Introducción a ELK

https://www.elastic.co/

INTRODUCCIÓN

Introducción

- Elasticsearch es un motor de búsqueda y análisis de código abierto basado en la biblioteca Apache Lucene. Inicialmente lanzado en 2010 por Elastic, Elasticsearch fue diseñado como solución Java distribuida para incorporar la funcionalidad de búsqueda de texto completo a documentos JSON sin esquema en múltiples tipos de base de datos.
- Apache Lucene es una biblioteca de motor de búsqueda de código abierto y gratuita, escrita completamente en Java. Lucene es reconocida principalmente por su implementación de motores de búsqueda. Lucene utiliza documentos como su principal unidad de búsqueda e indexación. Como indexa y almacena todo el contenido de los documentos en estructuras de datos centradas en palabras clave, Lucene puede lograr tiempos de respuesta de búsqueda extremadamente rápidos.
- Lucene hace todo el trabajo de almacenamiento de datos y consulta, pero Elasticsearch lo administra y convierte en un sistema distribuido fácil de usar y escalar. Elasticsearch proporciona prestaciones que permiten recopilar, buscar, analizar y visualizar datos de manera eficiente. Elasticsearch también cuenta con un interfaz API RESTful que brinda a los desarrolladores una flexibilidad increíble cuando llama a diferentes formatos de datos para tareas de búsqueda, visualización y análisis.

Introducción

- Elasticsearch fue lanzado en 2010 por Shay Banon y se podría definir una Base de Datos NoSQL orientada a documentos JSON. Originalmente concebido como un proyecto personal, a lo largo de los años, se ha convertido en un ecosistema de herramientas esencial en el ámbito del análisis de datos y ha evolucionado para incluir una amplia gama de características y mejoras, consolidándose como una solución líder en el mercado.
- Búsqueda (Enterprise Search) y análisis:
  - Búsqueda de texto completo, búsqueda de vectores (ML) y búsqueda de semántica.
  - Visualización Tabular, Graficas, Dashboards, Canvas, Maps.
- Ingesta:
  - ETL, Agentes e Integraciones.
- Observabilidad:
  - Monitoreo de logs (logging rápido y escalable que no se detendrá), Monitoreo de infraestructura (controla y visualiza las métricas del sistema), APM (obtiene información sobre el rendimiento de la aplicación), Monitoreo sintético (monitorea y reacciona a problemas de disponibilidad), Automatización del flujo de trabajo (configura reglas y realiza acciones automatizadas frente a alertas, anomalías e información)
- Seguridad:
  - SIEM Seguridad de endpoint: Prevé, detecta, busca y responde a amenazas.

Casos de uso

- **Búsqueda de texto completo**: Debido a su capacidad para indexar y buscar grandes volúmenes de texto rápidamente, Elasticsearch se utiliza ampliamente en la búsqueda de texto completo para aplicaciones como motores de búsqueda internos en sitios web, bases de datos de documentos, y cualquier sistema que requiera búsqueda avanzada de texto.
- **Análisis y visualización de datos**: Elasticsearch es una poderosa herramienta para el análisis y la visualización de datos. Integrado con Kibana, permite a los usuarios crear dashboards interactivos, gráficos y reportes, facilitando el análisis de grandes conjuntos de datos de manera visual e intuitiva.
- **Gestión del rendimiento de las aplicaciones**: En la gestión del rendimiento de las aplicaciones (APM), encontrar y superar obstáculos en su código muchas veces se reduce a una búsqueda fiable. Elasticsearch puede correlacionar registros y métricas para indexarlos y facilitar su búsqueda en toda su infraestructura. Esto proporciona a los equipos de desarrollo las herramientas que necesitan para acelerar el tratamiento de problemas críticos de rendimiento y evitar cuellos de botella costosos. Y, como es de código abierto, muchos desarrolladores ya han ideado formas útiles de optimizar las prestaciones de APM de Elasticsearch al máximo.
- **SIEM**: La gestión de sucesos y seguridad de la información (SIEM) es un componente crítico para reforzar la seguridad en el entorno digital actual. La velocidad, escalabilidad y potencia de análisis de Elasticsearch permite a los equipos de seguridad automatizar la correlación de miles de millones de líneas de datos de registro para buscar vulnerabilidades de red y posibles infracciones de datos.

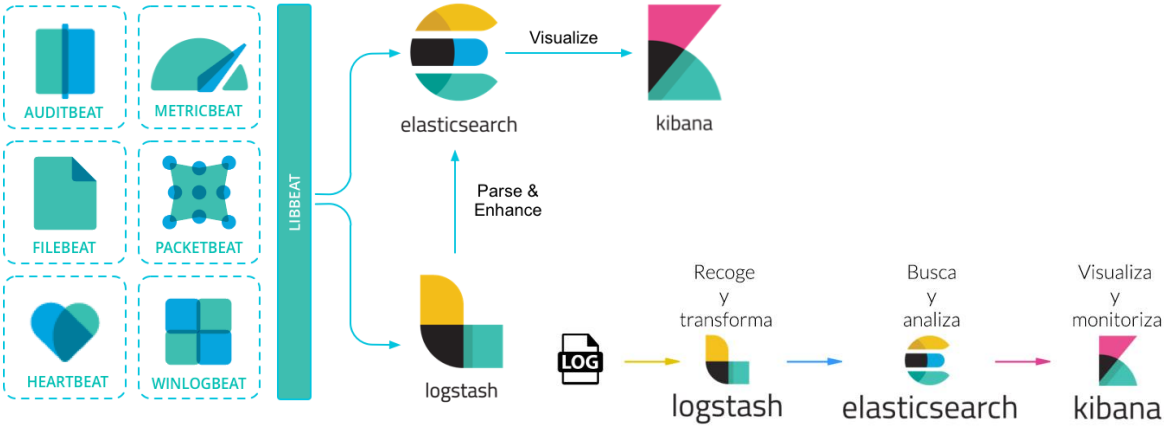
ELK (Elastic Stack)

- "ELK" son el acrónimo para tres proyectos open source: Elasticsearch, Logstash y Kibana. Más adelante, también se ha incorporado Beats. La pila ELK esta construida sobre una base gratuita y abierta, Elasticsearch, Logstash y Kibana, herramientas que juntas proporcionan una solución integrada para la búsqueda, el análisis y la visualización de datos en tiempo real y tiene una gran adaptabilidad al ser open source, de manera que permite personalizar y adaptar estas soluciones a cada empresa o necesidad concreta.
  1. Logstash ingiere, transforma y envía los datos al destino correcto.
  2. Elasticsearch indexa, analiza y busca los datos ingeridos.
  3. Kibana visualiza los resultados del análisis.
- Las características avanzadas de Elastic, como machine learning, seguridad, X-Pack, integraciones y otras, así como la nube, son módulos de pago.
- El 21 de enero de 2021, Elastic NV anunció que cambiaría su estrategia de licencias de software y no lanzaría nuevas versiones de Elasticsearch y Kibana bajo la licencia permisiva Apache, versión 2.0 (ALv2). El proyecto OpenSearch es una bifurcación de Elasticsearch y Kibana de código abierto impulsada por la comunidad y con licencia ALv2.

ELK: Elasticsearch, Logstash y Kibana

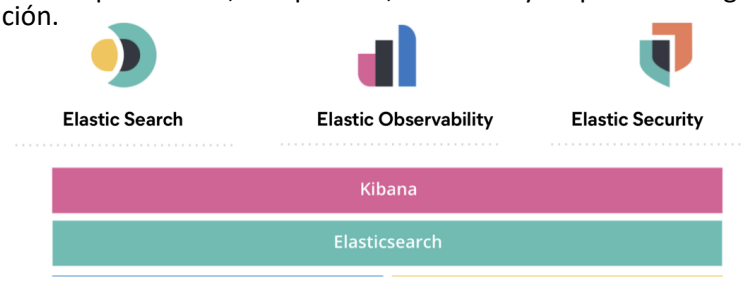
- **Elasticsearch** es un motor de búsqueda y analítica RESTful distribuido capaz de abordar multitud de casos de uso. Como núcleo del Elastic Stack, almacena e indexa de forma centralizada los datos para una búsqueda (estructuradas, no estructuradas, geográficas, métricas, …) a gran velocidad, con relevancia refinada y analíticas que escalan con facilidad.
- **Kibana** es una interfaz de usuario gratuita y abierta que permite visualizar los datos de Elasticsearch y navegar en el Elastic Stack, desde rastrear la carga de búsqueda hasta comprender la forma en que las solicitudes fluyen por las apps. Como una imagen vale más que mil líneas de log, Kibana envía los datos en forma de histogramas, grafos de líneas, gráficos circulares, proyecciones solares y más, para análisis de logs, monitoreo de infraestructura, APM (Application Performance Monitoring), operaciones de seguridad, analítica de negocios, …
- **Logstash** es un ETL obtiene, transforma y envía de forma dinámica los datos independientemente de su formato o complejidad. Admite una variedad de entradas de una manera de transmisión continua que extraen eventos de una multitud de fuentes comunes, todo al mismo tiempo: logs, métricas, aplicaciones web, almacenes de datos, varios servicios de AWS, … los filtros transforman cada evento, para que converjan en un formato común para el análisis y un valor comercial más poderosos. Los resultados se pueden enviar a Elasticsearch para búsquedas y análisis o a una variedad de salidas.
- **Beats** son agentes ligeros que recolectan información en origen y envían los datos directamente a Elasticsearch o, si necesitan transformación, a Logstash. Hay agentes para ficheros (logs y otros datos), datos de métricas, datos de red, logs de eventos de Windows, información de auditoría, salubridad (monitoreo de tiempo de actividad), …

ELK: Elasticsearch, Logstash y Kibana



Elasticsearch Service

- **Analytics**: Explora, visualiza y analiza tus datos utilizando un potente conjunto de herramientas y aplicaciones analíticas.
- **Enterprise Search**: Crea como usuario final experiencias de búsqueda poderosas y modernas para tu espacio de trabajo, sitio web o aplicaciones.
- **Observability**: Unifica tus logs, métricas y rastreos de APM a escala en un solo stack.
- **Security**: Obtén prevención, recopilación, detección y respuestas integrales para tu organización.



ELASTICSEARCH

Introducción

- Elasticsearch es el motor de búsqueda y análisis distribuido que se encuentra en el corazón del Elastic Stack, es donde se produce la magia de la indexación, la búsqueda y el análisis.
- Elasticsearch ofrece búsquedas y análisis casi en tiempo real para todo tipo de datos, ya sean texto estructurado o no estructurado, datos numéricos o datos geoespaciales, Elasticsearch puede almacenarlos e indexarlos de manera eficiente de una manera que admita búsquedas rápidas.
- Se puede ir mucho más allá de la simple recuperación de datos y agregar información para descubrir tendencias y patrones en los datos. Y, a medida que el volumen de datos y consultas crece, la naturaleza distribuida de Elasticsearch permite que la implementación crezca sin problemas junto con él.
- Si bien no todos los problemas son problemas de búsqueda, Elasticsearch ofrece velocidad y flexibilidad para manejar datos en una amplia variedad de casos de uso.

Casos de uso

- Agregar un cuadro de búsqueda a una aplicación o sitio web
- Almacenar y analizar registros, métricas y datos de eventos de seguridad
- Utilizar el aprendizaje automático para modelar automáticamente el comportamiento de los datos en tiempo real
- Utilizar Elasticsearch como una base de datos vectorial para crear, almacenar y buscar incrustaciones vectoriales
- Gestionar, integrar y analizar información espacial utilizando Elasticsearch como sistema de información geográfica (GIS)
- Automatizar los flujos de trabajo empresariales utilizando Elasticsearch como motor de almacenamiento
- Almacenar y procesar datos genéticos utilizando Elasticsearch como herramienta de investigación bioinformática

Arquitectura de Elasticsearch

- Elasticsearch es un almacén de documentos distribuido. En lugar de almacenar información como filas de datos en columnas, Elasticsearch almacena estructuras de datos complejas que se han serializado como documentos JSON y se almacenan en índices (el equivalente a tablas o colecciones).
- Cuando se almacena un documento, se indexa y se puede buscar en él casi en tiempo real (en menos de un segundo). Elasticsearch utiliza una estructura de datos denominada índice invertido que admite búsquedas de texto completo muy rápidas. Un índice invertido enumera cada palabra única que aparece en cualquier documento e identifica todos los documentos en los que aparece cada palabra.
- Apache Lucene, la biblioteca Java en la que se basa Elasticsearch, introdujo el concepto de búsqueda por segmento. Un segmento es similar a un índice invertido, pero la palabra índice en Lucene significa "una colección de segmentos más un punto de confirmación". Después de una confirmación, se agrega un segmento nuevo al punto de confirmación y se borra el búfer.
- Entre Elasticsearch y el disco se encuentra la memoria caché del sistema de archivos. Los documentos que se encuentran en el búfer de indexación en memoria se escriben en un segmento nuevo. El segmento nuevo se escribe primero en la memoria caché del sistema de archivos (lo que es económico) y solo después se vacía en el disco (lo que es costoso). Sin embargo, una vez que un archivo se encuentra en la memoria caché, se puede abrir y leer como cualquier otro archivo.

Arquitectura de Elasticsearch

- Un índice puede considerarse como una colección optimizada de documentos y cada documento es una colección de campos, que son los pares clave-valor que contienen sus datos. De forma predeterminada, Elasticsearch indexa todos los datos de cada campo, y cada campo indexado tiene una estructura de datos dedicada y optimizada. Por ejemplo, los campos de texto se almacenan en índices invertidos, y los campos numéricos y geográficos se almacenan en árboles BKD. La capacidad de usar las estructuras de datos por campo para ensamblar y devolver resultados de búsqueda es lo que hace que Elasticsearch sea tan rápido.
- Elasticsearch también tiene la capacidad de no tener esquemas, lo que significa que los documentos se pueden indexar sin especificar explícitamente cómo manejar cada uno de los diferentes campos que pueden aparecer en un documento. Cuando se habilita el mapeo dinámico, Elasticsearch detecta y agrega automáticamente nuevos campos al índice. Este comportamiento predeterminado facilita la indexación y exploración de sus datos: simplemente comience a indexar documentos y Elasticsearch detectará y mapeará valores booleanos, de punto flotante y enteros, fechas y cadenas a los tipos de datos de Elasticsearch adecuados.

Clusters, Nodes y Shards

- Elasticsearch está diseñado para estar siempre disponible y escalar según las necesidades. Esto se logra al ser distribuido por naturaleza. Se pueden agregar servidores (nodos) a un clúster para aumentar la capacidad y Elasticsearch distribuye automáticamente sus datos y la carga de consultas entre todos los nodos disponibles. No es necesario revisar la aplicación, Elasticsearch sabe cómo equilibrar clústeres de múltiples nodos para brindar escala y alta disponibilidad. Cuantos más nodos, mejor.
- Un índice de Elasticsearch es simplemente una agrupación lógica de uno o más fragmentos (shards) físicos, donde cada fragmento es en realidad un índice autónomo. Al distribuir los documentos de un índice en varios fragmentos y distribuir esos fragmentos en varios nodos, Elasticsearch puede garantizar la redundancia, lo que protege contra fallos de hardware y aumenta la capacidad de consulta a medida que se agregan nodos a un clúster. A medida que el clúster crece (o se reduce), Elasticsearch migra automáticamente los fragmentos para reequilibrarlo.
- Existen dos tipos de fragmentos: primarios y réplicas. Cada documento de un índice pertenece a un fragmento primario. Un fragmento de réplica es una copia de un fragmento primario. Las réplicas proporcionan copias redundantes de los datos para protegerse contra fallos de hardware y aumentar la capacidad para atender solicitudes de lectura, como la búsqueda o recuperación de un documento. La cantidad de fragmentos primarios en un índice es fija (cuando se define el índice), pero la cantidad de fragmentos de réplica puede cambiar en caliente y en cualquier momento.

Clusters

- Cuando se despliega como clúster, se debe considerar los roles que pueden tomar los nodos. Cada nodo del clúster puede ejecutar uno o varios roles.
  - Master node: Se encarga de realizar la gestión de los índices, la distribución de los shards en los nodos de datos (data nodes) y su seguimiento.
  - Data node: Encargados de almacenar los shards y ejecutar los procesos de búsqueda y de indexación.
  - Ingest node: Este tipo de nodo se encarga de realizar la ingesta de los datos. Para ello, escucha las entradas de datos que se producen y los escribe. Estas operaciones las realiza en memoria.
  - Coordinating node: Nodos encargados de balancear la carga y de preprocesar los datos en las fases de agregación.
- En clusters grandes (a partir de 10 nodos), es recomendable separar estos roles en diferentes nodos.
- Los servicios del clúster se comunican entre ellos por el puerto TCP 9300. También se puede habilitar el autodescubrimiento de nodos, de forma que se facilita la configuración del clúster.
- La configuración de Elasticsearch es muy flexible por lo que puede ser muy compleja.

API REST

- Elasticsearch ofrece una API REST sencilla y coherente para administrar el clúster e indexar y buscar los datos. Las API REST admiten consultas estructuradas (similares a los tipos de consultas que se pueden crear en SQL), consultas de texto completo y consultas complejas que combinan ambas.
- Las consultas de texto completo encuentran todos los documentos que coinciden con la cadena de consulta y los devuelven ordenados por relevancia (como de buenos son para los términos de búsqueda). Además de buscar términos individuales, se pueden realizar búsquedas de frases, semánticas, de similitud, de prefijos, de sugerencias de autocompletar …
- Elasticsearch indexa datos no textuales en estructuras de datos optimizadas que admiten consultas numéricas y geográficas de alto rendimiento.
- Se puede acceder a todas estas capacidades de búsqueda mediante el completo lenguaje de consulta de estilo JSON de Elasticsearch (Query DSL) o también se pueden crear consultas de **estilo** SQL para buscar y agregar datos de forma nativa dentro de Elasticsearch.
- Desde las aplicaciones, se puede utilizar el cliente Elasticsearch especifico para el lenguaje, disponibles en: Java, JavaScript, Go, .NET, PHP, Perl, Python o Ruby. Los controladores JDBC y ODBC permiten que una amplia gama de aplicaciones de terceros interactúen con Elasticsearch a través de SQL. Se pueden enviar solicitudes directamente desde la línea de comandos (curl) o a través de la consola de desarrollador en Kibana.

