Implementación de un sistema de monitorización con elastic stack

Juan de Dios Cárdenas Deza

Resumen — El siguiente articulo realiza una síntesis sobre el desarrollo e implementación de un sistema de monitorización de datos basado en un conjunto de herramientas software llamado elastic stack, para el procesamiento e ingesta en diferentes cuotas de datos de tipo texto, independiente del sistema u origen que genera la información y para diversos casos de uso a tiempo real. Por el lado de la implementación para la gestion de la información, está se realiza bajo 3 entornos on premise (local), sistemas de contenedores con Docker y en un entorno IAAS en cloud, que trabajaran en 2 modalidades de tipo modo Single node y Cluster. Como parte del proceso de control y monitorización del propio sistema se realiza el desarrollo de una SPA (single page application), utilizando el framework progresivo de vuejs con JavaScript como leguaje de programación en un entorno de node.js, para hacer uso de la API RESTful de elasticsearch que permite administrar y consultar los datos indexados del sistema desde sus EndPoints.

Palabras clave — Elasticsearch, logtash, kibana, beats, monitorización, clúster, API, shards, dashboard, cloud, desarrollo, implementación.

Abstract— The following article explains the development and implementation of a data monitoring system based on a set of software tools called elastic stack, for processing and ingesting in different quotas of text-type data, independent of the system that generates it and for various use cases in real time. On the side of the implementation for the management of the information it is carried out under 3 environments on premise (local), systems of containers with Docke and in an environment IAAS in the cloud that worke in 2 modalities of type Single node and Cluster. And as part of the process of control and monitoring of the system itself is the development of a SPA (single page application), using the progressive framework of vuejs with Javascript, to make use of the API RESTful of elasticsearch that allows you to manage and consult the system's indexed data.

Index Terms— Elasticsearch, logtash, kibana, beats, monitoring, cluster, API, shards, dashboard, cloud, development, implementation.

1 Introducción

La informática hoy en día es posible que sea considerada como una de las nuevas eras digitales de estos tiempos; desde sus inicios uno de los aspectos importantes en términos de evolución ha sido la ralentización de su avance tecnológico hacia la democratización del hardware, esto hace referencia *al valor del coste de procesamiento necesario sobre infraestructura* para alcanzar un objetivo, hoy en día existe un cambio de paradigma del modelo, ya que es posible contar con los medios necesarios para adaptar una empresa a sus necesidades tecnológicas sin que esto repercute altamente a sus beneficios como costes de implementación y mantenimiento del propio sistema. En la actuacidad existen modelos de infraestructura de computo en la nube de tipo IaaS[1], PaaS[2] y SaaS[3]. Por otro lado, la conectividad y sus altas cuotas de velocidad y transferencia dan las condiciones idóneas para la recolección y el análisis en tiempo real de diferentes cuotas de volúmenes de datos y generación continua de eventos.

Estos factores además de traer grandes beneficios también incluyen enormes retos como a la seguridad, la fiabilidad, disponibilidad, el rendimiento, la elasticidad, escalabilidad y entre otros el tramiento de la información.

Hoy en día la información es conocimiento y para poder encontrar medios que permitan el tratamiento y su lectura acotando la complejidad de preprocesamiento, nos plantea el reto de buscar soluciones a corto plazo.

- *E-mail de contacto: JuanDeDios.Cardenas@e-campus.uab.cat*
- *Referencia a la Mención: Mención en Tecnologías de la Información.*
- *Tutor del Proyecto: Rafael Fernández González.*

[1] Un proveedor proporciona acceso a la infraestructura de almacenamiento, red y otros recursos en la nube, la gestion es propia de quien la contrata.

[2] Un proveedor proporciona Infraestructura para plataformas destinadas al desarrollo.

[3] Software funciona como Servicio donde esta incluida la infraestructura y la plataforma.

Elastic stack [1] es una solución entre muchas a uno de estos desafíos, desde la implementación como en el desarrollo, se expondrá el análisis y control de la información, para un punto de vista de la lectura, monitorización y vizualición de sistemas, hacia la información a procesar. Este stack es un proyecto *Open source* y cuenta con un motor de búsqueda y análisis de texto distribuido, está basado en el lenguaje de programación Java y como núcleo utiliza la librería lucene[4]. Dentro de su arquitectura se define y organiza la información en índices[5], los cuales están distribuidos y particionados como shards[6], para poder otorgar al sistema características de alta disponibilidad donde se distribuyen replicas por todos sus nodos. Toda la información esta almacenada con el formato de tipo JSON.

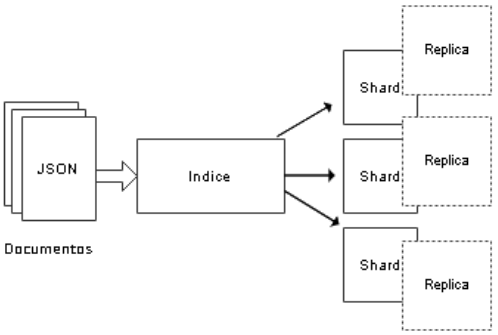


Fig. 1. Arquitectura interna de Elasticsearch una vez indexados los documentos.

Elastic está compuesto por una serie de soluciones software interconectadas entre sí para la ingesta, preprocesamiento, indexado y almacenamiento de la información, estos componentes son los siguientes:

- Elasticksearch es el nodo procesador de la base datos no relacional y distribuida.
- Kibana para la explotación de los datos y proporciona el entorno UI.
- Logtash la etapa que transforma, parsea y procesa la información, como un proceso similar a un ETL.
- Beats es un agente para la recolección e ingesta de los datos, alojado en el origen de la información.

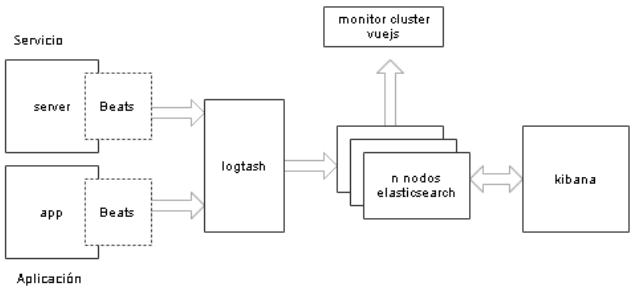


Fig. 2. Estructura del sistema elastic con la monitorización del cluster.

Para poder tener un control del funcionamiento correcto del cluster bajo la arquitectura con la que trabaja elasticsearch, se desarrolla una SPA[7](single page aplication) en JavaScript denominada herramienta *monitor del cluster* con el framework de Vuejs [2], que nos permita monitorizar el sistema en lo que respecta a el Cluster, los nodos y los índices.

2 Estado del Arte

Actualmente existen plataformas que ofrecen servicios similares como Hadoop [3], Sumologic [4], Splunk [5], Graylog [6], pero a diferencia de todas estas alternativas o dentro de los requerimientos de la elección por "Elastic", destaca su simplicidad de puesta en marcha bajo un solo nodo(Single node), la flexibilidad de integrarla a un producto en un entorno en desarrollo, evitando así grandes esquemas a la hora de implementar el sistema dentro de una organización. Ahora extenderla a otros patrones requiere conocimientos tecnológicos adecuados, como por ejemplo un conocimiento del producto y de tecnologías que permitan su crecimiento para una puesta en marcha en producción.

2.1 Comparativa

Algunos aspectos que clasifican y diferencia a otros productos de Elastic, es Hadoop producto capaz de procesar *cualquier tipo de dato* y no solo está limitado a datos de texto, pero su programación con *MapReduce*[8] no es tan eficiente para las tareas analíticas e iterativas [7]. Splunk, Graylog y Sumologic son más similares, pero cuentan con características cada una de ellas que hacen apropiada o no la elección como, por ejemplo, Splunk es software propietario(Licencia de pago) y Graylog necesita de "Elasticserch" para el almacenamiento y motor de búsqueda. Otra característica que hace de Elastic una alternativa válida es la integración hacia servicios de terceros, esto significa que, al realizar tareas de recolección, parseo, procesado, indexación y almacenamiento es una buena solución para adaptarla a otros productos.

Existen otras diferencias más al detalle que pueden ser consultadas en sus respectivas fuentes oficiales.

Finalmente, al realizar una consulta en Google trends [8] para ver una comparativa como termino de búsqueda de estos servicios y así poder darnos una idea bajo que demanda está considerada esta herramienta como alternativa entre las fechas del 2004 y Actualmente.

