Sistema de monitorización de una herramienta de gestión de rendimiento

Mohamed Salim Ben Jelloul

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Máster

Director: Antonio Sarasa Cabezuelo

Convocatoria: Julio

Nota: 8

Autorización de difusión

Mohamed Salim Ben Jelloul

Fecha

El abajo firmante, matriculado en el MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "Sistema de monitorización de una herramienta de gestión de rendimiento" realizado durante el curso académico 2020-2021 bajo la dirección de Antonio Sarasa Cabezuelo en el Departamento de Sistemas Informáticos y Computación.

Resumen

Este proyecto constituye el Trabajo de Fin de Máster Mohamed Salim Ben Jelloul, alumno del MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA de la Universidad Complutense de Madrid.

La idea sobre la que se basa mi trabajo de fin de máster surgió en las prácticas que he realizado en una empresa. He colaborado en la construcción de un sistema de gestión de rendimiento que facilita la supervisión de la red para una empresa de telecomunicaciones. Dicho sistema está compuesto por múltiples subsistemas que debían ser monitorizados de forma conjunta, por tanto, este trabajo de fin de máster trata sobre la construcción de un sistema de monitorización.

El objetivo del Proyecto de Fin de Máster es construir un sistema que permita la resolución automática de los errores de los diferentes sistemas involucrados. En este sentido, se pretende realizar la monitorización de los distintos servicios y sistemas mediante la generación de alarmas. Estas alarmas proporcionarán información a los equipos de soporte para actuar sobre los sistemas.

El resultado de las acciones realizadas será almacenado en una BBDD (o motor de búsqueda) como una serie temporal que podrá ser utilizada para conseguir:

- Mayor rapidez a la hora de detectar un problema en la caída de un servicio y su recuperación.
- Almacenar la información de los resultados de las acciones realizadas que permitan analizar los datos de rendimiento para realizar un mantenimiento preventivo del sistema.

Palabras clave

NiFi, Elastic, Big data, Redis.

Tabla de contenido

Tabla de ilustraciones

Capítulo 1 – Introducción

En este capítulo se realizará una introducción al proyecto y sus objetivos. Además, se definirá el plan de trabajo y la metodología usada.

1.1 Introducción

Durante las prácticas del máster, he participado en el desarrollo de un sistema de gestión del rendimiento y supervisión de la red para una empresa de telecomunicaciones. El sistema construido está compuesto por la integración de varios subsistemas que necesitan ser monitorizados para el buen funcionamiento de este. Dada la complejidad del sistema desarrollado (sistema de gestión del rendimiento y supervisión de la red), surge la necesidad de monitorizar automáticamente todas las tareas posibles, y por tanto construir un sistema que permita la resolución automática de los errores de los diferentes sistemas involucrados.

El trabajo de fin de máster aquí presentado es el desarrollo de un sistema de monitorización de una herramienta de gestión de rendimiento [1], que facilita la supervisión de la red y los servicios de una empresa de telecomunicaciones.

Las tecnologías usadas en este trabajo de fin de master son Apache NiFi, Elastic (Elasticsearch, Kibana), Redis.

1.2 Motivación

En los proyectos que utilizan diferentes tecnologías y donde la mayoría de las tareas son automatizadas, pueden darse muchos problemas que afecten a todo el sistema tales como:

- Consumo excesivo de memoria de los sistemas
- Uso excesivo de disco de los sistemas, CPU
- Desconexión de los nodos de clúster de NiFi
- Ausencia y pérdida de datos a procesar
- Problemas en el servicio de las distintas herramientas que componen la plataforma

Así pues, surge la necesidad de monitorización del funcionamiento de los diferentes componentes de la plataforma, así como la resolución de los errores que puedan ocurrir. En concreto, en este proyecto se pretende ofrecer una solución que permita monitorizar los diferentes sistemas y resolver automáticamente los errores que se puedan producir.

1.3 Objetivos

Los principales objetivos de este trabajo de fin de máster son:

- La construcción de una herramienta que permita la monitorización de los diferentes sistemas existentes y la resolución automática de los errores que puedan ocurrir.
- Proveer de interfaces de usuario que permitan a los usuarios hacer un seguimiento de los trabajos de supervisión y mantenimiento del sistema.

Los objetivos específicos del trabajo son:

- Monitorizar el funcionamiento de diferentes componentes del proyecto.
- Detectar los errores que se puedan producir.
- Generar alarmas que representen los errores producidos y sean enviadas a los equipos de soporte.
- Solucionar los errores detectados de forma automática.

1.4 Plan de trabajo

Para realizar el proyecto se ha seguido el siguiente plan de trabajo:

- Búsqueda de información sobre el estado del arte.
- Durante esta fase se recopiló información sobre las distintas herramientas que permitan visualizar los datos y entornos y las que permiten actuar sobre los diferentes sistemas de manera automatizada y recurrente. Por último, se realizó una investigación sobre aplicaciones que propusiesen soluciones similares a la planteada en este proyecto.
- Especificación de requisitos
- En esta fase se definió la funcionalidad que se iba a implementar como parte el proyecto.
- Diseño y desarrollo del sistema
- En esta fase se llevó a cabo la implementación del sistema de acuerdo con la especificación establecida.
- Evaluación técnica de la aplicación.
- En esta fase, se ha sometido a diferentes pruebas el sistema con el objetivo de solucionar problemas que pudieran surgir y comprobar el correcto funcionamiento del sistema de monitorización automática.

1.5 Metodología de trabajo

Para la realización de este proyecto han sido necesarias tres iteraciones SCRUM (para más detalle ver el ANEXO I):

- Durante la primera iteración, se ha creado la configuración el entorno de Docker con las imágenes de herramientas utilizadas. Después se ha creado las funcionalidades de monitorización de diferentes plataformas desde NiFi.
- Durante la segunda iteración, se hace la detección de los diferentes errores definidos en cada herramienta e intentar solucionarlos.
- Durante la tercera iteración, la plataforma ya puede monitorizar y solucionar los problemas detectados de forma automatizada y está bien para implementarlo en diferentes proyectos.

Chapter 1 – Introduction

This chapter will provide an introduction to the project and its objectives. In addition, the work plan and the methodology used are defined.

1.1 Introduction

During my master's internship, I participated in the development of a performance management and network monitoring system for a telecommunications company. The system built is composed of the integration of several subsystems that need to be monitored for the proper functioning of the system. Given the complexity of the developed system (performance management system and network supervision), the need arises to automatically monitor all possible tasks, and therefore build a system that allows the automatic resolution of the errors of the different systems involved.

The master's thesis presented here is the development of a monitoring system for a performance management tool [1],which facilitates the supervision of the network and services of a telecommunications company.

The technologies used in this work are Apache NiFi, Elastic (Elasticsearch, Kibana), Redis.

1.2 Approach

In projects that use different technologies and where most of the tasks are automated, there can be many problems that affect the whole system such as:

- Excessive memory consumption of systems.
- Excessive disk usage of the systems, CPUs
- Disconnection of NiFi cluster nodes
- Absence and loss of data to be processed
- Problems in the service of the different tools that make up the platform.

Thus, the need arises to monitor the operation of the different components of the platform, as well as the resolution of errors that may occur. Specifically, this project aims to provide a solution to monitor the different systems and automatically resolve any errors that may occur.

1.3 Objectives

The main objectives of this master's thesis are:

- The construction of a tool that allows the monitoring of the different existing systems and the automatic resolution of errors that may occur.
- To provide user interfaces that allow users to keep track of the monitoring and maintenance of the system.

The specific objectives of the work are:

- Monitor the operation of different components of the project.
- Detect errors that may occur.
- Generate alarms that represent the errors produced and send them to the support teams.
- Solve the detected errors automatically.

1.4 Working Plan

The following work plan has been followed to carry out the project:

- Search for information on the state of the art.
- During this phase, information was gathered on the different tools that allow visualizing data and environments and those that allow acting on the different systems in an automated and recurrent way. Finally, research was carried out on applications proposing solutions similar to the one proposed in this project.
- Requirements specification
- In this phase, the functionality to be implemented as part of the project was defined.
- System design and development
- In this phase the implementation of the system was carried out according to the established specification.
- Technical evaluation of the application.
- In this phase, the system was subjected to different tests in order to solve problems that might arise and to verify the correct operation of the automatic monitoring system.

1.5 Methodology of work

Para la realización de este proyecto han sido necesarias tres iteraciones SCRUM (para más detalle ver el ANEXO I):

- Durante la primera iteración, se ha creado la configuración el entorno de Docker con las imágenes de herramientas utilizadas. Después se ha creado las funcionalidades de monitorización de diferentes plataformas desde NiFi.
- Durante la segunda iteración, se hace la detección de los diferentes errores definidos en cada herramienta e intentar solucionarlos.
- Durante la tercera iteración, la plataforma ya puede monitorizar y solucionar los problemas detectados de forma automatizada y está bien para implementarlo en diferentes proyectos.

Capítulo 2 – Estado del arte

En este capítulo se hará una revisión del estado del arte relacionado con el desarrollo del proyecto. Se comentará, por una parte, las herramientas de automatización de flujo de datos entre sistemas y las herramientas de monitorización y resolución de problemas, su importancia, en qué consisten y su evolución día a día. Y por otra parte, como no hay proyectos en el mercado que realicen todas las tareas de forma unificada, se detallarán proyectos similares divididos en dos partes:

- Monitorización de sistemas y generación de alarmas.
- Automatización de procesos y flujo de datos entre sistemas.

2.1 Herramientas de automatización de procesos y flujo de datos entre sistemas

Un proceso es un conjunto de tareas/actividades. Por ejemplo:

- Ejecutar comandos o scripts en diferentes sistemas
- Enrutamiento y transformación de datos

El flujo de datos [2] es el movimiento de datos a través de uno o diferentes sistemas compuestos por software, hardware o una combinación de ambos.

La integración de datos de diferentes sistemas consiste en 3 procesos importantes (ETL): [3]

- Extracción
  El primer paso del proceso ETL es extraer los datos de varias fuentes. Cada uno de los sistemas fuente puede almacenar sus datos en un formato completamente diferente al del resto.
- Transformación
  Consiste en transformar los datos de acuerdo con un conjunto de reglas de negocio. La transformación de los datos puede incluir varias operaciones, como filtrar, ordenar, agregar, unir datos, limpiar datos...
- Carga
  El último paso del ETL consiste en cargar los datos transformados en el destino, que puede ser una base de datos o un almacén de datos.

Cuando la empresa tiene diferentes sistemas, pueden producirse muchos errores. Los problemas y patrones de solución que surgieron se han debatido y articulado ampliamente [4].

Una herramienta de automatización consiste en diseñar procesos con el fin de usar la capacidad de los sistemas para llevar a cabo determinadas tareas anteriormente realizadas por seres humanos, pudiendo ser controladas, corregidas y visibles a través de dichos procesos. [5]

Es básicamente eficiente para automatizar la ejecución de aquellas tareas más delicadas, pesadas y repetitivas.

Hoy en día muchas de esas herramientas disponen de una consola de gestión centralizada, siendo más fácil crear procesos y visualizar su funcionamiento sin la necesidad de usar código y disminuyendo la complejidad para la implantación de nuevas funcionalidades.

Los principales objetivos del proceso de automatización son:

- Mejorar la productividad y eficiencia, reduciendo los costos de producción y mejorando la calidad. [6]
- Ahorrar tiempo y energía.
- Optimizar la planificación y el control.
- Realizar aquellas operaciones imposibles de controlar intelectual o manualmente.
- Conseguir resultados con el mínimo esfuerzo y coste, o que al menos no sean tan variables.
- Recibir mediante notificaciones automáticas alertas de lo que pasa o deja de pasar en cualquier punto del proceso.
- Gestión global en todas las fases de desarrollo de software con una misma herramienta.
- Automatizar el desarrollo del software, la documentación, la generación de código, las pruebas de errores y la gestión del proyecto.
- Controlar y dar seguimiento del proceso en todo momento de forma detallada y completa, pudiendo conocer su estatus de forma inmediata

Hablando del proceso de automatización se pueden distinguir distintos tipos de control dependiendo del grado de intervención humana. De este modo, denominaremos “control automático” cuando éste se lleva a cabo sin ninguna intervención directa del ser humano, mientras que el “control semi automatizado” se referirá a aquel en el que existe una intervención humana.

2.2 Herramientas de visualización de datos

2.2.1 Definición

La visualización de datos suele consistir en gráficos y diagramas que son representaciones visuales de los datos. Estas representaciones gráficas ofrecen una poderosa forma de resumir y presentar los datos de una manera que resulta más fácil de comprender [7]. Cuando se trata de conjuntos de datos que incluyen cientos de miles o millones de puntos de datos, por ejemplo, en el caso de series temporales [8], la automatización del proceso de creación de una visualización facilita considerablemente el trabajo del diseñador.

Sin las visualizaciones de datos, sacar conclusiones de grandes conjuntos de datos o incluso discernir información útil de ellos es casi imposible. Con los métodos de visualización de datos, los diseñadores pueden hacer que la información sea comprensible para los interesados. En la Ilustración 1 [7] se muestra cómo se utilizan una serie de cuadros y gráficos para describir las características principales de los datos.

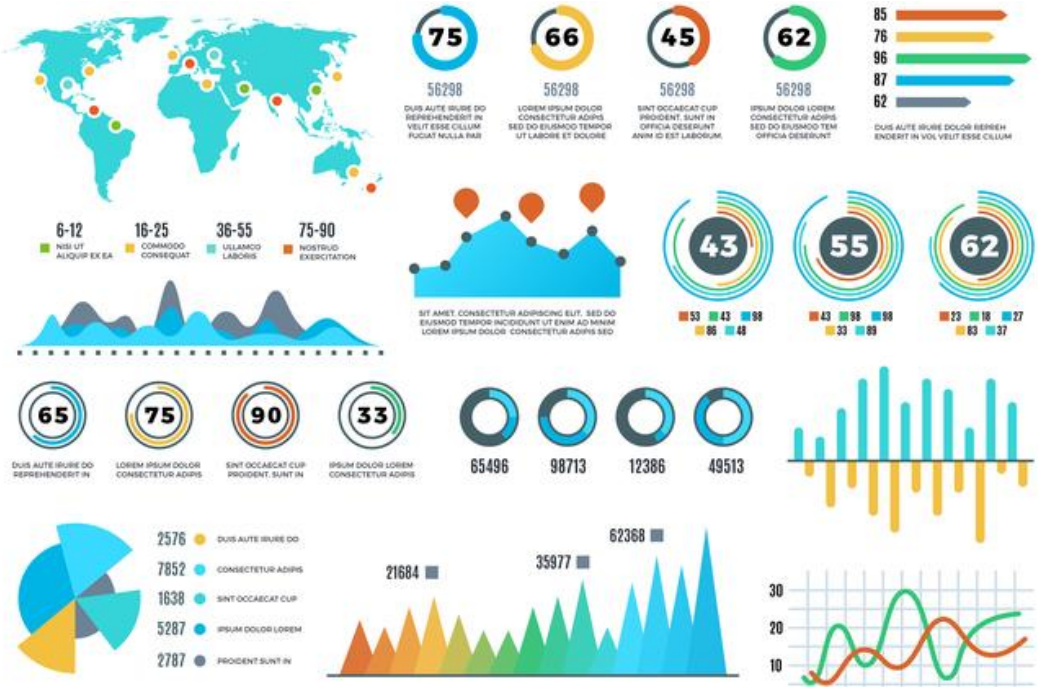

Ilustración 1 Ejemplos de presentación visual mediante diferentes gráficos y visualizaciones

2.2.1 Big Data

Tal y como lo define Gartner: "Los Big Data son activos de información de gran volumen, alta velocidad o variedad que requieren nuevas formas de procesamiento para permitir una mejor toma de decisiones, el descubrimiento de conocimientos y la optimización de procesos." [9]

El término "Big Data" se explica por sí mismo: una colección de enormes conjuntos de datos que las técnicas informáticas normales no pueden procesar. El término no sólo se refiere a los datos, sino también a los diversos marcos, herramientas y técnicas implicados. El avance tecnológico y la aparición de nuevos canales de comunicación (como las redes sociales) y de nuevos dispositivos más potentes han supuesto un reto

para los agentes del sector, en el sentido de que tienen que encontrar otras formas de manejar los datos.

Aproximadamente hasta el año 2003, se estimaba que se habían generado alrededor de cinco mil millones de gigabytes de datos. La misma cantidad de datos se generó en sólo dos días en 2011. En 2013, este volumen se generaba cada diez minutos. Por tanto, no es de extrañar que la generación del 90% de todos los datos gestionados en la actualidad se haya producido en los últimos años.

Los datos son útiles cuando se procesan, sin embargo, no se habían considerado su utilidad hasta que apareció el fenómeno Big data [10]. Algunas fuentes de generación de datos masivos son:

- Datos de la caja negra
  Son los datos generados por los aviones, incluidos los jets y los helicópteros. Los datos de las cajas negras incluyen las voces de la tripulación de vuelo, las grabaciones de los micrófonos y la información sobre el rendimiento de la aeronave.

- Datos de las redes sociales
  Se trata de datos desarrollados por sitios de medios sociales como Twitter, Facebook, Instagram, Pinterest y Google+.

- Datos bursátiles
  Se trata de datos de las bolsas de valores sobre las decisiones de compra y venta de acciones tomadas por los clientes.

- Datos de las redes eléctricas
  Son datos de las redes eléctricas. Contienen información sobre nodos concretos, como la información de uso.

- Datos de transporte
  Incluye la capacidad posible, el modelo de vehículo, la disponibilidad y la distancia recorrida por un vehículo.

- Datos de los motores de búsqueda
  Es una de las fuentes más importantes de Big data. Los motores de búsqueda tienen enormes bases de datos donde obtienen sus datos.

Unas de las aplicaciones de Big Data son:

- Las aplicaciones de streaming
  Trabajan sobre datos parciales. Por definición, los datos en streaming tienen un tamaño ilimitado y son difíciles. Para manejar un flujo continuo de datos, es necesario crear resúmenes de las ventanas de datos temporales y utilizarlos en

el procesamiento posterior del flujo. Puede haber muchas formas de definir las ventanas de datos, incluidas las ventanas basadas en el tiempo y las ventanas basadas en el recuento de datos.

- Los pipelines de datos
  Se utilizan principalmente para operaciones de extracción, transformación y carga (ETL), aunque pueden incluir pasos como la ejecución de un algoritmo complejo. En su mayoría tratan con datos no estructurados almacenados en bruto o con datos semiestructurados almacenados en bases de datos NoSQL [11]. Los pipelines de datos trabajan con los conjuntos de datos más grandes posibles de los tres tipos de aplicaciones.

- Las aplicaciones de aprendizaje automático.
  Ejecutan operaciones algebraicas complejas y pueden hacerse funcionar en paralelo utilizando operaciones paralelas sincronizadas. En la mayoría de los casos, los datos pueden equilibrarse entre los trabajadores a medida que se utilizan los datos curados. Los algoritmos pueden ser regulares o irregulares y pueden necesitar un equilibrio de carga dinámico de los cálculos y los datos.

- Los servicios
  Están evolucionando hacia un modelo impulsado por eventos para conseguir escalabilidad, eficiencia y rentabilidad en la nube. Los antiguos servicios monolíticos están siendo sustituidos por microservicios más ágiles. Se prevé que estos microservicios se compongan de pequeñas funciones dispuestas en un workflow [12] de datos para lograr la funcionalidad requerida.

2.2.2 Series temporales

Una serie temporal [13] es una sucesión de observaciones de una variable recogidos en determinados instantes y ordenados cronológicamente. Las observaciones pueden estar recogidas en intervalos iguales o desiguales. El análisis de series temporales ayuda detectar patrones de comportamiento entre la propia serie y otras series temporales que permiten interpolar los datos y de este modo sacar conclusiones como realizar análisis predictivos del comportamiento de los datos, detectar anomalías en que sirvan para ayudar a predecir comportamientos sobre datos en el pasado, futuro y momentos intermedios.

El análisis de las series temporales resulta útil en casos como los que se exponen a continuación:

- Analizar datos meteorológicos recogidos con diferentes frecuencias (Horas, días, etc.)
- Analizar las variaciones en el rendimiento de las aplicaciones que permitan detectar problemas en el funcionamiento o el incremento del uso de estas.

- Disponer de información en tiempo real de los datos en campos como la medicina mediante dispositivos médicos
- Seguir los registros de la red
- Analizar el comportamiento de los índices bursátiles y el volumen de transacciones [14]
- Análisis de las variaciones de los caudales fluviales a lo largo del tiempo [14]
- Análisis del uso de los transportes durante el año. En la Ilustración 2 muestra el número de embarques mensuales de pasajeros para el período comprendido entre enero de 2004 y diciembre de 2013 [14]

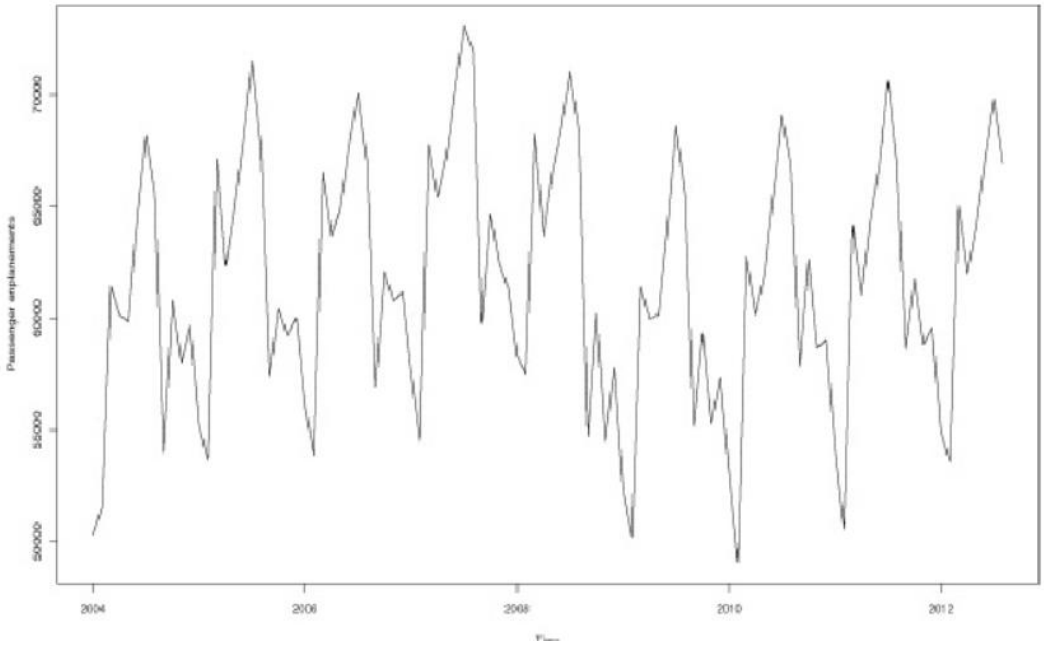

Ilustración 2 el número de pasajeros mensuales que suben a un avión en los estados unidos para el período comprendido entre enero de 2004 y diciembre de 2013.

2.2.3 Técnicas de visualización de datos

Existen diferentes técnicas que se pueden utilizar para satisfacer necesidades específicas de visualización en función de los datos con los que trabajan.

Los gráficos más habituales cuando se requiere realizar una representación visual de los datos son los siguientes: [15]

- Tablas
  Se utilizan para representar información en formato tabular (filas y columnas), normalmente números, fechas y palabras, que pueden ser listados parciales o completos de la información existente, pudiendo representar agregaciones o resúmenes que permiten entender la información disponible.

- Gráficos de Barras

Se utilizan para representar variables numéricas y su evolución en el tiempo o bien agrupaciones de información y valores, permitiendo comparar las magnitudes de las barras o columnas que se visualizan.

- Gráficos de Líneas
  Los gráficos de líneas son especialmente útiles a la hora de representar información numérica y su variación en el tiempo, cada magnitud en un momento del tiempo se representa por un punto que se une mediante una línea con el resto de magnitudes representadas.

- Gráficos Circulares
  Los gráficos circulares o gráficos de tarta (pie en inglés), son aplicables en la representación de los porcentajes que suponen las distintas magnitudes respecto a la suma total de las mismas.

- Gráficos de Dispersión
  Se trata de gráficos que se emplean para visualizar las relaciones de similitud o diferenciación entre las magnitudes que representan.

- Gráficos de Burbujas
  Son una variación visual de los gráficos de dispersión que muestran círculos que agrupan magnitudes en lugar de puntos, normalmente el tamaño de los círculos es utiliza para incorporar una dimensión adicional.

- Treemap
  Este gráfico permite representar variables de forma agrupada, mostrando un conjunto de rectángulos cuyo tamaño se calcula de forma proporcional al porcentaje que supone una variable respecto al área del rectángulo total.