Mapeo

- El mapeo es el proceso de definir cómo se almacenan e indexan un documento y los campos que contiene.
- Cada documento es una colección de campos, cada uno de los cuales tiene su propio tipo de datos. Al asignar los datos, se crea una definición de asignación, que contiene una lista de campos que son pertinentes al documento. Una definición de asignación también incluye campos de metadatos, como el campo _source que personalizan cómo se manejan los metadatos asociados a un documento. Un índice puede contener diferentes tipos de documentos, por lo que definirá todos los campos de los diferentes tipos, reutilizando los coincidentes. Se puede utilizar asignación dinámica y asignación explícita para definir el mapeo de datos.
- El mapeo dinámico permite experimentar y explorar datos cuando se está comenzando. Elasticsearch agrega nuevos campos automáticamente, simplemente al indexar (añadir) un documento. Se pueden agregar campos al mapeo de nivel superior y a los object campos internos nested.
- El mapeo explícito permite elegir con precisión los campos disponibles: tipos, formatos, indexaciones, … Los documentos no podrán contener campos no mapedos.

Mapeo dinámico

- Para indexar un documento, no es necesario que crear primero un índice, definir un tipo de mapeo y definir los campos. Se puede simplemente indexar un documento: el índice se creará, se mapearán los campos y agregaran los datos automáticamente:

  PUT demo/_doc/1

  { "count": 5 }
- La detección y adición automática de nuevos campos se denomina mapeo dinámico. Las reglas de mapeo dinámico se pueden personalizar para adaptarse a sus propósitos con:
  - Mapeos de campos dinámicos (las reglas que rigen la detección de campos dinámicos)
  - Plantillas dinámicas (reglas personalizadas para configurar la asignación de campos agregados dinámicamente)

Mapeos de campos dinámicos

- Cuando Elasticsearch detecta un nuevo campo en un documento, agrega dinámicamente el campo a la asignación de tipos de forma predeterminada. Elasticsearch utiliza reglas para determinar cómo asignar los tipos de datos para cada campo: true o false (boolean), number (double, float, long, …), object (object), array (array), string (text, date, long, float, …), null (no se mapea), …
- Se pueden personalizar la detección de fechas y numéricos para las reglas de asignación de campos dinámicos:

  PUT demo

  { "mappings": { "dynamic_date_formats": ["dd/MM/yyyy"] }}
- La detección dinámica de fecha y numéricos se puede activar y desactivar:

  PUT demo

  { "mappings": { "date_detection": false, "numeric_detection": true } }

Plantillas dinámicas

- Las plantillas dinámicas de índices permiten un mayor control de cómo Elasticsearch asigna los datos más allá de las reglas de asignación de campos dinámicos predeterminadas.
- Se pueden usar plantillas dinámicas para definir asignaciones personalizadas que se pueden aplicar a campos agregados dinámicamente según la condición de coincidencia:
  - match_mapping_type y unmatch_mapping_type operan sobre el tipo de datos que Elasticsearch detecta
  - match y unmatch usan un patrón para que coincida con el nombre del campo
  - path_match y path_unmatch operan sobre la ruta separada por puntos hasta el campo
- Si una plantilla dinámica no define match_mapping_type, match o path_match, no coincidirá con ningún campo.

```
PUT my-index-000001
{
  "mappings": {
    "dynamic_templates": [
      {
        "longs_as_strings": {
          "match_mapping_type": "string",
          "match":   "long_*", "unmatch": "*_text",
          "mapping": {
            "type": "long"
          }
```


Mapeo explícito

- Se puede crear asignaciones de campos cuando se crea un índice y, posteriormente, agregar campos a un índice existente.

```
PUT /my-index-000001
{ "mappings": {
    "properties": {
      "age":    { "type": "integer" },
      "email":  { "type": "keyword"  },
      "name":   { "type": "text"  }
    }}}
PUT /my-index-000001/_mapping
{ "properties": {
    "employee-id": { "type": "keyword", "index": false }
  } }
```

- A excepción de los parámetros de asignación admitidos, no se puede cambiar la asignación ni el tipo de campo de un campo existente. Cambiar un campo existente podría invalidar los datos que ya están indexados.
- Puede utilizar la API de obtención de mapeo para ver el mapeo de un índice existente o ver la asignación de uno o más campos específicos.

```
GET /my-index-000001/_mapping
GET /my-index-000001/_mapping/field/employee-id
```

Tipos de datos de campo

- Cada campo tiene un tipo de datos de campo o field type. Este tipo indica el tipo de datos que contiene el campo, como cadenas o valores booleanos, y su uso previsto. En función al tipo, Elasticsearch genera la indexación apropiada y restringe las operaciones de búsqueda que se pueden realizar sobre ellos. Por ejemplo, puede indexar cadenas en los campos text o keyword pero los valores de campo text se analizan para la búsqueda de texto completo, mientras que las cadenas keyword se dejan tal como están para filtrarlas y ordenarlas.
- Los tipos de campos se agrupan por familia dado que todos los tipos de la misma familia tienen exactamente el mismo comportamiento de búsqueda, pero pueden tener diferentes características de rendimiento o espacio de almacenamiento. Los nombres de las familias aparecen en PascalCase.
- El tipo alias define un nombre alternativo para un campo en el índice que se pueden utilizar en lugar del campo de destino en las solicitudes de búsqueda y en otras API, evitando costosas transformaciones.

Tipos de datos de campo

- Tipos comunes
  - binary: Valor binario codificado como una cadena Base64.
  - boolean: Valores true y false.
  - Keywords: La familia de palabras clave, que incluye keyword, constant_keyword, y wildcard.
  - Numbers: Los tipos numéricos, como long y double, se utilizan para expresar cantidades.
  - Dates: Tipos de fecha, incluidos date y date_nanos.
  - alias: Define un alias para un campo existente.
- Objetos y tipos relacionales
  - object: Un objeto JSON.
  - flattened: Un objeto JSON completo como un único valor de campo.
  - nested: Un objeto JSON que conserva la relación entre sus subcampos.
  - join: Define una relación padre/hijo para los documentos en el mismo índice.
- Tipos de datos estructurados
  - Range: Tipos de rango, como long_range, double_range, date_range e ip_range.
  - ip: Direcciones IPv4 e IPv6.
  - version: Versiones de software. Admite reglas de precedencia de versiones semánticas.
  - murmur3: Calcula y almacena hash de valores.
- Tipos de datos agregados
  - aggregate_metric_double: Valores métricos preagregados.
  - histogram: Valores numéricos preagregados en forma de histograma.

Tipos de datos de campo

- Tipos de búsqueda de texto
  - Text: La familia textual, que incluye text y match_only_text. Texto analizado y no estructurado.
  - annotated-text: Texto que contiene marcado especial. Se utiliza para identificar entidades con nombre.
  - completion: Se utiliza para sugerencias de autocompletar.
  - search_as_you_type: text-tipo similar para completar a medida que escribe.
  - semantic_text: Se utiliza para realizar búsquedas semánticas.
  - token_count: Un recuento de tokens en un texto.
- Tipos de clasificación de documentos
  - dense_vector: Registra vectores densos de valores float.
  - sparse_vector: Registra vectores dispersos de valores float.
  - rank_feature: Registra una función numérica para aumentar los resultados en el momento de la consulta.
  - rank_features: Registra características numéricas para aumentar los resultados en el momento de la consulta.
- Tipos de datos espaciales
  - geo_point: Puntos de latitud y longitud.
  - geo_shape: Formas complejas, como polígonos.
  - point: Puntos cartesianos arbitrarios.
  - shape: Geometrías cartesianas arbitrarias.
- Otros tipos
  - percolator: Consultas de índices escritas en Query DSL.

Mapeo de contenido no estructurado

- Se puede asignar un campo que contenga contenido no estructurado a un campo de la familia Text o de la familia Keyword. El mejor tipo de campo depende de la naturaleza del contenido y de cómo planea buscar en el campo.
- Se utiliza un campo de la familia Text si:
  - El contenido es legible para humanos, como el cuerpo de un correo electrónico o la descripción de un producto.
  - Si se planea buscar en el campo palabras o frases individuales mediante consultas de texto completo. Elasticsearch analiza los campos text para devolver los resultados más relevantes para estas consultas.
- Se utiliza un campo de la familia Keyword si:
  - El contenido es generado por máquina, como un mensaje de registro o información de solicitud HTTP.
  - Si se planea buscar en el campo valores completos exactos o secuencias de caracteres parciales (comodin) mediante consultas a nivel de término o se utilizan a menudo en ordenaciones y agregaciones.
- Si se utiliza un campo de la familia Keyword se puede asignar como un campo keyword o wildcard según la cardinalidad y el tamaño de los valores del campo.
  - Se utiliza el tipo wildcard si se planea buscar regularmente en el campo mediante una consulta wildcard (con comodines) o regexp y cumple con uno de los siguientes criterios:
    - El campo contiene más de un millón de valores únicos y se planea buscar regularmente en el campo usando un patrón con comodines iniciales (*fooo).
    - El campo contiene valores mayores a 32 KB y se planea buscar regularmente en el campo usando cualquier patrón comodín.
  - De lo contrario, se utiliza el tipo keyword para contenido estructurado y realizar búsquedas más rápidas, una indexación más veloz y menor coste de almacenamiento. Se puede usar constant_keyword para campos que siempre contienen el mismo valor.

Objetos y tipos relacionales

- Los documentos JSON son de naturaleza jerárquica: el documento puede contener objetos internos que, a su vez, pueden contener objetos internos en sí mismos.
- El tipo object está indexado como una lista simple y plana de pares clave-valor, donde clave es el path (campo.subcampo.subsubcampo):

```
"author": {
 "type": "object",
 "properties": {
  "age":  { "type": "integer" },
  "name": {
   "properties": { "first": { "type": "text" }, "last":  { "type": "text" } }
  }
 } }
```

- El tipo flattened proporciona un enfoque alternativo, en el que todo el objeto se asigna como un único valor al campo, no define sus properties y se indexará como palabras clave.
- El tipo nested es una versión especializada del tipo object de datos que permite indexar matrices de objetos de manera que puedan consultarse independientemente unos de otros.

Campos de metadatos

- Cada documento tiene metadatos asociados. El comportamiento de algunos de estos campos de metadatos se puede personalizar cuando se crea una asignación.
  - _index: El índice al que pertenece el documento.
  - _id: El ID del documento.
  - _source: El JSON original que representa el cuerpo del documento.
  - _size: El tamaño del campo _source en bytes, proporcionado por el complemento mapper-size.
  - _doc_count: Un campo personalizado que se utiliza para almacenar recuentos de documentos cuando un documento representa datos previamente agregados.
  - _field_names: Todos los campos del documento que contienen valores no nulos.
  - _ignored: Todos los campos del documento que se han ignorado en el momento de la indexación debido a ignore_malformed.
  - _routing: Un valor de enrutamiento personalizado que enruta un documento a un fragmento particular.
  - _meta: Metadatos específicos de la aplicación.
  - _tier: La preferencia de nivel de datos actual del índice al que pertenece el documento.

Campos en tiempo de ejecución

- Un campo en tiempo de ejecución (runtime fields) es un campo calculado que se evalúa en el momento de la consulta. Los campos de tiempo de ejecución permiten:
  - Agregar campos a documentos existentes sin reindexar sus datos
  - Empezar a trabajar con datos sin comprender cómo están estructurados
  - Anular el valor devuelto desde un campo indexado en el momento de la consulta
  - Definir campos para un uso específico sin modificar el esquema subyacente
- Los campos en tiempo de ejecución se definen en uno de los lenguajes de scripting soportados por Elasticsearch.
- Se pueden definir los runtime fields en la asignación de índice o en una solicitud de búsqueda. Se puede acceder a estos campos desde la API de búsqueda como cualquier otro campo, Elasticsearch no ve los campos de tiempo de ejecución de manera diferente.
- Se deben solicitar explícitamente utilizando el parámetro fields en la API _search para recuperar los valores de los runtime fields. Los runtime fields no se mostrarán en _source, pero la API fields funciona para todos los campos, incluso aquellos que no se enviaron como parte del original _source.
- Los runtime fields son útiles cuando se trabaja con datos de registro, especialmente cuando no se está seguro de la estructura de los datos. La velocidad de búsqueda disminuye, pero el tamaño del índice es mucho menor y puede procesar registros más rápidamente sin tener que indexarlos.

Data Streams

- Un flujo de datos (data stream) permite almacenar datos de series temporales de solo anexión en varios índices y, al mismo tiempo, brinda un único recurso con nombre para las solicitudes. Los flujos de datos son adecuados para registros, eventos, métricas y otros datos generados de manera continua.
- Un flujo de datos consta de uno o más índices de respaldo ocultos y generados automáticamente. Se pueden enviar solicitudes de indexación y búsqueda directamente a un flujo de datos. El flujo enruta automáticamente la solicitud a los índices de respaldo que almacenan los datos del flujo. Cuando envía una solicitud de lectura a un flujo de datos, el flujo enruta la solicitud a todos sus índices de respaldo. El índice de respaldo creado más recientemente es el índice de escritura del flujo de datos y el flujo agrega nuevos documentos únicamente a este índice.
- Un flujo de datos requiere su correspondiente plantilla de índice. La plantilla contiene las asignaciones y configuraciones utilizadas para configurar los índices de respaldo del flujo. La misma plantilla de índice se puede utilizar para varios flujos de datos.
- Se puede utilizar la gestión del ciclo de vida del índice (ILM) para automatizar la gestión de estos índices de respaldo.

Plantillas de índice

- Una plantilla de índice es una forma de indicarle a Elasticsearch cómo configurar un índice cuando se crea. Las plantilla definen, mediante un patrón de nombre, a que índices se deben aplicar, cuando estos se creen, y una prioridad para el caso de que el nombre del índice satisfaga mas de un patrón.
- En el caso de los flujos de datos, la plantilla de índice configura los índices de respaldo del flujo a medida que se crean. Las plantillas se configuran antes de la creación del índice. Cuando se crea un índice (ya sea de forma manual o mediante la indexación de un documento), la configuración de la plantilla se utiliza como base para crear el índice.
- Existen dos tipos de plantillas: plantillas de índice y plantillas de componentes. Las plantillas de componentes son bloques de construcción reutilizables que configuran asignaciones, configuraciones y alias. Las plantillas de índice pueden contener una colección de plantillas de componentes, así como especificar directamente configuraciones, asignaciones y alias.

Scripting

- Con los scripts, se pueden evaluar expresiones personalizadas en Elasticsearch. Por ejemplo, puede usar un script para devolver un valor calculado como un campo runtime o evaluar una puntuación personalizada para una consulta.
- Los lenguajes de script disponibles son:
  - painless: Diseñado específicamente para Elasticsearch
  - expression (Lucene expressions language): Clasificación y ordenación personalizadas y rápidas
  - mustache: Plantillas
  - java: API de expertos
- El lenguaje de programación predeterminado es Painless. Painless está diseñado específicamente para Elasticsearch, se puede utilizar para cualquier propósito en las API de scripting y ofrece la mayor flexibilidad. Los demás lenguajes son menos flexibles, pero pueden resultar útiles para propósitos específicos.
- Hay complementos adicionales disponibles para ejecutar scripts escritos en otros lenguajes. Se puede especificar el lenguaje del script en cualquier lugar donde se ejecuten scripts.

Scripting

- Siempre que se admitan scripts en las API de Elasticsearch, la sintaxis sigue el mismo patrón que: especifica el lenguaje del script (opcional, por defecto Painless), proporciona la lógica del script (mediante un identificador o el código fuente) y los parámetros que se pasan al script (opcionales):

```
"script": {
  "lang":   "...",
  "source" | "id": "...",
  "params": { … }
}
```

- Se puede almacenar y recuperar scripts mediante la Scripts API. Los scripts almacenados reducen el tiempo de compilación y agilizan las búsquedas. A diferencia de los scripts regulares, los scripts almacenados requieren que se especifique un lenguaje de script en el parámetro lang.

Búsquedas

- El punto fuerte Elasticsearch es la flexibilidad en las consultas o búsquedas, de ahí su nombre. Elasticsearch dispone de una amplia capacidad para indexar los datos por lo que permite realizar los siguiente tipos de consultas de una forma eficiente y rápida:
  - Consultas a nivel de término: para filtrar documentos en función de valores precisos en datos estructurados.
  - Consultas de búsqueda texto completo: basadas en la aparición y posición de palabras en un texto.
  - Consultas geográficas: basadas en geopuntos (pares latitud/longitud) o geoformas (puntos, líneas, círculos, polígonos, …).
  - Consultas de formas: basadas en la capacidad de indexar geometrías arbitrarias de dos dimensiones (no geoespaciales).
  - Consultas vectoriales: basadas en la semántica y conocimiento (machine learning, PLN, …).
  - Consultas de span: basadas en intervalos posicionales de bajo nivel mas precisos que las búsquedas texto completo.
  - Consultas especializadas: basadas en distancias, reglas, documentos (similares, coincidentes, …), ranking, …
  - Consultas de mezcla (JOIN): basadas en campos de tipo nested, has_child y has_parent.
  - Consultas compuestas: consultas que envuelven otras consultas, ya sea para combinar sus resultados y puntajes, para cambiar su comportamiento o para cambiar el contexto.
- Elasticsearch dispone de múltiples lenguajes de consultas especializados.

Search API

- Una búsqueda consta de una o más consultas que se combinan y se envían a Elasticsearch. Los documentos que coinciden con las consultas de una búsqueda se devuelven en los hits (o resultados de búsqueda) de la respuesta.
- Una búsqueda también puede contener información adicional que se utiliza para procesar mejor las consultas:
  - Recuperar campos específicos
  - Ordenar resultados
  - Paginar resultados
  - Destacar resultados
  - Agregaciones
  - Y muchos mas
- Se puede utilizar la API de búsqueda (_search) para buscar y resumir datos almacenados en índices o flujos de datos de Elasticsearch. El parámetro "query" del cuerpo de la solicitud de la API acepta consultas escritas en Query DSL.
- Se pueden almacenar consultas predefinidas con las plantillas de búsqueda.

Query DSL

- Elasticsearch ofrece un completo Query DSL (lenguaje específico de dominio) basado en JSON para definir consultas. Es un lenguaje sumamente flexible y de gran alcance, además de simple, que permite conocer y explorar los datos de la mejor manera. Al ser utilizado a través de una interfaz de tipo JSON, las consultas son muy sencillas de leer y, lo más importante, de depurar.
- Se puede pensar en el Query DSL como un AST (árbol de sintaxis abstracta) de consultas, que consta de dos tipos de cláusulas:
  - Cláusulas de consulta de hojas (leaf)
    - Las cláusulas de consulta de hoja buscan un valor particular en un campo particular, como las consultas match, term o range. Estas consultas se pueden utilizar por sí solas.
  - Cláusulas de consulta compuestas (compound)
    - Las cláusulas de consulta compuestas envuelven otras consultas de hoja o compuestas y se utilizan para combinar múltiples consultas de manera lógica (como la consulta bool o dis_max) o para alterar su comportamiento (como la consulta constant_score).
- El parámetro "query" del cuerpo de la solicitud solo puede contener una clausula, hoja o compuestas, y, salvo en consultas simples, suele ser una cláusula de consulta compuesta.

