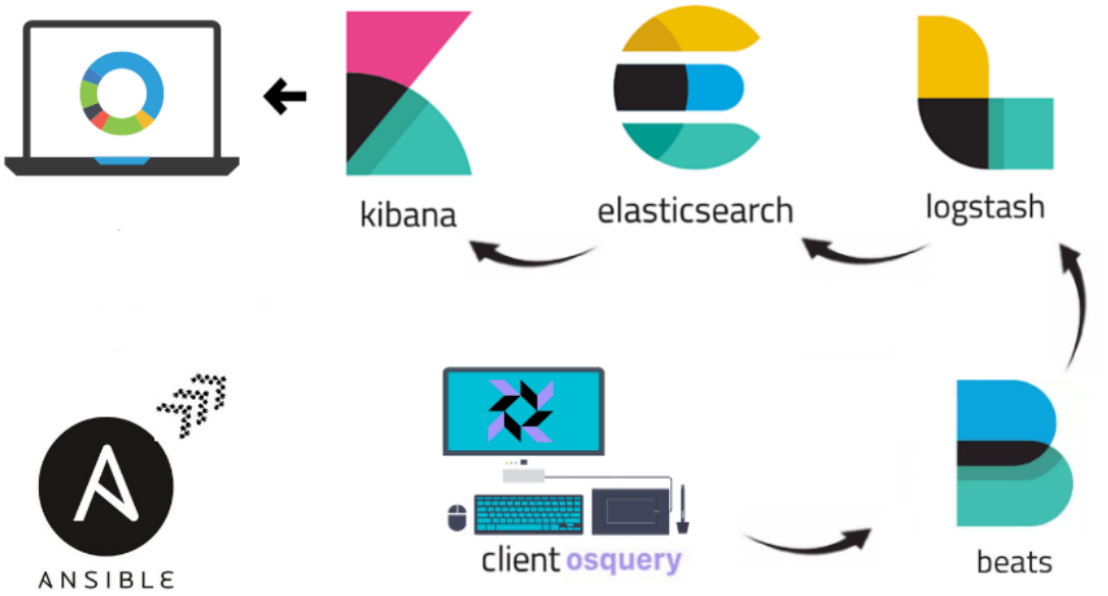


Elastic Stack + Osquery + Ansible

(Projecte d'investigació)
CFGS Administración de Sistemas Informáticos y Redes

Maria Madroñal Martínez & Aure Gonzales Jiménez
ASIX - Proyecto Final

RESUMEN

2020-21. “El gran año del salto tecnológico”, así ha sido clasificado debido a las expectativas que se han ido generando durante el transcurso de los años en base al desarrollo de todo tipo de tecnología, IT.

Si bien es verdad que este último año nos hemos encontrado en medio de una pandemia global lo cual ha dejado devastadores daños; es importante tener en cuenta la evolución que está teniendo el sector tecnológico, donde se avecinan grandes avances haciéndonos a la idea del deber de tener los pies en el suelo, estar firmes, fortaleciendo las áreas más débiles de esta en la transformación de la nueva era digital.

El mundo está cambiando a una velocidad sorprendente; pocos están preparados para afrontarlo y es en esta situación, en mitad de la incertidumbre del caos que está sacudiendo a la humanidad, hemos visto que la tecnología ha tomado relevancia e importancia siendo un “salvavidas” para todas aquellas organizaciones y negocios donde los elementos de IT han sido una necesidad básica.

Teniendo en cuenta esto, hemos decidido realizar un proyecto basado en lo aprendido a lo largo del grado, combinando Ansible, Elastic Stack (Kibana, ElasticSearch, Logstash) y Osquery.

Enfocado en la centralización y procesamiento de datos desde el lado del servidor el cual podrá ingresar datos de una multitud de fuentes/clientes de manera simultánea, transformando e interpretando estos para luego poder ser visualizados en gráficos y la administración de todas estas fuentes desde un servidor central.

Palabras clave
Ansible, Elastic Stack (ElasticSearch, Logstash, Kibana), Beats (Filebeat), Osquery.

ABSTRACT

2020-21. "The Great Year of technological Leap", this has been classified due to the expectations that have been generated over the years based on the development of all kinds of technology, IT.

While it is true that this past year we have found ourselves in the midst of a global pandemic which has left devastating damage; it is important to take into account the developments that the technology sector is having, where great advances loom making us to the idea of the duty to have our feet on the ground, to stand firm, strengthening the weakest areas of this in the transformation of the new digital age.

The world is changing at an astonishing rate; few are prepared to face it and it is in this situation, a middle of the uncertainty of the chaos that is shaking humanity, we have seen that technology has taken on relevance and importance by being a "lifeguard" for all those organizations and businesses where IT elements have been a basic necessity.

With this in mind, we have decided to carry out a project based on what we learned throughout the degree, combining Ansible, Elastic Stack (Kibana, ElasticSearch, Logstash) and Osquery.

Focused on centralization and server-side data processing which will be able to enter data from a multitude of sources/customers simultaneously, transforming and interpreting these so that they can then be viewed in charts and managing all these sources from a central server.

Keywords

Ansible, Elastic Stack (ElasticSearch, Logstash, Kibana), Beats (Filebeat), Osquery.

Índice

1. Introducción
   1.1 Contexto 2
   1.2 Justificación 2
   1.3 Objetivos 2
   1.4 Estrategia y planificación del proyecto 3
   1.5 Metodología de trabajo 3
   1.6 Estudio económico y presupuestario 3
2. Descripción del proyecto 4
   2.1 Análisis de requisitos [proyecto de desarrollo] 4
   2.1.1 Requisitos funcionales 4
   2.1.2 Requisitos no funcionales 4
   2.1.3 Previsión de tareas de investigación 5
   2.2 Tecnologías 5
   2.2.1 Comparativa de las tecnologías valoradas 5
   2.2.2 Tecnologías escogidas 6
   2.2.3 Tecnologías descartadas 8
   2.3 Estructura del proyecto 10
   2.4 Definición de las tareas [proyecto de investigación] 10
3. Anexos 11
4. Conclusiones 29
   4.1 Conclusiones generales del proyecto 29
   4.2 Consecución de los objetivos 29
   4.3 Valoración de la metodología y planificación 30
   4.4 Visión de futuro 30
5. Glosario 31
6. Bibliografía 32

1. Introducción

En un mundo que está cada vez más informatizado, las empresas necesitan un sistema informático y una red que funcione correctamente, con lo que la figura de un administrador de sistemas es imprescindible ya que es el responsable de la planificación e implementación de medidas de seguridad que impidan que elementos externos accedan, dañen y sustraigan información.

Es por esto que nuestro proyecto se basa en la monitorización, que implica supervisar todos los sistemas y comprobar su disponibilidad, rendimiento, respaldo, etc. Hoy en día existen muchas herramientas disponibles para este fin, tanto comerciales como gratuitas.

Nuestra idea es la de monitorizar y administrar distintos equipos de cualquier sistema operativo mediante el programa Osquery que es una herramienta de código abierto que sirve para escanear cualquier ordenador conectado a una determinada red, analizando y almacenando los detalles de esas conexiones para exponer cualquier sistema operativo como si fuese una base de datos. Se basa en realizar peticiones SQL para analizar comportamientos a bajo nivel en la red, encontrar rastros sospechosos y detectar también aplicaciones vulnerables instaladas en esos sistemas. Los datos serán interpretados en el terminal o bien mediante alguna interfaz gráfica.

