Lab 1.1 - Introduction to SIEM Architecture

Objectives

- · Be comfortable with using the Elastic Stack
- · Establish a high-level understanding of SIEM architecture
- Learn how to collect logs manually
- Interact with the various components of a SIEM
- · Use an alert engine to create an alert

Exercise Preparation

Log into the Sec-555 VM

• Username: student

Password: sec555



The overall objective of this lab is to learn the various components of a SIEM by using them. Installation of each component has already been performed, and all configuration files have been pre-built.

Logstash (log aggregator) configuration files are in /labs/1.1/files/

Filebeat has been preconfigured for this lab and can be run using the command below. This is informational is for students who want to know how to run filebeat for the no hints version. If you are doing the step-by-step walkthrough ignore this command until you are asked to run it in the walkthrough.

```
filebeat -c /labs/1.1/filebeat.yml
```

This lab deals with reading logs from /var/log/ on your student virtual machine. Thus, the number of logs represented, and the timestamps will not match the pictures in the step-by-step instructions.

Exercises

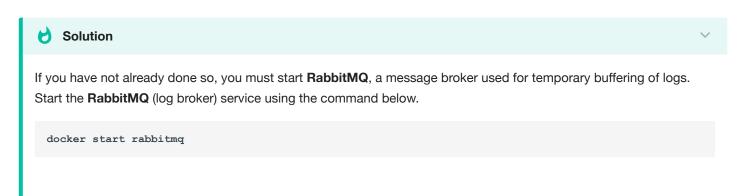
Before starting the lab, you must start **RabbitMQ**, a message broker used for temporary buffering of logs. Start the **RabbitMQ** (log broker) service using the command below.

docker start rabbitmq



These services are not started by default to save system resources. In a production environment, this would be configured to start automatically.

1. Send logs from /var/log/*.log to Logstash (log aggregator) using Filebeat (log agent). Output logs to the screen



Note

These services are not started by default to save system resources. In a production environment, this would be configured to start automatically.

In this section, the config files will pick up logs using **Filebeat** and send them to **Logstash**, which will only display them to the screen. This demonstrates a log agent sending logs to a central location.

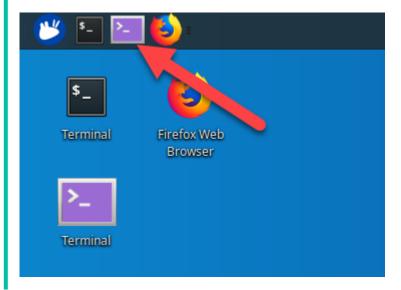
To send logs from a log agent to a log aggregator, the log aggregator must first be running. The Elastic Stack uses **Logstash** as a log aggregator, but for this class, Logstash is not set up as a service. Manually run **Logstash** and have it use the configuration file called **debug.conf**. Do this by running the command below.