Relevancia

- Las consulta de búsqueda de texto completo y otras devuelven una gran cantidad de coincidencias. De forma predeterminada, Elasticsearch ordena los resultados por puntuación de relevancia, que mide el grado de coincidencia en qué el documento satisface una consulta. La puntuación de relevancia es un valor positivo en coma flotante que se muestra en el campo _score del resultado. Cuanto mayor sea el valor _score, más relevante será el documento, su valor no es significativo salvo para la ordenación de los resultados dado que cada tipo de consulta puede calcular las puntuaciones de relevancia de forma diferente, el cálculo de la puntuación también depende de si la cláusula de consulta se ejecuta en un contexto de consulta o de filtro.
- Elasticsearch utiliza un modelo de puntuación denominado Función de puntuación práctica (BM25) de forma predeterminada. Este modelo se basa en la teoría de recuperación de información probabilística y tiene en cuenta factores como la frecuencia de términos, la frecuencia inversa de documentos y la normalización de la longitud de campo:
  - Frecuencia de término (TF): representa la cantidad de veces que aparece un término en un documento. Una frecuencia de término más alta indica una relación más fuerte entre el término y el documento.
  - Frecuencia inversa de documentos (IDF): este factor mide la importancia de un término en toda la colección de documentos. Un término que aparece en muchos documentos se considera menos importante, mientras que un término que aparece en menos documentos se considera más importante.
  - Longitud del campo normalizada: este factor tiene en cuenta la longitud del campo en el que aparece el término. A los campos más cortos se les da más peso, ya que el término se considera más significativo en un campo más corto.
- Para obtener la explicación del _score se añade "explain": true o
  GET /<index>/_explain/<document_id>

$$\sum_{i}^{n} IDF(q_i) \frac{f(q_i, D) * (k1 + 1)}{f(q_i, D) + k1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})}$$

Contexto de consulta y filtrado

- De forma predeterminada, Elasticsearch ordena los resultados de búsqueda coincidentes por la puntuación de relevancia, que mide el grado de coincidencia en qué el documento satisface una consulta. El cálculo de la puntuación también depende de si la cláusula de consulta se ejecuta en un contexto de consulta o de filtro.
- En el contexto de consulta, una cláusula de consulta responde a la pregunta "¿Qué grado de coincidencia tiene este documento con esta cláusula de consulta? ". Además de decidir si el documento coincide o no, la cláusula de consulta también calcula una puntuación de relevancia en el campo de metadatos _score. El contexto de consulta entra en vigor siempre que se pasa una cláusula de consulta a un parámetro query.
- En un contexto de filtro, una cláusula de filtrado responde a la pregunta "¿Este documento coincide con esta cláusula de consulta? " La respuesta es un simple Sí o No y no se calculan puntuaciones. El contexto de filtro se utiliza principalmente para filtrar datos estructurados de coincidencia exacta. Elasticsearch almacenará en caché automáticamente los filtros utilizados con frecuencia para acelerar el rendimiento. El contexto de filtro entra en vigor siempre que se pasa en una cláusula de consulta un parámetro filter.
- Cuando se combinan contextos de consulta y de filtrado, el filtrado no influye en el calculo del _score, solo se aplica al conjunto de resultados.

```
GET /_search
{
  "query": {
    "bool": {
      "must": [ { "match": { "content": "Elasticsearch" }} ],
      "filter": [ { "term":  { "status": "published" }}} ]
```

Consultas compuestas

- Las consultas compuestas envuelven otras consultas compuestas o de hoja, ya sea para combinar sus resultados y puntajes, para cambiar su comportamiento o para cambiar del contexto de consulta al contexto de filtro.
  - bool: La consulta predeterminada para combinar varias cláusulas de consulta compuesta o de hoja, como las cláusulas must, should, must_not o filter. Las cláusulas must y should tienen sus puntuaciones combinadas (cuantas más cláusulas coincidentes, mejor), mientras que las cláusulas must_not y filter se ejecutan en el contexto del filtro.
  - boosting: Devuelve los documentos que coinciden con una consulta positive, pero reduce la puntuación de los documentos que también coinciden con una consulta negative.
  - constant_score: Una consulta que envuelve otra consulta, pero la ejecuta en un contexto de filtro. A todos los documentos coincidentes se les asigna a _score la misma "constante".
  - dis_max: Una consulta que acepta múltiples consultas y devuelve todos los documentos que coinciden con cualquiera de las cláusulas de consulta. A diferencia de bool que combina las puntuaciones de todas las consultas coincidentes, dis_max se queda con la puntuación de la cláusula que mejor coincide.
  - function_score: Permite modifcar las puntuaciones devueltas por la consulta principal con funciones para tener en cuenta factores como la popularidad, la actualidad, la distancia o algoritmos personalizados implementados con scripts.
- Es muy habitual tener que envolver las consultas hoja en las consultas compuestas.

Consultas bool

- Una consulta bool combina varias consultas con las que deben coincidir los documentos. Se crean utilizando una o más cláusulas (consultas) booleanas, cada cláusula con una ocurrencia tipificada. Los tipos de ocurrencia son:
  - must: La cláusula debe aparecer en los documentos coincidentes y contribuirá a la puntuación.
  - should: La cláusula puede aparecer en el documento correspondiente y contribuirá positiva (si aparece) o negativamente (si no aparece) a la puntuación. Se puede utilizar el parámetro minimum_should_match para especificar el número o porcentaje de cláusulas should que deben coincidir con los documentos devueltos. Si la consulta bool incluye al menos una cláusula should y ninguna cláusula must o filter, el valor predeterminado es 1 o 0 en caso contrario.
  - filter: La cláusula debe aparecer en los documentos coincidentes. Las cláusulas de filtro se ejecutan en el contexto de filtro, lo que significa que se ignora la puntuación y las cláusulas se tienen en cuenta para el almacenamiento en caché.
  - must_not: La cláusula no debe aparecer en los documentos coincidentes. Las cláusulas se ejecutan en el contexto del filtro, lo que significa que se ignora la puntuación y se tienen en cuenta para el almacenamiento en caché. Como se ignora la puntuación, se devuelve una puntuación de 0 para todos los documentos.
- La consulta bool adopta un enfoque de "cuantas más coincidencias, mejor", por lo que la puntuación de cada coincidencia must o should se sumará para proporcionar el _score final de cada documento.

Consultas bool

```
POST _search
{
  "query": {
    "bool" : {
      "must" : { "term" : { "user.id" : "kimchy" } },
      "filter": { "term" : { "tags" : "production" } },
      "must_not" : { "range" : { "age" : { "gte" : 10, "lte" : 20 } } },
      "should" : [
        { "term" : { "tags" : "env1" } },
        { "term" : { "tags" : "deployed" } }
      ],
      "minimum_should_match" : 1,
      "boost" : 1.0
    }
  }
}
```


Búsqueda de valores exactos

- Son consultas hoja a nivel de término para buscar documentos en función de valores precisos en datos estructurados y establecer filtros. Devuelve documentos que:
  - exists: contienen cualquier valor indexado para un campo.
  - fuzzy: contienen términos similares al término de búsqueda. Elasticsearch mide la similitud o imprecisión mediante una distancia de edición de Levenshtein.
  - ids: en función de sus ID de documento.
  - prefix: contienen un prefijo específico en un campo proporcionado.
  - range: contienen términos dentro de un rango proporcionado.
  - regexp: contienen términos que coinciden con una expresión regular.
  - term: contienen un término exacto en un campo proporcionado.
  - terms: contienen uno o más términos exactos en un campo proporcionado.
  - terms_set: contienen una cantidad mínima de términos exactos en un campo proporcionado. Puede definir la cantidad mínima de términos coincidentes mediante un campo o un script.
  - wildcard: contienen términos que coinciden con un patrón comodín.

Búsqueda de valores exactos

```
"term": {
    "user.id": {
      "value": "kimchy",
      "boost": 1.0
    }
}
"term": { "full_text": "Hola mundo" }
"terms": { "user.id": [ "kimchy", "elkbee" ], "boost": 1.0 }
"prefix": { "user.id": "ki" }
"wildcard": { "user.id": "ki*y" }
"range": {
    "age": { "gte": 10, "lte": 20, "boost": 2.0 }
}
"regexp": {
    "user.id": {
      "value": "k.*y",
      "flags": "ALL",
      "case_insensitive": true,
      "max_determinized_states": 10000,
      "rewrite": "constant_score_blended"
    }
  }
```


Parámetros habituales para <field>

- case_insensitive: Permite la coincidencia sin distinción entre mayúsculas y minúsculas del patrón con los valores del campo indexado cuando se establece como verdadero, por defecto es falso.
- boost: Número en coma flotante utilizado para disminuir o aumentar los puntajes de relevancia de una consulta. El valor predeterminado es 1.0. Se puede utilizar el parámetro boost para ajustar las puntuaciones de relevancia para búsquedas que contengan dos o más consultas. Los valores de refuerzo son relativos al valor predeterminado de 1.0, un valor mayor que 1.0 aumenta la puntuación de relevancia y un valor entre 0 y 1.0 la disminuye.

Consultas de texto completo

- Las consultas hoja de texto completo permiten realizar búsquedas en campos de texto analizados.
  - match: La consulta estándar para realizar consultas de texto completo, incluidas consultas de coincidencias aproximadas y consultas de frase o proximidad.
  - match_bool_prefix: Crea una consulta bool que coincide con cada término como una consulta term, excepto el último término, que coincide como una consulta prefix.
  - match_phrase: Similar a la consulta match, pero se utiliza para hacer coincidir frases exactas o coincidencias de proximidad de palabras.
  - match_phrase_prefix: Me gusta la consulta match_phrase, pero hago una búsqueda con comodín en la última palabra.
  - multi_match: La versión multicampo de la consulta match.
  - match_all: La consulta más sencilla, coinciden con todos los documentos, otorgándoles un 1.0 de _score.
  - intervals: Una consulta de texto completo que permite un control detallado del orden y la proximidad de los términos coincidentes.
  - combined_fields: Coincide con varios campos como si hubieran sido indexados en un campo combinado.
  - query_string: Admite la sintaxis de cadena de consulta compacta de Lucene, lo que le permite especificar condiciones AND | OR | NOT y búsquedas en varios campos dentro de una sola cadena de consulta.
  - simple_query_string: Una versión más simple y robusta de la sintaxis query_string, adecuada para exponer directamente a los usuarios.

Consultas de texto completo

- La consulta match es la consulta estándar para realizar una búsqueda de texto completo, incluidas opciones de coincidencia aproximada. Devuelve documentos que coinciden con un texto, número, fecha o valor booleano del texto proporcionado. El texto proporcionado se analiza antes de la comparación.

```
GET blogs/_search
{
  "query": {
    "bool": {
      "should": [
        { "match": { "content": { "query": "kibana project weekly", "operator": "and" } } },
        { "match": { "content": "Welcome ny" } },
        { "match_phrase": { "title": "with kibana" } }
      ]
    }
  }
}
```

- Por defecto busca alguna de las palabra (or), el parámetro operator se puede configurar como and para contenga todas las palabras. La consulta match admite la expansión de sinónimos de varios términos: ny → (ny OR (new AND york)).
- La consulta match_phrase analiza el texto y crea una phrase (subcadena literal sin separación) que buscar.

Sintaxis de la cadena de consulta

- La propiedad query_string devuelve documentos basados en una cadena de consulta proporcionada, utilizando un analizador con una sintaxis estricta. Esta consulta utiliza una sintaxis para analizar y dividir la cadena de consulta proporcionada en función de operadores como AND o NOT. Luego, la consulta analiza cada texto dividido de forma independiente antes de devolver los documentos coincidentes. Se puede utilizar para crear una búsqueda compleja que incluya caracteres comodín, búsquedas en varios campos y más. Si bien es versátil, la consulta es estricta y devuelve un error si la cadena de consulta incluye alguna sintaxis no válida.
- La cadena de consulta se analiza en una serie de términos y operadores. Un término puede ser una sola palabra o una frase, rodeada de comillas dobles que busca todas las palabras de la frase, en el mismo orden.
- Si se necesita utilizar literalmente alguno de los caracteres utilizados como operadores se debe escaparlos con \.
- Se pueden ejecutar búsquedas con comodines en términos individuales, utilizándolos ? para reemplazar un solo carácter y * para reemplazar cero o más caracteres. Por defecto no se permiten al principio de una palabra.
- Los patrones de expresiones regulares se pueden incorporar a la cadena de consulta envolviéndolos entre /.

Sintaxis de la cadena de consulta

- Puede ejecutar consultas fuzzy (términos similares) utilizando el operador ~n: la distancia de edición (cantidad de cambios de un carácter necesarios para convertir un término en otro) predeterminada es 2, pero una distancia de edición de 1 debería ser suficiente para detectar el 80 % de todos los errores ortográficos humanos: *palabra~1*.
- La búsquedas de proximidad permite que las palabras especificadas en una frase estén más separadas o en un orden diferente utilizando el operador ~n: distancia máxima de las palabras de la frase: *"john smith"~2*.
- Se pueden especificar rangos para campos de fecha, numéricos o de cadena. Los rangos inclusivos se especifican con corchetes [min TO max] *(count:[1 TO 5])* y los rangos exclusivos con llaves {min TO max} *(tag:{alpha TO omega})*. Los rangos con un lado ilimitado pueden utilizar: *edad:>18*.
- El operador boost ^n al final hace que un término sea más relevante que otro.
- Los operadores de preferencia son + (este término **debe** estar presente) y - (este término **no debe** estar presente).
- También se admiten los operadores booleanos conocidos AND, OR y NOT (que también se escriben como &&, || y !). Se pueden agrupar varios términos o cláusulas mediante paréntesis para formar subconsultas.

Sintaxis de la cadena de consulta

- Los campos donde buscar se pueden incorpora a la propia cadena (campo:valor):
  - donde el campo status contiene active: *status:active*
  - donde el campo title contiene quick o brown: *title:(quick OR brown)*
  - donde el campo autor contiene la frase exacta "john smith": *author:"John Smith"*
  - donde el campo first name contiene Alice (se debe escapar el espacio): *first\ name:Alice*
  - donde cualquiera de los campos de book contienen quick o brown (se debe el *): *book.*:(quick OR brown)*
  - donde el campo title tiene cualquier valor no nulo: _exists_:title
  - donde son equivalentes:

```
"query": { "query_string": { "fields": [ "content", "name" ], "query": "this AND that" } }
"query": { "query_string": {  "query": "(content:this OR name:this) AND (content:that OR name:that)" } }
```

- Los espacios en blanco se consideran separadores, ni términos y ni operadores. Si la cadena de consulta está vacía o solo contiene espacios en blanco, la consulta producirá un conjunto de resultados vacío.
- La consulta query_string admite la expansión de sinónimos de varios términos con el filtro de token synonym_graph:
  - ny produce (ny OR ("new york"))

Sintaxis de la cadena de consulta

- Puede ejecutar consultas fuzzy (términos similares) utilizando el operador ~n: la distancia de edición (cantidad de cambios de un carácter necesarios para convertir un término en otro) predeterminada es 2, pero una distancia de edición de 1 debería ser suficiente para detectar el 80 % de todos los errores ortográficos humanos: *palabra~1*.
- La búsquedas de proximidad permite que las palabras especificadas en una frase estén más separadas o en un orden diferente utilizando el operador ~n: distancia máxima de las palabras de la frase: *"john smith"~2*.
- Se pueden especificar rangos para campos de fecha, numéricos o de cadena. Los rangos inclusivos se especifican con corchetes [min TO max] *(count:[1 TO 5])* y los rangos exclusivos con llaves {min TO max} *(tag:{alpha TO omega})*. Los rangos con un lado ilimitado pueden utilizar: *edad:>18*.
- El operador boost ^n al final hace que un término sea más relevante que otro.
- Los operadores de preferencia son + (este término **debe** estar presente) y - (este término **no debe** estar presente).
- También se admiten los operadores booleanos conocidos AND, ORy NOT (que también se escriben como &&, || y !). Se pueden agrupar varios términos o cláusulas mediante paréntesis.
- Los espacios en blanco se consideran separadores, ni términos y ni operadores. Si la cadena de consulta está vacía o solo contiene espacios en blanco, producirá un conjunto de resultados vacío.

Sintaxis simple de la cadena de consulta

- La propiedad simple_query_string devuelve documentos basados en una cadena de consulta proporcionada, utilizando un analizador con una sintaxis limitada pero tolerante a fallos. Esta consulta utiliza una sintaxis simple para analizar y dividir la cadena de consulta proporcionada en términos según operadores especiales. Luego, la consulta analiza cada término de forma independiente antes de devolver los documentos coincidentes. Si bien su sintaxis es más limitada que la de query_string, no devuelve errores por sintaxis no válida.
- La consulta simple_query_string admite los siguientes operadores:
  - `+` : significa operación AND
  - `|` : significa operación OR
  - `-` : niega un solo token
  - `"` : envuelve una serie de tokens para indicar una frase para buscar
  - `*` : al final de un término significa una consulta de prefijo.
  - `( )` : significar precedencia
  - `~N` : después de una palabra significa distancia (cantidad de cambios de un carácter)
  - `~N` : después de una frase significa cantidad de basura

Consultas de mezcla (JOIN)

- Realizar mezclas completas de índices al estilo SQL JOIN en un sistema distribuido como Elasticsearch es prohibitivamente costoso. En cambio, Elasticsearch ofrece dos formas de mezcla que están diseñadas para escalar horizontalmente.
- Consultas nested
  - Los documentos pueden contener campos de tipo nested. Estos campos se utilizan para indexar matrices de objetos, donde cada objeto puede consultarse (con la busqueda nested) como un documento independiente.
- Consultas has_child y has_parent
  - Puede existir una relación join de campos entre documentos dentro de un mismo índice. La consulta has_child devuelve los documentos principales cuyos documentos secundarios coinciden con la consulta especificada, mientras que la consulta has_parent devuelve los documentos secundarios cuyo documento principal coincide con la consulta especificada.

