



Universidad
Nacional
de Quilmes

Trabajo Práctico de
Laboratorio de Redes y Sistemas
Operativos

2do cuatrimestre 2019

Fluentd



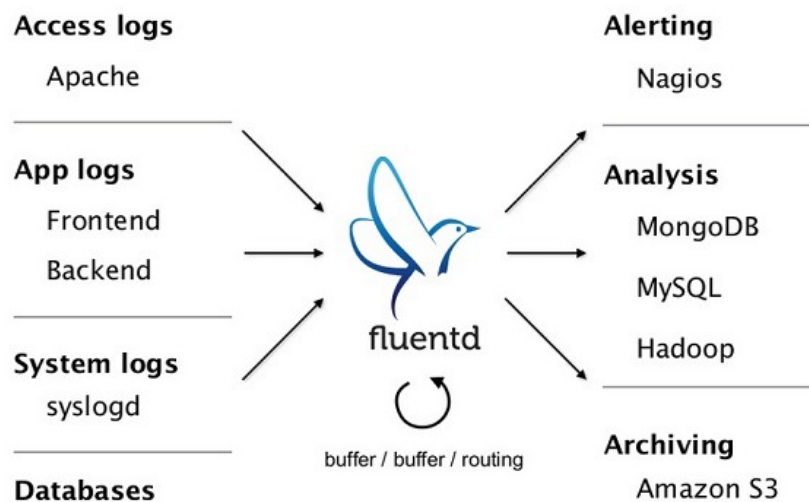
Integrantes: Cameriere Federico, Valenzuela Horacio

Índice

1. ¿Qué es Fluentd?
2. Descripción del proyecto
3. Computadoras utilizadas
4. Instructivo de instalación, configuración y uso
5. Problemas encontrados
6. Fuentes

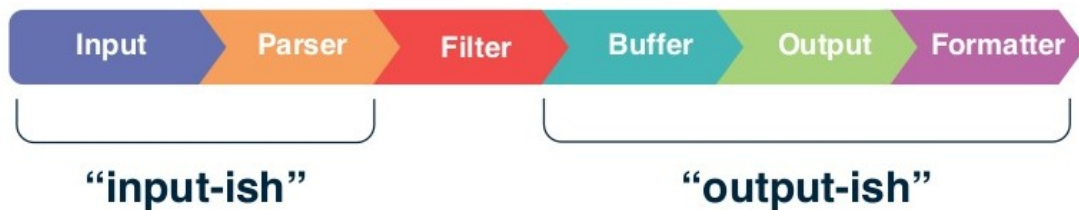
1) ¿Qué es Fluentd?

Fluentd es un colector de datos opensource, particularmente pensado para manejo de logs de múltiples orígenes y/o hacia múltiples salidas (almacenamiento de diferentes índices).



Su estructura se divide en lo que es el “core” (núcleo) y los plugins.

En el core se manejan operaciones de buffering, reintentos de requests, manejo de errores, etc. Ahí se encuentra el desarrollo más performante de la herramienta, que está codeado en C (y un poco en Ruby), a diferencia de los plugins que están diseñados para ser codeados en Ruby (esto es una estrategia para abarcar un espectro más amplio de desarrolladores que puedan contribuir a la herramienta opensource).



Imágen de los diferentes tipos de plugins que maneja actualmente Fluentd.

Fluentd está pensado para no tener que programar demasiado, sino más bien entender sus plugins y utilizarlos de forma combinada de ser necesario (también se puede desarrollar un plugin propio en Ruby, y de ser funcional idealmente compartirlo via github para que otros lo usen). Por ello es que cuenta con una muy amplia gama de plugins (algunos desarrollados y mantenidos por el equipo encargado del desarrollo de la herramienta, y una mayoría desarrollados por programadores que contribuyeron al proyecto opensource con su/s plugin/s).

Para manejar los plugins, la herramienta cuenta con un archivo de configuración para el que la mayoría de los plugins más usados tienen ejemplos (en sus respectivas documentaciones) de los parámetros que se deben utilizar para ellos.

La unidad más pequeña de datos para fluentd que "se mueve" entre los plugins, es el "evento" (que tiene forma JSON). El mismo está compuesto de 3 partes:

1. Timestamp, que tiene el dato de la fecha y hora del evento inicial (usualmente un log).

2. Tag, que es un identificador para el evento.
3. Registro (record), que es donde van los datos de interés del evento en sí.

