Logstash y Kibana

Administración de Sistemas

Unai Lopez Novoa

unai.lopez@ehu.eus

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Contenido

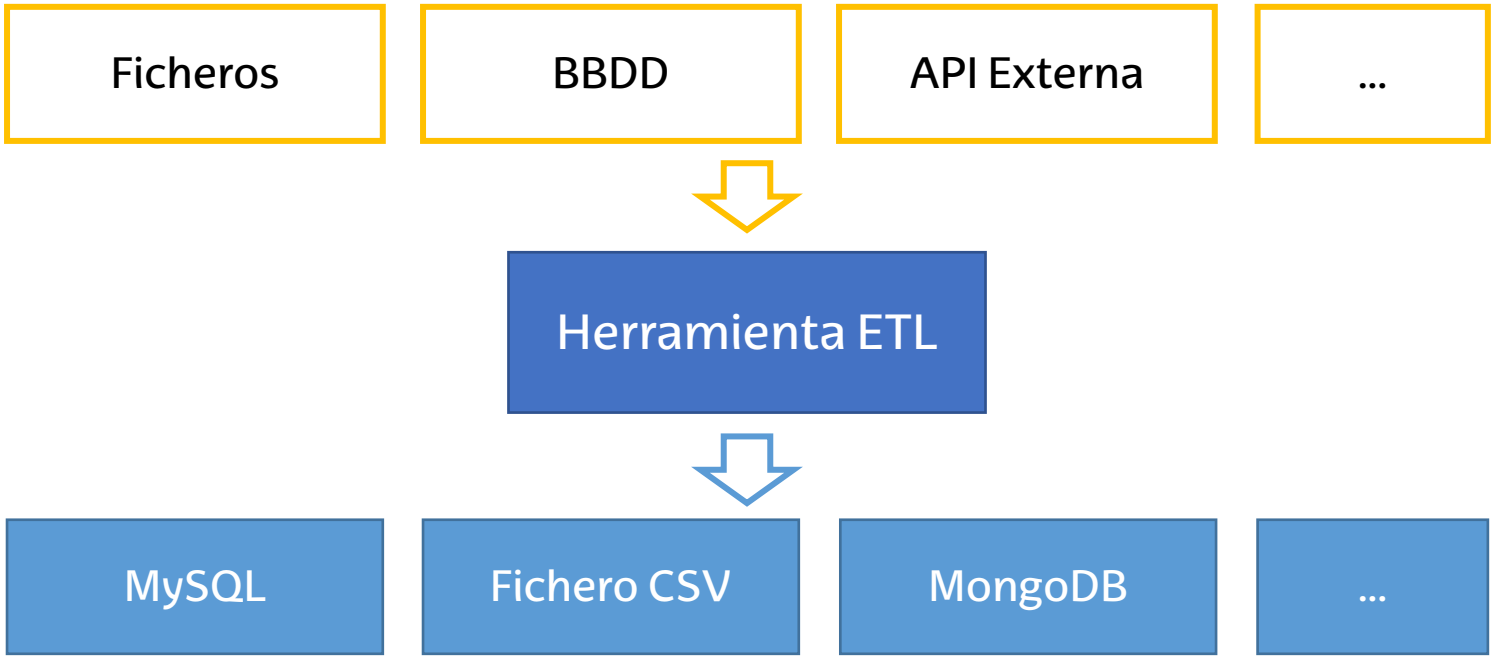
Introducción

- En sistemas con muchos componentes, los *logs* suelen tener diferente formato, estructura y tamaño.
  - Y no todo su contenido es relevante.

- Es necesario una herramienta para recopilar y filtrar *logs* antes de almacenarlos.

Introducción

- Las herramientas ETL (*Extract, Transform, Load*) permiten recopilar y filtrar datos de diferentes fuentes.