Consultas de mezcla (JOIN)

- La consulta terms es una de las formas más efectivas de unir dos índices en Elasticsearch. Esta consulta se utiliza para recuperar documentos que contienen uno o más términos exactos en un campo específico:

```
get blogs/_search
{
  "query": {
    "terms": {
       "author" : { "index" : "temas", "id" : "utilidades", "path" : "autores" }
    }
  }
}
```

- El procesador de enriquecimiento es otra herramienta potente que se puede utilizar para unir dos índices en Elasticsearch. Este procesador enriquece los datos de los documentos entrantes agregando datos de un índice de enriquecimiento predefinido al documento. Se debe crear una política de enriquecimiento donde se define qué índice utilizar para el enriquecimiento y con qué campo realizar la coincidencia. La mayor desventaja de este enfoque es que una vez enriquecido no se actualiza ni se sincroniza automáticamente los documentos creados.

Consultas geográficas y de formas

- Las consultas geográficas buscan documentos con geoformas o geopuntos:
  - geo_bounding_box: que intersecan el rectángulo especificado.
  - geo_distance: dentro de la distancia especificada de un punto central.
  - geo_grid: que intersecan el geohash especificado, el mosaico de mapa especificado o el bin H3 especificado
  - geo_polygon: que intersequen el polígono especificado.
  - geo_shape: relacionados con la geoforma especificada. Las relaciones espaciales posibles para especificar son: intersecciones, contenidos, dentro y disjuntos.
- La consultas de formas es:
  - shape: Encuentra documentos con:
    - shapes que se intersecan, están contenidos por, están dentro o no se intersecan con la forma especificada
    - points que intersecan la forma especificada

Consultas especializadas

distance_feature: calcula puntuaciones en función de las distancias calculadas dinámicamente entre el origen y los campos date, date_nanos y de los documentos geo_point. Permite omitir de manera eficiente los resultados no competitivos.

more_like_this: encuentra documentos que son similares al texto, documento o colección de documentos especificados.

percolate: busca consultas almacenadas como documentos que coinciden con el documento especificado.

rank_feature: calcula puntuaciones basadas en los valores de características numéricas y es capaz de omitir de manera eficiente los resultados no competitivos.

script: permite que un script actúe como filtro.

script_score: permite modificar la puntuación de una subconsulta con un script.

wrapper: acepta otras consultas como cadenas json o yaml.

pinned: promueve documentos seleccionados sobre otros que coinciden con una consulta determinada.

rule: admite la aplicación de reglas contextuales basadas en consultas, definidas mediante la API de reglas de consulta, a una consulta determinada.

Recuperar campos seleccionados

- De forma predeterminada, cada resultado de la respuesta de búsqueda incluye el documento _source, que es el objeto JSON completo que se proporcionó al indexar el documento.
- Para recuperar campos específicos en la respuesta de búsqueda, se utiliza el parámetro fields, que ofrece varias ventajas con respecto a la referencia directa a _source:
  - Devuelve cada valor de una manera estandarizada que coincide con su tipo de mapeo
  - Acepta campos múltiples y alias de campo.
  - Permite formatos de fechas y tipos de datos espaciales
  - Recupera valores de campos runtime
  - Devuelve los campos calculados por un script
  - Devuelve campos de índices relacionados mediante campos runtime

Recuperar campos seleccionados

- Para especificar los campos a devolver:

```
GET blogs/_search
{
  "fields": [
    "author",
    "title",
    { "field": "publish_date", "format": "epoch_millis" }
  ],
  "_source": false,
  "query": {
```

- Los hints de la respuesta incluyen la propiedad fields con los campos seleccionados. La propiedad fields de la respuesta siempre devuelve una matriz de valores para cada campo, incluso cuando hay un solo valor en el _source. Esto se debe a que Elasticsearch no tiene un tipo de matriz dedicado y cualquier campo podría contener varios valores y no garantiza que los valores de la matriz se devuelvan en un orden específico.
- Establecer el parámetro _source a false evita que se incluya el documento en el _source de la respuesta.

Ordenar resultados de búsqueda

- Por defecto Elasticsearch ordena los resultados por relevancia (_score) pero permite agregar una o más clasificaciones en campos específicos.

```
GET /my-index-000001/_search
{
  "sort" : [
    { "last_date" : {"order" : "asc", "format": "strict_date_optional_time_nanos"}},
    "name",
    { "type" : "desc" },
    {"prices" : {"order" : "asc", "mode" : "avg"}}
    "_score"
  ],
  "query" : {
```

- La opción order puede tener los valores asc y desc. El orden predeterminado es desc cuando se ordena por _score y asc cuando se ordena por cualquier otra cosa.
- La opción mode controla qué valor de matriz se selecciona para ordenar el documento al que pertenece: min, max, sum, avg, median. El modo de ordenación predeterminado en el orden ascendente es min y en descendente es max.
- La respuesta de búsqueda incluye la propiedad sort con los valores para cada documento. Si no importa el orden en el que se devuelven los documentos, debe ordenar por _doc.

Paginar resultados de búsqueda

- De forma predeterminada, las búsquedas devuelven los 10 resultados más coincidentes. Para recorrer un conjunto más amplio de resultados, se puede utilizar los parámetros from y size de la API de búsqueda. El parámetro from define la cantidad de resultados que se omitirán (0 como valor predeterminado. El parámetro size es la cantidad máxima de resultados que se devolverán. Juntos, estos dos parámetros definen una página de resultados. De forma predeterminada, no se puede utilizar from y size para recorrer más de 10.000 resultados.

```
GET /_search
{
  "from": 40,
  "size": 20,
  "query": {
```

- Se puede utilizar el parámetro search_after para recuperar la siguiente página de resultados utilizando un conjunto de valores de clasificación de la página anterior. El uso de los campos search_after requiere varias solicitudes de búsqueda con los mismos valores query y sort.

Agregaciones

- Una agregación resume datos como métricas, estadísticas u otros análisis. Elasticsearch organiza las agregaciones en tres categorías:
  - Agregaciones Metric que calculan métricas, como una suma o un promedio, a partir de valores de campo.
  - Agregaciones Bucket que agrupan documentos en contenedores según valores de campo, rangos, muestradores u otros criterios.
  - Agregaciones Pipeline que toman información de otras agregaciones en lugar de documentos o campos.
- El parámetro "aggs" permite definir las agregaciones en la API de búsqueda:

```
GET /blogs/_search?size=0
{
  "aggs": {
    "idiomas-poco-usados": { "rare_terms": { "field": "locales", "max_doc_count": 1 } },
    "cuenta-documentos-por-autor": {
      "terms": { "field": "author" },
      "aggs": {
        "url_stats": { "string_stats": { "field": "url" } }
      } } } }
```

Agregaciones

- El parámetro query de la consulta determina los documentos que participan en la agregación.
- Los resultados de la agregación aparecen en la propiedad aggregations de la respuesta.
- De forma predeterminada, las búsquedas que contienen una agregación devuelven tanto resultados de búsqueda como de agregación, con "size": 0 se evita la devolución de los documentos.
- Se pueden especificar múltiples agregaciones en la misma solicitud, por lo que hay que asignarles un nombre único.
- Las agregaciones de contenedores admiten subagregaciones de contenedores o métricas. La respuesta anida los resultados de subagregación bajo su agregación principal contenedora. No existe ningún límite de nivel o profundidad para anidar subagregaciones.

Contraer

- Al trabajar con Elasticsearch, los resultados duplicados a veces pueden ser un problema, especialmente cuando se trabaja con grandes conjuntos de datos. La función de contracción de Elasticsearch está diseñada para solucionar este problema agrupando los resultados similares y devolviendo solo el resultado más relevante de cada grupo. Utilizada para:
  - Eliminar resultados duplicados: en situaciones en las que el mismo documento aparece varias veces en los resultados de búsqueda, se puede utilizar la función de contraer para eliminar duplicados y presentar un conjunto de resultados más limpio al usuario.
  - Agrupar resultados por un campo específico: cuando desee agrupar los resultados de búsqueda por un campo en particular, como una categoría o una etiqueta, puede utilizar la función de contracción para lograrlo.
  - Agregación de datos: se puede utilizar junto con las agregaciones para proporcionar una vista más completa de los datos, como calcular la calificación promedio para cada grupo de productos.
- Para implementar la función de contracción en Elasticsearch, debe utilizar la propiedad collapse en la solicitud de búsqueda con la propiedad que especifica el campo utilizado para contraer los resultados.

```
GET blogs/_search
{
  "collapse": { "field": "author  }
```

Resaltar

- El resaltado permite obtener fragmentos destacados de uno o más campos en los resultados de búsqueda para que se pueda mostrar a los usuarios dónde se encuentran las coincidencias de la consulta. Cuando se solicita resaltados, la respuesta contiene un elemento adicional highlight para cada resultado de búsqueda que incluye los campos resaltados y los fragmentos resaltados.

```
GET blogs/_search
{
  "query": {…},
  "highlight": {
    "fields": {
      "title": {
        "pre_tags" : ["<mark>"], "post_tags" : ["</mark>"],
        "require_field_match": false
      },
      "content": {}
    }
  }
}
```


Plantillas de búsqueda

- Una plantilla de búsqueda es una búsqueda almacenada que puede ejecutar con diferentes variables. Si se utiliza Elasticsearch como backend de búsqueda, se puede pasar la información introducida por el usuario desde una barra de búsqueda como parámetros para una plantilla de búsqueda, esto permite ejecutar búsquedas sin exponer la sintaxis de consulta de Elasticsearch a los usuarios y optimizarlas. Si se usa Elasticsearch para una aplicación personalizada, las plantillas de búsqueda permiten cambiar las búsquedas, modificando las plantillas almacenadas directamente en Elasticsearch, sin modificar el código de la aplicación.
- Las plantillas de búsqueda son scripts, por lo que para crear o actualizar una plantilla de búsqueda se utiliza la Scripts API. El código del script (source) admite los mismos parámetros que el cuerpo de la solicitud de la API de búsquedas, pero también acepta la interpolación con Mustache. Las plantillas de búsqueda deben usar mustache como lang. Por lo general, las variables de Mustache se incluyen entre llaves dobles: {{my-var}} y, cuando se ejecuta la búsqueda con plantilla, Elasticsearch reemplaza las variables con los valores de params (o sus valores predeterminados). Mustache permite Url encode, concatenar listas, convertir a json o list, condicionales, …

Plantillas de búsqueda

```
PUT _scripts/search-in-content-template
{
  "script": {
    "lang": "mustache",
    "source": {
      "query": {
        "match": {
          "content": "{{query_string}}"
        }
      },
      "from": "{{from}}{{^from}}0{{/from}}", ← Con valor por defecto
      "size": "{{size}}{{^size}}10{{/size}}"
    }
  }
}
```


Plantillas de búsqueda

- Para probar una plantilla con diferentes params:

```
POST _render/template
{
  "id": "search-in-content-template",
  "params": {
    "query_string": "hello world", "from": 20, "size": 10
  }
}
```

- Para ejecutar una búsqueda con una plantilla de búsqueda:

```
GET blogs/_search/template
{
  "id": "search-in-content-template",
  "params": {
    "query_string": "hello world", "from": 0, "size": 10
  }
}
```

- Para recuperar y borrar plantillas almacenadas:

```
GET _scripts/search-in-content-template
DELETE _scripts/search-in-content-template
```

```
{
    "template_output": {
        "query": {
            "match": {
                "content": "hello world"
            }
        },
        "from": "20",
        "size": "10"
    }
}
```

Plantillas de búsqueda

- Para desarrollar y probar una plantilla antes de almacenarla:

```
POST _render/template
{
   "source": {
    "query": {
      "match": {
        "content": "{{query_string}}"
      }
    },
    "from": "{{from}}{{^from}}0{{/from}}",
    "size": "{{size}}{{^size}}10{{/size}}"
   },
  "params": {
    "query_string": "hello world", "from": 20, "size": 10
  }
}
```


Document APIs

- El API de documentos (_doc) gestiona las operaciones CRUD tanto de documentos únicos como de múltiples documentos. El cuerpo de la solicitud debe contener los nuevos datos del documento o los datos del documento actualizados.
- Con GET se recupera un documento y su fuente o campos almacenados de un índice en particular y con HEAD se verifica que un documento existe (200 ó 404). Se puede utilizar el recurso _source para recuperar solo la fuente del documento o verificar que existe. Con head ?_source=false no se recupera la fuente del documento.
  - GET <index>/_doc/<_id>
  - HEAD <index>/_doc/<_id>
  - GET <index>/_source/<_id>
  - HEAD <index>/_source/<_id>
- Se puede indexar un nuevo documento JSON con el recurso _doc o _create. El uso de _create garantiza que el documento solo se indexe si aún no existe.
  - PUT /<target>/_doc/<_id>
  - POST /<target>/_doc/
  - PUT /<target>/_create/<_id>
  - POST /<target>/_create/<_id>

Document APIs

- Si el destino es un índice, se especifica el ID del documento y el documento ya existe, la solicitud actualiza el documento e incrementa su versión.
  - PUT /<target>/_doc/<_id>
  - POST /<target>/_doc/<_id>
- Para actualiza un documento utilizando el script especificado en cuerpo de la solicitud:
  - POST /<index>/_update/<_id>
- Se utiliza DELETE para eliminar un documento de un índice, se debe especificar el ID del documento.
  - DELETE /<index>/_doc/<_id>

API Multi get (_mget)

- Permite recupera múltiples documentos JSON por ID del mismo índice o de varios índices:

```
GET /_mget
{
  "docs": [
    { "_index": "my-index-000001", "_id": "1" },
    { "_index": "my-index-000001", "_id": "2" }
  ]
}
GET /my-index-000001/_mget
{
  "docs": [ { "_id": "1" }, { "_id": "2" } ]
}
GET /my-index-000001/_mget
{
  "ids" : ["1", "2"]
}
```

Bulk API

- Realiza múltiples operaciones de indexación o eliminación en una única llamada API. Esto reduce la sobrecarga y puede aumentar considerablemente la velocidad de indexación.

```
POST _bulk
{ "index" : { "_index" : "test", "_id" : "1" } }
{ "field1" : "value1" }
{ "delete" : { "_index" : "test", "_id" : "2" } }
{ "create" : { "_index" : "test", "_id" : "3" } }
{ "field1" : "value3" }
{ "update" : {"_id" : "1", "_index" : "test"} }
{ "doc" : {"field2" : "value2"} }
POST kk/_bulk
{ "create" : {} }
{ "brand": "gucci", "color": "red", "model": "slim" }
{ "create" : {} }
{ "brand": "gucci", "color": "blue", "model": "slim" }
```


Update By Query API y Delete by query API

- Actualiza las versiones de los documentos que coinciden con la consulta especificada. Si no se especifica ninguna consulta, realiza una actualización en cada documento del flujo de datos o índice sin modificar la fuente, lo que resulta útil para detectar cambios en la asignación.

```
POST shirts/_update_by_query?conflicts=proceed
{
  "query": { "term": { "model": "other" } }
}
```

- Para eliminar los documentos que coinciden con la consulta especificada:

```
POST shirts/_delete_by_query
{
  "query": { "term": { "model": "other" } }
}
```

Reindex API

- Copia documentos de un origen a un destino. La fuente puede ser cualquier índice, alias o flujo de datos existente. El destino debe ser diferente de la fuente. Por ejemplo, no se puede volver a indexar un flujo de datos en sí mismo.

```
POST _reindex
{
  "source": {
    "index": "my-index-000001"
  },
  "dest": {
    "index": "my-new-index-000001"
  }
}
```

- Extrae el documento de origen del índice de origen e indexa los documentos en el índice de destino. Puede copiar todos los documentos en el índice de destino o volver a indexar un subconjunto de los documentos.

Parámetro de consulta refresh

- Las API Index, Update, Delete y Bulk admiten la configuración refreshpara controlar cuándo se hacen visibles para la búsqueda los cambios realizados por esta solicitud.
- Cuando el parámetro de consulta opcional refresh es true, Elasticsearch actualiza los fragmentos primarios y de réplica relevantes (no todo el índice) inmediatamente después de que se realice la operación para que esta operación sea visible para la búsqueda, si es wait_for espera a un refresco para que la operación sea visible para la búsqueda y si es false (por defecto) no hace nada con las actualizaciones, los cambios realizados por esta solicitud se harán visibles en algún momento después de que la solicitud se complete.
- A menos que se tenga una buena razón para esperar a que el cambio se haga visible, se debe utilizar siempre ?refresh=false (la configuración predeterminada). La opción más sencilla y rápida es omitir el parámetro refresh de la URL.

SQL

- Elasticsearch SQL es una función que permite ejecutar consultas similares a SQL en tiempo real en Elasticsearch y devolver resultados en formato tabular. Ya sea que se utilice la interfaz REST, la línea de comandos (elasticsearch-sql-cli) o JDBC, cualquier cliente puede utilizar SQL para buscar y agregar datos de forma nativa dentro de Elasticsearch. Se puede pensar en Elasticsearch SQL como un traductor, que comprende tanto SQL como Elasticsearch y facilita la lectura y el procesamiento de datos en tiempo real, a escala, aprovechando las capacidades de Elasticsearch.

