UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Gestión centralizada de Logs. Herramientas de código abierto.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Michis , Stefan Vasile

Tutor/a: Acebrón Linuesa, Floreal

CURSO ACADÉMICO: 2021/2022

Dedicatoria

A mi familia por permitirme estudiar sin poder trabajar, que es un privilegio, aunque no deba serlo nunca.

A Mario, Laura y Marcel por ayudarme cuando los necesitaba y por ayudarme a creer en mi para poder seguir con la carrera.

A Vicente, Víctor y Quique por ser las mejores personas que me he encontrado durante el grado haciendo que el paso durante este sea agradable, ameno y muy satisfactorio.

Finalmente, y no por ello menos importante a Floreal por ayudarme con el desarrollo y con mi motivación personal para este último capítulo del grado.

Resumen

Cualquier máquina o aplicación genera registros de todo tipo, indicando todo tipo de información que puede o debe ser gestionada y controlada, hoy en día en las empresas se hace uso generalmente de la monitorización de parámetros de hardware e ignorando por completo los registros que se puedan generar por parte de su software o sus computadores. La práctica de la observabilidad permite hacer uso de dichos registros para la gestión de los servidores de manera que se puedan ver y analizar desde un punto central, cambiando el paradigma de uso.

Hacer una gestión óptima de los computadores es necesario para una rápida solución de problemas en sistemas que pueden llegar a ser críticos.

Con las herramientas que proporciona la empresa Elastic se puede hacer una práctica de la gestión centralizada de calidad en cualquier entorno que soporte la ejecución de su software. Es también necesario tener en cuenta que herramientas tienen los entornos en la nube como Google Cloud pues cada vez se usan más esos entornos en escalas cada vez más grandes, por lo que la práctica de la observabilidad se hace prácticamente obligatoria.

Palabras clave: Elastic, Métricas, Máquinas Virtuales, Google Cloud, Observabilidad

Abstract

Any machine or application generates logs of all kinds, indicating all types of information that can or should be managed and controlled, nowadays companies generally make use of monitoring hardware parameters and completely ignoring the logs that can be generated by their software or computers. The practice of observability makes it possible to make use of these logs for the management of servers so that they can be viewed and analyzed from a central point, changing the paradigm of use.

Optimal computer management is necessary for quick troubleshooting of potentially critical systems.

With the tools provided by Elastic it can be possible to make a practice of quality centralized management in any environment that supports the execution of that software. It is also necessary to consider the tools that cloud environments such as Google Cloud have, as these environments are increasingly used on larger and larger scales, making the practice of observability practically mandatory.

Keywords: Elastic, Logs, Virtual Machines, Google Cloud, Observability.

Tabla de Contenidos

1. Introducción

Hoy en día existe una amplia cantidad de aplicaciones que se usan o que se deben controlar en cualquier entorno mínimamente realista, con una amplia cantidad de elementos críticos que deben funcionar correctamente de manera constante. Estos elementos pueden ser servidores web, bases de datos, aplicaciones en red, software de terceros o incluso la propia máquina que esté ejecutando dichas aplicaciones. Proceder a hacer un control de estos elementos o tratar de encontrar el detonante de un problema que haya ocurrido sin ningún tipo de herramienta que facilite dichas tareas resulta en una labor muy tediosa pues además de haber muchas aplicaciones y/o nodos, cada uno de ellos genera una gran cantidad de información indicando su estado. Querer consultar manualmente estos datos podría conllevar más tiempo del deseado o del que se tenga disponible para el análisis de un problema que esté generando dicha aplicación. Para más inri, para cada consulta suele ser necesario acceder física o remotamente al nodo dificultando o retrasando la tarea en algunos casos. Es muy importante tener en cuenta el tiempo de acción en la detección de problemas sobre todo si se están dando en sistemas críticos. Es por ello por lo que se considera necesario proponer una herramienta de gestión que sea centralizada, rápida y fiable.

Cada vez más empresas virtualizan su información en servidores tanto propios como contratándolos como servicio en un tercero y una parte crucial de este movimiento es la gestión de dichas máquinas que pueden almacenar datos o proporcionar varios servicios específicos. Es necesario destacar que cada empresa tendrá unos objetivos y una metodología de funcionamiento distinta a las demás dando lugar a que los datos que tenga que almacenar en un servidor o las aplicaciones que tengan activas sean totalmente diferentes una de otra, se pretende hacer énfasis en que la gestión de las máquinas de cada empresa será prácticamente independiente entre proyectos o empresas distintas.

Ante dicha problemática, se propone, con este proyecto de fin de grado, mostrar una solución utilizando varios tipos de software en entornos tanto locales como en la nube que permitan tener un tiempo de respuesta aceptable ante la mayoría de los problemas que se puedan dar en entornos de este tipo, los cuales, pueden ser totalmente independientes o diferentes entre sí, ofreciendo mayor fiabilidad, rapidez y comodidad de uso ante la alternativa de no usar la propuesta escogida.

2. Objetivos

El principal objetivo de este proyecto es demostrar que, utilizando unas herramientas que permitan una gestión centralizada, es posible ahorrar mucho tiempo, gestionar el entorno y entender la raíz de un problema con mucha más sencillez que no usando esta metodología y software.

El primer paso para cumplir dicho objetivo es explicar por qué una gestión centralizada de logs es útil. Será imprescindible pues definir el concepto de observabilidad y los 3 pilares de esta: métricas, logs (o registros) y tracing (o trazabilidad). Se afianzará la versatilidad de practicar la observabilidad haciendo una comparación con el clásico concepto de monitorización. Puesto que este proyecto trata la gestión de los logs, se hará especial énfasis en dicho concepto justificando su importancia y su potencia si se da un uso correcto de dicha información.

Teniendo definidas las bases teóricas del proyecto se procederá a definir las herramientas open source que se van a usar primeramente para hacer práctica de la observabilidad y por ende de la gestión centralizada y se construirá un clúster local justo a esas de manera que se dé un uso práctico de dichas herramientas. Estas son Elasticsearch, Logstash y Kibana. Se definirá qué objetivo tiene cada herramienta en concreto para, una vez en conjunto, crear casos de uso en nuestro entorno local que simulen situaciones realistas donde se muestre la potencia que tiene utilizar este software en conjunto. Ciertamente estos programas no son la única solución al problema que se plantea de manera que se harán comparativas, una de ellas será utilizar una herramienta equivalente a Logstash: Fluentd y analizar cuál es la mejor de las 2 en nuestro contexto. Posteriormente se creará un clúster en la nube de Google Cloud donde se simularán los mismos casos de uso con las herramientas nativas posteriormente haciendo una comparación e indicando si hacer el gasto de utilizar las herramientas nativas de Google es más beneficioso que hacer uso del software que es gratuito.

3. Observabilidad

Hoy en día muchas empresas practican la monitorización de sus sistemas o de sistemas de sus clientes, es decir, se va comprobando un valor numérico que indica el estado de un elemento a lo largo del tiempo con el objetivo de tener la capacidad de detectar un problema en dicho elemento. Cierto es que, si se da, por ejemplo, un alto consumo de procesador con esta metodología es posible detectar la irregularidad, pero ¿a qué se debe tal incremento de consumo? ¿Este consumo va a seguir durante mucho tiempo? ¿Es realmente una irregularidad o se hubiese podido predecir el consumo en ese momento en concreto? Con meramente la monitorización no es posible responder tal pregunta si se desconoce el contexto del entorno de manera que se debe acceder a la máquina en cuestión para proceder a hacer un análisis. He aquí donde se introduce la observabilidad.

La observabilidad tiene el mismo objetivo que la monitorización, habilitar la capacidad de resolver los problemas que surjan en los sistemas a controlar, no obstante, el enfoque es diferente, no se busca hacer un análisis de las métricas únicamente, se busca hacer un análisis más completo utilizando la máxima información posible para poder hacer una búsqueda a grano fino de cuál es el problema. Para esta búsqueda se hará uso, por ejemplo, de información como los logs siendo esto mucho más útil y descriptivo en gran parte de los casos de uso puesto que, un alto consumo de CPU vendrá dado probablemente por parte de una aplicación que tendrá un uso irregular y dicho uso irregular puede estar justificado por los logs o registros que tenga dicha aplicación. Y todo esto desde un sistema central que controlará, en este caso, todos los logs que interesen. Otro caso de uso interesante es el momento en el que una aplicación falla o en el proceso de inicio de una aplicación ocurre un error, sin una herramienta que permita echar un vistazo a información sumamente relevante para esta casuística como los logs dificulta el análisis y solución del problema. Cabe recordar que, utilizar solamente métricas es una mala práctica, pero eso no quiere decir que no se deba utilizar o que no haya más maneras de aprovecharlas aparte de crear gráficas valor/tiempo [1] esto último se verá cuando se haga mayor énfasis a una de las herramientas que usaremos, Kibana.

Se debe tener en cuenta también que solo con tener la información disponible no es suficiente, esa información debe ser procesada, almacenada, y mostrada o graficada de manera que la información sea fácil de ver y fácil de buscar, para ello, se usan varias herramientas que tienen dicho cometido, no obstante, aunque estas herramientas facilitan mucho esta tarea, no practicarán la observabilidad de por sí para el administrador, será él mismo quien conociendo el contexto, el entorno de trabajo y teniendo el conocimiento y la experiencia pertinentes quien practique la observabilidad correctamente. [1]

A la hora de practicar la observabilidad se debe tener en cuenta 3 tipos de información con los que trabajar:

- Logs: Es este elemento de la observabilidad el que va a tener la mayor participación en este proyecto. Un log o registro es un texto junto a una fecha que indica un estado, una acción o un paso de un proceso de una aplicación, por ejemplo, para saber con certeza si es una aplicación como un servidor Apache se ha iniciado con facilidad, se buscan los logs de la susodicha aplicación y se comprueba si en realidad se ha iniciado correctamente. Es la manera más precisa de saber cómo está funcionando cualquier aplicación o sistema pues la información que se genera suele ser bastante descriptiva, de hecho, lo natural para cualquier persona que tenga algo de mano con servidores ante el fallo de una aplicación será "pelearse" con los logs. Cierto es que se genera mucha información muy descriptiva, pero hacer un filtrado de esta información de manera manual es muy tedioso cuanto menos, es por ello que, como se menciona al principio de este capítulo, que existen aplicaciones centradas únicamente en la observabilidad para poder filtrar y analizar dicha información. Tener un acceso a los logs necesarios de todas las aplicaciones de todas las máquinas en un mismo punto, sin tener que cambiar de máquina o de aplicación para poder observarlo todo además de tener la información preprocesada y filtrada trae muchas ventajas recayendo en la facilidad de uso una vez configurado el sistema y la mayor velocidad de reacción al detectar con mayor rapidez la causa y el lugar de un problema.

- Métricas: Las métricas describen el estado de los componentes de un nodo, es decir, se muestra su "estado de salud". Un nodo servidor, además de recibir los registros que se indiquen, puede recibir métricas de los clientes tales como el nivel de CPU, el nivel de RAM, el ancho de banda usado, la ocupación de disco, los programas que se estén ejecutando, etcétera. Esta información es muy útil para tenerla como complementación a los logs que se reciban, por ejemplo, se ve que en un cliente hay un alto consumo de CPU, se busca que aplicación está consumiendo ese extra de CPU y cuando se encuentra se procede a buscar logs de esa aplicación para intentar descubrir qué está pasando. Se debe hacer énfasis que tener métricas de monitorización es útil pero su objetivo es complementar a los logs pues se debe deshacer la idea de que las métricas son el elemento principal para controlar. Como se indicó antes, en la observabilidad no entran solamente métricas referentes al hardware de las máquinas, es interesante tener la posibilidad de crear métricas en base a logs para controlar el software necesario también.

- Tracing: Este apartado de la observabilidad no se tratará en este proyecto, pero cabe mencionarla pues en entornos como Kubernetes es crucial tener el tracing en cuenta. Se explicará qué es el tracing con un ejemplo: se hace una petición http a un servidor para que devuelva una web en concreto, pero resulta que antes de

llegar al servidor que procese dicha petición, hay un proxy por delante y dicho proxy no tiene la web por lo que redirige la petición al servidor resolviendola y respondiendo con dicha web. Tracing nos informa de por cuáles máquinas ha pasado una petición de cualquier tipo, en un sistema con muchos contenedores para un servicio, por ejemplo, saber por cuáles contenedores está pasando una petición es información muy interesante y útil, pero como este trabajo se centra en los propios logs y no se usa un entorno con contenedores no se cree necesario integrarlo en la gestión.

Este proyecto busca hacer un uso parcial de la observabilidad, es decir, hacer un gran énfasis en los logs y complementarlos con métricas a la par de ignorar la trazabilidad. Según las necesidades dadas, la interpretación de la observabilidad varía. Una gestión centralizada de logs permite tener un enfoque a la gestión de los propios servidores con logs que genera el propio sistema operativo o tener un enfoque a aplicaciones que use, bien sean nativas o de terceros. Se verá posteriormente que la gran potencia de este esquema de uso es su flexibilidad, se podrá adaptar el sistema para que procese la información que nos interese y aumentarla o reducirla según las necesidades. Las soluciones que hacen uso de la observabilidad se hacen indispensables en sistemas muy escalados, de 50 o 100 servidores, no es lo mismo tener 3 máquinas y tener un fallo desconociendo el nodo origen del problema que tener la misma situación con 75 computadoras.

4. Pila ELK

Haciendo referencia al segundo objetivo del proyecto: "*Teniendo definidas las bases teóricas del proyecto se procederá a definir las herramientas (…) que se van a usar (…) y se construirá un clúster local justo a esas de manera que se dé un uso práctico de dichas herramientas*" se pretende definir en la introducción a este capítulo un primer ecosistema que hará uso de las principales herramientas de este proyecto en lo que respecta la práctica de observabilidad y, por ende, de gestión centralizada de logs. Se tendrá un servidor con una pila de aplicaciones llamadas en conjunto ELK (por ende, su nombre es ELK Server). Esta pila, formada por Elasticsearch, Logstash y Kibana, recibirá la información, la procesará, la almacenará y la mostrará al administrador. Posteriormente, habrá 2 máquinas que tendrán como función principal enviar información a la pila formando el ecosistema, estas 2 máquinas, cuyo nombre será Cliente 1 y Cliente 2 respectivamente, tendrán 2 programas de misma naturaleza, pero diferente función con dicho objetivo: Filebeat y Metricbeat, que enviarán pulsos (o beats) cada cierto tiempo al Servidor ELK.

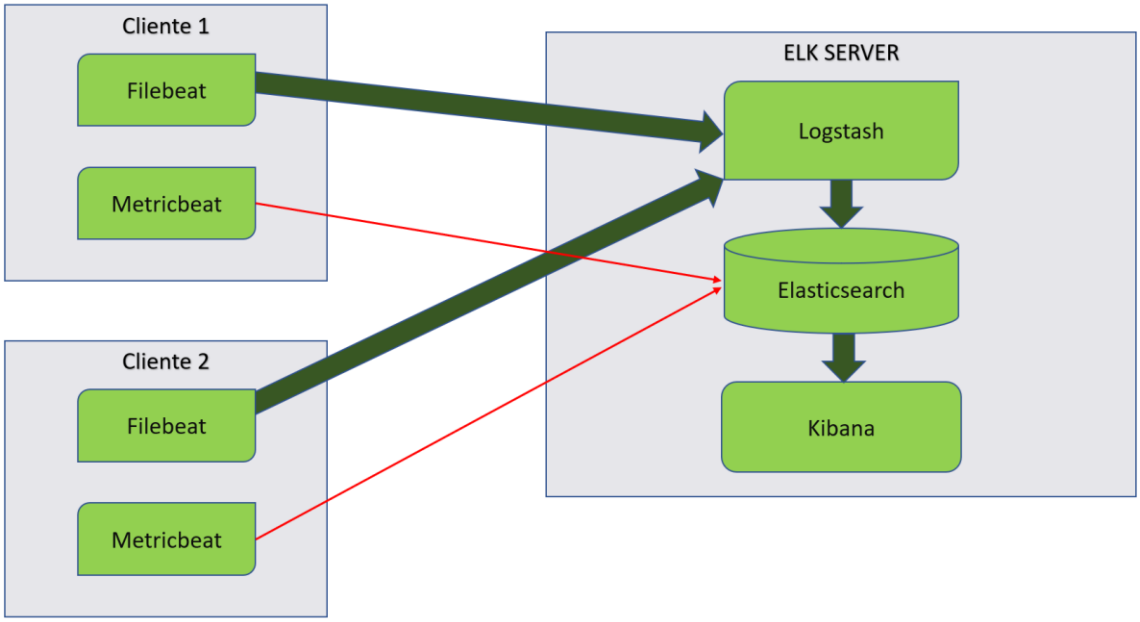


Figura 1: Esquema de la Pila ELK

En la Figura 1 se puede ver la funcionalidad básica de la pila, Filebeat enviará la información de los logs a Logstash, el cual procesará los datos y los enviará a Elasticsearch, el cual almacenará los datos de manera que Kibana se aproveche de estos datos con la interfaz gráfica que proporciona. Finalmente, Metricbeat enviará los datos directamente a Elasticsearch sin pasar por Logstash, este hecho se explicará posteriormente en el capítulo dedicado a los beats.

4.1. Elasticsearch

Elasticsearch es una base de datos temporal, es decir, almacena los datos junto a un "timestamp" indicando el momento de llegada o generación de dichos datos que cuenta con motor de búsqueda y de análisis distribuido de gran potencia construido sobre Apache Lucene permitiendo las rápidas búsquedas sobre cualquier tipo de información sea estructurada, no estructurada, numérica, geoespacial, etc. Elasticsearch es el corazón de la pila puesto que al fin y al cabo es la herramienta que almacena la información que interesa y si se tiene en cuenta, Logstash envía la información a Elasticsearch y Kibana explota la información y las analíticas que tiene disponibles por parte de esta base de datos temporal [2].

Además de esta funcionalidad, Elasticsearch proporciona otras funcionalidades como monitorización sobre sí mismo, integración con SQL, herramientas de seguridad, herramientas que tratan el tema de Alerting y funcionalidad con machine learning. Se hará un énfasis en el alacenamiento y análisis pues son estos factores los que son más relevantes en lo que la observabilidad refiere.

4.1.1. ¿Cómo Funciona Elasticsearch?

Cualquier elemento que reciba Elasticsearch será serializado a JSON (excepto en el caso donde el elemento recibido ya esté formateado a JSON), este elemento será almacenado con un timestamp el cual puede añadirlo Elasticsearch o puede formar parte de los datos que se reciben. Estos datos serán principalmente almacenados en índices, por ejemplo, los logs que se reciban se almacenarán en un índice mientras que las métricas se almacenarán en otros. Esta funcionalidad permite un mejor tiempo de búsqueda para permitir una mejor práctica de la observabilidad.

En Elasticsearch un índice no funciona como en una base de datos relacional, un índice en este caso es la propia información almacenada de manera óptima, según los datos que reciba Elasticsearch, los índices que se generen tendrán un formato u otro, en el caso de recibir momentáneamente 2 tipos de datos diferentes, se guardaran los datos en 2 tipos de índices distintos. Además, los datos se "mapean" automáticamente, es decir, no es necesario definir los tipos de datos antes de dárselos a Elastic, este automáticamente les da el tipado [3].

Elasticsearch recibirá los datos de 2 fuentes diferentes: los datos de Filebeat que envíe Logstash y los datos de Metricbeat que recibirá directamente desde los clientes. Los datos que se reciban desde Logstash, es decir, los logs, estarán formateados en JSON y mejor preparados para ser explotados (esa es la funcionalidad de Logstash), de manera que Elastic creará un índice para esos datos y no los formateará. No obstante, los datos de Metricbeat no pasarán por el filtro de Logstash y serán enviados directamente a Elasticsearch. Metricbeat tiene la funcionalidad de enviar las métricas formateadas ya a JSON y Elastic viene con índices predeterminados, uno de ellos está preparado para

Metricbeat de manera que se generará un índice predefinido y se podrán ver las métricas inmediatamente.

Una vez los datos están almacenados, entra en escena la verdadera potencia de Elasticsearch: Análisis y Búsqueda. Elasticsearch ofrece una API de tipo REST para hacer las búsquedas y para gestionar la aplicación. Esta interfaz ofrece las búsquedas por texto explícito, también conocidas como búsquedas full-text o búsquedas estructuradas, que simulan consultas SQL y, de hecho, existe una integración para hacer consultas SQL a Elasticsearch. También es posible hacer uso del lenguaje de peticiones que proporciona Elastic (QueryDSL) sin hacer uso directamente de la API [4].

A la hora del análisis al tener los datos en formato JSON, Elastic los aprovecha y genera conjuntos de datos principalmente estadísticos como, por ejemplo: La moda de los valores, valores poco comunes o Top 5 valores más comunes que se ven muy fácilmente desde Kibana. También cabe destacar que el uso de los índices de Elasticsearch hace que la búsqueda sea muy rápida. Dichos índices se habilitan con una cantidad de datos no muy grande (aproximadamente 10 entradas desde la creación de un índice o del añadido de campos nuevos) y teniendo un retraso de aplicación del índice prácticamente nulo. Según la documentación oficial, dicho retraso sería de 1 segundo aproximadamente [5]. Para usos más estadísticos de los datos, Elasticsearch hace uso de machine learning para detectar anomalías en los valores o rarezas estadísticas.

El aprovechamiento de datos por parte de Kibana no tiene mucha complejidad de explicación, como bien se ha comentado anteriormente, ElasticSearch tiene una API REST por la cual se pueden hacer búsquedas, obtener un análisis y recoger datos del propio Elasticsearch. Toda la información que obtenga Kibana será a través de esta API.

4.1.2. ¿Qué Diferencia Tiene una Base de Datos Relacional de Elasticsearch?

Un elemento a destacar es que, aunque Elasticsearch tenga la capacidad de funcionamiento de una base de datos, es totalmente diferente de una base de datos relacional como una de Oracle, Elasticsearch tiene la posibilidad de funcionar de manera totalmente distribuida sin tener que ejecutar réplicas de un nodo a otro o solo teniendo una instancia de escritura, ElasticSearch se puede configurar como un conjunto de nodos y funcionará de manera distribuida automáticamente permitiendo replicación de datos añadiendo una mayor disponibilidad y tolerancia a fallos ante una base de datos relacional. Se recuerda que en este proyecto sólo se construye una instancia ElasticSearch en el nodo servidor de manera que no se construye un sistema distribuido, pero es una propiedad muy interesante de esta aplicación.

Otra diferencia relevante es que es una base de datos NoSQL, es decir, el motor de búsqueda no se basa en el lenguaje SQL, sino que se basa en búsquedas a parámetros y/o objetos JSON (QueryDSL). Esto supone que los datos no se guardarán como tablas sino como objetos JSON. Además, Elasticsearch, está construido sobre Apache Lucene, es

decir, todas las funcionalidades y propiedades de búsqueda están basadas en la dicha aplicación. Una de estas propiedades que cabe comentar es que tiene la propiedad de tener búsquedas *full-text,* las cuales comprenden y permiten búsquedas sobre texto natural (como un artículo periodístico o un documento de la universidad). Es decir, en una base de datos relacional se hace una búsqueda propiedad -> valor sobre una tabla (además que no tiene sentido guardar un documento como un libro o un artículo periodístico en una base de datos relacional) pero en una búsqueda *full-text*, con una entrada de texto normal sobre un documento, se obtienen resultados. El documento está almacenado como un objeto JSON, dicho objeto podría tener propiedades como Autor, Fecha de Publicación, Texto, etc. Para hacer una búsqueda lo que sucede es que se enviará un objeto JSON con la propiedad query que tendrá como valor el texto a buscar y esta petición atacará a los objetos JSON y consecuentemente a la propiedad Texto de dichos objetos que estén catalogados como documentos. [6]

La última diferencia que cabe destacar, pero no por ello menos importante es el añadido de los timestamps, tener una fecha para la entrada de un dato es indispensable en este proyecto pues si se busca un log, se debe saber en qué momento ocurrió ese log para poder conocer más detalles del problema que se pretende solucionar mientras que en una base de datos relacional no es relevante saber cuál es el timestamp de una entrada.

4.2. Logstash

Logstash es un motor de colección de datos open source, es decir, una herramienta que permite recibir información, darle un formato, unificar, y enviar la información al "alijo" que se desee. La idea reside en que a partir de datos "sin significado" tales como elementos en formato textual darles un sentido transformándolos en datos "con significado", en este caso, objetos JSON, de manera que los datos tienen varios atributos y valores por ejemplo la fecha y hora del log, el programa que está lanzando el log, la IP de la máquina que está generando el log, etcétera.

4.2.1. ¿Cómo Funciona Logstash?