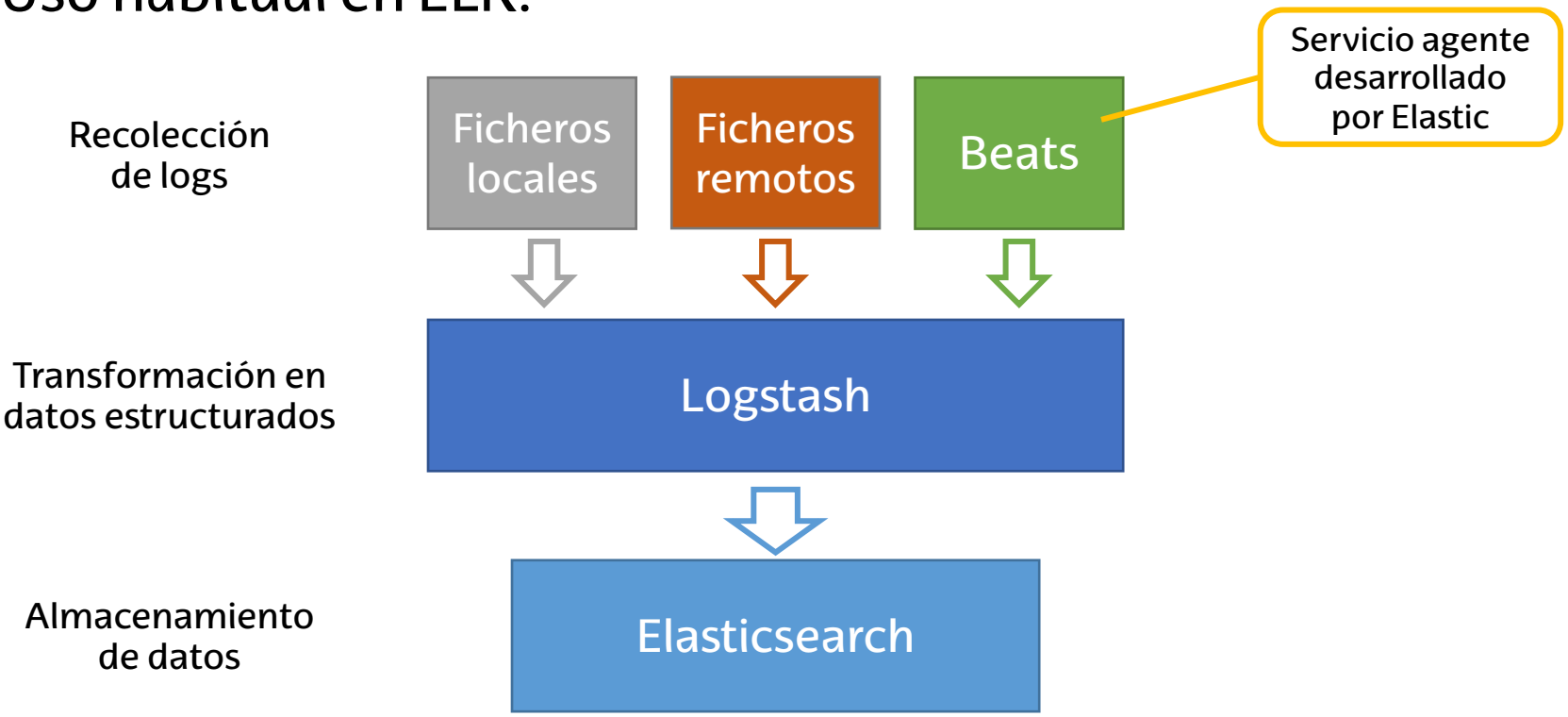
Logstash

- Logstash es la herramienta ETL de la pila ELK
  - Escalable a múltiples nodos
  - Tolerante a fallos
  - Web: https://www.elastic.co/es/logstash

- Se puede utilizar para:
  - Parsear, transformar y filtrar los datos que recoge.
  - Estructurar datos no estructurados.
  - Anonimizar datos.
  - Geo-localizar datos.
  - ...

Logstash

- Uso habitual en ELK:



Instalación

- Instalación de ELK con Docker Compose
  - Fichero YAML en: https://github.com/ulopeznovoa/ELK-8-setup

```
services:
  logstash:
    …
    volumes:
      - ./pipeline:/usr/share/logstash/pipeline
    …
  elasticsearch:
    …
  kibana:
    …
```

Bind mount para compartir configuración con el contenedor

  - Pila ELK requiere, al menos, 1 CPU dedicada y 6 GB de RAM.

Configuración

- Fichero logstash.conf
  - Por defecto, se busca en /usr/share/logstash/pipeline
  - Indica el "pipeline" de procesado de datos en 3 partes
    - Input: Orígenes de datos
    - Filter: Transformaciones sobre los datos
    - Output: A donde enviar los datos
  - Esquema general:

```
input {
        …
}
filter {
        …
}
output {
        …
}
```

Configuración

- Fichero logstash.conf
  - Sección **input**: plug-ins para recogida de datos
    - File: plug-in para monitorizar ficheros locales

```
input {
        file {
              path => “/home/unai/access_log”
              start_position => “beginning”
              sincedb_path => "/dev/null"
        }
}
```

Ruta al fichero a monitorizar

Monitorizar desde el comienzo del fichero

Opcional. Ruta a una BBDD que almacena la última línea leída del fichero. Al asignar /dev/null, no hay BBDD.

Configuración

- Fichero logstash.conf
  - Sección **input**: plug-ins para recogida de datos
    - jdbc: acceder una tabla de una BBDD MySQL

```
input {
        jdbc {
                jdbc_connection_string => "jdbc:mysql//localhost:3306/nombre-tabla"
                jdbc_user => "usuario-mysql"
                jdbc_password => "contraseña-mysql"
                jdbc_driver_library => "ruta-conector-java-mysql.jar"
                jdbc_driver_class => "com.mysql.jdbc.Driver"
                statement => "SELECT * FROM tabla"
        }
}
```

IP y puerto de la BBDD

Sentencia SQL que queramos para extraer datos

Configuración

- Fichero logstash.conf
  - Sección **input**: plug-ins para recogida de datos
    - tcp: Recibir desde un puerto TCP
    - udp: Recibir desde un puerto UDP
    - http: Recibir una conexión HTTP
    - Se les puede añadir un campo "codec" para indicar el formato a recibir

```
input {
        http {
              port => '9563'
              codec => "json"
        }
}
```

Puerto en escucha

Configuración

- Fichero logstash.conf
  - Sección **filter**: transformaciones en los datos.
    - JSON: Parsear campos de una estructura JSON
      - Si el mensaje se ha enviado como texto plano, permite indicar cómo se trata la parte que es JSON.
      - No es necesario si el mensaje completo es un JSON y se ha indicado "codec => json" en la parte "input".
      - Ejemplo:

```
filter {
        json {
                source => "message"
                target => "doc"
        }
}
```

Indica en qué campo del mensaje está el JSON a parsear

Indica en qué campo del documento generado se va a ubicar el JSON parseado

Configuración

- Fichero logstash.conf
  - Sección **filter**: transformaciones en los datos.
    - CSV: Formato Comma Separated Value (CSV)
      - Ejemplo junto con el plug-in "file" de entrada

```
input {
        file {
                path => "/home/unai/data.csv"
                start_position => "beginning" }
}
filter {
        csv {
                separator => ","
                skip_header => "true"
                columns => ["Nombre", "Autor", "Isbn"]
        }
}
```

Opcional. Omitir 1ª fila con los nombres de las columnas.

Configuración

- Fichero logstash.conf
  - Sección **filter**: transformaciones en los datos.
    - Mutate: Transformaciones sobre los datos
      - Más información:
        - https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html
      - Ejemplo:

```
filter {
        if [version] == "3.11" {
                drop {}
        }
        mutate {
                remove_field => ["version", "nombre"]
        }
}
```

No procesar los objetos JSON cuyo campo "version" sea "3.11"

Eliminar los campos "version" y "nombre" antes de enviar el objeto.