- Grafos de relación social
  Los grafos sirven para representar relaciones entre variables y cómo interaccionan unas con otras. En los grafos se utilizan nodos para representar entidades, personas o elementos y flechas para mostrar las relaciones entre ellos. Son muy útiles en el análisis de información de redes sociales ya que permiten visualizar las interacciones que se producen.

- Nubes de palabras o tag cloud
  Representación de un conjunto de palabras que tienen relevancia en el análisis que se está realizando, el tamaño de las palabras en estas nubes suele tener relación con el volumen o uso de las mismas.

- Infografías
  Se trata de una representación de información en la que se aplican elementos gráficos que facilitan el entendimiento de lo que se quiere transmitir, especialmente utilizadas para acciones de marketing y difusión.

2.3 Herramientas para la resolución de problemas

Un sistema que utiliza diferentes tecnologías podría tener muchos errores, retrasos y otros problemas que afectan al funcionamiento de este. Esos problemas pueden estar relacionados con diferentes aspectos tal como [16]: problemas de hardware, problemas de software y aplicaciones, problemas de red, problemas de arranque, y otros.

A veces se tienen que realizar varios ajustes en el sistema para solucionar estos problemas, y puede ser muy complicado, y a veces arriesgado. Sin embargo, existen herramientas para la resolución de problemas que facilitan el trabajo, y que permiten ahorrar mucho tiempo.

2.4 Herramientas similares al sistema construido

2.4.1 Zenoss

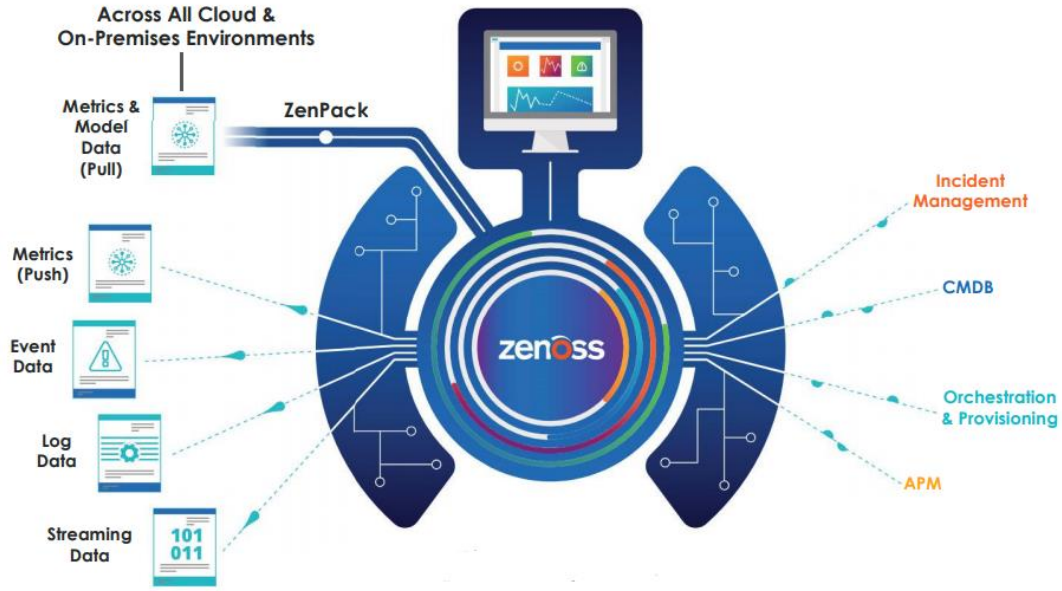

Ilustración 3 Arquitectura de Zenoss

Construido sobre una única plataforma, Zenoss Core es una solución de monitorización unificada que ofrece la visión más rápida y precisa de los servicios potencialmente afectados para que pueda abordar los problemas antes de que su negocio y sus usuarios se vean afectados. [17]

Como se muestra en la Ilustración 3, Zenoss supervisará todo lo que soporte agentes compatibles con los estándares como SNMP, SSH y WinRS/WinRM para supervisar todas las cosas.

Zenoss también aprovechará las API de destino para la nube y la infraestructura convergente como CloudStack, OpenStack, AWS y UCS.

Además, Zenoss es compatible con una amplia variedad de otros plugins de código abierto, lenguajes de scripting y aplicaciones.

Las principales funcionalidades de Zenoss son:

- Supervisión de la disponibilidad de los dispositivos de red mediante SNMP, SSH, WMI
- Supervisión de los servicios de red (HTTP, POP3, NNTP, SNMP, FTP)
- Monitorización de los recursos del host (procesador, uso del disco) en la mayoría de los sistemas operativos de red como se muestra en la Ilustración 4
- Monitorización del rendimiento de los dispositivos en series temporales
- Supervisión ampliada de Microsoft Windows a través de Windows
- Extensión de las funcionalidades de gestión y monitorización mediante el empleo de componentes open source
- Visualización de problemas en el sistema a través de una consola de visualización y gestión de alertas, siendo estas alertas originadas en base a un conjunto de reglas y calendarios que se definen a través de plantillas
- Descubrimiento automático de los recursos de red y los cambios en la configuración de la red

Ventajas Principales de Zenoss: [18]

- Análisis inmediato de la causa raíz
    - Obtener la máxima visibilidad mediante el modelado en tiempo real de los riesgos relacionados con la infraestructura de extremo a extremo, y aislar los problemas inmediatamente para mejorar el tiempo medio de resolución (MTTR) y eliminar las pérdidas por interrupción del servicio.
    - Obtener una visibilidad total del del estado general de los servicios de TI con cuadros de mando e informes. Colaborar con todos los equipos para coordinarla investigación y la resolución de problemas.

- Prevención de interrupciones informáticas
    - Aprovechar los datos de alta cardinalidad para garantizar la fiabilidad de los sistemas efímeros y reducir los tiempos de respuesta de TI mediante el análisis predictivo impulsado por el aprendizaje y la IA.
    - Evolucionar de la disponibilidad y el rendimiento a capacidad y optimización, y eliminar los riesgos asociados con la transformación digital.

- Rendimiento optimizado de las aplicaciones

- Ver el rendimiento y las anomalías en todas las infraestructuras locales y en la nube con información de AIOps para predecir el estado del servicio y los problemas de rendimiento.
- Aplicar políticas de monitorización consistentes en todos los sistemas locales.
- Ofrecer la gestión como un servicio para equipos de DevOps

- Automatización inteligente
  - Compartir datos e información clave con otras herramientas de ITOM para automatizar una resolución rápida y preparar su plataforma de monitorización para que funcione a cualquier escala.
  - Permitir una TI ágil mientras elimina la fatiga de los empleados al reducir las alertas.

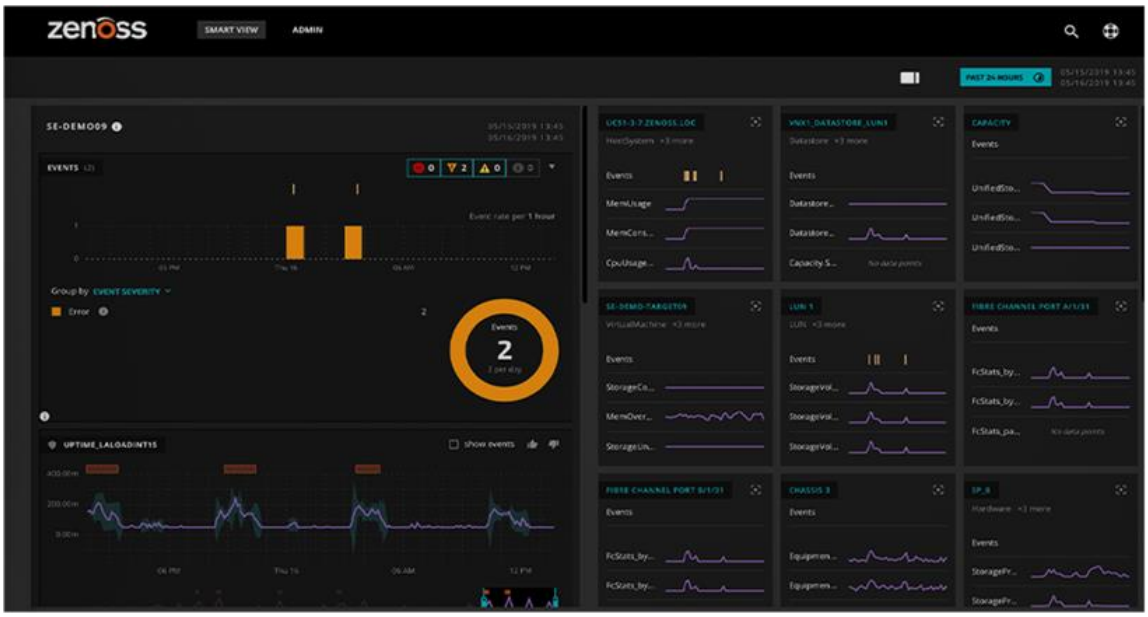

Ilustración 4 Ejemplo de un dashboard en Zenoss

2.4.2 Apache Airflow

Apache Airflow [19] es una herramienta de código abierto que permite la creación, programación y supervisión de procesos de trabajo. Fue desarrollada por Airbnb en octubre de 2014 [20] como una solución que les permitiese gestionar de forma ágil los procesos de trabajo cada vez más complejos de la empresa. La creación de Airflow permitió a Airbnb programar sus procesos de trabajo y supervisarlos a través de una interfaz de usuario integrada en la propia herramienta. [21]

Airflow está basada en el paradigma DAG (grafos acíclicos dirigidos), permitiendo gestionar la orquestación de múltiples flujos de trabajo desde una interfaz visual y unificada. Pyhton es el lenguaje de programación utilizado por Airflow para la definición de tareas y dependencias, mientras que la programación de los flujos de trabajos, su ejecución y gestión se realizan desde la interfaz visual de Airflow. Los DAGs definidos

en Airflow se pueden ejecutar de forma periódica o puntual, ya sea mediante reglas temporales (por ejemplo, todos los días a una hora determinada, cada 12 horas, cada 10 minutos, …) o en base a eventos que se produzcan (por ejemplo o en función de eventos externos (por ejemplo, el cumplimiento de una condición tras la consulta a una base de datos de un archivo en Hive [22]). Airflow, con respecto a otras herramientas basadas en el paradigma DAGs, como Oozie y Azkaban, que requerían de un gran conocimiento para poder realizar las configuraciones y estructurar la información de forma adecuada, supone una mejora significativa ya que la programación de los DAGs puede realizarse en un único fichero Python [23].

En la Ilustración *5* se muestra un ejemplo de la lógica para crear un flujo en Apache Airflow.

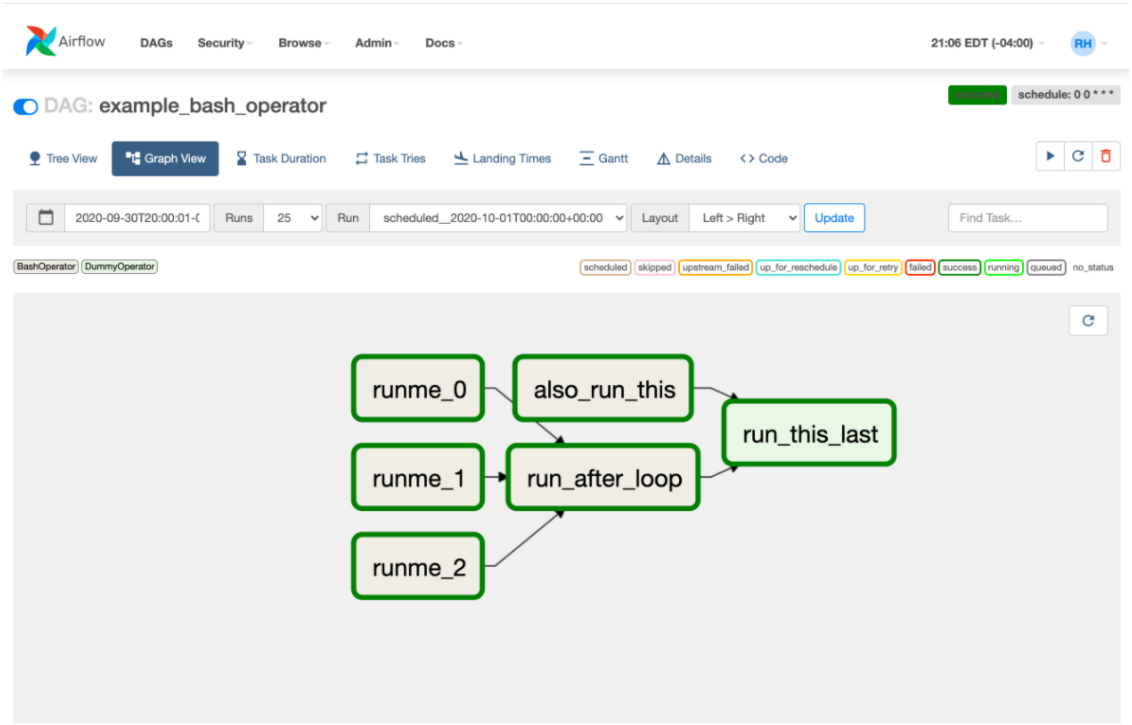

Ilustración 5 *Un workflow en Apache Airflow*

Principios de Apache AirFlow [19]:

- Dinámicos
  Los pipelines de AirFlow se configuran como código (Python), lo que permite la generación dinámica de pipelines. Esto permite escribir código que instancie los pipelines dinámicamente.

- Extensible
  Define fácilmente tus propios operadores, ejecutores y extiende la librería para que se adapte al nivel de abstracción que se adapte a tu entorno.

- Sencillez

Los pipelines de AirFlow son sencillos y explícitos. La parametrización de sus scripts está integrada en el núcleo de AirFlow mediante el potente motor de plantillas Jinja.

- Escalable
  AirFlow tiene una arquitectura modular y utiliza una cola de mensajes para orquestar un número arbitrario de trabajadores. AirFlow está preparado para escalar hasta el infinito

2.4.3 Apache Solr

Apache Solr [24] es un servidor de búsqueda de texto completo Enterprise Java de código abierto. Se inició en 2004 en CNET (en ese momento, uno de los sitios más conocidos de noticias y reseñas sobre tecnología), luego se convirtió en un proyecto Apache en 2007, y desde entonces se ha utilizado para muchos proyectos y sitios web.

Inicialmente fue concebido como una aplicación web para proporcionar una amplia gama de capacidades de búsqueda de texto completo, utilizando y ampliando la potencia de la conocida biblioteca Apache Lucene. Desde 2011 [25], los dos proyectos se han fusionado en un único esfuerzo de desarrollo, con un modularidad mejorada.

Solr está diseñado (Ilustración 7) para ser un servidor web independiente, que expone el texto completo y otras funcionalidades a través de sus propios servicios tipo REST, que pueden ser consumidos de muchas maneras diferentes desde casi cualquier plataforma o lenguaje. Este es el caso de uso más común, y en el que se centrará el trabajo.

Además, Solr no es una base de datos; es muy diferente de las relacionales, ya que está diseñada para gestionar índices de los datos reales (digamos, metadatos útiles para buscar sobre los datos reales) y no los datos en sí mismos o las relaciones entre ellos.

En la Ilustración *6*, se muestra la interfaz de Solr.

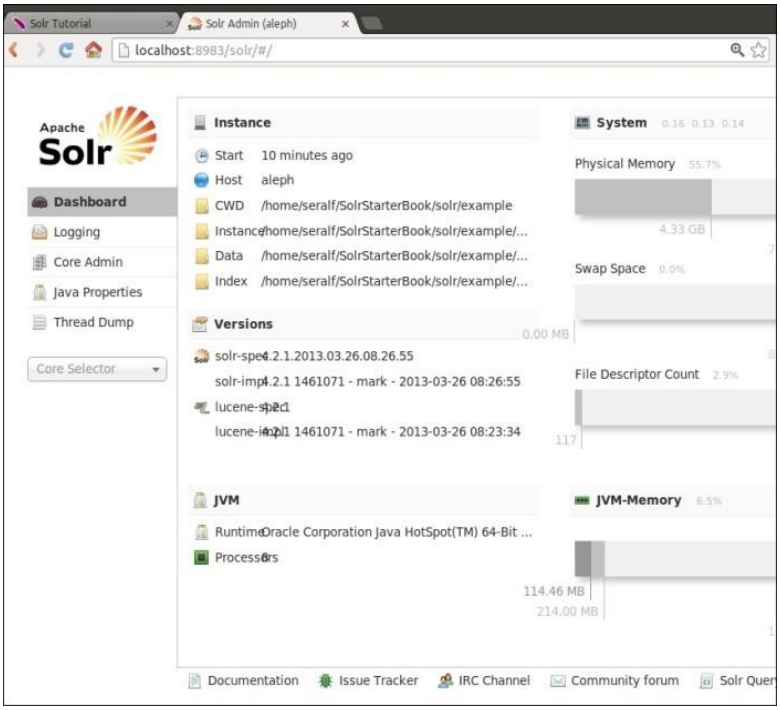

Ilustración 6 Interfaz de Solar

Entre las características más destacadas de Solr [26] se encuentran la búsqueda robusta de texto completo, la búsqueda facetada, la indexación en tiempo real, la agrupación, el manejo de documentos (Word, PDF, etc.) y la búsqueda geoespacial. Las capacidades de fiabilidad, escalabilidad y tolerancia a fallos hacen que Solr sea aún más exigente para los desarrolladores, especialmente para los profesionales de SEO y DevOp.

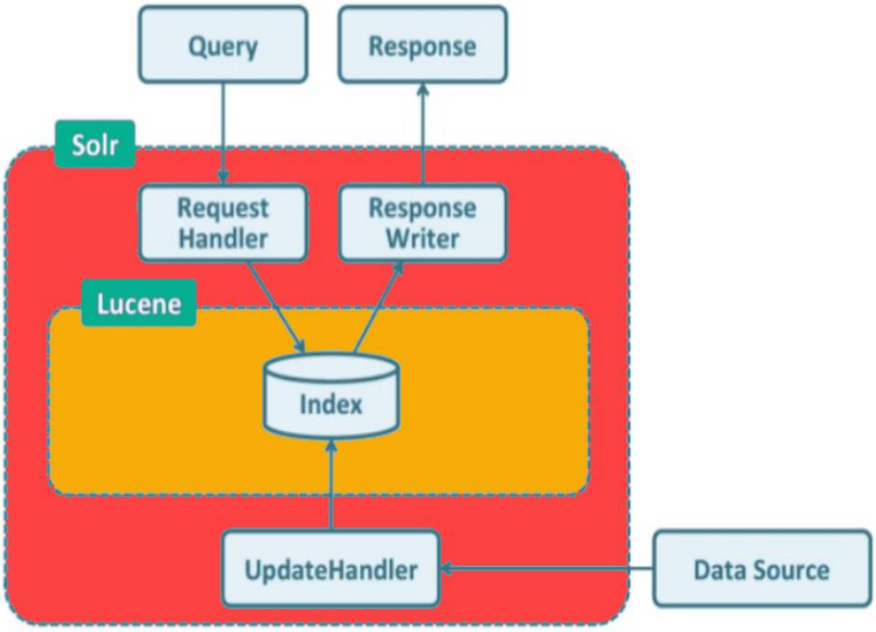

Ilustración 7 Arquitectura de Solr

Apache Solr cuenta con los siguientes componentes:

- Consulta

El analizador de consultas analiza las consultas que debe pasar a Solr. Verifica su consulta para comprobar los errores sintácticos. Después de analizar las consultas, las traduce a un formato conocido por Lucene.

- Manejador de solicitudes
  Las peticiones que se envían a Apache Solr son procesadas por el gestor de peticiones. La solicitud puede ser una solicitud de consulta o de actualización de índices. Debe seleccionar el gestor de peticiones según sus necesidades. Para pasar una solicitud a Solr, debe asignar el gestor a un punto final de URL específico.

- Redactor de respuestas
  Un escritor de respuestas generará salidas formateadas para las consultas de entrada. Soporta varios formatos como XML, JSON, CSV, etc. Puede tener diferentes escritores de respuesta para diferentes tipos de solicitudes.

- API REST para gestión de actualizaciones
  Cuando se envía una solicitud de actualización a Apache Solr, ésta se ejecuta a través de un conjunto de plugins, firma, registro e indexación. Este proceso se conoce como procesador de solicitudes de actualización. El gestor de actualizaciones también es responsable de las modificaciones, como la adición o eliminación de archivos, etc.

Capítulo 3. Herramientas tecnológicas

A continuación, se describen las principales herramientas tecnológicas que se han utilizado en el desarrollo del proyecto.

3.1 Apache NiFi

Apache NiFi es una plataforma integrada de software libre para el procesamiento y la logística de datos en tiempo real. Dicha plataforma está escrita en Java y automatiza el movimiento de datos entre diferentes sistemas de forma rápida, es decir, carga datos de diferentes fuentes, los pasa por un flujo de procesos para su tratamiento, y los vuelca en otra fuente. [27]

Una de sus características principales es su potente e intuitiva aplicación web (Ilustración 8), la cual permite diseñar, configurar y monitorizar el flujo de datos de forma visual. De una manera más específica, te posibilita operar sobre el proceso mediante su arranque y/o parada, te facilita el seguimiento y modificación del flujo de datos de principio a fin e incluso te deja integrar nuevos procesos.

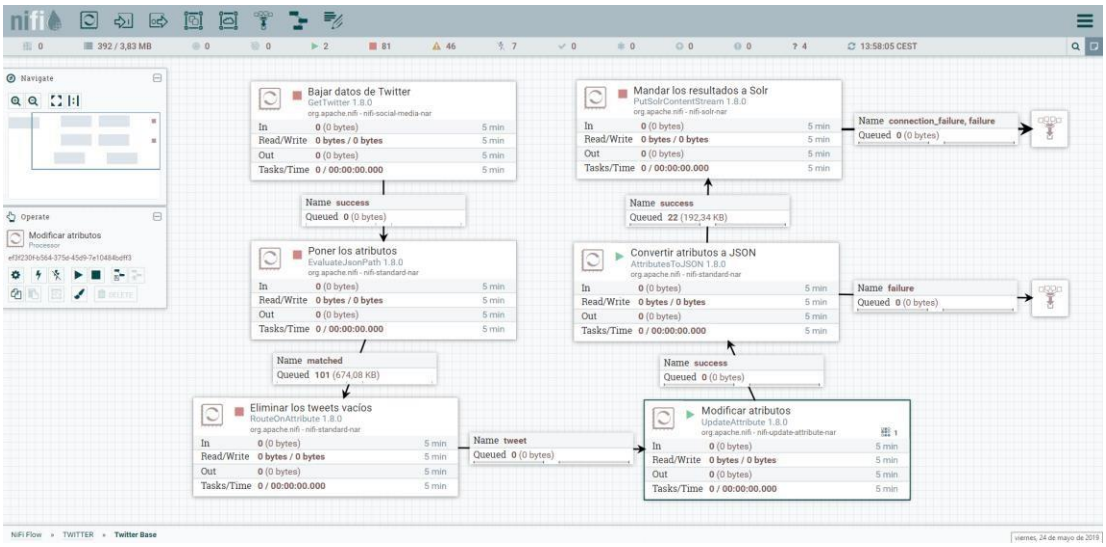

Ilustración 8 Interfaz web de NiFi. Flujo para la extracción y tratamiento de tweets.

Cada procesador, es decir, cada una de las cajas que forman parte del flujo, aporta una funcionalidad concreta. NiFi incluye por defecto con una larga lista de operaciones sobre los datos, en forma de procesadores.

Entre sus muchas utilidades, se encuentran las siguientes:

- Ingesta de información.

- Consulta e inserción de datos en diferentes tecnologías (ElasticSearch, MongoDB, MySQL…).
- Cambios en la estructura de la información entre diferentes formatos de datos (JSON, CSV).
- Operaciones de unión y división de los datos, así como su validación y transformación.
- Delegación a otros sistemas de procesamiento (Kafka, Flume).
- Ejecución en paralelo (mediante Apache ZooKeeper).
- Rastreo de datos en tiempo real.

Los conceptos de diseño fundamentales de NiFi están estrechamente relacionados con las ideas principales de la Programación Basada en el Flujo [28].

Entre sus componentes, se encuentran las siguientes:

- FlowFile
  Conjunto de información que se mueve como una sola entidad por el sistema. Está formado por el contenido y sus atributos o metadatos. Este sistema sería por ejemplo un fichero que se acaba de añadir al filer.

- Processor
  Elemento de código que procesa el FlowFile (cada una de las cajitas).

- Connection
  Caminos o conexiones entre Processors (queue). Pueden ser configuradas para priorizar el paso de los flowfiles, numero de flowfiles y tamaño de estas.

- Process Group
  Conjunto de Processors y sus Connections con entidad propia.

- Data Flow
  Lógica de gestión de los elementos anteriores.

- Templates
  Fichero de extensión XML que guarda el dashboard de trabajo o sus componentes.