En el apartado de entrada de datos, Logstash abre "pipelines" en uno o varios puertos donde se recibe la información, en los Beats se define la salida a dicho nodo más puerto donde esté escuchando Logstash.

Logstash trabaja con un archivo (o varios) de configuración donde se especifica todo el funcionamiento, en ese archivo se definirán 3 campos que a la vez cada uno justifica una funcionalidad del programa: La entrada de datos, el filtrado y la salida de datos (Figura 61).

En el archivo de configuración se definen los puertos de escucha y los datos de aplicación esperados, en el caso del proyecto, los Beats. En el apartado de salida de datos se define el servicio o directorio donde Logstash debe enviar el resultado final de procesar la información. Es una posibilidad enviar los datos a un directorio local o remoto, en nuestro caso y el caso más común es enviar los datos al servicio de Elasticsearch de manera que lo almacene y posteriormente aprovechados por Kibana.

Una vez recogidos los datos de entrada, en la sección de filtrado se puede añadir información extra, hacer modificaciones y/o sustracciones. Este es el apartado más importante del archivo de configuración y a la vez, la característica más útil del programa. Aquí se pueden definir patrones de manera que cada vez que llegue un log se pueda filtrar por dichos patrones y, si hay un patrón coincidente, añadir, por ejemplo, un campo extra que defina un valor en concreto que interese mostrar posteriormente en Kibana. Cabe indicar que, aunque varios logs estén en un mismo archivo, los Beats los envían de tal forma que Logstash procesa los logs de uno en uno. El filtrado de patrones es muy versátil y potente por ese mismo motivo, al definir varios patrones se están definiendo varios tipos de logs que muy probablemente sea equivalente a logs de varias aplicaciones. Los directorios de logs que se definan en los Beats para el envío junto al filtrado de patrones de Logstash da la potencia de elegir qué registros se quieren controlar y cómo se quiere hacerlo. ¿Interesa añadir un campo que defina si una conexión SSH ha sido aceptada o denegada? Se define un patrón donde coincida el log de las conexiones aceptadas/fallidas y se añade un campo que indique el estado de la conexión o se coge el patrón predefinido para los logs del tipo syslog y se añaden condiciones para crear el mismo campo. ¿Se desea procesar solamente los errores que esté generando un servidor Apache? Se crea un patrón que coincida con los logs de Apache que indiquen errores, o se coge el patrón predefinido que se proporciona Grok y se añaden las condiciones necesarias. La potencia que tiene el filtrado de Logstash es muy grande y da muchas facilidades para crear, modificar y ver el contenido necesario para el contexto necesario del clúster del proyecto. Es importante mencionar que, además del filtrado que se haga y de los datos que se procesan, Logstash de por sí añade campos tales como la dirección IP de quién envía el log, el timestamp y demás información genérica [7].

4.2.2. Filtros Grok

Se ha estado comentando sobre la potencia del filtrado de Logstash pero no se ha comentado como lo hace, una de las formas de filtrado tiene de nombre filtro Grok. Grok delimita sintácticamente la información de manera que, a partir de una pieza entera de texto, se obtenga un número arbitrario de piezas con un significado que las difiera, es decir, como se ha comentado anteriormente, a partir de un log que es un texto lineal con un significado único, separarlo en piezas con un significado cada una como la aplicación que genera el log, el tipo de log, el propio mensaje del log, etcétera. Además, Grok también permite definir el formato de los datos, es decir, la información que recibimos está en formato textual, pero se pueden formatear los números como enteros, los textos que sean "true" o "false" como booleanos, etcétera. Toda la potencia de generar un objeto JSON a partir de un texto la permiten los filtros Grok.

4.2.2.1. Sintáxis de Grok

Para poder explicar la sintaxis de Grok se usará con un ejemplo, se pone el caso de que exista un log como el siguiente (Figura 2):

```
Apr 20 14:45:52 client2 sshd[935]: pam_unix(sshd:session): session
opened for user root by (uid=0)
```

Figura 2: Log de syslog de Ejemplo

Se pueden apreciar a primera vista campos interesantes que interesaría poder analizar independientemente de los otros campos del log tales como la máquina que genera el mensaje, la aplicación que genera el mensaje y su PID, el timestamp o el mensaje de por sí.

Aplicando un filtro de esta forma se podría "recoger" el log cuando llegue al pipeline (Figura 3):

```
%{SYSLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOST:syslog_hostname}
%{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?:
%{GREEDYDATA:syslog_message}
```

Figura 3: Filtro Grok para logs de syslog.

Se pueden ver en este filtro Grok varios detalles:

- Los elementos a recoger y las variables declaradas se indican de la siguiente forma: `%{TIPO_DE_DATOS:variable}`, de manera que una vez se pase el filtrado no habrá un solo elemento de tipo texto sino varios elementos de varios tipos distintos. La separación de campos vendrá o bien por lo que el tipo de datos sea capaz de reconocer o por el separador explícito que haya posteriormente al campo.

- En casi todo este filtro la separación entre elementos será un espacio en blanco explícito, si se tiene en cuenta el primer elemento a filtrar, se recoge completamente el objeto del tipo `SYSLOGTIMESTAMP`, donde en el propio tipo se esperan dos espacios y los pertinentes campos que tiene un timestamp de syslog por lo que la variable recogerá lo siguiente: `syslog_timestamp: Apr 20 14:45:52` y no sólo `Apr`.

- Se puede apreciar que la variable `syslog_pid` viene precedida de unos paréntesis y de un símbolo de interrogación, esto indica que el campo puede aparecer o no, por lo que, si llega un log sin esa parte, pasa por el mismo filtro sin tener que crear otro filtro para otro tipo de logs. Sucede lo mismo con los dos puntos de después, esos dos puntos no pertenecen a ningún campo y poniéndolos explícitamente dan la función de separador explícito, al igual que el espacio en blanco, poner ": " es equivalente a después de esta variable hay que saltar esos 2 caracteres para empezar con la siguiente. Estos campos explícitos se pueden poner como

`\campo_explícito` en caso de que haya alguna inconsistencia con las expresiones regulares que se deseen poner.

Una vez pasado el log por el filtro tendríamos lo siguiente en formato JSON (Tabla 1):

Variable	Valor
syslog_timestamp	Apr 20 14:45:52
syslog_hostname	client2
syslog_program	sshd
syslog_pid	935
syslog_message	pam_unix(sshd:session): session opened for user root by (uid=0)

Tabla 1: Esquema de log transformado a JSON por el filtro Grok.

Grok también permite añadir texto explícito para el filtrado, esto se ha podido ver anteriormente y podemos aprovecharlo de la siguiente forma (Figura 4) (Figura 5):

```
Apr 20 15:19:06 client2 sshd[1261]: Accepted password for root from
192.168.56.111 port 34818 ssh2
```

Figura 4: Log de ejemplo de SSH

```
%{SYSLOGTIMESTAMP:ssh_timestamp} %{SYSLOGHOST:ssh_hostname}
sshd(?:\[%{POSINT:ssh_pid}\])?: %{WORD:ActOrDny} password for
%{WORD:Remote_User} from %{GREEDYDATA:remote_address} port
%{NUMBER:port} ssh2
```

Figura 5: Filtro Grok para logs de SSH

Se puede ver que inicialmente el filtro funciona de la misma manera que el anterior pero después de la asignación a `syslog_pid`, cuyo valor sería `1261`, y los dos puntos explícitos se puede presenciar que el mensaje del propio log, que en el anterior caso equivalía a la variable `syslog_message`, no va enteramente asignado a una variable como ocurre en el caso anterior, sino que se divide en varias subsecciones:

- La variable `ActOrDny` que tendrá en este caso la palabra "`Accepted`"

- El texto explícito "`password for`"
- La variable `Remote_IP`, que tendrá como valor `192.168.56.111`
- El texto explícito `from`
- La variable `pre_syslog_message` que contendrá la parte restante del mensaje.

Este filtro está preparado explícitamente para filtrar las conexiones y/o intentos de conexiones por SSH a un nodo que se esté controlando de manera que todo log que indique si se acepta o si se falla una contraseña será filtrado por este filtro especial y no por el filtro genérico de los mensajes syslog definido previamente. La utilidad interesante y el elemento que diferencia este filtro del anterior reside principalmente en `ActOrDny` donde esta variable sólo puede tener 2 valores, "`Accepted`" o "`Failed`". A partir de esto se da la posibilidad, con esta variable, de hacer un conteo de las conexiones aceptadas y fallidas en los nodos y, por ejemplo, graficarlos o que se puedan buscar todos los logs con la variable `ActOrDny` a `Failed` y obtener información como a qué hora se produjo la conexión o desde qué IP se produjo. Como también se ha visto, en este caso, el mensaje está incompleto debido a que se necesitan poner al menos la palabra "`password`" para poder filtrar este tipo de logs en concreto, con la función `add_field` de Logstash se puede añadir explícitamente un texto a la variable (Figura 6).

```
add_field => {"syslog_message" => "%{ActOrDny} password for
%{Remote_User} from %{Remote_IP} %{pre_syslog_message}"}
```

Figura 6: Ejemplo de método de añadido de campo en Logstash.

De esta forma se obtiene el mensaje con el campo `syslog_message` que tienen todos los demás.

Este ejemplo dado se usará para poner un caso de uso donde el objetivo sea practicar observabilidad sobre las conexiones SSH que se produzcan en nuestro entorno para poder capturar un caso de acceso ilegal rápidamente.

4.3. Kibana

Kibana es el punto final de la pila, es la interfaz gráfica donde se exponen los datos que haya en Elasticsearch, de manera que, esos datos sean mucho más aprovechables en más de un sentido pues Kibana permite hacer búsquedas de todo tipo contra la base de datos, pueden ser por palabra clave, por campo concreto, por fecha, etcétera. Además, Kibana permite graficar de muchas formas, están las gráficas típicas de barras o de tarta, pero también hay gráficas de localidad, de mapas de calor y demás tipos.

4.3.1. ¿Cómo Funciona Kibana?

Kibana tiene como función dar una visión usable de toda la información que proporciona Elasticsearch y potenciar esa información con varios elementos. Primeramente, Kibana accede a la información con constantes peticiones a la API de Elasticsearch por lo que saca información en tiempo real de todos los índices existentes que tenga la base de datos temporal. Estos datos tienen información como el timestamp o variables JSON predefinidas como la IP origen del generador del mensaje. Estos datos tienen el potencial de ser buscados con facilidad, por ejemplo, sabiendo la fecha y hora de la generación de un log. Desde Kibana se puede filtrar por los logs de los últimos 10 días, a este filtro se puede añadir que se busca una variable con un valor en concreto, como una IP concreta o un índice en específico y así recursivamente se pueden filtrar y procesar los datos a gusto del administrador, una forma de procesar los datos sería hacer un conteo de estos o sacar el valor medio del conteo de dichos logs o incluso se puede hacer una comparación con otro tipo de log para saber cuál es más común en el cluster, las posibilidades que ofrece Kibana son muchas y muy variadas. [8]

A partir de este punto, Kibana ofrece muchos menús y muchas posibilidades de trabajar con estos datos, como se ha mencionado antes, los datos se pueden filtrar y graficar, pero a partir de este punto se generan muchas ramificaciones tales como geolocalizar la información, añadir graficas de varias fuentes de información y de diferentes formas, por ejemplo, desde el submenú Canvas, se pueden generar gráficas con la intención de presentarlas en Powerpoint o desde Dashboard se puede hacer un menú general con las gráficas que nos interesen para monitorizar dicha información. Kibana también ofrece funcionalidades de seguridad y funciones de alerting para evitar tener que estar monitorizando el sistema constantemente. Estas funciones de seguridad destacan logs que se puedan considerar maliciosos o tener en cuenta otro tipo de alertas, tanto predefinidas como personalizadas. Cabe destacar que varias de las funcionalidades comentadas son de pago y no vienen con la versión de uso gratuito que se utiliza para este proyecto. Aunque haya muchas funcionalidades, en este proyecto se hará uso únicamente de apartados dedicados únicamente a la observabilidad y a la ayuda de su práctica, cabe recordar que la función principal de Kibana es esta, aunque ramifique en muchas más opciones. Se hará especial énfasis en los menús Observability, Analytics.

4.3.2. Observability

Como bien dice su nombre, este menú va dedicado a la observabilidad, desde aquí, junto a los Beats indicados se puede ver los elementos troncales de la observabilidad, es decir, monitorización, registros y trazabilidad. Se puede indicar que este es más bien un menú general y dentro de esta sección están las secciones dedicadas a cada uno de los subelementos donde la información mostrada es más precisa y con más opciones de visibilidad. Por ejemplo, desde el menú general de observabilidad se puede ver la cantidad de logs que llegan y el listado de las máquinas que están enviando Beats al servidor junto a datos del estado de dichas máquinas en formato numérico. Pero si se quiere profundizar en qué logs están llegando o en más datos de las métricas junto a

gráficas de dichos valores se debe acceder a los submenús. El menú de Observability vendrá muy bien para poder observar métricas, pero en cuanto al tema de logs se utilizará Analytics, esto es debido que en Observability no sucede el procesado de logs como sí lo hace en el siguiente menú (Figura 7).

Figura 7: Menú principal de Observability

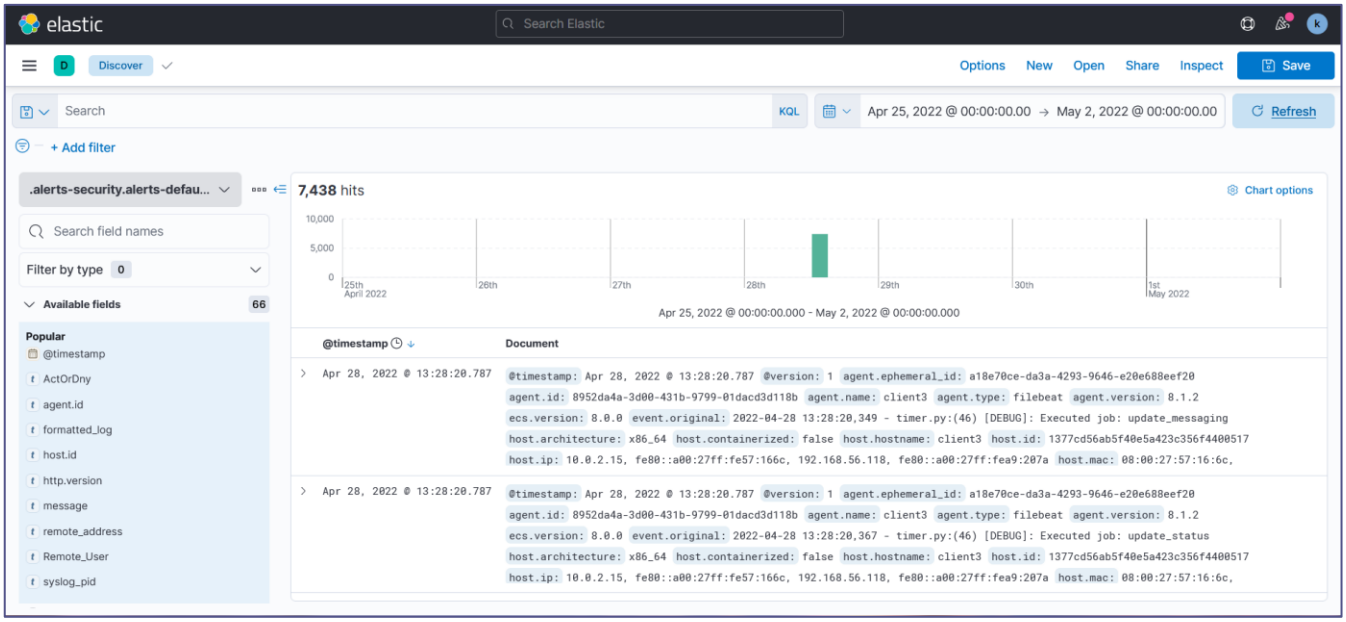


Figura 8: Entrada general de Discover

4.3.3. Analytics

Desde aquí se puede modificar y observar los datos al gusto del administrador, es una función muy potente para poder practicar la observabilidad, de manera sencilla se pueden observar los casos de uso que interesen según el contexto que se quiera dar. Se hará énfasis en Discover y en Dashboard.

El menú de Discover permite coger la información de los logs ya procesados por Logstash o FluentD y modificarlos. Se puede filtrar por campos y valores y se puede elegir qué información interesa ver (Figura 8). Por ejemplo, es posible ver para cada campo metadatos como tales como los 5 valores más comunes que puede tener. Es una herramienta muy potente pues a partir de toda la información que se recibe, se filtra exactamente qué es lo que se quiere ver, por ejemplo, se tiene el objetivo de ver todas las conexiones SSH fallidas que hay recibidas, a qué máquina se han hecho esas conexiones, el mensaje del log y ver quién ha intentado conectarse a esa máquina. Antes que nada, se debe tener los campos preparados, en el capítulo anterior se procedía con el filtrado a este tipo de logs ya hecho, en este caso, lo relevante es tener un campo dedicado a identificar si el intento de conexión ha sido aceptado o denegado. A partir de los campos necesarios simplemente se filtran los valores que interese ver, en este caso interesa hacer lo siguiente:

1. Primero se quiere ver sólo los logs de conexiones SSH fallidas por lo que pulsando en Add Filter (esquina superior izquierda) se añadirá que se filtra por el campo `ActOrDny` (el campo que indica si una conexión ha sido aceptada o fallida) y se indicará que se filtrará por valor y que este sea `Failed`.

2. Ya están disponibles todos los logs de conexiones fallidas, pero en los logs se ve demasiada información y es difícil identificar información clave como a cuál de los clientes se está haciendo una conexión o desde que IP se está haciendo la conexión. En el submenú izquierdo se pueden añadir qué campos se quieren ver de manera que se muestren los campos que indiquen a quién se está intentando hacer la conexión y el mensaje original. Es posible también añadir con qué usuario se está haciendo la conexión SSH y desde qué IP se está haciendo, en este caso se puede ver fácilmente esta información con el mensaje original SSH, pero tener esta información disponible para filtrar es muy útil si se busca un intruso en concreto. Al finalizar habrá una vista predefinida para utilizarla posteriormente se guardará y se le pondrá un nombre. Una vez guardada si interesa ver ese menú simplemente en la barra de búsqueda se podrá el nombre del filtro y se cargará (Figura 9).

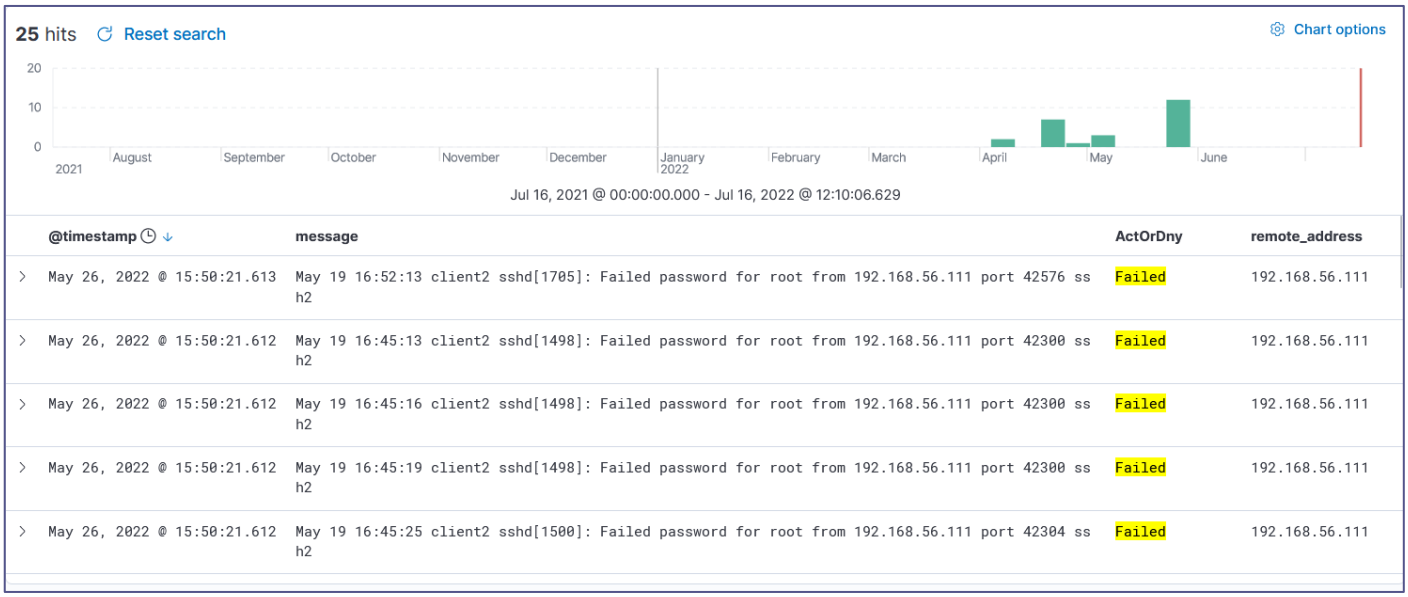


Figura 9: Esquema personalizado de conexiones fallidas desde Discovery

Se explicará los menús de Dashboard y Lens puesto que el primero es dependiente del segundo para su funcionamiento. Lens permite crear gráficas a partir de la información de los logs, las gráficas pueden ser de cualquier tipo y pueden tener la información filtrada de la misma manera que en Discovery. Dashboard junta las gráficas formadas por Lens y las agrupa en un menú de visualización general. Es muy interesante tener la posibilidad de poder graficar algunos tipos de información para identificar aún más rápido un problema. A partir del ejemplo anterior se indicará como ejemplo demostrativo cómo hacer una gráfica que indique un conteo de las conexiones SSH recibidas:

1. Primero deberá estar preparado el filtro en Discovery, también se deberá tener claro sobre qué variables se quiere generar una gráfica, en este caso, la principal variable a observar es `ActOrDny` que indica si se han aceptado o denegado conexiones SSH.

2. Yendo al submenú de Dashboard pulsaremos la opción "Create Dashboard" en el caso que interese tener el panel personalizado. Una vez dentro, se seleccionará el modo editar en el caso que no esté activado (esquina superior derecha) y se seleccionará la opción "Create visualization" (Figura 10). Esto abrirá el menú de Lens, donde se podrá ver en una columna en la izquierda qué datos se desean insertar y de qué índice (esto será relevante posteriormente para diferenciar datos de Logstash y de FluentD). Seleccionando la variable deseada, es decir, ActOrDny, se selecionará explícitamente o se arrastrará al campo blanco y se generará una gráfica automáticamente junto a sugerencias por parte de Kibana. Para este caso en concreto se seleccionará una gráfica de barras, no obstante, hay muchas posibilidades como graficas de área, mapas de calor, graficas de tarta y demás opciones (Figura 11). Es posible que las opciones de filtrado que se preseleccionan no sean las que se deseen, en este caso, interesa tener en el eje y el conteo de `ActOrDny` distinguiendo sus valores y en el eje x el paso del tiempo. Para más inri se pueden definir los posibles valores con la paleta de colores

"Status" y que los colores coincidan alfabéticamente. De esta manera las conexiones aceptadas saldrán en verde y las fallidas en rojo.

