ManualKibanaOCDS-multilang-test Documentation

Versión master

01 de abril de 2019

1. Herramientas y datos en OCDS

El manual es una iniciativa del Proyecto sobre Organización, Desarrollo, Educación e Investigación (PODER), la Iniciativa Latinoamericana por los Datos Abiertos y Open Contracting Parnershipt (OCP) escrito por **Ricardo Vega**, **Martín Szyszlican** y **Eduard Martín-Borregón** con la supervisión de Juan Pane y Yohanna Lisnichuk.

CAPÍTULO 1

Introducción

El Open Contracting Parnership indica en septiembre de 2017 que 13 países de América Latina y el Caribe (Argentina, Chile, Colombia, Costa Rica, Guadalupe, Guatemala, Honduras, Martinica, México, Panamá, Paraguay, República Dominicana y Uruguay) están activos. Esto significa que están usando el estándar de datos en contrataciones abiertas, tienen el compromiso de implementarlo y/o tienen alguna innovación en el monitoreo de contrataciones públicas.

Para incrementar el uso y el impacto de la publicación de datos se ha realizado esta guía, que muestran cómo instalar el paquete ELK, importar datos con formato de contrataciones abiertas (OCDS) y analizarlos de forma visual e intuitiva. Si bien todos los ejemplos están basados el caso mexicano, hay instrucciones para importar cualquier conjunto de datos que cumpla el estándar OCDS.

Esperamos que este sea un granito de arena más para potenciar y promover la transparencia y la rendición de cuentas en la región.

1.1 Introducción al Estándar de Datos de Contrataciones Abiertas

> *Esta introducción es una traducción del capítulo An introduction to the Open Contracting Data Standard del Manual Analyzing Open Contracting data using the R programming language de Rodrigo Parra*

Se considera un dato abierto aquel que está en un formato estructurado, reutilizable y legible por máquina; más allá de los requerimientos técnicos, los datos abiertos abren nuevas oportunidades para la participación y la rendición de cuentas ciudadana. El Estándar de Datos de Contrataciones Abiertas (OCDS por sus siglas en inglés) fue creado para aplicar estos principios a los datos relacionados con el ciclo de vida completo de contratación, que incluye planificación, convocatoria, adjudicación, contratación e implementación.

El estándar de datos, diseñado y desarrollado mediante un proceso abierto por la Open Contracting Partnership (OCP), facilita a los gobiernos y ciudades de todo el mundo a compartir sus datos de contratación, permitiendo una mayor transparencia en la contratación pública y respaldando el análisis accesible y en profundidad de la eficiencia, efectividad, equidad e integridad de los sistemas de contratación pública.

La intención de esta sección es presentar al lector el estándar, los casos de uso para los que fue diseñado y los conceptos básicos necesarios para aplicarlo. La mayoría del contenido fue tomado de la documentación oficial del estándar; para obtener una introducción más completa, consulte la guía de inicio rápida de la OCP.

1.1.1 Usuarios y casos de uso

El estándar fue diseñado para satisfacer las cuatro necesidades principales que se detectaron en los usurarios:

- Lograr poner en valor el gasto gubernamental
- Fortalecimiento de la transparencia, la rendición de cuentas y la integridad de la contratación pública
- Permitir que el sector privado compita de forma justa por los contratos públicos
- Monitoreo de la efectividad de la prestación del servicio contratado

Para saber quién está publicando datos que cumplen con OCDS y cómo lo están haciendo, consulte la web de OCP. Cuatro posibles casos de uso para los datos de contratación abierta son:

- Valor para el dinero en la adquisición: ayudar a los funcionarios a obtener una buena relación calidad-precio durante el proceso de adquisición, y analizar si estos objetivos se lograron después.
- Detección de fraude y corrupción: identificación de banderas rojas que podrían indicar corrupción mediante el estudio de adquisiciones individuales o redes basadas en fondos, propiedad e intereses.
- Competir por contratos públicos: permitir que las empresas privadas comprendan el potencial de las oportunidades de adquisición al observar información relacionada con adquisiciones pasadas y actuales.
- Supervisión de la prestación de servicios: ayuda a los actores interesados a aprovechar la trazabilidad en el proceso de adquisición para fines de supervisión, vinculando los presupuestos y los datos de los donantes con los contratos y los resultados.

1.1.2 El proceso de contratación

El estándar define un proceso de contratación como:

> Toda la información de planificación, publicación de la convocatoria, adjudicaciones, contratos e implementación de contratos relacionada con un solo proceso de iniciación.

El estándar cubre todas las etapas de un proceso de contratación, aunque algunos procesos pueden no incluir todos los pasos posibles. Para fines de identificación, a todos los procesos de contratación se les asigna un Id. de contratación abierta único (ocid), que se puede utilizar para unir datos de diferentes etapas. Para evitar clústers de ocid entre editores, un editor puede anteponer un prefijo a los identificadores generados localmente. Se anima a los editores a registrar su prefijo aquí.

1.1.3 Documentos

Los procesos de contratación se representan como **documentos** en el OCDS. Cada documento se compone de varias **secciones**, que se mencionan a continuación:

- **Metadatos de lanzamiento**: información contextual sobre cada lanzamiento de datos;
- **Participantes**: información sobre las organizaciones y otros actores involucrados en el proceso de contratación;
- **Planificación**: información sobre los objetivos, presupuestos y proyectos relacionados con un proceso de contratación;
- **Convocatoria**: información sobre cómo se llevará a cabo una licitación;
- **Adjudicación**: información sobre las adjudicaciones realizadas como parte de un proceso de contratación;
- **Contrato**: información sobre contratos firmados como parte de un proceso de contratación;
- **Implementación**: información sobre el progreso de cada contrato hacia la finalización.

Un ejemplo de fragmento de JSON compatible con esta estructura se ve de la siguiente manera:

```
{
   "language": "en",
   "ocid": "contracting-process-identifier",
   "id": "release-id",
   "date": "ISO-date",
   "tag": ["tag-from-codelist"],
   "initiationType": "tender",
   "parties": {},
   "buyer": {},
   "planning": {},
   "tender": {},
   "awards": [ {} ],
   "contracts":[ {
       "implementation":{}
   }]
}
```

Hay dos tipos de documentos definidos en el estándar:

- **Releases** son inmutables y representan actualizaciones sobre el proceso de contratación. Por ejemplo, se pueden usar para notificar a los usuarios de nuevas convocatorias, premios, contratos y otras actualizaciones. Como tal, un único proceso de contratación puede tener muchos lanzamientos.
- **Registros** son instantáneas del estado actual de un proceso de contratación. Un registro debe actualizarse cada vez que se publique una nueva versión asociada a su proceso de contratación; por lo tanto, solo debe haber un registro por proceso de contratación.

1.1.4 Campos

Cada sección puede contener varios **campos** especificados en el estándar, que se utilizan para representar datos. Estos objetos pueden aparecer varias veces en diferentes secciones del mismo documento; por ejemplo, los artículos pueden presentarse en convocatoria (para indicar los artículos que un comprador desea comprar), en un objeto de adjudicación (para indicar los artículos para los que se ha realizado una adjudicación) y en un objeto contractual (para indicar los artículos enumerados en el contrato). Algunos campos de ejemplo, acompañados por los fragmentos de JSON correspondientes, se presentan a continuación.

Participantes (Organizaciones)

```
{
    "address": {
        "countryName": "United Kingdom",
        "locality": "London",
        "postalCode": "N11 1NP",
        "region": "London",
        "streetAddress": "4, North London Business Park, Oakleigh Rd S"
    },
    "contactPoint": {
        "email": "procurement-team@example.com",
        "faxNumber": "01234 345 345",
        "name": "Procurement Team",
        "telephone": "01234 345 346",
        "url": "http://example.com/contact/"
    },
    "id": "GB-LAC-E09000003",
```

(continué en la próxima página)

(proviene de la página anterior)

```
    "identifier": {
        "id": "E09000003",
        "legalName": "London Borough of Barnet",
        "scheme": "GB-LAC",
        "uri": "http://www.barnet.gov.uk/"
    },
    "name": "London Borough of Barnet",
    "roles": [ ... ]
}
```

Valores

```
{
    "amount": 11000000,
    "currency": "GBP"
}
```

Items

```
{
    "additionalClassifications": [
       {
            "description": "Cycle path construction work",
            "id": "45233162-2",
            "scheme": "CPV",
            "uri": "http://cpv.data.ac.uk/code-45233162.html"
        }
    ],
    "classification": {
        "description": "Construction work for highways",
        "id": "45233130",
        "scheme": "CPV",
        "uri": "http://cpv.data.ac.uk/code-45233130"
    },
    "description": "string",
    "id": "0001",
    "quantity": 8,
    "unit": {
        "name": "Miles",
        "value": {
            "amount": 137000,
            "currency": "GBP"
        }
    }
}
```

Periodos de tiempo

```
{
    "endDate": "2011-08-01T23:59:00Z",
    "startDate": "2010-07-01T00:00:00Z"
}
```

Documentos

```
{
    "datePublished": "2010-05-10T10:30:00Z",
    "description": "Award of contract to build new cycle lanes to AnyCorp Ltd.",
    "documentType": "notice",
    "format": "text/html",
    "id": "0007",
    "language": "en",
    "title": "Award notice",
    "url": "http://example.com/tender-notices/ocds-213czf-000-00001-04.html"
}
```

Hitos

```
{
    "description": "A consultation period is open for citizen input.",
    "dueDate": "2015-04-15T17:00:00Z",
    "id": "0001",
    "title": "Consultation Period"
}
```

1.1.5 Extensiones y listas de códigos

Además de los campos regulares, el esquema OCDS define algunos campos que solo se pueden usar en ciertas secciones, p. *títulos* y *descripciones* de licitaciones, premios y contratos. En algunos casos, los editores pueden requerir campos que no son proporcionados por el esquema central; una **extensión** permite definir nuevos campos que se pueden usar en estos casos. Una lista de las extensiones disponibles se puede encontrar aquí; si ninguna extensión existente satisface las necesidades de un editor, se alienta al editor a colaborar en la creación de una nueva extensión de comunidad.