logstash -f /labs/1.1/files/debug.conf

Wait until you see "Pipeline running." The output will reflect as below. Moving forward only the last line of "Pipelines running" will be shown.

```
Sending Logstash's logs to /usr/share/logstash/logs which is now configured via log4j2.properties
[2018-05-20T19:49:48,765][INFO][logstash.modules.scaffold] Initializing module
{:module name=>"netflow", :directory=>"/usr/share/logstash/modules/netflow/configuration"}
[2018-05-20T19:49:48,785][INFO][logstash.modules.scaffold] Initializing module
{:module name=>"fb apache", :directory=>"/usr/share/logstash/modules/fb apache/configuration"}
[2018-05-20T19:49:48,950][INFO ][logstash.setting.writabledirectory] Creating directory
{:setting=>"path.queue", :path=>"/usr/share/logstash/data/queue"}
[2018-05-20T19:49:48,955][INFO][logstash.setting.writabledirectory] Creating directory
{:setting=>"path.dead_letter_queue", :path=>"/usr/share/logstash/data/dead_letter_queue"}
[2018-05-20T19:49:49,410][WARN][logstash.config.source.multilocal] Ignoring the 'pipelines.yml'
file because modules or command line options are specified
[2018-05-20T19:49:49,458][INFO][logstash.agent
                                                        ] No persistent UUID file found.
Generating new UUID {:uuid=>"defed98f-d0b1-4416-a0c8-ba498d60105e", :path=>"/usr/share/logstash/
data/uuid"}
[2018-05-20T19:49:50,079][INFO ][logstash.runner
                                                        ] Starting Logstash
{"logstash.version"=>"6.2.2"}
[2018-05-20T19:49:50,469][INFO ][logstash.agent ] Successfully started Logstash API
endpoint {:port=>9600}
[2018-05-20T19:50:00,136][INFO][logstash.pipeline ] Starting pipeline {:pipeline id=>"main",
"pipeline.workers"=>4, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>50}
[2018-05-20T19:50:00,528][INFO ][logstash.filters.geoip | Using geoip database {:path=>"/usr/
share/logstash/vendor/bundle/jruby/2.3.0/gems/logstash-filter-geoip-5.0.3-java/vendor/GeoLite2-
City.mmdb"}
[2018-05-20T19:50:00,552][INFO ][logstash.filters.geoip | Using geoip database {:path=>"/usr/
share/logstash/vendor/bundle/jruby/2.3.0/gems/logstash-filter-geoip-5.0.3-java/vendor/GeoLite2-
ASN.mmdb"}
[2018-05-20T19:50:00,916][INFO ][logstash.inputs.beats ] Beats inputs: Starting input listener
{:address=>"0.0.0.0:5044"}
[2018-05-20T19:50:00,983][INFO][logstash.pipeline
                                                       ] Pipeline started successfully
{:pipeline id=>"main", :thread=>"#<Thread:0x7fefa656 run>"}
[2018-05-20T19:50:01,046][INFO][org.logstash.beats.Server] Starting server on port: 5044
[2018-05-20T19:50:01,125][INFO ][logstash.agent
                                                        ] Pipelines running
{:count=>1, :pipelines=>["main"]}
```

This means that **Logstash** is running. Next, open a new terminal that will be used to demonstrate a log agent. To do this, left click on the purple terminal icon at the top of the screen.



This terminal will be referred to as the **Agent Terminal**. The next steps are to be performed on the **Agent Terminal**. This is used to visually distinguish between **Logstash**, a log aggregator on the **black terminal**, and **Filebeat**, a log agent on the purple terminal. Typically, **Filebeat** would be running on a remote machine. In the **Agent Terminal**, run **Filebeat** using the command below.

```
filebeat -c /labs/1.1/filebeat.yml
```

Switch back to the **black terminal**, and you should see that **Logstash** has accepted the logs sent from Filebeat. The output should look like the image below but **WILL NOT** be the same. The logs and timestamps will be specific to your system.

```
"syslog_severity" => "notice",
                "tags" => [
    [0] "beats input codec plain applied",
    [1] " grokparsefailure",
    [2] "_geoip_lookup_failure"
],
                "host" => "filebeat",
             "message" => "2018-04-26 18:25:33 trigproc mime-support:all 3.60ubuntu1 <none>",
             "source" => "/var/log/dpkg.log",
"syslog_facility_code" => 1,
           "@version" => "1",
               "beat" => {
        "name" => "filebeat",
   "hostname" => "filebeat",
    "version" => "6.2.4"
},
             "offset" => 1019662,
          "@timestamp" => 2018-05-20T20:17:20.895Z,
          "prospector" => {
   "type" => "log"
},
"syslog_severity_code" => 5,
     "syslog_facility" => "user-level"
```

Note

Outputting logs to the screen can be helpful for debugging. That is what this step is demonstrating.

Within the **black terminal**, hit **CTRL** + **C** to stop **Logstash**. You will receive a "**Pipeline has terminated**" message such as below. Moving forward only the last line of "**Pipeline has terminated**" will be referenced.

```
[2018-05-20T20:17:28,261][WARN][logstash.runner] Received shutdown signal, but pipeline is still waiting for in-flight events to be processed. Sending another ^C will force quit Logstash, but this may cause data loss.
[2018-05-20T20:17:28,544][WARN][logstash.shutdownwatcher] {"inflight_count"=>0,
```

```
stalling thread info"=>{"other"=>[{"thread id"=>36, "name"=>"[main]<bedts", "current call"=>"[...]/
vendor/bundle/jruby/2.3.0/qems/logstash-input-beats-5.0.6-java/lib/logstash/inputs/beats.rb:199:in
`run'"}], ["LogStash::Filters::GeoIP", {"default_database_type"=>"ASN", "source"=>"source_ip",
"id"=>"f3af272bc6887631c19738e0819932a4c0de091b7e111f5826c9cba98f476e86"}]=>[{"thread id"=>31,
"name"=>nil, "current call"=>"[...]/logstash-core/lib/logstash/util/wrapped synchronous queue.rb:
90:in `read batch'"}, {"thread id"=>32, "name"=>nil, "current call"=>"[...]/logstash-core/lib/
logstash/util/wrapped synchronous queue.rb:90:in `read batch'"}, {"thread id"=>33, "name"=>nil,
"current call"=>"[...]/logstash-core/lib/logstash/util/wrapped synchronous queue.rb:90:in
`read batch'"}, {"thread id"=>34, "name"=>nil, "current call"=>"[...]/logstash-core/lib/logstash/
util/wrapped synchronous queue.rb:90:in `read batch'"}]}}
[2018-05-20T20:17:32,078][INFO][logstash.pipeline
                                                         ] Pipeline has terminated
{:pipeline_id=>"main", :thread=>"#<Thread:0x1c5a3ccb run>"}
```