Configuración

- Fichero logstash.conf
  - Sección **output**: donde enviar la información.
    - Ejemplo: enviar a 2 sitios a la vez:
      - Instancia de Elasticsearch alcanzable en localhost:9200, los datos se escriben en el índice "mi-índice".
      - Salida estándar (consola) utilizando el formato "rubydebug"

```
output {
        elasticsearch {
                hosts => [ "localhost:9200" ]
                index => "mi-índice"
        }
        stdout {
                codec => rubydebug
        }
}
```

Opcional. Si no se especifica, Logstash elige un nombre basado en la fecha.

Opcional. Especifica formato para el texto.

Configuración

- Fichero logstash.conf
  - Se pueden procesar múltiples entradas de manera selectiva.
    - Utilizar la etiqueta “type” en la sección input.
    - Segregar filtros y salidas con sentencias “if”.
    - Ejemplo:

```
input {
  tcp {
       port => 5045
       type => 'datos-tcp’  }
  udp {
       port => 5045
       type => 'datos-udp’  }
}
filter {
  if [type] == 'datos-tcp’ {
       grok {
           match => ["message" , " … "] }
       }
  else if [type] == 'datos-udp’ {
       grok {
           …
```

Configuración

- Fichero logstash.conf
  - Hay muchos plug-ins y opciones de configuración
  - Input:
    - https://www.elastic.co/guide/en/logstash/current/input-plugins.html
  - Filtros:
    - https://www.elastic.co/guide/en/logstash/current/filter-plugins.html
  - Output:
    - https://www.elastic.co/guide/en/logstash/current/output-plugins.html

Configuración

- Si hay errores en la configuración del pipeline:
  - Logstash lo mostrará en su log de arranque y abortará su inicio.

- Ejemplo:
  - Error en logstash.conf: *output* mal escrito.

```
10      …
11      outupt{
12        stdout{}
13      }
```

  - Mensaje en log de arranque:

```
            …
logstash  | [2023-12-01T11:00:30,040][ERROR][logstash.agent ] Failed to execute
action {:action=>LogStash::PipelineAction::Create/pipeline_id:main,
:exception=>"LogStash::ConfigurationError", :message=>"Expected one of [
\\t\\r\\n], \"#\", \"input\", \"filter\", \"output\" at line 11, column 1 (byte
74) after ", :backtrace=>[" … '"]}
            …
```

Ejercicio 1

- Los siguientes JSON representan 2 líneas de Syslog:

```
{
 "timestamp": "Nov 28 07:55:04",
 "device" : "server",
 "process" : "kernel",
 "event-number" : "1150.308049",
 "message" : "port 2(veth78fa91b) entered blocking state"
}
```

```
{
 "timestamp": "Nov 28 07:55:05",
 "device" : "server",
 "process" : "systemd-networkd",
 "event-number" : "730",
 "message" : "Gained IPv6LL"
}
```

  - *Ver enunciado del ejercicio en siguiente diapositiva*

Ejercicio 1

- Configurar pipeline Logstash:
  - Recibir datos a través de conexiones HTTP en el puerto 9900.
    - Formato: JSON
  - Escribir cada objeto JSON en el índice “mis-logs” de Elasticsearch.
- Iniciar Logstash y Elasticsearch con Compose
  - Redirigir puerto 9900 para Logstash
- Enviar los objetos JSON a Logstash.
  - Mostrados en la diapositiva anterior.
  - Utilizar curl u otro cliente REST.
- Verificar que los datos están en el índice “mis-logs” de Elasticsearch.

Logstash

- Generalmente, los mensajes de *log* no son texto estructurado.
  - Ejemplo: Fragmento de la salida de syslog
    - El mensaje de cada línea tiene un formato diferente

```
Nov 28 10:58:06 as-vm dockerd[865]: time="2023-11-28T10:58:06.710552620Z" level=error
msg="attach failed with error: error attaching stdout stream: write unix /run/docker.sock-
>@: write: broken pipe"
Nov 28 10:58:06 as-vm kernel: [ 3456.685240] veth6170430: renamed from eth0
Nov 28 10:58:06 as-vm systemd-networkd[727]: veth6b43ad5: Lost carrier
```

  - Es conveniente estructurar el texto plano para poder analizarlo posteriormente con más facilidad.

Grok

- Grok es un plug-in que permite detectar patrones comunes de texto en un mensaje.

- Contiene 71 patrones.
  - P.e.: dirección IP, URL, dirección de e-mail, ...
  - Listado: https://github.com/logstash-plugins/logstash-patterns-core/blob/main/patterns/ecs-v1/grok-patterns

- Internamente utiliza expresiones regulares.
  - P.e.: expresión regular para detectar direcciones de e-mail[1]:

```
^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$
```

[1]Obtenida de: https://regexr.com/3e48o

Grok

- Sintaxis genérica de Grok:

```
%{PATRON:etiqueta}
```

  - donde:
    - PATRON El patrón que buscamos
    - etiqueta Contendrá el texto coincidente con el patrón

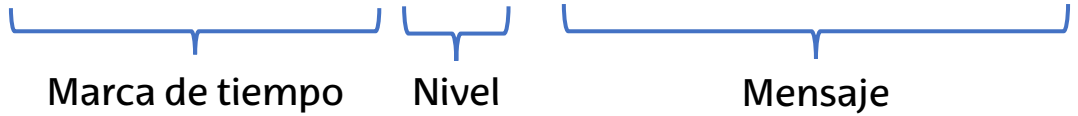
  - Ejemplo:
    - Detectar una dirección de e-mail en una secuencia de texto.

```
%{EMAILADDRESS:dir_email}
```

Grok

- Considerando una línea genérica de Log:

```
Nov 28 10:58:06 debug: This is a log message
```



- El patrón Grok a crear sería el siguiente:

```
%{SYSLOGTIMESTAMP:tiempo} %{LOGLEVEL:nivel} %{GREEDYDATA:mensaje}
```

Grok

- Pasos para utilizar Grok en Logstash
  1) Describir la estructura de cada línea de texto con patrones Grok
     - Ejemplo:

```
%{SYSLOGTIMESTAMP:tiempo} %{LOGLEVEL:nivel} %{GREEDYDATA:mensaje}
```

  2) Configurar Grok como filtro de Logstash e incluir los patrones

```
input {
        …
}
filter {
  grok {
    match => { "message" => "%{SYSLOGTIMESTAMP:tiempo} %{LOGLEVEL:nivel} %{GREEDYDATA:mensaje}" }
        }
}
```

  3) Iniciar Logstash
     - Cada nuevo objeto se parsea y envía a su destino.