3.2 Elasticsearch

Elasticsearch es un motor de búsqueda desarrollado en Java y basado en Apache Lucene [29]. Dispone de un motor de búsqueda de texto libre, puede ser distribuido y facilita

una interfaz web RESTful. Elasticsearch tiene doble licencia: la licencia pública del lado del servidor y la licencia de Elasticsearch [30] , mientras que otras partes [31] están sujetas a la licencia de Elasticsearch.

Elasticsearch permite realizar diversos tipos de búsquedas y combinarlas entre sí, con datos estructurados y desestructurados, mediante las cuales es capaz de responder a cualquier pregunta que se realice sobre la marcha de un negocio o una aplicación. Al ser un sistema distribuido, Elasticsearch es capaz de funcionar en un solo equipo o bien a lo largo de cientos de servidores en los cuales manejar cantidades enormes de información. Distribuye toda la información en todos los nodos, por tanto, es tolerante a fallos y tiene alta disponibilidad. Al igual que distribuye la información, distribuye el procesamiento. Cuando se realiza una consulta o búsqueda y esa información se encuentra distribuida, será cada nodo el que procese dicha información y devuelva los resultados. Por tanto, se pueden obtener mejores rendimientos.

Estructura de la información en el programa: [32]

- Index
  Una búsqueda en Elasticsearch nunca arroja el contenido como respuesta, sino el índice, en el cual se almacenan, ya preparados, todos los contenidos de todos los documentos. De esta forma la búsqueda requiere muy poco tiempo. Es el llamado inverted index (índice inverso): para cada término de búsqueda se indica el lugar donde se puede encontrar dicho término.

- Document
  La salida para el índice son los documentos, en los cuales aparecen los datos. No tienen que ser necesariamente textos completos (por ejemplo, artículos de blogs) es suficiente que se trate de archivos con información.

- Field
  Un documento, a su vez, consta de varios campos. Además del campo de contenido propiamente dicho, hay otros metadatos que también forman parte de un documento. Por ejemplo, Elasticsearch puede utilizarse para buscar metadatos sobre el autor o el momento de la creación. Además, se pueden realizar mappings para establecer cualquier cosa acerca de los campos o las operaciones. Por ejemplo, los mapping types determinan como se indexa el índice. A medida que los conjuntos de datos aumentan en tamaño y complejidad, el esfuerzo humano requerido para inspeccionar dashboards o mantener reglas para detectar problemas de infraestructura, ataques cibernéticos o problemas de negocios se vuelve poco práctico. Las funciones de aprendizaje automático de Elasticsearch modelan automáticamente el comportamiento normal de sus datos de series de tiempo (tendencias de aprendizaje, periodicidad y más) en tiempo real para identificar anomalías, agilizar el análisis de las posibles causas y reducir los falsos positivos.

3.3 Kibana

Kibana [33] es un software de código abierto de cuadros de mando de ELK Stack, y es una muy buena herramienta para crear diferentes visualizaciones, gráficos, mapas e histogramas, y mediante la integración de diferentes visualizaciones juntas, se pueden crear cuadros de mando. Forma parte de ELK Stack, por lo que es bastante fácil leer los datos de Elasticsearch. Esto no requiere ninguna habilidad de programación. Kibana tiene una hermosa interfaz de usuario para crear diferentes tipos de visualizaciones, incluyendo gráficos, histogramas y cuadros de mando.

A partir de Kibana se pueden realizar fácilmente análisis de datos avanzados, visualizar los datos en una variedad de tablas, gráficos y mapas, análisis automáticos en tiempo real y algoritmos de búsqueda muy flexibles. A través de dashboards (Ilustración 9), las diversas visualizaciones interactivas pueden combinarse para formar una imagen general dinámica que permita su filtrado y análisis. Además, Kibana permite identificar anomalías, comportamientos, eventos, picos, etc…., de forma gráfica y visual.

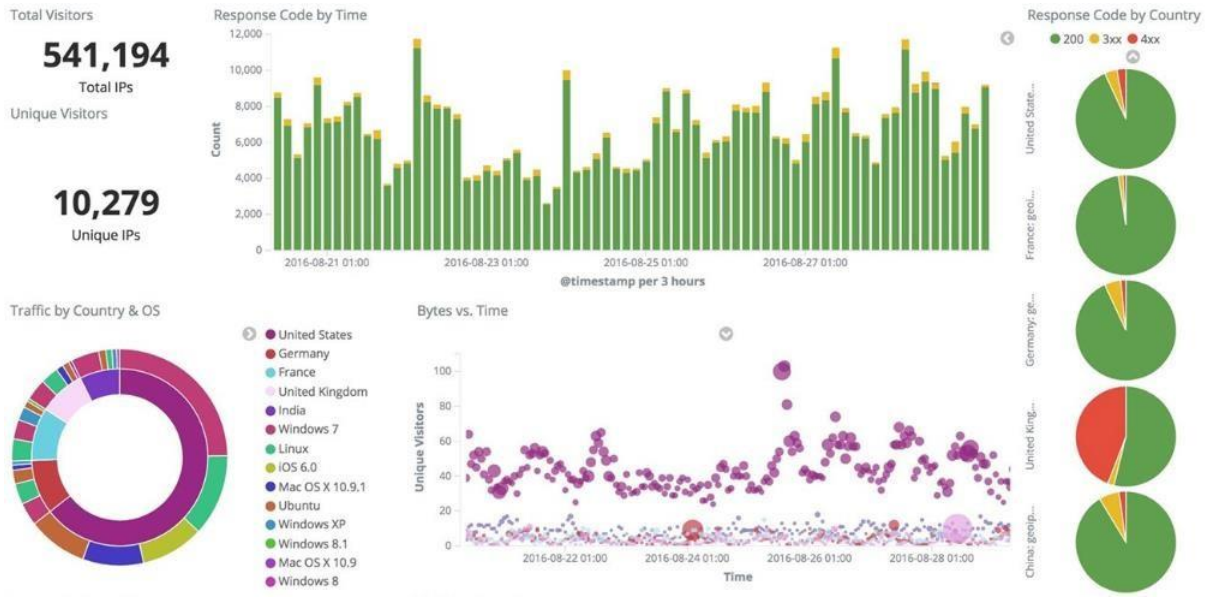

Ilustración 9 Ejemplo de un dashboard en kibana

Al ser una aplicación web escrita en JavaScript, Kibana puede utilizarse en todas las plataformas y navegadores.

Algunas de las distintas secciones que forman la interfaz: [34]

- Discover
  Pantalla que permite filtrar y buscar registros en un intervalo específico, para una primera exploración de los datos.

- Visualize

Pantalla donde se pueden crear, modificar, ver y guardar visualizaciones personalizadas (Gráficos, tablas …) para el análisis de datos.

- Dashboard
  Pantalla donde se pueden crear, modificar, ver y guardar tableros de control personalizados que se componen de visualizaciones y/o búsquedas.

- Timelion
  Pantalla que permite visualizar series temporales de datos de distintos índices en una única visualización, así como la realización de cálculos con dichos datos.

- Machine Learning
  Interfaz de usuario para la detección de anomalías y comprensión de los resultados.

- Dev Tools
  Contiene herramientas de desarrollo para interactuar con los datos a través de una consola.

- Management
  Pantalla que permite modificar las propiedades de los distintos objetos que puede guardar Kibana, como las búsquedas, visualizaciones o dashboard.

- Settings
  Pantalla que permite cambiar la configuración por defecto o patrones de índice.

3.4 Redis

Redis [35] es un avanzado almacén de valores clave de código abierto que es capaz de almacenar estructuras de datos como cadenas, listas, hashes y conjuntos. Redis, que significa Remote Dictionary Server [36], es una base de datos en memoria en la que todo el conjunto de datos debe estar disponible en la memoria durante el tiempo de ejecución.

Redis está basado en una estructura de tablas hash donde cada clave tiene un valor asociado. En comparación con otras bases de datos de tipo Clave-Valor, Redis permite el uso de estructuras más complejas y flexibles que abren una serie de posibilidades ante las distintas necesidades de aplicaciones de negocio.

Para lograr su excelente rendimiento, Redis trabaja con un conjunto de datos en memoria. Dependiendo de su caso de uso, puede persistir volcando el conjunto de datos al disco de vez en cuando, o agregando cada comando a un registro. En comparación con las bases de datos tradicionales basadas en disco, donde la mayoría

de las operaciones implican ir y volver al disco, los almacenes de datos en memoria como Redis no se ven afectados de la misma manera. Por lo tanto, pueden admitir un mayor orden de magnitud, más operaciones y tiempos de respuesta más rápidos. El resultado es un rendimiento increíblemente rápido con operaciones de lectura o escritura promedio que se ejecutan en menos de un milisegundo y capacidad para procesar millones de operaciones por segundo.

En la Ilustración *10* se muestra un ejemplo de flujo de trabajo de Redis

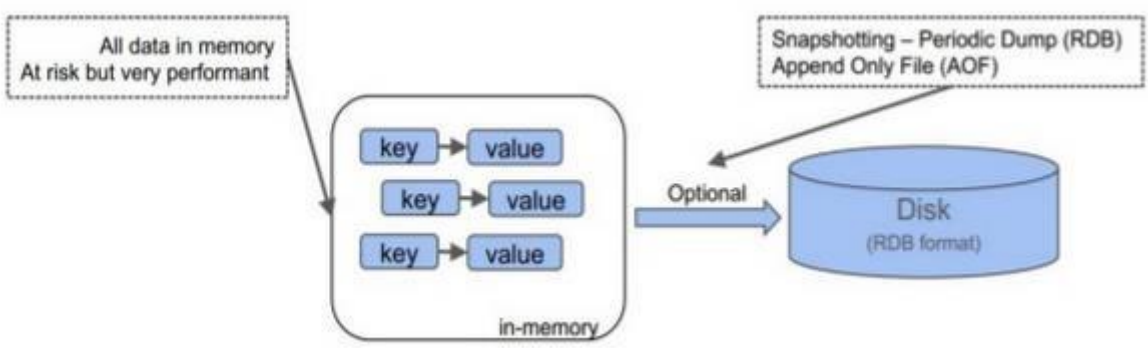

Ilustración 10 *flujo de trabajo de Redis*

Dado que la información queda almacenada y persistida en disco, Redis ofrece rápidos accesos en la recuperación de los datos; además, Apache NiFi contiene servicios que permiten su conexión con Redis.

3.5 Docker

Docker [37] es una plataforma de contenedores de código abierto que mejora el proceso de envío de un ciclo de vida de desarrollo.

Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución.

Con Docker, se puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutara.

De otra parte, Docker proporciona una manera estándar de ejecutar su código. Como se muestra su arquitectura en Ilustración 11, Docker es un sistema operativo para contenedores. [38] De manera similar a cómo una máquina virtual virtualiza (elimina la necesidad de administrar directamente) el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor. Docker se instala en cada servidor y proporciona comandos sencillos que puede utilizar para crear, iniciar o detener contenedores.

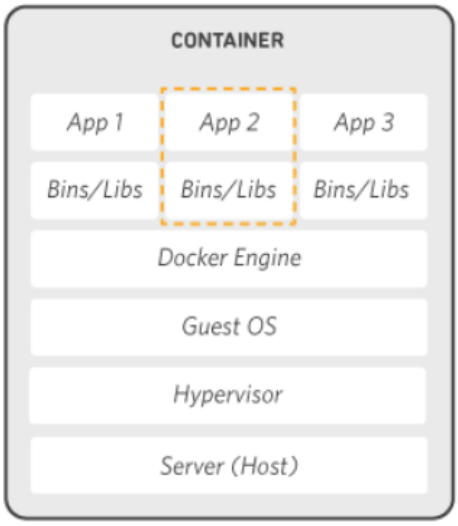

Ilustración 11 Arquitectura Docker

3.6 Herramientas auxiliares

3.6.1 Sublime Text 3

Sublime Text 3 [39] es un editor de texto multiplataforma que permite la edición de código fuente y además incluye librerías que permiten resaltar la sintaxis de infinidad de lenguajes de programación. El editor dispone funcionalidades como el autoguardado, múltiples opciones de personalización, así como edición de código y automatización de tareas.

3.6.2 YAML

YAML [40] es un lenguaje de serialización de datos que se utiliza a menudo para crear archivos de configuración y funciona en concurrencia con cualquier lenguaje de programación.

YAML es diseñado para la interacción humana. Es un superconjunto estricto de JSON, otro lenguaje de serialización de datos. Pero como es un superconjunto estricto, puede hacer todo lo que JSON puede y más. Una diferencia importante es que las nuevas líneas y la sangría significan algo en YAML, a diferencia de JSON, que utiliza corchetes y llaves.

Capítulo 4. Especificación de requisitos

A continuación, se presentan los actores identificados que interactuarán con el sistema, así como los casos de uso que se han implementado.

4.1. Actores del sistema

Los actores que se han detectado han sido dos:

- Actor sistema.

  Este actor representa al sistema. Es el actor encargado de monitorizar, detectar los errores y solucionarlos.

- Actor usuario.

  Este actor representa al usuario final del sistema.

4.2. Casos de uso

En la Ilustración 12 se muestra el diagrama de casos de uso del sistema.

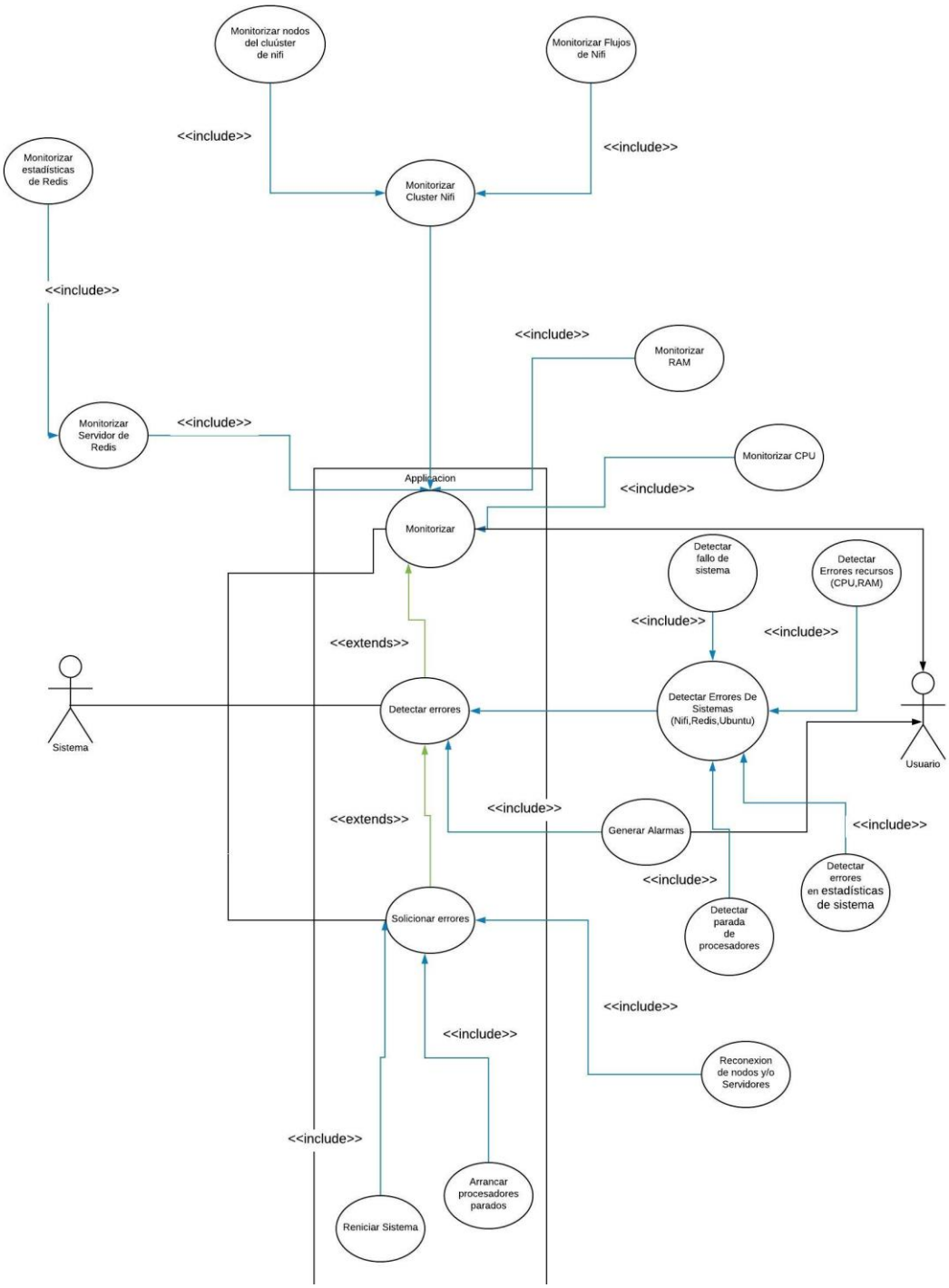

Ilustración 12 Casos de uso

A continuación, se muestran los casos de uso definidos agrupados de acuerdo con el actor que los realiza (Tabla 1).

Caso de uso	Monitorizar
Actores	Sistema, Usuario
Dependencias	Ninguno

Descripción	El Sistema debe monitorizar el funcionamiento de diferentes componentes del proyecto
Precondición	Los componentes están lanzados, funcionando y se puede acceder a los puertos de servidores.
Secuencia normal	1-Acceder a los componentes desde el protocolo asociado 2- Monitorizar el Funcionamiento de cada componente 3-Almacenar todos los resultados cada "x" tiempo en Elasticsearch para visearlos después
Postcondición	Los componentes están parados
Excepciones	1-Intentar lanzar el componente 2-Enviar un mail de error al usuario
Caso de uso	Monitorizar RAM
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe monitorizar el consumo de RAM en cada maquina
Precondición	Los máquinas están lanzados y funcionando
Secuencia normal	1-Acceder a las maquinas desde Apache NiFi 2- Monitorizar el Consumo de RAM en cada maquina
Postcondición	Los componentes están parados
Excepciones	1-Intentar lanzar el componente 2-Enviar un mail de error al usuario
Caso de uso	Monitorizar CPU
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe monitorizar el consumo de CPU en cada maquina
Precondición	Los máquinas están lanzados y funcionando
Secuencia normal	1- Colectar el consumo de CPU en la maquina cada "x" 2-Encryptiar los datos y enviarlos a Apache NiFi 3- Monitorizar el Consumo de CPU
Postcondición	Los componentes están parados
Excepciones	1-Intentar lanzar el componente 2-Enviar un mail de error al usuario
Caso de uso	Monitorizar Clúster NiFi
Actores	Sistema
Dependencias	Ninguno

Descripción	El Sistema debe monitorizar el funcionamiento del clúster de NiFi y sus procesos
Precondición	NiFi está lanzado y funcionando
Secuencia normal	1- Acceder a Apache NiFi 2-Monitorizar el funcionamiento de cada nodo y cada flujo
Postcondición	El clúster de NiFi esta parrado
Excepciones	1-Intentar reconectar el clúster 2-Enviar un mail de error al usuario
Caso de uso	Monitorizar nodos del clúster de NiFi
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe monitorizar el funcionamiento y las estadísticas de nodos del clúster de Nifi
Precondición	Los nodos del clúster están conectados
Secuencia normal	1- Acceder a Apache NiFi 2-Monitorizar el funcionamiento y las estadísticas de los nodos
Postcondición	Uno o más nodos están desconectados
Excepciones	1-Reconectar los nodos parados 2-Enviar un mail de error al usuario
Caso de uso	Monitorizar procesos de Nifi
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe monitorizar el funcionamiento de los procesos en el clúster de Nifi
Precondición	Los procesos de Nifi están arrancados
Secuencia normal	1- Acceder a Apache NiFi 2-Monitorizar el funcionamiento de los procesadores en cada flujo
Postcondición	Uno o más procesadores están parados
Excepciones	1-Arrancar los procesadores parados 2-Enviar un mail de error al usuario
Caso de uso	Monitorizar Servidor de Redis

Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe monitorizar el funcionamiento del servidor de Redis
Precondición	El servidor de Redis está funcionando
Secuencia normal	1- Conectar al servidor de Redis desde el puerto especifico 2-Monitorizar el funcionamiento de Redis y sus estadísticas.
Postcondición	El servidor esta parrado
Excepciones	1-Intentar reconectar el servidor de Redis 2-Enviar un mail de error al usuario
Caso de uso	Detectar errores
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe Detectar los errores que pueden ocurrir
Precondición	El sistema está monitorizando los componentes del proyecto
Secuencia normal	1-Detectar los fallos que ocurren
Postcondición	El sistema no está monitorizando los componentes del proyecto
Excepciones	1-Intentar el reinicio del sistema 2-Enviar un mail de error al usuario
Caso de uso	Detectar errores Sistemas (Clúster NiFi, Servidor De Redis, Ubuntu)
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe Detectar los errores que pueden ocurrir
Precondición	El sistema está monitorizando los componentes del proyecto
Secuencia normal	1-Detectar los fallos que ocurren
Postcondición	El sistema no está monitorizando los componentes del proyecto
Excepciones	1-Intentar el reinicio del sistema 2-Enviar un mail de error al usuario
Caso de uso	Detectar errores Recursos (CPU, RAM)
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe Detectar los errores que pueden ocurrir en los recursos

Precondición	El sistema este monitorizando los componentes del proyecto
Secuencia normal	1-Detectar el consumo excesivo de RAM, CPU
Postcondición	El sistema no está monitorizando los componentes del proyecto
Excepciones	1-Intentar el reinicio del sistema 2-Enviar un mail de error al usuario
Caso de uso	Detectar fallo de sistema
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe Detectar los errores que pueden ocurrir en los sistemas
Precondición	El sistema este monitorizando los componentes del proyecto
Secuencia normal	1-Detectar la desconexión de los nodos del Clúster de NiFi 2-Detectar los fallos o la desconexión de servidor en el funcionamiento de Redis 3-Detectar la parada de la máquina de Ubuntu
Postcondición	El sistema no está monitorizando los componentes del proyecto
Excepciones	1-Intentar el reinicio del sistema 2-Enviar un mail de error al usuario
Caso de uso	Detectar parada de procesadores
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe detectar la parada de procesadores de los procesos de Nifi
Precondición	El sistema este monitorizando el clúster de Nifi
Secuencia normal	1-Detectar uno o más procesadores parados
Postcondición	El sistema no está monitorizando los componentes del proyecto
Excepciones	1-Intentar el reinicio del sistema 2-Enviar un mail de error al usuario
Caso de uso	Generar Alarmas
Actores	Sistema, Usuario
Dependencias	Ninguno

Descripción	El Sistema debe generar alarmas que representan los errores que se envían al usuario
Precondición	El sistema ha detectado errores en uno de los componentes
Secuencia normal	1-Generar un fichero con todos los errores 2-Enviar la alarma al usuario para notificarlo
Postcondición	El sistema no ha detectado errores en ninguno de los componentes
Excepciones	1-Almacenar los resultados en Elasticsearch 2-Enviar un mail al usuario
Caso de uso	Solucionar Problemas
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe intentar solucionar los errores detectados
Precondición	El sistema ha detectado errores en uno de los componentes
Secuencia normal	1-Intentar la solución de problema 2-Notificar el usuario con las acciones hechas
Postcondición	El sistema no ha detectado errores en ninguno de los componentes
Excepciones	1-Almacenar los resultados en Elasticsearch 2-Enviar un mail al usuario
Caso de uso	Reiniciar la maquina
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe intentar el reinicio de la maquina en caso de los errores recursos
Precondición	El sistema ha detectado un consumo alto en CPU y/o RAM por varias ocasiones de tiempo
Secuencia normal	1-Intentar el reinicio de la maquina 2-Notificar el usuario con las acciones hechas
Postcondición	El sistema no ha detectado un consumo alto en CPU y/o RAM o ha bajado después de una ocasión de tiempo
Excepciones	1-Generar alarma para notificar al usuario 1-Almacenar los resultados en Elasticsearch 2-Enviar un mail al usuario
Caso de uso	Arrancar procesadores parados
Actores	Sistema

Dependencias	Ninguno
Descripción	El Sistema debe arrancar los procesadores parados de los procesos de Nifi
Precondición	El sistema ha detectado la parada de uno o más procesadores
Secuencia normal	1-Arrancar los procesadores parados 2-Notificar el usuario con las acciones hechas
Postcondición	El sistema no ha detectado ningún procesador parado
Excepciones	1-Almacenar los resultados en Elasticsearch 2-Enviar un mail al usuario
Caso de uso	Reconexión de nodos y/o servidor
Actores	Sistema
Dependencias	Ninguno
Descripción	El Sistema debe intentar el reinicio de la maquina en caso de los errores sistemas
Precondición	El sistema ha detectado errores en el clúster de NiFi o servidor de Redis
Secuencia normal	1-Intentar la reconexión de nodo de clúster de NiFi o Redis que ha fallado 2-Notificar el usuario con las acciones hechas
Postcondición	El sistema no ha detectado errores en NiFi y Redis
Excepciones	1-Almacenar los resultados en Elasticsearch 2-Enviar un mail al usuario

Tabla 1 Tabla de casos de uso

Capítulo 5. Arquitectura de la aplicación

En este capítulo se comentará la arquitectura diseñada y la relación entre los diferentes componentes.