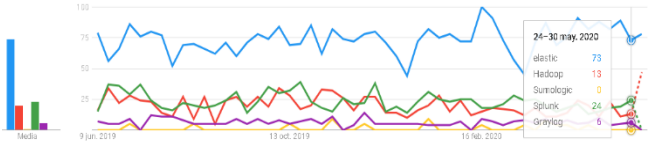


Fig. 3. Google trend comparativa en términos de búsqueda de productos software para análisis de datos.

Siendo Elastic de color azul, obtiene más demanda bajo el termino de búsqueda como se observa en la Fig. 3.

[4] Librería para la recuperación de información de código abierto, útil para cualquier aplicación que requiere indexado y búsqueda a texto completo.

[5] Dentro de la arquitectura del elastic, un indice puede ser conciderado como una tabla dentro del modelo de base de datos relacionales.

6 Un shard se considera como una instancia y partición de un indice.

[7] Aplicacion de pagina única, cuyo proposito es dar una experiència mas fluida a los usuarios.

[8] modelo de programación para dar soporte a la computación paralela sobre grandes colecciones de datos

Una vez analizados los diversos productos actuales del mercado y teniendo en cuenta sus características y demanda, es" Elastic" la elección al software que se utiliza en este Proyecto.

Para el ámbito del desarrollo de una herramienta que permita monitorizar el sistema, se ha elegido JavaScript como lenguaje por interés personal y por ser un lenguaje con demanda actualmente. Esta herramienta está enfocada a ser una SPA por tanto para realizar este enfoque nos decantamos por utilizar un framework que nos facilite el planteamiento. Si repasamos algunas encuentras populares en relación al software como en Stackoverflow [9], podemos hacernos una idea del estado actual de las tendencias del software en diferentes aspectos. Repasando de esta manera la información encontramos que existen 3 posibles soluciones(React| Vuejs |Angular) donde la elección más adecuada es "Vuejs".

Las razón es porque se adapta bien a nuestras necesidades como la curva de aprendizaje y el DOM virtual [10] muy eficiente para herramientas orientadas a ser reactivas cómo para la monitorización.

3 OBJETIVOS

La implementación y desarrollo de este proyecto esta divido en *5 objetivos*, los *3 primeros* están orientados a la implementación del sistema Elastic Stack, el *cuarto* al desarrollo de la herramienta para monitorizar el funcionamiento del sistema y el *ultimo* a la documentación de todo el proyecto.

3.1 Conocimiento del sistema a implementar

Uno de los requerimientos para la implementación es el conocimiento del sistema, por tanto, es preciso conocer en profundidad su funcionamiento y como está construido para poder lograr una explotación adecuada. Para ello se ha recurrido a la documentación oficial de Elastic y Vuejs.

Esto nos ha permitido dentro del periodo planificado poder implementar el sistema de manera fluida, con pocos inconvenientes a la hora de ponerlo en marcha en los 2 modos, *single node* para desarrollo y *cluster* para producción.

Además de tener en cuenta como está construido para implementar el sistema, también es necesario conocer al respecto cada parte del software para poder ponerlo en práctica en los casos de uso propuestos, para su correcta configuración.

En el modo *single Node* para un uso en desarrollo, requerimos de los 4 componentes principales de Elastic, pero siendo "Elasticsearch" un solo nodo en lugar de 3 como se observa en la Fig. 2. Porque va enfocada también a nuestro primer caso de uso.

En modo cluster se requiere n nodos o tantos nodos como se plante el caso de uso para "Elasticsearch", lo cual representa una necesidad de recursos importante, *esta configuración se realiza normalmente al pasar a un estado en producción.*

Por su arquitectura los nodos de "Elasticsearch" realizan diversas tareas y pueden estar definidos en distintos roles que son los siguientes:

- Nodo Master, requiere como mínimo para establecer un cluster de 2 nodos y cuenta con las funcionalidades de creación y eliminación de índices, estado del cluster, decidir la ubicación de los shards, dirigir las búsquedas a la correspondiente data node, en conclusión, puede realizar el rol de todos los nodos.
- Nodo Data, almacenar datos y contiene las funcionalidades CRUD (Créate |Ready | Update | Delete).
- Nodo Ingest, realiza tareas de preprocesamiento de la información antes de indexar los datos.
- Nodo Coordinador, balanceo de carga, encamina las búsquedas y distribuye los índices.

Esta información es importante conocerla para lo que requiere la puesta en marcha del sistema, ahora en la parte de desarrollo, se ha podido estructurar la herramienta monitor en diferentes componentes debido a la estructura del framework, esto ha permitido estructurar la SPA como estaba planificado, apuntando a los EndPoint[9] proporcionados por "Elastisearch".

En el apartado de preparar entorno se pondrá en marcha ya el control de versiones, para ello se hace uso de GitLab para tener una metodología del control de toda la información que se va generando durante la implementantión y se generara todas las configuraciones necesarias mediante Script Shell para automatizar el proceso de entorno.

3.2 Preparar entorno

Esta tecnología funciona para diversos sistemas y entornos, por tanto, para experimentar su funcionamiento y a su vez según los requerimientos, se debe de testear primero para conocer cuál sería el entorno adecuado y hacer uso de ello en la práctica por tanto los implementamos en los 3 entornos, el on premise[10], contenedores Docker y cloud.

Distribuciones Open Source

En este Apartado se evaluó que distribuciones como base eran las adecuadas para ser implementado el sistema, la elección fue Ubuntu 18.04 server y CentOS 7 server. Una vez conocida la arquitectura de "Elasticsearch", se ha podido *automatizar la preparación de los nodos* mediante un script Shell con toda la configuración recomendada por la documentación. Al mantener estas pautas, permite un buen rendimiento en relación entre el hardware y el software para los productos de Elastic y son los siguientes:

- Solo RAM y CPU, memoria SWAP no recomendada.
- Funciona sobre Java no menos de 8 Gb, y no más de 32 Gb de memoria.
- Crecimiento horizontal en nodos para "Elasticsearch". El objetivo de Elastic es la distribución

[9] Son las URLs de un API o un backend que responden a una petición.
[10] Modelo de instalación, distribución y ejecución de software en ámbito local de una organización o persona.

de la carga.
- Librería lucene de Java para el indexado versión 1.8 lib recomendado.
- 50% RAM destinado como mínimo para el nodo a "Elasticsearch".
- DISCO ver sobre todo tasas de IOPS[11].
- USO DE RAID 0, alterna los datos de un disco al otro (NO HACER REPLICA | ESPEJO).
- No usar almacenamiento de tipo NAS[12], incrementa latencias.

Se ha utilizado CentOS 7 server en los entornos on premise y en las imágenes oficiales para Docker en los 2 modos, *Single node y Cluster*.

Para el entorno Cloud se utiliza Ubuntu 18.04 server par el entorno *cluster*. Pero para el entorno *cluster* en on premise se utilizó CentOS 7 con un máximo de 3 nodos "Elasticsearch" por limitación de recursos.

Microsoft Windows

Microsoft Windows estuvo planteado inicialmente como cliente o "Beats", pero para los usos prácticos en este proyecto finalmente no se ha contemplado.

Docker (Docker file, Docker compose, Docker swarm)

Mediante el fichero de configuración en Yaml (.YML) con Docker compose[13], permite de forma rápida y sencilla implementar los 2 modos, *single node* para desarrollo y en *cluster* para producción.

Para iniciar una configuración hay que tener en cuenta la versión con que trabaja Docker y Docker compose para evitar tener problemas de configuración con los parámetros que utiliza Docker y los parámetros que están preconfigurados en las imágenes base, como variables de entorno. Las configuraciones redundantes para todos los componentes (elasticsearch | kibana | logstash | beats) son las de network que sirve para la conexión entre contenedores y volumen para que permanezcan los datos después de parar un contenedor, el resto son independientes.