Los datos serán obtenidos mediante la base de datos Osquery de las máquinas clientes, y mediante la suite Elastic podremos procesar, interpretar y mostrar los datos.

Estos datos serán mostrados a través de Kibana, donde podremos llevar un seguimiento en un entorno gráfico en diferentes tiempos como ejemplo, en tiempo real.

Y finalmente hemos decidido también implementar como servicio administrador de los clientes, al grandioso y poderoso Ansible.

1.1 Contexto

Actualmente en el progreso de los medios informáticos y la aplicación de estos como medios de defensa, donde el mundo se encuentra en una gran guerra en el ciberespacio a causa de diversidad de actores de sociedades internacionales por diversidad de motivos, es importante también realizar un seguimiento de nuestros servicios, servidores, máquinas en general donde la intención es tratar con estos datos en busca de errores, de anomalías o incluso de cualquier tipo de "hackeo".

Y aunque no seamos aún foco de ningún tipo de hacker que tengan intenciones maliciosas contra nuestra empresa, servicio web, clientes, etc…
Siempre será aconsejable llevar un seguimiento de todo lo que sucede en las máquinas.

Debido a que es mejor prevenir que curar.

1.2 Justificación

Esto nos ha dado la idea de desarrollar un proyecto donde se implementen diversos tipos de servicios/programas para realizar un seguimiento de las máquinas, donde estas sean administradas y monitorizadas en búsqueda de cualquier falla.
Lo que más nos llamó la atención para realizar este proyecto son:

- El gran potencial que tiene la herramienta Osquery.
- El potencial que nos aporta Filebeat, ElasticSearch, Kibana.
- La gran herramienta Ansible para administrar sistemas.
- Monitoreo personalizado dependiendo de nuestras necesidades.

1.3 Objetivos

1.3.1 Objetivo general

Monitoreo y administración de sistemas mediante aplicaciones permitiendo realizar un seguimiento constante de cada uno de los ordenadores de los cuales se crean gráficos con Kibana, interfaz web para observar con más detalle en búsqueda de errores, problemas, etc... dentro de estas, donde cada ordenador utilizará osquery + filebeat los cuales enviaran datos a un servidor (ElasticSearch + Kibana) que lo interpretará.

Estas máquinas estarán administradas por el servidor que contará con Ansible para la gestión, instalación y configuración de estos.

1.3.2 Objetivos específicos

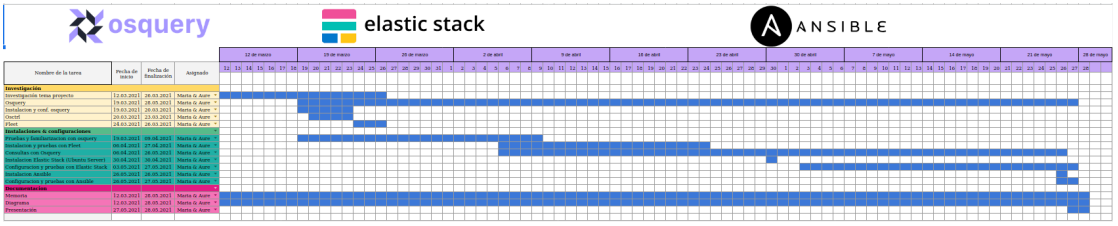
Centralizar todos los registros de los hosts/clientes y el poder administrar estos desde la misma máquina.

1.4 Estrategia y planificación del proyecto

La estrategia que hemos seguido se ha basado en utilizar nuevas tecnologías y otras ya conocidas. Con las nuevas tecnologías queríamos aprender una manera diferente de hacerlo e implementarlo con lo ya conocido para obtener los resultados deseados.

La planificación del proyecto la hemos realizado con un diagrama de Gantt donde hemos establecido tareas y fechas para llevar un seguimiento y tratar de conseguir los objetivos marcados.

Diagrama de Gantt:



1.5 Metodología de trabajo

El tipo de metodología que hemos utilizado es la de tipo Scrum. Es la que mejor se adapta ya que se basa en la flexibilidad en la adopción de cambios y nuevos requisitos, la colaboración e interacción y el desarrollo iterativo como forma de asegurar unos buenos resultados.

Además también hemos contado con la planificación del diagrama de Gantt que nos ha ayudado a seguir el desarrollo del proyecto de lo que teníamos planificado a lo que luego ha sido la realidad.

1.6 Estudio económico y presupuestario

En este proyecto no hemos tenido que adquirir materiales o componentes extras que nos haya hecho tener que realizar un presupuesto de costes.

Para llevarlo a cabo nos hemos basado en programas de software libre que nos han permitido poder desarrollar el proyecto sin problema.

2 Descripción del proyecto

2.1 Análisis de requisitos

Para el proyecto seleccionado pueda considerarse finalizado, debe cumplir con las siguientes condiciones:

- El servidor central debe de poder obtener los datos, procesarlos y mostrarlos a través de la suite Elastic.

- El servidor central debe de poder realizar una administración de forma automatizada de los clientes a través de Ansible.

- Los hosts/clientes deben de poder estar disponibles en la red.

2.1.1 Requisitos funcionales

- Se debe de poder centralizar los registros.

- Contar con el espacio necesario para albergar la cantidad de datos de todas las máquinas hosts/clientes.

- El servidor central debe de ser resistente a cualquier tipo de ataques.

- El servidor central debe de poder crear tipos de alertas para prevenir cualquier tipo de falla dentro de cualquier sistema administrado y monitorizado.

- Implementación de la administración de todos los hosts/clientes de manera automática, donde se pueda administrar todos estos dispositivos desde el servidor central.

2.1.2 Requisitos no funcionales

El proyecto deberá de cumplir los siguientes objetivos/requisitos no funcionales:

- Seguridad de todos los sistemas hosts/clientes.
- Seguridad de todos los datos/registros obtenidos de los hosts/clientes.

2.1.3 Previsión de tareas de investigación

Para la realización e implementación correcta de proyecto es necesario poder conocer previamente todos los servicios que serán utilizados durante y después de el proyecto:

- **Suite Elastic - Elastic Stack:** Entender el funcionamiento y configuración necesaria para su correcta implementación de todos los servicios que nos ofrece la suite.
  - **ElasticSearch**
  - **Logstash**
  - **Kibana**
  - **Filebeat**

- **Osquery:** Entender su funcionamiento y la manera en la que realiza los procesamientos, y la obtención de todos los datos.
  - **SQL:** Conocer lo básico de este lenguaje de base de datos para poder realizar cualquier tipo de consulta dentro del servicio Osquery.

- **Ansible:** Entender su funcionamiento, familiarizarse con la manera en la que realizará las tareas programadas.

2.2 Tecnologías

2.2.1 Comparativa de las tecnologías valoradas

Durante el desarrollo del proyecto se ha empezado a valorar y realizar una búsqueda de todos aquellos servicios que podrían habernos proporcionado lo necesario para poder realizarlo.

Entre estas opciones que hemos podido escoger han sido varias las que nos han puesto contra la pared: (Hemos escogido **ELK+Ansible**)

ElasticStack + Ansible	Fleet
Nos permite realizar de manera más controlada, una monitorización más adecuada a nuestra manera junto a una libertad en la hora de escoger qué datos de verdad queríamos mostrar y representar en infinidad de tipos de gráficos.	Monitorización básica.
Poder administrar de cualquier manera todos los hosts/clientes con Ansible.	Administración básica, añadir hosts, apagar y encender.

