DEPLOYMENT GUIDE

# Elastic Stack + Infoblox B1TD & NIOS DNSTAP Logging Integration

July 2023

## Table of Contents

# Introduction

Elastic Stack, formerly known as the ELK Stack, is a popular suite of tools that provides advanced logging, storing, searching and visualization functionality to data of many types from any source. Elasticsearch, Logstash, Kibana, and newcomer Beats work together to make up the core products of Elastic Stack. Elasticsearch handles search and storage of data, Logstash is the pipeline for retrieving data to send to Elasticsearch, and Kibana provides the web browser user interface used to visualize and query this data. Elastic Stack is available as a free, open source local download, but it also provides a paid-for cloud solution. We will be working with the open source version.

This deployment guide is two-fold. As well as providing a detailed introduction into Elastic, it contains deployment guides for integrating both BloxOne Threat Defense data and NIOS dnstap logging data.

Drastically enhance the ability to analyze your network by integrating Elastic Stack's powerful data exploration tools with Infoblox's extensive security and query/response data.

# Requirements

## For Integrating BloxOne Threat Defense Data

The following items are required to incorporate Infoblox BloxOne Threat Defense DNS security data into Elastic Stack:

- Access to an Infoblox BloxOne Threat Defense subscription
- Access to an Elastic Stack instance
  - Composed of Elasticsearch, Logstash and Kibana

## For Integrating NIOS dnstap Logs

The following items are required to incorporate NIOS dnstap for high-performance query logging into Elastic Stack:

- A NIOS Grid with dnstap enabled
- Access to an Elastic Stack instance
  - Composed of Elasticsearch, Logstash and Kibana
- A dnstap receiver to read and translate dnstap logs from NIOS

# Tested Hardware & Software

- Windows 10 VM
- Ubuntu 18.04 VM
    - Elastic Stack version 8.3.2 installed
        - Elastic Stack composed of Elasticsearch, Logstash and Kibana
    - Python3 installed
    - dnstap-receiver module for Python installed
- IB-FLEX NIOS VM with DNS Cache Acceleration and dnstap enabled
    - Running NIOS 8.5.2

# Install Elastic Stack

There are several ways to install Elastic Stack. You may prefer different package formats or operating systems depending on your needs and preferences. Find more information and further links for installing each Elastic Stack component at https://www.elastic.co/guide/en/elastic-stack/current/installing-elastic-stack.html.

A comprehensive one-all guide for installing and configuring Elastic Stack and its dependencies on Ubuntu 18.04/20.04 can be found at https://phoenixnap.com/kb/how-to-install-elk-stack-on-ubuntu.

# Deployment Instructions: B1TD

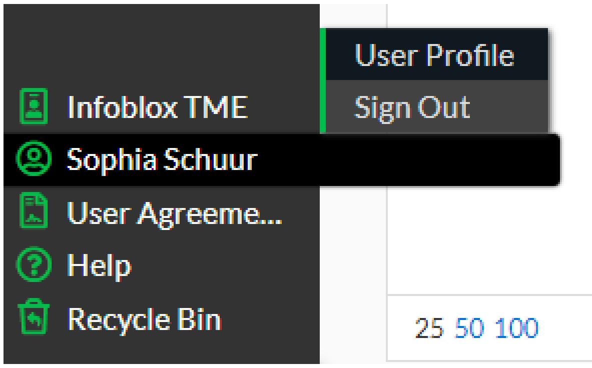
The following instructions provide an intro into the Elastic Stack as well as the steps required to integrate BloxOne Threat Defense DNS Security data with Elastic.

## CSP API Key Retrieval

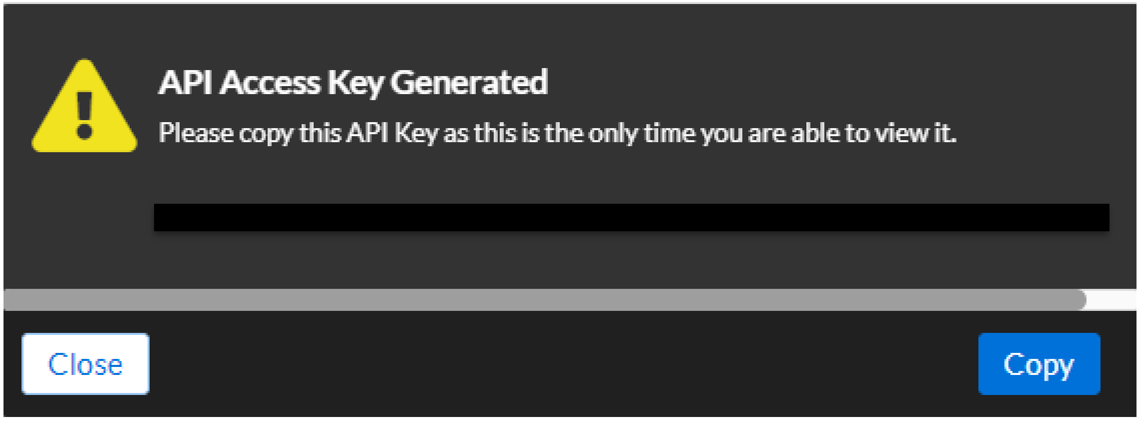
You will need a BloxOne Threat Defense API key to pull the DNS data. You can access this key through the Cloud Services Portal (CSP). API keys are unique identifiers found in many applications to both identify the application making the API calls and verify the application making the calls has access to do so.

To access your API key:

1. Log into the CSP at https://csp.infoblox.com.
2. Upon logging in, hover over your username in the bottom-left corner of the CSP and select **User Profile**.



3. Click the **User API Keys** tab. Click **Create**.
    a. Give the key a meaningful name.
    b. Set an expiration date. Elastic will be constantly polling with this key, so setting a date far into the future is recommended.
    c. Click **Save & Close**.
4. A popup will appear. Click **Copy** to copy your API key to your clipboard. Copy it somewhere you can easily access and copy from later, such as Notepad. This will be the key you copy into the Python script later.


API Access Key Generated

Please copy this API Key as this is the only time you are able to view it.

[redacted]

## Python Configuration

We will be using a Python script to gather the most recent ten minutes of DNS event data from Infoblox's BloxOne Threat Defense REST API and write it into `json` log files. To ensure the data is always recent and updated, we will tell Ubuntu to run this script every ten minutes. Later we will configure Logstash to read the `json` and send it to Kibana to be visualized.

Let's create the Python script. You can save this script anywhere easily accessible to you, such as Documents or the Desktop. *Note: For this demo we will be saving it in* *`/home/<username>/dataconnector`*.

1. Access the machine where your Logstash instance is installed. *Note: For this demo Elastic Stack was installed on Ubuntu 18.04.*

2. Open a terminal.

3. **Python3** must be installed for the script to work properly. If it is not already installed, install it with:

   ```
   sudo apt install python3.8
   ```

4. Navigate to `/home/<username>/dataconnector`:

   ```
   cd /home/infoblox/dataconnector
   ```

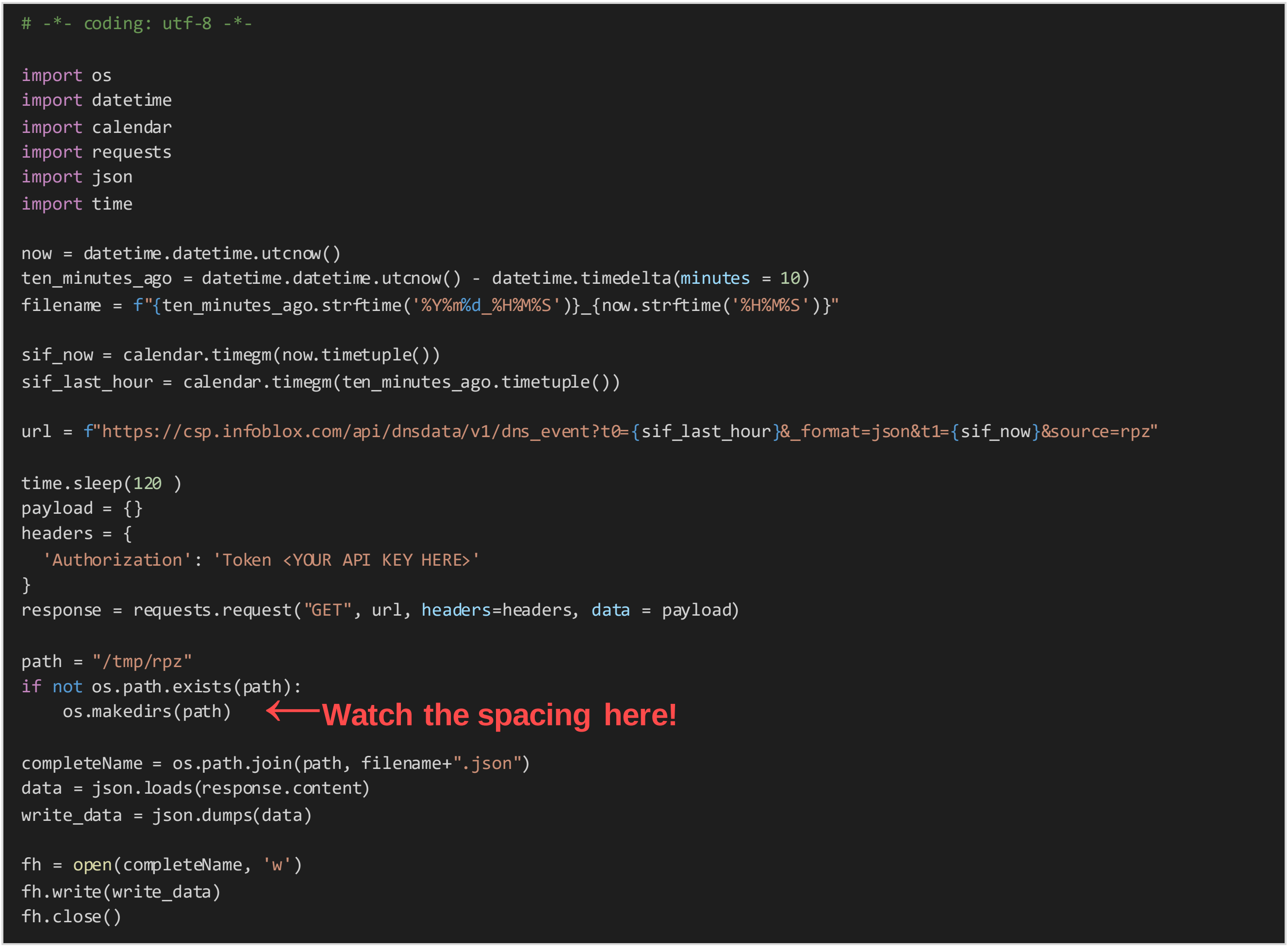
5. Create a new Python file:

   ```
   touch cspscript.py
   ```

6. Open the file with `gedit` for editing:

   ```
   gedit cspscript.py
   ```

7. Copy and paste the following into the file. **Careful to note Python's spacing and tabbing syntax**. **Indent nested Python newlines with four spaces**. Replace the text **<YOUR API KEY HERE>** with the BloxOne TD API key acquired in the CSP API Key Retrieval section of this document. Save and close the file when finished. To ensure your formatting is correct, you can also download the script here on InfobloxOpen's Github repo.



