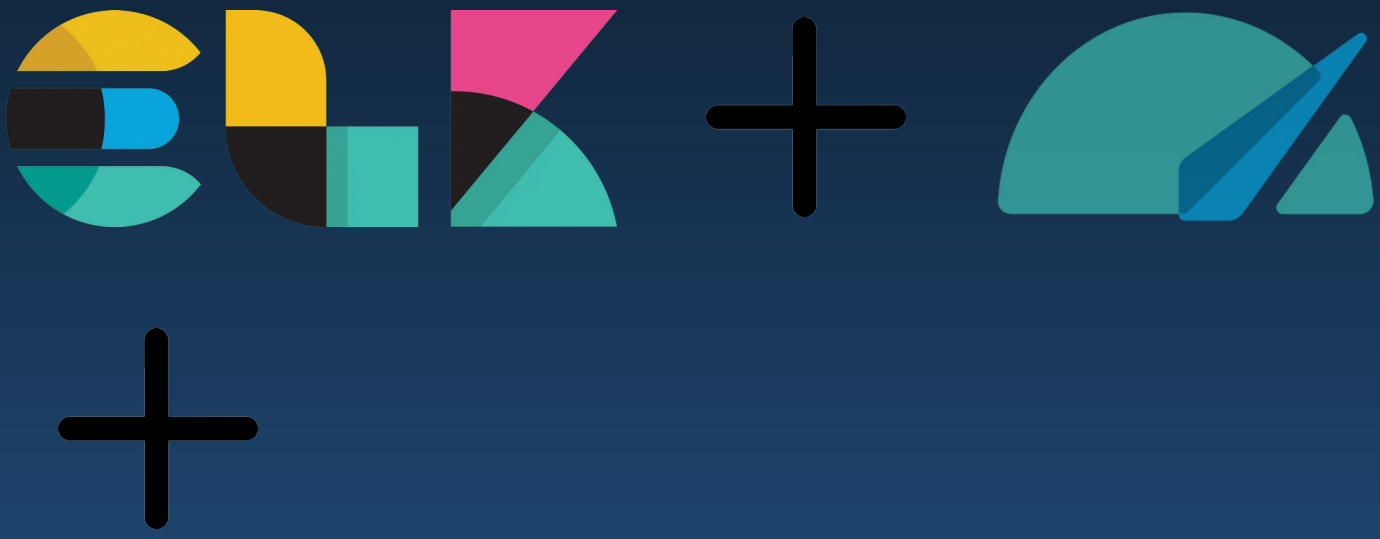
Monitorización y análisis de logs y métricas en tiempo real utilizando ELK stack, Metricbeat y Filebeat



Autor: Ángel Suárez Pérez
Centro Educativo: I.E.S Gonzalo Nazareno
Administración de Sistemas Informáticos en Red

INDICE

1. Introducción

Como administrador de sistemas informáticos, en muchos casos, es importante comprender la importancia de la monitorización y análisis de logs y métricas en tiempo real para el correcto funcionamiento y mantenimiento de los sistemas informáticos.
Para lograr este objetivo, en este documento se describirá como utilizar el stack ELK (Elasticsearch, Logstash y Kibana) junto a Metricbeat y Filebeat para la monitorización y análisis en tiempo real de logs y métricas de sistemas.

1.2 Motivación y justificación de su realización

He optado por este tema por varias razones, entre ellas, me interesaba aprender acerca de la monitorización y análisis de logs y métricas en tiempo real, investigando descubrí que es un tema relevante en el ámbito de la administración de sistemas y que a día de hoy es un tema muy demandado en el mercado laboral, de hecho, muchas empresas requieren profesionales capaces de utilizar la pila ELK que yo voy a utilizar para la realización del proyecto. Además existe una amplia documentación y una gran cantidad de recursos en línea para aprender sobre este tema, lo que hace más accesible su estudio y comprensión.

1.3 Objetivos y alcance del trabajo

En cuanto a objetivos que busco cumplir, son los siguientes:

- **Identificar y solucionar problemas en la infraestructura de la empresa de forma proactiva:** Buscamos anticiparnos a los fallos y a tomar medidas preventivas para evitarlos o solucionarlos antes de que afecten a los usuarios finales.
- **Facilitar la colaboración y la toma de decisiones:** De cara a ofrecer esta solución a alguna empresa, nuestro objetivo es facilitarles información detallada sobre sus sistemas y aplicaciones para que puedan mejorar la toma de decisiones.
- **Automatizar la recolección y el análisis de datos:** Esto nos permitirá ahorrar tiempo y recursos además de reducir la posibilidad de cometer errores humanos.

Al finalizar este proyecto busco tener un escenario capaz de monitorizar logs y métricas de una serie de máquinas con distintas funcionalidades y rendimiento y visualizar los datos recopilados desde Kibana

2. Marco teórico

En este apartado voy a hablar de los fundamentos teóricos sobre la monitorización y análisis de logs y métricas. También voy a mencionar algunas herramientas que podrían ser utilizadas, aunque no serán el foco principal, ya que, más adelante, profundizaremos en las herramientas que sí vamos a utilizar y sus conceptos relevantes.

2.1 Registro sobre análisis de logs

El registro y análisis de logs es una práctica fundamental en la gestión de sistemas informáticos. Los logs son registros detallados de eventos que se han producido en el sistema, y su análisis puede proporcionar información valiosa sobre el rendimiento, la seguridad y la estabilidad del sistema.

Los logs pueden contener información sobre errores, fallos, intentos de acceso no autorizado, actividad de usuarios, tráfico de red y otros eventos relevantes.

Una vez que se han recopilado los logs, el siguiente paso es su análisis. El análisis de logs puede ser manual o automatizado, y puede proporcionar información valiosa para la toma de decisiones y la resolución de problemas. Por ejemplo, el análisis de logs puede ayudar a identificar patrones de comportamiento anormal, detectar errores en el código de programación, detectar intentos de acceso no autorizado y optimizar el rendimiento del sistema.

2.2 Métricas y monitoreo de sistemas

El monitoreo y la medición de métricas son fundamentales en la gestión de sistemas informáticos. Las métricas son medidas cuantitativas que indican el rendimiento, la capacidad y el estado de los sistemas y aplicaciones. El monitoreo de métricas permite supervisar el estado del sistema y detectar problemas o cuellos de botella antes de que afecten el rendimiento o la disponibilidad del sistema.

Existen diferentes tipos de métricas que se pueden medir, como la utilización de la CPU, la memoria RAM, el tráfico de red, la latencia y la velocidad de transferencia de datos. Estas métricas se miden a intervalos regulares y se registran en logs para su posterior análisis.

Los datos recopilados se pueden utilizar para analizar y ajustar la configuración del sistema, lo que puede mejorar el rendimiento y la eficiencia.

2.3 Herramientas de monitoreo y análisis de logs y métricas

En la actualidad, el monitoreo y análisis de logs y métricas se ha vuelto esencial para el correcto funcionamiento de los sistemas informáticos. Para realizar esta tarea, existen diversas herramientas que ofrecen diferentes soluciones y características para cumplir con los objetivos deseados.

Si bien es cierto que personalmente me parece que la pila ELK y Metricbeat son muy interesante, no son la única opción disponible, y otras herramientas también pueden ofrecer soluciones efectivas para diferentes necesidades y situaciones. En este apartado, voy a presentar algunas de ellas que pueden ser muy útiles y que merecen ser consideradas. Algunas herramientas:

- ***Grafana***: Herramienta de visualización y análisis de datos, diseñada para proporcionar una interfaz de usuario intuitiva y fácil de usar para el monitoreo y análisis de métricas en tiempo real. Se integra con numerosas fuentes de datos y permite crear paneles personalizados y dashboards para monitorear diferentes sistemas y servicios.

- ***Nagios***: Herramienta de monitoreo de red y servicios que se utiliza para supervisar la disponibilidad y el estado de los servicios de red, servidores y dispositivos de red. Proporciona alertas en tiempo real y notificaciones en caso de que se produzcan problemas en los sistemas monitoreados.

- ***Prometheus***: Herramienta de monitoreo y alerta de métricas diseñada para trabajar con sistemas distribuidos y altamente escalables. Permite la recolección, el almacenamiento y la consulta de métricas en tiempo real, y también ofrece una plataforma para crear alertas en función de umbrales y condiciones definidas.

- ***Sensu***: Herramienta de monitoreo y análisis de eventos que se utiliza para monitorear y analizar servicios, infraestructura y aplicaciones. Proporciona una plataforma escalable y de alta disponibilidad para recopilar y procesar eventos en tiempo real, y también ofrece alertas personalizables para notificar a los equipos de operaciones de cualquier problema que se produzca.

- ***Zabbix***: Herramienta de monitoreo de red y servidores que se utiliza para supervisar y administrar sistemas y redes. Permite la recolección de métricas y datos de rendimiento, así como la configuración de alertas y notificaciones en caso de problemas en los sistemas monitoreados.

En definitiva, el objetivo es ampliar nuestro conocimiento y dar una visión más completa sobre las diferentes herramientas disponibles para realizar el monitoreo y análisis de logs y métricas en tiempo real, y no solo limitarnos a la pila ELK y Metricbeat.