Grok

- GrokDebugger
  - Herramienta online para verificar patrones Grok
  - URL: https://grokdebugger.com/
  - Ejemplo:
    - Utilizando los valores de la diapositiva anterior.

Grok

- Se pueden añadir múltiples patrones en el filtro.
  - Grok parsea cada objeto con el primer patrón que encaje.

```
{
 grok {
       match => { "message" => [
                                 '%{SYSLOGTIMESTAMP:tiempo} %{LOGLEVEL:nivel}%{GREEDYDATA:mensaje}',
                                 '%{IP:direccionIP} %{LOGLEVEL:nivel}%{GREEDYDATA:mensaje}',
                                 ]}
      }
}
```

- Si un objeto no encaja con ningún patrón descrito:
  - El objeto se envía a su destino igualmente sin parsear.
  - Se le añade un nuevo campo "tags" y una etiqueta "_grokparsefailure".

Ejercicio 2

- Parsear y almacenar en un índice Elasticsearch las siguientes líneas de log:
  - 4 líneas log aleatorias obtenidas de un servidor Apache[1].
    - Versiones simplificadas respecto a sus originales.
  - *Ver siguiente diapositiva.*

```
124.173.67.77 - - 23/07/2016 - 0400 GET http://www.059boss.com/index.php

195.182.131.107 - - 23/07/2016 - 0400 GET http://asconprofi.ru/common/proxy.php

155.94.224.168 - - 23/07/2016 - 0400 GET http://www.daqimeng.com/user/login

119.29.32.85 - - 23/07/2016 - 0400 GET http://www.tianx.top
```

[1]Adaptado de: https://askubuntu.com/questions/802222/random-urls-in-apache-access-log

Ejercicio 2

- Crear un patrón Grok que encaje con el formato de Logs mostrados en la diapositiva anterior.
- Configurar pipeline Logstash:
  - Recibir datos como conexiones HTTP al puerto 9901.
  - Utilizar el patrón Grok para parsear cada línea recibida.
  - Escribir cada línea log en el índice “logs-apache” de Elasticsearch.
- Iniciar Logstash y Elasticsearch con Docker Compose
- Enviar las líneas de Log a Logstash.
  - Utilizar curl u otro cliente REST.
- Verificar que los datos se almacenan correctamente en el índice ”logs-apache” de Elasticsearch.

Kibana

- Es el Dashboard Web de la pila ELK
- Permite analizar datos de manera interactiva

