ElastAlert 2 Documentation

Release 0.0.1

Quentin Long

Aug 08, 2022

CONTENTS

Contents:

CHAPTER

ONE

ELASTALERT 2 - AUTOMATED RULE-BASED ALERTING FOR ELASTICSEARCH

ElastAlert 2 is a simple framework for alerting on anomalies, spikes, or other patterns of interest from data in Elasticsearch and OpenSearch.

If you have data being written into Elasticsearch in near real time and want to be alerted when that data matches certain patterns, ElastAlert 2 is the tool for you.

1.1 Overview

We designed ElastAlert 2 to be *reliable*, highly *modular*, and easy to *set up* and *configure*.

It works by combining Elasticsearch with two types of components, rule types and alerts. Elasticsearch is periodically queried and the data is passed to the rule type, which determines when a match is found. When a match occurs, it is given to one or more alerts, which take action based on the match.

This is configured by a set of rules, each of which defines a query, a rule type, and a set of alerts.

Several rule types with common monitoring paradigms are included with ElastAlert 2:

- "Match where there are X events in Y time" (`frequency` type)
- "Match when the rate of events increases or decreases" (`spike` type)
- "Match when there are less than X events in Y time" (`flatline` type)
- "Match when a certain field matches a blacklist/whitelist" (`blacklist` and `whitelist` type)
- "Match on any event matching a given filter" (`any` type)
- "Match when a field has two different values within some time" (`change` type)

Currently, we have support built in for these alert types:

- Alerta
- Alertmanager
- AWS SES (Amazon Simple Email Service)
- AWS SNS (Amazon Simple Notification Service)
- Chatwork
- Command
- Datadog
- Debug

- Dingtalk
- Discord
- Email
- Exotel
- Gitter
- GoogleChat
- HTTP POST
- HTTP POST 2
- Jira
- Line Notify
- Mattermost
- Microsoft Teams
- OpsGenie
- PagerDuty
- PagerTree
- Rocket.Chat
- Squadcast
- ServiceNow
- Slack
- Splunk On-Call (Formerly VictorOps)
- Stomp
- Telegram
- Tencent SMS
- TheHive
- Twilio
- Zabbix

Additional rule types and alerts can be easily imported or written. (See *Writing rule types* and *Writing alerts*)

In addition to this basic usage, there are many other features that make alerts more useful:

- Alerts link to Kibana Discover searches
- Aggregate counts for arbitrary fields
- Combine alerts into periodic reports
- Separate alerts by using a unique key field
- Intercept and enhance match data

To get started, check out *Running ElastAlert 2 For The First Time*.

1.2 Reliability

ElastAlert 2 has several features to make it more reliable in the event of restarts or Elasticsearch unavailability:

- ElastAlert 2 *saves its state to Elasticsearch* and, when started, will resume where previously stopped
- If Elasticsearch is unresponsive, ElastAlert 2 will wait until it recovers before continuing
- Alerts which throw errors may be automatically retried for a period of time

1.3 Modularity

ElastAlert 2 has three main components that may be imported as a module or customized:

1.3.1 Rule types

The rule type is responsible for processing the data returned from Elasticsearch. It is initialized with the rule configuration, passed data that is returned from querying Elasticsearch with the rule's filters, and outputs matches based on this data. See *Writing rule types* for more information.

1.3.2 Alerts

Alerts are responsible for taking action based on a match. A match is generally a dictionary containing values from a document in Elasticsearch, but may contain arbitrary data added by the rule type. See *Writing alerts* for more information.

1.3.3 Enhancements

Enhancements are a way of intercepting an alert and modifying or enhancing it in some way. They are passed the match dictionary before it is given to the alerter. See *Enhancements* for more information.

1.4 Configuration

ElastAlert 2 has a global configuration file, `config.yaml`, which defines several aspects of its operation:

`buffer_time`: ElastAlert 2 will continuously query against a window from the present to `buffer_time` ago. This way, logs can be back filled up to a certain extent and ElastAlert 2 will still process the events. This may be overridden by individual rules. This option is ignored for rules where `use_count_query` or `use_terms_query` is set to true. Note that back filled data may not always trigger count based alerts as if it was queried in real time.

`es_host`: The host name of the Elasticsearch cluster where ElastAlert 2 records metadata about its searches. When ElastAlert 2 is started, it will query for information about the time that it was last run. This way, even if ElastAlert 2 is stopped and restarted, it will never miss data or look at the same events twice. It will also specify the default cluster for each rule to run on. The environment variable `ES_HOST` will override this field. For multiple host Elasticsearch clusters see `es_hosts` parameter.

`es_port`: The port corresponding to `es_host`. The environment variable `ES_PORT` will override this field.

`es_hosts` is the list of addresses of the nodes of the Elasticsearch cluster. This parameter can be used for high availability purposes, but the primary host must also be specified in the `es_host` parameter. The `es_hosts` parameter can be

overridden within each rule. This value can be specified as `host:port` if overriding the default port. The environment variable `ES_HOSTS` will override this field, and can be specified as a comma-separated value to denote multiple hosts.

`use_ssl`: Optional; whether or not to connect to `es_host` using TLS; set to `True` or `False`. The environment variable `ES_USE_SSL` will override this field.

`verify_certs`: Optional; whether or not to verify TLS certificates; set to `True` or `False`. The default is `True`.

`ssl_show_warn`: Optional; suppress TLS and certificate related warnings; set to `True` or `False`. The default is `True`.

`client_cert`: Optional; path to a PEM certificate to use as the client certificate.

`client_key`: Optional; path to a private key file to use as the client key.

`ca_certs`: Optional; path to a CA cert bundle to use to verify SSL connections

`es_username`: Optional; basic-auth username for connecting to `es_host`. The environment variable `ES_USERNAME` will override this field.

`es_password`: Optional; basic-auth password for connecting to `es_host`. The environment variable `ES_PASSWORD` will override this field.

`es_bearer`: Optional; Bearer token for connecting to `es_host`. The environment variable `ES_BEARER` will override this field. This authentication option will override the password authentication option.

`es_api_key`: Optional; Base64 api-key token for connecting to `es_host`. The environment variable `ES_API_KEY` will override this field. This authentication option will override both the bearer and the password authentication options.

`es_url_prefix`: Optional; URL prefix for the Elasticsearch endpoint. The environment variable `ES_URL_PREFIX` will override this field.

`es_send_get_body_as`: Optional; Method for querying Elasticsearch - `GET`, `POST` or `source`. The default is `GET`

`es_conn_timeout`: Optional; sets timeout for connecting to and reading from `es_host`; defaults to `20`.

`rules_loader`: Optional; sets the loader class to be used by ElastAlert 2 to retrieve rules and hashes. Defaults to `FileRulesLoader` if not set.

`rules_folder`: The name of the folder or a list of folders which contains rule configuration files. ElastAlert 2 will load all files in this folder, and all subdirectories, that end in .yaml. If the contents of this folder change, ElastAlert 2 will load, reload or remove rules based on their respective config files. (only required when using `FileRulesLoader`).

`scan_subdirectories`: Optional; Sets whether or not ElastAlert 2 should recursively descend the rules directory - `true` or `false`. The default is `true`

`run_every`: How often ElastAlert 2 should query Elasticsearch. ElastAlert 2 will remember the last time it ran the query for a given rule, and periodically query from that time until the present. The format of this field is a nested unit of time, such as `minutes: 5`. This is how time is defined in every ElastAlert 2 configuration.

`misfire_grace_time`: If the rule scheduler is running behind, due to large numbers of rules or long-running rules, this grace time settings allows a rule to still be executed, provided its next scheduled runt time is no more than this grace period, in seconds, overdue. The default is 5 seconds.

`writeback_index`: The index on `es_host` to use.

`max_query_size`: The maximum number of documents that will be downloaded from Elasticsearch in a single query. The default is 10,000, and if you expect to get near this number, consider using `use_count_query` for the rule. If this limit is reached, ElastAlert 2 will scroll using the size of `max_query_size` through the set amount of pages, when `max_scrolling_count` is set or until processing all results.

`max_scrolling_count`: The maximum amount of pages to scroll through. The default is `990`, to avoid a stack overflow error due to Python's stack limit of 1000. For example, if this value is set to `5` and the `max_query_size` is set to `10000` then `50000` documents will be downloaded at most.

`max_threads`: The maximum number of concurrent threads available to process scheduled rules. Large numbers of long-running rules may require this value be increased, though this could overload the Elasticsearch cluster if too many complex queries are running concurrently. Default is 10.

`scroll_keepalive`: The maximum time (formatted in Time Units) the scrolling context should be kept alive. Avoid using high values as it abuses resources in Elasticsearch, but be mindful to allow sufficient time to finish processing all the results.

`max_aggregation`: The maximum number of alerts to aggregate together. If a rule has `aggregation` set, all alerts occuring within a timeframe will be sent together. The default is 10,000.

`old_query_limit`: The maximum time between queries for ElastAlert 2 to start at the most recently run query. When ElastAlert 2 starts, for each rule, it will search `elastalert_metadata` for the most recently run query and start from that time, unless it is older than `old_query_limit`, in which case it will start from the present time. The default is one week.

`disable_rules_on_error`: If true, ElastAlert 2 will disable rules which throw uncaught (not EAException) exceptions. It will upload a traceback message to `elastalert_metadata` and if `notify_email` is set, send an email notification. The rule will no longer be run until either ElastAlert 2 restarts or the rule file has been modified. This defaults to True.

`show_disabled_rules`: If true, ElastAlert 2 show the disable rules' list when finishes the execution. This defaults to True.

`notify_email`: An email address, or list of email addresses, to which notification emails will be sent. Currently, only an uncaught exception will send a notification email. The from address, SMTP host, and reply-to header can be set using `from_addr`, `smtp_host`, and `email_reply_to` options, respectively. By default, no emails will be sent.

single address example:

```
notify_email: "one@domain"
```

or

multiple address example:

```
notify_email:
    - "one@domain"
    - "two@domain"
```

`from_addr`: The address to use as the from header in email notifications. This value will be used for email alerts as well, unless overwritten in the rule config. The default value is "ElastAlert".

`smtp_host`: The SMTP host used to send email notifications. This value will be used for email alerts as well, unless overwritten in the rule config. The default is "localhost".

`email_reply_to`: This sets the Reply-To header in emails. The default is the recipient address.

`aws_region`: This makes ElastAlert 2 to sign HTTP requests when using Amazon OpenSearch Service. It'll use instance role keys to sign the requests. The environment variable `AWS_DEFAULT_REGION` will override this field.

`profile`: AWS profile to use when signing requests to Amazon OpenSearch Service, if you don't want to use the instance role keys. The environment variable `AWS_DEFAULT_PROFILE` will override this field.

`replace_dots_in_field_names`: If `True`, ElastAlert 2 replaces any dots in field names with an underscore before writing documents to Elasticsearch. The default value is `False`. Elasticsearch 2.0 - 2.3 does not support dots in field names.

`string_multi_field_name`: If set, the suffix to use for the subfield for string multi-fields in Elasticsearch. The default value is `.keyword`.

add_metadata_alert: If set, alerts will include metadata described in rules (category, description, owner and priority); set to True or False. The default is False.

skip_invalid: If True, skip invalid files instead of exiting.

jinja_root_name: When using a Jinja template, specify the name of the root field name in the template. The default is _data.

jinja_template_path: When using a Jinja template, specify filesystem path to template, this overrides the default behaviour of using alert_text as the template.

custom_pretty_ts_format: This option provides a way to define custom format of timestamps printed in log messages and in alert messages. If this option is not set, default timestamp format ('%Y-%m-%d %H:%M %Z') will be used. (Optional, string, default None)

Example usage and resulting formatted timestamps:

```
(not set; default)                              -> '2021-08-16 21:38 JST'
custom_pretty_ts_format: '%Y-%m-%d %H:%M %z'    -> '2021-08-16 21:38 +0900'
custom_pretty_ts_format: '%Y-%m-%d %H:%M'       -> '2021-08-16 21:38'
```

1.4.1 Logging

By default, ElastAlert 2 uses a simple basic logging configuration to print log messages to standard error. You can change the log level to INFO messages by using the --verbose or --debug command line options.

If you need a more sophisticated logging configuration, you can provide a full logging configuration in the config file. This way you can also configure logging to a file, to Logstash and adjust the logging format.

For details, see the end of examples/config.yaml.example where you can find an example logging configuration.

CHAPTER

TWO

RUNNING ELASTALERT 2

ElastAlert 2 can easily be run as *a Docker container* or directly on your machine as *a Python package*. If you are not interested in modifying the internals of ElastAlert 2, the Docker container is recommended for ease of use.

2.1 Configuration flags

However you choose to run ElastAlert 2, the ElastAlert 2 process is started by invoking `python elastalert/elastalert.py`.

This command accepts several configuration flags:

`--config` will specify the configuration file to use. The default is `config.yaml`. See *here* to understand what behaviour can be configured in this file.

`--debug` will run ElastAlert 2 in debug mode. This will increase the logging verboseness, change all alerts to `DebugAlerter`, which prints alerts and suppresses their normal action, and skips writing search and alert metadata back to Elasticsearch. Not compatible with *–verbose*.

`--end <timestamp>` will force ElastAlert 2 to stop querying after the given time, instead of the default, querying to the present time. This really only makes sense when running standalone. The timestamp is formatted as `YYYY-MM-DDTHH:MM:SS` (UTC) or with timezone `YYYY-MM-DDTHH:MM:SS-XX:00` (UTC-XX).

`--es_debug` will enable logging for all queries made to Elasticsearch.

`--es_debug_trace <trace.log>` will enable logging curl commands for all queries made to Elasticsearch to the specified log file. `--es_debug_trace` is passed through to elasticsearch.py which logs *localhost:9200* instead of the actual `es_host:es_port`.

`--pin_rules` will stop ElastAlert 2 from loading, reloading or removing rules based on changes to their config files.

`--prometheus_port` exposes ElastAlert 2 Prometheus metrics on the specified port. Prometheus metrics disabled by default.

`--rule <rule.yaml>` will only run the given rule. The rule file may be a complete file path or a filename in `rules_folder` or its subdirectories.

`--silence <unit>=<number>` will silence the alerts for a given rule for a period of time. The rule must be specified using `--rule`. <unit> is one of days, weeks, hours, minutes or seconds. <number> is an integer. For example, `--rule noisy_rule.yaml --silence hours=4` will stop noisy_rule from generating any alerts for 4 hours.

`--silence_qk_value <value` will silence the rule only for the given query key value. This parameter is intended to be used with the `--rule` parameter.

`--start <timestamp>` will force ElastAlert 2 to begin querying from the given time, instead of the default, querying from the present. The timestamp should be ISO8601, e.g. `YYYY-MM-DDTHH:MM:SS` (UTC) or with timezone

`YYYY-MM-DDTHH:MM:SS-08:00` (PST). Note that if querying over a large date range, no alerts will be sent until that rule has finished querying over the entire time period. To force querying from the current time, use "NOW".

`--verbose` will increase the logging verboseness, which allows you to see information about the state of queries. Not compatible with *–debug*.

2.2 As a Docker container

If you're interested in a pre-built Docker image check out the elastalert2 container image on Docker Hub or GitHub Container Registry. Both images are published for each release. Use GitHub Container Registry if you are running into Docker Hub usage limits.

Be aware that the `latest` tag of the image represents the latest commit into the master branch. If you prefer to upgrade more slowly you will need utilize a versioned tag, such as `2.6.0` instead, or `2` if you are comfortable with always using the latest released version of ElastAlert 2.

A properly configured config.yaml file must be mounted into the container during startup of the container. Use the example file as a template.

The following example assumes Elasticsearch container has already been started with Docker. This example also assumes both the Elasticsearch and ElastAlert2 containers are using the default Docker network: `es_default`

Create a rule directory and rules file in addition to elastalert.yaml, and then mount both into the ElastAlert 2 container:

```
elastalert.yaml
rules/
  a.yaml
```

elastalert.yaml

```
rules_folder: /opt/elastalert/rules

run_every:
  seconds: 10

buffer_time:
  minutes: 15

es_host: elasticsearch
es_port: 9200

writeback_index: elastalert_status

alert_time_limit:
  days: 2
```

a.yaml

```
name: "a"
type: "frequency"
index: "mariadblog-*"
is_enabled: true
num_events: 2
realert:
  minutes: 5
```

(continues on next page)

(continued from previous page)

```
terms_size: 50
timeframe:
  minutes: 5
timestamp_field: "@timestamp"
timestamp_type: "iso"
use_strftime_index: false
alert_subject: "Test {} 123 aa"
alert_subject_args:
  - "message"
  - "@log_name"
alert_text: "Test {}  123 bb"
alert_text_args:
  - "message"
filter:
  - query:
      query_string:
        query: "@timestamp:*"
alert:
  - "slack"
slack_webhook_url: 'https://hooks.slack.com/services/xxxxxxxxx'
slack_channel_override: "#abc"
slack_emoji_override: ":kissing_cat:"
slack_msg_color: "warning"
slack_parse_override: "none"
slack_username_override: "elastalert"
```

Starting the container via Docker Hub (hub.docker.com)

```
docker run --net=es_default -d --name elastalert --restart=always \
-v $(pwd)/elastalert.yaml:/opt/elastalert/config.yaml \
-v $(pwd)/rules:/opt/elastalert/rules \
jertel/elastalert2 --verbose

docker logs -f elastalert
```

Starting the container via GitHub Container Registry (ghcr.io)

```
docker run --net=es_default -d --name elastalert --restart=always \
-v $(pwd)/elastalert.yaml:/opt/elastalert/config.yaml \
-v $(pwd)/rules:/opt/elastalert/rules \
ghcr.io/jertel/elastalert2/elastalert2 --verbose

docker logs -f elastalert
```

For developers, the below command can be used to build the image locally:

```
docker build . -t elastalert2
```

2.3 As a Kubernetes deployment

The Docker container for ElastAlert 2 can be used directly as a Kubernetes deployment, but for convenience, a Helm chart is also available. See the instructions provided on Github for more information on how to install, configure, and run the chart.