La arquitectura elegida será el siguiente (Ilustración *13*).

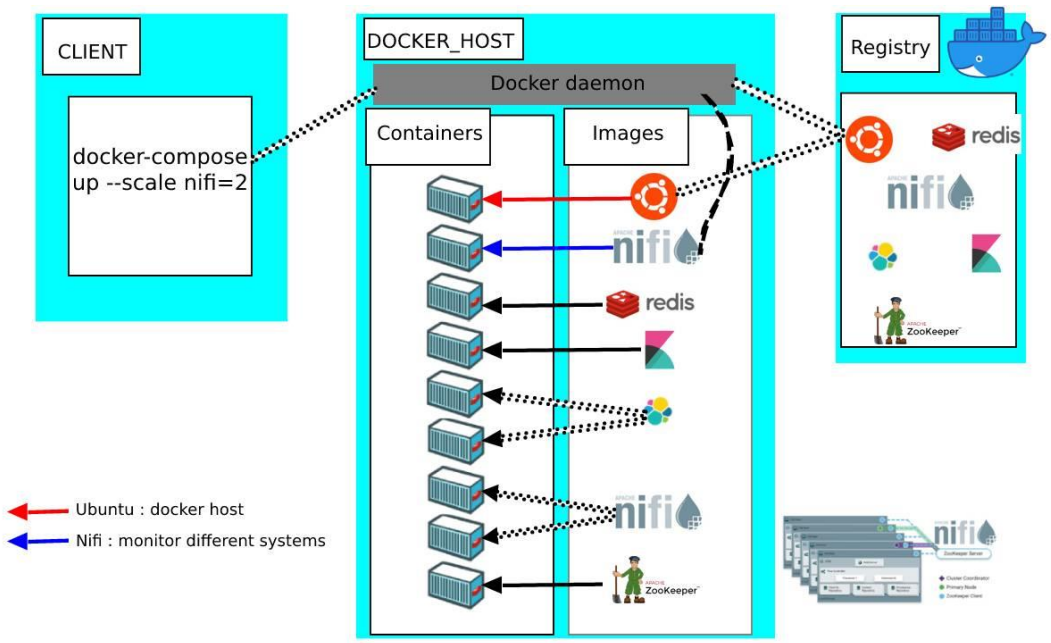

Ilustración 13 *Arquitectura del sistema*

La construcción de la plataforma se realizará como un desarrollo basado en dockers. Y para ello se utiliza una arquitectura cliente-servidor. El cliente Docker habla con el demonio Docker, que hace el trabajo pesado de construir, ejecutar y distribuir los contenedores Docker. El cliente Docker y el demonio se ejecutan en el mismo sistema. El cliente Docker y el demonio se comunican utilizando una API REST, a través de sockets UNIX o una interfaz de red. El cliente Docker es Docker Compose, que permite trabajar con aplicaciones formadas por un conjunto de contenedores.

Un registro Docker almacena imágenes Docker. Docker Hub es un registro público que cualquiera puede utilizar, y Docker está configurado para buscar imágenes en Docker Hub por defecto. Incluso une persona puede ejecutar su propio registro privado.
Cuando se usa el comando docker-compose up, las imágenes requeridas se extraen del registro configurado.

Una imagen es una plantilla de sólo lectura con instrucciones para crear un contenedor Docker. A menudo, una imagen se basa en otra imagen, con alguna personalización adicional. Por ejemplo, la imagen de Ubuntu usada se basa en la imagen original de Ubuntu, pero instala el servidor ssh, así como los detalles de configuración necesarios para hacer que su aplicación se ejecute. Se puede crear sus propias imágenes o se puede

utilizar sólo las creadas por otros y publicadas en un registro. Para construir su propia imagen, se crea un Dockerfile con una sintaxis simple para definir los pasos necesarios para crear la imagen y ejecutarla. Cada instrucción en un Dockerfile crea una capa en la imagen. Cuando se cambia el Dockerfile y se reconstruye la imagen, sólo se reconstruyen las capas que han cambiado. Esto es parte de lo que hace que las imágenes sean tan ligeras, pequeñas y rápidas, en comparación con otras tecnologías de virtualización.

Un contenedor es una instancia ejecutable de una imagen. Puede crear, iniciar, detener, mover o eliminar un contenedor utilizando la API de Docker o la CLI. Se puede conectar un contenedor a una o más redes, adjuntarle almacenamiento o incluso crear una nueva imagen basada en su estado actual. Por defecto, un contenedor está relativamente bien aislado de otros contenedores y de su máquina anfitriona. Se controla el grado de aislamiento de la red, el almacenamiento u otros subsistemas subyacentes de un contenedor con respecto a otros contenedores o a la máquina anfitriona.

Un contenedor está definido por su imagen, así como por las opciones de configuración que se le proporciona cuando se crea o se inicia. Cuando se elimina un contenedor, cualquier cambio en su estado que no esté almacenado en el almacenamiento persistente desaparece. Buscando alguna analogía con el mundo real se pueden asemejar a los contenedores que son transportados en barco de un sitio a otro, no importa su contenido sino su forma modular para ser almacenados y transportados de un sitio a otro como cajas.

Este tipo de arquitecturas permiten escalar la solución en función de las necesidades de los sistemas construidos sobre ellos, además de ofrecer de forma sencilla una arquitectura orientada a servicios que permita añadir nuevos componentes de forma sencilla a la solución.

En la plataforma, se va a usar Docker para crear el entorno de trabajo que contiene los siguientes contenedores:

- Un clúster de NiFi de dos nodos
- NiFi standalone que va a monitorizar todo el sistema
- Zookeeper para manejar los nodos del clúster
- Un clúster de Elasticsearch de dos nodos
- Kibana
- Redis
- Una máquina de Ubuntu

Desde NiFi se pueden ejecutar comandos y capturar datos de los diferentes sistemas para verificar si han producido errores. Una vez se encuentre un fallo, se genera un fichero json con los datos del fallo, se intenta solucionar el problema y se envía alarma al cliente y se indexan los datos en Elasticsearch.

En Elasticsearch, se crea un índice para almacenar los datos y con Kibana se crea visualizaciones de unas series temporales para que puedan ser analizadas posteriormente y poder extraer conclusiones (rapidez a la hora de detectar un problema en la caída de un servicio y su recuperación, detección de patrones en los fallos del sistema, análisis de los datos de rendimiento que permitan realizar un mantenimiento preventivo del sistema, etc).

Capítulo 6. Modelo de datos

En este capítulo se comentará el modelo de datos. De una parte, se define la estructura de datos de Elasticsearch. De otra parte, se define los mapeos de cada índice usado.

Se ha usado Elasticsearch para almacenar los datos. Elasticsearch es un almacén de documentos distribuido. En lugar de almacenar información como filas de datos en columnas, Elasticsearch almacena estructuras de datos complejas que han sido serializadas como documentos JSON.

Elasticsearch utiliza por medio de la implementación de apache Lucene una estructura de datos denominada índice inverso que permite realizar búsquedas de texto libre de forma rápida. Un índice inverso es un tipo de estructura semejante al glosario de un libro basado en la asociación de determinados términos y documentos. El índice inverso se crea en tiempo de análisis de los documentos sobre los que se basará la búsqueda y asocia a cada documento los términos que contiene, en una BBDD relacional los índices se crean cuando se diseña el modelo de datos. Un índice puede considerarse como una colección optimizada de documentos y cada documento es una colección de campos, que son los pares clave-valor que contienen sus datos.

Para cada índice, se crea un mapping que es el proceso de definir cómo se almacena e indexa un documento y los campos que contiene. Por ejemplo, se utilizan mapeos para definir:

- Qué campos de cadena deben ser tratados como campos de texto completo.
- Qué campos contienen números, fechas o geolocalizaciones.
- El formato de los valores de las fechas.
- Reglas personalizadas para controlar la asignación de los campos añadidos dinámicamente.

Cada índice tiene un tipo de mapeo que determina cómo se indexará el documento. Un tipo de mapeo tiene:

- Meta-campos: los meta-campos se utilizan para personalizar el tratamiento de los metadatos asociados a un documento. Algunos ejemplos de meta-campos son los campos _index, _type, _id y _source del documento.
- Campos o propiedades
- Un tipo de mapeo contiene una lista de campos o propiedades pertinentes al documento.

Cada campo tiene un tipo de datos que puede ser:

- Un tipo simple como: text, keyword, date, long, double, boolean or ip.
- Un tipo que soporte la naturaleza jerárquica de JSON como object o nested.

- Un tipo especializado como geo_point, geo_shape o completion.

Para la definición de índices se usa:

- Template:
  Las plantillas de índices definen configuraciones y mapeos que se pueden aplicar automáticamente al crear nuevos índices. Elasticsearch aplica las plantillas a los nuevos índices basándose en un patrón de índice que coincide con el nombre del índice. Las plantillas de índice sólo se aplican durante la creación del índice. Los cambios en las plantillas de índices no afectan a los índices existentes. Las configuraciones y mapeos especificados en las solicitudes de la API de creación de índices anulan cualquier configuración o mapeo especificado en una plantilla de índice.

A continuación, se muestra la lista de los mapeos de cada índice usado (Tabla 2)

Ubuntu	Redis	NiFi	Elasticsearch
{ "_id": { "type": "date" }, "fecha": { "type": "date" }, "nodename": { "type": "keyword" }, "statistics": { "cpu-load": { "idle": { "type": "float" } }, "memory": { "avail": { "type": "long" }, "memfree": { "type": "long" }, "memused-percent": { "type": "float" }	{ "_id": { "type": "date" }, "fecha": { "type": "date" }, "redis": { "clients": { "blocked_clients": { "type": "long" }, "connected_clients": { "type": "long" }, "maxclients": { "type": "long" }, "tracking_clients": { "type": "long" } }, "cluster": { "cluster_enabled": { "type": "long" } }, "cpu": { "used_cpu_sys": { "type": "float" }, "used_cpu_user": {	{ "_id": { "type": "date" }, "cluster": { "nodes": { "address": { "type": "keyword" }, "nodeId": { "type": "keyword" }, "status": { "type": "keyword" } } }, "fecha": { "type": "date" }, "systemDiagnostics": { "aggregateSnapshot": { "properties": { "heapUtilization": { "type": "float" }, "maxHeapBytes": { "type": "long" },	{ "_id": { "type": "date" }, "jvm": { "mem": { "heap_max_in_bytes": { "type": "long" }, "heap_used_in_bytes": { "type": "long" }, "heap_used_percent": { "type": "long" } } }, "name": { "type": "keyword" }, "os": { "cpu": { "percent": { "type": "long" } },

```
    },
    "neatwork": {
      "net-dev": {
        "iface": {
          "type": "keyword"
        },
        "ifutil-percent": {
          "type": "long"
        },
        "rxpck": {
          "type": "long"
        },
        "txpck": {
          "type": "long"
        }
      }
    }
  }
}
```

```
          "type": "float"
        }
      },
      "errorstats": {
        "type": "object"
      },
      "keyspace": {
        "type": "object"
      },
      "memory": {
        "maxmemory": {
          "type": "long"
        },

"mem_clients_normal": {
          "type": "long"
        },
        "mem_clients_slaves": {
          "type": "long"
        },

"mem_fragmentation_ratio": {
          "type": "float"
        },
        "used_memory": {
          "type": "long"
        },

"used_memory_dataset": {
          "type": "long"
        },
        "used_memory_lua": {
          "type": "long"
        }
      },
      "persistence": {

"rdb_changes_since_last_save": {
          "type": "long"
        },
        "rdb_last_save_time": {
          "type": "long"
        }
      },
      "replication": {
        "connected_slaves": {
          "type": "long"
        }
      },
```

```
"nonHeapUtilization": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",

"ignore_above": 256
            }
          }
        },

"processorLoadAverage": {
          "type": "float"
        }
      }
    }
  }
}
```

```
    "mem": {
      "free_in_bytes": {
        "type": "long"
      },
      "free_percent": {
        "type": "long"
      },
      "total_in_bytes": {
        "type": "long"
      },
      "used_in_bytes": {
        "type": "long"
      },
      "used_percent": {
        "type": "long"
      }
    },
    "swap": {
      "free_in_bytes": {
        "type": "long"
      },
      "total_in_bytes": {
        "type": "long"
      },
      "used_in_bytes": {
        "type": "long"
      }
    }
  }
}
```

	"stats": {
	
	"instantaneous_ops_per_sec": {
	"type": "long"
	},
	"keyspace_hits": {
	"type": "long"
	},
	"keyspace_misses": {
	"type": "long"
	},
	
	"rejected_connections": {
	"type": "long"
	}
	}
	}
	}		

Tabla 2 Tabla de mapeos

Cada índice contiene la información relativa a campos que interesa para evaluar el funcionamiento de la máquina de Ubuntu, servidor Redis, clúster NiFi y clúster Elasticsearch. La mayoría de los campos usados están definidos en la tabla de mapeos (Tabla 2). Además, cada uno tiene un campo de fecha que representa la hora y día en el que se va a indexar los datos, lo que permite crear visualizaciones de series temporales y visualizar datos en el discover de kibana por cada ROP.

Capítulo 7. Diseño del sistema

En ese capítulo se explicará el diseño del sistema. Primero, se explicará el entorno de Docker desarrollado. Después, se definirá la tabla de métricas que hay que monitorizar. Finalmente, se detallará un ejemplo de caso de uso del flujo de Apache Nifi y un ejemplo de dashboard de Kibana donde se visualizan los datos almacenados en Elasticseach.

7.1 Entorno Docker

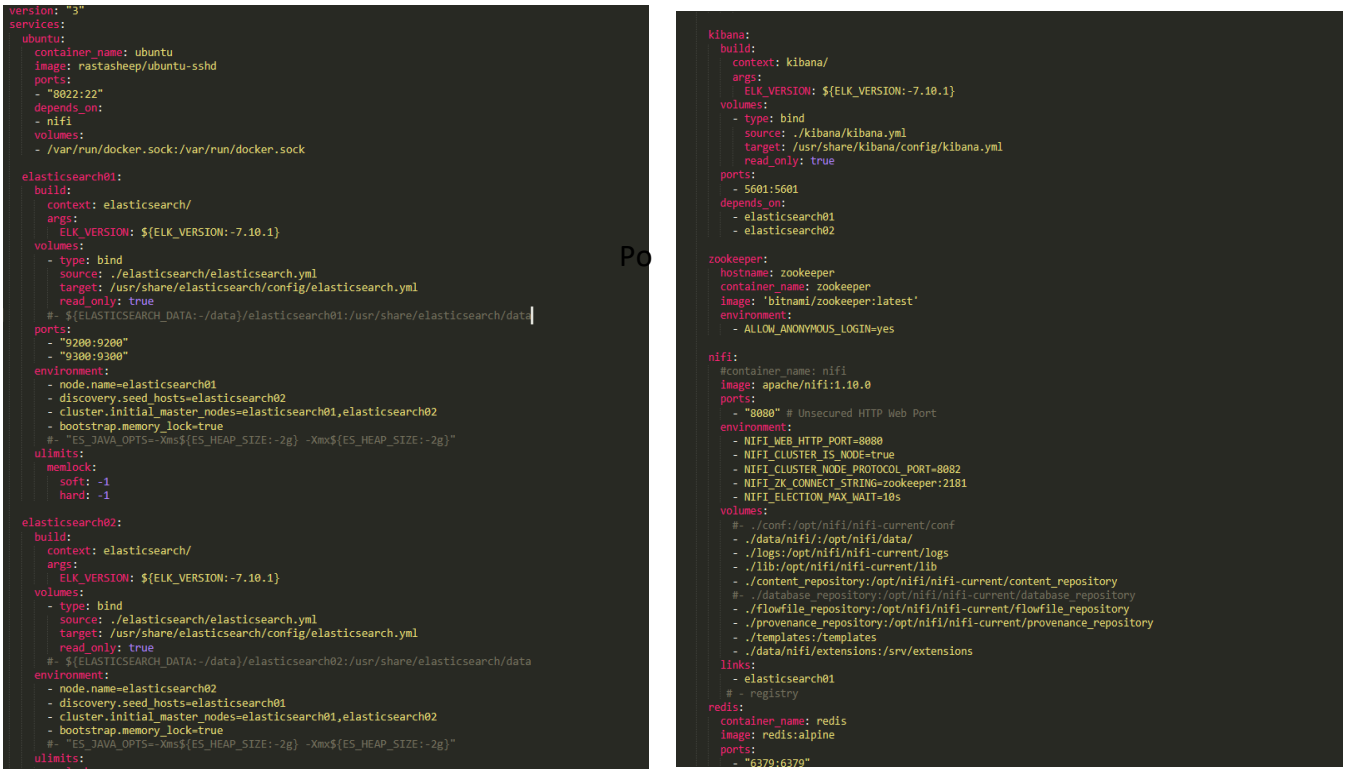

Ilustración 14 Entorno Docker

Se ha creado un fichero de docker-compose que lleva todos los sistemas que se van a monitorizar. Se ha definido unos parámetros para el buen funcionamiento de estos y para guardar el trabajo hecho en cada container.

7.2 Tabla de métricas y errores, alarmas y acciones

Sistema	Métricas y errores	Alarmas	Acciones
Ubuntu	-Uso excesivo de Memoria	>80%	-Enviar Alarma al cliente
	-Consumo excesivo de CPU	>90%	-Enviar Alarma al cliente
	-Perdida del servicio	No se puede conectar a la	-Enviar Alarma al cliente

		maquina	-Reinicio del sistema
Redis	-Uso excesivo de memoria	>80%	-Enviar Alarma al cliente
	-mem_fragmentation_ratio	>1,5	-Enviar Alarma al cliente
	-clientes bloqueados	>0	-Enviar Alarma al cliente
	-keyspace_misses	>0	-Enviar Alarma al cliente
	-instantaneous_ops_per_sec (Número de comandos procesados per segundo.)	None	None
	-rdb_changes_since_last_save	None	None
	-rdb_last_save_time	None	None
	-conexiones rechazadas	None	None
	-Perdida del servicio	No se puede conectar a la maquina	-Enviar Alarma al cliente -Reinicio del sistema
NiFi	-Nodos Conectados	-Uno o mas nodos desconectados	-Enviar Alarma al cliente -Reinicio del nodo desconectado
	-Funcionamiento de procesos	-Uno o mas Procesadores parados	-Enviar Alarma al cliente -Arrancar los procesadores desde NiFi Api
Elasticsearch	-Cluster health	¡=green	-Enviar Alarma al cliente
	-Nodos Conectados	-Uno o mas nodos desconectados	-Enviar Alarma al cliente -Reinicio del nodo desconectado
	-Uso excesivo de memoria JVM	>80%	-Enviar Alarma al cliente
	-Uso excesivo de memoria de sistema	>85%	-Enviar Alarma al cliente
	-Consumo de CPU	>90%	-Enviar Alarma al cliente

Tabla 3 Tabla de métricas y errores, alarmas, acciones

7.3 Flujo Apache NiFi

Se explicará un ejemplo de caso de uso de monitorización y solución de problemas de Redis desde NiFi. En la Ilustración 15 se muestra los dos procesos creados para monitorizar Redis.

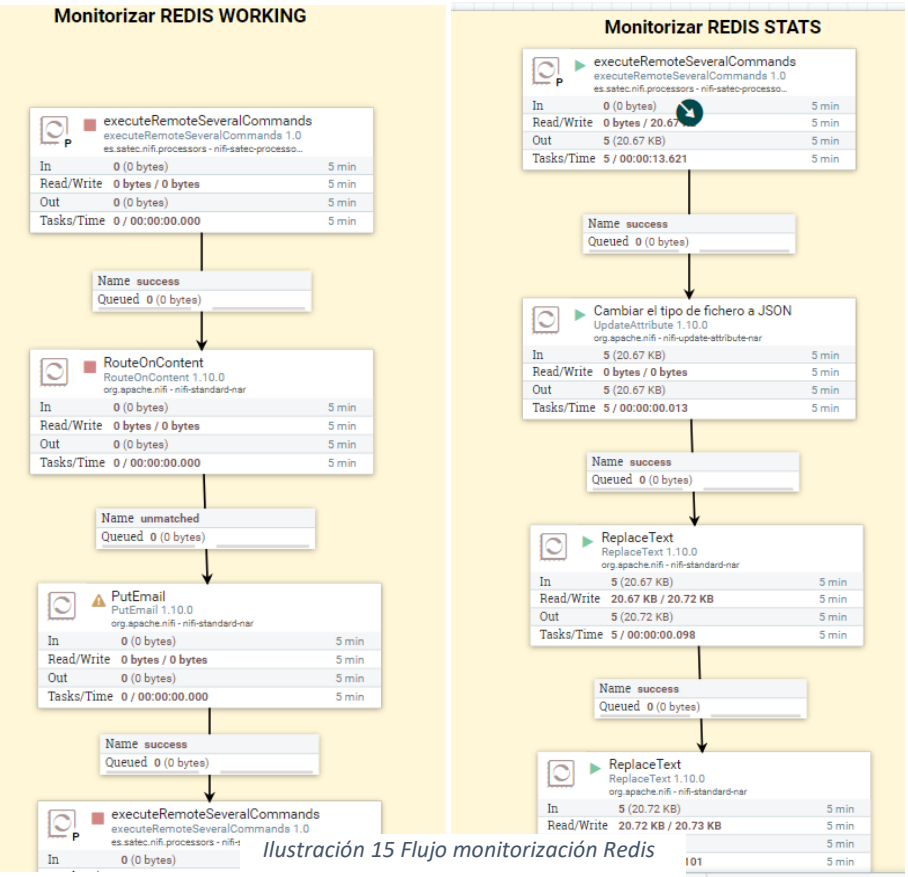

Ilustración 15 Flujo monitorización Redis

El flujo con el nombre Monitorizar REDIS STATS está encargado de las siguientes tareas:

- Recuperar las estadísticas de Redis con el comando redis-cli INFO
- Sacar como atributos las métricas que se quieren monitorizar
- Verificar si hay errores
- Enviar alarma en caso de encontrar problemas
- Ingesta de datos en Elasticsearch

De otra parte, el flujo Monitorizar Redis Working está encargado de las siguientes tareas:

- Hacer un ping al servidor de Redis
- Verificar si hay repuesta
- En caso de pérdida del servicio envía una alarma al cliente
- Reiniciar el servicio de Redis

7.4 Dashboards kibana

Una vez se tienen los datos de estatus de Redis almacenados en Elasticsearch, se puede crear un dashboard (Ilustración 16) con visualizaciones para que pueda ser analizada posteriormente y poder extraer conclusiones.

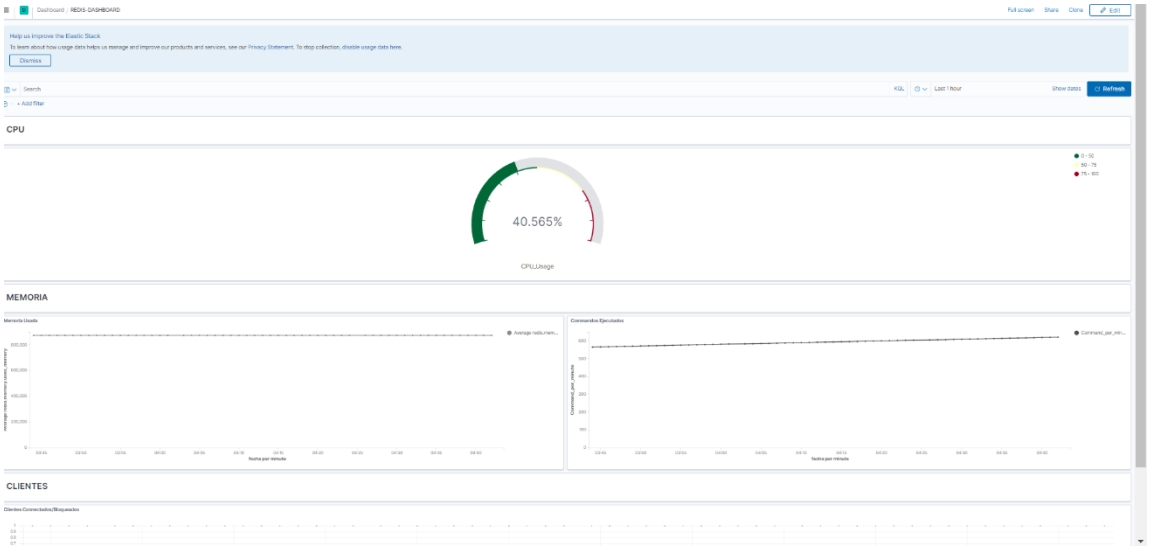

Ilustración 16 Dashboard Redis

Capítulo 8. Evaluación

En este capítulo se va a probar unos casos de uso de la plataforma que permitan mostrar algunas de las funcionalidades del sistema construido.

