

Adding a New Telemetry Data Source

Date of publish: 2017-11-06



Legal Notice

© Cloudera Inc. 2019. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

- Prerequisites..... 4**
- Stream Data into CCP..... 4**
 - Install Your New Data Source..... 5
 - Install NiFi..... 5
 - Create a NiFi Flow to Stream Events to CCP..... 5
- Parse the Squid Data Source to CCP..... 13**
 - Parse the Squid Telemetry Event..... 13
 - Create an Index Template..... 20
 - Verify That the Events Are Indexed..... 23
- Add New Data Source to the Metron Dashboard..... 23**
 - Configure a New Data Source Index in the Metron Dashboard..... 23
 - Review the New Data Source Data..... 25

Prerequisites

Before you add a new telemetry device, you must meet both the Cloudera Cybersecurity Platform (CCP) and your user requirements.

- Install HDP and HDF, and then install CCP.
- The following use case assumes the following user requirements:
 - Proxy events from Squid logs must be ingested in real-time.
 - Proxy logs must be parsed into a standardized JSON structure suitable for analysis by Metron.
 - In real-time, the Squid proxy events must be enriched so that the domain names contain the IP information.
 - In real-time, the IP within the proxy event must be checked against for threat intelligence feeds.
 - If there is a threat intelligence hit, an alert must be raised.
 - The SOC analyst must be able to view new telemetry events and alerts from Squid.
 - CCP must profile the Squid data and incorporate statistical features of the profile into the threat triage rules.
 - CCP must be able to deploy a machine learning model that derives additional insights from the stream.
 - CCP must incorporate user's machine learning model into user's threat triage rule along with threat intel, static rules, and statistical rules.
 - Set CCP values.

When you install CCP, you will set up several hosts. You will need the locations of these hosts, along with port numbers, and the Metron version. These values are listed below.

- KAFKA_HOST = The host where a Kafka broker is installed.
- ZOOKEEPER_HOST = The host where a ZooKeeper server is installed.
- PROBE_HOST = The host where your sensor, probes are installed. If don't have any sensors installed, pick the host where a Storm supervisor is running.
- SQUID_HOST = The host where you want to install SQUID. If you don't care, install SQUID on the PROBE_HOST.
- NIFI_HOST = Host where you will install NIFI. This should be the same host on which you installed Squid.
- HOST_WITH_ENRICHMENT_TAG = The host in your inventory hosts file that you put under the group "enrichment."
- SEARCH_HOST = The host where you have Elastic or Solr running. This is the host in your inventory hosts file that you put under the group "search". Pick one of the search hosts.
- SEARCH_HOST_PORT = The port of the search host where indexing is configured (for example, 9300).
- METRON_UI_HOST = The host where your Metron UI web application is running. This is the host in your inventory hosts file that you put under the group "web."
- METRON_VERSION = The release of the Metron binaries you are working with (for example, 0.2.0BETA-RC2).

Stream Data into CCP

The first step in adding a new data source telemetry is to stream all raw events from the telemetry data source into its own Kafka topic.

Install Your New Data Source

Prior to adding a new telemetry data source, you must install it. If you have already installed your data source and have a log source, you can skip this section and move to the next one. All procedures in the Runbook use the Squid telemetry data source.

Procedure

1. Log into the sensor node on which you want to install Squid:

```
ssh $SQUID_HOST
```

2. Install and start Squid:

```
sudo yum install squid  
sudo service squid start
```

Install NiFi

NiFi is built to automate the flow of data between systems. As a result, NiFi works well to collect, ingest, and push data to CCP. If you haven't already done so, install NiFi.



Important:

NiFi cannot be installed on top of HDP, so you must install NiFi manually to use it with CCP.

Create a NiFi Flow to Stream Events to CCP

You can use NiFi to create a data flow to capture events from Squid and push them into Cloudera Cybersecurity Platform (CCP). For this task we will use NiFi to create two processors, one TailFile processor that will ingest Squid events and one PutKafka processor that will stream the information to Metron. When we create and define the PutKafka processor, Kafka will create a topic for the Squid data source. We'll then connect the two processors, generate some data in the Squid log, and watch the data flow from one processor to the other.

Procedure

- 1.



Drag the first icon on the toolbar (the processor icon) to your workspace.
NiFi displays the Add Processor dialog box.