Las posibilidades de uso de Fluentd son variadas. En principio hay 2 tipos de plugins que vamos a querer utilizar siempre, "Input" y "Output", para obtener datos de un lugar y enviarlos a otro lugar respectivamente. Se puede usar más de 1 tipo de cada uno, del mismo modo que con los demás tipos de plugin.

Para ver una buena lista de formas usuales de usar la herramienta, dirigirse a <https://www.fluentd.org/guides>.

2) Descripción del proyecto

En principio nuestra idea era obtener mails, mensajes de diferentes redes sociales (al menos 1) y mensajes de herramientas de chat (whatsapp/telegram). Sabíamos que era ambicioso, pero no supimos cuánto hasta que nos chocamos con que debíamos desarrollar uno o varios plugins en Ruby (lenguaje que desconocemos) para alcanzar los objetivos propuestos. Dado esto, bajamos la vara y nos propusimos al menos lograr obtener mails o mensajes de alguna red social.

Entonces tras buscar plugins y tratar de entender el funcionamiento de los mismos (hasta cierto punto, ya que el código era Ruby), nos encontramos con uno de Twitter (desarrollado por terceros a la herramienta). Asimismo encontramos uno de Slack (también desarrollado por terceros), que nos parecía más interesante que volcar los logs a una base de datos (lo que era nuestro proyecto inicial).

El nuevo proyecto entonces era:

- A) Obtener tweets de una cuenta de twitter de prueba y los tweets de las cuentas que siguiéramos con esa cuenta de prueba (sin retweets).
- B) Finalmente volcarlos a un canal de slack en un servidor creado de prueba.

3) Computadoras utilizadas

PC1 (desktop):

- Ram: 16GB
- CPU: AMD Ryzen 3 2200G
- Sistema Operativo: Ubuntu 18.04.3 LTS (Bionic)

PC2 (notebook):

- Ram: 2GB
- CPU: Intel Core 2 Duo T5870
- Sistema Operativo: Ubuntu 18.04.3 LTS (Bionic) (Distro: Lubuntu de 32bits)

4) Instructivo de instalación, configuración y uso

a) Instalación

- ▶ Instalación de la herramienta

Para Ubuntu de 64 bits:

Vamos a instalar “td-agent”, que es una herramienta muy similar a la original de “fluentd” pero que facilita el manejo de los y la instalación de plugins por ejemplo, además de ya traer un ejecutable para hacer pruebas y el archivo de configuración con algunas configuraciones de prueba. Básicamente es una versión

más estable que “fluentd” de base, con pequeños agregados y principalmente el mantenimiento de “Treasure Data” que es la compañía desarrolladora de ambas herramientas (igualmente usa fluentd en su núcleo).

Para la instalación de esta herramienta utilizaremos el siguiente comando:

```
curl -L https://toolbelt.treasuredata.com/sh/install-ubuntu-bionic-td-agent3.sh | sh
```

Luego de eso, iniciamos el servicio con el siguiente comando:

```
sudo systemctl start td-agent.service
```

Y este otro para verificar el estado del mismo:

```
sudo systemctl status td-agent.service
```

Ahora vamos a usar el ejecutable que facilita el uso de la herramienta:

```
sudo /etc/init.d/td-agent restart
```

Y verificamos el estado:

```
sudo /etc/init.d/td-agent status
```

Ya tenemos funcionando el servicio de fluentd.

Ahora podemos detener el servicio, de así desearlo, con:

```
sudo /etc/init.d/td-agent stop
```

Para Ubuntu de 32 bits:

Vamos a instalar Ruby (para poder instalar y correr la herramienta..además que serviría en caso de querer desarrollar nuestro propio plugin a futuro).