2.2.2 Tecnologías escogidas

Después de pensarlo mucho realizando pruebas, hemos podido llegar a la conclusión de utilizar diversos servicios que nos podrán proporcionar todo lo necesario para poder desarrollar el proyecto.

Las tecnologías escogidas han sido:

Osquery

Osquery es una herramienta que puede utilizarse en cualquier sistema operativo. Hace que el análisis y la supervisión del sistema operativo de bajo nivel sea eficaz e intuitivo.

Trata el sistema operativo como una base relacional de alto rendimiento y esto le permite escribir consultas SQL para explorar los datos del sistema. Las tablas SQL representan conceptos abstractos como módulos del kernel, procesos en ejecución, complementos del navegador, conexiones de red abiertas, hashes de archivos y eventos de hardware.

El paquete Osquery instala tres componentes básicos:

- **osqueryctl** → Es un script de ayuda para probar la configuración y despliegue de osquery, así como gestionar el servicio osqueryd.
- **osqueryd** → Es un demonio para programar consultas y registrar los cambios en el estado del sistema operativo.
- **osqueryi** → Es un intérprete de órdenes interactivas. Desde este intérprete podemos ejecutar diversas consultas para explorar el estado del sistema operativo.

Ansible:

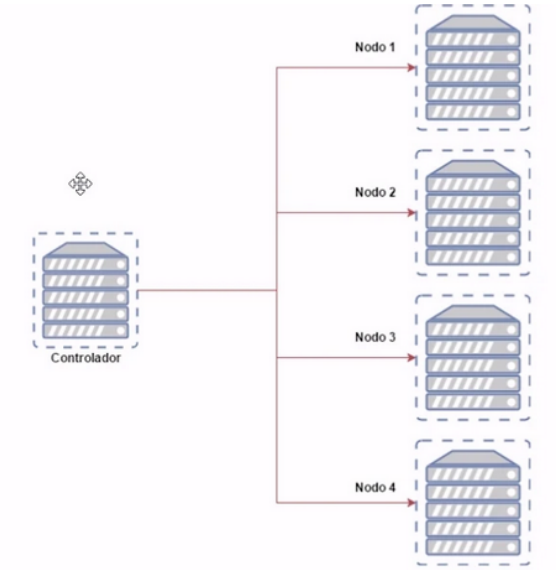
Software que nos automatiza la administración/gestión de sistemas, el cual permite realizar configuración y desplegar aplicaciones en estas máquinas de forma secuencial en distintas máquinas al mismo tiempo lo cual es muy útil para los administradores de sistemas.

Este realiza la gestión de sus nodos a través de SSH y únicamente requiere de Python en las máquinas clientes donde se vaya a ejecutar el contenido de distintos “playbooks” creados ya en el servidor.

Este consta de 2 tipos de servicios:

- **Controlador:** Es la máquina desde la que se realiza la gestión en diferentes nodos.

- **Nodo:** Son aquellas máquinas controladas por el controlador.



ElasticStack:

Elasticstack es un conjunto de programas donde su función es recoger datos de todo tipo del sistema/s para ayudarnos y facilitar la tarea de monitorizar. Está formado por los siguientes proyectos:

- **Beats:** Se encarga de recolectar y del envío de logs a Logstash o ElasticSearch. El beat escogido es **Filebeat**.
- **Logstash:** Recoge la información que le envían los Beats, la filtra y transforma antes de ser enviada a ElasticSearch.
- **ElasticSearch:** Recoge la información enviada por los beats y Logstash en índices que después se podrán utilizar para generar datos sobre el uso de los diferentes sistemas.
- **Kibana:** Es una interfaz de usuario gratuita que permite visualizar los datos de ElasticSearch de forma que el análisis y la monitorización de los datos sea mucho más sencillo.

2.2.3 Tecnologias descartadas

Osctrl:

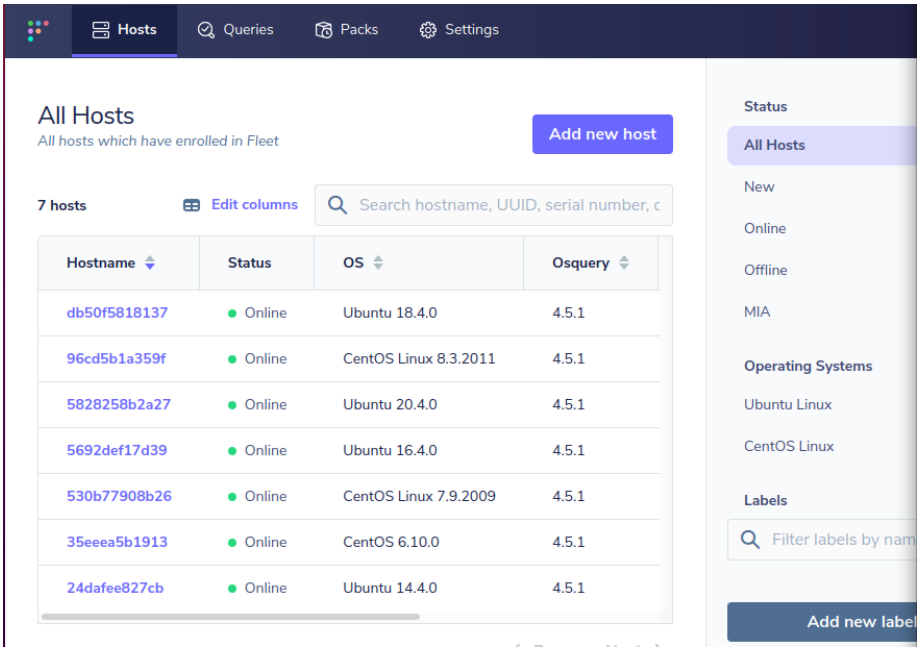
Tras una búsqueda sobre herramientas gráficas con las que usar osquery, la primera que quisimos implementar fue Osctrl. Los videos junto con la información que encontramos, nos pareció una buena opción y muy potente. Intentamos instalarlo de varias maneras que vimos en distintas páginas pero finalmente desistimos porque no nos funcionaba a causa de su complejidad.

Aunque si es una solución para gestionar osquery de forma rápida y eficaz que implementa su api remota como punto final TLS.

Se pueden supervisar todos los sistemas que ejecuten osquery, distribuyendo su configuración rápidamente, recoger todos los registros de estado y resultado, y permitir ejecutar consultas a la carta.