Add Processor

Tag Cloud: **amazon attributes** **avro aws** **consume** **database fetch** **files filesystem** **get hadoop http** **ingest input** **insert json listen** **logs message** **put remote** **restricted source** **split update**

Displaying 188 of 188 Filter


Type	Tags
AttributesToJSON	flowfile, json, attributes
Base64EncodeContent	encode, base64
CompressContent	lzma, decompress, compress, snappy framed,...
ConnectWebSocket	subscribe, consume, listen, WebSocket
ConsumeAMQP	receive, amqp, rabbit, get, consume, message
ConsumeIMAP	imap, Email, Consume, Ingest, Message, Get, I...
ConsumeJMS	jms, receive, get, consume, message
ConsumeKafka	PubSub, Consume, Ingest, Get, Kafka, Ingress,...
ConsumeKafka_0_10	0.10.x, PubSub, Consume, Ingest, Get, Kafka, I...
ConsumeMQTT	MQTT, subscribe, consume, listen, IOT
ConsumePOP3	Email, Consume, Ingest, Message, POP3, Get, ...
ConsumeWindowsEventLog	event, windows, ingest

Selected Processor:
AttributesToJSON

Generates a JSON representation of the input FlowFile Attributes. The resulting JSON can be written to either a new Attribute 'JSONAttributes' or written to the FlowFile as content.

CANCEL ADD

2. Select the TailFile type of processor and click **Add**.
NiFi displays a new TailFile processor.



TailFile
TailFile

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

3. Right-click the processor icon and select **Configure** to display the **Configure Processor** dialog box.
 - a) In the **Settings** tab, change the name to Ingest Squid Events.

Configure Processor

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Name
Ingest Squid Events

Id
13a1a081-015c-1000-7972-7dc6816628b0

Type
TailFile

Penalty Duration 30 sec

Yield Duration 1 sec

Bulletin Level WARN

Enabled

Automatically Terminate Relationships
☐ success
 All FlowFiles are routed to this Relationship.

CANCEL APPLY

- b) In the **Properties** tab, enter the path to the squid access.log file in the **Value** column for the **File(s) to Tail** property.

Configure Processor

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Required field +

Property	Value
Tailing mode	Single file
File(s) to Tail	/usr/log/squid/access.log
Rolling Filename Pattern	No value set
Base directory	No value set
Initial Start Position	Beginning of File
State Location	Local
Recursive lookup	false
Rolling Strategy	Fixed name
Lookup frequency	10 minutes
Maximum age	24 hours

CANCEL APPLY

4. Click **Apply** to save your changes and dismiss the **Configure Processor** dialog box.
5. Add another processor by dragging the **Processor** icon to the main window.
6. Select the **PutKafka** type of processor and click **Add**.
7. Right-click the processor and select **Configure**.
8. In the Settings tab, change the name to Stream to Metron and then select the Automatically Terminate Relationships check boxes for **Failure** and **Success**.

The screenshot shows the 'Configure Processor' dialog box with the 'SETTINGS' tab selected. The dialog has four tabs: SETTINGS, SCHEDULING, PROPERTIES, and COMMENTS. The 'SETTINGS' tab contains the following fields and options:

- Name:** Stream to Metron
- Id:** 13ada490-015c-1000-1805-77a61f75a5ab
- Type:** PutKafka
- Penalty Duration:** 30 sec
- Yield Duration:** 1 sec
- Bulletin Level:** WARN (with a dropdown arrow)
- Enabled:** ☒ Enabled
- Automatically Terminate Relationships:**
 - ☒ failure: Any FlowFile that cannot be sent to Kafka will be routed to this Relationship
 - ☒ success: Any FlowFile that is successfully sent to Kafka will be routed to this Relationship

At the bottom right of the dialog are 'CANCEL' and 'APPLY' buttons.

9. In the Properties tab, set the following three properties:
 - Known Brokers: \$KAFKA_HOST:6667
 - Topic Name: squid
 - Client Name: nifi-squid

Configure Processor

SETTINGS
SCHEDULING
PROPERTIES
COMMENTS

Required field +

Property	Value
Known Brokers	? BKAFKA_HOST:6667
Topic Name	? squid
Partition Strategy	? Round Robin
Partition	? No value set
Kafka Key	? No value set
Delivery Guarantee	? Best Effort
Message Delimiter	? No value set
Max Buffer Size	? 5 MB
Max Record Size	? 1 MB
Communications Timeout	? 30 secs
Batch Size	? 16384
Queue Buffering Max Time	? No value set
Compression Codec	? None
Client Name	? nifi-squid

CANCEL
APPLY

10. Click **Apply** to save your changes and dismiss the **Configure Processor** dialog box.

11. Create a connection by dragging the arrow from the Ingest Squid Events processor to the Stream to Metron processor.

NiFi displays a **Create Connection** dialog box.



