Redborder Vault documentation

Redborder

Version 0.0.3

Índice

Capítulo 1: Introducción

En este capítulo introduciremos el servicio de Logstash y su integración dentro de la arquitectura de Redborder.

Redborder Vault

Cuando se depende de varios servicios para la operatividad de un producto o infraestructura es necesario disponer de un sistema que nos permita una rápida y simple gestión de los múltiples logs que se generan. Es aquí donde entra en juego Redborder Vault.

Redborder Vault es nuestro módulo de análisis de logs y eventos del sistema que permite la monitorización de logs así como su integración con el CEP para la correlación de eventos. Para llevar a cabo todo el tratamiento de datos se hace uso del conocido servicio Logstash.

Logstash

Logstash es una herramienta open source que permite el procesamiento de datos. El procesamiento se hace a través de la definición de un pipeline que permite la ingesta de datos desde múltiples fuentes, los transforma a través de uno o varios filtros y finalmente los envía a distintos silos de almacenamiento.

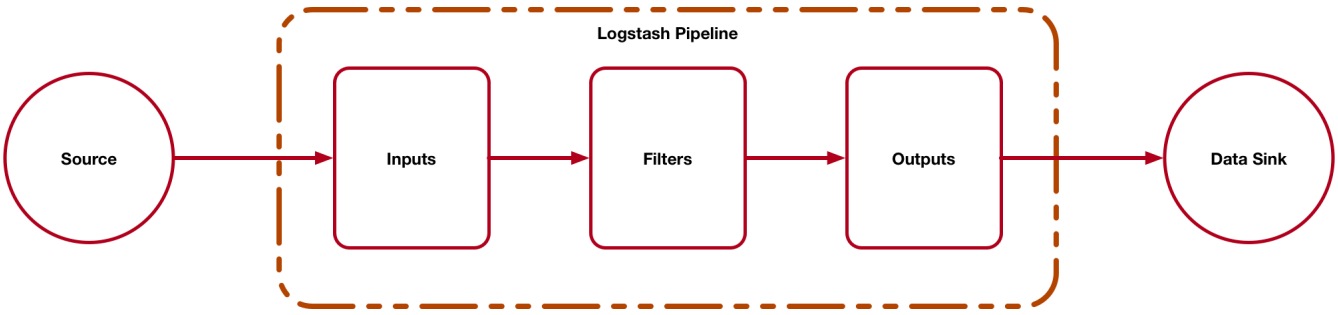


Figure 1. Logstash Pipeline

Además, Logstash cuenta con una comunidad muy activa que está en continuo crecimiento ofreciendo una gran variedad de plugins y filtros.

Arquitectura en Redborder Vault

Gracias a la combinación de Logstash y Kafka podemos ofrecer un servicio realmente potente y escalable para la monitorización de logs de múltiples servicios.