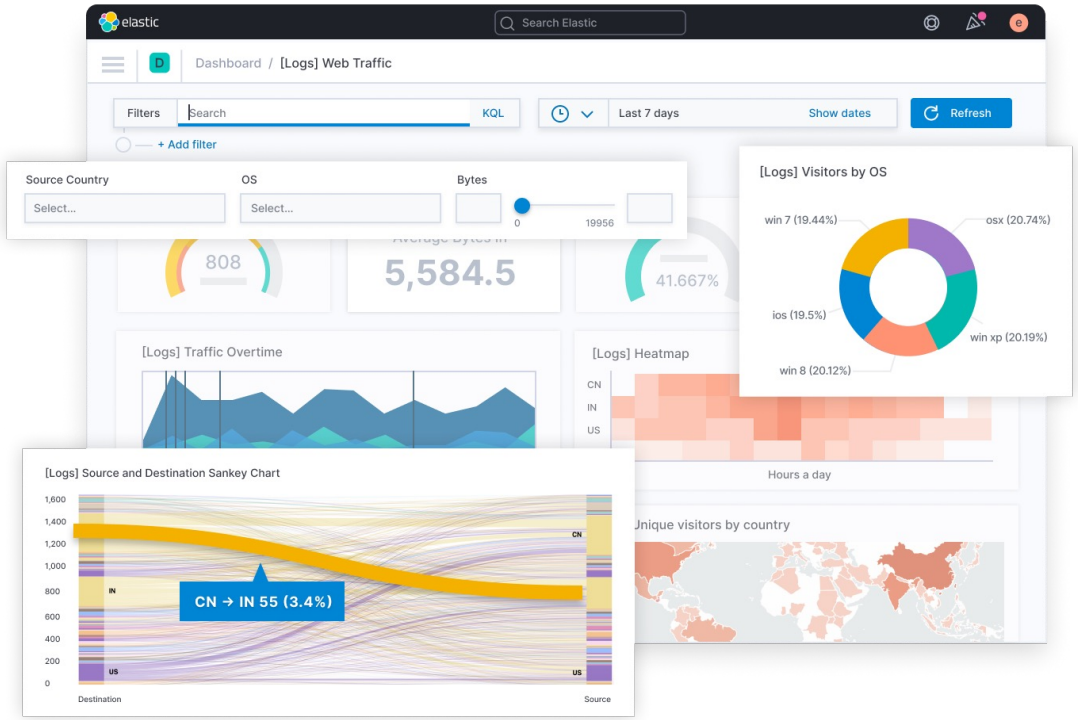
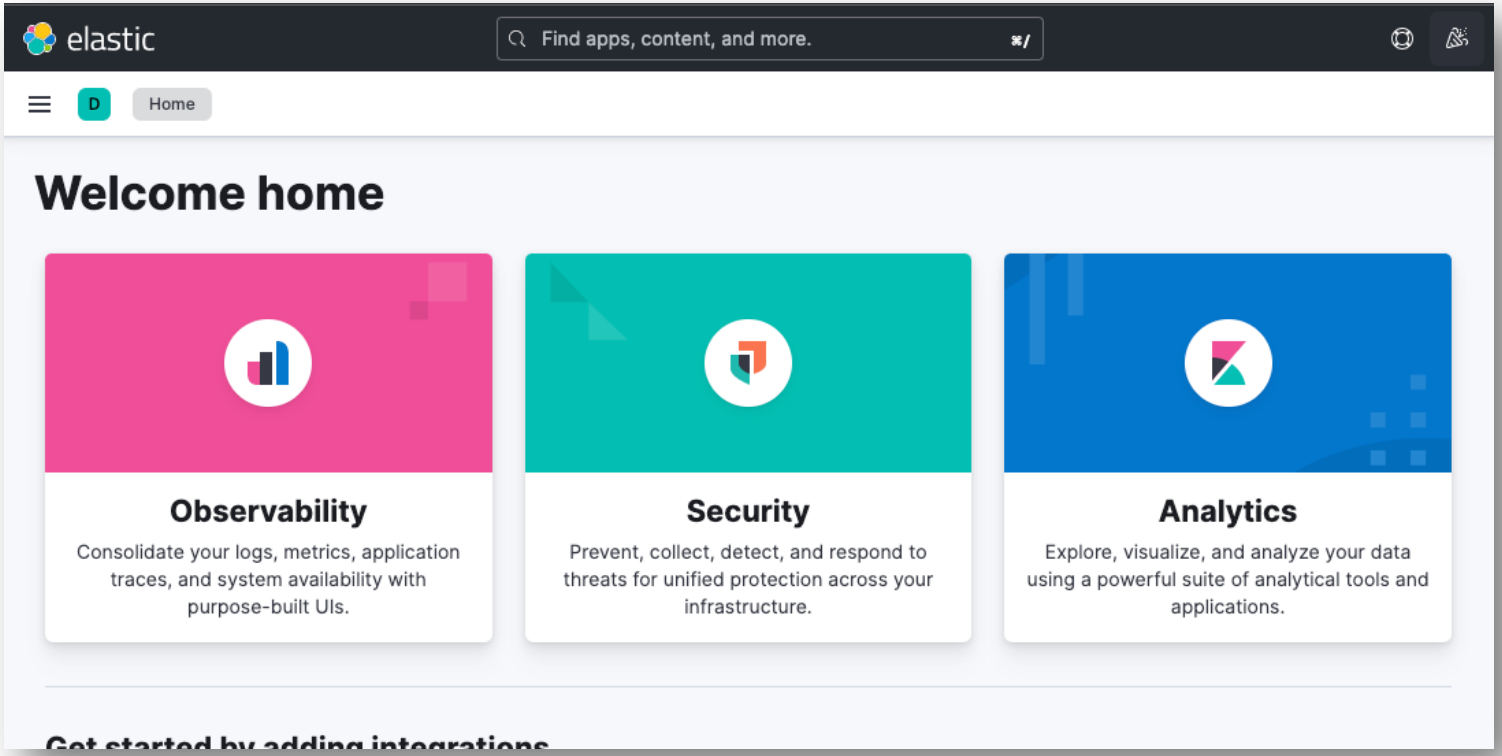


Imagen de: https://www.elastic.co/es/kibana

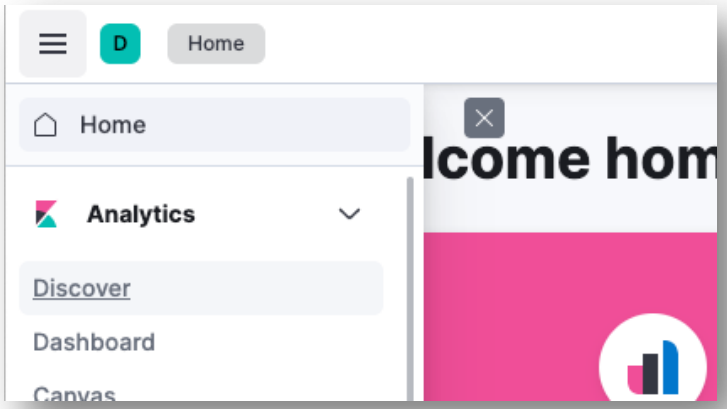
Kibana

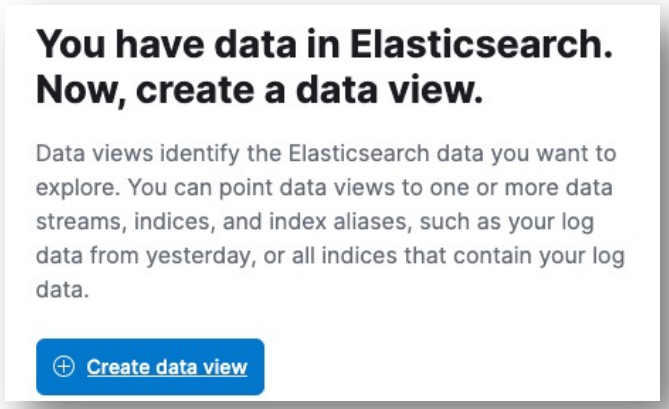
- Acceder a http://‹IP› desde un navegador
  - Donde ‹IP› es la IP/dirección de la máquina con ELK en marcha.



Kibana

- Mostrar datos de Elasticsearch
  - Es necesario crear un “Data View”
    - Objeto de Kibana que conecta con los índices Elasticsearch
  - Abrir la sección “Discover”:
    - Menu lateral, apartado “Analytics”.
    - En el aviso emergente, seleccionar “Create data view”.