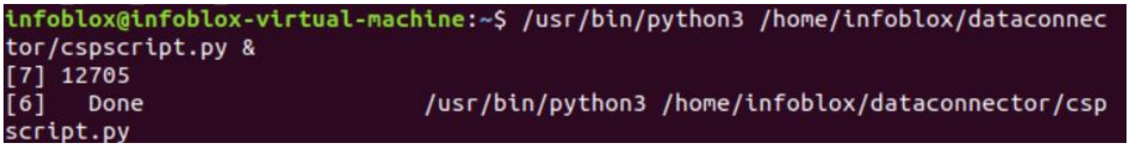
```
# -*- coding: utf-8 -*-

import os
import datetime
import calendar
import requests
import json
import time

now = datetime.datetime.utcnow()
ten_minutes_ago = datetime.datetime.utcnow() - datetime.timedelta(minutes = 10)
filename = f"{ten_minutes_ago.strftime('%Y%m%d_%H%M%S')}_{now.strftime('%H%M%S')}"

sif_now = calendar.timegm(now.timetuple())
sif_last_hour = calendar.timegm(ten_minutes_ago.timetuple())

url = f"https://csp.infoblox.com/api/dnsdata/v1/dns_event?t0={sif_last_hour}&_format=json&t1={sif_now}&source=rpz"

time.sleep(120 )
payload = {}
headers = {
  'Authorization': 'Token <YOUR API KEY HERE>'
}
response = requests.request("GET", url, headers=headers, data = payload)

path = "/tmp/rpz"
if not os.path.exists(path):
    os.makedirs(path)

completeName = os.path.join(path, filename+".json")
data = json.loads(response.content)
write_data = json.dumps(data)

fh = open(completeName, 'w')
fh.write(write_data)
fh.close()
```

⟵Watch the spacing here!

Let's make sure the script is working. Run:

```
/usr/bin/python3 /home/<username>/dataconnector/cspscript.py &
```

You should see an output like this:



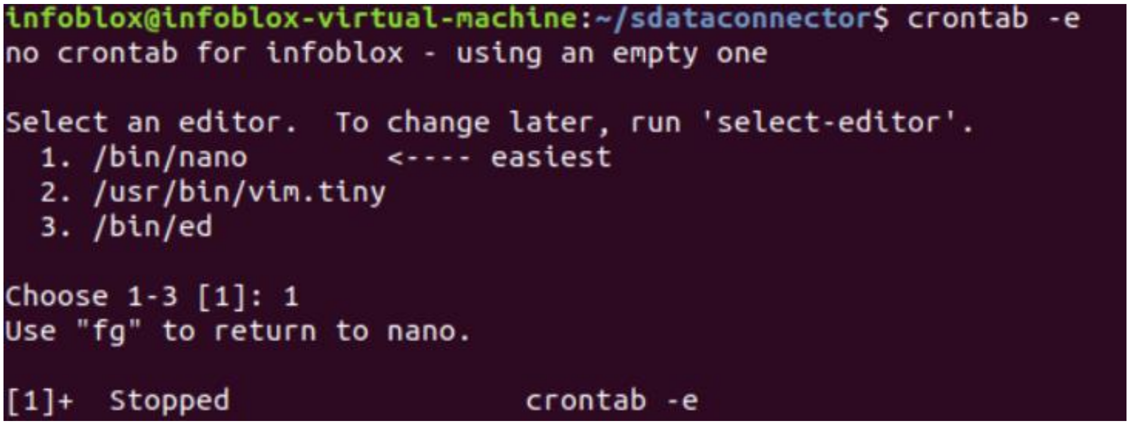
```
infoblox@infoblox-virtual-machine:~$ /usr/bin/python3 /home/infoblox/dataconnec
tor/cspscript.py &
[7] 12705
[6]   Done                    /usr/bin/python3 /home/infoblox/dataconnector/csp
script.py
```

The software utility cron is a time-based job scheduler in Unix-like operating systems. Let's configure cron to run the Python script every ten minutes.

1. Open a terminal.
2. Open a crontab file for editing:

```
crontab -e
```

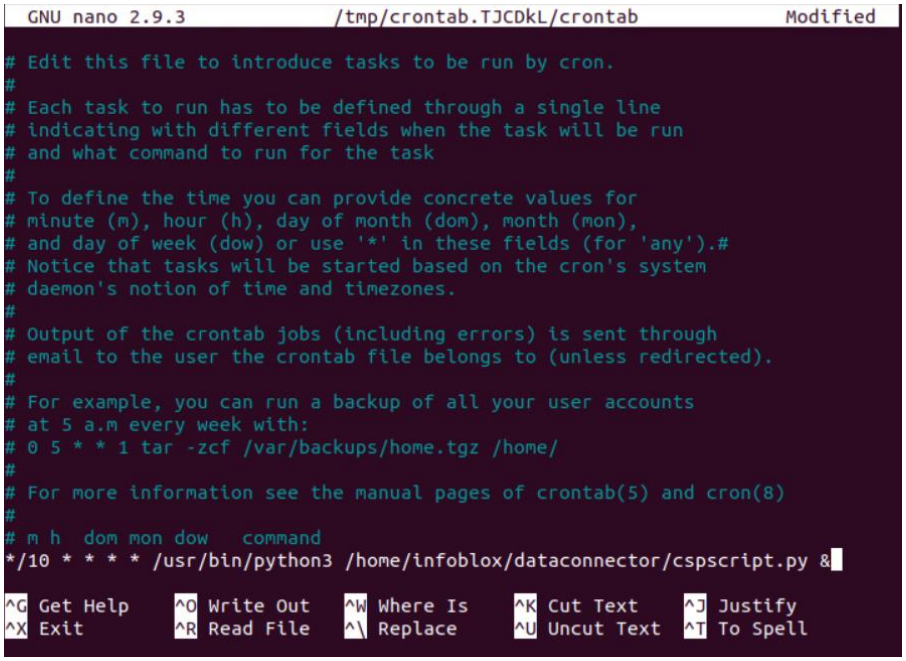
3. Choose your preferred editor if prompted.



```
infoblox@infoblox-virtual-machine:~/sdataconnector$ crontab -e
no crontab for infoblox - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.tiny
  3. /bin/ed

Choose 1-3 [1]: 1
Use "fg" to return to nano.

[1]+  Stopped                 crontab -e
```

4. Insert the following line into the file as shown below. Save and exit the editor.

```
*/10 * * * * /usr/bin/python3 /home/<username>/dataconnector/cspscript.py &
```



```
  GNU nano 2.9.3              /tmp/crontab.TJCDkL/crontab              Modified

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
*/10 * * * * /usr/bin/python3 /home/infoblox/dataconnector/cspscript.py &
```

## Logstash Configuration for TD

Logstash is a highly customizable part of Elastic Stack that retrieves data. It can be configured to collect data from many different sources, such as log files, REST API requests, and more, to be sent to Elasticsearch and later visualized in Kibana. Hundreds of plugins are available to expand its functionality, and many are included with the software at installation. We will be using the input plugin **file** to read the `json` logs generated by the Python script. A complete list of available plugins and links to their documentation can be found at https://www.elastic.co/support/matrix#matrix_logstash_plugins.

Logstash configuration is governed by special configuration files. Where to retrieve data, how to filter it, and where to output it are configured by these files. For this demo, Elastic Stack was installed on Ubuntu 18.04 via `apt-get`, so these files are set by default to live in the `/etc/logstash/conf.d` directory. Your directories may be different depending on how Elastic Stack was installed. More information about the Logstash directory layout can be found at https://www.elastic.co/guide/en/logstash/current/dir-layout.html.

Let's configure Logstash to grab the DNS security data found in the `json` files generated by the Python script.

1. Access the machine where your Logstash instance is installed. *Note: For this demo Elastic Stack was installed on Ubuntu 18.04.*

2. Open a terminal.

3. Navigate to where your Logstash configuration (`.conf`) files are located. In this demonstrative environment, these files are located in `/etc/logstash/conf.d`. Input the following command to navigate to the correct directory:

   ```
   cd /etc/logstash/conf.d
   ```

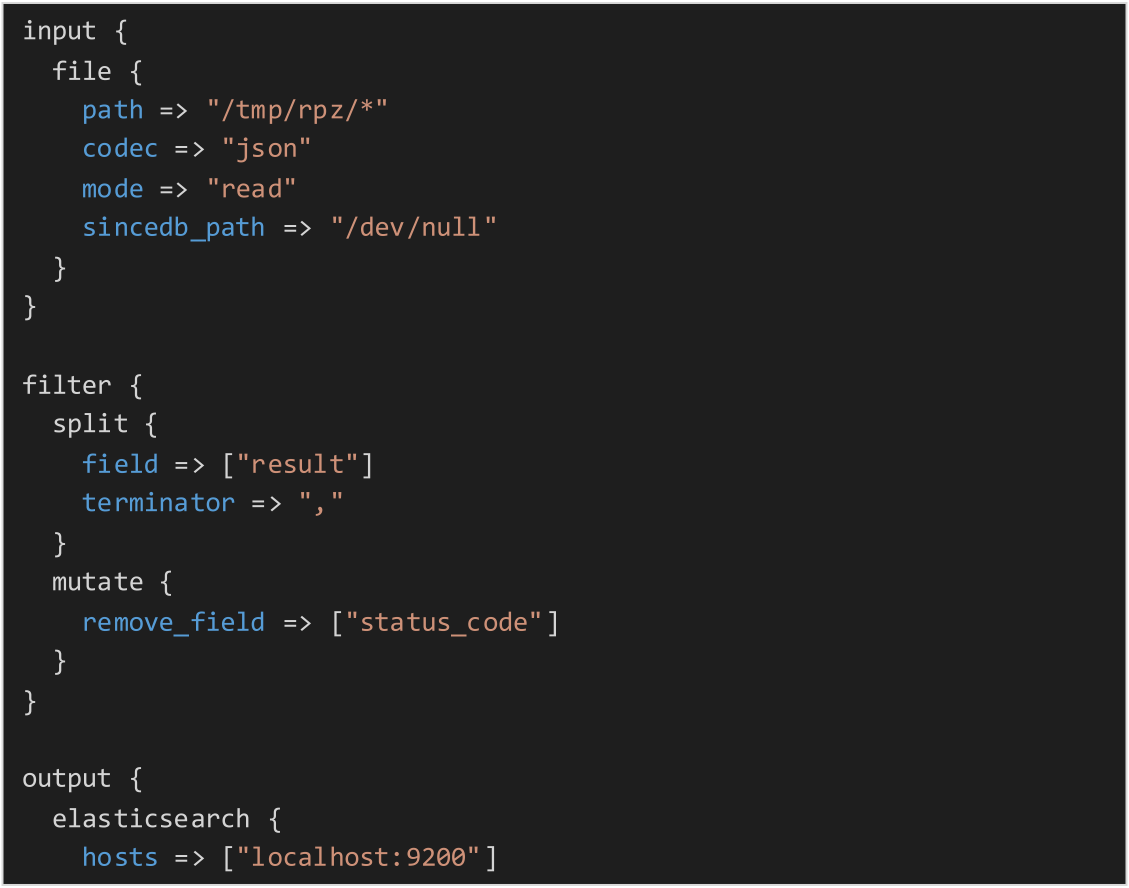
4. Create a new file called `csp-dns-events.conf`:

   ```
   sudo touch csp-dns-events.conf
   ```

5. Open the file with `gedit` for editing:

   ```
   sudo gedit csp-dns-events.conf
   ```

6. Copy and paste the following into the file. Save and close the file when finished.



```
input {
  file {
    path => "/tmp/rpz/*"
    codec => "json"
    mode => "read"
    sincedb_path => "/dev/null"
  }
}

filter {
  split {
    field => ["result"]
    terminator => ","
  }
  mutate {
    remove_field => ["status_code"]
  }
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
```

- Let's get a breakdown of what is happening in this code.

   a. **`Input`**: Here is where we read the `json` files created by the Python script. We use the input plugin [`file`](#).

   b. **`Filter`**: This is where we [`split`](#) every record found in the `json` files into individual hits in Kibana. By doing so, we can directly search and organize by any field returned by the GET request in the Python script. The returned body of the request is placed in one field called `result` with each record terminated by a comma, so we tell that to Logstash. Using the [`mutate`](#) plugin we remove the extra `status_code` field, since it creates unnecessary clunky data.

   c. **`Output`**: Send the data to Elasticsearch and give this index a name. This is the name that will appear in Kibana when creating a new index.

