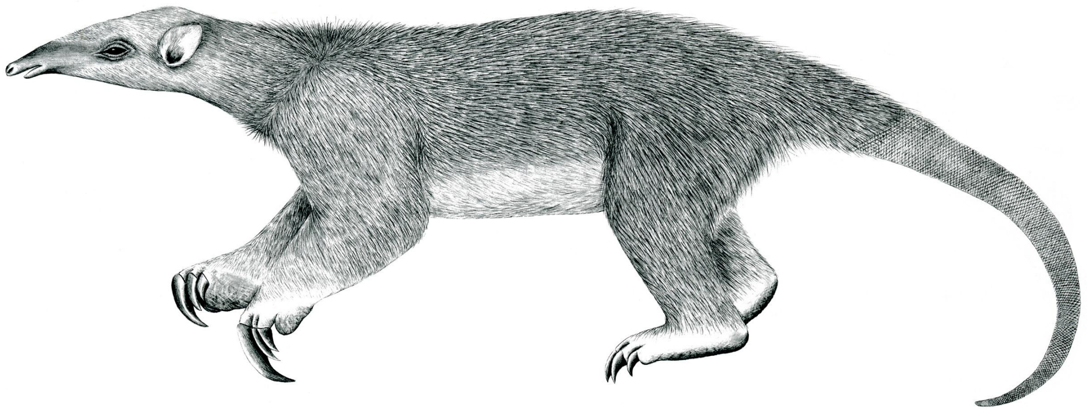
Tranalyzer2

kafkaSink

Apache Kafka



Tranalyzer Development Team

Contents

1 kafkaSink

1.1 Description

The kafkaSink plugin outputs flows to an Apache Kafka event streaming platform.

1.2 Dependencies

1.2.1 External Libraries

This plugin depends on the **librdkafka** library.

		OPT2=1
Ubuntu:	`sudo apt-get install`	`librdkafka-dev`
Arch:	`sudo pacman -S`	`librdkafka`
Gentoo:	`sudo emerge`	`librdkafka`
openSUSE:	`sudo zypper install`	`librdkafka-devel`
Red Hat/Fedora[1]**:**	`sudo dnf install`	`librdkafka-devel`
macOS[2]**:**	`brew install`	`librdkafka`

1.2.2 Core Configuration

This plugin requires the following core configuration:

- *$T2HOME/tranalyzer2/src/tranalyzer.h*:
  - `BLOCK_BUF=0`

1.3 Services Initialization

The kafkaSink plugin requires a ZooKeeper and a Kafka broker service running on `KAFKA_BROKERS` (default address and port are `127.0.0.1:9092`).

```
# Start the ZooKeeper server and send it to the background
$ zookeeper-server-start.sh /etc/kafka/zookeeper.properties &

# Start the Kafka server and send it to the background
$ kafka-server-start.sh /etc/kafka/server.properties &
```

[1]If the `dnf` command could not be found, try with `yum` instead
[2]Brew is a packet manager for macOS that can be found here: https://brew.sh

1.4 Configuration Flags

The following flags can be used to control the output of the plugin:

Name	Default	Description
`KAFKA_BROKERS`	`"127.0.0.1:9092"`	Broker address(es) (comma separated list of host[:port])
`KAFKA_TOPIC`	`"tranalyzer.flows"`	Topic to produce to
`KAFKA_PARTITION`	-1	Target partition: ≥ 0: fixed partition -1: automatic partitioning (unassigned)
`KAFKA_RETRIES`	3	Max. number of retries when message production failed [0 - 255]
`KAFKA_DEBUG`	0	Print debug messages

1.4.1 Environment Variable Configuration Flags

The following configuration flags can also be configured with environment variables (`ENVCNTRL>0`):

- `KAFKA_BROKERS`
- `KAFKA_TOPIC`
- `KAFKA_PARTITION`

1.5 Plugin Report Output

The following information is reported:

- Number of flows discarded

1.6 Example

In this example, the flows will be sent to Kafka to the `tranalyzer.flows` topic. In addition, Tranalyzer information (`[INF]`) and warnings (`[WRN]`) will be sent to the `tranalyzer.out`, while errors (`[ERR]`) will use the `tranalyzer.err` topic.

First, we want to prevent Tranalyzer from coloring the output:

```
$ t2conf tranalyzer2 -D T2_LOG_COLOR=0
```

In this example, only the `basicFlow` and kafkaSink plugins will be used:

```
$ t2build tranalyzer2 basicFlow kafkaSink
```

Now, the fun part! Run Tranalyzer as per usual, but redirect `stdout` and `stderr` to a `kcat`[3] process, which will send the data to Kafka:

[3] `kcat` was formerly known as `kafkacat`

```
$ t2 -r file.pcap \
      1> >(grep -F -e "[INF]" -e "[WRN]" | kcat -P -b 127.0.0.1:9092 -t tranalyzer.out) \
      2> >(kcat -P -b 127.0.0.1:9092 -t tranalyzer.err)
```

The messages can now be consumed with the help of `kafka-console-consumer`.

```
# Consume messages for tranalyzer.flows topic
$ kafka-console-consumer \
      --bootstrap-server localhost:9092 \
      --from-beginning \
      --topic tranalyzer.flows

# Consume messages for tranalyzer.out topic
$ kafka-console-consumer \
      --bootstrap-server localhost:9092 \
      --from-beginning \
      --topic tranalyzer.out

# Consume messages for tranalyzer.err topic
$ kafka-console-consumer \
      --bootstrap-server localhost:9092 \
      --from-beginning \
      --topic tranalyzer.err
```