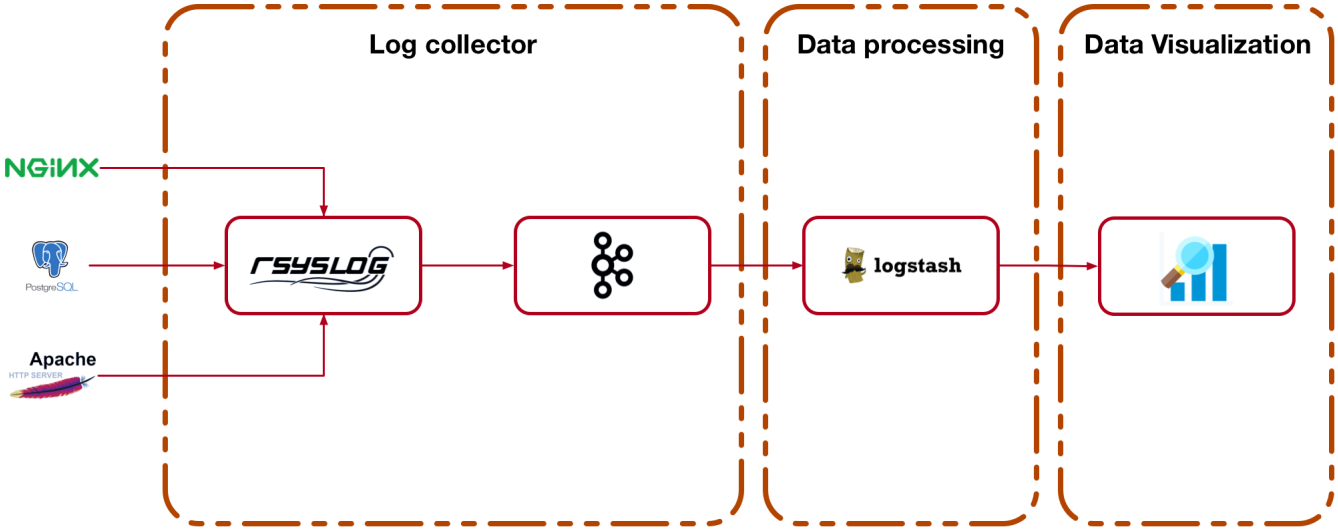


Figure 2. Arquitectura Redborder Vault

Con una interfaz sencilla y muy intuitiva podrá visualizar todos los eventos de su entorno para asegurarse de que tiene la mejor respuesta en cada uno de los servicios.

Capítulo 2: Arquitectura interna de Vault

En este capítulo explicaremos y analizaremos el funcionamiento interno de redborder Vault.

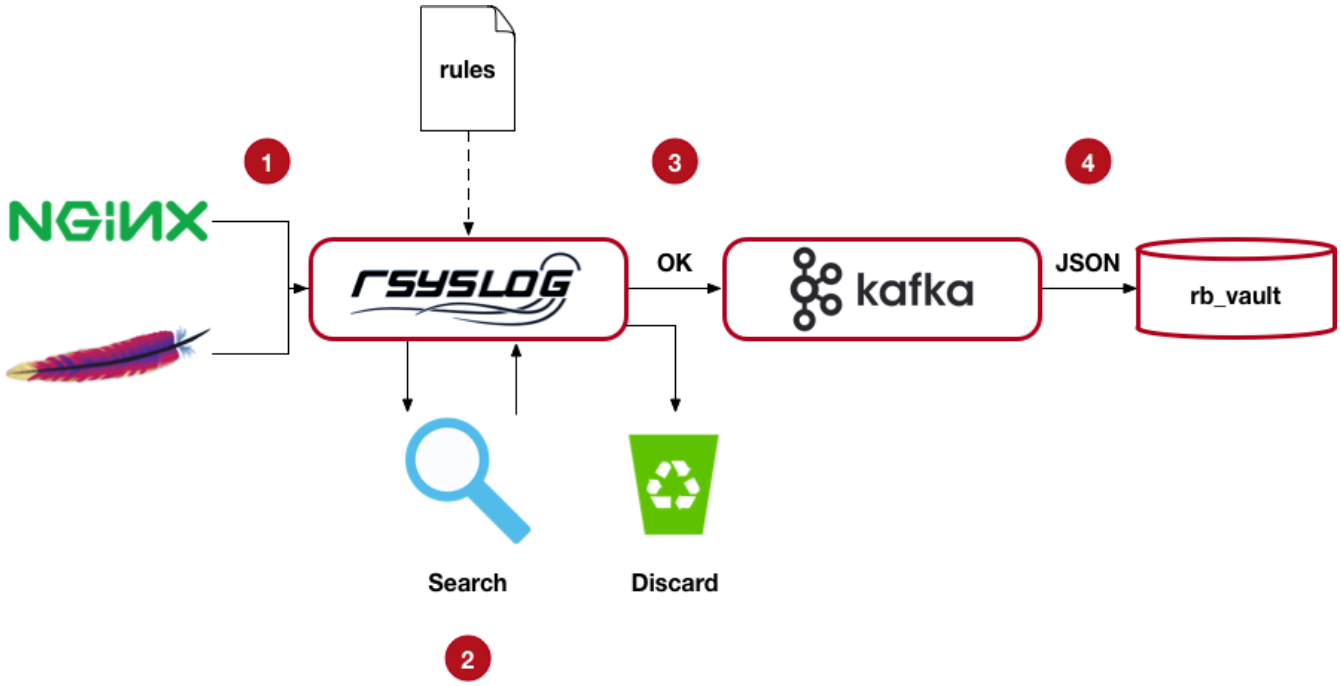
Ruta de datos

Desde que los datos son recolectados hasta que son mostrados a través de la interfaz de usuario, pasa por tres fases para facilitar la recolección, el procesado y visualización de los datos.

Cada una de estas fases tiene un peso significativo y son críticas para el correcto funcionamiento de redborder Vault.

Colector de logs

En esta primera fase se recolectan los logs de los distintos servicios, esto se hace utilizando la herramienta rsyslog. Una vez obtenidos los logs son clasificados por aplicación y posteriormente enviados a Kafka para su persistencia y posterior procesado.