Logstash config files follow a specific schema. More information on the structure of config files can be found at https://www.elastic.co/guide/en/logstash/current/configuration-file-structure.html

7. Navigate to your home directory for Logstash. For this demo, this is `/usr/share/logstash/`. Input the following command to navigate to the correct directory:

```
cd /usr/share/logstash
```

8. Run Logstash with your new configuration:

```
sudo bin/logstash -f /etc/logstash/conf.d/csp-dns-events.conf
```

Allow several minutes of processing. The console will inform you if there are any syntax errors with your config file.

- Alternatively, you can simply restart the Logstash service, but the console will not warn you of any errors with your config file:

```
sudo systemctl restart logstash
```

## Kibana Data Discovery

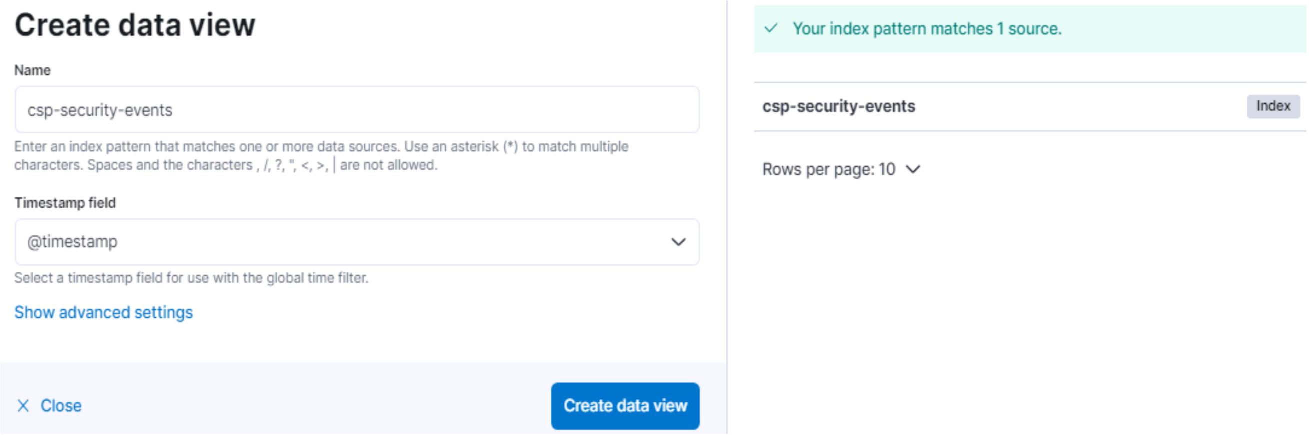
Kibana is the visualization part of the Elastic Stack. It provides a web-based user interface for viewing and charting the data stored in Elasticsearch. Using Data Views (formerly known as Index Patterns), we can map Kibana with the data that our Logstash configuration is outputting to Elasticsearch.

1. Access your Kibana instance. *Note: If desired, you must configure Kibana to allow remote access, such as from a secondary Windows machine. Find instructions [here](#).*

2. Navigate to **Stack Management** → **Data Views**. Your configuration will appear here as an available index pattern. Click **Create data view**.

    a. Name the index pattern **csp-dns-events**.

    b. In the Timestamp field, select **@timestamp**. Click **Create data view**.



Create data view

Name

csp-security-events

Enter an index pattern that matches one or more data sources. Use an asterisk (*) to match multiple characters. Spaces and the characters , /, ?, ", <, >, | are not allowed.

Timestamp field

@timestamp

Select a timestamp field for use with the global time filter.

Your index pattern matches 1 source.

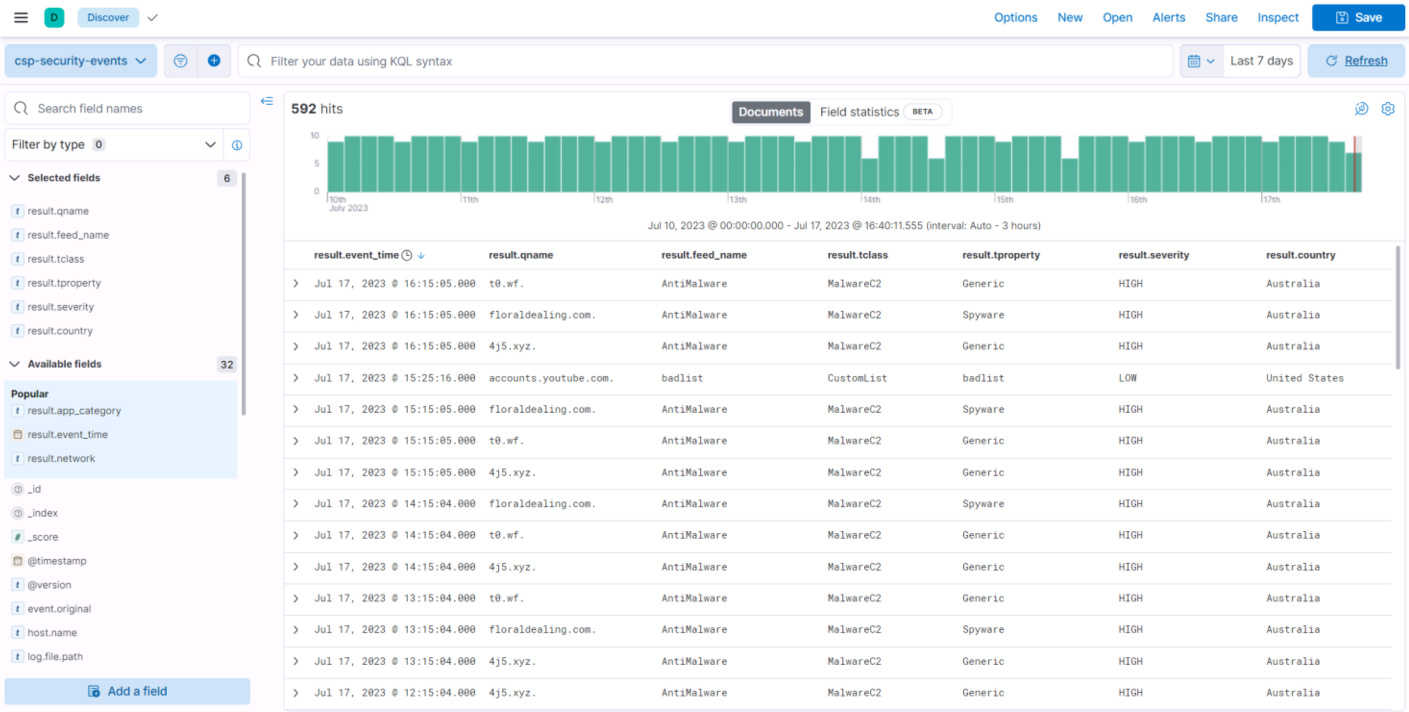
| | |
|---|---|
| csp-security-events | Index |

Rows per page: 10

3. Click the ☰ menu icon in the topbar. Select **Discover**.

4. Here you will see all the DNS security records retrieved from the CSP.

   You can perform extensive searching, displaying, and filtering here. Click on some **Available fields** in the left panel to view field totals for this dataset and organize your hits.

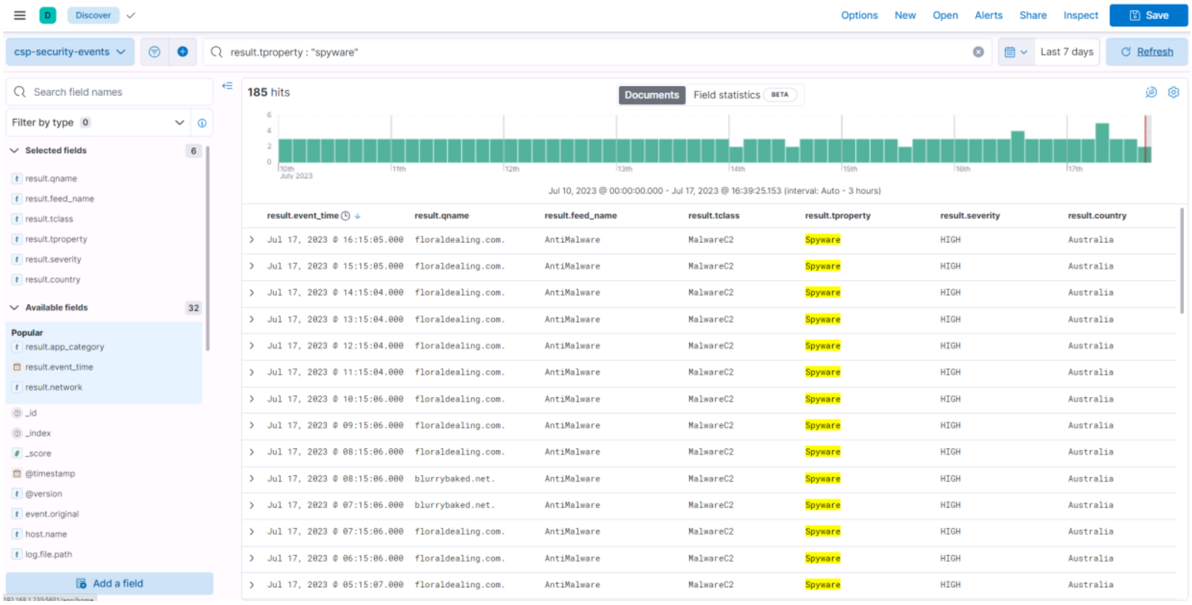


592 hits

Jul 10, 2023 @ 00:00:00.000 - Jul 17, 2023 @ 16:40:11.555 (interval: Auto - 3 hours)