  POST /_sql?format=txt

  { "query": "SELECT * FROM library WHERE release_date < '2000-01-01'" }
- Si bien SQL y Elasticsearch tienen términos diferentes para la forma en que se organizan los datos (y semántica diferente), un modelo relacional frente a un modelo jerarquico, esencialmente su propósito es el mismo y tienen cierta correspondencia:
  - Índice → Tabla
  - Documento → Fila
  - Campo → Columna

EQL

- Event Query Language (EQL) es un lenguaje de consulta para datos de series de tiempo basados en eventos, como registros, métricas y seguimientos.
- Las ventajas de EQL son:
  - EQL permite expresar relaciones entre eventos.
  - Muchos lenguajes de consulta permiten hacer coincidir eventos individuales pero EQL permite hacer coincidir una secuencia de eventos en diferentes categorías de eventos y períodos de tiempo.
  - EQL tiene una curva de aprendizaje baja.
  - La sintaxis de EQL se parece a la de otros lenguajes de consulta comunes, como SQL. EQL permite escribir y leer consultas de manera intuitiva, lo que permite realizar búsquedas rápidas e iterativas.
  - EQL está diseñado para casos de uso de seguridad.
  - Si bien se puede usar para cualquier dato basado en eventos, EQL se creo para la búsqueda de amenazas. EQL no solo admite búsquedas de indicadores de compromiso (IOC), sino que también puede describir actividades que van más allá de los IOC

  sequence with maxspan=1d

  [process where process.name == "cmd.exe"]

  ![process where stringContains(process.command_line, "ocx")]

  [file where stringContains(file.name, "scrobj.dll")]

ES|QL

- El lenguaje de consulta Elasticsearch (ES|QL) ofrece una forma eficaz de filtrar, transformar y analizar datos almacenados en Elasticsearch y, en el futuro, en otros entornos de ejecución. Está diseñado para que sea fácil de aprender y usar por parte de usuarios finales, equipos de SRE, desarrolladores de aplicaciones y administradores.
- Los usuarios pueden crear consultas ES|QL para buscar eventos específicos, realizar análisis estadísticos y generar visualizaciones. Admite una amplia gama de comandos y funciones que permiten a los usuarios realizar diversas operaciones de datos, como filtrado, agregación, análisis de series temporales y más.
- El lenguaje de consulta Elasticsearch (ES|QL) utiliza "pipes" (|) para manipular y transformar datos paso a paso. Este enfoque permite a los usuarios componer una serie de operaciones, donde el resultado de una operación se convierte en la entrada de la siguiente, lo que permite realizar transformaciones y análisis de datos complejos.

```
FROM logs-*
| WHERE event.code IS NOT NULL
| STATS event_code_count = COUNT(event.code) BY event.code,host.name
| ENRICH win_events ON event.code WITH event_description
| WHERE event_description IS NOT NULL and host.name IS NOT NULL
| RENAME event_description AS event.description
| SORT event_code_count DESC
| KEEP event_code_count,event.code,host.name,event.description
```


Kibana Query Language

- El lenguaje de consulta Kibana (KQL) es un lenguaje de consulta simple basado en texto para filtrar datos: solo filtra datos y no tiene ninguna función en la agregación, transformación u ordenación de datos y no debe confundirse con el lenguaje de consulta Lucene, que tiene un conjunto de características diferente.
- Se utiliza KQL en las Data View para filtrar documentos donde:
  - existe un valor para un campo: *mi_campo: **
  - coincide con un valor determinado: *mi_campo: un_valor*
    - En campos keyword, numéricos, de fecha o booleanos, el valor debe ser una coincidencia exacta, incluida la puntuación y el uso de mayúsculas y minúsculas. Pueden utilizar comodín * (cero o más caracteres), no se permiten los comodines iniciales por razones de rendimiento (query:allowLeadingWildcards en configuración avanzada).
    - En campos text, el orden de los términos de búsqueda no importa, para buscar los términos en el orden indicado se encierre el valor entre comillas.
    - Se deben escapar con \ los caracteres: \():<>"*
  - está dentro de un rango determinado (<,<=,>,>=): *mi_campo < un_valor*
- Para negar o excluir un conjunto de documentos se utiliza la palabra clave not, para combinar varias consultas, and y or, usando paréntesis para especificar la precedencia al combinar, se puede utilizar el * en los nombres de campos (mismo tipo), las {} delimitan los valores de los campos anidados. La palabras clave no distinguen entre mayúsculas y minúsculas.

INGESTA DE DATOS

Logstash

- Logstash es un motor de recopilación de datos de código abierto con capacidades de canalización en tiempo real. Logstash puede unificar dinámicamente datos de distintas fuentes y normalizarlos en destinos de su elección. Si bien Logstash impulsó originalmente la innovación en la recopilación de registros (logs), sus capacidades se extienden mucho más allá de ese caso de uso. Cualquier tipo de evento se puede enriquecer y transformar con una amplia gama de complementos de entrada, filtro y salida, y muchos códecs nativos simplifican aún más el proceso de ingesta. Logstash acelera la obtención de información al aprovechar un mayor volumen y variedad de datos.
- Logstash es un procesador de datos capaz de recoger, transformar y enviar (ETL) datos a un "stash" como Elasticsearch. Es extremadamente útil en la ingesta de datos, especialmente cuando provienen de múltiples fuentes y formatos, proporcionando una amplia variedad de input plugins para facilitar este proceso. Logstash permite transformar los datos antes de almacenarlos, lo cual es crucial para asegurar que el formato de los datos sea el adecuado para su análisis posterior.
- Entre sus características principales destacan:
  - Flexibilidad en la recolección de datos: Logstash puede recibir datos de múltiples fuentes simultáneamente, adaptándose a casi cualquier tipo de entrada de datos.
  - Filtrado y transformación poderosos: A través de su variedad de filtros, Logstash permite modificar y transformar los datos antes de enviarlos a su destino final, lo que facilita la adaptación de los datos a las necesidades específicas de almacenamiento y análisis.
  - Pluriformidad en la salida de datos: Logstash no sólo se limita a enviar datos a Elasticsearch, sino que también puede exportarlos a otros destinos, ofreciendo una gran versatilidad en la gestión de salidas.

Pipelines

- El flujo de procesamiento de eventos de Logstash tiene tres etapas: entradas → filtros → salidas. Las entradas generan eventos, los filtros los modifican y las salidas los envían a otra parte. Las entradas y las salidas admiten códecs que permiten codificar o decodificar los datos a medida que ingresan o salen del flujo de procesamiento sin tener que usar un filtro independiente. Los complementos de entrada consumen datos de una fuente, los complementos de filtro modifican los datos según lo especifique y los complementos de salida escriben los datos en un destino.
- La canalización de procesamiento de eventos de Logstash coordina la ejecución de entradas, filtros y salidas.
- Cada etapa de entrada en la canalización de Logstash se ejecuta en su propio subproceso. Las entradas escriben eventos en una cola central que se encuentra en la memoria (predeterminado) o en el disco. Cada subproceso de trabajo de la canalización toma un lote de eventos de esta cola, ejecuta el lote de eventos a través de los filtros configurados y, luego, ejecuta los eventos filtrados a través de cualquier salida. El tamaño del lote y la cantidad de subprocesos de trabajo de la canalización son configurables. De forma predeterminada, Logstash utiliza colas limitadas en memoria entre las etapas de la canalización (entrada → filtro y filtro → salida) para almacenar en búfer los eventos. Si Logstash finaliza de forma insegura, se perderán todos los eventos almacenados en la memoria. Para ayudar a evitar la pérdida de datos, puede habilitar Logstash para que conserve los eventos en curso en el disco (Colas persistentes (PQ)).

Configuración y ejecución

- Logstash tiene dos tipos de archivos de configuración: archivos de configuración de canalización (pipelines) y archivos de configuración.
- Los archivos de configuración de la canalización, que definen la canalización de procesamiento de Logstash, se crean cuando se definen las etapas de la canalización de procesamiento de Logstash. En deb y rpm, se colocan en el directorio /etc/logstash/conf.d, y en Docker en /usr/share/logstash/pipeline/. Logstash intenta ejecutar todos los archivos con extensión .conf y no tiene en cuenta los demás archivos. Para ejecutar Logstash desde la línea de comandos con un pipeline:

  ```
  logstash -f logstash.conf
  ```

- Los archivos de configuración, que especifican opciones que controlan el inicio y la ejecución de Logstash, ya están definidos en la instalación de Logstash. Logstash incluye los siguientes archivos de configuración:
  - logstash.yml: Contiene indicadores de configuración de Logstash. Puede configurar indicadores en este archivo en lugar de pasarlos en la línea de comandos. Cualquier indicador que configure en la línea de comandos anula la configuración correspondiente en el archivo logstash.yml.
  - pipelines.yml: Contiene el marco y las instrucciones para ejecutar múltiples pipelines en una única instancia de Logstash.
  - jvm.options: Contiene indicadores de configuración de JVM. Utilice este archivo para establecer los valores inicial y máximo para el espacio total del montón. También puede utilizar este archivo para establecer la configuración regional de Logstash. Especifique cada indicador en una línea independiente. Todas las demás configuraciones de este archivo se consideran configuraciones expertas.
  - log4j2.properties: Contiene la configuración predeterminada de log4j 2la biblioteca.

Múltiples pipelines

- Por defecto, Logstash es un proceso dedicado que ejecuta una única canalización que puede estar configurada en múltiples ficheros.
- Si se necesita ejecutar más de una canalización en el mismo proceso, Logstash ofrece una forma de hacerlo a través de un archivo de configuración llamado pipelines.yml. Este archivo debe ubicarse en la carpeta path.settings y sigue esta estructura:

```
- pipeline.id: my-pipeline_1
  path.config: "/etc/path/to/p1.config"
  pipeline.workers: 3
- pipeline.id: my-other-pipeline
  path.config: "/etc/different/path/p2.cfg"
  queue.type: persisted
```

- Este archivo tiene formato YAML y contiene una lista de diccionarios, donde cada diccionario describe una canalización y cada par clave/valor especifica una configuración para esa canalización.
- El uso de múltiples pipelines es especialmente útil si la configuración actual tiene flujos de eventos que no comparten las mismas entradas/filtros y salidas y se separan entre sí mediante etiquetas y condicionales.
- Tener múltiples pipelines en una sola instancia también permite que estos flujos de eventos tengan diferentes parámetros de rendimiento y durabilidad (por ejemplo, diferentes configuraciones para los trabajadores de pipeline y las colas persistentes). Esta separación significa que una salida bloqueada en un pipeline no ejercerá contrapresión en el otro.

Estructura de una canalización

- Se puede crear una canalización uniendo complementos (entradas, salidas, filtros y, a veces, códecs) para procesar datos. Para crear una canalización de Logstash, se crea un archivo de configuración (.conf) para especificar qué complementos se desean utilizar y las configuraciones para cada complemento.
- Una canalización muy básica puede contener solo una entrada y una salida. La mayoría de las canalizaciones incluyen al menos un complemento de filtro porque ahí es donde ocurre la parte de "transformación" de la magia ETL (extracción, transformación, carga). Puede hacer referencia a campos de eventos en una canalización y usar condicionales para procesar eventos cuando cumplan con ciertos criterios.
- El pipeline tiene una sección independiente para cada tipo de complemento que desee agregar a la canalización de procesamiento de eventos. Cada sección contiene opciones de configuración para uno o más complementos. Si especifica varios filtros, se aplican en el orden en que aparecen en el archivo de configuración. Si especifica varias salidas, los eventos se envían a cada destino de forma secuencial, en el orden en que aparecen en el archivo de configuración. La configuración de un complemento consta del nombre del complemento seguido de un bloque de configuraciones para ese complemento.
- Un comentario comienza con el carácter # y no es necesario que esté al principio de una línea

Estructura de una canalización

```
input {
  file {
    path => "/tmp/access_log"
    start_position => "beginning"
  }
}
filter {
  if [path] =~ "access" {
    mutate { replace => { "type" => "apache_access" } }
    grok {
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
  }
  date {
    match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}
output {
  elasticsearch {
    hosts => ["localhost:9200"]
  }
  stdout { codec => rubydebug }
}
```


Entradas

- Para obtener datos en Logstash, se utilizan entradas. Algunas de las entradas más utilizadas son:
  - file: lee desde un archivo en el sistema de archivos, de forma muy similar al comando UNIX tail -0F
  - syslog: escucha en el puerto conocido 514 los mensajes de syslog y los analiza según el formato RFC3164
  - redis: lee desde un servidor redis, utilizando tanto los canales como las listas de redis. Redis se utiliza a menudo como "agente" en una instalación centralizada de Logstash, que pone en cola los eventos de Logstash de los "transportadores" remotos de Logstash.
  - beats: procesa eventos enviados por Beats.

Campos de eventos

- Todos los eventos tienen propiedades, Logstash llama a estas propiedades "campos".
- Algunas opciones de configuración de Logstash requieren la existencia de campos para funcionar. Debido a que las entradas generan eventos, no hay campos para evaluar dentro del bloque de entrada, no funcionan en los bloques de entrada porque aún no existen.
- La sintaxis básica para acceder a un campo es [fieldname]. Si se hace referencia a un campo de nivel superior, se puede omitir el [] y simplemente utilizar fieldname. Para hacer referencia a un campo anidado, se especifica la ruta completa a ese campo: [top-level field][nested field].
- El formato de referencia de campo también se utiliza en lo que Logstash llama formato sprintf. Este formato le permite incrustar valores de campo en otras cadenas.

  increment => "apache.%{[response][status]}"
- De manera similar, se puede convertir la marca de tiempo UTC en el campo @timestamp en una cadena. En lugar de especificar un nombre de campo dentro de las llaves, se utiliza la sintaxis %{{FORMAT}} donde FORMAT es un formato de tiempo Java.

  index => "log-%{[@metadata][beat]}-%{+YYYY.MM.dd}"

Campos reservados en eventos de Logstash

- Algunos campos de los eventos de Logstash están reservados o deben cumplir con una determinada forma. El uso de estos campos puede provocar excepciones en tiempo de ejecución cuando la API de eventos o los complementos encuentran valores incompatibles.
  - @metadata
    - Un mapa clave/valor.
    - API de complemento basada en Ruby: el valor es un org.jruby.RubyHash.
    - API de complemento basada en Java: el valor es un org.logstash.ConvertedMap.
    - En forma serializada (como JSON): un mapa clave/valor donde las claves deben ser cadenas y los valores no están restringidos a un tipo particular.
  - @timestamp
    - Un objeto que contiene la representación de un momento específico en el tiempo.
    - API de complemento basada en Ruby: el valor es org.jruby.RubyTime.
    - API de complemento basada en Java: el valor es java.time.Instant.
    - En formato serializado (como JSON) o cuando se configura con Event#set: es aceptable un valor de cadena compatible con ISO8601.
  - @version
    - Una cadena que contiene un valor entero.
  - tags
    - Una matriz de cadenas distintas

Filtros

- Los filtros son complementos de procesamiento intermedios en la secuencia de comandos de Logstash. Puede combinar filtros con condicionales para realizar una acción en un evento si cumple con ciertos criterios. Algunos filtros útiles incluyen:
  - grok : analiza y estructura texto arbitrario. Actualmente, Grok es la mejor manera en Logstash de analizar datos de registro no estructurados y convertirlos en algo estructurado y consultable. Con 120 patrones integrados en Logstash, es más que probable que encuentres uno que satisfaga tus necesidades.
  - mutate : realiza transformaciones generales en campos de eventos. Puede cambiar el nombre, eliminar, reemplazar y modificar campos en sus eventos.
  - drop: descarta un evento por completo, por ejemplo, depurar eventos.
  - clone: hacer una copia de un evento, posiblemente añadiendo o quitando campos.
  - geoip: agrega información sobre la ubicación geográfica de las direcciones IP (¡también muestra gráficos increíbles en Kibana!)

Condicionales

- A veces, se desea filtrar o generar un evento solo bajo ciertas condiciones. Para eso, puede usar un condicional.
- Los condicionales en Logstash tienen el mismo aspecto y funcionan de la misma manera que en los lenguajes de programación. Los condicionales admiten las instrucciones *if*, *else if* y *else*, y pueden estar anidados.

```
if EXPRESSION {
  …
} else if EXPRESSION {
  …
} else {
  …
}
```

- Las expresiones pueden ser largas y complejas. Pueden contener otras expresiones, se pueden negar con ! y se pueden agrupar con paréntesis (...).

Condicionales

- Los operadores de comparación son:
  - relacionales: ==, !=, <, >, <=,>=
  - expresiones regulare: =~, !~ (comprueba un patrón a la derecha delimitado por / contra un valor de cadena a la izquierda)
  - inclusión: in, not in
- Los operadores booleanos admitidos son:
  - and, or, nand, xor
- Los operadores unarios admitidos son:
  - !
- Se puede comprobar la existencia de un campo específico, pero actualmente no hay forma de diferenciar entre un campo que no existe y un campo que es simplemente falso. La expresión if [foo] se devuelve false cuando:
  - [foo] no existe en el evento,
  - [foo] existe en el evento, pero es falso, o
  - [foo] existe en el evento, pero es nulo

Salidas

- Las salidas son la fase final del flujo de trabajo de Logstash. Un evento puede pasar por varias salidas, pero una vez que se completa todo el procesamiento de salida, el evento ha finalizado su ejecución. Algunas salidas que se utilizan comúnmente son:
  - elasticsearch: envía datos de eventos a un índice de Elasticsearch.
  - file: escribe datos de eventos en un archivo en el disco.
  - csv: escribe eventos en el disco en un formato delimitado.
  - exec: ejecuta un comando para un evento coincidente.
  - http: envía eventos a un punto final HTTP o HTTPS genérico.
  - graphite: envía datos de eventos a Graphite, una popular herramienta de código abierto para almacenar y graficar métricas.
  - statsd: envía datos de eventos a statsd (salida estandar), un servicio que "escucha estadísticas, como contadores y temporizadores, enviadas a través de UDP y envía agregados a uno o más servicios backend conectables".

El campo @metadata

- En Logstash, hay un campo especial llamado @metadata. El contenido de @metadata no forma parte de ninguno de los eventos en el momento de la salida, lo que lo hace ideal para usar condicionales o para ampliar y crear campos de eventos con referencias de campo y formato sprintf.
- Se utiliza este campo @metadata cada vez que se necesita un campo temporal que no esté en el resultado final.

```
filter {
  if "%{+HH}:%{+mm}" < "08:00" {
    mutate { add_field => { "[@metadata][show]" => true } }
  }
}

output {
  if [@metadata][show] {  stdout { codec => rubydebug }  }
}
```

- Quizás uno de los casos de uso más comunes para este nuevo campo es con el filtro date y tener una marca de tiempo temporal.

```
filter {
  grok { match => [ "message", "%{HTTPDATE:[@metadata][timestamp]}" ] }
  date { match => [ "[@metadata][timestamp]", "dd/MMM/yyyy:HH:mm:ss Z" ] }
}
```


Variables de entorno

- Se puede establecer referencias de variables de entorno en la configuración de los complementos de Logstash mediante ${var}.

```
input { tcp { port => "${TCP_PORT}" } }
```

- Al iniciar Logstash, cada referencia se reemplaza por el valor de la variable de entorno. El reemplazo distingue entre mayúsculas y minúsculas. Las variables de entorno son inmutables, si se actualiza la variable de entorno, se deberá reiniciar Logstash para que se aplique el valor actualizado.
- Las referencias a variables no definidas generan un error de configuración de Logstash. Se puede proporcionar un valor predeterminado mediante el formato ${var:default value}. Logstash utiliza el valor predeterminado si la variable de entorno no está definida.

```
input { tcp { port => "${TCP_PORT:8080}" } }
```

- Se puede agregar referencias de variables de entorno en cualquier tipo de opción de complemento: cadena, número, booleano, matriz o hash.
- Las variables de entorno para parámetros URI de tipo lista pueden admitir listas de valores delimitados por espacios.