Otro concepto que vale la pena mencionar es el de las listas de códigos. Las listas de códigos son conjuntos de cadenas sensibles a mayúsculas y minúsculas con etiquetas asociadas, disponibles en cada idioma en el que se ha traducido el OCDS. Los editores deben usar valores de lista de códigos siempre que sea posible para mapear sus sistemas de clasificación existentes; si es necesario, los campos de detalles pueden usarse para proporcionar información de clasificación más detallada. Hay dos tipos de listas de códigos:

- **Las listas de códigos cerradas** son conjuntos de valores fijos. Si un campo está asociado con una lista de códigos cerrada, solo debe aceptar una opción de la lista publicada.
- **Las listas de códigos abiertas** son conjuntos de valores recomendados. Si un campo está asociado con una lista de códigos abierta, acepta opciones de la lista, pero también otros valores.

El OCDS se mantiene utilizando un esquema JSON. En esta sección, hemos introducido y descrito las secciones principales y los objetos comunes utilizados en el esquema, proporcionando fragmentos JSON como ejemplos de estos bloques básicos. Si le intereasa en la referencia completa del esquema JSON, consulte la documentación oficial.

Si bien la mayoría de la información en este manual puede ser utilizada para el análisis de cualquier conjunto de datos, nuestros esfuerzos están orientados a analizar datos de Contrataciones Abiertas publicadas por la Secretaría de Hacienda y Crédito Público.

1.2 Datos de Contrataciones Abiertas

Desde hace muchos años existen datos abiertos sobre contrataciones en México, un historial de los mismos se puede encontrar en el sitio de CompraNet, que dispone de información en formato Excel desde el año 2002. Estos datos han sido objeto de múltiples análisis a lo largo del tiempo y están disponibles en una variedad de plataformas de tecnología cívica que simplifican su análisis. Para enumerar algunas de las que están actualmente disponibles:

- QuiénEsQuién.Wiki
- ContratoBook
- CompranNetFacil
- Data Analytics for Procurement

1.3 Datos mexicanos en OCDS

Desde el año 2015 hay diferentes proyectos en la administración pública mexicana para publicar datos en el estándar de contrataciones abiertas OCDS. El primero que estuvo disponible fue el sitio de la Ciudad de México. La fuente de datos gubernamentales en OCDS la publica la Secretaría de Hacienda y Crédito Público y son los datos que se usan en este manual.

Adicionalmente hay que destacar el esfuerzo para traducir el OCDS al español realizado por la Alianza para las contrataciones Abiertas de México; otros organismos que publican en OCDS son el Instituto Nacional de Acceso a la Información y Protección de Datos, el Grupo Aeroportuario de la Ciudad de México que publica los contratos en para la construcción del Nuevo Aeropuerto, la Secretaría de Comunicaciones y Transportes pública los contratos relacionados con el proyecto Red Compartida y a nivel subnacional está la Secretaría de Planificación, Administración y Finanzas del Gobierno de Jalisco.

1.4 Datos en OCDS en América Latina

Además de México en América Latina hay en cinco países seis conjuntos de datos en formato OCDS :

- ChileCompra
- Colombia Compra Eficiente
- Superintendencia de Alianza Público-Privada de Honduras
- Portal de Contrataciones Públicas de la República de Paraguay
- Portal de Datos Abiertos del Ministerio de Hacienda de Paraguay
- Compras Estatales de la Agencia de Compras y Contrataciones del Estado de Uruguay

En el manual OCDS Kingfisher tool se explica como descargarlos e importarlos.

1.5 Los datos para este manual

Como se dijo en el apartado anterior en este manual vamos a estar utilizando los datos publicados por el área de Transparencia Presupuestaria de la Secretaría de Hacienda y Crédito Público disponibles aquí: https://www.gob.mx/contratacionesabiertas

En particular vamos a trabajar con el concentrado de contrataciones abiertas de la Administración Pública Federal en formato JSON, obtenido desde esta URL: https://datos.gob.mx/busca/dataset/concentrado-de-contrataciones-abiertas-de-la-apf/resource/0252e19f-bdd6-43de-af7b-106d4c7a82c8

Este dataset incluye datos de contrataciones realizadas por todas las dependencias del gobierno federal, organismos autónomos, empresas paraestatales y aquellos contratos de nivel estatal y municipal que cuentan con aportes del gobierno federal. Los datos disponibles en este formato son a partir del año 2017 y se actualizan continuamente. Si bien la frecuencia de actualización del archivo en el portal de datos no está explicitada, y al momento de la escritura la última versión es de principios de 2018, sí se pueden obtener datos más nuevos mediante el uso de la API descripta en este documento: http://transparenciapresupuestaria.gob.mx/work/models/PTP/programas/OpenDataDay/Resultados/Guia%20_uso_API_contrataciones%

Adicionalmente, fuimos capaces de obtener un volcado completo de los datos en OCDS por solicitud directa al área responsable.

En el siguiente capítulo profundizaremos sobre las herramientas necesarias para realizar análisis sobre estos datos y cómo importarlos.

1.6 Las herramientas Elastic

1.6.1 Introducción

Es un conjunto de herramientas que al combinarse crean una plataforma robusta de administración de datos permitiendo el monitoreo, consolidación y análisis de los mismos.

Las herramientas que componen este sistema son: ElasticSearch, Logstash y Kibana. Y por las iniciales de estos, el conjunto también es conocido como «stack ELK» o simplemente «ELK».

Estas herramientas son creadas, mantenidas y distribuidas por la compañía Elastic desde 2012 y han evolucionado según las necesidades del mercado. La primera de las herramientas en ser creada fue ElasticSearch en 2004 bajo el nombre de Compass, pero la primera versión oficial surge en 2010.

Elastic ofrece sus productos en dos modalidades:

- Como software de código abierto, bajo la licencia Apache 2, salvo algunas funcionalidades adicionales que son distribuidas con licencia propietaria. Ofrece la oportunidad de ver, utilizar y hasta modificar las herramientas sin costo alguno, pero toda administración de las herramientas debe ser llevado a cabo personalmente.
- Como servicio de pago, Elastic Cloud pone a disposición todas las herramientas y las funcionalidades adicionales en servidores administrados por ellos.

1.6.2 Las ventajas de la plataforma ELK

Hay muchas soluciones distintas para cubrir la necesidad de procesamiento, monitoreo y visualización de datos, tanto de paga como libres, pero lo que distingue a ELK sobre otras es principalmente:

- **Potencia**: Ofrece mucha funcionalidad con un bajo costo técnico, las configuraciones son mínimas para empezar. Y las optimizaciones disponibles son amplias.
- **Escalabilidad**: Elasticsearch es una herramienta diseñada para manejar terabytes de datos sin ningún problema. Su arquitectura le permite expandirse de forma rápida y fácil.
- **Flexibilidad**: La configuración es flexible y puede adaptarse a cualquier necesidad y entorno.
- **Apertura**: Elastic fomenta un ecosistema de extensiones (plugins) alrededor de sus herramientas que han creado un número importante de funcionalidades extras y gratuitas para facilitar el trabajo con ellas.

- **Código Abierto**: Hoy en día el código abierto ofrece ventajas competitivas sobre otras plataformas porque permite la rápida corrección de errores gracias a la comunidad, la creación de extensiones e incluso incremente la base de usuarios al permitir utilizar las herramientas sin requerir pago alguno, lo que incrementa el conocimiento compartido de las herramientas.

1.6.3 Los componentes

ELK está compuesta por tres pilares fundamentales: Elasticsearch, Logstash y Kibana.

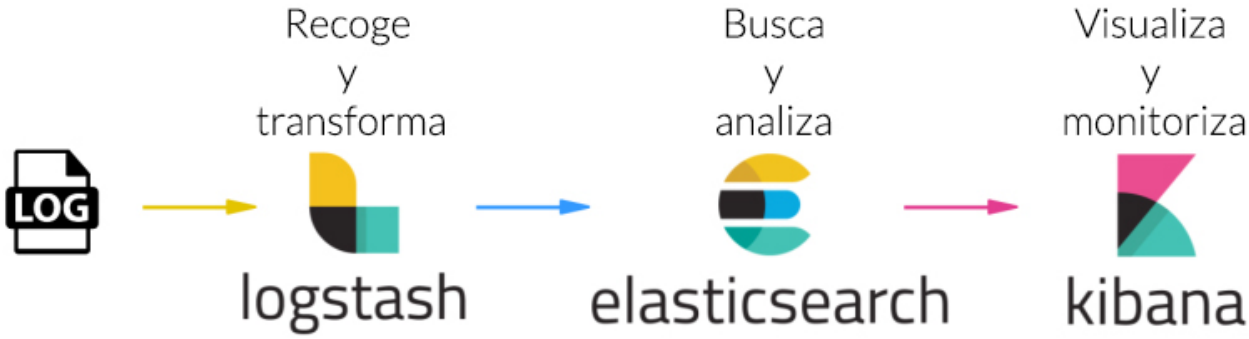
Plataforma ELK

Cada componente tiene una funcionalidad y arquitectura específica, veamos cómo se relacionan y cómo podemos usarlos como una plataforma.

1.6.4 Arquitectura de la plataforma

En este capítulo vamos a profundizar en el uso de las herramientas de Elastic para el análisis de datos de Contrataciones Abiertas en México. Para eso es necesario comprender mejor cómo se relacionan las diferentes herramientas antes de explicar cómo importaremos los datos.

La plataforma ELK tiene una arquitectura lineal entre sus componentes.