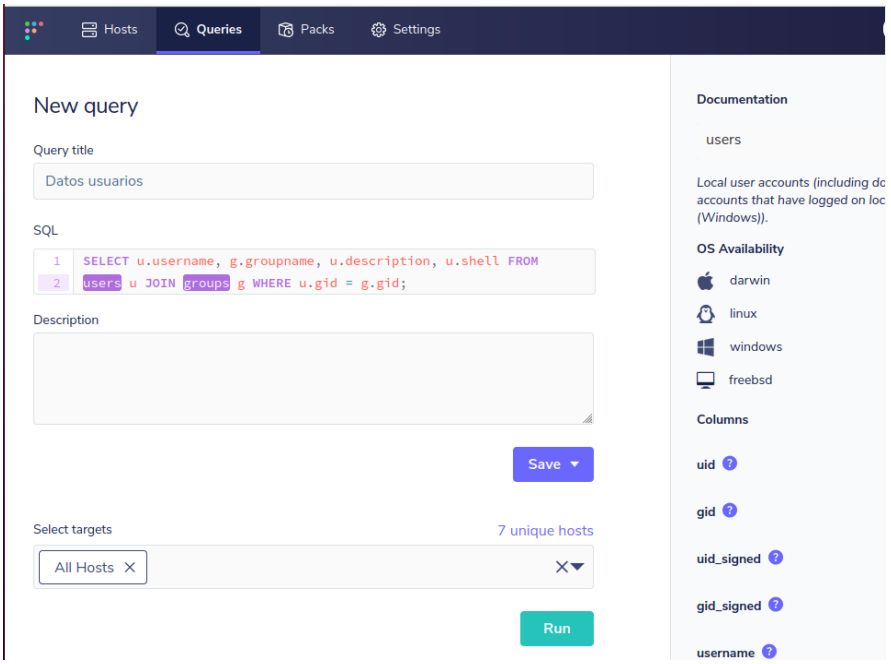
Fleet:

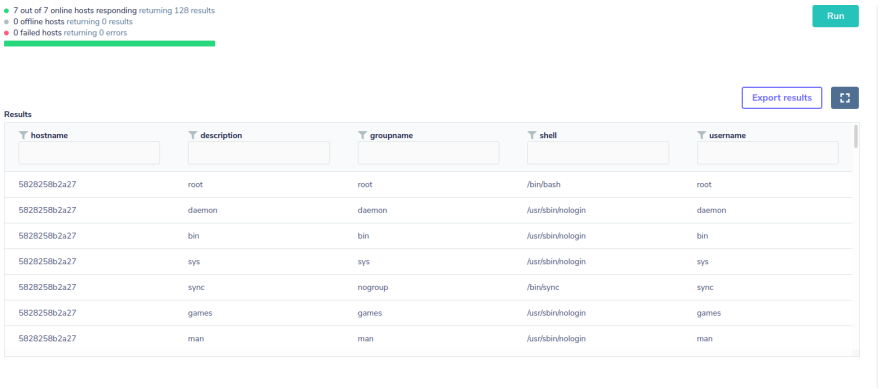
Después de que Osctrl fuera descartada, decidimos usar Fleet. Una vez terminamos de instalar la versión de prueba de Fleet nos pusimos a probar las opciones que nos aporta esta herramienta gráfica.



En el apartado de hosts vimos que desde aquí se puede ver los distintos ordenadores que monitorizamos y se pueden editar las columnas en función de los datos que nos interese ver.

Al intentar añadir un nuevo host tuvimos problemas ya que era muy complejo y finalmente no pudimos añadir otro más.





En el apartado de “queries” te da la opción de añadir consultas e ir almacenandolas para luego en vez de tener que estar escribiendo poder ejecutarlas con un simple click y puedes exportar esos resultados en un archivo. Resulta bastante útil y simplifica las tareas.

A pesar de eso, después de probar el programa nos encontramos que era muy simple y no tenía todo el potencial y herramientas que queríamos implementar para llevar a cabo nuestro proyecto.

Por eso decidimos dejarlo de lado y probar otro programa que cumpliera nuestras expectativas.

2.3 Estructura del proyecto

La estructura del proyecto se ha basado en crear un servidor encargado de centralizar todos los registros.

- Este obtendrá los datos de diversos hosts/clientes para poderlos procesar y mostrarlos de forma más gráfica para poder llevar un seguimiento con más facilidad.

Los hosts/clientes deberán de partir de una base ya creada que contendrá un par de configuraciones previas para poder ser monitorizados y administrados por el servidor central.

2.4 Definición de las tareas [proyecto de investigación]

ELK

El conjunto de paquetes de la Suite Elastic.

- **ElasticSearch**: Recogerá todos los registros ya filtrados enviado por el servicio Logstash en índices para después poder transferirlo a kibana.

- **Logstash**: Recoge la información en sucio de que enviará el Filebeat desde la máquina host/cliente.

- **Filebeat:** Enviará al servicio Logstash los datos en sucio que recogerá desde un archivo log del servicio a monitorizar.

- **Kibana:** La interfaz web que permitirá a los usuarios que accedan a la web poder monitorizar los datos obtenidos a través de los demás servicios.

Osquery

Software encargado de recoger los datos del sistema albergando estos dentro de una base de datos propia.
Utilizando el demonio podrá ejecutar las queries prediseñadas para mandar el registro a un log para luego poder ser recogido por filebeat.

Ansible

Encargado de la gestión de configuración automática y remota, mediante el protocolo ssh, permitiendo centralizar la configuración de diversos hosts/clientes.

3. Anexos

Presentación guía

En este apartado se podrán observar todas las instalaciones, configuraciones implementados para realizar el proyecto.

Se parte de la base donde encontramos un servidor central y las máquinas clientes:

- Servidor central [ELK-A]:
  - **ElasticSearch, Logstash, Kibana** y **Ansible**
- Host/Cliente:
  - **Filebeat** y **Osquery**

Instalación ELK en Linux Ubuntu 20.04 ~ [Servidor ELK-A]

ELK: ElasticSearch, Logstash y Kibana.

Guía para la instalación de la colección de software de "**código abierto**" (Entre comillas debido a que hay ciertas partes de este que es privativo) que nos permitirá realizar una búsqueda, análisis y visualización de los registros generados desde Osquery pasados por Filebeat en un registro centralizado.

Este registro centralizado nos ayudará/permitirá identificar problemas o simplemente conocer lo que sucede dentro de las máquinas clientes que se conectarán con la máquina servidora.

Explicación previa de los componentes principales de Elastic Stack:

- **ElasticSearch**: Motor de búsqueda que almacena todos los datos recopilados.
- **Logstash**: Componente ocupado del procesamiento de los datos.
- **Kibana**: Interfaz web encargado de realizar la búsqueda y mostrar una visualización de los registros obtenidos.
- **Beats**: Encargados de transportar los datos a **Logstash** o **ElasticSearch**.

Teniendo conocimiento de lo anterior procederemos a mostrar un tutorial simple donde se instalará **ElasticSearch**, **Logstash** y **Kibana** en el servidor **ELK-A (ELK-(A)** de ansible**)** y el beat **Filebeat** se instalará en las máquinas clientes, pero antes de llegar a esa historia instalaremos ELK en un servidor que centralizará los registros.

Los paquetes ELK no están disponibles en los repositorios de Ubuntu por defecto, es necesario agregar la lista de fuentes de paquetes de Elastic para instalarlo con **APT.**

1. **Importación del repositorio Elastic**

Descargar e importar la clave GPG pública a APT:

```
curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
```

Agregar la lista de fuentes Elastic a la **sources.list**:

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

Tras lo anterior, actualizamos la lista de paquetes:

```
sudo apt update
```

2. **Instalación y configuración ElasticSearch**:

En este proyecto se instalará una versión concreta utilizada en clase.

```
sudo apt-get install elasticsearch-7.9.1
```

Una vez instalado **ElasticSearch** editaremos el archivo de configuración principal con cualquier editor, ejemplo **nano**, **vi**, **vim**, etc...

```
sudo nano /etc/elasticsearch/elasticsearch.yml
```

Dentro de este archivo configuremos solo los siguientes datos, de tal manera que queden así (Si es necesario, se deberá de descomentar las líneas):

```
network.host 0.0.0.0
http.port: 9200
discovery.seed_hosts: ["0.0.0.0", "[::1"]
```

De esta manera permitiremos (Aunque no sería lo más adecuado, pero para el proyecto lo podremos implementar así) a cualquier host conectarse a este.