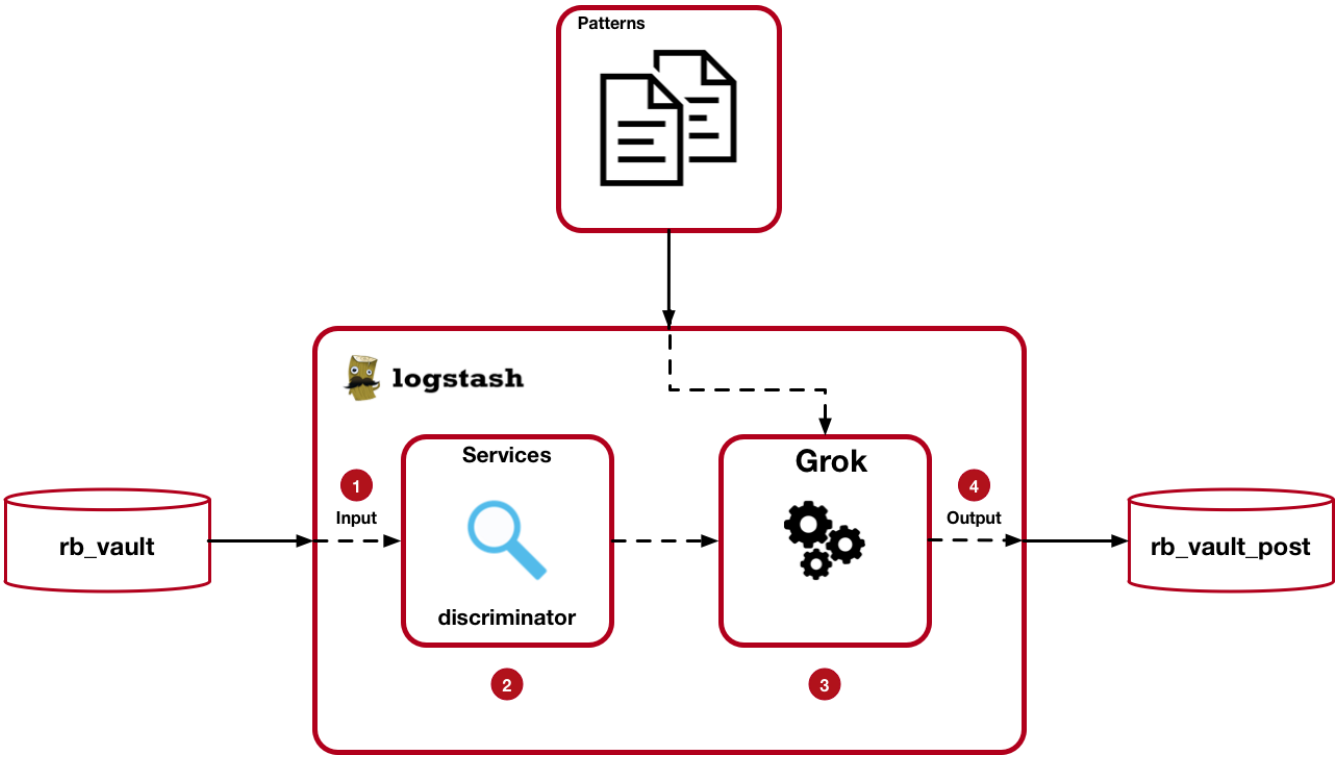


En la imagen anterior podemos ver en detalle el proceso que sigue el recolector de log. Podemos resumir esta fase en cuatro pasos:

1. El primer paso consiste en la recepción de los logs de los diferentes servicios en rsyslog.
2. A continuación rsyslog procederá a buscar aquellos logs que haga match con una serie de reglas definidas con los logs más comunes y útiles.
3. Una vez que se ha buscado esos logs específicos, rsyslog los envía Kafka en formato JSON. Aquellos logs que no hagan match serán desechados.
4. Por último, al recibir Kafka los logs los almacena en su correspondiente topic para su posterior procesado.

Procesado de datos

El procesado de los logs suele ser una tarea bastante compleja, sobretodo por la variedad de logs y los distintos fabricantes y versiones de un mismo servicio. Por ello logstash es una buena opción, ya que el uso de uno de sus plugins más famoso nos permite crear reglas y aplicarlas para la obtención de información relevante.