ELK Stack

1. Los datos son recogidos y procesados por LogStash
2. LogStash envía los datos ya procesados a ElasticSearch para ser indexados
3. ElasticSearch proporciona los datos a la interfaz de Kibana para poder ser consultados

Pero veamos un poco más a detalle cada herramienta:

- ElasticSearch

- Logstash
- Kibana

1.7 Plataforma ELK para el Análisis de Contrataciones en formato OCDS

Como establecimos en la introducción de este manual, el análisis de los datos de contrataciones públicas de México es una tarea indispensable para la correcta fiscalización democrática de las operaciones gubernamentales.

Para lograr este análisis debemos contar con las herramientas tecnológicas para tomar el Estándar OCDS y ponerlo a disposición del público de una forma amigable que permita refinar la información así como crear visualizaciones de estos datos.

La plataforma ELK (ElasticSearch, Logstash, Kibana) proveé de las herramientas necesarias para lograr este objetivo.

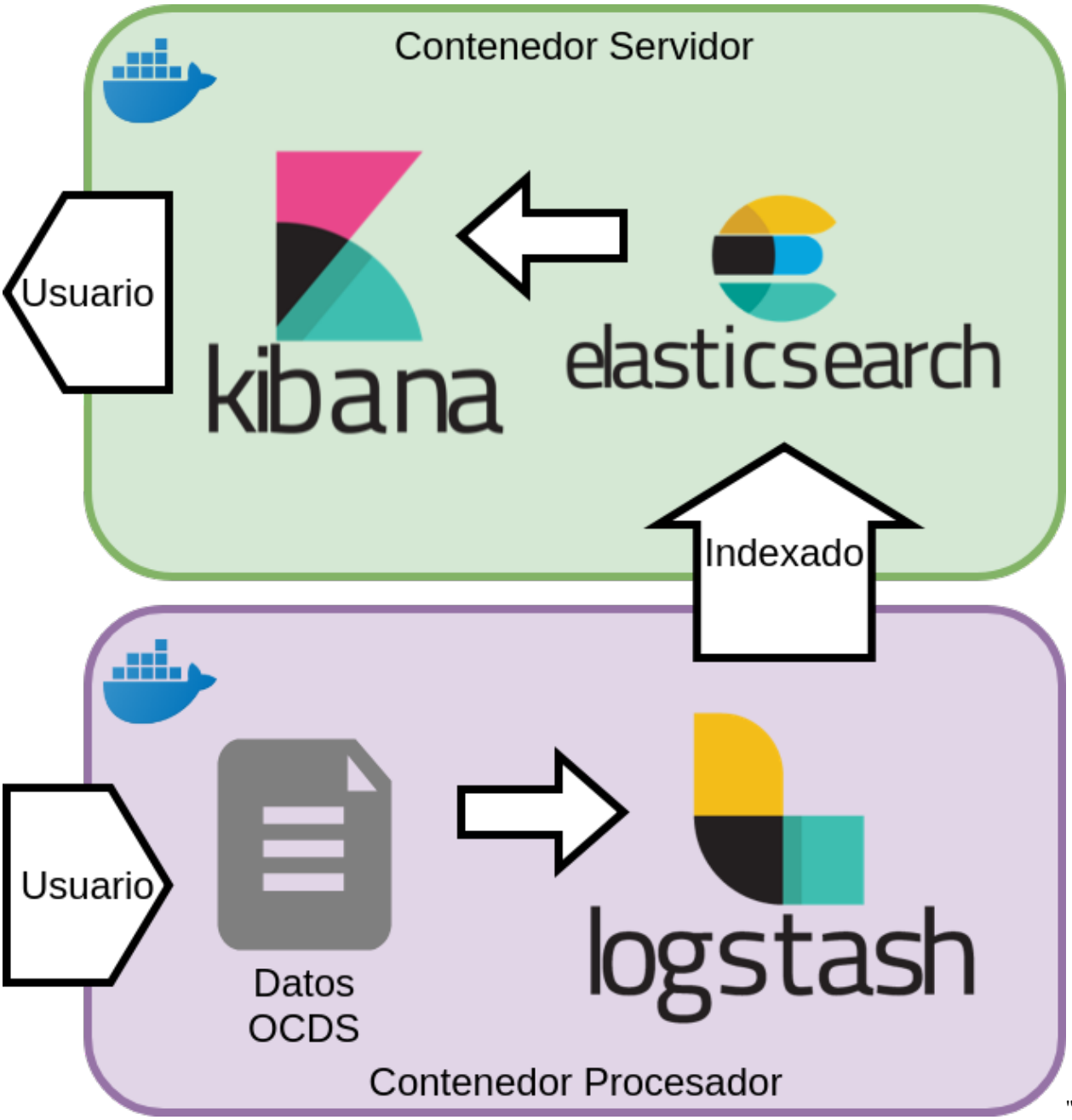
El presente manual permitirá realizar un sistema con las siguientes características:

- Interfaz gráfica de uso amigable pero robusto para consultar los datos.
- Un motor de búsqueda e indexado que permita la actualización y corrección de los datos en todo momento.
- Un motor de ingesta de datos, que permita que según se realicen las publicaciones de los datos por parte del gobierno mexicano éstos puedan ser procesados e ingresados a la base de datos con un mínimo esfuerzo.

1.7.1 Arquitectura

Cada una de estas funcionalidades estarán sustentadas en las herramientas Kibana, ElasticSearch y Logstash, y la arquitectura para su utilización quedará definida de la siguiente manera:

- Clúster ElasticSearch para indexar y contener los datos.
  - Inicialmente el clúster tendrá un solo nodo, según la demanda lo indique se podría incrementar el número de nodos.
  - Un índice especializado con los datos completos liberados.
- Interfaz de Kibana para visualizar los datos en el índice.
  - Tanto Kibana como ElasticSearch estarán inicialmente contenidos en un mismo contenedor Docker para su sencilla distribución.
- Un pipeline para Logstash preparado para tomar los datos OCDS, procesarlos e indexarlos en ElasticSearch
  - Este pipeline estará contenido en un contenedor Docker para su fácil ejecución.



"Plataforma ELK"

Llamaremos **Contenedor Servidor** al que aloja a ElasticSearch y Kibana, ya que este contenedor se mantendrá en ejecución tanto tiempo como se quiera ofrecer este servicio.

Y **Contenedor Procesador** al que solo ejecuta Logstash para procesar los datos y se da por terminado.

Extra: Contenedores Docker

Un contenedor Docker es una herramienta que utilizaremos para empaquetar nuestra solución, su arquitectura y las herramientas.

Este manual no contempla enseñar los detalles de Docker y su tecnología, pero podríamos definirlo de forma sencilla como «cajas» o «contenedores» (como de trailers o de barcos de carga) de Software donde va incluido absolutamente

todo lo que necesitamos para ejecutar nuestro proyecto.

En teoría esto debería facilitar mucho la distribución de cualquier herramienta de Software pues el único prerequisito a instalar es Docker en sí mismo. Este paso debería ser muy parecido a instalar cualquier otro software en la plataforma de cómputo que se elija.

Una vez instalado Docker, nuestro software estará listo para iniciarse automáticamente, sin necesitar de ningún tipo de software auxiliar ni dependencias.

Otra ventaja de usar Docker es que mantiene estabilidad entre lo que se desarrolla y lo que se distribuye, se evitan problemas del tipo «funciona correctamente en mi computadora».

> En las siguientes secciones utilizaremos diversos comandos de docker, entre los detalles más importantes se encuentra el concepto de «volúmenes» que podriamos entender como «carpetas compartidas» entre el contenedor docker y nuestra computadora. `docker run -v CARPETA_LOCAL:CARPETA_DEL_CONTENEDOR imagen` La opción `-v A:B` le indica a docker que queremos compartir la carpeta `A` de nuestra computadora con el contenedor, pero para el contenedor se llamará `B` Un ejemplo: `docker run -v $HOME/Descargas:/input imagen` «Comparte» la carpeta Descargas de mi computadora con el contenedor, dentro del contenedor la carpeta se llamará `/input`

Para saber más sobre Contenedores y Docker, se recomiendan las siguientes lecturas:

- Amazon Web Services - Qué es Docker?
- OpenWebinars - Docker
- 1and1.mx - Instalación de Docker

1.8 Puesta en marcha de la Plataforma ELK para el Análisis de Contrataciones en formato OCDS

Al final de esta sección tendremos todo lo necesario para hacer consultas y visualizaciones sobre los datos de Contrataciones en formato OCDS.

Para una mejor comprensión de esta sección se recomienda tener familiaridad básica con la terminal de comandos:

- Mac
- Windows
- Linux/Ubuntu

1.8.1 Objetivos

1. Iniciar un servidor ElasticSearch con Kibana.
2. Cargar los datos publicados de Contrataciones en un formato sencillo de consultar.
3. Realizar una consulta sobre los datos.

El presente proyecto está desarrollado para poder iniciar los 3 servicios, según sea necesario de forma fácil y rápida.

1.8.2 Prerequisitos

1. Abrir la terminal para comandos del Sistema Operativo
2. Instalar Docker CE

3. Instalar Docker Compose
4. Descargar el archivo que contiene las herramientas
5. Descomprimir el archivo descargado y entrar a la carpeta que se acaba de crear
   - En la terminal: `cd ManualKibanaOCDS-master`

1.8.3 Iniciar el *Contenedor Servidor* con ElasticSearch y Kibana

Ahora podremos iniciar el servidor ejecutando el siguiente comando en la terminal:

```
docker-compose -f elastic-kibana.yaml up
```

> Este comando le indica al programa Docker que debe crear un contenedor según lo indicado en el archivo `elastic-kibana.yaml`, en el mismo indicamos que ambos programas deben iniciarse.

Pasados unos minutos, depende de los recursos disponibles, deberíamos poder abrir en el navegador web la dirección http://localhost:5601/app/kibana y Kibana se mostrará disponible.

A partir de este momento ElasticSearch y Kibana están listos para ser usados. Aunque aún no tenemos datos disponibles.