The image shows a 'Create Connection' dialog box with two tabs: 'DETAILS' and 'SETTINGS'. The 'SETTINGS' tab is active. It contains two columns of options. The left column has 'From Processor' (Ingest Squid Events, TailFile), 'Within Group' (NiFi Flow), and 'For Relationships' (a checked checkbox labeled 'success'). The right column has 'To Processor' (Stream to Metron, PutKafka) and 'Within Group' (NiFi Flow). At the bottom right are 'CANCEL' and 'ADD' buttons.

From Processor	To Processor
Ingest Squid Events	Stream to Metron
TailFile	PutKafka

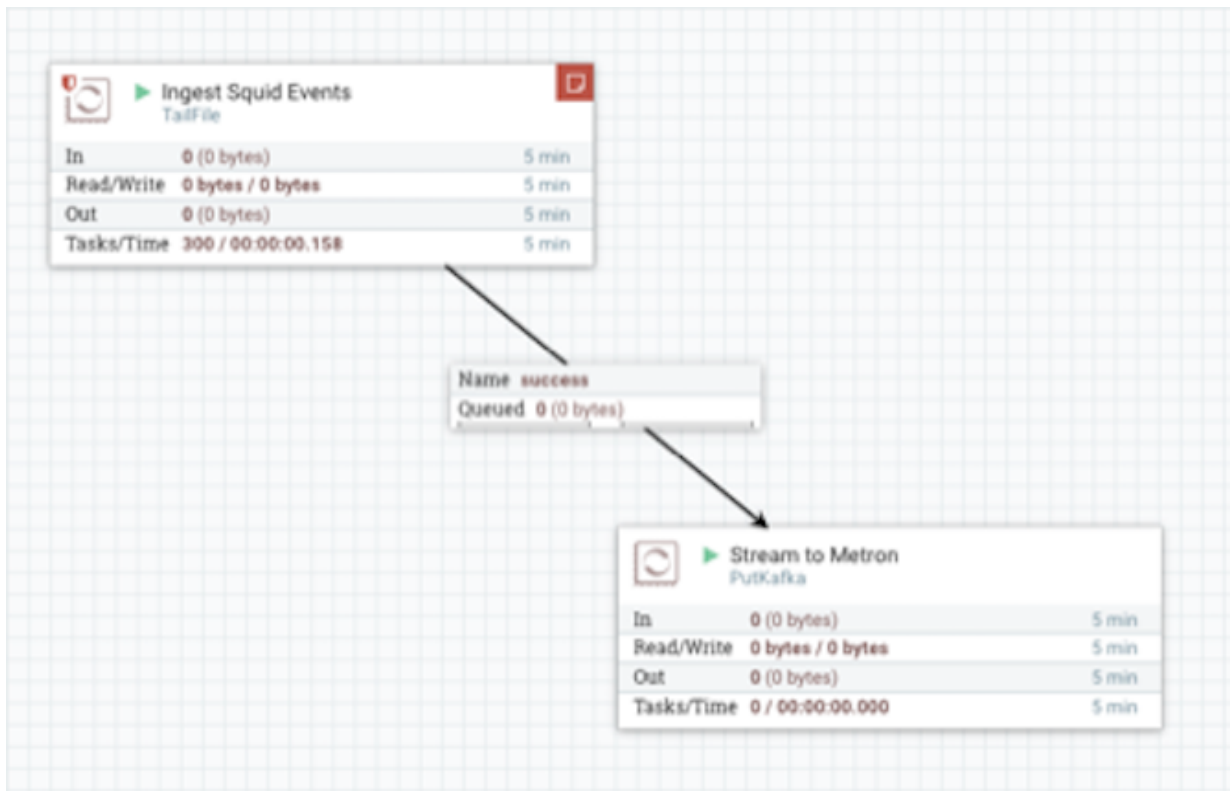
Within Group	Within Group
NiFi Flow	NiFi Flow