Para levantar el sistema, se lanza el Docker client con el comando “docker-compose up --scale NiFi=2” que permitirá generar y levantar los diferentes contenedores del entorno (--scale NiFi=2 permite lanzar un clúster de NiFi con dos nodos).

8.1 Monitorizar CPU y Memoria de Ubuntu et NiFi

En la Ilustración *17* se muestra los dos procesos usados para monitorizar la máquina de Ubuntu y el clúster de Apache Nifi

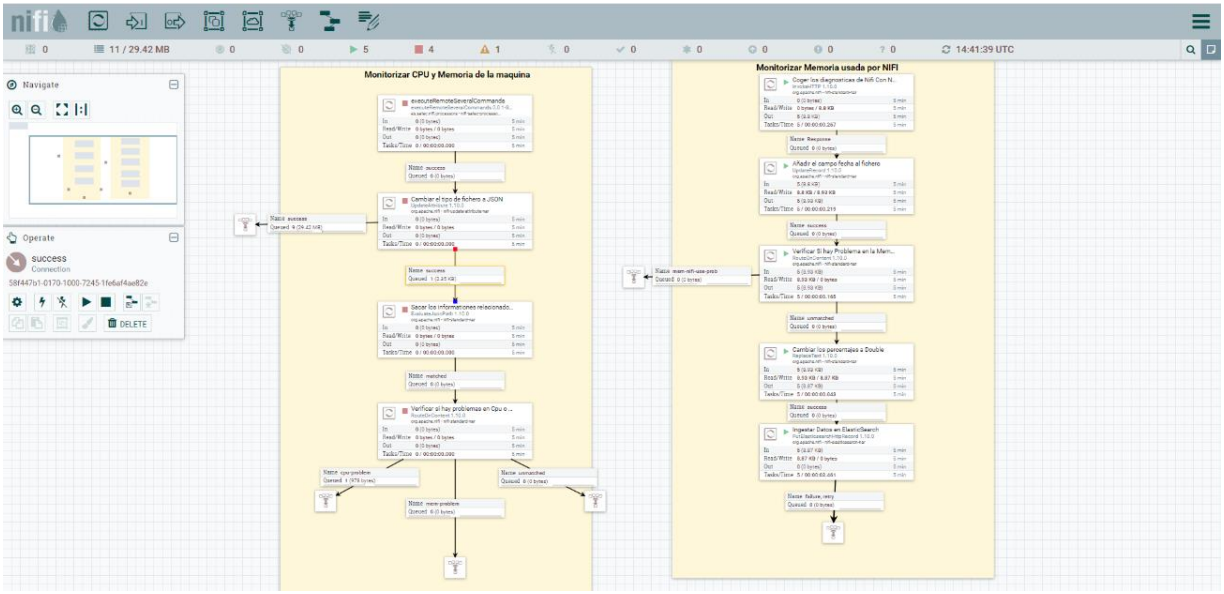

Ilustración 17 Flujo Monitorización Ubuntu y Nifi

Monitorizar CPU y Memoria de la maquina (Flujo 1):

- Primero se ha desarrollado un custom procesador para ejecutar diferentes comandos desde Apache NiFi a una maquina Linux con ssh. El procesador se llama ExecuteRemoteSeveralComandos. En la Ilustración 18 se puede ver su configuración.

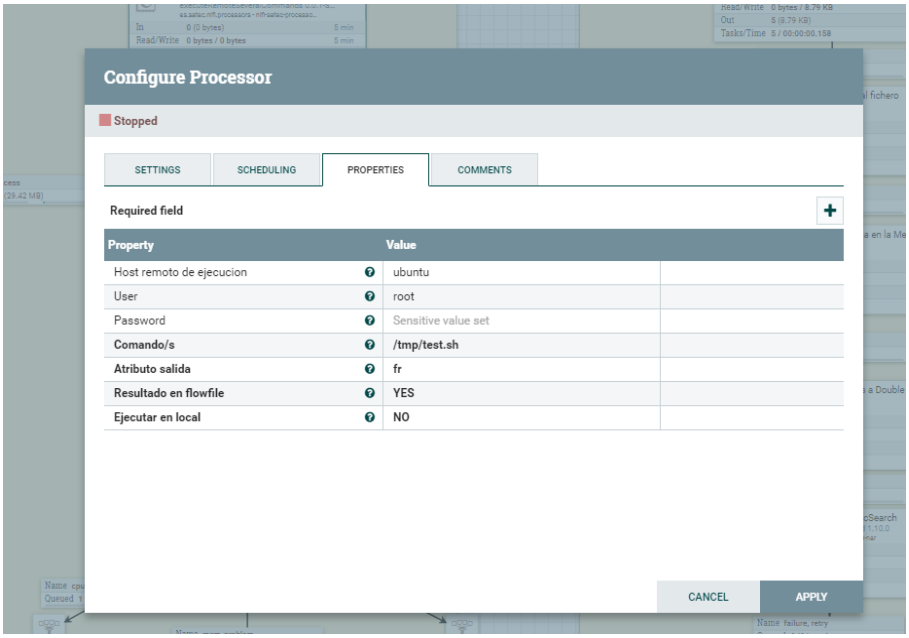

Ilustración 18 Configuración de ExecuteRemoteSeveralComandos Processor

- Se han evaluado dos scripts y al final se ha quedado con el segundo script porque genera un JSON que contiene los datos necesarios. Que permitirá de forma sencilla procesarlo después e indexarlo en Elasticsearch. En la Ilustración 19 y la Ilustración 20 se muestran los resultados de los dos scripts.

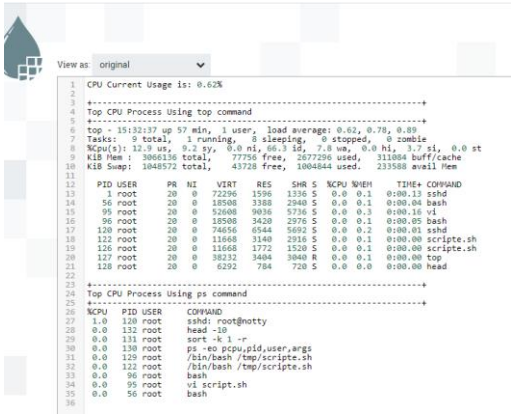

Ilustración 19 Resultado del primer script

Ilustración 20 Resultado del segundo Script

- Después se realiza una transformación del fichero a JSON para simplificar la indexación en Elasticsearch.

- Después se verifica si hay un consumo de CPU o Memoria > 80%. En la siguiente imagen se ve que hay un consumo alto en CPU.

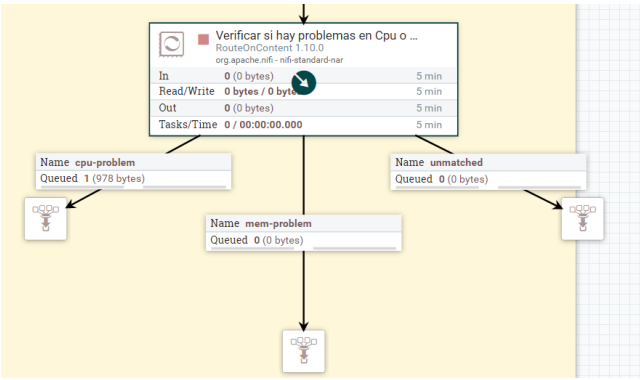

Ilustración 21 Problema en el CPU

- Finalmente, se almacenan los datos en Elasticsearch.

Monitorizar Memoria usada por NIFI:

- Se usa el NiFi REST api para sacar las estadísticas del consumo de NiFi (La diferencia aquí es que se ve el consumo de NiFi de la memoria de java asociado a él). Para ello se usa el InvokeHttp processor definido por NiFi. En la Ilustración 22 se muestra la configuración del procesador.

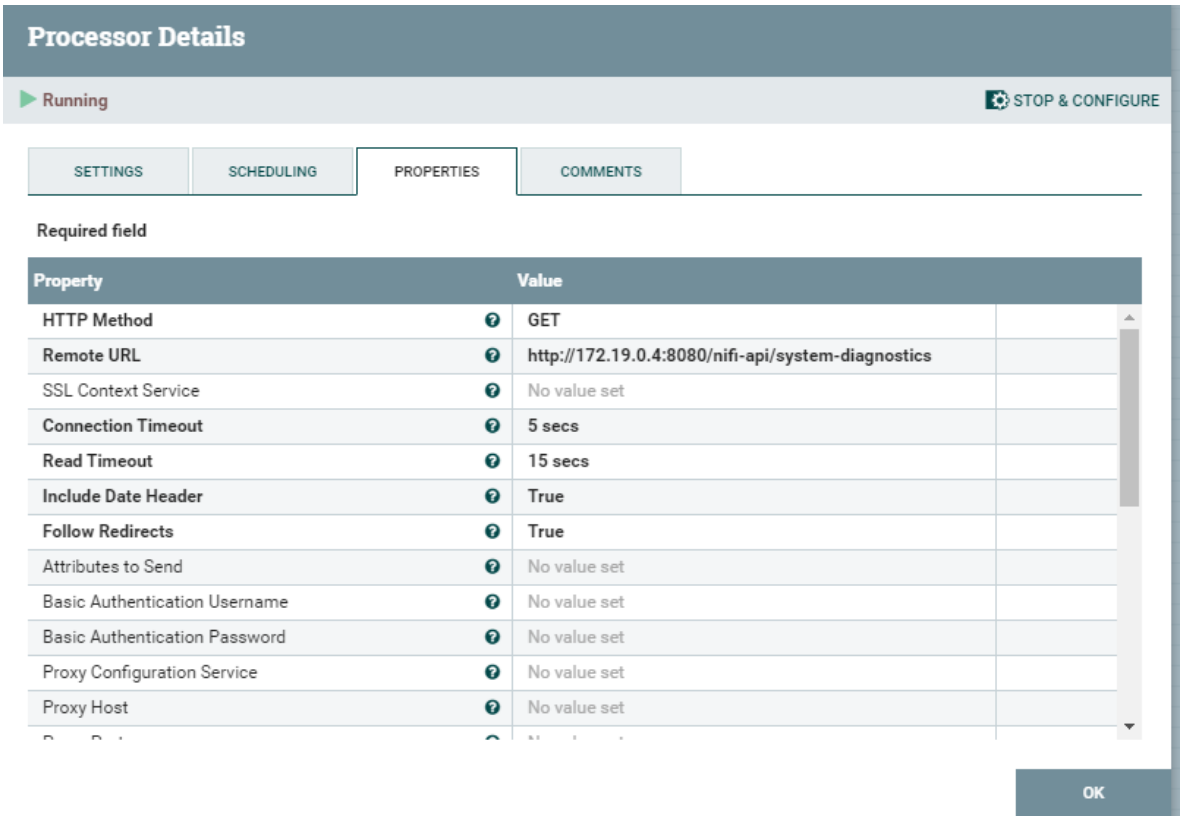

Ilustración 22 Configuración del InvokeHttp

View as: formatted


```
[ {
  "systemDiagnostics" : {
    "aggregateSnapshot" : {
      "totalNonHeap" : "219.15 MB",
      "totalNonHeapBytes" : 229793792,
      "usedNonHeap" : "206.29 MB",
      "usedNonHeapBytes" : 216315376,
      "freeNonHeap" : "12.85 MB",
      "freeNonHeapBytes" : 13478416,
      "maxNonHeap" : "-1 bytes",
      "maxNonHeapBytes" : -1,
      "totalHeap" : "506 MB",
      "totalHeapBytes" : 530579456,
      "usedHeap" : "268.88 MB",
      "usedHeapBytes" : 281940480,
      "freeHeap" : "237.12 MB",
      "freeHeapBytes" : 248638976,
      "maxHeap" : "506 MB",
      "maxHeapBytes" : 530579456,
      "heapUtilization" : "53.0%",
      "availableProcessors" : 2,
      "processorLoadAverage" : 4.16,
      "totalThreads" : 77,
      "daemonThreads" : 36,
      "uptime" : "03:03:36.958",
      "flowFileRepositoryStorageUsage" : {
        "freeSpace" : "162.13 GB",
        "totalSpace" : "464.82 GB",
        "usedSpace" : "302.69 GB",
        "freeSpaceBytes" : 174083514368,
        "totalSpaceBytes" : 499091906560,
        "usedSpaceBytes" : 325008392192,
        "utilization" : "65.0%"
      },
      "contentRepositoryStorageUsage" : [ {
        "identifier" : "default",
        "freeSpace" : "162.13 GB",
        "totalSpace" : "464.82 GB",
        "usedSpace" : "302.69 GB",
        "freeSpaceBytes" : 174083514368,
        "totalSpaceBytes" : 499091906560,
        "usedSpaceBytes" : 325008392192,
        "utilization" : "65.0%"
      } ],
      "provenanceRepositoryStorageUsage" : [
        "identifier" : "default",
        "freeSpace" : "162.13 GB",
        "totalSpace" : "464.82 GB",
        "usedSpace" : "302.69 GB",
        "freeSpaceBytes" : 174083514368,
        "totalSpaceBytes" : 499091906560,
        "usedSpaceBytes" : 325008392192,
        "utilization" : "65.0%"
      } ],
      "garbageCollection" : [ {
        "name" : "PS Scavenge",
        "collectionCount" : 131,
        "collectionTime" : "00:00:56.940",
        "collectionMillis" : 56940
      }, {
        "name" : "PS MarkSweep",
        "collectionCount" : 4,
        "collectionTime" : "00:00:00.906"
```


Ilustración 23 Resultado de la recuesta al api

- Desde el resultado de la recuesta al api que se muestra en la Ilustración 23 se cogen los datos de la memoria y verificamos si el consumo de memoria > 80%
- Después hay que tomar la decisión de la solución y alertas
- Al final se indexan los datos en Elasticsearch.
- Se ha creado un índice. Se configura todo su policy, mapping, rollover, ...
- Después se ha creado un dashboard que contiene una visualización línea del consumo de Memoria Java desde NiFi y una búsqueda en el que se puede ver maxHeap, Freeheap, usedHeap. Se puede controlar el Intervalo de tiempo. Como se muestra en la Ilustración 24 por ejemplo se ha consultado sobre la última hora.

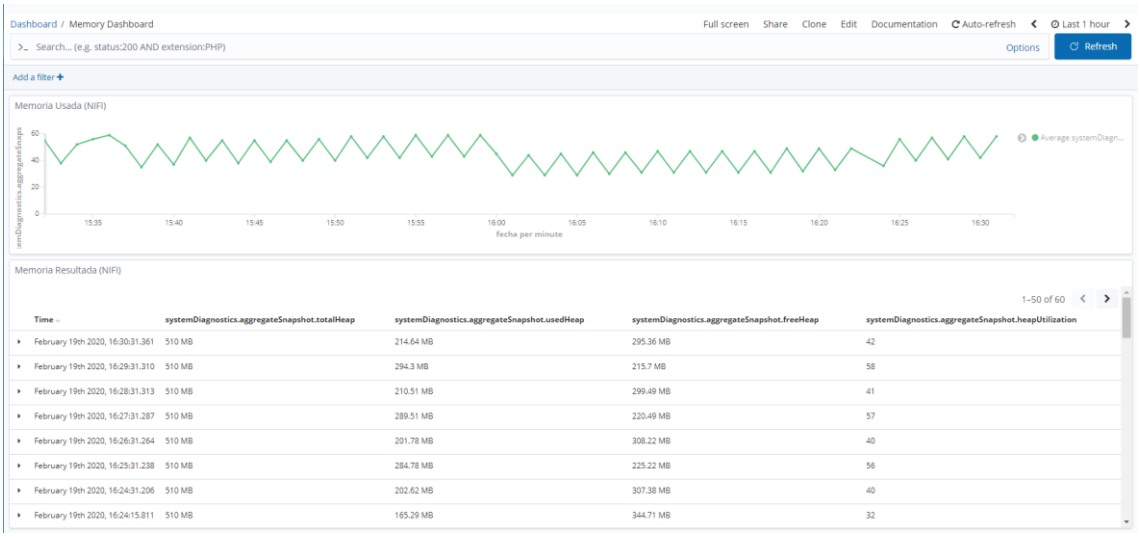

Ilustración 24 Dashboard de la Memoria usada por NIFI

8.2 Monitorizar, detectar y solucionar errores de Redis

8.2.1 Monitorizar estadísticas de Redis

En la Ilustración 25 se muestra el flujo para monitorizar las estadísticas de Redis

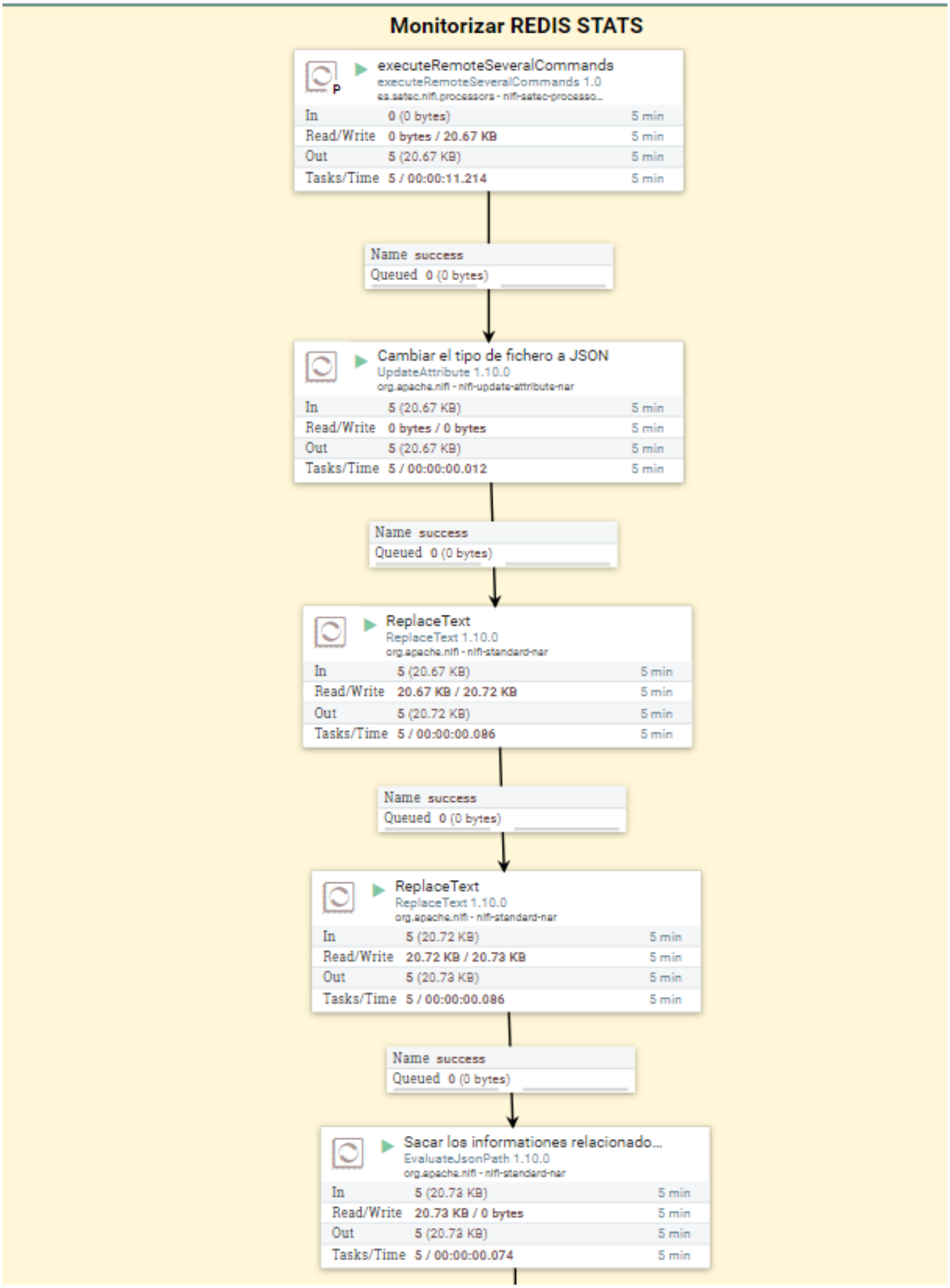

Ilustración 25 Flujo de monitorización de estadísticas de Redis

- Primero se ejecuta redis-cli INFO por el servidor de Redis que usa un script para generar como resultado el json que se muestra en la Ilustración 26.

View as: formatted


```
{
  "redis" : {
    "server" : {
      "redis_version" : "6.2.2",
      "redis_git_sha1" : 0,
      "redis_git_dirty" : 0,
      "redis_build_id" : "4c2797f59647982a",
      "redis_mode" : "standalone",
      "os" : "Linux 5.4.72-microsoft-standard-WSL2 x86_64",
      "arch_bits" : 64,
      "multiplexing_api" : "epoll",
      "atomicvar_api" : "atomic-builtin",
      "gcc_version" : "10.2.1",
      "process_id" : 1,
      "process_supervised" : "no",
      "run_id" : "ca675e616145d334c431320b29809e3c988d7de0",
      "tcp_port" : 6379,
      "server_time_usec" : 1621221746459303,
      "uptime_in_seconds" : 90991,
      "uptime_in_days" : 1,
      "hz" : 10,
      "configured_hz" : 10,
      "lru_clock" : 10609010,
      "executable" : "/data/redis-server",
      "config_file" : null,
      "io_threads_active" : 0
    },
    "clients" : {
      "connected_clients" : 1,
      "cluster_connections" : 0,
      "maxclients" : 10000,
      "client_recent_max_input_buffer" : 0,
      "client_recent_max_output_buffer" : 0,
      "blocked_clients" : 0,
      "tracking_clients" : 0,
      "clients_in_timeout_table" : 0
    },
    "memory" : {
      "used_memory" : 873552,
      "used_memory_human" : "853.08K",
      "used_memory_rss" : 11345920,
      "used_memory_rss_human" : "10.82M",
      "used_memory_peak" : 873552,
      "used_memory_peak_human" : "853.08K",
      "used_memory_peak_perc" : "100.00%",
      "used_memory_overhead" : 809824,
      "used_memory_startup" : 809824,
      "used_memory_dataset" : 63728,
      "used_memory_dataset_perc" : "100.00%",
      "allocator_allocated" : 889960,
      "allocator_active" : 1196032,
      "allocator_resident" : 4009984,
      "total_system_memory" : 26699833344,
      "total_system_memory_human" : "24.87G",
      "used_memory_lua" : 37888,
      "used_memory_lua_human" : "37.00K",
      "used_memory_scripts" : 0,
      "used_memory_scripts_human" : "0B",
      "number_of_cached_scripts" : 0,
      "maxmemory" : 0,
      "maxmemory_human" : "0B",
      "maxmemory_policy" : "noeviction",
      "allocator_frag_ratio" : 1.34,
      "allocator_frag_bytes" : 306072,
      "allocator_rss_ratio" : 3.35,
      "allocator_rss_bytes" : 2813952,
```


Ilustración 26 Redis estadísticas

- Después se sacan las métricas que se quieren monitorizar. En la Ilustración 27 se muestra los atributos definidos para coger los valores de métricas con JsonPathy en la Ilustración 28 se muestra ejemplos de valores por las métricas *"rdb_changes_since_last_save, rdb_last_save_time, conexiones rechazadas y memoria usada"*

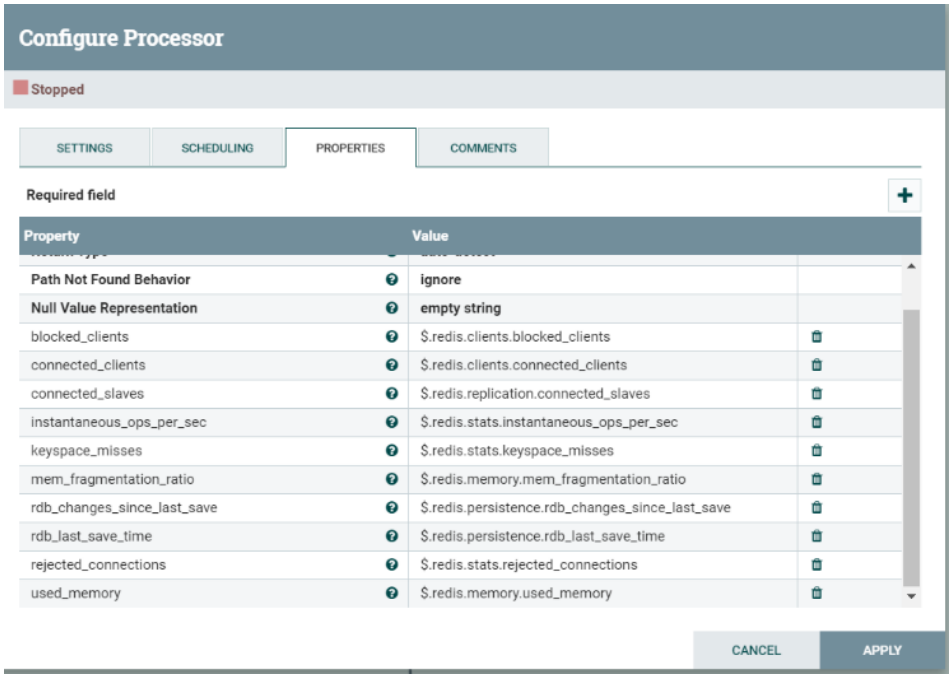

Ilustración 27 Sacar las métricas con JsonPath

rdb_changes_since_last_save
0

rdb_last_save_time
1621130755

rejected_connections
0

used_memory
873552

Ilustración 28 Ejemplo de valores de métricas sacadas

Redis	-Uso excesivo de memoria	>80%	-Enviar Alarma al cliente
	-mem_fragmentation_ratio	>1,5	-Enviar Alarma al cliente
	-clientes bloqueados	>0	-Enviar Alarma al cliente
	-keyspace_misses	>0	-Enviar Alarma al cliente
	-instantaneous_ops_per_sec (Número de comandos procesados per segundo.)	None	None

	rdb_changes_since_last_save	None	None
	-rdb_last_save_time	None	None
	-conexiones rechazadas	None	None
	-Perdida del servicio	No se puede conectar a la maquina	-Enviar Alarma al cliente -Reinicio del sistema