Para levantar nodos de "Elastisearch" las configuraciones son redundantes, la misma configuración que utiliza elasticsearch(es01) la utilizaran los siguientes nodos que se vallan a integrar al cluster excepto con excepción del nombre y el volumen. Aunque pueden variar los parámetros de "Elasticsearch" según el rol de tipo de nodo (Master| Data | Ingest| Coordinador) que requiera el caso de uso para el sistema.

```
version: '3.2'
services:
  es01:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.6.0
    container_name: es01
    environment:
      - node.name=es01
      - cluster.name=es-docker-cluster
      - discovery.seed_hosts=es02,es03
      - cluster.initial_master_nodes=es01,es02,es03
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - data01:/usr/share/elasticsearch/data
    ports:
      - 9201:9200
      - 9301:9300
    networks:
      - elasticluster
volumes:
  data01:
    driver: local
  data02:
    driver: local
  data03:
    driver: local
networks:
  elasticluster:
    driver: bridge
```

Para "Logstash" se configuran los volúmenes, en nuestro caso de tipo es bind que sirve para mapear rutas y configurar ficheros desde el lado local y los puertos.

```
lgt01:
    image: docker.elastic.co/logstash/logstash:7.6.2
    container_name: lgt01
    volumes:
    - type: bind
      source: ./logstash/config/logstash.yml
      target: /usr/share/logstash/config/logstash.yml
      read_only: true
    - type: bind
      source: ./logstash/pipeline/logstash.conf
      target: /usr/share/logstash/pipeline/logstash.conf
      read_only: true
    ports:
    - "5000:5000/tcp"
    - "5000:5000/udp"
    - "9600:9600"
    environment:
    - "LS_JAVA_OPTS=-Xms512m -Xmx512m"
    networks:
    - elasticluster
    depends_on:
    - es01
    - es02
    - es03
```

Para" kibana" sobre todo se configuran los puertos y con qué nodos de "Elasticsearch" se establecerá conexión.

```
kib01:
  image: docker.elastic.co/kibana/kibana:7.6.0
  container_name: kib01
  ports:
    - 5601:5601
  environment:
    ELASTICSEARCH_URL: http://es01:9200
```

[11] unidad de benchmark usada para medir el rendimiento de dispositivos informáticos de almacenamiento.

[12] Almacenamiento en Red.

[13] Herramienta que permite definir y ejecutar múltiples contenedores, la configuración está definida en .yml.

ELASTICSEARCH_HOSTS: http://es01:9200
networks:
- elasticluster

Para "Beats" la configuración va en relación del agente (Auditbeat|Filebeat|Functionbeat|Heartbeat |Journalbeat |Metricbeat |Packetbeat | Winlogbeat) del que se va a ser uso.

Cloud

Se ha recurrido al servicio de Amazon web service (AWS) educate [11], sobre todo para montar el *modo cluster* de 9 nodos de "Elasticsearch" (3 Master|6 Data). Para montar nuestro caso de uso se necesita crear y configurar instancias donde se hará en función de los recursos que se requiera disponer, ya que en cloud todo recurso es facturable en función del uso que se le otorgue al recurso.

Crear y configurar VPC[14] .

1. existe una red por defecto
2. crea una nueva red:
   * elegimos una región AWS
   * create VPC, p.e:
     - Name tag : entornoCloudElastic
     - IPv4 CIDR block : 192.168.0.0/16
     - IPv6 CIDR block : No
     - Tenancy : default
3. crea una subred en base a la red creada :
   * create subnets
     - Name tag : entornoCloudElastic-prod-subnet-a
     - VPC* : vpc-0915376a9cd517904 (entornoCloudElastic)
     - Availability Zone: us-east-1a
     - IPv4 CIDR block* : 192.168.1.0/24 (251 IPs)
4. crear salida a internet
   * Create internet gateway
     - Name tag : entornoCloudElastic-prod-i
   * detached a nuestra VPC (solo puede 1 VPC a la vez, botón derecho y attach)
     - VPC* : vpc-0915376a9cd517904
5. ver tablas de ruta
   + subnet, selecciona la subred y debajo el tab Route Table
     - vemos la red a donde va dirigido toda la ruta
6. asignar en la Route Table a nuestra VPCs
   - seleccionamos la VPCs y le damos un name : entornoCloudElastic-prod-pub
   - seleccionamos la VPCs (vpc-0915376a9cd517904 | entornoCloudElastic-prod)
     - tab routes, edit routes
       - add route
         - 0.0.0.0/0 : todo internet
         - igw-03d8ac2dff6db034f : internet gateways
   - Asociar en el tab subnet associations
     - edit subnet associations : check
7. asignar DNS hostname a mi instancia
   - dentro de VPCs en la red creada hacer clic botón derecho: DNS hostname enable (activar)
8. Modify auto-assign IP settings en la subred
   - dentro de subnets, en la subred creada clic derecho Modify auto-assign IP settings
     - Auto-assign IPv4 : Enable auto-assign public IPv4 address (activar).

Configurar un Segurity group[15].

1. Por tipo de Instancia en la Consola de administración de AWS
* informática -> EC2 -> network & segurity -> Security Group
  + clic en Create security group
    - Security group name : sg-elastic
    - Description : elastic
    - VPC : entornoCloudElastic-prod
    - Inbound rules
      - ssh :: Anywhere

Crear un crear un key pairs[16]

Crear una y guardarla en local la private key
conexión por ssh a las instancia -> key pairs

Una realizado todo lo necesario para montar nuestro entorno en AWS, ya se pueden iniciar las configurar las instancias necesarias para nuestro sistema e iniciar las instalaciones.

3.3 Instalaciones

Una vez conocida la arquitectura y preparado el entorno, podemos implementar los servicios del Elastic según el modo requerido (nodo single | cluster) .

Una vez instalado el software desde los repositorios para Debian o Redhat, se realiza la instalación directamente sobre los paquetes Package Manager (RPM) o Debian GNU/Linux y derivadas (DEB). Este proceso de instalación es recurrente para todos productos de la Elastic Stack (Elasticsearch | Kibana| Logstash| beats). En cambio, para las configuraciones cada componente software requiere de una especifica configuración.

Elasticsearch

Para los 2 modos requiere una configuración sobre el fichero de configuración elasticsearch.yml (.YML) .

Para un *single node* la configuración en on premise y docker se utilizó la siguiente:

cluster.name: ELK
node.name: ${HOSTNAME}
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch
network.host: 127.0.0.1
http.host: 0.0.0.0

Para una configuración en cluster requiere algunos parámetros más al detalle en función del caso de uso. Para nuestra configuración en *Cluster* sobre AWS se utilizó la siguiente:

cluster.name: clusteraws
node.name: ip-192-168-1-92.ec2.internal
node.master: true
node.data: true
http.cors.enabled: true
http.cors.allow-origin: "*"
http.cors.allow-methods: OPTIONS, HEAD, GET, POST, PUT, DELETE
http.cors.allow-headers: X-Requested-With, X-Auth-Token, Content-Type, Content-Length, kbn-version, Origin,
path.data: /var/lib/elasticsearch

[14] Virtual private cloud, termino para denominar una red privada dentro de un entorno cloud.

[15] Reglas de acceso a una instancia tanto de entra como de salida a nivel de capa de transporte.

[16] Configuración de pares de llaves, clave publica y privada.

```
path.logs: /var/log/elasticsearch
bootstrap.memory_lock: false
network.host: 0.0.0.0
http.port: 9200
discovery.seed_hosts:
["192.168.1.92:9300","192.168.1.84:9300","192.168.1.53:9300","192.168.1.
45:9300","192.168.1.114:9300","192.168.1.37:9300","192.168.1.67:9300","
192.168.1.63:9300","192.168.1.35:9300"]
cluster.initial_master_nodes:
  - ip-192-168-1-92.ec2.internal
  - ip-192-168-1-84.ec2.internal
  - ip-192-168-1-53.ec2.internal
```

kibana

Se puede mejorar su uso, configurando el sistema de manera que se va implementando el rol del Nodo coordinador juntamente con kibana en la misma instancia, esto permite más rendimiento a la UI para consultas de búsqueda, pero para nuestros casos de uso requerimos la siguiente:

```
server.host: "192.168.0.21"
elasticsearch.hosts:
        - http://192.168.0.20:9200
        - http://192.168.0.21:9200
        - http://192.168.0.22:9200
logging.dest: /var/log/kibana/kibana.log
```

Logstash

Es posible ampliar este proceso de la pila si lo utilizamos como un *Cluster*, esto nos permite realizar un balance de carga, porque si observamos en la Fig. 2, existe un posible *cuello de botella* en este proceso entre la ingesta de "Beats" y "Logstash", para solucionar este posible problema se recurre a gestores de colas (REDIS | KAFKA) para ser implementado entre "Beats" y "Logstash" para que permita la distribución y evitar el atollamiento del envió de datos.