For Relationships
☒ success

CANCEL ADD

12. Click **Add** to accept the default settings for the connection.

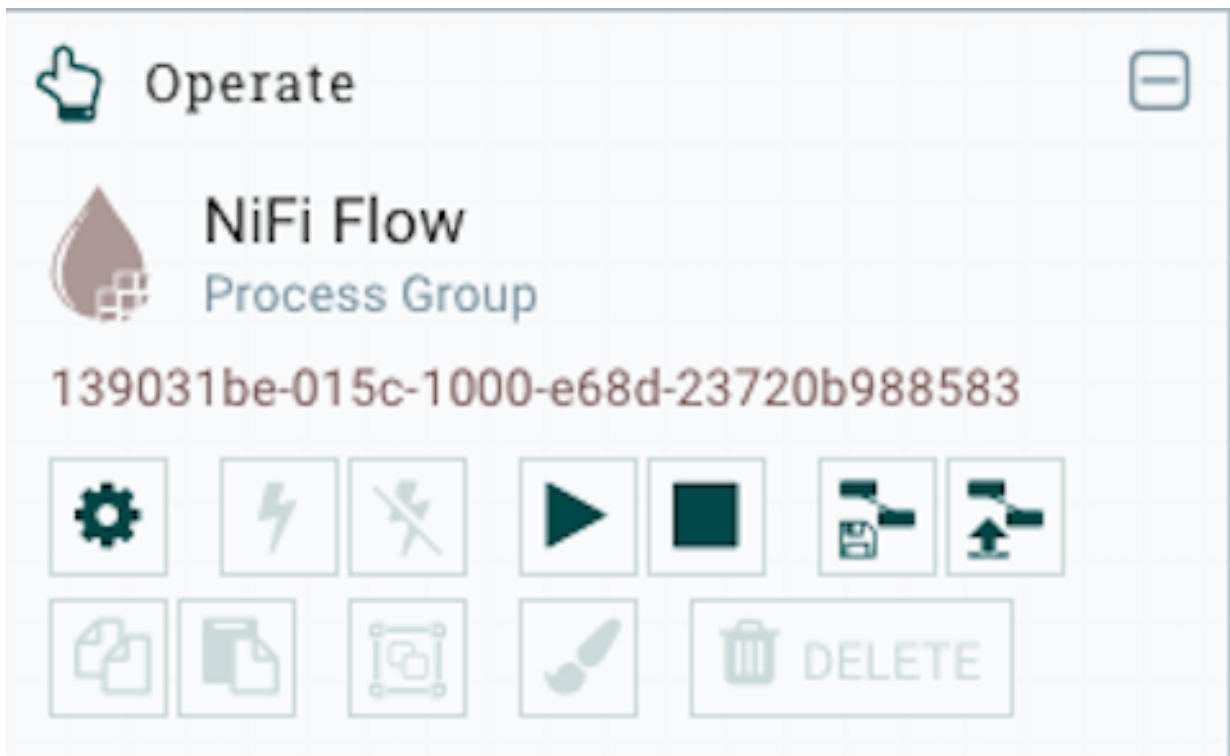
13. Press the Shift key and draw a box around both parsers to select the entire flow.



14.



Click (Start button) in the **Operate** panel.



15. Generate some data using the new data processor client.

a) Use ssh to access the host for the new data source.

- b) With Squid started, look at the different log files that get created:

```
sudo su -
cd /var/log/squid
ls
```

The file you want for Squid is the access.log, but another data source might use a different name.

- c) Generate entries for the log so you can see the format of the entries.

```
squidclient -h 127.0.0.1 "http://www.atmape.ru"
```

You will see the following data in the access.log file.

```
1481143984.330 1111 127.0.0.1 TCP_MISS/301 714 GET http://
www.atmape.ru/ - DIRECT/212.109.217.71 text/html
```

- d) Using the Squid log entries, you can determine the format of the log entries is:

```
timestamp | time elapsed | remotehost | code/status | bytes | method |
URL rfc931 peerstatus/peerhost | type
```

You should see metrics on the processor of data being pushed into Metron.

16. Look at the Storm UI for the parser topology and you should see tuples coming in.

- Navigate to Ambari UI.
- From the **Quick Links** pull-down menu in the upper center of the window, select **Storm UI**.