Switch back to the Agent Terminal and hit CTRL + C to stop Filebeat. Filebeat will not display anything when stopped. Instead, it will simply terminate.



Do not continue unless you have stopped Filebeat by pressing CTRL + C. The student VM will not let you run two instances of Filebeat at the same time.

2. Send logs from /var/log/*.log to Logstash using Filebeat and output logs to Elasticsearch (storage) in an index called lab1.1-aggregator_only



Solution

In this section, the config files will pick up logs using Filebeat and send them to Logstash, which will then forward the logs to Elasticsearch for storage. This demonstrates a log agent sending logs to a central location which in turn parses and stores the logs.

Switch back to the black terminal and run Logstash with the aggregator_only.conf configuration file. Do this by running the command below. This is a single line command.

```
logstash -f /labs/1.1/files/aggregator only.conf
```

Wait until you see "Pipeline running." Switch back to the Agent Terminal and run Filebeat using the command below.

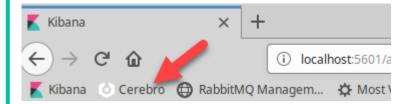
```
filebeat -c /labs/1.1/filebeat.yml
```

This time, the black terminal will not show logs. This is because they are being sent directly to Elasticsearch, the back-end storage system. If logs are properly received, then an index called lab1.1-aggregator_only should be created and contain all the logs Filebeat sent. There are multiple ways to see if logs were accepted and the index was created. One method is to use a web-based management tool. Most SIEMs provide a GUI method for this. For

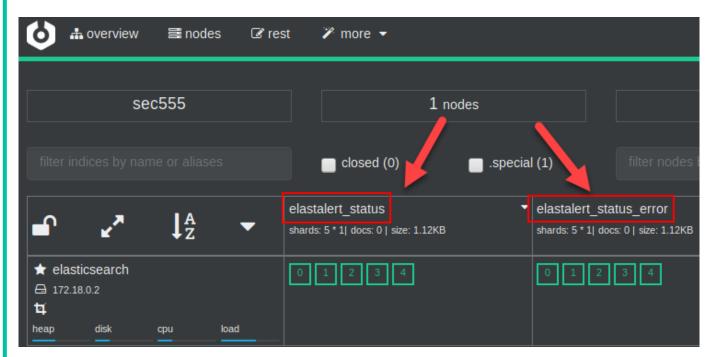
Elasticsearch, you can use **Marvel**, scripts, or community plugins such as **Cerebro**. This lab uses **Cerebro**. First, open **Firefox**.



In **Firefox**, switch to **Cerebro** by clicking on it in the **Bookmarks Toolbar**. **Cerebro** is a GUI management interface for monitoring and changing **Elasticsearch**.



The page displayed is a web front end to manage **Elasticsearch** settings and indexes. The names reflected in the columns are index names. These are placeholders for similar logs and are covered in more detail throughout the course.





Indexes are special files that split logs across shards. If this terminology makes your head hurt, then think of an index as a traditional database. Under the hood, a shard operates differently than a traditional database but conceptually is similar. At the time this screenshot was taken, the **elastalert_status** index had **0** logs. You can tell this by looking at the number of docs found immediately under the index name. This index is a special index that stores alerts. Your student VM may have some alerts pre-generated in the **elastalert_status**.