Códecs

- Los códecs son básicamente codificadores/decodificadores de flujo que pueden funcionar como parte de una entrada o salida. Los códecs le permiten separar fácilmente el transporte de sus mensajes del proceso de serialización. Los códecs más populares son:
  - json: codifica o decodifica datos en formato JSON.
  - csv: codifica o decodifica datos en un formato delimitado.
  - dot: envía 1 punto por evento a stdout para realizar el seguimiento.
  - multiline: fusiona eventos de texto de varias líneas, como excepciones de Java y mensajes de seguimiento de pila, en un solo evento.
  - plain: lee texto simple sin delimitación entre evento
  - msgpack: lee el contenido codificado de MessagePack (formato de serialización binaria que busca ser ultra eficiente tanto en términos de tamaño como de velocidad)

Gestión de eventos multilínea

- Por defecto cada línea genera un evento, pero hay casos de uso que los eventos que abarcan varias líneas de texto. Para gestionar correctamente estos eventos de varias líneas, Logstash necesita saber cómo identificar qué líneas forman parte de un único evento.
- El procesamiento de eventos multilínea es complejo y depende de un orden adecuado de los eventos. La mejor manera de garantizar un procesamiento ordenado de los registros es implementar el procesamiento lo antes posible en el flujo de trabajo.
- El códec multiline es la herramienta preferida para gestionar eventos multilínea en la secuencia de comandos de Logstash. El códec multiline fusiona líneas de una única entrada mediante un conjunto simple de reglas.
- Los aspectos más importantes de la configuración del códec multiline son los siguientes:
  - La opción pattern especifica una expresión regular. Las líneas que coinciden con la expresión regular especificada se consideran continuaciones de una línea anterior o el inicio de un nuevo evento de varias líneas. Se pueden utilizar plantillas de expresión regular de grok con esta opción de configuración.
  - La opción what toma uno de los dos valores: previous o next. El valor previous especifica que las líneas que coinciden con el valor de la opción pattern son parte de la línea anterior. El valor next especifica que las líneas que coinciden con el valor de la opción pattern son parte de la línea siguiente.
  - La opción negate a "true" aplica el códec multiline a las líneas que no coinciden con la expresión regular especificada en la opción pattern.

Gestión de eventos multilínea

- Los seguimientos de pila de Java constan de varias líneas, y cada línea después de la línea inicial comienza con un espacio en blanco:

```
codec => multiline {
    pattern => "^\s"
    what => "previous"
}
```

- Los registros de actividad de los servicios generalmente comienzan con una marca de tiempo, seguida de información sobre la actividad específica:

```
codec => multiline {
    pattern => "^%{TIMESTAMP_ISO8601} "
    negate => true
    what => previous
}
```


Administrar complementos

- Logstash cuenta con una amplia colección de complementos de entrada, filtro, códec y salida. Por defecto, los paquetes de lanzamiento de Logstash incluyen complementos comunes.
- Los complementos están disponibles en paquetes independientes llamados gemas (gems) y alojados en RubyGems.org. Se utiliza el script de administrador de complementos bin/logstash-plugin para administrarlos.
- Para enumerar los complementos que se encuentran actualmente disponibles en su implementación:

```
logstash-plugin list
```

- Para recuperar complementos alojados en el repositorio público RubyGems.org e instalarlos sobre la instalación local de Logstash:

```
logstash-plugin install logstash-codec-csv
```

- Para actualizar todos los complementos instalados o solo el que especifique:

```
logstash-plugin update
logstash-plugin update logstash-codec-csv
```

- Para eliminar un complemento instalado:

```
logstash-plugin remove logstash-codec-csv
```

Transformar datos

- Realizar mutaciones generales en los campos. Se puede cambiar el nombre, eliminar, reemplazar y modificar campos en los eventos.

```
mutate {
  rename => { "HOSTORIP" => "client_ip" }
  strip => ["ident", "auth"]
  split => { "hostname" => "." }
  add_field => { "shortHostname" => "%{[hostname][0]}" }
  capitalize => [ "name" ]
}
```

- Las mutaciones disponibles son: coerce, rename, update, replace, convert, gsub, uppercase, capitalize, lowercase, strip, split, join, merge, copy.
- Omitir eventos, se eliminan de la salida.

```
if [loglevel] == "debug" {
  drop { }
}
```


Transformar datos

- Analiza las fechas de los campos para usarlas como marcas de tiempo de Logstash para eventos.

```
date {
  match => [ "logdate", "MMM dd yyyy HH:mm:ss" ]
}
```

- El complemento de filtro de DNS realiza una búsqueda de DNS estándar o inversa.

```
dns {
  reverse => [ "source_host" ]
  action => "replace"
}
```

- El filtro geoip agrega información geográfica sobre la ubicación de las direcciones IP.

```
geoip {
  source => "clientip"
  target => "clientgeo"
}
```

- El filtro de agente de usuario analiza las cadenas del agente de usuario en campos separados.

```
useragent {
  source => "agent"
  target => "user_agent"
}
```

Transformar datos

- Analiza los datos de valores separados por comas en campos individuales. De forma predeterminada, el filtro genera automáticamente los nombres de los campos (column1, column2, etc.) o puedes especificar una lista de nombres. También puedes cambiar el separador de columnas.

```
filter {
  csv {
    separator => ","
    columns => [ "Transaction Number", "Date", "Description", "Amount Debit", "Amount Credit", "Balance" ]
  }
```

- Decodifica (a través de entradas) y codifica (a través de salidas) contenido con formato JSON, creando un evento por elemento en una matriz JSON.

```
input {
  file {
    path => "/path/to/myfile.json"
    codec =>"json"
  }
```

- Convierte una cadena JSON en campos:

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


Transformar datos

- Extrae datos de eventos no estructurados en campos mediante el uso de delimitadores. El filtro de disección no utiliza expresiones regulares y es muy rápido. Sin embargo, si la estructura de los datos varía de una línea a otra, el filtro grok es más adecuado.

```
dissect {
  mapping => { "message" => "%{ts} %{+ts} %{+ts} %{src} %{prog}[%{pid}]: %{msg}" }
}
```

- Analiza pares clave-valor, como key=value:

```
kv { }
```

Transformar datos

- Grok es una excelente manera de analizar datos de registro no estructurados y convertirlos en algo estructurado y consultable. Grok funciona combinando patrones de texto (RegEx) en algo que coincide con sus registros.
- La sintaxis de un patrón grok es %{SYNTAX:SEMANTIC}. Donde SYNTAX es el nombre predefinido del patrón que coincidirá con el texto y SEMANTIC es el identificador a dar al fragmento de texto encontrado. Opcionalmente, se puede agregar una conversión de tipo de datos aunque las únicas admitidas son int y float.

  %{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes:int} %{NUMBER:duration:float}
- Grok se basa en expresiones regulares, por lo que cualquier expresión regular también es válida en grok. A veces, Logstash no tiene el patrón necesitado, se puede usar la sintaxis Oniguruma (?<*SEMANTIC*>*regex*) que permitirá hacer coincidir una expresión regular y guardarlo como un campo:

  (?<nif>\d{1,8}[A-Za-z])
- Alternativamente, se puede crear un archivo de patrones personalizados.

Transformar datos

- Hay disponibles SYNTAX, como COMMONAPACHELOG, COMBINEDAPACHELOG o TOMCATLOG con patrones para líneas completas.
- La siguiente configuración analiza el mensaje en campos:

```
grok {
   match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration} %{GREEDYDATA:message}" }
   overwrite => [ "message" ]
  }
```

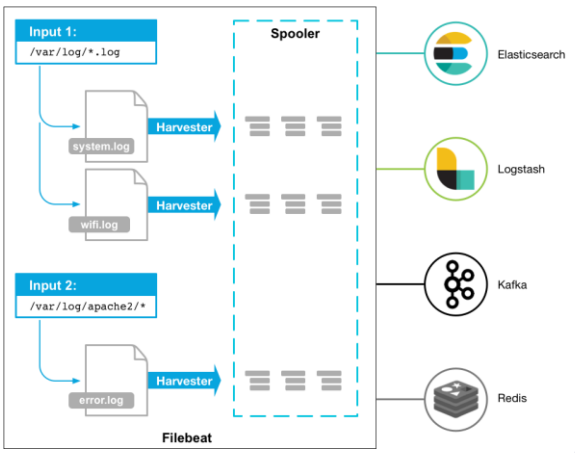
- Si se necesita hacer coincidir varios patrones con un solo campo, el valor puede ser una matriz de patrones. Si un patrón depende de un campo creado por un patrón anterior, es necesario usar filtros grok separados
- overwrite permite sobrescribir un valor en un campo que ya existe.
- Si se necesita ayuda para crear patrones de grok, se puede utilizar el depurador de Grok de las Dev Tools en Kibana. El depurador de Grok es una función de X-Pack con licencia básica y, por lo tanto, su uso es gratuito.

Beats

- Los Beats son aplicaciones livianas de envío de datos de código abierto que se instalan como agentes en los servidores para enviar datos operativos a Elasticsearch.
- Un Beats puede enviar datos directamente a Elasticsearch, Redis y Kafka o a través de Logstash, donde puede procesarlos y mejorarlos aún más, antes de visualizarlos en Kibana.
- Los Beats disponibles son:
  - Filebeat: Agente ligero para logs y otros datos
  - Metricbeat: Agente ligero para datos de métricas
  - Heartbeat: Agente ligero para monitoreo de tiempo de actividad
  - Auditbeat: Agente ligero para información de auditoría
  - Winlogbeat: Agente ligero para logs de eventos de Windows
  - Packetbeat: Agente ligero para datos de red

Filebeat

- Filebeat es una aplicación liviana para reenviar y centralizar datos de registro almacenados en ficheros. Filebeat, que se instala como agente en sus servidores, monitorea los archivos de registro o las ubicaciones que usted especifica, recopila eventos de registro y los reenvía a Elasticsearch o Logstash para indexarlos.
- Así es como funciona Filebeat: cuando inicia Filebeat, inicia una o más entradas que buscan en las ubicaciones que especificó para los datos de registro. Por cada registro que Filebeat localiza, inicia un recolector. Cada recolector lee un solo registro en busca de contenido nuevo y envía los nuevos datos de registro a libbeat, que agrega los eventos y envía los datos agregados a la salida que configuró para Filebeat.



Metricbeat

- Metricbeat es una aplicación liviana que se puede instalar en los servidores para recopilar métricas periódicas del sistema operativo y de los servicios que se ejecutan en el servidor. Metricbeat toma las métricas y estadísticas que recopila y las envía a la salida que especifique, como Elasticsearch o Logstash.
- Metricbeat ayuda a supervisar los servidores mediante la recopilación de métricas del sistema y los servicios que se ejecutan en el servidor, como:
  - Apache
  - HAProxy
  - MongoDB
  - MySQL
  - Nginx
  - PostgreSQL
  - Redis
  - System
  - Zookeeper

Heartbeat

- Heartbeat es un servicio liviano que se instala en un servidor remoto para verificar periódicamente el estado de sus servicios y determinar si están disponibles. A diferencia de Metricbeat, que solo le indica si los servidores están activos o inactivos, Heartbeat indica si los servicios están disponibles.
- Heartbeat es útil cuando se necesita verificar el cumplimiento de los acuerdos de nivel de servicio en cuanto al tiempo de actividad del servicio. También es útil para otros escenarios, como casos de uso de seguridad, cuando necesitas verificar que nadie externo pueda acceder a los servicios en tu servidor empresarial privado.
- Se puede configurar Heartbeat para que haga ping a todas las direcciones IP que se puedan resolver mediante DNS para un nombre de host específico. De esa manera, se puede verificar todos los servicios que tienen balanceo de carga para ver si están disponibles.

Auditbeat

- Auditbeat es un servicio ligero que puede instalar en sus servidores para auditar las actividades de los usuarios y los procesos en sus sistemas.
- Por ejemplo, se puede utilizar Auditbeat para recopilar y centralizar eventos de auditoría desde Linux Audit Framework.
- También se puede utilizar Auditbeat para detectar cambios en archivos críticos, como archivos binarios y de configuración, e identificar posibles violaciones de las políticas de seguridad.

Winlogbeat

- Winlogbeat envía registros de eventos de Windows a Elasticsearch o Logstash. Se puede instalar como un servicio de Windows.
- Winlogbeat lee uno o más registros de eventos mediante las API de Windows, filtra los eventos según criterios configurados por el usuario y luego envía los datos de los eventos a las salidas configuradas (Elasticsearch o Logstash). Winlogbeat supervisa los registros de eventos para que los nuevos datos de eventos se envíen de manera oportuna. La posición de lectura de cada registro de eventos se conserva en el disco para permitir que Winlogbeat se reanude después de los reinicios.
- Winlogbeat puede capturar datos de eventos de cualquier registro de eventos que se ejecute en su sistema. Por ejemplo, puede capturar eventos como:
  - eventos de aplicación
  - eventos de hardware
  - eventos de seguridad
  - eventos del sistema

Packetbeat

- Packetbeat es un analizador de paquetes de red en tiempo real que puede utilizar con Elasticsearch para proporcionar un sistema de análisis de rendimiento y monitoreo de aplicaciones. Packetbeat completa la plataforma Beats al brindar visibilidad entre los servidores de su red.
- Packetbeat funciona capturando el tráfico de red entre sus servidores de aplicaciones, decodificando los protocolos de la capa de aplicación (HTTP, MySQL, Redis, etc.), correlacionando las solicitudes con las respuestas y registrando los campos interesantes para cada transacción.
- Packetbeat puede ayudar a detectar fácilmente problemas con su aplicación back-end, como errores o problemas de rendimiento, y hace que su resolución (y, por lo tanto, su reparación) sea mucho más rápida.
- Después de decodificar los mensajes de la capa de aplicación (OSI), Packetbeat correlaciona las solicitudes con las respuestas en lo que llamamos transacciones. Packetbeat rastrea el tráfico entre sus servidores, analiza los protocolos de nivel de aplicación sobre la marcha y correlaciona los mensajes en transacciones.
- Packetbeat puede ejecutarse en los mismos servidores que los procesos de su aplicación o en sus propios servidores. Cuando se ejecuta en servidores dedicados, Packetbeat puede obtener el tráfico de los puertos espejo del conmutador o de dispositivos de interceptación. En este tipo de implementación, no hay ninguna sobrecarga en la aplicación monitoreada.

Elastic Agent

- Ahora Elastic ofrece dos formas principales de enviar datos a Elasticsearch:
  - Los Beats son aplicaciones livianas que envían datos operativos a Elasticsearch. Elastic proporciona Beats separados para diferentes tipos de datos, como registros, métricas y tiempo de actividad. Según los datos que desee recopilar, es posible que deba instalar varios transportadores en un solo host.
  - Elastic Agent es un agente único para registros, métricas, datos de seguridad y prevención de amenazas. Elastic Agent se puede implementar en dos modos diferentes:
    - Administrado por Fleet : las políticas y el ciclo de vida de Elastic Agent se administran de forma centralizada mediante la aplicación Fleet en Kibana. La aplicación Integraciones también permite agregar integraciones de forma centralizada con otros servicios y sistemas populares. Esta es la opción recomendada para la mayoría de los usuarios.
    - Modo independiente: todas las políticas se aplican al Elastic Agent de forma manual con un archivo YAML. Esta opción está pensada para usuarios más avanzados.
- Elastic Agent es un binario único diseñado para brindar la misma funcionalidad que brindan los distintos Beats en la actualidad, sin embargo todavía no se ha alcanzado la paridad de características.

Integraciones

- Las Elastic Integrations brindan una manera sencilla de conectar Elastic a servicios y sistemas externos, y obtener información o tomar medidas rápidamente. Pueden recopilar nuevas fuentes de datos y, a menudo, se envían con recursos listos para usar, como paneles, visualizaciones y canales para extraer campos estructurados de registros y eventos. Esto facilita la obtención de información en segundos.
- Estas integraciones suelen usar configuraciones, dashboards, visualizaciones y pipelines prediseñados para ayudar a darle sentido a las métricas, los logs y los eventos.
- Las integraciones están disponibles para servicios y plataformas populares como Nginx o AWS, así como para muchos tipos de entrada genéricos, como archivos de registro.
- Kibana ofrece una interfaz de usuario basada en la web para agregar y administrar integraciones.

Elastic Agent

- Elastic Agent es una forma única y unificada de agregar monitoreo de registros, métricas y otros tipos de datos a un host. También puede proteger a los hosts de amenazas de seguridad, consultar datos de sistemas operativos, reenviar datos de servicios remotos o hardware, y más. Un solo agente facilita y agiliza la implementación del monitoreo en toda su infraestructura. Cada agente tiene una sola política que puede actualizar para agregar integraciones para nuevas fuentes de datos, protecciones de seguridad y más.
- Las políticas son recopilaciones de configuraciones e integraciones que definen cómo funcionará un Elastic Agent. Se pueden asignar varias integraciones a una política de Agent que brinden flexibilidad respecto a lo que pueden capturar los agentes de datos. Asignar una política de Elastic Agente a varios agentes te permite gestionar y configurar muchos agentes a mayor escala con el uso de Fleet.
- Los datos recopilados por Elastic Agent se almacenan en índices que son más granulares que los que se obtienen de forma predeterminada con los proveedores de Beats o APM Server. Esto brinda más visibilidad sobre las fuentes del volumen de datos y control sobre las políticas de administración del ciclo de vida y los permisos de índice. Estos índices se denominan flujos de datos.

Fleet

- Fleet es la interfaz de usuario dentro de Kibana que permite la gestión centralizada de Elastic Agent y las políticas asociadas. Esta interfaz de usuario te brinda la posibilidad de ver el estado de cada Agent, la versión instalada, la hora de registro o actividad más reciente, e información sobre las políticas. La comunicación con cada Elastic Agent la facilita Fleet a través del servidor de Fleet. Esto permite distribuir de forma remota nuevas actualizaciones de políticas durante el registro, además de actualizar los binarios o las integraciones de Agent.
- Fleet Server es el mecanismo para conectar los Elastic Agent a Fleet. Un servidor de Fleet es una instancia de Elastic Agent que se ejecuta como coordinador de comunicación entre Fleet y todas las instancias de Elastic Agent desplegadas. Permite una infraestructura escalable y es compatible con Elastic Cloud y clústeres autogestionados. Fleet Server es un proceso independiente que se comunica con los agentes elásticos implementados.
- Toda la comunicación entre la interfaz de usuario de Fleet y el servidor de Fleet se realiza a través de Elasticsearch. Fleet escribe políticas, acciones y cualquier cambio en los índices fleet-* de Elasticsearch. Cada servidor de Fleet supervisa los índices, recoge los cambios y los envía a los Elastic Agent. Para comunicarse con Fleet sobre el estado de los Elastic Agent y la implementación de políticas, los servidores de Fleet escriben actualizaciones en los índices fleet-*.