Para esto primero hacemos:

```
sudo apt update
```

Y luego usamos:

```
sudo apt install ruby-full
```

Para verificar que se instaló correctamente, usamos:

```
ruby --version
```

Que nos debería mostrar algo como lo siguiente:

```
ruby 2.5.1p57 (2018-03-29 revision 63029) [i386-linux-gnu]
```

Ahora que tenemos Ruby instalado, vamos a pasar a instalar fluentd con el siguiente comando:

```
sudo gem install fluentd --no-ri --no-rdoc
```

Teniendo instalado fluentd, pasamos a correr los siguientes comandos para verificar que se instaló correctamente:

```
sudo fluentd --setup ./fluent
```

```
fluentd -c ./fluent/fluent.conf -vv &
```

```
echo '{"json":"message"}' | fluent-cat debug.test
```

Habiendo corrido dichos comandos, debería mostrarse lo siguiente en pantalla:

```
2019-08-12 13:44:02 -0300 debug.test: {"json":"message"}
```

Ahora se está corriendo el daemon de fluentd. Para detenerlo podemos usar el siguiente comando:

```
kill -f fluentd
```

► Instalación de los plugins

Para Ubuntu de 64 bits:

El próximo paso para preparar el servicio como lo queremos es instalar los plugins necesarios:

Para empezar vamos a instalar el plugin de Twitter.

Lo primero es instalar la librería de las dependencias:

```
sudo apt-get install build-essential libssl-dev
```

Luego pasamos a instalar el plugin:

```
sudo td-agent-gem install eventmachine  
sudo td-agent-gem install fluent-plugin-twitter
```

Por ahora no lo vamos a agregar al archivo de configuración, porque necesitamos una “app” de twitter con sus claves de acceso (lo dejamos para más adelante).

Ahora pasamos a instalar el plugin de Slack:

```
sudo td-agent-gem install fluent-plugin-slack
```

De nuevo vamos a dejar para más adelante la modificación del archivo de configuración, ya que necesitamos crear una url de webhook junto con una app de Slack para usar el plugin.

El próximo plugin que instalaremos será filter_keys:

```
sudo td-agent-gem install fluent-plugin-filter_keys
```

El plugin “filter_keys” lo vamos a usar para filtrar los retweets, ya que por defecto el plugin de twitter va a

obtener todos los tweets de las cuentas de twitter seleccionadas, pero además los (indeseados) retweets de los mismos.

Ahora instalamos el plugin record_modifier:

```
sudo td-agent-gem install fluent-plugin-record-modifier
```

Este plugin último plugin lo usaremos para modificar el registro (dato de interés) de los eventos que manejaremos, para que tengan el formato y el mensaje deseado a la hora de pasarlo al plugin de slack (cabe aclarar que el plugin “record_modifier” es bastante robusto y nos permitirá incluso usar lógica de código Ruby en el archivo de configuración).

Para Ubuntu de 32 bits:

El próximo paso para preparar el servicio como lo queremos es instalar los plugins necesarios:

Para empezar vamos a instalar el plugin de Twitter.

Lo primero es instalar la librería de las dependencias:

```
sudo apt-get install build-essential libssl-dev
```

Luego pasamos a instalar el plugin:

```
sudo gem install eventmachine  
sudo gem install fluent-plugin-twitter
```

Por ahora no lo vamos a agregar al archivo de configuración, porque necesitamos una “app” de twitter con sus claves de acceso (lo dejamos para más adelante).

Ahora pasamos a instalar el plugin de Slack:

```
sudo fluent-gem install fluent-plugin-slack
```

De nuevo vamos a dejar para más adelante la modificación del archivo de configuración, ya que necesitamos crear una url de webhook junto con una app de Slack para usar el plugin.

El próximo plugin que instalaremos será filter_keys:

```
sudo gem install fluent-plugin-filter_keys
```

El plugin "filter_keys" lo vamos a usar para filtrar los retweets, ya que por defecto el plugin de twitter va a obtener todos los tweets de las cuentas de twitter seleccionadas, pero además los (indeseados) retweets de los mismos.

Ahora instalamos el plugin record_modifier:

```
sudo fluent-gem install fluent-plugin-record-modifier --no-document
```

Este plugin último plugin lo usaremos para modificar el registro (dato de interés) de los eventos que manejaremos, para que tengan el formato y el mensaje deseado a la hora de pasarlo al plugin de slack (cabe aclarar que el plugin "record_modifier" es bastante robusto y nos permitirá incluso usar lógica de código Ruby en el archivo de configuración).

► Obtención de claves para apps de twitter y slack

Perfecto, ya tenemos los plugins que vamos a usar.