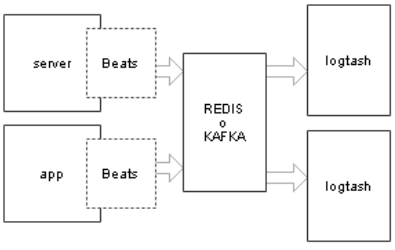


Fig. 4. Estructura para balanceo de carga en Logstash haciendo uso de gestores de colas.

En nuestros casos de uso, no generamos grandes volúmenes de eventos al segundo, por ello no es necesario implementar este modelo, pero es importante conocer que un sistema de análisis de datos genera miles de eventos al segundo y optimizar el funcionamiento es posible.

Logstash al ser el proceso de parseo o Extract, Transform and Load (ETL), se configuran 3 pasos en el fichero de configuración (.conf).

Input

Proceso donde se configura el origen de la ingesta de datos. En los casos de uso planteados la configuración proviene de 3 fuentes de origen, para el primer caso de uso "Beats", recoge los datos provenientes del agente "Filebeat" y para segundo caso de uso el plugin *file* hace lectura de algún directorio en concreto y *http_poller* para coger los datos desde una API rest publica.

```
beats {
  port => 5044
 }

file{
  path => "/etc/logstash/conf.d/UID_ISO_FIPS_LookUp.csv"
  start_position => "beginning"
 }

http_poller {
  urls => {
   test => {
    # Supports all options supported by ruby's Manticore HTTP
client
    method => get
    url => "https://coronavirus-19-
api.herokuapp.com/countries"
    headers => {
     Accept => "application/json"
    }
   }
  }
  request_timeout => 60
  # Supports "cron", "every", "at" and "in" schedules by rufus
scheduler
  schedule => { cron => "* * * * * UTC"}
  codec => "json"
  # A hash of request metadata info (timing, response headers,
etc.) will be sent here
  metadata_target => "http_poller_metadata"
 }
```

Filter

Es el proceso de transformación y procesado de los datos, funciona a travez de plugins para realizar alguna tarea en concreto, dentro de los plugins más utilizados tenemos *mutate*[17], *grok*[18], *geoip*[19], en nuestros casos de uso utilizamos los plugins *csv, mutate, y geoip grok*.

```
csv {
    columns => [
"UID","iso2","iso3","code3","FIPS","Admin2","Province_State","Count
ry_Region","Lat","Long_","Combined_Key","Population"]
    }
mutate {
    add_field => {"locations" => "%{Lat},%{Long_}" }
    }
grok {
    match => { "message" => "%{IP:ip} - - \[%{HTTPDATE:data}\]
\"%{WORD:request} %{URIPATHPARAM:uriRequest}
HTTP/%{NUMBER:httpversion}\" %{NUMBER:response}
(?:%{NUMBER:bytes}|-) \"%{URI:url}\" %{QS:agent}" }
```

[17] Permite añadir, borrar, modificar los campos en los eventos.

[18] Permite descomponer un evento de una manera más compacta y legible.

[19] agrega información sobre la ubicación geográfica de las direcciones IP, en función de los datos de las bases de datos Maxmind GeoLite2

```
}
geoip {
  source  => "ip"
}
```

Output

Sirve para configurar la salida de los datos cargados y modificados por los anteriores procesos, se registra el template asignado y el destino del nodo "Elastisearch" para ser indexados los datos.

Logstash debe asignar el template con los parámetros necesarios para ser indexada la información por "Elasticseach", este proceso cubre 3 etapas, los "settings" que permiten dar el número de shards y replicas para el índice, los "mappings" que configura el tipo de dato a ser indexado y los "analizadores" para que generan tokens que se utilizan para realizar consultas de búsquedas de los documentos indexados, el tercer paso se aplica cuando se realizan consultas desde sus EndPoints.

Beats

Al ser agentes instalados en los nodos que van a generar los datos, las configuraciones van relacionadas según el tipo de caso de uso para un agente en concreto. Para nuestro caso utilizamos Filebeat.

```
filebeat.inputs:
- type: log
paths:
    - /root/Fake-Apache-Log-Generator/*.log
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.1.14:5044"]
```

No se realiza ningún preprocesado dentro del agente ya que no da la opción para realizar cambios necesarios en los datos para nuestro caso de uso, para ello enviamos los datos a "Logstash" para su procesamiento y transformación antes de ser indexados a "Elastisearch". Los agentes Beats pueden realizar transformación de los datos antes de ser enviados, pero no es recomendable porque realizaría consumos altos de CPU RAM para la transformación dentro del sistema que genera los datos y además no son transformación completas como lo realiza "Logstash".

3.4 Desarrollo

Este punto está orientado al control y visualización del funcionamiento del sistema en cualquiera de los 2 modos *Single node o Clúster*, ya que en el Elastic los nodos de las instancias "Elasticsearch" dentro de sus características de escalabilidad, tolerante a fallos y alta disponibilidad cuenta con una implementación en su arquitectura con la que dispone y es efectiva, pero tiene algunos inconvenientes. Debido a que distribuye los índices en shards y replicas, es posible perder información por la caída de uno de sus nodos, aunque el servicio de la aplicación podría seguir funcionando y dar la sensación falsa de que no ha ocurrido nada y por ello es preciso monitorizar el servicio.

"Elasticserach" proporciona una serie de EndPoints sobre API REST que permiten visualizar el estado de todo el sistema tanto el clúster en general, como sus nodos que lo componen y sus índices. Al contener esta información nos permite enfocar estos datos en una SPA (single page application).

El desarrollo es con Vuejs que nos permite modularizar y trasladar esta información distribuida a la aplicación en componentes y como finalidad otorgar una UI de control y monitorización.

La estructura de la aplicación se genera por el gestor CLI de vuejs que nos da la flexibilidad de implementar la estructura de un proyecto en Vue de manera rápida y cómoda.

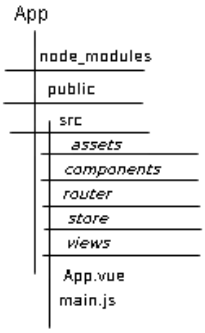


Fig. 5. Vue CLI gestor para la implementación de proyectos.

El desarrollo de la aplicación utiliza el modelo DOM virtual, que nos permite renderizar solo los últimos cambios realizados, esto es útil para nuestro objetivo de monitorizar cualquier cambio en el cluster sin necesidad de tener que actualizar en todo momento toda la aplicación.

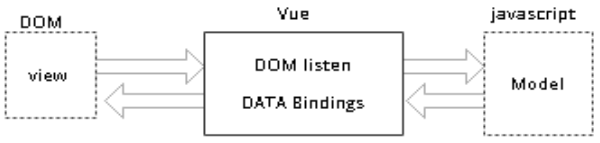


Fig. 6. Modelo Virtual DOM de vue.

Para la monitorización de nuestro sistema se hace uso de las siguientes EndPoints de Elasticsearch:

1. GET _cluster/health/<index>
2. GET _nodes/stats/fs?human
3. GET _cat/nodes?format=json&pretty=true
4. GET _cat/shards?format=json&pretty=true
5. GET _nodes/stats/fs?human
6. GET _search_shards?pretty

Tambien para monitorizar Logstash utilizamos su API "GET _node/stats?pretty"

La herramienta está distribuida en 2 vistas, la primera vista es para el logear al *Cluster* para ser monitorizado y si se requiere también "Logstash"(opcional). Esta vista cuenta con 2 controles internos, *el primer control* para poder acceder debe ser una URL valida, donde este campo del input es controlado con expresiones regulares y el

segundo controla si el servidor cluster está conectado y espera que devuelve el estado de "200" hacia una petición GET. El campo para "Logstash" también cuenta con el mismo proceso de control para el acceso a su API.

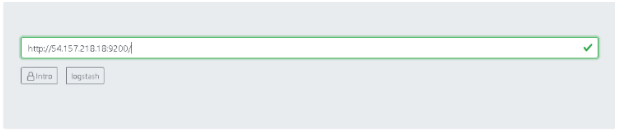


Fig. 7. Login con la url del cluster para acceder a los EndPoints de elasticsearch.

