 'Portátil', 'precio': 999.99, 'categoria': 'tecnología'}
ID: 3, Producto: {'nombre': 'Auriculares', 'precio': 49.99, 'categoria': 'tecnología'}

Productos entre 20 y 100 euros:
ID: 1, Producto: {'nombre': 'Libro de Elasticsearch', 'precio': 25.99, 'categoria': 'libros'}
ID: 3, Producto: {'nombre': 'Auriculares', 'precio': 49.99, 'categoria': 'tecnología'}
ID: 6, Producto: {'id': 6, 'nombre': 'Silla', 'precio': 100, 'categoria': 'Muebles'}

Tecnología entre 0 y 500 euros:
ID: 3, Producto: {'nombre': 'Auriculares', 'precio': 49.99, 'categoria': 'tecnología'}
```

8.Integración con ElasticVue.

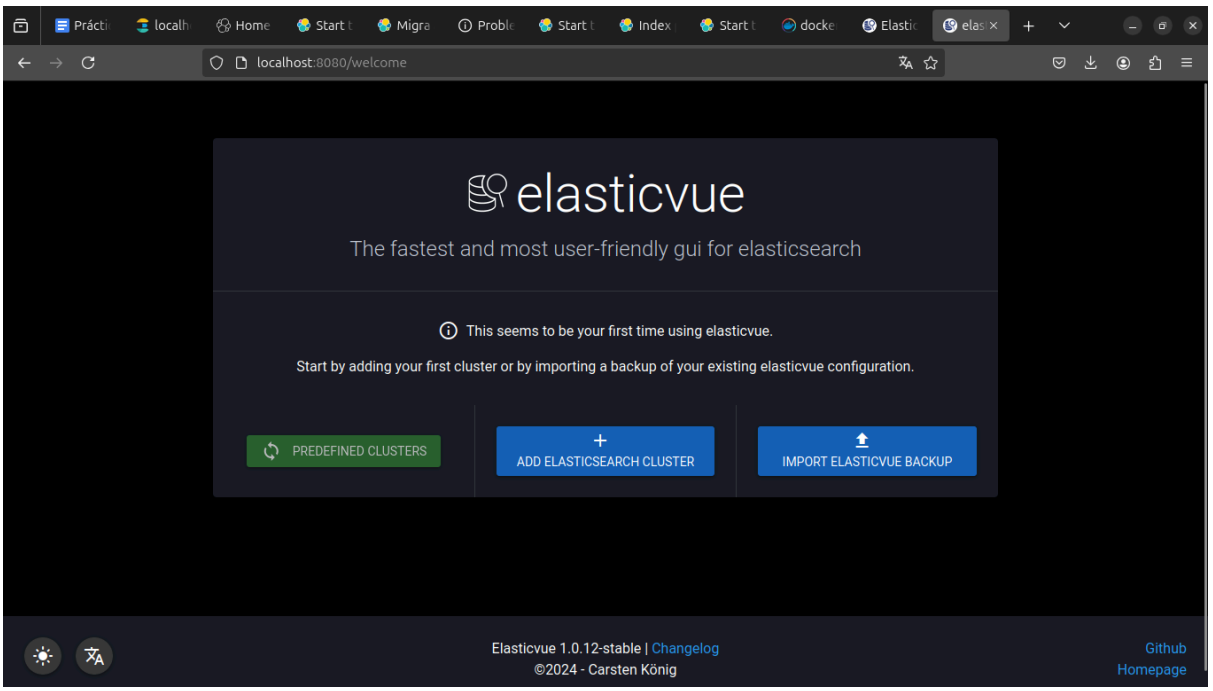
Para esta integración vamos a utilizar un contenedor ya que manera sencilla podemos utilizarlo con unas pocas líneas en nuestro docker-compose.yml a partir de una imagen y que estaría disponible en el puerto 8080 para la gestión de datos de Elasticsearch quedando el docker-compose.yml de la siguiente manera:

```
services:
 elastic_search:
   image: docker.elastic.co/elasticsearch/elasticsearch:8.16.0
   container_name: elastic_search
   environment:
     - discovery.type=single-node
     - xpack.security.enabled=false
     - ES_JAVA_OPTS=-Xms2g -Xmx2g # Limitar la memoria de la JVM
     - http.cors.enabled=true
     - http.cors.allow-origin=http://localhost:8080
     -
http.cors.allow-headers=X-Requested-With,Content-Type,Content-Length,Au
thorization
   ports:
     - 9200:9200
     - 9300:9300
   networks:
     - elastic_search_network
   volumes:
     - elastic_search_data:/usr/share/elasticsearch/data
 kibana:
   image: docker.elastic.co/kibana/kibana:8.10.1
   container_name: kibana
   environment:
     - ELASTICSEARCH_HOSTS=http://elastic_search:9200
   ports:
     - 5601:5601
   networks:
     - elastic_search_network

 elasticvue:
     image: cars10/elasticvue
     ports:
       - "8080:8080"
     depends_on:
       - elastic_search
```

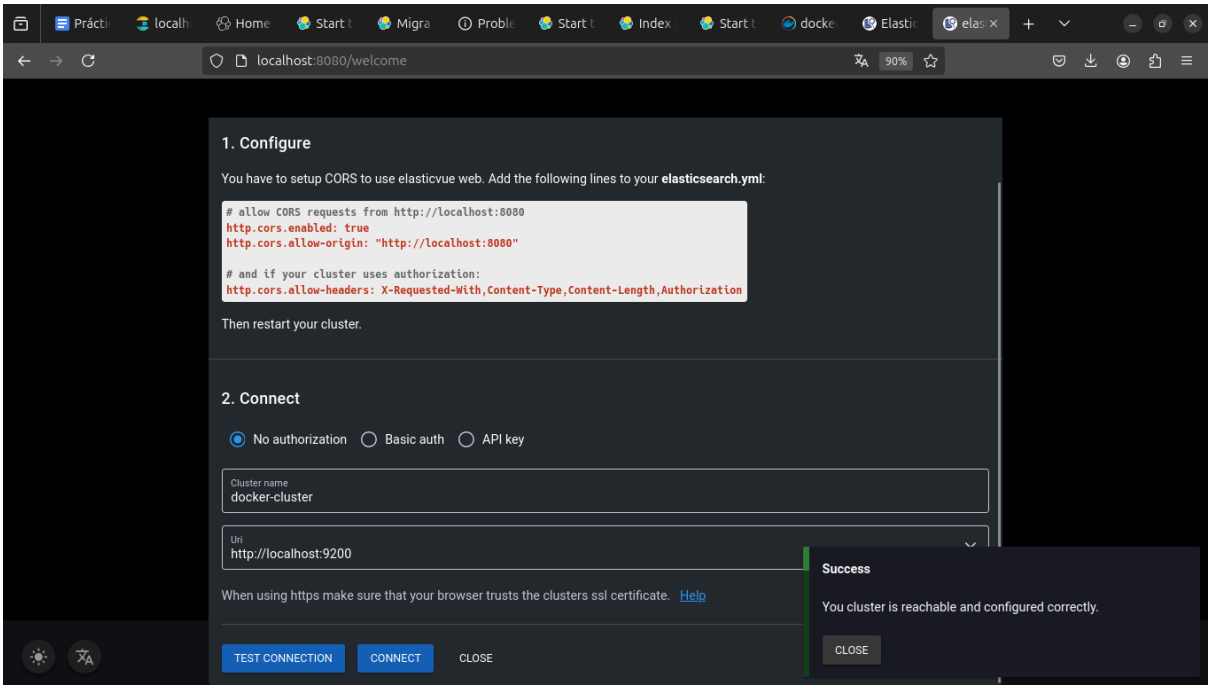
```
volumes:
 elastic_search_data:
networks:
 elastic_search_network:
   driver: bridge
```

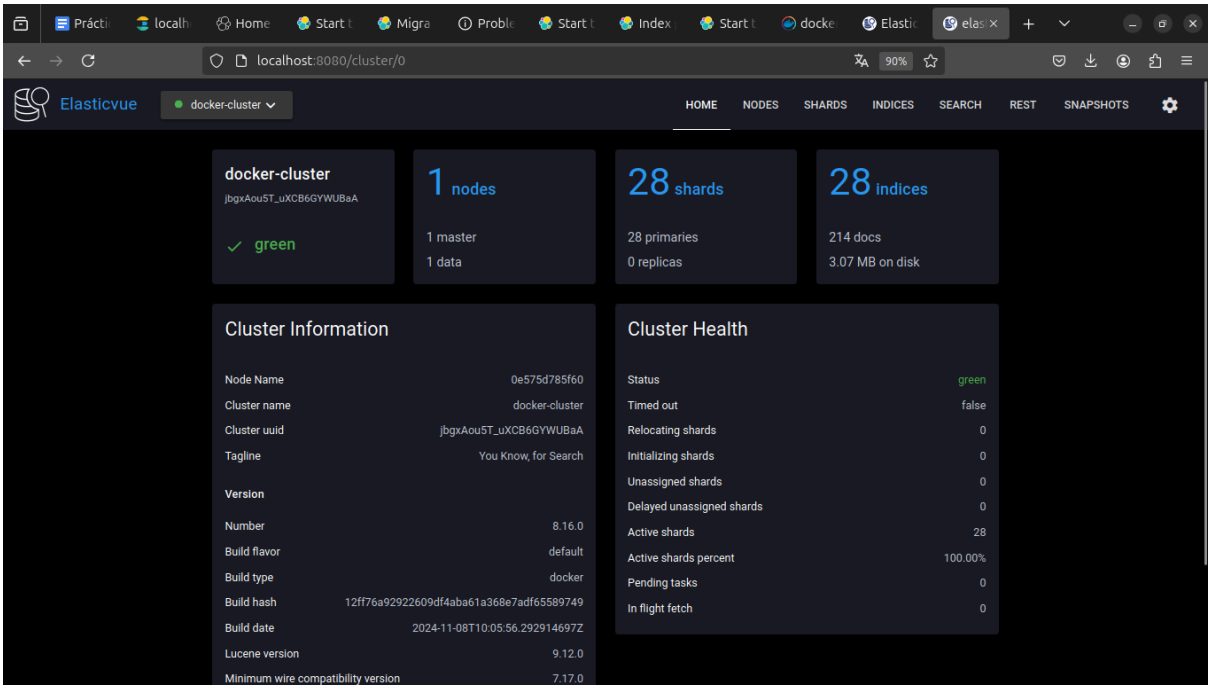
Para que este servicio funcione tendremos que parar los contenedores de nuestro proyecto con docker compose stop y después volverlos a iniciar con docker compose up -d entonces estará disponible en http://localhost:8080 pudiendo hacer las siguientes tareas sobre elasticsearch:

- Ver y gestionar índices
- Realizar consultas
- Ver el estado del cluster
- Gestionar documentos
- Ver estadísticas
- Ejecutar operaciones de mantenimiento

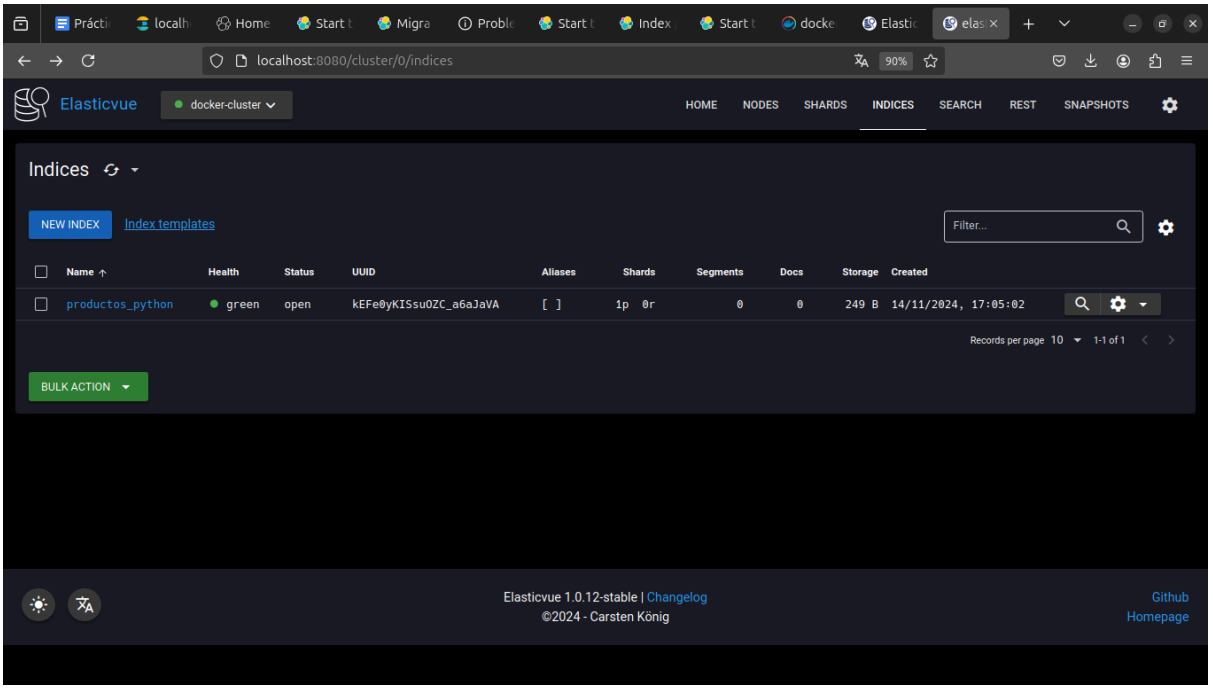


Clicaremos en ADD ELASTICSEARCH CLUSTER y nos pedirá que indiquemos el cluster y que por defecto se conectará al puerto 9200 de nuestro localhost,las advertencias que