2.4 As a Python package

2.4.1 Requirements

- Elasticsearch 7.x or 8.x, or OpenSearch 1.x or 2.x
- ISO8601 or Unix timestamped data
- Python 3.10. Require OpenSSL 1.1.1 or newer.
- pip
- Packages on Ubuntu 21.x: build-essential python3-pip python3.10 python3.10-dev libffi-dev libssl-dev

If you want to install python 3.10 on CentOS, please install python 3.10 from the source code after installing 'Development Tools'.

2.4.2 Downloading and Configuring

You can either install the latest released version of ElastAlert 2 using pip:

```
$ pip install elastalert2
```

or you can clone the ElastAlert2 repository for the most recent changes:

```
$ git clone https://github.com/jertel/elastalert2.git
```

Install the module:

```
$ pip install "setuptools>=11.3"
$ python setup.py install
```

Next, open up `examples/config.yaml.example`. In it, you will find several configuration options. ElastAlert 2 may be run without changing any of these settings.

`rules_folder` is where ElastAlert 2 will load rule configuration files from. It will attempt to load every .yaml file in the folder. Without any valid rules, ElastAlert 2 will not start. ElastAlert 2 will also load new rules, stop running missing rules, and restart modified rules as the files in this folder change. For this tutorial, we will use the `examples/rules` folder.

`run_every` is how often ElastAlert 2 will query Elasticsearch.

`buffer_time` is the size of the query window, stretching backwards from the time each query is run. This value is ignored for rules where `use_count_query` or `use_terms_query` is set to true.

`es_host` is the primary address of an Elasticsearch cluster where ElastAlert 2 will store data about its state, queries run, alerts, and errors. Each rule may also use a different Elasticsearch host to query against. For multiple host Elasticsearch clusters see `es_hosts` parameter.

`es_port` is the port corresponding to `es_host`.

`es_hosts` is the list of addresses of the nodes of the Elasticsearch cluster. This parameter can be used for high availability purposes, but the primary host must also be specified in the `es_host` parameter. The `es_hosts` parameter can be overridden within each rule. This value can be specified as `host:port` if overriding the default port.

`use_ssl`: Optional; whether or not to connect to `es_host` using TLS; set to `True` or `False`.

`verify_certs`: Optional; whether or not to verify TLS certificates; set to `True` or `False`. The default is `True`

`ssl_show_warn`: Optional; suppress TLS and certificate related warnings; set to `True` or `False`. The default is `True`.

`client_cert`: Optional; path to a PEM certificate to use as the client certificate

`client_key`: Optional; path to a private key file to use as the client key

`ca_certs`: Optional; path to a CA cert bundle to use to verify SSL connections

`es_username`: Optional; basic-auth username for connecting to `es_host`.

`es_password`: Optional; basic-auth password for connecting to `es_host`.

`es_bearer`: Optional; bearer token authorization for connecting to `es_host`. If bearer token is specified, login and password are ignored.

`es_url_prefix`: Optional; URL prefix for the Elasticsearch endpoint.

`statsd_instance_tag`: Optional; prefix for statsd metrics.

`statsd_host`: Optional; statsd host.

`es_send_get_body_as`: Optional; Method for querying Elasticsearch - `GET`, `POST` or `source`. The default is `GET`

`writeback_index` is the name of the index in which ElastAlert 2 will store data. We will create this index later.

`alert_time_limit` is the retry window for failed alerts.

Save the file as `config.yaml`

2.4.3 Setting Up Elasticsearch

ElastAlert 2 saves information and metadata about its queries and its alerts back to Elasticsearch. This is useful for auditing, debugging, and it allows ElastAlert 2 to restart and resume exactly where it left off. This is not required for ElastAlert 2 to run, but highly recommended.

First, we need to create an index for ElastAlert 2 to write to by running `elastalert-create-index` and following the instructions. Note that this manual step is only needed by users that run ElastAlert 2 directly on the host, whereas container users will automatically see these indexes created on startup.:

```
$ elastalert-create-index
New index name (Default elastalert_status)
Name of existing index to copy (Default None)
New index elastalert_status created
Done!
```

For information about what data will go here, see *ElastAlert 2 Metadata Index*.

2.4.4 Creating a Rule

Each rule defines a query to perform, parameters on what triggers a match, and a list of alerts to fire for each match. We are going to use `examples/rules/example_frequency.yaml` as a template:

```
# From examples/rules/example_frequency.yaml
es_host: elasticsearch.example.com
es_port: 14900
name: Example rule
type: frequency
index: logstash-*
num_events: 50
timeframe:
  hours: 4
filter:
- term:
    some_field: "some_value"
alert:
- "email"
email:
- "elastalert@example.com"
```

`es_host` and `es_port` should point to the Elasticsearch cluster we want to query.

`name` is the unique name for this rule. ElastAlert 2 will not start if two rules share the same name.

`type`: Each rule has a different type which may take different parameters. The `frequency` type means "Alert when more than `num_events` occur within `timeframe`." For information other types, see *Rule types*.

`index`: The name of the index(es) to query. If you are using Logstash, by default the indexes will match `"logstash-*"`.

`num_events`: This parameter is specific to `frequency` type and is the threshold for when an alert is triggered.

`timeframe` is the time period in which `num_events` must occur.

`filter` is a list of Elasticsearch filters that are used to filter results. Here we have a single term filter for documents with `some_field` matching `some_value`. See *Writing Filters For Rules* for more information. If no filters are desired, it should be specified as an empty list: `filter: []`

`alert` is a list of alerts to run on each match. For more information on alert types, see *Alerts*. The email alert requires an SMTP server for sending mail. By default, it will attempt to use localhost. This can be changed with the `smtp_host` option.

`email` is a list of addresses to which alerts will be sent.

There are many other optional configuration options, see *Common configuration options*.

All documents must have a timestamp field. ElastAlert 2 will try to use `@timestamp` by default, but this can be changed with the `timestamp_field` option. By default, ElastAlert 2 uses ISO8601 timestamps, though unix timestamps are supported by setting `timestamp_type`.

As is, this rule means "Send an email to elastalert@example.com when there are more than 50 documents with `some_field == some_value` within a 4 hour period."

2.4.5 Testing Your Rule

Running the `elastalert-test-rule` tool will test that your config file successfully loads and run it in debug mode over the last 24 hours:

```
$ elastalert-test-rule examples/rules/example_frequency.yaml
```

If you want to specify a configuration file to use, you can run it with the config flag:

```
$ elastalert-test-rule --config <path-to-config-file> examples/rules/example_frequency.
↪yaml
```

The configuration preferences will be loaded as follows:

1. Configurations specified in the yaml file.
2. Configurations specified in the config file, if specified.
3. Default configurations, for the tool to run.

See *the testing section for more details*

2.4.6 Running ElastAlert 2

There are two ways of invoking ElastAlert 2. As a daemon, through Supervisor (http://supervisord.org/), or directly with Python. For easier debugging purposes in this tutorial, we will invoke it directly:

```
$ python -m elastalert.elastalert --verbose --rule example_frequency.yaml  # or use the␣
↪entry point: elastalert --verbose --rule ...
No handlers could be found for logger "Elasticsearch"
INFO:root:Queried rule Example rule from 1-15 14:22 PST to 1-15 15:07 PST: 5 hits
INFO:Elasticsearch:POST http://elasticsearch.example.com:14900/elastalert_status/
↪elastalert_status?op_type=create [status:201 request:0.025s]
INFO:root:Ran Example rule from 1-15 14:22 PST to 1-15 15:07 PST: 5 query hits (0␣
↪already seen), 0 matches, 0 alerts sent
INFO:root:Sleeping for 297 seconds
```

ElastAlert 2 uses the python logging system and `--verbose` sets it to display INFO level messages. `--rule example_frequency.yaml` specifies the rule to run, otherwise ElastAlert 2 will attempt to load the other rules in the `examples/rules` folder.

Let's break down the response to see what's happening.

`Queried rule Example rule from 1-15 14:22 PST to 1-15 15:07 PST: 5 hits`

ElastAlert 2 periodically queries the most recent `buffer_time` (default 45 minutes) for data matching the filters. Here we see that it matched 5 hits:

```
POST http://elasticsearch.example.com:14900/elastalert_status/elastalert_status?op_
↪type=create [status:201 request:0.025s]
```

This line showing that ElastAlert 2 uploaded a document to the elastalert_status index with information about the query it just made:

```
Ran Example rule from 1-15 14:22 PST to 1-15 15:07 PST: 5 query hits (0 already seen), 0␣
↪matches, 0 alerts sent
```

The line means ElastAlert 2 has finished processing the rule. For large time periods, sometimes multiple queries may be run, but their data will be processed together. `query hits` is the number of documents that are downloaded from Elasticsearch, `already seen` refers to documents that were already counted in a previous overlapping query and will be ignored, `matches` is the number of matches the rule type outputted, and `alerts sent` is the number of alerts actually sent. This may differ from `matches` because of options like `realert` and `aggregation` or because of an error.

`Sleeping for 297 seconds`

The default `run_every` is 5 minutes, meaning ElastAlert 2 will sleep until 5 minutes have elapsed from the last cycle before running queries for each rule again with time ranges shifted forward 5 minutes.

Say, over the next 297 seconds, 46 more matching documents were added to Elasticsearch:

```
INFO:root:Queried rule Example rule from 1-15 14:27 PST to 1-15 15:12 PST: 51 hits
...
INFO:root:Sent email to ['elastalert@example.com']
...
INFO:root:Ran Example rule from 1-15 14:27 PST to 1-15 15:12 PST: 51 query hits, 1␣
↪matches, 1 alerts sent
```

The body of the email will contain something like:

```
Example rule

At least 50 events occurred between 1-15 11:12 PST and 1-15 15:12 PST

@timestamp: 2015-01-15T15:12:00-08:00
```

If an error occurred, such as an unreachable SMTP server, you may see:

```
ERROR:root:Error while running alert email: Error connecting to SMTP host: [Errno 61]␣
↪Connection refused
```

Note that if you stop ElastAlert 2 and then run it again later, it will look up `elastalert_status` and begin querying at the end time of the last query. This is to prevent duplication or skipping of alerts if ElastAlert 2 is restarted.

By using the `--debug` flag instead of `--verbose`, the body of email will instead be logged and the email will not be sent. In addition, the queries will not be saved to `elastalert_status`.

2.4.7 Disabling a Rule

To stop a rule from executing, add or adjust the *is_enabled* option inside the rule's YAML file to *false*. When ElastAlert 2 reloads the rules it will detect that the rule has been disabled and prevent it from executing. The rule reload interval defaults to 5 minutes but can be adjusted via the *run_every* configuration option.

Optionally, once a rule has been disabled it is safe to remove the rule file, if there is no intention of re-activating the rule. However, be aware that removing a rule file without first disabling it will _not_ disable the rule!

CHAPTER

THREE

RULE TYPES AND CONFIGURATION OPTIONS

Examples of several types of rule configuration can be found in the `examples/rules` folder.

Note: All "time" formats are of the form `unit: X` where unit is one of weeks, days, hours, minutes or seconds. Such as `minutes: 15` or `hours: 1`.

3.1 Rule Configuration Cheat Sheet

FOR ALL RULES	
`es_host` (string)	Required
`es_port` (number)	
`index` (string)	
`type` (string)	
`alert` (string or list)	
`es_hosts` (list, no default)	Optional
`name` (string, defaults to the filename)	
`use_strftime_index` (boolean, default False)	
`use_ssl` (boolean, default False)	
`verify_certs` (boolean, default True)	
`ssl_show_warn` (boolean, default True)	
`es_username` (string, no default)	
`es_password` (string, no default)	
`es_bearer` (string, no default)	
`es_api_key` (string, no default)	
`es_url_prefix` (string, no default)	
`statsd_instance_tag` (string, no default)	
`statsd_host` (string, no default)	
`es_send_get_body_as` (string, default "GET")	
`aggregation` (time, no default)	
`limit_execution` (string, no default)	
`description` (string, default empty string)	
`kibana_url` (string, default from es_host)	
`kibana_username` (string, no default)	
`kibana_password` (string, no default)	
`generate_kibana_discover_url` (boolean, default False)	
`shorten_kibana_discover_url` (boolean, default False)	

continues on next page

Table 1 – continued from previous page

FOR ALL RULES	
kibana_discover_app_url (string, no default)	
kibana_discover_version (string, no default)	
kibana_discover_index_pattern_id (string, no default)	
kibana_discover_security_tenant (string, no default)	
kibana_discover_columns (list of strs, default _source)	
kibana_discover_from_timedelta (time, default: 10 min)	
kibana_discover_to_timedelta (time, default: 10 min)	
use_local_time (boolean, default True)	
realert (time, default: 1 min)	
exponential_realert (time, no default)	
match_enhancements (list of strs, no default)	
top_count_number (int, default 5)	
top_count_keys (list of strs)	
raw_count_keys (boolean, default True)	
include (list of strs, default ["*"])	
filter (ES filter DSL, no default)	
max_query_size (int, default global max_query_size)	
query_delay (time, default 0 min)	
owner (string, default empty string)	
priority (int, default 2)	
category (string, default empty string)	
scan_entire_timeframe (bool, default False)	
query_timezone (string, default empty string)	
import (string) IGNORED IF use_count_query or use_terms_query is true	
buffer_time (time, default from config.yaml)	
timestamp_type (string, default iso)	
timestamp_format (string, default "%Y-%m-%dT%H:%M:%SZ")	
timestamp_format_expr (string, no default)	
_source_enabled (boolean, default True)	
alert_text_args (array of strs)	
alert_text_kw (object)	
alert_missing_value (string, default "<MISSING VALUE>")	
is_enabled (boolean, default True)	
search_extra_index (boolean, default False)	

RULE TYPE	Any	Black-list	Whitelist	Change	Fre-quency	Spike	Flat-line	New_term	Car-di-nality	Met-ric Ag-gre-ga-tion	Spike Ag-gre-ga-tion	Per-cent-age Match
`compare_key` (list of strs, no de-fault)		Req	Req	Req								
`blacklist` (list of strs, no de-fault)		Req										
`whitelist` (list of strs, no de-fault)			Req									
`ignore_null` (boolean, de-fault False)			Req	Req								
`query_key` (string or list, no de-fault)	Opt			Req	Opt	Opt	Opt	Req	Opt	Opt	Opt	Opt
`aggregation_key` (string, no de-fault)	Opt											
`summary_table_fields` (list, no de-fault)	Opt											

continues on next page

Table 2 – continued from previous page

RULE TYPE	Any	Black-list	Whitelist	Change	Fre-quency	Spike	Flat-line	New_term	Car-di-nality	Met-ric Ag-gre-ga-tion	Spike Ag-gre-ga-tion	Per-cent-age Match
`timeframe` (time, no de-fault)				Opt	Req	Req	Req		Req		Req	
`num_events` (int, no de-fault)					Req							
`attach_related` (boolean, de-fault False)					Opt							
`use_count_query` (boolean, de-fault False)					Opt	Opt	Opt					
`use_terms_query` (boolean, de-fault False) `query_key` (string or list, no de-fault) `terms_size` (int, de-fault 50)					Opt	Opt		Opt				
`spike_height` (int, no de-fault)						Req					Req	

continues on next page

Table 2 – continued from previous page

RULE TYPE	Any	Black-list	Whitelist	Change	Fre-quency	Spike	Flat-line	New_term	Car-di-nality	Met-ric Ag-gre-ga-tion	Spike Ag-gre-ga-tion	Per-cent-age Match
`spike_type` ([up|down|both], no de-fault)						Req					Req	
`alert_on_new_data` (boolean, de-fault False)						Opt						
`threshold_ref` (int, no de-fault)						Opt						
`threshold_ref` (num-ber, no de-fault)											Opt	
`threshold_cur` (int, no de-fault)						Opt						
`threshold_cur` (num-ber, no de-fault)											Opt	
`threshold` (int, no de-fault)							Req					
`fields` (string or list, no de-fault)								Req				

continues on next page

Table 2 – continued from previous page

RULE TYPE	Any	Black-list	Whitelist	Change	Fre-quency	Spike	Flat-line	New_term	Car-di-nality	Met-ric Ag-gre-ga-tion	Spike Ag-gre-ga-tion	Per-cent-age Match
`terms_window_size` (time, de-fault 30 days)								Opt				
`window_step_size` (time, de-fault 1 day)								Opt				
`alert_on_missing_field` (boolean, de-fault False)								Opt				
`cardinality_field` (string, no de-fault)									Req			
`max_cardinality` (boolean, de-fault False)									Opt			
`min_cardinality` (boolean, de-fault False)									Opt			
`metric_agg_key` (string, no de-fault)										Req		
`metric_agg_type` (no de-fault, ([min|max|avg|sum|cardinality|value_count|percentiles])										Req	Req	
`metric_agg_script` (no de-fault)										Opt	Opt	

continues on next page

Table 2 – continued from previous page

RULE TYPE	Any	Black-list	Whitelist	Change	Fre-quency	Spike	Flat-line	New_term	Car-di-nality	Met-ric Ag-gre-ga-tion	Spike Ag-gre-ga-tion	Per-cent-age Match
`percentile_range` ++re-quired if per-centiles is used										Req++	Req++	
`max_threshold` (num-ber, no de-fault) `min_threshold` (num-ber, no de-fault) Re-quires at least one of the two op-tions										Opt		
`min_doc_count` (int, de-fault 1)										Opt	Opt	
`use_run_every_query_size` (boolean, de-fault False)										Opt		Opt
`allow_buffer_time_overlap` (boolean, de-fault False)										Opt		Opt

continues on next page

Table 2 – continued from previous page

RULE TYPE	Any	Blacklist	Whitelist	Change	Frequency	Spike	Flatline	New_term	Cardinality	Metric Aggregation	Spike Aggregation	Percentage Match
`bucket_interval` (time, no default) `sync_bucket_interval` (boolean, default False)										Opt		Opt
`metric_format_string` (string, no default)										Opt		
`match_bucket_filter` (no default)												Req
`min_percentage` (number, no default) `max_percentage` (number, no default) Requires at least one of the two options												Req
`percentage_format_string` (string, no default)												Opt

continues on next page

Table 2 – continued from previous page

RULE TYPE	Any	Black-list	Whitelist	Change	Fre-quency	Spike	Flat-line	New_term	Car-di-nality	Met-ric Ag-gre-ga-tion	Spike Ag-gre-ga-tion	Per-cent-age Match
`min_denominator` (int, de-fault 0)												Opt

3.2 Common Configuration Options

Every file that ends in `.yaml` in the `rules_folder` will be run by default. The following configuration settings are common to all types of rules.

