



Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

IVÁN PINAR DOMÍNGUEZ ®

¿Qué es Elastic Stack (ELK)?



**Búsqueda, análisis eficiente y
almacenamiento flexible de ingente
volumen de datos**

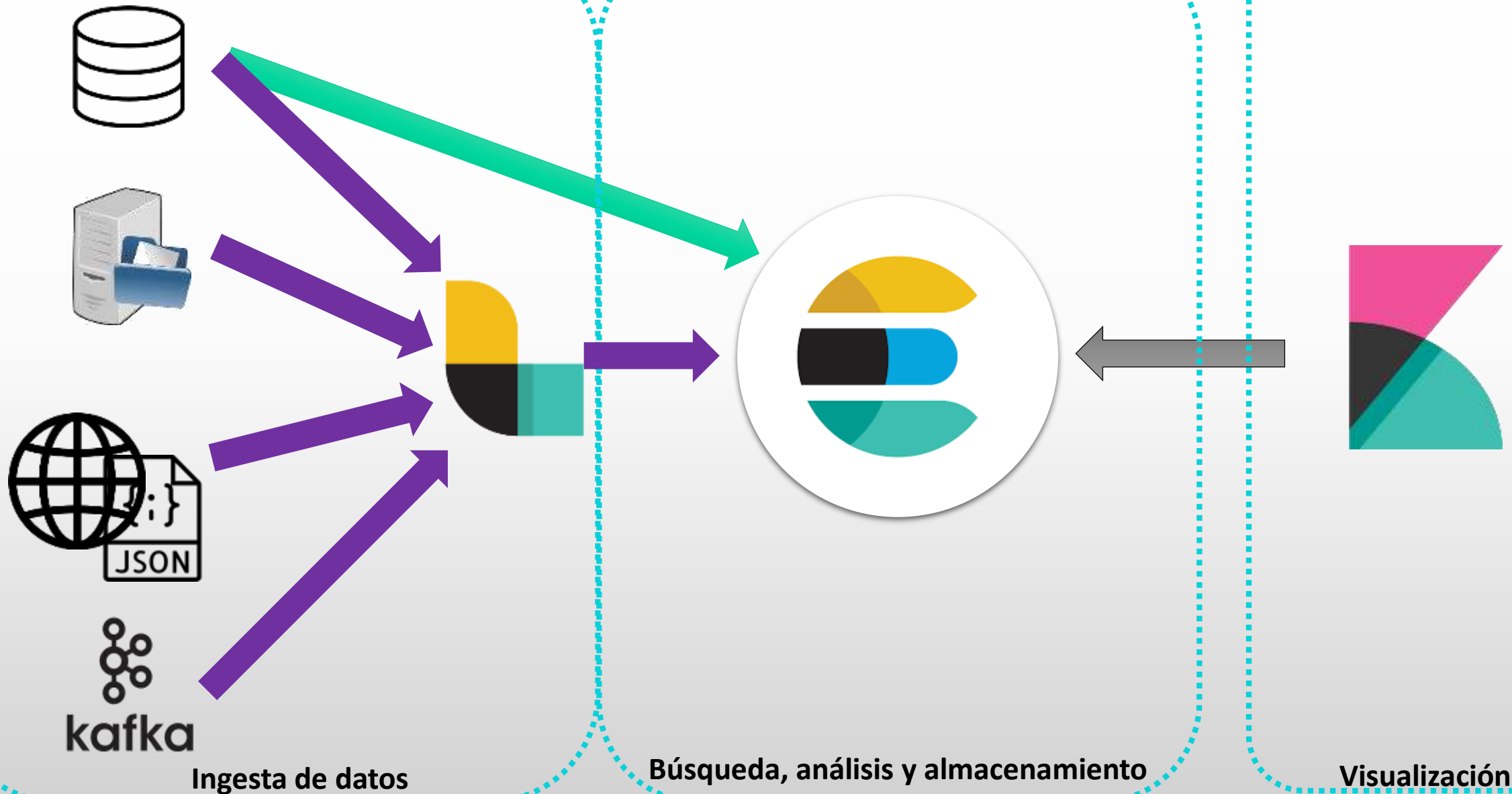


**Pipeline de procesamiento de datos
capaz de ingestar datos de cualquier
naturaleza, transformarlos y enviarlos
a la salida que necesitamos**



**Plataforma de visualización y análisis
con capacidad de construir potentes
dashboards invocando al motor
Elasticsearch**

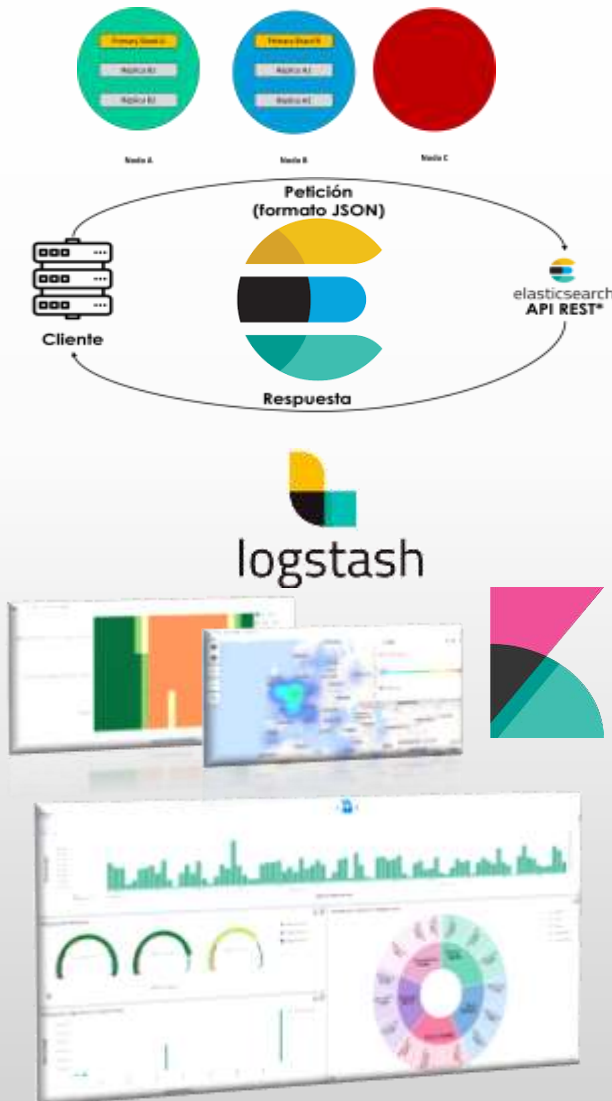
¿Qué es Elastic Stack (ELK)?



Objetivos del Curso

Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

- 01 Aprender los fundamentos y la arquitectura de Elastic Stack (ELK) para entender por qué es tan flexible, escalable y eficiente.
- 02 Crear un potente motor de búsqueda con Elasticsearch.
- 03 Escribir consultas Elasticsearch simples y complejas.
- 04 Ejecutar pipelines de ingesta, transformación y carga de datos con Logstash.
- 05 Crear potentes visualizaciones en Kibana (línea, circular, histograma, heatmap, mapa geográfico,...).
- 06 Crear dashboards interactivos en Kibana y enlazar múltiples dashboards.
- 07 Gestionar acceso y seguridad para compartir dashboards con otros usuarios configurando roles y permisos.



Estructura del Curso



elastic



Estructura del Curso

Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

01

INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK



Estructura del Curso

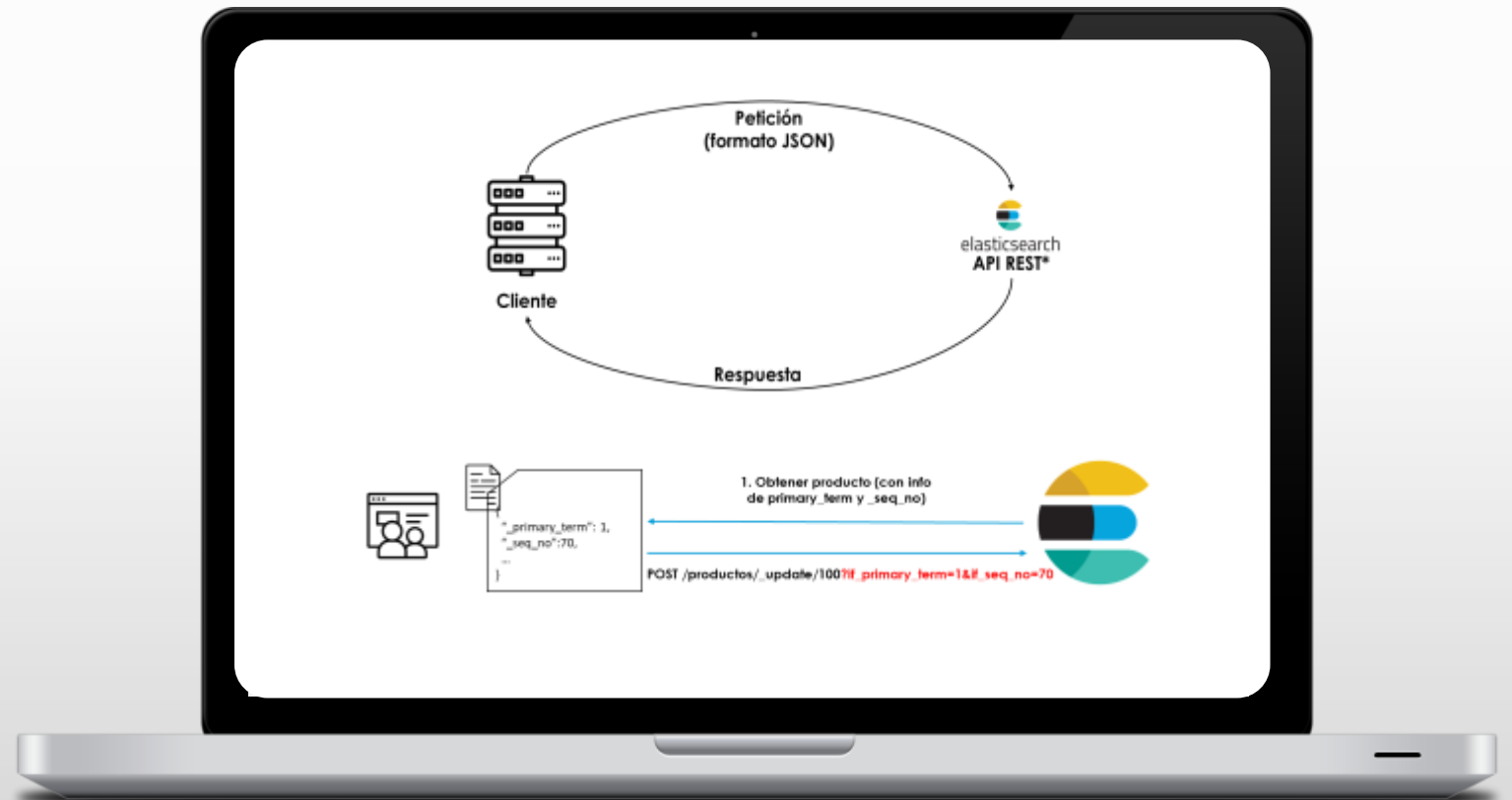
Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

01

INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK

02

ELASTICSEARCH - MANEJO DE DOCUMENTOS



Estructura del Curso

Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

01

INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK

02

ELASTICSEARCH - MANEJO DE DOCUMENTOS

03

ELASTICSEARCH - TÉCNICAS DE MAPPING Y ANÁLISIS



Estructura del Curso

Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

01

INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK

02

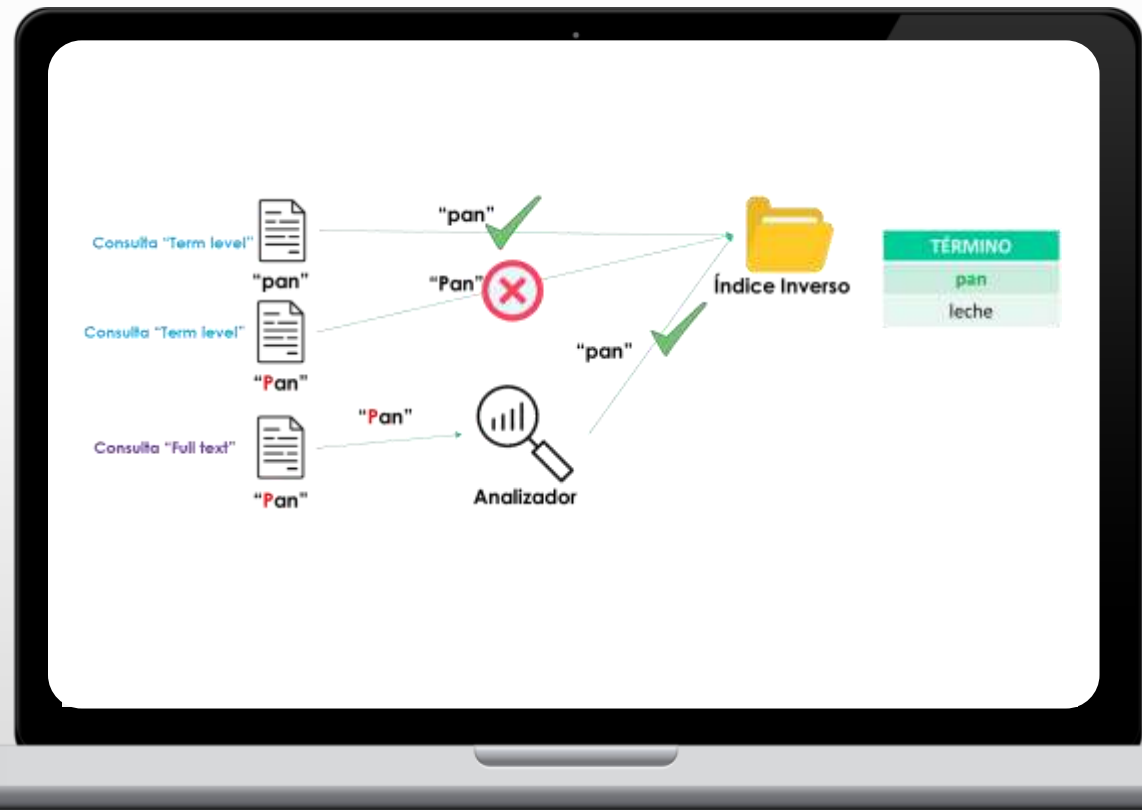
ELASTICSEARCH - MANEJO DE DOCUMENTOS

03

ELASTICSEARCH - TÉCNICAS DE MAPPING Y ANÁLISIS

04

ELASTICSEARCH - BÚSQUEDAS TERM-LEVEL, FULL-TEXT Y BOOLEANAS



Estructura del Curso

Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

01

INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK

02

ELASTICSEARCH - MANEJO DE DOCUMENTOS

03

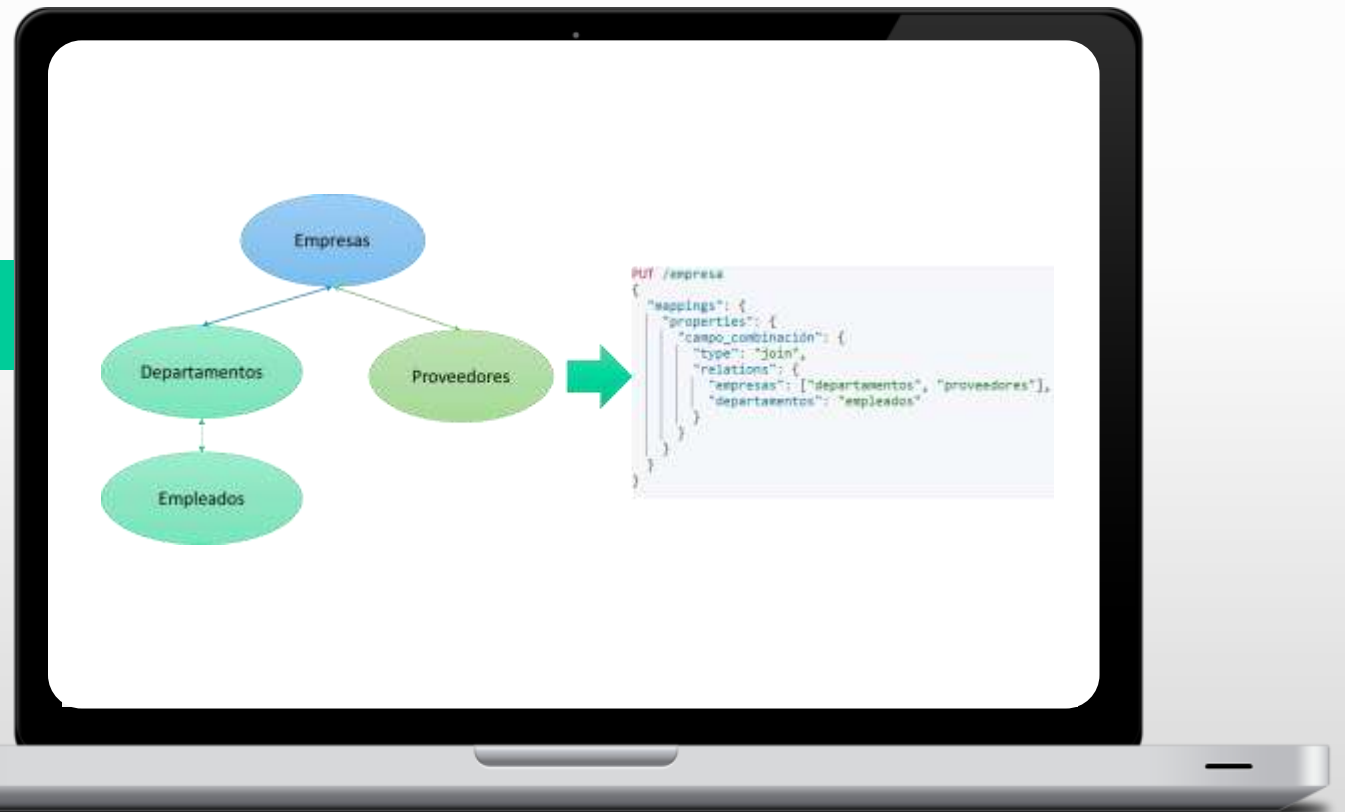
ELASTICSEARCH - TÉCNICAS DE MAPPING Y ANÁLISIS

04

ELASTICSEARCH - BÚSQUEDAS TERM-LEVEL, FULL-TEXT Y BOOLEANAS

05

ELASTICSEARCH - CONSULTAS PARA RELACIONES ENTRE DOCUMENTOS



Estructura del Curso

Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

01

INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK

02

ELASTICSEARCH - MANEJO DE DOCUMENTOS

03

ELASTICSEARCH - TÉCNICAS DE MAPPING Y ANÁLISIS

04

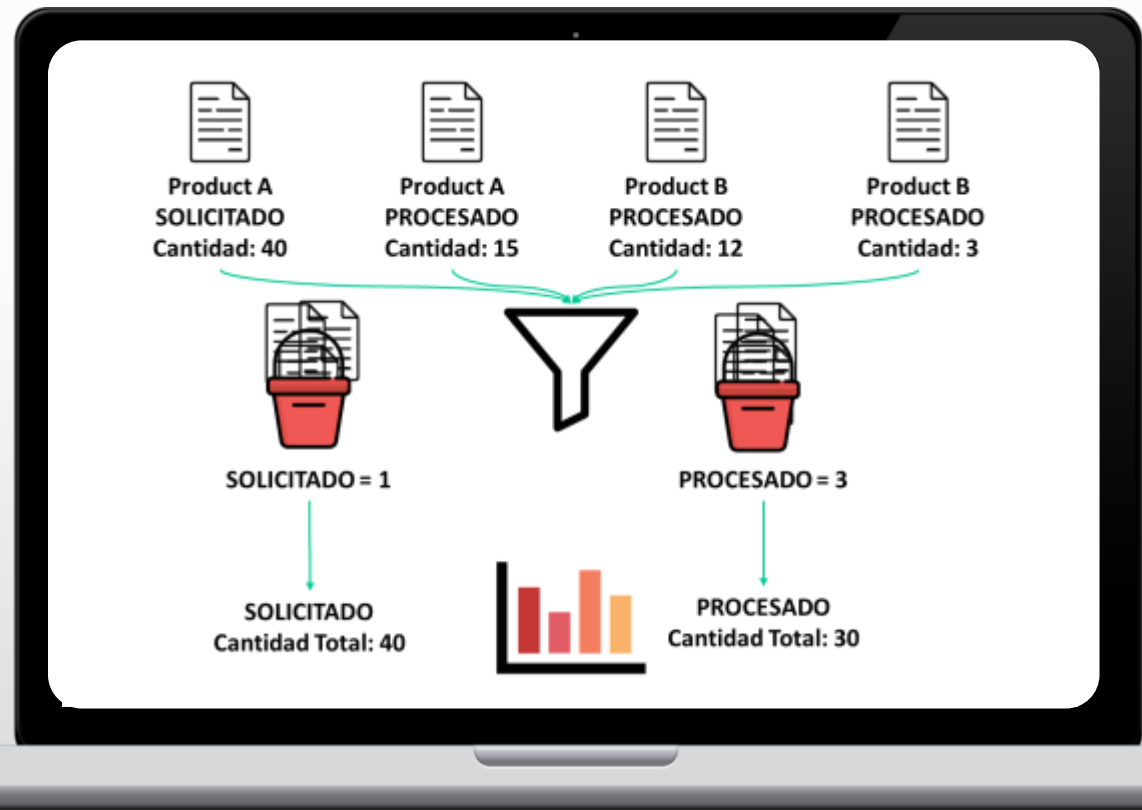
ELASTICSEARCH - BÚSQUEDAS TERM-LEVEL, FULL-TEXT Y BOOLEANAS

05

ELASTICSEARCH - CONSULTAS PARA RELACIONES ENTRE DOCUMENTOS

06

ELASTICSEARCH - AGREGACIONES



Estructura del Curso

Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

01

INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK

02

ELASTICSEARCH - MANEJO DE DOCUMENTOS

03

ELASTICSEARCH - TÉCNICAS DE MAPPING Y ANÁLISIS

04

ELASTICSEARCH - BÚSQUEDAS TERM-LEVEL, FULL-TEXT Y BOOLEANAS

05

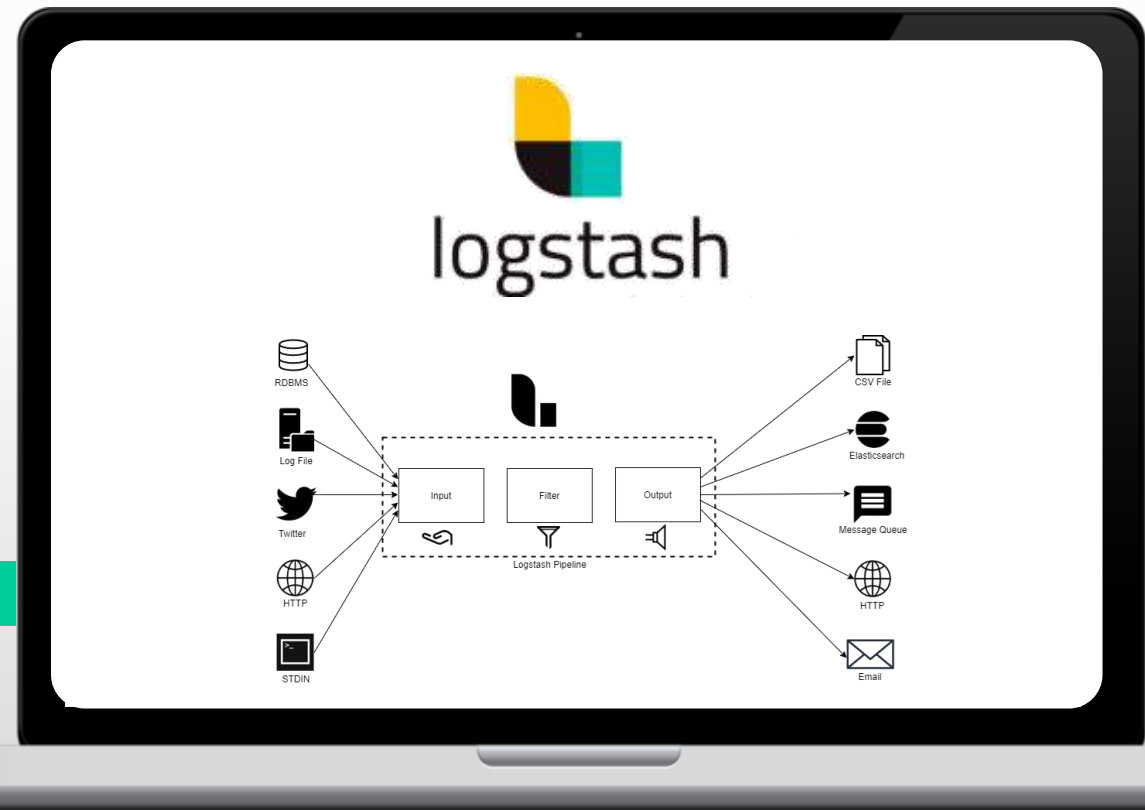
ELASTICSEARCH - CONSULTAS PARA RELACIONES ENTRE DOCUMENTOS

06

ELASTICSEARCH - AGREGACIONES

07

LOGSTASH - INGESTA, TRANSFORMACIÓN Y SALIDA



Estructura del Curso

Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

- 01 INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK
- 02 ELASTICSEARCH - MANEJO DE DOCUMENTOS
- 03 ELASTICSEARCH - TÉCNICAS DE MAPPING Y ANÁLISIS
- 04 ELASTICSEARCH - BÚSQUEDAS TERM-LEVEL, FULL-TEXT Y BOOLEANAS
- 05 ELASTICSEARCH - CONSULTAS PARA RELACIONES ENTRE DOCUMENTOS
- 06 ELASTICSEARCH - AGREGACIONES
- 07 LOGSTASH - INGESTA, TRANSFORMACIÓN Y SALIDA
- 08 KIBANA - INTERFAZ, INGESTA Y VISUALIZACIONES



Estructura del Curso

Máster Elasticsearch, Logstash y Kibana (Elastic Stack)

- 01 INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK
- 02 ELASTICSEARCH - MANEJO DE DOCUMENTOS
- 03 ELASTICSEARCH - TÉCNICAS DE MAPPING Y ANÁLISIS
- 04 ELASTICSEARCH - BÚSQUEDAS TERM-LEVEL, FULL-TEXT Y BOOLEANAS
- 05 ELASTICSEARCH - CONSULTAS PARA RELACIONES ENTRE DOCUMENTOS
- 06 ELASTICSEARCH - AGREGACIONES
- 07 LOGSTASH - INGESTA, TRANSFORMACIÓN Y SALIDA
- 08 KIBANA - INTERFAZ, INGESTA Y VISUALIZACIONES
- 09 KIBANA – CREACIÓN DE DASHBOARDS, ROLES Y PERMISOS



¿Por qué adquirir este curso?