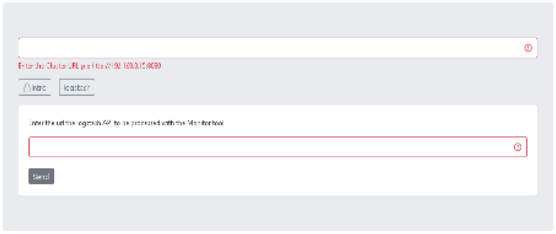


Fig. 8. Login con la url con logstash.

La segunda vista está orientada a la monitorización y está dividida en 2 partes, *la primera* del navbar para las opciones de la aplicación (Overview| Logstash | Cluster |Refresh | Reload | Logout), *la segunda* está compuesta por los componentes internos e independientes para la monitorización del cluster, los índices y los nodos.

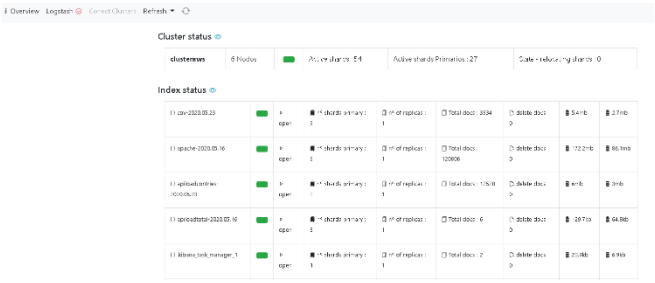


Fig. 9 Vista de la monitorización Cluster y sus índices.

Fig. 10 Vista de la monitorización del cluster y sus nodos.

3.5 Documentación

Durante el proceso de implementación y desarrollo, la documentación ha sido punto necesario y no menos importante, ya que todo el proceso esta descrito tanto para la parte evaluadora como para sus posibles usos a futuro. La guía de evaluación del TFG ya describe una serie de fases dentro de la documentación a desarrollar. Toda la documentación esta centralizada en un dossier como se observa en la Fig. 11, dentro del informe contiene todas las configuraciones realizadas para este proyecto y una guía general en Markdown (.MD).

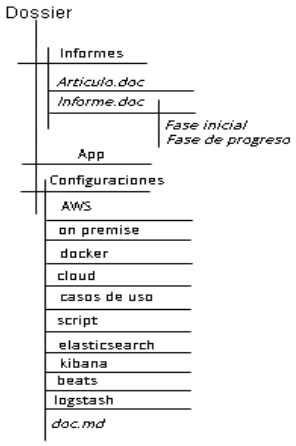


Fig. 11 Estructura para el dossier del TFG.

4 METODOLOGÍA

Para el seguimiento y control de la implementación y desarrollo dentro de un entorno "Ágil", se ha establecido el método Kanban [12]. Junto a esta metodología se ha acompañado en una serie de 3 procesos de control para la planificación de los requisitos y tareas establecidas inicialmente en los objetivos principales. De esta manera simplificamos el seguimiento ya que todas las herramientas se pueden integrar entre ellas y facilitar así la gestion de los datos.

Por tanto, los procesos de control están establecido en 3 etapas, como se observa en la Fig. 12.

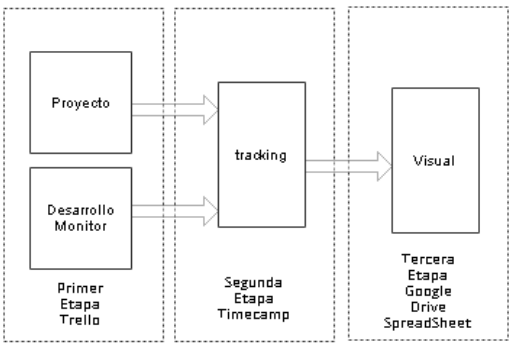


Fig. 12. Proceso de control para el seguimiento temporal de los requisitos y tareas.

El proyecto está establecido para la implementación de elastic stack y el desarrollo de la aplicación monitor, por ello dentro para la *primera etapa* se establece 2 tableros, con la aplicación Trello [13].

El *primer proceso* consiste en establecer los 2 tableros que están compuesto por 4 estados (Para hacer | Desarrollo | Prueba | Hecho), esto nos permite tener un control de toda la implementación y conocer la situación de cada

requisito dentro de la implementación o desarrollo (Apéndice A.1 Tableros Trello).

El *segundo proceso* consiste en contabilizar el tiempo dedicado a cada requisito con la herramienta Time Camp [14], porque para la planificación se estableció un tiempo determinado a cada requisito. En el contenido del Dossier se adjuntará un resumen del control en csv de los requisitos (Apéndice A.2 TimeCamp).

Finalmente, el *tercer proceso* consiste en implementar una plantilla en Google drive con spreadsheet donde se importa los datos generados por las aplicaciones Trello y Time Camp. Aquí tenemos un control automatizado con una plantilla generada que cubra todas las necesidades para establecer un control completo de nuestro proyecto (Apéndice A.3 Google SpreadSheed).

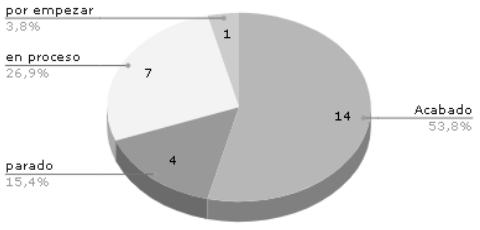


Fig. 13 Estado de Progreso de los requisitos durante el desarrollo e implementación.

De esta manera hemos podido generar métricas para evaluar el estado del proyecto durante las fases de progreso para la documentación del progreso del proyecto.

4.1 Requisitos

Al inicio de la implementación del proyecto se establecieron unos requisitos en función de los 5 objetivos principales. Aquí se establecieron las pautas necesarias para poder organizar y establecer lo que se necesitaba para el sistema pudiese estar en marcha (Apéndice A.4 Requisitos).

4.2 Restricciones

Este proyecto está orientado a la implementación del sistema "Elastic Stack" bajo diversos entornos, los cuales se implementan en función del caso de uso o requerimientos de una empresa o particular y así poder dar uso a este sistema teniendo en cuenta sus necesidades. Por tal motivo los casos de uso implementados en este proyecto son orientativos y demostrativos para ver las funcionalidades del sistema, acompañado del desarrollo de una herramienta para su monitorización.

El proyecto total está establecido para un total de 260 horas de las cuales sobre dimensionamos un 10% en la propuesta inicial, esto se realizó de esta manera para poder asegurar el desarrollo del proyecto en general dentro de los plazos establecidos.

4.3 Control de versiones

Se hace uso de Gitlab como gestor de versiones para todo el proyecto, pero no se implementa un sistema de tipo Git Flow, básicamente porque está orientado a un sistema de versiones de trabajo en equipo. Pero es necesario usarlo para el control de versiones en el desarrollo del monitor del clúster y sobre todo para ir manteniendo las configuraciones actualizadas y recopilación de la documentación de toda la implementación.

5 Resultados

Se ha podido alcanzar los objetivos marcados en la fase inicial dentro del periodo planificado.

Para la verificación del funcionamiento de todo el proyecto se propuso 2 casos de uso de los cuales sus configuraciones han sido comentadas durante este artículo en el apartado de Objetivos pag.3. Se ha podido implementar en los 2 modos *Single node y Cluster* para los entornos de on premise, docker y Cloud donde finalmente se llegó a desarrollar la herramienta *Monitor* para hacer un seguimiento del funcionamiento del sistema testeando los casos de uso que mencionamos a continuación.

5.1 Caso de Uso Fake Apache Log Generator

Fake Apache Log generator [15] es un software desarrollado en Python que permite generar logs similares a los generados por un servidor apache, permitiendo así ser procesados por el sistema y dar el caso de una simulación de un servidor en funcionamiento.

81.75.130.8 - - [12/Apr/2020:10:34:24 +0200] "GET /wp-admin HTTP/1.0" 200 5022 "http://hunt.com/login.php" "Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10_5_3; rv:1.9.2.20) Gecko/2011-01-16 23:38:40 Firefox/3.6.3".

Los datos fueron generados desde "Filebeat" para ser procesados por "Logstash" y pasar de estar indexados en un solo campo o Key(JSON) a ser distribuidos en diversos campos, esto es necesario para su respectivo análisis por Kibana.

