ElastAlert2 Alerting Guide

Forward: In this guide I will outline how to create ElastAlert alerts with ElastAlert2 running through Docker. I will go through the process of the first alert we created – the lockout event detection.

Step 1 – Create your configuration file:

The configuration file that we are using is named `elastalert.yaml`. Its contents are as follows:

```
# This is the folder that contains the rule yaml files
# This can also be a list of directories
# Any .yaml file will be loaded as a rule
rules_folder: /etc/elastalert/elastalert_rules

# How often ElastAlert will query Elasticsearch
# The unit can be anything from weeks to seconds
run_every:
  minutes: 1

# ElastAlert will buffer results from the most recent
# period of time, in case some log sources are not in real time
buffer_time:
  minutes: 15

# The Elasticsearch hostname for metadata writeback
# Note that every rule can have its own Elasticsearch host
es_host: 10.100.10.36

# The Elasticsearch port
es_port: 9200

# The AWS region to use. Set this when using AWS-managed elasticsearch
```

```
   #aws_region: us-east-1


# The AWS profile to use. Use this if you are using an aws-cli profile.
# See http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html
# for details
   #profile: test


# Optional URL prefix for Elasticsearch
#   es_url_prefix: elasticsearch


# Optional prefix for statsd metrics
statsd_instance_tag: elastalert


# Optional statsd host
   #statsd_host: dogstatsd


# Connect with TLS to Elasticsearch
use_ssl: False


# Verify TLS certificates
verify_certs: False


# Show TLS or certificate related warnings
ssl_show_warn: True


# GET request with body is the default option for Elasticsearch.
# If it fails for some reason, you can pass 'GET', 'POST' or 'source'.
# See http://elasticsearch-py.readthedocs.io/en/master/connection.html?highlight=send_get_body_as#transport
# for details
es_send_get_body_as: GET
```

```
# Option basic-auth username and password for Elasticsearch
es_username: elastic
es_password: changeme

# Use SSL authentication with client certificates client_cert must be
# a pem file containing both cert and key for client
   #ca_certs: /path/to/cacert.pem
   #client_cert: /path/to/client_cert.pem
   #client_key: /path/to/client_key.key

# The index on es_host which is used for metadata storage
# This can be a unmapped index, but it is recommended that you run
# elastalert-create-index to set a mapping
writeback_index: elastalert_status

# If an alert fails for some reason, ElastAlert will retry
# sending the alert until this time period has elapsed
alert_time_limit:
  days: 2

# Optional timestamp format.
# ElastAlert will print timestamps in alert messages and in log messages using
this format.
   #custom_pretty_ts_format: '%Y-%m-%d %H:%M'

# Custom logging configuration
#
#If you want to setup your own logging configuration to log into
# files as well or to Logstash and/or modify log levels, use
# the configuration below and adjust to your needs.
# Note: if you run ElastAlert with --verbose/--debug, the log level of
# the "elastalert" logger is changed to INFO, if not already INFO/DEBUG.
```

```
logging:
  version: 1
  incremental: false
  disable_existing_loggers: false
  formatters:
    logline:
      format: '%(asctime)s %(levelname)+8s %(name)+20s %(message)s'

  handlers:
    console:
      class: logging.StreamHandler
      formatter: logline
      level: DEBUG
      stream: ext://sys.stderr

  loggers:
    elastalert:
      level: DEBUG
      handlers: []
      propagate: true

    elasticsearch:
      level: DEBUG
      handlers: []
      propagate: true

    elasticsearch.trace:
      level: DEBUG
      handlers: []
      propagate: true

    '':  # root logger
```

```
    level: DEBUG
    handlers:
      - console
    propagate: false
```

You can check the docker container logs to see if the configuration file above loads without issue (if there are no problems the Elastalert indices will be created). You can also look for any errors that may be associated with the config. A common error that I have seen is: `yaml.parser.ParserError`. A tool that can help with debugging parser errors is here: https://yaml-online-parser.appspot.com/.