Aprendizaje
desde cero



Material
extenso con
Casos
prácticos



Scripts y
consultas
reutilizables
a su caso de
uso



Precio
asequible



Certificación
Finalización



Garantía
100% - 30
días





Iván Pinar Domínguez

**Ingeniero de Telecomunicación
Director de Operaciones/ Data Scientist**

**12 años de experiencia con plataformas de
Análisis de Datos y Automatización de Procesos**

Master en Project Management







BLOQUE 1: INTRODUCCIÓN Y ARQUITECTURA DE ELASTIC STACK

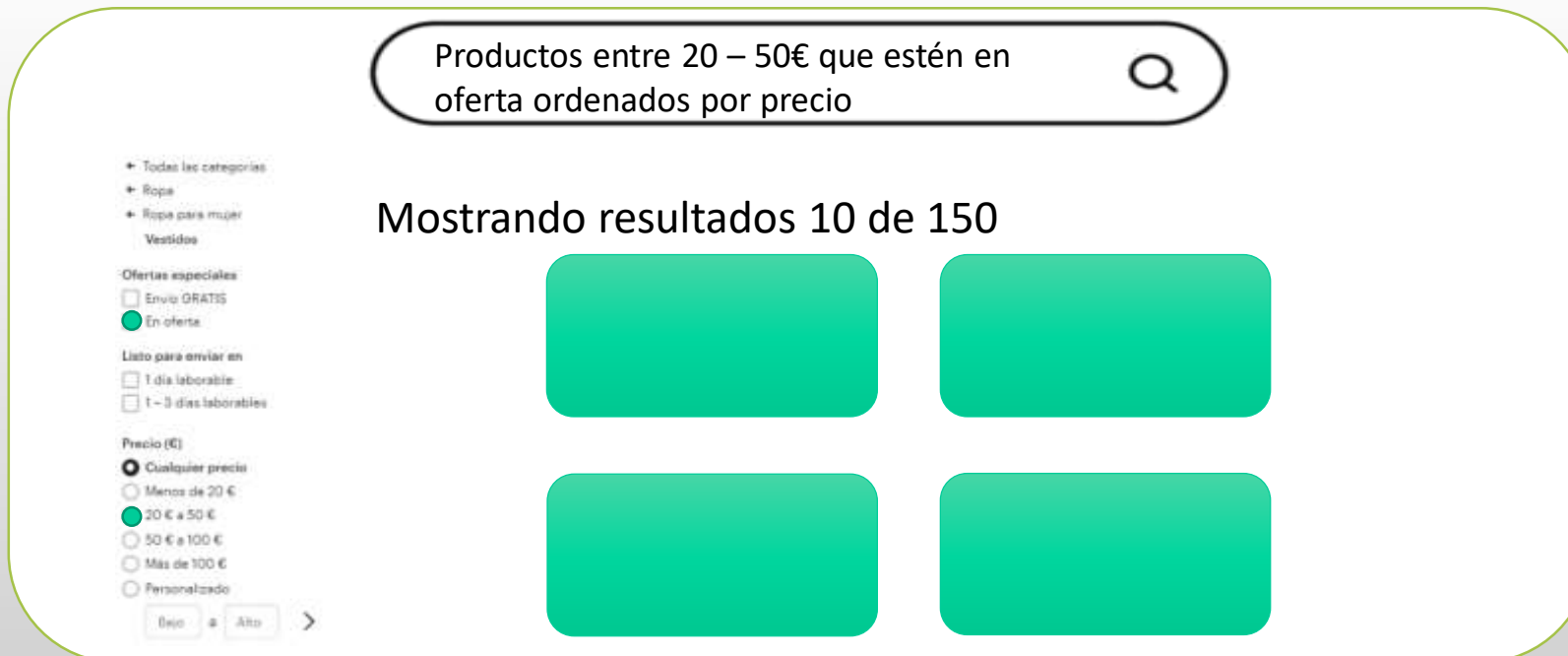
Lo que aprenderemos en este bloque....

- ✓ ¿Qué es Elastic Stack?
- ✓ ¿Cómo instalar e implementar Elastic Stack en local o cloud?
- ✓ ¿Cuál es la arquitectura básica de Elastic?
- ✓ ¿Qué es el sharding, escalabilidad, replicación y roles en Elastic?

¿Qué es Elasticsearch?

- Elasticsearch es un **motor de búsqueda** analítico open-source full-text fácil de usar y **altamente escalable**.
- Es posible realizar búsquedas complejas de datos sobre toda la información **indexada** que tengamos incluyendo:
 - *Auto-complección*
 - *Corrección de errores*
 - *Resaltar emparejamientos*
 - *Manejar sinónimos*
 - *Ajuste de relevancia*

- **Ejemplo:**



Productos entre 20 – 50€ que estén en oferta ordenados por precio

Mostrando resultados 10 de 150

Filters:

- + Todas las categorías
- + Ropa
- + Ropa para mujer
- Vestidos
- Ofertas especiales
 - ☐ Envío GRATIS
 - ☒ En oferta
- Listo para enviar en
 - ☐ 1 día laborable
 - ☐ 1 - 3 días laborables
- Precio (€)
 - ☒ Cualquier precio
 - ☐ Menos de 20 €
 - ☒ 20 € a 50 €
 - ☐ 50 € a 100 €
 - ☐ Más de 100 €
 - ☐ Personalizado

Bejo * Alto >

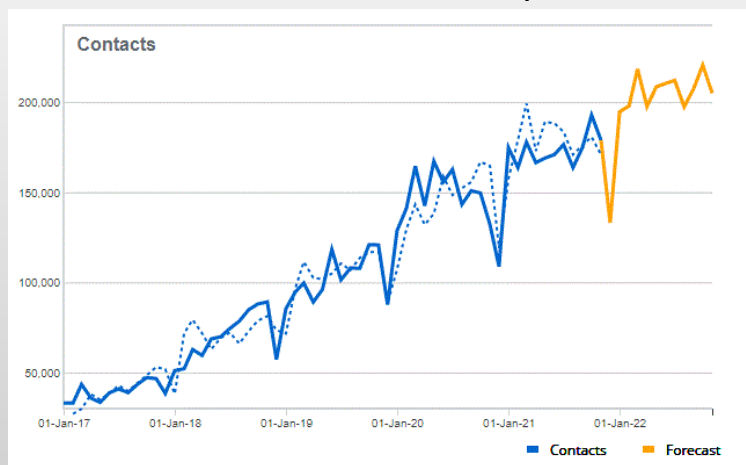
The main content area displays four large teal rectangular placeholders arranged in a 2x2 grid, representing the search results.

¿Qué es Elasticsearch?

- Elasticsearch también permite **consultas sobre datos estructurados** como números o agregaciones para ser usado como plataforma analítica.



- Elasticsearch es una herramienta potente a la hora de manejar **grandes volúmenes de datos** (uso de índices, shards,...).
- Ejemplo:** Predicción de llamadas en un Call Center para dimensionar el equipo



¿Qué es Elasticsearch?

- En Elasticsearch los datos son guardados como **documentos** (unidad de información, similar a una fila en una base de datos SQL).
- Un documento es un objeto JSON que contiene **campos** (similar a las columnas en una base de datos).

```
{  
  "nombre": "Iván",  
  "apellido": "Pinar",  
  "intereses": ["Elastic Stack", "Elasticsearch", "análisis"]  
}
```



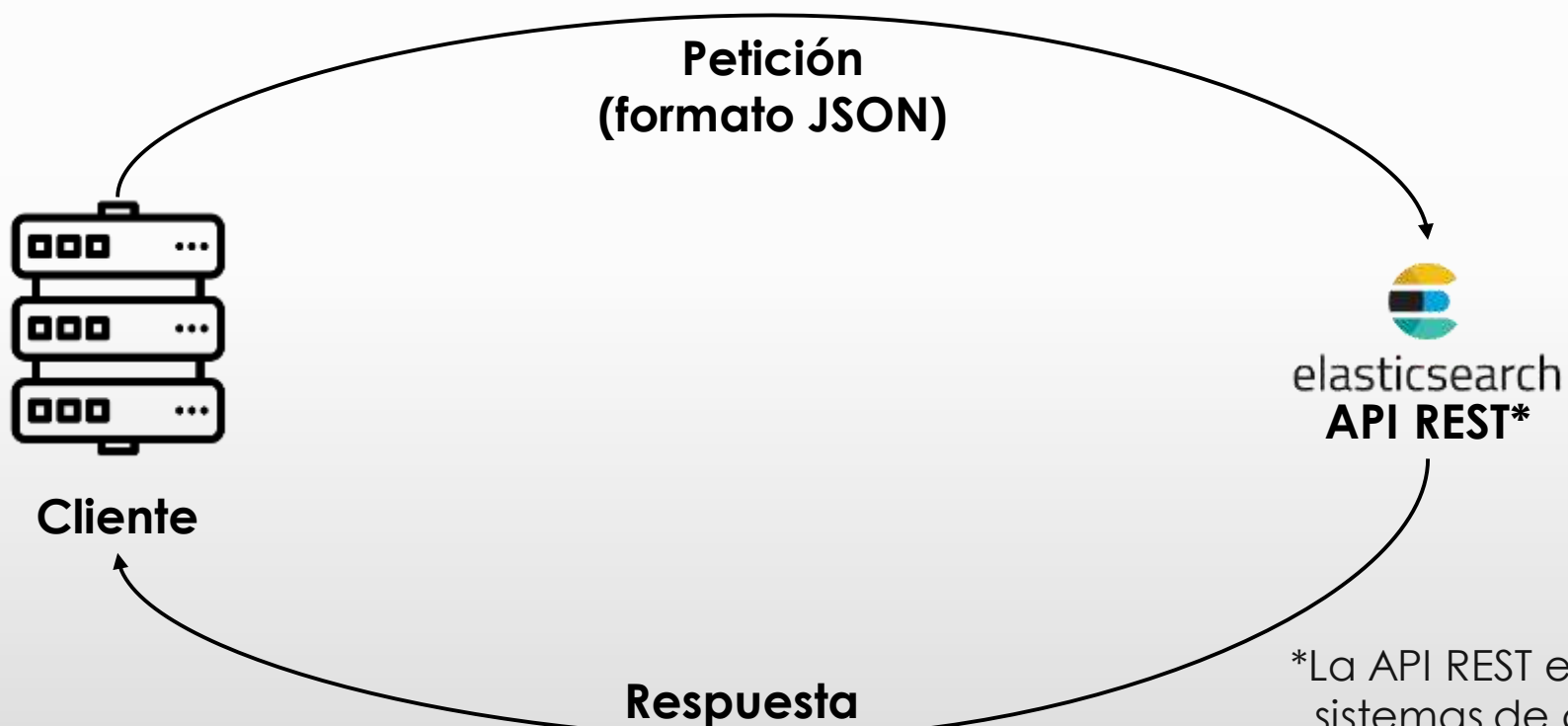
**Nombres de
campos**



Valores de campos

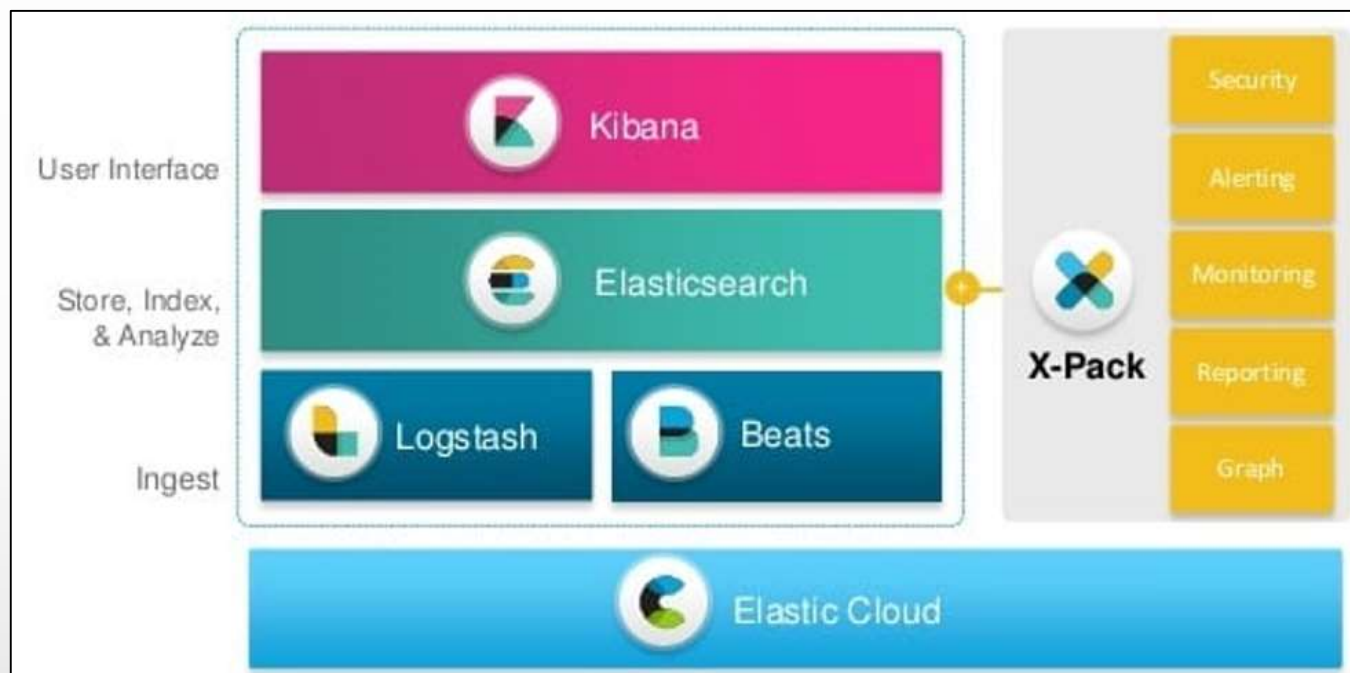
¿Qué es Elasticsearch?

¿Cómo hacemos consultas en Elasticsearch?



*La API REST es una interfaz que dos sistemas de computación utilizan para intercambiar información (obtener datos o ejecutar una función), de manera que el sistema comprenda la solicitud y la cumpla.

Resumen de Elastic Stack (Elasticsearch, Logstash, Kibana,...)

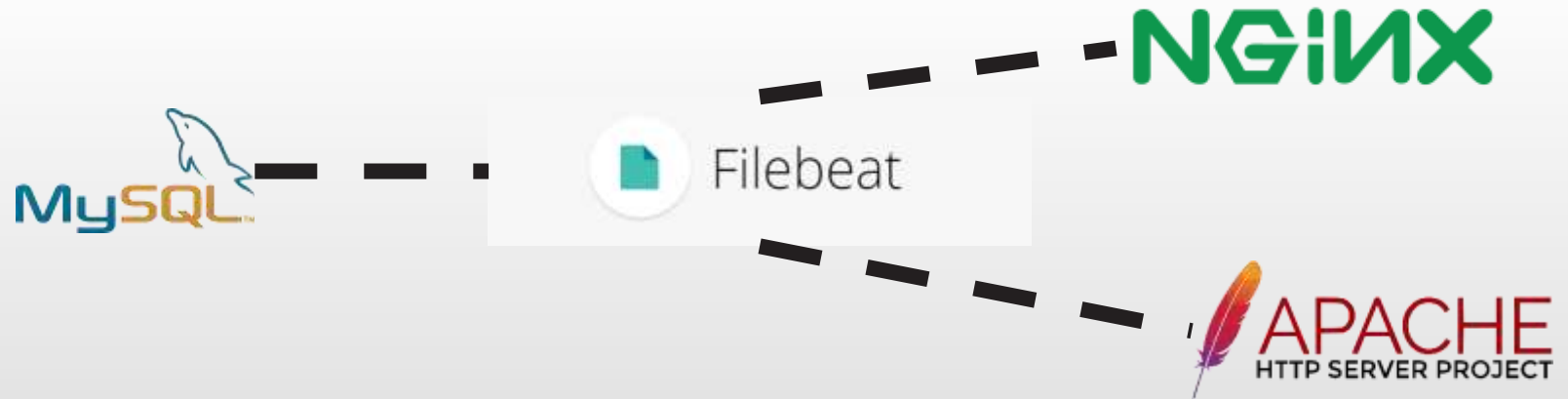


ELASTIC STACK (ELK)

Conjunto de herramientas utilizadas en el ecosistema Elastic

Resumen de Elastic Stack (Elasticsearch, Logstash, Kibana,...)

- Colección de “*data shippers*” (recolectores de datos).
- Agentes que se instalan en un **servidor** y envían datos a Logstash o Elasticsearch.
- **Filebeat** es el más usado para **recolectar logs** de diferentes tipos de servidor (nginx, apache web server, BBDD MySQL,...)

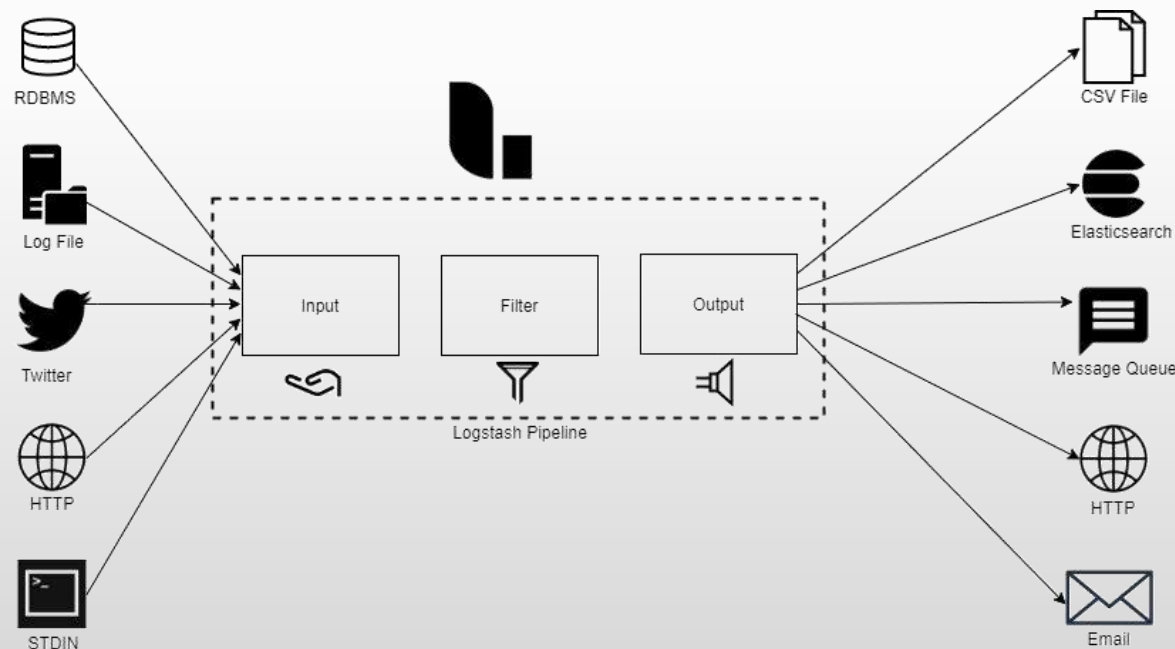


- También existen otros como **Metricbeat** para **recolectar métricas del sistema** (ejemplo uso de CPU o RAM).

Resumen de Elastic Stack (Elasticsearch, Logstash, Kibana,...)



- Logstash es un pipeline de procesamiento de datos.
- Recibe datos que son manejados como **eventos**, estos datos pueden ser de cualquier tipo de fuente.
- Tiene 3 fases: Input – Filter – Output que forman el **pipeline**:



- Existen **plugins** para cada una de las 3 fases.

Resumen de Elastic Stack (Elasticsearch, Logstash, Kibana,...)



Ejemplo



beats

10.0.0.27 – ivan [20/04/2023] "GET /productos/view/123" 200



10.0.0.27 – ivan [20/04/2023] GET /productos/view/123 200



```
{
  "ip_adress": "10.0.0.27",
  "user": "ivan",
  "http_verb": "GET",
  "request_path": "/productos/view/123",
  "http_status": 200
}
```

Resumen de Elastic Stack (Elasticsearch, Logstash, Kibana,...)

- Plataforma de visualización y análisis con capacidad de construir potentes **dashboards**:



- Configuración de aplicaciones **Machine Learning**:
 - Previsiones (forecasting)
 - Detección de anomalías
 - ...
- Puede manejar componentes de Logstash o Elasticsearch (como la autenticación)
- **Invoca a Elasticsearch** para obtener resultados a través de consultas en una interfaz gráfica.

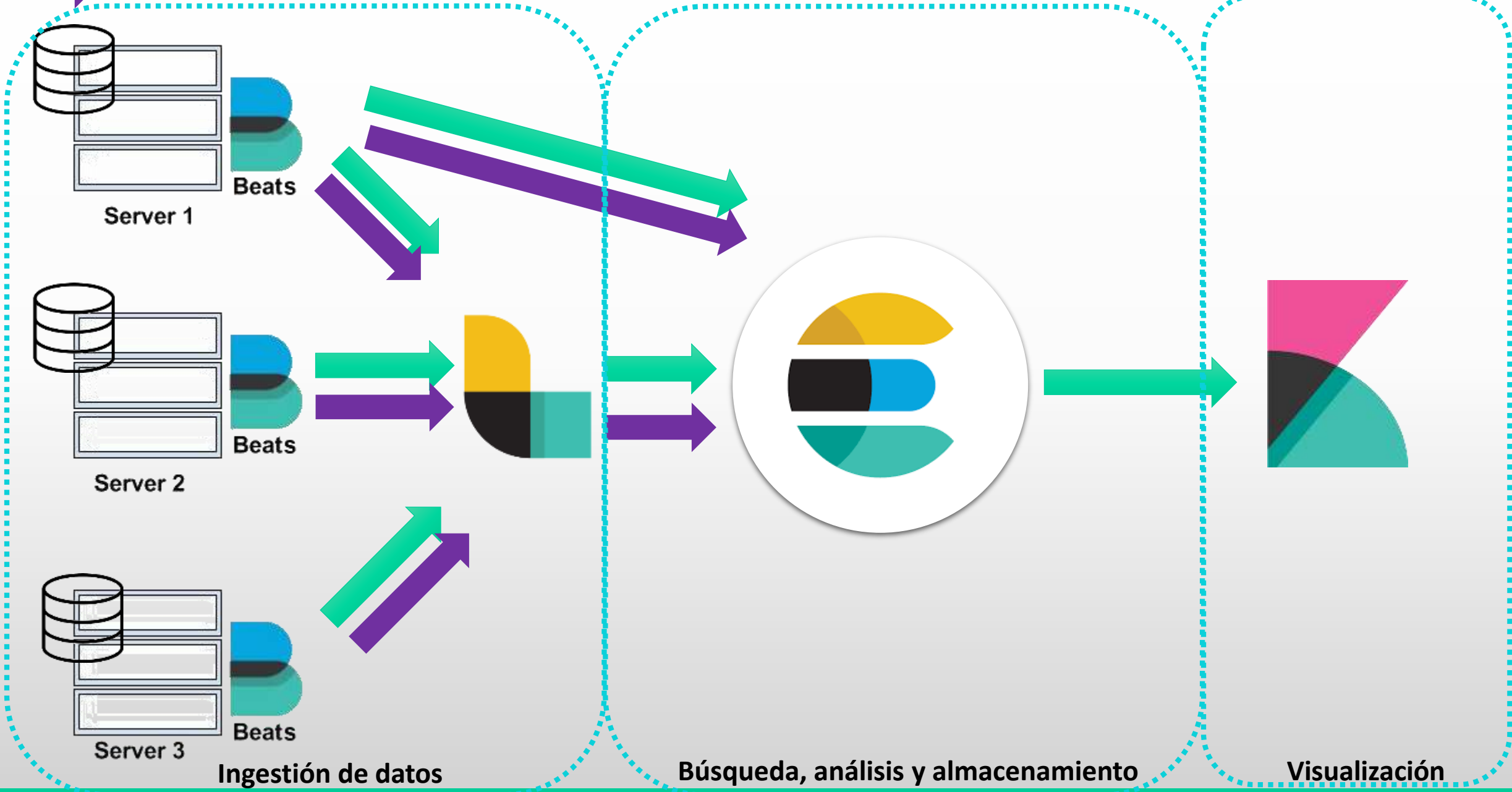
Resumen de Elastic Stack (Elasticsearch, Logstash, Kibana,...)



- Conjunto de **funcionalidades adicionales** a Elasticsearch y Kibana.
- **Seguridad:**
 - Proporciona autenticación y autorización para Elasticsearch y Kibana.
 - Se puede integrar con proveedores de autenticación
 - Puede configurar roles y permisos a usuarios
- **Monitoring & Alerting:**
 - Verifica cómo funcionan los componentes y su rendimiento
 - Se puede configurar para enviar alertas (ejemplo: "Notifícame si el servidor web tiene un uso de CPU mayor que 90%")
- **Reporting:**
 - Exportado de datos y visualizaciones de Kibana
 - Programación de reportes periódicos
 - Pueden ser generados en función de un disparador (como una alerta)
- **Machine Learning:** Habilita el uso de ML en Elasticsearch y Kibana.
- **Graphs:** Analiza las relaciones en los datos basándose en la relevancia con Elasticsearch (ejemplo, mostrar al usuario productos relacionados o canciones recomendadas en una lista de reproducción).



Escenario típico de Elastic Stack



Implementar Elasticsearch y Kibana en Elastic Cloud



1. Visitar la página: <https://www.elastic.co/> y pulsar en “Try Free”



2. Insertar datos personales y configurar los parámetros para el nuevo despliegue.
3. Pulsar en el proveedor de infraestructura Cloud oportuno (ejemplo Azure), no es necesario crear ninguna cuenta previa en el proveedor de Cloud.
4. Seleccionar versión de Elastic (8.x)
5. Guardar credenciales de acceso a Elastic proporcionadas

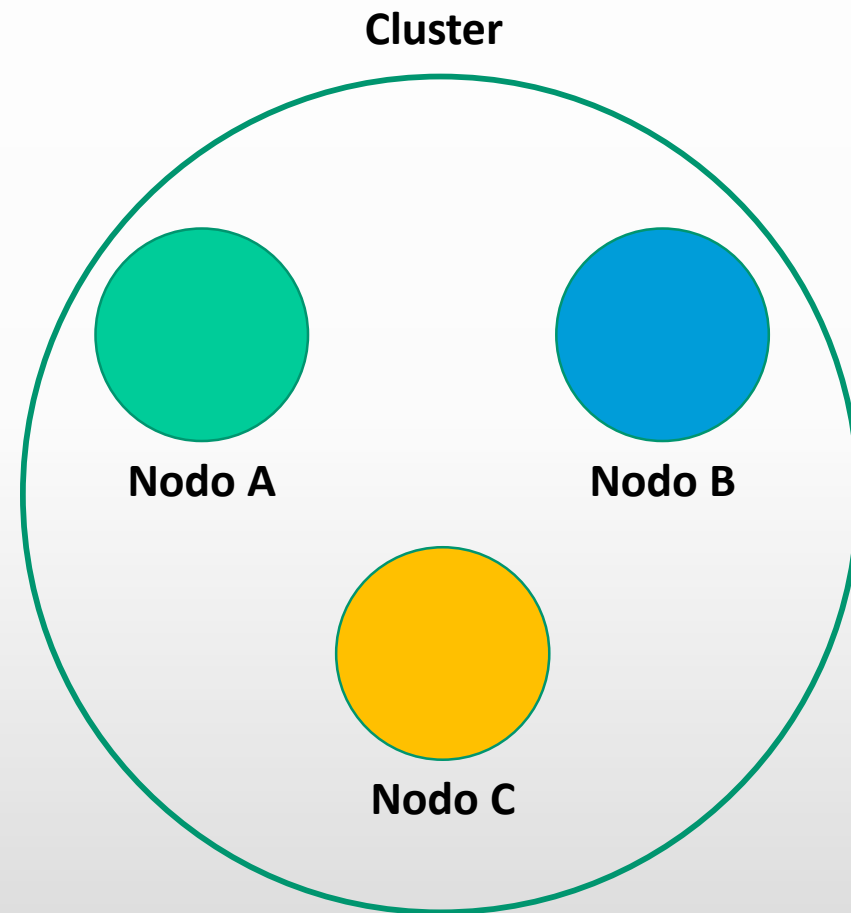
Instalación de Elasticsearch y Kibana en local

1. Visitar la página: <https://www.elastic.co/downloads/elasticsearch>
2. Seleccionar la plataforma de descarga (Windows / MAC / Linux)
3. En consola de comandos ir a la ruta donde tenemos la descarga de elasticsearch (con comando cd en Windows) e insertar la sentencia "bin\elasticsearch.bat"
4. Copiar el password generado para el usuario "elastic" que aparece en la terminal.
5. Si se necesita resetear el password, se puede usar el comando: "bin\elasticsearch-reset-password.bat -u elastic."
6. Copiar el token generado para el "enrollment" posterior de Kibana.
7. Realizamos los mismos pasos para instalar Kibana desde la URL <https://www.elastic.co/downloads/kibana> teniendo en cuenta que ahora el comando para la instalación es "bin\kibana.bat"
8. Acceder a la URL local generada e insertar el token generado anteriormente para el enrollment.
9. Acceder con el usuario "elastic" que se generó.

NOTA: En Linux/MAC la diferencia es que en lugar de insertar "bin\elasticsearch.bat" y "bin\kibana.bat", solo es necesario insertar "bin\elasticsearch" y "bin\kibana" en la terminal.