%{IP:ip} - - \[%{HTTPDATE:data}\] \"%{WORD:request} %{URIPATHPARAM:uriRequest} HTTP/%{NUMBER:httpversion}\" %{NUMBER:response} (?:%{NUMBER:bytes}|-) \"%{URI:url}\" %{QS:agent}

Además, también generamos el campo geoip para poder ser localizadas la conexiones y respectivamente analizadas (Apéndice A.6 Mapa GeoIP).

5.2 Caso de Uso COVID-19

Actualmente a fecha en la que se redacta este artículo se está viviendo una pandemia global [16] y para poder hacer estudios y tomas de decisiones de las autoridades pertinentes con todos los datos, existen repositorios que están centralizando toda la información generada por la pandemia. Por esto hacemos uso también de este sistema que nos permita recopilar estos datos de sus repositorios oficiales y ser procesados.

Del repositorio [17] publico en GitHub de The Center for Systems Science and Engineering (CSSE) is a research collective housed within the Department of Civil and Systems Engineering (CaSE) at Johns Hopkins University (JHU) [18] (Apéndice A.6 HeapMap de casos Confirmados COVID-19).

A partir de una API [19] publica con licencia de tipo MITcreada por NovelCOVID [20] (Apéndice A.7 Tabla de estado del COVID-19).

Finalmente, los índices generados en ambos casos de

uso fueron monitorizados desde la herramienta monitor para controlar la fiabilidad de nuestro cluster.

6 CONCLUSIONES

Este proyecto ha permitido hacer uso de tecnologías actuales para implementar el sistema y poder preparar una herramienta para diversos entornos informáticos y empresariales.

Dentro de los aspectos importantes de manera personal, este proyecto me ha permitido explorar muchas tecnologías y a destacar un pequeña experiencia necesaria para la vida laboral en un futuro.

Por el lado del software utilizado, Elastic como producto cuenta cada vez más con una serie de soluciones interesantes que nos permitan generar más valor al estudio de los datos.

7 AGRADECIMIENTOS

Finalizando este articulo quiero agradecer a mi tutor Rafael Fernández González por la orientación y adaptación dentro de las circunstancias que han surgido actualmente en el ámbito académico y facilitarme el desarrollo de este proyecto.

Y a mi familia que, aunque no esté a mi lado siempre cuento ellos, mis amigos y compañeros demostrando interes en las etapas de mi vida.

REFERENCIAS

[1] Elastic, «elastic.co,» Shay Banon, 8 08 2010. [En línea]. Available: https://www.elastic.co/es/about/history-of-elasticsearch. [Último acceso: 12 01 2020].

[2] vuejs, «vuejs.org,» Evan You, 11 feb 2014. [En línea]. Available: https://es.vuejs.org/v2/guide/index.html#. [Último acceso: 28 feb 2020].

[3] Apache Hadoop, «hadoop.apache.org/,» Apache Software Foundation, 01 Abr 2006. [En línea]. Available: https://hadoop.apache.org/docs/stable/. [Último acceso: 07 Mar 2020].

[4] C. B. Kumar Saurabh, «sumologic.com,» Computer software, Security management, Enterprise software, 01 Ene 2010. [En línea]. Available: https://www.sumologic.com/. [Último acceso: 10 Jun 2020].

[5] E. S. R. D. Michael Baum, «https://www.splunk.com/,» splunk, 2003. [En línea]. Available: https://docs.splunk.com/Documentation. [Último acceso: 07 Mar 2020].

[6] L. Koopmann, «graylog.org/,» Enterprise software Opensource software, 2009. [En línea]. Available: http://docs.graylog.org/en/3.2/index.html. [Último acceso: 07 Mar 2020].

[7] https://www.sas.com/es_es/company-information.html, «sas.com,» [En línea]. Available: https://www.sas.com/es_es/insights/big-data/hadoop.html. [Último acceso: 11 Jun 2020].

[8] Google, «https://trends.google.es/trends/?geo=ES,» google, 4 Sep 1998. [En línea]. Available: https://trends.google.es/trends/explore?date=2004-07-03%202020-03-07&geo=ES&q=elasticsearch,loggly,splunk,graylog,Haddop. [Último acceso: 7 Mar 2020].

[9] J. S. y. J. Atwood, «stackoverflow,» Stack Exchange, Inc., 15 Sep 2008. [En línea]. Available: https://insights.stackoverflow.com/survey/2019#technology. [Último acceso: 11 Jun 2020].

[10] Vuejs, «vuejs.org,» Evan You, 11 Feb 2014. [En línea]. Available: https://es.vuejs.org/v2/guide/render-function.html#El-DOM-Virtual. [Último acceso: 11 Jun 2020].

[11] «AWS Educate,» Amazon inc, 5 Jul 1994. [En línea]. Available: https://aws.amazon.com/es/education/awseducate/. [Último acceso: 22 May 2020].

[12] BusinessMap , «https://kanbanize.com/,» kanbanize.com, 01 Ene 2008 . [En línea]. Available: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban/. [Último acceso: 03 Mar 2020].

[13] J. Spolsky, «Trello,» Atlassian, 13 Sep 2011. [En línea]. Available: https://trello.com/. [Último acceso: 30 Mar 2020].

[14] K. Rudnicki, «timecamp,» TimeCamp Inc, 01 Ene 2009. [En línea]. Available: https://app.timecamp.com/auth/login#/timesheets/timer. [Último acceso: 30 Mar 2020].

[15] Repositorio, «Repositorio,» kiritbasu, 15 Mar 2018. [En línea]. Available: https://github.com/kiritbasu/Fake-Apache-Log-Generator. [Último acceso: 11 Abr 2020].

[16] «Organización Mundial de la Salud (OMS),» 07 Abr 1948. [En línea]. Available: https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019. [Último acceso: 12 May 2020].

[17] CSSEGISandData, «COVID-19,» [En línea]. Available: https://github.com/CSSEGISandData/COVID-19. [Último acceso: 15 May 2020].

[18] The Johns Hopkins University, 22 Feb 1876. [En línea]. Available: https://systems.jhu.edu/. [Último acceso: 12 May 2020].

[19] J. Aviles, «forked from NovelCOVID/API,» [En línea]. Available: https://github.com/javieraviles/covidAPI. [Último acceso: 13 May 2020].

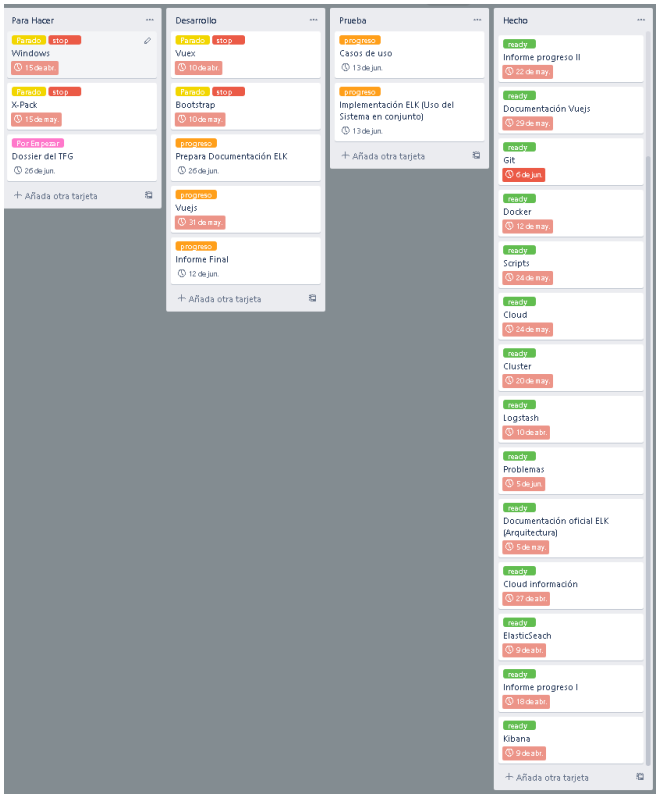
[20] NovelCOVID, «NovelCOVID,» [En línea]. Available: https://disease.sh/docs/. [Último acceso: 12 May 2020].