| result.event_time | result.qname | result.feed_name | result.tclass | result.tproperty | result.severity | result.country |
|---|---|---|---|---|---|---|
| Jul 17, 2023 @ 16:15:05.000 | t0.wf. | AntiMalware | MalwareC2 | Generic | HIGH | Australia |
| Jul 17, 2023 @ 16:15:05.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 16:15:05.000 | 4j5.xyz. | AntiMalware | MalwareC2 | Generic | HIGH | Australia |
| Jul 17, 2023 @ 15:25:16.000 | accounts.youtube.com. | badlist | CustomList | badlist | LOW | United States |
| Jul 17, 2023 @ 15:15:05.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 15:15:05.000 | t0.wf. | AntiMalware | MalwareC2 | Generic | HIGH | Australia |
| Jul 17, 2023 @ 15:15:05.000 | 4j5.xyz. | AntiMalware | MalwareC2 | Generic | HIGH | Australia |
| Jul 17, 2023 @ 14:15:04.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 14:15:04.000 | t0.wf. | AntiMalware | MalwareC2 | Generic | HIGH | Australia |
| Jul 17, 2023 @ 14:15:04.000 | 4j5.xyz. | AntiMalware | MalwareC2 | Generic | HIGH | Australia |
| Jul 17, 2023 @ 13:15:04.000 | t0.wf. | AntiMalware | MalwareC2 | Generic | HIGH | Australia |
| Jul 17, 2023 @ 13:15:04.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 13:15:04.000 | 4j5.xyz. | AntiMalware | MalwareC2 | Generic | HIGH | Australia |
| Jul 17, 2023 @ 12:15:04.000 | 4j5.xyz. | AntiMalware | MalwareC2 | Generic | HIGH | Australia |

5. Use the top search bar to return hits with specified field values. Try searching for **result.tproperty : "spyware”**. You can also just type **spyware**. You can save searches, open searches, apply filters and more.



result.tproperty : "spyware"

185 hits

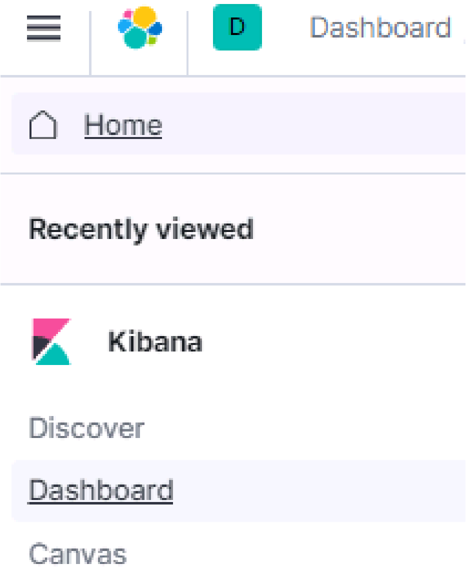
Jul 10, 2023 @ 00:00:00.000 - Jul 17, 2023 @ 16:39:25.153 (interval: Auto - 3 hours)

| result.event_time | result.qname | result.feed_name | result.tclass | result.tproperty | result.severity | result.country |
|---|---|---|---|---|---|---|
| Jul 17, 2023 @ 16:15:05.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 15:15:05.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 14:15:04.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 13:15:04.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 12:15:04.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 11:15:04.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 10:15:06.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 09:15:06.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 08:15:06.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 08:15:06.000 | blurrybaked.net. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 07:15:06.000 | blurrybaked.net. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 07:15:06.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 06:15:06.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |
| Jul 17, 2023 @ 05:15:07.000 | floraldealing.com. | AntiMalware | MalwareC2 | Spyware | HIGH | Australia |

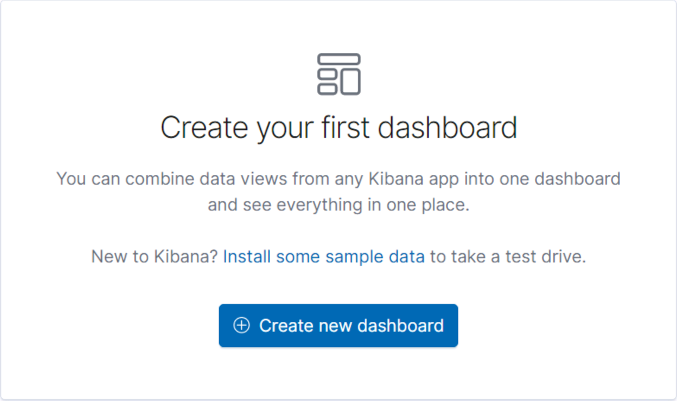
## Kibana Data Visualization

Kibana offers many ways of charting data. Let’s build a pie chart based on the field **tclass**.

1. Click the ☰ menu icon in the topbar. Select **Dashboard**.



2. Click **Create new dashboard**. We’ll add the chart to this dashboard. Many objects, such as charts, maps, and saved searches, can live on a dashboard for convenient access and visualization.



Create your first dashboard

You can combine data views from any Kibana app into one dashboard and see everything in one place.

New to Kibana? Install some sample data to take a test drive.

3. Click **Create visualization**.

   a. Select **Pie** as the type.

   b. Select the data view **csp-security-events**.

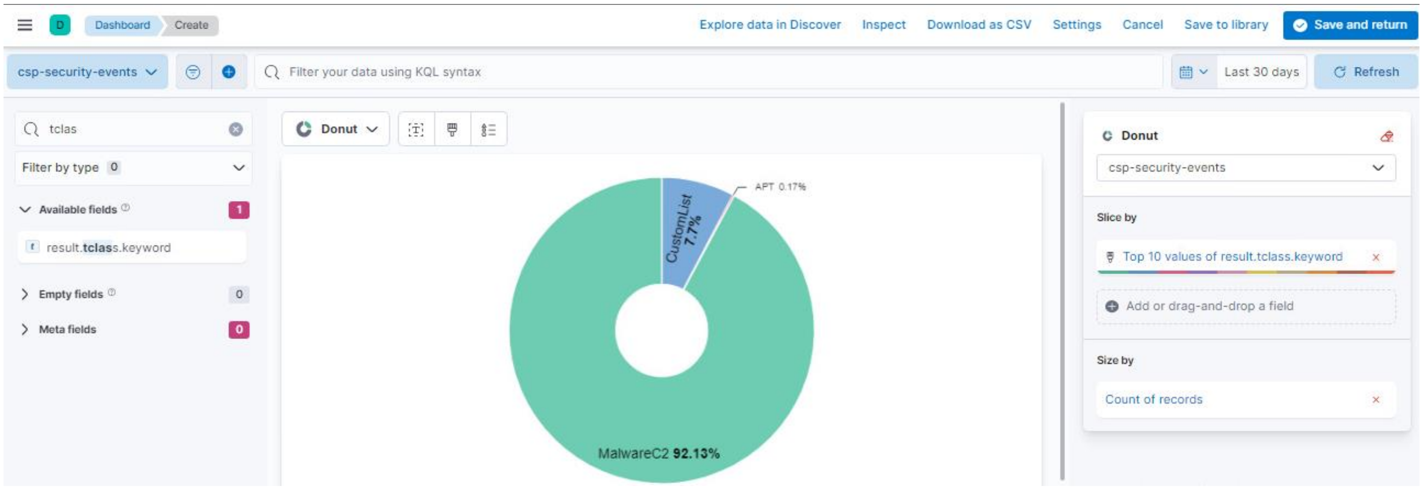
   c. Drag the **result.tclass.keyword** field in the left column to the center. This will automatically:

      i. Slice by **result.tclass.keyword.**

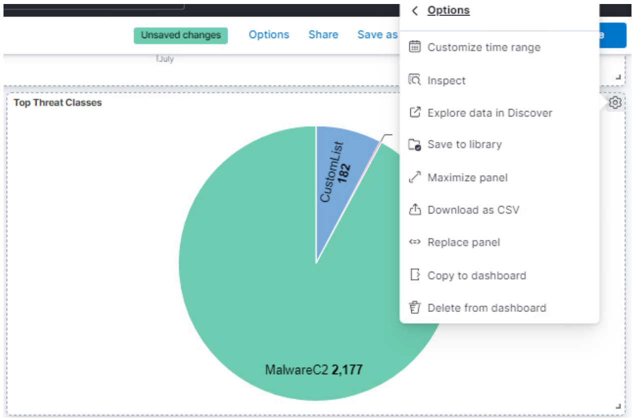
      ii. Size by **Count of records**.

   d. Narrow or expand your dataset with the calendar in the upper right corner as desired.

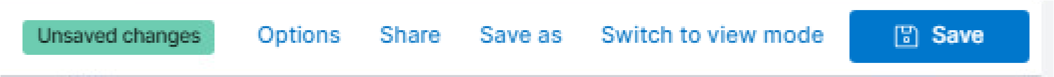
   e. Your chart should look something like below**.** Click **Save and return**.



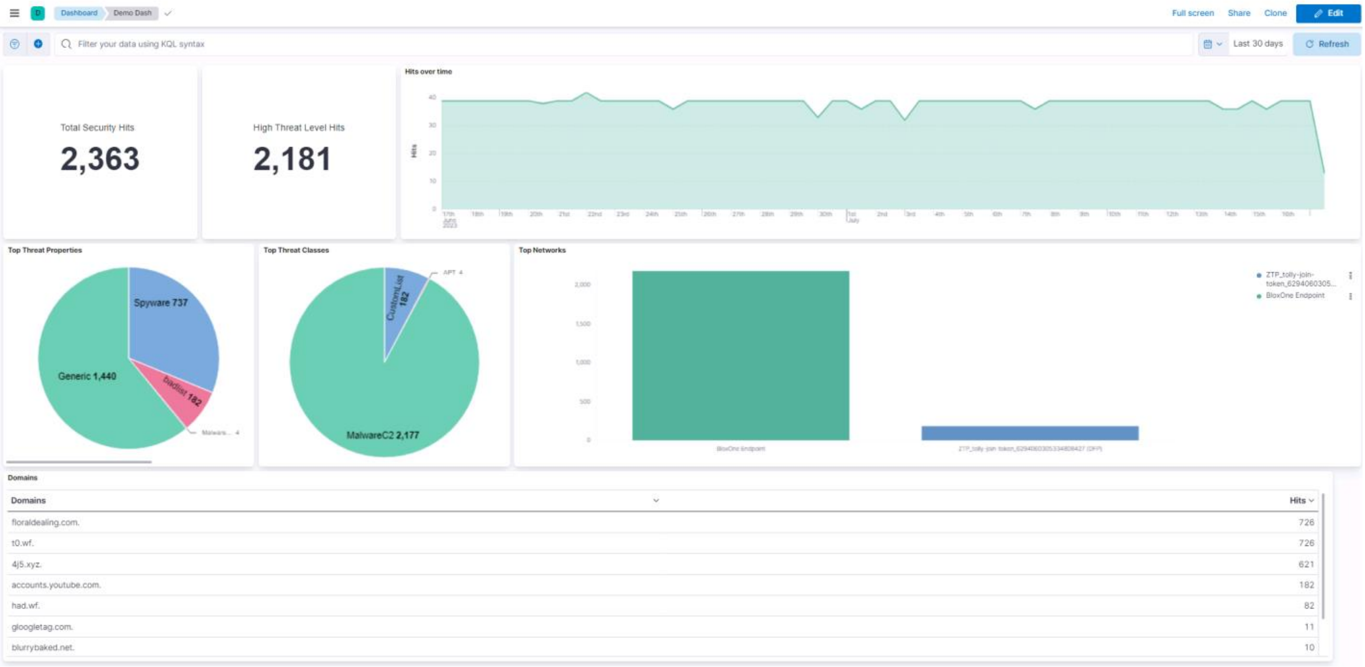
4. Your chart now appears on your dashboard. Click the gears ⚙ options icon.
    a. You can perform several actions here. Click **Edit panel title** to give the panel a name, such as **Top Threat Classes**.
    b. Click **Inspect** to see a breakdown of data for the chart.
    c. You can even **download most charts as a CSV**.



    d. Click **Save** at the top of the dashboard.