Tabla 4 Tabla de métricas, alarmas y acciones

- Se verifica si hay una alarma que hay que enviar al cliente. Las condiciones para generar alarmas están definidas en la Tabla 4
- Se indexan los datos en Elasticsearch y se crea un dashboard. En la Ilustración 29 se muestra la media de valores de unas métricas en 1 hora

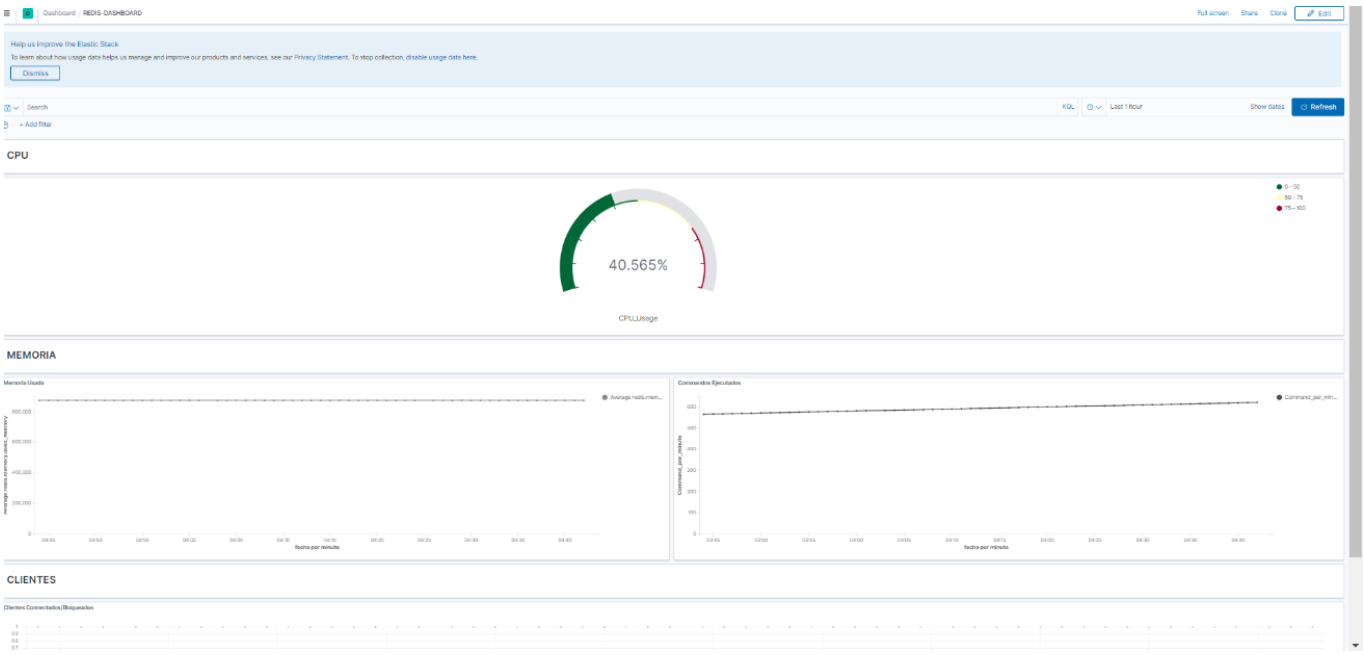

Ilustración 29 Dashboard Redis

8.2.2 Monitorizar el funcionamiento de Redis

En la Ilustración 30 se muestra el flujo para monitorizar el funcionamiento de Redis

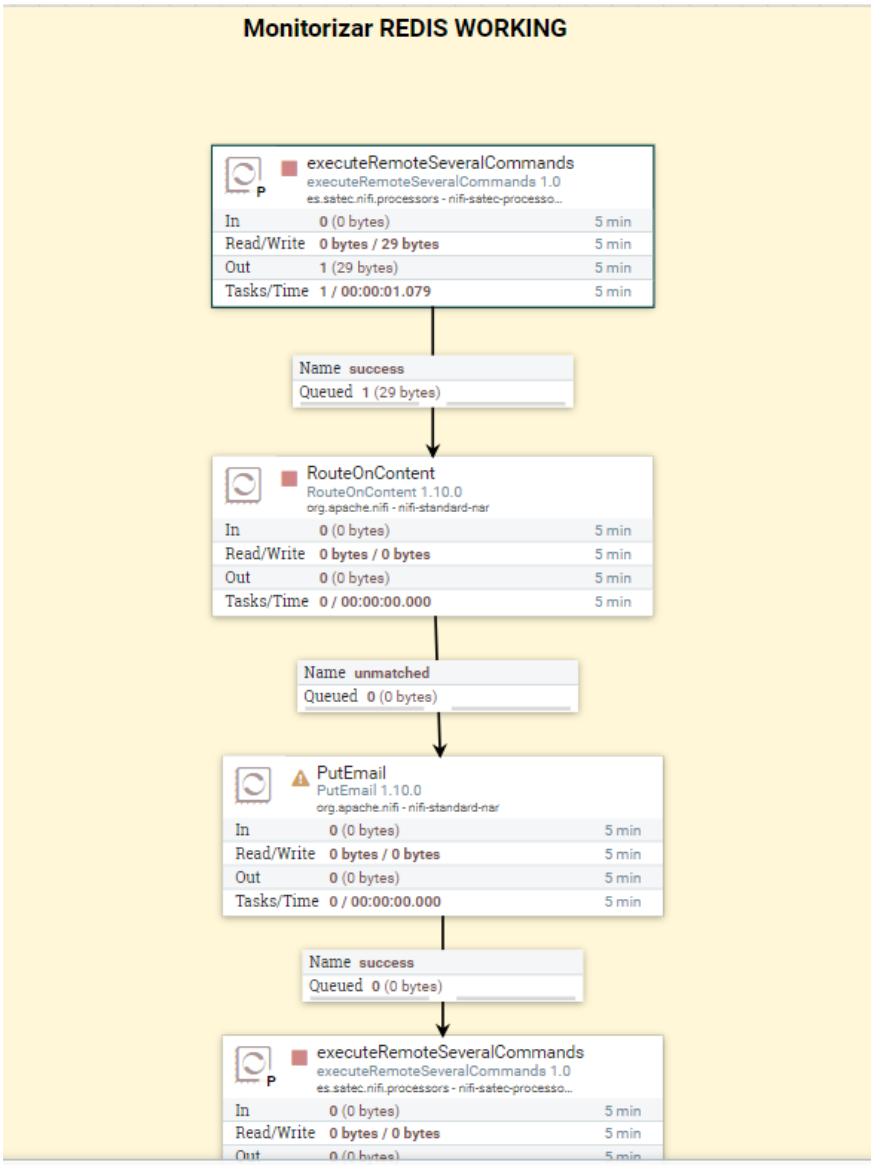

Ilustración 30 Flujo de monitorización del servidor de Redis

- Primero se hace un ping al servidor de Redis. Si el servidor funciona se debe tener la repuesta PONG (Ilustración 31)

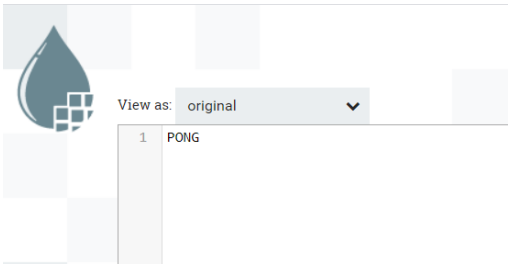

Ilustración 31 Resultado del comando

- En el caso que el servidor falla se enviar un correo al cliente y se reinicia el contenedor de Redis desde NiFi con el comando que se muestra en la Ilustración 32 (se ha usado $(docker ps -a | grep "redis") porque como necesitamos que todo el sistema sea automatizado y el id del contenedor cambia a cada ejecución del docker-compose se usa ese comando para que coge el id desde el nombre de imagen)

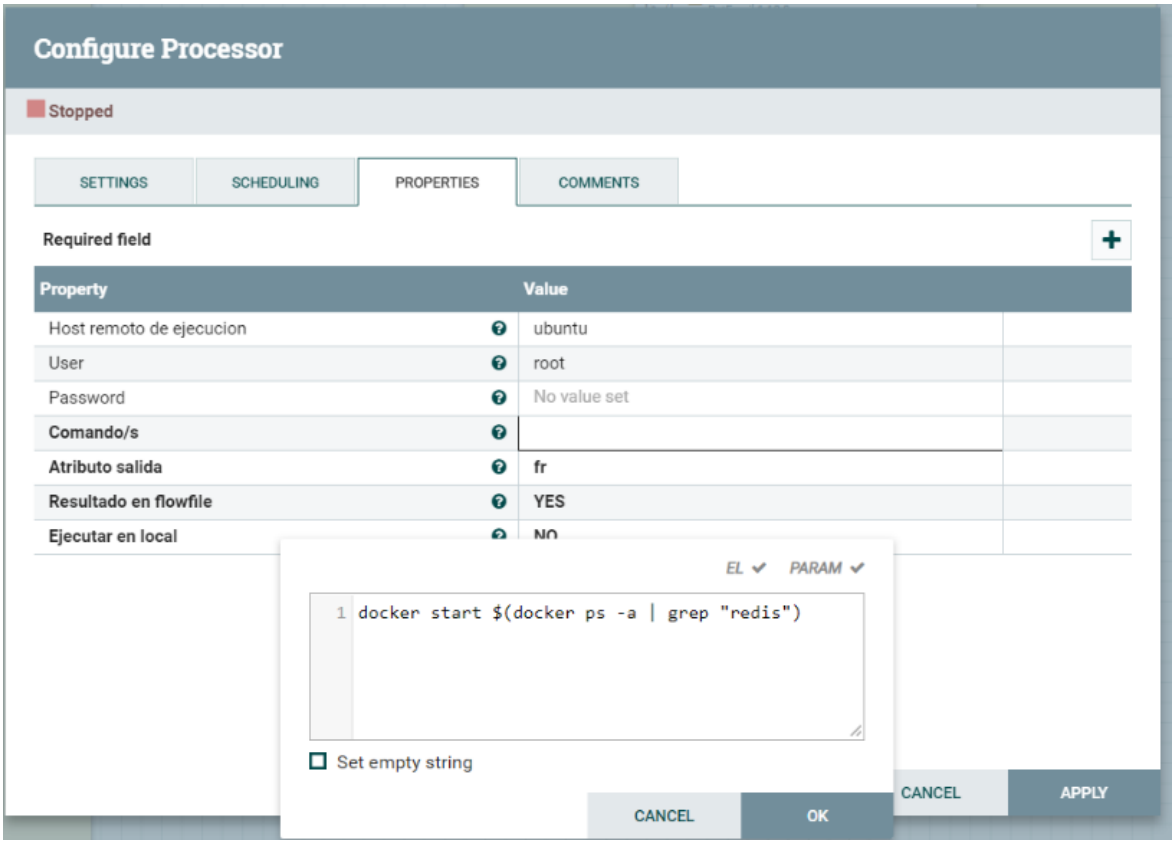

Ilustración 32 Comando para ejecutar el servidor de Redis

8.3 Monitorizar, detectar y solucionar errores de clúster Elasticsearch

En la Ilustración 33 se muestra una parte del flujo de Monitorizar Elasticsearch. Empieza desde la parte de coger los datos hasta definir errores. A continuación, se detallará la configuración de cada procesador y los resultados obtenidos.

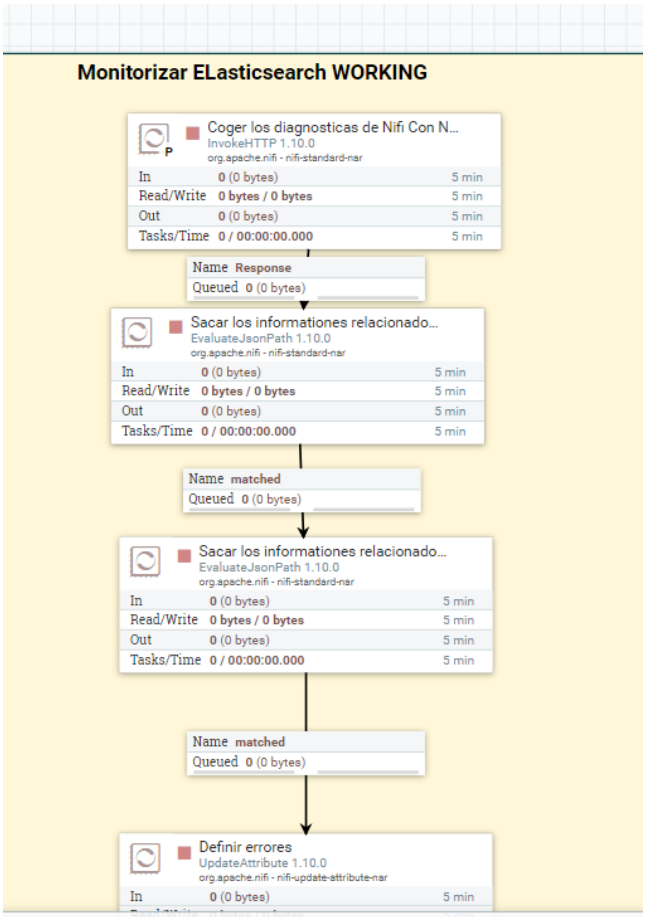

Ilustración 33 Una parte del flujo de Monitorizar Elasticsearch

- Primero se coge los estados de nodos con el Api nodes de Elasticsearch: https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-nodes-stats.html. En la Ilustración 34 se muestra la configuración de InvokeHttp.

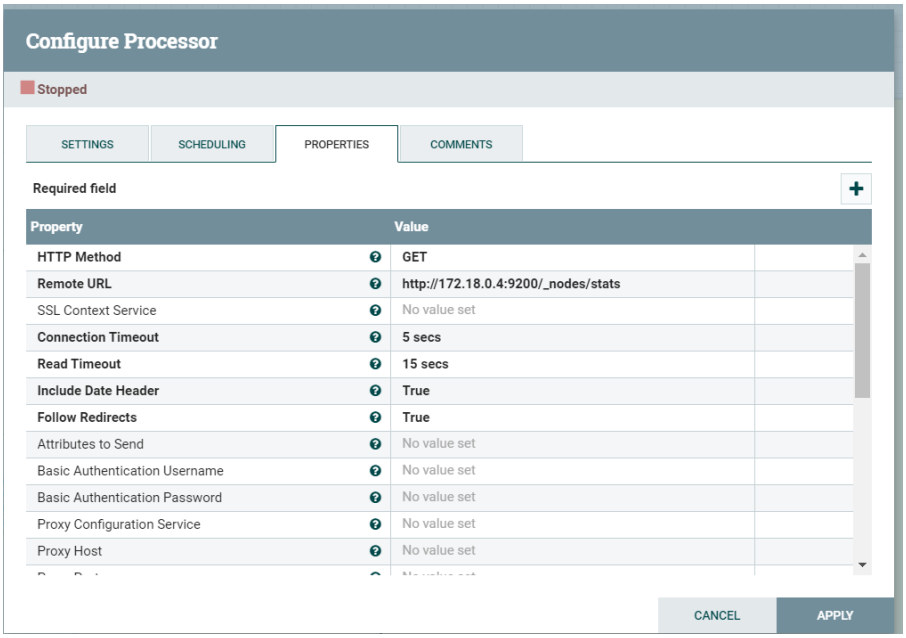

Ilustración 34 Configuración de InvokeHttp

- El resultado va a ser un JSON con los estatus de nodos conectados (Ilustración 35)

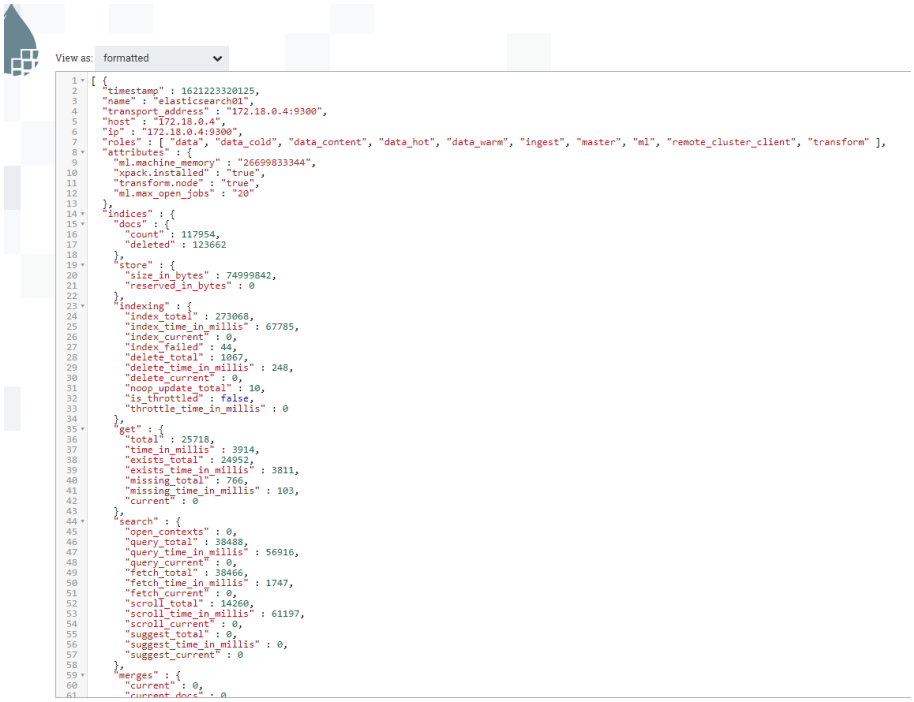

Ilustración 35 Resultado en Json

- Después se sacan las métricas que se quieren monitorizar y también los nodos conectados (Ilustración 36)

Elasticsearch	-Cluster health	¡=green	-Enviar Alarma al cliente
	-Nodos Conectados	-Uno o mas nodos desconectados	-Enviar Alarma al cliente -Reinicio del nodo desconectado
	-Uso excesivo de memoria JVM	>80%	-Enviar Alarma al cliente
	-Uso excesivo de memoria de sistema	>85%	-Enviar Alarma al cliente
	-Consumo de CPU	>90%	-Enviar Alarma al cliente

Tabla 5 Tabla de métricas, Alarmas y acciones

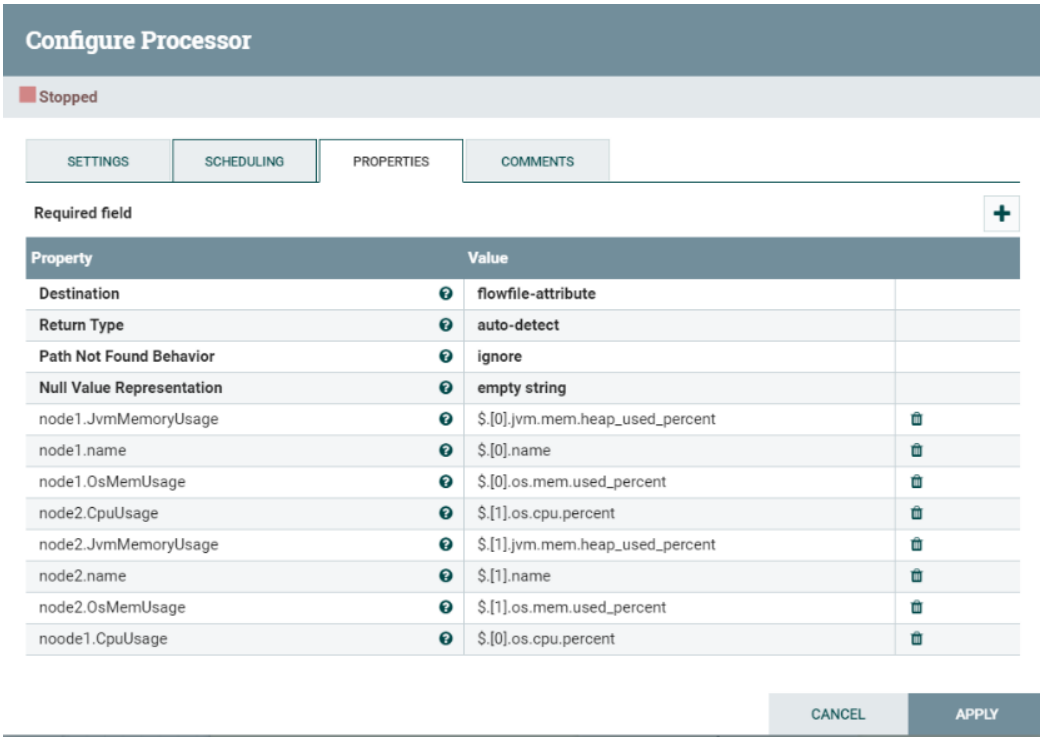

Ilustración 36 Sacar las métricas con JsonPath

- Se verifica si hay errores en las métricas (se ha usado valores deferentes de Tabla para las condiciones para la prueba como por ejemplo JVM_Consumo>20 o CPU_Consumo>40). Las condiciones usadas y los valores se muestran en la Ilustración 37.

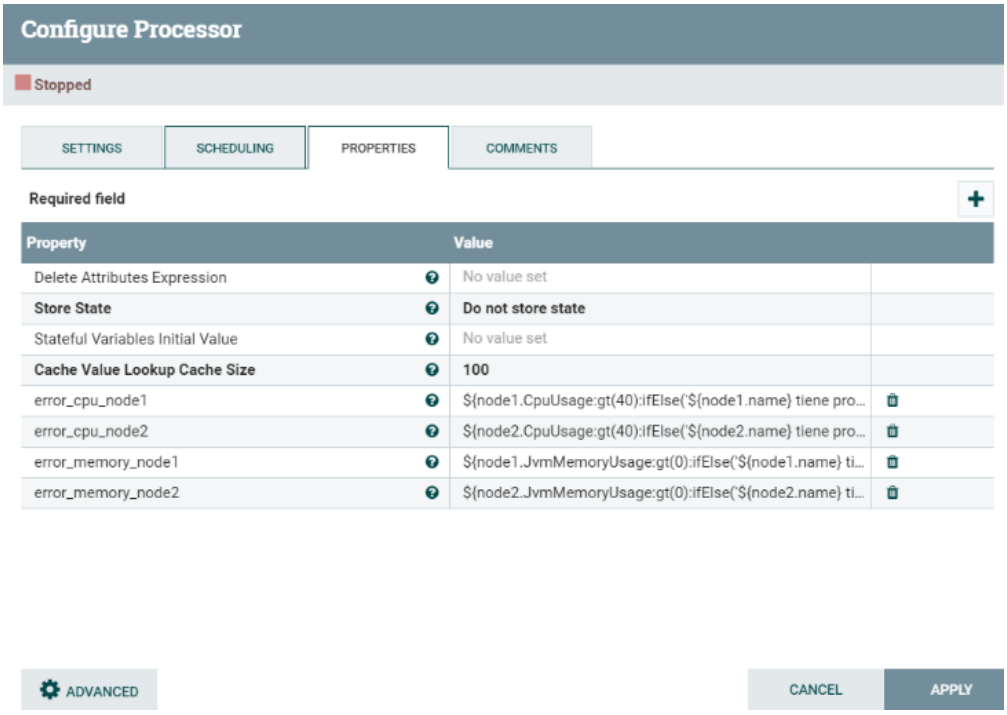

Ilustración 37 Definir Errores

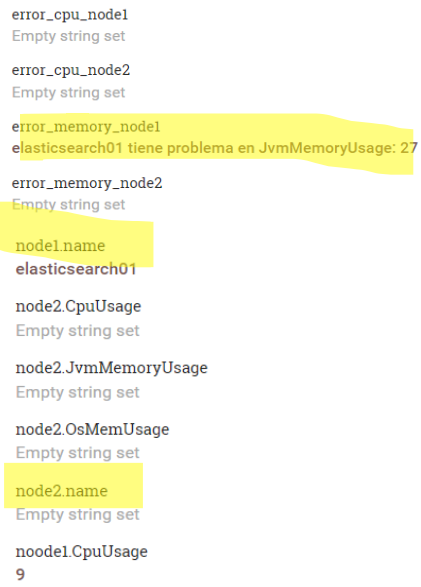

Ilustración 38 Sacar errores y nombres de nodos conectados

- Como se puede ver en la Ilustración 38 hay solamente un nodo conectado porque node2.name y todas las métricas relacionadas con nodo2 están vacías. Así, el siguiente paso es crear el texto del error como una concatenación entre los diferentes de errores de las métricas e iniciar el nodo desconectado.