APÉNDICE

A.1 Tableros Trello

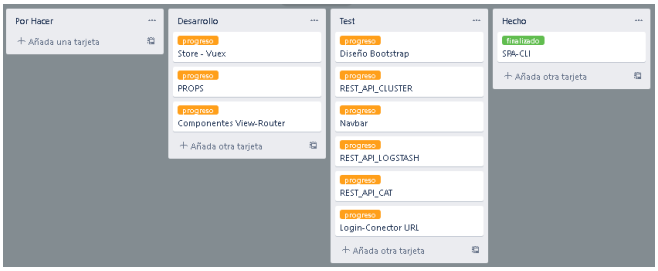
A.1.1 Tablero para la implementación del sistema

Nos permite establecer los estados de los requisitos que están en 4 etapas como lo estable la metodología Kanban y tener una perspectiva global del estado de todo el sistema.



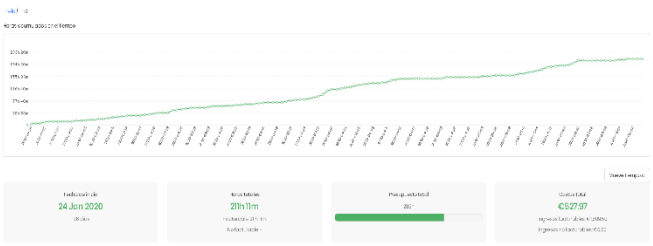
A. 1.2 Tablero para la implementación de la herramienta monitor

El siguiente tablero está establecido para los requisitos que contendrán el proceso de desarrollo de la herramienta monitor.

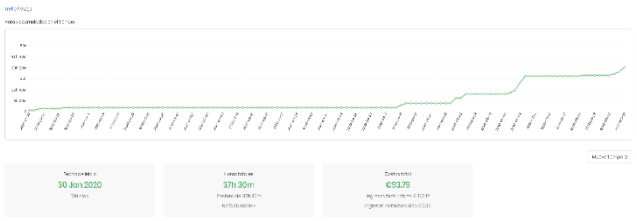


A.2 TimeCamp

Esta herramienta contiene más opciones además de contabilizar el tiempo por tareas o requisitos como establecer costes y generar métricas para cualquier tipo de proyecto. Para un uso práctico se establecido una cuota a 8 € por hora y con un 25% porciento dedicado a impuestos directos e indirectos.

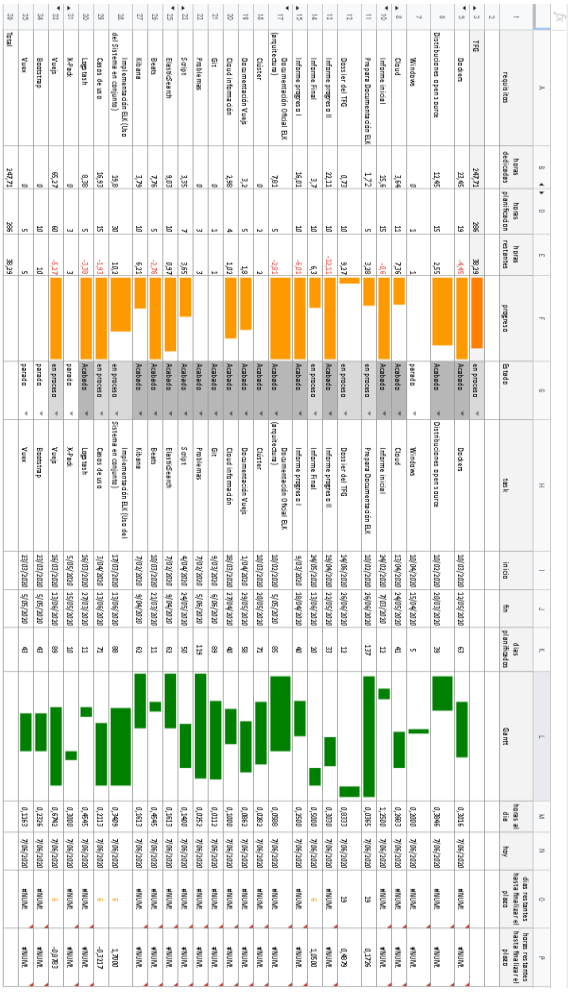


Generamos 2 graficas ya que contamos con 2 tableros para en trello para contabilizar el tiempo de la implementación y el desarrollo.



A.3 Google SpreadSheed

Creamos una plantilla para establecer un control automatizado y poner tener una visión global del estado del proyecto, como con barras de visualización de progreso, una columna Gantt y tablas dinámicas de inicio y fin.



Esta plantilla cubre los siguientes aspectos:

- Horas dedicadas, son horas contabilizadas hasta el momento.
- Horas planificación, son horas totales establecidas como necesarias para completar ese requisito.
- Horas restantes, son horas restantes establecidas como necesarias para completar ese requisito.
- Progreso, es una barra de métrica para observar la evolución.
- Estado, es una métrica de situación o circunstancia de cada requisito.
- Data inicio, es fecha de planificación de inicio.
- Data fin, es fecha de planificación de final.
- Días planificados, va ligado a horas al día.
- Gantt, métrica donde figura el estado de inicio y final entre todos los requisitos.
- Horas al día, tiempo necesario al día entre cuando empieza y debería estar lista el requisito.
- días restantes, días que necesarios antes de que se cumpla el plazo para estar listo el requisito.
- Horas restantes, horas necesarias restantes en función de los días restantes.

T	U	W	X
requisito	inicio	*requisito*	fin
	43868	Informe inicial	7/03/2020
ElasticSearch	7/02/2020	Distribuciones open source	20/03/2020
Problemas	7/02/2020	Logstash	27/03/2020
Documentación Oficial ELK (arquitectura)	10/02/2020	ElasticSearch	9/04/2020
Prepara Documentación ELK	10/02/2020	Kibana	9/04/2020
Distribuciones open source	10/02/2020	Windows	15/04/2020
Git	9/03/2020	Cloud información	27/04/2020
Informe progreso I	9/03/2020	Bootstrap	5/05/2020
Beats	10/03/2020	Documentación Oficial ELK (arquitectura)	5/05/2020
Clúster	10/03/2020	Vuex	5/05/2020
Dockers	10/03/2020	Dockers	12/05/2020
Logstash	16/03/2020	X-Pack	15/05/2020
Implementación ELK (Uso del Sistema en conjunto)	17/03/2020	Informe progreso II	22/05/2020
Cloud información	18/03/2020	Cloud	24/05/2020
Bootstrap	23/03/2020	Script	24/05/2020
Vuex	23/03/2020	Documentación Vuejs	29/05/2020
Documentación Vuejs	1/04/2020	Problemas	5/06/2020
Casos de uso	3/04/2020	Git	6/06/2020
Script	4/04/2020	Casos de uso	13/06/2020
Cloud	13/04/2020	Informe Final	13/06/2020
Informe progreso II	19/04/2020	Vuejs	13/06/2020
X-Pack	5/05/2020	Dossier del TFG	26/06/2020
Informe Final	24/05/2020	Prepara Documentación ELK	26/06/2020
Dossier del TFG	14/06/2020		44010

A.4 Requisitos

Id	Nombre	Finalidad	Descripción	Precondición
R.T 01	Documentación Oficial ELK	documentación	Elastic, documentación oficial, recursos.	
R.T 02	Documentación Vuejs	documentación	Documentación oficial y recursos de la comunidad, para conocer en profundidad el Framework, de manera que se pueda desarrollar en modo CLI o CDN.	
R.T 03	Problemas	documentación	Documentación de la gestion de los posibles problemas que se presenten durante la implementación de todo el TFG y sus recursos. Es importante este apartado ya que permitirá conocer cualquier inconveniente que surja y la razón por la que el proyecto pueda estancarse o tenga posibles modificaciones, debido a imprevistos inesperados propios de un proyecto de implementación y desarrollo.	
R.T 04	Script - Bash – Node – Python	implementar – desarrollar - documentación	como parte del correcto funcionamiento es necesario comprobar la recogida y recolección de datos que realizaran los agentes beats, pero al no tener sistemas en producción, otra manera de generar datos es con scripts propios o de terceros. Además, es posible automatizar la implementación de un servidor o nodo en un entorno onpremise, ya que se ira añadiendo nodos a un clúster para su crecimiento horizontal, hasta un punto este proceso es redundante y es más apropiado que se realice esta tarea con un script de manera	R.T 01 - R.T 12