3.2.1 Required Settings

es_host

`es_host`: The hostname of the Elasticsearch cluster the rule will use to query. (Required, string, no default) The environment variable `ES_HOST` will override this field. For multiple host Elasticsearch clusters see `es_hosts` parameter.

es_port

`es_port`: The port of the Elasticsearch cluster. (Required, number, no default) The environment variable `ES_PORT` will override this field.

index

`index`: The name of the index that will be searched. Wildcards can be used here, such as: `index: my-index-*` which will match `my-index-2014-10-05`. You can also use a format string containing `%Y` for year, `%m` for month, and `%d` for day. To use this, you must also set `use_strftime_index` to true. (Required, string, no default)

name

`name`: The name of the rule. This must be unique across all rules. The name will be used in alerts and used as a key when writing and reading search metadata back from Elasticsearch. (Required, string, no default)

type

`type`: The `RuleType` to use. This may either be one of the built in rule types, see *Rule Types* section below for more information, or loaded from a module. For loading from a module, the type should be specified as `module.file.RuleName`. (Required, string, no default)

alert

`alert`: The `Alerter` type to use. This may be one or more of the built in alerts, see *Alert Types* section below for more information, or loaded from a module. For loading from a module, the alert should be specified as `module.file.AlertName`. (Required, string or list, no default)

3.2.2 Optional Settings

es_hosts

`es_hosts`: The list of nodes of the Elasticsearch cluster that the rule will use for the request. (Optional, list, default none). Values can be specified as `host:port` if overriding the default port. The environment variable `ES_HOSTS` will override this field, and can be specified as a comma-separated value. Note that the `es_host` parameter must still be specified in order to identify a primary Elasticsearch host.

import

`import`: If specified includes all the settings from this yaml file. This allows common config options to be shared. Note that imported files that aren't complete rules should not have a `.yml` or `.yaml` suffix so that ElastAlert 2 doesn't treat them as rules. Filters in imported files are merged (ANDed) with any filters in the rule. You can only have one import per rule, though the imported file can import another file or multiple files, recursively. The filename can be an absolute path or relative to the rules directory. (Optional, string or array of strings, no default)

use_ssl

`use_ssl`: Whether or not to connect to `es_host` using TLS. (Optional, boolean, default False) The environment variable `ES_USE_SSL` will override this field.

ssl_show_warn

`ssl_show_warn`: Whether or not to show SSL/TLS warnings when `verify_certs` is disabled. (Optional, boolean, default True)

verify_certs

`verify_certs`: Whether or not to verify TLS certificates. (Optional, boolean, default True)

client_cert

`client_cert`: Path to a PEM certificate to use as the client certificate (Optional, string, no default)

client_key

`client_key`: Path to a private key file to use as the client key (Optional, string, no default)

ca_certs

`ca_certs`: Path to a CA cert bundle to use to verify SSL connections (Optional, string, no default)

es_username

`es_username`: basic-auth username for connecting to `es_host`. (Optional, string, no default) The environment variable `ES_USERNAME` will override this field.

es_password

`es_password`: basic-auth password for connecting to `es_host`. (Optional, string, no default) The environment variable `ES_PASSWORD` will override this field.

es_bearer

`es_bearer`: bearer-token authorization for connecting to `es_host`. (Optional, string, no default) The environment variable `ES_BEARER` will override this field. This authentication option will override the password authentication option.

es_api_key

`es_api_key`: api-key-token authorization for connecting to `es_host`. (Optional, base64 string, no default) The environment variable `ES_API_KEY` will override this field. This authentication option will override both the bearer and the password authentication options.

es_url_prefix

`es_url_prefix`: URL prefix for the Elasticsearch endpoint. (Optional, string, no default)

statsd_instance_tag

`statsd_instance_tag`: prefix for statsd metrics. (Optional, string, no default)

statsd_host

`statsd_host`: statsd host. (Optional, string, no default)

es_send_get_body_as

`es_send_get_body_as`: Method for querying Elasticsearch. (Optional, string, default "GET")

use_strftime_index

`use_strftime_index`: If this is true, ElastAlert 2 will format the index using datetime.strftime for each query. See https://docs.python.org/2/library/datetime.html#strftime-strptime-behavior for more details. If a query spans multiple days, the formatted indexes will be concatenated with commas. This is useful as narrowing the number of indexes searched, compared to using a wildcard, may be significantly faster. For example, if `index` is `logstash-%Y.%m.%d`, the query url will be similar to `elasticsearch.example.com/logstash-2015.02.03/...` or `elasticsearch.example.com/logstash-2015.02.03,logstash-2015.02.04/...`.

search_extra_index

`search_extra_index`: If this is true, ElastAlert 2 will add an extra index on the early side onto each search. For example, if it's querying completely within 2018-06-28, it will actually use 2018-06-27,2018-06-28. This can be useful if your timestamp_field is not what's being used to generate the index names. If that's the case, sometimes a query would not have been using the right index.

aggregation

`aggregation`: This option allows you to aggregate multiple matches together into one alert. Every time a match is found, ElastAlert 2 will wait for the `aggregation` period, and send all of the matches that have occurred in that time for a particular rule together.

For example:

```
aggregation:
  hours: 2
```

means that if one match occurred at 12:00, another at 1:00, and a third at 2:30, one alert would be sent at 2:00, containing the first two matches, and another at 4:30, containing the third match plus any additional matches occurring before 4:30. This can be very useful if you expect a large number of matches and only want a periodic report. (Optional, time, default none)

If you wish to aggregate all your alerts and send them on a recurring interval, you can do that using the `schedule` field.

For example, if you wish to receive alerts every Monday and Friday:

```
aggregation:
  schedule: '2 4 * * mon,fri'
```

This uses Cron syntax, which you can read more about here. Make sure to *only* include either a schedule field or standard datetime fields (such as `hours`, `minutes`, `days`), not both.

By default, all events that occur during an aggregation window are grouped together. However, if your rule has the `aggregation_key` field set, then each event sharing a common key value will be grouped together. A separate aggregation window will be made for each newly encountered key value.

For example, if you wish to receive alerts that are grouped by the user who triggered the event, you can set:

```
aggregation_key: 'my_data.username'
```

Then, assuming an aggregation window of 10 minutes, if you receive the following data points:

```
{'my_data': {'username': 'alice', 'event_type': 'login'}, '@timestamp': '2016-09-
↪20T00:00:00'}
{'my_data': {'username': 'bob', 'event_type': 'something'}, '@timestamp': '2016-09-
↪20T00:05:00'}
{'my_data': {'username': 'alice', 'event_type': 'something else'}, '@timestamp': '2016-
↪09-20T00:06:00'}
```

This should result in 2 alerts: One containing alice's two events, sent at `2016-09-20T00:10:00` and one containing bob's one event sent at `2016-09-20T00:16:00`

For aggregations, there can sometimes be a large number of documents present in the viewing medium (email, Jira ticket, etc..). If you set the `summary_table_fields` field, ElastAlert 2 will provide a summary of the specified fields from all the results.

The formatting style of the summary table can be switched between `ascii` (default) and `markdown` with parameter `summary_table_type`. `markdown` might be the more suitable formatting for alerters supporting it like TheHive.

The maximum number of rows in the summary table can be limited with the parameter `summary_table_max_rows`.

For example, if you wish to summarize the usernames and event_types that appear in the documents so that you can see the most relevant fields at a quick glance, you can set:

```
summary_table_fields:
    - my_data.username
    - my_data.event_type
```

Then, for the same sample data shown above listing alice and bob's events, ElastAlert 2 will provide the following summary table in the alert medium:

```
+------------------+--------------------+
| my_data.username | my_data.event_type |
+------------------+--------------------+
|      alice       |       login        |
|       bob        |     something      |
|      alice       |   something else   |
+------------------+--------------------+
```

Note: By default, aggregation time is relative to the current system time, not the time of the match. This means that running ElastAlert 2 over past events will result in different alerts than if ElastAlert 2 had been running while those events occured. This behavior can be changed by setting `aggregate_by_match_time`.

limit_execution

`limit_execution`: This option allows you to activate the rule during a limited period of time. This uses the cron format.

For example, if you wish to activate the rule from monday to friday, between 10am to 6pm:

```
limit_execution: "* 10-18 * * 1-5"
```

aggregate_by_match_time

Setting this to true will cause aggregations to be created relative to the timestamp of the first event, rather than the current time. This is useful for querying over historic data or if using a very large buffer_time and you want multiple aggregations to occur from a single query.

realert

`realert`: This option allows you to ignore repeating alerts for a period of time. If the rule uses a `query_key`, this option will be applied on a per key basis. All matches for a given rule, or for matches with the same `query_key`, will be ignored for the given time. All matches with a missing `query_key` will be grouped together using a value of `_missing`. This is applied to the time the alert is sent, not to the time of the event. It defaults to one minute, which means that if ElastAlert 2 is run over a large time period which triggers many matches, only the first alert will be sent by default. If you want every alert, set realert to 0 minutes. (Optional, time, default 1 minute)

exponential_realert

`exponential_realert`: This option causes the value of `realert` to exponentially increase while alerts continue to fire. If set, the value of `exponential_realert` is the maximum `realert` will increase to. If the time between alerts is less than twice `realert`, `realert` will double. For example, if `realert: minutes: 10` and `exponential_realert: hours: 1`, an alerts fires at 1:00 and another at 1:15, the next alert will not be until at least 1:35. If another alert fires between 1:35 and 2:15, `realert` will increase to the 1 hour maximum. If more than 2 hours elapse before the next alert, `realert` will go back down. Note that alerts that are ignored (e.g. one that occurred at 1:05) would not change `realert`. (Optional, time, no default)

buffer_time

`buffer_time`: This options allows the rule to override the `buffer_time` global setting defined in config.yaml. This value is ignored if `use_count_query` or `use_terms_query` is true. (Optional, time)

query_delay

`query_delay`: This option will cause ElastAlert 2 to subtract a time delta from every query, causing the rule to run with a delay. This is useful if the data is Elasticsearch doesn't get indexed immediately. (Optional, time)

For example:

```
query_delay:
  hours: 2
```

owner

`owner`: This value will be used to identify the stakeholder of the alert. Optionally, this field can be included in any alert type. (Optional, string)

priority

`priority`: This value will be used to identify the relative priority of the alert. Optionally, this field can be included in any alert type (e.g. for use in email subject/body text). (Optional, int, default 2)

category

`category`: This value will be used to identify the category of the alert. Optionally, this field can be included in any alert type (e.g. for use in email subject/body text). (Optional, string, default empty string)

max_query_size

`max_query_size`: The maximum number of documents that will be downloaded from Elasticsearch in a single query. If you expect a large number of results, consider using `use_count_query` for the rule. If this limit is reached, a warning will be logged but ElastAlert 2 will continue without downloading more results. This setting will override a global `max_query_size`. (Optional, int, default value of global `max_query_size`)

filter

`filter`: A list of Elasticsearch query DSL filters that is used to query Elasticsearch. ElastAlert 2 will query Elasticsearch using the format `{'filter': {'bool': {'must': [config.filter]}}}` with an additional timestamp range filter. All of the results of querying with these filters are passed to the `RuleType` for analysis. For more information writing filters, see *Writing Filters*. (Required, Elasticsearch query DSL, no default)

include

`include`: A list of terms that should be included in query results and passed to rule types and alerts. When set, only those fields, along with '@timestamp', `query_key`, `compare_key`, and `top_count_keys` are included, if present. (Optional, list of strings, default all fields)

top_count_keys

`top_count_keys`: A list of fields. ElastAlert 2 will perform a terms query for the top X most common values for each of the fields, where X is 5 by default, or `top_count_number` if it exists. For example, if `num_events` is 100, and `top_count_keys` is - `"username"`, the alert will say how many of the 100 events have each username, for the top 5 usernames. When this is computed, the time range used is from `timeframe` before the most recent event to 10 minutes past the most recent event. Because ElastAlert 2 uses an aggregation query to compute this, it will attempt to use the field name plus ".keyword" to count unanalyzed terms. To turn this off, set `raw_count_keys` to false.

top_count_number

`top_count_number`: The number of terms to list if `top_count_keys` is set. (Optional, integer, default 5)

raw_count_keys

`raw_count_keys`: If true, all fields in `top_count_keys` will have `.keyword` appended to them. This used to be ".raw" in older Elasticsearch versions, but the setting name *raw_count_keys* was left as-is to avoid breaking existing installations. (Optional, boolean, default true)

description

`description`: text describing the purpose of rule. (Optional, string, default empty string) Can be referenced in custom alerters to provide context as to why a rule might trigger.

kibana_url

`kibana_url`: The base url of the Kibana application. If not specified, a URL will be constructed using `es_host` and `es_port`.

This value will be used if `generate_kibana_discover_url` is true and `kibana_discover_app_url` is a relative path

(Optional, string, default `http://<es_host>:<es_port>/_plugin/kibana/`)

kibana_username

`kibana_username`: The username used to make basic authenticated API requests against Kibana. This value is only used if `shorten_kibana_discover_url` is true.

(Optional, string, no default)

kibana_password

`kibana_password`: The password used to make basic authenticated API requests against Kibana. This value is only used if `shorten_kibana_discover_url` is true.

(Optional, string, no default)

generate_kibana_discover_url

`generate_kibana_discover_url`: Enables the generation of the `kibana_discover_url` variable for the Kibana Discover application. This setting requires the following settings are also configured:

- `kibana_discover_app_url`
- `kibana_discover_version`
- `kibana_discover_index_pattern_id`

```
generate_kibana_discover_url:  true
```

Example usage:

```
generate_kibana_discover_url: True
kibana_discover_app_url: "http://localhost:5601/app/discover#/"
kibana_discover_index_pattern_id: "4babf380-c3b1-11eb-b616-1b59c2feec54"
kibana_discover_version: "7.15"
kibana_discover_from_timedelta:
  minutes: 10
kibana_discover_to_timedelta:
  minutes: 10
alert_text: '{0}'
alert_text_args: [ kibana_discover_url ]
alert_text_type: alert_text_only
```

shorten_kibana_discover_url

`shorten_kibana_discover_url`: Enables the shortening of the generated Kibana Discover urls. In order to use the Kibana Shorten URL REST API, the `kibana_discover_app_url` must be provided as a relative url (e.g. app/discover?#/).

ElastAlert may need to authenticate with Kibana to invoke the Kibana Shorten URL REST API. The supported authentication methods are:

- Basic authentication by specifying `kibana_username` and `kibana_password`
- AWS authentication (if configured already for ElasticSearch)

(Optional, bool, false)

kibana_discover_app_url

`kibana_discover_app_url`: The url of the Kibana Discover application used to generate the `kibana_discover_url` variable. This value can use *$VAR* and *${VAR}* references to expand environment variables. This value should be relative to the base kibana url defined by `kibana_url` and will vary depending on your installation.

`kibana_discover_app_url: app/discover#/`

(Optional, string, no default)

kibana_discover_security_tenant

`kibana_discover_security_tenant`: The Kibana security tenant to include in the generated `kibana_discover_url` variable.

(Optional, string, no default)

kibana_discover_version

`kibana_discover_version`: Specifies the version of the Kibana Discover application.

The currently supported versions of Kibana Discover are:

- *7.0*, *7.1*, *7.2*, *7.3*, *7.4*, *7.5*, *7.6*, *7.7*, *7.8*, *7.9*, *7.10*, *7.11*, *7.12*, *7.13*, *7.14*, *7.15*, *7.16*, *7.17*
- *8.0*, *8.1*, *8.2*, *8.3*

```
kibana_discover_version:  '7.15'
```

kibana_discover_index_pattern_id

`kibana_discover_index_pattern_id`: The id of the index pattern to link to in the Kibana Discover application. These ids are usually generated and can be found in url of the index pattern management page, or by exporting its saved object.

