UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Sistema de alertas basado en procesamiento en tiempo real de logs en una plataforma con disponibilidad 24/7

Autor/a:
Adrián Bernárdez Cornes

Titores:
Manuel Lama Penín

Máster universitario en Tecnologías de Análisis de Datos Masivos: Big Data

Marzo 2021

Traballo de Fin de Máster presentado na Escola Técnica Superior de Enxeñaría da Universidade de Santiago de Compostela para a obtención do Máster Interuniversitario en Big Data: Tecnologías de Análisis de Datos Masivos

Índice general

Capítulo 1

Introdución

Para un sistema de máxima disponibilidad como es el caso de sistemas de dispensación farmacéutica, los cuales deben estar en funcionamiento las 24 horas del día, los 7 días de la semana se generan una cantidad masiva de logs.

Para cumplir con esta disponibilidad se deberá disponer de un equipo de desarrollo y de soporte que cubrirán todas las horas del día haciendo guardias durante las horas no laborables. Sin embargo, en caso de que un componente falle, el tiempo de respuesta en informar del problema y buscar una solución deberá ser el mínimo posible independientemente de la causa y circunstancia en que se de el problema.

El objetivo del proyecto será aportar una plataforma que se encargue de generar alertas en caso de una incidencia importante, analizando los logs de actividad hasta dar con el componente afectado. De esta forma el sistema deberá responder en tiempo real con el fin de agilizar el proceso y mejorar el servicio. Con esta nueva operativa se podría registrar estas alertas para poder dotar al equipo de un informe de las incidencias y sus componentes afectados, con el fin de elaborar soluciones a problemas comunes y poder formar de una manera más eficiente y exacta a los encargados del correcto funcionamiento del sistema principal.

Se buscará que la alerta generada sea lo más descriptible posible y que consiga dar un indicio de la causa del problema. De esta manera se podrá dar una respuesta con tiempos mínimos tiempos de espera, lo cual supondría tanto una

mejora en el servicio como la facilitación del trabajo de cara al equipo encargado de la supervisión de la aplicación.

A la hora realizar el diseño de la plataforma se deberá tener los siguientes puntos como claves:

- Dada una aplicación que tenga una disponibilidad total y sobre la que opere un número elevado de usuarios, se generan una cantidad de logs masiva, que en caso de que se produzca un error, el proceso de revisión de logs y la diagnosis posterior puede ocupar un tiempo demasiado largo en caso de que no se pueden producir cortes en el servicio.

- La finalidad del proyecto será disponer de una plataforma para detectar en tiempo real posibles problemas e incidencias y clasificarlas según experiencias pasadas reducirá ese tiempo de espera en gran medida. La funcionalidad de la clasificación de la alerta entre problemas conocidos supondría una mejora significativa en la gestión operacional de la aplicación.

- Registrar las incidencias alertadas de errores no conocidos supone un extra añadido, ya que se dispondrá de un histórico con el que saldrán a la luz posibles bugs o errores frecuentes de cara a poder llevar un control y análisis de la aplicación más exhaustiva.

- El proyecto deberá estar desacoplado de la aplicación principal y estar preparado para adaptarse a cambios y nuevas funcionalidades.

- En caso de detectar un detectar errores desconocidos, de los cuales la aplicación no conoce la causa, se deberá implementar un sistema de obtención de anomalías en los logs para mostrar las excepciones causantes del error.

1.1. Objetivos

Una vez planteado el contexto se identifican los siguientes objetivos:

OBJ-01	**Emisión de alertas con información relevante tras la detección de posibles errores o incidencias. Se buscará enviar la posible causa del problema raíz.**
Descripción	El principal objetivo del proyecto es la construcción de una plataforma con las herramientas necesarias para la emisión de alertas tempranas en caso de producirse una incidencia en la aplicación a monitorizar. Se espera que la alertas se generen en near real time con el fin de una recuperación del servicio los más rápida y eficiente posible. La plataforma deberá ser configurable ya que nace de la necesidad de coexistir servicio a un sistema con disponibilidad total que está en continuo desarrollo y evolución.

OBJ-02	**Registro de las incidencias encontradas con la información necesaria para su posterior tratamiento.**
Descripción	Se deberá llevar un registro de las incidencias detectadas de errores no identificados y de los logs de la aplicación obtenidos. De esta forma una vez solucionada la incidencia se podrá catalogar y registrar los errores producidos en la aplicación con el fin de un análisis posterior en el que se podrán detectar posibles vulnerabilidades o bugs.