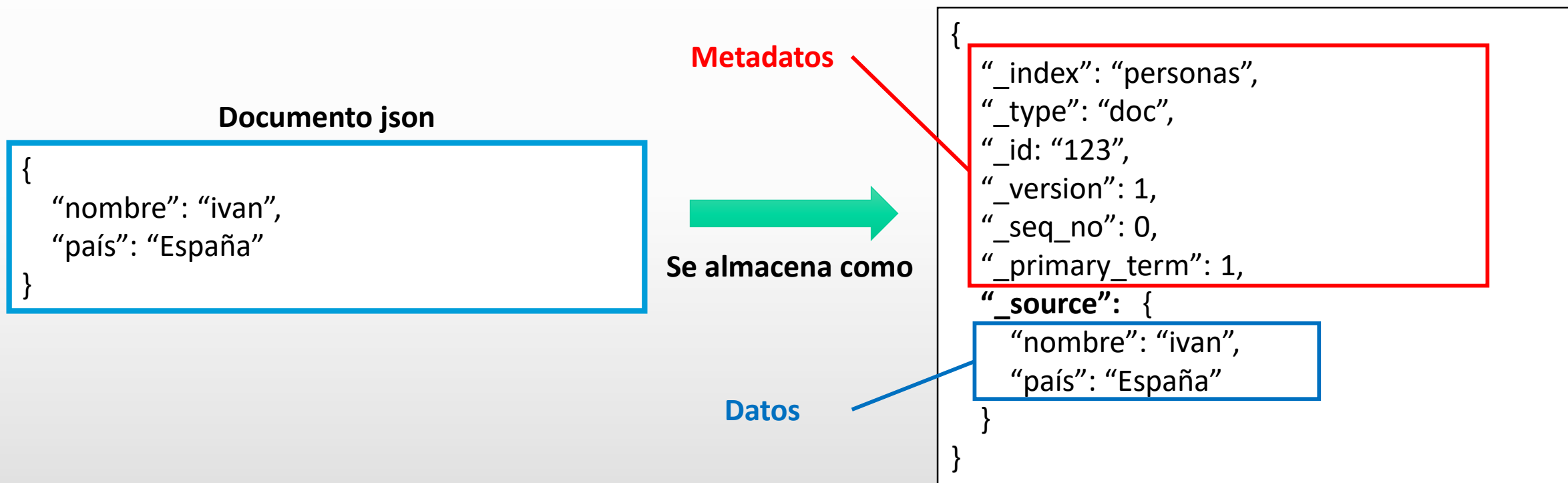
¿Cuál es la arquitectura básica de Elastic?

- **Nodo:** Instancia de Elasticsearch que almacena datos.
 - Se pueden crear todos los nodos que queramos, cada nodo tiene un límite del orden de TB.
 - Cada nodo se puede instalar en una máquina física o máquina virtual (incluso compartiendo el mismo disco)
 - Nodo = instancia (nodo no es cada máquina), se pueden instalar **hasta 5 nodos en la misma máquina** (si se quieren más habría que usar máquinas virtuales o contenedores).
 - Lo normal es que cada nodo se ejecute en una máquina diferente (o máquina virtual / contenedor diferente)
- **Cluster:** Colección de nodos relacionados.
 - Los clusters son **independientes** de los demás.
 - Se suelen utilizar diferentes clusters para separaciones lógicas (cluster para un ecommerce, cluster para el ERP,...)
 - Cuando se crea un primer nodo, **el cluster se crea automáticamente**, posteriores nodos pueden crear nuevos clusters o adherirse a uno existente.



¿Cuál es la arquitectura básica de Elastic?

¿Cómo se almacena la información en Elasticsearch?



¿Cuál es la arquitectura básica de Elastic?

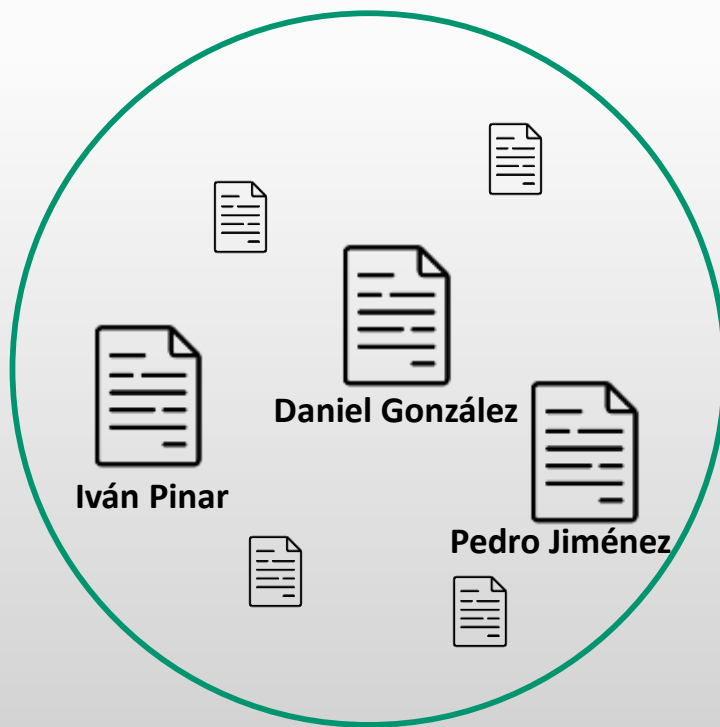
¿Cómo se organizan los documentos?



Índices

- **Índice:** Agrupan documentos de manera lógica:
 - Cada documento se almacena dentro de un index
 - Proporcionan opciones de configuración particulares para cada índice.
 - Son una colección de documentos con características similares.
 - Pueden contener todos los documentos que queramos.
 - En las consultas Elastic que se hagan hay que especificar el index al que corresponde.

Index Personas



Index Departamentos



Inspección del cluster y envío de consultas mediante consola

- La comunicación con Elastic la realizaremos a través de la **API** con mensajes **HTTP** (GET, PUT,...)
- Para comunicarnos con la API de Elasticsearch y realizar consultas (queries) podemos usar:

- Consola de Kibana (la más sencilla y recomendada)
- cURL
- Postman



- Para obtener información, usaremos el comando HTTP **GET**, ejemplo:

• **GET** **/_cluster/health**

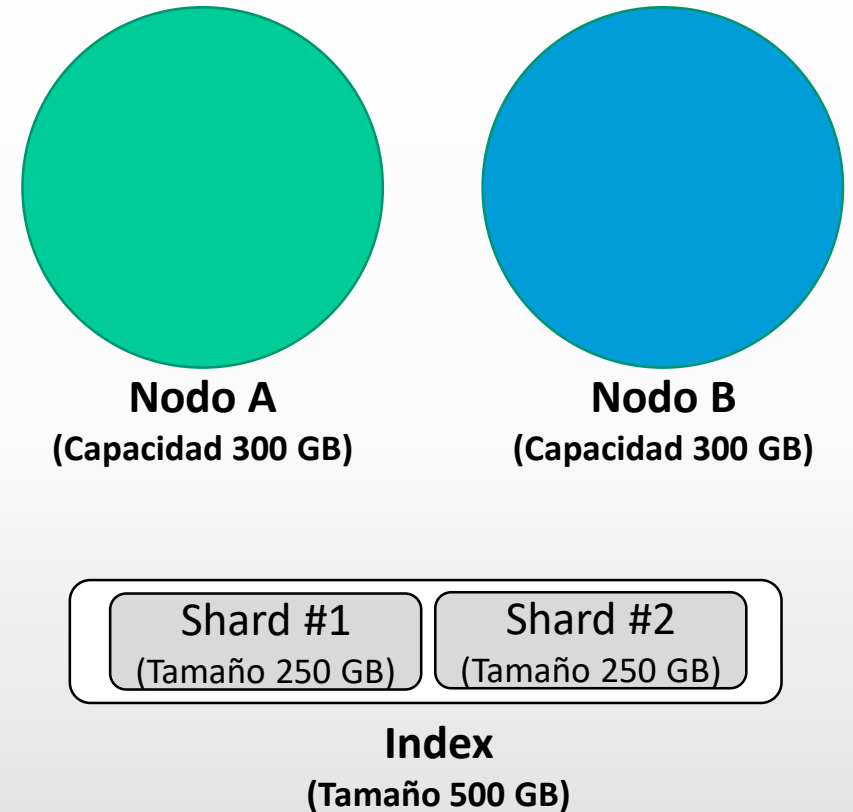
API

Comando

- Consulta de nodos: GET `/_cat/nodes?v`
- Consulta de índices: GET `/_cat/indices?v`

Sharding y escalabilidad en Elastic Stack

- **Sharding:** Técnica para dividir índices en piezas más pequeñas:
 - Cada pieza se denomina “shard”
 - Sharding se realiza a nivel de índice (no a nivel de cluster o nodo). Por defecto un índice solo contiene 1 shard.
 - **Objetivo:** Escalar horizontalmente el volumen de datos para poder alojar índices de gran tamaño.
 - Un shard solo puede alojarse en un nodo (es la división mínima)
 - Cada shard es un índice casi **independiente** (realmente es un Apache Lucene index)
 - Un índice de Elasticsearch por tanto consiste en 1 o más índices de Apache Lucene.
 - Los shards no tienen un tamaño predefinido y aumentarán conforme se añadan documentos.
 - Cada shard puede almacenar hasta 2 billones de documentos.
 - **Objetivo adicional:** *Mejorar el rendimiento*
 - **Paralelización** de consultas para incrementar el rendimiento del index (la consulta se puede realizar en múltiples shards al mismo tiempo y por tanto utilizar la capacidad de varios nodos en paralelo)
 - Incrementar el número de shards: **Split API**
 - Reducir el número de shards: **Shrink API**



RECOMENDACIÓN NÚMERO DE SHARDS: Depende muchos factores (# nodos, capacidad/nodo, # índices, # consultas,...)

Si hay millones de documentos considerar añadir 2 shards

Replicación en Elastic Stack

¿Qué ocurre si en un nodo falla el disco duro?

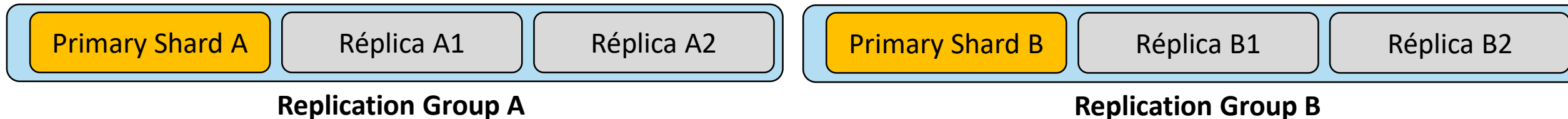


¡Perdemos los datos asociados!

SOLUCIÓN: Elasticsearch soporta Replicación (habilitada por defecto)

- A diferencia de las BBDD donde la replicación es muy compleja, en **Elasticsearch es muy sencilla** (se realiza automáticamente).
- La replicación se realiza a nivel de **índice**.
- La replicación crea copias de los shards ("**replicas**").
- Un shard que ha sido replicado se le denomina "**primary shard**".
- Un "primary shard" y sus réplicas forman el "**replication group**".
- Una réplica puede servir consultas exactamente como el "primary shard".
- Al crear un índice, podemos definir cuántas réplicas queremos (1 por defecto).

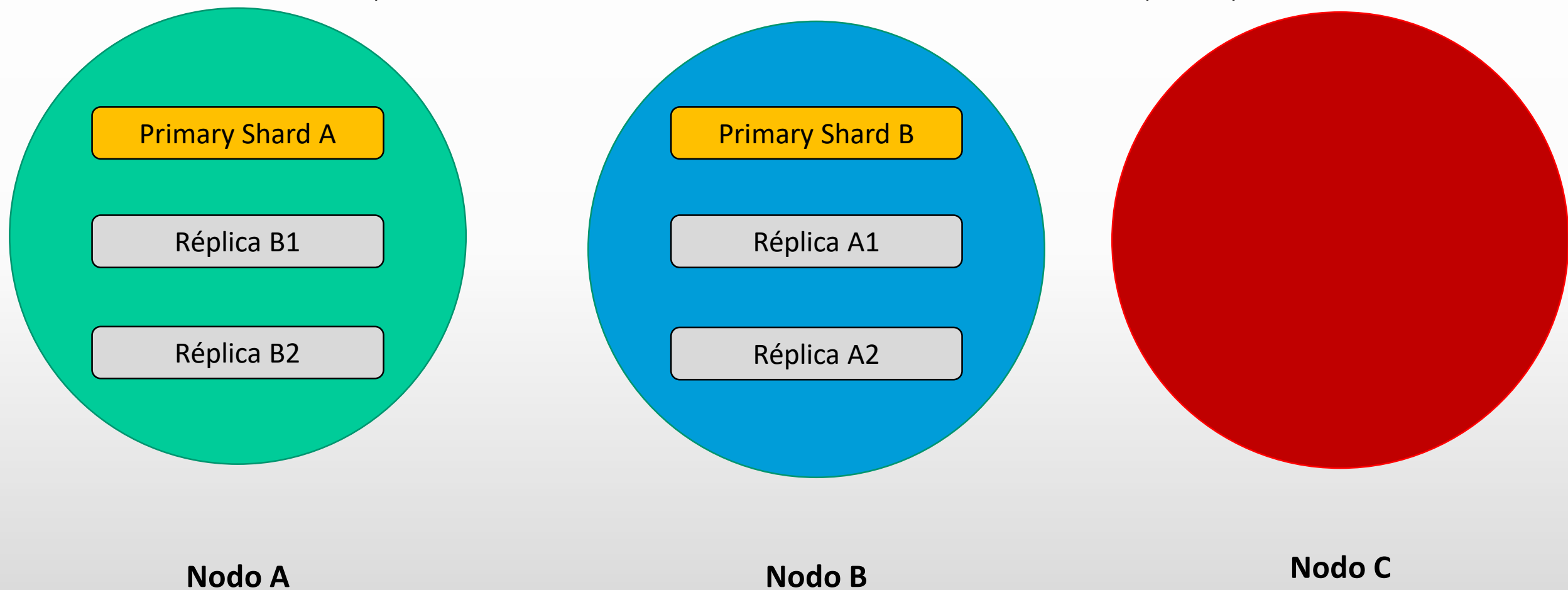
Índice



Replicación en Elastic Stack

¿Cómo se reparten las réplicas?

- Las réplicas deben estar almacenadas en un nodo diferente del primary shard



- Elasticsearch reconoce automáticamente cuando se añaden más nodos al cluster para distribuir las réplicas de manera autónoma.

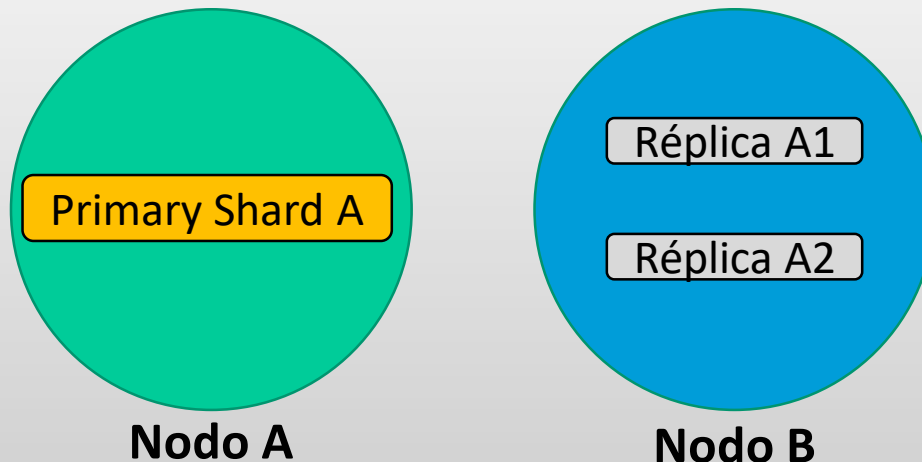
Replicación en Elastic Stack

¿Cuál es el número de replicas recomendado?

- Depende de la **críticidad** del caso de uso:
 - ¿Tenemos un respaldo de los datos en una RDBMS?
 - ¿Podemos permitir periodos de indisponibilidad durante la restauración?
- Para escenarios críticos se recomienda replicar al menos 2 veces.
- Es posible utilizar **snapshots** de Elasticsearch para realizar backup a un momento específico en el tiempo (a nivel de índice o el cluster al completo, típico realizar backups diarios), la replicación proporciona alta disponibilidad de los datos generados en tiempo real.

¿Pueden las réplicas aumentar el rendimiento?

- Las réplicas también pueden servir solicitudes igual que las primary shards y Elasticsearch direcciona las solicitudes a la “mejor shard” mejorando la paralelización de CPU:



Añadir la Réplica B2 no mejora la disponibilidad pero sí el rendimiento (por ejemplo ejecutar 3 solicitudes a la vez sobre el mismo índice)

***Importante: Cada nodo normalmente tiene varias CPUs**

Replicación en Elastic Stack

¿Cómo creamos un índice?

- PUT /Nombre_index

¿Cómo visualizamos los shards?

- GET /_cat/shards?v

¿Qué son los roles de los nodos de Elastic?

Rol Master node

- Responsable de crear/eliminar índices (entre otras).
- El rol master node no habilita a que automáticamente un nodo sea maestro.
- Puede ser útil en el caso de tener un nodo maestro dedicado (clusters de gran tamaño que requieren un nodo dedicado a ello para mantener la estabilidad).
- **Configuración:**
 - `node.master: true | false`

Rol Data node

- Habilita al nodo a almacenar datos y realizarle solicitudes.
- Para clusters pequeños o medianos, siempre está activo este rol.
- Si hay un nodo master dedicado se le puede quitar este rol.
- **Configuración:**
 - `node.data: true | false`

Rol Ingest node

- Habilita a ejecutar **pipelines de ingestión de datos** (pasos de transformación antes de indexar documentos).
- Sería una versión simplificada de Logstash directamente dentro de Elasticsearch para transformaciones simples (si hay mayor complejidad es mejor usar Logstash).
- Este rol se usa cuando hay nodos dedicados a la ingesta.
- **Configuración:**
 - `node.ingest: true | false`

¿Qué son los roles de los nodos de Elastic?

Rol Machine Learning node

- Permite al nodo ejecutar trabajos de Machine Learning.
- Podemos habilitar que el nodo pueda operar como API Machine Learning.
- Típico utilizarlo para activar nodos dedicados a ML.
- **Configuración:**
 - `node.ml`: true | false
 - `xpack.ml.enable`: true | false

Rol Coordination node

- Rol para distribución de consultas internamente en el cluster y agregación de resultados.
- Útil para nodos de solo coordinación (clusters de gran tamaño).
- Sería como un repartidor de carga del cluster.
- **Configuración:**
 - Deshabilitar todos los demás roles

Rol Voting node

- Usado para habilitar al nodo a votar quién es el nodo master en el cluster.
- Muy poco usado
- **Configuración:**
 - `node.voting_only`: true | false

Por regla general, no se modifican los roles de los nodos a menos que haya una razón de peso (como por ejemplo si el cluster es de un tamaño considerable)



elastic

BLOQUE 2:
ELASTICSEARCH –
MANEJO DE DOCUMENTOS

Lo que aprenderemos en este bloque....

- ✓ ¿Cómo crear un index y sus documentos?
- ✓ ¿Cómo actualizar y eliminar documentos en Elasticsearch?
- ✓ ¿Cómo se controla la concurrencia de solicitudes en Elasticsearch?
- ✓ ¿Cómo podemos realizar operaciones masivas?

Creación de un índice, indexar documentos y consultarlos mediante el ID

¿Cómo creamos un índice?

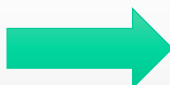


PUT /Nombre_index

```
{  
  "settings": {  
    "number_of_shards": 2,  
    "number_of_replicas": 2  
  }  
}
```

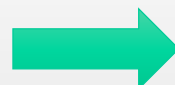
Indicar la configuración solo si hay un motivo para ellos, sino mejor dejarlo por defecto

¿Cómo eliminamos un índice?



• DELETE /Nombre_index

¿Cómo indexamos documentos?

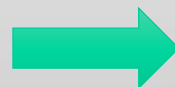


POST /Nombre_index/_doc

```
{  
  "atributo_1": valor,  
  "atributo_2": valor  
  ...  
}
```

Indicar todos los atributos del documento a añadir (si queremos especificar el atributo _id, se utilizar PUT en lugar de POST y especificamos el valor tras "_doc")

¿Cómo consultar documento a partir de ID?



GET /Nombre_index/_doc/ID

Actualización de documentos manual y mediante scripts en Elastic

¿Cómo actualizamos un documento?



Especificar el atributo a actualizar

```
#ACTUALIZAR UN DOCUMENTO POR ID
POST /productos/_update/1
{
  "doc": {
    "precio": 4
  }
}
```

¿Cómo añadimos un nuevo atributo?



Indicar el nuevo campo

```
POST /productos/_update/1
{
  "doc": {
    "etiquetas": ["comida"]
  }
}
```

¿Cómo actualizar documentos mediante scripts?



```
POST /productos/_update/1
{
  "script": {
    "source": "ctx._source.stock -= params.cantidad_comprada",
    "params": {
      "cantidad_comprada": 2
    }
  }
}
```

Uso de un parámetro

```
POST /productos/_update/1
{
  "script": {
    "source": ""
    if (ctx._source.stock > 0) {
      ctx._source.stock--;
    }
  }
}
```

Uso de una condición

*Los documentos son inmutables (no modificables), realmente lo que hace Elasticsearch es un reemplazo del documento

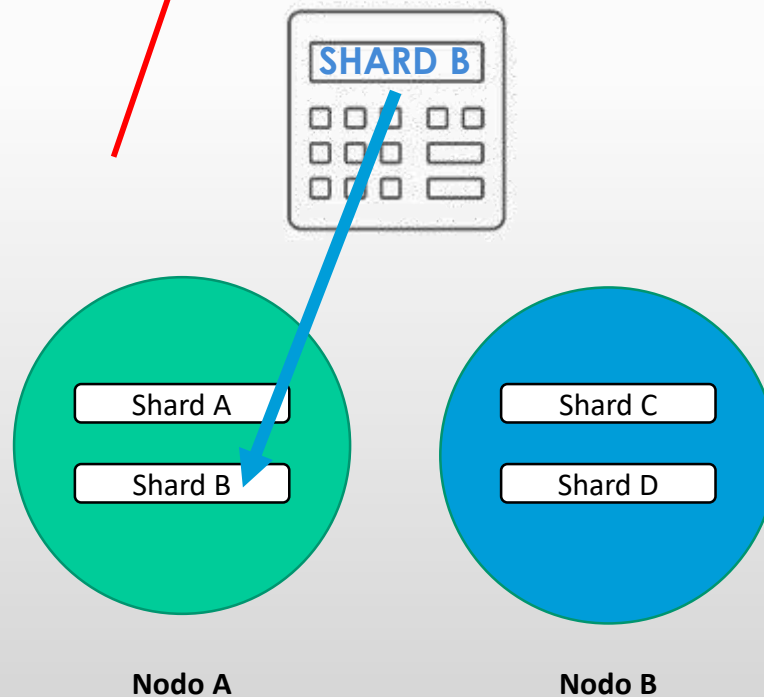
¿Cómo Elasticsearch lee y escribe datos?

Proceso de “Routing” (¿Dónde se encuentra mi documento?)

GET /productos/_doc/**100**
ID

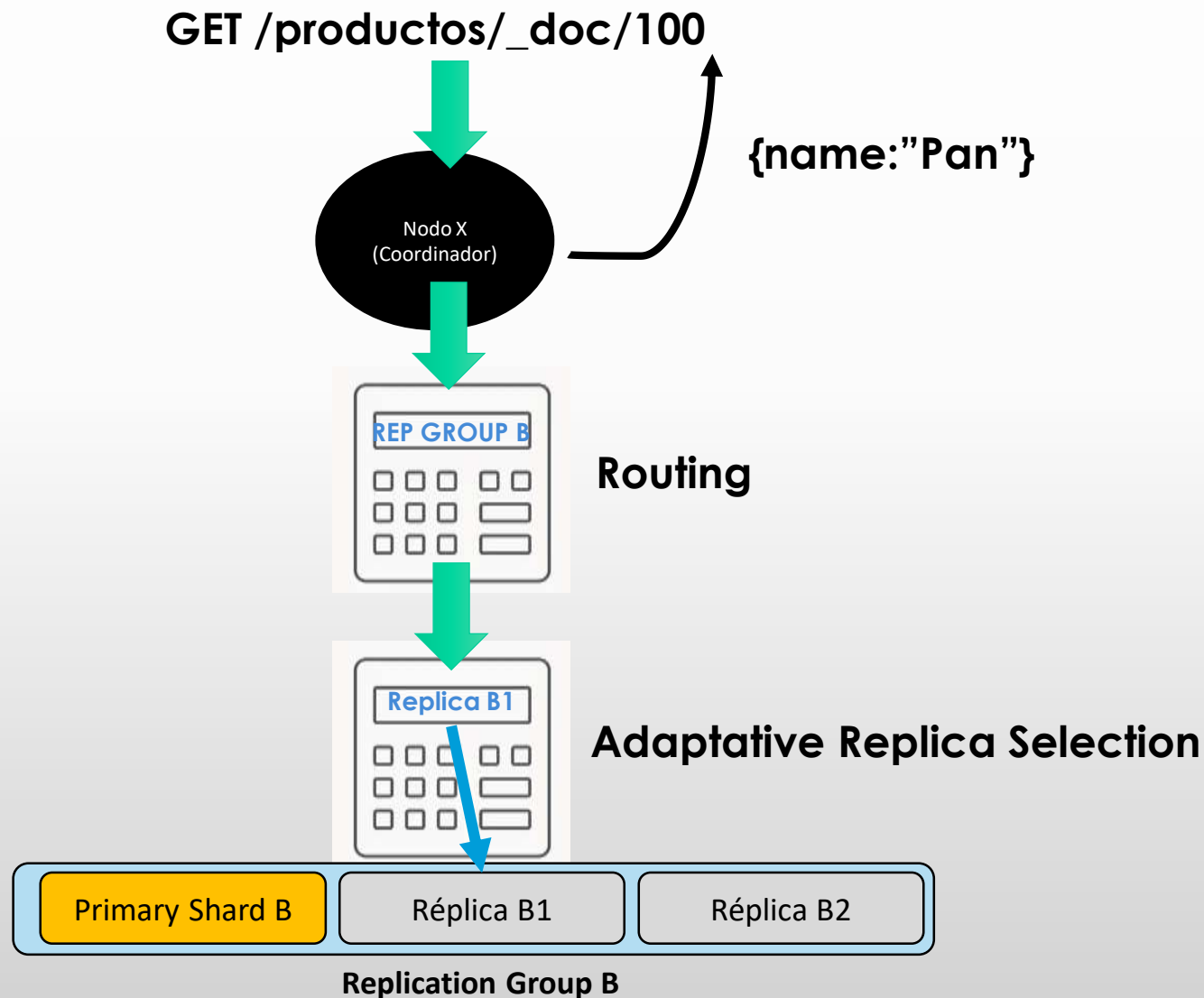
Shard_num = hash(**_routing**) % número_primary_shards

Valor que depende del ID del documento



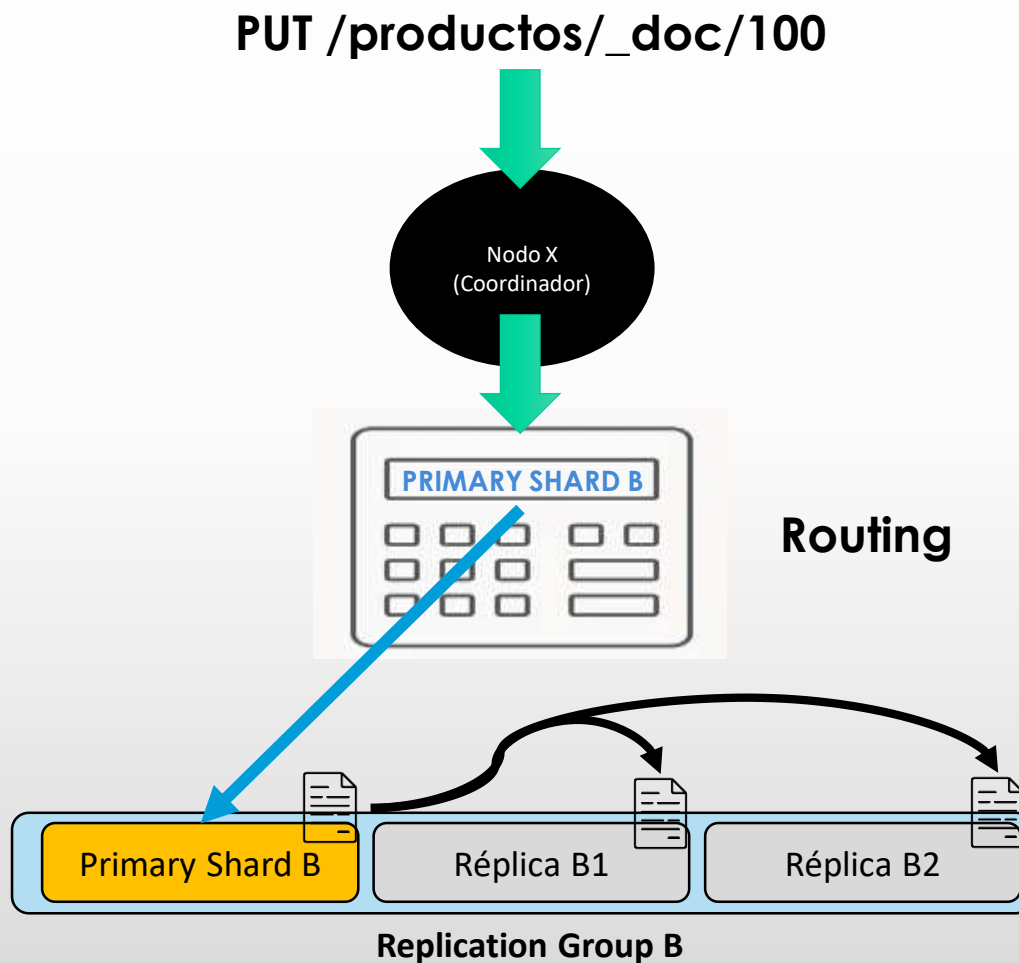
¿Cómo Elasticsearch lee y escribe datos?