In this documentation all references of "index pattern" refer to the similarly named concept in Kibana 8 called "data view".

Example export of an index pattern's saved object:

```
[
    {
        "_id": "4e97d188-8a45-4418-8a37-07ed69b4d34c",
        "_type": "index-pattern",
        "_source": { ... }
    }
]
```

You can modify an index pattern's id by exporting the saved object, modifying the `_id` field, and re-importing.

```
kibana_discover_index_pattern_id:  4e97d188-8a45-4418-8a37-07ed69b4d34c
```

kibana_discover_columns

`kibana_discover_columns`: The columns to display in the generated Kibana Discover application link. Defaults to the `_source` column.

```
kibana_discover_columns:  [ timestamp, message ]
```

kibana_discover_from_timedelta

`kibana_discover_from_timedelta`: The offset to the *from* time of the Kibana Discover link's time range. The *from* time is calculated by subtracting this timedelta from the event time. Defaults to 10 minutes.

```
kibana_discover_from_timedelta:  minutes:  2
```

kibana_discover_to_timedelta

`kibana_discover_to_timedelta`: The offset to the *to* time of the Kibana Discover link's time range. The *to* time is calculated by adding this timedelta to the event time. Defaults to 10 minutes.

```
kibana_discover_to_timedelta:  minutes:  2
```

use_local_time

`use_local_time`: Whether to convert timestamps to the local time zone in alerts. If false, timestamps will be converted to UTC, which is what ElastAlert 2 uses internally. (Optional, boolean, default true)

match_enhancements

`match_enhancements`: A list of enhancement modules to use with this rule. An enhancement module is a subclass of enhancements.BaseEnhancement that will be given the match dictionary and can modify it before it is passed to the alerter. The enhancements will be run after silence and realert is calculated and in the case of aggregated alerts, right before the alert is sent. This can be changed by setting `run_enhancements_first`. The enhancements should be specified as `module.file.EnhancementName`. See *Enhancements* for more information. (Optional, list of strings, no default)

run_enhancements_first

`run_enhancements_first`: If set to true, enhancements will be run as soon as a match is found. This means that they can be changed or dropped before affecting realert or being added to an aggregation. Silence stashes will still be created before the enhancement runs, meaning even if a `DropMatchException` is raised, the rule will still be silenced. (Optional, boolean, default false)

query_key

`query_key`: Having a query key means that realert time will be counted separately for each unique value of `query_key`. For rule types which count documents, such as spike, frequency and flatline, it also means that these counts will be independent for each unique value of `query_key`. For example, if `query_key` is set to `username` and `realert` is set, and an alert triggers on a document with `{'username':  'bob'}`, additional alerts for `{'username':  'bob'}` will be ignored while other usernames will trigger alerts. Documents which are missing the `query_key` will be grouped together. A list of fields may also be used, which will create a compound query key. This compound key is treated as if it were a single field whose value is the component values, or "None", joined by commas. A new field with the key "field1,field2,etc" will be created in each document and may conflict with existing fields of the same name.

aggregation_key

`aggregation_key`: Having an aggregation key in conjunction with an aggregation will make it so that each new value encountered for the aggregation_key field will result in a new, separate aggregation window.

summary_table_fields

summary_table_fields: Specifying the summmary_table_fields in conjunction with an aggregation will make it so that each aggregated alert will contain a table summarizing the values for the specified fields in all the matches that were aggregated together.

summary_table_type

summary_table_type: Either ascii or markdown. Select the table type to use for the aggregation summary. Defaults to ascii for the classical text based table.

summary_table_max_rows

summary_table_max_rows: Limit the maximum number of rows that will be shown in the summary table.

summary_prefix

summary_prefix: Specify a prefix string, which will be added in front of the aggregation summary table. This string is currently not subject to any formatting.

summary_suffix

summary_suffix: Specify a suffix string, which will be added after the aggregation summary table. This string is currently not subject to any formatting.

timestamp_type

timestamp_type: One of iso, unix, unix_ms, custom. This option will set the type of @timestamp (or timestamp_field) used to query Elasticsearch. iso will use ISO8601 timestamps, which will work with most Elasticsearch date type field. unix will query using an integer unix (seconds since 1/1/1970) timestamp. unix_ms will use milliseconds unix timestamp. custom allows you to define your own timestamp_format. The default is iso. (Optional, string enum, default iso).

timestamp_format

timestamp_format: In case Elasticsearch used custom date format for date type field, this option provides a way to define custom timestamp format to match the type used for Elastisearch date type field. This option is only valid if timestamp_type set to custom. (Optional, string, default '%Y-%m-%dT%H:%M:%SZ').

timestamp_format_expr

timestamp_format_expr: In case Elasticsearch used custom date format for date type field, this option provides a way to adapt the value obtained converting a datetime through timestamp_format, when the format cannot match perfectly what defined in Elastisearch. When set, this option is evaluated as a Python expression along with a *globals* dictionary containing the original datetime instance named dt and the timestamp to be refined, named ts. The returned value becomes the timestamp obtained from the datetime. For example, when the date type field in Elasticsearch uses milliseconds (yyyy-MM-dd'T'HH:mm:ss.SSS'Z') and timestamp_format option is '%Y-%m-%dT%H:%M:%S.%fZ', Elasticsearch would fail to parse query terms as they contain microsecond values - that is it gets 6 digits instead of 3 - since the %f placeholder stands for microseconds for Python *strftime* method calls. Setting timestamp_format_expr:

`'ts[:23] + ts[26:]'` will truncate the value to milliseconds granting Elasticsearch compatibility. This option is only valid if `timestamp_type` set to `custom`. (Optional, string, no default).

_source_enabled

`_source_enabled`: If true, ElastAlert 2 will use _source to retrieve fields from documents in Elasticsearch. If false, ElastAlert 2 will use `fields` to retrieve stored fields. Both of these are represented internally as if they came from `_source`. See https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-fields.html for more details. The fields used come from `include`, see above for more details. (Optional, boolean, default True)

scan_entire_timeframe

`scan_entire_timeframe`: If true, when ElastAlert 2 starts, it will always start querying at the current time minus the timeframe. `timeframe` must exist in the rule. This may be useful, for example, if you are using a flatline rule type with a large timeframe, and you want to be sure that if ElastAlert 2 restarts, you can still get alerts. This may cause duplicate alerts for some rule types, for example, Frequency can alert multiple times in a single timeframe, and if ElastAlert 2 were to restart with this setting, it may scan the same range again, triggering duplicate alerts.

Some rules and alerts require additional options, which also go in the top level of the rule configuration file.

query_timezone

`query_timezone`: Whether to convert UTC time to the specified time zone in rule queries. If not set, start and end time of query will be used UTC. (Optional, string, default empty string)

Example value : query_timezone: “Europe/Istanbul”

3.3 Testing Your Rule

Once you’ve written a rule configuration, you will want to validate it. To do so, you can either run ElastAlert 2 in debug mode, or use `elastalert-test-rule`, which is a script that makes various aspects of testing easier.

It can:

- Check that the configuration file loaded successfully.
- Check that the Elasticsearch filter parses.
- Run against the last X day(s) and the show the number of hits that match your filter.
- Show the available terms in one of the results.
- Save documents returned to a JSON file.
- Run ElastAlert 2 using either a JSON file or actual results from Elasticsearch.
- Print out debug alerts or trigger real alerts.
- Check that, if they exist, the primary_key, compare_key and include terms are in the results.
- Show what metadata documents would be written to `elastalert_status`.

Without any optional arguments, it will run ElastAlert 2 over the last 24 hours and print out any alerts that would have occurred. Here is an example test run which triggered an alert:

```
$ elastalert-test-rule my_rules/rule1.yaml
Successfully Loaded Example rule1

Got 105 hits from the last 1 day

Available terms in first hit:
    @timestamp
    field1
    field2
    ...
Included term this_field_doesnt_exist may be missing or null

INFO:root:Queried rule Example rule1 from 6-16 15:21 PDT to 6-17 15:21 PDT: 105 hits
INFO:root:Alert for Example rule1 at 2015-06-16T23:53:12Z:
INFO:root:Example rule1

At least 50 events occurred between 6-16 18:30 PDT and 6-16 20:30 PDT

field1:
value1: 25
value2: 25

@timestamp: 2015-06-16T20:30:04-07:00
field1: value1
field2: something


Would have written the following documents to elastalert_status:

silence - {'rule_name': 'Example rule1', '@timestamp': datetime.datetime( ... ),
↪'exponent': 0, 'until':
datetime.datetime( ... )}

elastalert_status - {'hits': 105, 'matches': 1, '@timestamp': datetime.datetime( ... ),
↪'rule_name': 'Example rule1',
'starttime': datetime.datetime( ... ), 'endtime': datetime.datetime( ... ), 'time_taken
↪': 3.1415926}
```

Note that everything between "Alert for Example rule1 at …" and "Would have written the following …" is the exact text body that an alert would have. See the section below on alert content for more details. Also note that datetime objects are converted to ISO8601 timestamps when uploaded to Elasticsearch. See *the section on metadata* for more details.

Other options include:

`--schema-only`: Only perform schema validation on the file. It will not load modules or query Elasticsearch. This may catch invalid YAML and missing or misconfigured fields.

`--count-only`: Only find the number of matching documents and list available fields. ElastAlert 2 will not be run and documents will not be downloaded.

`--days N`: Instead of the default 1 day, query N days. For selecting more specific time ranges, you must run ElastAlert 2 itself and use `--start` and `--end`.

`--save-json FILE`: Save all documents downloaded to a file as JSON. This is useful if you wish to modify data while testing or do offline testing in conjunction with `--data FILE`. A maximum of 10,000 documents will be downloaded.

`--data FILE`: Use a JSON file as a data source instead of Elasticsearch. The file should be a single list containing objects, rather than objects on separate lines. Note than this uses mock functions which mimic some Elasticsearch query methods and is not guaranteed to have the exact same results as with Elasticsearch. For example, analyzed string fields may behave differently.

`--alert`: Trigger real alerts instead of the debug (logging text) alert.

`--formatted-output`: Output results in formatted JSON.

Note: Results from running this script may not always be the same as if an actual ElastAlert 2 instance was running. Some rule types, such as spike and flatline require a minimum elapsed time before they begin alerting, based on their timeframe. In addition, use_count_query and use_terms_query rely on run_every to determine their resolution. This script uses a fixed 5 minute window, which is the same as the default.

3.4 Rule Types

The various `RuleType` classes, defined in `elastalert/ruletypes.py`, form the main logic behind ElastAlert 2. An instance is held in memory for each rule, passed all of the data returned by querying Elasticsearch with a given filter, and generates matches based on that data.

To select a rule type, set the `type` option to the name of the rule type in the rule configuration file:

`type: <rule type>`

3.4.1 Any

`any`: The any rule will match everything. Every hit that the query returns will generate an alert.

3.4.2 Blacklist

`blacklist`: The blacklist rule will check a certain field against a blacklist, and match if it is in the blacklist.

This rule requires two additional options:

`compare_key`: The name of the field to use to compare to the blacklist. If the field is null, those events will be ignored.

`blacklist`: A list of blacklisted values, and/or a list of paths to flat files which contain the blacklisted values using `"!file /path/to/file"`; for example:

```
blacklist:
    - value1
    - value2
    - "!file /tmp/blacklist1.txt"
    - "!file /tmp/blacklist2.txt"
```

It is possible to mix between blacklist value definitions, or use either one. The `compare_key` term must be equal to one of these values for it to match.

3.4.3 Whitelist

`whitelist`: Similar to `blacklist`, this rule will compare a certain field to a whitelist, and match if the list does not contain the term.

This rule requires three additional options:

`compare_key`: The name of the field to use to compare to the whitelist.

`ignore_null`: If true, events without a `compare_key` field will not match.

`whitelist`: A list of whitelisted values, and/or a list of paths to flat files which contain the whitelisted values using - `"!file /path/to/file"`; for example:

```
whitelist:
    - value1
    - value2
    - "!file /tmp/whitelist1.txt"
    - "!file /tmp/whitelist2.txt"
```

It is possible to mix between whitelisted value definitions, or use either one. The `compare_key` term must be in this list or else it will match.

3.4.4 Change

For an example configuration file using this rule type, look at `examples/rules/example_change.yaml`.

`change`: This rule will monitor a certain field and match if that field changes. The field must change with respect to the last event with the same `query_key`.

This rule requires three additional options:

`compare_key`: The names of the field to monitor for changes. Since this is a list of strings, we can have multiple keys. An alert will trigger if any of the fields change.

`ignore_null`: If true, events without a `compare_key` field will not count as changed. Currently this checks for all the fields in `compare_key`

`query_key`: This rule is applied on a per-`query_key` basis. This field must be present in all of the events that are checked.

There is also an optional field:

`timeframe`: The maximum time between changes. After this time period, ElastAlert 2 will forget the old value of the `compare_key` field.

3.4.5 Frequency

For an example configuration file using this rule type, look at `examples/rules/example_frequency.yaml`.

`frequency`: This rule matches when there are at least a certain number of events in a given time frame. This may be counted on a per-`query_key` basis.

This rule requires two additional options:

`num_events`: The number of events which will trigger an alert, inclusive.

`timeframe`: The time that `num_events` must occur within.

Optional:

use_count_query: If true, ElastAlert 2 will poll Elasticsearch using the count api, and not download all of the matching documents. This is useful is you care only about numbers and not the actual data. It should also be used if you expect a large number of query hits, in the order of tens of thousands or more.

use_terms_query: If true, ElastAlert 2 will make an aggregation query against Elasticsearch to get counts of documents matching each unique value of query_key. This must be used with query_key. This will only return a maximum of terms_size, default 50, unique terms.

terms_size: When used with use_terms_query, this is the maximum number of terms returned per query. Default is 50.

query_key: Counts of documents will be stored independently for each value of query_key. Only num_events documents, all with the same value of query_key, will trigger an alert.

attach_related: Will attach all the related events to the event that triggered the frequency alert. For example in an alert triggered with num_events: 3, the 3rd event will trigger the alert on itself and add the other 2 events in a key named related_events that can be accessed in the alerter.

3.4.6 Spike

spike: This rule matches when the volume of events during a given time period is spike_height times larger or smaller than during the previous time period. It uses two sliding windows to compare the current and reference frequency of events. We will call this two windows "reference" and "current".

This rule requires three additional options:

spike_height: The ratio of number of events in the last timeframe to the previous timeframe that when hit will trigger an alert.

spike_type: Either 'up', 'down' or 'both'. 'Up' meaning the rule will only match when the number of events is spike_height times higher. 'Down' meaning the reference number is spike_height higher than the current number. 'Both' will match either.

timeframe: The rule will average out the rate of events over this time period. For example, hours: 1 means that the 'current' window will span from present to one hour ago, and the 'reference' window will span from one hour ago to two hours ago. The rule will not be active until the time elapsed from the first event is at least two timeframes. This is to prevent an alert being triggered before a baseline rate has been established. This can be overridden using alert_on_new_data.

Optional:

field_value: When set, uses the value of the field in the document and not the number of matching documents. This is useful to monitor for example a temperature sensor and raise an alarm if the temperature grows too fast. Note that the means of the field on the reference and current windows are used to determine if the spike_height value is reached. Note also that the threshold parameters are ignored in this mode.

threshold_ref: The minimum number of events that must exist in the reference window for an alert to trigger. For example, if spike_height: 3 and threshold_ref: 10, then the 'reference' window must contain at least 10 events and the 'current' window at least three times that for an alert to be triggered.

threshold_cur: The minimum number of events that must exist in the current window for an alert to trigger. For example, if spike_height: 3 and threshold_cur: 60, then an alert will occur if the current window has more than 60 events and the reference window has less than a third as many.