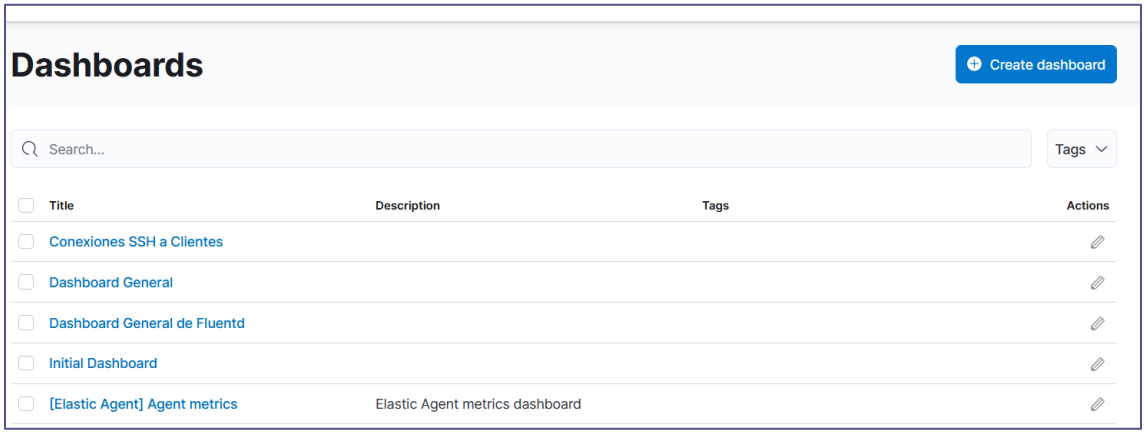


Figura 10: Menú inicial de Dashboard

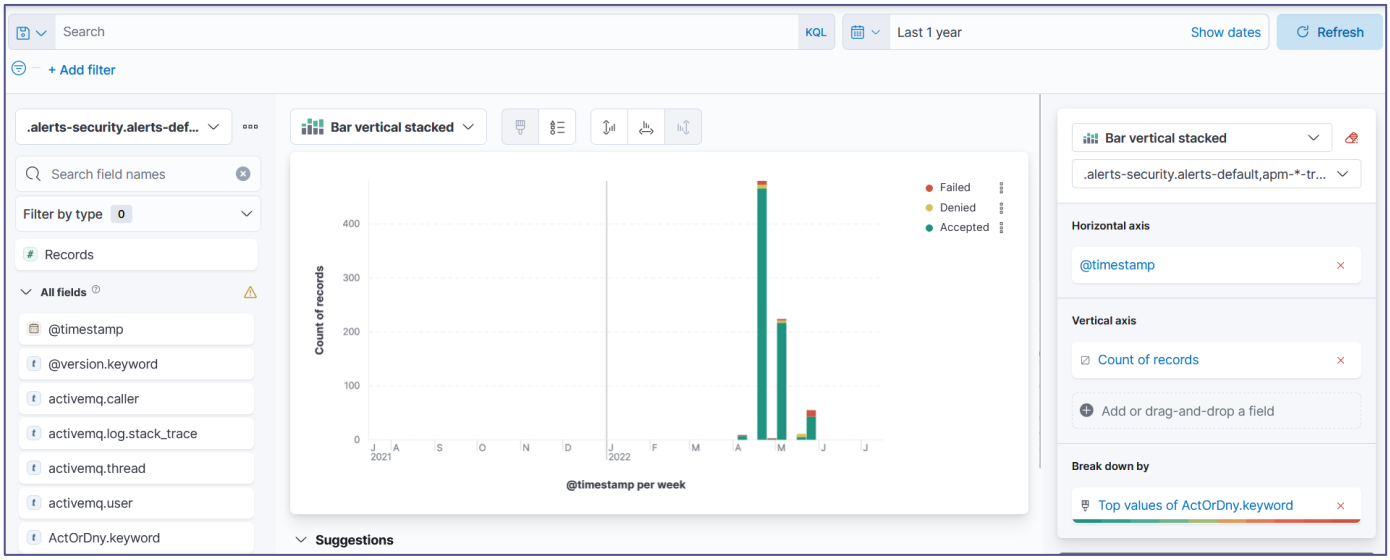


Figura 11: Esquema de creación de métricas de Lens

4.4. Beats

Se ha estado comentando durante todo este tiempo cómo funciona el elemento central que recibe los logs y los procesa, pero ¿cómo se envían los logs o las métricas? Aquí es donde entran los Beats. Los Beats tienen 2 funciones, leer la información pertinente y enviarla al destinatario adecuado cada cierto intervalo de tiempo, como si fueran pulsos (de ahí su nombre). Hay varios tipos de Beats que leen e interpretan la información de manera distinta tales como Filebeat, que lee y envía logs, Metricbeat, que lee y envía métricas, Packetbeat, que trabaja con datos de red tales como envío y recepción de paquetes, Heartbeat, que comprueba si cierta aplicación está funcionando o Functionbeat, que tiene la misma funcionalidad en arquitecturas descentralizadas [9]. Se hará uso de los dos primeros tipos de Beats, Filebeat y Metricbeat.

4.4.1. Filebeat

Filebeat lee los archivos indicados y los envía donde proceda, este Beat funciona principalmente para enviar logs a una herramienta que los procese, de hecho, desde Filebeat sólo se encuentran 2 opciones de envío por defecto: ElasticSearch y Logstash. Se puede enviar los datos directamente a Elastic pero no serían procesados pertinentemente, solo serían transformados a JSON sin pasar por la serialización que hace Logstash. Enviar los datos a Elasticsearch puede ser útil en el caso donde se envíe información ya procesada o se envíe información que no necesite dicho procesamiento como métricas.

Cabe destacar que, aunque solo existan estas opciones, Filebeat enviará la información a un host y a un puerto sin hacer ninguna identificación de a quién se le está enviando la información por defecto a no ser que se indique lo contrario haciendo uso de certificados de seguiridad y conexiones https. Lo que se quiere indicar con esto es que existe la posibilidad de "engañar" a Filebeat indicando que se está enviando la información a Logstash pero realmente se está enviando a FluentD, por ejemplo. En este caso concreto es una ventaja pues evita la necesidad de configurar Filebeat a la hora de cambiar Logstash por FluentD.

4.4.2. Metricbeat

Metricbeat tiene la misma mecánica de funcionamiento que Filebeat pero no está preparado para enviar archivos sino que lee y envía métricas del host donde esté alojado. Desde un servidor central se pueden ver datos de los hosts tales como consumo de CPU, consumo de memoria, uso de las interfaces de red e incluso que procesos se están ejecutando en dichos hosts siendo esto último equivalente al comando top de los sistemas Unix. Metricbeat envía las métricas formateadas por lo que no es necesario que Logstash las procese, es por ello que se envía la información a Elasticsearch directamente.

5. Puesta en Marcha de la Pila ELK

Una vez se han definido cuáles son las herramientas que se van a usar en la pila ELK y cómo funcionan, se procederá a usarlas todas en un clúster de 3 máquinas virtuales que serán montadas en VirtualBox de manera que una de dichas máquinas será el servidor central que recogerá la información y las otras 2 harán de nodos que envíen los logs y demás información al servidor central.

Una vez esté el clúster virtual y el software necesario en funcionamiento se harán pruebas de funcionamiento que simularán casos de uso realistas para poder demostrar la ventaja que supondría tener una gestión centralizada de los logs en contra de no tenerla.

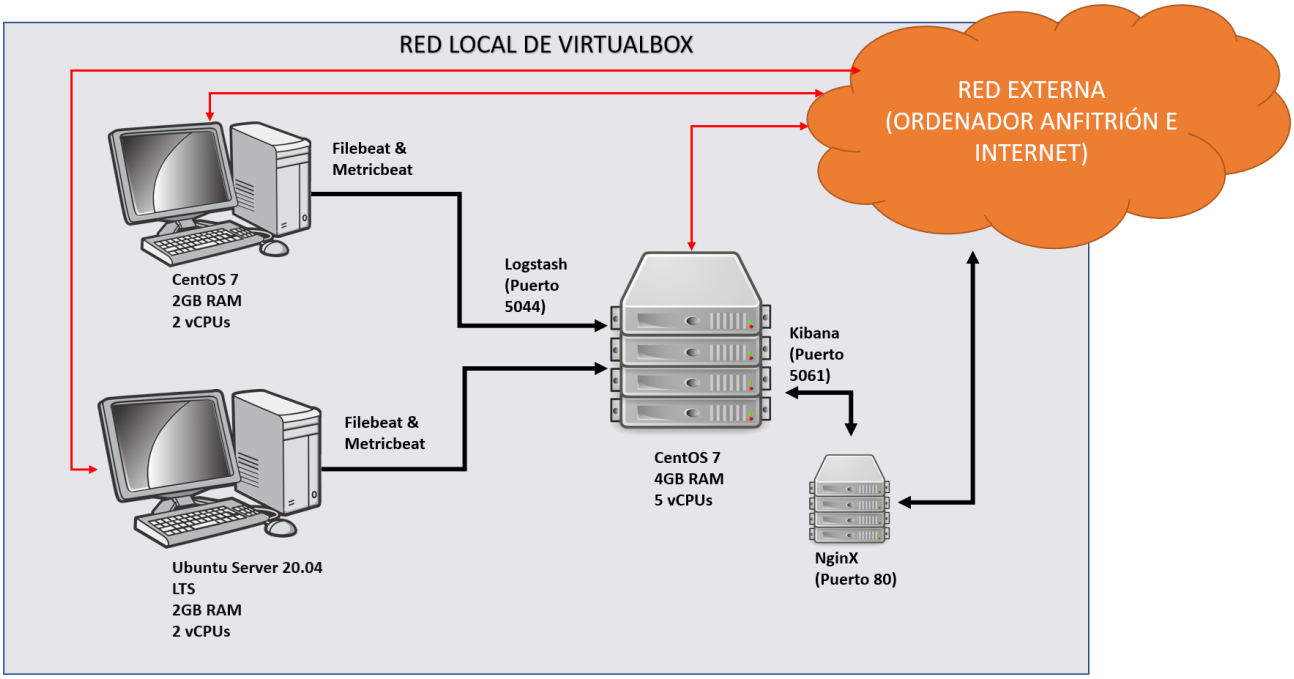


Figura 12: Esquema de la red local que se va a usar.

5.1. Montado de Máquinas Virtuales

Para crear y gestionar el despliegue de las máquinas virtuales se hará uso de VirtualBox, una herramienta de open source muy conocida para la virtualización. A través de esta herramienta se montarán 3 máquinas virtuales; 2 clientes y un servidor. El servidor tendrá como sistema operativo CentOS 7 y contará con 5 procesadores virtuales junto a 4GB de memoria RAM. El servidor contará con la Pila ELK junto a un servidor proxy de uso libre llamado NginX para poder acceder a Kibana por el puerto 80 debido que en el

caso de no tener dicho proxy el acceso a Kibana sería por el puerto 5601. Cada cliente tendrá un sistema operativo distinto, el primero tendrá el mismo sistema operativo que el servidor, es decir, CentOS 7, mientras que el otro cliente tendrá como sistema operativo Ubuntu Server 20.04 LTS. Ambos clientes tendrán 2 CPUs virtuales y 2 GB de memoria RAM. Para sincronizar los relojes de las 3 máquinas se hará uso de un programa integrado en los sistemas operativos (timedate-ctl) que hará acceso a un servidor NTP. (Figura 12)

6. Casos de Uso en la Pila ELK

Teniendo todo configurado toca poner a prueba los sistemas y demostrar la potencia que realmente tiene utilizar un esquema como este. Se utilizarán varios casos de uso para indicar estos hechos: Se una comparativa de comprobación de conexiones SSH a los clientes y se configurará un ejemplo de práctica de observabilidad sobre un servidor Apache.

6.1. Conexiones SSH

Se supone el siguiente caso; se quiere hacer una comprobación de cuantas conexiones SSH se han hecho a los clientes y cuáles de estas han sido conexiones fallidas, bien por despistes, por negligencia o por tener intenciones de infiltrarse en el sistema. Se puede escalar el caso y suponer que en vez de tener 2 clientes como el caso práctico que se usa, hay 100. El procedimiento manual para comprobar dichas conexiones sería el siguiente:

1. Conectarse a la máquina pertinente:
   `ssh user@client.`
2. Buscar los logs y filtrarlos con grep o con una herramienta equivalente:
   `cat /var/log/secure | grep ssh | grep Failed.`
3. Hacer una búsqueda manual del resultado, interpretándolo.
4. Repetir el proceso en la siguiente máquina.

Si se pide una revisión diaria del conteo de las conexiones haciendo saber que no es una tarea aislada o única sino una rutina, se crearía una tarea tediosa por no decir técnicamente imposible en términos de tiempo a sabiendas que no es la única tarea por hacer en la jornada laboral. Ciertamente en este caso de uso sólo se usan 2 clientes por motivos lógicos, aunque con las herramientas que se usan se podrían configurar 100 clientes siempre y cuando existan los recursos necesarios. Este proyecto no pretende simular un entorno totalmente real sino en intentar educar al lector/a en la potencia y utilidad de estas herramientas con ejemplos como este mismo.

Para permitir este funcionamiento se deben tener en cuenta varios factores:

- Configurar el envío del archivo `secure` en `/var/log` desde Filebeat. Este es un paso fácil pues en este caso no añadiremos ninguna etiqueta, en el filestream por defecto añadimos el archivo del directorio y se comprueba que el usuario `filebeat` tenga permisos de lectura.

- Se tendrá que definir un filtro Grok para esta información de manera que se pueda obtener información relevante de los logs que se lean, en este caso, si la conexión es aceptada o fallida o de que IP proviene el intento de conexión. El filtro Grok funcionará de la siguiente forma (Figura 13):

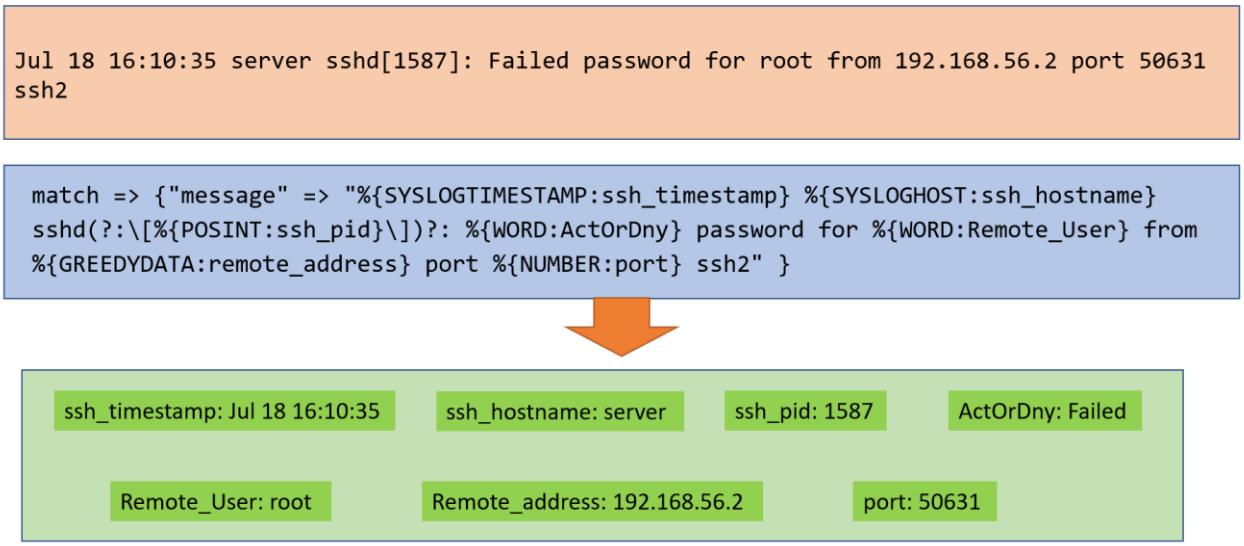


Figura 13: Esquema de funcionamiento del filtro Grok para conexiones SSH.

- Tener menús y gráficas en Kibana que permitan leer esta información y graficarla. Para esto se utilizará el campo `ActOrDny` como elemento principal para la monitorización, es decir, se harán los menús y las gráficas según el valor que tenga esta variable. Para poder leer los logs se harán 2 filtros predefinidos desde Discover, uno para ver todas las conexiones SSH y otro para ver solamente las fallidas. Después, en Dashboard se generarán 2 gráficas, una que indique las conexiones y su tipo según el paso del tiempo (Figura 14) y otra que diferenciará proporcionalmente las conexiones y con qué tipo de usuario se ha intentado entrar a dicho cliente (Figura 15).

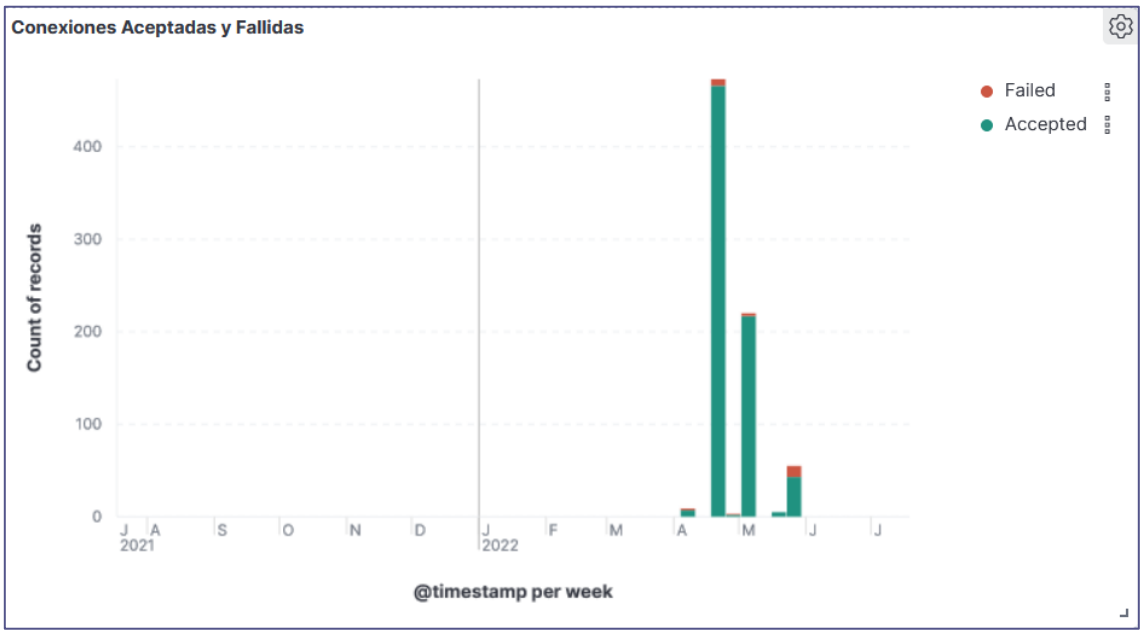


Figura 14: Grafica de conexiones SSH por el paso del tiempo.

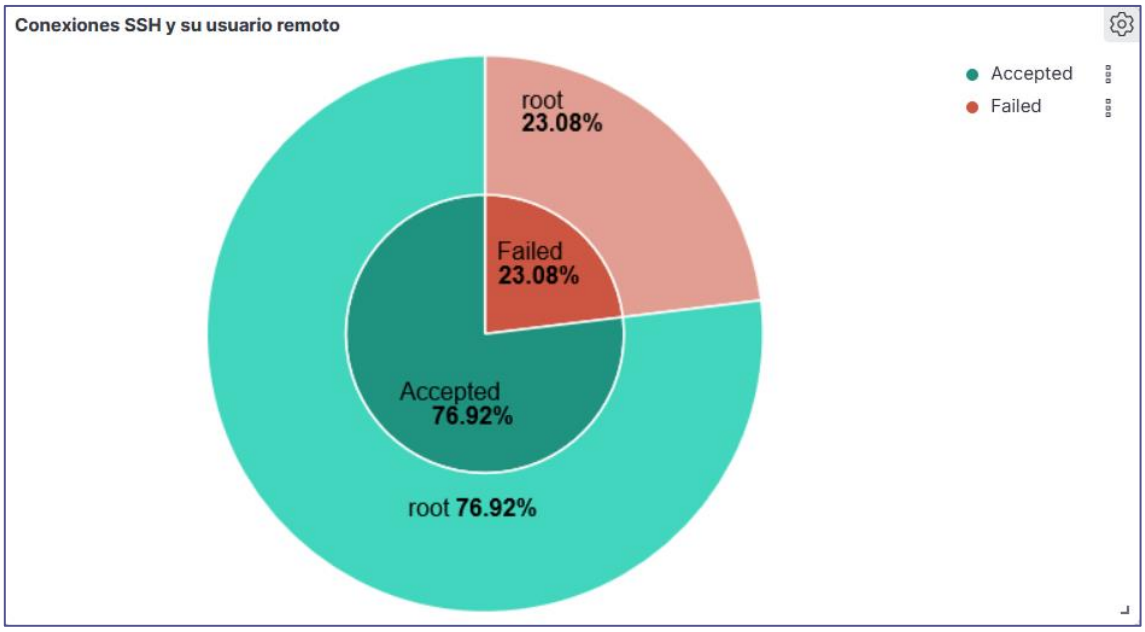


Figura 15: Gráfica de tarta indicando conexiones y los usuarios con los que se ha intentado hacer esa conexión.

Teniendo la configuración finalizada se volverá a plantear el caso de uso, existen 100 clientes y se nos solicita hacer una monitorización de las conexiones SSH que se producen a lo largo del día, se presupone como base que todos los clientes tienen configurado ya Filebeat para que lea de todos los directorios necesarios y que Logstash tiene los filtros Grok necesarios, además, se supone que ya hay construida una vista previa que identifica las conexiones SSH que sean fallidas (Figura 9) por lo que para hacer dicha revisión simplemente habrá que acceder a dicha vista previamente guardada.

Se puede comprobar que resulta en una búsqueda ínfimamente más rápida, además, al poder filtrar por muchas posibilidades y valores, es posible seleccionar un cliente en concreto cuando nos interese sin perder tiempo prácticamente. Otro posible uso que tiene esta práctica es identificar intentos de conexión de direcciones IP que no sean de confianza, esto se podría resolver añadiendo un filtro como el siguiente (Figura 16):

Cabe destacar que, en la figura anterior, cuando se menciona que *"se pide a Kibana que muestre..."* realmente la petición se hace a Elasticsearch a través de Kibana. Podemos definir vistas y gráficas a nuestro gusto según lo que se nos pida en Logstash de manera que una vez definidas, para proceder a mirar las conexiones podemos mirar la vista de Analytics o las gráficas en Dashboard, podemos filtrar por cada cliente en Analytics si lo deseamos, incluso en el caso de mirar los logs cliente por cliente en vez de mirar una gráfica o verlos todos juntos en una vista nos ahorramos el tiempo de hacer la conexión a la máquina y acceder al directorio de los logs. Una tarea que nos podría llevar el día entero nos lleva unos pocos minutos si utilizamos la gestión centralizada de logs.

Figura 16: Añadido de filtro para conexiones que no sean de confianza.

6.2. Conexiones y Logs de Apache

Otro caso de uso que se va a tratar conlleva la gestión de los accesos a un servidor Apache y también la gestión de dicho servidor a través de los logs que genera en el sistema. Para ello, primeramente, se montará un servidor Apache en un cliente, en este caso, se instalará el Apache en el cliente que tiene montado el sistema operativo CentOS 7. Lo que se pretende demostrar con este caso de uso es agilizar una gestión de las peticiones que se estén haciendo a las páginas webs que estén montadas, de manera que, para ver los accesos rápidamente a uno o varios servidores Apache, no sea necesario el acceso a dichas máquinas, sino que se pueda gestionar toda la información desde la pila ELK. Es crucial, por ejemplo, saber de dónde llegan las peticiones, a qué recurso de dirigen dichas peticiones y qué respuesta da el servidor a éstas para poder llevar una gestión correcta de Apache, además, si se hace un control de los logs que genera la aplicación per se a través de journalctl se puede deducir por qué la aplicación no se ha inicializado o por qué dejó de funcionar repentinamente, por ejemplo, sin las herramientas pertinentes supone una búsqueda manual aumentando mucho el tiempo de búsqueda del problema a solucionar.

Para poder hacer práctica del caso de uso se tienen que hacer configuraciones parecidas al caso anterior:

- Un filtro Grok, que en este caso será el filtro predefinido que existe y filtrará de la siguiente manera (Figura 17):

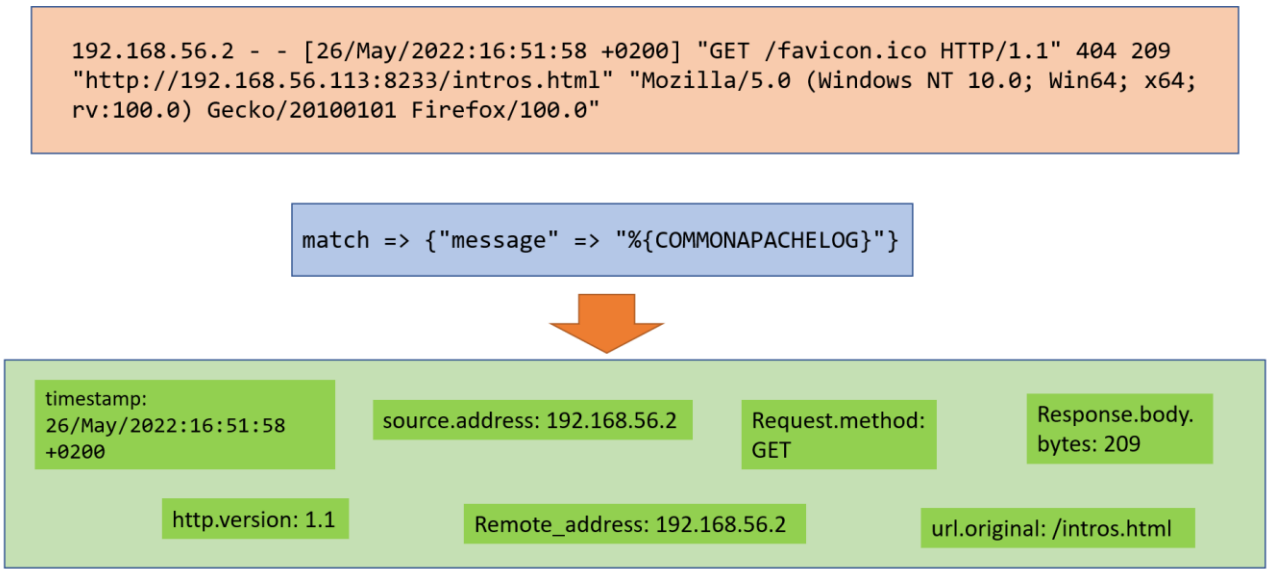


Figura 17: Esquema de funcionamiento del filtro Grok para conexiones al servidor Apache.