You can add many different objects to dashboards, connect them, move them around, filter fields, and much more. Here are some examples showcasing more of Kibana's visualization capabilities:



| Domains | Hits |
|---|---|
| floraldealing.com. | 726 |
| t0.wf. | 726 |
| 4j5.xyz. | 621 |
| accounts.youtube.com. | 182 |
| had.wf. | 82 |
| gloogletag.com. | 11 |
| blurrybaked.net. | 10 |

# Deployment Instructions: NIOS dnstap Logging

You can configure NIOS to use the dnstap log format to log DNS queries and responses at high rates to well-known destinations, such as an Ubuntu VM. dnstap is a flexible, structured binary log format for DNS software. It reduces the workload on NIOS to allow for logging queries and/or responses at higher speeds and performance than regular logging. This section shows you how to ingest dnstap formatted logs from NIOS into Elastic.

Several components are required to ingest dnstap logs into Elastic from NIOS. You need:

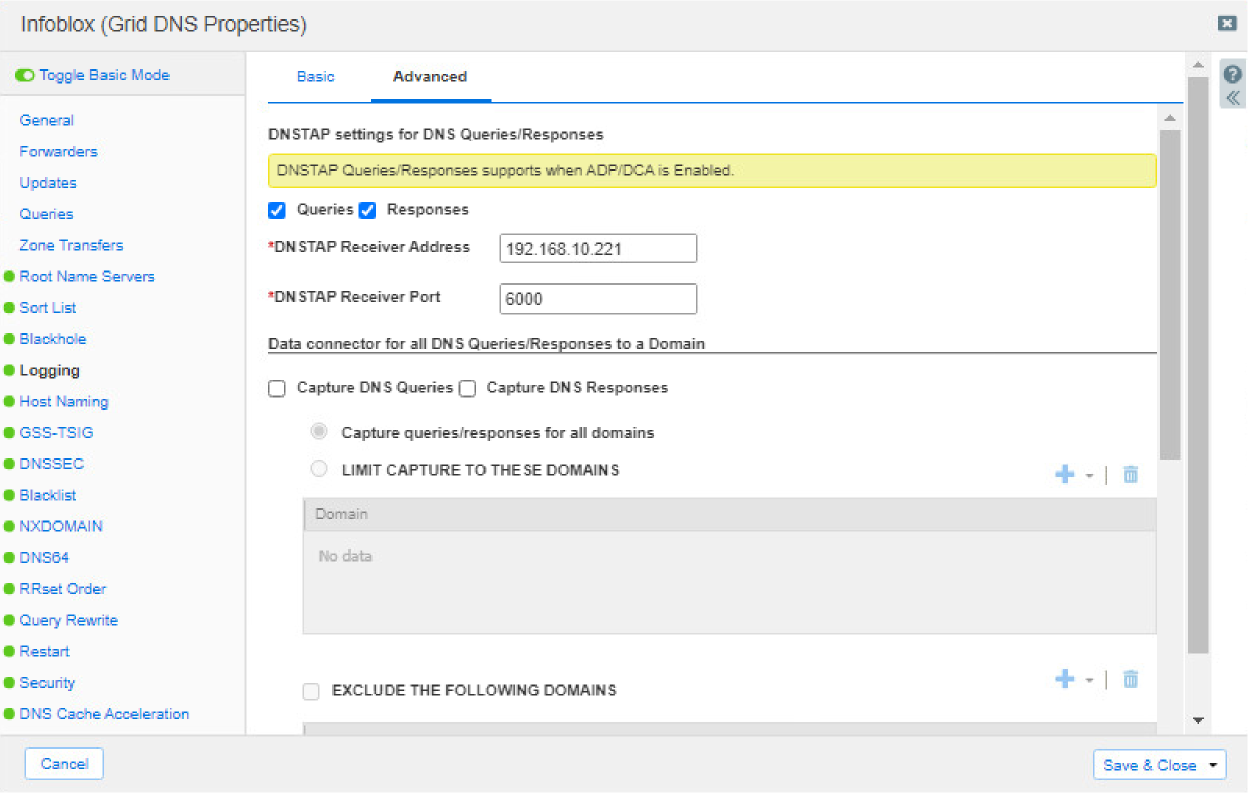
- NIOS with dnstap enabled. *For this demo, an IB-FLEX box was used with the DNS Cache Acceleration service running.*
- An external client to receive dnstap logs from NIOS. *For this demo, an Ubuntu 18.04 VM was used.*
- A way to receive, store and process dnstap logs after logging queries to the external client. *For this demo, the Python module dnstap-receiver was used.*
- Elastic Stack. *For this demo, Elastic Stack was installed on the same Ubuntu VM as dnstap-receiver.*

It is likely that if you are using dnstap, you are logging a large number of queries. It is highly recommended to secure DNS queries and responses between a server and client. Another layer of extra security that NIOS offers is the DNS over TLS and DNS over HTTPS services. These services encrypt DNS queries and responses to secure communication between a DNS server and a DNS client.

## NIOS Configuration

dnstap is only available on NIOS 8.5.1 and higher and can only be configured on specific NIOS boxes running certain services. More information on requirements as well as detailed instructions on configuring dnstap for NIOS can be found on the official Infoblox documentation here.

For this demo, dnstap was enabled on an IB-FLEX box running DNS Cache Acceleration (DCA). The following screenshot shows the NIOS configuration under the *Grid DNS Properties* editor. Observe that both queries and responses are being logged. The **DNSTAP Receiver Address** is the IP of the Ubuntu VM for which Elastic and the dnstap-receiver are installed. The default port used is **6000**.



DNSTAP settings for DNS Queries/Responses

DNSTAP Queries/Responses supports when ADP/DCA is Enabled.

☑ Queries ☑ Responses

*DNSTAP Receiver Address: 192.168.10.221

*DNSTAP Receiver Port: 6000

Data connector for all DNS Queries/Responses to a Domain

□ Capture DNS Queries □ Capture DNS Responses

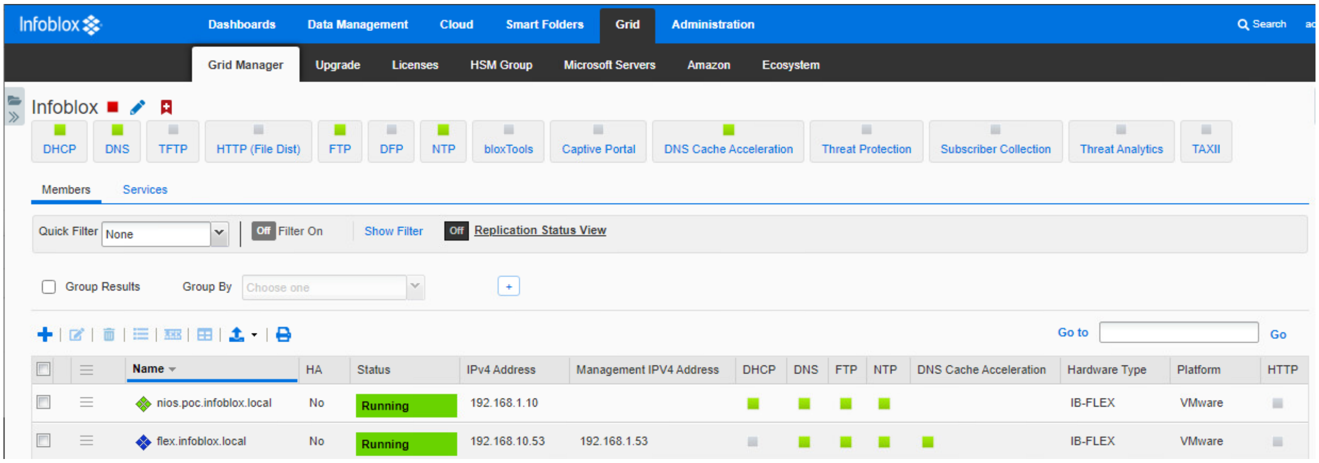
○ Capture queries/responses for all domains

○ LIMIT CAPTURE TO THESE DOMAINS

| Domain |
|---|
| No data |

□ EXCLUDE THE FOLLOWING DOMAINS

For reference, the below screenshot shows the IP configuration and services enabled for the Grid.



| Name | HA | Status | IPv4 Address | Management IPV4 Address | DHCP | DNS | FTP | NTP | DNS Cache Acceleration | Hardware Type | Platform | HTTP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| nios.poc.infoblox.local | No | Running | 192.168.1.10 | | 🟩 | 🟩 | 🟩 | 🟩 | | IB-FLEX | VMware | ⬜ |
| flex.infoblox.local | No | Running | 192.168.10.53 | 192.168.1.53 | ⬜ | 🟩 | 🟩 | 🟩 | 🟩 | IB-FLEX | VMware | ⬜ |

## dnstap-receiver Configuration

NIOS currently has no way to store or process dnstap logs after they leave the Grid. You will need some way to unpack and read the incoming dnstap messages from NIOS. A simple solution is to install *dnstap-receiver*, a python module that receives dnstap messages and outputs them in a way Elastic can ingest. It supports several input stream types and can be configured to output readable data in many ways, such as to a syslog server, stdout, a file, and more.

We will be using dnstap-receiver to ingest the dnstap messages from NIOS and output them to a file. Later, we will configure Logstash to ingest the file into Kibana. dnstap-receiver is configured with external config files, similar to Logstash. Let's create the config file that will output the readable messages to a file. *Note: For this demo, dnstap-receiver was installed on the same Ubuntu VM as Elastic. It is best practice to keep these pieces of software installed on separate VMs.*

1. Access the machine where your dnstap-receiver instance is installed. Follow the installation instructions on its Python module page to install it.

2. Open a terminal.

3. To keep tidy, create a new directory for which dnstap will output the logfile that will be ingested by Elastic:

   ```
   sudo mkdir /var/log/dnstap
   ```

4. Then create the logfile. *Note: We do this because the logfile must exist before executing dnstap-receiver. Otherwise it will throw an error.*

   ```
   sudo touch /var/log/dnstap/dnstap.json
   ```

5. You must allow the logfile to be written to by dnstap-receiver. Enter the following command to allow all the files inside `/var/log/dnstap` to be readable, writable, and executable to all users on the computer. You can store the logfile in a writeable directory somewhere else, such as Documents, if you do not wish to change permissions.