To illustrate the use of threshold_ref, threshold_cur, alert_on_new_data, timeframe and spike_height together, consider the following examples:

```
" Alert if at least 15 events occur within two hours and less than a quarter of that␣
↪number occurred within the previous two hours. "
```

(continues on next page)

(continued from previous page)

```
timeframe: hours: 2
spike_height: 4
spike_type: up
threshold_cur: 15

hour1: 5 events (ref: 0, cur: 5) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled
hour2: 5 events (ref: 0, cur: 10) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled
hour3: 10 events (ref: 5, cur: 15) - No alert because (a) spike_height not met, (b) ref␣
↪window not filled
hour4: 35 events (ref: 10, cur: 45) - Alert because (a) spike_height met, (b) threshold_
↪cur met, (c) ref window filled

hour1: 20 events (ref: 0, cur: 20) - No alert because ref window not filled
hour2: 21 events (ref: 0, cur: 41) - No alert because ref window not filled
hour3: 19 events (ref: 20, cur: 40) - No alert because (a) spike_height not met, (b) ref␣
↪window not filled
hour4: 23 events (ref: 41, cur: 42) - No alert because spike_height not met

hour1: 10 events (ref: 0, cur: 10) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled
hour2: 0 events (ref: 0, cur: 10) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled
hour3: 0 events (ref: 10, cur: 0) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled, (c) spike_height not met
hour4: 30 events (ref: 10, cur: 30) - No alert because spike_height not met
hour5: 5 events (ref: 0, cur: 35) - Alert because (a) spike_height met, (b) threshold_
↪cur met, (c) ref window filled

" Alert if at least 5 events occur within two hours, and twice as many events occur␣
↪within the next two hours. "
timeframe: hours: 2
spike_height: 2
spike_type: up
threshold_ref: 5

hour1: 20 events (ref: 0, cur: 20) - No alert because (a) threshold_ref not met, (b) ref␣
↪window not filled
hour2: 100 events (ref: 0, cur: 120) - No alert because (a) threshold_ref not met, (b)␣
↪ref window not filled
hour3: 100 events (ref: 20, cur: 200) - No alert because ref window not filled
hour4: 100 events (ref: 120, cur: 200) - No alert because spike_height not met

hour1: 0 events (ref: 0, cur: 0) - No alert because (a) threshold_ref not met, (b) ref␣
↪window not filled
hour2: 20 events (ref: 0, cur: 20) - No alert because (a) threshold_ref not met, (b) ref␣
↪window not filled
hour3: 100 events (ref: 0, cur: 120) - No alert because (a) threshold_ref not met, (b)␣
↪ref window not filled
hour4: 100 events (ref: 20, cur: 200) - Alert because (a) spike_height met, (b)␣
↪threshold_ref met, (c) ref window filled
```

(continues on next page)

(continued from previous page)

```
hour1: 1 events (ref: 0, cur: 1) - No alert because (a) threshold_ref not met, (b) ref␣
↪window not filled
hour2: 2 events (ref: 0, cur: 3) - No alert because (a) threshold_ref not met, (b) ref␣
↪window not filled
hour3: 2 events (ref: 1, cur: 4) - No alert because (a) threshold_ref not met, (b) ref␣
↪window not filled
hour4: 1000 events (ref: 3, cur: 1002) - No alert because threshold_ref not met
hour5: 2 events (ref: 4, cur: 1002) - No alert because threshold_ref not met
hour6: 4 events: (ref: 1002, cur: 6) - No alert because spike_height not met

hour1: 1000 events (ref: 0, cur: 1000) - No alert because (a) threshold_ref not met, (b)␣
↪ref window not filled
hour2: 0 events (ref: 0, cur: 1000) - No alert because (a) threshold_ref not met, (b)␣
↪ref window not filled
hour3: 0 events (ref: 1000, cur: 0) - No alert because (a) spike_height not met, (b) ref␣
↪window not filled
hour4: 0 events (ref: 1000, cur: 0) - No alert because spike_height not met
hour5: 1000 events (ref: 0, cur: 1000) - No alert because threshold_ref not met
hour6: 1050 events (ref: 0, cur: 2050)- No alert because threshold_ref not met
hour7: 1075 events (ref: 1000, cur: 2125) Alert because (a) spike_height met, (b)␣
↪threshold_ref met, (c) ref window filled

" Alert if at least 100 events occur within two hours and less than a fifth of that␣
↪number occurred in the previous two hours. "
timeframe: hours: 2
spike_height: 5
spike_type: up
threshold_cur: 100

hour1: 1000 events (ref: 0, cur: 1000) - No alert because ref window not filled

hour1: 2 events (ref: 0, cur: 2) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled
hour2: 1 events (ref: 0, cur: 3) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled
hour3: 20 events (ref: 2, cur: 21) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled
hour4: 81 events (ref: 3, cur: 101) - Alert because (a) spike_height met, (b) threshold_
↪cur met, (c) ref window filled

hour1: 10 events (ref: 0, cur: 10) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled
hour2: 20 events (ref: 0, cur: 30) - No alert because (a) threshold_cur not met, (b) ref␣
↪window not filled
hour3: 40 events (ref: 10, cur: 60) - No alert because (a) threshold_cur not met, (b)␣
↪ref window not filled
hour4: 80 events (ref: 30, cur: 120) - No alert because spike_height not met
hour5: 200 events (ref: 60, cur: 280) - No alert because spike_height not met
```

`alert_on_new_data`: This option is only used if `query_key` is set. When this is set to true, any new `query_key` encountered may trigger an immediate alert. When set to false, baseline must be established for each new `query_key` value, and then subsequent spikes may cause alerts. Baseline is established after `timeframe` has elapsed twice since

first occurrence.

use_count_query: If true, ElastAlert 2 will poll Elasticsearch using the count api, and not download all of the matching documents. This is useful is you care only about numbers and not the actual data. It should also be used if you expect a large number of query hits, in the order of tens of thousands or more.

use_terms_query: If true, ElastAlert 2 will make an aggregation query against Elasticsearch to get counts of documents matching each unique value of query_key. This must be used with query_key. This will only return a maximum of terms_size, default 50, unique terms.

terms_size: When used with use_terms_query, this is the maximum number of terms returned per query. Default is 50.

query_key: Counts of documents will be stored independently for each value of query_key.

3.4.7 Flatline

flatline: This rule matches when the total number of events is under a given threshold for a time period.

This rule requires two additional options:

threshold: The minimum number of events for an alert not to be triggered.

timeframe: The time period that must contain less than threshold events.

Optional:

use_count_query: If true, ElastAlert 2 will poll Elasticsearch using the count api, and not download all of the matching documents. This is useful is you care only about numbers and not the actual data. It should also be used if you expect a large number of query hits, in the order of tens of thousands or more.

use_terms_query: If true, ElastAlert 2 will make an aggregation query against Elasticsearch to get counts of documents matching each unique value of query_key. This must be used with query_key. This will only return a maximum of terms_size, default 50, unique terms.

terms_size: When used with use_terms_query, this is the maximum number of terms returned per query. Default is 50.

query_key: With flatline rule, query_key means that an alert will be triggered if any value of query_key has been seen at least once and then falls below the threshold.

forget_keys: Only valid when used with query_key. If this is set to true, ElastAlert 2 will "forget" about the query_key value that triggers an alert, therefore preventing any more alerts for it until it's seen again.

3.4.8 New Term

new_term: This rule matches when a new value appears in a field that has never been seen before. When ElastAlert 2 starts, it will use an aggregation query to gather all known terms for a list of fields.

This rule requires one additional option:

fields: A list of fields to monitor for new terms. query_key will be used if fields is not set. Each entry in the list of fields can itself be a list. If a field entry is provided as a list, it will be interpreted as a set of fields that compose a composite key used for the ElasticSearch query.

Note: The composite fields may only refer to primitive types, otherwise the initial ElasticSearch query will not properly return the aggregation results, thus causing alerts to fire every time the ElastAlert 2 service initially launches with the

rule. A warning will be logged to the console if this scenario is encountered. However, future alerts will actually work as expected after the initial flurry.

Optional:

`terms_window_size`: The amount of time used for the initial query to find existing terms. No term that has occurred within this time frame will trigger an alert. The default is 30 days.

`window_step_size`: When querying for existing terms, split up the time range into steps of this size. For example, using the default 30 day window size, and the default 1 day step size, 30 invidivdual queries will be made. This helps to avoid timeouts for very expensive aggregation queries. The default is 1 day.

`alert_on_missing_field`: Whether or not to alert when a field is missing from a document. The default is false.

`use_terms_query`: If true, ElastAlert 2 will use aggregation queries to get terms instead of regular search queries. This is faster than regular searching if there is a large number of documents. If this is used, you may only specify a single field, and must also set `query_key` to that field. Also, note that `terms_size` (the number of buckets returned per query) defaults to 50. This means that if a new term appears but there are at least 50 terms which appear more frequently, it will not be found.

Note: When using use_terms_query, make sure that the field you are using is not analyzed. If it is, the results of each terms query may return tokens rather than full values. ElastAlert 2 will by default turn on use_keyword_postfix, which attempts to use the non-analyzed version (.keyword) to gather initial terms. These will not match the partial values and result in false positives.

`use_keyword_postfix`: If true, ElastAlert 2 will automatically try to add .keyword to the fields when making an initial query. These are non-analyzed fields added by Logstash. If the field used is analyzed, the initial query will return only the tokenized values, potentially causing false positives. Defaults to true.

3.4.9 Cardinality

`cardinality`: This rule matches when a the total number of unique values for a certain field within a time frame is higher or lower than a threshold.

This rule requires:

`timeframe`: The time period in which the number of unique values will be counted.

`cardinality_field`: Which field to count the cardinality for.

This rule requires one of the two following options:

`max_cardinality`: If the cardinality of the data is greater than this number, an alert will be triggered. Each new event that raises the cardinality will trigger an alert.

`min_cardinality`: If the cardinality of the data is lower than this number, an alert will be triggered. The `timeframe` must have elapsed since the first event before any alerts will be sent. When a match occurs, the `timeframe` will be reset and must elapse again before additional alerts.

Optional:

`query_key`: Group cardinality counts by this field. For each unique value of the `query_key` field, cardinality will be counted separately.

3.4.10 Metric Aggregation

`metric_aggregation`: This rule matches when the value of a metric within the calculation window is higher or lower than a threshold. By default this is `buffer_time`.

This rule requires:

`metric_agg_key`: This is the name of the field over which the metric value will be calculated. The underlying type of this field must be supported by the specified aggregation type. If using a scripted field via `metric_agg_script`, this is the name for your scripted field

`metric_agg_type`: The type of metric aggregation to perform on the `metric_agg_key` field. This must be one of 'min', 'max', 'avg', 'sum', 'cardinality', 'value_count', 'percentiles'. Note, if *percentiles* is used, then `percentile_range` must also be specified.

Note: When Metric Aggregation has a match, match_body includes an aggregated value that triggered the match so that you can use that on an alert. The value is named based on `metric_agg_key` and `metric_agg_type`. For example, if you set `metric_agg_key` to 'system.cpu.total.norm.pct' and `metric_agg_type` to 'avg', the name of the value is 'metric_system.cpu.total.norm.pct_avg'. Because of this naming rule, you might face conflicts with jinja2 template, and when that happens, you also can use 'metric_agg_value' from match_body instead.

This rule also requires at least one of the two following options:

`max_threshold`: If the calculated metric value is greater than this number, an alert will be triggered. This threshold is exclusive.

`min_threshold`: If the calculated metric value is less than this number, an alert will be triggered. This threshold is exclusive.

`percentile_range`: An integer specifying the percentage value to aggregate against. Must be specified if `metric_agg_type` is set to `percentiles`. See https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-metrics-percentile-aggregation.html for more information.

Optional:

`query_key`: Group metric calculations by this field. For each unique value of the `query_key` field, the metric will be calculated and evaluated separately against the threshold(s).

`metric_agg_script`: A *Painless* formatted script describing how to calculate your metric on-the-fly:

```
metric_agg_key: myScriptedMetric
metric_agg_script:
    script: doc['field1'].value * doc['field2'].value
```

`min_doc_count`: The minimum number of events in the current window needed for an alert to trigger. Used in conjunction with `query_key`, this will only consider terms which in their last `buffer_time` had at least `min_doc_count` records. Default 1.

`use_run_every_query_size`: By default the metric value is calculated over a `buffer_time` sized window. If this parameter is true the rule will use `run_every` as the calculation window.

`allow_buffer_time_overlap`: This setting will only have an effect if `use_run_every_query_size` is false and `buffer_time` is greater than `run_every`. If true will allow the start of the metric calculation window to overlap the end time of a previous run. By default the start and end times will not overlap, so if the time elapsed since the last run is less than the metric calculation window size, rule execution will be skipped (to avoid calculations on partial data).

`bucket_interval`: If present this will divide the metric calculation window into `bucket_interval` sized segments. The metric value will be calculated and evaluated against the threshold(s) for each segment. If

`bucket_interval` is specified then `buffer_time` must be a multiple of `bucket_interval`. (Or `run_every` if `use_run_every_query_size` is true).

`sync_bucket_interval`: This only has an effect if `bucket_interval` is present. If true it will sync the start and end times of the metric calculation window to the keys (timestamps) of the underlying date_histogram buckets. Because of the way elasticsearch calculates date_histogram bucket keys these usually round evenly to nearest minute, hour, day etc (depending on the bucket size). By default the bucket keys are offset to allign with the time ElastAlert 2 runs, (This both avoid calculations on partial data, and ensures the very latest documents are included). See: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-datehistogram-aggregation.html#_offset for a more comprehensive explaination.

`metric_format_string`: An optional format string applies to the aggregated metric value in the alert match text and match_body. This adds 'metric_{metric_agg_key}_formatted' value to the match_body in addition to raw, unformatted 'metric_{metric_agg_key}' value so that you can use the values for `alert_subject_args` and `alert_text_args`. Must be a valid python format string. Both str.format() and %-format syntax works. For example, "{:.2%}" will format '0.966666667' to '96.67%', and "%.2f" will format '0.966666667' to '0.97'. See: https://docs.python.org/3.4/library/string.html#format-specification-mini-language

3.4.11 Spike Aggregation

`spike_aggregation`: This rule matches when the value of a metric within the calculation window is `spike_height` times larger or smaller than during the previous time period. It uses two sliding windows to compare the current and reference metric values. We will call these two windows "reference" and "current".

This rule requires:

`metric_agg_key`: This is the name of the field over which the metric value will be calculated. The underlying type of this field must be supported by the specified aggregation type. If using a scripted field via `metric_agg_script`, this is the name for your scripted field

`metric_agg_type`: The type of metric aggregation to perform on the `metric_agg_key` field. This must be one of 'min', 'max', 'avg', 'sum', 'cardinality', 'value_count', 'percentiles'. Note, if *percentiles* is used, then `percentile_range` must also be specified.

`spike_height`: The ratio of the metric value in the last `timeframe` to the previous `timeframe` that when hit will trigger an alert.

`spike_type`: Either 'up', 'down' or 'both'. 'Up' meaning the rule will only match when the metric value is `spike_height` times higher. 'Down' meaning the reference metric value is `spike_height` higher than the current metric value. 'Both' will match either.

`buffer_time`: The rule will average out the rate of events over this time period. For example, `hours: 1` means that the 'current' window will span from present to one hour ago, and the 'reference' window will span from one hour ago to two hours ago. The rule will not be active until the time elapsed from the first event is at least two timeframes. This is to prevent an alert being triggered before a baseline rate has been established. This can be overridden using `alert_on_new_data`.

`percentile_range`: An integer specifying the percentage value to aggregate against. Must be specified if `metric_agg_type` is set to `percentiles`. See https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-metrics-percentile-aggregation.html for more information.

Optional:

`query_key`: Group metric calculations by this field. For each unique value of the `query_key` field, the metric will be calculated and evaluated separately against the 'reference'/'current' metric value and `spike height`.

`metric_agg_script`: A *Painless* formatted script describing how to calculate your metric on-the-fly:

```
metric_agg_key: myScriptedMetric
metric_agg_script:
    script: doc['field1'].value * doc['field2'].value
```

threshold_ref: The minimum value of the metric in the reference window for an alert to trigger. For example, if spike_height: 3 and threshold_ref: 10, then the 'reference' window must have a metric value of 10 and the 'current' window at least three times that for an alert to be triggered.

threshold_cur: The minimum value of the metric in the current window for an alert to trigger. For example, if spike_height: 3 and threshold_cur: 60, then an alert will occur if the current window has a metric value greater than 60 and the reference window is less than a third of that value.

min_doc_count: The minimum number of events in the current window needed for an alert to trigger. Used in conjunction with query_key, this will only consider terms which in their last buffer_time had at least min_doc_count records. Default 1.

3.4.12 Percentage Match

percentage_match: This rule matches when the percentage of document in the match bucket within a calculation window is higher or lower than a threshold. By default the calculation window is buffer_time.

This rule requires:

match_bucket_filter: ES filter DSL. This defines a filter for the match bucket, which should match a subset of the documents returned by the main query filter.

ssThis rule also requires at least one of the two following options:

min_percentage: If the percentage of matching documents is less than this number, an alert will be triggered.

max_percentage: If the percentage of matching documents is greater than this number, an alert will be triggered.

Optional:

query_key: Group percentage by this field. For each unique value of the query_key field, the percentage will be calculated and evaluated separately against the threshold(s).

use_run_every_query_size: See use_run_every_query_size in Metric Aggregation rule

allow_buffer_time_overlap: See allow_buffer_time_overlap in Metric Aggregation rule

bucket_interval: See bucket_interval in Metric Aggregation rule

sync_bucket_interval: See sync_bucket_interval in Metric Aggregation rule

percentage_format_string: An optional format string applies to the percentage value in the alert match text and match_body. This adds 'percentage_formatted' value to the match_body in addition to raw, unformatted 'percentage' value so that you can use the values for alert_subject_args and alert_text_args. Must be a valid python format string. Both str.format() and %-format syntax works. For example, both "{:.2f}" and "%.2f" will format '96.6666667' to '96.67'. See: https://docs.python.org/3.4/library/string.html#format-specification-mini-language

min_denominator: Minimum number of documents on which percentage calculation will apply. Default is 0.

3.5 Alerts

Each rule may have any number of alerts attached to it. Alerts are subclasses of `Alerter` and are passed a dictionary, or list of dictionaries, from ElastAlert 2 which contain relevant information. They are configured in the rule configuration file similarly to rule types.