1.8.4 Cargar los datos OCDS a ElasticSearch

Para este proceso debemos posicionarnos en la carpeta `elk-gobmx-csv-master/pipeline`, en la terminal:

```
cd pipeline
```

Descargando los paquetes de datos

Si queremos trabajar con el total de los contratos publicados en estándar OCDS sin importar los últimos datos publicados, lo primero que debemos hacer es asegurarnos de haber descargado el archivo de **Contrataciones en formato OCDS por paquetes json** publicado en el sitio datos.gob.mx

> Al 2 de Septiembre de 2018 este archivo lleva por nombre `contratacionesabiertas_bulk_paquetes.json.zip` y tiene un tamaño de 310.5 MB aproximadamente.

Es importante mencionar que el formato de esta información se conoce en el estándar OCDS como recordPackages o paquete de Registros; las herramientas y código incluidas en éste manual utilizan este formato, para utilizar alguno otro como *releases* o *releasePackages* o alguna otra estructura no definida por el estándar OCDS serían necesarias modificaciones al código.

> Estos archivos pueden ser bastante grandes en tamaño, es recomendable tener por lo menos 2GB libres de espacio en disco duro antes de continuar.

Ahora debemos descomprimir el archivo `contratacionesabiertas_bulk_paquetes.json.zip`, esto generará múltiples archivos `.json` dentro de una carpeta:

```
carpeta/contratacionesabiertas_bulk_paquete1.json
carpeta/contratacionesabiertas_bulk_paquete2.json
carpeta/contratacionesabiertas_bulk_paquete3.json
...
```

IMPORTANTE: Debemos saber la ruta completa de esta carpeta con los archivos .json, pues será necesaria para el paso de carga.

A manera de ejemplo asumamos que los archivos fueron descargados y descomprimidos dentro de la carpeta de Descargas del sistema operativo. La ruta completa a esta carpeta **debería ser** 1

- **Linux/Ubuntu/Mac**: `/home/{nombre de usuario}/Descargas` Se puede abreviar como `$HOME/Descargas`
- **Windows**: `C:\Users\{nombre de usuario}\Descargas` Se puede abreviar como `%HOMEPATH%\Descargas`

Al confirmar esto u obtener la ruta completa de la carpeta con los archivos `.json` podemos continuar.

1.8.5 Procesando y cargando los datos

IMPORTANTE

El proceso actual carga específicamente la parte `compiledRelease` de cada documento OCDS, esto en función de poder realizar el análisis sobre la ultima versión disponible de los *releases* OCDS, se recomienda leer los capítulos anteriores antes de proceder

En esta misma carpeta tenemos disponible otra herramienta diseñada para la carga de los datos, que también hace uso de un contenedor Docker. Usaremos dos comandos unicamente: el primero para preparar el contenedor, el segundo para ejecutarlo.

```
docker build . -t logstash-sfp-compranet-ocds:latest
```

> Con este comando Docker estará preparando el contenedor con todo lo necesario para procesar y cargar los datos.

Una vez finalizado, podemos ejecutar el proceso de carga según el sistema operativo disponible.

Linux/Ubuntu o Mac

```
docker run --net="host" -v $HOME/Descargas:/input logstash-sfp-compranet-ocds
```

Windows

```
docker run --net="host" -v %HOMEPATH%\Descargas:/input logstash-sfp-compranet-ocds
```

> Este comando utilizará el contenedor preparado con anterioridad ejecutando el proceso y carga de los datos. Para mayores detalles puede consultar la documentación técnica de este proceso.

La pantalla ahora debe mostrar información del proceso. Esto puede tomar algunos minutos.

Al finalizar exitosamente deberemos ver la leyenda: `Carga finalizada` y `Kibana esta listo`.

Ahora podremos visitar la pagina de Kibana y consultar ver los datos cargados.

Para conocer más sobre los detalles técnicos de como logramos hacer la carga de los datos, en la siguiente sección hablaremos de como utilizamos LogStash para este proceso.

Extra: Descargando los datos OCDS directo de la API de datos.gob.mx

Anteriormente se explicó cómo descargar el conjunto completo de los datos en OCDS mediante un sólo archivo, en esta sección presentamos una alternativa para descargar sólo los contratos que buscamos o contratos más actualizados que aún no se hayan publicado en el archivo completo, para esto usaremos la API datos.gob.mx proporcionada por el Gobierno Mexicano.

> Para ver la documentación completa de la API se puede revisar la Guía básica de uso de la API donde se detallan las opciones específicas de filtrado.

Para realizar la acción de descarga y para manipular un poco los datos utilizaremos las herramientas: cURL y jq.

> El comando `curl` nos permitirá descargar la información de forma automática mientras el comando `jq` nos ayudará a darle un formato manejable a los datos JSON. Más adelante en este manual se incluye una breve introducción a `jq`.

Se pueden instalar ambos programas de forma local, por ejemplo para Linux Ubuntu con una instrucción como:

```
sudo apt-get install -y curl jq
```

Para Windows o Mac, tendríamos que descargar los archivos ejecutables por separado, pero también tenemos la opción de usar el *contenedor docker* incluido en el presente código, para ello solo tenemos que ejecutar el siguiente comando de docker:

```
docker run --rm -it -v $HOME/Descargas:/input --entrypoint=bash logstash-sfp-
↪compranet-ocds
```

> Recuerda que `$HOME/` es una abreviación propia de sistemas Linux y Mac, en Windows usaremos `%HOMEPATH%\`

Al ejecutar éste comando obtendremos una nueva línea de comandos, que debe lucir como:

```
bash-4.2$
```

En esta [línea de comando]((https://es.wikipedia.org/wiki/Bash) ya podremos ejecutar los comandos presentados a continuación.

Para descargar los últimos 100 procesos de contratación y guardarlo en un archivo `.json`:

```
curl https://api.datos.gob.mx/v2/contratacionesabiertas | jq -crM ".results" >␣
↪contratacionesabiertas_ultimos_100.json
```

> Hay que considerar que para conservar los archivos creados dentro del contenedor deberemos moverlos o crearlos en las carpetas compartidas entre la computadora y el contenedor. De lo contrario, estos archivos se perderán al momento de «apagar» el contenedor.

Para descargar los procesos de contratación que involucran a una unidad compradora determinada (limitado a 1000, pero se puede cambiar)

```
curl https://api.datos.gob.mx/v2/contratacionesabiertas?records.compiledRelease.
↪parties.name=Servicio%20de%20Administraci%C3%B3n%20Tributaria&pageSize=1000&page=1␣
↪| jq -crM ".results"  > contratacionesabiertas_SAT_1000.json
```

Para comprender mejor este último comando, vamos a detallarlo parte por parte:

- Primero se invoca a `curl`

- Luego se incluye la dirección URL base de la API: https://api.datos.gob.mx/v2/contratacionesabiertas
- A continuación los parámetros de filtrado:
  - `records.compiledRelease.parties.name`: filtra por el valor de ese campo, es decir, el nombre de alguna de las partes del contrato.
  - `pageSize`: especifica cuántos resultados devolverá en cada pedido
  - `page`: permite ir pidiendo las páginas siguientes en caso de que haya más de una.
- Luego usamos el comando `jq` para extraer únicamente la parte de los resultados.
- Finalmente indicamos que el resultado de la operación debe ser almacenado en un archivo, es importante que este nombre de archivo represente la consulta realizada para simplificar luego el archivado.

  Estos archivos deben almacenarse y tratarse de la misma forma que en la sección anterior, poniéndolos en la carpeta de descargas, para poder continuar con el próximo paso.

1.9 Procesamiento de datos con Logstash

Ahora que hemos logrado poner en marcha la plataforma podemos ahondar en los detalles técnicos de la colección, procesamiento e indexación de los datos, que como habíamos revisado con anterioridad es la tarea realizada por la herramienta Logstash.

IMPORTANTE: Todo lo mencionado a continuación se encuentra implementado en el código incluido en los contenedores Docker. NO es necesario repetir estos pasos, unicamente se menciona aqui para entender mejor el proceso.

1.9.1 Preparación de los datos OCDS por paquetes

Formato disponible y formato requerido

El archivo obtenido desde gob.mx se presenta en formato de colección de Paquete de Registros

> El esquema de paquete de registros (record package) describe la estructura del contenedor para publicar registros. Los contenidos de un registro se basan en el esquema de entregas (releases). El paquete contiene metadatos importantes.

```
[
    { paquete de registros ocds },
    { paquete de registros ocds },
]
```

Esto se traduce en una estructura como la siguiente:

```
[
    {
        "uri": "..."
        "version": "..."
        ... otros meta datos ...
        "records": [
            { documento ocds },
            { documento ocds },
            ...
        ]
    },
    {
```

(continué en la próxima página)

(proviene de la página anterior)

```
        "uri": "..."
        "version": "..."
        ... otros meta datos ...
        "records": [
            { documento ocds },
            { documento ocds },
            ...
        ]
    }
]
```

Para poder trabajar con este documento necesitaremos convertirlo a un formato donde cada línea del archivo sea un documento OCDS.

```
{ documento ocds }
{ documento ocds }
```

De esta forma podremos procesarlo con Logstash para después enviar los documentos uno a uno a ElasticSearch.

Convirtiendo el formato con la herramienta `jq`

Para poder trabajar con archivos JSON existe una herramienta disponible llamada jq de código libre y licencia MIT.