When searching for indices in **Cerebro**, you might have to click on the arrows to see the **lab1.1-aggregator_only** index or type **1.1** in the **filter indices by name** search bar. Type **1.1** in the **filter indices by name** or alias bar.



Go back to the black terminal and hit CTRL + C to stop Logstash.

```
[2018-05-21T02:25:32,161][WARN ][logstash.runner ] SIGINT received. Shutting down.
[2018-05-21T02:25:32,600][INFO ][logstash.pipeline ] Pipeline has terminated
{:pipeline_id=>"main", :thread=>"#<Thread:0x4aeac31f run>"}
```

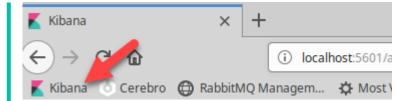
Switch back to the **Agent Terminal** and hit **CTRL** + **C** to stop **Filebeat**. **Filebeat** will not display anything when stopped. Instead, it will simply terminate.

3. View logs from the lab1.1-aggregator_only index using Kibana (search/report system)

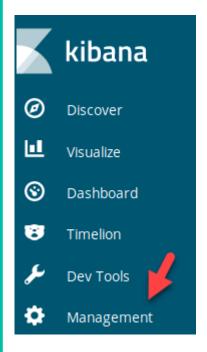


In this section, the **Kibana** is used to view the logs stored in **Elasticsearch**. This demonstrates the ability to search and report on logs once they have been collected.

While still in Firefox, click on the Kibana bookmark.



To see logs from a new index, you must tell **Kibana** about the index. To do this, **click** on **Management**.



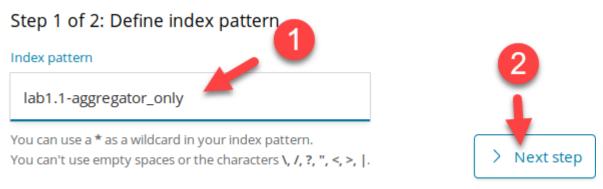
Next, click on Index Patterns.



Then click on Create Index Pattern.



In the Index pattern field, enter the index name of lab1.1-aggregator_only. Then click on Next step.

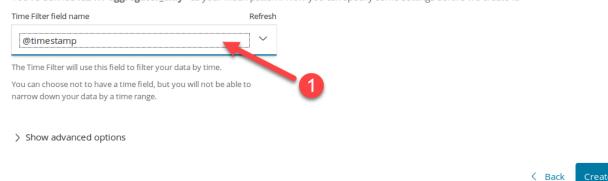


✓ Success! Your index pattern matches 1 index.

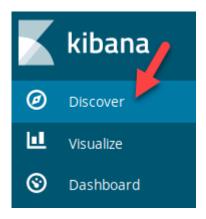
Then select the Time Filter field name and click on @timestamp and click Create index pattern.

Step 2 of 2: Configure settings

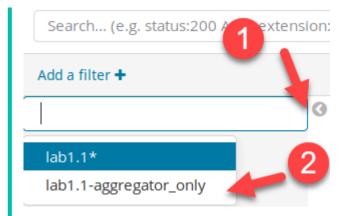
You've defined lab1.1-aggregator_only as your index pattern. Now you can specify some settings before we create it.



Now you can switch back to the **Discover** tab by **clicking** on **Discover**.



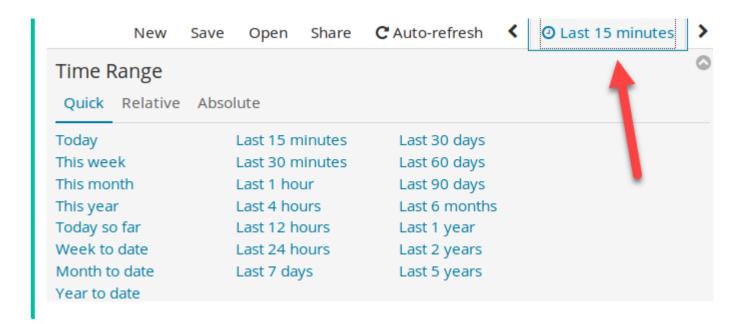
Then select lab1.1-aggregator_only as your index to view the logs.