To set the alerts for a rule, set the `alert` option to the name of the alert, or a list of the names of alerts:

```
alert:  email
```

or

```
alert:
  - alerta
  - alertmanager
  - chatwork
  - command
  - datadog
  - debug
  - dingtalk
  - discord
  - email
  - exotel
  - gitter
  - googlechat
  - hivealerter
  - jira
  - linenotify
  - mattermost
  - ms_teams
  - opsgenie
  - pagerduty
  - pagertree
  - post
  - post2
  - rocketchat
  - servicenow
  - ses
  - slack
  - sns
  - stomp
  - telegram
  - tencent_sms
  - twilio
  - victorops
  - zabbix
```

Options for each alerter can either defined at the top level of the YAML file, or nested within the alert name, allowing for different settings for multiple of the same alerter. For example, consider sending multiple emails, but with different 'To' and 'From' fields:

```
alert:
 - email
from_addr: "no-reply@example.com"
email: "customer@example.com"
```

versus

```
alert:
 - email:
     from_addr: "no-reply@example.com"
     email: "customer@example.com"
 - email:
     from_addr: "elastalert@example.com""
     email: "devs@example.com"
```

If multiple of the same alerter type are used, top level settings will be used as the default and inline settings will override those for each alerter.

3.5.1 Alert Subject

E-mail subjects, Jira issue summaries, PagerDuty alerts, or any alerter that has a "subject" can be customized by adding an `alert_subject` that contains a custom summary. It can be further formatted using standard Python formatting syntax:

```
alert_subject: "Issue {0} occurred at {1}"
```

The arguments for the formatter will be fed from the matched objects related to the alert. The field names whose values will be used as the arguments can be passed with `alert_subject_args`:

```
alert_subject_args:
- issue.name
- "@timestamp"
```

It is mandatory to enclose the `@timestamp` field in quotes since in YAML format a token cannot begin with the `@` character. Not using the quotation marks will trigger a YAML parse error.

In case the rule matches multiple objects in the index, only the first match is used to populate the arguments for the formatter.

If the field(s) mentioned in the arguments list are missing, the email alert will have the text `alert_missing_value` in place of its expected value. This will also occur if `use_count_query` is set to true.

3.5.2 Alert Content

There are several ways to format the body text of the various types of events. In EBNF:

```
rule_name           = name
alert_text          = alert_text
ruletype_text       = Depends on type
top_counts_header   = top_count_key, ":"
top_counts_value    = Value, ": ", Count
top_counts          = top_counts_header, LF, top_counts_value
field_values        = Field, ": ", Value
```

Similarly to `alert_subject`, `alert_text` can be further formatted using Jinja2 Templates or Standard Python Formatting Syntax

1. Jinja Template

By setting `alert_text_type: alert_text_jinja` you can use jinja2 templates in `alert_text` and `alert_subject`.

```
alert_text_type: alert_text_jinja

alert_text: |
  Alert triggered! *({{num_hits}} Matches!)*
  Something happened with {{username}} ({{email}})
  {{description|truncate}}
```

Top fields are accessible via *{{field_name}}* or *{{_data['field_name']}}*, *_data* is useful when accessing *fields with dots in their keys*, as Jinja treat dot as a nested field. If *_data* conflicts with your top level data, use `jinja_root_name` to change its name.

2. Standard Python Formatting Syntax

The field names whose values will be used as the arguments can be passed with `alert_text_args` or `alert_text_kw`. You may also refer to any top-level rule property in the `alert_subject_args`, `alert_text_args`, `alert_missing_value`, and `alert_text_kw fields`. However, if the matched document has a key with the same name, that will take preference over the rule property.

```
alert_text: "Something happened with {0} at {1}"
alert_text_type: alert_text_only
alert_text_args: ["username", "@timestamp"]
```

By default:

```
body                = rule_name

                      [alert_text]

                      ruletype_text

                      {top_counts}

                      {field_values}
```

With `alert_text_type: alert_text_only`:

```
body                = rule_name

                      alert_text
```

With `alert_text_type: alert_text_jinja`:

```
body                = rule_name

                      alert_text
```

With `alert_text_type: exclude_fields`:

```
body                = rule_name

                      [alert_text]
```

(continues on next page)

(continued from previous page)

```
                    ruletype_text

                    {top_counts}
```

With `alert_text_type: aggregation_summary_only`:

```
body                = rule_name

                    aggregation_summary
```

ruletype_text is the string returned by RuleType.get_match_str.

field_values will contain every key value pair included in the results from Elasticsearch. These fields include "@timestamp" (or the value of `timestamp_field`), every key in `include`, every key in `top_count_keys`, `query_key`, and `compare_key`. If the alert spans multiple events, these values may come from an individual event, usually the one which triggers the alert.

When using `alert_text_args`, you can access nested fields and index into arrays. For example, if your match was `{"data": {"ips": ["127.0.0.1", "12.34.56.78"]}}`, then by using `"data.ips[1]"` in `alert_text_args`, it would replace value with `"12.34.56.78"`. This can go arbitrarily deep into fields and will still work on keys that contain dots themselves.

3.5.3 Alerter

For all Alerter subclasses, you may reference values from a top-level rule property in your Alerter fields by referring to the property name surrounded by dollar signs. This can be useful when you have rule-level properties that you would like to reference many times in your alert. For example:

Example usage:

```
jira_priority: $priority$
jira_alert_owner: $owner$
```

3.5.4 Alerta

Alerta alerter will post an alert in the Alerta server instance through the alert API endpoint. See https://docs.alerta.io/en/latest/api/alert.html for more details on the Alerta JSON format.

For Alerta 5.0

Required:

`alerta_api_url`: API server URL.

Optional:

`alerta_api_key`: This is the api key for alerta server, sent in an `Authorization` HTTP header. If not defined, no Authorization header is sent.

`alerta_use_qk_as_resource`: If true and query_key is present, this will override `alerta_resource` field with the `query_key value` (Can be useful if `query_key` is a hostname).

`alerta_use_match_timestamp`: If true, it will use the timestamp of the first match as the `createTime` of the alert. otherwise, the current server time is used.

`alerta_api_skip_ssl`: Defaults to False.

alert_missing_value: Text to replace any match field not found when formating strings. Defaults to <MISSING_TEXT>.

The following options dictate the values of the API JSON payload:

alerta_severity: Defaults to "warning".

alerta_timeout: Defaults 84600 (1 Day).

alerta_type: Defaults to "elastalert".

The following options use Python-like string syntax {<field>} or %(<field>)s to access parts of the match, similar to the CommandAlerter. Ie: "Alert for {clientip}". If the referenced key is not found in the match, it is replaced by the text indicated by the option alert_missing_value.

alerta_resource: Defaults to "elastalert".

alerta_service: Defaults to "elastalert".

alerta_origin: Defaults to "elastalert".

alerta_environment: Defaults to "Production".

alerta_group: Defaults to "".

alerta_correlate: Defaults to an empty list.

alerta_tags: Defaults to an empty list.

alerta_event: Defaults to the rule's name.

alerta_text: Defaults to the rule's text according to its type.

alerta_value: Defaults to "".

The attributes dictionary is built by joining the lists from alerta_attributes_keys and alerta_attributes_values, considered in order.

Example usage using old-style format:

```
alert:
  - alerta
alerta_api_url: "http://youralertahost/api/alert"
alerta_attributes_keys:   ["hostname",   "TimestampEvent",  "senderIP" ]
alerta_attributes_values: ["%(key)s",    "%(logdate)s",     "%(sender_ip)s"  ]
alerta_correlate: ["ProbeUP","ProbeDOWN"]
alerta_event: "ProbeUP"
alerta_text:  "Probe %(hostname)s is UP at %(logdate)s GMT"
alerta_value: "UP"
```

Example usage using new-style format:

```
alert:
  - alerta
alerta_attributes_values: ["{key}",    "{logdate}",     "{sender_ip}"  ]
alerta_text:  "Probe {hostname} is UP at {logdate} GMT"
```

3.5.5 Alertmanager

This alert type will send alerts to Alertmanager postAlerts. `alert_subject` and `alert_text` are passed as the annotations labeled `summary` and `description` accordingly. The labels can be changed. See https://prometheus.io/docs/alerting/clients/ for more details about the Alertmanager alert format.

Required:

`alertmanager_hosts`: The list of hosts pointing to the Alertmanager.

Optional:

`alertmanager_api_version`: Defaults to *v1*. Set to *v2* to enable the Alertmanager V2 API postAlerts.

`alertmanager_alertname`: `alertname` is the only required label. Defaults to using the rule name of the alert.

`alertmanager_labels`: Key:value pairs of arbitrary labels to be attached to every alert. Keys should match the regular expression `^[a-zA-Z_][a-zA-Z0-9_]*$`.

`alertmanager_annotations`: Key:value pairs of arbitrary annotations to be attached to every alert. Keys should match the regular expression `^[a-zA-Z_][a-zA-Z0-9_]*$`.

`alertmanager_fields`: Key:value pairs of labels and corresponding match fields. When using `alertmanager_fields` you can access nested fields and index into arrays the same way as with `alert_text_args`. Keys should match the regular expression `^[a-zA-Z_][a-zA-Z0-9_]*$`. This dictionary will be merged with the `alertmanager_labels`.

`alertmanager_alert_subject_labelname`: Rename the annotations' label name for `alert_subject`. Default is `summary`.

`alertmanager_alert_text_labelname`: Rename the annotations' label name for `alert_text`. Default is `description`.

`alertmanager_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to Alertmanager. Set this option using `hostname:port` if you need to use a proxy. only supports https.

`alertmanager_ca_certs`: Set this option to `True` if you want to validate the SSL certificate.

`alertmanager_ignore_ssl_errors`: By default ElastAlert 2 will verify SSL certificate. Set this option to `True` if you want to ignore SSL errors.

`alertmanager_timeout`: You can specify a timeout value, in seconds, for making communicating with Alertmanager. The default is 10. If a timeout occurs, the alert will be retried next time ElastAlert 2 cycles.

`alertmanager_basic_auth_login`: Basic authentication username.

`alertmanager_basic_auth_password`: Basic authentication password.

Example usage:

```
alert:
  - "alertmanager"
alertmanager_hosts:
  - "http://alertmanager:9093"
alertmanager_alertname: "Title"
alertmanager_annotations:
  severity: "error"
alertmanager_labels:
  source: "elastalert"
alertmanager_fields:
  msg: "message"
  log: "@log_name"
```

3.5.6 AWS SES (Amazon Simple Email Service)

The AWS SES alerter is similar to Email alerter but uses AWS SES to send emails. The AWS SES alerter can use AWS credentials from the rule yaml, standard AWS config files or environment variables.

AWS SES requires one option:

`ses_email`: An address or list of addresses to sent the alert to.

single address example:

```
ses_email: "one@domain"
```

or

multiple address example:

```
ses_email:
  - "one@domain"
  - "two@domain"
```

`ses_from_addr`: This sets the From header in the email.

Optional:

`ses_aws_access_key`: An access key to connect to AWS SES with.

`ses_aws_secret_key`: The secret key associated with the access key.

`ses_aws_region`: The AWS region in which the AWS SES resource is located. Default is us-east-1

`ses_aws_profile`: The AWS profile to use. If none specified, the default will be used.

`ses_email_reply_to`: This sets the Reply-To header in the email.

`ses_cc`: This adds the CC emails to the list of recipients. By default, this is left empty.

single address example:

```
ses_cc: "one@domain"
```

or

multiple address example:

```
ses_cc:
  - "one@domain"
  - "two@domain"
```

`ses_bcc`: This adds the BCC emails to the list of recipients but does not show up in the email message. By default, this is left empty.

single address example:

```
ses_bcc: "one@domain"
```

or

multiple address example:

```
ses_bcc:
  - "one@domain"
  - "two@domain"
```

Example When not using aws_profile usage:

```
alert:
  - "ses"
ses_aws_access_key_id: "XXXXXXXXXXXXXXXXXX'"
ses_aws_secret_access_key: "YYYYYYYYYYYYYYYYYYYY"
ses_aws_region: "us-east-1"
ses_from_addr: "xxxx1@xxx.com"
ses_email: "xxxx1@xxx.com"
```

Example When to use aws_profile usage:

```
# Create ~/.aws/credentials

[default]
aws_access_key_id = xxxxxxxxxxxxxxxxxxx
aws_secret_access_key = yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy

# Create ~/.aws/config

[default]
region = us-east-1

# alert rule setting

alert:
  - "ses"
ses_aws_profile: "default"
ses_from_addr: "xxxx1@xxx.com"
ses_email: "xxxx1@xxx.com"
```

3.5.7 AWS SNS (Amazon Simple Notification Service)

The AWS SNS alerter will send an AWS SNS notification. The body of the notification is formatted the same as with other alerters. The AWS SNS alerter uses boto3 and can use credentials in the rule yaml, in a standard AWS credential and config files, or via environment variables. See http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html for details.

AWS SNS requires one option:

`sns_topic_arn`: The SNS topic's ARN. For example, `arn:aws:sns:us-east-1:123456789:somesnstopic`

Optional:

`sns_aws_access_key_id`: An access key to connect to SNS with.

`sns_aws_secret_access_key`: The secret key associated with the access key.

`sns_aws_region`: The AWS region in which the SNS resource is located. Default is us-east-1

`sns_aws_profile`: The AWS profile to use. If none specified, the default will be used.

Example When not using aws_profile usage:

```
alert:
  - sns
sns_topic_arn: 'arn:aws:sns:us-east-1:123456789:somesnstopic'
sns_aws_access_key_id: 'XXXXXXXXXXXXXXXXXX''
sns_aws_secret_access_key: 'YYYYYYYYYYYYYYYYYYYY'
sns_aws_region: 'us-east-1' # You must nest aws_region within your alert configuration␣
↪so it is not used to sign AWS requests.
```

Example When to use aws_profile usage:

```
# Create ~/.aws/credentials

[default]
aws_access_key_id = xxxxxxxxxxxxxxxxxxx
aws_secret_access_key = yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy

# Create ~/.aws/config

[default]
region = us-east-1

# alert rule setting

alert:
  - sns
sns_topic_arn: 'arn:aws:sns:us-east-1:123456789:somesnstopic'
sns_aws_profile: 'default'
```

3.5.8 Chatwork

Chatwork will send notification to a Chatwork application. The body of the notification is formatted the same as with other alerters.

Required:

`chatwork_apikey`: Chatwork API KEY.

`chatwork_room_id`: The ID of the room you are talking to in Chatwork. How to find the room ID is the part of the number after "rid" at the end of the URL of the browser.

`chatwork_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to Chatwork. Set this option using `hostname:port` if you need to use a proxy. only supports https.

`chatwork_proxy_login`: The Chatwork proxy auth username.

`chatwork_proxy_pass`: The Chatwork proxy auth password.

Example usage:

```
alert:
  - "chatwork"
chatwork_apikey: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
chatwork_room_id: "xxxxxxxxx"
```

3.5.9 Command

The command alert allows you to execute an arbitrary command and pass arguments or stdin from the match. Arguments to the command can use Python format string syntax to access parts of the match. The alerter will open a subprocess and optionally pass the match, or matches in the case of an aggregated alert, as a JSON array, to the stdin of the process.

This alert requires one option:

`command`: A list of arguments to execute or a string to execute. If in list format, the first argument is the name of the program to execute. If passed a string, the command is executed through the shell.

Strings can be formatted using the old-style format (`%`) or the new-style format (`.format()`). When the old-style format is used, fields are accessed using `%(field_name)s`, or `%(field.subfield)s`. When the new-style format is used, fields are accessed using `{field_name}`. New-style formatting allows accessing nested fields (e.g., `{field_1[subfield]}`).

In an aggregated alert, these fields come from the first match.

Optional:

`pipe_match_json`: If true, the match will be converted to JSON and passed to stdin of the command. Note that this will cause ElastAlert 2 to block until the command exits or sends an EOF to stdout.

`pipe_alert_text`: If true, the standard alert body text will be passed to stdin of the command. Note that this will cause ElastAlert 2 to block until the command exits or sends an EOF to stdout. It cannot be used at the same time as `pipe_match_json`.

`fail_on_non_zero_exit`: By default this is `False`. Allows monitoring of when commands fail to run. When a command returns a non-zero exit status, the alert raises an exception.

Example usage using old-style format:

```
alert:
  - command
command: ["/bin/send_alert", "--username", "%(username)s"]
```

Warning: Executing commmands with untrusted data can make it vulnerable to shell injection! If you use formatted data in your command, it is highly recommended that you use a args list format instead of a shell string.

Example usage using new-style format:

```
alert:
  - command
command: ["/bin/send_alert", "--username", "{match[username]}"]
```

3.5.10 Datadog

This alert will create a Datadog Event. Events are limited to 4000 characters. If an event is sent that contains a message that is longer than 4000 characters, only his first 4000 characters will be displayed.

This alert requires two additional options:

`datadog_api_key`: Datadog API key

`datadog_app_key`: Datadog application key

Example usage:

```
alert:
  - "datadog"
datadog_api_key: "Datadog API Key"
datadog_app_key: "Datadog APP Key"
```

3.5.11 Debug

The debug alerter will log the alert information using the Python logger at the info level. It is logged into a Python Logger object with the name `elastalert` that can be easily accessed using the `getLogger` command.

3.5.12 Dingtalk

Dingtalk will send notification to a Dingtalk application. The body of the notification is formatted the same as with other alerters.

Required:

`dingtalk_access_token`: Dingtalk access token.

`dingtalk_msgtype`: Dingtalk msgtype, default to `text`. `markdown`, `single_action_card`, `action_card`.

dingtalk_msgtype single_action_card Required:

`dingtalk_single_title`: The title of a single button..