En la imagen anterior podemos ver un poco más en detalle la fase de procesado de datos. Los pasos que se llevan a cabo son los siguientes:

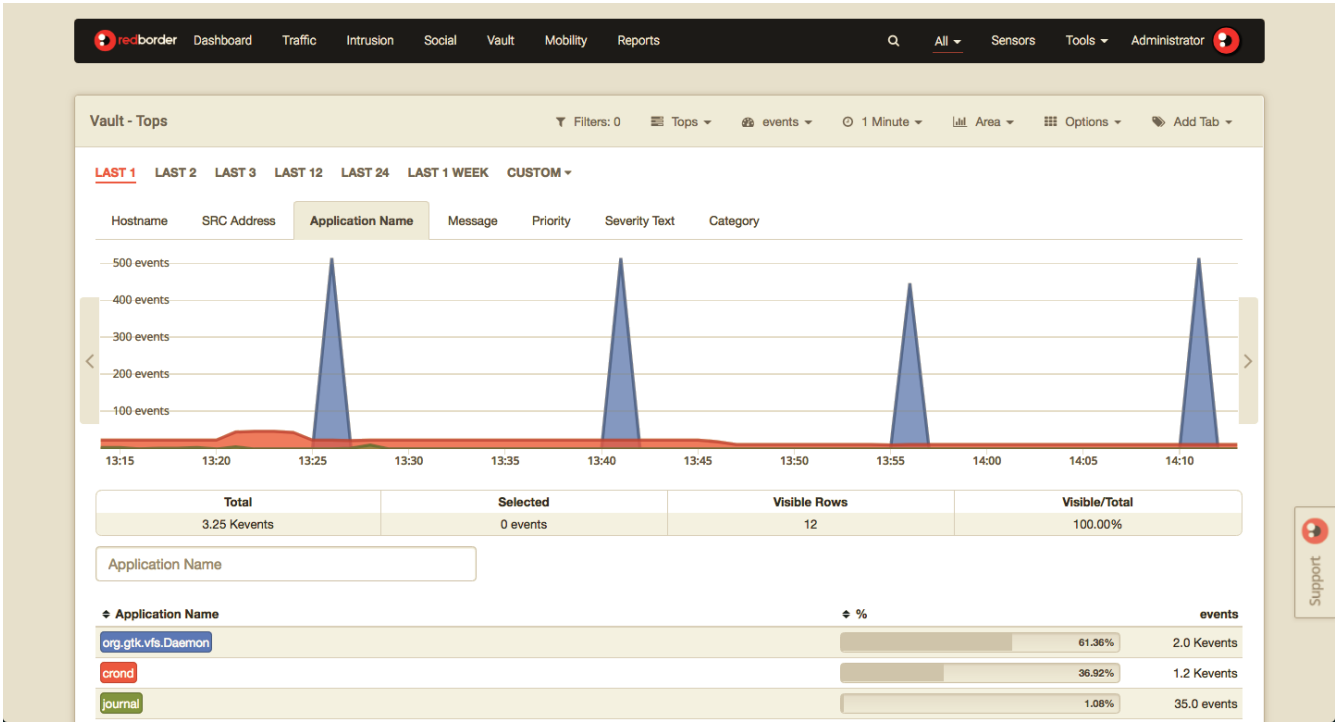
1. El primer paso es alimentar a logstash con los datos obtenidos por la fase de recolección de logs.
2. En cada JSON tenemos un campo denominado `app_name` que permite discriminar por aplicación y aplicar filtrado.
3. Gracias al plugin de logstash Grok, podemos interpretar los logs y obtener la información relevante. Para ello se carga una serie de patrones.
4. Por último, una vez que se ha obtenido la información necesaria se envía a un nuevo topic de kafka para su análisis en la plataforma de redborder.

En el último paso se hace una clasificación de los campos `target`, `status`, `source` y `action` en base a las reglas definidas por los logs, la definición de estos campos se puede encontrar en la tabla inferior. Los ejemplos expuestos son los obtenidos a partir de un servicio web de apache:

Campo	Definición	Ejemplo
action	Acción realizada	GET, POST, PUT, DELETE ...
target	Objetivo de la acción realizada	https://redborder.com
status	Efecto que ha tenido la acción	200, 404, 503 ...
source	Origen de la acción	24.109.71.77

Visualización de datos

Una vez que se han recolectado y analizado los logs para obtener los campos relevantes, la plataforma de redborder analiza e indexa los datos para su consulta e interpretación.