OBJ-03	**Clasificar el error mediante los logs de la aplicación**
Descripción	La plataforma deberá clasificar de la incidencia en base a los logs producidos en la aplicación. De esta forma se podrá plantear el envío de los pasos conocidos para su solución en cuanto se detecte el error. En caso de que se trate de un error no conocido se aplicarán técnicas de Machine Learning para la obtención de anomalías en los registros.

1.2. Alcance

La solución alcanzada deberá cumplir los objetivos definidos en el apartado anterior. Por un lado la plataforma a desarrollar deberá incorporar las herramientas necesarias que permitan de forma simple para un usuario la creación de alertas y modificación de las mismas en base a criterios acordados y a la experiencia obtenida. Por otro deberá utilizar los datos obtenidos con la finalidad de aplicar herramientas de Machine Learning con el fin de obtener la información más útil para la solución de la incidencia.

Se deberá tener en cuenta que no se podrán utilizar datos reales de la aplicación actual debido a que se estarían utilizando datos personales de terceros lo que incumpliría la normativa de protección de datos. En su lugar se utilizarán logs de entornos de desarrollo sobre los cuales se implementará una solución que permita crear un escenario equivalente al entorno real sin datos personales y a menor escala.

Una vez finalizado el desarrollo se deberá proporcionar una plataforma que pueda ser agregada a la aplicación actual y que deberá ser validada a posteriori. Para ello deberá proponerse una arquitectura como una extensión de la plataforma actual con el fin de que coexista. Por ello se ha de tener en cuenta que la instalación y despliegue será controlado en un entorno preparado para este fin permitiendo un control de versiones y registro del resultado.

Como entregables del proyecto serán los siguiente:

1. Plataforma encargada de la emisión de alertas con reglas de alertas configurables.
2. Plataforma encargada de la clasificación de las incidencias y su registro.
3. Propuesta de integración en el sistema real

Capítulo 2

Revisión de solucións comerciais

2.1. Gestión de alertas

La monitorización de la actividad en la red puede ser un trabajo tedioso, sin embargo existen buenas razones pare hacerlo. Por un lado permite buscar e investigar inicios de sesión sospechosos en estaciones de trabajo, dispositivos conectados a redes y servidores mientras identifica las posibles deficiencia de seguridad en la adminitración del sistema. También puede reastrar las instalaciones software y las transferencias de datos para identificar problemas potenciales en tiempo real en lugar que después de que el daño esté hecho.

Estos logs trambién contribuyen en gran medida a que la organización cumpla con el reglamento general de protección de datos (GHDPR) que se aplica a cualquier entidad que opere en la unión europea.

El registro, tanto el seguimiento como el análisis debe ser un proceso fundamental en cualquier infraestructura de monitoreo. Es necesario un archivo de registro de transacciones para recuperar una base de datos de un desastre. Además, al rastrear los archivos de registro, los equipo de DevOps y los administradores base de datos (DBA) pueden mantener un rendimiento óptimo de la base de datos o encontrar evidencia de actividad no autorizada en el caso de un ciberataque. Por esta razón, es importante monitorear y analizar regularmente los regitro del sistema. Es una forma confiable de recrear la cadena de eventos que condujo a

cualquier problema que haya surgido.

Debido a esta serie de problemáticas, existen un número considerable de registro de código abierto y herramientas de análisis disponibles en la actualidad, lo que hace que elegir los recursos adecuados para los registros de actividad sea un trabajo considerable. La comunidad de software de código abierto y gratuito ofrece diseños de registros que funcionan con todo tipo de sitios y prácticamente con cualquier sistema operativo.

Para la gestión de los logs y producción de alertas se tuvieron en cuenta las siguientes herramientas:

2.1.1. Graylog

Graylog comenzó en Alemania en 2011 y ahora se ofrece como una herramienta de código abierto o una solución comercial. Está diseñado para ser un sistema de administración de registros centralizado que recibe flujos de datos de varios servidores o puntos finales y le permite navegar o analizar esa información rápidamente.