Proceso de Lectura (¿Cómo extraigo los datos?)



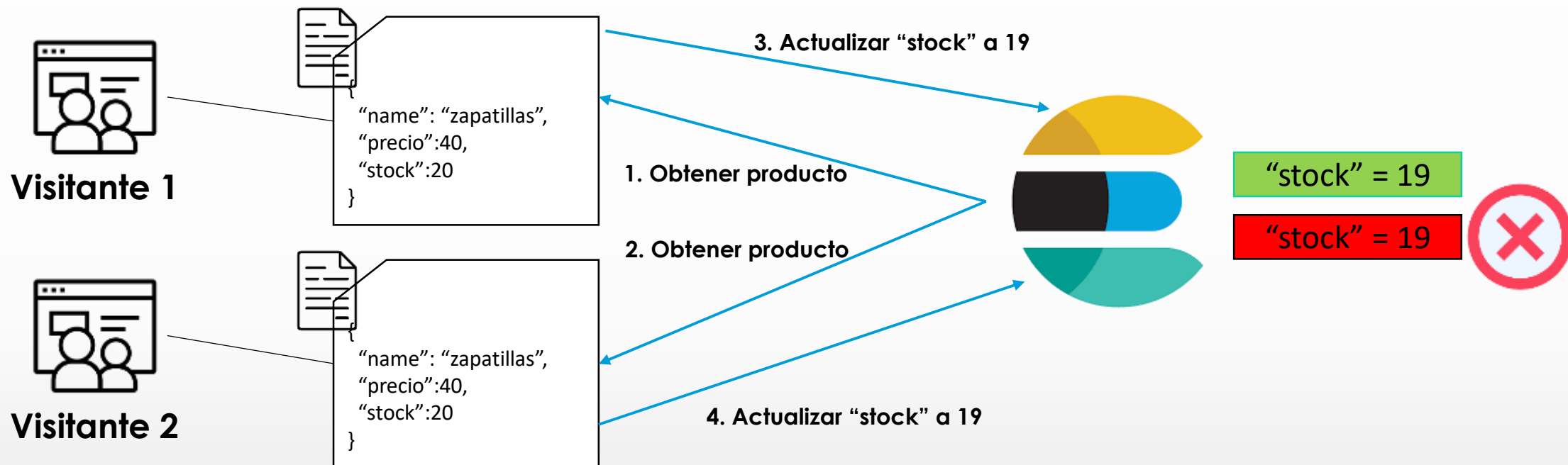
¿Cómo Elasticsearch lee y escribe datos?

Proceso de Escritura (¿Cómo escribo los datos?)

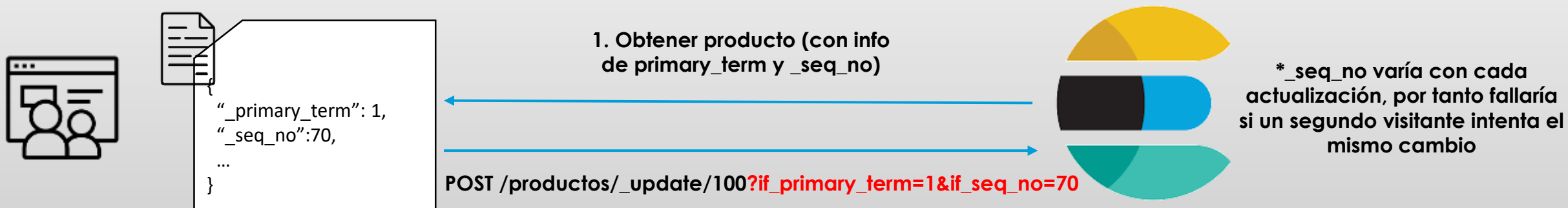


¿Cómo controlamos la concurrencia de solicitudes en Elasticsearch?

PROBLEMA DE OPERACIONES CONCURRENTES



SOLUCIÓN (uso de `primary_term` / `seq_no`)



Actualizar y eliminar masivamente a partir de consulta (Query)

¿Qué ocurre si quiero actualizar varios documentos?



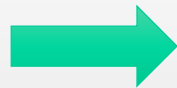
Usar “update_by_query”

Dentro de “Query” se establece la lógica de los documentos que serán actualizados (similar a UPDATE WHERE de SQL)

```
#ACTUALIZAR MASIVAMENTE DOCUMENTOS
POST /productos/_update_by_query
{
  "script": {
    "source": "ctx._source.stock--"
  },
  "query": {
    "match_all": {}
  }
}
```

Indicamos en “source” la modificación a realizar (ejemplo restar 1 al atributo “stock” de todos los documentos)

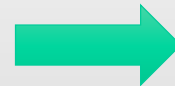
¿Cómo puedo consultar masivamente los documentos de un index?



Usar “_search” y la “Query” que especifica las condiciones de búsqueda

```
GET /productos/_search
{
  "query": {
    "match_all": {}
  }
}
```

¿Cómo puedo eliminar masivamente los documentos de un index?



Usar “delete_by_query”

Dentro de “Query” se establece la lógica de los documentos que serán eliminados

```
#ELIMINAR MASIVAMENTE DOCUMENTOS
POST /productos/_delete_by_query
{
  "query": {
    "match_all": {}
  }
}
```

Procesamiento masivo mediante bulk

¿Qué ocurre si queremos importar gran cantidad de datos o bien actualizarlos masivamente?



Usar Bulk API (mayor eficiencia)

Acción = "index" para crear un nuevo documento o reemplazar existente

Acción = "create" para crear un nuevo documento si no existe previamente

```
POST /_bulk
{ "index": { "_index": "productos", "_id": 200 } }
{ "name": "Cafetera", "precio": 199, "stock": 5 }
{ "create": { "_index": "productos", "_id": 201 } }
{ "name": "Aspirador", "precio": 149, "stock": 14 }
```

Acción = "update" para actualizar documento

Acción = "delete" para eliminar documento

```
POST /_bulk
{ "update": { "_index": "productos", "_id": 201 } }
{ "doc": { "precio": 129 } }
{ "delete": { "_index": "productos", "_id": 200 } }
```

Si solo modificamos un index entonces se puede insertar en la request de HTTP

```
POST /productos/ bulk
{ "update": { "_id": 201 } }
{ "doc": { "precio": 129 } }
{ "delete": { "_id": 200 } }
```

*Este formato es NDJSON, la consola nos realiza el trabajo pero si no comunicamos a través de una aplicación, entonces hay que especificar application/x-ndjson

Importación de datos con cURL

1. Preparar documento ndjson



2. Ejecutar cURL



3. Lanzar comando

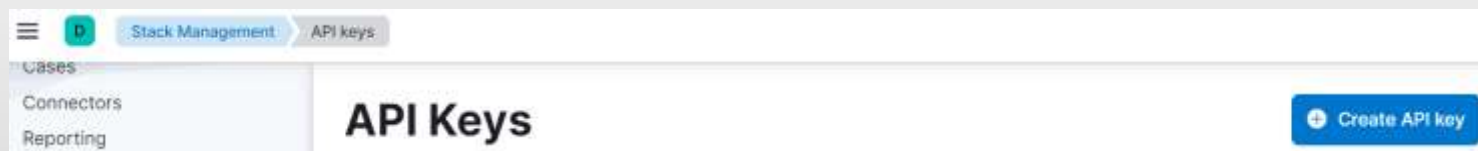
```
1360 {"name":"Pork - loin Bone - In","price":100,"in_stock":44,"sold":499,"tags":["Meat"],"description":"P"}
1361 {"index":{"_id":905}}
1362 {"name":"Cheese and pumpkin risotto","price":49,"in_stock":17,"sold":99,"tags":[],"description":"Crea"}
1363 {"index":{"_id":906}}
1364 {"name":"Mapkin - Beverage White 2 - Fly","price":10,"in_stock":40,"sold":100,"tags":[],"description":""}
1365 {"index":{"_id":907}}
1366 {"name":"Lid Tray - 16in Dome","price":10,"in_stock":99,"sold":497,"tags":[],"description":"Nulla m"}
1367 {"index":{"_id":908}}
1368 {"name":"Sage - Fresh","price":140,"in_stock":40,"sold":100,"tags":["Spice"],"description":"Integer a"}
1369 {"index":{"_id":909}}
1370 {"name":"Pepper - Yellow Bell","price":90,"in_stock":3,"sold":100,"tags":["Fruit"],"description":"Se"}
1371 {"index":{"_id":910}}
1372 {"name":"Corn - Mini","price":90,"in_stock":11,"sold":100,"tags":["Vegetable"],"description":"Eciam v"}
1373 {"index":{"_id":911}}
1374 {"name":"Yogurt - Peach 175 Oz","price":100,"in_stock":1,"sold":100,"tags":[],"description":"Cras m"}
1375 {"index":{"_id":912}}
1376 {"name":"Wine - Alsace Riesling Reserve","price":10,"in_stock":10,"sold":100,"tags":["Beverage"],"Alco"}
1377 {"index":{"_id":913}}
1378 {"name":"Sauce - Black Currant Dry Mix","price":170,"in_stock":11,"sold":100,"tags":[],"description":""}
1379 {"index":{"_id":914}}
1380 {"name":"Lamb - Loin Trimmed Boneless","price":100,"in_stock":10,"sold":100,"tags":["Meat"],"descripti"}
1381 {"index":{"_id":915}}
1382 {"name":"Toothpick Filled","price":101,"in_stock":10,"sold":100,"tags":[],"description":"Integer ac n"}
1383 {"index":{"_id":916}}
```

Obligatorio que haya un salto de línea
(\n) al final del documento

- Comprobar que cURL está disponible:

```
C:\Users\ivan_pinar>curl --version
```

- Si versión de Windows antigua entonces descargar cURL:
<https://curl.se/download.html>



```
curl -k -H "Content-Type: application/x-ndjson" -H "Authorization: ApiKey
cadena_api" -X POST https://nombre_url:9243/nombre_index/_bulk --data-
binary "@nombre_fichero.json"
```



elastic

BLOQUE 3:
ELASTICSEARCH –
TÉCNICAS DE MAPPING Y ANÁLISIS

Lo que aprenderemos en este bloque....

- ✓ ¿Qué tipos de datos tenemos en Elasticsearch?
- ✓ ¿Cómo podemos hacer un mapeo a los campos?
- ✓ ¿Cómo reindexar un índice?
- ✓ ¿Cómo podemos crear analizadores?

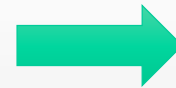
Introducción al análisis y al uso de la API Analyze



```
1 {  
2   "_index": "productos",  
3   "_id": "10001",  
4   "_version": 1,  
5   "_seq_no": 1425,  
6   "_primary_term": 1,  
7   "found": true,  
8   "_source": {  
9     "nombre": "aspirador",  
10    "precio": 49,  
11    "descripcion": "Robot aspirador con garantía de 5 años"  
12  }  
13 }
```

Los datos están en el campo `_source` del documento

¿Realiza Elasticsearch la búsqueda de campos de texto en base al campo `_source` de todos los documentos?



NO, cuando se indexan documentos, los campos de texto son analizados por Elasticsearch



Introducción al análisis y al uso de la API Analyze



Analizador

Filtros de caracteres

- Añade, elimina o modifica caracteres
- Los analizadores pueden tener varios filtros de caracteres
- Son aplicados en el orden especificado

Ejemplo: filtro `html_strip`

Entrada: "Quiero **** modificar**** caracteres"

Salida: "Quiero modificar caracteres"

Analizar estándar

"Ninguno"

Tokenizador

- División de una cadena en tokens
- Los analizadores contiene un tokenizador
- Los caracteres especiales pueden ser eliminados en el proceso.

Ejemplo:

Entrada: "¡Me ENCANTAN los coches!"

Salida: ["Me", "ENCANTAN", "los", "coches"]

"standard"

Filtros de token

- Añade, elimina o modifica tokens
- Los analizadores pueden tener varios filtros de tokens
- Son aplicados en el orden especificado

Ejemplo: filtro `lowercase`

Entrada: ["Me", "ENCANTAN", "los", "coches"]

Salida: ["me", "encantan", "los", "coches"]

"lowercase"

¿Qué son los índices invertidos para mejorar la eficiencia de Elasticsearch?

- Para conseguir máxima eficiencia, Elasticsearch utiliza el concepto de **“índices invertidos”**



Entrada 1: “¡Me ENCANTAN los coches!” → Salida 1: [“me”, “encantan”, “los”, “coches”]



Entrada 2: “Nos reuniremos tras aparcar los coches” → Salida 2: [“nos”, “reuniremos”, “tras”, “aparcar”, “los”, “coches”]



Entrada 3: “Los coches nos ENCANTAN” → Salida 3: [“los”, “coches”, “nos”, “encantan”]

TÉRMINO	DOCUMENTO#1	DOCUMENTO#2	DOCUMENTO#3
me	X		
encantan	X		X
los	X	X	X
coches	X	X	X
nos		X	X
reuniremos		X	
tras		X	
aparcar		X	

Ordena
alfabéticamente
y se establece
puntuación de
relevancia*

***¡Se realiza por
cada campo de
texto de los
documentos!**

Tipos de datos en Elasticsearch

Object (=json)

Float

Text

Integer

Date

Keyword

<https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-types.html>

Tipos de datos en Elasticsearch

Tipo de dato: Keyword

- Se usa para la búsqueda **exacta** de valores
- Útil para **filtrado, agregaciones y ordenaciones** (ejemplo, buscar artículos por estado “PUBLICADO”)
- Para búsquedas de texto completo, se usar el tipo “texto” en lugar de “keyword” (ejemplo, buscar un texto dentro del contenido de un artículo)
- Los campos “keyword” son analizados con el **analizador keyword** (se genera un token que contiene todo el campo sin modificar nada)

Documento 1

```
{  
  "nombre": "Iván",  
  "email": "usuarioelastic@aac.com"  
}
```

Documento 2

```
{  
  "nombre": "Eduardo",  
  "email": "aprendiendoelk@aac.com"  
}
```

Documento 3

```
{  
  "nombre": "Marta",  
  "email": "elastic@aac.com"  
}
```

Tipo keyword

TÉRMINO	DOCUMENTO#1	DOCUMENTO#2	DOCUMENTO#3
usuarioelastic@aac.com	X		
aprendiendoelk@aac.com		X	
elastic@aac.com			X

Tipos de datos en Elasticsearch

¿Cómo detecta Elasticsearch el tipo de datos?

- Los tipos de datos se inspeccionan cuando se indexan documentos.
- Es importante usar el tipo de datos correcto para el campo, sobre todo cuando se indexa por **primera vez el campo**.
- Se usa la técnica “coerción” por defecto a menos que se desactive para los siguientes valores de ese campo.

Inspección “precio” → float

Mapping:

“precio”:{“type”:“float”}

```
PUT /Nombre_index/_doc/1
{
  “precio”: 7.4
}
```

7.4 (float)

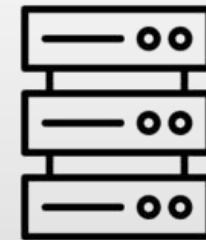


7.4 (float)

```
PUT /Nombre_index/_doc/2
{
  “precio”: “5.3”
}
```

“5.3” (string)

5.3 (float)



Almacenamiento

Coercion

Se inspecciona el campo para mapearlo al tipo de datos si es posible

¿Cómo definir mapeos explícitos y añadir nuevos mapeos?

¿Cómo hacemos un mapeo manual del index? →

Objeto “mapping” y sus “properties”

Definir todos los “type” de cada campo

```
PUT /valoraciones
{
  "mappings": {
    "properties": {
      "puntuacion": { "type": "float" },
      "contenido": { "type": "text" },
      "producto_id": { "type": "integer" },
      "autor": {
        "properties": {
          "nombre": { "type": "text" },
          "apellidos": { "type": "text" },
          "email": { "type": "keyword" }
        }
      }
    }
  }
}
```

¿Cómo consultamos el mapeo de un index? →

```
GET /valoraciones/_mapping
```

```
GET /valoraciones/_mapping/field/contenido
```

Campo específico

¿Cómo añadimos el mapeo de un nuevo campo? →

```
PUT /valoraciones/_mapping
{
  "properties": {
    "fecha_creación": {
      "type": "date"
    }
  }
}
```

Comando “_mapping”

Insertar nuevo campo en “properties”

¿Qué son los parámetros de mapeo y cómo aplicarlos?

- **Parámetros:** Se usan para modificar el comportamiento y características de un campo.

<https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-params.html>

Parámetro “format”

- Permiten personalizar el formato de un campo “**date**” (fechas).
- El formato por defecto de las fechas (y recomendado) sigue la ISO8601: “2024-05-21T13:07:41+01:00”
- Se pueden utilizar múltiples formatos como por ejemplo sintaxis de Java como “dd/MM/yyyy”
- Los formatos posibles están en <https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-date-format.html>

```
PUT /Nombre_index
{
  "mappings": {
    "properties": {
      "fecha_compra": {
        "type": "date",
        "format": "dd/MM/yyyy"
      }
    }
  }
}
```


¿Qué son los parámetros de mapeo y cómo aplicarlos?

Parámetro “coerce”

- Usado para habilitar o deshabilitar la coerción de valores (habilitado por defecto)

```
PUT /Nombre_index
{
  "mappings": {
    "properties": {
      "importe": {
        "type": "float",
        "coerce": false
      }
    }
  }
}
```

¿Qué son los parámetros de mapeo y cómo aplicarlos?

Parámetro “copy_to”

- Usado para copiar múltiples campos en un campo “agregado” (campo objetivo).
- Se copian los valores, pero no los términos y tokens.
- El campo objetivo no es parte del objeto _source.

```
PUT /Nombre_index
{
  "mappings": {
    "properties": {
      "nombre": {
        "type": "text",
        "copy_to": "nombre_completo"
      }
      "apellidos": {
        "type": "text",
        "copy_to": "nombre_completo"
      }
      "nombre_completo": {
        "type": "text",
      }
    }
  }
}
```

```
POST /Nombre_index/_doc
{
  "nombre": "Iván",
  "apellidos": "Pinar"
}
```

¿Qué son los parámetros de mapeo y cómo aplicarlos?

¿Qué podemos hacer si queremos liberar espacio en disco?

Parámetro “doc_values”

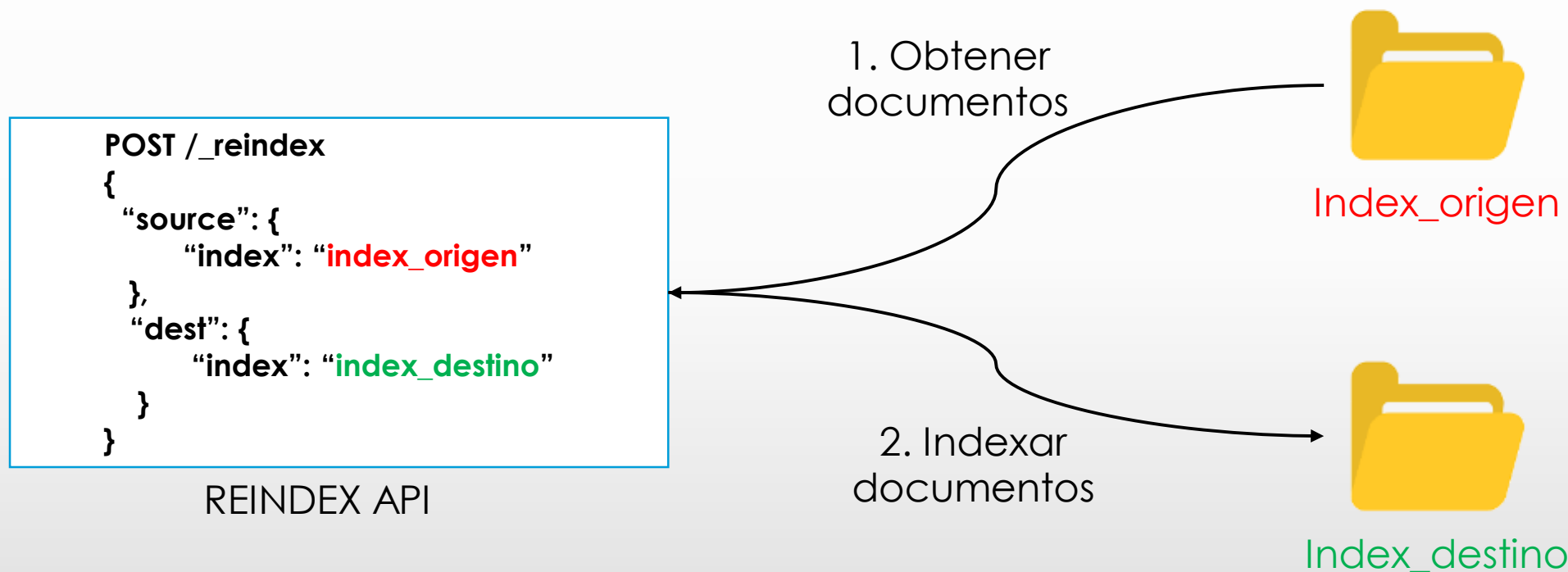
- Cambia el tipo de estructura de datos por defecto afectando al índice inverso.
- **Solo deshabilitar si no queremos hacer agregaciones, ordenaciones o scripting sobre el campo (“doc_values”: False)**
- No se puede modificar sin reindexar documentos en un nuevo index.

Parámetro “norms”

- Parámetros utilizado para valorar la relevancia (normalmente no solo queremos filtrar resultados, sino conseguir un ranking)
- **Solo deshabilitar si no el campo para ordenar por relevancia, por ejemplo, un campo de etiquetas (“norms”: False)**

Reindexación de documentos con la API Reindex

- **Reindexar:** Mover documentos a un nuevo índice por necesidades de cambio de mapping.



Reindexación de documentos con la API Reindex

¿Podemos reindexar con modificaciones?

```
#REINDEXAR MODIFICANDO producto_id A STRING
POST /_reindex
{
  "source": {
    "index": "valoraciones"
  },
  "dest": {
    "index": "valoraciones_nuevo"
  },
  "script": {
    "source": ""
    if (ctx._source.product_id != null) {
      ctx._source.product_id = ctx._source.product_id.toString();
    }
    ""
  }
}
```

Si el documento
cumple una condición

```
POST /_reindex
{
  "source": {
    "index": "valoraciones",
    "query": {
      "range": {
        "puntuacion": {
          "gte": 4.5
        }
      }
    }
  },
  "dest": {
    "index": "valoraciones_nuevo"
  }
}
```

Modificando el tipo de dato

```
POST /_reindex
{
  "source": {
    "index": "valoraciones",
    "_source": ["puntuacion", "autor"]
  },
  "dest": {
    "index": "valoraciones_nuevo"
  }
}
```

Modificando el
nombre del campo

```
POST /_reindex
{
  "source": {
    "index": "valoraciones"
  },
  "dest": {
    "index": "valoraciones_nuevo"
  },
  "script": {
    "source": ""
    ctx._source.descripcion = ctx._source.remove("contenido");
    ""
  }
}
```

Solo algunos campos

Aplicación de plantillas de mapeo a índices

¿Podemos reutilizar un mapeo para ser aplicable en múltiples índices?

- logs_2023-01
- logs_2023-02
- logs_2023-03
- logs_2023-04
- ...



Aplicación de templates con
el mapping usando el
asterisco (*)



“logs_”

Índices mensuales (Alto volumen de datos)

- Un índice puede cumplir criterios de varias plantillas.
- Se define el parámetro “order” en la definición de la plantilla para indicar la prioridad (cuanto más bajo el valor más prioritaria la plantilla).

Recomendaciones prácticas de mapeo

- Usar mapeo dinámico por defecto es conveniente, pero cuando se pasa a producción o hay alto volumen de datos se puede liberar espacio con mapeos explícitos:
 - Si necesitas búsquedas de texto completa → *text* mapping
 - Si necesitas realizar agregaciones, ordenaciones o filtrados por valores exactos → *keyword* mapping
 - Para valores enteros normalmente es suficiente con *integer* (en lugar de *long* que ocupa más espacio)
 - Para valores decimales normalmente es suficiente con *float* (en lugar de *double*)
- Setear dentro de “*mappings*” la propiedad “*dynamic*” a “*strict*” para solo indexar documentos cuyos campos cumplan todas las condiciones de tipo de mapeo.
- Deshabilitar “*doc_values*” si no queremos hacer agregaciones, ordenaciones o scripting.
- Deshabilitar “*norms*” si no queremos el campo para ordenar por relevancia.

Solo merece la pena cambiar el comportamiento por defecto si indexamos gran cantidad de documentos

Técnicas stemming y palabras de parada

“Me gustaba saltar mientras estaba corriendo por las aceras”

Verbo en pasado

Gerundio

Plural

```
POST /Nombre_index/_doc
{
  "descripción": "Me gustaba saltar
mientras estaba corriendo por las
aceras"
}
```

```
GET /Nombre_index/_search
{
  "query": {
    "match": {
      "descripción": "gustar"
    }
  }
}
```



Analizador
estándar



Sin resultado

TÉRMINO	DOCUMENTO#1
me	X
gustaba	X
saltar	X
mientras	X
estaba	X
corriendo	X
por	X
las	X
aceras	X

Técnicas stemming y palabras de parada

- **Steeming**: Reducir las palabras a su forma raíz (se realiza por defecto en Elasticsearch).
 - Ejemplo: “gustaba” → “gustar”

“Me gustaba saltar mientras estaba corriendo por las aceras”



*“Me **gustar** saltar mientras **estar** correr por la **acera**”*

- **Palabras de parada**: Palabras que son filtradas durante el análisis de texto ya que tienen poca relevancia (“a”, “el”, “de”, “un”,...)
 - Muy común eliminarlas (por ejemplo en buscador Google).
 - En **Elasticsearch** el tratamiento se realiza automáticamente, aunque es menos común puesto que los algoritmos de relevancia han mejorado drásticamente.

Analizadores predefinidos (built-in)

“¿Es el perro de Pedro?”

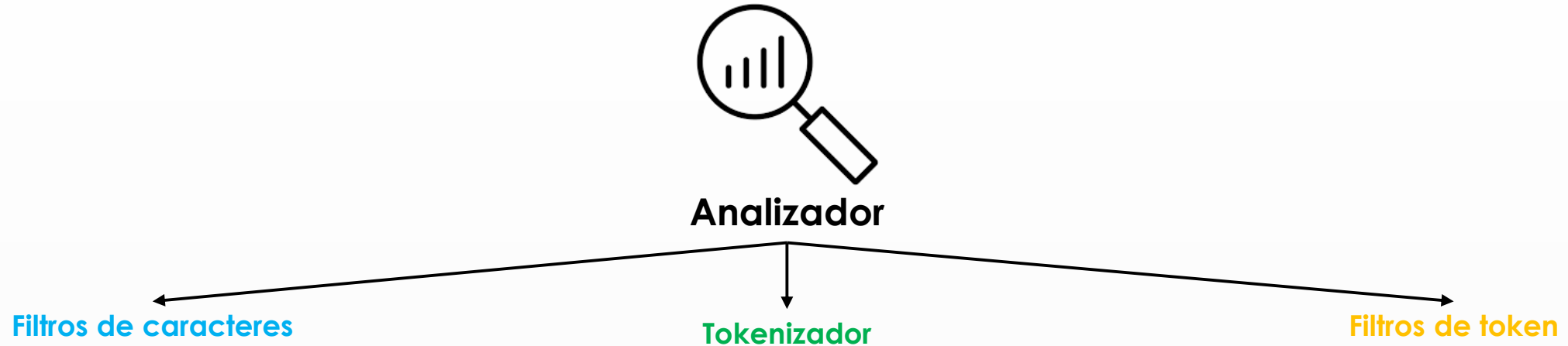
Nombre analizador	Características	Resultado
Standard	<ul style="list-style-type: none">Divide el texto por palabras y elimina símbolos de puntuación.Convierte a minúsculas	<i>[“es”, “el”, “perro”, “de”, “pedro”]</i>
Keyword	<ul style="list-style-type: none">Deja la entrada de texto intactaUtilizado para campos de tipo keyword	<i>[“¿Es el perro de Pedro?”]</i>
Whitespace	<ul style="list-style-type: none">Divide el texto en tokens a partir de los espacios en blancoNo convierte a minúscula	<i>[“¿Es”, “el”, “perro”, “de”, “Pedro?”]</i>
Pattern	<ul style="list-style-type: none">Se evalúa una expresión regular para separar en tokens (ejemplo: “//”)Muy flexible, por defecto divide por los caracteres que no son palabrasConvierte a minúsculas	<i>[“es”, “el”, “perro”, “de”, “pedro”]</i>

Web referencia analizadores built-in:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-analyzers.html>

```
PUT /Nombre_index
{
  "mappings": {
    "properties": {
      "descripción": {
        "type": "text",
        "analyzer": "spanish"
      }
    }
  }
}
```

Analizadores personalizados



```
PUT /index_analizador_test_1
{
  "settings": {
    "analysis": {
      "analyzer": {
        "mi_analizador_personalizado": {
          "type": "custom",
          "char filter": ["html_strip"],
          "tokenizer": "standard",
          "filter": [
            "lowercase"
          ]
        }
      }
    }
  }
}
```

Nombre analizador

Web referencia analizadores personalizados

<https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-custom-analyzer.html>

- Es posible **actualizar** los analizadores.
- Hay que intentar definir correctamente los analizadores **antes** de indexar documentos, si no es posible, habrá una **indisponibilidad** durante la actualización (o bien se puede reindexar los documentos si no se permiten tiempos de indisponibilidad).