Esta herramienta nos permitirá manipular el documento JSON y llevarlo al formato requerido. Una vez que tenemos instalada esta herramienta y disponible el comando `jq` podemos usar un comando como:

```
jq -crM ".[].records[]" "archivo.json" > "archivo.ocds_por_linea"
```

> Recomendamos ampliamente consultar el manual de `jq` pero a continuación explicaremos qué hace este comando específico.

```
jq
    -c = Presenta el documento JSON de forma compacta
    -r = Presenta el documento JSON con valores sin formatos especiales
    -M = Sin color (Monocromatico)
    ".[].records[]" = Filtro de jq
    "archivo.json" = El archivo por leer
    "archivo.ocds_por_linea" = El archivo generado con el resultado
```

El filtro jq y la estructura de datos

El filtro es la parte más importante de este comando; para entenderlo debemos revisar con cuidado la estructura de datos presentada en el archivo original.

```
[
    {
        "uri": "..."
        "version": "..."
        ... otros meta datos ...
        "records": [
            { documento ocds },
            { documento ocds },
            ...
```

(continué en la próxima página)

(proviene de la página anterior)

```
        ]
    },
    ...
]
```

Para efectos de este proyecto nos interesa obtener cada documento OCDS por separado, de acuerdo a la notación de documentos JSON una ruta para acceder a ellos sería:

1. Entremos a cada elemento del arreglo raíz: `.[]`
2. De cada elemento, entremos a la propiedad records: `.records`
3. Obtengamos cada elemento de este arreglo: `.records[]`

Uniendo todas las instrucciones y en notación de filtro de jq obtenemos: `.[].records[]`

Los archivos producidos por este comando son adecuados para procesarlos con Logstash, así que continuemos con la creación del pipeline, pero primero revisemos algunos conceptos importantes.

1.9.2 Conceptos básicos para Pipelines de Logstash

Ahora que estamos listos para enviar los datos a Logstash, revisemos algunos conceptos requeridos para entender mejor las mecánicas de Logstash.

Sintaxis

Las definiciones de Pipelines para Logstash utilizan un lenguaje similar a bloques de código de programación simplificado.

Cada filtro o plugin es definido por un bloque:

```
bloque {

}
```

Algunas veces estos bloques pueden estar vacíos

```
bloque { }
```

Pero comúnmente utilizaremos opciones y argumentos para estos bloques, y esto se define como:

```
bloque { opcion => valor }
```

Los valores de las opciones pueden ser de distintos tipos:

- Texto `opcion => "Texto"`
- Numerico `opcion => 123`
- Boolean (Verdadero / Falso) `opcion => true` o `opcion => false`
- Arreglos `opcion => [ "Texto", 123, false ]`

  Los arreglos son conjuntos de otros tipos.

1.9.3 Pipeline

En el archivo pipeline.conf podemos encontrar el pipeline ya diseñado para este dataset; revisemos cada uno de los bloques que lo componen.

Entrada (input)

Este componente le indica a Logstash de dónde y cómo leerá los datos originales.

```
input {
  stdin {
    codec => "json"
  }
}
```

Para este pipeline hemos decidido leer el archivo desde la entrada estándar del programa, por cada línea de texto que reciba el programa esta será tratada como un documento JSON y almacenada en memoria para el siguiente paso.

Transformación (filter)

Este bloque le indica a Logstash qué debe hacer con cada uno de los registros que ha leído desde el módulo de Entrada.

```
filter {
  ruby {
    code => '
      event.get("[compiledRelease]").each do |k, v|
        event.set(k, v)
      end
    '
    remove_field => [ "releases", "compiledRelease", "host", "path" ]
  }
}
```

Este puede ser el proceso más complicado del Pipeline, y también el más interesante y poderoso para nuestras tareas.

Este bloque se compone por una serie de filtros que actúan de forma secuencial, en este caso solo ocupamos un filtro: ruby.

Filtro Ruby

> Este filtro es más avanzado y requiere de conocimientos de programación en lenguaje Ruby.

El objetivo de esta sección es tomar de cada documento JSON recibido la propiedad `compiledRelease`, y a su vez, leer cada propiedad que lo compone, y copiarla sobre la raíz del documento.

Ejemplo

```
{
  "compiledRelease": {
    "a": "A",
    "bc": [ "B", "C" ],
    "tercero": {
      "a": "3.A",
      "b": "3.B"
    }
```

(continué en la próxima página)

(proviene de la página anterior)

```
  }
}
```

Sería transformado como:

```
{
  "a": "A",
  "bc": [ "B", "C" ],
  "tercero": {
    "a": "3.A",
    "b": "3.B"
  }
}
```

Al final la propiedad `compiledRelease` es removida de la misma forma que `releases`, `host` y `path`.

Salida (output)

Esta sección le indica a Logstash qué debe hacer con los nuevos documentos, en nuestro caso queremos que los resultado sean enviados a ElasticSearch.

```
output {
  file {
    path => "/logs/sfp-compranet-ocds-pipeline.log"
    create_if_deleted => true
    write_behavior => "overwrite"
  }
  elasticsearch {
    index => "${ES_INDEX}"
    hosts => ["${ES_HOST}"]
    document_id => "%{ocid}"
  }
}
```

Aquí realizamos dos cosas:

1. Guardar en un archivo log todos los documentos procesados, uno por cada línea.
2. Enviar los documentos a ElasticSearch.

Para lo primero utilizamos el Plugin Output File y en las opciones especificamos el nombre del archivo log, que debe ser creado si no existe y que debe sobrescribir lo existente.

Para enviar los documentos a ElasticSearch usamos otro plugin que dispone de múltiples opciones; en nuestro caso especificamos tres pero recomendamos consultar el manual Output ElasticSearch.

Las opciones utilizadas son las siguientes:

- `index`: Indica el nombre del indice al que vamos a enviar el documento.
- `hosts`: Indica el `hostname` del servidor ElasticSearch.
- `document_id`: Esta opción es **MUY** importante ya que permite que Logstash identifique el documento con un identificador único, que a su vez permitirá a ElasticSearch saber cuando un documento ya existía previamente. En este caso el documento OCDS tiene un id único llamado `ocid`.

Como pudimos constatar la creación de un pipeline para procesamiento con Logstash es la codificación de un proceso lógico determinado. Cada dataset puede requerir distintos procesos, pero ahí radica el poder de Logstash que nos permite plasmar estos pasos de forma concisa y ordenada.

1.10 Resumen

Llegados a este punto hemos logrado tomar, procesar y visualizar datos publicados por el Gobierno Mexicano sobre sus contrataciones, en este caso específico bajo el formato estándar de contrataciones abiertas (OCDS).

Los presentes apuntes aunque creados para este caso de uso pueden ser replicados para otros, resaltando la importancia de los fundamentos de la plataforma Elastic (ELK).

Procesar, Indexar y Visualizar cualquier conjunto de datos abiertos es posible con estos conocimientos básicos.

A manera de resumen, recordemos los siguientes puntos.

Plataforma ELK

1. Existen 3 componentes de la plataforma ELK: ElasticSearch, Logstash y Kibana, cada uno con una tarea específica:
   - ElasticSearch almacena e indexa la información, es «la base de datos».
   - Kibana visualiza y ayuda a consultar la información.
   - Logstash compila, transforma e inserta los datos originales en ElasticSearch.
2. Una vez iniciado un servidor de ElasticSearch con Kibana podemos comenzar a enviar documentos al mismo para ser indexados.
3. Logstash es una herramienta muy flexible para tomar una colección de datos, leerla, transformarla para finalmente enviarla a ElasticSearch.
4. Logstash utiliza «Pipelines» para procesar los datos, éstos están compuestos de 3 partes: Entrada, Filtro, Salida.
5. El Pipeline está escrito en un «lenguaje» propio que describe cada proceso de forma lógica y clara, con la flexibilidad disponible para realizar acciones complejas con instrucciones de código de programación.
6. Una vez escrito el Pipeline éste puede ser usado múltiples veces, incluso para crear índices distintos dentro de un mismo servidor ElasticSearch.

1.11 Introducción a Kibana

> PODER pone a disposición de los lectores del manual una instancia de Kibana con el Concentrado de contrataciones abiertas de la APF, los mismos con los que se ejemplifica el manual. Para poder acceder a ellos, favor de solicitar una cuenta al email kibana@quienesquien.wiki.

Kibana es una herramienta de análisis y visualización de datos Elasticsearch integrada en el «stack ELK», cuyo funcionamiento técnico se explica en la sección «Las herramientas Elastic» de este Manual.

Para acceder a la herramienta tendemos que ir a una url (si está usando la instancia de PODER es kibana.quienesquien.wiki) donde habitualmente tendremos que poner nuestro usuario y contraseña. Las funciones multiusuario dependen de cada instancia por lo que no se explicarán aquí. En cualquier caso, es importante advertir al lector que tanto se puede encontrar accediendo a instancia limpia como a instancia donde ya hay búsquedas, visualizaciones y dashboards guardados.

Al acceder a Kibana podemos realizar varias acciones en este manual abordaremos solo cuatro apartados:

- Discover
- Management
- Visualize
- Dashboards

1.12 Descubrir (Discover)

Tal y como su nombre indica, este primer apartado sirve para hacer una primera exploración de los datos. La pantalla de búsqueda está dividida en tres partes principales:

- Un buscador
- Un mapeo de campos
- Un espacio de resultados

"Discover"

Las principales acciones tanto para buscar como para configurar la visualización de la pantalla, son:

1. **Selector de índices**: En el desplegable se encuentran los distintos índices importados en la instancia de Kibana, este desplegable nos permite mover a través de ellos. Incluso algunos filtros se mantienen entre índices si hay campos coincidentes.

2. **Buscador**: Nos permite hacer toda una serie de preguntas sobre nuestra base de datos, una forma de ver si nuestra query está funcionado es comprobar el recuento de «hits» que aparece justo encima del buscador. Algunas de las queries más habituales que se pueden usar son:

Para conocer más opciones repasar la documentación de Query String Query y de Lucene Query Syntax.