17. Before leaving this section, run the following commands to fill the access.log with data. You'll need this data when you enrich the telemetry data.

```
squidclient -h 127.0.0.1 "http://www.aliexpress.com/af/shoes.html?
ltype=wholesale&d=y&origin=n&isViewCP=y&catId=0&initiative_id=SB_20160622082445&Search"
squidclient -h 127.0.0.1 "http://www.help.1and1.co.uk/domains-c40986/
transfer-domains-c79878"
squidclient -h 127.0.0.1 "http://www.pravda.ru/science/"
squidclient -h 127.0.0.1 "http://
www.brightsideofthesun.com/2016/6/25/12027078/anatomy-of-a-deal-phoenix-
suns-pick-bender-chriss"
squidclient -h 127.0.0.1 "https://www.microsoftstore.com/store/msusa/
en_US/pdp/Microsoft-Band-2-Charging-Stand/productID.329506400"
squidclient -h 127.0.0.1 "https://tfl.gov.uk/plan-a-journey/"
squidclient -h 127.0.0.1 "https://www.facebook.com/Africa-Bike-
Week-1550200608567001/"
squidclient -h 127.0.0.1 "http://www.ebay.com/itm/02-Infiniti-QX4-Rear-
spoiler-Air-deflector-Nissan-Pathfinder-/172240020293?fits=Make%3AInfiniti
%7CModel%3AQX4&hash=item281a4e2345:g:iMkAAOSwoBtW4Iwx&vxp=mtr"
squidclient -h 127.0.0.1 "http://www.recruit.jp/corporate/english/company/
index.html"
squidclient -h 127.0.0.1 "http://www.lada.ru/en/cars/4x4/3dv/about.html"
squidclient -h 127.0.0.1 "http://www.help.1and1.co.uk/domains-c40986/
transfer-domains-c79878"
squidclient -h 127.0.0.1 "http://www.aliexpress.com/af/shoes.html?
ltype=wholesale&d=y&origin=n&isViewCP=y&catId=0&initiative_id=SB_20160622082445&Search"
```

What to do next

For more information about creating a NiFi data flow, see the NiFi documentation.

Parse the Squid Data Source to CCP


Parsers transform raw data (textual or raw bytes) into JSON messages suitable for downstream enrichment and indexing by Cloudera Cybersecurity Platform (CCP). There is one parser for each data source and the information is piped to the Enrichment/Threat Intelligence topology.

Parse the Squid Telemetry Event

Parsers transform raw data into JSON messages suitable for downstream enrichment and indexing by Cloudera Cybersecurity Platform (CCP). There is one parser for each data source and CCP pipes the information to the Enrichment/Threat Intelligence topology. You can transform the field output in the JSON messages into information and formats that make the output more useful. For example, you can change the timestamp field output from GMT to your timezone.

The following procedures use the Management UI whenever possible. If you would like to see these steps performed using the CLI, see the CCP Administration Guide.

Procedure

1. Launch the Management UI:
 - a) From the Ambari Dashboard panel, click **Metron**.
Make sure you have the **Summary** tab selected.
 - b) Select **Management UI** from **Quick Links**.
The **Management UI** tool should display in a separate browser tab.
2. Click **Sensors** on the left side of the window, under **Operations**.
3.  (add button) in the lower right corner of the screen.
The Management UI displays a panel used to create the new sensor.

4. Enter the following information:

```
Name: mysquid
Kafka Topic: mysquid
Parser Type: Grok
```




Important: You must include timestamp in the Grok expression to ensure that the system uses the event time rather than the system time. You will not be able to save the Grok statement if the **TIMESTAMP** field is empty.

If a Kafka topic already exists for the sensor name, the UI displays a message similar to **Kafka Topic Exists. Emitting** and displays the name of the **Parser Type**. You can skip to the next section, Verify Events are Indexed.

5. Create a Grok statement for the new parser:

a)



In the Grok Statement box, click the  (expand window) to display the **Grok Validator** panel.

- b) Choose or enter a name for the Grok statement in the **PATTERN LABEL** field.
For our example, we chose MYSQUID.

- c) Enter a sample log entry for the data source.

To create sample log entries, see Step 15 in *Create a NiFi Flow to Stream Events to CCP*.