3. Descripción de Elastic Stack, Metricbeat y Filebeat

3.1 Características y funcionalidades de Elastic Stack

3.1.1 Los comienzos de la pila ELK

La pila ELK fue creada por la compañía Elastic en 2010. Originalmente, la compañía se enfocó en el desarrollo de Elasticsearch, que es un motor de búsqueda y análisis de datos distribuido y escalable. A medida que Elasticsearch se volvió más popular, la compañía decidió expandir su oferta y desarrollar herramientas adicionales que pudieran integrarse con Elasticsearch para proporcionar una solución completa de recopilación, análisis y visualización de datos.

En 2013, la compañía lanzó Logstash, una herramienta de procesamiento de datos que permite la ingestión, transformación y envío de datos a Elasticsearch o a otros destinos. Posteriormente, en 2014, se lanzó Kibana, una herramienta de visualización de datos que permite la creación de paneles de control interactivos y gráficos para análisis de datos.

A medida que la popularidad de Elasticsearch, Logstash y Kibana creció, la comunidad de usuarios comenzó a referirse a esta combinación de herramientas como la pila ELK. Desde entonces, la compañía Elastic ha continuado desarrollando y mejorando la pila ELK, agregando nuevas funcionalidades y mejorando la integración entre las herramientas.

3.1.2 ¿Qué es Elastic Stack?

Elastic Stack es un conjunto de herramientas de software libre y de código abierto utilizadas para la búsqueda, análisis y visualización de datos. A continuación voy a hablar un poco sobre estas herramientas y algunos conceptos que considero de importancia.

3.1.3 Elasticsearch

Elasticsearch es un motor de búsqueda y análisis de datos distribuido y escalable. Algunas de sus características y funcionalidades incluyen:

- **Búsqueda de texto completo:** Elasticsearch es capaz de indexar y buscar grandes cantidades de texto en tiempo real, lo que lo hace ideal para aplicaciones de búsqueda de texto completo.
- **Escalabilidad:** Elasticsearch es altamente escalable y puede manejar grandes cantidades de datos y consultas de búsqueda.
- **Análisis de datos:** Elasticsearch permite realizar análisis en tiempo real de los datos indexados, incluyendo agregaciones, estadísticas y más.
- **Integración con otros componentes del Elastic Stack:** Elasticsearch se integra con otros componentes del Elastic Stack para proporcionar una solución completa de búsqueda y análisis de datos.

Cluster de Elasticsearch

Un clúster de Elasticsearch es un conjunto de nodos que trabajan juntos para proporcionar una plataforma de búsqueda y análisis escalable y tolerante a fallos. Los nodos en un clúster de Elasticsearch se comunican entre sí y se coordinan para indexar, almacenar y buscar datos.

Cada clúster de Elasticsearch tiene un nodo maestro, que se encarga de supervisar y coordinar las actividades del clúster. El nodo maestro es elegido automáticamente por el clúster y puede cambiar si el nodo actual falla o se desconecta. Además, el clúster puede tener varios nodos de datos, que almacenan y procesan los datos indexados. Los nodos de datos también pueden realizar búsquedas y consultas en los datos almacenados.

Tipos de nodos

Dentro de un clúster de Elasticsearch, podemos identificar distintos tipos de nodos, los cuales son:

MASTER NODE

Es el nodo responsable de coordinar las actividades del clúster, como la asignación de shards a los nodos de datos y la elección de un nuevo nodo maestro en caso de que el actual falle o se desconecte. Un clúster de Elasticsearch solo tiene un nodo maestro activo en cualquier momento.

DATA NODE

Es el nodo que almacena los datos y procesa las búsquedas y consultas en los datos almacenados. Los nodos de datos también pueden realizar operaciones de indexación, en las que se agregan nuevos documentos al índice.

INGEST NODE

Es el nodo que se utiliza para procesar y transformar los datos antes de que se almacenen en el clúster. Los nodos de ingestión se utilizan comúnmente para extraer, transformar y cargar (ETL) los datos en Elasticsearch.

COORDINATING NODE

Es un nodo especial que actúa como intermediario entre los clientes y los nodos del clúster de Elasticsearch. Su función principal es recibir las solicitudes de los clientes y dirigirlas a los nodos apropiados en el clúster que pueden procesar la solicitud de manera más eficiente.

¿Cómo almacena la información?

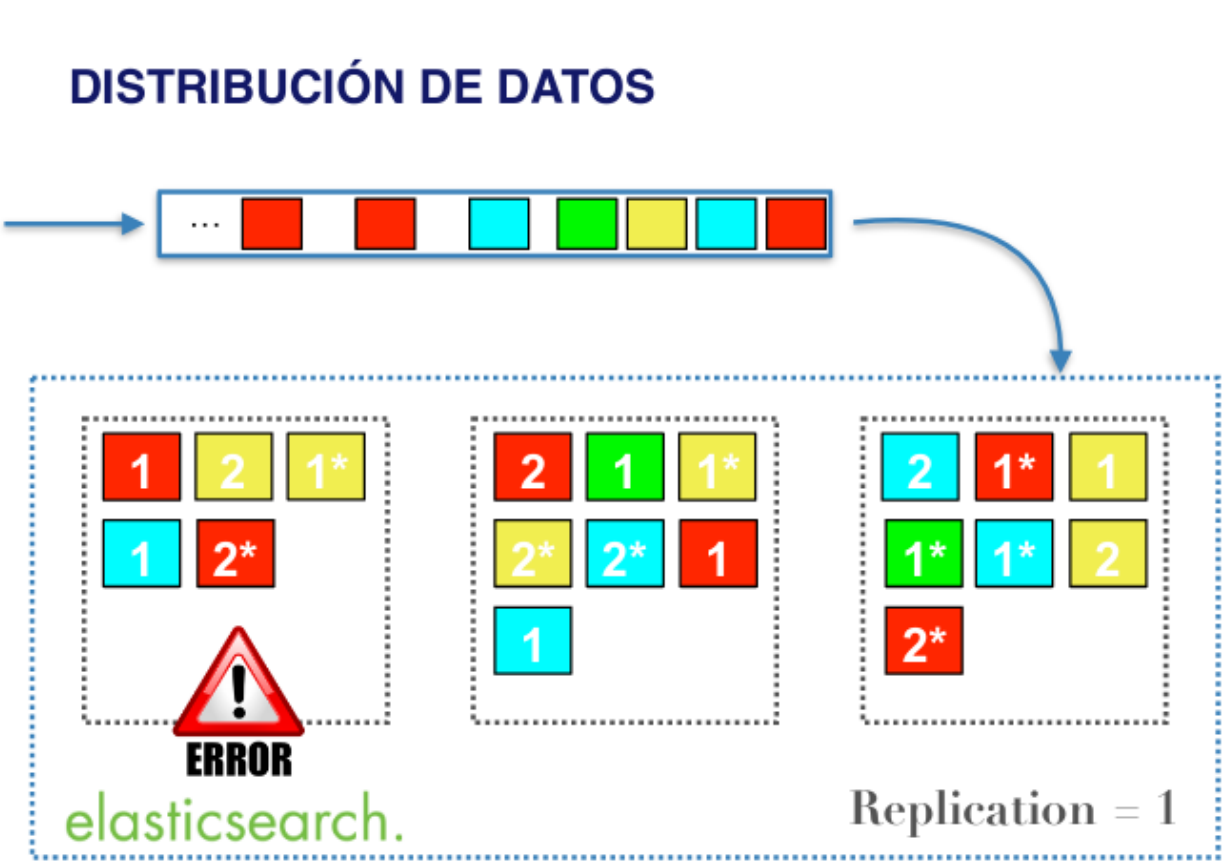
Digamos que al igual que en las bases de datos relacionales, elasticsearch utiliza algo parecido a lo que conocemos como tablas, solo que aquí se llaman índices y sus registros se llaman documentos.

Cuando agregamos un documento a elasticsearch este se divide en unos fragmentos llamados “shards” (definición completa en conceptos de importancia). Elasticsearch asigna automáticamente los shards a los nodos, utilizando un algoritmo que equilibra la carga de trabajo y asegura que los shards estén disponibles en caso de fallos.

Los shards se replican automáticamente para garantizar la disponibilidad y la tolerancia a fallos.

Las réplicas se distribuyen por diferentes nodos en el clúster y se actualizan en tiempo real para

mantener la coherencia de los datos.



3.1.4 Logstash

Logstash es una herramienta de procesamiento de datos que permite la ingestión, transformación y envío de datos a Elasticsearch o a otros destinos. Algunas de sus características y funcionalidades incluyen:

- **Ingestión de datos**: Logstash puede recibir datos de diferentes fuentes, incluyendo archivos de registro, bases de datos, servicios web y más.

- **Transformación de datos**: Logstash permite transformar los datos antes de enviarlos a Elasticsearch o a otros destinos, lo que puede incluir filtrar, enriquecer y modificar los datos.

- **Conexión con diferentes sistemas**: Logstash se conecta a diferentes sistemas y servicios, como bases de datos SQL, MongoDB, Kafka y más.

● **Escalabilidad**: Logstash es altamente escalable y puede manejar grandes cantidades de datos y procesamiento de datos.

3.1.5 Kibana

Kibana es una interfaz de usuario web para Elasticsearch que permite visualizar y analizar los datos indexados. Algunas de sus características y funcionalidades incluyen:

● **Visualización de datos**: Kibana permite crear visualizaciones de datos en tiempo real, incluyendo gráficos, tablas, mapas y más.