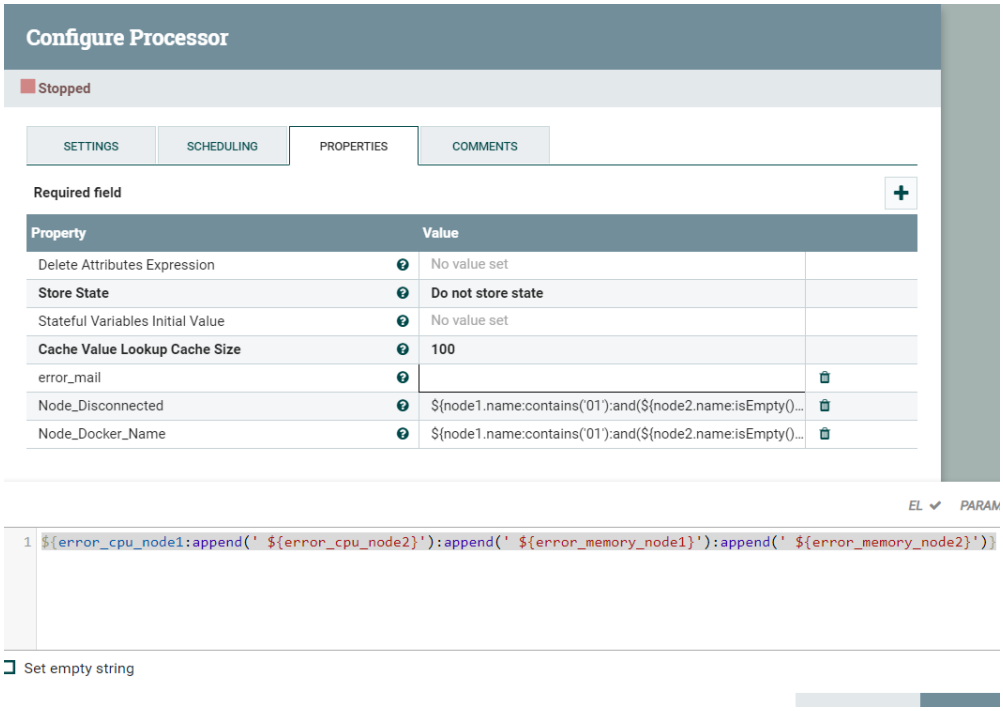

Ilustración 39 Crear texto de error, sacar el nombre de nodo desconectado

- Si se verifica el clúster de Elasticsearch desde kibana se va a ver que hay solamente un nodo desconectado (Ilustración 40)

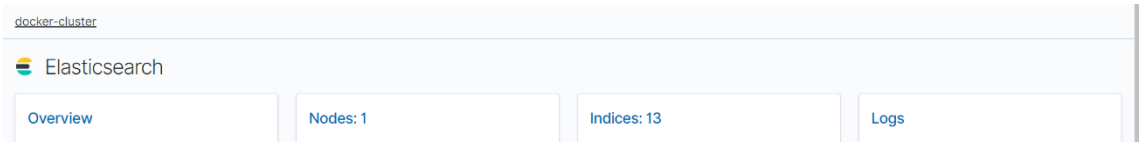

Ilustración 40 Clúster Elasticsearch

- El siguiente paso es ejecutar el nodo desconectado mediante el procesador executeStreamCommand que lo se ha desarrollado en java para ejecutar comandos en remote desde Apache NiFi y enviar alarma al cliente. En la Ilustración 41 se muestra su configuración. Hay que definir:
  - Hostname de la máquina
  - Usuario
  - Contraseña
  - Comando a ejecutar
  - Si se quiere que el resultado se almacena en el contenido de flowfile o en un atributo.

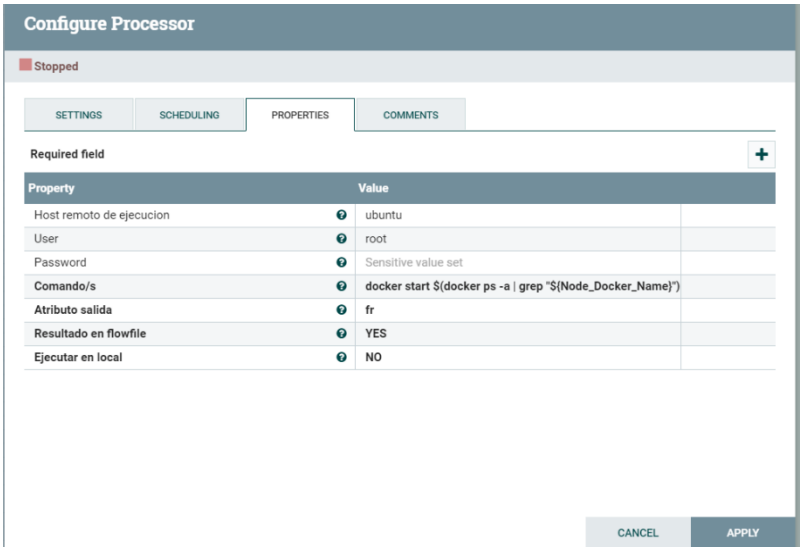

Ilustración 41 Configuración del procesador executeStreamCommand

- Ahora si se mira otra vez Kibana se puede comprobar que solo hay dos nodos en el clúster (Ilustración 42)

Ilustración 42 Clúster Elasticsearch después de conexión de nodo fallado

- Se crea un dashboard como el dashboard creado para Redis para visualizar las métricas sacadas de Elasticsearch

8.4 Monitorizar procesos de Nifi

En la Ilustración 43, Ilustración 44 e Ilustración 45 se muestran partes del flujo para monitorizar los procesos del clúster de Nifi

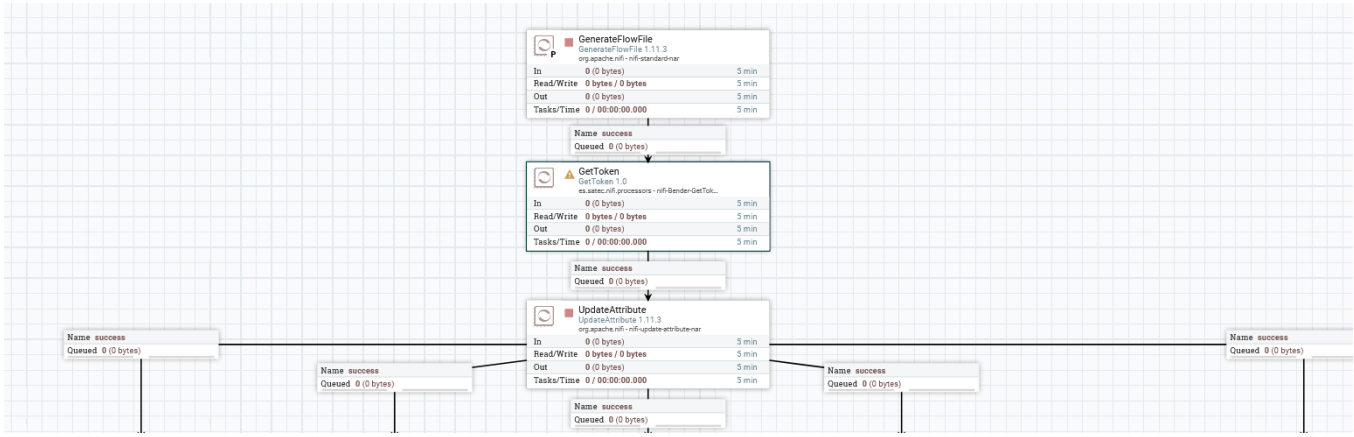

Ilustración 43 Primera parte del flujo para monitorizar los procesos del clúster

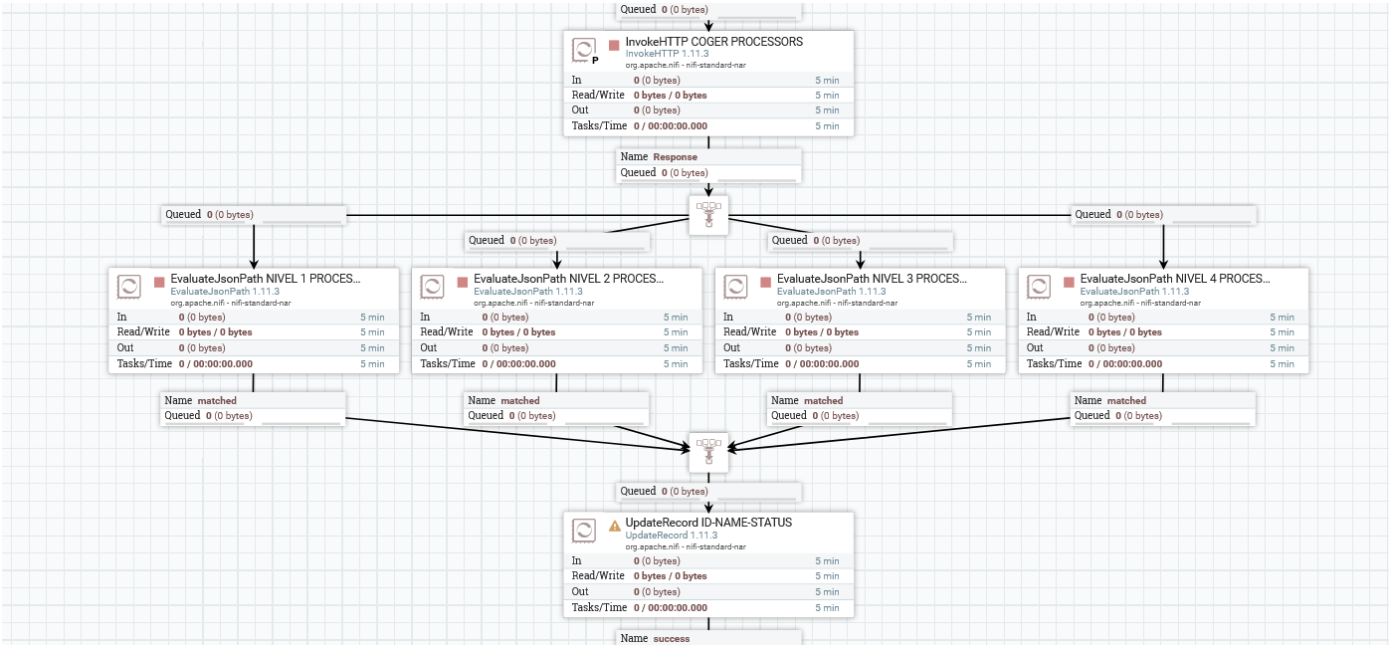

Ilustración 44 Segunda parte del flujo para monitorizar los procesos del clúster

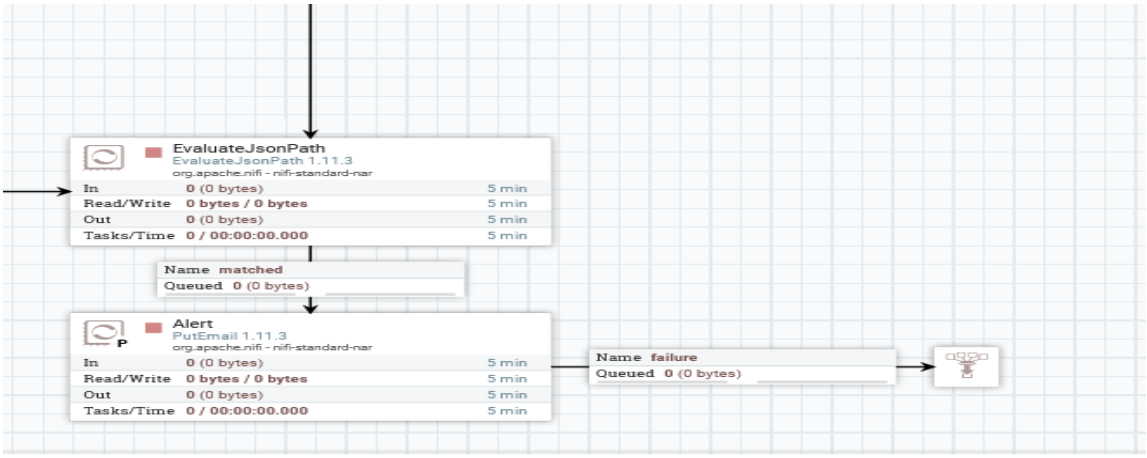

Ilustración 45 Tercera parte del flujo para monitorizar los procesos del clúster

- Primero se genera un token para acceder a los procesadores parados
- Después se coge cada grupo de procesadores según su nombre
- Después se coge la lista de procesadores de cada grupo de procesadores. En la Ilustración 46 se muestra la configuración del procesador "EvaluateJsonPath"

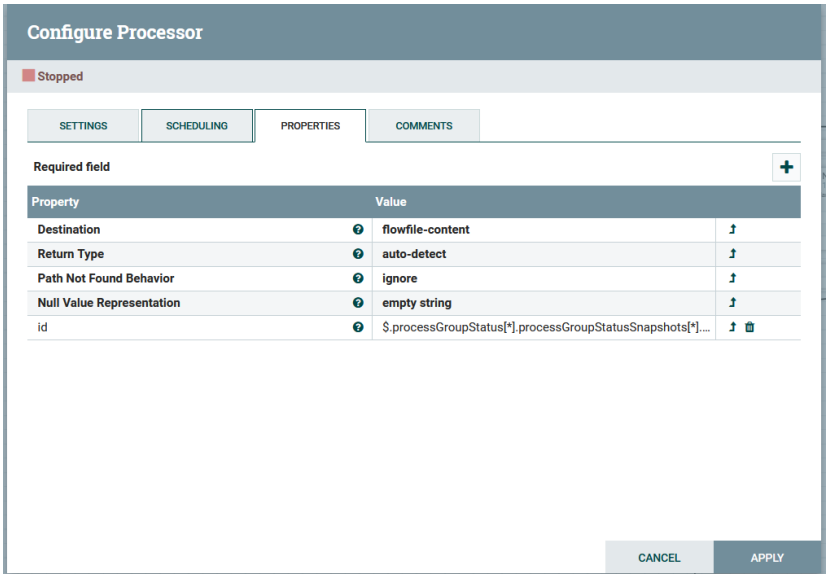

Ilustración 46 Configuración del procesador EvaluateJsonPath

- De los resultados obtenidos, quedamos con el id, nombre y estado de cada procesador (Ilustración 47)

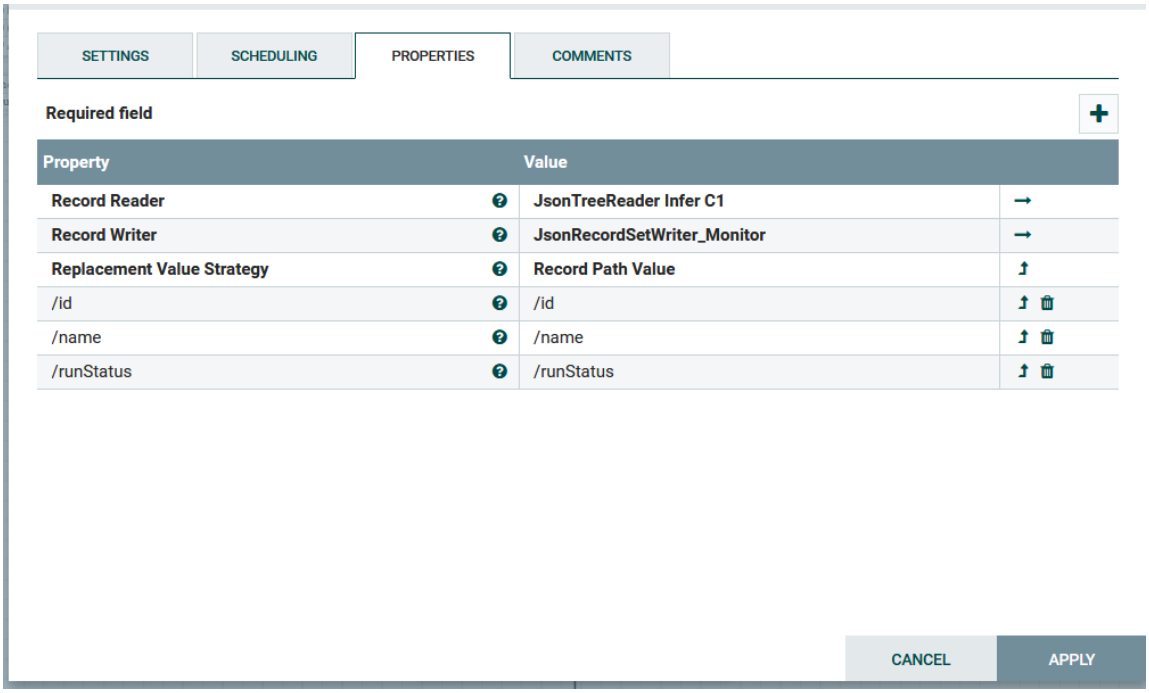

Ilustración 47 Configuración del UpdateRecord

- Si está parado se envía un correo al cliente. Tiene los siguientes detalles (Ilustración 48):
    - Entorno: Nombre del clúster
    - Estado: arrancado o parado
    - Nombre e id del procesador

ENTORNO : Cluster 1

STATUS : Stopped

NAMES : ["PutElasticsearchHttpRecord ; 019d1bbe-0349-344c-a29d-79ce84916448","PutElasticsearchHttpRecord ; c2e4e147-9c75-3e6c-b698-64779559460f","SplitRecord ; 4e3c2857-ca71-3221-a8b4-b331212b42c0","RetryFlowFile ; 1eb53768-6751-32e9-9f7f-f2b458f2ff0b","RetryFlowFile ; 87db7ac7-2703-3025-8863-c3589f657a70"]

Responder | Responder a todos | Reenviar

Ilustración 48 Correo enviado al cliente

- Se coge el id del procesador parado y se arranca con el Nifi Api Rest. Se utiliza ese comando “PUT/processors/{id}/run-status” https://nifi.Apache.org/docs/nifi-docs/rest-api/index.html
- Una vez esta arrancado el procesador se envía un correo al cliente como se muestra en la Ilustración 49.

ENTORNO : Cluster 1

STATUS : Running

NAMES : ["CompressContent - Unzip ; 4632334d-b5f2-3983-b418-b188d6355cdc","LookupRecord Martes con Nodo ; 27b1ef0a-4b54-387a-b9c3-9ba3b5d9b672","JoltTransformRecord - Transformar Nombres ; 3cc2b3c1-1a8b-30aa-9e37-86b3cc9c5cd9","LookupRecord Martes Nodo Sin Letra ; b9a90415-27c7-3755-9892-821fdafeec98","LookupRecord Batch KPI.AVG ; 0533647c-31dd-3d23-a531-d221f9e75e3f","UpdateRecord - Limpiar Operador ; 6deb37cc-b3cf-3697-aa4b-0c603b595738"]

Responder | Responder a todos | Reenviar

Ilustración 49 Correo enviado al cliente tras arrancar un procesador

Capítulo 9. Discusión

En base a los resultados obtenidos, se ha conseguido desarrollar una plataforma que permite aprovechar los diferentes aspectos de las herramientas de automatización de procesos y las herramientas de visualización de datos para crear procesos automatizados que permitan monitorizar los diferentes sistemas y resolver de forma automática los problemas detectados.

Con respecto a las herramientas de automatización de procesos se ha conseguido las siguientes ventajas:

- Plataforma escalable horizontal y verticalmente.
- Permite su implantación en alta disponibilidad (clúster) o standalone.
- Procesos automatizados para monitorizar el sistema.
- Los usuarios pueden ejecutar comandos y controlar los procesos visualmente.
- Soporte para la transmisión, gestión y tratamiento de datos.
- Permite mejorar, ordenar, modificar, combinar, dividir y verificar los datos.
- Facilita la creación de procesos de supervisión de forma visual, empleando componentes parametrizables y ejecutarlos sin necesidad de codificación adicional.
- Permite el desarrollo de nuevos componentes mediante implementación en Java o invocación externa de programas (python, groovy, shell script, etc).

Con respecto a las herramientas de visualización de datos se han obtenido las siguientes ventajas:

- Plataforma flexible de análisis y visualización
- Los resultados están almacenados en índices e implementado en visualizaciones, lo que permite generar dashboards y analizar series temporales
- Facilita el análisis de los datos para evitar errores en el futuro

Además, la herramienta presenta otras ventajas como:

- Envío automático de correos cuando se detecta errores
- Resolución de errores (reinicio de máquinas caídas, arrancar procesadores parados de Nifi)

Sin embargo, la herramienta desarrollada sigue teniendo algunas limitaciones y posibles puntos mejora:

- Ampliación de las funcionalidades para gestión de errores son pocos. Actualmente, se permite:
    - Reiniciar las máquinas
    - Arrancar procesadores parados

        - Los procesadores pueden parar a causa de un error
        - Si una persona para un procesador y se olvida puede causar muchos problemas. Como los procesos son automatizados, se puede coger muchos datos y quedan en cola sin procesar. Eso puede causar fallos en el nodo de Nifi, a causa de consumo excesivo de recursos o retrasos en las visualizaciones por falta de datos
- Ampliación de la seguridad del entorno
- Implementación de un sistema de versionado (Apache Registry)

Capítulo 10. Conclusiones y trabajo futuro

Con respecto a los objetivos planteados al inicio del proyecto, se puede decir que se han alcanzado y cumplido la mayoría con éxito. Se ha desarrollado una plataforma capaz de monitorizar diferentes sistemas, recopilar datos, solucionar problemas, crear dashboards y visualizaciones que sirven para evitar tales problemas y avisar al cliente mediante un correo con los errores que han ocurrido.

Este proyecto puede considerarse un piloto que se puede mejorar y ampliar con el objetivo de monitorizar otras herramientas, adicionales a las que se incluyen en el mismo.

Se ha empleado Docker para el despliegue de los diferentes componentes del sistema en un entorno de desarrollo local. Docker permite el despliegue de los mismos componentes en un entorno de producción real sin apenas modificaciones.

Este trabajo permite dar solución a las empresas con proyectos grandes ofreciendo una solución que puede monitorizar y solucionar problemas 7/24h. Por otra parte, la interfaz visual de Apache Nifi y sus workflows permiten modificar o actualizar los cambios de manera sencilla y fácil de entender. Con los análisis creados por los dashboards para mejorar los diferentes sistemas y las alarmas creadas, la plataforma puede ser una primera aproximación para tener un sistema completo en todos los aspectos.

Las ventajas y desventajas del proyecto se pueden resumir en:

- Ventajas:
    - Procesos automatizados para monitorizar el sistema 7/24h
    - Interfaz visual para la facilidad de actualizar y adaptar el piloto cuando se cambia o se actualiza un sistema
    - Datos recuperados e implantados entre diferentes visualizaciones para analizar y evitar problemas en el futuro
    - Reinicio automático de máquinas y nodos de clúster cuando se cae un servicio
    - Posibilidad de monitorizar los procesos de Nifi y arrancar los procesadores parados.
    - Uso de un solo sistema (NiFi) para tener acceso, recopilar datos, iniciar o ejecutar comandos entre diferentes tipos de sistema
    - Envío de correos automáticos al cliente a cada vez que se detecta un problema.
    - Posibilidad de añadir otros procesadores para Nifi cuando se necesita añadir una funcionalidad o ejecutar líneas de código directamente

- Desventajas:

- Limitaciones de acciones para solucionar los problemas
- Falta implementar la seguridad del entorno
- Falta tener un sistema de versionado (Se puede utilizar Apache Registry, pero como el trabajo es un piloto se guarda los avances del trabajo en Docker directamente)

A continuación, se plantean algunas líneas de trabajo futuro:

- Se puede mejorar los procesos y monitorizar más cosas en cada sistema según sus necesidades. Por ejemplo, se puede monitorizar procesos en el clúster de Apache Nifi y controlar el arranque/ parada de los procesadores.
- Hay que mejorar también la implementación con respecto a los errores encontrados. En este sentido, la plataforma solo puede como máximo reiniciar la máquina y/o el nodo desconectado de un clúster de NiFi y/o Elasticsearch, enviar correos para advertir al cliente de los problemas, y arrancar los procesadores parados. Así, por ejemplo, sería interesante poder apagar los programas cuando se la RAM y/o la CPU tienen un consumo masivo por parte del cliente.
- Otro elemento a mejorar sería la seguridad del entorno. En este piloto se usa NiFi, Elasticsearch, Kibana… pero se implementan en la web de una manera no segura. Sería conveniente usar un sistema de versionado como Apache NiFi Registry que es un subproyecto de Apache Nifi que permite el versionado de procesos.