Grok Validator

SAMPLE (1 of 1)

1496873870.043 728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/-DIRECT/104.16.27.236 text/html

PATTERN LABEL

SQUID

STATEMENT

1 SQUID

TEST SAVE

PREVIEW

- d) Refer to the format of the log entries you determined in Step 11d in *Create a NiFi Flow to Stream Events to CCP*.

For example, the log entry format for Squid is:

```
timestamp | time elapsed | remotehost | code/status | bytes | method |
URL rfc931 peerstatus/peerhost | type
```

- e) In the **STATEMENT** field, enter the first element in the sensor log file.

For Squid, the first element in the sensor log file is timestamp which is a number, so we enter `%{NUMBER:timestamp}`.



Note: The Management UI automatically completes partial words in your Grok statement as you enter them.

Grok Validator

SAMPLE (1 of 1)

```
1496873870.043 728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/-DIRECT/104.16.27.236 text/html
```

PATTERN LABEL

SQUID

STATEMENT

```
1 SQUID %{NUMBER:timestamp}
```

TEST SAVE

- f) Click **TEST**.

If the validator finds an error, it displays the error information. If the validation succeeds, it displays the valid mapping in the **PREVIEW** field.

Because you entered the timestamp element, the validator parses the timestamp correctly and leaves the rest of the information as random data.

Grok Validator ✕

SAMPLE (1 of 1)

```
1496873870.043 728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/
-DIRECT/104.16.27.236 text/html
```

PATTERN LABEL

SQUID_DELIMITED

STATEMENT

```
1 SQUID_DELIMITED %{NUMBER:timestamp}
```

TEST SAVE

PREVIEW

original_string	1496873870.043 728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/ - DIRECT/104.16.27.236 text/html
timestamp	1496873870.043

- g) Click **Save**.
 - h) Verify that the sensor name and topic name are "mysquid" with NO extra spaces or special characters.
6. Click **Save** on the mysquid sensor.
- The mysquid sensor appears in the Management UI. You might need to refresh your screen to see the new sensor.
7. Click the pencil icon to edit the mysquid sensor.
8. Scroll down to the **Parser Config** section.
9. In the first open field, indicated by **enter field**, enter timestampField.

PARSER CONFIG

grokPath	
/apps/metron/patterns/mysquid	—
patternLabel	
MYSQUID	—
timestampField	
timestamp	+

SAVE CANCEL

10. In next open field, enter timestamp.

11. Click **Save**.

12. Continue to build and test the Grok statement until you have entries for each element in the log entry.
For example, for Squid, the second element is elapsed, so we enter `%{INT:elapsed}` and click **TEST** again.



Grok Validator

SAMPLE (1 of 1)

```
1496873870.043 728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/
- DIRECT/104.16.27.236 text/html
```

PATTERN LABEL

SQUID_DELIMITED

STATEMENT

```
1 SQUID_DELIMITED %{NUMBER:timestamp} %{INT:elapsed}
```

TEST SAVE

PREVIEW

elapsed	85547
original_string	1496873870.043 728 127.0.0.1 TCP_MISS/200 85547 GET https://tfl.gov.uk/plan-a-journey/ - DIRECT/104.16.27.236 text/html
timestamp	200

13. When your Grok statement is complete and valid, click **SAVE** to save the Grok statement for the sensor.

14. Click **SAVE** to save the sensor information and add it to the list of Sensors.

This new data source processor topology ingests from the \$Kafka topic and then parses the event with the CCP Grok framework using the Grok pattern. The result is a standard JSON Metron structure that then is added to the "enrichment" Kafka topic for further processing.

15. Test that a Kafka topic has been created for the Squid parser:

a) Navigate to the following directory:

```
/usr/hdp/current/kafka-broker/bin
```

b) List all of the Kafka topics:

```
./kafka-topics.sh --zookeeper localhost:2181 --list
```

You should see the following list of Kafka topics:

- bro
- enrichments
- ubdexubg
- snort
- mysquid

Create an Index Template

To work with a new data source data in the Metron dashboard, you need to ensure that the data is landing in the search index (Elasticsearch) with the correct data types. You can achieve this by defining an index template. The index template specifies how to interpret the metron events and how to index strings using either a keyword or full text search.

Procedure

1. Launch the Metron dashboard in the browser.
2. Select **Dev Tools** from the left hand side of the Kibana page.

The Dev Tools console is an easy way to interact with the index REST api. If the **Welcome** window appears, click the **Get to work** button.