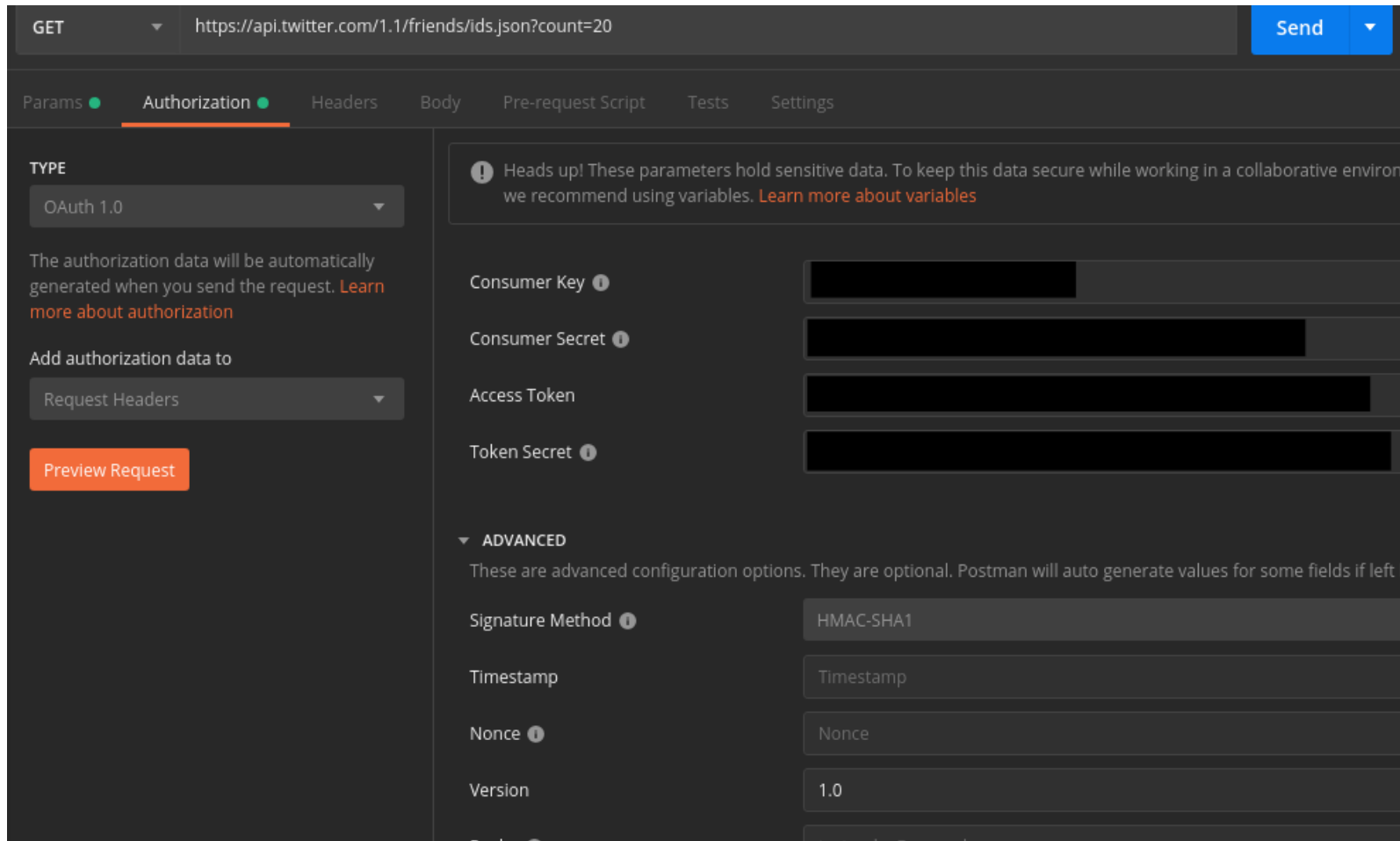
Ahora vamos a encargarnos de obtener las claves necesarias para el plugin de twitter y el webhook para el de slack.

Primero con el de twitter: (un video ágil para obtener la cuenta de twitter con permisos de dev y crear la app para obtener las claves es el siguiente: https://www.youtube.com/watch?v=M_gGUqhCJoU)

- I. Vamos a registrarnos en twitter (o si ya tenemos una cuenta, podemos saltar este paso).