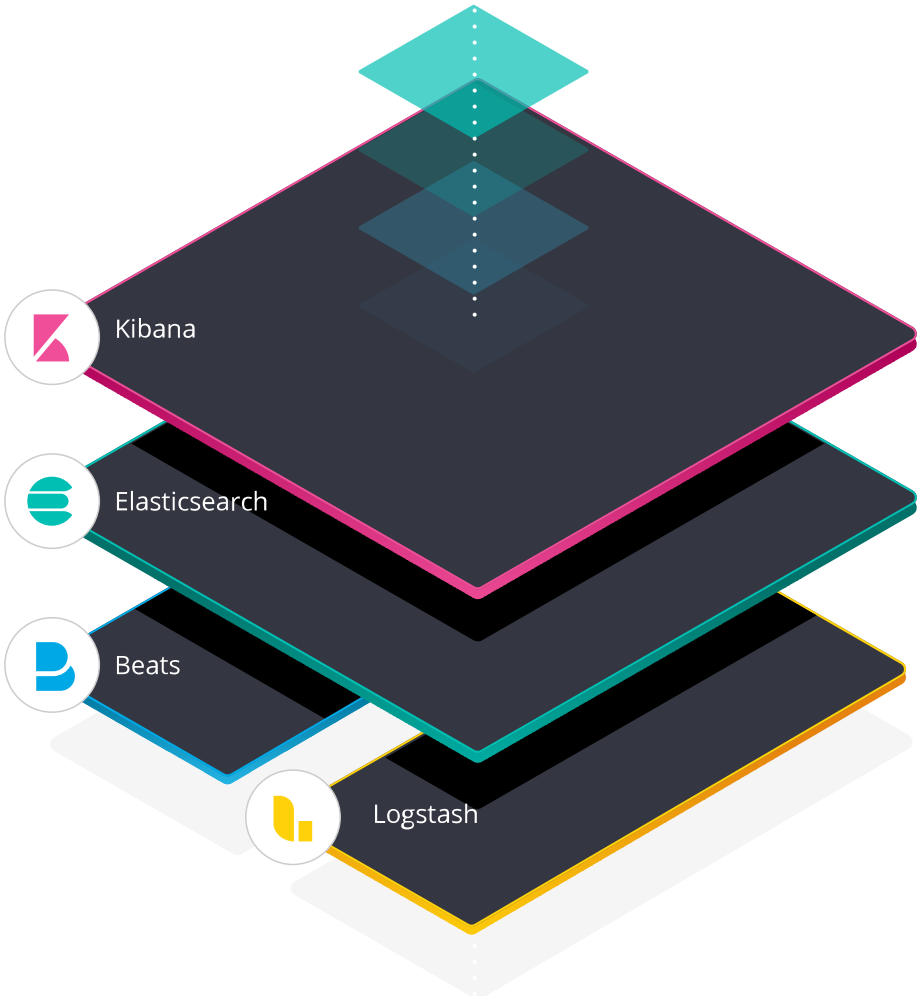
● **Análisis de datos**: Kibana permite realizar análisis de datos en tiempo real, incluyendo agregaciones, estadísticas y más.

● **Búsqueda de texto completo**: Kibana permite realizar búsquedas de texto completo en los datos indexados.

● **Integración con Elasticsearch**: Kibana se integra con Elasticsearch para proporcionar una solución completa de búsqueda y análisis de datos.

3.1.6 Conceptos de importancia

Beats: Los beats de la pila ELK son agentes de envío de datos que recopilan y envían datos a Elasticsearch o a otros destinos. Estos agentes se encargan de la recopilación de datos, envío de datos, ofrece escalabilidad para el manejo de grandes cantidades de datos y nos permite la integración con Elasticsearch



Shards: Los shards son unidades de procesamiento y almacenamiento que se distribuyen por los nodos del clúster.

Replica de shards: Copia de shards de Elasticsearch, proporciona alta disponibilidad en caso de que un nodo falle.

3.2 Características y funcionalidades de Metricbeat

3.2.1 Los comienzos de Metricbeat

La idea de Metricbeat nació en respuesta a la necesidad de recopilar y analizar métricas del sistema y servicios en tiempo real, lo que es esencial para la administración y monitoreo de la infraestructura en entornos de producción. En lugar de confiar en herramientas de terceros para recopilar y enviar datos de métricas, Elastic decidió desarrollar una herramienta interna que pudiera integrarse perfectamente con Elasticsearch y otros componentes de la pila ELK.

En 2016, Elastic lanzó Metricbeat como parte de la versión 5.0 de la pila ELK. Desde entonces, Metricbeat se ha convertido en una herramienta esencial para la recopilación y análisis de métricas de sistemas y servicios en tiempo real en una amplia variedad de casos de uso. Metricbeat ha evolucionado continuamente con nuevas características y módulos para adaptarse a las cambiantes necesidades de los usuarios y de la infraestructura de sistemas y servicios.

3.2.2 ¿Qué es Metricbeat?

Metricbeat es una herramienta desarrollada por Elastic como parte de la pila ELK para recopilar y enviar datos de métricas del sistema y servicios a Elasticsearch o a otros destinos. A continuación, se presentan algunas de las principales características y funcionalidades de Metricbeat:

- **Recopilación de métricas**: Metricbeat recopila una amplia variedad de métricas de diferentes sistemas y servicios, como CPU, memoria, red, disco, bases de datos, servidores web y contenedores.

- **Modularidad**: Metricbeat está diseñado con un enfoque modular que permite agregar y configurar módulos específicos para recopilar métricas de sistemas y servicios específicos. Los módulos preconstruidos incluyen, por ejemplo, Apache, Docker, Elasticsearch, MySQL, MongoDB, Redis, entre otros.

- **Monitoreo en tiempo real**: Metricbeat envía los datos de métricas recopilados a Elasticsearch o a otros destinos en tiempo real, lo que permite el monitoreo y análisis en tiempo real de la infraestructura.

- **Bajo consumo de recursos**: Metricbeat es un agente ligero que consume muy pocos recursos del sistema, lo que lo hace ideal para su uso en sistemas y entornos de producción.
- **Integración con Elasticsearch**: Metricbeat se integra perfectamente con Elasticsearch, lo que permite un fácil análisis y visualización de los datos de métricas recopilados a través de Kibana.
- **Automatización**: Metricbeat se puede configurar y automatizar mediante herramientas de orquestación de infraestructura como Ansible, Puppet y Chef, lo que facilita su implementación y administración en entornos de producción.

3.3 Características y funcionalidades de Filebeat

3.3.1 Los comienzos de Filebeat

Filebeat es un componente del conjunto de herramientas de Elastic Stack, desarrollado por Elastic. Fue lanzado por primera vez en 2014 y se ha convertido en una herramienta popular para la recopilación y envío de logs y otros datos a distintos destinos.

Antes de Filebeat, la forma común de recopilar logs en Elastic Stack era a través de Logstash, que es una herramienta muy versátil pero más pesada y con mayor consumo de recursos. Filebeat fue diseñado como una alternativa más ligera y eficiente para la recopilación de logs.

3.3.2 ¿Qué es Filebeat?

Filebeat es un componente del conjunto de herramientas de Elastic Stack, desarrollado por Elastic. Se utiliza para la recopilación, el envío y el procesamiento ligero de logs y otros datos de diferentes fuentes. Algunas de las principales características y funcionalidades de Filebeat son las siguientes:

- **Recopilación de logs**: Filebeat puede leer y recopilar logs de archivos de texto, logs de sistema y eventos de diferentes fuentes, como servidores, aplicaciones, servicios y más. Puede seguir de forma continua los cambios en los archivos de logs o enviar logs de forma periódica.
- **Envío de logs a destinos**: Filebeat puede enviar logs y otros datos a distintos destinos, como Elasticsearch, Logstash, Kafka y otros sistemas de almacenamiento o procesamiento. Esto permite almacenar, analizar y visualizar los datos de logs de forma centralizada.
- **Módulos preconfigurados**: Filebeat proporciona una serie de módulos preconfigurados para la recopilación de logs de fuentes populares, como Apache, Nginx, MySQL, Docker, sistema operativo, entre otros. Estos módulos simplifican la configuración y permiten una integración rápida con diferentes fuentes de logs.

3.4 Requisitos del Sistema y compatibilidad

En cuanto a requisitos del sistema todo varía en función de la cantidad de datos a procesar pero sería algo similar a lo siguiente:

● **Elasticsearch**: Se recomienda al menos 8 GB de RAM y 16 GB de espacio de almacenamiento para el sistema de archivos en cada nodo del clúster.

● **Logstash**: Se recomienda al menos 4 GB de RAM y 2 CPU para el procesamiento de datos de tamaño mediano.

● **Kibana**: Se recomienda al menos 4 GB de RAM y 2 CPU para su uso en producción.