Ingest Pipelines

- Las canalizaciones de ingesta permiten realizar transformaciones comunes en los datos antes de indexarlos. Por ejemplo, puede usar canalizaciones para eliminar campos, extraer valores de texto y enriquecer sus datos.
- Una canalización consta de una serie de tareas configurables llamadas procesadores. Cada procesador se ejecuta de forma secuencial y realiza cambios específicos en los documentos entrantes. Una vez que se han ejecutado los procesadores, Elasticsearch agrega los documentos transformados a su flujo de datos o índice.
- Se pueden crear y administrar canales de ingesta mediante la función Ingest pipelines de Kibana o las API de ingesta.
- Se utiliza el parámetro pipeline de consulta para aplicar una canalización a los documentos en solicitudes de indexación individuales o masivas y actualización por consulta o reindexación.
  - POST my-data-stream/_doc?pipeline=demo-format-log { … }
- Se utilice la configuración de índice index.default_pipeline para establecer una secuencia predeterminada que se aplica si no se especifica ningún parámetro pipeline. Con index.final_pipeline se establece una canalización final que se aplica después de la canalización de solicitud o predeterminada, incluso si no se especifica ninguna.

KIBANA

Introducción

- Kibana surgió un par de años después de Elasticsearch, como una herramienta web de análisis y visualización de datos (originalmente de la ingesta de logs), para aportarle una interfaz gráfica de usuario alternativa a las APIs originales, actuando como la ventana visual a los datos almacenados en Elasticsearch. Con el paso de los años ha ido asumiendo responsabilidades que la convierten en una herramienta multipropósito.
- Kibana permite dar forma a los datos y navegar por Elastic Stack:
  - **Analice sus datos**: Busque información oculta, visualice lo que encontró en gráficos, indicadores, mapas y más, y combínelos en un panel.
  - **Busque, observe y proteja sus datos**: Desde descubrir documentos hasta analizar registros y encontrar vulnerabilidades de seguridad, Kibana es el portal para acceder a estas funciones y mucho más.
  - **Administre, monitoree y proteja Elastic Stack**: Administre sus datos, monitoree el estado de su clúster Elastic Stack y controle qué usuarios tienen acceso a qué funciones.
- Kibana está destinado a administradores, analistas y usuarios empresariales. Como administrador, su función es gestionar Elastic Stack, desde la creación de su implementación hasta la introducción de datos de Elasticsearch en Kibana y, luego, la gestión de los datos. Como analista, desea descubrir información valiosa en los datos, visualizarlos en paneles y compartir sus hallazgos. Como usuario empresarial, desea ver paneles existentes para acceder a la información y profundizar en los detalles.
- Kibana trabaja con todo tipo de datos. Sus datos pueden ser texto estructurado o no estructurado, datos numéricos, datos de series temporales, datos geoespaciales, registros, métricas, eventos de seguridad y más. Sin importar cuáles sean sus datos, Kibana puede ayudarlo a descubrir patrones y relaciones y visualizar los resultados.

Analiticas

- Con Kibana Analytics, se puede buscar rápidamente en grandes cantidades de datos, explorar campos y valores, y luego usar la interfaz de arrastrar y soltar para crear rápidamente gráficos, tablas, métricas y más. El análisis de los datos (explorar, visualizar y analizar) consiste en:
  1. **Obtener y agregar datos**: La mejor manera de agregar datos a Elastic Stack es usar una de las muchas integraciones. También puede agregar un conjunto de datos de muestra o cargar un archivo. Las tres opciones están disponibles en la página de inicio.
  2. **Explorar:** Con Discover, puede buscar en sus datos información y relaciones ocultas. Formule sus preguntas y luego filtre los resultados para obtener solo los datos que desea. Puede limitar los resultados a los documentos más recientes agregados a Elasticsearch.
  3. **Visualizar:** Kibana ofrece muchas opciones para crear visualizaciones de sus datos, desde datos basados en agregación hasta datos de series temporales y datos geográficos. Dashboard es su punto de partida para crear visualizaciones y luego reunirlas para mostrar sus datos desde múltiples perspectivas. Use Canvas para darle a sus datos el factor "sorpresa" para mostrarlos en una pantalla grande. Use Graph para explorar patrones y relaciones.
  4. **Modelar el comportamiento de los datos:** Utilice el aprendizaje automático para modelar el comportamiento de sus datos: pronostique comportamientos inusuales y realice análisis de detección, regresión y clasificación de valores atípicos.
  5. **Compartir**: Una vez se este listo para compartir loas hallazgos con una audiencia más amplia, Kibana ofrece muchas opciones: incorporar un panel, compartir un vínculo, exportar a PDF y más.

Buscar, observar y proteger

- Poder buscar, observar y proteger sus datos es un requisito para cualquier analista. Kibana ofrece soluciones para cada uno de estos casos de uso:
  - **Enterprise Search**, centrada en los usuarios finales, permite crear una experiencia de búsqueda para una aplicación, lugar de trabajo y sitio web.
  - **Elastic Observability** permite a administrdores y operaciones monitorear y aplicar análisis en tiempo real a los eventos que ocurren en todos sus entornos. Se puede analizar eventos de registro, monitorear las métricas de rendimiento del host o contenedor en el que se ejecutaron, rastrear una transacción y verificar la disponibilidad general del servicio.
  - **Elastic Security**, diseñado para analistas de seguridad, ofrece una descripción general de los eventos y alertas de su entorno. Elastic Security ayuda a defender a la organización de amenazas antes de que se produzcan daños y pérdidas.

Administrar Elasticsearch

- Kibana le ayuda a realizar sus tareas de gestión de datos desde la comodidad de una interfaz de usuario. Puede:
  - Actualice, vacíe y borre la memoria caché de sus índices.
  - Define el ciclo de vida de un índice a medida que envejece.
  - Defina una política para tomar instantáneas de su clúster.
  - Reúne datos de uno o más índices en un índice nuevo y compacto.
  - Replicar índices en un clúster remoto y copiarlos en un clúster local.
- Así mismos, permite la gestión de:
  - Pipelines de ingesta de datos
  - Data views, ficheros, objetos guardados, etiquetas, … de Kibana
  - Alertas, Reglas, Perspectivas, …
  - Seguridad: usuarios, roles, API Keys, …

Explorar y visualizar

- Explorar y visualizar
  - Visualizaciones
  - Kibana Lens
  - Time Series Visual Builder
  - Análisis geoespacial
  - Gráficos
  - Métricas
  - Tablas de datos
  - Vega (personalizado)
  - Plugins de Kibana
  - Canvas
  - User Experience
  - Editor de campos de tiempo de ejecución de Kibana
- Exploración de datos
  - Dashboards
  - Discover
  - Estadísticas de campo
  - Interfaz de la consola
  - Analítica de grafo
  - Consola
- Dashboards preconfigurados
  - Módulos de servidor web
  - Módulos de base de datos
  - Módulos de infraestructura
- Compartir y colaborar
  - Dashboards insertables
  - Modo de solo dashboard
  - Spaces
  - Reportes en PDF/PNG
  - Banners personalizados para Kibana Spaces
  - Exportaciones de CSV
  - Importación/exportación de objetos guardados
  - Etiquetas
- Machine learning
  - Pronóstico de series temporales
  - Detección de anomalías en series temporales
  - Alerta de anomalías
  - Análisis de poblaciones/entidades
  - Categorización de mensajes de log
  - Indicación de causa raíz
  - Data Visualizer
  - Inferencia
  - Identificación de idioma
  - Gestión de snapshots modelo

Explorar, visualizar y analizar

- Si algo abunda hoy en nuestras vidas, son los datos. Generamos datos constantemente sobre cualquier actividad humana o fenómeno natural. Vivimos rodeados de datos. El bombardeo es continuo lo que hace más necesario que nunca desarrollar nuestra capacidad para leer, interpretar y tomar decisiones basadas en datos.
- En la actualidad, las herramientas de análisis de datos se convierten en algo imprescindible para cualquier empresa. Los procesos de manipulación de datos se basan en secuencias que aseguran el manejo y análisis efectivo. La observación minuciosa de la información y el diagnóstico a tiempo, fundamenta la asertividad en las estrategias propuestas y la toma de decisiones por medio de la exploración y visualización de datos. En suma, la cultura contemporánea es visual, por lo que estos procesos facilitan la transformación de datos en representaciones visuales que ayudan la comprensión de la información.
- La exploración de datos es la primera etapa de un análisis de datos. Generalmente, esta se presta para identificar tendencias o patrones desde el primer acercamiento a los macrodatos. A causa de esto, las empresas lo requieren como un diagnóstico que marque el punto de partida para tomar acción de manera eficaz y rápida. Gracias a esta se podrás reducir el tiempo dedicado al análisis de datos y plantear una ruta de acción práctica. Para propiciar la información hallada de manera efectiva se emplean recursos de visualización informativa.
- Una vez se ha recolectado la información, se realiza una representación visual para que sea más accesible la observación y estudio de los datos. La comprensión del diagnóstico realizado en la exploración de datos se facilita gracias a elementos esquemáticos como los cuadros, las barras, los gráficos de dispersión, las tablas de texto, los histogramas, las burbujas, los mapas, los gráficos de bala, etc. Gracias a este análisis visual se genera la configuración y diseño avanzado de cuadros de mando.

Análisis

- El **análisis descriptivo** ayuda a responder preguntas sobre lo que ha sucedido, en función de datos históricos. Las técnicas de análisis descriptivo resumen grandes modelos semánticos para describir resultados para las partes interesadas. Mediante el desarrollo de indicadores clave de rendimiento (KPI), estas estrategias pueden facilitar el seguimiento del éxito o el fracaso de los objetivos clave. En muchos sectores se usan métricas como la rentabilidad de la inversión (ROI), y las métricas especializadas se desarrollan para realizar un seguimiento del rendimiento en sectores específicos.Un ejemplo de análisis descriptivo es la generación de informes para proporcionar una visión de los datos financieros y de ventas de una organización.
- El **análisis de diagnóstico** ayuda a responder preguntas sobre por qué se ha producido un evento. Las técnicas de análisis de diagnóstico complementan el análisis descriptivo básico y usan los resultados del análisis descriptivo para identificar la causa de estos eventos. Después, los indicadores de rendimiento se investigan aún más para descubrir por qué estos eventos han mejorado o empeorado. Este proceso se suele realizar en tres pasos:
  - Identificación de anomalías en los datos que pueden ser cambios inesperados en una métrica o en un mercado determinado.
  - Recopilación de datos relacionados con estas anomalías.
  - Uso de técnicas estadísticas para detectar relaciones y tendencias que expliquen estas anomalías.
- El **análisis predictivo** ayuda a responder a preguntas sobre lo que ocurrirá en el futuro. Las técnicas de análisis predictivo usan datos históricos para identificar tendencias y determinar la probabilidad de que se repitan. Las herramientas de análisis predictivo proporcionan conclusiones valiosas sobre lo que podría ocurrir en el futuro. Engloban diversas técnicas estadísticas y de aprendizaje automático, como las de redes neuronales, árboles de decisión y regresión.

Análisis

- El **análisis prescriptivo** ayuda a responder preguntas sobre las acciones que se deben llevar a cabo para lograr un objetivo. Las conclusiones obtenidas con el análisis prescriptivo permiten a las organizaciones tomar decisiones basadas en datos. Esta técnica permite que, en caso de incertidumbre, las empresas tomen decisiones fundamentadas. Las técnicas de análisis prescriptivo dependen utilizan el aprendizaje automático como una de sus estrategias para buscar patrones en modelos semánticos de gran tamaño. Mediante el análisis de eventos y decisiones anteriores, las organizaciones pueden calcular la probabilidad de otros resultados.
- El **análisis cognitivo** intenta obtener inferencias a partir de datos y patrones existentes, derivar conclusiones en función de bases de conocimiento existentes y, después, devolver estos resultados a la base de conocimiento para futuras inferencias, un bucle de comentarios de autoaprendizaje. El análisis cognitivo ayuda a saber lo que podría ocurrir si cambiaran las circunstancias y a determinar cómo se podrían controlar estas situaciones. Las inferencias no son consultas estructuradas basadas en una base de datos de reglas, sino supuestos no estructurados que se recopilan de varios orígenes y se expresan con distintos grados de confianza. El análisis cognitivo eficaz depende de algoritmos de aprendizaje automático y usa varios conceptos del procesamiento de lenguaje natural para entender orígenes de datos desaprovechados anteriormente, como los registros de conversaciones de centros de llamadas y revisiones de productos.

Roles

- Hasta hace poco tiempo, roles como los de analistas de negocios y desarrolladores de inteligencia empresarial eran los habituales para el procesamiento y la comprensión de los datos. Pero el aumento excesivo del tamaño de los datos y sus diferentes tipos ha provocado que estos roles evolucionen hacia conjuntos de aptitudes más especializadas que modernizan y simplifican los procesos de ingeniería y análisis de datos:
  - **Analista de negocios**: está más cerca de la empresa y es un especialista en la interpretación de los datos que proceden de la visualización. A menudo, las tareas del analista de datos y el analista de negocios pueden ser responsabilidad de una misma persona.
  - **Analista de datos**: permiten a las empresas maximizar el valor de sus recursos de datos a través de herramientas de visualización y creación de informes. Sus responsabilidades incluyen la generación de perfiles, la limpieza y la transformación de los datos, el diseño y la creación de modelos semánticos escalables y eficaces, y la habilitación e implementación de las funciones de análisis avanzado en informes para su análisis. Trabaja con las partes interesadas pertinentes para identificar los requisitos de datos y de creación de informes necesarios y, después, se encarga de convertir los datos sin procesar en conclusiones relevantes y significativas.
  - **Ingeniero de datos**: aprovisionan y configuran las tecnologías de plataforma de datos locales y en la nube. Administran y protegen el flujo de datos estructurados y no estructurados procedentes de múltiples orígenes. Aseguran de que los servicios de datos se integren de forma segura y sin problemas en las plataformas de datos. Sus responsabilidades incluyen el uso de servicios datos locales y en la nube, y herramientas para la ingesta, la salida y la transformación de datos procedentes de múltiples orígenes.
  - **Científico de datos**: realizan un análisis avanzado para extraer valor de los datos. Su trabajo puede variar del análisis descriptivo al análisis predictivo.
  - **Administrador de base de datos**: implementa y administra los aspectos operativos de las soluciones de plataforma de datos locales, híbridas y nativas de la nube que se basan en servicios de datos. También es responsable de la disponibilidad general, las optimizaciones y el rendimiento coherentes de las soluciones de base de datos.

Artefactos Kibana

- Kibana necesita una **vista de datos** que le indique a qué datos de Elasticsearch desea acceder y si los datos están basados en el tiempo. Una vista de datos puede señalar uno o más flujos de datos, índices o alias de índice de Elasticsearch por nombre. Las vistas de datos las crea normalmente un administrador al enviar datos a Elasticsearch. Se puede crear o actualizar vistas de datos en Stack Management o mediante un script que acceda a la API de Kibana.
- Kibana utiliza la vista de datos para mostrar una lista de campos. Se puede personalizar el nombre y el formato de visualización de cada campo. Kibana tiene formateadores de campos para cadenas, fechas, puntos geográficos y números.
- Kibana ofrece varias formas de crear **consultas de búsqueda**, lo que reducirá la cantidad de coincidencias de documentos que obtiene de Elasticsearch. Las aplicaciones de Kibana ofrecen un filtro de tiempo y la mayoría de las aplicaciones también incluyen búsqueda semiestructurada y filtros adicionales.
- El **filtro de tiempo global** limita el intervalo de tiempo de los datos que se muestran. En la mayoría de los casos, el filtro de tiempo se aplica al campo de tiempo en la vista de datos, pero algunas aplicaciones permiten utilizar un campo de tiempo diferente.
- La **búsqueda semiestructurada** filtrará los documentos en busca de coincidencias y solo devolverá los documentos que coincidan. Combinan la búsqueda de texto libre con la búsqueda basada en campos mediante el lenguaje de consulta Kibana (KQL).
- Kibana permite guardar objetos para su uso futuro propio o para compartirlos con otros. Para organizarlos, cada objeto guardado puede tener un nombre, etiquetas y tipo.

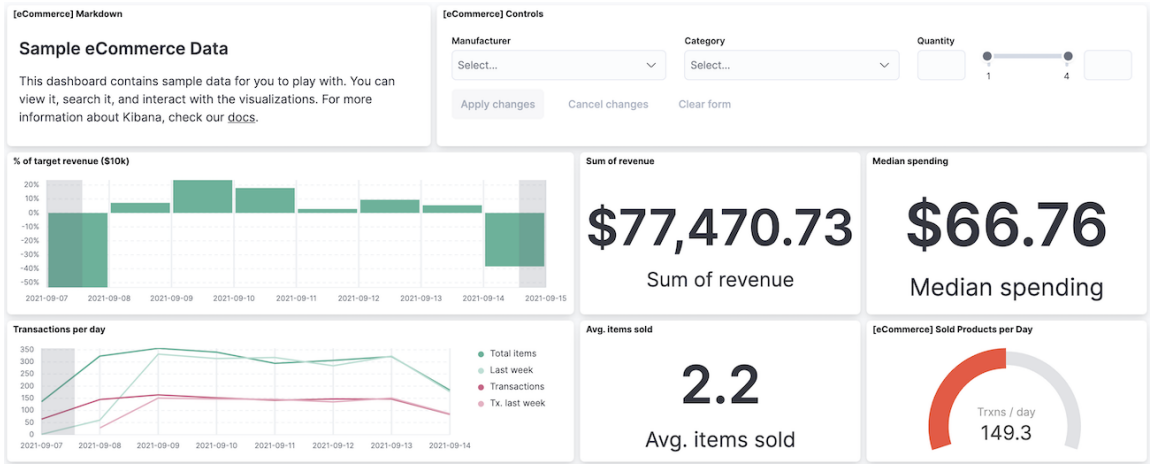
Discover

- Con Discover, se puede buscar y filtrar rápidamente sus datos, obtener información sobre la estructura de los campos y mostrar los hallazgos en una visualización. También se puede personalizar y guardar las búsquedas y colocarlas en un panel.
- Las principales acciones son:
  - Seleccionar la fuente de datos: mediante un Data View o una consulta ES|QL.
  - En caso de ser una serie temporal, ajustar el rango de tiempo para ver los datos.
  - Explorar los campos de los datos: Discover incluye una tabla que muestra todos los documentos que coinciden con tu búsqueda. De forma predeterminada, la tabla de documentos incluye una columna para el campo de tiempo y una columna que enumera todos los demás campos del documento. Se puede personalizar la tabla incluyendo, retirando y ordenando campos en columnas.
  - Buscar y Filtrar los datos: Una de las funciones exclusivas de Discover es la posibilidad de combinar la búsqueda de texto libre con el filtrado basado en datos estructurados. Mientras que la busqueda define el conjunto de documentos que le interesan, los filtros le permiten centrarse en los documentos a mostrar. Para buscar en campos específicos y crear consultas más complejas, se utiliza el KQL. Los filtros se crean mediante el asistente o con Query DSL.
  - Consultar un documento y estadísticas de campos: Permite abrir un documento individual para ver sus campos y los documentos que ocurrieron antes y después de él. Discover proporciona las estadísticas del documento, los valores mínimo, medio y máximo, una lista de los valores más altos y un gráfico de distribución.
  - Visualizar campos: Si un campo se puede agregar, entonces se puede visualizar como una grafica o mapa para explorar sus valores individuales y generar los componentes de los Dashboard.
  - Guardar la búsqueda para usarla más tarde.
  - Compartir hallazgos: desde Discover se pueden compartir los enlaces y exportar los datos directamente.
  - Generar alertas: desde Discover se pueden crear alertas en función de los resultados de la consulta.