- II. Ahora necesitamos registrar nuestra cuenta como una cuenta de devs de twitter para poder crear la app. Para esto se puede utilizar el siguiente link (a la hora de realizar este documento): <https://developer.twitter.com/en/apply-for-access>
- III. Una vez con nuestra cuenta de devs, vamos a crear al app necesaria para obtener las claves que debemos pasarle al plugin de twitter. Una vez logueados con la cuenta de dev de twitter podemos acceder a este link (al momento de realizar este documento): <https://developer.twitter.com/en/apps> y allí crear la app. Para el campo de la "url" que es requerido podemos poner una url de nuestro github por ejemplo, ya que no es importante para este ejercicio.
- IV. Con la app creada, vamos al tab de "keys and tokens" (o "claves y tokens") y copiamos las claves de "consumer" y creamos y copiamos los tokens. Esas son las claves necesarias para usar el plugin de twitter.
- V. Recomiendo seguir cuentas populares con la cuenta de twitter que vayamos a usar de prueba, para facilitar las pruebas (ya que queremos que se genere al menos 1 tweet por minuto para testear).
- VI. Finalmente necesitamos obtener los ids de las cuentas, tanto de las que seguimos, como de la que queremos controlar. Para obtener el id de la que queremos controlar, podemos hacerlo directamente desde twitter.
- VII. El proceso no es complicado (a la hora de escribir este documento): vamos a "More" > "Settings and Privacy" > "Account" > "Data and Permissions > Your Twitter data" > "Account".. ahí en el tab "Username" veremos nuestro ID, que debemos copiar para pasarselo al plugin de twitter.
- VIII. Ahora para obtener los ids de las cuentas que seguimos podemos usar postman y consumir la siguiente api de twitter: <https://api.twitter.com/1.1/friends/ids.json> con un GET y con autorizacion por medio de Oauth 1.0 en el header. Como se muestra en la imagen, completando los campos de las claves e ignorando el parametro en la url "?count=20" que solo limita el

resultado a las primeras 20 ids. Estas claves también se las vamos a pasar al plugin de twitter.

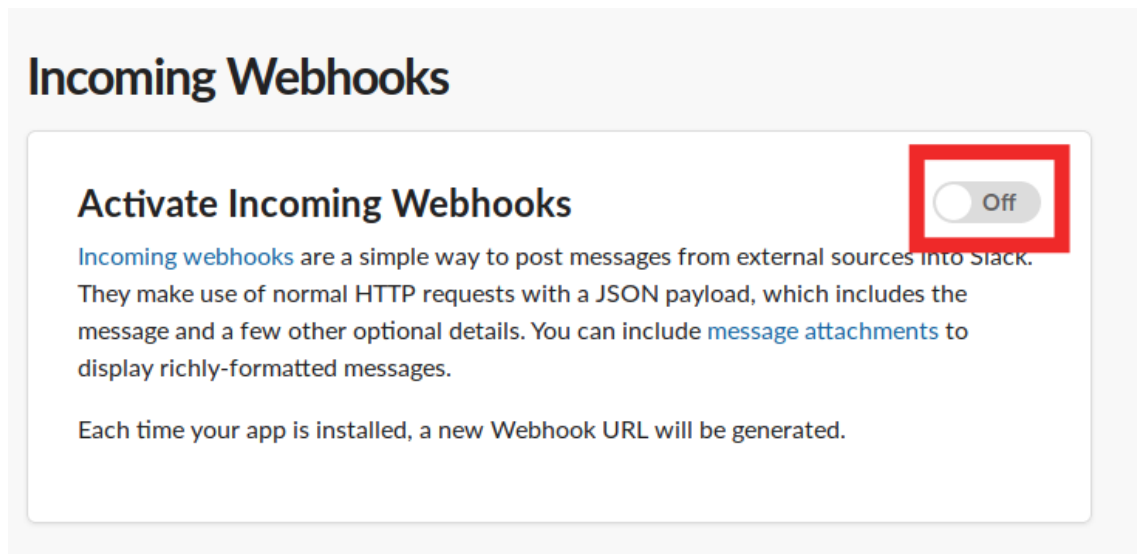


Ahora hacemos lo propio para el webhook de slack:

- I. Primero creamos un espacio de trabajo de slack de prueba (o usamos uno que tengamos).
- II. Luego tenemos que crear la app de slack para poder asociarle un webhook como se indica en el siguiente link (click en el boton verde que dice "Create app"):
https://api.slack.com/messaging/webhooks#enable_webhook
s. Allí le ponemos un nombre y la asignamos al espacio de trabajo que creamos. Ahora hacemos click en "Incoming

Webhooks” y allí le hacemos click al “toggle” para activarlos (como se aprecia en la siguiente imagen).