3. Paste the following command into the left side of **Dev Tools** window:

```
PUT _template/mysquid
{
  "template": "mysquid_index*",
  "settings": {},
  "mappings": {
    "mysquid_doc": {
      "dynamic_templates": [
        {
          "geo_location_point": {
            "match": "enrichments:geo:*:location_point",
            "match_mapping_type": "*",
            "mapping": {
              "type": "geo_point"
            }
          }
        }
      ],
      {
        "geo_country": {
          "match": "enrichments:geo:*:country",
          "match_mapping_type": "*",
          "mapping": {
            "type": "keyword"
          }
        }
      }
    },
    {
      "geo_city": {
        "match": "enrichments:geo:*:city",
        "match_mapping_type": "*",
        "mapping": {
          "type": "keyword"
        }
      }
    }
  ]
}
```

```

    }
  },
  {
    "geo_location_id": {
      "match": "enrichments:geo*:locID",
      "match_mapping_type": "*",
      "mapping": {
        "type": "keyword"
      }
    }
  },
  {
    "geo_dma_code": {
      "match": "enrichments:geo*:dmaCode",
      "match_mapping_type": "*",
      "mapping": {
        "type": "keyword"
      }
    }
  },
  {
    "geo_postal_code": {
      "match": "enrichments:geo*:postalCode",
      "match_mapping_type": "*",
      "mapping": {
        "type": "keyword"
      }
    }
  },
  {
    "geo_latitude": {
      "match": "enrichments:geo*:latitude",
      "match_mapping_type": "*",
      "mapping": {
        "type": "float"
      }
    }
  },
  {
    "geo_longitude": {
      "match": "enrichments:geo*:longitude",
      "match_mapping_type": "*",
      "mapping": {
        "type": "float"
      }
    }
  },
  {
    "timestamps": {
      "match": "*:ts",
      "match_mapping_type": "*",
      "mapping": {
        "type": "date",
        "format": "epoch_millis"
      }
    }
  },
  {
    "threat_triage_score": {
      "mapping": {
        "type": "float"
      },
      "match": "threat:trriage:*score",

```

```

        "match_mapping_type": "*"
      }
    },
    {
      "threat_triage_reason": {
        "mapping": {
          "type": "text",
          "fielddata": "true"
        },
        "match": "threat:triage:rules*:reason",
        "match_mapping_type": "*"
      }
    }
  ],
  "properties": {
    "action": {
      "type": "keyword"
    },
    "bytes": {
      "type": "long"
    },
    "code": {
      "type": "long"
    },
    "domain_without_subdomains": {
      "type": "keyword"
    },
    "elapsed": {
      "type": "long"
    },
    "full_hostname": {
      "type": "keyword"
    },
    "guid": {
      "type": "keyword"
    },
    "ip_dst_addr": {
      "type": "ip"
    },
    "ip_src_addr": {
      "type": "ip"
    },
    "is_alert": {
      "type": "keyword"
    },
    "is_potential_typosquat": {
      "type": "boolean"
    },
    "method": {
      "type": "keyword"
    },
    "original_text": {
      "type": "text"
    },
    "source:type": {
      "type": "keyword"
    },
    "timestamp": {
      "type": "date",
      "format": "epoch_millis"
    },
    "url": {
      "type": "keyword"
    }
  },

```

```
    "alert": {  
      "type": "nested"  
    }  
  }  
}  
}  
}
```

4. Press the green play button.

The result on the right hand side of the screen will display "acknowledged" : true.

Verify That the Events Are Indexed

After you finish adding your new data source, you should verify that the data source events are indexed and the output matches any Stellar transformation functions you used.

Procedure

From the Alerts UI, search the source:type filter for squid messages.

By convention, the index where the new messages are indexed is called squid_index_[timestamp] and the document type is squid_doc.