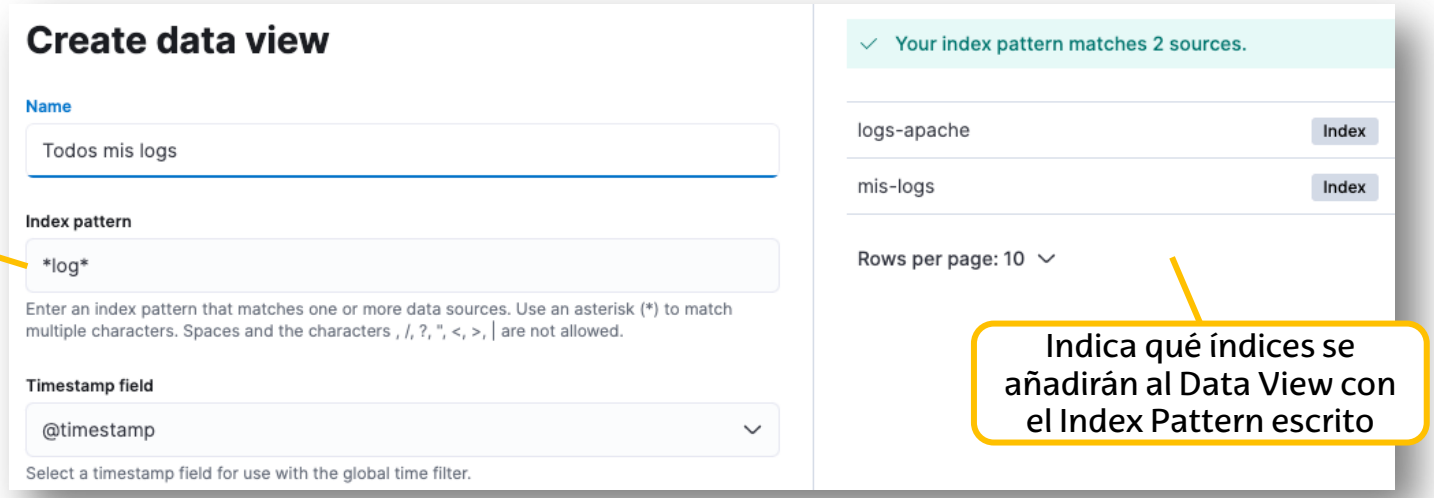




Kibana

- Mostrar datos de Elasticsearch
  - Un "Data View" recupera datos a través de un "Index Pattern"[1].
  - Un "Index Pattern" es una expresión que indica qué índices de Elasticsearch se quieren explorar.
    - Permite obtener datos de múltiples índices simultáneamente.
    - Puede ser una expresión regular simple.

P.e. **log** hace que se el Data View tenga los datos de todos los índices con "*log*" en su nombre



Indica qué índices se añadirán al Data View con el Index Pattern escrito

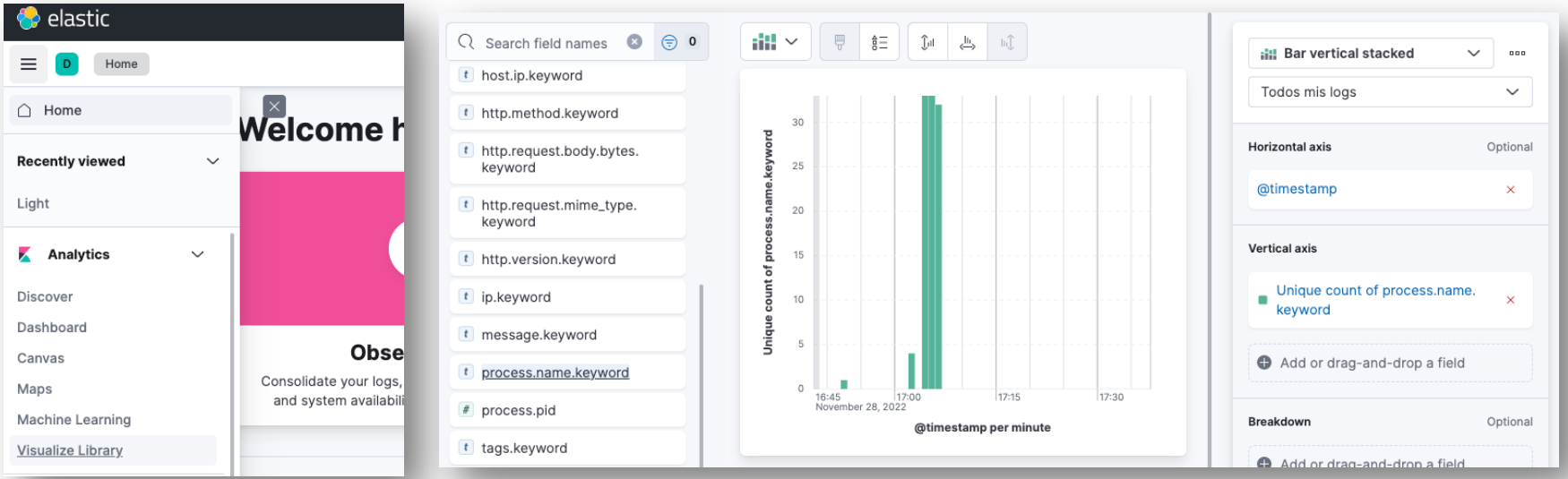
[1]Create a Data View, Kibana, Elastic.co: https://www.elastic.co/guide/en/kibana/current/data-views.html

Kibana

- Mostrar datos de Elasticsearch
  - Se muestra la vista Discover con los datos recuperados de los índices definidos con el Index Pattern