- III. Una vez hecho eso, tenemos que crear el webhook con el boton de “Add New Webhook to Workspace”. Le asignamos un canal, que será al cual mandará los tweets.
- IV. Finalmente copiamos la url del webhook, que es la que le pasaremos al plugin de slack.



b) Configuración

Por fin, ya tenemos tanto las claves/credenciales necesarias de la app de twitter, como la url del webhook de la app de slack.

Para Ubuntu de 64 bits:

Ahora si, vamos a meternos en el archivo de configuración de fluentd que se encuentra en la ruta: /etc/td-agent/td-agent.conf

Lo abrimos con el editor de texto de preferencia (por ejemplo):

```
sudo gedit /etc/td-agent/td-agent.conf
```

Para Ubuntu de 32 bits:

Ahora si, vamos a meternos en el archivo de configuración de fluentd que se encuentra en la ruta: /etc/fluent/fluent.conf

Lo abrimos con el editor de texto de preferencia (por ejemplo):

```
sudo gedit /etc/fluent/fluent.conf
```

Para ambas versiones de ubuntu:

Entonces vamos a borrar lo que teniamos en el archivo y ponemos lo siguiente:

```
<source>
  @type twitter
  consumer_key XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  consumer_secret XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  access_token XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  access_token_secret XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  tag input.twitter
  timeline tracking
  follow_ids 123, 456, 789, 10
  lang en,es
  output_format flat
</source>
```

Reemplazando las "X"s por las claves de la app de twitter.

Luego de eso ponemos lo siguiente:

```

<match input.twitter>
  type filter_keys

  add_tag_prefix filter_keys.
  denied_keys    retweeted_status_id
</match>

<filter filter_keys.input.twitter>
  @type record_modifier

  # remove all keys except for text and extended_tweet_full_text
  whitelist_keys text, extended_tweet_full_text
</filter>

<filter filter_keys.input.twitter>
  @type record_modifier
  remove_keys _dummy_
  <record>
    _dummy_ ${if !record.has_key?('extended_tweet_full_text'); record['message'] = record['text']; end; nil}
  </record>
</filter>

<filter filter_keys.input.twitter>
  @type record_modifier
  remove_keys _dummy_
  <record>
    _dummy_ ${if record.has_key?('extended_tweet_full_text'); record['message'] = record['extended_tweet_full_text']; end; nil}
  </record>
</filter>

<filter filter_keys.input.twitter>
  @type record_modifier

  # remove all keys except for message
  whitelist_keys message
</filter>

```

Finalmente vamos a poner:

```
<match filter_keys.input.twitter>
  @type copy
  <store>
    @type stdout
  </store>

  <store>
    @type slack
    webhook_url https://hooks.slack.com/services/WEBHOOK\_TEMPLATE
    channel NOMBRE DEL CANAL DE SLACK
    username NOMBRE DE LA APP DE SLACK
    flush_interval 60s
  </store>
</match>
```

Reemplazando la url falsa por la url del webhook creado para la app de slack. El argumento de “channel” debe ser el nombre del canal de slack en el que querramos que envíe los mensajes la app. Y finalmente el argumento de “username” debe ser el nombre de la app de slack.

c) Uso

¡Ya está todo listo! Solo queda guardar el archivo y correr el servicio y verificar que cada 60 segundos nos lleguen los tweets correctos.

Para Ubuntu de 64 bits:

Para esto usamos el siguiente comando:

```
sudo /etc/init.d/td-agent restart
```

Cabe aclarar que el último “match” de la configuración (la última imagen) tiene 2 “store”, el primero siendo solo para verificar en consola los tweets que va a enviar a la app de slack. De querer dejar de ver en consola los tweets, solo se debe borrar el @type copy, junto con el primer “store” (que tiene @type stdout).

Para parar el servicio (como antes) usamos:


```
sudo /etc/init.d/td-agent stop
```

Para Ubuntu de 32 bits:

Para esto usamos el siguiente comando:

```
fluentd -c ./fluent/fluent.conf -vv
```

Cabe aclarar que el último “match” de la configuración (la última imagen) tiene 2 “store”, el primero siendo solo para verificar en consola los tweets que va a enviar a la app de slack. De querer dejar de ver en consola los tweets, solo se debe borrar el @type copy, junto con el primer “store” (que tiene @type stdout).