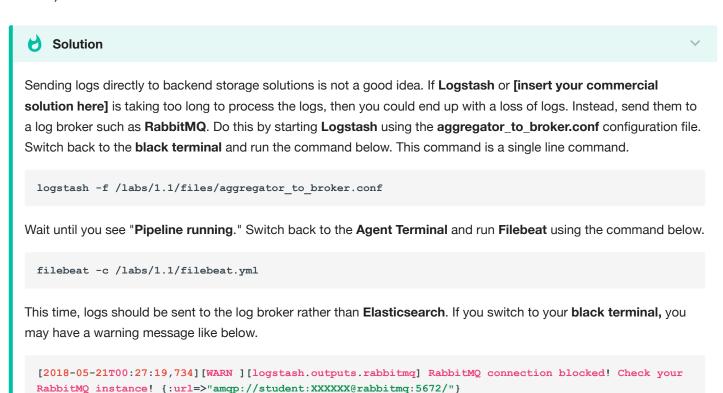
You should now see the logs you have collected.

Note

If you cannot see any logs, it may be that the logs are older than the last 15 minutes. This is the default time span selected in **Kibana**. You can change this by clicking on the **date picker** in the top right corner. This will allow you to pick a longer period such as the **Last 5 years**.



4. Send logs from /var/log/*.log to Logstash using Filebeat and output logs to RabbitMQ (log broker)



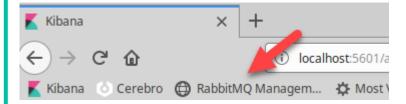
{:url=>"amqp://student:XXXXXX@rabbitmq:5672/"}

[2018-05-21T00:27:28,645] [WARN] [logstash.outputs.rabbitmq] RabbitMQ connection unblocked!



This is nothing to worry about and is a bonus if it happens. What this is demonstrating is that **Filebeat** sent logs extremely rapidly and **RabbitMQ** throttled the connection due to a hard-coded memory limit. This forced **Logstash** to hold temporarily before sending over more logs. In effect, this is buffering, but in production, this would happen at the log aggregator rather than the message queue.

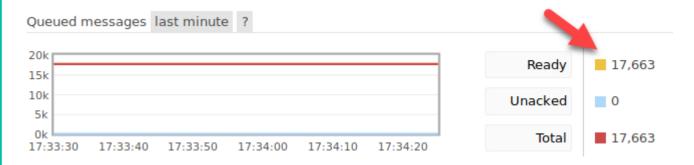
To see that the logs reached **RabbitMQ**, switch back to **Firefox** and click on the bookmark to **RabbitMQ Management**.



Login with the username student and password of sec555.



The home page shows how many logs are currently in the log broker.



In this case, the quantity is **17,663**. This number **may be different** on your system as the configuration file used by **Filebeat** is reading new logs generated in **/var/log/** on your virtual machine. This view is helpful, but it does not break down the logs. Next, **click** on the **Queues** tab.



Overview

Connections

Channels

Exchanges

Oueues

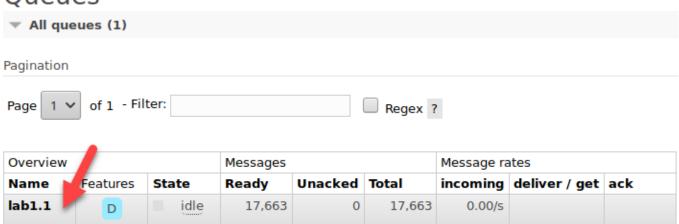
Admin



At this point, logs are sent unparsed. The goal is to get them into a log broker as quickly as possible to avoid bottlenecks.

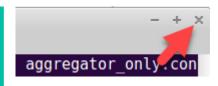
This view shows the total number of logs, # of incoming logs, and # of outgoing logs per queue. In this example, only one queue exists. However, a production log broker may have a queue for Windows logs, firewall logs, and any other logs going through the broker.

Queues



Switch back to the black terminal and hit CTRL + C to stop Logstash.

Switch back to the **Agent Terminal** and hit **CTRL** + **C** to stop **Filebeat**. **Filebeat** will not display anything when stopped. Instead, it will simply terminate. Close out of the **Agent Terminal** by clicking the **X** in the top right corner of the terminal.



5. Use Logstash to pull logs out of RabbitMQ and send them to Elasticsearch in an index called lab1.1-broker



Solution

In this section, **Logstash** is used to pull logs out of **RabbitMQ** so that they can be parsed and enriched. After parsing and enrichment, logs are sent to **Elasticsearch** for storage. This step demonstrates one or more log aggregators pulling logs out of a log broker for processing.