1. **Filtros**: Los filtros gráficos pueden hacer más o menos las mismas operaciones de filtro que acabamos de ver en el buscador, con la ventaja de que se pueden sumar varios filtros con facilidad y que hay una opción para editar el filtro y hacerlo mucho más complejo siguiendo este tutorial. Si se están haciendo filtros sobre campos que contienen strings verán que aparecen duplicadas, una con el nombre definido y otra que termina en *.keyword*, se recomienda usar la segunda.
2. **Available fields**: La barra lateral sirve para poder inspeccionar las cabeceras de los datos, dar una primera visión de los datos que contienen y configurar el panel de resultados.
   - *Configuración*: La ruedita que está al lado de «Availaible Fields» despliega una serie de opciones para que se muestren más o menos campos. En caso de datos no tabulares, como los de OCDS, se aconseja desmarcar «Hide missing fields», para que se muestren los campos que están dentro de otros campos.
   - *Campos*: Todos los campos van acompañados de un símbolo que identifica el tipo de datos que contiene, el reloj cuando es temporal, el numeral o tecla gato para identificar numero, la «t» para identificar los textos o strings, un esfera mitad negra significa que el campo es un booleano, y el símbolo interrogante que desconoce que tipo de campo tiene, normalmente será porque contiene a su vez más campos a su interior. Al cliquear sobre un campo se desplegará un grafico que mapea los primeros 500 valores del campo.
   - *Add*: El botón add, que está iluminado por el círculo rojo, sirve para que el panel de resultados en lugar de mostrar toda la tira de datos muestre solo aquel o aquellos valores seleccionados. Por ejemplo podríamos usar el campo `tender.numberOfTenderers` que muestra el número de oferentes en cada proceso de contratación, y una vez lo tenemos en el espacio de resultados ordenarlo para poder ver que contratos tiene más contendientes.
3. **Save**: Como se dice al principio del apartado la opción visualizar en el apartado Discover sirve para una primera exploración, pero una vez ya conseguimos los resultados deseados podemos guardar esta búsqueda para graficarla o mandarla a un dashboard.

1.13 Panel de administración (Management)

El panel de administración de datos de Kibana permite modificar las propiedades de los datos importados, reformatearlos o crear nuevos campos calculados.

En la primera pantalla de «Management» podemos administrar tanto la base de datos de «Elaticsearch» como la visualización en «Kibana». En este Manual solo abordaremos la parte de «Index Patterns», que nos permite tanto reconfigurar las propiedades de los campos como crear campos calculados. Para expandir los conocimientos de la administración de Kibana se recomienda consultar el manual oficial.

1.13.1 Reconfigurar un campo (Field)

Al entrar encontraremos una tabla con un buscador con todos los campos de cada uno de los íindices. En cada campo hay información del tipo, del formato (se da de forma individual), si es buscables, agregable (verán que los strings terminados en *keyword* son agregables) y si lo hemos excluíido. Al final de cada renglón aparece un síimbolo de edición, dónde podremos reformatearlo.

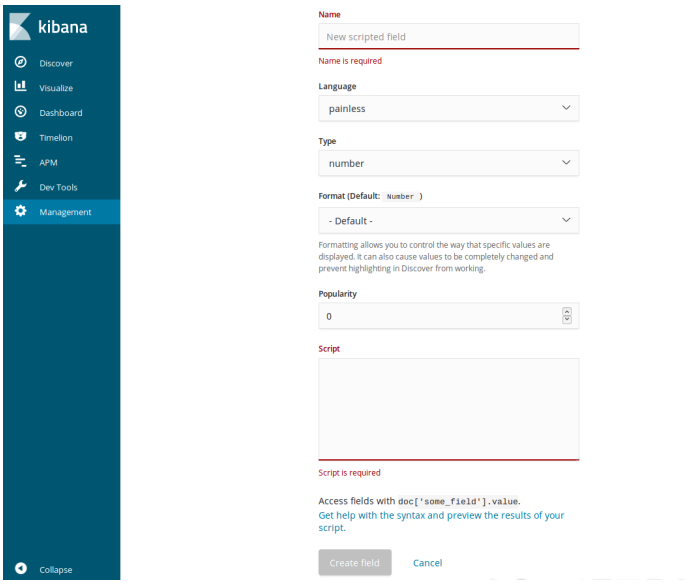
Un caso de uso sería que importamos un identificador que solo tiene números (award ID, cóodigo postal, partidas presupuestarias, etc.) y está identificado como número y no como string. Para modificarlos solo deberíamos entrar en la pantalla y en la parte de «format» ponerle el nuevo formato y cómo queremos que se visualice.

1.13.2 Campos calculados (Scripted filds)

Kibana permite crear nuevos campos a partir de cálculos y que los podamos usarpoder usarlos de forma permanente en la aplicación. Para explicar cóomo funcionan, calcularemos la cantidad de días transcurridos desde la publicación

de licitación hasta la recepción de oferta.

En la pantalla «Index patterns» hay tres pestañas, «Fields», «Scripted filds» y «Source Fields»;, cliquearemos en «Sripted fields» donde aparecen los campos calculados y luego en el botón «Add scripted field» para empezar a calcular.



"Scripted Fields"

Antes de empezar hemos de detectar los campos con los que vamos a trabajar; en nuestro caso serán `tender.awardPeriod.startDate` y `tender.awardPeriod.endDate` que ambos son fechas y Kibana las reconoce como fechas. El resultado que queremos es el número de días de diferencia.

Siguiendo el formulario de la imagen:

1. **Name:** El nombre con el que identificaremos nuestro campo, por ejemplo `tender.awardPeriod.duration` para seguir con el mismo lenguaje del dataset de OCDS.
2. **Language:** Es un desplegable con las opciones «painless» y «expresión», se sugiere trabajar con painless ya que es la sintaxis con la que nos familiarizamos en Discover y la que seguro va ser soportada en próximas versiones.
3. **Type:** Desplegable pare elegir el tipo de campo que vamos a generar, en este caso usaremos «Number».
4. **Format:** Definiremos el número como «Duration», y allí nos aparecen dos desplegables más. El «Input format» donde seleccionaremos «Milliseconds» (porque así lo definimos en la fórmula) y el «Output format» como «Days» ya que estamos buscando la cantidad de días (Nota: la opción «Human Redable» dificulta las queries sobre el campo). También tenemos un campo numérico para los decimales, seguiremos con 2 decimiales por preferencias del autor.
5. **Popularity:** Este campo numérico Kibana lo va calculando a partir del uso para mostrar los campos destacados en varias pantallas de la aplicación. Si queremos tenerlo destacado desde que terminamos se sugiere ponerle un valor alto, en este caso le pondremos un 10.
6. **Script:** Este es el campo donde haremos nuestros cálculo; para una comprensión profunda sobre como hacer scripts se sugiere leer la guía oficial. El sript que vamos a usar será:

```
(doc['tender.awardPeriod.endDate'].value.getMillis() - doc['tender.awardPeriod.
↪startDate'].value.getMillis())
```

El sript lo que hace es llamar a los campos de inicio y fin de la apertura de licitación, mostrar sus valores, convertirlos a milisegundos y restar el fin del inicio. Para hacer este resultado usable y que podamos trabajar en el resto de la plataforma lo hemos formateado a días con dos decimales.

Analizando al detalle los elementos del sript:

- `doc['Nombre.Campo']` Sirve para llamar al campo de nuestra base datos.
- El `.value` nos devuelve el valor que nos servirá para hacer operaciones matemáticas.
- `.getMillis()` convierte el valor en milesegundos
- – Es el operador de resta, se pueden usar otros operadores matemáticos.

Para más detalles consultar la sintaxis completa.

7. **Create field:** Este es el botón final que crea el campo para toda la aplicación. En caso que el sript dé error, nos saltará un mensaje de advertencia y no lo procesará en la aplicación.

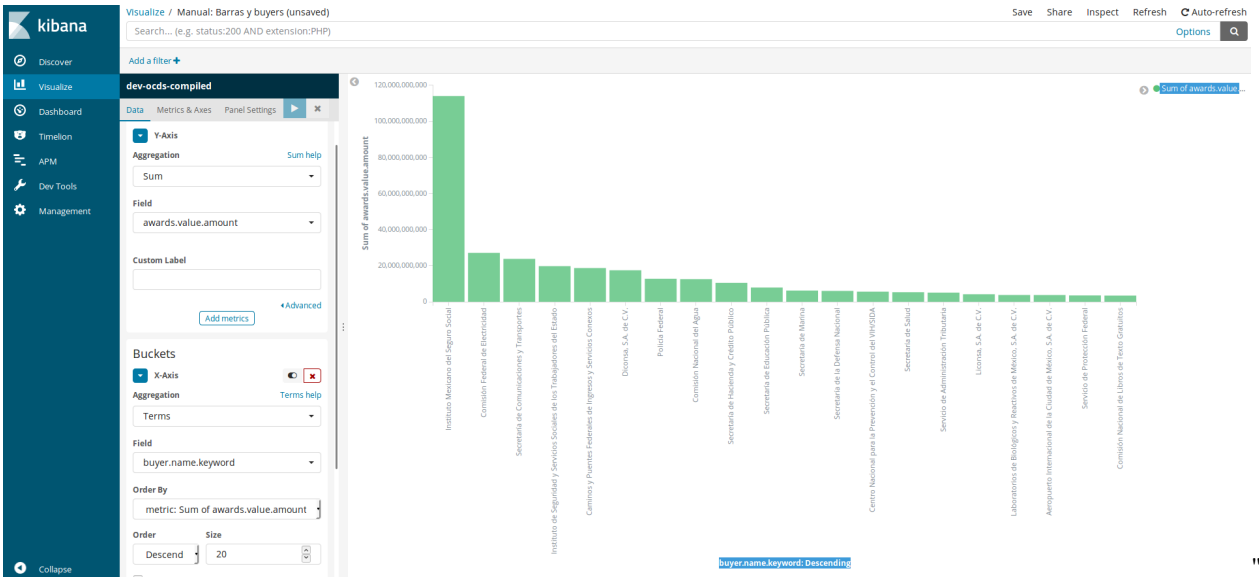
1.14 Visualizar (Visualize)

Kibana además de un gran buscador es un potente visualizador de datos, que nos permite crear y guardar gráficas de muchos tipos para el análisis de datos.