Add New Data Source to the Metron Dashboard

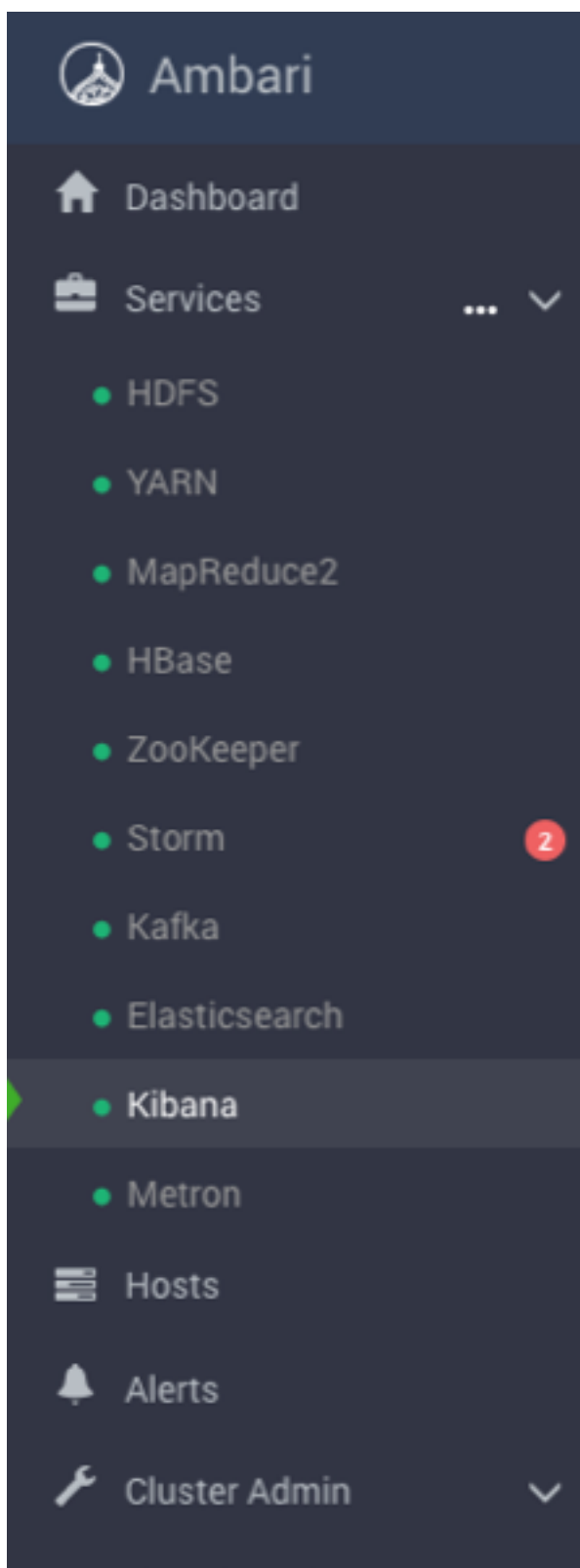
After a new data telemetry source has been added to Cloudera Cybersecurity Platform (CCP), you must add it to the Metron dashboard before you can create queries and filters for it and add telemetry panels displaying its data.

Configure a New Data Source Index in the Metron Dashboard

Now that you have an index for the new data source with all of the right data types, you need to tell the Metron dashboard about this index.

Procedure

1. Launch the Metron dashboard if you have not already done so:
 - a) From Ambari, click Kibana in the list of quick tasks.




- b) Select **Metron UI** from the Quick Links menu in the top center of the window.
2. Click the **Settings** tab on the Metron dashboard.
 3. Make sure you have the **Indices** tab selected, then click **+Add New**.

Kibana displays the **Configure an index pattern** window. Use the index pattern window to identify your telemetry source.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

4. In the **Index name or pattern** field, enter the name of the index pattern of your data telemetry source. In most cases the name of the index pattern will match the sensor name. For example, the 'bro' sensor has an index pattern of 'bro-*'.
5. If your data telemetry source does not contain time-based events, clear the **Index contains time-based events** check box.
If your data telemetry source does contain time-based events, leave the check box as is. Most of your data telemetry sources will contain time-based events.
6. Click **Create** to add the index pattern for your new data telemetry source.

If you would like this new index pattern to be the default, click the Green Star icon ().

Review the New Data Source Data

Now that the Metron dashboard is aware of the new data source index, you can look at the data.

Procedure

1. Display the Metron dashboard.
2. Click on the **Discover** tab and then choose the newly created data source index pattern.
3. Click any of the fields in the left column to see a representation of the variety of data for that specific field.
4. Click the Right Facing Arrow icon next to a specific record in the center of the window (the **Document** table) to expand the record and display the available data.