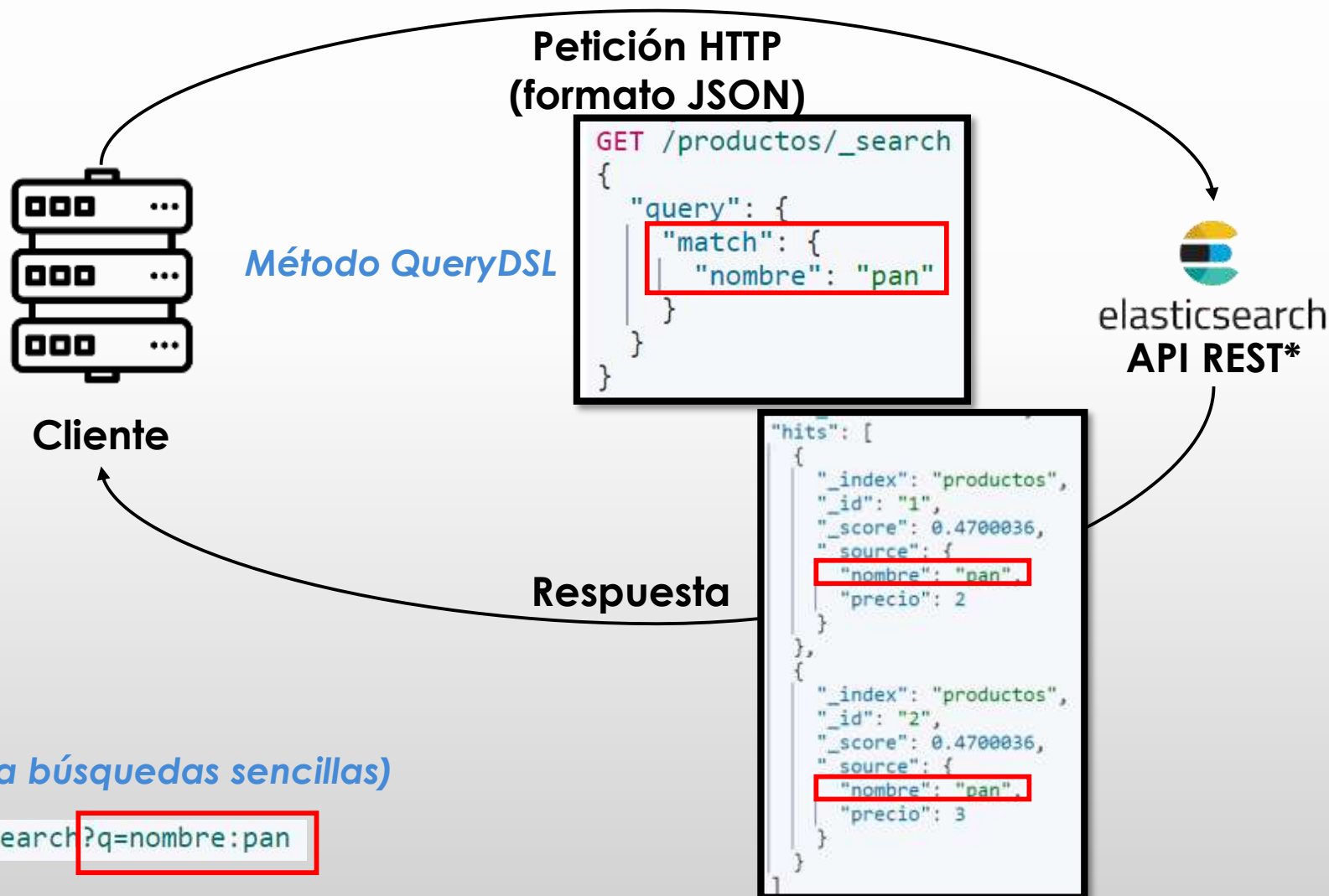
BLOQUE 4:
ELASTICSEARCH –
BÚSQUEDAS TERM-LEVEL, FULL-TEXT Y
BOOLEANAS

Lo que aprenderemos en este bloque....

- ✓ ¿Qué métodos de búsqueda existen en Elasticsearch?
- ✓ ¿Por qué Elasticsearch utiliza la puntuación de relevancia?
- ✓ ¿Para qué son las consultas “term-level”?
- ✓ ¿Para qué son las consultas “full-text”?
- ✓ ¿Cómo aplicar potentes consultas booleanas con **MUST**, **MUST NOT**, **FILTER** y **SHOULD**?

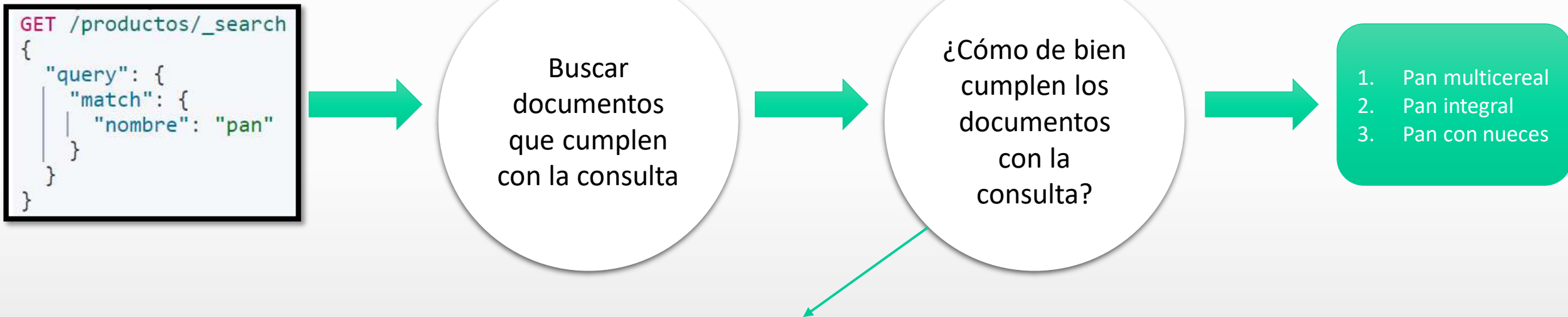
Métodos de búsqueda QueryDSL vs búsqueda URI

¿Cómo hacemos consultas en Elasticsearch?



¿Qué es la puntuación de relevancia en las búsquedas?

- A diferencia de una BBDD relacional, en Elasticsearch se evalúan los resultados que tienen una mayor relevancia respecto a la búsqueda realizada:



- Actualmente en Elasticsearch se utiliza el algoritmo **Okapi BM25**:
 - **Term Frequency**: ¿Cuántas veces aparece el término de búsqueda en el documento?
 - **Inverse Document Frequency**: ¿Cuántas veces aparece el término en el index (en todos los documentos)?
 - **Field length norm**: ¿Cómo de largo es el campo? Si aparece el término en campos más pequeños entonces tendrá más peso

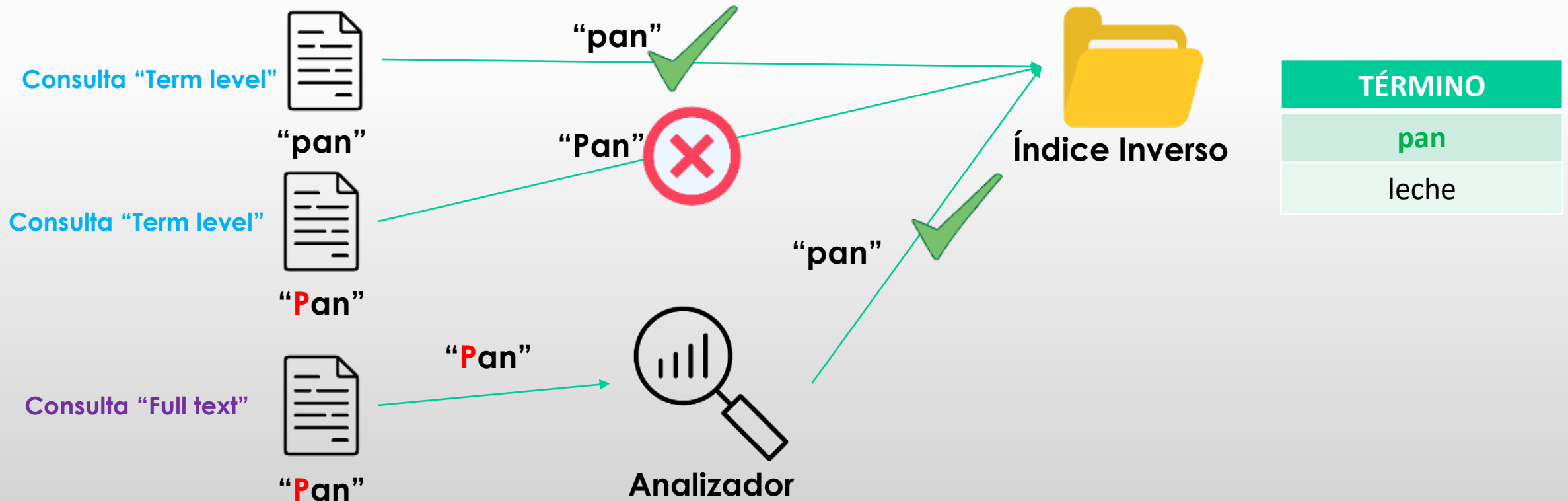
Diferencia entre consultas “full-text” y “term level”

Consulta “Term level”

- Realiza la búsqueda por el término de manera exacta
- **No** se aplica analizador antes de buscar en el índice inverso.
- Utilizadas para consulta de fechas, identificadores, categorías,...

Consulta “Full-text”

- Realiza búsqueda en todo el campo sin que la coincidencia sea completa.
- Se aplica el analizador sobre la consulta antes de buscar en el índice inverso.
- Utilizadas para consultar dentro de campos de texto, descripciones, encabezados, artículos,...



Búsquedas “Term level” – 1 o múltiples términos o IDs

Consulta por un 1 término

```
GET /productos/_search
{
  "query": {
    "term": {
      "nombre": "peras"
    }
  }
}
```

El tipo de consulta es “term”

Clave : valor por la que buscar

El tipo de consulta es “terms”

Consulta por varios términos

```
GET /productos/_search
{
  "query": {
    "terms": {
      "nombre": ["peras", "naranjas"]
    }
  }
}
```

Array de valores por los que buscar

Consulta por IDs

```
GET /productos/_search
{
  "query": {
    "ids": {
      "values": [ 100, 102, 103 ]
    }
  }
}
```

El tipo de consulta es “ids”

Array de IDs por los que buscar

Búsquedas “Term level” – Rango de valores o de fechas

Consulta por rango de valores

```
GET /productos/_search
{
  "query": {
    "range": {
      "stock": {
        "gte": 25,
        "lte": 45
      }
    }
  }
}
```

El tipo de consulta es “range”

Condiciones mayor o igual que (“gte”) y menor o igual que (“lte”)

Formato por defecto: “yyyy/MM/dd”

Consulta por rango de fechas

```
GET /productos/_search
{
  "query": {
    "range": {
      "fecha_reg": {
        "gte": "2024/04/01",
        "lte": "2024/07/01"
      }
    }
  }
}
```

Consulta por rango de fechas con formato personalizado

```
GET /productos/_search
{
  "query": {
    "range": {
      "fecha_reg": {
        "gte": "01-04-2024",
        "lte": "01-07-2024",
        "format": "dd-MM-yyyy"
      }
    }
  }
}
```

Especificamos el formato personalizado de la búsqueda

Búsquedas “Term level” – Trabajar con fechas relativas

Formato:

Fecha anclaje + | | + **redondeo** /x + desplazamiento fechas

Ejemplo “3 días después del 03/04/2024”:

“2024/04/03 | | +3d”

Ejemplo “3 días después del 03/04/2024
redondeado al mes”:

“2024/04/03 | | /M+3d”

Ejemplo “Fecha mayor que el momento
actual menos 1 mes”:

“now-1M”

Web referencia fechas relativas

<https://www.elastic.co/guide/en/elasticsearch/reference/current/common-options.html#date-math>

y	Years
M	Months
w	Weeks
d	Days
h	Hours
H	Hours
m	Minutes
s	Seconds

Búsquedas “Term level” – No nulos, prefijo, comodín y expresión regular

Búsqueda por no nulos:

```
GET /productos/_search
{
  "query": {
    "exists": {
      "field": "precio"
    }
  }
}
```

El tipo de consulta es “exists”

Campo por el que obtener resultados no nulos

Búsqueda por un prefijo:

```
GET /productos/_search
{
  "query": {
    "prefix": {
      "nombre": "pa"
    }
  }
}
```

El tipo de consulta es “prefix”

Campo y prefijo por el que buscar

Búsqueda con comodín:

```
GET /productos/_search
{
  "query": {
    "wildcard": {
      "nombre": "mer*da"
    }
  }
}
```

```
GET /productos/_search
{
  "query": {
    "wildcard": {
      "nombre": "me?melada"
    }
  }
}
```

El tipo de consulta es “wildcard”

“*” sustituye cualquier cadena intermedia
“?” solo sustituye 1 carácter
(cuidado con el rendimiento)

Búsqueda con expresión regular:

```
GET /productos/_search
{
  "query": {
    "regexp": {
      "nombre": "mermel[a-zA-Z]da"
    }
  }
}
```

El tipo de consulta es “regexp”

Insertar expresión regular esperada

Web referencia expresiones regulares

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-regexp-query.html#regexp-syntax>

Búsquedas “Full-text” – Coincidencia flexible con “match”

```
GET /productos2/_search
{
  "query": {
    "match": {
      "name": "productos que tienen chocolate"
    }
  }
}
```

El tipo de consulta es “match”

Indicamos campo en el que buscar y la búsqueda de texto completo (por defecto devuelve resultados que contengan algunas de las palabras – modo OR)

```
GET /productos2/_search
{
  "query": {
    "match": {
      "name": {
        "query": "productos que tienen chocolate",
        "operator": "and"
      }
    }
  }
}
```

Especificamos el operador “and” si deben aparecer todas las palabras en el campo

Búsquedas “Full-text” – Múltiples campos y frases completas

Búsqueda múltiples campos:

```
GET /productos2/_search
{
  "query": {
    "multi_match": {
      "query": "chocolate",
      "fields": [ "name", "etiquetas" ]
    }
  }
}
```

El tipo de consulta es “multi_match”

Valor a buscar

En qué campos buscar

Búsqueda frases completas:

```
GET /productos2/_search
{
  "query": {
    "match_phrase": {
      "name": "chocolate milk"
    }
  }
}
```

El tipo de consulta es “match_phrase”

Frase a buscar (PUEDE HABER PALABRAS EN MEDIO PERO SE DEBE RESPETAR EL ORDEN INDICADO)

Búsquedas “booleanas” – Must, must not, should y filter

Búsqueda booleanas - MUST

```
GET /productos2/_search
```

```
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "name": "chocolate"
          }
        },
        {
          "range": {
            "precio": {
              "lte": 150
            }
          }
        }
      ]
    }
  }
}
```

El tipo de consulta es “must” para indicar que es obligatorio que se cumplan las condiciones para devolver resultados

CONDICIÓN 1

CONDICIÓN 2

Búsqueda booleanas – MUST + FILTER

```
GET /productos2/_search
```

```
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "name": "chocolate"
          }
        }
      ],
      "filter": [
        {
          "range": {
            "precio": {
              "lte": 150
            }
          }
        }
      ]
    }
  }
}
```

El tipo de consulta es “filter” si lo que necesitamos es solo filtrar datos por una condición si/no (búsqueda de filtro) en lugar de que se dé una puntuación de valoración (búsqueda de contexto)

Búsquedas “booleanas” – Must, must not, should y filter

Búsqueda booleanas – MUST NOT

```
GET /productos2/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "name": "chocolate"
          }
        }
      ],
      "must_not": [
        {
          "match": {
            "name": "milk"
          }
        }
      ],
      "filter": [
        {
          "range": {
            "precio": {
              "lte": 150
            }
          }
        }
      ]
    }
  }
}
```

Condición que
NO se debe
cumplir

Búsqueda booleanas – SHOULD

```
GET /productos2/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "name": "chocolate"
          }
        }
      ],
      "must_not": [
        {
          "match": {
            "name": "milk"
          }
        }
      ],
      "should": [
        {
          "range": {
            "vendido": {
              "gte": 200
            }
          }
        }
      ]
    }
  }
}
```

Los resultados que
cumplan la condición de
“should” serán priorizados
con mayor peso

*Si no existe “must” o
“must_not”, es obligatorio
que al menos una condición
“should” se cumpla

Mejorar búsquedas con tratamiento de errores mediante "fuzziness"

¿Cómo podemos solventar un error tipográfico en la búsqueda del usuario?



Búsquedas "fuzziness"

"choc0late"

Búsqueda fuzziness – Reemplazo automático

```
GET /productos2/_search
{
  "query": {
    "match": {
      "name": {
        "query": "choc0late",
        "fuzziness": "auto"
      }
    }
  }
}
```

Añadimos en parámetros fuzziness a "auto" para solucionar errores.
El número de errores que solventa "auto" depende de la longitud de la palabra:
1-2 caracteres → 0 reemplazo
3-5 caracteres → 1 reemplazo
>5 caracteres → 2 reemplazos

Búsqueda fuzziness – Reemplazo específico

```
GET /productos2/_search
{
  "query": {
    "match": {
      "name": {
        "query": "chocloate",
        "fuzziness": "1"
      }
    }
  }
}
```

Fuzziness también corrige trasposiciones de caracteres y podemos indicar el número de cambios en lugar de dejarlo a "auto"

Búsquedas aplicando stemming y sinónimos

- La derivación regresiva (**stemming**) se utilizar para encontrar la palabra raíz eliminando afijos (prefijos o sufijos).

Cantando



Cant

Cantante

Derivación Regresiva Cant

- Optimiza los resultados de búsqueda sin necesidad de coincidencia exacta.
- Se pueden declarar **sinónimos** a buscar que permiten obtener resultados relevantes.

PASO 1: CREAR FILTROS PERSONALIZADOS

```
PUT /stemming_esp_
{
  "settings": {
    "analysis": {
      "filter": {
        "test_sinonimo": {
          "type": "synonym",
          "synonyms": [
            "amar, disfrutar"
          ]
        },
        "test_stemming": {
          "type": "stemmer",
          "name": "spanish"
        }
      }
    }
  }
}
```

PASO 2: CREAR ANALIZADOR CON FILTROS PERSONALIZADOS

```
"analyzer": {
  "mi_analizador": {
    "tokenizer": "standard",
    "filter": [
      "lowercase",
      "test_sinonimo",
      "test_stemming"
    ]
  }
}
```

PASO 3: APLICAR MAPEO AL CAMPO CON EL ANALIZADOR CREADO

```
"mappings": {
  "properties": {
    "descripcion": {
      "type": "text",
      "analyzer": "mi_analizador"
    }
  }
}
```



BLOQUE 5:
ELASTICSEARCH –
CONSULTAS PARA RELACIONES ENTRE
DOCUMENTOS

Lo que aprenderemos en este bloque....

- ✓ ¿En qué se diferencia una BBDD relacional y Elasticsearch?
- ✓ ¿Cómo mapear las relaciones entre documentos?
- ✓ ¿Cómo buscar documentos en base a las relaciones?
- ✓ ¿Cómo crear relaciones multinivel y buscar en base a este modelo?
- ✓ ¿Cómo controlamos los resultados de búsqueda?

¿En qué se diferencia una BBDD relacional y Elasticsearch?

BBDD Relacional

Elasticsearch

TABLA DE VENTAS

MODELO DE DATOS

TABLA DE EMPLEADOS

ID_Producto	ID_Cliente	ID_Empleado	Fecha compra	Estado Pedido
1	C2	E1	05/06/2021	Enviado
2	C3	E1	05/06/2021	Entregado
5	C3	E3	07/06/2021	Entregado

ID_Empleado	Nombre	Puesto	Fecha incorporación	País
E1	JSD	Comercial Nivel 1	05/04/2018	Perú
E2	PED	Comercial Nivel 2	06/05/2020	México
E3	TEZ	Gerente comercial	01/02/2015	México

TABLA DE PRODUCTOS

ID_Producto	Tipo Producto	Color	Precio
1	Pantalón	Blanco	40
2	Pantalón	Amarillo	28
3	Camiseta	Azul	15
4	Camiseta	Blanco	22
5	Gorra	Verde	10

TABLA DE CLIENTES

ID_Cliente	Nombre	Dirección	País
C1	IPD	C/ Leonardo, 3	España
C2	BCB	Av. Libertad, 7	México
C3	JLT	C/ Augusto, 6	Venezuela

- Existen claves (IDs) en las tablas para poder relacionarlas entre ellas

```
{
  "Fecha compra": "05/06/2021",
  "Estado Pedido": "Enviado"
  "Empleado": {
    "Nombre": "JSD",
    "Puesto": "Comercial Nivel 1",
    "Fecha incorp": "05/04/2018",
    "País": "Perú"
  }
  .
  .
}
```

- Desnormalización de los datos
- Similar a BBDD NO SQL
- Mejor rendimiento a base de mayor espacio en disco.
- Si problemas de espacio, no usar Elasticsearch como el Data Store principal.

¿En qué se diferencia una BBDD relacional y Elasticsearch?

TECNOLOGÍA

Empleado 1 Empleado 2



```
PUT /departamento/_doc/1
{
  "nombre departamento": "Tecnología",
  "empleados": [
    {
      "name": "Eric Green",
      "age": 39,
      "gender": "M",
      "position": "Big Data Specialist"
    },
    {
      "name": "James Taylor",
      "age": 27,
      "gender": "M",
      "position": "Software Developer"
    }
  ]
}
```

Los empleados son guardados dentro de los documentos de cada departamento



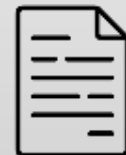
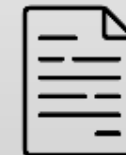
La actualización sería muy costosa



SOLUCIÓN: RELACIONAR DOCUMENTOS



Departamento Tecnología



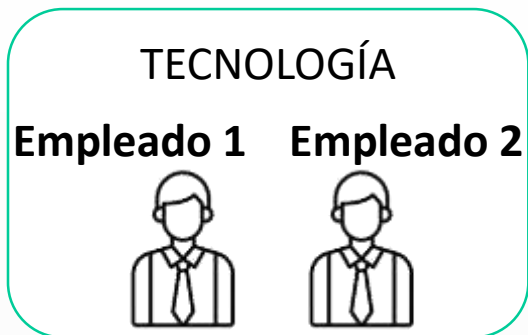
Empleado 1 Empleado 2

PARENT

CHILDREN



Mapear relaciones entre documentos y añadir documentos



Mapear relación entre documentos

```
PUT /departamento
PUT /departamento/_mapping
{
  "properties": {
    "campo combinación": {
      "type": "join",
      "relations": {
        "departamentos": "empleados"
      }
    }
  }
}
```

Definimos la relación



Crear Departamentos (PARENT)

```
PUT /departamento/_doc/1
{
  "nombre": "Tecnología",
  "campo_combinación": "departamentos"
}

PUT /departamento/_doc/2
{
  "nombre": "Marketing",
  "campo_combinación": "departamentos"
}
```

Definimos a que lado de la relación pertenece el documento

Definimos el lado de la relación y el parent ID

Crear empleados (CHILDREN)

```
PUT /departamento/_doc/3?routing=1
{
  "name_empleado": "Iván",
  "edad": 36,
  "género": "M",
  "campo_combinación": {
    "name": "empleados",
    "parent": 1
  }
}
```

```
PUT /departamento/_doc/5?routing=2
{
  "name_empleado": "Ana",
  "edad": 38,
  "género": "F",
  "campo_combinación": {
    "name": "empleados",
    "parent": 2
  }
}
```

Búsqueda de Children por Parent y viceversa

BUSCAR CHILD POR PARENT ID

```
GET /departamento/_search
{
  "query": {
    "parent_id": {
      "type": "empleados",
      "id": 2
    }
  }
}
```

Indicar lado de la relación a devolver y el ID del Parent

BUSCAR CHILD POR CONDICIÓN PARENT

```
GET /departamento/_search
{
  "query": {
    "has_parent": {
      "parent_type": "departamentos",
      "score": true,
      "query": {
        "term": {
          "nombre.keyword": "Tecnología"
        }
      }
    }
  }
}
```

Tipo consulta "has_parent"

Lado de la relación por la que buscar

Si queremos ver la puntuación de valoración de cada resultado (por ejemplo en búsquedas de contexto como match o bool)

Condición de búsqueda

Búsqueda de Children por Parent y viceversa

BUSCAR PARENT POR CONDICIÓN CHILD

```
GET /departamento/_search
{
  "query": {
    "has_child": {
      "type": "empleados",
      "score_mode": "sum",
      "min_children": 2,
      "max_children": 10,
      "query": {
        "bool": {
          "must": [
            {
              "range": {
                "edad": {
                  "gte": 37
                }
              }
            }
          ],
          "should": [
            {
              "term": {
                "género": "M"
              }
            }
          ]
        }
      }
    }
  }
}
```

Tipo consulta "has_child"

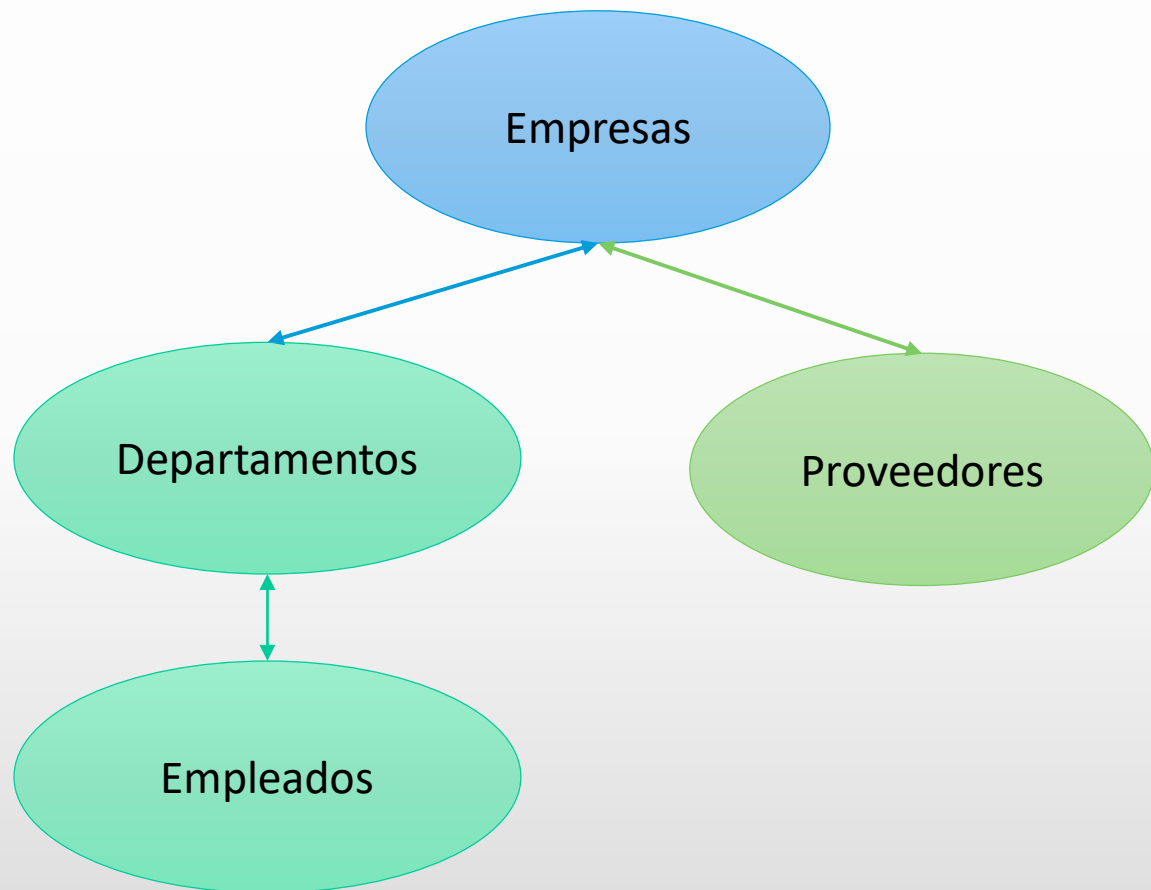
Lado de la relación por la que buscar

"score mode" para el modo de puntuación agregada de los childs (sum, min, max, avg)