Una vez realizado la configuración procederemos a reiniciar y habilitar **elasticsearch**:

```
sudo systemctl start elasticsearch
sudo systemctl enable elasticsearch
```

Comprobar si el servicio se ejecuta correctamente enviando una petición HTTP:

```
curl -X GET "localhost:9200"
```

3. **Instalación y configuración Kibana**:

Tras el paso previo de importar la lista de fuentes de Elastic podemos seguir realizando la instalación de forma sencilla:

En este proyecto se instalará una versión concreta utilizada en clase.

```
sudo apt install kibana-7.9.1
```

Una vez instalado **Kibana** editaremos el archivo de configuración principal con cualquier editor, ejemplo **nano**, **vi**, **vim**, etc…

Como la instalación de kibana será realizado dentro de un servidor Ubuntu deberemos editar el archivo de configuración principal:

```
sudo nano /etc/kibana/kibana.yml
```

Donde solo realizaremos una modificación:

```
server.host: "0.0.0.0"
```

Una vez realizado la configuración procederemos a reiniciar y habilitar **Kibana**:

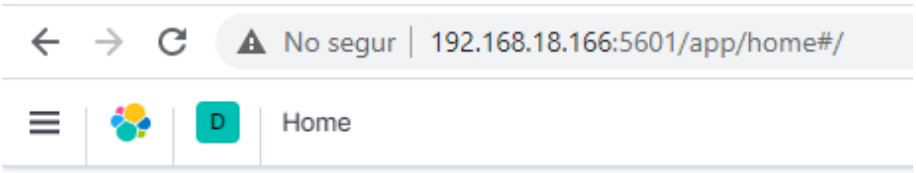
```
sudo systemctl enable kibana
sudo systemctl start kibana
```

Tras lo anterior, ya podrías acceder desde cualquier buscador web a través de:

```
http://ipservidor:5601
```

En nuestro caso:

- http://192.168.18.166:5601



4. **Instalación y configuración Logstash**:

Instalamos Logstash:

```
sudo apt install logstash
```

Una vez instalado **Logstash** crearemos 1 archivo (aunque puede ser en 2) con cualquier editor, ejemplo **nano**, **vi**, **vim**, etc…
En este archivo estableceremos la entrada de Filebeat y la salida en donde logstash almacenará los datos de los beats (en este caso Filebeat), en este caso ElasticSearch:

Creamos el archivo dentro del directorio:

```
/etc/logstash/conf.d/02-in-filebeat-out-elastics.conf
```

El cual contendrá:

```
input {
 beats {
  port => 5044
 }
}
output {
 elasticsearch {
  port => ["localhost:9200"]
  index => "osquerylog-%{+YYYY.MM.dd}"
 }
}
```

Comprobar si la configuración está escrita de manera correcta:

```
sudo -u logstash /usr/share/logstash/bin/logstash --path.settings /etc/logstash -t
```

Si nos muestra una salida de **OK** está todo correcto, si no, no, obviamente.

Una vez realizado la configuración procederemos a reiniciar y habilitar **Logstash**:

```
sudo systemctl start logstash
sudo systemctl enable logstash
```

Instalación y configuración de Osquery & Filebeat en Linux Ubuntu 20.04 ~ [Cliente]

1. Instalación y configuración de Osquery

Importación de la fuente e Instalación de Osquery.

El método elegido para realizar la instalación dentro de un sistema Ubuntu con versión 20.04 ha sido la instalación del repositorio apt específico para posteriormente instalar el servicio Osquery.

En una línea de comandos, ahora en adelante "**Terminal**":

```
export OSQUERY_KEY=1484120AC4E9F8A1A577AEEE97A80C63C9D8B80B
```

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys $OSQUERY_KEY
```

```
sudo add-apt-repository 'deb [arch=amd64] https://pkg.osquery.io/deb deb main'
```

```
sudo apt-get update
```

Instalación de osquery con apt:

```
sudo apt-get install osquery
```

Tras realizar la instalación de Osquery, creamos un archivo de configuración dentro de la ruta con el siguiente nombre de archivo (Ir para apreciarlo mejor).

```
sudo nano /etc/osquery/osquery.conf
```

Dentro de este archivo se especificará qué datos de cada tabla vamos a querer mostrar a través de Kibana.

El contenido de este archivo lo podremos encontrar en este repositorio de github al cual le subimos los archivos principales:
Enlace:

- https://github.com/ahades/filebeat-osquery/blob/main/playbooks/osquery/conf/osquery.conf

Los SELECTS ubicados en este archivo muestra (5 Ejemplos):

1. El consumo de CPU por cada Servicio, es decir, mostraremos qué servicios han estado consumiendo más CPU, en Kibana podremos especificar desde qué momento queremos ver qué servicios han consumido más.
2. Conexiones establecidas, que IP establece una conexión con otra máquina a través de un puerto/protocolo.
3. Usuarios conectados en la máquina.
4. IPv4 de las máquinas.
5. Muestra los usuarios, su descripción y su directorio home.

Comprobar si la configuración está escrita de manera correcta:

```
sudo osqueryctl config-check
```

Una vez realizado la configuración procederemos a reiniciar y habilitar **Osqueryd**, con lo cual tras iniciar el demonio de **Osquery** este empezará a utilizar la configuración donde se especifican los SELECTS para obtener los datos de la máquina y meterlos dentro del log especificado dentro del archivo de configuración del módulo Osquery de Filebeat.

```
sudo systemctl start osqueryd
```

2. **Instalación y configuración de Filebeat:**

De la misma manera en la que instalamos **ElasticSearch**, **Logstash** y **Kibana**, debemos importar la clave GPG de Elastic, añadir la fuente para la instalación posterior con **APT**.

Importación del repositorio Elastic

Descargar e importar la clave GPG pública a APT:

```
curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
```

Agregar la lista de fuentes Elastic a la **sources.list**:

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

Tras lo anterior, actualizamos la lista de paquetes:

```
sudo apt update
```

Instalamos Filebeat:

```
sudo apt-get install filebeat
```

Una vez instalado, se debe realizar una edición del archivo de configuración principal de Filebeat donde se especificará a donde, en este caso a la **IP** del **(Servidor ELK-A)** el cual mandará los datos, en este caso, Filebeat, mandará los datos al servicio Logstash ubicado en el servidor ELK-A.

Como en la configuración de Osquery, añadimos este archivo de configuración a nuestro repositorio de Github (Ir allí para apreciarlo mejor).

Enlace:

- https://github.com/ahades/filebeat-osquery/blob/main/playbooks/filebeat/filebeat.yml

Por último se debe de activar el módulo **osquery** de filebeat, esta habilitación del módulo no es más el cambio de nombre del archivo del módulo (Para que lo entendamos):

- Módulo desactivado: **osquery.yml.disabled**
- Módulo activado: **osquery.yml**

Este archivo, **osquery.yml** se ubica en el siguiente directorio:

```
/etc/filebeat/modules.d/
```

Con la siguiente configuración donde se especifica, el módulo habilitado y el archivo log de este, del cual Filebeat cogerá los datos y los mandará hacía logstash de la máquina server ELK-A:

```
- module: osquery
  result:
    enabled: true
    var.paths: ["/var/log/osquery/osqueryd.results.log*"]
```

Una vez realizada lo anterior, es necesario iniciar el servicio Filebeat:

```
sudo systemctl start filebeat
```

Deshabilitar IPv6 ~ [Cliente]

En este modo de recolección de datos a través de osquery se implementó una query para obtener la IP de la máquina donde se hospedan los servicios osquery y filebeat.

Y es por esto, que a la hora de obtener la IP nos mostraba la IPv4 y la IPv6 en el mismo valor de la columna y solo queríamos obtener la IPv4 entonces decidimos deshabilitar esta IPv6 por mero capricho, utilizando/ejecutando los siguientes comandos:

```
sysctl -w net.ipv6.conf.all.disable_ipv6=1
```

Entendiendo la configuración de osquery.conf

Una explicación simple para poder comprender de manera más fácil y rápida de lo que se realiza/añade dentro de este archivo.

Podemos encontrar la información de cada apartado de manera más detallada, ya sea desde los enlaces que hemos añadido en cada apartado o bien ir directamente a realizar una búsqueda del mismo sitio web oficial de Osquery, nos ofrece una cantidad de documentación de maneras de crear,utilizar los apartados que se explican a continuación.

Documentación de osquery:
- https://osquery.readthedocs.io/

Se puede componer de 4 apartados:
- **Options**
  - Define un mapa de pares de nombre con valor de "opciones''. Es decir, nos permite indicar las opciones de configuración de osquery.
  - Más información y ejemplos:
    - https://osquery.readthedocs.io/en/latest/deployment/configuration/#options
- **Schedule**
  - Define el mapa de las **queries**, consultas programadas.
  - Más información de como utilizarlo (Ejemplo más explicación):
    - https://osquery.readthedocs.io/en/stable/deployment/configuration/#schedule
- **Decorators**
  - Se utilizan para agregar "decoraciones" adicionales a los registros de resultados y de instantáneas. Hay tres tipos de consultas de decoradores en función de cuándo y cómo desea los datos de decoración.
  - Más información de como utilizarlo (Ejemplo más explicación):
    - https://osquery.readthedocs.io/en/stable/deployment/configuration/#decorator-queries
- **Packs**
  - Sin tanto rodeo, nos permite agregar "packs" lo que es igual a nombres de paquetes con un valor igual a una cadena o un diccionario (objeto) de queries prediseñada.
  - Más información de como utilizarlo (Ejemplo más explicación):
    - https://osquery.readthedocs.io/en/stable/deployment/configuration/#packs

Entendiendo nuestro archivo de configuración osquery.conf

Hemos añadido nuestra configuración en nuestro repositorio de Github, donde podrás encontrar el "código":
Enlace:

- https://github.com/ahades/filebeat-osquery/blob/main/playbooks/osquery/conf/osquery.conf

Puedes abrir el archivo de configuración en el enlace anterior mientras lees lo siguiente donde explicamos lo que hemos definido dentro de este archivo mientras visualizas la menera (ejemplo) de cómo se puede componer.

options:

- Se especifica plugin a utilizar por osquery de la manera en la que obtendrá los datos.
- Archivos de log a donde irán a parar todas las consultas.
  - De estos log file beat obtendrá los registros para transferirlos al servidor ELK-A.
- Definición del tiempo a transcurrir tras cada ejecución de queries escritas en schedule, se define 10, 10 segundos (mínimo) de diferencia entre cada consulta.
- Definición de la base de datos osquery a utilizar.
- Identificador de la máquina.

schedule:

- **(servicio_cpu_porcentaje)**
  - Query que nos mostrará la utilización de la CPU de los servicios. (Solo los principales 15 servicios que consumen CPU mayor que 0.)
  - Utiliza la siguiente QUERY: **CPU-Porcentaje-Serv**.
- **(conexiones_establecidas)**
  - Query que nos mostrará las conexiones establecidas entre la máquina local y remota.
  - Utiliza la siguiente QUERY: **Conexiones_establecidas.**
- **(usuarios_on)**
  - Otra query SELECT que muestra los usuarios "personas" conectados en ese momento en el sistema:
  - Utiliza la siguiente QUERY: **Usuarios_On.**
- **(ipv4)**
  - Mostrará las columnas IPv4 e interfaz de red de la máquina.

decorators:

- Hemos indicado una query SELECT que al cargar por primera vez (iniciar osqueryd), guardará un registro en el log **/var/log/osquery/osqueryd.results.log** donde se mostraran los usuarios, la descripción y la ruta "home" de estos mismos.
    - Utiliza la siguiente QUERY: **Usuario-Descripción-Directorio**.

packs:

- Ninguno debido a que en nuestro proyecto no utilizaremos estos packs los cuales mostrarían contenido ya predefinidos por esta aplicación y nuestro enfoque ha sido "realizarlo manualmente".

Queries implementadas en el proyecto:

CPU-Porcentaje-Serv: Calcular y mostrar el porcentaje de utilización de la CPU de un servicio en ejecución:

```
SELECT name, percentage FROM (SELECT name, ROUND((
(user_time + system_time) / (cpu_time.tsb - cpu_time.itsb)
) * 100, 2) AS percentage
FROM processes, (
SELECT (
 SUM(user) + SUM(nice) + SUM(system) + SUM(idle) * 1.0) AS tsb,
 SUM(COALESCE(idle, 0)) + SUM(COALESCE(iowait, 0)) AS itsb
 FROM cpu_time
) AS cpu_time
ORDER BY user_time+system_time DESC) WHERE percentage > 0.0;
```

Usuario-Descripción-Directorio: Select que muestra los usuarios, la descripción y la ruta "home" de estos mismos:

```
SELECT username,description,directory FROM users;
```

Usuarios_On: Select que muestra los usuario "personas" que se encuentran conectados en ese momento:

```
SELECT DISTINCT user AS usuario_on FROM logged_in_users WHERE tty NOT LIKE "~";
```

Conexiones_establecidas: Select que muestra las conexiones establecidas.

```
SELECT local_address, remote_address, local_port, remote_port FROM process_open_sockets
s JOIN processes p ON s.pid = p.pid WHERE local_port NOT IN (0) AND remote_port NOT IN
(0);
```

IPv4: Mostrará la IPv4 de la máquina:

```
SELECT interface,address AS ipv4 FROM interface_addresses WHERE interface NOT LIKE
'%lo%';
```

Consultas varias Osquery

```
SELECT u.username, g.groupname, u.description, u.shell FROM users u JOIN groups g WHERE u.gid = g.gid;
```

```
SELECT interface,address,mask FROM interface_addresses WHERE interface NOT LIKE '%lo%';
```

```
SELECT name, path, pid, FROM processes WHERE on_disk = 0;
```

```
SELECT pid,name,uid FROM processes;
```

```
SELECT * FROM processes LIMIT 5;
```

```
SELECT pid,name,uid FROM processes WHERE uid != 0;
```

```
SELECT md5 FROM file JOIN hash USING (path) WHERE path = '/etc/issue';
```

```
SELECT * FROM deb_packages;
```

```
SELECT * FROM users; // SELECT * FROM users where uid >=1000;
```

```
SELECT * FROM kernel_modules;
```

```
SELECT hostname FROM system_info; // SELECT * FROM system_info;
```

```
SELECT uid, username FROM users;
```

```
SELECT pid, name, path FROM processes;
```

```
SELECT * FROM uptime;
```

```
SELECT p.pid, p.name, p.path, u.username, u.shell FROM processes AS p JOIN users AS u ON p.uid = u.uid;
```

```
SELECT DISTINCT process.name, listening.port, listening.address, process.pid FROM processes AS process JOIN listening_ports AS listening ON process.pid = listening.pid;
```