`dingtalk_single_url`: Jump link for a single button.

dingtalk_msgtype action_card Required:

`dingtalk_btns`: Button.

dingtalk_msgtype action_card Optional:

`dingtalk_btn_orientation`: “0”: Buttons are arranged vertically “1”: Buttons are arranged horizontally.

Example msgtype : text:

```
alert:
  - "dingtalk"
dingtalk_access_token: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
dingtalk_msgtype: "text"
```

Example msgtype : markdown:

```
alert:
  - "dingtalk"
dingtalk_access_token: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
dingtalk_msgtype: "markdown"
```

Example msgtype : single_action_card:

```
alert:
  - "dingtalk"
dingtalk_access_token: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
dingtalk_msgtype: "single_action_card"
dingtalk_single_title: "test3"
dingtalk_single_url: "https://xxxx.xxx"
```

Example msgtype : action_card:

```
alert:
  - "dingtalk"
dingtalk_access_token: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
dingtalk_msgtype: "action_card"
dingtalk_btn_orientation: "0"
dingtalk_btns: [{"title": "a", "actionURL": "https://xxxx1.xxx"}, {"title": "b",
↪"actionURL": "https://xxxx2.xxx"}]
```

Optional:

`dingtalk_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to Dingtalk. Set this option using `hostname:port` if you need to use a proxy. only supports https.

`dingtalk_proxy_login`: The Dingtalk proxy auth username.

`dingtalk_proxy_pass`: The Dingtalk proxy auth username.

3.5.13 Discord

Discord will send notification to a Discord application. The body of the notification is formatted the same as with other alerters.

Required:

`discord_webhook_url`: The webhook URL.

Optional:

`discord_emoji_title`: By default ElastAlert 2 will use the `:warning:` emoji when posting to the channel. You can use a different emoji per ElastAlert 2 rule. Any Apple emoji can be used, see http://emojipedia.org/apple/ . If discord_embed_icon_url parameter is provided, emoji is ignored.

`discord_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to Discord. Set this option using `hostname:port` if you need to use a proxy. only supports https.

`discord_proxy_login`: The Discord proxy auth username.

`discord_proxy_password`: The Discord proxy auth username.

`discord_embed_color`: embed color. By default `0xffffff`.

`discord_embed_footer`: embed footer.

`discord_embed_icon_url`: You can provide icon_url to use custom image. Provide absolute address of the pciture.

Example usage:

```
alert:
- "discord"
discord_webhook_url: "Your discord webhook url"
discord_emoji_title: ":lock:"
discord_embed_color: 0xE24D42
discord_embed_footer: "Message sent by  from your computer"
discord_embed_icon_url: "https://humancoders-formations.s3.amazonaws.com/uploads/course/
↪logo/38/thumb_bigger_formation-elasticsearch.png"
```

3.5.14 Email

This alert will send an email. It connects to an smtp server located at `smtp_host`, or localhost by default. If available, it will use STARTTLS.

This alert requires one additional option:

`email`: An address or list of addresses to sent the alert to.

single address example:

```
email: "one@domain"
```

or

multiple address example:

```
email:
  - "one@domain"
  - "two@domain"
```

Optional:

`email_from_field`: Use a field from the document that triggered the alert as the recipient. If the field cannot be found, the `email` value will be used as a default. Note that this field will not be available in every rule type, for example, if you have `use_count_query` or if it's `type: flatline`. You can optionally add a domain suffix to the field to generate the address using `email_add_domain`. It can be a single recipient or list of recipients. For example, with the following settings:

```
email_from_field: "data.user"
email_add_domain: "@example.com"
```

and a match `{"@timestamp": "2017", "data": {"foo": "bar", "user": "qlo"}}`

an email would be sent to `qlo@example.com`

`smtp_host`: The SMTP host to use, defaults to localhost.

`smtp_port`: The port to use. Defaults to port 25 when SSL is not used, or 465 when SSL is used.

`smtp_ssl`: Connect the SMTP host using TLS, defaults to `false`. If `smtp_ssl` is not used, ElastAlert 2 will still attempt STARTTLS.

`smtp_auth_file`: The path to a file which contains SMTP authentication credentials. The path can be either absolute or relative to the given rule. It should be YAML formatted and contain two fields, `user` and `password`. If this is not present, no authentication will be attempted.

`smtp_cert_file`: Connect the SMTP host using the given path to a TLS certificate file, default to `None`.

`smtp_key_file`: Connect the SMTP host using the given path to a TLS key file, default to `None`.

`email_reply_to`: This sets the Reply-To header in the email. By default, the from address is ElastAlert@ and the domain will be set by the smtp server.

`from_addr`: This sets the From header in the email. By default, the from address is ElastAlert@ and the domain will be set by the smtp server.

`cc`: This adds the CC emails to the list of recipients. By default, this is left empty.

single address example:

```
cc: "one@domain"
```

or

multiple address example:

```
cc:
  - "one@domain"
  - "two@domain"
```

bcc: This adds the BCC emails to the list of recipients but does not show up in the email message. By default, this is left empty.

single address example:

```
bcc: "one@domain"
```

or

multiple address example:

```
bcc:
  - "one@domain"
  - "two@domain"
```

email_format: If set to 'html', the email's MIME type will be set to HTML, and HTML content should correctly render. If you use this, you need to put your own HTML into alert_text and use alert_text_type: alert_text_jinja Or alert_text_type: alert_text_only.

assets_dir: images dir. default to /tmp.

email_image_keys: mapping between images keys.

email_image_values: mapping between images values

Example assets_dir, email_image_keys, email_image_values:

```
assets_dir: "/opt/elastalert/email_images"
email_image_keys: ["img1"]
email_image_values: ["my_logo.png"]
```

3.5.15 Exotel

Developers in India can use the Exotel alerter, which can send an alert to a mobile phone as an SMS from your ExoPhone. The SMS will contain both the alert name and the specified message body.

The alerter requires the following option:

exotel_account_sid: The SID of your Exotel account.

exotel_auth_token: The auth token associated with your Exotel account.

Instructions for finding the SID and auth token associated with your account can be found on the Exotel website.

exotel_to_number: The phone number to which you would like to send the alert.

exotel_from_number: The ExoPhone number from which the alert will be sent.

The alerter has one optional argument:

exotel_message_body: The contents of the SMS. If you don't specify this argument, only the rule name is sent.

Example usage:

```
alert:
  - "exotel"
exotel_account_sid: "Exotel Account SID"
exotel_auth_token: "Exotel Auth token"
exotel_to_number: "Exotel to number"
exotel_from_number: "Exotel from number"
```

3.5.16 Gitter

Gitter alerter will send a notification to a predefined Gitter channel. The body of the notification is formatted the same as with other alerters.

The alerter requires the following option:

`gitter_webhook_url`: The webhook URL that includes your auth data and the ID of the channel (room) you want to post to. Go to the Integration Settings of the channel https://gitter.im/ORGA/CHANNEL#integrations , click 'CUSTOM' and copy the resulting URL.

Optional:

`gitter_msg_level`: By default the alert will be posted with the 'error' level. You can use 'info' if you want the messages to be black instead of red.

`gitter_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to Gitter. Set this option using `hostname:port` if you need to use a proxy. only supports https.

Example usage:

```
alert:
  - "gitter"
gitter_webhook_url: "Your Gitter Webhook URL"
gitter_msg_level: "error"
```

3.5.17 GoogleChat

GoogleChat alerter will send a notification to a predefined GoogleChat channel. The body of the notification is formatted the same as with other alerters.

The alerter requires the following options:

`googlechat_webhook_url`: The webhook URL that includes the channel (room) you want to post to. Go to the Google Chat website https://chat.google.com and choose the channel in which you wish to receive the notifications. Select 'Configure Webhooks' to create a new webhook or to copy the URL from an existing one. You can use a list of URLs to send to multiple channels.

Optional:

`googlechat_format`: Formatting for the notification. Can be either 'card' or 'basic' (default).

`googlechat_header_title`: Sets the text for the card header title. (Only used if format=card)

`googlechat_header_subtitle`: Sets the text for the card header subtitle. (Only used if format=card)

`googlechat_header_image`: URL for the card header icon. (Only used if format=card)

`googlechat_footer_kibanalink`: URL to Kibana to include in the card footer. (Only used if format=card)

3.5.18 HTTP POST

This alert type will send results to a JSON endpoint using HTTP POST. The key names are configurable so this is compatible with almost any endpoint. By default, the JSON will contain all the items from the match, unless you specify http_post_payload, in which case it will only contain those items.

Required:

`http_post_url`: The URL to POST.

Optional:

`http_post_payload`: List of keys:values to use as the content of the POST. Example - ip:clientip will map the value from the clientip index of Elasticsearch to JSON key named ip. If not defined, all the Elasticsearch keys will be sent.

`http_post_static_payload`: Key:value pairs of static parameters to be sent, along with the Elasticsearch results. Put your authentication or other information here.

`http_post_headers`: Key:value pairs of headers to be sent as part of the request.

`http_post_proxy`: URL of proxy, if required. only supports https.

`http_post_all_values`: Boolean of whether or not to include every key value pair from the match in addition to those in http_post_payload and http_post_static_payload. Defaults to True if http_post_payload is not specified, otherwise False.

`http_post_timeout`: The timeout value, in seconds, for making the post. The default is 10. If a timeout occurs, the alert will be retried next time elastalert cycles.

`http_post_ca_certs`: Set this option to `True` if you want to validate the SSL certificate.

`http_post_ignore_ssl_errors`: By default ElastAlert 2 will verify SSL certificate. Set this option to `True` if you want to ignore SSL errors.

Example usage:

```
alert: post
http_post_url: "http://example.com/api"
http_post_payload:
  ip: clientip
http_post_static_payload:
  apikey: abc123
http_post_headers:
  authorization: Basic 123dr3234
```

3.5.19 HTTP POST 2

This alert type will send results to a JSON endpoint using HTTP POST. The key names are configurable so this is compatible with almost any endpoint. By default, the JSON will contain all the items from the match, unless you specify http_post_payload, in which case it will only contain those items. This alert is a more flexible version of the HTTP Post alerter.

Required:

`http_post2_url`: The URL to POST.

Optional:

`http_post2_payload`: List of keys:values to use for the payload of the HTTP Post. You can use {{ field }} (Jinja2 template) in the key and the value to reference any field in the matched events (works for nested ES fields and nested

payload keys). If not defined, all the Elasticsearch keys will be sent. Ex: *"description_{{ my_field }}": "Type: {{ type }}\nSubject: {{ title }}"*.

`http_post2_raw_fields`: List of keys:values to use as the content of the POST. Example - ip:clientip will map the value from the clientip field of Elasticsearch to JSON key named ip. This field overwrite the keys with the same name in *http_post2_payload*.

`http_post2_headers`: List of keys:values to use for as headers of the HTTP Post. You can use {{ field }} (Jinja2 template) in the key and the value to reference any field in the matched events (works for nested fields). Ex: *"Authorization": "{{ user }}"*. Headers *"Content-Type": "application/json"* and *"Accept": "application/json;charset=utf-8"* are present by default, you can overwrite them if you think this is necessary.

`http_post2_proxy`: URL of proxy, if required. only supports https.

`http_post2_all_values`: Boolean of whether or not to include every key value pair from the match in addition to those in http_post2_payload and http_post2_static_payload. Defaults to True if http_post2_payload is not specified, otherwise False.

`http_post2_timeout`: The timeout value, in seconds, for making the post. The default is 10. If a timeout occurs, the alert will be retried next time elastalert cycles.

`http_post2_ca_certs`: Set this option to `True` if you want to validate the SSL certificate.

`http_post2_ignore_ssl_errors`: By default ElastAlert 2 will verify SSL certificate. Set this option to `True` if you want to ignore SSL errors.

Example usage:

```
alert: post2
http_post2_url: "http://example.com/api"
http_post2_payload:
  description: "An event came from IP {{clientip}}"
  username: "{{user.name}}"
http_post2_raw_fields:
  ip: clientip
http_post2_headers:
  authorization: Basic 123dr3234
  X-custom-type: {{type}}
```

3.5.20 Jira

The Jira alerter will open a ticket on Jira whenever an alert is triggered. You must have a service account for ElastAlert 2 to connect with. The credentials of the service account are loaded from a separate file. Credentials can either be username and password or the Personal Access Token. The ticket number will be written to the alert pipeline, and if it is followed by an email alerter, a link will be included in the email.

This alert requires four additional options:

`jira_server`: The hostname of the Jira server.

`jira_project`: The project to open the ticket under.

`jira_issuetype`: The type of issue that the ticket will be filed as. Note that this is case sensitive.

`jira_account_file`: The path to the file which contains Jira account credentials.

> For an example Jira account file, see `examples/rules/jira_acct.yaml`. The account file is a YAML formatted file.
>
> When using user/password authentication, the Jira account file must contain two fields:

`user`: The username to authenticate with Jira.

`password`: The password to authenticate with Jira.

When using a Personal Access Token, the Jira account file must contain a single field:

`apikey`: The Personal Access Token for authenticating with Jira.

Optional:

`jira_assignee`: Assigns an issue to a user.

`jira_component`: The name of the component or components to set the ticket to. This can be a single string or a list of strings. This is provided for backwards compatibility and will eventually be deprecated. It is preferable to use the plural `jira_components` instead.

`jira_components`: The name of the component or components to set the ticket to. This can be a single string or a list of strings.

`jira_description`: Similar to `alert_text`, this text is prepended to the Jira description.

`jira_label`: The label or labels to add to the Jira ticket. This can be a single string or a list of strings. This is provided for backwards compatibility and will eventually be deprecated. It is preferable to use the plural `jira_labels` instead.

`jira_labels`: The label or labels to add to the Jira ticket. This can be a single string or a list of strings.

`jira_priority`: The index of the priority to set the issue to. In the Jira dropdown for priorities, 0 would represent the first priority, 1 the 2nd, etc.

`jira_watchers`: A list of user names to add as watchers on a Jira ticket. This can be a single string or a list of strings.

`jira_bump_tickets`: If true, ElastAlert 2 search for existing tickets newer than `jira_max_age` and comment on the ticket with information about the alert instead of opening another ticket. ElastAlert 2 finds the existing ticket by searching by summary. If the summary has changed or contains special characters, it may fail to find the ticket. If you are using a custom `alert_subject`, the two summaries must be exact matches, except by setting `jira_ignore_in_title`, you can ignore the value of a field when searching. For example, if the custom subject is "foo occured at bar", and "foo" is the value field X in the match, you can set `jira_ignore_in_title` to "X" and it will only bump tickets with "bar" in the subject. Defaults to false.

`jira_ignore_in_title`: ElastAlert 2 will attempt to remove the value for this field from the Jira subject when searching for tickets to bump. See `jira_bump_tickets` description above for an example.

`jira_max_age`: If `jira_bump_tickets` is true, the maximum age of a ticket, in days, such that ElastAlert 2 will comment on the ticket instead of opening a new one. Default is 30 days.

`jira_bump_not_in_statuses`: If `jira_bump_tickets` is true, a list of statuses the ticket must **not** be in for ElastAlert 2 to comment on the ticket instead of opening a new one. For example, to prevent comments being added to resolved or closed tickets, set this to 'Resolved' and 'Closed'. This option should not be set if the `jira_bump_in_statuses` option is set.

Example usage:

```
jira_bump_not_in_statuses:
  - Resolved
  - Closed
```

`jira_bump_in_statuses`: If `jira_bump_tickets` is true, a list of statuses the ticket *must be in* for ElastAlert 2 to comment on the ticket instead of opening a new one. For example, to only comment on 'Open' tickets – and thus not 'In Progress', 'Analyzing', 'Resolved', etc. tickets – set this to 'Open'. This option should not be set if the `jira_bump_not_in_statuses` option is set.

Example usage:

```
jira_bump_in_statuses:
  - Open
```

`jira_bump_only`: Only update if a ticket is found to bump. This skips ticket creation for rules where you only want to affect existing tickets.

Example usage:

```
jira_bump_only: true
```

`jira_transition_to`: If `jira_bump_tickets` is true, Transition this ticket to the given Status when bumping. Must match the text of your Jira implementation's Status field.

Example usage:

```
jira_transition_to: 'Fixed'
```

`jira_bump_after_inactivity`: If this is set, ElastAlert 2 will only comment on tickets that have been inactive for at least this many days. It only applies if `jira_bump_tickets` is true. Default is 0 days.

Arbitrary Jira fields:

ElastAlert 2 supports setting any arbitrary Jira field that your Jira issue supports. For example, if you had a custom field, called "Affected User", you can set it by providing that field name in `snake_case` prefixed with `jira_`. These fields can contain primitive strings or arrays of strings. Note that when you create a custom field in your Jira server, internally, the field is represented as `customfield_1111`. In ElastAlert 2, you may refer to either the public facing name OR the internal representation.

In addition, if you would like to use a field in the alert as the value for a custom Jira field, use the field name plus a # symbol in front. For example, if you wanted to set a custom Jira field called "user" to the value of the field "username" from the match, you would use the following.

Example:

```
jira_user: "#username"
```

Example usage:

```
jira_arbitrary_singular_field: My Name
jira_arbitrary_multivalue_field:
  - Name 1
  - Name 2
jira_customfield_12345: My Custom Value
jira_customfield_9999:
  - My Custom Value 1
  - My Custom Value 2
```

3.5.21 Line Notify

Line Notify will send notification to a Line application. The body of the notification is formatted the same as with other alerters.