Step 2 – Create your rules directory and rules file:

The rules file should be in a directory dedicated only to rule files. This is because it allows us to run multiple different rule files at the same time. ElastAlert will read all files in the specified rules directory. This is the current rule file (titled `rule.yaml`) that we are using for the lockout alerting:

```
name: ElastAlert Rules File
type: any
num_events: 1
filter:
- query:
    query_string:
      query: "\"A user account was locked out\""
index: winlogbeat-*
realert:
  minutes: 0
query_key:
  - source.ip
include:
  - message
include_match_in_root: true
alert_subject: "Lockout Event Detected from User"
alert_subject_args:
```

```
  - winlog.event_dataTargetUserName
alert_text: |-
  Lockout Event detected.
  Details of the event:
          - message: {0}
alert_text_args:
  - message
is_enabled: true
alert:
  - email:
     from_addr: "relay2@uhtasi.org"
     email: "info@uhtasi.org"
     smtp_host: "mail.uhtasi.org"
     smtp_port: 587
     smtp_ssl: false
     smtp_auth_file: "/etc/elastalert/smtp_auth_user.yaml"
alert_text_type: alert_text_only
```

This rule file searches winlogbeat logs that contain the string “A user account was locked out” and filters for only those logs. The email text is also utilizing the `alert_text` field which uses data from the `message` field in the logs. Note that you do not need to create a new container every time you adjust the rule file. Therefore you can keep the original running container and still change the volume that is linked to that container without issue.

For more info on ElastAlert rules and configuration, look here:

https://elastalert.readthedocs.io/en/latest/ruletypes.html

Step 3 – Create the SMTP authentication file:

The SMTP auth file is needed to access the TASI email. Here are the contents:

```
user: ********
password: ********
```

Step 4 – Execute the *docker run* command:

This command creates the container and starts the ElastAlert process. The command must be executed in the `/etc/elastalert` directory because of the use of `pwd` in the command. Here is the command:

```
docker run -d --name elastalert2 --restart=always
  -v $(pwd)/elastalert.yaml:/opt/elastalert/config.yaml
  -v $(pwd)/elastalert_rules:/etc/elastalert/elastalert_rules
  -v $(pwd)/smtp_auth_user.yaml:/etc/elastalert/smtp_auth_user.yaml
  ghcr.io/jertel/elastalert2/elastalert2 --verbose
```

Note that the `-v` tag is need for each file being used in the ElastAlert implementation (for example files mentioned in the configuration files). After running this command, ElastAlert should be running on the VM!

Step 5 – Docker container maintenance and log checking:

The command `docker ps` will display the statuses of all containers on the VM. Use the command `docker logs <container name>` to check the ElastAlert rule queries (if `--verbose` has been enabled in `docker run`) as well as any errors. To stop a container, run `docker stop <container name>.` To delete a container permanently, run `docker rm <container name>` to remove the individual container or `docker container prune` which will remove all stopped containers on the VM.

Tips/Tricks – Run this command to search for specific ElastAlert logs in the docker container. It is useful for finding specific errors/alerts when the amount of logs is large:

```
docker logs <container name> 2>&1 | grep "your text here"
```

Additionally, run this command to obtain only the most recent log lines (because sometimes the amount of total logs is extremely long):

`docker logs --tail n <container name>` (where *n* is the number of log lines to print)

You can also combine the two commands mentioned above like so:

```
docker logs --tail n <container name> 2>&1 | grep "your text here"
```

If you want to enter the container and view the files that are mounted into the container, run this command:

```
docker exec -it <container name> bash
```

This will open an interactive bash session in the terminal where you can browse files, run commands, etc belonging to the container. This is useful for seeing what files your container is running. Note that you have to specify the paths for the commands while in the container (`/bin/cat` for example).