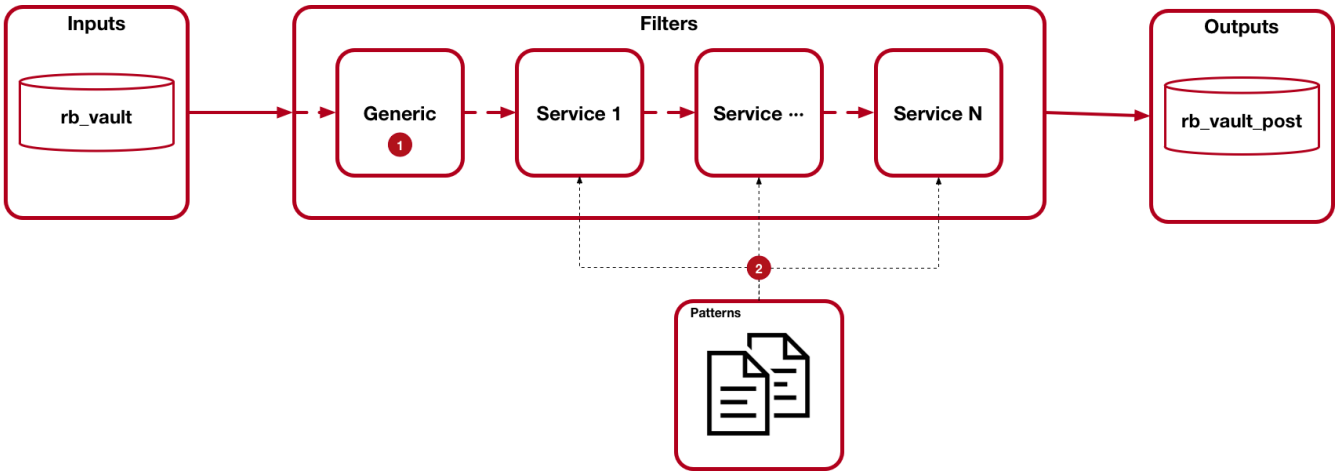
Capítulo 3: Procesado de datos

En este capítulo analizaremos un poco más a fondo el procesado de los datos. La fase de procesado de datos es importante para la obtención de los campos `action`, `target`, `status` y `source` y obtener información relevante de los logs que se reciban en el sistema.

Pipeline

El procesado de eventos de logstash tiene tres fases: inputs → filters → outputs. Los inputs generan los eventos, los filtros los modifican y los outputs los envía. Tanto los inputs como los outputs soportan codecs para la codificación y decodificación de datos sin la necesidad de utilizar filtros adicionales.

Actualmente el pipeline de Vault tiene un input y un output. Ambos están conectados a un broker de kafka para leer y escribir de los topics `rb_vault` y `rb_vault_post` respectivamente.



En la imagen anterior podemos ver un poco más en detalle el pipeline de logstash utilizado para el procesado de los datos de Vault.

Obviando el input y output, cuya funcionalidad es leer y escribir datos a los topics de kafka comentados, podemos observar que hay una serie de filtros que sigue los siguientes pasos:

1. En primer lugar pasa por el filtro `Generic`. Este filtro se encarga de eliminar algunos campos y de generar un hash del mensaje en bruto.
2. En este paso el mensaje pasa por los filtros de forma secuencial, obteniendo la información relevante aplicando para ello una serie de reglas.

Filtros de servicios

Los filtros definido para cada uno de los servicios tienen la siguiente estructura:

```
filter {
  if "nginx" in [app_name] {
    grok {
      patterns_dir => ["/etc/logstash/conf.d/patterns"]
      match => {
        "message" => [
          "%{NGINXACCESS}"
        ]
      }
    }
    mutate {
      add_field => {"source" => "%{remote_ip}"}
      add_field => {"status" => "%{response}"}
      add_field => {"target" => "%{request}"}
      add_field => {"action" => "%{request_action}"}
    }
  }

}
```

Como puede observarse la estructura es bastante sencilla. En primer lugar discriminamos el servicio con el campo `app_name` si coincide se aplica los patrones correspondientes.

Además de llevar a cabo este paso se añaden los campos que posteriormente se utilizarán para su análisis y consulta.

Si observamos el conjunto de reglas, podremos ver que tienen la siguiente estructura:

```
NGINXACCESS %{IPORHOST:remote_ip} - %{DATA:user_name} \[%{HTTPDATE:time}\]
"%{WORD:request_action} %{DATA:request} HTTP/%{NUMBER:http_version}"
%{NUMBER:response} %{NUMBER:bytes} "%{DATA:referrer}" "%{DATA:agent}"
```

Se trata pues de una abstracción de las conocidas expresiones regulares. Muchas de estas reglas ya están definidas, sin embargo otras tienen que definirse completamente.

Para más información sobre el plugin Grok y la creación de reglas, puede consultar el siguiente enlace.