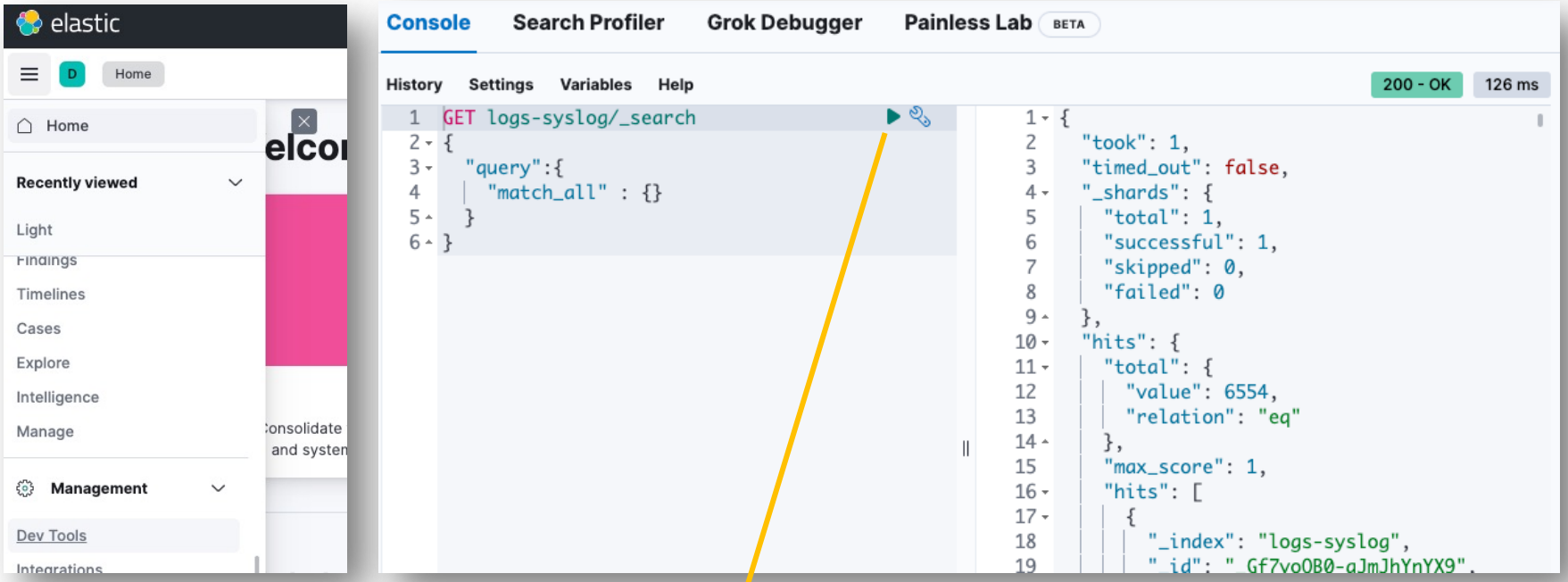
Kibana

- Crear diagramas con los datos
  - Usar la función “Visualize Library” en el apartado “Analytics”
  - El modo **Lens** permite crear gráficos al estilo “Drag and drop”.



Kibana

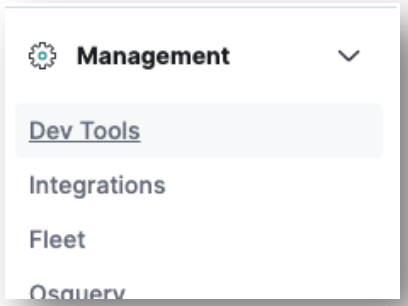
- El modo *Dev Tools* permite hacer consultas en formato REST a ElasticSearch