"min_children" / "max_children" condición de cuántos children tiene que haber como mínimo y máximo para devolver el Parent

Condiciones de búsqueda

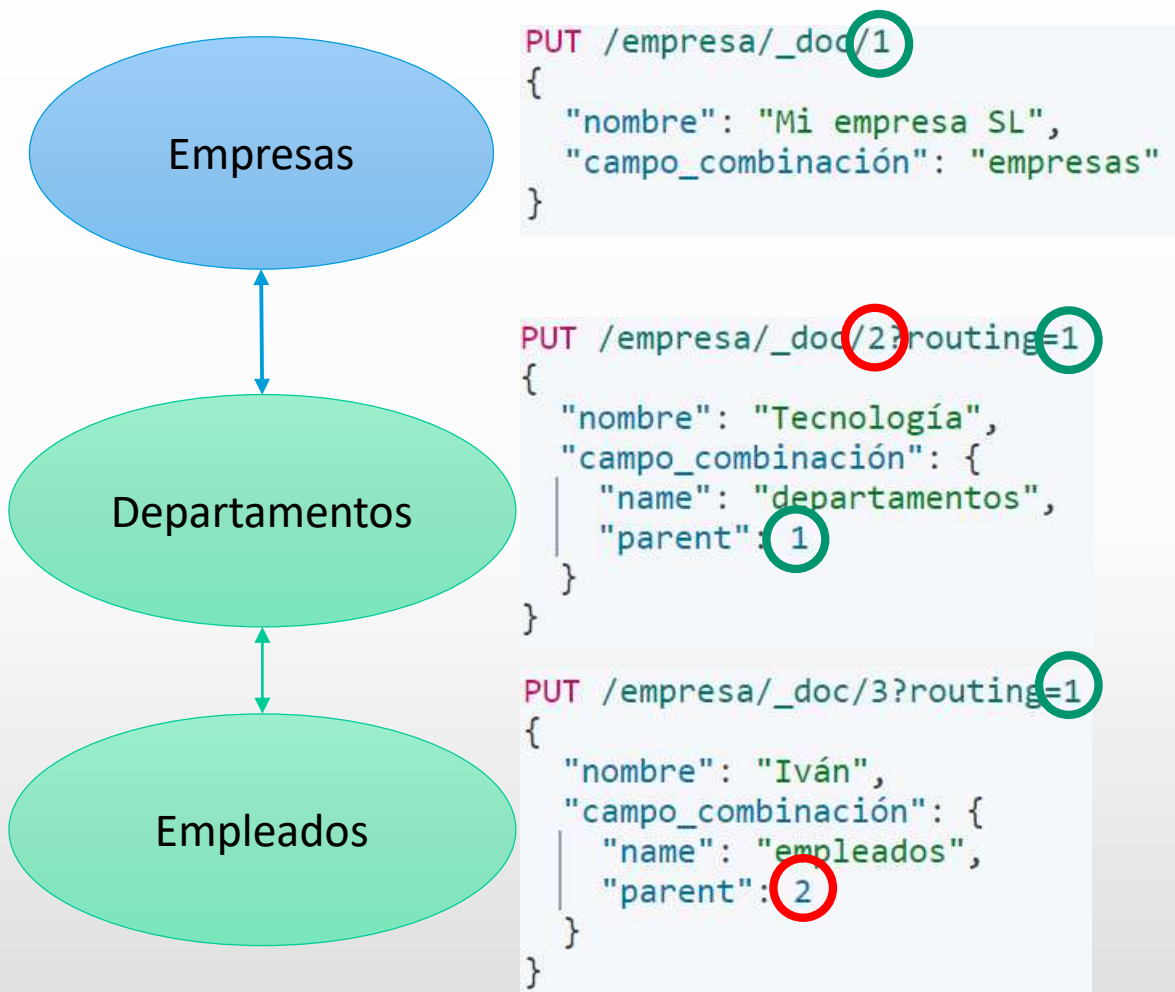
Relaciones multinivel



```
PUT /empresa
{
  "mappings": {
    "properties": {
      "campo_combinación": {
        "type": "join",
        "relations": {
          "empresas": ["departamentos", "proveedores"],
          "departamentos": "empleados"
        }
      }
    }
  }
}
```

- Definir relaciones de cada nivel individualmente
- Definir en un array si hay varias relaciones al mismo nivel

Relaciones multinivel



BUSQUEDA ANIDADA MULTINIVEL

```

GET /empresa/_search
{
  "query": {
    "has_child": {
      "type": "departamentos",
      "query": {
        "has_child": {
          "type": "empleados",
          "query": {
            "term": {
              "nombre.keyword": "Iván"
            }
          }
        }
      }
    }
  }
}

```

Múltiples "has_child"

- Las combinaciones son útiles para algunos escenarios, pero la filosofía de Elasticsearch es tener los **datos desnormalizados** para conseguir mayor rendimiento a diferencia de BBDD relacional (ejemplo, en cada documento de empleado añadir el departamento y la empresa asociada).

Control de resultados de búsqueda

LIMITAR TAMAÑO RESULTADOS

```
GET /productos3/_search
{
  "size": 5,
  "query": {
    "match_all": {
    }
  }
}
```

Parámetro "size" para especificar tamaño de visualización

FILTRAR CAMPOS DE RETORNO

```
GET /productos3/_search
{
  "_source": ["name", "stock"],
  "query": {
    "match_all": {
    }
  }
}
```

Campos a devolver

```
GET /productos3/_search
{
  "source": {
    "excludes": "descripcion"
  },
  "query": {
    "match_all": {
    }
  }
}
```

Campos a excluir

ORDENAR RESULTADOS

```
GET /productos3/_search
{
  "size": 5,
  "query": {
    "match_all": {
    }
  },
  "sort": [
    {"precio": "asc"},
    {"created": "desc"}
  ]
}
```

Especificar en objeto "sort" los campos por los que ordenar



elastic

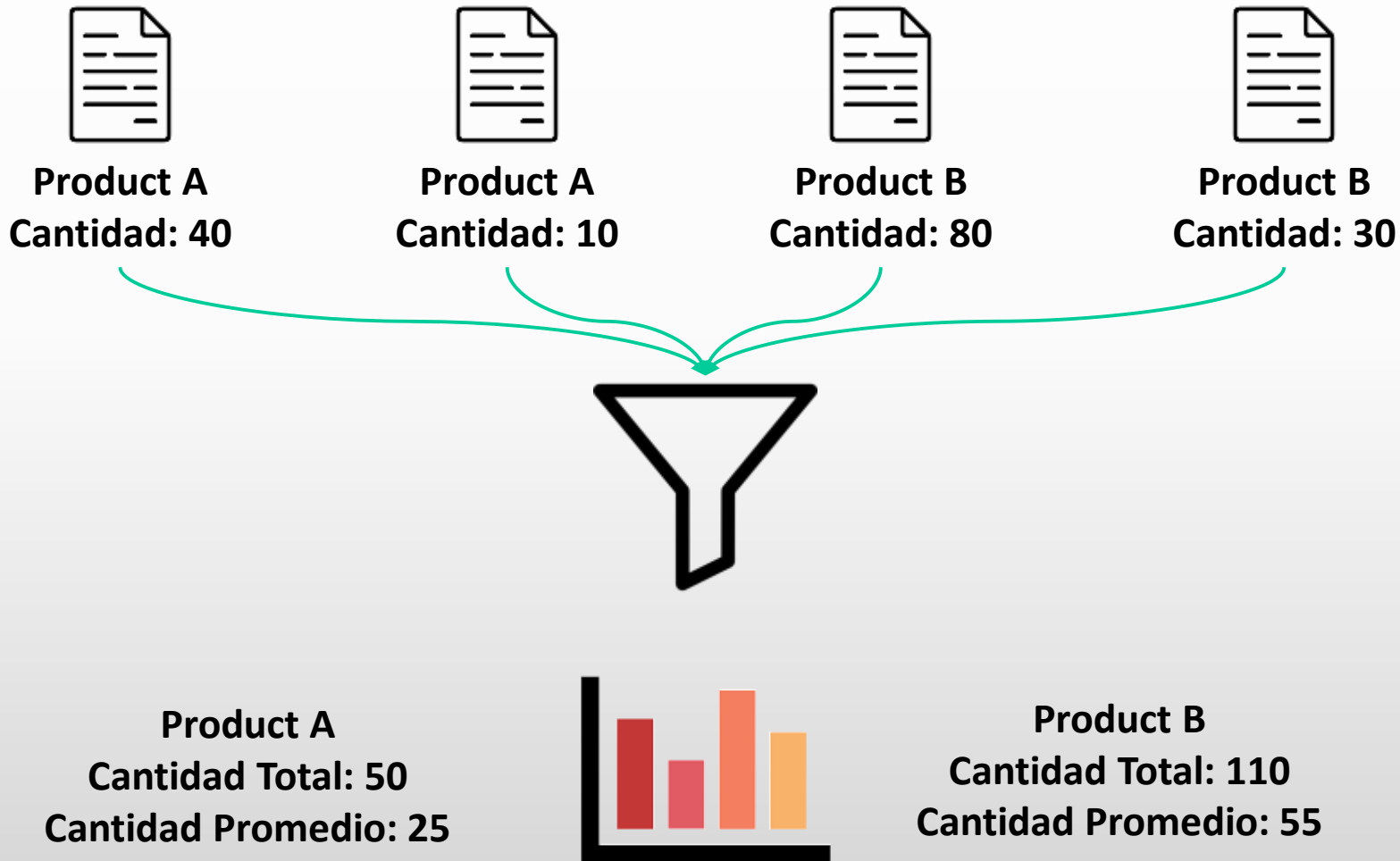
BLOQUE 6: ELASTICSEARCH – AGREGACIONES

Lo que aprenderemos en este bloque....

- ✓ ¿Para qué nos sirven las agregaciones y por qué son tan potentes?
- ✓ ¿Cómo crear agregaciones de tipo métrica y de tipo bucket?
- ✓ ¿Cómo realizar agregaciones combinadas?
- ✓ ¿Cómo realizar agregaciones por rangos de valores y fechas?
- ✓ ¿Cómo crear histogramas?

¿Qué son las agregaciones de tipo métrica?

- Las **agregaciones de tipo métrica** nos permiten realizar cálculos globales con nuestros datos para extraer conclusiones relevantes.



¿Qué son las agregaciones de tipo métrica?

¿Cómo realizamos una agregación?

```
GET /pedidos/_search
{
  "size": 0,
  "aggs": {
    "total_venta": {
      "sum": {
        "field": "total_importe"
      }
    }
  }
}
```

“size” es cero para indicar que es una agregación

Indicamos nombre del objeto donde nos va a devolver las agregaciones

Tipo de agregación (sum, avg, min, max,...) y campo a utilizar para la agregación

OBTENER ESTADÍSTICAS COMUNES CON “STATS”

```
GET /pedidos/_search
{
  "size": 0,
  "aggs": {
    "total_estadísticas": {
      "stats": {
        "field": "total_importe"
      }
    }
  }
}
```

Tipo de agregación “stats” devuelve suma, mínimo, máximo, promedio y contador de valores

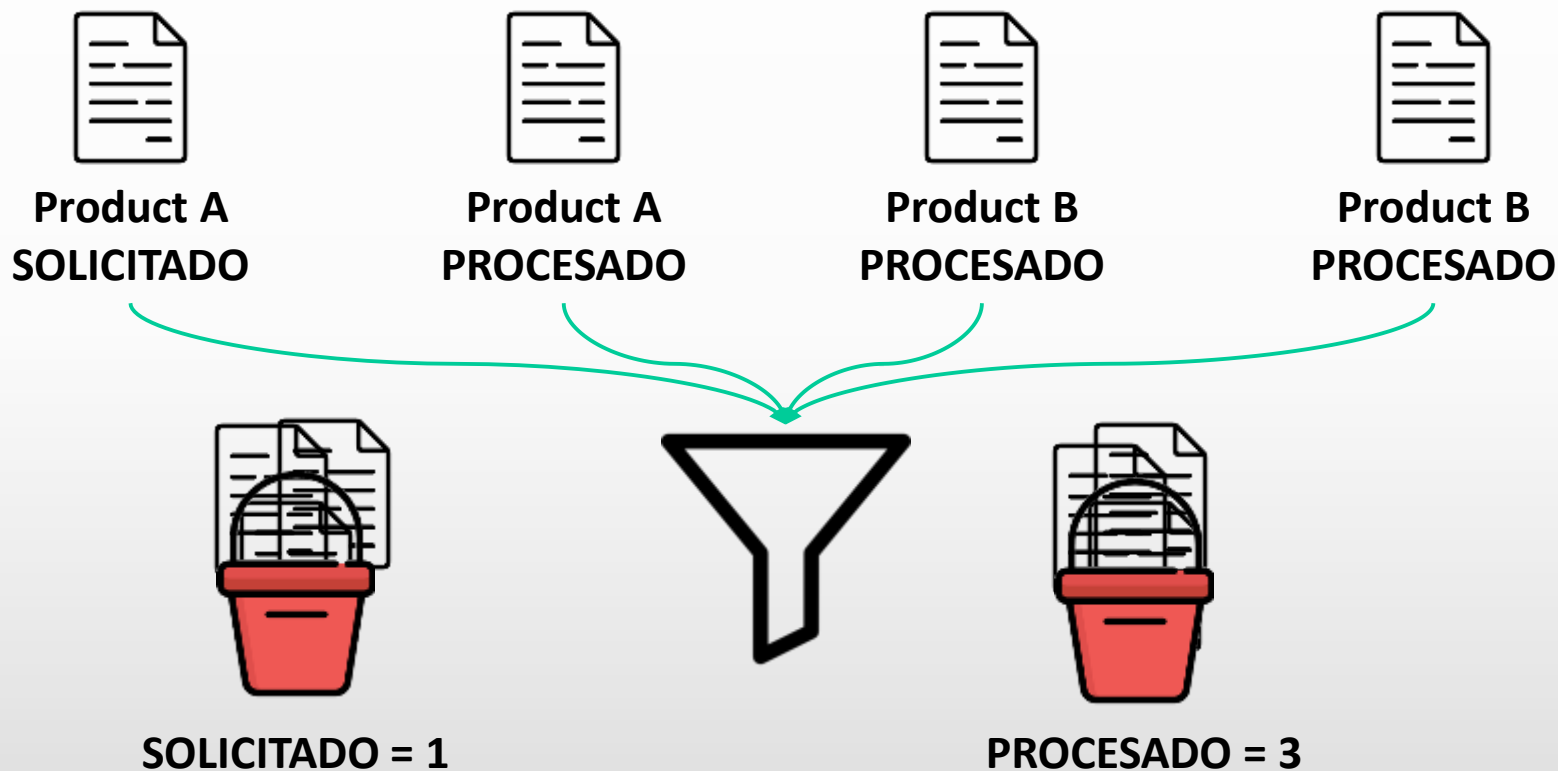
- Por defecto, las agregaciones hacen el cálculo considerando todos los documentos del index (como si usáramos una Query “match_all”), pero podríamos añadir una Query concreta sobre la que después aplicar las agregaciones.

Web referencia agregaciones tipo métrica:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-metrics.html>

¿Qué son las agregaciones de tipo “bucket”?

- Las **agregaciones de tipo “bucket”** nos permiten realizar contenedores de documentos (agrupaciones de documentos).



¿Qué son las agregaciones de tipo “bucket”?

Agregación bucket tipo “terms”

```
GET /pedidos/_search
{
  "size": 0,
  "aggs": {
    "bucket_estados": {
      "terms": {
        "field": "estado"
      }
    }
  }
}
```

Nombre del bucket

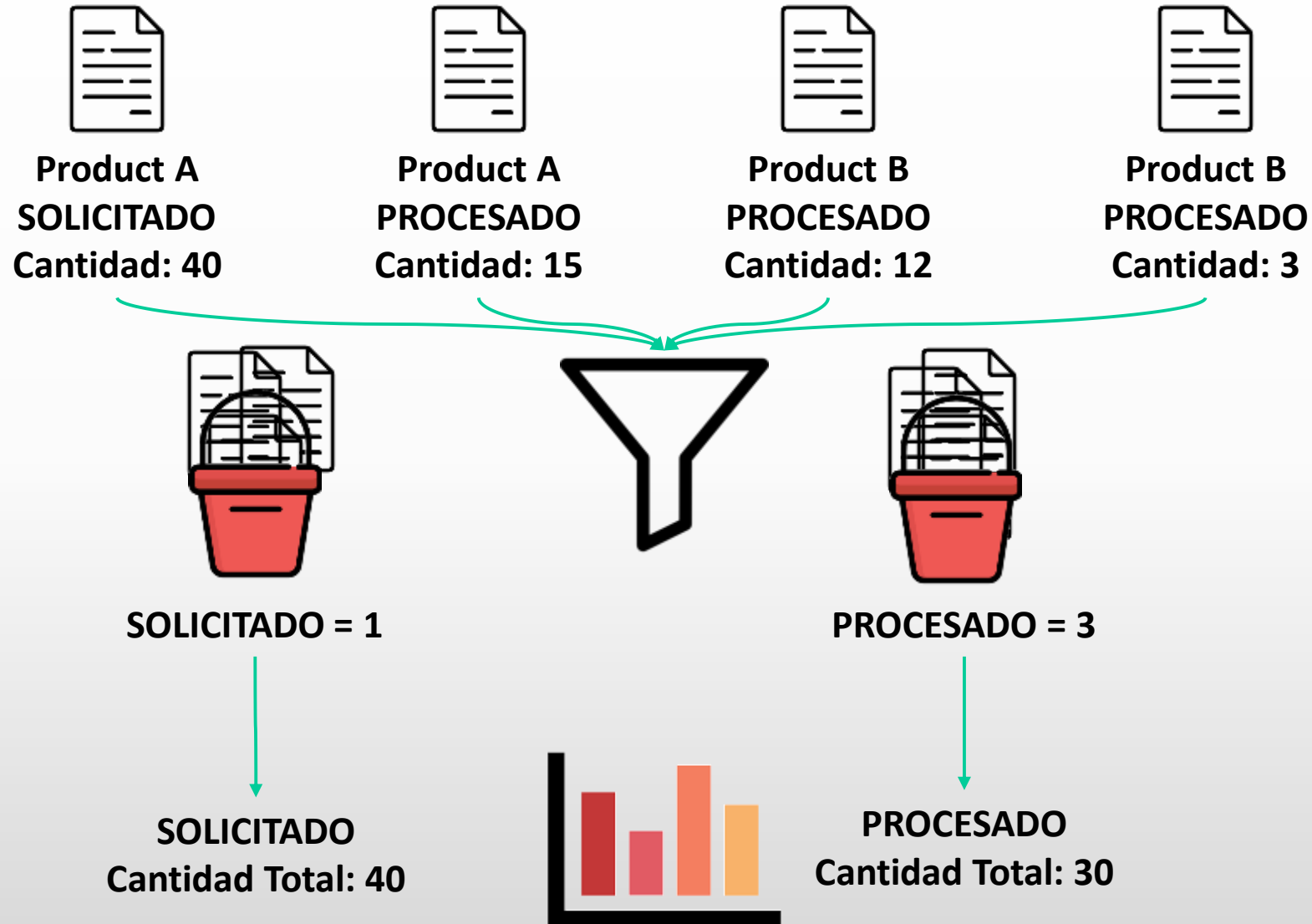
**Agregación bucket tipo
“terms” por cada
“estado” encontrado**

```
GET /pedidos/_search
{
  "size": 0,
  "aggs": {
    "bucket_estados": {
      "terms": {
        "field": "estado",
        "size": 20,
        "missing": "No aplica"
      }
    }
  }
}
```

**Número máximo de
buckets**

**Especificar bucket para
los documentos que
tengan el campo
“estado” vacío**

Agregaciones combinadas “nested”



Agregaciones combinadas (bucket + métricas)

Agregación combinada bucket + métrica

GET /pedidos/_search

```
{
  "size": 0,
  "aggs": {
    "bucket_estados": {
      "terms": {
        "field": "estado"
      }
    },
    "aggs": {
      "stats_estados": {
        "stats": {
          "field": "total_importe"
        }
      }
    }
  }
}
```

1. Para cada uno de los buckets de "estado"...

2. ...quiero obtener las estadísticas del total_importe

Agregación combinada bucket + métrica + query

GET /pedidos/_search

```
{
  "size": 0,
  "query": {
    "range": {
      "total_importe": {
        "gte": 100
      }
    }
  },
  "aggs": {
    "bucket_estados": {
      "terms": {
        "field": "estado"
      },
      "aggs": {
        "stats_estados": {
          "stats": {
            "field": "total_importe"
          }
        }
      }
    }
  }
}
```

3. Teniendo en cuenta solo los documentos en los que total_importe sea mayor que 100

Agregaciones con filtros y reglas

Crear buckets en base a filtrado y aplicar métricas

Aplicar un filtrado en la agregación

```
GET /pedidos/_search
{
  "size": 0,
  "aggs": {
    "valor_bajo": {
      "filter": {
        "range": {
          "total_importe": {
            "lte": 50
          }
        }
      }
    },
    "aggs": {
      "promedio_importe": {
        "avg": {
          "field": "total_importe"
        }
      }
    }
  }
}
```

**Especificar en
"filter" la condición
de filtrado**

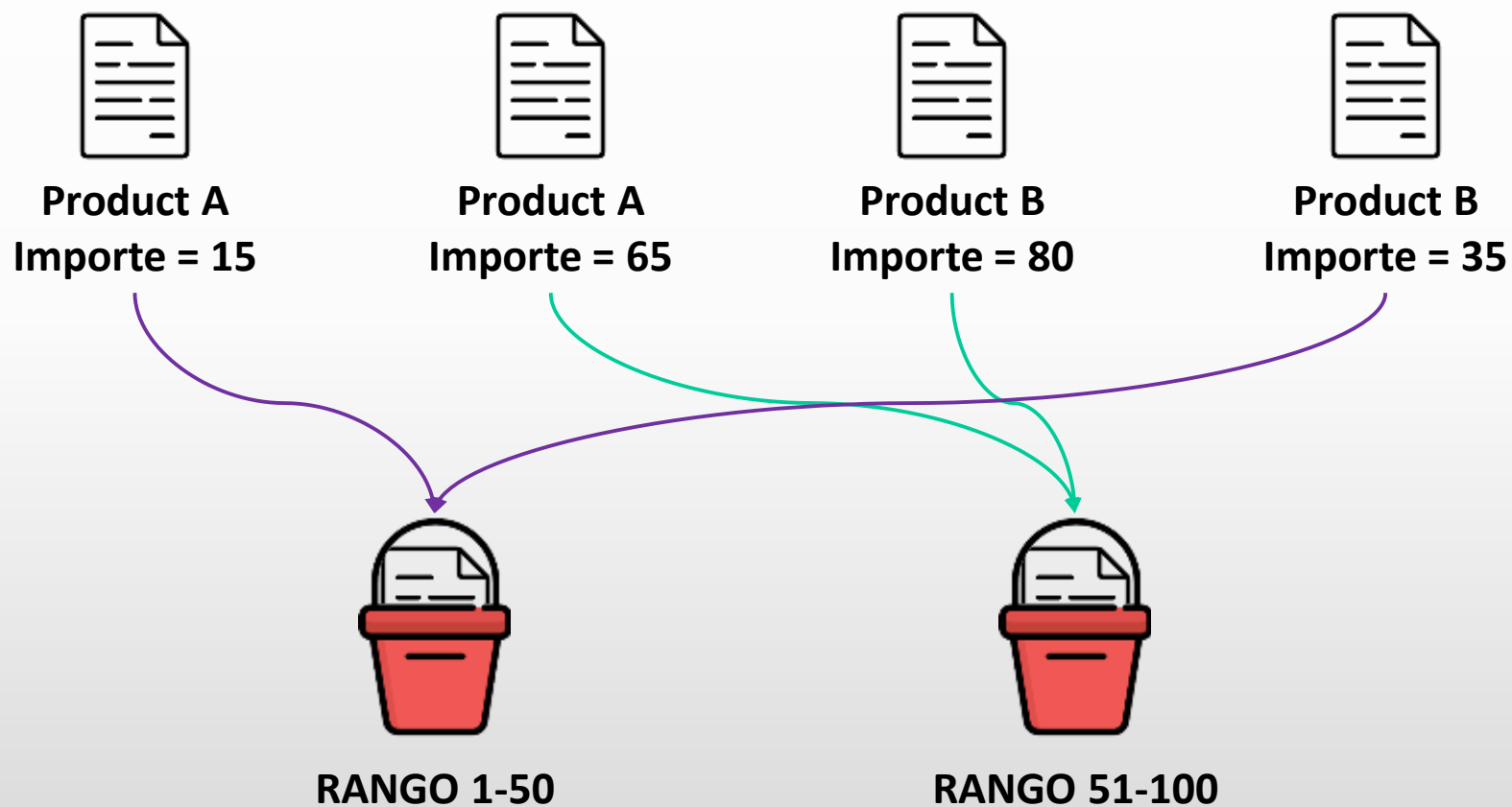
**Combinar con la
agregación de métrica
deseada**

```
GET /pedidos/_search
{
  "size": 0,
  "aggs": {
    "mi_filtro": {
      "filters": {
        "filters": {
          "PENDIENTE": {
            "match": {
              "estado": "pending"
            }
          },
          "CONFIRMADO": {
            "match": {
              "estado": "confirmed"
            }
          }
        }
      }
    },
    "aggs": {
      "promedio_importe": {
        "avg": {
          "field": "total_importe"
        }
      }
    }
  }
}
```

**Crear buckets con las
reglas de filtrado
personalizado:
Bucket 1: "pending"
Bucket 2: "confirmed"**

**Combinar con la
agregación de métrica
deseada**

Agregaciones con rangos de valores y fechas



Agregaciones con rangos de valores y fechas

Agregación por rangos de valores

```
GET /pedidos/_search
{
  "size": 0,
  "aggs": {
    "distribución importe": {
      "range": {
        "field": "total importe",
        "ranges": [
          {
            "to": 50
          },
          {
            "from": 50,
            "to": 100
          },
          {
            "from": 100
          }
        ]
      }
    }
  }
}
```

Especificar tipo
"range" y el campo

Rangos de valores

Agregación por rangos de fechas

```
GET /pedidos/_search
{
  "size": 0,
  "aggs": {
    "rangos_compra": {
      "date_range": {
        "field": "Fecha compra",
        "ranges": [
          {
            "from": "2016-01-01",
            "to": "2016-01-01|+6M"
          },
          {
            "from": "2016-01-01|+6M",
            "to": "2016-01-01|+1y"
          }
        ]
      }
    }
  }
}
```