   ```
   sudo chmod -R 777 /var/log/dnstap/
   ```

6. Now create a new directory for which the config file will live:

   ```
   sudo mkdir /etc/dnstap-receiver
   ```

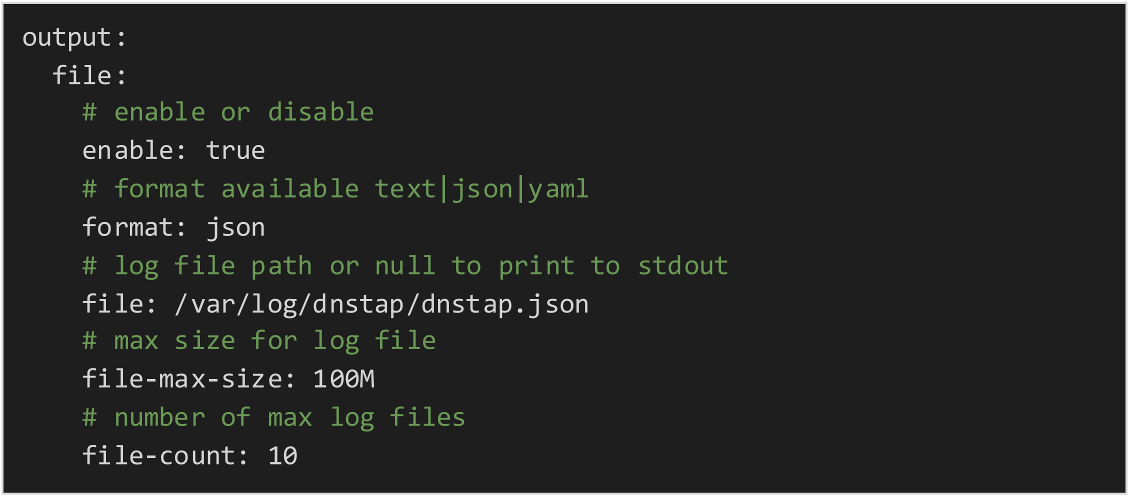
7. Then create the config file:

   ```
   sudo touch /etc/dnstap-receiver/dnstap.conf
   ```

8. Open the file with `gedit` for editing:

   ```
   sudo gedit /etc/dnstap-receiver/dnstap.conf
   ```

9. Copy and paste the following into the file. Save and close the file when finished.



```
output:
  file:
    # enable or disable
    enable: true
    # format available text|json|yaml
    format: json
    # log file path or null to print to stdout
    file: /var/log/dnstap/dnstap.json
    # max size for log file
    file-max-size: 100M
    # number of max log files
    file-count: 10
```

This file tells dnstap-receiver to output the dnstap messages in json format to the `dnstap.json` file we created earlier. You can set various other parameters here, such as the max file size of the logfile or the max number of files to keep. **These files can potentially become very large so adjust and rotate according to your needs. Logstash offers many modules to simplify this.**

## Logstash Configuration for dnstap

Now let's configure Logstash to grab the data in `dnstap.json` logged by dnstap-receiver. View the Logstash Configuration for TD section of this document for more details on Logstash and config files.

1. Access the machine where your Logstash instance is installed. *Note: For this demo Elastic Stack was installed on Ubuntu 18.04.*

2. Open a terminal.

3. Navigate to where your Logstash configuration (`.conf`) files are located. In this demonstrative environment, these files are located in `/etc/logstash/conf.d`. Input the following command to navigate to the correct directory:

```
cd /etc/logstash/conf.d
```

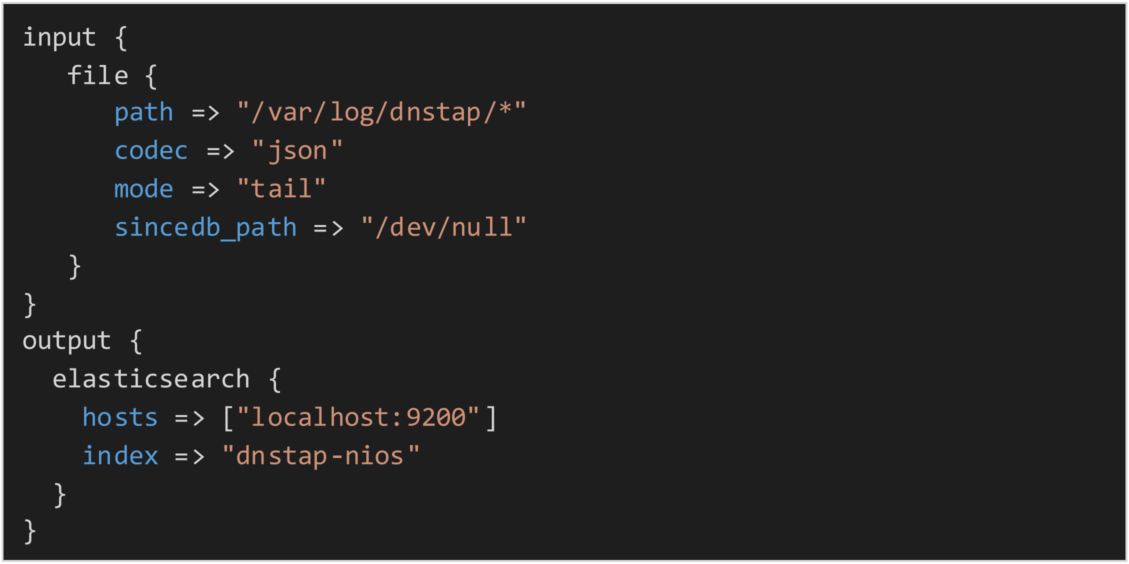
4. Create a new file called `dnstap-nios.conf`:

```
sudo touch dnstap-nios.conf
```

5. Open the file with `gedit` for editing:

```
sudo gedit dnstap-nios.conf
```

6. Copy and paste the following into the file. Save and close the file when finished.



```
input {
   file {
      path => "/var/log/dnstap/*"
      codec => "json"
      mode => "tail"
      sincedb_path => "/dev/null"
   }
}
output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "dnstap-nios"
  }
}
```

Note we are grabbing everything in the `/var/log/dnstap` directory we created earlier. Because dnstap-receiver is set to *append* the `dnstap.json` file with all the dnstap messages, we set the mode to `tail`.

7. Navigate to your home directory for Logstash. For this demo, this is `/usr/share/logstash/`. Input the following command to navigate to the correct directory:

```
cd /usr/share/logstash
```

8. Run Logstash with your new configuration:

```
sudo bin/logstash -f /etc/logstash/conf.d/dnstap-nios.conf
```

Allow several minutes of processing. The console will inform you if there are any syntax errors with your config file.

- Alternatively, you can simply restart the Logstash service, but the console will not warn you of any errors with your config file:

```
sudo systemctl restart logstash
```

## Run & Test Configuration

Let’s run our new configuration.

First, we need to run dnstap-receiver with the config file created in the dnstap-receiver Configuration section of this document. Run this command on the Ubuntu VM during the entire time you wish to collect dnstap messages from NIOS.

1. Open a terminal.
2. Run dnstap-receiver using the config file as a parameter:

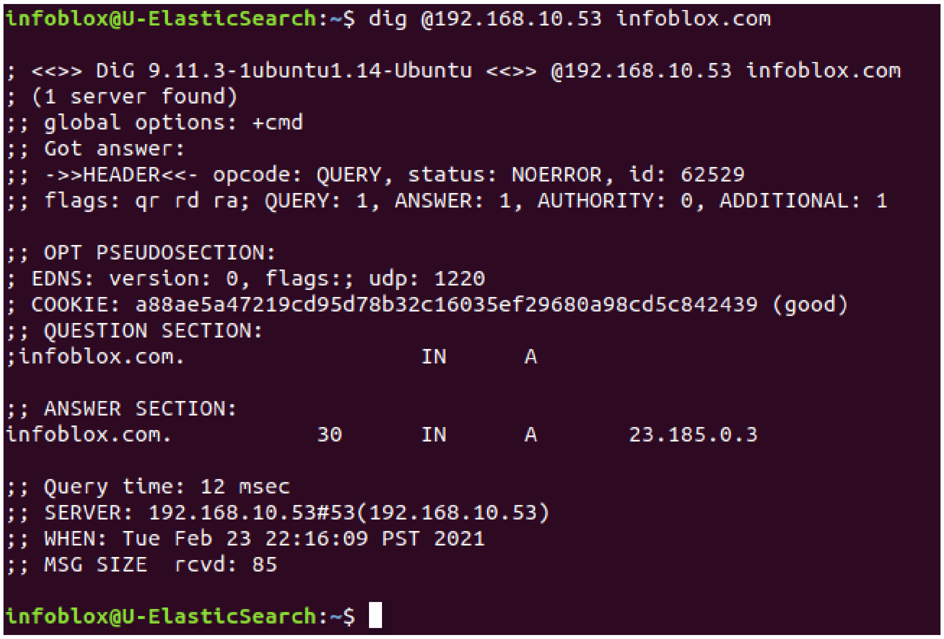
```
dnstap_receiver -c /etc/dnstap-receiver/dnstap.conf
```

The console will tell you if there are syntax errors with your config file.

Let’s run a few test queries now. We will use the `dig` command to do so.