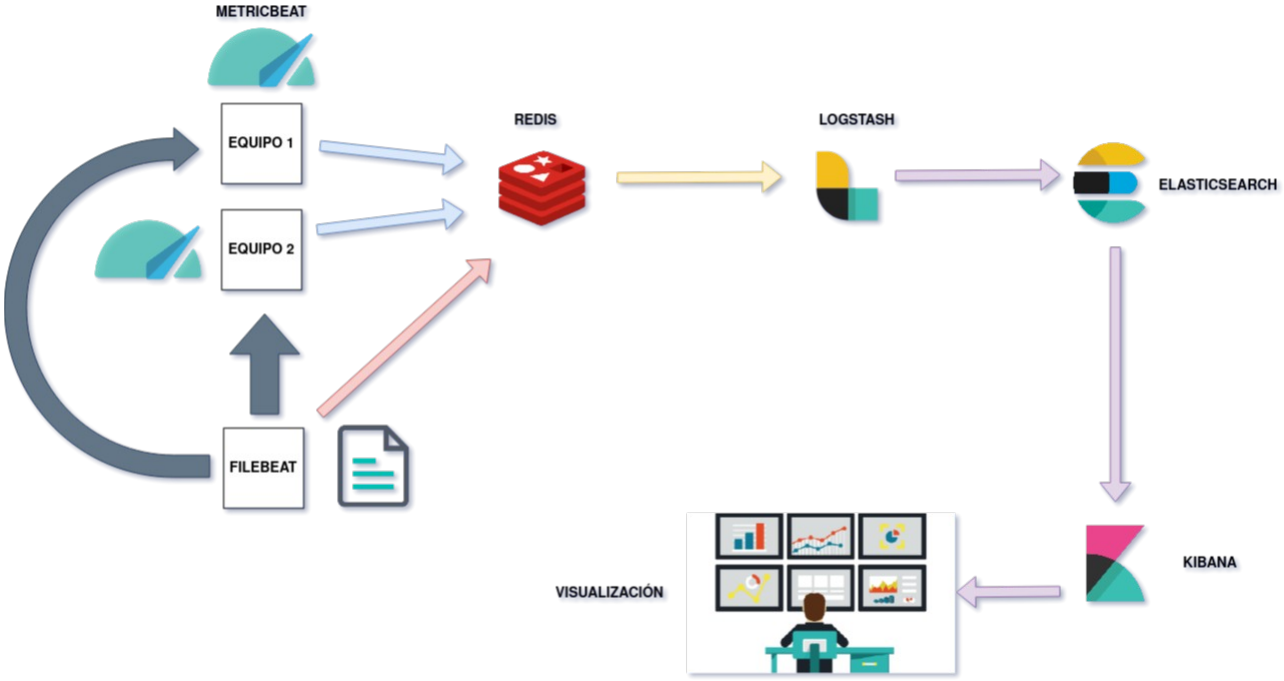
● **Metricbeat**: Se recomienda al menos 2 GB de RAM y 1 CPU para su uso en producción.

● **Filebeat**: Se recomienda al menos 2 GB de RAM para su uso en producción.

En cuanto a la compatibilidad, la pila ELK es compatible con una amplia variedad de sistemas operativos, incluyendo Linux, Windows y macOS. Además, Elasticsearch y Kibana son compatibles con una amplia variedad de navegadores web, incluyendo Chrome, Firefox, Safari y Edge.

En resumen, los requisitos del sistema y la compatibilidad de la pila ELK son bastante flexibles y se adaptan a una amplia variedad de casos de uso y entornos de producción. Se recomienda verificar los requisitos específicos para cada componente antes de implementar la pila ELK en producción.

4. Escenario sobre el que voy a trabajar



En cuanto al escenario sobre el que voy a trabajar, voy a hacerlo todo el local utilizando contenedores docker, instalando y configurando cada herramienta según lo que queramos llegar a obtener, mi escenario estará planteado de la siguiente manera:
Tendré 2 hosts de los cuales recopilaré logs y métricas, las métricas que recopilaré serán métricas de sistema, de apache, de nginx, de docker y de mysql.
Cada una de estas máquinas tendrán funcionando servicios web, base de datos y aplicaciones, que serán las fuentes de las cuales voy a recopilar los logs y las métricas.
Luego en cada una de estas máquinas instalaré Metricbeat para que recopile métricas y lo configuraré para que envíe los datos recopilados a Redis, que estará instalado en otro contenedor.

Tendré un contenedor de Filebeat que recopilará los logs de todos los contenedores docker y enviará los logs recopilados a Redis.

Todos los datos recopilados son enviados a Redis para que se almacenen siempre incluso cuando falle la pila ELK, si por ejemplo tenemos un problema con el usuario de monitorización que estamos utilizando y cambia la contraseña o el procesamiento de Logstash falla, todos estos datos se perderían el tiempo que la pila ELK esté inactiva, por lo tanto, he visto con buenos ojos implementar una base de datos de tipo clave-valor como lo es Redis para almacenar todos los datos

recopilados en este y que Logstash lea la cola de Redis para obtener los datos a procesar, de esta manera, incluso con la pila ELK caida, los datos de métricas y logs seguirían recopilandose y almacenándose para ser procesados en cuanto la pila ELK vuelva a estar activa.

Logstash procederá a leer la cola de Redis y procesará y enriquecerá estos datos obtenidos (agregando información adicional, como el nombre del host, la fecha y hora, etc) para posteriormente enviarlos a Elasticsearch, que estará instalado en otra máquina para almacenar estos datos.
Por último tendremos una última máquina con Kibana para ofrecer dashboards para la visualización de los datos almacenados en Elasticsearch.

Por lo tanto, tendremos un escenario con 1 contenedor por herramienta de la pila ELK (3) más otros 2 hosts con Metricbeat para la recopilación de métricas y por último un contenedor con Filebeat que recopilará los logs de todos los contenedores docker.

4.1 Demostraciones que voy a realizar

- Simulación de problemas para crear situaciones que produzcan problemas en la infraestructura y demostrar que este sistema es capaz de detectar y ofrecer información para resolver el problema de manera eficiente.

- Visualización de datos recopilados a lo largo del tiempo con Kibana y realizaré un análisis de tendencias para demostrar que este sistema es capaz de identificar patrones y tendencias en función del comportamiento de los sistemas y aplicaciones de mi infraestructura.

- Búsqueda de logs para demostrar que puedes encontrar rápidamente información específica en los registros.

5. Instalación y configuración de Elastic Stack, Metricbeat y Filebeat

5.1 Instalación y configuración de Elastic Stack

Este manual está pensado para realizar las instalaciones en un sistema operativo Debian.

Para las tres herramientas de la pila ELK vamos a necesitar el siguiente paquete y repositorio de elastic

- Instalamos el paquete apt-transport-https

```
apt install apt-transport-https
```

- Importamos la clave GPG del repositorio oficial de Elasticsearch

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | gpg --dearmor -o /usr/share/keyrings/elastic-keyring.gpg
```

- Almacenamos el repositorio en /etc/apt/sources.list.d/elastic-8.x-list

```
echo "deb [signed-by=/usr/share/keyrings/elastic-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | tee -a /etc/apt/sources.list.d/elastic-8.x.list
```

5.1.1 Instalación y configuración de Elasticsearch

- Actualizamos paquetería e instalamos el paquete elasticsearch

```
apt update && apt install elasticsearch
```

Puerto sobre el que trabaja: 9200

Fichero elasticsearch.yml

El fichero elasticsearch.yml es el fichero de configuración principal de Elasticsearch. En él, se especifican diferentes ajustes y opciones de configuración para el funcionamiento de un nodo de Elasticsearch.

El fichero elasticsearch.yml que voy a utilizar en este escenario es el siguiente:

```
cluster.name: "docker-cluster"
network.host: 0.0.0.0
discovery.type: single-node
```

Donde:

- **cluster.name**: Esta opción se utiliza para asignar un nombre a tu clúster de Elasticsearch. Cada clúster debe tener un nombre único para evitar conflictos con otros clústeres en la red.

- **network.host**: Esta opción especifica la dirección IP en la que Elasticsearch escucha y acepta conexiones entrantes. Al establecerlo en 0.0.0.0, Elasticsearch escuchará en todas las interfaces de red disponibles. Esto significa que Elasticsearch estará accesible tanto desde la dirección IP interna de la máquina como desde direcciones IP externas si el servidor está expuesto a Internet.

- **discovery.type**: Indica que Elasticsearch se está ejecutando en modo de descubrimiento de nodo único.

API Elasticsearch y Query DSL

La API de Elasticsearch es una interfaz de programación de aplicaciones que proporciona un conjunto de endpoints para interactuar con el clúster de Elasticsearch. Esta API permite realizar una variedad de operaciones, como indexar, buscar, actualizar y eliminar documentos, administrar índices y configuraciones, realizar agregaciones y mucho más. La API de Elasticsearch es accesible a través de solicitudes HTTP/RESTful y responde en formato JSON.

El Query DSL es un lenguaje específico de dominio (DSL) utilizado para construir consultas y búsquedas avanzadas en Elasticsearch. El Query DSL permite especificar criterios de búsqueda precisos, combinar múltiples condiciones, realizar consultas de texto completo, aplicar filtros, ordenar resultados y realizar agregaciones para obtener métricas y estadísticas sobre los datos. El Query DSL utiliza una sintaxis JSON que permite construir consultas complejas de manera legible y expresiva.

Tanto la API como el Query DSL nos ofrecen capacidades de búsqueda y análisis de datos en Elasticsearch.

Tras la instalación de Elasticsearch, podemos comprobar que está en funcionamiento realizando una solicitud GET al endpoint raíz de Elasticsearch (http://localhost:9200), esta solicitud nos devuelve información sobre el estado y la configuración del clúster de Elasticsearch. Una salida correcta sería la siguiente:

```
{
  "name" : "a40a767775b4",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "D4b15w7QQJWAZ9WrgkZOmA",
  "version" : {
    "number" : "8.0.0",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "1b6a7ece17463df5ff54a3e1302d825889aa1161",
    "build_date" : "2022-02-03T16:47:57.507843096Z",
    "build_snapshot" : false,
    "lucene_version" : "9.0.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
```

```
  "tagline" : "You Know, for Search"
}
```

Elasticsearch por defecto tiene usuarios creados para su administración, como el usuario elastic, pero nosotros como administradores vamos a crear nuestro propio usuario de monitorización, el cual se encargará, por ejemplo, de acceder a Kibana para visualizar los datos recopilados y otras muchas funciones.

Para ello primero generaremos contraseñas aleatorias para los usuarios administradores por defecto y crearemos el rol y el usuario con el usuario elastic y la contraseña generada aleatoriamente

Para generar la contraseña aleatoria de los usuarios ya definidos por Elasticsearch tenemos que utilizar la siguiente sentencia

```
bin/elasticsearch-setup-passwords auto
```

Obtendremos una salida como la siguiente, donde cada usuario creado automáticamente por Elasticsearch tendrá una contraseña aleatoria

```
Changed password for user apm_system
PASSWORD apm_system = WPHqyaCmubDoS6YmUWIQ

Changed password for user kibana_system
PASSWORD kibana_system = rcS5shRNLe3woiBYLaVL

Changed password for user kibana
PASSWORD kibana = rcS5shRNLe3woiBYLaVL

Changed password for user logstash_system
PASSWORD logstash_system = kSqMbRG0Dkx8mr0maBMp

Changed password for user beats_system
PASSWORD beats_system = vCnwyTzlQtOwmqQY1lfM

Changed password for user remote_monitoring_user
PASSWORD remote_monitoring_user = DIwmZKUudgjbb6IKB6b0

Changed password for user elastic
PASSWORD elastic = aPvUyEMp8WXbaS7NW9wH
```

A continuación utilizaremos la API de Elasticsearch para crear el usuario de monitorización que vamos a utilizar para la gestión de índices de Elastic.

Este usuario lo creamos para no trabajar por así decirlo con el super usuario “elastic” el cual tiene todos los privilegios posibles.

Siguiendo la documentación oficial de Elasticsearch para la creación de un usuario de monitorización tenemos que crear primero un rol con determinados privilegios sobre un índice específico y luego crear el usuario con ese rol creado y otros dos, llamados “kibana_admin” y “monitoring_user”

- El rol “kibana_admin” sirve para administrar y configurar Kibana, la interfaz de usuario de Elasticsearch. Al asignar el rol "kibana_admin" a un usuario, le estás otorgando privilegios para realizar tareas administrativas en Kibana, como crear y administrar paneles, visualizaciones, consultas y configuraciones de Kibana. Este rol es útil para los administradores de Kibana que necesitan un amplio acceso y control sobre la configuración y los datos en Kibana.

● El rol “monitoring_user” sirve para usuarios que necesitan acceder a datos de monitoreo en Elasticsearch. Al asignar el rol "monitoring_user" a un usuario, le estás otorgando privilegios para acceder a las métricas y registros de monitoreo del clúster de Elasticsearch. Los usuarios con este rol pueden visualizar y analizar datos de monitoreo, lo que les permite supervisar la salud y el rendimiento del clúster. Este rol es útil para los equipos de operaciones y los administradores que necesitan tener información detallada sobre el estado del clúster y realizar un seguimiento de su rendimiento.

Como he dicho, primero creamos el rol:

- Le indicamos el usuario elastic:contraseña autogenerada porque necesitamos un usuario con privilegios para crear roles.

- El nombre del rol se encuentra en la URL sobre la que hacemos la petición, “usuario_monitoring”

- Le damos privilegios sobre todos los índices

- Le asignamos los privilegios necesarios para la gestión de estos índices

```
curl -H 'Content-Type: application/json' -u
elastic:aPvUyEMp8WXbaS7NW9wH -XPUT
"http://localhost:9200/_security/role/usuario_monitoring" -d '{
  "indices": [
    {
      "names": ["*"],
      "privileges": ["create_index", "create_doc", "manage",
"auto_configure", "all"]
    }
  ],
  "cluster": ["monitor"],
  "applications": [],
```

```
  "run_as": [],
  "metadata": {}
}'
```

Procedemos a continuación a crear el usuario agregándole los roles mencionados y una contraseña a nuestro gusto, recomendable que esta contraseña sea segura, incluso si hace falta generada con pwgen, ya que este usuario va a ser capaz de actuar sobre todos los registros almacenados en Elasticsearch, puede ser peligroso que lo utilice alguien no autorizado.

Creamos el usuario

- Vemos el nombre del usuario, “angelsuarez”
- Contraseña generada con pwgen para mayor seguridad
- Roles asignados siguiendo la documentación oficial de Elasticsearch

```
curl -H 'Content-Type: application/json' -u
elastic:aPvUyEMp8WXbaS7NW9wH -XPOST
"http://localhost:9200/_security/user/angelsuarez" -d '{
  "password" : "Thieth5ta:u?faig",
  "roles" : ["kibana_admin", "monitoring_user",
"usuario_monitoring"]
}'
```

Para comprobar que el usuario se ha creado correctamente, tratamos de hacer una consulta al servidor de Elasticsearch para comprobar el estado del servidor con este usuario creado y vemos que es capaz de realizar correctamente la consulta.

```
elasticsearch@a40a767775b4:~$ curl -u angelsuarez:Thieth5ta:u?faig http://localhost:9200
{
  "name" : "a40a767775b4",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "D4b15w7QQJWAZ9WrgkZOmA",
  "version" : {
    "number" : "8.0.0",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "1b6a7ece17463df5ff54a3e1302d825889aa1161",
    "build_date" : "2022-02-03T16:47:57.507843096Z",
    "build_snapshot" : false,
    "lucene_version" : "9.0.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
elasticsearch@a40a767775b4:~$
```

De esta manera, podemos dar por concluida la correcta configuración de Elasticsearch con nuestro usuario de monitorización listo para gestionar índices y kibana.

5.1.2 Instalación y configuración de Logstash

Primero vamos a proceder a la instalación de Logstash, vamos paso a paso:

- Actualizamos paquetería e instalamos el paquete Logstash

```
apt update && apt install logstash
```

Puertos sobre los que trabaja:

- 5044: Puerto utilizado para recibir datos de logs mediante el protocolo de entrada Beats.

- 5000: Puerto utilizado para recibir datos de logs mediante el protocolo de entrada syslog.

- 9600: Puerto utilizado para acceder a la interfaz de línea de comandos (CLI) de Logstash para realizar consultas y obtener información de estado.

Fichero logstash.yml

Es el archivo de configuración principal de Logstash. Contiene configuraciones generales que se aplican a todo el entorno de Logstash. Algunas de las configuraciones que puedes encontrar en este archivo incluyen la configuración de red, la configuración de memoria, la configuración de logging y otras opciones de configuración a nivel global.

El fichero de configuración de logstash.yml que voy a utilizar en mi escenario tendrá la siguiente línea:

```
http.host: "0.0.0.0"
```

Esta línea se refiere a la configuración de la dirección IP en la que Logstash escuchará las solicitudes HTTP entrantes, significa que Logstash estará escuchando en todas las interfaces de red disponibles en el sistema.

Ficheros de definición de pipelines

Estos archivos contienen la configuración específica de Logstash, incluyendo las entradas (inputs), filtros (filters) y salidas (outputs) que se utilizan para procesar los datos. Son los ficheros almacenados en /etc/logstash/conf.d/ y se pueden tener varios ficheros dependiendo del filtro, inputs y outputs. Cuando ejecutas Logstash, puedes especificar que todos los pipelines definidos en /etc/logstash/conf.d/ se utilicen simultáneamente. Esto se logra utilizando la ruta /etc/logstash/conf.d/*.conf como argumento, lo que indica que todos los archivos .conf en ese directorio deben ser considerados como parte de la configuración.

El pipeline a utilizar en nuestro caso va a ser por ejemplo el siguiente:

```
input {
  redis {
    host => "redis-service"
    port => 6379
    password => "passentrada"
```

```
    ssl => true
    data_type => "list"
    key => "metricas-logs-escenario"
  }
}

filter {
  if [agent][type] == "metricbeat" {
    mutate {
      add_field => { "tipoRegistro" => "metrica" }
      remove_field => [ "[agent][ephemeral_id]", "[host][mac]",
"[ecs][version]", "@version", "[event][duration]", "[host]
[architecture]"]
    }
  } else if [agent][type] == "filebeat" {
    mutate {
      add_field => { "tipoRegistro" => "log" }
      remove_field => [ "[agent][ephemeral_id]", "[ecs][version]",
"@version", "[log][offset]"]
    }
  }
}

output {
  elasticsearch {
```

```
    hosts => ["http://elasticsearch:9200"]
    user => "angelsuarez"
    password => "Thieth5ta:u?faig"
    index => "datos-recopilados"
  }
}
```

Donde:

El pipeline se divide en tres secciones, **input**, **filter** y **output:**

● **Input:** Se utiliza el plugin de redis para obtener los datos de entrada leyendo la cola de una key de redis, se indica el puerto, una contraseña, que se utiliza protocolo ssl, el tipo de dato y la key donde vamos a buscar los datos recopilados que queremos procesar y enviar al output. En este caso, para poder utilizar el plugin input de redis he tenido que instalarlo con la siguiente sentencia (esto puede ser necesario para otros plugins)