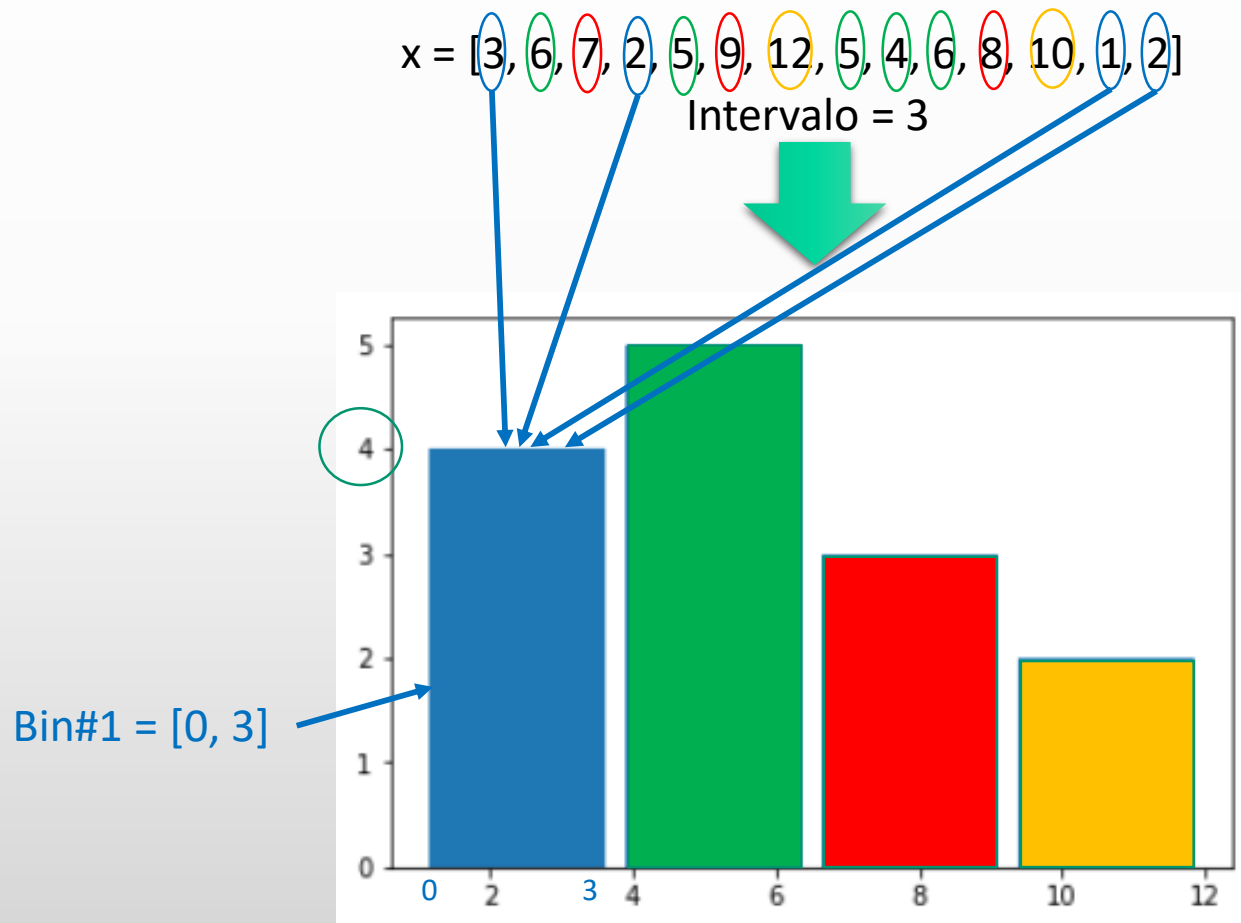
Especificar tipo
"date_range" y el campo

Rango de fechas

Histogramas

Histograma: Agrupación de los valores de una variables en contenedores (**intervalos** - **bins**). Proporciona la distribución de un campo a lo largo de los intervalos

x: Campo número de pedidos de mis clientes



Histogramas

Creación de histograma – intervalo valores

```
GET /pedidos/_search
```

```
{
  "size": 0,
  "aggs": {
    "distribución_importe": {
      "histogram": {
        "field": "total_importe",
        "interval": 25
      }
    }
  }
}
```

Especificar tipo
"histogram" y el
campo

Indicar el intervalo de
valores

Creación de histograma – intervalo fechas

```
GET /pedidos/_search
```

```
{
  "size": 0,
  "aggs": {
    "agregación_pedidos_mensual": {
      "date_histogram": {
        "field": "Fecha_compra",
        "calendar_interval": "month"
      }
    }
  }
}
```

Especificar tipo
"date_histogram" y el
campo

Indicar intervalo de
fechas



elastic

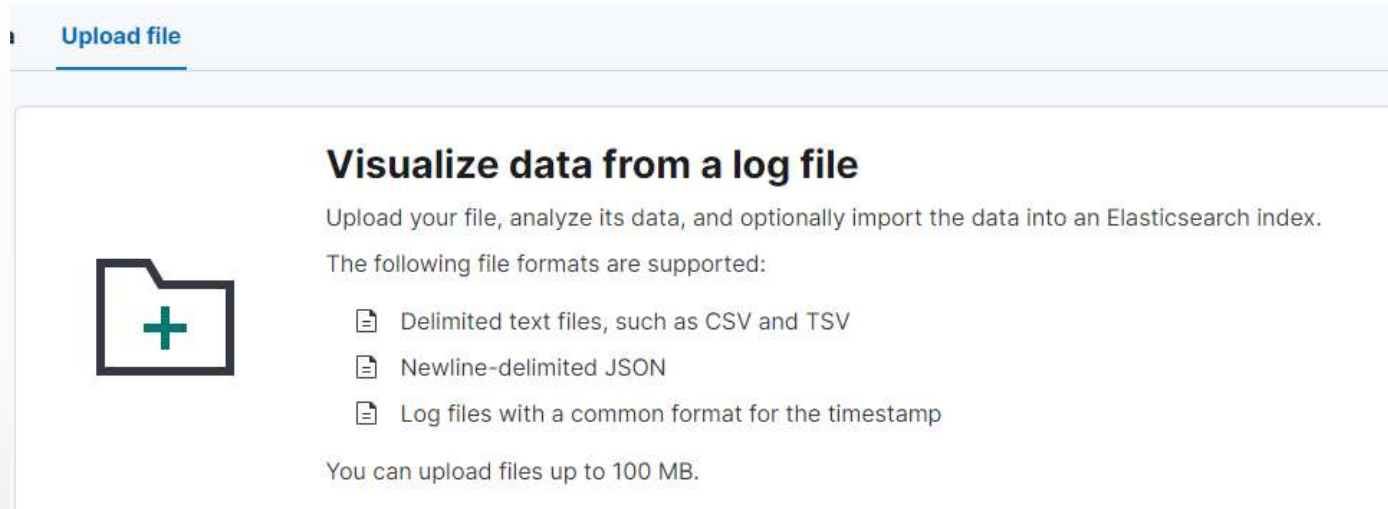
BLOQUE 7:
LOGSTASH –
INGESTA, TRANSFORMACIÓN Y SALIDA

Lo que aprenderemos en este bloque....

- ✓ ¿Cómo podemos ingestar datos en Elasticsearch?
- ✓ ¿Cómo instalar Logstash?
- ✓ ¿Cómo crear pipelines de Logstash?
- ✓ ¿Cómo ejecutar los pipelines y realizar cargas en Elasticsearch?
- ✓ ¿Qué otros métodos de ingesta, transformación y carga podemos utilizar?

¿Cómo ingestamos datos en Elasticsearch?

- **MÉTODO 1:** Desde Elastic Cloud (CSV, TSV, NDJSON, log files,...)



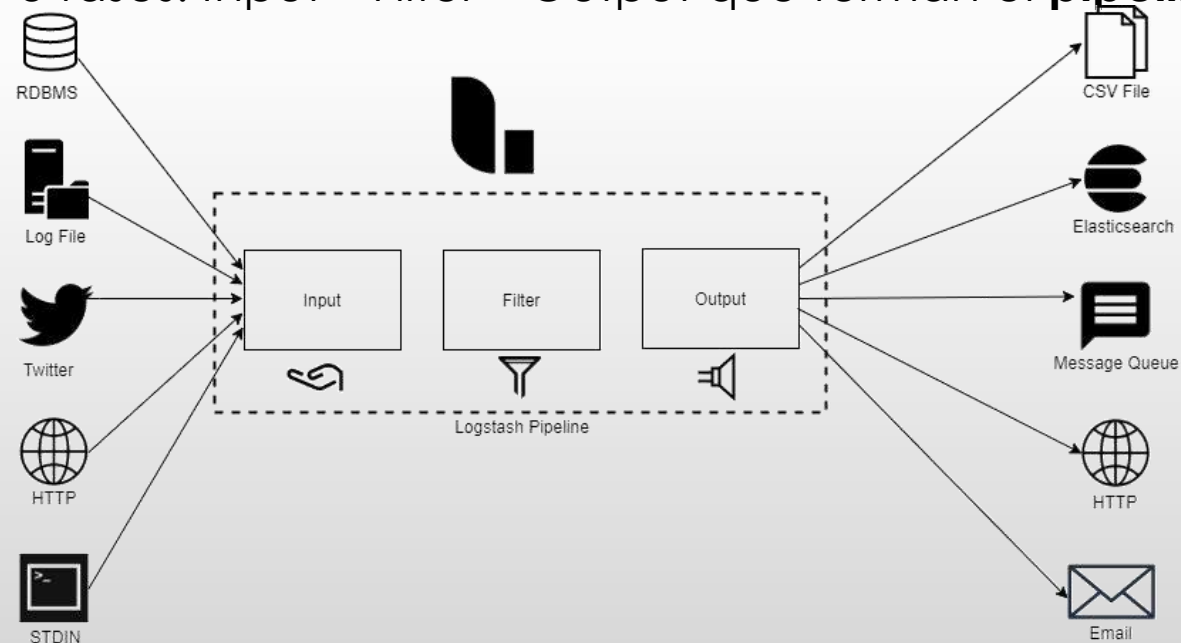
- **MÉTODO 2:** Importación masiva mediante cURL o aplicaciones web:

```
1960 {"name":"Pork - loin Bone - In","price":120,"in_stock":44,"sold":439,"tags":["Meat"],"description":"P
1961 {"index":{"_id":1011}}
1970 {"name":"Cheese and pumpkin risotto","price":49,"in_stock":17,"sold":94,"tags":[],"description":"Crea
1971 {"index":{"_id":1011}}
1972 {"name":"Mapkin - Beverage White 2 - Fly","price":118,"in_stock":61,"sold":145,"tags":[],"description"
1973 {"index":{"_id":1017}}
1974 {"name":"Lid Tray - 16in Dome","price":118,"in_stock":97,"sold":687,"tags":[],"description":"Nulla m
1975 {"index":{"_id":1017}}
1976 {"name":"Sage - fresh","price":141,"in_stock":44,"sold":224,"tags":["Spice"],"description":"Integer a
1977 {"index":{"_id":1017}}
1978 {"name":"Pepper - Yellow Bell","price":193,"in_stock":9,"sold":221,"tags":["Fruit"],"description":"Se
1979 {"index":{"_id":1017}}
1980 {"name":"Corn - Mini","price":90,"in_stock":11,"sold":240,"tags":["Vegetable"],"description":"Eclia v
1981 {"index":{"_id":1017}}
1982 {"name":"VoQuart - Peach 175 Gr","price":119,"in_stock":1,"sold":189,"tags":[],"description":"Cras mi
1983 {"index":{"_id":1017}}
1984 {"name":"Wine - Alsace Riesling Reserve","price":10,"in_stock":17,"sold":116,"tags":["Beverage"],"Alco
1985 {"index":{"_id":1017}}
1986 {"name":"Sauce - Black Currant Dry Mix","price":179,"in_stock":11,"sold":479,"tags":[],"description":
1987 {"index":{"_id":1017}}
1988 {"name":"Lamb - loin Trimmed Boneless","price":170,"in_stock":12,"sold":487,"tags":["Meat"],"descripti
1989 {"index":{"_id":1017}}
1990 {"name":"Toothpick Filled","price":181,"in_stock":20,"sold":43,"tags":[],"description":"Integer ac n
1991 {"index":{"_id":1017}}
```

¿Cómo ingestamos datos en Elasticsearch?

- **MÉTODO 3:** Utilizar Logstash desde la máquina donde residen los datos:

- Logstash es un pipeline de procesamiento de datos.
- Recibe datos que son manejados como **eventos**, estos datos pueden ser de cualquier tipo de fuente.
- Tiene 3 fases: Input – Filter – Output que forman el **pipeline**:

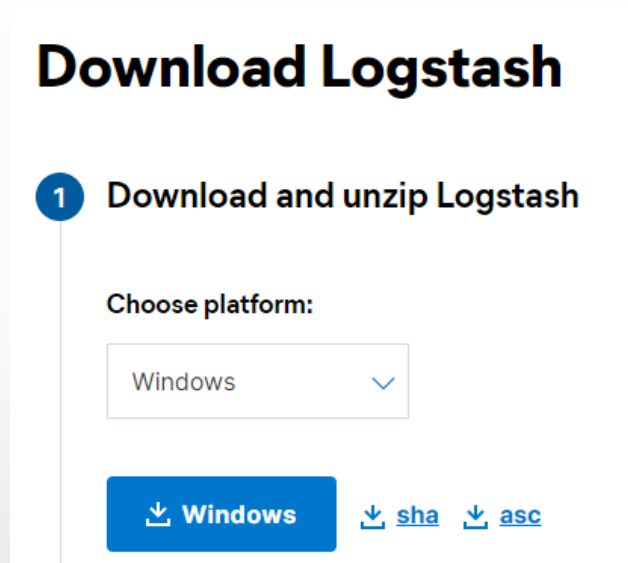


- Existen **plugins** para cada una de las 3 fases.

Instalación de Logstash

1) Descargar Logstash:

<https://www.elastic.co/es/downloads/logstash>



2) Descomprimir carpeta Logstash

- Carpeta "bin": logstash.bat (Windows) / logstash (Linux)

Creación de un pipeline (input, filter y output)

1) Crear **Input** del pipeline:

```
input{
  file{
    path => "C:/Users/ivan_pinar/Downloads/Info_pais.csv"
    start_position => beginning
    sincedb_path => "NULL"
  }
}
```

En path indicamos la ruta del fichero

2) Aplicar **Filtros** en el pipeline:

```
filter {
  csv{
    separator => ";"
    columns => ["Pais", "Poblacion", "Renta per capita", "Esperanza de vida"]
  }
}
```

Renombrado de columnas (se pueden aplicar múltiples pasos de transformación)

3) Asignar la **Salida** (output) del pipeline:

- Utilizar el Cloud ID (cloud_id => "tu_cloud_id")
- O bien la dirección local de Elasticsearch (hosts => "http\\localhost:9200").

```
output {
  stdout{}
  elasticsearch{
    cloud id => "0cee7a2e06ea467"
    user => logstash_internal
    password => logstash
    index => "países_test"
  }
}
```

Index donde cargar los datos

Usuario creado en Elastic cloud con los permisos necesarios

Web referencia documentación Logstash:

<https://www.elastic.co/guide/en/logstash/current/index.html>

Ejecución del pipeline y carga en Elasticsearch

1) Definir el index en Elasticsearch (se puede aplicar un mapping previo al index)

```
PUT /países_test
{
  "mappings": {
    "properties": {
      "Esperanza de vida": {
        "type": "keyword"
      },
      "Pais": {
        "type": "keyword"
      },
      "Poblacion": {
        "type": "long"
      },
      "Renta per capita": {
        "type": "long"
      }
    }
  }
}
```

2) Definir rol para permitir la escritura en el index

3) Crear usuario en Elastic Cloud con ese rol

```
POST /_security/role/logstash_escritura
{
  "run_as": [ "logstash_escritura" ],
  "cluster": [ "monitor" ],
  "indices": [
    {
      "names": [ "países_test" ],
      "privileges": [ "write", "all" ]
    }
  ]
}
```



4) Ejecutar logstash con el pipeline creado

```
>C:\Users\ivan pinar\Downloads\logstash-8.6.2-windows-x86_64\logstash-8.6.2\bin\logstash.bat -f C:\Users\ivan pinar\Downloads\pipeline simp.conf
```

Ruta completa logstash

Ruta completa pipeline "conf"

5) Indexar en Kibana para poder visualizar los datos posteriormente (Management / Stack Management):



Otros métodos de ingesta, transformación y carga

¿Qué otras fuentes de datos se pueden usar?

```
input {
  sqlite {
    path => "/tmp/example.db"
    type => weblogs
  }
}
```



Beats

```
input {
  beats {
    port => 5044
  }
}
```

```
input {
  s3 {
    access_key_id => "1234"
    secret_access_key => "secret"
    bucket => "logstash-test"
    additional_settings => {
      force_path_style => true
      follow_redirects => false
    }
  }
}
```



Amazon S3

¿Qué transformaciones se pueden aplicar?

```
filter {
  mutate {
    split => { "hostname" => "." }
    add_field => { "shortHostname" => "%{[hostname][0]}" }
  }

  mutate {
    rename => { "shortHostname" => "hostname" }
  }
}
```

```
filter {
  grok {
    match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request}" }
  }
}
```

¿Hacia dónde puede enviar los datos además de Elasticsearch?



```
output {
  kafka {
    codec => json
    topic_id => "mytopic"
  }
}
```

```
output {
  csv {
```



```
output {
  if "shouldmail" in [tags] {
    email {
      to => 'technical@example.com'
      from => 'monitor@example.com'
      subject => 'Alert - %{title}'
      body => "Tags: %{tags}\\n\\Content:\\n\\n%{message}"
      template_file => "/tmp/email_template.mustache"
      domain => 'mail.example.com'
      port => 25
    }
  }
}
```

Web referencia documentación Logstash:

<https://www.elastic.co/guide/en/logstash/current/index.html>



elastic

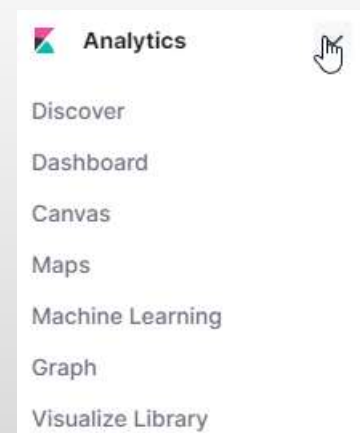
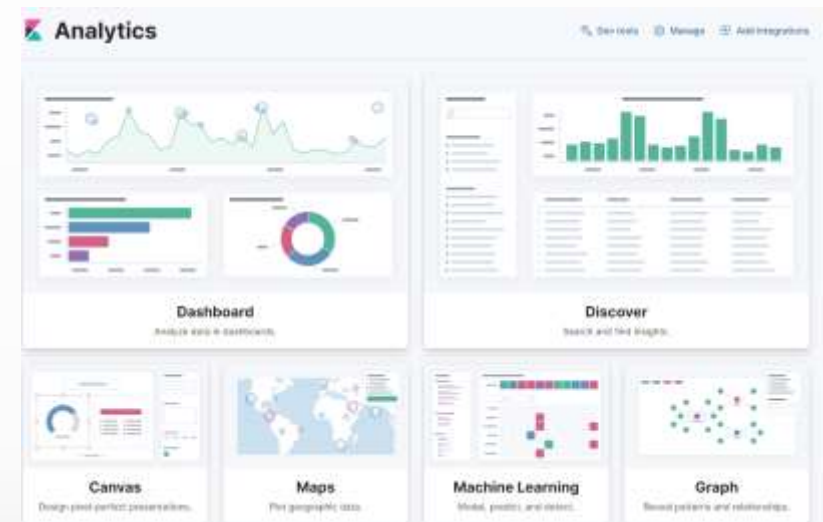
BLOQUE 8:
KIBANA –
INTERFAZ, INGESTA Y VISUALIZACIONES

Lo que aprenderemos en este bloque....

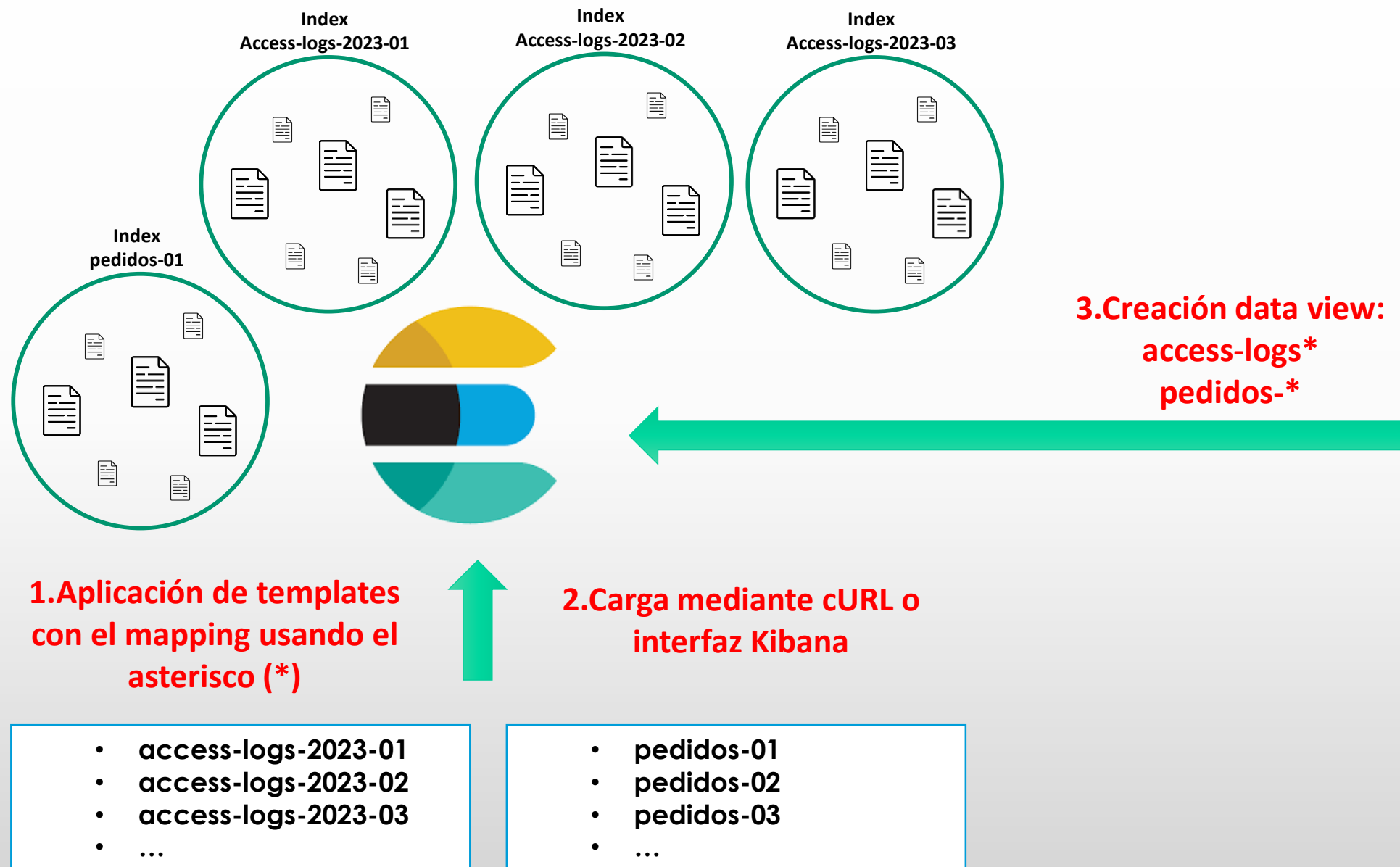
- ✓ ¿Qué nos proporciona Kibana y cuáles son sus componentes?
- ✓ ¿Cómo creamos data views con la ingesta de datos?
- ✓ ¿Qué nos permite analizar la herramienta Discover?
- ✓ ¿Cómo crear multitud de tipos de visualización como área, línea, barras, circular, histogramas, heatmap, KPI, mapas geográficos,..?

¿Qué nos proporciona Kibana y cuáles son sus componentes?

- 1) **Discover:** Analizar datos mediante consultas KQL filtros.
- 2) **Dashboard:** Creación de dashboards (combinaciones de visualizaciones) para presentación de datos.
- 3) **Canvas:** Presentación de visualizaciones más avanzada y personalizada (incluso CSS).
- 4) **Maps:** Creación de mapas geográficos avanzada con diferentes capa.
- 5) **Machine learning:** Detección de anomalías, forecasting,...
- 6) **Graph:** Visualizar conexiones entre elementos.
- 7) **Librería visualizaciones:** Creación de las visualizaciones.



Ingesta de datos y creación de data views



Menú Discover – Paneles y lenguaje KQL

- 1) Modificación filtro temporal
- 2) Vista general paneles Discover
- 3) Sintaxis KQL: Lenguaje abreviado que se transforma automáticamente a consulta QueryDSL de Elasticsearch.

COMPARADORES KQL

<=>	:	equals	some value
<=>	<=	is less than or equal to	some value
<=>	>=	is greater than or equal to	some value
<=>	<	is less than	some value
<=>	>	is greater than	some value
<=>	:	*	exists in any form

OPERADORES BOOLEANOS

	and	Requires both arguments to be true
	or	Requires one or more arguments to be true

- 4) Guardar consultas:



- 5) Inspeccionar consulta y respuesta

Web referencia documentación KQL:

<https://www.elastic.co/guide/en/kibana/current/query-query.html>

Visualización tipo métrica

- 1) Pulsamos en Analytics > Visualize Library > Create > Metric
- 2) Elegimos el Data View
- 3) Configuramos la métrica:



Pedidos

Data Options

Metrics

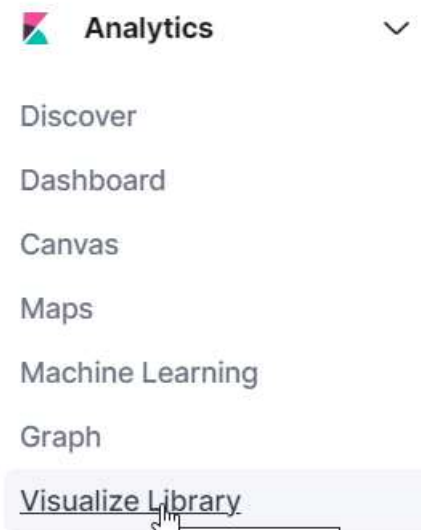
▼ Metric

Aggregation [Average help](#)

Average ▼

Field

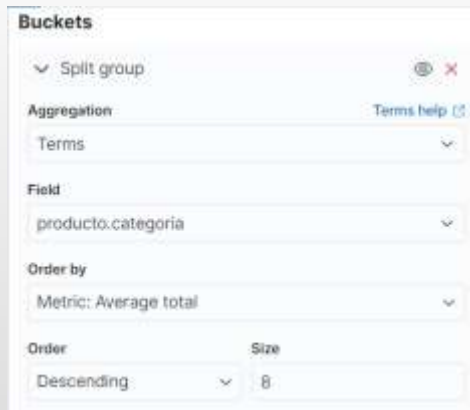
total ▼



8 **Metric**

Show a calculation as a single number

- 4) Dividimos en buckets si es preciso:



Buckets

▼ Split group

Aggregation [Terms help](#)

Terms ▼

Field

producto.categoria ▼

Order by

Metric: Average total ▼

Order

Descending ▼

Size

8

- 5) Guardamos la visualización:



- 6) Modificamos el formato del campo en el data view > Set format

Visualización tipo barra, área y línea

1) Elegimos el tipo de visualización agregación:



Vertical bar



Present data in vertical bars on an axis.



Area

Emphasize the data between an axis and a line.



Line

Display data as a series of points.

2) Elegimos el Data View

3) Configuramos el bucket “x-axis” para la escala temporal:

▼ X-axis ⓘ = ✕

Aggregation [Date Histogram help](#) ⓘ

Date Histogram ▼

Field

@timestamp ▼

Minimum interval

Auto ⓘ ▼

Select an option or create a custom value.
Examples: 30s, 20m, 24h, 2d, 1w, 1M

4) Configuramos un bucket subagregación Split-series:

▼ Split series ⓘ = ✕

Sub aggregation [Terms help](#) ⓘ

Terms ▼

Field

http.response.status_code ▼

Order by

Metric: Count ▼

Order

Descending ▼

Size

5

5) Bucket para dividir el gráfico con Split-chart y subir a la parte alta:

▼ Split chart ⓘ = ✕

Rows Columns

Aggregation [Terms help](#) ⓘ

Terms ▼

Field

url.path ▼

Order by

Metric: Count ▼

6) Modificación “Metrics & axes” y se puede modificar entre tipos de gráfico.