Graylog dispone de las siguientes funcionalidades:

- **Recolección y procesado de logs**: implemente la necesidad de leer distintos tipos de logs de distintas fuentes para ser tratados de forma descentralizada, así como la posibilidad de integración con directorios LDAP.
- **Transformación de datos**: ofrece la posibilidad de transformar los datos de entrada para publicador a otras aplicaciones como fuente de datos refinada.
- **Análisis e investigación**: proporciona la capacidad para buscar y analizar información de distintas fuentes de datos en una única consulta.
- **Grafos y estadísticas**: dispone de múltiples gráficos para analizar los datos de forma sencilla accesibles mediante interfaz web.
- **Alertas**: provee de la capacidad de crear disparadores y acciones que se activen cuando se produce un evento determinado en los logs.

La arquitectura está compuesta por una serie de nodos que se encargan de leer la información de las máquinas a monitorizar y enviarla a los nodos de Elastic Search que almacenan y procesan todos los logs y mensajes recibidos. Para almacenar los metadatos se utiliza MongoDB y fivalmente se proporciona una interfaz web para visualizar los datos recolectados y procesados.

Los administradores de TI encontrarán que la interfaz frontend de Graylog es fácil de usar y robusta en su funcionalidad. Graylog se basa en el concepto de paneles, que le permite elegir qué métricas o fuentes de datos le parecen más valiosas y ver rápidamente las tendencias a lo largo del tiempo.

Cuando ocurre un incidente de seguridad o rendimiento, los administradores de TI quieren poder rastrear los síntomas hasta una causa raíz lo más rápido posible. La función de búsqueda en Graylog lo hace fácil. Tiene tolerancia a fallas incorporada que puede ejecutar búsquedas de múltiples subprocesos para que pueda analizar varias amenazas potenciales juntas.

2.1.2. Nagios

Nagios comenzó con un solo desarrollador en 1999 y desde entonces se ha convertido en una de las herramientas de código abierto más confiables para administrar datos de registro. La versión actual de Nagios puede integrarse con servidores que ejecutan Microsoft Windows, Linux o Unix.

Su producto principal es un servidor de registros, cuyo objetivo es simplificar la recopilación de datos y hacer que la información sea más accesible para los administradores del sistema. El motor del servidor de registro de Nagios capturará datos en tiempo real y los enviará a una poderosa herramienta de búsqueda. La integración con un nuevo punto final o aplicación es fácil gracias al asistente de configuración integrado.

Nagios se usa con mayor frecuencia en organizaciones que necesitan monitorear la seguridad de su red local. Puede auditar una variedad de eventos relacionados con la red y ayudar a automatizar la distribución de alertas. Nagios incluso se puede configurar para ejecutar scripts predefinidos si se cumple una determinada condición, lo que le permite resolver problemas antes de que un humano tenga que involucrarse.

Como parte de la auditoría de red, Nagios filtrará los datos de registro según la ubicación geográfica donde se originan. Eso significa que puede crear paneles de control completos con tecnología de mapeo para comprender cómo fluye su tráfico web.

2.1.3. ELK Stack

Elastic Stack, a menudo llamado ELK Stack, es una de las herramientas de código abierto más populares entre las organizaciones que necesitan examinar grandes conjuntos de datos y dar sentido a los registros de su sistema.

Se trata de un proyecto OpenSource multiplataforma que contempla una serie de productos y soluciones con el objetivo de dotar facilitar al usuario el tratamiento de la información. Las funciones básicas con extraer, buscar, analizar y visualizar datos en tiempo real.

ElasticSerch proporciona distintas formas de implementación, tanto de forma local e una única máquina o de forma distribuida utilizando un modelo SaaS (Software as service).

Su oferta principal se compone de tres productos separados: Elasticsearch, Kibana y Logstash:

- Como sugiere su nombre, Elasticsearch está diseñado para ayudar a los usuarios a encontrar coincidencias dentro de conjuntos de datos utilizando una amplia gama de tipos y lenguajes de consulta. La velocidad es la ventaja número uno de esta herramienta. Se puede expandir en grupos de cientos de nodos de servidor para manejar petabytes de datos con facilidad.
- Kibana es una herramienta de visualización que se ejecuta junto con Elasticsearch para permitir a los usuarios analizar sus datos y crear informes potentes. Cuando instale por primera vez el motor Kibana en su clúster de servidores, obtendrá acceso a una interfaz que muestra estadísticas, gráficos e incluso animaciones de sus datos.
- La última pieza de ELK Stack es Logstash, que actúa como una canalización puramente del lado del servidor hacia la base de datos Elasticsearch. Puede

integrar Logstash con una variedad de lenguajes de codificación y API para que la información de sus sitios web y aplicaciones móviles se alimente directamente a su potente motor de búsqueda Elastic Stalk.