```
/usr/share/logstash/bin/logstash-plugin install logstash-input-redis
```

● **Filter:** Procesamiento de datos en el que en función del tipo de agente le añado y elimino ciertas claves, en caso del que tipo de agente sea “metricbeat” añadiré una columna llamada “tipoRegistro” con el valor “métrica” y eliminaré las siguientes claves de este registro: “agent.ephemeral_id”, “host.mac”, “ecs.version”, “@version”, “event.duration”, “host.architecture”.

En caso de que el valor de que el tipo de agente sea “filebeat” añadiré la misma columna “tipoRegistro” pero con el valor “log” y eliminaré las siguientes claves de este registro: “agent.ephemeral_id”, “ecs.version”, “@version”, “log.offset”

● **Output:** Se utiliza el plugin de Elasticsearch como output para mandar los datos recopilados y procesados a un servidor de Elasticsearch, le indicamos el usuario y la contraseña a utilizar (nuestro

usuario de monitorización para autenticarnos en Elasticsearch) y el índice donde queremos que se almacenen los datos recopilados.

Para hacer una demostración de que el procesamiento de datos es correcto debemos realizar una consulta desde la API de Elasticsearch o directamente desde Kibana a este índice. En caso de no estar procesando correctamente los datos nos dirigiremos a los logs para saber que está pasando.

Si queremos realizar una consulta a un índice concreto desde la API de Elasticsearch, utilizaremos la siguiente sentencia:

```
curl -u angelsuarez:Thieth5ta:u?faig http://localhost:9200/hosts-metricbeat-datos/_search
```

Donde:

- Nos identificamos con nuestro usuario de monitorización
- Especificamos el índice del que queremos ver sus documentos registrados (hosts-metricbeat-datos)

5.1.3 Instalación y configuración de Kibana

Primero vamos a proceder a la instalacaión de Kibana, vamos paso a paso:

- Actualizamos paquetería e instalamos el paquete Kibana

```
apt update && apt install kibana
```

Puerto sobre el que trabaja: 5601

Fichero kibana.yml

Es el archivo de configuración principal de Kibana. Contiene diversas configuraciones que permiten personalizar el comportamiento y la apariencia de Kibana. Este archivo se encuentra en el directorio de instalación de Kibana y se utiliza para configurar opciones como la conexión con Elasticsearch, el puerto de escucha, la configuración de seguridad, las opciones de visualización y muchas más.

El fichero que voy a utilizar en mi escenario es el siguiente:

```
server.host: "0.0.0.0"
server.shutdownTimeout: "5s"
elasticsearch.hosts: ["http://elasticsearch:9200"]
elasticsearch.username: "kibana"
elasticsearch.password: "uJVoquLVRxw5nLZtxKXD"
monitoring.ui.container.elasticsearch.enabled: true
```

Donde:

- **server.host: "0.0.0.0":** Indica que el servidor Kibana escucha en todas las interfaces de red disponibles

- **server.shutdownTimeout: "5s":** Establece el tiempo máximo permitido para que Kibana cierre las conexiones y se detenga correctamente. En este caso, 5 segundos

- **elasticsearch.hosts: ["http://elasticsearch:9200"]:** Apunta al cluster de Elasticsearch al que nos queremos conectar

- **elasticsearch.username: "kibana":** Indica el usuario con el que el Kibana se autentica en Elasticsearch, en este caso no nos sirve nuestro usuario de monitorización, tenemos que indicar el usuario predefinido “Kibana”

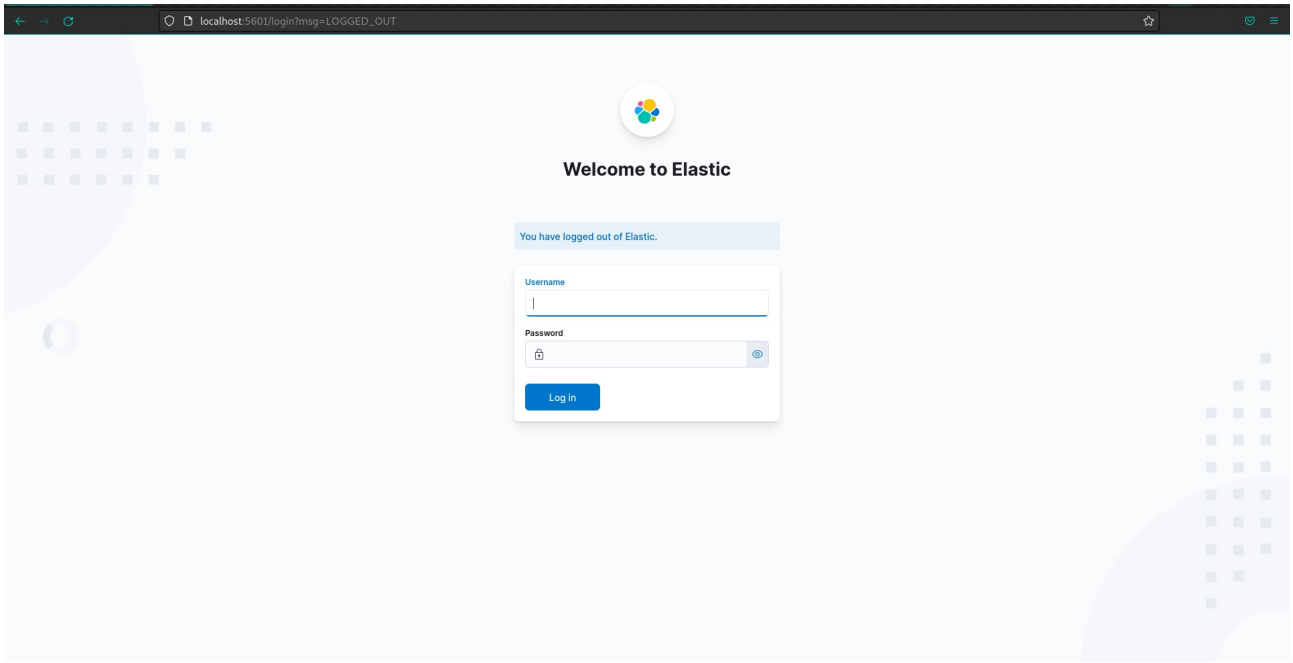
- **elasticsearch.password: "uJVoquLVRxw5nLZtxKXD":** Contraseña del usuario indicado (la generada previamente de forma automática)

- **monitoring.ui.container.elasticsearch.enabled: true:** Habilita o deshabilita la integración de monitoreo en Kibana con Elasticsearch. Cuando se establece en "true", Kibana muestra información de monitoreo relacionada con Elasticsearch en su interfaz de usuario.

Tras esto, lanzamos nuestro Kibana y accedemos a nuestro navegador a la URL http://localhost:5601.

Al acceder nos pedirá un usuario para acceder a Elastic, aquí entra en juego nuestro usuario de monitorización, capaz de gestionar cualquier índice de Elasticsearch y de configurar Kibana a su antojo (Dashboards, alertas…)

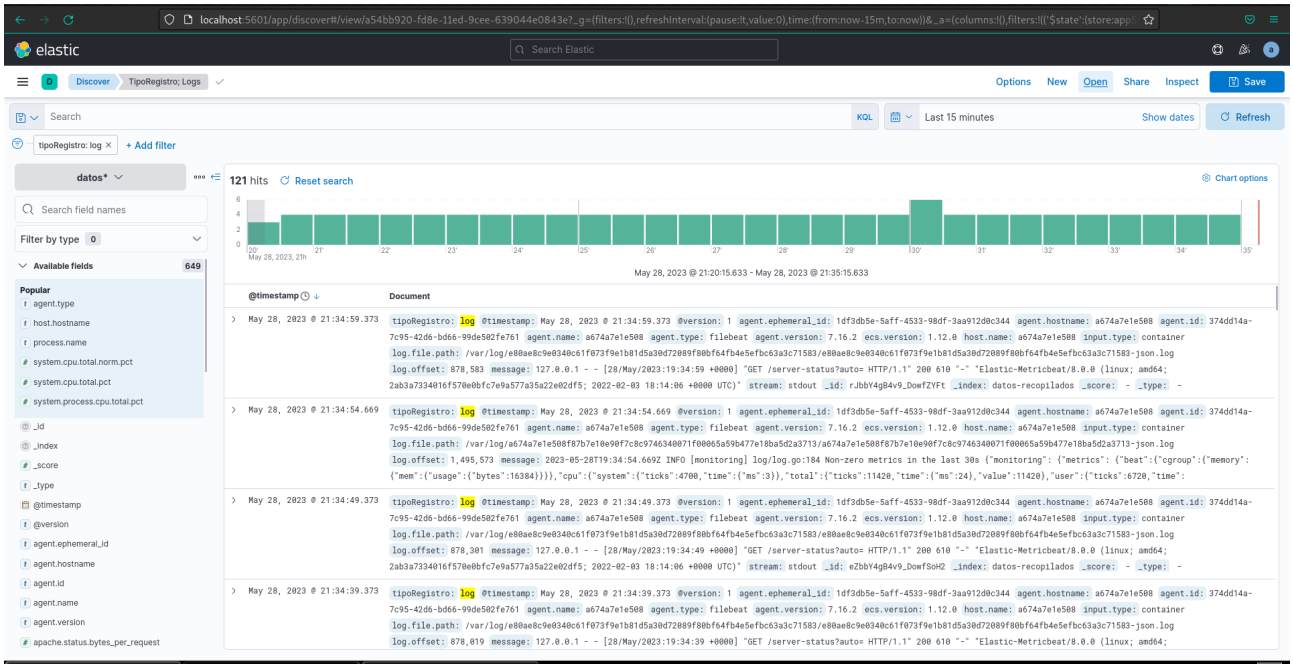
Una vez introduzcamos las credenciales, podemos proceder a probar nuestra interfaz web de Kibana (iniciando sesión con el usuario de monitorización creado en la API de Elasticsearch)



Discover

En el apartado de discover de Kibana podemos crear una nueva visualización de datos apuntando a nuestro índice de Elasticsearch y tener una visualización de nuestros datos recopilados, podemos especificar el rango de tiempo sobre el que queremos ver los datos recopilados y otras muchas opciones como aplicar filtros...

Filtro aplicado a clave “tipoRegistro”:”log” en los últimos 15 minutos



Dashboards

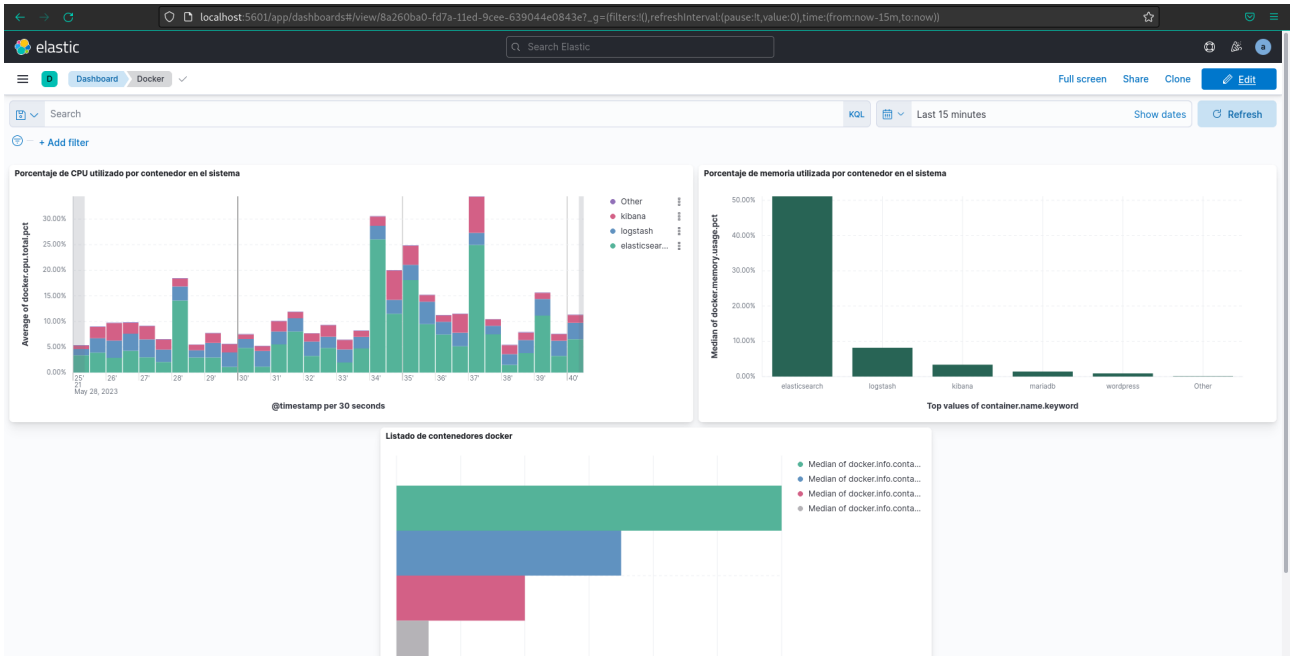
Los dashboards de Kibana son interfaces visuales que permiten mostrar y explorar datos de manera intuitiva y personalizada.

En estos dashboards podemos crear paneles, filtrar, explorar y analizar datos recopilados de manera más intuitiva.

Para crear un dashboard tenemos que tener clara la función de cada métrica recopilada para saber que datos nos ofrece y que panel podemos crear utilizando “x” métrica de los campos que hayamos exportado: https://www.elastic.co/guide/en/beats/metricbeat/current/exported-fields.html.

Para crear un dashboard vamos al apartado de Dashboards → Create dashboard → Create visualization. Una vez aquí elegimos el tipo de panel y las opciones que deseamos configurar.

Ejemplo de un dashboard con algunos paneles creado por mí.:



Este sería un ejemplo de como quedaría un dashboard con algunos paneles.

5.2 Instalación y configuración de Metricbeat

Para instalar Metricbeat tenemos que ejecutar las siguientes sentencias:

```
curl -L -O https://artifacts.elastic.co/downloads/beats/metricbeat/metricbeat-8.0.0-amd64.deb

sudo dpkg -i metricbeat-8.0.0-amd64.deb
```

De esta manera tendremos metricbeat instalado como servicio y podremos proceder a configurar el servicio.

Fichero metricbeat.yml

El fichero metricbeat.yml es el archivo de configuración principal de Metricbeat. Aquí es donde se definen los ajustes globales y se habilitan los módulos específicos que deseas utilizar.

Este fichero constará de dos partes, una en la que definamos los módulos a utilizar para saber que métricas queremos recopilar y otra en la que indicaremos donde queremos dirigir estas métricas.

El fichero utilizado en mi escenario será el siguiente

```
metricbeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false

setup.template.settings:
  index.number_of_shards: 1
```

```
  index.codec: best_compression


output.redis:
  hosts: ["redis-service:6379"]
  password: "passentrada"
  key: "metricas-logs-escenario"
  ssl:
    enabled: true
    verification_mode: none
```

Donde:

● **metricbeat.config.modules**: Esta sección configura la carga de módulos en Metricbeat. Especifica la ruta donde se encuentran los archivos de configuración de los módulos y deshabilita la recarga automática de módulos.

● **setup.template.settings**: Aquí se configuran las opciones relacionadas con las plantillas de índice de Elasticsearch. Se establece el número de fragmentos (shards) del índice en 1 y se utiliza la mejor compresión para el codec.

● **output.redis**: Aquí se configura la salida de Metricbeat hacia un servidor Redis. Se especifica la dirección del host y el puerto de Redis, la contraseña para la autenticación, y se define la clave para almacenar las métricas y los registros.

Módulos

Los módulos de Metricbeat son componentes preconfigurados que facilitan la recolección de métricas específicas de diversos servicios y tecnologías.

Cada módulo tiene su propio conjunto de métricas y configuraciones específicas.

Los módulos de metricbeat se almacenan en la ruta "/etc/metricbeat/modules.d", por defecto todos los módulos están deshabilitados, excepto el módulo de sistema, que por defecto recopila datos de cpu, carga, memoria, red…
Estos módulos simplemente están deshabilitados por su nombre, me explico:

Una vez instalamos metricbeat, el fichero metricbeat.yml establece los módulos a utilizar de la siguiente manera:

```
metricbeat.config.modules:
  path: ${path.config}/modules.d/*.yml
```

Lo que quiere decir que va a utilizar cualquier módulo de la ruta de módulos que termine en ".yml", y los módulos que están deshabilitados terminan en ".disabled"

Ejemplo entre el nombre por defecto de un módulo habilitado y uno deshabilitado:

```
# Habilitado
system.yml

# Deshabilitado
docker.yml.disabled
```

Es decir, si queremos habilitar un módulo tenemos que modificar el nombre (o la sentencia del fichero metricbeat.yml) y configurar el módulo, cuyo contenido, por ejemplo del módulo de docker, es el siguiente:

```
- module: docker
  metricsets:
    - container
    - cpu
    - event
```

```
    - info
    - memory
    - network
    - network_summary
  period: 10s
  hosts: ["unix:///var/run/docker.sock"]
```

Donde:

● **container**: Recopila métricas específicas de cada contenedor Docker en ejecución, como el uso de CPU, la memoria, el rendimiento de la red y las estadísticas de E/S.

● **cpu**: Recopila métricas de uso de CPU de los contenedores Docker, incluyendo el uso de CPU por contenedor y las estadísticas de uso de CPU en el host.

● **event**: Recopila eventos generados por Docker, como la creación o eliminación de contenedores, errores y otros eventos importantes.

● **info**: Recopila información general sobre el entorno Docker, como el número de contenedores en ejecución, imágenes disponibles, versiones de Docker, etc.

● **memory**: Recopila métricas de uso de memoria de los contenedores Docker, incluyendo el uso de memoria por contenedor y las estadísticas de uso de memoria en el host.

● **network**: Recopila métricas de uso de red de los contenedores Docker, incluyendo el tráfico de red entrante y saliente, el número de paquetes y los errores de red.

● **network_summary**: Recopila métricas de resumen relacionadas con la red de los contenedores Docker, incluyendo el número total de contenedores, el número de contenedores con red y el número de interfaces de red en el host.

Una vez visto un ejemplo y sabiendo como funcionan los módulos de metricbeat:
En mi escenario voy a utilizar los siguientes módulos:
- System
- Nginx

- Apache

- MySQL

- Docker

* El módulo docker está explicado anteriormente

Configuración del módulo system:

```
- module: system
  period: 10s
  metricsets:
    - cpu
    - load
    - memory
    - network
    - process
    - process_summary
    - socket_summary
```

Donde:

● **cpu**: Recopila métricas relacionadas con el uso de la CPU, como la carga promedio, el tiempo de CPU utilizado por cada núcleo y el tiempo de CPU por proceso.

● **load**: Recopila métricas relacionadas con la carga del sistema, como la carga promedio en el último minuto, en los últimos cinco minutos y en los últimos quince minutos.

● **memory**: Recopila métricas relacionadas con el uso de la memoria del sistema, como la memoria total, la memoria utilizada, la memoria libre, la memoria en búfer y la memoria en caché.

● **network**: Recopila métricas relacionadas con el uso de red, como el número de paquetes recibidos y enviados, los bytes recibidos y enviados, y las estadísticas de errores y descartes de la interfaz de red.

● **process**: Recopila métricas relacionadas con los procesos en ejecución, como el número de procesos en ejecución, el uso de CPU y memoria por proceso, y las estadísticas de E/S.

● **process_summary**: Recopila métricas de resumen relacionadas con los procesos, como el número total de procesos, el número de procesos por estado (durmiendo, en ejecución, detenidos, etc.) y el número de procesos por usuario.

● **socket_summary**: Recopila métricas de resumen relacionadas con los sockets, como el número total de sockets abiertos, el número de sockets por estado (establecido, escucha, cierre, etc.) y el número de sockets por tipo de protocolo.

Configuración del módulo nginx:

```
- module: nginx
  metricsets:
    - stubstatus
  period: 10s
  hosts: ["http://127.0.0.1"]
```

Donde:

● **metricsets**: Especifica el metricset o conjunto de métricas a utilizar. En este caso, se utiliza el metricset "stubstatus", que permite obtener información de las estadísticas de Nginx.

● **period**: Establece el período de recolección de métricas. En este caso, se recolectarán las métricas cada 10 segundos.

● **hosts**: Indica los hosts de Nginx de los cuales se recopilarán las métricas. En este caso, se utiliza "http://127.0.0.1" como el host local de Nginx.

Configuración del módulo apache:

```
- module: apache
  metricsets:
    - status
  period: 10s
  hosts: ["http://127.0.0.1"]
```

Donde:

● **stubstatus**: Especifica el metricset o conjunto de métricas a utilizar. En este caso, se utiliza el metricset "status", que permite obtener información del estado de Apache.

● **period**: Establece el período de recolección de métricas. En este caso, se recolectarán las métricas cada 10 segundos.

● **hosts**: Indica los hosts de Apache de los cuales se recopilarán las métricas. En este caso, se utiliza "http://127.0.0.1" como el host local de Apache.

Configuración del módulo mysql:

```
- module: mysql
  metricsets:
    - status
    - galera_status
    - performance
  period: 10s
```

Donde:

● **status**: Recopila métricas relacionadas con el estado general del servidor MySQL, como el tiempo de actividad, la cantidad de conexiones y la cantidad de consultas ejecutadas.

● **galera_status**: Este metricset se utiliza específicamente en entornos de clústeres Galera de MySQL. Recopila métricas relacionadas con el estado de la replicación Galera, como el estado de los nodos del clúster, el estado de la replicación, las transacciones pendientes, el tiempo de latencia de replicación, entre otros.

● **performance**: Recopila métricas relacionadas con el rendimiento del servidor MySQL, como el tiempo de respuesta de las consultas, las transacciones por segundo y los comandos de almacenamiento en caché.

5.3 Instalación y configuración de Filebeat

Para realizar la instalación de Filebeat, tenemos que ejecutar las siguientes sentencias

```
curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.16.0-amd64.deb


dpkg -i filebeat-7.16.0-amd64.deb
```

De esta manera tendremos filebeat instalado como servicio y podremos proceder a configurar el servicio.

Fichero filebeat.yml

El fichero filebeat.yml es el archivo de configuración principal de Filebeat, en este archivo definimos como Filebeat debe recolectar, procesar y enviar los archivos de logs a diferentes destinos.
En mi escenario, el fichero de configuración a utilizar va a ser el siguiente:

```
filebeat.inputs:
  - type: container
    paths:
      - /var/log/*/*.log


output.redis:
  hosts: ["redis:6379"]
  password: "passentrada"
  key: "metricas-logs-escenario"
  ssl:
```

```
    supported_protocols: [ "TLSv1.0", "TLSv1.1", "TLSv1.2",
"SSLv3" ]
    enabled: true
    verification_mode: none
```

Donde:

Dividimos el fichero en dos apartados, input y output.

El fichero está configurado de manera que el contenedor de Filebeat va a leer ficheros con extensión ".log" almacenados en una ruta donde he montado el directorio de logs de los contenedores docker.

Luego utilizaremos el output para configurar el destino de salida de los registros recopilados a un servidor Redis, configurado en el puerto 6379, con una contraseña configurada para el acceso (**password**), a una clave que almacenará los datos (**key**) y configurado con SSL, donde no hacemos comprobación alguna del certificado utilizado porque es un certificado autofirmado.

De esta manera, los logs de los contenedores docker estarían siendo recopilados y enviados a un servidor Redis mediante el uso de Filebeat.

6. Bibliografía

Documentación oficial pila ELK, Metricbeat y Filebeat

https://www.elastic.co/guide/en/logstash/current/configuration.html

https://www.elastic.co/guide/en/logstash/current/configuration-file-structure.html

https://www.elastic.co/guide/en/beats/auditbeat/current/privileges-to-publish-monitoring.html

https://www.elastic.co/guide/en/beats/metricbeat/current/exported-fields.html

https://www.elastic.co/guide/en/beats/filebeat/current/redis-output.html

https://www.elastic.co/guide/en/kibana/current/settings.html

Documentación redis

https://redis.io/docs/management/

https://redis.io/docs/management/security/encryption/

Otros

Además, he aprovechado recursos internos y documentación confidencial de la empresa durante mis prácticas, así como he participado en cursos y talleres especializados sobre la pila ELK y sus componentes, como los beats, ofrecidos por instituciones reconocidas como OpenWebinars y otros proveedores de formación.

7. Dificultades encontradas

En cuanto al escenario y todas las herramientas con las que he trabajado, las mayores dificultades que he encontrado han sido comprender correctamente la API de Elasticsearch y la creación de dashboards en Kibana, que como tal no es algo complicado pero tienes que comprender que hace cada clave extraída con Metricbeat para saber como combinarla y llegar a mostrar dashboards con sentido y fáciles de entender, recomiendo mucho para este apartado de dashboards mirar la documentación oficial (se encuentra en la bibliografía) acerca de que hace cada campo extraido.

También los problemas de compatibilidad con la pila ELK me han resultado algo incómodo, ya que tienes que encontrar las versiones sobre las que puedes combinar las tres herramientas y que además cumplan con las configuraciones que quieres implementar en tu escenario, ya que, hay opciones que en las últimas versiones quedan obsoletas y se sustituyen para ser configuradas de otra manera.

En cuanto a mi escenario también se puede hacer un poco tedioso configurar todas las herramientas para que cada una cumpla su función y los datos lleguen a su destino, como por ejemplo que Redis almacene correctamente los datos que le hacen llegar en la clave donde quieres que se almacene y que Logstash lea la cola de Redis y procese correctamente esos datos y los haga llegar con el formato deseado a Elasticsearch. Entender el procesamiento de Logstash es más complicado de lo que parece.

8. Conclusiones y propuestas para seguir trabajando sobre este tema

Las conclusiones que saco de este PI es que hay muchas fuentes que son capaces de recopilar métricas y logs y hay otras muchísimas herramientas que se pueden combinar para llegar a formar un escenario bastante interesante en el que se puede hacer un manejo de datos mucho más cómodo. También que todos los datos recopilados se pueden transformar y enriquecer en función de lo que queramos llegar a mostrar.

La visualización final de los datos debe ser intuitiva para detectar fácilmente patrones, tendencias y anomalías.

Considero que este tema tiene mucho más sobre lo que aprender y como he dicho en un apartado, hay empresas que buscan especialistas en la pila ELK, que considero que es lo más importante de mi PI. Yo tengo un escenario por así decirlo de prueba, en el que vemos el funcionamiento de las herramientas y algunas de las muchísimas cosas que podemos hacer, considero que se le puede sacar mucho juego a este tema y por supuesto tengo pensado aprender más. Si te interesa todo el tema de manejo de grandes cantidades de datos y su procesamiento, vas a querer aprender mucho acerca de esta pila y todos los beats y herramientas que puedes combinar en un escenario.

Continuar trabajando sobre este tema implicaría explorar y aprender acerca de más módulos y métricas, integración con otras herramientas, seguridad y autenticación y una configuración más avanzada de todas las herramientas, aprendiendo ajustes específicos para optimizar el rendimiento, mejorar la seguridad…

De esta manera, llegamos al final de mi proyecto sobre la monitorización de logs y métricas utilizando Filebeat, Metricbeat y la pila ELK. Espero que sea tan útil como me ha sido a mí para aprender sobre estos temas.

FIN