- Para los logs generados por Apache no será necesario crear un filtro Grok pues ya existe uno creado para los mensajes genéricos de Syslog (Figura 3), no obsante, para diferenciar los mensajes de los logs de Apache de los mensajes de syslog en general lo que se hará será poner dichos logs de apache en un archivo aparte, esta tarea se hará de la siguiente forma, se habilitará en crontab un comando que filtre en journalctl únicamente los logs de apache y los reescriba en otro archivo, dicho archivo será leído por Filebeat (deberá tener permisos para ello) y este le añadirá una etiqueta, por esa etiqueta, los logs de Apache podrán ser filtrados de los demás (Figura 18) (Figura 19).

```
* * * * * root journalctl -u httpd | tee /var/log/journal_apache
```

Figura 18: Comando a ejecutar en crontab cada minuto.

```
- type: filestream
  enabled: true
  paths:
    - /var/log/journal_apache
  fields:
    custom_service: apache_info_journal
  fields_under_root: true
```

Figura 19: Sección de la configuración de filebeat donde se leen los logs de un archivo y se añade una etiqueta.

- Se necesitará crear 2 vistas en Discover, una para las conexiones al servidor y otra para los logs que genere Apache. La primera se creará con un filtro que indique la existencia de uno de los campos que genera el filtro Grok genérico de Logstash (Figura 20) mientras que la otra vista se generará con la etiqueta, en este caso, se ha creado una etiqueta que es una variable por lo que si se busca esa variable tal como custom_service: apache_info_journal se conseguirá la vista (Figura 21).

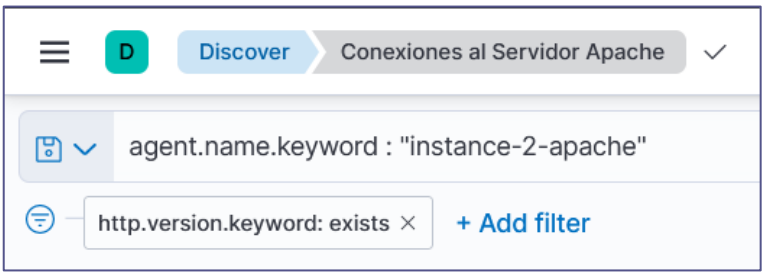


Figura 20: Configuración de la vista de las conexiones al servidor Apache.

Figura 21: Configuración de la vista de los logs que genera el servidor Apache.

- Una gráfica de tipo donut en Dashboard que muestre un porcentaje de las respuestas que envía el servidor Apache a las peticiones que recibe (Figura 22). Hubiese sido perfectamente posible poner en vez del porcentaje de respuestas, poner el conteo de estas, pero se prefirió poner el porcentaje dado que daba un resultado más visual.

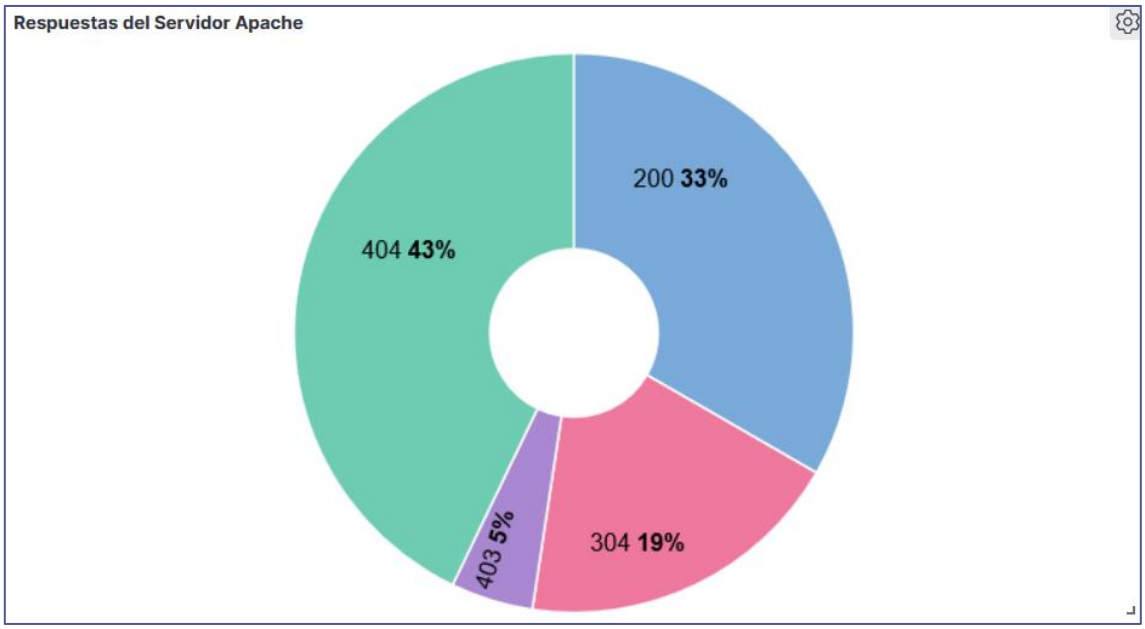


Figura 22: Gráfica de Donut indicando el porcentaje de las respuestas por parte del servidor Apache.

Teniendo las configuraciones hechas, al igual que en caso anterior, se replantea el caso de uso, para poder comprobar si un nodo que no forma parte del grupo de nodos de confianza, intenta una conexión y recibe un código 200 simplemente accediendo a la vista que se haya construido en Discover, filtrando por las IPs que no sean de confianza (equivalente a la Figura 11) y por el resultado de la petición, se obtendrían todos los mensajes de este tipo, facilitando mucho el tiempo de acción ante una situación de este tipo.

7. Pila FEK

Hasta ahora, para poder practicar la observabilidad y obtener una gestión centralizada de logs se hizo uso de la pila ELK, compuesta por Elasticsearch, Logstash y Kibana, no obstante, se propone también otra solución para obtener el mismo objetivo intercambiando Logstash por otro procesador y transformador de información llamado FluentD.

Sus desarrolladores definen FluentD como un colector de datos unificado que permite la colección, el consumo y una mejor comprensión de los datos con una filosofía muy parecida a la de Logstash por la cual, desde un punto de entrada de datos, que, en este caso serán logs, se unificarán procesando dichos logs y transformándolos a JSON y dichos datos procesados irían a uno o varios almacenes de datos temporales. Esta herramienta utiliza la misma premisa que las herramientas de la pila ELK y de la observabilidad en general para justificar su existencia: cada vez las empresas usan más herramientas y más software y dichos elementos cada vez generan más logs, para poder utilizar dicha información es necesario un elemento gestor que normalice la información, que la haga accesible y mejor visible en contra de ver esa información en su estado "natural", sobre todo, para los ordenadores que procesen dicha información porque, aunque los humanos comprendan dicha información en formato textual mejor que en cualquier otro formato, los ordenadores son "terribles" trabajando con ese formato y a la hora de trabajar con una gran cantidad de información no se puede delegar dicha tarea a una persona [10].

La idea pues, es simular el mismo funcionamiento haciendo uso de la pila FEK FluentD, Elasticsearch y Kibana. La intención es que desde el punto de la base de datos temporal y de la interfaz web no habrá ningún cambio general y hacer un análisis del proceso de configuración, de rendimiento y de funcionalidad (Figura 23).

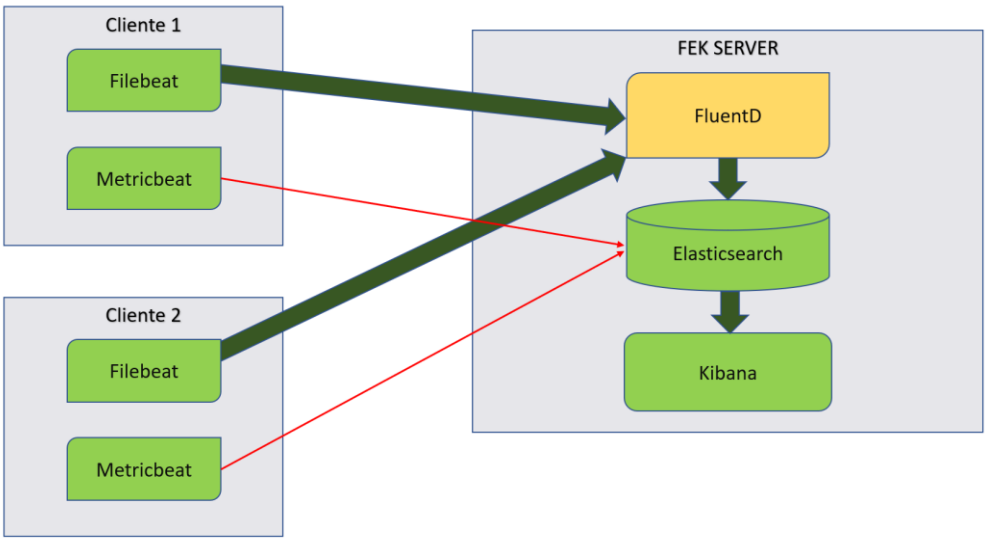


Figura 23: Esquema de funcionamiento de la pila FEK.

Se puede ver en la figura anterior como, para enviar la información se seguirán usando los Beats de que se usaban en la pila ELK, como se vio en el capítulo 4.4.1, es posible "engañar" a los Beats sin hacer ninguna configuración en ellos debido a la configuración de FluentD, que es lo suficientemente versátil para permitir trabajar con muchos tipos de tecnologías, lo cual es una gran ventaja.

7.1. ¿Cómo Funciona FluentD?

Como bien se ha mencionado previamente, FluentD tiene la misma filosofía de funcionamiento que Logstash, necesita de una entrada de datos, un filtro o procesador de dichos datos de entrada y un objetivo al que enviarle la información procesada, no obstante, a la hora de configurar la herramienta la metodología de uso difiere mucho respecto a la herramienta anterior, FluentD para los casos de uso de este proyecto trabaja principalmente con 2 campos.

El primer campo que se explicará es `source` que contiene principalmente 2 funcionalidades para el contexto del proyecto, la primera y la principal será la de receptor de datos. Al tener muchas opciones para recibir información por la red, según el tipo que se defina en `source`, FluentD abrirá un puerto u otro, el tipo que se indicará en la configuración será `beats` y, por ende, se abrirá el puerto que abre por defecto Logstash, permitiendo que desde Filebeat no se haga ningún cambio en la configuración para el envío de los archivos. Una funcionalidad incluida que tiene `source` es `parse`, que sirve explícitamente para los logs, es decir, desde el propio receptor de los datos se encuentra el campo que permite añadir los filtros Grok o filtros regex (que se usarán en un caso posterior) que permitan generar los objetos JSON. Una gran ventaja que trae FluentD y que es necesaria para este proyecto es el uso de plugins, se pueden añadir plugins tanto aprobados por los desarrolladores como no, por lo que cualquier persona puede publicar un plugin, cosa que no sucede con las herramientas de la pila ELK. Gracias a los filtros se puede o añadir metodologías de filtrado específicas bien en `parse` o bien en `filter` añadiendo mucha potencia a la herramienta. Para hacer uso de Grok es necesario instalar su plugin. FluentD cuenta con un campo `filter` que tiene una funcionalidad parecida, pero utilizando recursos propios y con su propio lenguaje para funciones como añadido, borrado o modificación de campos que haya procesado FluentD por su cuenta. También es cierto que `filter` tiene un campo `parse`, pero no tiene funcionalidad con logs [11]. Como bien se ha mencionado, para el procesado de los datos se utilizará la misma metodología que en Logstash, los filtros Grok, de manera que los datos transformados serán exactamente iguales, facilitando mucho la tarea del cambio de Logtsash a FluentD.

El segundo campo que se usará es `match`, y aunque se pueda confundir con un campo como `filter`, match recoge los diferentes campos `source` que hay por su etiqueta, si es que tienen una, o en caso contrario, recoger todos los campos posibles. Una vez recogidos, se escogerá el destino y tipo que sea, este tipo puede ser una base de datos o una aplicación, en este caso, el tipo de salida será `elasticsearch`. Por lo que los mensajes se recibirán desde Filebeat, por el puerto que usa Logstash por defecto, se

procesarán los logs con la misma técnica y los mismos filtros y se enviarán de la misma manera a Elasticsearch. Se puede ver un ejemplo de la configuración en la figura 64.

7.2. Casos de Uso en la Pila FEK y comparativa con la Pila ELK

Como se ha comentado en el punto anterior, FluentD es tan flexible en su uso que se puede simular prácticamente a la perfección el funcionamiento de Logstash por lo que los logs se procesarán igual que en los otros casos de uso y por ende el funcionamiento de estos será exactamente igual exceptuando una diferencia relevante. En la pila ELK, los mensajes procesados por Logstash pasan por un índice genérico (el cual se puede ver en la figura 8) para varios tipos de entradas donde se tienen en cuenta, por ejemplo, todos los beats existentes, en FluentD se debe definir un índice donde se envíen los logs procesados, en este caso se puede definir indicando que se usará el formato de Logstash, el cual es `logstash-DDMMYYYY` (Figura 24):

```
<match **>
  @type elasticsearch
  port 9200
  user kibanaAdmin
  password ******
  logstash_format true
</match>
```

Figura 24: Campo match del archive de configuración de FluentD

La única diferencia en Kibana será que para poder ver los datos de FluentD se deberá cambiar el índice genérico que se puede seleccionar en todos menús relacionados con recoger información de los índices de Elasticsearch por el índice de con el formato de Logstash, el cual será `logstash-*`, desde Kibana, el índice de uso se puede cambiar desde el desplegable de la esquina superior izquierda (Figura 25). Cabe destacar que no se hará mayor énfasis en los casos de uso pues el objetivo que tienen en este caso se cumple de la misma manera en FluentD al usar los mimos filtros Grok, es decir, los logs se transformarán en los objetos JSON de la misma forma y, por ende, se utilizarán de la misma manera que en la pila ELK. Los mensajes SSH se procesarán de la misma forma, pasando por la variable `ActOrDny` indicando si es `Accepted` o `Failed` y para los mensajes de Apache se hará uso del mismo filtro genérico.

Figura 25: Menú Discover genérico para los mensajes procesados por FluentD

Aunque esto último sea cierto, en términos de rendimiento hay una clara desventaja respecto Logstash, FluentD tarda más tiempo en iniciarse y en procesar los logs que recibe lo cual podría suponer una gran problemática en entornos muy críticos. FluentD compensa la pérdida de velocidad con su flexibilidad pues, no es solamente capaz de procesar logs sino que puede recibir, procesar y enviar todo tipo de información textual a todo tipo de arquitecturas de una forma mucho más automatizada tal y como se ha podido ver con la definición de los tipos, además, en FluentD hay una gran cantidad de plugins que añaden más tipos en las entradas y en las salidas, dando más adaptabilidad y una mayor facilidad de configuración al no tener que filtrar los mensajes, eventos o valores a mano como sí ocurre en Logstash. FluentD está pensado para tener muchos tipos de entradas de datos y muchos tipos distintos de salidas y poder jerarquizar la información de manera que según el dato que se procese, se prepare debidamente y se envíe con las especificaciones necesarias y concretas al programa o máquina que necesite dicha información. Logstash, aunque si tenga como posibilidad enviar información a un destino que no sea Elasticsearch y pueda recibir información que no sean logs o Beats, no está preparado para para un entorno o un contexto que esté fuera de una casuística como la pila ELK, eso quiere decir que estará optimizado para ese funcionamiento, de manera que funcionará mucho más rápido y eficazmente que FluentD, es por ello, que en el siguiente punto, cuando se abarque poner una pila local en Google Cloud, se usará la pila ELK, puesto que es lo más óptimo y funcional, conseguir demostrar que la pila que use FluentD sea más útil supondría o bien cambiar todos los elementos de la pila o bien añadir muchos más elementos al clúster dando a destacar la flexibilidad de dicho procesador de datos.

8. Gestión Centralizada de Logs en Google Cloud

Después de hacer uso de un clúster local con herramientas open source, el siguiente paso del proyecto es trasladar el clúster con la pila ELK para comprobar el funcionamiento en un entorno que esté en la nube, en este caso, Google Cloud. La idea reside en demostrar el funcionamiento y ver las posibles comparaciones en la instalación y en los casos de uso que se den en el despliegue en la nube. Posteriormente, se usarán las herramientas nativas de observabilidad de Google Cloud con el fin de compararlas a la pila ELK y poder concluir con qué solución es mejor para esta casuística.

Google Cloud es un entorno en la nube cuyo objetivo es que se puedan virtualizar y/o migrar a dicho entorno la mayor cantidad posible de infraestructuras de red y/o servicios informáticos que tenga una empresa, centralizándolos de manera que Google pueda gestionar dicha infraestructura y con sus herramientas, tanto de observabilidad como de construcción de máquinas virtuales, de gestión de costes o incluso de seguridad, suponga una mayor facilidad de uso y gestión, a la par de un coste menor, que tener dichas infraestructuras y servicios de manera física, este tipo de servicios llegan a ser bastante útiles y se usan mucho por el mero hecho del ahorro del costo inicial que supondría una infraestructura grande además de su mantenimiento y actualizaciones posteriores [12].

Como bien se ha mencionado anteriormente, Google Cloud ofrece costos más asequibles que tener una infraestructura física, pero esto obviamente supone tener unos costos superiores a 0, es por ello que Google ofrece 300 dólares americanos durante 90 días con el mero hecho de crearse una cuenta y poner una tarjeta de crédito a nombre del administrador para poder probar gran parte de las herramientas [13], aunque cabe mencionar que se restan 20 dólares al configurar la cuenta con dicho crédito. De esos 300 dólares se exceptúan elementos que ofrece Google que no tienen afectación para este proyecto como el uso de GPU's o tener máquinas virtuales basadas en Windows Server. En este proyecto se hará uso de dicho crédito para poder probar las herramientas necesarias y poder aplicar los casos de uso pertinentes.

8.1. Pila ELK en Google Cloud

Una vez creada la cuenta y habilitado el crédito de 300$USD, se puede proceder con la implementación de la pila ELK, dicha implementación sigue el siguiente esquema:

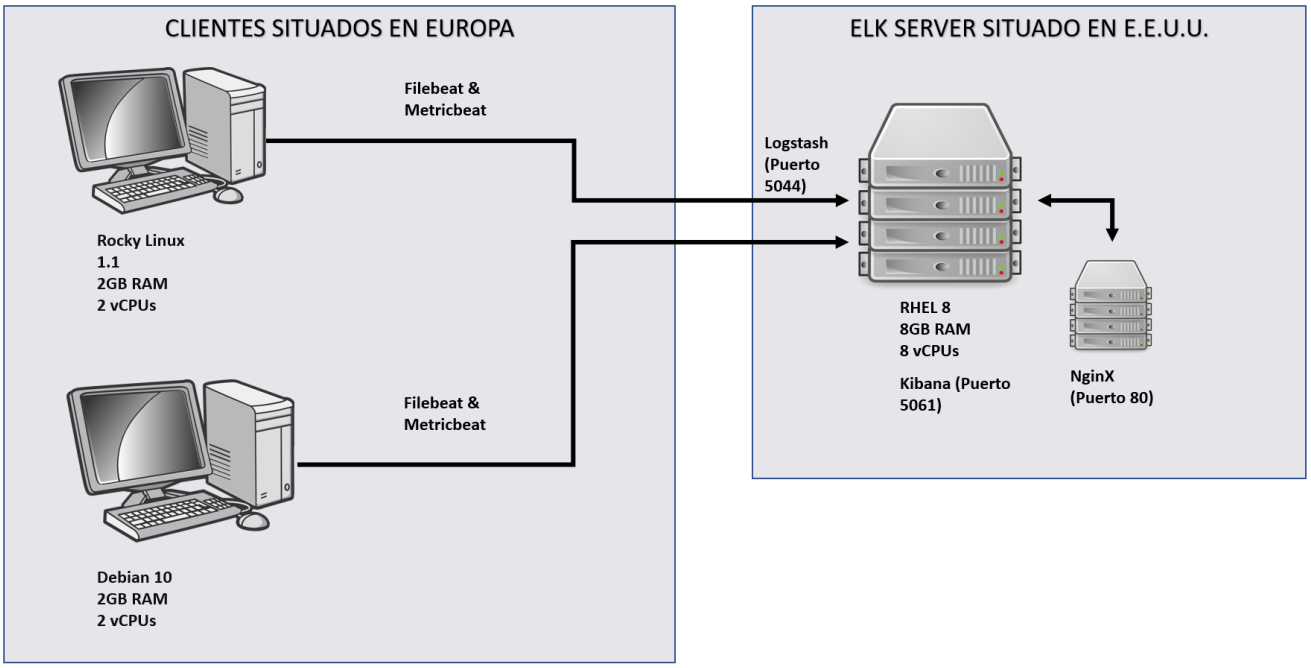


Figura 26: Esquema de configuración de los nodos en Google Cloud

En la figura 26 se puede ver cómo se harán uso de 2 regiones, la de Europa y la de Estados Unidos de manera que los clientes residirán en Europa mientras que el servidor residirá en Estados Unidos. Se puede observar el uso de otros sistemas operativos, esto es debido a que llegó una noticia indicando que en Diciembre de 2020, Red Hat hizo una publicación anunciando que CentOS 8 ya no tendría soporte [14] debido al cambio de uso que se le va a dar, se usará como un entorno de test donde todas las pruebas y actualizaciones se harán en CentOS en vez de RHEL como sucedía anteriormente, en otra publicación se da a saber que se darán licencias de RHEL 8 a cambio de esta situación para proyectos open source [15]. Sabiendo también que CentOS 7 dejará de tener soporte en 2024 se propone que el servidor ELK tenga una licencia RHEL 8 que no será gratuita pues se escogerá la licencia del listado de imágenes de Google Cloud, no obstante, será poco costosa y estará dentro de las capacidades de uso del crédito ofrecido por Google por lo que realmente no se generará un coste, además, se dará más potencia a este servidor para un funcionamiento más fluido. Otra alternativa de uso que no sea RHEL 8 y que sea totalmente open source es Rocky Linux, desarrollado por un ex desarrollador del proyecto CentOS, habrá un cliente que tendrá Rocky Linux que será el que tenga el servidor Apache. Finalmente, en la última máquina, habrá un Debian 10 en vez de un

Ubuntu Server puesto que realmente son sistemas operativos muy similares que tienen un comportamiento muy parecido, además de demostrar que la instalación de las herramientas pertinentes sólo varía en la herramienta de instalación y la configuración de dichas herramientas varía únicamente en qué directorios se debe leer la información.

Cabe también informar que a diferencia de la figura 12, donde hay un único acceso a Internet el cual es la tarjeta de red virtual de Virtual Box, en estas instancias de Google Cloud, cada una tiene su dirección privada de red, que sirve para comunicar todas las instancias de la nube entre sí y también cada una cuenta con una dirección pública variable de manera que en acceso a Internet es posible desde las tarjetas de red de las propias máquinas virtuales.

8.2. Casos de Uso de la Pila ELK en Google Cloud

Posteriormente a la instalación de la Pila ELK en el servidor y de los Beats pertinentes en cada instancia además del servidor Apache se procederá con los mismos casos de uso que en local para poder hacer una comparativa, no obstante, hay una diferencia principal a destacar, las versiones del software usado en Google Cloud son posteriores a las versiones usadas en el clúster local, en cuanto a Logstash y Elasticsearch este cambio es irrelevante, no obstante, en Kibana, el cambio de la versión 8.1 a la versión 8.2 supone la modificación de algunas visualizaciones respecto las figuras mostradas en local.