Una vez entramos en el apartado visualizar nos aparece una tabla con las visualizaciones ya guardadas y un botón azul con un símbolo +; al cliquearlo nos aparece toda una serie de opciones para trabajar y visualizar datos.

1.14.1 Gráfico de barras

Los gráficos de barras sirven para comparar un mismo rango de datos (por ejemplo el importe total) en distintas instancias (en este caso las dependencias de gobierno). Al seleccionarlo llegamos a una pantalla como la siguiente con todos los valores vacíos.



"Gráfico Barras"

Para replicar el gráfico el proceso es:

- **Y-Axis**
  - *Agregation:* En este punto seleccionamos cómo se agregaran los datos en nuestro gráfico. Por defecto nos da «Count», pero para conseguir los valores totales hemos de seleccionar la opción «Sum» y se abre un nuevo desplegable llamado *Field* donde seleccionaremos `awards.value.amount`, que es el campo dónde está el valor de contrato.

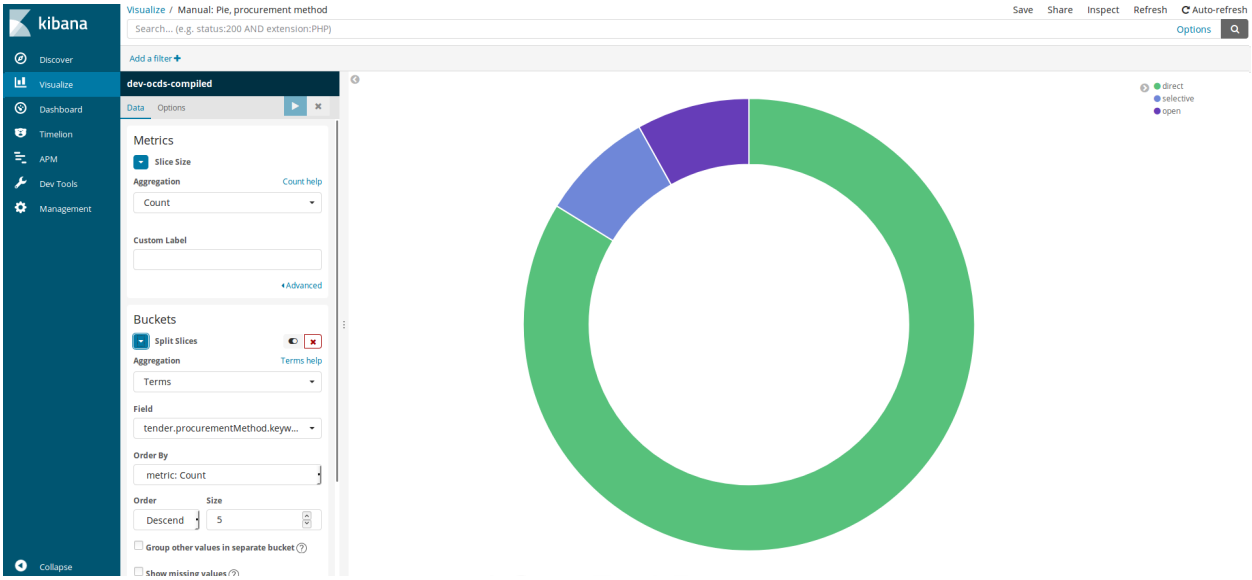
- *Custom label:* Campo de personalización del gráfico.
- *Add metric:* El botón azul sirve por si queremos añadir otra métrica en el eje de la Y. Si quisiéramos hacer un gráfico con barras agrupadas deberíamos seguir por este camino.

- **Buckets / X-Axis** - Aunque hay otras dos opciones **Split Series** y **Split Chart** para realizar este gráfico nos quedaremos con la primera opción.
  - *Agregation:* Nos abre un desplegable con múltiples opciones, seleccionaremos «Terms» y aparecerá una serie de campos.
  - *Field*: Seleccionamos la opción `buyer.name.keyword` (observarán que no hay la opción sin keyword).
  - *Order by*: Ordenaremos por «metric.Sum of awards.value.amount», el valor que pusimos en el inicio, aunque también podríamos ordenar alfabéticamente o definir otro orden de la base de datos.
  - *Order*: Seleccionaremos «Descending» para que aparezcan los mayores primeros. Si hubiéramos seleccionado otro *Orden by* las opciones se modificarían.
  - *Size*: El número de valores que vamos a mostrar en el gráfico, 20 es un valor razonable
  - *Custom label:* Campo de personalización del gráfico.

Cuando ya tenemos todo el panel completo hemos de darle al botón de play en el recuadro azul y el gráfico aparecerá en la pantalla. Podemos modificar las distintas opciones para analizar a profundidad.

1.14.2 Gráfico de tarta

Los gráficos de tarta para saber qué peso tiene cada uno de los elementos (procedimientos de contratación) sobre el conjunto (todo el dataset).



"Grafico de tarta"

Para replicar el gráfico el proceso es:

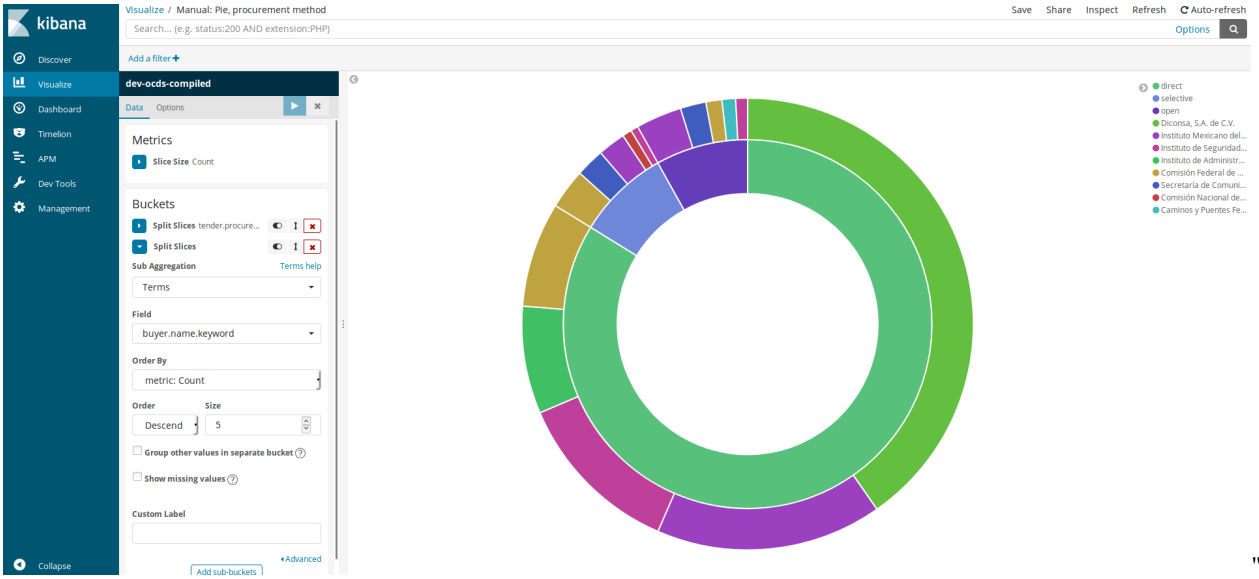
- **Metrics**
  - *Agregation:* Dejaremos seleccionada el «Count»
- **Bucket / Splits Slices** ya que nos interesa generar los espacios de la tarta y no crear varias gráficas de tarta.
  - *Agregation:* Otra vez Terms

- *Field*: `tender.procurementMethod.keyword`
- *Order by*: «Count»
- *Oder* Descent
- *Size*: Dejamos 5 aunque son tres.

Le damos al «play» y aparecerá el gráfico. Pero una vez visualizado queremos ver cuáles son las dependencias que más han usado este dataset para y para esto hemos de cliquear al botón con letra azul «Add sub-buckets» que está en el apartado de Buckets. Una vez allí repetiremos el proceso.

- **Bucket / Splits Slices**
  - *Agregation:* Otra vez Terms
  - *Field*: `buyer.name.keyword`
  - *Order by*: «Count»
  - *Oder* Descent
  - *Size*: Dejamos 5 aunque son tres.

Cuando le demos al play nos saldrá esta gráfica con las 5 dependencias que más veces han echo ese tipo de contratación en el dataset.

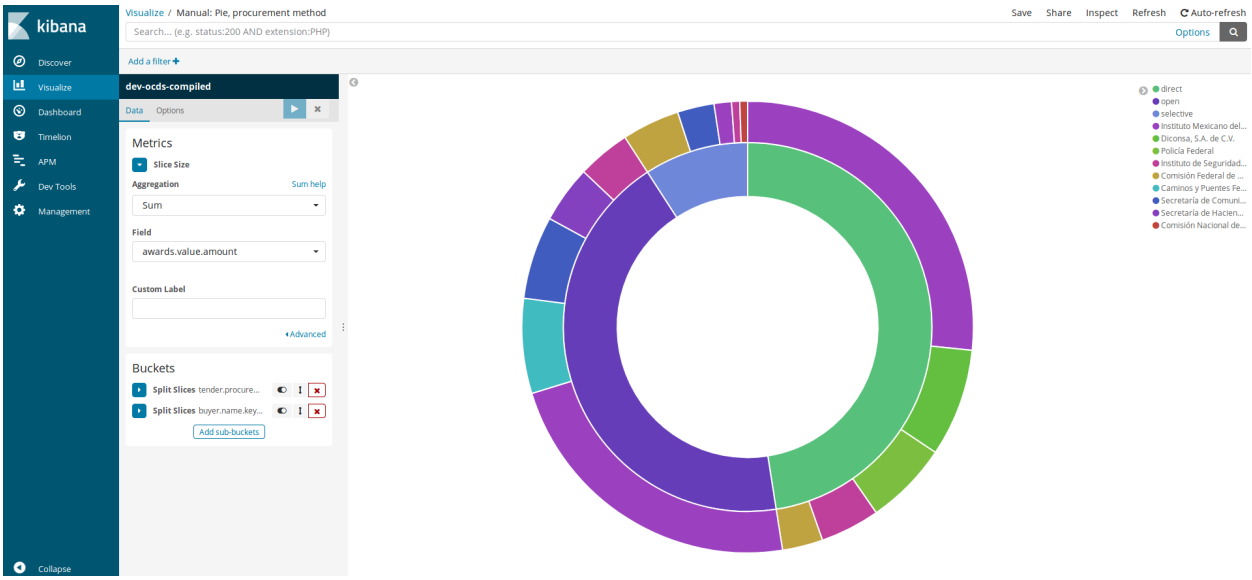


"Grafico de tarta2"

Siguiendo con el análisis queremos ahora cambiar el agregado, no queremos que sea cuenta sino importe total por procedimiento y dependencia.