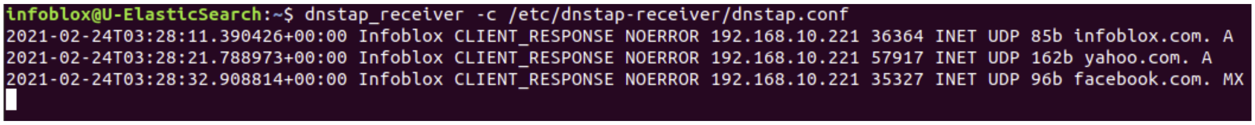
1. Open a new terminal or terminal tab without halting the terminal where dnstap-receiver is running.
2. Run a couple dig commands using the IP address of the NIOS Grid Member running dnstap. Try querying `infoblox.com`. *Note: For this demo, the IP used in the `dig` is the LAN Interface IP of the IB-FLEX box running DNS Cache Acceleration.*

```
dig @192.168.10.53 infoblox.com
```



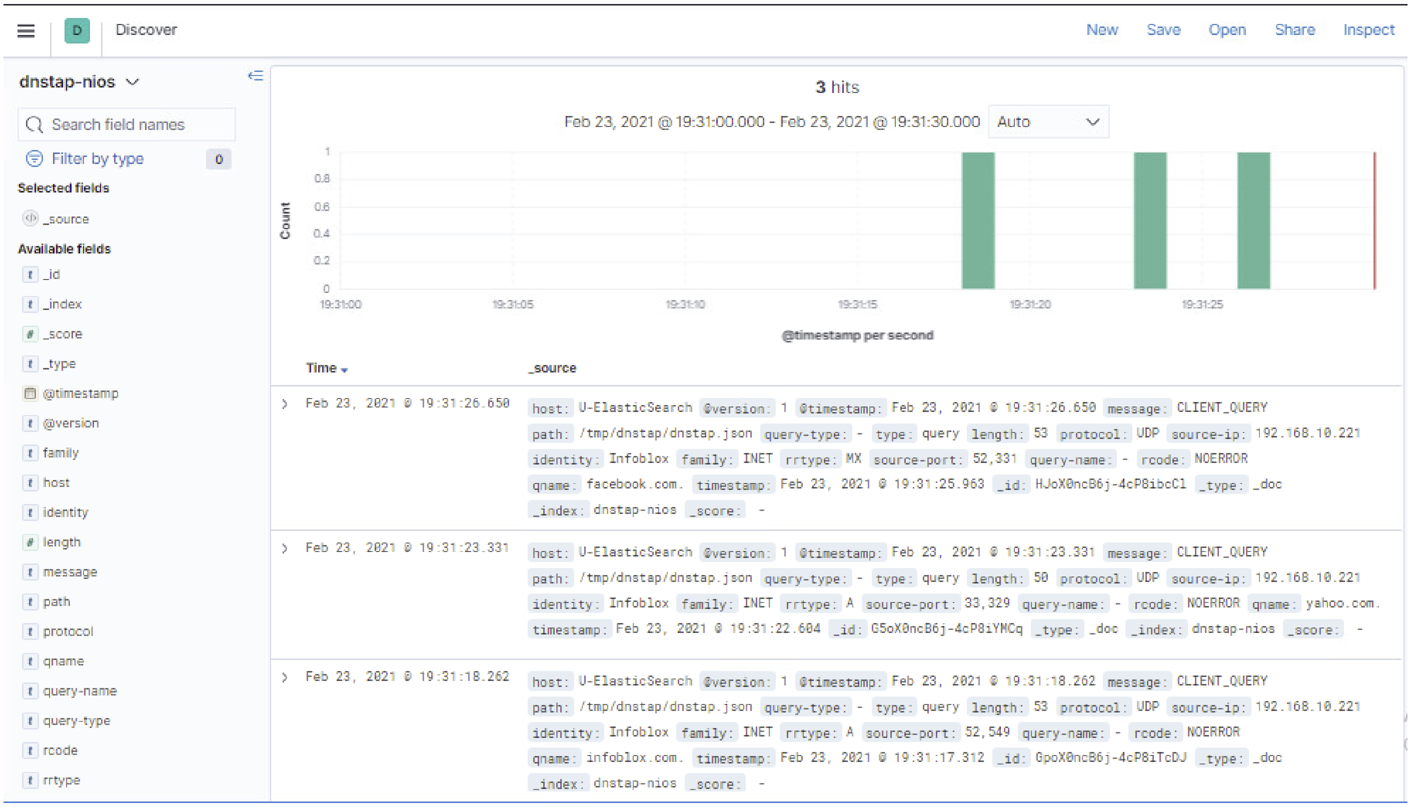
```
infoblox@U-ElasticSearch:~$ dig @192.168.10.53 infoblox.com

; <<>> DiG 9.11.3-1ubuntu1.14-Ubuntu <<>> @192.168.10.53 infoblox.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62529
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1220
; COOKIE: a88ae5a47219cd95d78b32c16035ef29680a98cd5c842439 (good)
;; QUESTION SECTION:
;infoblox.com.                  IN      A

;; ANSWER SECTION:
infoblox.com.           30      IN      A       23.185.0.3

;; Query time: 12 msec
;; SERVER: 192.168.10.53#53(192.168.10.53)
;; WHEN: Tue Feb 23 22:16:09 PST 2021
;; MSG SIZE  rcvd: 85

infoblox@U-ElasticSearch:~$ 
```

When queries are received by dnstap-receiver, they will appear in the dnstap-receiver stdout.



```
infoblox@U-ElasticSearch:~$ dnstap_receiver -c /etc/dnstap-receiver/dnstap.conf
2021-02-24T03:28:11.390426+00:00 Infoblox CLIENT_RESPONSE NOERROR 192.168.10.221 36364 INET UDP 85b infoblox.com. A
2021-02-24T03:28:21.788973+00:00 Infoblox CLIENT_RESPONSE NOERROR 192.168.10.221 57917 INET UDP 162b yahoo.com. A
2021-02-24T03:28:32.908814+00:00 Infoblox CLIENT_RESPONSE NOERROR 192.168.10.221 35327 INET UDP 96b facebook.com. MX
```

3. Open Kibana and observe the queries are ingested into Kibana. For more information on Kibana, indices and creating visualizations, see the Kibana Data Discovery and Kibana Data Visualization sections of this document.



3 hits

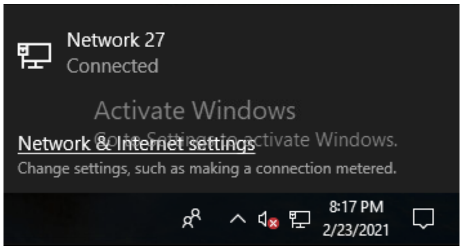
Feb 23, 2021 @ 19:31:00.000 - Feb 23, 2021 @ 19:31:30.000 Auto

| Time | _source |
|---|---|
| Feb 23, 2021 @ 19:31:26.650 | host: U-ElasticSearch @version: 1 @timestamp: Feb 23, 2021 @ 19:31:26.650 message: CLIENT_QUERY path: /tmp/dnstap/dnstap.json query-type: - type: query length: 53 protocol: UDP source-ip: 192.168.10.221 identity: Infoblox family: INET rrtype: MX source-port: 52,331 query-name: - rcode: NOERROR qname: facebook.com. timestamp: Feb 23, 2021 @ 19:31:25.963 _id: HJoX0ncB6j-4cP8ibcCl _type: _doc _index: dnstap-nios _score: - |
| Feb 23, 2021 @ 19:31:23.331 | host: U-ElasticSearch @version: 1 @timestamp: Feb 23, 2021 @ 19:31:23.331 message: CLIENT_QUERY path: /tmp/dnstap/dnstap.json query-type: - type: query length: 50 protocol: UDP source-ip: 192.168.10.221 identity: Infoblox family: INET rrtype: A source-port: 33,329 query-name: - rcode: NOERROR qname: yahoo.com. timestamp: Feb 23, 2021 @ 19:31:22.604 _id: G5oX0ncB6j-4cP8iYMCq _type: _doc _index: dnstap-nios _score: - |
| Feb 23, 2021 @ 19:31:18.262 | host: U-ElasticSearch @version: 1 @timestamp: Feb 23, 2021 @ 19:31:18.262 message: CLIENT_QUERY path: /tmp/dnstap/dnstap.json query-type: - type: query length: 53 protocol: UDP source-ip: 192.168.10.221 identity: Infoblox family: INET rrtype: A source-port: 52,549 query-name: - rcode: NOERROR qname: infoblox.com. timestamp: Feb 23, 2021 @ 19:31:17.312 _id: GpoX0ncB6j-4cP8iTcDJ _type: _doc _index: dnstap-nios _score: - |

We can also test with a Windows machine to see the ingesting in action instead of sending `dig` requests.

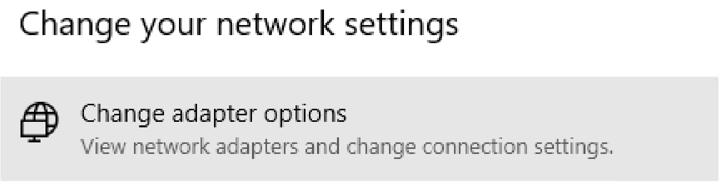
1. Access a Windows machine on the same network as your Grid. *Note: For this demo, we will use the same Windows machine used to access the Kibana UI.*

2. Open your **Internet and Network settings**.

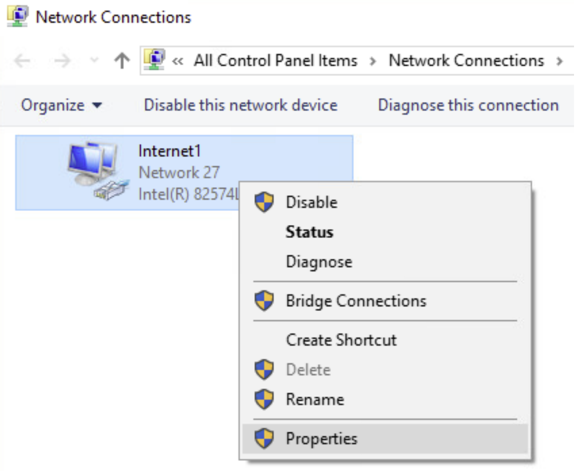


Network 27
Connected
Activate Windows
Network & Internet settings
Go to Settings to activate Windows.
Change settings, such as making a connection metered.
8:17 PM
2/23/2021

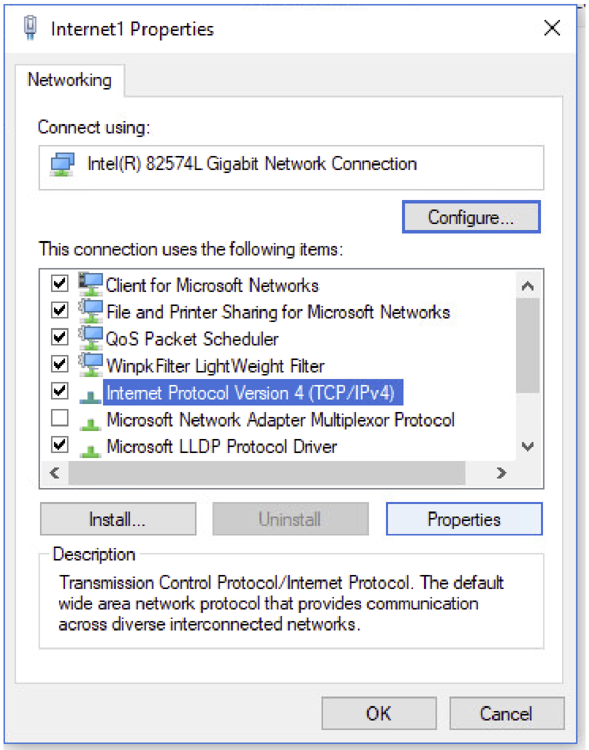
3. Click **Change adapter options**.


Change your network settings
Change adapter options
View network adapters and change connection settings.