2.1.4. Conclusiones

Tras el análisis realizado de posibles herramientas utilizadas en el mercado se ha optado utilizar por una de ellas, concretamente ELK debido a la extensa documentación y comunidad, así como su posibilidad de escalado y a la rapidez de búsqueda. De esta forma aun teniendo que emplear tiempo en el conocimiento de las funcionalidades de la herramienta, será más eficiente que desarrollar una herramienta desde 0. De esta forma se podrá asignar mas tiempo a las posibilidades de parametrización de la solución para poder ser aplicada a otros proyectos y en otros entornos.

Capítulo 3

Proposta de solución

Los archivos de registro de actividad que sobre los que se obtendrá la información del sistema se dividen en varios archivos, cada uno con un objetivo concreto. En concreto se monitorizarán tres ficheros, los cuales poseen un ciclo de vida que consiste en pasar por un fichero plano, para una vez alcanzado un tamaño determino comprimirse y guardarse como histórico. Si el histórico tiene una antigüedad superior a los días establecidos como máximo se enviarán a un servidor que funcionará como almacenamiento. Estos tres ficheros tienen los siguientes acometidos:

- Registro de interacciones (peticiones y respuestas) con otros sistemas externos e internos.
- Registro de las transformaciones y cálculos que reciben los datos en los distintos servicios de los que se compone la aplicación.
- Registro de la aplicación de entrada que utilizarán los usuarios finales con registro de las operaciones solicitadas y pantallas accedidas.

En los logs de actividad se registrarán el momento en que se registran, la clase en la que se originó el comportamiento y el texto con la información que se está registrando en ese momento.

En este capítulo se va a describir la plataforma implementada para la emisión de alertas. La arquitectura diseñada para la plataforma es la siguiente:

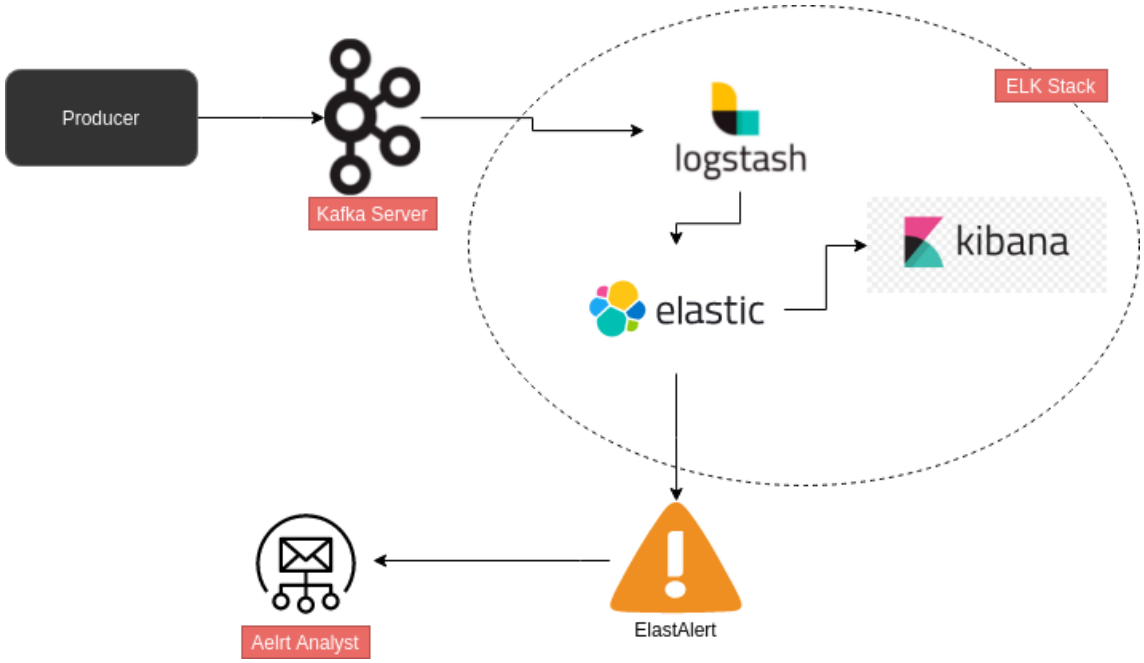

Figura 3.1: Arquitectura de la Plataforma de Gestión de Alertas

La arquitectura estará desplegado en varios contenedores Docker. Se pueden distinguir las siguientes partes:

- **Producer**: como se ha comentado en el capítulo de introducción no es posible utilizar los logs de la aplicación real en el entorno de producción ya que se estaría incumpliendo la ley de protección de datos. Para ello se utilizarán logs ya registrados de los que se tiene certeza que se produjo una incidencia pero procedente de un entorno de desarrollo con datos que no contengan información sensitiva. Para reproducir el comportamiento de la aplicación se implementará un script python que se encargue de leer cada línea de log y trasladarla a kafka, de esta forma replicará el comportamiento de escribir registros a lo largo del tiempo.
- **Kafka server**: se utilizará kafka con el fin de trasladar los datos producidos en el productos al ELK Stack. La plataforma Apache Kafka es un sistema de transmisión de datos distribuido con capacidad de escalado y tolerante a fallos. Gracias a su alto rendimiento permitirá transmitir datos en tiempo real utilizando el patrón de mensajería publish/subscribe.
- **ELK Stack**: conjunto de aplicaciones (Elastic Search, Logstash y Kivana). Permite recoger datos de cualquier tpo de fuente y cualquier formato para

realizar búsquedas, análisis y visualización de los datos en tiempo real. En la arquitectura planteada se para la aplicación real se obtendrán los datos a partir de ficheros de logs mediante Logstash y se almacenarán en el motor de búsquedas y análisis de Elasticsearch. Además, se permite la visualización, monitorización y explotación de datos en tiempo real mediante kibana.

- **Elastalert**: herramienta open source que se integra con ELK y permite detectar anomalías e inconsistencias en los datos de Elasticsearch y lanzar alertas. El conjunto de herramientas de **ELK (Elastic Stack)** más ElastAlert permite crear un sistema de monitorización para la extracción, almacenamiento y explotación de los datos de los procesos o sistemas a monitorizar.

3.0.1. Generador de alertas

Para detectar y emitir alertas como se comentó en el apartado anterior se utilizará ElastAlert. ElastAlert es un componente originalmente diseñado por Yelp que es capaz de detectar anomalías, picos u otros patrones de interés. Se trata de un producto *production-ready* y es un estándar de alerta conocido y aceptado dentro del ecosistema de ElasticSearch. En la documentación se especifica que "Si puede verlo en Kibana, ElastAlert puede alertarlo". ElastAlert podrá realizar distintas operaciones dependiendo que como se configuren las reglas en los ficheros de configuración.

ElastAlert consulta periódicamente a ElasticSearch y los datos se trasladan al tipo de regla, la cual determina cuando se encuentra una coincidencia. Cuando se produce una coincidencia, se envía a una o más alertas que toman medidas en función de la coincidencia. Esto se configura mediante un conjunto de reglas, cada una de las cuales define una consulta, un tipo de regla y un conjunto de alertas. Se podrán definir tipos de regla del estilo:

- **Frequency**: existen una serie de X eventos en un tiempo Y.
- **Spike**:la tasa de eventos aumenta/disminuye
- **Flatline**: menos de X eventos en Y tiempo.
- **Blacklist/Whitelist**: cuando un determinado campo coincide con una lista negra.

- **Any**: cualquier evento que coincida con un filtro dado.

- **Change**: un campo tiene dos valores diferentes dentro de un tiempo.4

En la solución propuesta se distinguirán dos tipos de alertas principales definidas tras un estudio de los logs.

- **Errores conocidos**: en la aplicación se pueden dar ciertos errores relativos a problemas de comunicaciones con otros servicios, caídas de los propios servicios así como problemas de conexión o caída con las bases de datos. Cuando se produce un error de este tipo la mayoría de operaciones van a devolver en la respuesta un código de ERROR del cual se puede obtener la información necesaria para su diagnóstico.

  Para este tipo de errores se generara una lista de reglas con un comportamiento similar:

  1. Se detectará que se está repitiendo un código de error un número configurable de veces en un tiempo determinado.
  2. Cada regla tendrá asignado un diagnóstico y las acciones a seguir en cada caso.
  3. Se enviará un correo especificando el error detectado, el diagnóstico y los pasos a seguir al correo de notificación de incidencias.