			automática.	
R.T 05	Docker	documentación	contar con la infraestructura necesaria para implementar un clúster requiere de un procesamiento de hardware importante, por tanto, al ser este proyecto más un entorno de pruebas y testeo, por motivos de costes, una opción interesante es utilizar contenedores Docker file y Docker compose[20], ya que a diferencia de la virtualización nos dan más flexibilidad debido a que su arquitectura nos permite saltarnos la capa del hipervisor, presente en la virtualización de Hardware, la cual sería otra opción. Además, esto no descarta que en un entorno empresarial se requiera una implementación por contenedores en cloud.	R.T 01
R.T 06	Cloud	documentación	Posibles entornos para la implementación del stack, pero es conveniente antes entender cómo opera el sistema de instancia miento en internet, tanto por costes y operabilidad.	R.T 01
R.T 07	Clúster	documentación	El stack ELK puede ser implementado de 2 maneras y todo es en función del uso que se le quiera dar, puede ser single node[21] o como clúster y es la segunda la que nos da concretamente las características de escalabilidad horizontal, tolerante a fallos y alta disponibilidad.	R.T 01
R.T 08	Distribuciones Linux	implementar	CentOS y Ubuntu en nuestro caso, precisamente estas 2 distribuciones cuentan con una serie de características adecuadas, como orientado al entorno empresarial, estabilidad y amplia comunidad.	R.T 01
R.T 09	Microsoft Windows	implementar	aunque no se dará uso para la implementación del propio clúster, si es conveniente tenerlo en cuenta como cliente para la extracción de datos.	
R.T 10	Docker	implementar	(Docker file, Docker compose, Docker swarm), como anteriormente se menciona en el documento, es importante para implementación de esta tecnología para hacer uso del clúster en un entorno onpremise, por temas de costes infraestructura. Aunque una manera de escalar un entorno on premise podría ser con Docker swarm.	R.T 01 - R.T 05 - R.T 08
R.T 11	Cloud	implementar	AWS, GCP, Azure, en el mercado existen muchas alternativas y estas serán algunas a evaluar para sus posibles implementaciones bajo estos entornos.	R.T 01 - R.T 06 - R.T 08
R.T 12	Elasticsearch	documentación - implementar	Documentación para su implementación en función del entorno a imple-	R.T 01 - R.T 08

[20] Herramienta que permite definir y ejecutar múltiples contenedores, la configuración esta definida en .yaml.

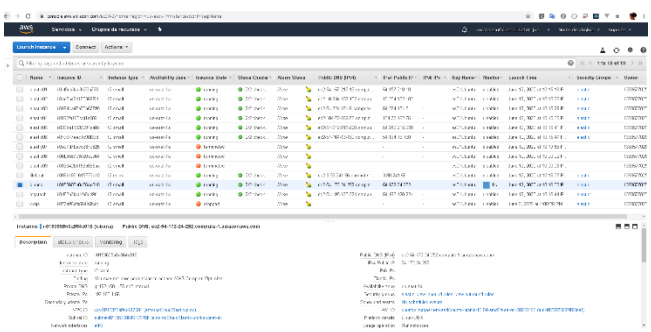
[21] Hace referencia a un solo nodo dentro de implementación ELK, el sistema puede operar con un solo nodo, ya que teóricamente un clúster está basado *desde* un solo nodo, hasta su escalabilidad a n nodos.

			mentar y su compleja configuración de uso.	
R.T 13	Kibana	documentación - implementar	Documentación para su implementación en función del entorno a implementar y su compleja configuración de uso.	R.T 01 - R.T 08
R.T 14	Beats	documentación - implementar	Documentación para su implementación en función del entorno a implementar y su compleja configuración de uso. Testeo de: Auditbeat, información de un sistema Unix. Winlogbeat, información de un sistema Windows. Packetbeat, Información de la red. Filebeat, información de registros logs.	R.T 01 - R.T 08
R.T 15	Logstash	documentación - implementar	Documentación para su implementación en función del entorno a implementar y su compleja configuración de uso.	R.T 01 - R.T 08
R.T 16	X-Pack	documentación	Documentación para su implementación, ya que es una extensión del stack que proporciona otras capacidades como aprendizaje automatico (machine learning), módulo de seguridad entre otras, pero estas ya incluyen de una suscripción al servicio de pago.	R.T 01
R.T 17	Implementación ELK	documentación - implementar	Uso y procesado de la información del stack, para ello como requisito debe de tener instalado y configurado los módulos necesarios del de la pila, como mínimo 2 de ellos, elasticsearch (solo 1 nodo) y kibana.	R.T 01 - R.T 12 - R.T 13 - R.T 14 - R.T 15 - R.T 18
R.T 18	Casos de uso	documentación - implementar	el testeo de la implementación de todo el stack.	R.T 01 - R.T 12 - R.T 13 - R.T 14 - R.T 15 - R.T 17
R.T 19	Documentación Oficial ELK (arquitectura)	documentación	Durante el proceso de implementación y desarrollo, es necesario documentar cada paso para su posterior documentación oficial de todo TFG.	
R.T 20	Informe Inicial	documentación	Redacción y preparación para su entrega programada.	
R.T 21	Informe de progreso I	documentación	Redacción y preparación para su entrega programada.	R.T 20
R.T 22	Informe de progreso II	documentación	Redacción y preparación para su entrega programada.	R.T 20
R.T 23	Informe Final	documentación	Redacción y preparación para su entrega programada.	R.T 20 - R.T 21 - R.T 22
R.T 24	Dossier del TFG	documentación	Redacción y preparación para su entrega programada.	R.T 20 - R.T 21 - R.T 22 - R.T 23
R.T 25	Metodología	documentación	hacer uso de la metodología planificada.	
R.T 26	Vuejs	documentación - desarrollar	Hacer uso del framework, basado en JS (JavaScript), además hacer uso de su extensa documentación.	R.T 01 - R.T 02 - R.T 12 - R.T 15
R.T 27	Vuex	documentación - desarrollar	librería complementaria para la conexión entre módulos dentro de la arquitectura vuejs. Sera necesaria utilizar para la conexión entre componentes desarrollados.	R.T 26

			Esto permite utilizar datos de manera global.	
R.T 28	Bootstrap	documentación - desarrollar	biblioteca de código abierto para el diseño de aplicativo.	R.T 26
R.T 29	REST_API_CAT	desarrollar		R.T 02 - R.T 12 - R.T 37 - R.T 36
R.T 30	REST_API_CLUSTER	desarrollar		R.T 02 - R.T 12 - R.T 37 - R.T 36
R.T 31	REST_API_LOGTASH	desarrollar		R.T 02 - R.T 12 - R.T 37 - R.T 36
R.T 32	PROPS	desarrollar		R.T 37
R.T 33	Store Vuex	desarrollar		R.T 37
R.T 34	Navbar	desarrollar		R.T 37, R.T 35
R.T 35	Login-Conector URL	desarrollar		R.T 37
R.T 36	Componentes View-Router	desarrollar		R.T 37
R.T 37	SPA-CLI	desarrollar		
R.T 38	Git	Documentación	Se hace uso de gitlab como gestor de versiones para todo el proyecto.	

A.5 Consola AWS Instances

Instanciamos 9 máquinas para el Cluster de "Elasticsearch".

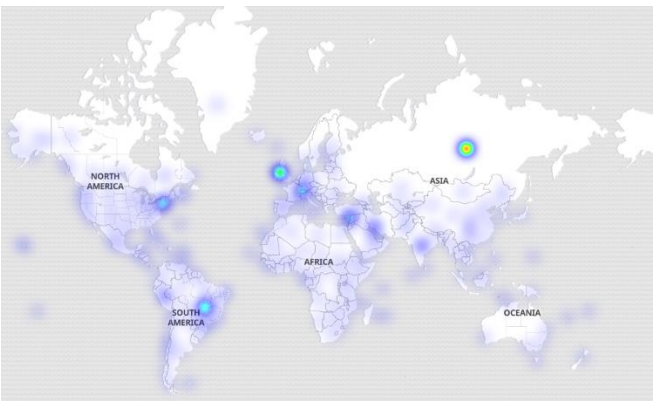


A.6 Mapa GeoIP

En el primer caso de uso se manipularon los logs de Fake Apache Log Generator para separar el campo IP y por poder generar el campo IP de tipo *geoip* con el plugin de "Logstash" en filter.

A.6 HeapMap de casos Confirmados COVID-19

En el segundo caso de uso generamos el campo location de tipo Geo-point con el plugin *mutate* .



A.7 Tabla de estado del COVID-19

Los datos extraídos mediante el plugin *http_poller* que nos permite extraer la información desde una API pública.