Bibliografía

[1] «Gestión del Desempeño Corporativo,» [En línea]. Available: https://www.grupoecuacopia.com/soluciones-para-empresas/cpm-corporate-performance-management/.

[2] P. C. P. R. a. D. P. N. Halbwachs, «"The synchronous data flow programming language LUSTRE,",» Proceedings of the IEEE, vol. 79, no. 9, pp. 1305-1320, Septiembre 1991. [En línea]. Available: doi: 10.1109/5.97300..

[3] V. Ankam, Big Data Analytics, 2016.

[4] G. Hohpe, Enterprise Integration Patterns.

[5] NORA_MILLOR, «Tarea_ISE2_1_1_Formación_Español».

[6] R. P. Gómez, «Monografias.com,» [En línea]. Available: http://www.monografias.com/trabajos14/herramicase/herramicase.shtml#herr.

[7] A. Sahay, Data Visualization, Volume II : Uncovering the Hidden Pattern in Data Using Basic and New Quality Tools, 2017.

[8] J. D. Miller, Big Data Visualization, 28 févr. 2017.

[9] Gartner, Hype Cycle for Emerging Technologies, 2015.

[10] S. Matteson. [En línea]. Available: https://www.techrepublic.com/blog/big-data-analytics/big-data-basic-concepts-and-benefits-explained/.

[11] H. E. G. L. J. D. Jing Han, «Survey on NoSQL database,» 2011 6th International Conference on Pervasive Computing and Applications, 26-28 Octubre 2011. [En línea]. Available: doi: 10.1109/ICPCA.2011.6106531.

[12] «AWS Step Functions.,» 19 July 2017. [En línea]. Available: https://aws.amazon.com/step-functions/.

[13] Erica, «Introduction to the Fundamentals of Time Series Data and Analysis,» 13 Septiembre 2019. [En línea]. Available: aptech.com/blog/introduction-to-the-fundamentals-of-time-series-data-and-analysis/#:~:text=Time%20series%20data%20is%20data,potential%20for%20correlation%20between%20observations..

[14] W. Palma, Time Series Analysis, 2016-03-07.

[15] G. Vialcanet, « «Visualización de Datos: ¿Qué dicen los Expertos?,» 4 Junio 2015. [En línea]. Available: https://dbibyhavas.io/en/.

[16] M. Halsey, «Windows 10 Troubleshooting,» Apress, Berkeley, CA, 2016. [En línea]. Available: https://doi-org.bucm.idm.oclc.org/10.1007/978-1-4842-0925-7.

[17] A. Kirch, *Intro to Zenoss,* 2015.

[18] M. Badger, Zenoss Core Network and System Monitoring, June 2008.

[19] «Apache Airflow Documentation,» 2019. [En línea]. Available: https://airflow.apache.org/docs/apache-airflow/2.0.1/.

[20] « Apache Airflow,» 30 September 2019. [En línea]. Available: https://airflow.apache.org/docs/apache-airflow/stable/project.html.

[21] M. Beauchemin, «Airflow: a workflow management platform,» 2 Jun 2015. [En línea]. Available: https://medium.com/airbnb-engineering/airflow-a-workflow-management-platform-46318b977fd8.

[22] M. Trencseni, «Airflow review,» Wed January 2016. [En línea]. Available: https://bytepawn.com/airflow.html.

[23] «AirflowProposal,» 28 marzo 2019. [En línea]. Available: https://cwiki.apache.org/confluence/display/incubator/AirflowProposal.

[24] A. Serafini, Apache Solr Beginner's Guide, 26 deciembre 2013.

[25] [En línea]. Available: https://lucene.apache.org/#31+March+2011+-+Lucene+Core+3.1+and+Solr+3.1+Available.

[26] S. Mohan, Apache Solr High Performance, 2014.

[27] A. N. Overview, 15 11 2018. [En línea]. Available: https://docs.cloudera.com/HDPDocuments/HDF3/HDF-3.3.0/apache-nifi-overview/hdf-apache-nifi-overview.pdf.

[28] Wikipedia, «Flow Based Programming,» 28 12 2014. [En línea]. Available: http://en.wikipedia.org/wiki/Flow-based_programming#Concepts.

[29] «DB-Engines Ranking of Search Engines,» 10 June 2021. [En línea]. Available: https://db-engines.com/en/ranking/search+engine.

[30] p. a. p. o. t. Protocol — The people, «'It's not OK': Elastic takes aim at AWS, at the risk of major collateral damage,» 22 1 2021. [En línea].

[31] C. Hull, «No, Elastic X-Pack is not going to be open source – according to Elastic themselves,» 2 March 2018. [En línea]. Available: https://www.flax.co.uk/blog/2018/03/02/no-elastic-x-pack-not-going-open-source-according-elastic/.

[32] A. Snivastara, Learning Elasticsearch 7.x: Index, Analyze, Search and Aggregate Your Data using Elasticsearch, 3 deciembre 2020.

[33] A. Srivastava, Mastering Kibana 6. x : Visualize Your Elastic Stack Data with Histograms, Maps, Charts, and Graphs, 31 Julio 2018.

[34] B. A. Anurag Srivastava, Learning Kibana 7: Build powerful Elastic dashboards with Kibana's data visualization capabilities, 2nd Edition, 19 juil. 2019.

[35] A. Chinnachamy, Instant Redis Optimization How-to, 23 Mayo 2013.

[36] [En línea]. Available: https://redis.io/topics/faq.

[37] J. Muli, Beginning DevOps with Docker : Automate the Deployment of Your Environment with the Power of the Docker Toolchain, 29 Mayo 2018.

[38] « Docker Documentation: Kernel Requirements,» 4 enero 2014. [En línea]. Available: https://web.archive.org/web/20140821065734/http://docker.readthedocs.org/en/v0.7.3/installation/kernel/.

[39] J. Skinner, «Anatomy of a Next Generation Text Editor,» 30 noviembre 2007. [En línea]. Available: http://www.sublimetext.com/blog/articles/anatomy-of-a-next-generation-text-editor.

[40] «Yet Another Markup Language (YAML) 1.0,» 10 Deciembre 2001. [En línea]. Available: https://yaml.org/spec/history/2001-12-10.html.

[41] a. J. M. Q. Kim H. Pries, Scrum Project Management, 17 Augusto 2010.

[42] «Guia Scrum,» 2021. [En línea]. Available: https://www.scrumio.com/scrum/incremento-de-producto/.

ANEXO I

SCRUM [41] es el marco de trabajo que se ha utilizado para realizar este proyecto dado que permite un planteamiento flexible y se adecua mejor a las necesidades del proyecto. Además, permite disponer de un prototipo funcional en poco tiempo y ayuda a dimensionar el proyecto y a disminuir los riesgos.

1. Planteamiento con una metodología agile basada en SCRUM

De un tiempo a esta parte, las nuevas tendencias de gestión y ejecución de proyectos de desarrollo de software están marcando un cambio hacia planteamientos de gestión de proyectos basados en el manifiesto agile, publicado en el año 2001, y que tiene como principal objetivo establecer un marco de trabajo que sea capaz de responder de manera mucho más eficiente que las metodologías predictivas o en cascada al entorno tan cambiante sobre el que normalmente se trabaja en los proyectos de desarrollo de software.

2. Introducción a Scrum en equipos de trabajo

En este contexto en el que se pretende crear un producto mantenido en el tiempo, creemos que esta metodología orientada a maximizar el valor final de la plataforma es la elección adecuada. Se debe intensificar la colaboración con el cliente para aprovechar toda su experiencia con la antigua plataforma para poder crear una nueva que sea realmente útil. Por tanto, dado que existe un producto que delimita el alcance del proyecto hace que la prioridad metodológica sea crear algo de calidad y útil cogiendo lo bueno, desechando lo malo y aportando además toda la experiencia en este tipo de proyectos para conseguir el mejor de los resultados.

Scrum no es un proceso o una técnica en sí mismo, sino un marco de trabajo donde se pueden emplear diferentes conjuntos de procesos y técnicas enfocados a la mejora continua del producto, equipos y entorno laboral. Scrum lo componen los Scrum Teams, sus roles, eventos, artefactos y reglas asociadas. Cada componente sirve a un propósito específico.

Las metodologías ágiles aportan mucho valor en la composición, organización, valores y dinámicas de los equipos de trabajo. Los llamados Scrum Teams están formados por: Product Owner, Development Team y Scrum Master. Dichos equipos, como veremos un poco más adelante, son auto-organizados, multidisciplinares y entregan los productos de forma iterativa e incremental maximizando de esta forma las oportunidades para obtener retroalimentación. Dichos modelos de equipo están pensados para optimizar la flexibilidad, la creatividad y la productividad.

El equipo propuesto para la realización de este proyecto se compone de un Scrum Team compuesto por tres roles:

- Jefe de Proyecto (Product Owner) a tiempo completo

- Equipo de Trabajo (Development Team) a tiempo completo
- Scrum Master perfil a tiempo parcial.

3. Pilares y valores de los equipos

Scrum emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo apoyándose en la experiencia adquirida (empirismo). Para canalizar esa experiencia se utiliza la transparencia, inspección y adaptación siendo los pilares sobre los que se sustentan los equipos.

- Transparencia
  Todo el proceso debe ser visible para los responsables del resultado. Para ello se definirá un lenguaje común por el cual el entendimiento pueda ser compartido y existirá una definición de 'Done' (Terminado) para indicar de manera inequívoca la finalización de una tarea incremental.

- Inspección
  La inspección sobre los procesos y resultados permite detectar desajustes tanto en los elementos en teoría acabados como errores en la metodología de trabajo.

- Adaptación
  Una vez que se detectan aspectos desviados y que no son aceptables en el producto el proceso o el material deberá ser ajustado. Dicha adaptación debe ser realizada 'as soon as possible'.

La transparencia, inspección y adaptación es llevada a cabo por el equipo especialmente durante la realización de cuatro eventos formales: Sprint Planning, Daily Scrum, Sprint Review y Sprint Retrospective. Estos eventos serán descritos más adelante.

Para materializar estos tres pilares que permiten mejorar y adquirir conocimientos de la experiencia durante la prestación del servicio, el equipo debe trabajar siguiendo los valores Scrum que se muestra en la ilustración**¡Error! No se encuentra el origen de la r eferencia.**:

- Coraje, valentía y confianza para hacer bien las cosas y trabajar en problemas difíciles
- Mantener la concentración y el foco sobre los objetivos perseguidos
- Mente abierta y positiva ante los desafíos
- Respeto entre los miembros del equipo

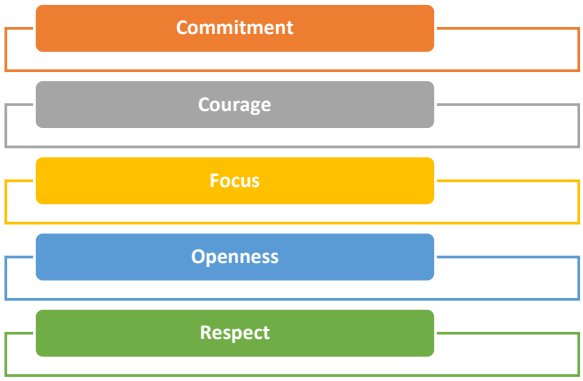

Ilustración 50 Los valores Scrum

4. Scrum Team

Tal como se ha visto en la introducción, el equipo Scrum (Scrum Team) está formado por un propietario del producto (Product Owner), el equipo de desarrollo (Development Team) y un Scrum Master. Lo que caracteriza a estos equipos de manera común en toda metodología ágil es que son auto-organizados y multidisciplinares.

5. Responsable del Producto (Product Owner)

Es el responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo. Además, en colaboración con el responsable del cliente, es el encargado de gestionar la llamada pila del producto (Product Backlog). Este artefacto es un panel de actividades o funcionalidades a realizar del producto. La gestión de esta pila incluye:

- Ordenar los elementos para alcanzar los objetivos de la mejor manera posible
- Deberá ser visible, transparente y clara para todos los y que muestre lo que el equipo trabajará a continuación

Por tanto, esta pila es un 'roadmap' de producto ordenado principalmente por las actividades más valiosas a las menos valiosas.

6. Equipo de Desarrollo (Development Team)

El equipo de desarrollo se compone de profesionales que realizan el trabajo de entregar incrementos de producto 'Done' que potencialmente puedan ser desplegados en producción. En general el despliegue en producción no es un requisito específico, pero la experiencia indica que lo mejor es que el paso a producción sea lo menos traumático posible, sobre todo una vez que el producto vea la luz. Por ello, el paso a producción deberá estar bien documentado, procedimentado y con la posibilidad de reversión.

Todo esto sigue la filosofía CI (Continuos Integration) & CD (Continuos Delivery) que será explicada más adelante como parte de la metodología de desarrollo.

Como adelantamos en la introducción, el equipo dispondrá de las siguientes características:

- Auto-organizados y auto-suficientes:
  Una vez asimilado el servicio nadie debería indicar al equipo cómo convertir elementos de la pila del producto (Product Backlog) en incrementos de funcionalidad entregable.

- Multidisciplinares:
  Contarán con todas las habilidades necesarias para crear un incremento.

- Jerarquía plana y sin sub-grupos:
  No se reconocen títulos para los miembros del equipo, todos son desarrolladores, independientemente del trabajo que realice cada persona. Además, sin importar los dominios particulares como pruebas o análisis no se realizarán sub-equipos específicos para ese fin.

- Responsabilidad compartida:
  Los miembros individuales pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en todo el equipo.

7. Scrum Master

La figura del Scrum Master será el que ayude al equipo a aplicar la metodología ágil. El Scrum Master está al servicio del y para el equipo.

- Al servicio del equipo de desarrollo

    - Guiarlos para conseguir ser auto-organizados, multidisciplinares y autosuficientes. Trabajar con ellos en la transferencia de conocimiento a través de todos los miembros del equipo y concienciarles en que todos son responsables del trabajo fomentando el compañerismo y el trabajo en equipo.
    - Motivar al equipo a ser creativos y que ayuden en la creación de productos de alto valor.
    - Facilitar que los eventos de Scrum, todas las reuniones internas del equipo, ser realicen y guiarles en cómo realizarlas.
    - Recoger sus opiniones sobre el funcionamiento del servicio y ayudar a preparar mejoras en sus dinámicas de trabajo.

- Al servicio del responsable de la plataforma:

    - Ayudar a encontrar técnicas para gestionar la pila de trabajo de manera efectiva.
    - Entender y practicar la agilidad en la forma acordada.
    - Facilitar los eventos de Scrum según se necesite.

    - Ayudar a encontrar técnicas y mejoras para los paneles de seguimiento del servicio
- Al servicio del cliente

    - Trabajar con la organización para la mejora continua en la metodología ágil.
    - Planificar las implementaciones necesarias para converger en la medida de lo posible con Scrum, asimilando lo que empíricamente resulte efectivo y desechando lo que no.
    - Motivar cambios que incrementen la productividad del equipo
    - Trabajar para incrementar la efectividad de la aplicación de la metodología ágil.

8. Eventos

Existen diferentes eventos con el fin de crear automatismos, rutinas y minimizar la necesidad de reuniones no definidas. Todos los eventos son de duración limitada (time-boxes) de manera que todos tienen una duración máxima. Todos estos eventos están orientados a habilitar los pilares vitales de transparencia e inspección.

El evento en el que intervendrá el cliente, con su responsable asignado y los 'stakeholders' que considere oportuno, es la revisión del Sprint (Sprint Review)

9. El Sprint

El Sprint es el corazón del Scrum siendo el único evento time-box (menos de un mes) de duración fija y que no puede acortarse o alargase. Se recomienda que la duración de los mismos sea consistente a lo largo de todo el esfuerzo de desarrollo. Es tremendamente positivo este aspecto para crear rutinas y disminuir la complejidad. La complejidad metodológica consume recursos. Para el desarrollo de esta plataforma se ha definido una duración de Sprints de dos a cuatro semanas. Todo el equipo conoce la duración de los Sprints y cuando se celebran los eventos correspondientes.

Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint anterior. Los Sprints contienen a los demás eventos: *Sprint Planning, Daily Srums, Sprint Review y Sprint Retrospective.*

En la siguiente ilustración**¡Error! No se encuentra el origen de la referencia.** se muestra l a relación entre esos eventos.

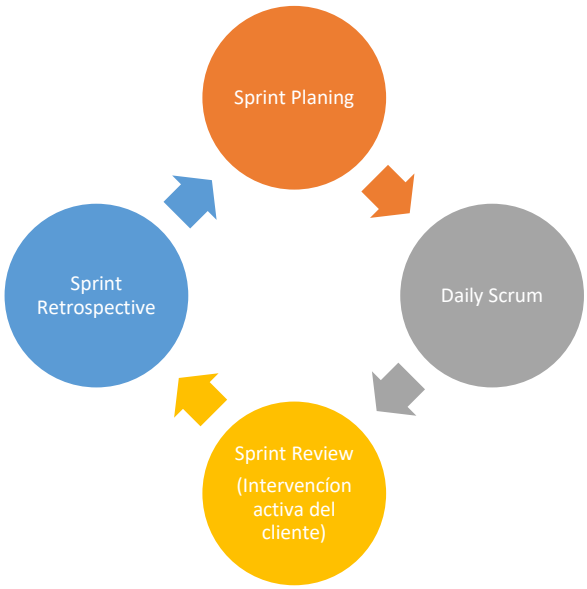

Ilustración 51 Eventos de los Sprints

El objetivo del Sprint (Sprint Goal), como veremos más adelante se fija al inicio de este durante el Sprint Planning y nunca se modifica. La esencia, el propósito general por el que se creó nunca cambia.

Los objetivos de calidad nunca disminuyen y el alcance puede clarificarse y renegociarse entre el responsable de producto (Product Owner) y el equipo de desarrollo. Como vemos la duración del Sprint nunca se modifica se completen o no todas las tareas planificadas. Lo que no este 'Done' (Terminado) volverá a la pila del producto para realizarse si así se estima conveniente en el siguiente Sprint y lo que este 'Done' es aceptado por el Product Owner.

Así pues, cada Sprint puede considerarse como un proyecto con un horizonte de dos semanas. El Sprint requiere de la definición de lo que se construirá.

10. Planificación del Sprint (Sprint Planning)

El trabajo a realizar durante el Sprint se planifica en esta reunión. Todo nuevo Sprint empieza con este evento. Tiene una duración máxima de ocho horas para un Sprint de un mes. Para Sprints más cortos el evento es normalmente más corto. A modo de ayuda se recomienda centrarse durante la reunión en contestar qué incremento desarrollar y cómo hacerlo. La carga de trabajo del Sprint es pactada por el responsable del producto, el equipo de desarrollo y el cliente.

En general se guarda el 10% del tiempo del Sprint para refinar la cabeza de la pila del producto y desgranar funcionalidad en tareas o 'ítems ready', así como, valorar y categorizar nuevas funcionalidades.

A la finalización de este evento se debe tener claro el objetivo del Sprint, se establece una meta para que todos puedan focalizarse en cumplirla. La planificación no tiene que ser exhaustiva. Lo principal es que se haya preparado trabajo para los siguientes días. Durante el Sprint se tiene una lista de tareas a realizar expuestas en lo que se denomina un 'Sprint Backlog'.

11. Scrum Diario (Daily Scrum)

Esta reunión probablemente sea la más famosa de los eventos de Scrum. Su duración máxima es de 15 minutos y el objetivo principal es que los miembros del equipo de desarrollo sincronicen sus tareas y creen un plan para las siguientes 8 horas laborables.

Es buena praxis realizar la reunión a la misma hora y en el mismo lugar todos los días para, como dijimos anteriormente, reducir la complejidad: todo el mundo tiene claro siempre dónde y cuándo se produce.

Los Scrum diarios mejoran la comunicación, identifican impedimentos relativos al desarrollo, promueven la toma de decisiones, mejoran el nivel de conocimiento, dan transparencia y ayudan a que todo el equipo se sienta responsable de todo el trabajo.

Existen muchas técnicas que ayudan a conseguir los objetivos del Scrum diario: realizarlos de pie, limitar la intervención de cada miembro a dos minutos, uso de una pelota de goma que otorga la palabra y pasarla entre los miembros del equipo, respuestas a las tres preguntas básicas etc... Sea como fuere es positivo que el equipo sea el que la vaya configurando con la ayuda del Scrum Master.

12. Revisión del Sprint (Sprint Review)

Es la reunión que se realiza al acabar el Sprint y sirve para revisar el Incremento producido. Esta reunión está restringida a un tiempo máximo de cuatro horas. El resultado de la revisión del Sprint es una pila de producto revisada que define la hoja de ruta para el siguiente Sprint.

Pautas interesantes a seguir en esta reunión son:

- El responsable del proyecto explica qué elementos se han terminado ('Done') y cuales no

- El equipo de desarrollo habla acerca de qué estuvo bien durante el Sprint, qué problemas aparecieron y cómo fueron resueltos esos problemas
- Se puede mostrar el incremento a través de una 'demo', para recibir el feedback y mejorar la experiencia en futuros Sprints.

13. Retrospectiva del Sprint (Sprint Retrospective)

Es la oportunidad que tiene el equipo Scrum para inspeccionarse a sí mismo y de crear un plan de mejoras que sean abordadas durante el siguiente Sprint. La duración máxima de esta reunión son tres horas para Sprints de un mes.

Es en esta reunión donde se intenta mejorar los procesos y dinámicas de trabajo. Uno de los puntos fuertes es la inspección de la definición de 'Done' y su mejora continua.

Un elemento de mejora suele denominarse 'kaizen'. Es una práctica recomendada implementar un 'kaizen' por Sprint. La implementación de 'kaizens', a veces, requieren adaptaciones o implementaciones técnicas específicas, el uso de nuevas herramientas etc. Esto deberá ser tenido en cuenta durante el Sprint Planning del día siguiente para reservar un tiempo estimado a la consecución de dicho 'kaizen'.

En general, estas mejoras, aunque sean beneficiosas para el servicio y para la organización en sí, no forman parte de las metas de los Sprints. Las metas son incrementos de productos, evolutivos o resolución de incidencias graves nunca mejoras metodológicas.

14. Artefactos

Los artefactos de Scrum representan el trabajo o el valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. El objetivo principal para los que se crean es la transparencia de la información clave, necesaria para asegurar que todos tengan el mismo entendimiento del artefacto.

15. Pila del producto (Product Backlog)

La pila del producto, ampliamente mencionada a lo largo de este documento, no es más que una lista ordenada de todo lo que hay que hacer relativo a la funcionalidad del producto a entregar. El orden de las tareas implica las actividades a realizar en el siguiente Sprint.

16. Pila del Sprint (Sprint Backlog)

La pila del Sprint es un conjunto de elementos de la pila del producto seleccionados para el Sprint, más un plan para entregar el 'Incremento' y conseguir el objetivo del Sprint. Esta lista de tareas hace visible todo el trabajo del equipo de desarrollo.

Las tareas se introducen en la pila del Sprint en función de prioridades y de la capacidad de trabajo de todos los miembros del Sprint. No olvidemos dos aspectos muy importantes:

- Todo incremento realizado debe cumplir con la definición de 'Done' o lo que es lo mismo.
- Los trabajos integrados y entregados deben cumplir con los estándares de calidad pactados al inicio del Sprint.

Recordemos que en cualquier momento durante un Sprint es posible sumar trabajo restante total en los elementos de la pila del Sprint. El equipo de desarrollo hace seguimiento de este trabajo restante total al menos en cada Daily Scrum, para proyectar la posibilidad de conseguir el objetivo del Sprint. Haciendo el seguimiento del trabajo restante a lo largo del Sprint el equipo de desarrollo puede gestionar su progreso.

17. Incremento (Increment)

"El incremento es la suma de todos los elementos de la pila del producto completados durante un Sprint y el valor de todos los incrementos de todos los Sprints anteriores"[42]

Es muy interesante resaltar la segunda parte en la que informa acerca de los incrementos nuevos deben asegurar el correcto funcionamiento de los incrementos pasados. Esto es lo que lleva al uso casi necesario de la 'Integración Continua'.

18. Done

En general para un incremento de software, la definición de 'Done' incluye la implementación de la funcionalidad garantizando que ha pasado un plan de pruebas y se ha realizado la documentación requerida. No olvidemos que algo 'Done' también tiene que garantizar que todo lo que era 'Done' sigua siéndolo. Si los procesos de integración continua son los adecuados, la integración automática garantiza que los trabajos son 'Done'.