Para parar el servicio directamente podemos terminarlo con Ctrl+C.

5) Problemas encontrados

Nos encontramos con una buena variedad de problemas a la hora de hacer este proyecto.

Vamos a ignorar los previos a la decisión de reducir el alcance del proyecto (principalmente fue falta de conocimiento del lenguaje Ruby, junto con algo de fluentd, para poder desarrollar nuestros propios plugins).

El principal problema (bastante general) fue encontrar y entender los plugins para el proyecto en cuestión.

Esto es de nuevo debido a la falta de conocimiento de Ruby (por lo que si la documentación del plugin no era muy clara o estaba algo desactualizada, nos complicaba mucho el entender cómo usarlo/instalarlo).

Una vez que entendimos que podíamos instalar los plugins siempre con el mismo comando (solo cambiando el argumento del identificador del plugin), se nos facilitó bastante la resolución del proyecto.

Entender como usar la herramienta nos llevó un poco de tiempo, pero no fue lo más complejo.

Un problema que consumió mucho más tiempo del que previmos fue: la transformación que debíamos hacer al “log” que nos daba el plugin de twitter, para pasarla con el formato y los valores correctos al plugin de slack. Esto debiera ser sencillo, pero la falta de conocimiento de la herramienta sumado a la falta de conocimiento de expresion regulares (regex), que se usaban en múltiples plugins, y el desconocimiento de Ruby, nos complicaron mucho.



Además, al probar con varios plugins se nos generó conflictos en un momento por haber instalado versiones viejas de los mismos que no eran compatibles con la versión que usamos de td-agent. Fue sencillo de solucionar, solo desinstalamos las versiones viejas de los plugins (que fueron instaladas en un principio basadas en documentaciones desactualizadas).

Algún plugin también estaba muy desactualizado y no lo pudimos usar con la versión actual de td-agent, en ese caso lo descartamos (desinstalandolo también).

Por otro lado, habíamos hecho todo sobre un Ubuntu de 64 bits (sistema operativo soportado por las aplicación td-agent), pero al querer hacer lo mismo en la notebook con un Ubuntu de 32 bits tuvimos que cambiar la manera de instalar y correr la herramienta.

Como aprendizaje, recomendamos siempre usar plugins certificados. Se pueden ver en la siguiente página con su logo de certificación: <https://www.fluentd.org/plugins/all>

Ejemplo:

Certified	Download	Name	Author
	5263205	rewrite-tag-filter	Kentaro Yoshida
	3484130	record-modifier	Masahiro Nakagawa

Además de usar los plugins que ya son nativos de la herramienta y no hace falta instalar..dichos plugins se pueden encontrar en la documentación oficial.

Por ejemplo en: <https://docs.fluentd.org/output> hay una lista de todos los plugins nativos de Output, de los cuales nosotros usamos “copy” y “stdout”.

6) Fuentes

- Sitio oficial de fluentd: <https://www.fluentd.org/>
- Documentación oficial de fluentd: <https://docs.fluentd.org/>
- Lista de todos los plugins de fluentd: <https://www.fluentd.org/plugins/all>
- Resumen de los principales inputs de data populares de fluentd: <https://www.fluentd.org/datasources>
- Resumen de los principales outputs de data populares de fluentd: <https://www.fluentd.org/dataoutputs>
- Slides que explican funcionamiento con bastante detalle de fluentd: <https://www.fluentd.org/slides>
- Video con explicación muy básica de lo que es fluentd: <https://www.youtube.com/watch?v=bK46tr17nVk>
- Video con explicación básica y con ejemplos de configuración de fluentd: <https://www.youtube.com/watch?v=aeGADcC-hUA>

- Video muy informal (pero ágil y al grano) de cómo crear una app de twitter: https://www.youtube.com/watch?v=M_gGUqhCJoU
- Github del plugin “record-modifier”:
<https://github.com/repeatedly/fluent-plugin-record-modifier>
- Github del plugin “filter-keys”:
https://github.com/banyan/fluent-plugin-filter_keys
- Github del plugin de “slack”:
<https://github.com/sowawa/fluent-plugin-slack>
- Github del plugin de “twitter”:
<https://github.com/y-ken/fluent-plugin-twitter>
- Documentación del api de twitter para obtener los ids de las cuentas que uno sigue:
<https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/api-reference/get-friends-list>