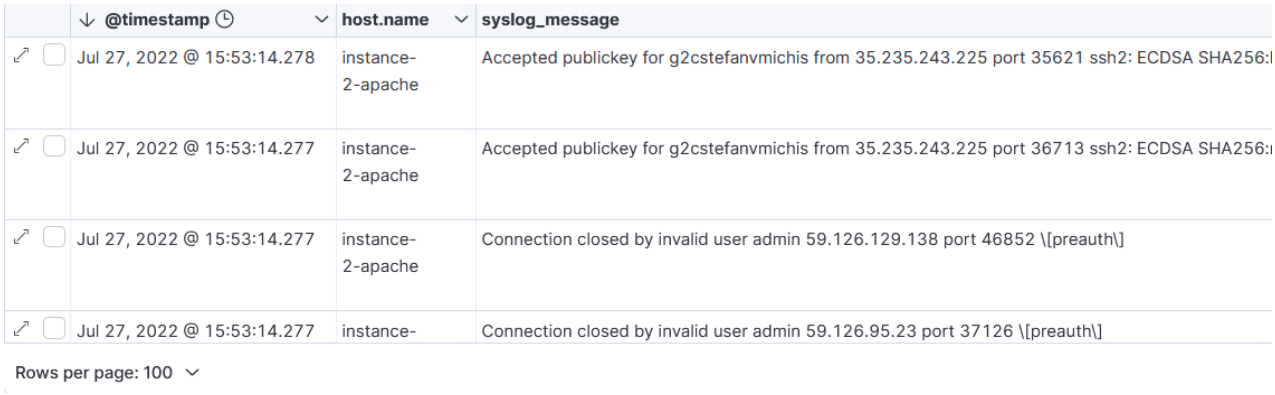
Respecto a los casos de uso que se van a ver posteriormente, se obviará el caso para el servidor Apache pues este no varía en ningún aspecto, no se puede decir lo mismo de los casos para las conexiones SSH.

8.2.1. Conexiones SSH en Google Cloud

Hay un factor muy importante a tener en cuenta y es que para los casos de uso de las conexiones SSH que se daban en local, se hacía énfasis en si dicha conexión había sido aceptada o fallida según si la contraseña introducida era correcta o errónea, en un entorno real como Google Cloud no están permitidas las conexiones SSH por contraseña, estas conexiones se hacen con un par de certificados SSL, es decir, con el uso de certificados públicos y privados, de manera que el administrador debe usar el certificado privado para conectarse a una máquina siempre que no sea desde la consola de Instancias VM de Google Cloud, donde, se usa el mismo procedimiento, pero el uso de la clave privada es automatizada.

Este cambio tiene afectación en los logs que se generan durante las conexiones SSH, ya no se aceptan o rechazan conexiones por contraseñas, ahora hay conexiones válidas, es

decir, se ha usado una clave privada de manera correcta, o inválidas, se ha usado una clave inválida o la conexión de por sí no se ha construido correctamente. Este último hecho será común pues las instancias cliente tienen una dirección IP pública, por lo que habrá máquinas en la red lanzando conexiones y peticiones a cualquier máquina con los puertos abiertos, esto también se puede aplicar al servidor Apache. Cabe mencionar que un uso de certificados SSL hace muy improbable por no decir imposible el acceso para un usuario ajeno por SSH.



	@timestamp	host.name	syslog_message
⤢ ☐	Jul 27, 2022 @ 15:53:14.278	instance-2-apache	Accepted publickey for g2cstefanvmichis from 35.235.243.225 port 35621 ssh2: ECDSA SHA256:I
⤢ ☐	Jul 27, 2022 @ 15:53:14.277	instance-2-apache	Accepted publickey for g2cstefanvmichis from 35.235.243.225 port 36713 ssh2: ECDSA SHA256:I
⤢ ☐	Jul 27, 2022 @ 15:53:14.277	instance-2-apache	Connection closed by invalid user admin 59.126.129.138 port 46852 \[preauth\]
⤢ ☐	Jul 27, 2022 @ 15:53:14.277	instance-	Connection closed by invalid user admin 59.126.95.23 port 37126 \[preauth\]

Rows per page: 100

Figura 27: Logs de Discover de las conexiones SSH en las máquinas del Cloud

Se puede apreciar en la figura 27 el cambio en la interfaz de Discover si se compara respecto con la figura 9, por ejemplo, además de 2 tipos de conexiones, las conexiones aceptadas donde aparece la clave pública y las conexiones inválidas durante la preautenticación, estas conexiones no son denegadas per se sino que se tratan de usuarios que abren una sesión SSH y que pasan cierto tiempo sin intentar autenticarse, lo que provoca que una vez pasado dicho tiempo se cierre la conexión generando ese log. Se puede presenciar también que entre los logs de conexiones aceptadas y los logs de conexiones denegadas hay diferencias esquemáticas y no siguen la misma base como en el caso de logs SSH del clúster local, por lo que se construirán 2 filtros (Figura 28) (Figura 29).

En ambas figuras se puede apreciar que el equivalente a la variable `ActOrDny` usada para el caso de uso en local tiene como nombre ahora `ssh_session`, pues `ActOrDny` simbolizaba la expresión *Accepted Or Denied* pero en este caso la variable no tiene esos valores. También se puede apreciar el añadido de variables, para las conexiones aceptadas, se añade la clave pública usada en la conexión mientras que en las conexiones inválidas, se muestra al final la fase de autenticación en el momento en el que se cerro el intento de conexión.

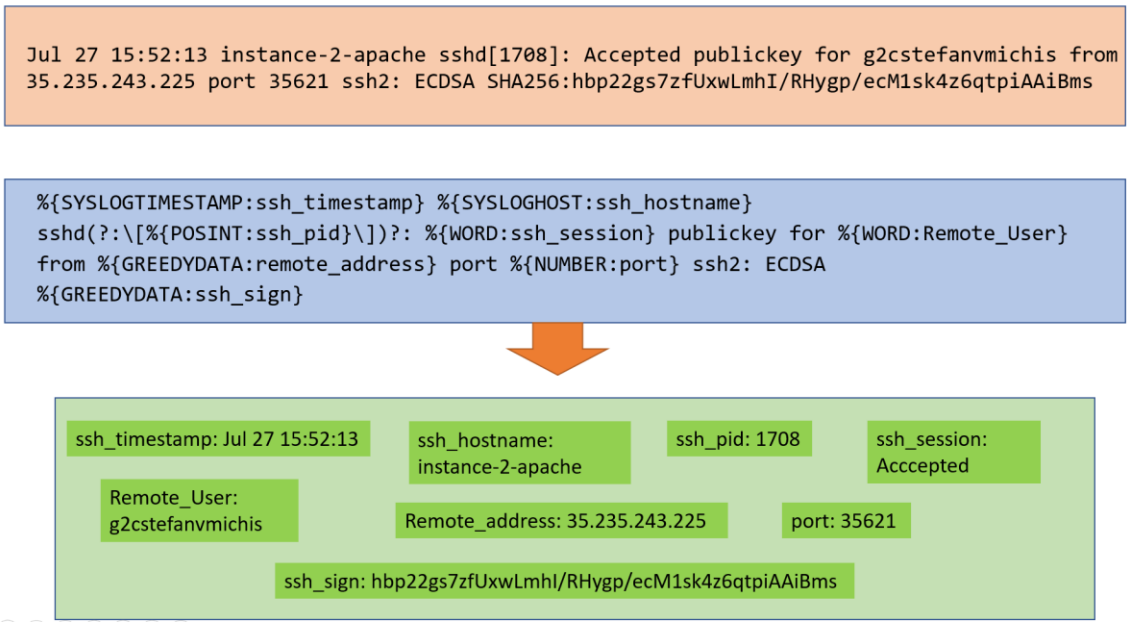


Figura 28: Esquema de conexiones SHH aceptadas con el uso del par de claves SSL/TLS en Google Cloud procesado por el filtro Grok

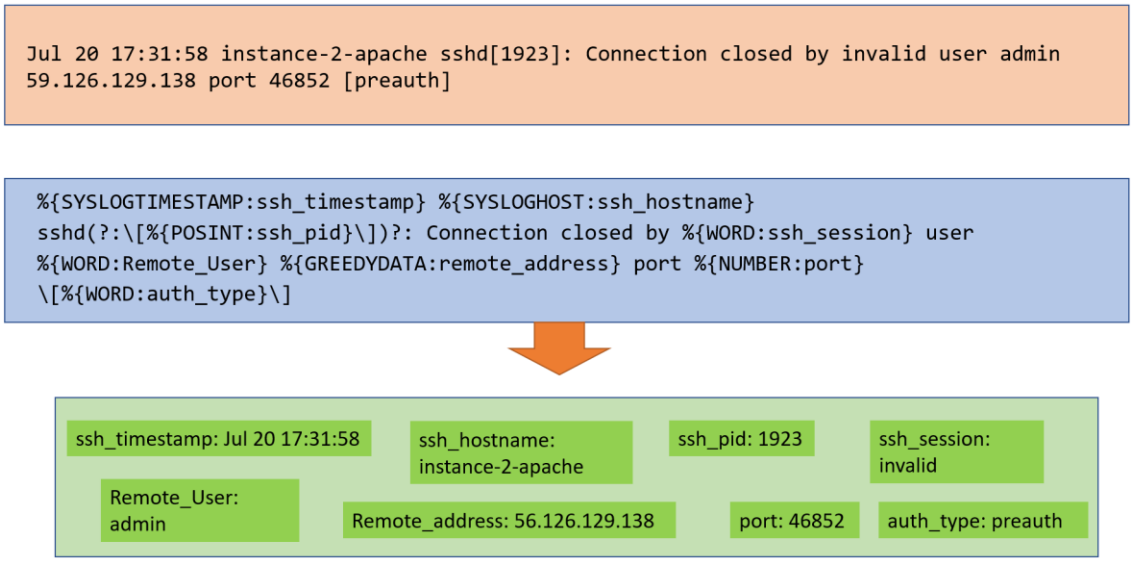


Figura 29: Esquema de conexiones SHH inválidas en Google Cloud procesado por el filtro Grok

8.3. Herramientas Nativas de Google Cloud para la Práctica de Observabilidad

Google Cloud dispone de sus propias herramientas para cualquier tipo de gestión de nuestras instancias, algunas de estas herramientas nativas son relevantes respecto la observabilidad y cumple con la misma función que la pila ELK, estas son Google Cloud Operations Agent o Google Cloud Ops, Google Logging y Google Monitoring.

8.3.1. Google Cloud Operations Agent

En la figura 30 se puede ver como el agente de operaciones de Google recopila tanto logs, como métricas y las transforma el mismo para subirlo a la nube donde se almacenará y se mostrará gráficamente, es decir, el agente de Google hará las funciones de ambos Beats y de Logstash a la vez mientras que la nube hará la función de Elasticsearch y de Kibana haciendo uso de Google Logging y Google Monitoring. El agente, basado en FluentBit y OpenTelemetry, permite enviar logs y telemetría consumiendo los mínimos recursos posibles, optimizando su funcionamiento. Respecto los logs, el objetivo es el mismo, coger información textual y transformarla en JSON de manera que las herramientas de Google puedan almacenar esa información para poder obtener una búsqueda mucho más estructurada y rápida además de la posibilidad de generar gráficas y alertas a partir de tanto los logs como las métricas. Google Cloud Operations Agent añade por defecto muchos tipos de métricas tales como métricas de la propia máquina monitorizando, pero también métricas de bases de datos, métricas de procesos y métricas del propio agente [16]. Una vez procesados y enviados los logs, la nube de Google los almacenará y podrán ser mostrados desde Google Logging.

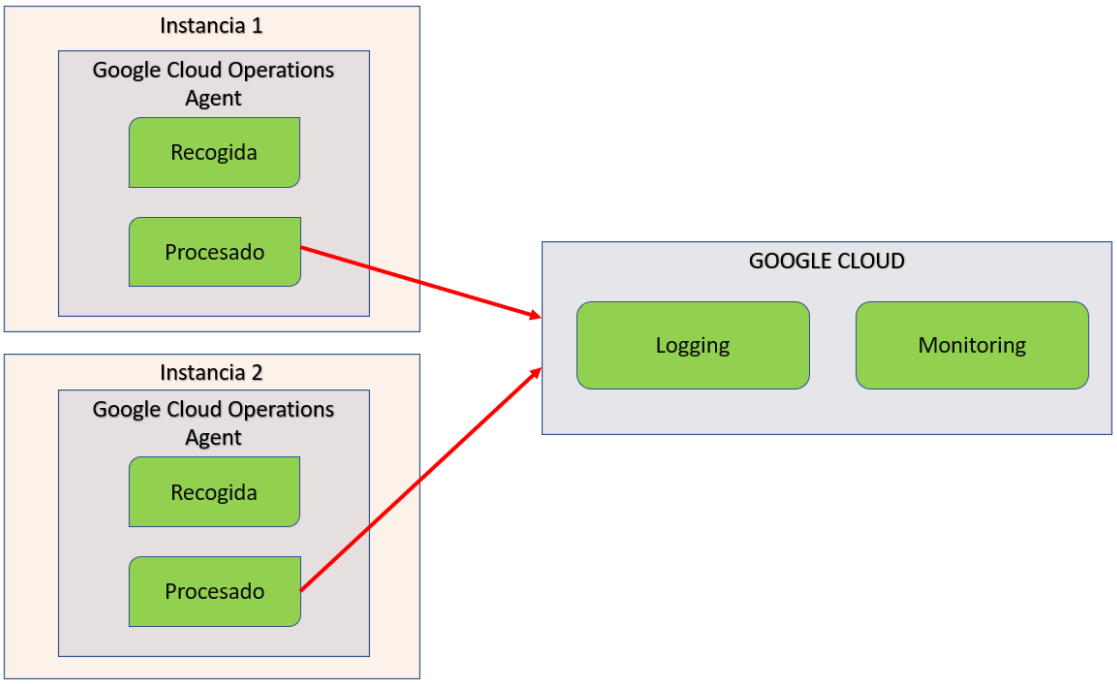


Figura 30: Esquema de funcionamiento de las herramientas de observabilidad de Google Cloud

Respecto a su configuración y funcionamiento, Google Cloud Ops tiene una metodología parecida a Logstash y a FluentD con el añadido que también se aplica a métricas. En el YAML de configuración se encuentran 2 divisiones, la división `logging` y la división `metrics`, en cada una habrá 3 campos: el campo `receivers` que recoge la información a procesar, el campo `processors` cuya función será procesar la información recibida y el campo `service` que define que información se enviará a la nube, es decir, en `service` se definirán que receptores y que procesadores se enviarán al pipeline. Este agente tiene configuraciones por defecto tanto para `logging` como para `metrics`, en `logging` se usarán las métricas por defecto y se añadirán métricas del propio agente para Apache mientras que en `logging` se usarán únicamente campos personalizados dejando de lado los receptores y procesadores por defecto [17].

8.3.1.1. Filtros con Expresiones Regulares

En la configuración de `logging` del agente de operaciones, específicamente en el campo Receivers se pondrán los directorios de los logs se quieran leer, pero en este caso, para el procesado de logs no será posible usar filtros Grok, por lo que se usarán expresiones regulares o regex para la transformación del log textual a un objeto JSON. Una expresión regular es una cadena de texto con la capacidad de comprender y modificar un lenguaje o una entrada de texto de manera genérica, es decir, un filtro regex tiene una filosofía muy parecida a Grok, si el filtro acepta la entrada, dicha entrada se separa por los campos

por defecto, generando un objeto JSON, dándole más significado a la información respecto a no tenerla procesada.

Al igual que en el capítulo 4.2.2, para explicar mejor cómo es el funcionamiento de los filtros regex se usará un ejemplo de funcionamiento, se supone el siguiente log (Figura 31):

```
Jul 27 15:55:13 instance-1 gpasswd[1598]: user stef added by root to group
google-sudoers
```

Figura 31: Log de ejemplo de syslog

Para un log de syslog genérico se escribiría el siguiente filtro (Figura 32):

```
^(?<time>\w* \d* [^ ]*) (?<client>[^ ]*) (?<program>[^ ]*):
(?<message>[^!?]*)$
```

Figura 32: Filtro regex para logs de syslog.

Para delimitar el filtro se usarán los caracteres ^ y $, cabe destacar que solamente el carácter ^ del principio tiene esa función de delimitador, los demás tienen la función de negación. Para separar el log y dividirlo en partes tal que este tenga un mayor significado, para ello se usará la siguiente expresión `(?<nombre_variable>valor)`.

Después de crear la variable habrá que darle un valor y un límite para que guarde únicamente los datos pertinentes. Para dicho límite o separador se pueden usar los espacios explícitos para separar la información o texto explícito también, de hecho, se puede apreciar que las variables van separadas por espacios y se puede ver que posteriormente a la variable `program` hay dos puntos explícitos, al igual que en el log para no añadirlos en la transformación.

El valor de estas variables se definirá con estas expresiones tal que:

- La expresión `\w` define un carácter del abecedario, bien sea en minúsculas o en mayúsculas.
- La expresión `\d` define un dígito, es decir un carácter del 0 al 9.
- La expresión `[^carácter/es]` indica que se almacene cualquier carácter que no esté en la lista precedida por ^, por ejemplo `[^ ]` recoge cualquier carácter que no sea un espacio o `[^?]` recoge cualquier carácter excepto el carácter de cierre de interrogación.
- El carácter "\" precedido de una letra no reservada indicará la aparición de un carácter explícito aunque el uso de la contra barra no es estrictamente necesario,

los caracteres explícitos se pueden poner directamente sin estar precedidos por ella.

- Finalmente la expresión * indica recursión, es decir, se recogerá la expresión previa tantas veces hasta que aparezca un carácter que no cumpla con esa expresión, por ejemplo, si existe una expresión que sea "hola que tal" y un filtro que sea `^(?<palabra>\w*)$,` entonces el valor de palabra será equivalente a hola puesto que en "hola que tal" hay espacios de por medio y `\w` no recoge dichos espacios.

Sabiendo esto, en el filtro de muestra junto al log de muestra, la variable time tendría un conjunto de caracteres del abecedario, lo cual sería, en este caso, equivalente a una palabra, junto a un espacio en banco, junto a un conjunto de números de una cifra, junto a otro espacio en blanco y se añadiría finalmente un conjunto de caracteres que no sean espacios en blanco (Figura 33).

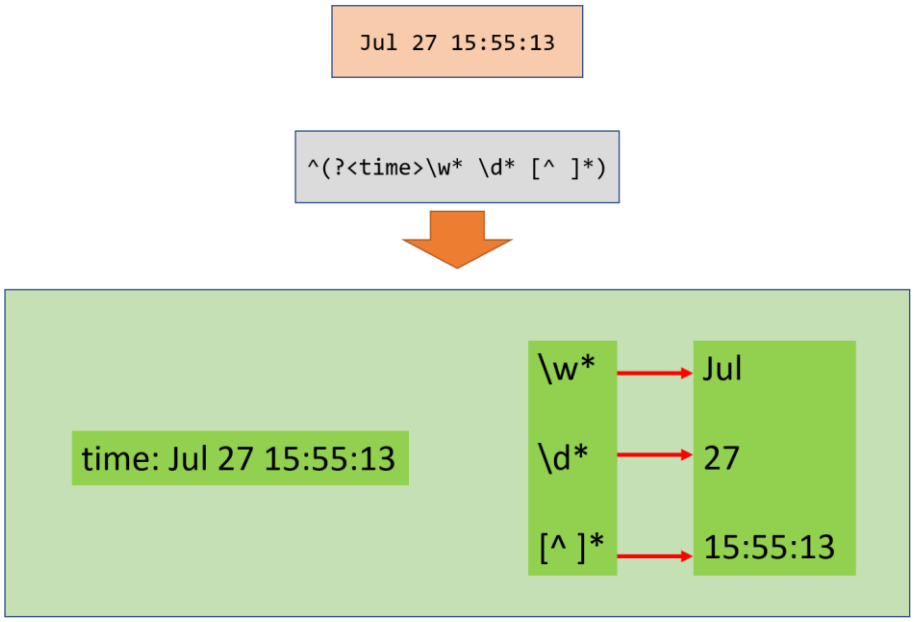


Figura 33: Creación de la variable time.

Finalmente, el filtro procesaría el log de la siguiente forma (Tabla 2):

time	Jul 27 15:55:13
client	Instance-1
program	gpasswd[1598]
message	user stef added by root to group google-sudoers

Tabla 2: Esquema de log transformado a JSON por el filtro regex.

Un factor importante que cabe destacar en la configuración de `processors` es que cada filtro regex que se añada deberá tener un nombre y dicho nombre tendrá que aparecer en el pipeline para su envío, esto permite poder separar los logs procesados por su nombre, facilitando las tareas de filtrado (Figura 66).

8.3.2. Google Cloud Logging

Google Cloud Logging es una herramienta que permite almacenar y analizar los datos recibidos del agente de operaciones, es decir, hace la función de base de datos temporal y de motor de búsqueda analítica, al igual que Elasticsearch. Los datos recibidos serán administrados y centralizados y podrán ser accedidos por una API que proporciona Google. Desde esta misma herramienta también se pueden observar los logs desde una interfaz web que tiene por nombre Explorador de Registros. Desde el explorador se pueden hacer las propias búsquedas con un lenguaje propio, se pueden generar vistas personalizadas, se pueden descargar los logs y se pueden crear alertas y métricas relevantes a las búsquedas personalizadas [18].

Figura 34: Vista principal del Explorador de Registros.

En la figura 34 se puede ver la vista inicial que tiene el explorador, este tiene un buscador, una gráfica con un conteo de los registros según el paso del tiempo, un panel lateral con donde se puedan buscar elementos en concreto y el panel principal mostrando los logs. La estructura no difiere mucho de Kibana, que es la alternativa con la que estamos comparado esta herramienta, no obstante, se pueden apreciar varias diferencias. Primero, el formato de muestra de los logs, el nombre de la instancia que genera el log está en primer lugar al lado del timestamp marcado en azul. En la propia figura se pueden presenciar logs propios que genera Google Cloud y posteriormente 2 tipos de logs, se puede ver el tipo más común que muestra un mensaje posteriormente al nombre de la máquina y otro que muestra un objeto JSON después del nombre. Ambos logs están

procesados y, de hecho, el primer tipo de logs tiene el mensaje incompleto si lo que se desea es mostrar el mensaje original, la principal diferencia reside en que el explorador muestra como mensaje principal cualquier campo del objeto JSON recibido llamado "message". El primer tipo de logs mencionado, es decir, el que no muestra el campo JSON ha sido procesado por el filtro regex mencionado en el capítulo anterior (Figura 34), el cual genera un campo llamado "message". Si este campo no existe Google Monitoring muestra el objeto al completo. Esta problemática no existe la pila ELK pues Logstash mientras procesa los logs mantiene el mensaje original también y en Kibana la visibilidad de los mensajes muestra dicho mensaje original por defecto.

Tener los datos mostrados de esta forma dificulta mucho su lectura y procesado por lo que existe una opción en el explorador para mostrar aquellos campos que interesen en la ventana que aparezca, para posteriormente guardar esa ventana y poder usar la búsqueda guardada en cualquier momento. Para ello existe una opción llamada "EDITAR" que añade los llamados campos de resumen a la vista, destacándolos (Figura 35).