- **Errores desconocidos**: por otro lado pueden darse errores desconocidos en la aplicación. Estos al contrario que los anteriores no tienen un fácil diagnóstico ni hay una serie de acciones por defecto a seguir. Se puede tratar de un gran abanico de posibilidades a la hora de buscar la causa, desde un mal tratamiento de una respuesta de un sistema o un bug que se introdujo en el código.

  En este tipo de errores se buscará realizar un tratamiento de los logs aplicando Machine Learning para detectar anomalías que puedan ayudar al responsable a diagnosticar y buscar una solución. En este caso se definirá un regla, que en caso de detectar un error no definido obtenga el identificador de la transacción de la operación y haga una llamada al servicio del componente de Alert Analist con el fin de que trate los logs obtenidos.

3.0.2. Reducción de log

Cuando la aplicación falla con un error indeterminado se genera un log distinto al normal. Encontrar la causa del error puede ser un trabajo muy tedioso ya que involucra el estudio de los logs para encontrar la fracción del registro que se escapa de la operativa normal. Para hacer esta tarea más simple se utilizará LogReduce, el cual es un librería de código abierto que permite la generación de modelos Machine Learning entrenándolo con ejecuciones exitosas de procesos anteriores para extraer anomalías de los registros de ejecuciones fallidas.

Los logs de la aplicación se componen de un registro de los distintos servicios que permite la aplicación. Cada vez que un servicio es invocado se guardará un registro de la petición, las transformaciones realizadas sobre los datos, información referente a las casuísticas concretas y de la respuesta. El conjunto de los registros comunes en la ejecución de un servicio que finalice correctamente se conocerá como ***Basenile***. En caso de se produzca algún tipo de error durante la ejecución del servicio se producirán excepciones que ayudarán a definir la causa del fallo.

Para eliminar los *Baselines* de un log errado se podrá utilizar un algoritmo de k vecinos más próximos (**k-nearest neighbors pattern recognition algorithm k-NN**).

Para ello, cada evento registrado en el log deberá ser convertido a un valor numérico aplicando algoritmos de Hashing Vectorizer. De esta forma se consigue mapear cada palabra y codificar cada evento en una matriz dispersa. Para mejorar este proceso se deberá aplicar una limpieza del texto quitando stop-words (palabras que no tienen significado por si solas), así como un proceso de tokenización para eliminar datos aleatorios como IPs y marcas temporales.

Una vez el modelo esté entrenado se podrá calcular la distancia de cada evento con respecto al *Baseline* y mostrar aquellos resultados con un valor de distancia mayor que un umbral fijado.