indica para que permita conectarse servicios a el cluster ya las hemos añadido a nuestro docker-compose.yml en el environment de nuestro cluster de elastic_search , por lo tanto probaremos la conexión y nos conectaremos:





Entonces podemos ver el último indice que creamos anteriormente:



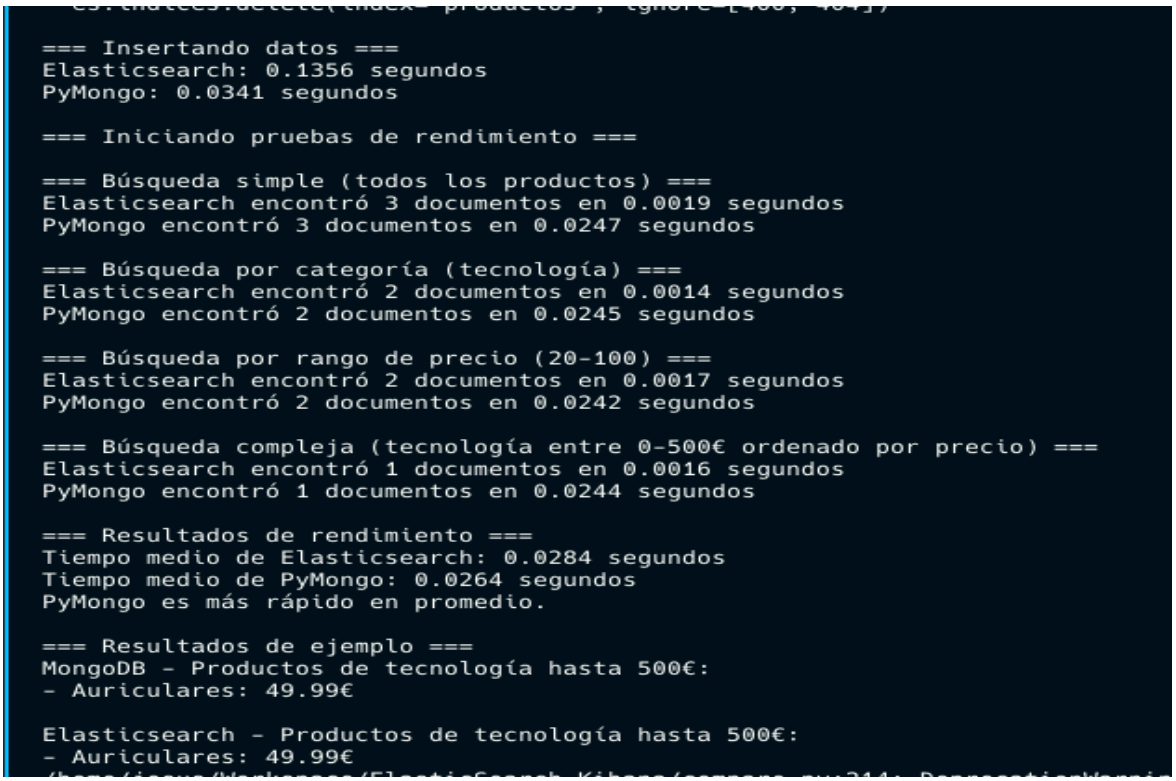
9.Comparación con MongoDBAtlas y Elastic.

En cuestión de consultas e inserción de datos con los mismos datos.
Para ello tendremos que instalar en nuestro entorno virtual anterior PyMongo para la creación de documentos y consultas eficientes.

pip install pymongo

Y para la ejecución del script y obtener los resultados ejecutamos el siguiente comando SIEMPRE Y CUANDO EL ENTORNO ESTE ACTIVADO

python3 compare.py



Con la media sacada según script podemos concluir que Pymongo es más rápido y eficiente para la inserción y consulta de datos en comparación a Elasticsearch.