The log broker is a temporary queue. It is not intended to be searched or used other than as a buffer. To pull the logs out of **RabbitMQ**, parse them, and then send them off to **Elasticsearch** use the **Logstash** configuration file called **broker_to_storage.conf**. Do this by running the command below.

```
logstash -f /labs/1.1/files/broker_to_storage.conf
```

Switch back to **Firefox** and look at the **RabbitMQ** queue for **lab1.1**. After a few seconds, it should show the total logs in the queue at **0**.



This means the logs have been retrieved out of the log broker, parsed, and then stored in **Elasticsearch**. You can verify this with **Cerebro** that a new index called **1.1-broker** has been created or you can move on to adding the index to **Kibana**. If you try to add an index to **Kibana** and the index does not exist, it will not let you select a time field.

Switch back to the black terminal and hit CTRL + C to stop Logstash.

```
[2018-05-21T02:30:32,161][WARN ][logstash.runner ] SIGINT received. Shutting down.
[2018-05-21T02:30:32,600][INFO ][logstash.pipeline ] Pipeline has terminated
{:pipeline_id=>"main", :thread=>"#<Thread:0x4aeac31f run>"}
```

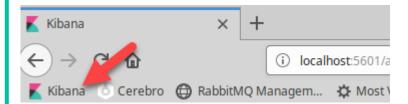
6. View the logs from the lab1.1-broker index using Kibana



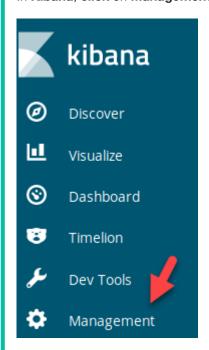
Solution

In this section, the **Kibana** is used to view the logs stored in **Elasticsearch**. This demonstrates the ability to search and report on logs once they have been collected.

Switch to Firefox and click on the Kibana bookmark.



In Kibana, click on Management.



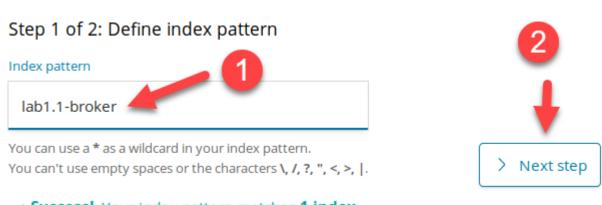
Next, click on Index Patterns.



Next, click on Create Index Pattern.



In the Index pattern field, enter the index name of lab1.1-broker. Then click on Next step.

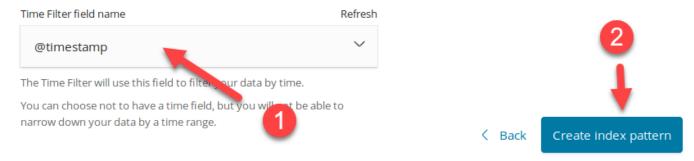


✓ Success! Your index pattern matches 1 index.

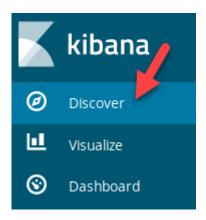
Then select the Time Filter field name and click on @timestamp and click Create index pattern.

Step 2 of 2: Configure settings

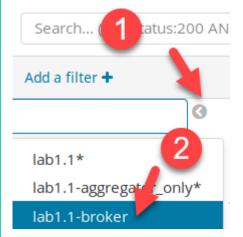
You've defined **lab1.1-broker** as your index pattern. Now you can specify some settings before we create it.



Now you can switch back to the **Discover** tab by **clicking** on **Discover**.



Then select lab1.1-broker as your index to view the logs.



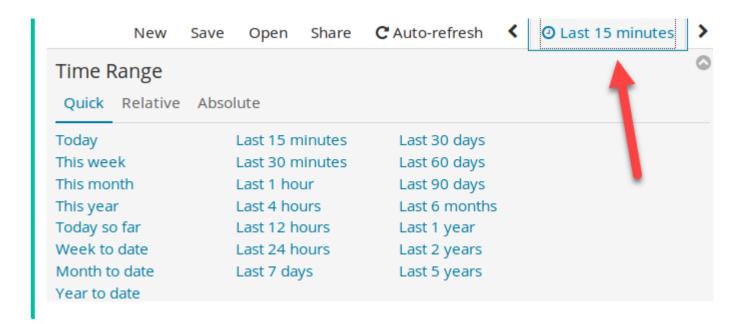
You should now see the logs you have collected.