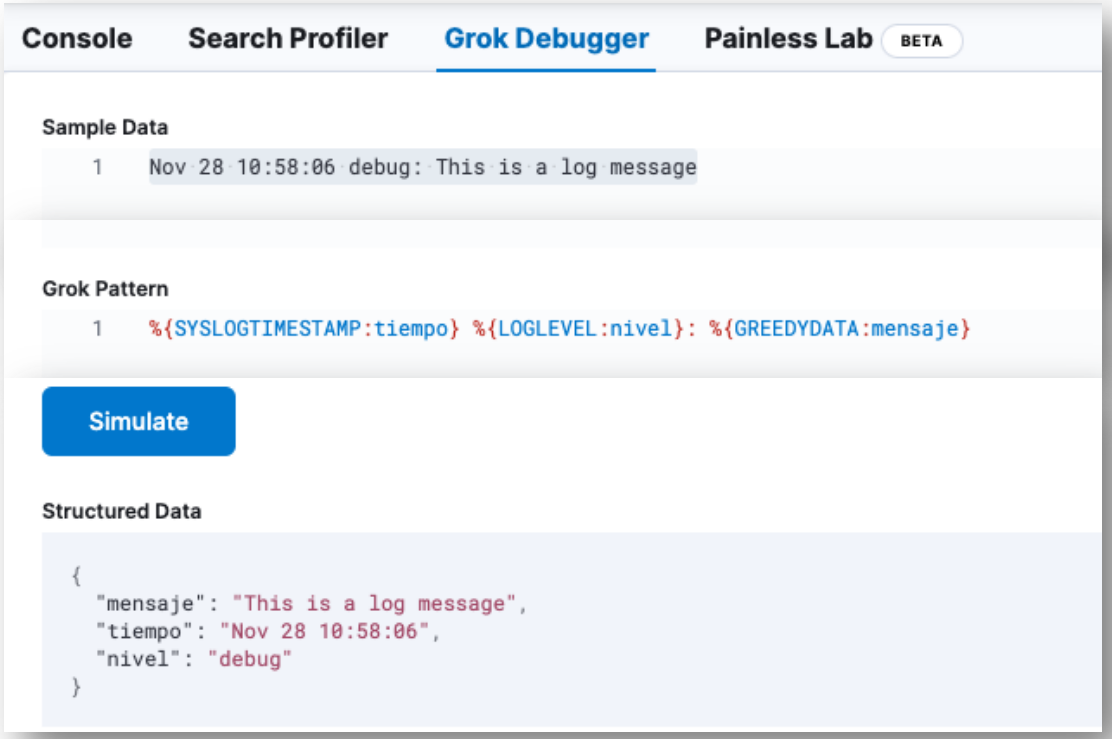


Ejecutar la consulta con el icono ▷

Kibana

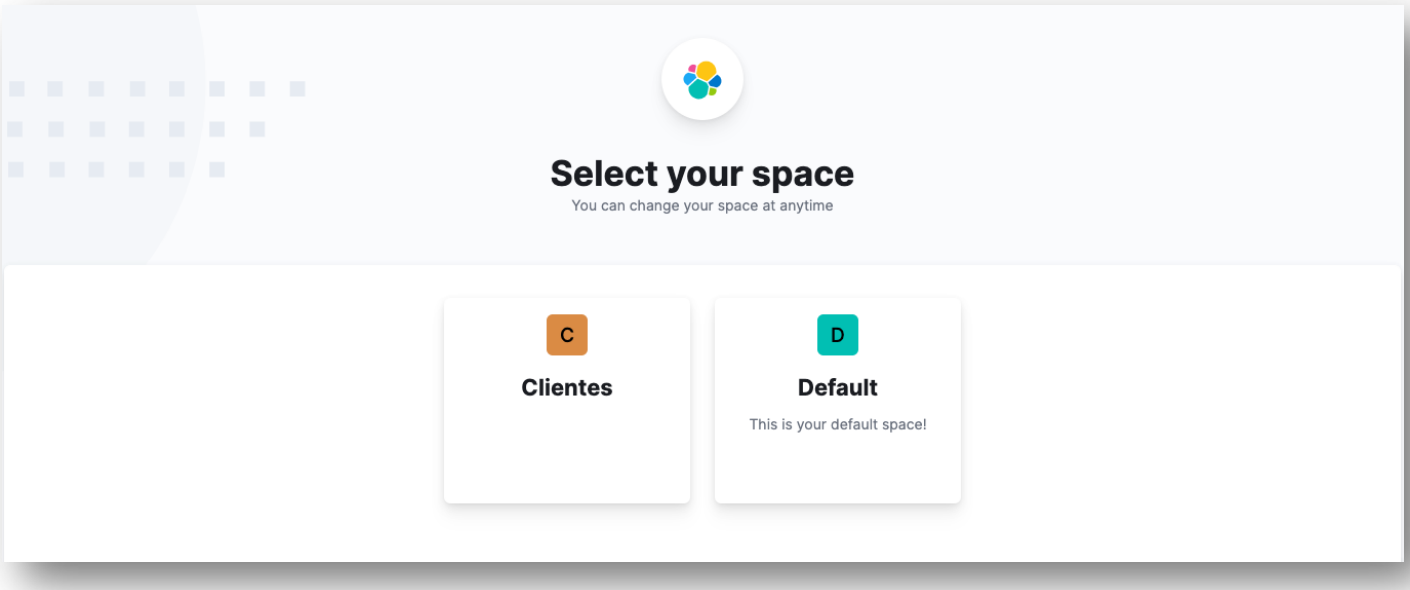
- Se incluye un Debugger de expresiones Grok.
  - Sección *Dev Tools*





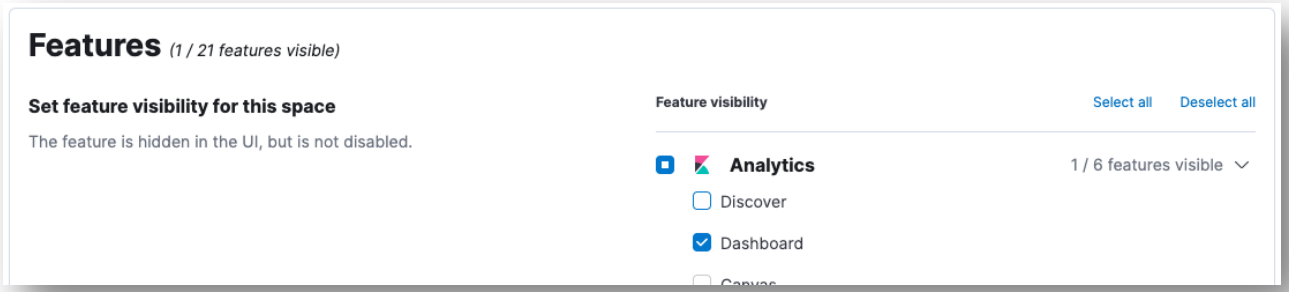
Kibana Spaces

- Permite crear espacios/Dashboards con diferentes características.
- Al crear varios Spaces, la pantalla inicial de Kibana muestra un selector:

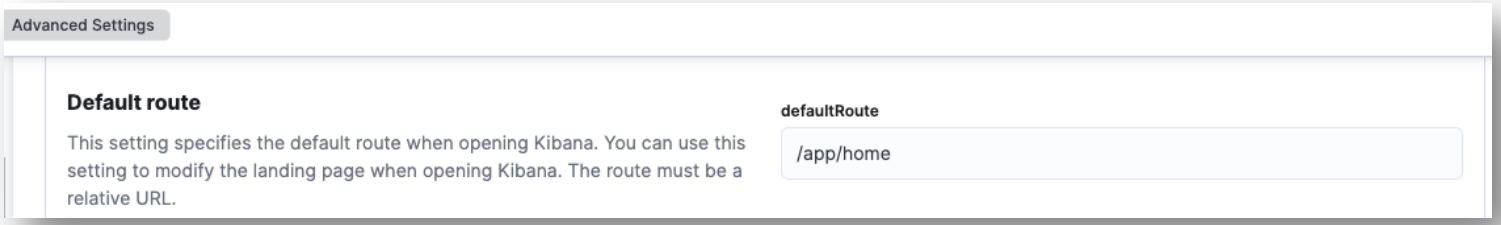


Kibana Spaces

- Se pueden limitar las características de cada Space.
  - Sección Stack Management / Spaces



- Se puede configurar Kibana para que cargue directamente un Space concreto.
  - Sección Stack Management / Advanced Settings, opción “route”.



Bibliografía

- Frank Kane. “Elasticsearch 7 and the Elastic Stack: In Depth and Hands On”, Udemy, 2020[1]:
  - https://www.udemy.com/course/elasticsearch-7-and-elastic-stack

- Logstash Docs, Elastic.co[2]:
  - https://www.elastic.co/guide/en/logstash/current

- Kibana Docs, Elastic.co[2]:
  - https://www.elastic.co/guide/en/kibana/current

- Consultados en noviembre 2020[1] y noviembre 2022[2].