Dashboard

- La mejor manera de comprender los datos es visualizándolos. Con los paneles de control, se pueden convertir los datos de una o más vistas de datos en una colección de paneles que aportan claridad a sus datos, cuentan una historia sobre ellos y permiten centrarse únicamente en los datos que son importantes. Un dashboard o panel de control reúne varios paneles simples.
- Los paneles muestran los datos en KPI, tablas, gráficos, mapas y más, lo que le permite comparar los datos en paralelo para identificar patrones y conexiones. Los dashboard admiten varios tipos de paneles para mostrar los datos y varias opciones para crear paneles. Los paneles configurados se pueden guardar y recuperar en biblioteca de visualizaciones (Visualize Library).
- Kibana permite a través de los paneles:
  - Utilizar y crear visualizaciones de sus datos a través de editores. Cada editor tiene distintas capacidades para todos los niveles de analistas.
  - Crear visualizaciones de datos geográficos.
  - Mostrar una tabla de registros de transmisión en vivo.
  - Mostrar los resultados de los trabajos de detección de anomalías de aprendizaje automático.
  - Mostrar un gráfico de anomalías desde el Explorador de anomalías.
  - Personaliza el tablero con imágenes personalizadas, agregar filtros interactivos al paneles de control y añadir contexto con paneles de texto (Markdown).

Dashboard



Lens

- Para crear una visualización, se arrastra los campos de datos que desea visualizar al espacio de trabajo; luego, Lens usa las mejores prácticas de visualización para aplicar los campos y crear una visualización que muestre mejor los datos.
- Con Lens, se puede:
  - Crear gráficos de áreas, líneas y barras con capas para mostrar múltiples índices y tipos de gráficos.
  - Cambiar la función de agregación para cambiar los datos en la visualización.
  - Crear tablas personalizadas.
  - Realizar cálculos matemáticos sobre agregaciones utilizando la Fórmula.
  - Utilizar cambios de tiempo para comparar los datos en dos intervalos de tiempo, como mes a mes.
  - Añadir anotaciones y líneas de referencia.

Lens

- Tabular
  - Table
- Bar
  - Bar horizontal
  - Bar horizontal percentage
  - Bar horizontal stacked
  - Bar vertical
  - Bar vertical percentage
  - Bar vertical stacked
- Line and area
  - Area
  - Area percentage
  - Area stacked
  - Line
- Goal and single value
  - Horizontal Bullet
  - Vertical Bullet
  - Semi-circular Gauge
  - Arc Gauge
  - Circular Gauge
  - Legacy Metric
  - Metric
- Magnitude
  - Heat map
- Map
  - Region map
- Proportion
  - Donut
  - Mosaic
  - Pie
  - Tag cloud
- Proportion
  - Donut
  - Mosaic
  - Pie
  - Tag cloud
  - Treemap
  - Waffle

TSVB

- TSVB es un conjunto de tipos de visualización que se configuran y muestran en los paneles.
- Con TSVB, se puede:
  - Combinar un número infinito de agregaciones para mostrar sus datos.
  - Anotar datos de series de tiempo con eventos con marca de tiempo de un índice de Elasticsearch.
  - Ver los datos en varios tipos de visualizaciones, incluidos gráficos, tablas de datos y paneles de rebajas.
  - Mostrar múltiples vistas de datos en cada visualización.
  - Utilizar funciones personalizadas y algo de matemática en las agregaciones.
  - Personalizar los datos con etiquetas y colores.

Vega

- Vega y Vega-Lite son gramáticas para crear visualizaciones personalizadas. Se recomiendan para usuarios avanzados que se sienten cómodos escribiendo consultas de Elasticsearch manualmente. Vega-Lite es un buen punto de partida para los usuarios que no conocen ambas gramáticas, pero no son compatibles.
- Los paneles Vega y Vega-Lite pueden mostrar una o más fuentes de datos, incluidos Elasticsearch, Elastic Map Service, URL o datos estáticos, y admiten extensiones de Kibana que le permiten integrar los paneles en su tablero y agregar herramientas interactivas.
- Se utiliza Vega o Vega-Lite cuando se desee crear visualizaciones con:
  - Agregaciones que utilizan mapeos nested o parent/child
  - Agregaciones sin una vista de datos
  - Consultas que utilizan filtros de tiempo personalizados
  - Cálculos complejos
  - Datos extraídos de _source en lugar de agregaciones
  - Gráficos de dispersión, gráficos de Sankey y mapas personalizados
  - Un tema visual sin soporte

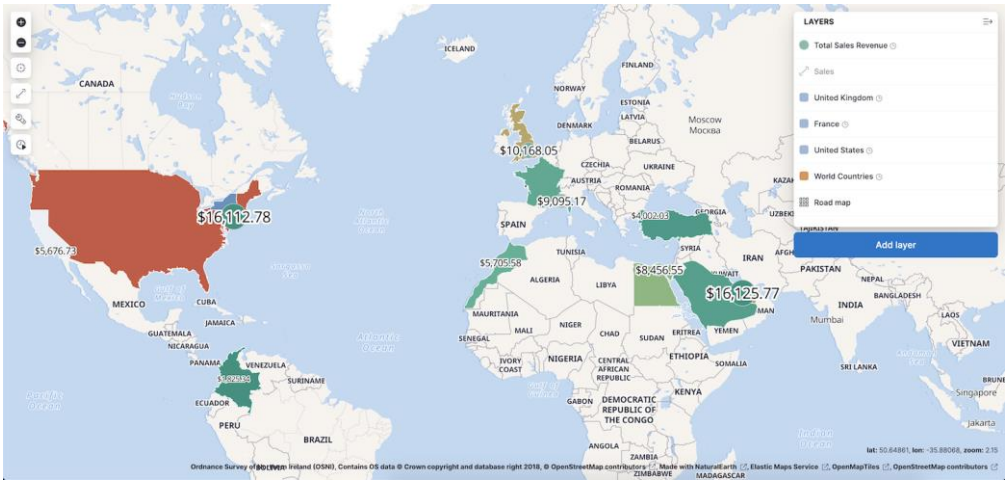
Basadas en agregación

- Las visualizaciones basadas en agregación son los paneles centrales de Kibana y no están optimizadas para un caso de uso específico.
- Con visualizaciones basadas en agregación, se puede:
  - Dividir los gráficos en hasta tres niveles de agregación, lo que es más que Lens y TSVB
  - Crear una visualización con datos que no sean de series temporales
  - Utilizar una búsqueda guardada como entrada
  - Ordenar tablas de datos y utilizar las funciones de fila de resumen y columna de porcentaje
  - Asignar colores a series de datos
- Las visualizaciones basadas en agregación incluyen las siguientes limitaciones:
  - Opciones de estilo limitadas
  - Las matemáticas no están soportadas
  - No se admiten índices múltiples

Mapas

- Se pueden crear hermosos mapas a partir de los datos geográficos. Con Maps, se puedes:
  - Construir mapas con múltiples capas e índices.
  - Animar datos espaciales temporales.
  - Subir archivos GeoJSON y shapefiles.
  - Incrustar el mapa en los paneles de control.
  - Simbolizar características utilizando valores de datos.
  - Concentrarse únicamente en los datos que son importantes.

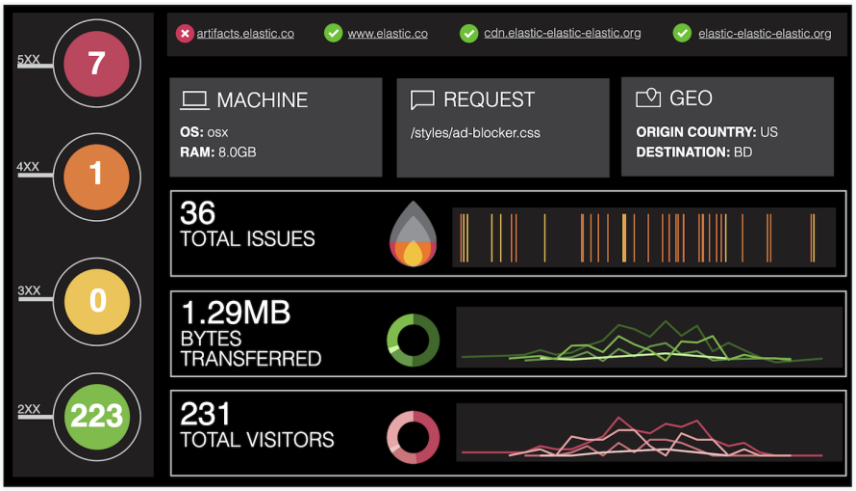
Mapas



Canvas

- Canvas es una herramienta de visualización y presentación de datos que permite extraer datos en vivo de Elasticsearch y luego combinarlos con colores, imágenes, texto y tu imaginación para crear presentaciones dinámicas, de varias páginas y con píxeles perfectos. Si eres un poco creativo, un poco técnico y muy curioso, Canvas es para ti.
- Con Canvas, se puede:
  - Crear y personalizar el espacio de trabajo con fondos, bordes, colores, fuentes y más.
  - Personalizar tu panel de trabajo con tus propias visualizaciones, como imágenes y texto.
  - Extraer los datos directamente de Elasticsearch y luego muéstrelos con gráficos, monitores de progreso y más.
  - Enfocar los datos que desea mostrar con filtros.

Canvas



Informar y compartir

- Kibana le ofrece varias opciones para compartir búsquedas guardadas de Discover, paneles, visualizaciones de Visualize Library y paneles de trabajo de Canvas.
- Las opciones para compartir incluyen lo siguiente:
  - Informes en PDF: genere y descargue archivos PDF de paneles, visualizaciones y paneles de trabajo de Canvas.
  - Informes PNG: genere y descargue archivos PNG de paneles y visualizaciones.
  - Informes CSV: genere informes CSV de búsquedas guardadas.
  - Descargar CSV: genere y descargue archivos CSV de visualizaciones de Lens.
  - Obtener enlaces: comparta enlaces directos a búsquedas guardadas, paneles y visualizaciones.
  - Descargar como JSON: genere y descargue archivos JSON de paneles Canvas.
  - Código para incrustar : incrusta paneles totalmente interactivos como un iframe en páginas web.

Machine learning

- A medida que los conjuntos de datos aumentan en tamaño y complejidad, el esfuerzo humano necesario para inspeccionar los paneles de control o mantener reglas para detectar problemas de infraestructura, ataques cibernéticos o problemas comerciales se vuelve poco práctico.
- Las funciones de aprendizaje automático Elastic, como la detección de anomalías y la detección de valores atípicos, facilitan la detección de actividades sospechosas con una mínima interferencia humana. También puede cargar un archivo CSV, NDJSON o de registro. El visualizador de datos identifica el formato de archivo y las asignaciones de campos. Luego, se pueden importar opcionalmente esos datos a un índice de Elasticsearch.
- La función de detección de anomalías de aprendizaje automático de Elastic modela automáticamente el comportamiento normal de los datos de series temporales (tendencias de aprendizaje, periodicidad y más) en tiempo real para identificar anomalías, optimizar el análisis de la causa raíz y reducir los falsos positivos. La detección de anomalías se ejecuta y escala con Elasticsearch, e incluye una interfaz de usuario intuitiva en la página de aprendizaje automático de Kibana para crear trabajos de detección de anomalías y comprender los resultados.

Observabilidad

- La observabilidad le permite agregar y monitorear los registros de logs, métricas del sistema, datos de tiempo de actividad y rastros de aplicaciones, como una sola pila.
- Con Observability, tienes:
  - Un lugar central para agregar y configurar sus fuentes de datos.
  - Una variedad de gráficos que muestran análisis relacionados con cada fuente de datos.
  - Vea las opciones de la aplicación para explorar en profundidad y analizar datos en las aplicaciones Registros, Métricas, Tiempo de actividad y APM.
  - Un cuadro de alertas para mantenerlo informado sobre cualquier problema que deba resolver rápidamente.
- Kibana ofrece instrucciones paso a paso para ayudar a agregar y configurar sus fuentes de datos.

Alertas

- Las alertas le permiten definir reglas que detectan condiciones complejas dentro de diferentes aplicaciones de Kibana y desencadenan acciones cuando se cumplen esas condiciones. Las alertas están integradas con Observability, Security, Maps y Machine Learning. Se pueden administrar de forma centralizada desde Stack Management y brindan un conjunto de reglas y conectores integrados para que se usen.
- Las alertas funcionan ejecutando comprobaciones según un cronograma para detectar condiciones definidas por una regla. Cuando se cumple una condición, la regla la registra como una alerta y responde activando una o más acciones. Las acciones generalmente implican la interacción con los servicios de Kibana (Index, Server log, …) o integraciones de terceros (correo electrónico, webhooks, Jira, Microsoft Teams, PagerDuty, ServiceNow, Slack, …). Los conectores permiten que las acciones se comuniquen con estos servicios e integraciones.
- Una regla consta de tres partes principales:
  - Condiciones: ¿Qué es necesario detectar?
  - Calendario: ¿cuándo y con qué frecuencia deben ejecutarse las comprobaciones de detección?
  - Acciones: ¿qué sucede cuando se detecta una condición?

Elastic Security

- Elastic Security combina análisis de detección de amenazas, seguridad nativa de la nube y capacidades de protección de puntos finales en una única solución, para que pueda detectar, investigar y responder rápidamente a amenazas y vulnerabilidades en todo su entorno.
- Elastic Security proporciona:
  - Un motor de detección que identifica una amplia gama de amenazas
  - Un espacio de trabajo para la clasificación de eventos, la investigación y la gestión de casos
  - Herramientas de visualización de datos interactivas
  - Integraciones para recopilar datos de diversas fuentes

Dev Tools

- Dev Tools contiene herramientas que puedes usar para interactuar con tus datos:
  - Consola
    - Para interactuar con las API REST de Elasticsearch y Kibana, incluido el envío de solicitudes y la visualización de la documentación de la API.
  - Search Profiler
    - Para inspeccionar y analizar la consultas de búsqueda.
  - Grok Debugger
    - Para crear y depurar patrones grok antes de usarlos en los pipelines de procesamiento de datos.
  - Painless Lab
    - Para probar y depurar scripts Painless en tiempo real.

Stack Monitoring

- Las funciones de monitoreo de Kibana tienen dos propósitos diferentes:
  - Para visualizar datos de monitoreo de Elastic Stack, puede ver datos de estado y rendimiento de Elasticsearch, Logstash, Enterprise Search, APM y Beats en tiempo real, así como analizar el rendimiento anterior.
  - Para monitorear Kibana en sí y enrutar esos datos al clúster de monitoreo.
- Si se habilita la supervisión en Elastic Stack, cada componente supervisado se considera único en función de su UUID persistente, que se escribe en el directorio path.data cuando se inicia el nodo o la instancia.

Stack Management

- Stack Management es el hogar de las interfaces de usuario para administrar todos los elementos de Elastic Stack: índices, clústeres, licencias, configuraciones de interfaz de usuario, vistas de datos, espacios y más.
- El acceso a funciones individuales se rige por los privilegios de Elasticsearch y Kibana.

ANEXOS

GenerateData

- GenerateData es una herramienta para la generación automatizada de juegos de datos.
- Ofrece ya una serie de datos precargados en BBDD y un conjunto de tipos de datos bastante amplio, así como la posibilidad de generar tipos genéricos.
- Podemos elegir en que formato se desea la salida de entre los siguientes:
  - CSV
  - Excel
  - HTML
  - JSON
  - LDIF
  - Lenguajes de programación (JavaScript, Perl, PHP, Ruby, C#)
  - SQL (MySQL, Postgres, SQLite, Oracle, SQL Server)
  - XML
- Online: http://www.generatedata.com/?lang=es
- Instalación (PHP): http://benkeen.github.io/generatedata/

Mockaroo

- https://www.mockaroo.com/
- Mockaroo permite descargar rápida y fácilmente grandes cantidades de datos realistas de prueba generados aleatoriamente en función de sus propias especificaciones que luego puede cargar directamente en su entorno de prueba utilizando formatos CSV, JSON, XML, Excel, SQL, … No se requiere programación y es gratuita (para generar datos de 1.000 en 1.000).
- Los datos realistas son variados y contendrán caracteres que pueden no funcionar bien con nuestro código, como apóstrofes o caracteres unicode de otros idiomas. Las pruebas con datos realistas harán que la aplicación sea más robusta porque detectará errores en escenarios de producción.
- El proceso es sencillo ya que solo hay que ir añadiendo nombres de campos y escoger su tipo. Por defecto nos ofrece mas de 140 tipos de datos diferentes que van desde nombre y apellidos (pudiendo escoger estilo, género, etc...) hasta ISBNs, ubicaciones geográficas o datos encriptados simulados.
- Además es posible hacer que los datos sigan una distribución Normal o de Poisson, secuencias, que cumplan una expresión regular o incluso que fuercen cadenas complicadas con caracteres extraños y cosas así. Tenemos la posibilidad de crear fórmulas propias para generarlos, teniendo en cuenta otros campos, condicionales, etc... Es altamente flexible.