Time - _source

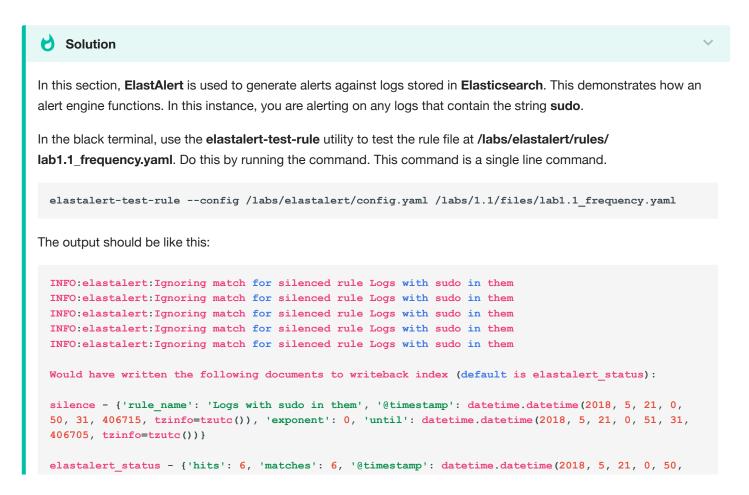
May 20th 2018, 17:27:19.388 beat.hostname: filebeat beat.name: filebeat
urce: /var/log/dpkg.log prospector.type: log
: 1,104,389 tags: beats_input_codec_plain_applitmq @timestamp: May 20th 2018, 17:27:19.388
52:37 status half-configured libc-bin:amd64 2.

Note

If you cannot see any logs, it may be that the logs are older than the last 15 minutes. This is the default time span selected in **Kibana**. You can change this by clicking on the **date picker** in the top right corner. This will allow you to pick a longer period such as the **Last 5 years**.



7. Use ElastAlert (alert engine) to test the rule /labs/1.1/files/lab1.1_frequency.yaml. This is an example rule that triggers an alert if sudo is found in any logs. ElastAlert has a utility called elastalert-test-rule that can be used to test rules.



```
31, 408633, tzinfo=tzutc()), 'rule_name': 'Logs with sudo in them', 'starttime': datetime.datetime(2018, 5, 20, 0, 50, 30, 949354, tzinfo=tzutc()), 'endtime': datetime.datetime(2018, 5, 21, 0, 50, 30, 949354, tzinfo=tzutc()), 'time_taken': 0.4532928466796875}
```

The output above shows the rule would have triggered **6** alerts. This number **may not match** the number on your system as it depends on how many times you have used the command **sudo** on your virtual machine. Scrolling up would show you some of the logs the alert would have triggered on such as this:

```
INFO:elastalert:Alert for Logs with sudo in them at 2018-05-20T20:39:59.544Z:
INFO:elastalert:Logs with sudo in them
At least 1 events occurred between 2018-05-20 20:34 UTC and 2018-05-20 20:39 UTC
@timestamp: 2018-05-20T20:39:59.544Z
@version: 1
id: ByhHf2MBsMHc5WHbxt3t
_index: lab1.1-aggregator_only-complete
 type: doc
beat: {
    "hostname": "filebeat",
    "name": "filebeat",
    "version": "6.2.4"
host: filebeat
message: Unpacking sudo (1.8.21p2-3ubuntu1) ...
num hits: 6
num matches: 6
offset: 46967
prospector: {
    "type": "log"
source: /var/log/bootstrap.log
syslog_facility: user-level
syslog facility code: 1
syslog severity: notice
syslog_severity_code: 5
tags: [
    "beats_input_codec_plain_applied",
    "_grokparsefailure",
    " geoip lookup failure"
]
```

At the end of this lab, stop **RabbitMQ** with the command below.

```
docker stop rabbitmq
```

Lab Conclusion

In this lab, you have experienced the major components and designs of a log pipeline. This included:

- · Sending logs from a log agent
- · A simple log aggregation collection method without a message broker
- A more complex log aggregation collection method using a message broker for added resiliency
- · Using a GUI to access and view logs
- · Interacting with an alert engine

Lab 1.1 is now complete!