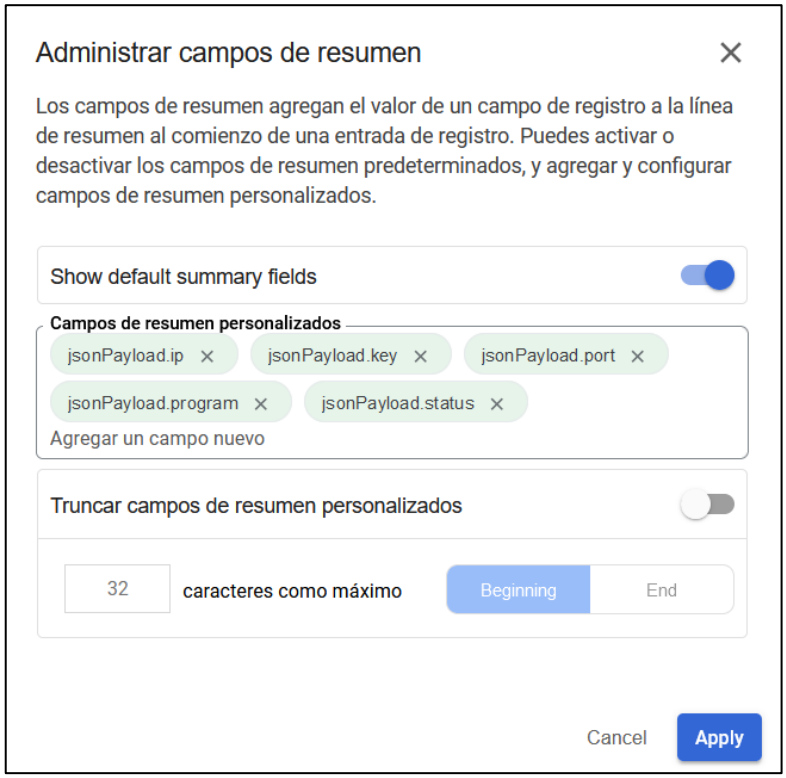


Figura 35: Menú para el añadido de campos en los mensajes del Explorador de Registros

Desde el filtro se puede ver que los campos que van a destacar indican una IP, una clave, un puerto, y un estado (haciendo referencia al estado de la conexión cuando esta sea inválida). Se puede ver que, este filtro coge todos los campos del filtrado de las conexiones SSH que se verá en el capítulo 8.3.6 además del campo `program` del filtro para los mensajes de syslog, esto se debe a que se está usando la vista principal, por lo que hay mensajes de syslog, conexiones SSH y demás. De estos mensajes de syslog

genéricos se cree que únicamente es conveniente destacar cual es el programa que genera el log para poder hacer una búsqueda en el caso que sea necesario. Una vez aplicado el filtro, la vista quedaría de la siguiente forma (Figura 36):

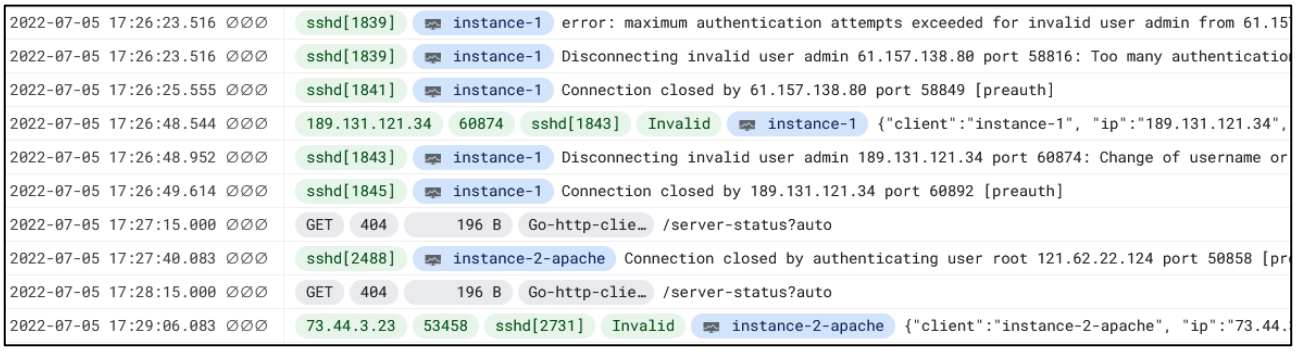


Figura 36: Vista principal de los logs en el Explorador de Métricas una vez aplicado el filtro.

Respecto a las búsquedas que se pueden hacer en el Explorador de Métricas de Google Cloud, se pueden diferenciar 2 elementos principales al igual que en Kibana, la búsqueda de valores de los elementos de los objetos JSON y la búsqueda por el tiempo transcurrido, es decir, está la posibilidad de hacer una búsqueda de un filtrado por el campo IP de manera que solamente aparezcan IP's que no sean de confianza en un transcurso de 10 días. También, para filtrar búsquedas para casos de uso concretos se pueden usar condicionantes como un campo que solamente aparezca si se da una petición al servidor Apache, esta metodología también se ha usado en Kibana. La potencia que tiene este buscador es equivalente a la del par Kibana-Elasticsearch, no obstante, el buscador que tiene Kibana es mucho más intuitivo y usable debido a su autocompletado.

Se puede crear una consulta tal que se busque un campo que exista en todos los logs que generan las máquinas, que puede ser genérico y no tener ningún significado. También interesa que no aparezcan logs que genere Google Cloud Ops, se indica en la consulta del buscador la inversa de filtrar por un par campo-valor que solamente aparezca en los logs del agente de operaciones (Figura 37). En cuanto a los filtrados de los casos de uso, como cada caso de uso tendrá su filtro regex y este nombre aparecerá en Google Monitoring como una de las fuentes del mensaje, se hará un filtrado con dicho nombre.

```
resource.type="gce_instance"
-logName:"ops-agent-fluent-bit"
```

Figura 37: Consulta de búsqueda para el filtrado por todos los logs que genere las instancias excepto los logs de Google Cloud Ops

8.3.3. Google Cloud Monitoring

Google dispone de una herramienta que permite monitorizar todo tipo de métricas, desde métricas por defecto de las máquinas hasta métricas personalizadas de servicios u aplicaciones además de ofrecer la posibilidad de creación de métricas que se deseen con los datos recibidor por parte del agente de operaciones. También es posible a partir de dichas métricas generar alertas para evitar una monitorización constante.

En Google Monitoring existe un menú general donde se muestra qué es lo que ofrece Monitoring, aquí se muestra como las herramientas de Google tienen la disponibilidad de usarse junto a Prometheus, que es una herramienta de monitorización muy potente. En el menú Paneles o Dashboards se muestran todos los paneles existentes, por defecto, hay varios paneles generales que muestran información general de las instancias con información general de sus recursos además de listados con información de las máquinas tales como la capacidad de sus discos o las reglas de firewall existentes (Figura 38).

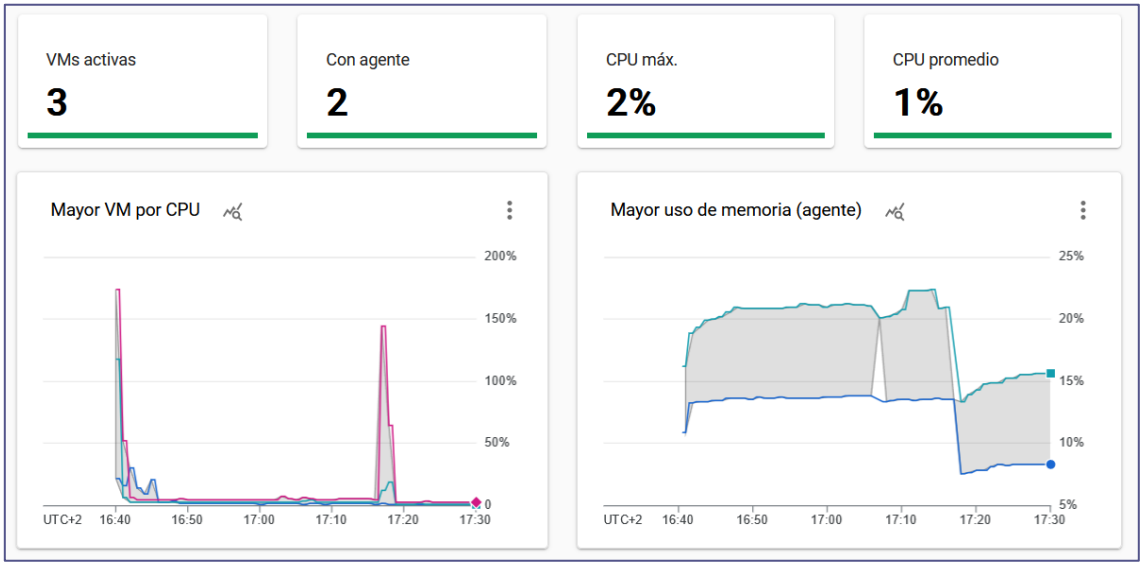


Figura 38: Graficas parciales de las instancias del panel por defecto.

Para crear métricas personalizadas basadas en registros se deberá acceder o bien a una vista personalizada de Google Logging y usar la opción "Crear Métrica" o directamente acceder al menú "Métricas basadas en registros" de Google Logging también. Desde este menú se podrá definir el nombre, unidades, tipo (contador o métrica de distribución), el filtro que seguirá, que por defecto es el filtro de la vista personalizada, en la figura 39 se ha accedido al menú desde la vista por defecto, que no tiene ningún filtro. Cabe destacar que se podrán crear etiquetas, las etiquetas son aquellos campos por los que se vaya a agrupar, lo común en estos casos es poner etiquetas que sean equivalentes a los campos que hayan sido transformados si interesa [19].

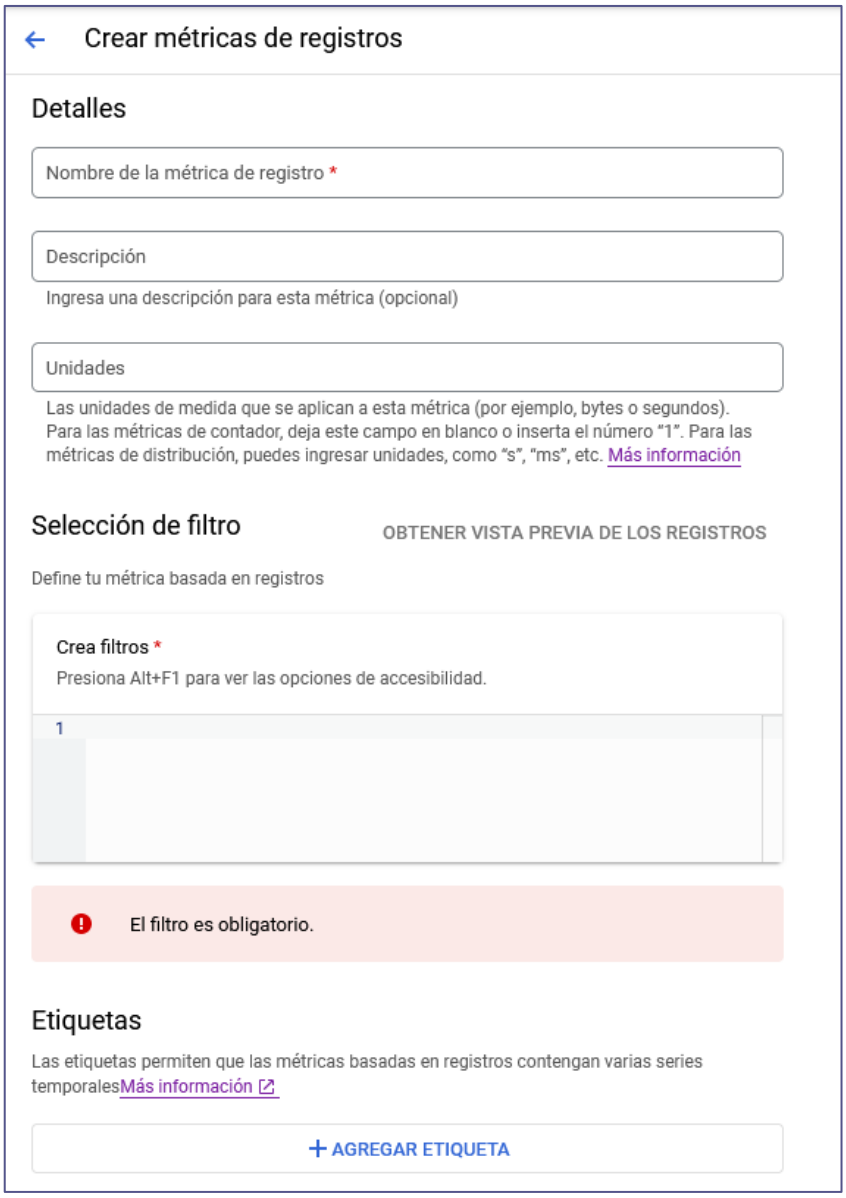


Figura 39: Submenú de creación de métricas.

En lo que respecta graficar las métricas que existan, se deberá acceder a un menú llamado Explorador de Métricas, desde dicho menú se pueden seleccionar varios casos, primeramente, se deberá seleccionar cuál es la métrica por monitorizar, para ello se podrá desplegar un menú que ofrece todos los datos que se puedan usar para crear métricas (Figura 40). Se pueden seleccionar varias métricas y estas se pueden agrupar por las etiquetas que tenga dicha métrica, por ejemplo, la métrica de consumo de CPU tiene 3 campos de agrupación, la instancia de donde se obtiene la información, el proyecto del que forma parte o la zona en la que está. Un factor importante por destacar es que no se puede modificar el eje x, que supondrá el paso del tiempo, mientras que en Kibana esto sí que es perfectamente posible y hasta aconsejable para hacer una mejor práctica de la observabilidad.

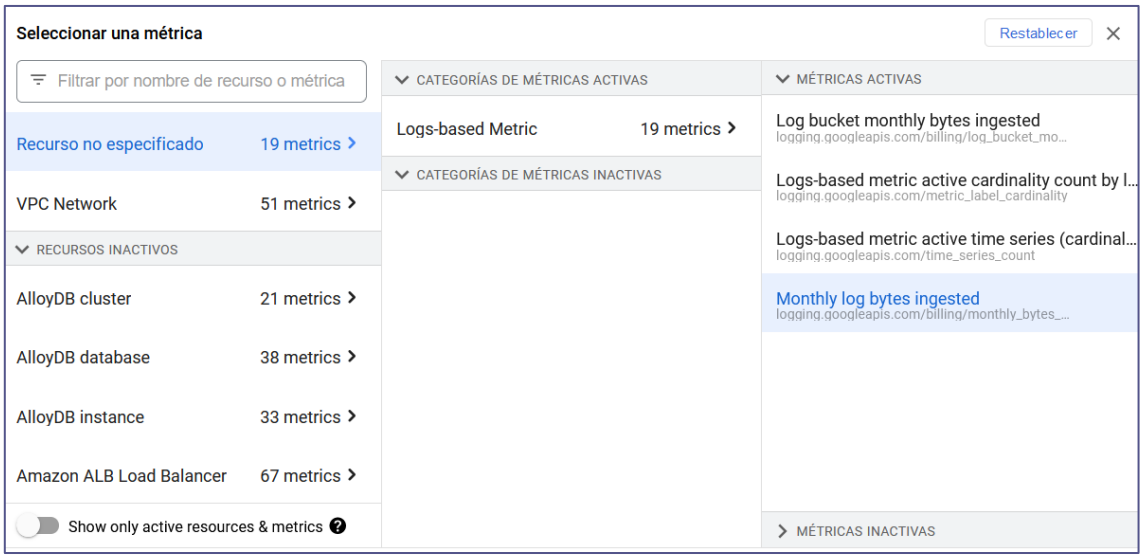


Figura 40: Menú de selección del registro a monitorizar.

8.3.4. Casos de Uso con Google Cloud Ops

En lo que respecta los casos de uso con las herramientas nativas de Google, la conclusión que se obtiene es la misma, es decir, la idea reside en ahorrar tiempo para encontrar una problemática respecto no usar las herramientas, encontrar una conexión SSH sospechosa o una petición al servidor Apache no permitida sin tener las herramientas presentadas en este proyecto, supone una tarea muy tediosa sobre todo en un clúster grande. La diferencia, pues, reside en cómo llegar a esa conclusión. En el caso de la pila FEK, al usar las mismas herramientas para el uso y el almacenamiento de la información, no era necesario mencionar dichos casos de uso, es por ello por lo que se mostrará cómo se practican los casos de uso con Google Cloud Logging, Google Cloud Monitoring y Google Cloud Operations Agent.

8.3.4.1. Conexiones SSH

Ya se mencionó en el capítulo 8.2.1 que las conexiones SSH en el entorno de la nube no se trataban igual que las conexiones en local, en la nube se tratan de conexiones válidas o inválidas debido al uso de certificados SSL/TLS, es decir, al crear un cuenta de Google Cloud, no se proporciona una contraseña para el acceso a las máquinas, no obstante, sí que existe un certificado para la conexión desde la consola de Google Cloud además de la posibilidad de crear usuarios aunque sigue siendo necesario tener un certificado aunque se haga una conexión con los usuarios. Es por ello por lo que se deberán definir los siguientes filtros regex para las conexiones SSH según sean válidas o no (Figura 41) (Figura 42):

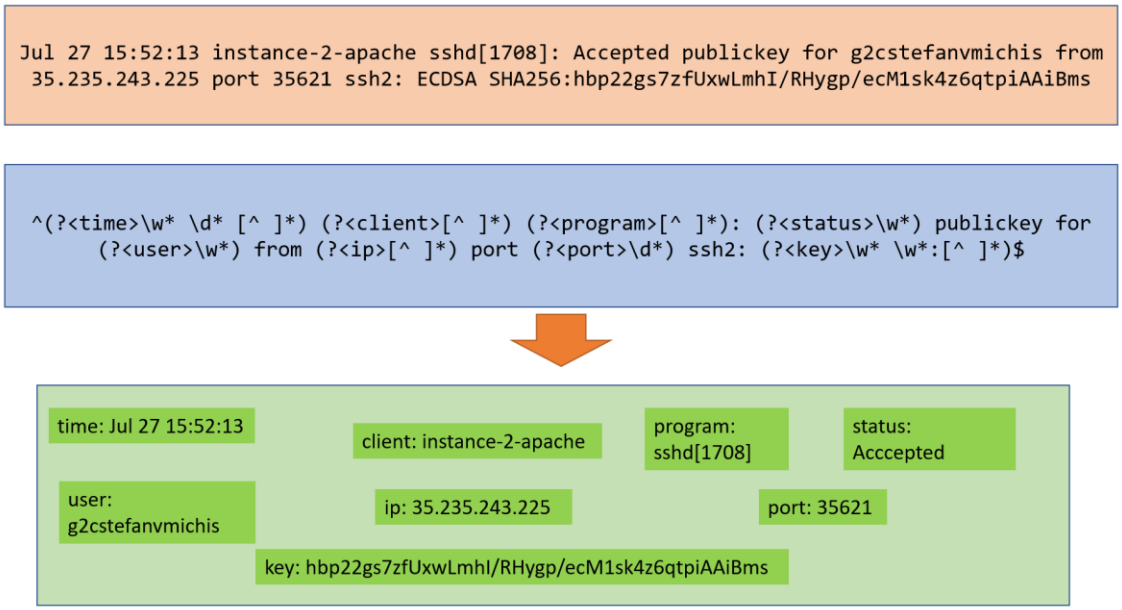


Figura 41: Esquema de conexiones SHH aceptadas con el uso del par de claves SSL/TLS en Google Cloud procesado por el filtro regex.

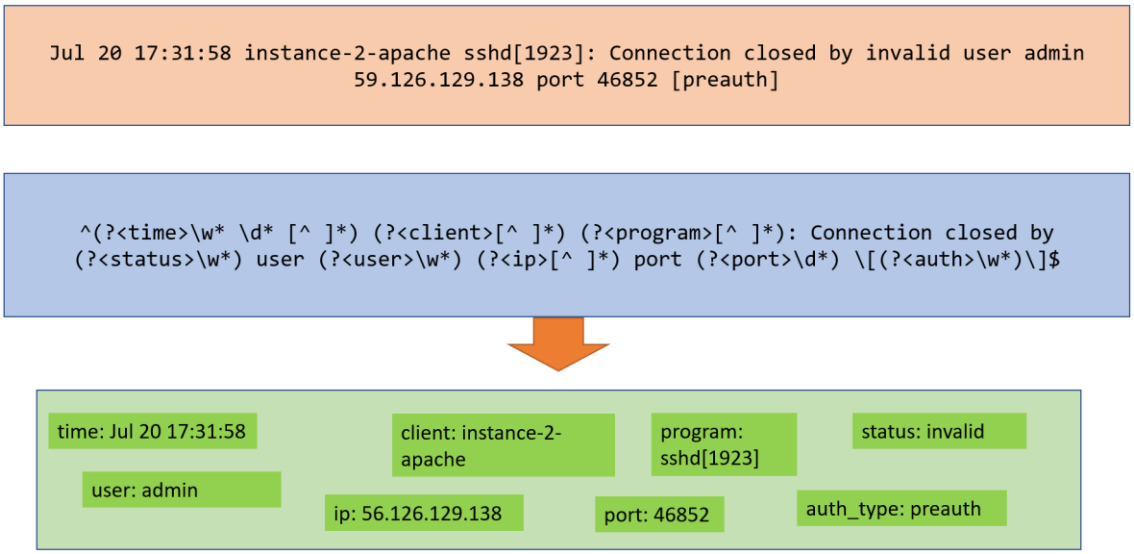


Figura 42: Esquema de conexiones SHH inválidas en Google Cloud procesado por el filtro regex.

Cabe destacar que existe la posibilidad de poner ambas situaciones en un único filtro, no obstante, se considera mucho más sencillo e igual de rápido configurar y usar 2 filtros distintos.

El uso que se le va a dar a Google Logging y Monitoring será el siguiente: en Logging se pueden definir las búsquedas que interesen según los datos recibidos y su formato, es por ello que se definirá, como se hace en Kibana, vistas para comprobar el estado de las conexiones según sean válidas o no, además, como se dispone del motor de búsqueda [20] y de un campo en el objeto JSON donde se indica la IP de origen, es una posibilidad

filtrar para obtener conexiones con IPs que no sean de confianza en una búsqueda (Figura 43). Se puede presenciar que el campo `status` estará solamente en este tipo de mensajes por lo que se puede filtrar por la existencia de dicho campo.

```
jsonPayload.status:*
NOT jsonPayload.ip:1.1.1.1
```

Figura 43: Entrada de búsqueda para la obtención de la vista de conexiones SSH excluyendo la IP 1.1.1.1.

En Google Monitoring se creará una gráfica en base a las conexiones válidas e inválidas usando como etiqueta la variable `status` según el paso del tiempo muy parecida a la que se usa en Kibana (Figura 44).

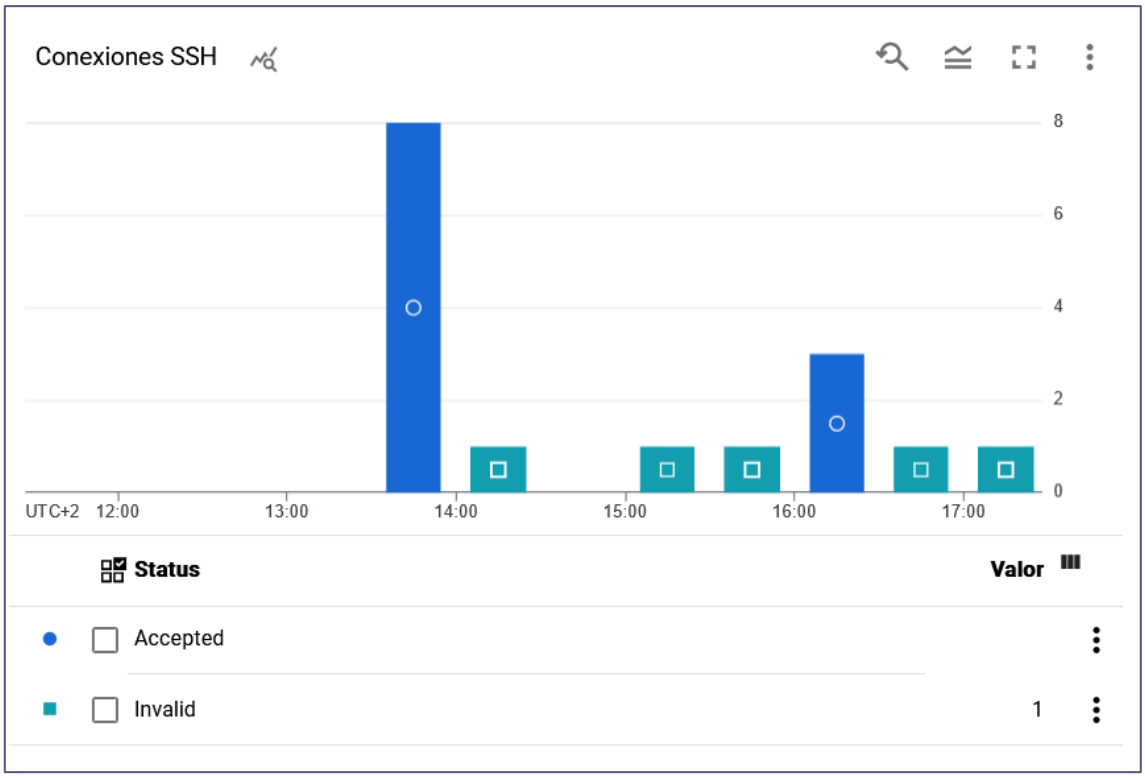


Figura 44: Gráfica de las conexiones SSH en Google Monitoring

8.3.4.2. Conexiones al Servidor Apache

En este caso de uso hay más variaciones respecto al caso de uso en local, al igual que en los filtros Grok había un filtro por defecto que recogía esa información, el agente de

operaciones de Google tiene una configuración completa por defecto para los logs del servidor Apache, donde se recoge el procesado de logs y de métricas para este caso de uso se utilizará la configuración genérica únicamente para los logs. [21]. Es por ello que no será necesario crear un filtro regex específico para las conexiones al servidor Apache, de hecho, no hará falta exponer gran parte de los campos que haya procesados tales como el método de la conexión o el código de respuesta en el explorador de registros, como sí sucede en los demás casos. Este hecho se puede presenciar observando que los cuadros filtrados, que aparecen en verde, en esta misma figura también aparece de por medio un log de acceso al servidor Apache donde dichos cuadros aparecen en gris, dando a saber que son campos creados automáticamente por la herramienta, es decir, en los mensajes normales aparecerá el campo `message` en formato textual y campos genéricos definidos por filtros genéricos en gris (Figura 45). Se puede observar, no obstante que hay un campo procesado por dicho filtro genérico que aparece en verde, esto se debe a que no aparece por defecto en el menú, por lo que se debe añadir explícitamente como se hace en la figura 35.