4. Right-click on your desired connection and click **Properties**.



5. Click on **Internet Protocol Version 4 (TCP/IPv4)** and click **Properties**.



Connect using:

Intel(R) 82574L Gigabit Network Connection

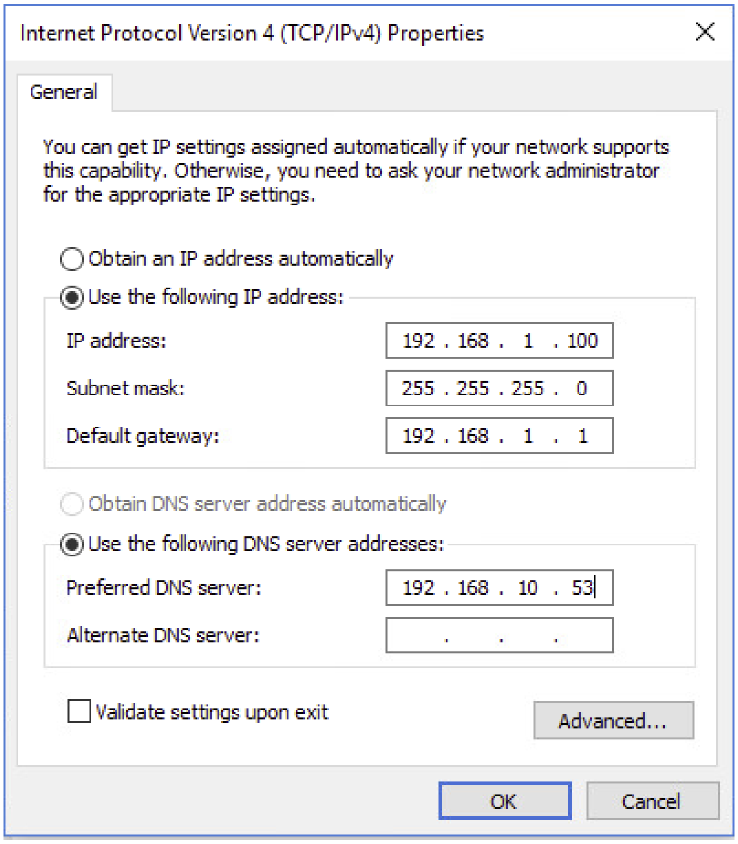
This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ File and Printer Sharing for Microsoft Networks
- ☑ QoS Packet Scheduler
- ☑ Winpk Filter LightWeight Filter
- ☑ Internet Protocol Version 4 (TCP/IPv4)
- ☐ Microsoft Network Adapter Multiplexor Protocol
- ☑ Microsoft LLDP Protocol Driver

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

6. Set the **Preferred DNS server** to the LAN IP address of the NIOS Grid Member running dnstap. Click **OK** when done. *Note: For this demo, the IP used is the LAN Interface IP of the IB-FLEX box running DNS Cache Acceleration.*



Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

| | |
|---|---|
| IP address: | 192 . 168 . 1 . 100 |
| Subnet mask: | 255 . 255 . 255 . 0 |
| Default gateway: | 192 . 168 . 1 . 1 |

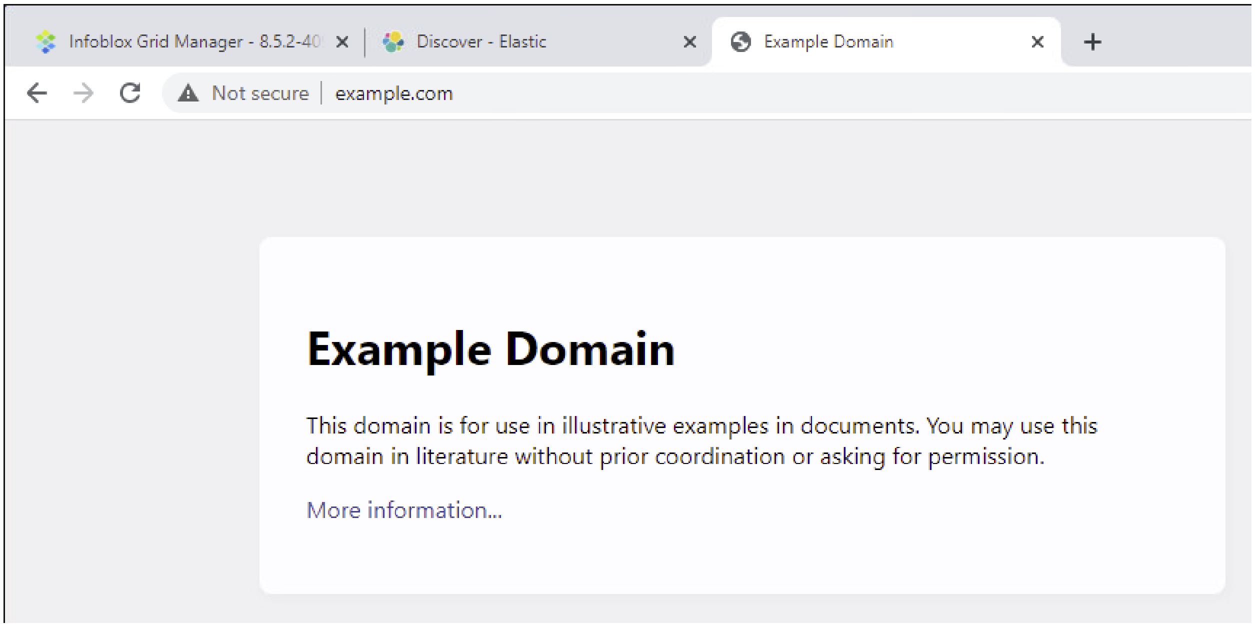
○ Obtain DNS server address automatically

● Use the following DNS server addresses:

| | |
|---|---|
| Preferred DNS server: | 192 . 168 . 10 . 53 |
| Alternate DNS server: | . . . |

□ Validate settings upon exit

7. Open Chrome. Go to http://example.com.

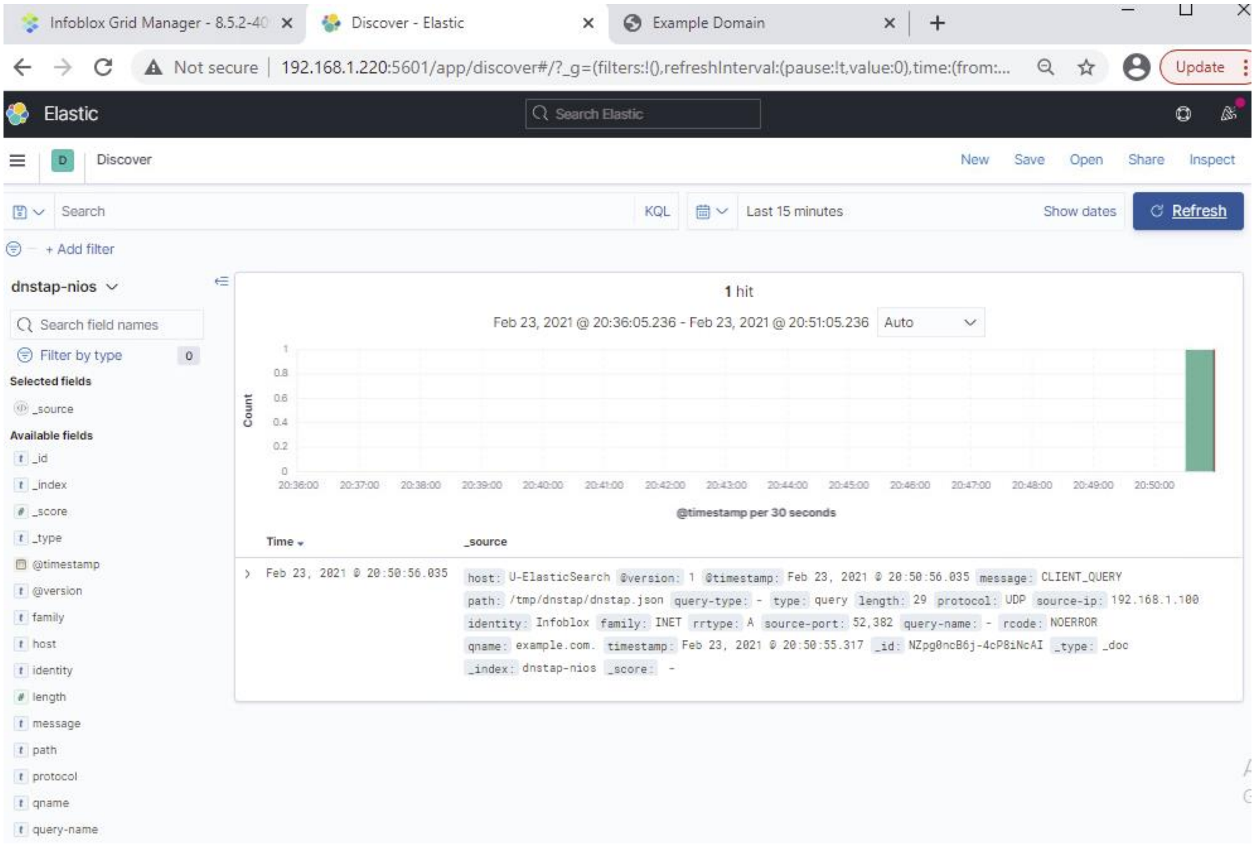


## Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

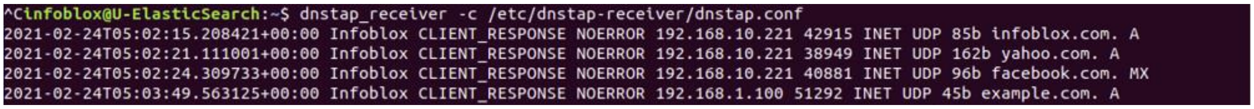
More information...

8. In Kibana, observe the query has been ingested.



```
Feb 23, 2021 @ 20:50:56.035   host: U-ElasticSearch @version: 1 @timestamp: Feb 23, 2021 @ 20:50:56.035 message: CLIENT_QUERY
                              path: /tmp/dnstap/dnstap.json query-type: - type: query length: 29 protocol: UDP source-ip: 192.168.1.100
                              identity: Infoblox family: INET rrtype: A source-port: 52,382 query-name: - rcode: NOERROR
                              qname: example.com. timestamp: Feb 23, 2021 @ 20:50:55.317 _id: NZpg0ncB6j-4cP8iNcAI _type: _doc
                              _index: dnstap-nios _score: -
```

The query also appears in the dnstap-receiver stdout.



```
^Cinfoblox@U-ElasticSearch:~$ dnstap_receiver -c /etc/dnstap-receiver/dnstap.conf
2021-02-24T05:02:15.208421+00:00 Infoblox CLIENT_RESPONSE NOERROR 192.168.10.221 42915 INET UDP 85b infoblox.com. A
2021-02-24T05:02:21.111001+00:00 Infoblox CLIENT_RESPONSE NOERROR 192.168.10.221 38949 INET UDP 162b yahoo.com. A
2021-02-24T05:02:24.309733+00:00 Infoblox CLIENT_RESPONSE NOERROR 192.168.10.221 40881 INET UDP 96b facebook.com. MX
2021-02-24T05:03:49.563125+00:00 Infoblox CLIENT_RESPONSE NOERROR 192.168.1.100 51292 INET UDP 45b example.com. A
```

Infoblox unites networking and security to deliver unmatched performance and protection. Trusted by Fortune 100 companies and emerging innovators, we provide real-time visibility and control over who and what connects to your network, so your organization runs faster and stops threats earlier.

Corporate Headquarters
2390 Mission College Blvd, Ste. 501
Santa Clara, CA 95054
+1.408.986.4000
www.infoblox.com