Data Metrics & axes Panel settings

Metrics

▼ Count

Value axis

LeftAxis-1 ▼

Chart type Mode

Area ▼ Stacked ▼

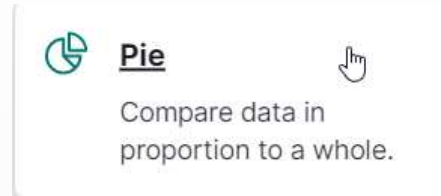
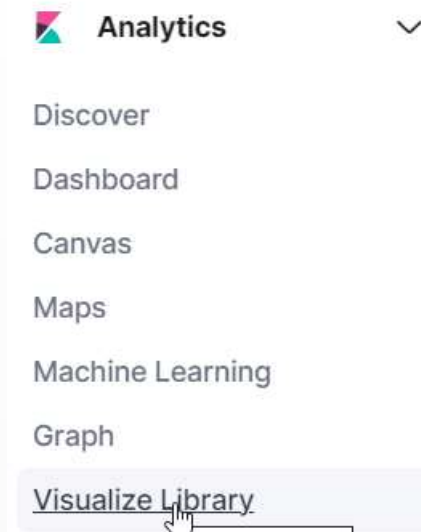
Line mode

Straight ▼

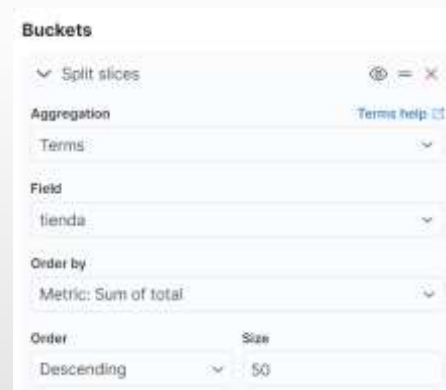
Show dots

Visualización tipo circular

- 1) Pulsamos en Analytics > Visualize Library > Create
- 2) Elegimos el Data View



- 3) Dividimos en buckets Split slices:

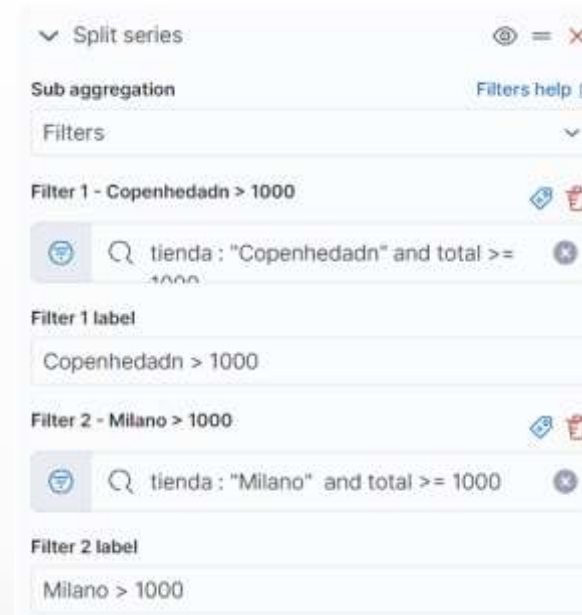


- 4) Agregamos nuevos buckets para sub-slices:



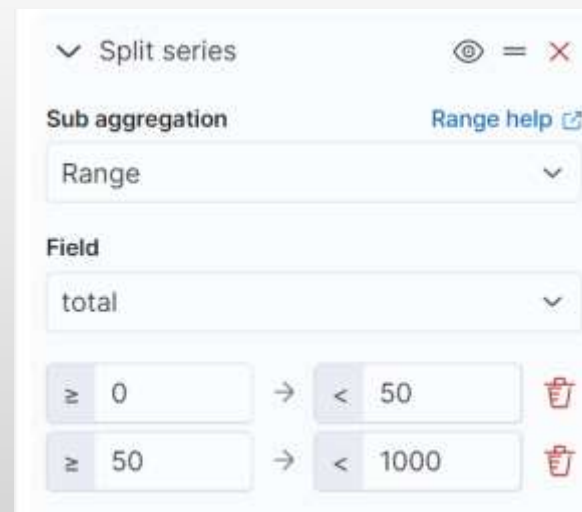
Split de series con filtros KQL y Ranges

1) Dividimos en buckets "Split series" de tipo "Filters":



The screenshot shows the 'Split series' configuration interface. At the top, there's a dropdown menu set to 'Filters' with a 'Filters help' link. Below this, two filter rules are defined. 'Filter 1 - Copenhedadn > 1000' has a KQL query 'tienda : "Copenhedadn" and total >= 1000' and a label 'Copenhedadn > 1000'. 'Filter 2 - Milano > 1000' has a KQL query 'tienda : "Milano" and total >= 1000' and a label 'Milano > 1000'. Each filter rule includes a search icon, a KQL query input, a plus icon to add more filters, and a trash icon to delete the filter.

2) Dividimos buckets "Split series" con "Ranges"



The screenshot shows the 'Split series' configuration interface. At the top, there's a dropdown menu set to 'Range' with a 'Range help' link. Below this, the 'Field' dropdown is set to 'total'. Two range rules are defined. The first rule is '≥ 0' to '< 50' with a trash icon. The second rule is '≥ 50' to '< 1000' with a trash icon. Each range rule includes a greater-than-or-equal-to icon, a value input, a right arrow, a less-than icon, another value input, and a trash icon.

Visualización tipo histogramas

- 1) Los buckets serán de agregación tipo “Histogram” e indicaremos el ancho del intervalo:

Buckets

▼ X-axis

Aggregation

Histogram help [Histogram help](#)

Histogram ▼

Field

producto.precio ▼

- 2) Podemos configurar el eje y para que en lugar de la cuenta nos proporcione otras agregaciones:

Metrics

▼ Y-axis

Aggregation

Sum help [Sum help](#)

Sum ▼



Field


total ▼

Visualización tipo tabla

1) Configuramos los buckets para dividir en filas:

Buckets

▼ Split rows  

Aggregation [Terms help](#) 

Terms ▼

Field

vendedor.id ▼

Order by

Custom metric ▼

2) Añadimos las métricas con las columnas de interés:

Metrics

> Metric Count  = 

▼ Metric  = 

Aggregation [Sum help](#) 

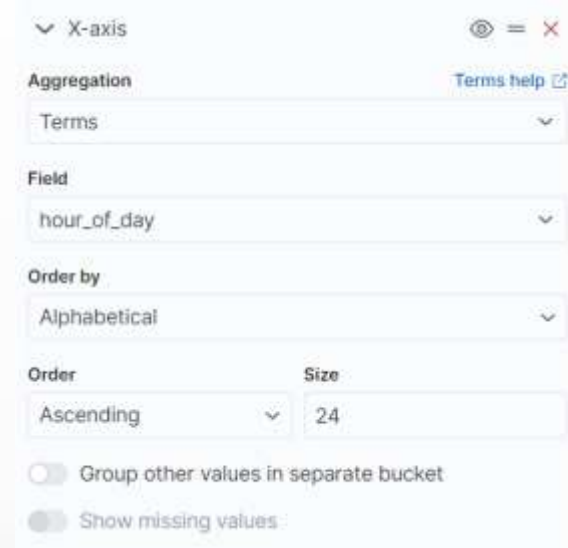
Sum ▼

Field

total ▼

Visualización tipo heatmap

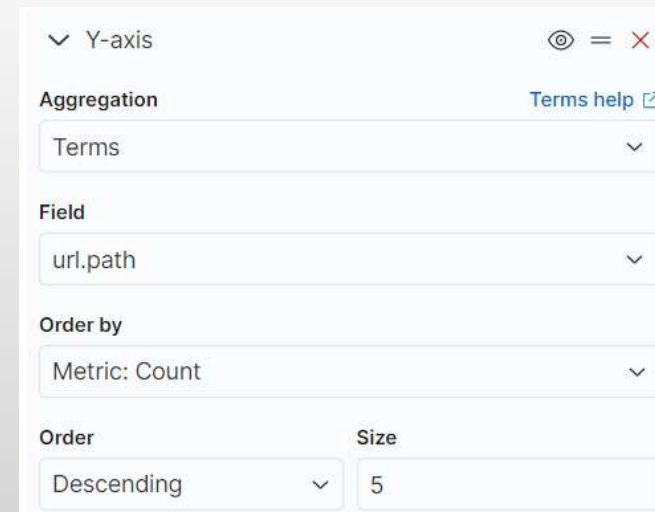
1) Configuramos los buckets para el x-axis:



Configuration for the X-axis:

- Aggregation: Terms
- Field: hour_of_day
- Order by: Alphabetical
- Order: Ascending
- Size: 24
- ☐ Group other values in separate bucket
- ☐ Show missing values

2) Configuramos los buckets para el y-axis:



Configuration for the Y-axis:

- Aggregation: Terms
- Field: url.path
- Order by: Metric: Count
- Order: Descending
- Size: 5

Visualización tipo KPI objetivo

1) Definir la métrica KPI:

Metrics

▼ Metric

Aggregation [Sum help](#)

Sum ▼

Field

total ▼

Custom label

2) Personalizar la visualización KPI:

Style

Gauge type

Arc ▼

Alignment

▼

Ranges

≥ 0	→	< 2000000	🗑
≥ 2000000	→	< 4000000	🗑
≥ 4000000	→	< 6000000	🗑

3) Definir buckets de división:

Buckets

▼ Split group [🗑](#) [×](#)

Aggregation [Date Histogram help](#)

Date Histogram ▼

Field

@timestamp ▼

Minimum interval

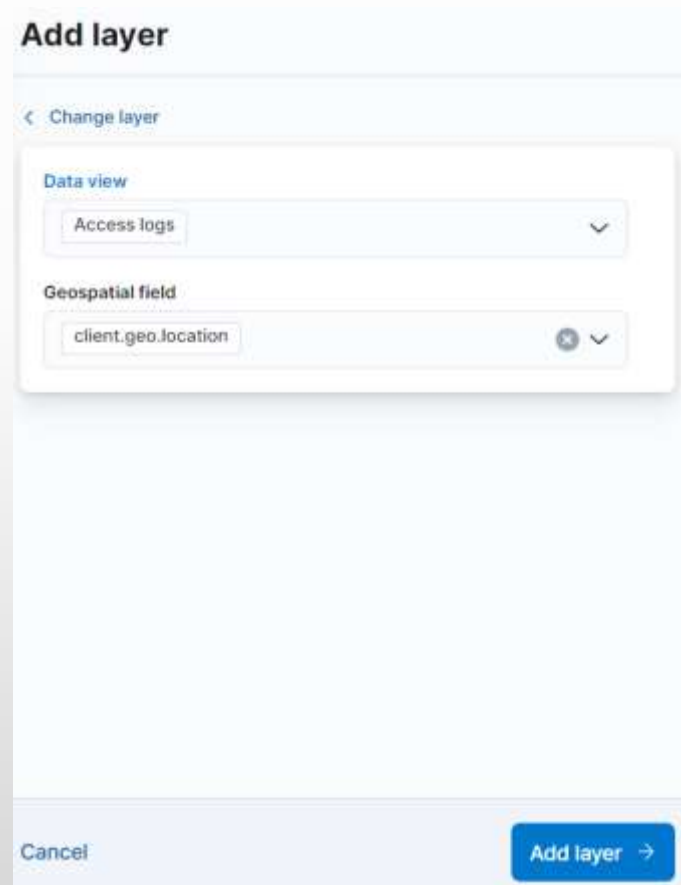
Month [⚙](#) ▼

Select an option or create a custom value. Examples: 30s, 20m, 24h, 2d, 1w, 1M



Visualización en mapa geográfico

- 1) Configuramos el basemap.
- 2) Añadimos todas las capas que necesitemos usando un campo de tipo geo_point:



Add layer

< Change layer

Data view

Access logs

Geospatial field

client.geo.location

Cancel

Add layer →



elastic

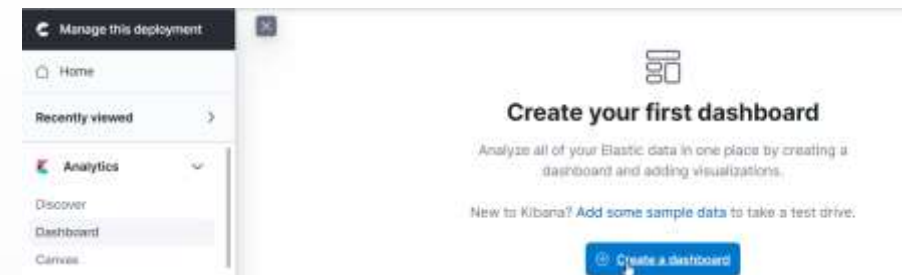
BLOQUE 9:
KIBANA –
CREACIÓN DE DASHBOARDS, ROLES Y PERMISOS

Lo que aprenderemos en este bloque....

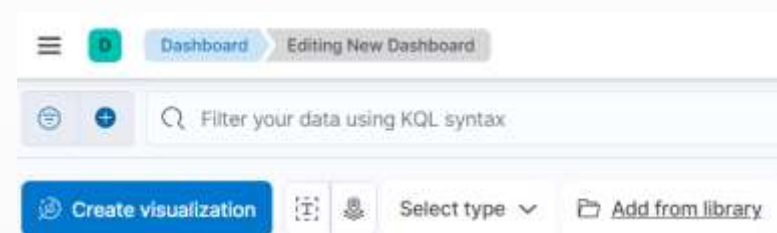
- ✓ ¿Cómo podemos crear completos dashboards utilizando las visualizaciones?
- ✓ ¿Cómo filtrar nuestros dashboards y añadir controles?
- ✓ ¿Cómo aplicar interactividad dentro del dashboard?
- ✓ ¿Cómo enlazar diferentes dashboards?
- ✓ ¿Cómo creamos usuarios para acceder a los dashboards y les aplicamos los roles y permisos oportunos?

Creación de un dashboard completo

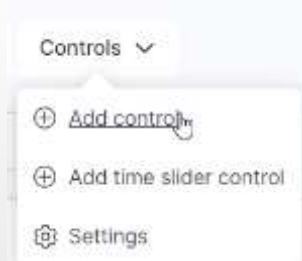
1) Pulsar en Analytics > Dashboard > Create



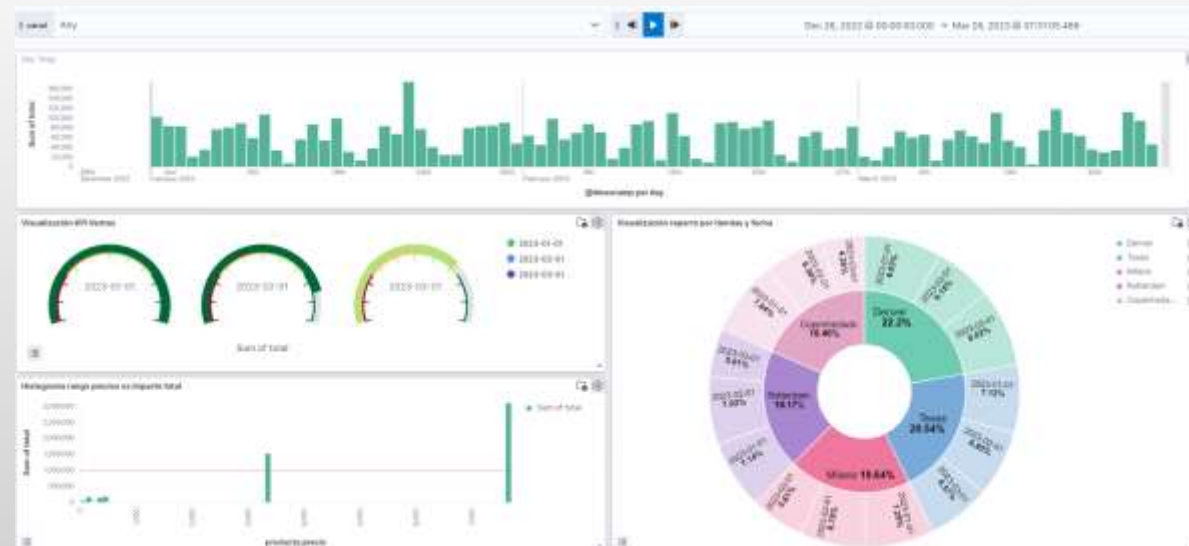
2) Crear nuevas visualizaciones o importar existentes.



3) Añadir controles

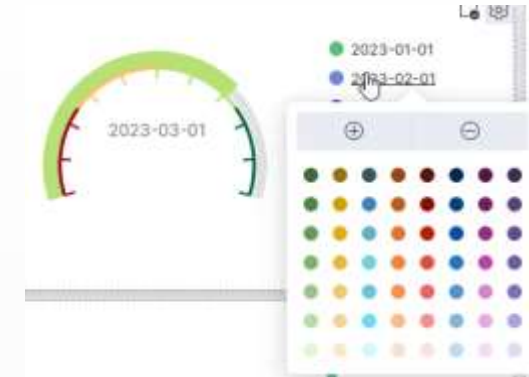


4) Ajustar layout



Editar visualizaciones y filtrar documentos

- 1) Personalizar colores si pulsamos en la leyenda.



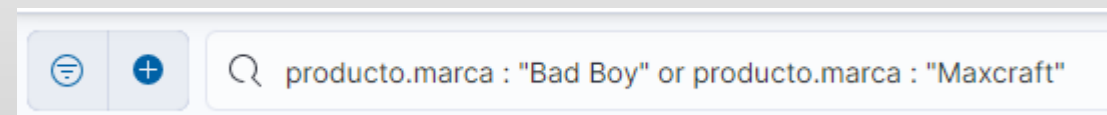
- 2) Es posible editar una visualización:



- 3) Inspeccionar la visualización para descargar datos o ver la consulta en Query DSL enviada a Elasticsearch

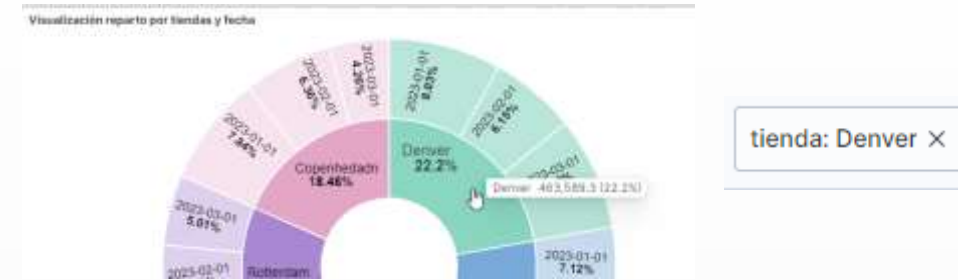


- 4) Aplicar consultas KQL al dashboard:

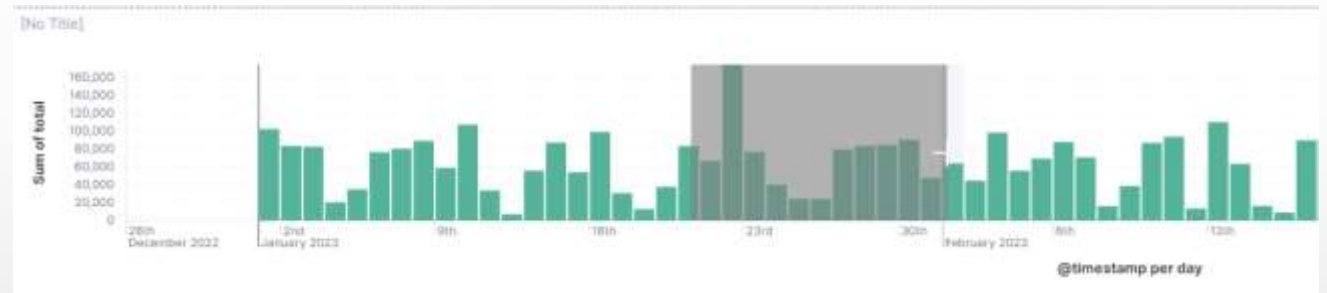


Interactividad en el dashboard

- 1) Pulsar en una categoría de la visualización para filtrar el dashboard al completo:



- 2) Seleccionar un intervalo de fechas:



- 3) Seleccionar 1 día concreto hará que se expanda a nivel horario:



Creación de dashboard logs de acceso

- 1) Reutilizamos las visualizaciones creadas previamente.
- 2) En las visualizaciones que tienen 2 campos (ejemplo heatmaps), la interactividad nos solicitará por qué campo filtrar.

×

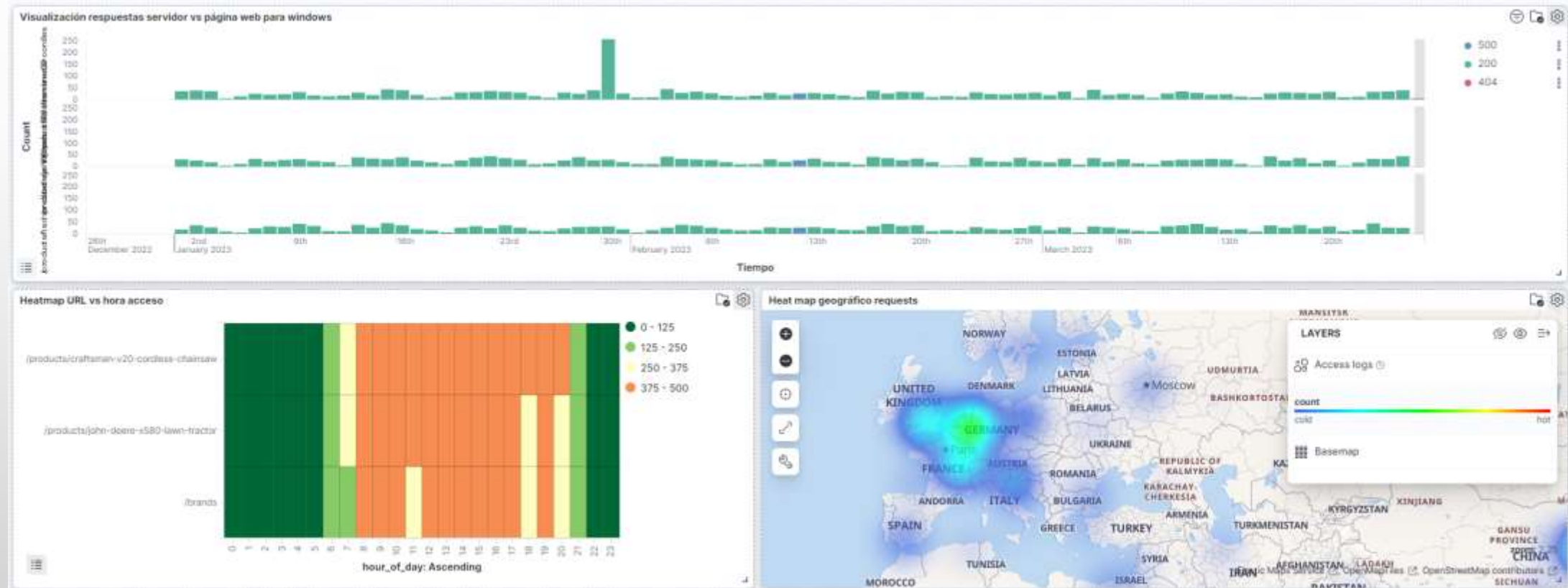
Select filters to apply

☒ hour_of_day: 17

☒ url.path: /products/craftsman-v20-cordless-chainsaw

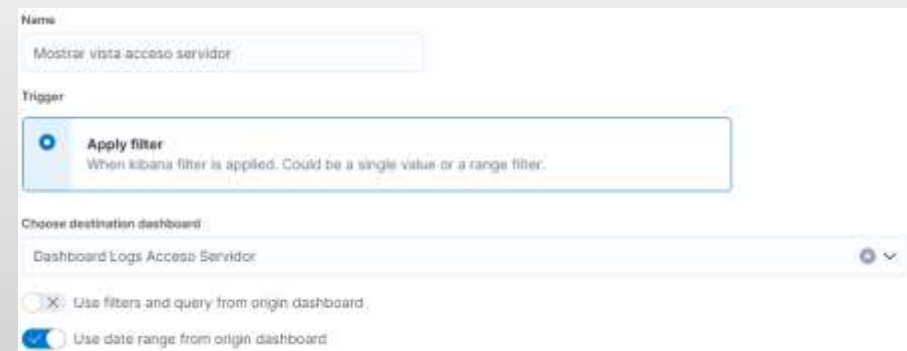
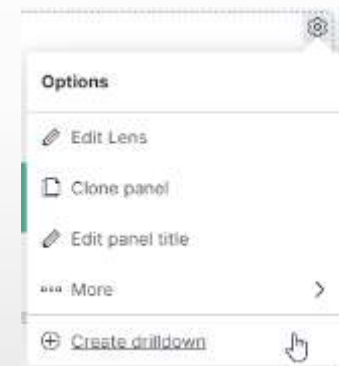
Cancel

Apply

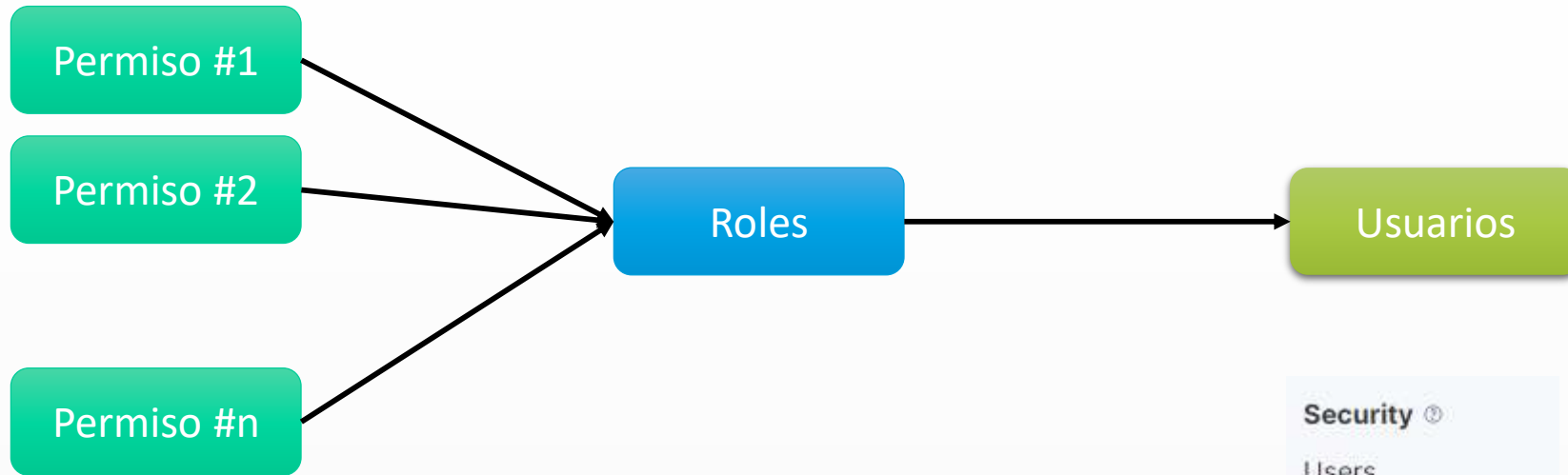


Enlazar dashboards (drilldown)

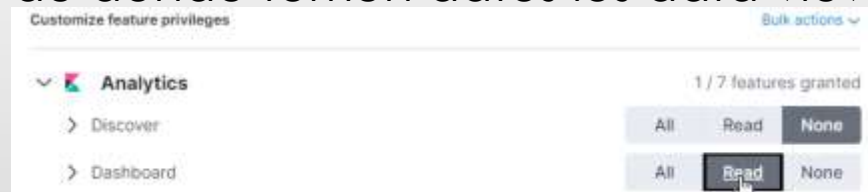
- 1) Es posible enlazar dashboards que tengan algún campo en común:
 - a) Dashboards que utilizan el mismo data view, ejemplo: Visualizar para una tienda concreta el dashboard de clientes asociados.
 - b) Dashboards que NO utilizan el mismo data view se pueden enlazar por el campo de fechas, ejemplo: Visualizar dashboards de solicitudes al servidor en el periodo de mayor número de ventas.
- 2) En cualquier visualización pulsamos configuración > “Create drilldown”.
- 3) Configuración drilldown, si los dashboards solo comparten rango de fechas (diferente data view), marcar “usar date range from origin dashboard”:



Creación de usuarios y roles



- 1) Crear rol: Stack Management > Security > Roles > Create Role
- 2) Configurar solo lo que queramos permitir para el rol (importante: debe tener permiso a los índices de Elasticsearch de donde tomen datos los data views del dashboard)



- 3) Crear usuario asignando el rol definido.





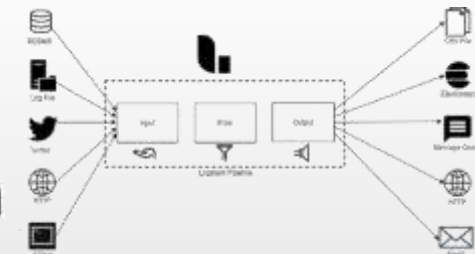
BLOQUE: CONCLUSIONES

Conclusiones

- **Elasticsearch** es un **motor de búsqueda** analítico open-source full-text fácil de usar y **altamente escalable**.
- Es posible realizar búsquedas complejas de datos sobre toda la información **indexada** que tengamos incluyendo **consultas sobre datos estructurados** como números o agregaciones para ser usado como plataforma analítica.



- **Logstash** es un pipeline de procesamiento de datos de cualquier naturaleza y tiene 3 fases: Input – Filter – Output.



- **Kibana** es una plataforma de visualización y análisis con capacidad de construir potentes **dashboards** y que invoca a Elasticsearch para obtener datos.
- La pila **Elastic Stack** al completo proporciona una **plataforma muy potente** y flexible para manejar grandes volúmenes de datos de una manera **muy eficiente** y creando potentes dashboards para analizar la información.



Gracias

- Iván Pinar Domínguez
- www.linkedin.com/in/ivanpinar