Figura 45: Vista filtrada por conexiones al servidor Apache del explorador de registros.

Para poder crear la vista personalizada se hará uso de la siguiente búsqueda en el explorador de registros de manera que recoja los logs de apache procesados en el agente de operaciones además de quitar los logs donde la IP de origen sea "::1" que es la IPv6 local para poder hacer un mejor filtrado de las conexiones al servidor (Figura 46). El filtrado de los errores se hará con una búsqueda explícita por el valor de la variable `program` indicando que contenga la palabra httpd por el que pasa dicho log.

```
logName="projects/radiant-planet-353011/logs/apache_access"
httpRequest.remoteIp!="::1"
```

Figura 46: Entrada de búsqueda para la obtención de la vista de conexiones al servidor Apache excluyendo la IP local.

En lo que respecta la gráfica de registros, no es posible hacer una gráfica de tipo donut que muestre un porcentaje de las respuestas que se dan por lo que se hará una gráfica que muestre las respuestas según el paso del tiempo (Figura 47):

Figura 47: Gráfica de conexiones al servidor Apache de Google Monitoring

8.3.5. Google Alerting

El concepto de alerta surge ante la imposibilidad de una monitorización constante por parte de los administradores de manera que se pueden generar espacios temporales donde no se practica una monitorización del entorno a controlar o se puede dar la situación de tener un entorno lo suficientemente grande como para no poder ser monitorizado por el equipo de administradores. Las políticas de alertas permiten generar un seguimiento constante de un elemento de hardware o de un software en específico para tener una rápida identificación del problema que haya surgido y generar una alerta que se pueda transmitir a través de un canal de monitorización.

Se hará uso únicamente del sistema de alertas de Google Cloud puesto que está disponible dentro del crédito gratuito de uso que se proporciona, en Kibana existe una función para generar alertas, no obstante, es de pago y en el proyecto se usa la licencia gratuita del programa por lo que no tiene cabida.

Google Alerting permite generar políticas de alertas que puedan identificar problemas respecto a las métricas existentes, tanto personalizadas como no y también permite ser configurado para identificar logs que describan un problema dado haciendo uso de dichas políticas [22].

Desde los exploradores de métricas y de registros está la posibilidad de crear una política de alerta de manera que saldrá una ventana emergente donde se deben indicar los datos necesarios para dicha alerta. En el caso que genere dicha alerta, se creará una incidencia indicando más detalles de esta además de recuadros de texto donde se puede indicar su solución para poder evitar que vuelva a ocurrir o para obtener una mayor cantidad de información precisa respecto al contexto del clúster, cabe recordar que la práctica de observabilidad puede indicar la existencia de un problema y qué es lo que lo ha causado pero la solución del problema la tiene que ofrecer el administrador.

Para este proyecto se hará uso de una alerta creada a partir de logs recibidos, en concreto, se hará una alerta que indique cuando se ha aceptado una conexión SSH desde una IP que no sea de confianza.

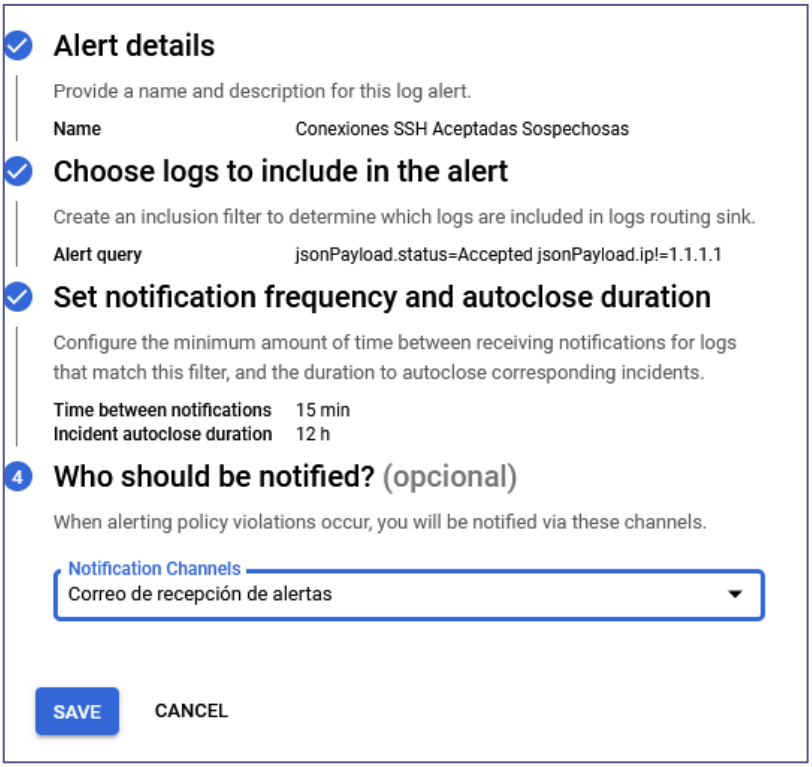


Figura 48: Submenú de creación de alertas

En los datos del segundo título de la figura 48 se ve como es necesario insertar la consulta que permite filtrar los logs recibidos y poder generar la alerta en el momento que llegue una conexión aceptada que no sea de la IP 1.1.1.1. También es necesario insertar un título a la alerta y una descripción. Posteriormente se indicará la frecuencia de revisión de logs para crear alertas y un tiempo de resolución automática de la incidencia que se genere a partir de la alerta. Finalmente se añadirá un canal de comunicación de la alerta, de manera que, en el momento que surja la alerta se envíe una notificación. Se puede notificar por correo electrónico, por un canal de Slack, por SMS y demás canales. Para esta alerta en concreto se hará uso de un correo electrónico.

Una vez configurada la alerta se pondrá a prueba haciendo una conexión SSH a cualquiera de las 2 instancias que se estén monitorizando. Aunque se acceda desde la consola de Google Cloud, el acceso será a través de una conexión SSH usando una IP pública que será diferente de 1.1.1.1 y en el momento que se acepte la conexión habrá llegado una alerta al correo electrónico definido (Figura 49). A su vez se habrá creado una incidencia de la alerta, a dicha incidencia se podrá acceder desde el correo o desde la propia consola de Google Cloud.

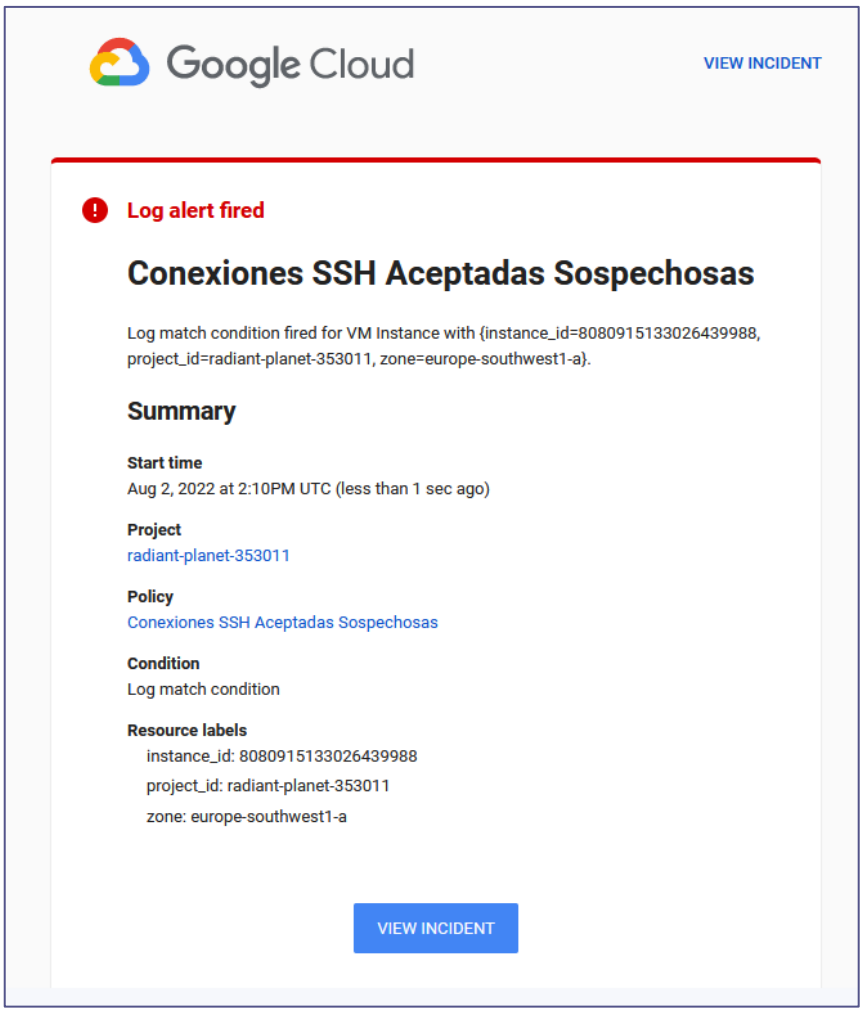


Figura 49: Mensaje de correo electrónico por la generación de una alerta.

En la incidencia se podrá ver el título y la descripción o documentación de la alerta junto a un listado de los logs que cumplan con la condición, observando los logs, se puede obtener información como cuál es la IP con la que se ha hecho la conexión o con qué usuario (Figura 50).

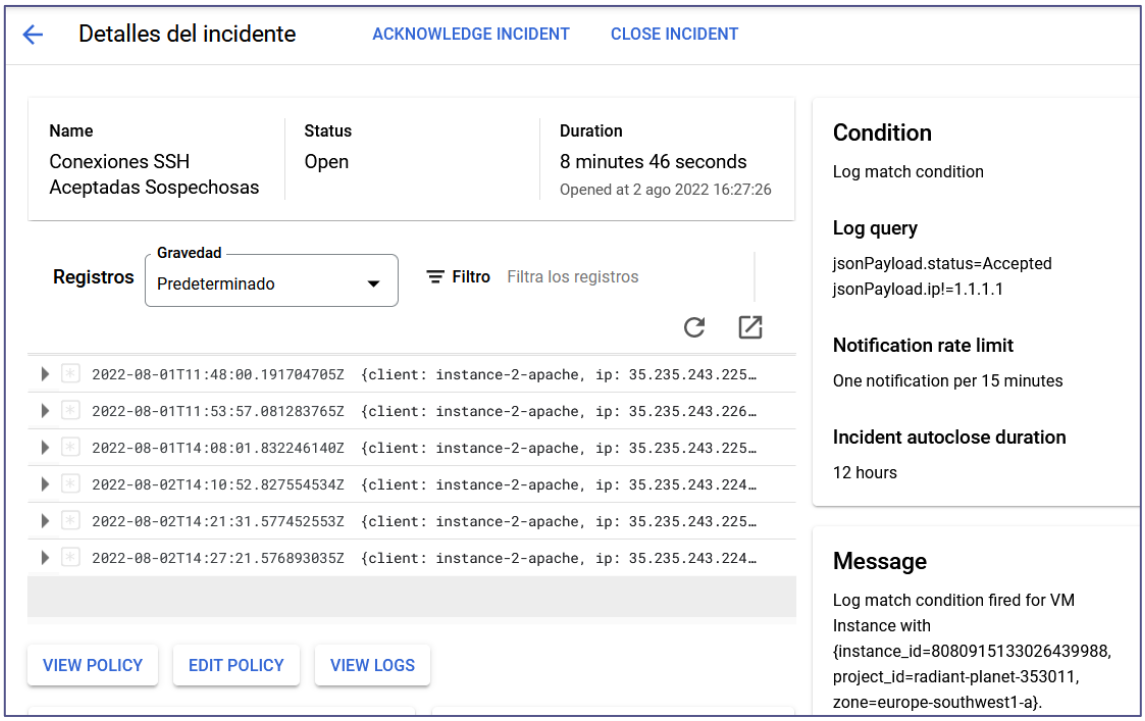


Figura 50: Interfaz de la incidencia generada por una conexión SSH aceptada proveniente de una IP sospechosa.

El uso de Google Alerting puede ser muy interesante debido a que hay contextos y casos de uso donde las alertas casan muy bien con los registros que puedan existir, como se da en el caso anterior, donde únicamente con ese registro se puede crear una política de alertas útil. Volviendo al primer caso de uso de las conexiones SSH, si se pide al administrador que monitorice diariamente dichas conexiones para encontrar sospechosos, junto a una gestión centralizada de logs y un sistema de alertas no habrá ninguna necesidad de estar controlando ese contexto pues cuando surja un problema, habrá un correo informativo y en la incidencia se mostrarán los logs que saltaron la alerta evitando su búsqueda explícita, de esta manera se puede obtener un sospechoso sin hacer ninguna búsqueda en el explorador de registros y mucho menos buscando los logs en la máquina cliente.

Es necesario mencionar que se debe hacer un buen uso del canal de comunicación de las alertas y de las alertas en sí, es una mala praxis inundar un correo de alertas que no sean relevantes o que sean muy repetitivas, de ahí la definición de un intervalo temporal hasta que vuelva a salir la alerta. Los canales de información deben estar bien preparados para evitar confusiones y para poder identificar problemas rápida y correctamente, es por ello por lo que se deben definir políticas de alertas en sistemas críticos y que sean lo suficientemente relevantes como para que sea justificado generar las alertas.

8.3.6. Costos de Uso de Google Cloud

Una vez se han observado los usos que se pueden dar tanto a la Pila ELK como a Google Cloud Ops y la comparación entre estos 2, es relevante mencionar cuales han sido los costos del crédito recibido por parte del Google durante el desarrollo de esta parte del proyecto. El desglose se dividirá principalmente en 2 partes: el uso de las instancias junto con las herramientas pertinentes y el costo de las licencias que se usen.

El primer costo, y el más obvio, es el costo de uso por hora de las máquinas, en el momento de la configuración sale un costo estimado mensual en el caso de que se mantengan encendidas por un mes seguido, este costo, no obstante, no es realista debido a que no es el caso, por lo que después de 40 días aproximados de uso se ha usado la sección de facturación para poder comprobar cuál ha sido el gasto (Figura 51).

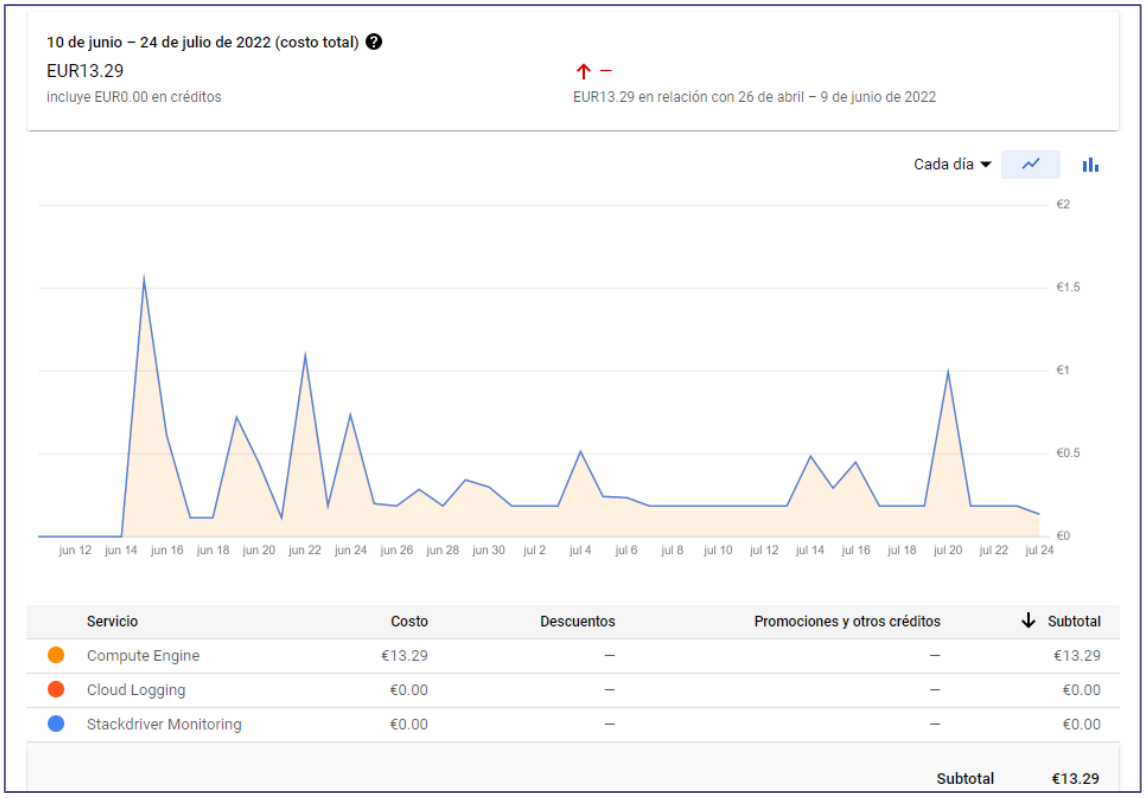


Figura 51: Gráfica de costos de uso desde el 10 de junio al 24 de Julio

Se puede ver en la figura anterior cómo el costo solamente está en Compute Engine, que es la herramienta de creación, uso y gestión de las instancias que se han usado. Mientras que el costo de las herramientas de Logging y Monitoring es de 0. Esto se debe a que hay una cuantía inicial de datos, que, si no se supera, no hay cobro. En el caso de Logging esta cuantía es de 50GiB mientras que en el caso de Monitoring, solamente se cobrará por métricas facturables siempre y cuando se supere la cuantía de 150 MiB [23].

El otro caso de costos hace referencia a las licencias, en este caso, solo hay gastos por una única licencia, la del uso Red Hat Enterprise Linux en una máquina virtual con 6 o más CPU'S virtuales, por cada vez que se inicie el servidor ELK habrá un gasto y por cada hora de uso este aumentará, de los 13,29€ que muestra la gráfica anterior, 2.16 son del uso de la licencia (Figura 52).

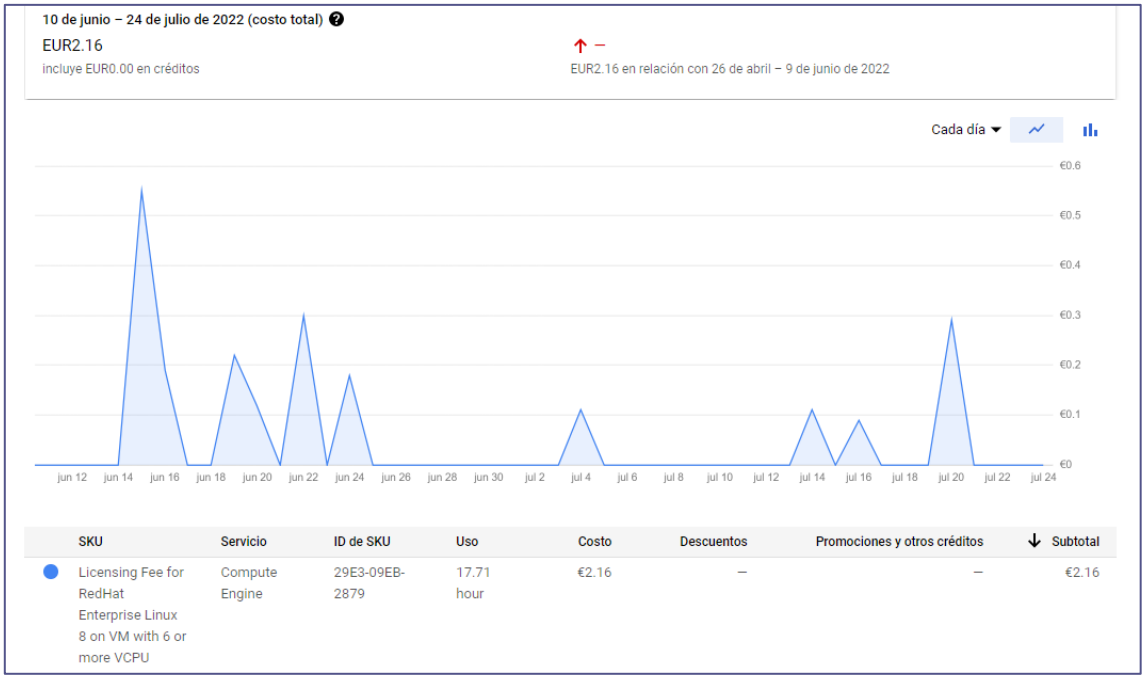


Figura 52: Gráfica de costos de la licencia de RHEL 8.

Se puede deducir, pues, que para un uso como el del proyecto, el costo se puede ignorar con totalidad y aunque no se diera el caso del uso del crédito ofrecido por Google, se trataría de un costo totalmente asumible.

8.4. Comparativa de la Pila ELK con las Herramientas Nativas de Google Cloud

Después de hacer uso de las dos pilas de herramientas se pueden deducir varios hechos:

- En lo que respecta a las métricas y las gráficas, Kibana tiene un mejor funcionamiento en todos los aspectos del uso de este proyecto, es más fácil y flexible de usar. Crear una gráfica de exactamente lo que se desea en Kibana se consigue en minutos mientras que en Google Monitoring, además de ser mucho menos flexible, es menos usable para el administrador, haciendo que se tarde más tiempo en la creación de métricas y gráficas e incluso sean de menor calidad.
- En cuanto a Google Logging, está más a la par con Kibana, no obstante, la búsqueda de elementos para obtener las vistas personalizadas es más rápida en Kibana por el autocompletado en las búsquedas además de no tener que ser escritas como una consulta, es decir, en cuanto a la gestión de los logs, Kibana una vez más es más usable que Google Logging.
- A la hora del procesado de logs, hacer uso de filtros regex es más óptimo respecto a los filtros Grok, no obstante, Logstash es más sencillo de configurar que Google Cloud Ops, se debe tener en cuenta que el agente de operaciones hace la función de 2 elementos de la Pila ELK con un consumo pequeño en las instancias por lo que generalmente es un programa muy ventajoso.

Se puede observar que, haciendo una aproximación, la pila ELK tiene herramientas más útiles y potentes que las herramientas de Google teniendo en cuenta además que se usa la versión gratuita de la pila, sin embargo, Google Cloud ofrece muchas más herramientas de administración de todo tipo mientras que la pila ELK está más centrada en la gestión centralizada de logs, también se debe tener en cuenta que las herramientas de Google Cloud solamente se pueden usar en su entorno de la nube mientras que la pila ELK se puede usar en cualquier entorno, es por ello que se ha probado su funcionamiento tanto en local como la nube. Se concluye pues que la pila ELK es más potente que las herramientas de Google Cloud en lo que respecta el objetivo de poder practicar la observabilidad sobre la gestión de logs.

9. Conclusiones

Se puede concluir que, a través del desarrollo de este proyecto se han cumplido los objetivos principales que tenía, es decir, se ha definido la necesidad de practicar la observabilidad en entornos empresariales en contra de practicar la limitada opción de únicamente monitorizar o de no practicar ninguna metodología. Este hecho se puede presenciar en varios aspectos siendo el más importante para el proyecto el usar logs para poder transformarlos en información esquematizada y poder practicar los análisis necesarios con dichos registros procesados. Se ha conseguido demostrar la ardua tarea que sería intentar hacer un análisis de los logs para encontrar problemas o hacer un control del entorno sin las herramientas necesarias que permitan enviar, transformar, almacenar, mostrar y graficar los logs.

Para dicha demostración se han construido satisfactoriamente los entornos necesarios para hacer práctica de los casos de uso pertinentes. Se ha construido un clúster local y se han comparado 2 pilas de herramientas de observabilidad, la pila ELK y la pila FEK con la conclusión que para el desarrollo del proyecto es más efectivo hacer uso de la pila ELK debido a que tiene un enfoque más específico para el entorno de la gestión centralizada de logs. Posteriormente se propuso replicar el clúster en la nube de Google junto a la pila ELK para después compararla con las herramientas nativas de Google, de dicha comparación se obtuvo que las herramientas de Google Cloud cumplen con el propósito de practicar la observabilidad, pero son mucho más limitantes que las herramientas de la pila ELK, sobre todo en el aspecto de la interfaz, donde Kibana hace un aprovechamiento mucho mejor de la información recibida.