```
SELECT username FROM users; // SELECT username, directory FROM users;
```

```
SELECT * FROM users WHERE username="root";
```

```
SELECT pid FROM processes WHERE name="firefox";
```

```
SELECT * FROM os_version;

SELECT memory_total FROM memory_info;

SELECT memory_free FROM memory_info;

SELECT cached FROM memory_info;

SELECT * FROM groups;

SELECT * FROM listening_ports;

SELECT DISTINCT processes.name, listening_ports.port, processes.pid FROM listening_ports
JOIN processes USING (pid) WHERE listening_ports.address = '0.0.0.0';
```

Más ejemplos de queries:
https://github.com/teoseller/osquery-attck/tree/master/Linux

Instalación e implementación de Ansible en Linux Ubuntu 20.04 ~ [Servidor ELK-A]

Instalación de Ansible.

Ansible no se encuentra dentro de los repositorios de APT, por esto, es necesario agregar la fuente de donde irá APT a buscar este servicio:

```
sudo apt-add-repository ppa:ansible/ansible -y
```

Una vez agregado el repositorio de Ansible, realizaremos un update para posteriormente poder realizar la instalación de Ansible:

```
sudo apt update
sudo apt install ansible
```

Para que el servicio Ansible pueda acceder a las máquinas "clientes" es necesario depender de la utilidad sshpass con la cual ansible podrá conectarse mediante ssh de forma no interactiva.

```
sudo apt install sshpass
```

Indicar a Ansible a hosts se conectará:
Para esto, debemos editar el archivo de configuración con el editor preferido por consola:

```
/etc/ansible/hosts
```

En este archivo indicaremos a qué máquinas Ansible se conectará cuando se ejecute un playbook.

Nosotros en el proyecto hemos realizado varias pruebas donde una de ellas es especificar un rango de hosts:

```
[clienteselka]
192.168.18.[230:235]
```

Uso de playbooks con Ansible.

Para la realización de varias tareas de forma coordinada, es necesario utilizar playbooks, estos son archivos de tipo YAML, que contendrán toda la lista de ítems que definirán las operaciones a realizar en los hosts/clientes a los que se conecte.

Por esto hemos creado nuestro propio archivo playbook:
Enlace (Nuestro Github):

- https://github.com/ahades/filebeat-osquery/blob/main/playbooks/filebeat-osquery.yml

Explicación del contenido del playbook.

- Hemos especificado el grupo al cual queremos que Ansible realice diversas operaciones conectándose a estas máquinas mediante el user **usuario**, el cual deberá ejecutar las siguientes listas de operaciones como **superusuario**.

- En la lista de ítems/tareas podemos encontrar diversas operaciones que hemos decidido que sean automatizadas:

- **Filebeat**:
  - Instalación (Añadir repositorio, Instalación e inicio del servicio)
  - Configuración (Copia archivo prediseñado ubicado en la máquina server ELK-A el cual será transferido a máquina Cliente)
  - Habilitación del módulo Osquery (Copia archivo prediseñado de máquina server ELK-A a máquina Cliente)

- **Osquery**:
  - Instalación (Añadir repositorio, Instalación e inicio del servicio)
  - Configuración (Copia archivo prediseñado ubicado en la máquina server ELK-A el cual será transferido a máquina Cliente)
  - Habilitación del módulo Osquery (Copia archivo prediseñado ubicado en la máquina server ELK-A el cual será transferido a máquina Cliente)

- **Deshabilitar IPv6**:
  - Esto es mera operación capricho de nosotros para solo mostrar la IPv4 de los hosts, donde se le indica al host/cliente que deshabilite la dirección IPv6.

Preparación de una máquina cliente [BASE]

En la necesidad de automatizar toda la implementación, instalación, configuración dentro de una máquina cliente, hemos decidido crear una máquina con unos servicios ya instalados para que estas máquinas puedan ser administradas mediante Ansible.

En estas máquinas solo hemos añadido 2 paquetes:

- **python3**
  - (Para que Ansible pueda administrar correctamente la máquina)

- **openssh-server**
  - (Para que Ansible pueda conectarse a la máquina)

La máquina cliente [BASE] además de contener los 2 paquetes necesarios, nombrados anteriormente, hemos decido crear un usuario ya diseñado que podría ser entendido como un usuario **Administrador** con el cual Ansible podrá realizar las operaciones necesarias, en este caso hemos decido crear el user llamado **Usuario** que tendrá poderes de superusuario:

Creamos el usuario:

```
sudo useradd -s /bin/bash -m usuario && sudo passwd usuario
```

Permitimos al usuario creado usar sudo, lo estamos añadiendo a sudoers.

```
sudo usermod -aG sudo usuario
```

Con todo lo anterior ya podríamos utilizar Ansible para automatizar y realizar o incluso añadir cambios dentro de las máquinas clientes controladas por él de forma remota.

Aquí os dejamos una imagen de la ejecución del playbook:

(Como la imagen sería muy grande, hemos decidido recortarla para mostrar que los procesos indicados en el playbook se han ejecutado correctamente)

Antes que nada hemos decidido ejecutar el playbook en un rango de IPv4 de máquinas:

- Rango: **192.168.18.230 a 192.168.18.233**

```
usuari@aelk:~/playbooks$ ansible-playbook filebeat-osquery.yml -k -K
SSH password:
BECOME password[defaults to SSH password]:

PLAY [Instalacion y Configuracion Filebeat mas Osquery] **************

TASK [Gathering Facts] ***********************************************
ok: [192.168.18.233]
ok: [192.168.18.230]
ok: [192.168.18.232]
ok: [192.168.18.231]
```

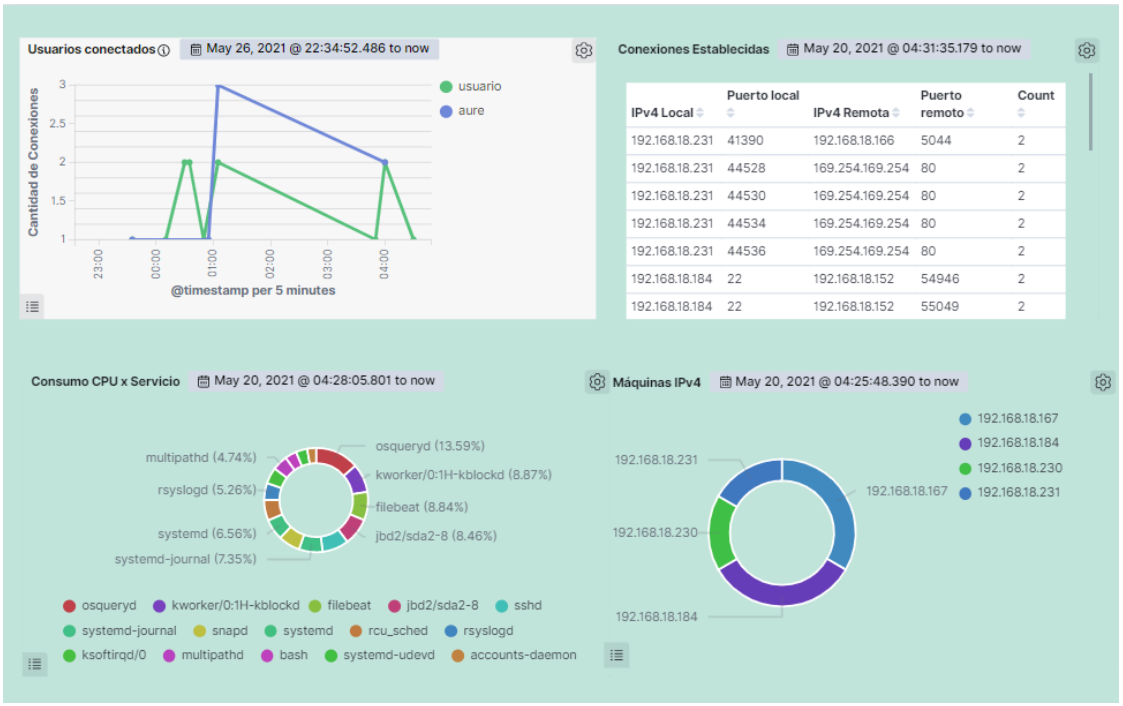
Como podemos comprobar, el playbook ha sido ejecutado de manera exitosa y los procesos en todas las máquinas han sido ejecutados también de forma correcta.