- **Metrics**
  - *Agregation:* seleccionaremos la opción de «Sum» y en el nuevo desplegable *Field* seleccionaremos `awards.value.amount`

Le volvemos a dar al play y nos da la siguiente gráfica.

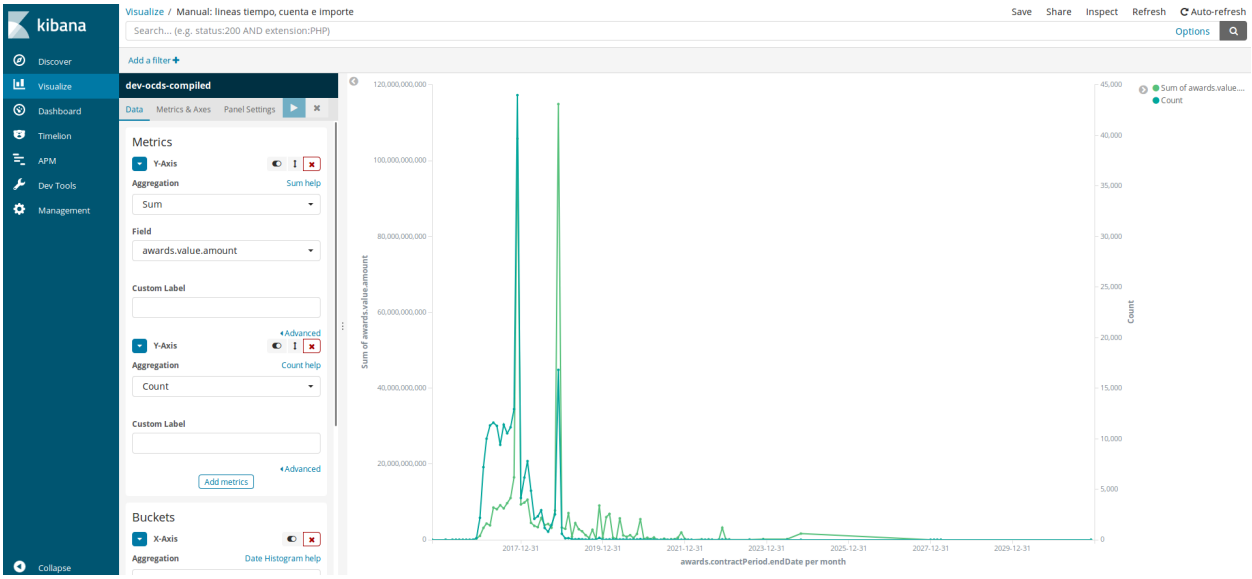


"Grafico de tarta3"

Podemos seguir analizando, añadiendo y quitando valores. Para acelerar el proceso, al lado de cada «Split Slices», hay tres botones que permiten hacer la acciones «Mostrar/esconder», «modificar el orden de aparición» y «borrar el slice». Cada vez que se hace una modificación hay que apretar el botón de play.

1.14.3 Gráfico de líneas

Los gráficos de líneas nos sirven para mostrar la evolución durante el tiempo; en esta ocasión haremos un gráfico con dos valores: cuenta e importe.



"Grafico de lineas"

- **Metrics**
  - *Agregation:* seleccionaremos la opción de «Sum» y en el nuevo desplagable *Field* seleccionaremos `awards.value.amount`
  - Cliquearemos en *Add metric*

    - *Agretation:* Dejaremos seleccionada el «Count»

- **Buckets / X-Axis** - Aunque hay otras dos opciones **Split Series** y **Split Chart** para realizar este gráfico nos quedaremos con la primera opción.
    - *Agregation:* Esta vez seleccionaremos «Date Histogram», ya que queremos hacer una serie temporal.
    - *Field:* Usaremos el campo `awards.contractPeriod.endDate`, la fecha en la que se otorga el contrato.
    - *Interval:* Seleccionaremos «Monthly», ya que nuestro dataset contiene datos de tres años.

Si le diéramos play ahora, el gráfico quedaría solo con la línea verde, y la azul quedaría rozando al 0, ya que los valores entre ambas son muy distintos. Para que esto aparezca bien, hemos de crear un segundo eje de valores en la gráfica. Para esto hemos de hacer clic en *Metrics & Axes* (el menú a la izquierda del botón play) y en el desplegable «Value Axis» elegir la opción «New Axis», que nos generará un nuevo eje que podemos editar en la siguiente caja **Y-Axis**.

Observando el gráfico podemos ver que termina en el 2020, cuando aún estamos en 2018. Para rectificar esto, podemos añadir un filtro a los datos, de la misma forma que lo hacemos en el apartado Discover

1.14.4 Otros gráficos disponibles

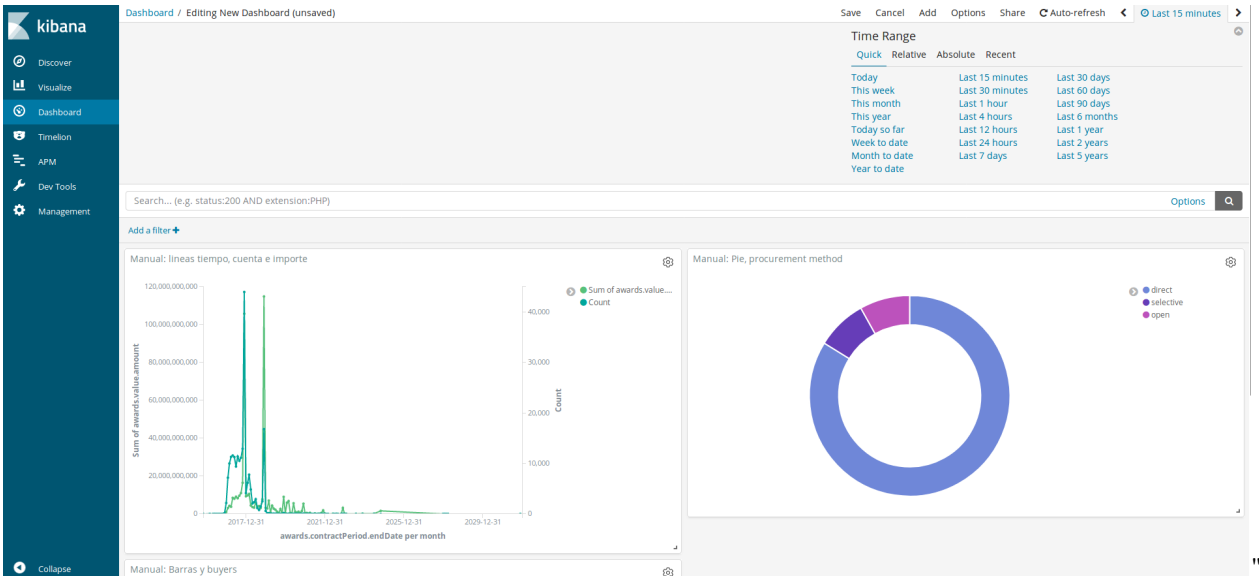
En Kibana hay muchas más opciones de crear gráficos, todos ellos con funcionamientos muy similares a los anteriormente descritos. Algunos recomendados son:

- **Tablas**: Permiten generar nuevas tablas y exportarlas a csv.
- **Área**: para valores acumulados en el tiempo.
- **Mapas**: cuando tenemos valores geográficos.
- **Redes**: Explorar las relaciones entre los distintos campos.

1.15 Creación de Paneles (dashboard)

Al tener toda una serie de gráaficas guardas que nos sirven para el análisis y con una fuente de entrada de datos constante, el dashboard nos permite hacernos tableros de control para ver como están entrando los datos. Su configuración es muy simple, solo hay que apretar el botón «Add», seleccionar las gráaficas y/o búsquedas que ya tenemos gravbadas y se mostraráan en el tablero, que podremos configurar a nuestro gusto.

Además (en la imagen está desplegado) tenemos un selector de tiempo para que rápidamente podamos seleccionar el intervalo de tiempo en el que tenemos interés.


kibana
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Management
Collapse
Dashboard / Editing New Dashboard (unsaved)
Save
Cancel
Add
Options
Share
Auto-refresh
Last 15 minutes
Time Range
Quick
Relative
Absolute
Recent
Today
This week
This month
This year
Today so far
Week to date
Month to date
Year to date
Last 15 minutes
Last 30 minutes
Last 1 hour
Last 4 hours
Last 12 hours
Last 24 hours
Last 7 days
Last 30 days
Last 60 days
Last 90 days
Last 6 months
Last 1 year
Last 2 years
Last 5 years
Search... (e.g. status:200 AND extension:PHP)
Options
Add a filter
Manual: lineas tiempo, cuenta e importe
Sum of awards.value....
Count
Sum of awards.value.amount
Count
awards.contractPeriod.endDate per month
Manual: Pie, procurement method
direct
selective
open
Manual: Barras y buyers

"Dashboard"