¿Es entonces la pila ELK superior ante la pila FEK y las herramientas nativas de Google Cloud? En este entorno, que pretende ser independiente y donde los datos van únicamente a un único punto sí que lo es, las herramientas de la pila ELK están pensadas para trabajar entre sí, de hecho, todas las herramientas usadas para la práctica de la observabilidad en la pila ELK son de la misma compañía, Elastic. Es por ello por lo que tienen un mejor funcionamiento que otra pila donde se use una herramienta que no forma parte de la empresa como podría ser la pila FEK. Una pila que tenga FluentD sería más funcional que una pila con Logstash en el caso donde se tengan que enviar los logs procesados a varios servidores, donde cada servidor tenga tecnologías distintas. Respecto a las herramientas de Google Cloud, son interesantes usarlas en algunos contextos, sobre todo en el uso de alertas. Se debe tener en cuenta que Google Cloud es un entorno de infraestructura en la nube cuyo principal servicio es ofrecer dicha infraestructura, se ofrecen herramientas para su control y gestión, pero al ser un entorno tan grande con tantas herramientas disponibles, es lógico pensar que las herramientas de observabilidad no sean tan profundas como sí lo son en la pila ELK, cuyo servicio principal sí que es referente a la observabilidad.

La observabilidad abarca muchos conceptos y muchos campos a varios niveles, en este trabajo ha habido mucho énfasis en los logs y un enfoque menor en las métricas ignorando la trazabilidad, pero en un entorno donde las métricas tengan prioridad tendría más sentido usar una herramienta como Grafana antes que Kibana. Un buen administrador deberá conocer bien el contexto del entorno para saber qué aplicaciones

funcionarán en este, pero, independientemente del contexto, siempre será mucho más práctico usar un conjunto de las herramientas disponibles, sean open source o no, que no usar ninguna.

9.1. Posibles Trabajos Futuros

A partir del desarrollo del proyecto se han observado los posibles trabajos futuros relacionados con el tema de la gestión centralizada de logs:

- Gestión centralizada de logs en un clúster Kubernetes: Trata el mismo concepto que este proyecto, pero a gran escala pues al tratarse de un clúster Kubernetes hay una cantidad alta de contenedores donde cada uno tiene una cantidad alta de pods y cada pod tiene la capacidad de generar logs. Es por este mismo motivo que para este trabajo se debería hacer un mayor énfasis en la trazabilidad para hacer un seguimiento de los pods en caso de un posible problema.
- Gestión centralizada de logs en AWS: Es interesante tener en cuenta otros entornos y profundizar más en estos, Amazon Web Services o AWS es el entorno en nube más usado del mundo, no se usó en el proyecto actual debido a que se denegó el crédito gratuito que sí que ofreció Google Cloud.
- Una posible ampliación de este proyecto supondría dotar de seguridad a la comunicación entre los clientes y el servidor, de manera que no pueda ocurrir un man-in-the-middle durante dicha comunicación demostrando la identidad de los clientes
- Otra posible ampliación sería dotar de alta disponibilidad al clúster con las herramientas de observabilidad replicando el servidor que ofrezca dicho servicio, permitiendo un funcionamiento constante en caso de que el nodo maestro caiga.

9.2. Relación con los Estudios Cursados

Este trabajo no ha podido ser posible si no fuera por las bases que existían previamente dadas durante el Grado de Ingeniería Informática, en concreto, el curso de las siguientes asignaturas fue crucial para el desarrollo de este proyecto:

- “Integración de Aplicaciones” debido a que se explicó en esta asignatura la importancia de formatear todos los datos recibidos a un formato más entendible para las máquinas además de lo útil que es el formato JSON.
- “Administración de Sistemas” puesto que la raíz de esta asignatura es la importancia que reside en configurar correctamente y controlar los sistemas a monitorizar. Es debido a esta asignatura que yo, el autor, tengo la motivación de querer ser un buen administrador.

- “Redes Corporativas” debido a que en esta asignatura se trata la comunicación entre máquinas además de los protocolos que se usan. En esta asignatura se hacen trabajos “en formato TFG” los cuales fueron muy útiles para el desarrollo de este proyecto.

10. Anexos

10.1. Instalación y Configuración

En este anexo se detallará con más precisión cómo han sido instaladas y configuradas las herramientas pertinentes.

10.1.1. Instalación y Configuración de los Elementos de la Pila ELK

Antes de hacer la instalación explícita se deberá tener preparado el repositorio pertinente para poder hacer dicha instalación, para ello, se importará una clave pública que permitirá hacer uso de un repositorio, para el cual se creará un archivo llamado `elasticsearch.repo` en el directorio `/etc/yum.repos.d`, en otros sistemas operativos el directorio y el gestor de paquetes varía pero el procedimiento es muy parecido. Una vez se haya configurado el repositorio pertinente según el sistema operativo, se procederá con la instalación (Figura 53). [24] [25] [26].

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

------------------------------------------------------------
## Se crea un archivo llamado elasticsearch.repo en /etc/yum.repos.d y se
añade lo siguiente:

[elasticsearch]
name=Elasticsearch repository for 8.x packages
baseurl=https://artifacts.elastic.co/packages/8.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=0
autorefresh=1
type=rpm-md

sudo yum install --enablerepo=elasticsearch elasticsearch, kibana, logstash
```

Figura 53: Instrucciones y configuración necesarias para la instalación de la pila ELK

Finalmente, en el servidor se instalará un servidor proxy NginX, para el cual se deberán instalar unas utilidades previas y posteriormente se deberá crear el archivo pertinente para importar el repositorio necesario (Figura 54) [27].

```
sudo yum install yum-utils

------------------------------------------------------------
## Se crea un archivo llamado nginx.repo en /etc/yum.repos.d y se añade lo
siguiente:

[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true

------------------------------------------------------------
sudo yum install nginx
```

Figura 54: Instrucciones y configuración necesarias para la instalación de NginX.

En cuanto a los Beats que se tengan que instalar en los clientes, se deberá repetir el proceso de importación del repositorio y posteriormente instalar Filebeat y Metricbeat (Figura 55).

```
## Se debe usar el gestor de paquetes que corresponda

sudo apt-get/yum install filebeat
sudo apt-get/yum install metricbeat
```

Figura 55: Instrucciones necesarias para la instalación de los Beats.

En la configuración de Elasticsearch se pueden poner parámetros como el nombre del clúster o el nombre del nodo que está ejecutando la Pila, no obstante el único parámetro que interesa modificar es `network.host` y cambiarlo a `0.0.0.0` para que escuche en

todas las interfaces del servidor. En la configuración de Elasticsearch hay un variable llamada `xpack.security` que está habilitada por defecto, esto supone que para poder interactuar con Elasticsearch será necesario crear un usuario (Figura 56), en este caso se creará un usuario con permisos totales para una interacción más directa con el sistema [28].

```
./elasticsearch-users useradd KibanaAdmin -p ad00min-r superuser
```

Figura 56: Comando de creación del usuario KibanaAdmin con máximos permisos en Elasticsearch.

La primera vez que se arranque Kibana y se acceda a la interfaz web aparecerá una ventana con un pequeño tutorial para conectar Kibana con Elasticsearch en el cual habrá que ejecutar un script de Elasticsearch que genere un código y ponerlo en la interfaz web (Figura 57).

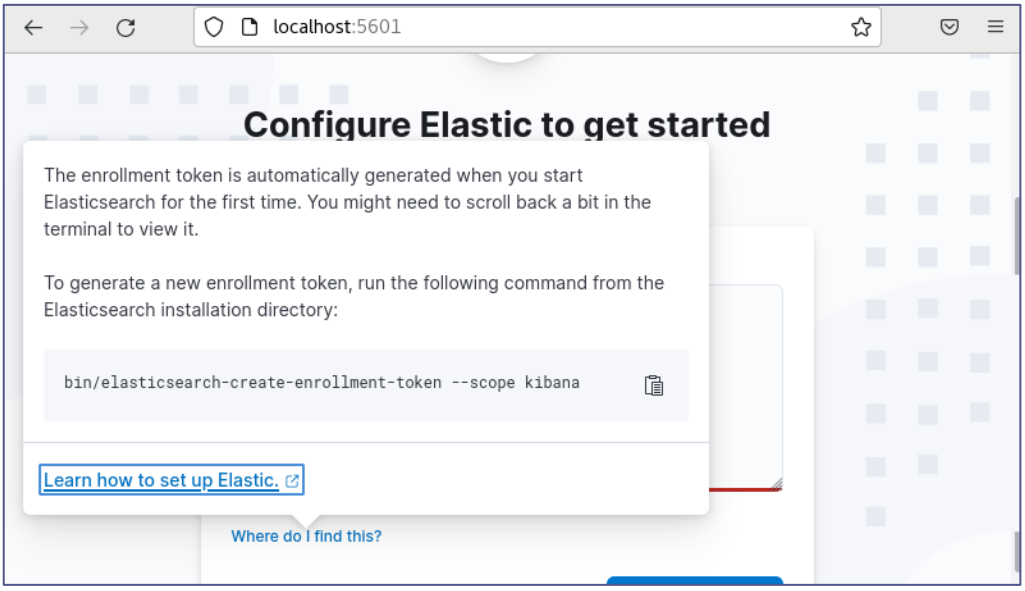


Figura 57: Configuración Inicial de Kibana.

Respecto a Logstash, primeramente, en `input` se configurará la entrada de datos para que reciba datos específicamente de los Beats por el puerto 5044, los logs se procesarán en el campo `filter` tal que para cada caso de uso se abra un campo Grok y en la sección `match` se ponga el filtro Grok y demás posibles datos como añadido de información. En la configuración para el envío de los logs procesados, es decir, `output`, se definirá la dirección y el puerto de Elasticsearch, el nombre del índice que se creará y, si ya está creado, se enviará la información a dicho índice. El índice tendrá el nombre y la versión del beat más la fecha de cuando se ejecute. También se indicarán un usuario y contraseña

creados previamente para poder acceder a Elasticsearch. La línea `stdout { codec => rubydebug }` indica que está habilitada la depuración de errores (Figura 58) [29].

```
input {
  beats {
    port => 5044
  }
}
filter {
  grok{
    match => {"message" => /FILTRO GROK 1/ }
      }
 grok{
    match => {"message" => /FILTRO GROK 2/ }
      }
    …
}
output {
  elasticsearch {
    hosts => ["http://IP_DE_ELASTICSEARCH:9200"]
    index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
    user => "kibanaAdmin"
    password => "ad00min"
  }
  stdout { codec => rubydebug }
}
```

Figura 58: Configuración del pipeline principal de Logstash.

Para finalizar la configuración del servidor se configurará NginX para que redirija las peticiones del puerto 5061 al puerto 80, facilitando el uso de Kibana (Figura 59).

```
server {
   listen 80;
      server_name server.tfg.com;
      location / {
        proxy_pass http://IP_DE_KIBANA:5601;
        }
}
```

Figura 59: Configuración de NginX

Respecto a Filebeat, en su archivo de configuración habrá que indicar de que archivos tendrá que leer la información y cuál es el destino, en este caso, Logstash. Se debe recordar que para cualquier archivo que se indique en la configuración, Filebeat deberá tener permisos de lectura, si no, no se podrán enviar los logs (Figura 60). La configuración de Metricbeat es más sencilla pues se deberá indicar únicamente el destino de la información, el cual es Elasticsearch (Figura 61).

```
…
filebeat.inputs:
   - type: filestream
    id: syslog
    paths:
           - /var/log/auth.log # archivo de logs de syslog en debian
# en centOS y RHEL el directorio de los logs de syslog es   /var/log/secure
…
output.logstash:
  # The Logstash hosts
  hosts: ["IP_DE_LOGSTASH:5044"]
…
```

Figura 60: Configuración a modificar de Filebeat.

```
output.elasticsearch:
  # The Elasticsearch hosts
  hosts: ["IP_DE_ELASTICSEARCH:9200"]
```

Figura 61: Configuración a modificar de Metricbeat.

10.1.2. Instalación y configuración de FluentD para la pila FEK

Para la instalación de FluentD se hará uso de la aplicación `td-agent` que es una versión estable de FluentD escrita en Ruby y en C adaptada para su instalación a través de repositorios, para la instalación será necesario descargar y ejecutar un script que actualizará los repositorios e instalará `td-agent` automáticamente (Figura 62) [30].

```
curl -L https://toolbelt.treasuredata.com/sh/install-redhat-td-agent4.sh |
sh
```

Figura 62: Comando de instalación de FluentD.

Previamente a la configuración de FluentD se deberá instalar un plugin que permita hacer uso de los filtros Grok (Figura 63) [31].

```
td-agent-gem install grok-parser
```

Figura 63: Instrucción de instalación del plugin para filtros Grok de FluentD.

La configuración de FluentD sigue la misma filosofía que la de Logstash, no obstante, sigue otra sintáxis. En el campo `source` se indicará que se reciben datos de tipo beats, en `parse` se podrán todos los filtros grok que haya y en `match` se enviarán los datos a Elasticsearch (Figura 64).

```
<source>
  @type beats
  tag other
  <parse>
    @type multiline_grok
      <grok>
        pattern /FILTRO_GROK_1/
        pattern /FILTRO_GROK_2/
      </grok>
  </parse>
</source>

<match **>
  @type elasticsearch
  port 9200
  user kibanaAdmin
  password ad00min
```

Figura 64: Configuración de FluentD

10.1.3. Instalación y Configuración del Agente de Operaciones de Google

Para la instalación de Google Cloud Ops Agent únicamente será necesario descargar y ejecutar un script en un directorio donde el usuario que ejecute dicho script tenga permisos de escritura, por ejemplo, su directorio home (Figura 65) [32].

```
curl -sSO https://dl.google.com/cloudagents/add-google-cloud-ops-agent-
repo.sh

sudo bash add-google-cloud-ops-agent-repo.sh --also-install
```

Figura 65: Comandos para la instalación del Agente de Operaciones.

A la hora de configurar el agente, se tendrá que indicar en receivers los datos indicando donde tiene que leer dicho agente, los campos genéricos de apache estarán separados del campo donde se indiquen que logs leer. Los procesadores mostrarán los filtros regex cada uno por separado y en service se asociará en cada pipeline, los receivers junto a sus processors. Se debe indicar que el `default_pipeline` está vacío para que no procese logs con los filtros por defecto que trae el agente (Figura 66) [19].

```
logging:
  receivers:
    apache_access:
      type: apache_access
    apache_error:
      type: apache_error
    syslog_data:
     type: files
     include_paths:
     - /var/log/auth.log #/var/log/secure en la instancia centOS

processors:
    accepted_ssh:
     type: parse_regex
     regex:”FILTRO_REGEX_1”
    invalid_ssh:
     type: parse_regex
     regex:”FILTRO_REGEX_2”
service:
    pipelines:
      default_pipeline:
        receivers: []
      custom_pipeline:
        receivers:
        - syslog_data
        processors:
        - accepted_ssh
        - invalid_ssh
      apache_pipeline:
        receivers:
        - apache_access
        - apache_error
```

Figura 66: Configuración del agente de operaciones de Google.

10.2. Objetivos de Desarrollo Sostenible

Los Objetivos de Desarrollo Sostenible se dan debido a la necesidad imperativa de proteger al planeta Tierra y a sus habitantes además de permitir a todos los habitantes las mismas posibilidades y los mismos derechos, estos objetivos tienen como fin acabar con las situaciones de desigualdad y con las situaciones que se dan en países menos desarrollados para 2030 [33].

A continuación, se indicará cuáles son los Objetivos de Desarrollo Sostenible que se pueden relacionar con el proyecto Gestión Centralizada de Logs en la siguiente tabla:

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.				X
ODS 4. Educación de calidad.		X		
ODS 5. Igualdad de género.				X
ODS 6. Agua limpia y saneamiento.				X
ODS 7. Energía asequible y no contaminante.				X
ODS 8. Trabajo decente y crecimiento económico.			X	
ODS 9. Industria, innovación e infraestructuras.	X			
ODS 10. Reducción de las desigualdades.				X
ODS 11. Ciudades y comunidades sostenibles.				X
ODS 12. Producción y consumo responsables.				X
ODS 13. Acción por el clima.			X	
ODS 14. Vida submarina.				X
ODS 15. Vida de ecosistemas terrestres.				X
ODS 16. Paz, justicia e instituciones sólidas.				X
ODS 17. Alianzas para lograr objetivos.				X

Tabla 3: Tabla de los Objetivos de Desarrollo Sostenible y su relación con el proyecto

Como se puede ver en la Tabla 3, son 4 los objetivos que se pueden relacionar con este proyecto:

- ODS4. Educación de calidad: Este proyecto se encuentra basado en el concepto de observabilidad, el cual no está muy desarrollado en España, pues tecnológicamente el país anda retrasado respecto Estados Unidos, que es de donde reside dicho concepto. La práctica de observabilidad supone la apertura a una nueva rama de la gestión de servidores y es por ello por lo que para una educación de calidad se debe tener en cuenta dicho concepto.

- ODS 8. Trabajo decente y crecimiento económico: Se puede relacionar este trabajo con el octavo objetivo ODS debido al ahorro de tiempo que supone hacer práctica de las herramientas mencionadas facilitando el trabajo y mejorando la economía como consecuencia directa a una mejor práctica laboral.

- ODS 9. Industria, innovación e infraestructuras: Este objetivo es una suma de la explicación de los 2 anteriores, el trabajo se basa en el concepto de observabilidad, el cual es innovador debido a los estándares actuales para la gestión de servidores, que se basan principalmente en la monitorización. Debido a dicho concepto innovador, el aplicarlo supondría innovar en la infraestructura haciendo uso de nuevo software y nuevas praxis que no se usaban antes. Además, el uso de infraestructuras en la nube supone una gran innovación respecto el uso de infraestructuras físicas en costes de creación y de uso de máquinas virtuales respecto máquinas físicas.

- ODS 13. Acción por el clima: Este Objetivo de Desarrollo Sostenible está indirectamente relacionado con el proyecto debido a que un ahorro de tiempo durante el uso implica una utilización óptima de los sistemas, lo que implica un menor tiempo de uso de dichos sistemas desembocando en un menor consumo eléctrico de estos, generando una huella de carbón menor. Si a el concepto anterior se añade que la infraestructura reside en la nube en vez de ser física, se reduce aún más la huella de carbono, en este caso, tanto directa como indirectamente. Esto se debe principalmente a 2 factores, el primero es que se evita la necesidad de la construcción de la infraestructura física lo que supone comprar servidores, los cuales al construirse generan una gran huella de carbono. El segundo factor refiere al consumo, el consumo de las máquinas virtuales el dinámico, solo se dará consumo energético mientras el servicio esté activo, en el momento que deje de estar habilitado el servicio, otro proyecto hará uso de dicha infraestructura, optimizando el consumo al máximo.

Se puede concluir que los Objetivos de Desarrollo Sostenible que se relacionan con el proyecto están centrados en mejorar la productividad y el conocimiento de la población además del pequeño extra del añadido del objetivo 13. Acción por el clima. Aunque los demás objetivos no procedan en la relación, cabe indicar lo importantes que son pues, al vivir en el llamado primer mundo, no se hace tanto énfasis en conceptos como la potabilidad del agua, la hambruna o la pobreza extrema, los cuales no se deben ignorar en absoluto.

11. Bibliografía

1. Majors, C., Fong-Jones, L., & Miranda, G. (2022). *Observability Engineering: Achieving Production Excellence*. (pp 18-31) O'Reilly Media.
2. *What is elasticsearch?* (s/f). Elastic.Co. Recuperado el 12 de abril de 2022, de https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html
3. *Data in: documents and indices.* (s/f). Elastic.Co. Recuperado el 12 de abril de 2022, de https://www.elastic.co/guide/en/elasticsearch/reference/current/documents-indices.html
4. Query DSL. (s/f). Elastic.Co. Recuperado el 12 de abril de 2022, de https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html
5. *Information out: search and analyze.* (s/f). Elastic.Co. Recuperado el 12 de abril de 2022, de https://www.elastic.co/guide/en/elasticsearch/reference/current/search-analyze.html
6. Noble, D. (2016). *Monitoring Elasticsearch*. (pp 1-8) Packt Publishing.
7. How logstash works. (s/f). Elastic.Co. Recuperado el 20 de abril de 2022, de https://www.elastic.co/guide/en/logstash/current/pipeline.html
8. *Kibana concepts*. (s/f). Elastic.Co. Recuperado el 22 de abril de 2022, de https://www.elastic.co/guide/en/kibana/current/kibana-concepts-analysts.html
9. *Beats*. (s/f). Elastic.Co. Recuperado el 10 de mayo de 2022, de https://www.elastic.co/es/beats/
10. Tamatura, K. (2014, 6 agosto). *Unified Logging Layer: Turning Data into Action | Fluentd*. Fluentd.org. https://www.fluentd.org/blog/unified-logging-layer
11. *Parser*. (s/f). Fluentd.org. Recuperado el 30 de mayo de 2022, de https://docs.fluentd.org/filter/parser
12. *Why Google Cloud*. (s/f). Google Cloud. Recuperado el 10 de junio de 2022, de https://cloud.google.com/why-google-cloud
13. *Google Cloud Free Program features*. (s/f). Google Cloud. Recuperado el 10 de junio de 2022, de https://cloud.google.com/free/docs/gcp-free-tier/
14. *CentOS Linux EOL*. (2021, 25 febrero). Centos.org. https://www.centos.org/centos-linux-eol/
15. McGovern, N. (2021, febrero 24). *Extending no-cost Red Hat Enterprise Linux to open source organizations*. Redhat.com. https://www.redhat.com/en/blog/extending-no-cost-red-hat-enterprise-linux-open-source-organizations
16. *Ops agent overview*. (s/f). Google Cloud. Recuperado el 30 de junio de 2022, de https://cloud.google.com/stackdriver/docs/solutions/agents/ops-agent
17. *Configure the Ops Agent*. (s/f). Google Cloud. Recuperado el 30 de junio de 2022, de https://cloud.google.com/stackdriver/docs/solutions/agents/ops-agent/configuration
18. *Using the logs explorer*. (s/f). Google Cloud. Recuperado el 10 de julio de 2022, de https://cloud.google.com/logging/docs/view/logs-explorer-interface
19. *Log-based metrics overview*. (s/f). Google Cloud. Recuperado el 10 de julio de 2022, de https://cloud.google.com/logging/docs/logs-based-metrics
20. *Logging query language. (s/f)*. Google Cloud. Recuperado el 10 de julio de 2022, de https://cloud.google.com/logging/docs/view/logging-query-language

21. *Apache web server (httpd)*. (s/f). Google Cloud. Recuperado el 10 de julio de 2022, de https://cloud.google.com/monitoring/agent/ops-agent/third-party/apache
22. *Introduction to alerting*. (s/f). Google Cloud. Recuperado el 23 de julio de 2022, de https://cloud.google.com/monitoring/alerts#types-of-policies
23. *Pricing*. (s/f). Google Cloud. Recuperado el 3 de agosto de 2022, de https://cloud.google.com/stackdriver/pricing#google-clouds-operations-suite-pricing
24. *Installing elasticsearch*. (s/f). Elastic.Co. Recuperado el 12 de agosto de 2022, de https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html
25. *Install kibana*. (s/f). Elastic.Co. Recuperado el 12 de agosto de 2022, de https://www.elastic.co/guide/en/kibana/current/install.html
26. *Installing logstash*. (s/f). Elastic.Co. Recuperado el 12 de agosto de 2022, de https://www.elastic.co/guide/en/logstash/current/installing-logstash.html
27. *Nginx: Linux packages - RHEL/CentOS*. (s/f). Nginx.org. Recuperado el 12 de agosto de 2022, de https://nginx.org/en/linux_packages.html#RHEL-CentOS
28. *Elasticsearch-users*. (s/f). Elastic.Co. Recuperado el 12 de agosto de 2022, de https://www.elastic.co/guide/en/elasticsearch/reference/current/users-command.html
29. *Structure of a pipeline*. (s/f). Elastic.Co. Recuperado el 12 de agosto de 2022, de https://www.elastic.co/guide/en/logstash/8.3/configuration-file-structure.html
30. *Install by RPM package (red hat Linux)*. (s/f). Fluentd.org. Recuperado el 12 de agosto de 2022, de https://docs.fluentd.org/installation/install-by-rpm
31. *Plugin Management*. (s/f). Fluentd.org. Recuperado el 12 de agosto de 2022, de https://docs.fluentd.org/deployment/plugin-management
32. *Installing the Cloud Logging agent on individual VMs*. (s/f). Google Cloud. Recuperado el 12 de agosto de 2022, de https://cloud.google.com/logging/docs/agent/logging/installation
33. *La Agenda para el Desarrollo Sostenible*. (s/f). Organización de las Naciones Unidas. Recuperado el 12 de agosto de 2022, de https://www.un.org/sustainabledevelopment/es/development-%20agenda/