```
PLAY RECAP ***********************************************************************************************************
192.168.18.230             : ok=11   changed=10   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.18.231             : ok=11   changed=10   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.18.232             : ok=11   changed=10   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.18.233             : ok=11   changed=10   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Presentación final de los datos

Vista final de la obtención y representación de los datos obtenidos de los hosts/clientes.

Tras esto podemos observar como en Kibana, tras que nosotros hemos creado varios gráficos como estos datos de las máquinas son obtenidos y mostrados:



4. Conclusiones

4.1 Conclusiones generales del proyecto

Tras meditar con cuidado hacia donde queríamos enfocar nuestro proyecto decidimos combinar los servicios utilizados en clase con otros nuevos que nos permitieran poder realizar de una forma distinta a lo aprendido durante el curso, para adquirir nuevos conocimientos y de este modo combinamos lo aprendido junto a nuestra pasión por la administración de los sistemas.

Esto nos permitiría realizar un trabajo del cual aprender por una parte y por otra poder fortalecer nuestros conocimientos.

En el transcurso de la instalación y configuración de los servicios hemos ido comprobando de buena mano la complejidad de cada uno de ellos, incluso llegando a descartar alguno por la falta de información de este mismo en internet, teniendo que seleccionar otros con una mejor documentación para poder implementarlos.

Esto nos ha llevado a cambiar nuestros planes iniciales, donde en principio se quería implementar unos servicios en concreto, ahora son otros distintos pero con la misma finalidad.

Como conclusión final, hemos adquirido gran riqueza de conocimientos, aprendido a trabajar en equipo, y con gratitud expresar que hemos podido alcanzar nuestros objetivos a pesar de las trabas que nos ha causado la falta de información sobre algunas tecnologías, pero gracias a ellas hemos podido sobreponernos a la adversidad.

4.2 Consecución de los objetivos.

Hemos podido cumplir todos nuestros objetivos e incluso crear un repositorio en Github del playbook creado para ansible el cual permite realizar la instalación de lo necesario en los hosts/clientes para ya ser administradas de manera automática por el servidor central.

4.3 Valoración de la metodología y planificación

Cuando realizamos la planificación inicial no esperábamos tener que descartar e investigar por varias veces diferentes programas, su uso, configuración, etc. A medida que iban pasando las semanas hemos tenido que adaptar los objetivos y el tiempo a las nuevas situaciones surgidas y donde pensábamos que nos haría falta menos tiempo, ha resultado ser lo contrario y viceversa.

Finalmente, hemos logrado alcanzar los objetivos propuestos una vez la idea del proyecto ya era la definitiva.

4.4 Visión de futuro

Respecto al futuro, los programas que hemos utilizado tienen muchísimas opciones y herramientas que no hemos podido investigar del todo o implementarlas debido al limitado tiempo.

Lo primero sería investigar más sobre los packs de consultas que trae Osquery, ya que en su página web habla de algunos como el de detección de intrusiones, respuesta a incidentes, TI, gestión de vulnerabilidades y algunos más. Sería interesante ver lo útiles que realmente pueden llegar a ser y la ayuda que nos puede proporcionar.

Por otra parte sería interesante añadir algún Beat mas como por ejemplo Metricbeat que nos daría información sobre metricas y estadisticas de cualquier tipo en cualquier equipo relacionadas con el sistema.

Y por último Auditbeat para llevar un registro y control de los eventos del kernel Linux, así como modificaciones a ficheros.

5. Glosario

Elastic Stack - ELK Stack: Conjunto de software de código abierto enfocados a la obtención de datos de cualquier tipo de fuente en diversos formatos para la búsqueda, análisis y visualización de los registros en tiempo real.

Ansible: Software enfocado a la gestión remota y automatizada mediante el protocolo SSH.

Osquery: Software encargado de recoger los datos del sistema albergando estos dentro de una base de datos propia.

Demonio: Tipo de proceso en informática no interactivo, el cual se ejecuta en segundo plano.

Playbook: Lista de tareas/procesos que se aplicará en los hosts/clientes.

6. Bibliografía

Elastic Stack

https://www.elastic.co/es/kibana
https://www.elastic.co/es/elastic-stack
https://enimbos.com/monitorizacion-de-aplicaciones-usando-elk-stack/
https://www.davincigroup.es/introduccion-a-elasticsearch-que-es-casos-de-uso-instalar/
https://www.hebergementwebs.com/tutorial-de-kibana/kibana-trabajar-con-graficos
https://www.elastic.co/es/what-is/elasticsearch-graph
https://www.ionos.es/digitalguide/online-marketing/analisis-web/tutorial-de-kibana/
https://techexpert.tips/es/elasticsearch-es/filebeat-envio-de-los-mensajes-de-syslog-a-elasticsearch/
https://www.elastic.co/guide/en/beats/filebeat/7.13/filebeat-installation-configuration.html
https://www.elastic.co/guide/en/logstash/current/installing-logstash.html
https://www.bujarra.com/instalando-logstash/
https://elpuig.xeill.net/Members/q2dg/seguretat-mp11/uf2/elk/elk-i-filebeat.pdf/view
https://elpuig.xeill.net/Members/q2dg/seguretat-mp11/uf2/elk/elk-ii-logstash.pdf/view
https://elpuig.xeill.net/Members/q2dg/seguretat-mp11/uf2/elk/elk-iii-elasticsearch.pdf/view
https://elpuig.xeill.net/Members/q2dg/seguretat-mp11/uf2/elk/elk-iv-kibana.pdf/view
https://elpuig.xeill.net/Members/vcarceler/articulos/ansible/index_html

Osquery

https://osquery.io/
https://github.com/osquery/osquery
https://osquery.readthedocs.io/en/latest/
https://pkg.osquery.io/deb/osquery_2.10.2_1.linux.amd64.deb

Fleet
https://github.com/kolide/fleet
https://fleetdm.com/

Ansible
https://github.com/juju4/ansible-ipv6/blob/master/tasks/ipv6-disable.yml
https://docs.ansible.com/ansible/latest/collections/ansible/builtin/systemd_module.html
https://docs.ansible.com/ansible/2.5/modules/copy_module.html
https://ualmtorres.github.io/CursoAnsible/tutorial/
https://www.elastic.co/guide/en/kibana/current/canvas-tutorial.html
https://atareao.es/tutorial/ansible/playbooks-de-ansible/
https://ansible.github.io/workshops/exercises/ansible_rhel/1.3-playbook/README.es.html