Step 6 – View ElastAlert index documents in Kibana:

In the Kibana Dev Tools run this to print all documents from an index (here I am using the `elastalert_status` index as an example):

```
GET elastalert_status/_search
{
  “query”: {
    “match_all”: {}
  }
}
```

To search for a specific document, you can attempt to query for it. Here I am searching for documents that contain `yaml.parser.ParserError`:

```
GET elastalert_error/_search
{
  “query”: {
    “query_string”: {
      “query”: “yaml.parser.ParserError”
            }
      }
}
```

For more info on ElastAlert indices, look here:
https://elastalert.readthedocs.io/en/latest/elastalert_status.html

Remark on index memory/storage management: Currently it is not possible to set an ILM policy for ElastAlert indices. Therefore it is required to manually delete the indices once they grow large and then run a new Docker container to re-create the indices. Here is a recent thread where this is mentioned:

https://github.com/jertel/elastalert2/discussions/713

Hopefully this is a feature that ElastAlert will eventually add.

Update! I have thought of two potential work-arounds to the issue brought up above. The first is to write a script that utilizes Kibana API commands to delete ElastAlert index documents by timestamp. This is the command that would be used:

```
curl -XDELETE
'elasticsearch.example.com:14900/elastalert_metadata/elastalert,elastalert_status
,silence,elastalert_error/_query' -d '{"query": {"range": {"@timestamp": {"lte":
"2015-07-13"}}}}'
```

The next solution would be to utilize the `--entrypoint` flag. This will completely skip the Elastalert index generation upon the creation of the container. Here is the command.

```
docker run -v <specify binds as documented> --entrypoint=elastalert
jertel/elastalert2:2.3.0 --config /opt/elastalert/config.yaml <any optional args>
```

Step 7 – Debugging ElastAlert:

Run this command to start ElastAlert in debug mode:

```
docker run -d --name elastalert2-debug --restart=always
  -v $(pwd)/elastalert.yaml:/opt/elastalert/config.yaml
  -v $(pwd)/elastalert_rules:/etc/elastalert/elastalert_rules
  -v $(pwd)/smtp_auth_user.yaml:/etc/elastalert/smtp_auth_user.yaml
  ghcr.io/jertel/elastalert2/elastalert2 --debug
```

The debug mode offers more granular logs. It also allows us to see the query made to Elasticsearch as well as the response from Elasticsearch. This can help with solving the `<MISSING VALUE>` error in our alerts because it lets us view the fields that Elasticsearch is actually using.

Here are some `DEBUG` log examples:

Initial Connection:

```
2022-02-18 23:45:09,472    DEBUG urllib3.connectionpool Starting new HTTP
connection (1): 10.100.10.36:9200
2022-02-18 23:45:10,422    DEBUG urllib3.connectionpool http://10.100.10.36:9200
"GET / HTTP/1.1" 200 552
2022-02-18 23:45:10,423     INFO        elasticsearch GET
http://10.100.10.36:9200/ [status:200 request:0.956s]
2022-02-18 23:45:10,424    DEBUG        elasticsearch > None
2022-02-18 23:45:10,424    DEBUG        elasticsearch < {
  "name" : "ELK-Stack.uhtasi.local",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "4KN6qWZWSF-TSgISyBmbqQ",
  "version" : {
    "number" : "7.16.1",
    "build_flavor" : "default",
    "build_type" : "rpm",
    "build_hash" : "5b38441b16b1ebb16a27c107a4c3865776e20c53",
```

```
    "build_date" : "2021-12-11T00:29:38.865893768Z",
    "build_snapshot" : false,
    "lucene_version" : "8.10.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```


** I decided to include the initial connection debug because it could help with debugging issues involving connections to the Elasticsearch cluster. The debug example above shows a successful connection.

Low-Priority IPS Query:


```
2022-02-18 23:30:25,833    DEBUG        elasticsearch >
{"query":{"bool":{"filter":{"bool":{"must":[{"range":{"@timestamp":{"gt":"2022-
02-18T23:15:25.765064Z","lte":"2022-02-
18T23:30:25.765064Z"}}},{"query_string":{"query":"\"Priority:
Low\""}}]}}}},"sort":[{"@timestamp":{"order":"asc"}}]}

2022-02-18 23:30:25,833    DEBUG        elasticsearch <
{"_scroll_id":"FGluY2x1ZGVfY29udGV4dF91dWlkDnF1ZXJ5VGhlbkZldGNoARZxMkhzTnROclFBT2
pzX01tUFM5R3N3AAAAAAFzWlUWTTNtcjR4eHdRRXl5ekVWNEptMEgzZw==","took":23,"timed_out"
:false,"_shards":{"total":2,"successful":2,"skipped":1,"failed":0},"hits":{"total
":{"value":2,"relation":"eq"},"max_score":null,"hits":[{"_index":"logstash-
2022.02.18-
000419","_type":"_doc","_id":"XFQgD38BQXFqgSjg18JP","_score":null,"_source":{"@ti
mestamp":"2022-02-18T23:18:03.684Z"},"sort":[1645226283684]},{"_index":"logstash-
2022.02.18-
000419","_type":"_doc","_id":"GVQjD38BQXFqgSjgYPxU","_score":null,"_source":{"@ti
mestamp":"2022-02-18T23:20:50.756Z"},"sort":[1645226450756]}]}}
```


Above we can see the query sent to Elasticsearch as well as Elasticsearch's response. The response shows why we are receiving `<MISSING VALUE>` in our alerts. The desired fields are simply not in the response.

Lockout Query:


```
2022-02-22 18:42:48,894    DEBUG        elasticsearch >
{"query":{"bool":{"filter":{"bool":{"must":[{"range":{"@timestamp":{"gt":"2022-
02-22T18:27:48.820360Z","lte":"2022-02-
22T18:42:48.820360Z"}}},{"query_string":{"query":"\"A user account was locked
out\""}}]}}}},"sort":[{"@timestamp":{"order":"asc"}}]}

2022-02-22 18:42:48,895    DEBUG        elasticsearch <
{"_scroll_id":"FGluY2x1ZGVfY29udGV4dF91dWlkDnF1ZXJ5VGhlbkZldGNoARZxMkhzTnROclFBT2
pzX01tUFM5R3N3AAAAAAG29ikWTTNtcjR4eHdRRXl5ekVWNEptMEgzZw==","took":66,"timed_out"
:false,"_shards":{"total":2,"successful":2,"skipped":1,"failed":0},"hits":{"total
```

":{"value":1,"relation":"eq"},"max_score":null,"hits":[{"_index":"winlogbeat-7.10.0-2022.02.22-000251","_type":"_doc","_id":"d1OrIn8BQXFqgSjgZ93Z","_score":null,"_source":{"@timestamp":"2022-02-22T18:31:28.864Z","message":"A user account was locked out.\n\nSubject:\n\tSecurity ID:\t\tS-1-5-18\n\tAccount Name:\t\tDCON3$\n\tAccount Domain:\t\tUHTASI\n\tLogon ID:\t\t0x3E7\n\nAccount That Was Locked Out:\n\tSecurity ID:\t\tS-1-5-21-3693063922-990322624-829951171-1332\n\tAccount Name:\t\tadmin\n\nAdditional Information:\n\tCaller Computer Name:\tDCON3"},"sort":[1645554688864]}]}}

The main difference between the query above and the IPS query is that we are receiving a field that we can actually use in the alerts – the `message` field – as shown in the Elasticsearch response.

Now, how do we actually get the desired fields we want into the Elasticsearch response? Unfortunately, I currently do not have a solution. I posted this question to various online forums so you can check that here:

https://github.com/jertel/elastalert2/discussions/734

https://discuss.elastic.co/t/elasticsearch-response-does-not-have-desired-fields-data/298237