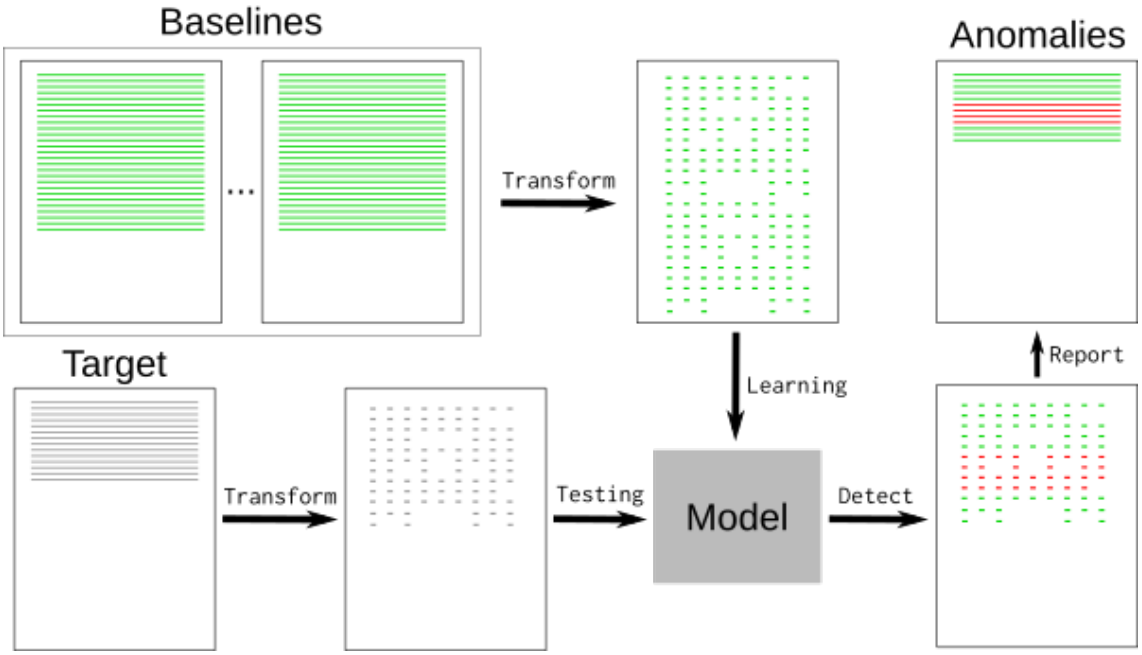

Figura 3.2: Pipeline logreduce

La librería LogReduce implementa este proceso con las ventajas de que se pueden especificar múltiples *Baselines* para entrenar el modelo, así como la generación de reportes html para la mejor visualización del proceso.

El proceso de entrenamiento por tanto podrá ser cíclico de forma que en la que se vayan generando nuevos *Baselines* añadirlos a la construcción del modelo. De esta forma el modelo estará preparado para cambios y evolucionará de igual manera que lo hace la aplicación.

Capítulo 4

Conclusiones y posibles ampliaciones

4.0.1. Conclusiones

Se consideran las siguientes conclusiones acerca de los objetivos propuestos:

- **Manejo de los logs**: Se ha conseguido una forma correcta de manejar la cantidad de logs generada dotándola de sentido para el observador gracias la utilización del ELK Stack. Kivana proporciona posibilidades de generación de informes con el fin de ayudar a los responsables a obtener información relevante a partir de los registros de la aplicación.
- **Generación de alertas**: teniendo en cuenta que el proceso de detección de alertas se está ejecutando sobre un entorno con una cantidad de datos inferior a la real es lo suficiente rápido como para que suponga una mejora en los servicios de operación de la aplicación. Para asegurar este comportamiento se realizarían pruebas de rendimiento sobre entornos de preproducción con cantidades de datos semejantes a producción.
- **Obtención de anomalías**: la aplicación de LogReduce sobre los registros de errores desconocidos genera informes en los que efectivamente se puede observar la causa raíz del error. En este punto se obtienen falsos positivos debido a la estructura compleja de los logs y a que existen un número muy

grande de posibilidades, sin embargo, aún no siendo un resultado óptimo si que se considera una mejora en el proceso de revisión.

4.0.2. Posibles ampliaciones

1. **Servidor Cloud ejecutando ELK**: la arquitectura propuesta de ELK podría estar alojada en un entorno Cloud como podría ser Amazon Web Services (AWS) o Windows Azure o bien servidores propios interconectados entre si. Tendía sentido dividir los distintos elementos en diferentes máquinas, por ejemplo, añadir un elemento como FileBeat de forma local, que envíe la información de los logs a un LogStash remoto y este, a su vez, enviara los datos refinados a otra máquina que ejecutaría Elastic Search y Kibana. Se podría garantizar la visualkización de los datos a través de Kibana en una url fija accesible desde equipos remotos.

2. **Alertas por picos de actividad o tiempos de respuesta altos**: a la hora de monitorizar la actividad aportaría valor el hecho del registro de eventos que suponen un peor rendimiento para la aplicación. Podrían ser tiempos de respuesta altos que no lleguen a ser TimeOuts o registrar picos de usuarios a ciertas horas que podrían para localizar momentos en los que se espera que podría producirse caídas.

3. **Detector de anomalías**: se podría plantear añadir un módulo que se encargue de validar los comportamientos normales de la aplicación, con el fin de localizar y minimizar el fraude de los responsables que utilizar la aplicación.

Bibliografía

[1] Logstash Reference (*https://www.elastic.co/guide/en/logstash/current/index.html*). Consultado el 1 de octubre de 2021.

[2] E. BV, .elastic.co,.Elasticsearch BV, 2021. [En línea]. Disponible: https://www.elastic.co/guide/en/kibana/current/index.html. Consultado el 1 de octubre de 2021.

[3] Han, J. Kamber, M. Pei, J. Data Mining: Concepts and Techniques. Ed. 3, Morgan Kaufmann Publishers Inc., 2003.