Required:

`linenotify_access_token`: The access token that you got from https://notify-bot.line.me/my/

Example usage:

```
alert:
  - "linenotify"
linenotify_access_token: "Your linenotify access token"
```

3.5.22 Mattermost

Mattermost alerter will send a notification to a predefined Mattermost channel. The body of the notification is formatted the same as with other alerters.

The alerter requires the following option:

`mattermost_webhook_url`: The webhook URL. Follow the instructions on https://docs.mattermost.com/developer/webhooks-incoming.html to create an incoming webhook on your Mattermost installation.

Optional:

`mattermost_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to Mattermost. Set this option using `hostname:port` if you need to use a proxy. only supports https.

`mattermost_ignore_ssl_errors`: By default ElastAlert 2 will verify SSL certificate. Set this option to `True` if you want to ignore SSL errors.

`mattermost_username_override`: By default Mattermost will use your username when posting to the channel. Use this option to change it (free text).

`mattermost_channel_override`: Incoming webhooks have a default channel, but it can be overridden. A public channel can be specified "#other-channel", and a Direct Message with "@username".

`mattermost_icon_url_override`: By default ElastAlert 2 will use the default webhook icon when posting to the channel. You can provide icon_url to use custom image. Provide absolute address of the picture or Base64 data url.

`mattermost_msg_pretext`: You can set the message attachment pretext using this option.

`mattermost_msg_color`: By default the alert will be posted with the 'danger' color. You can also use 'good', 'warning', or hex color code.

`mattermost_msg_fields`: You can add fields to your Mattermost alerts using this option. You can specify the title using *title* and the text value using *value*. Additionally you can specify whether this field should be a *short* field using *short: true*. If you set *args* and *value* is a formattable string, ElastAlert 2 will format the incident key based on the provided array of fields from the rule or match. See https://docs.mattermost.com/developer/message-attachments.html#fields for more information.

Example mattermost_msg_fields:

```
mattermost_msg_fields:
  - title: Stack
    value: "{0} {1}" # interpolate fields mentioned in args
    short: false
    args: ["type", "msg.status_code"] # fields from doc
```

(continues on next page)

(continued from previous page)

```
  - title: Name
    value: static field
    short: false
```

`mattermost_title`: Sets a title for the message, this shows up as a blue text at the start of the message. Defaults to "".

`mattermost_title_link`: You can add a link in your Mattermost notification by setting this to a valid URL. Requires mattermost_title to be set. Defaults to "".

`mattermost_footer`: Add a static footer text for alert. Defaults to "".

`mattermost_footer_icon`: A Public Url for a footer icon. Defaults to "".

`mattermost_image_url`: An optional URL to an image file (GIF, JPEG, PNG, BMP, or SVG). Defaults to "".

`mattermost_thumb_url`: An optional URL to an image file (GIF, JPEG, PNG, BMP, or SVG) that is displayed as thumbnail. Defaults to "".

`mattermost_author_name`: An optional name used to identify the author. . Defaults to "".

`mattermost_author_link`: An optional URL used to hyperlink the author_name. Defaults to "".

`mattermost_author_icon`: An optional URL used to display a 16x16 pixel icon beside the author_name. Defaults to "".

`mattermost_attach_kibana_discover_url`: Enables the attachment of the `kibana_discover_url` to the mattermost notification. The config `generate_kibana_discover_url` must also be `True` in order to generate the url. Defaults to `False`.

`mattermost_kibana_discover_color`: The color of the Kibana Discover url attachment. Defaults to `#ec4b98`.

`mattermost_kibana_discover_title`: The title of the Kibana Discover url attachment. Defaults to `Discover in Kibana`.

Example mattermost_attach_kibana_discover_url, mattermost_kibana_discover_color, mattermost_kibana_discover_title:

```
# (Required)
generate_kibana_discover_url: True
kibana_discover_app_url: "http://localhost:5601/app/discover#/"
kibana_discover_index_pattern_id: "4babf380-c3b1-11eb-b616-1b59c2feec54"
kibana_discover_version: "7.15"

# (Optional)
kibana_discover_from_timedelta:
  minutes: 10
kibana_discover_to_timedelta:
  minutes: 10

# (Required)
mattermost_attach_kibana_discover_url: True

# (Optional)
mattermost_kibana_discover_color: "#ec4b98"
mattermost_kibana_discover_title: "Discover in Kibana"
```

3.5.23 Microsoft Teams

Microsoft Teams alerter will send a notification to a predefined Microsoft Teams channel.

The alerter requires the following options:

`ms_teams_webhook_url`: The webhook URL that includes your auth data and the ID of the channel you want to post to. Go to the Connectors menu in your channel and configure an Incoming Webhook, then copy the resulting URL. You can use a list of URLs to send to multiple channels.

Optional:

`ms_teams_alert_summary`: MS Teams use this value for notification title, defaults to Alert Subject. You can set this value with arbitrary text if you don't want to use the default.

`ms_teams_theme_color`: By default the alert will be posted without any color line. To add color, set this attribute to a HTML color value e.g. `#ff0000` for red.

`ms_teams_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to MS Teams. Set this option using `hostname:port` if you need to use a proxy. only supports https.

`ms_teams_alert_fixed_width`: By default this is `False` and the notification will be sent to MS Teams as-is. Teams supports a partial Markdown implementation, which means asterisk, underscore and other characters may be interpreted as Markdown. Currenlty, Teams does not fully implement code blocks. Setting this attribute to `True` will enable line by line code blocks. It is recommended to enable this to get clearer notifications in Teams.

`ms_teams_alert_facts`: You can add additional facts to your MS Teams alerts using this field. Specify the title using *name* and a value for the field or arbitrary text using *value*.

Example ms_teams_alert_facts:

```
ms_teams_alert_facts:
  - name: Host
    value: monitor.host
  - name: Status
    value: monitor.status
  - name: What to do
    value: Page your boss
```

`ms_teams_attach_kibana_discover_url`: Enables the attachment of the `kibana_discover_url` to the MS Teams notification. The config `generate_kibana_discover_url` must also be `True` in order to generate the url. Defaults to `False`.

`ms_teams_kibana_discover_title`: The title of the Kibana Discover url attachment. Defaults to `Discover in Kibana`.

Example ms_teams_attach_kibana_discover_url, ms_teams_kibana_discover_title:

```
# (Required)
generate_kibana_discover_url: True
kibana_discover_app_url: "http://localhost:5601/app/discover#/"
kibana_discover_index_pattern_id: "4babf380-c3b1-11eb-b616-1b59c2feec54"
kibana_discover_version: "7.15"

# (Optional)
kibana_discover_from_timedelta:
  minutes: 10
kibana_discover_to_timedelta:
  minutes: 10
```

(continues on next page)

(continued from previous page)

```
# (Required)
ms_teams_attach_kibana_discover_url: True

# (Optional)
ms_teams_kibana_discover_title: "Discover in Kibana"
```

`ms_teams_ca_certs`: Set this option to `True` if you want to validate the SSL certificate.

`ms_teams_ignore_ssl_errors`: By default ElastAlert 2 will verify SSL certificate. Set this option to `True` if you want to ignore SSL errors.

Example usage:

```
alert:
  - "ms_teams"
ms_teams_theme_color: "#6600ff"
ms_teams_webhook_url: "MS Teams Webhook URL"
```

3.5.24 OpsGenie

OpsGenie alerter will create an alert which can be used to notify Operations people of issues or log information. An OpsGenie `API` integration must be created in order to acquire the necessary `opsgenie_key` rule variable. Currently the OpsGenieAlerter only creates an alert, however it could be extended to update or close existing alerts.

It is necessary for the user to create an OpsGenie Rest HTTPS API integration page in order to create alerts.

The OpsGenie alert requires one option:

`opsgenie_key`: The randomly generated API Integration key created by OpsGenie.

Optional:

`opsgenie_account`: The OpsGenie account to integrate with.

`opsgenie_addr`: The OpsGenie URL to to connect against, default is `https://api.opsgenie.com/v2/alerts`. If using the EU instance of Opsgenie, the URL needs to be `https://api.eu.opsgenie.com/v2/alerts` for requests to be successful.

`opsgenie_recipients`: A list OpsGenie recipients who will be notified by the alert.

`opsgenie_recipients_args`: Map of arguments used to format opsgenie_recipients.

`opsgenie_default_receipients`: List of default recipients to notify when the formatting of opsgenie_recipients is unsuccesful.

`opsgenie_teams`: A list of OpsGenie teams to notify (useful for schedules with escalation).

`opsgenie_teams_args`: Map of arguments used to format opsgenie_teams (useful for assigning the alerts to teams based on some data).

`opsgenie_default_teams`: List of default teams to notify when the formatting of opsgenie_teams is unsuccesful.

`opsgenie_tags`: A list of tags for this alert.

`opsgenie_message`: Set the OpsGenie message to something other than the rule name. The message can be formatted with fields from the first match e.g. “Error occurred for {app_name} at {timestamp}.”.

`opsgenie_description`: Set the OpsGenie description to something other than the rule body. The message can be formatted with fields from the first match e.g. “Error occurred for {app_name} at {timestamp}.”.

opsgenie_alias: Set the OpsGenie alias. The alias can be formatted with fields from the first match e.g "{app_name} error".

opsgenie_subject: A string used to create the title of the OpsGenie alert. Can use Python string formatting.

opsgenie_subject_args: A list of fields to use to format opsgenie_subject if it contains formaters.

opsgenie_priority: Set the OpsGenie priority level. Possible values are P1, P2, P3, P4, P5. Can be formatted with fields from the first match e.g "P{level}"

opsgenie_details: Map of custom key/value pairs to include in the alert's details. The value can sourced from either fields in the first match, environment variables, or a constant value.

opsgenie_proxy: By default ElastAlert 2 will not use a network proxy to send notifications to OpsGenie. Set this option using hostname:port if you need to use a proxy. only supports https.

opsgenie_source: Set the OpsGenie source, default is *ElastAlert*. Can be formatted with fields from the first match e.g "{source} {region}"

opsgenie_entity: Set the OpsGenie entity. Can be formatted with fields from the first match e.g "{host_name}"

Example usage:

```
opsgenie_details:
  Author: 'Bob Smith'          # constant value
  Environment: '$VAR'          # environment variable
  Message: { field: message }  # field in the first match
```

Example opsgenie_details with kibana_discover_url:

```
# (Required)
generate_kibana_discover_url: True
kibana_discover_app_url: "http://localhost:5601/app/discover#/"
kibana_discover_index_pattern_id: "4babf380-c3b1-11eb-b616-1b59c2feec54"
kibana_discover_version: "7.15"

# (Optional)
kibana_discover_from_timedelta:
  minutes: 10
kibana_discover_to_timedelta:
  minutes: 10

# (Required)
opsgenie_details:
  Kibana Url: { field: kibana_discover_url }
  Message: { field: message }
  Testing: 'yes'
```

3.5.25 PagerDuty

PagerDuty alerter will trigger an incident to a predefined PagerDuty service. The body of the notification is formatted the same as with other alerters.

The alerter requires the following option:

`pagerduty_service_key`: Integration Key generated after creating a service with the 'Use our API directly' option at Integration Settings

`pagerduty_client_name`: The name of the monitoring client that is triggering this event.

`pagerduty_event_type`: Any of the following: *trigger*, *resolve*, or *acknowledge*. (Optional, defaults to *trigger*)

Optional:

`alert_subject`: If set, this will be used as the Incident description within PagerDuty. If not set, ElastAlert 2 will default to using the rule name of the alert for the incident.

`alert_subject_args`: If set, and `alert_subject` is a formattable string, ElastAlert 2 will format the incident key based on the provided array of fields from the rule or match.

`pagerduty_incident_key`: If not set PagerDuty will trigger a new incident for each alert sent. If set to a unique string per rule PagerDuty will identify the incident that this event should be applied. If there's no open (i.e. unresolved) incident with this key, a new one will be created. If there's already an open incident with a matching key, this event will be appended to that incident's log.

`pagerduty_incident_key_args`: If set, and `pagerduty_incident_key` is a formattable string, ElastAlert 2 will format the incident key based on the provided array of fields from the rule or match.

`pagerduty_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to PagerDuty. Set this option using `hostname:port` if you need to use a proxy. only supports https.

V2 API Options (Optional):

These options are specific to the PagerDuty V2 API

See https://developer.pagerduty.com/api-reference/b3A6Mjc0ODI2Nw-send-an-event-to-pager-duty

`pagerduty_api_version`: Defaults to *v1*. Set to *v2* to enable the PagerDuty V2 Event API.

`pagerduty_v2_payload_class`: Sets the class of the payload. (the event type in PagerDuty)

`pagerduty_v2_payload_class_args`: If set, and `pagerduty_v2_payload_class` is a formattable string, ElastAlert 2 will format the class based on the provided array of fields from the rule or match.

`pagerduty_v2_payload_component`: Sets the component of the payload. (what program/interface/etc the event came from)

`pagerduty_v2_payload_component_args`: If set, and `pagerduty_v2_payload_component` is a formattable string, ElastAlert 2 will format the component based on the provided array of fields from the rule or match.

`pagerduty_v2_payload_group`: Sets the logical grouping (e.g. app-stack)

`pagerduty_v2_payload_group_args`: If set, and `pagerduty_v2_payload_group` is a formattable string, ElastAlert 2 will format the group based on the provided array of fields from the rule or match.

`pagerduty_v2_payload_severity`: Sets the severity of the page. (defaults to *critical*, valid options: *critical*, *error*, *warning*, *info*)

`pagerduty_v2_payload_source`: Sets the source of the event, preferably the hostname or fqdn.

`pagerduty_v2_payload_source_args`: If set, and `pagerduty_v2_payload_source` is a formattable string, ElastAlert 2 will format the source based on the provided array of fields from the rule or match.

pagerduty_v2_payload_custom_details: List of keys:values to use as the content of the custom_details payload. Example - ip:clientip will map the value from the clientip index of Elasticsearch to JSON key named ip.

pagerduty_v2_payload_include_all_info: If True, this will include the entire Elasticsearch document as a custom detail field called "information" in the PagerDuty alert.

3.5.26 PagerTree

PagerTree alerter will trigger an incident to a predefined PagerTree integration url.

The alerter requires the following options:

pagertree_integration_url: URL generated by PagerTree for the integration.

pagertree_proxy: By default ElastAlert 2 will not use a network proxy to send notifications to PagerTree. Set this option using hostname:port if you need to use a proxy. only supports https.

Example usage:

```
alert:
  - "pagertree"
pagertree_integration_url: "PagerTree Integration URL"
```

3.5.27 Rocket.Chat

Rocket.Chat alerter will send a notification to a predefined channel. The body of the notification is formatted the same as with other alerters. https://developer.rocket.chat/api/rest-api/methods/chat/postmessage

The alerter requires the following option:

rocket_chat_webhook_url: The webhook URL that includes your auth data and the ID of the channel (room) you want to post to. You can use a list of URLs to send to multiple channels.

Optional:

rocket_chat_username_override: By default Rocket.Chat will use username defined in Integration when posting to the channel. Use this option to change it (free text).

rocket_chat_channel_override: Incoming webhooks have a default channel, but it can be overridden. A public channel can be specified "#other-channel", and a Direct Message with "@username".

rocket_chat_emoji_override: By default ElastAlert 2 will use the :ghost: emoji when posting to the channel. You can use a different emoji per ElastAlert 2 rule. Any Apple emoji can be used, see http://emojipedia.org/apple/ .

rocket_chat_msg_color: By default the alert will be posted with the 'danger' color. You can also use 'good' or 'warning' colors.

rocket_chat_text_string: Notification message you want to add.

rocket_chat_proxy: By default ElastAlert 2 will not use a network proxy to send notifications to Rocket.Chat. Set this option using hostname:port if you need to use a proxy. only supports https.

rocket_chat_ca_certs: Set this option to True if you want to validate the SSL certificate.

rocket_chat_ignore_ssl_errors: By default ElastAlert 2 will verify SSL certificate. Set this option to True if you want to ignore SSL errors.

rocket_chat_timeout: You can specify a timeout value, in seconds, for making communicating with Rocket.Chat. The default is 10. If a timeout occurs, the alert will be retried next time ElastAlert 2 cycles.

`rocket_chat_attach_kibana_discover_url`: Enables the attachment of the `kibana_discover_url` to the Rocket.Chat notification. The config `generate_kibana_discover_url` must also be `True` in order to generate the url. Defaults to `False`.

`rocket_chat_kibana_discover_color`: The color of the Kibana Discover url attachment. Defaults to `#ec4b98`.

`rocket_chat_kibana_discover_title`: The title of the Kibana Discover url attachment. Defaults to `Discover in Kibana`.

Example rocket_chat_attach_kibana_discover_url, rocket_chat_kibana_discover_color, rocket_chat_kibana_discover_title:

```
# (Required)
generate_kibana_discover_url: True
kibana_discover_app_url: "http://localhost:5601/app/discover#/"
kibana_discover_index_pattern_id: "4babf380-c3b1-11eb-b616-1b59c2feec54"
kibana_discover_version: "7.15"

# (Optional)
kibana_discover_from_timedelta:
  minutes: 10
kibana_discover_to_timedelta:
  minutes: 10

# (Required)
rocket_chat_attach_kibana_discover_url: True

# (Optional)
rocket_chat_kibana_discover_color: "#ec4b98"
rocket_chat_kibana_discover_title: "Discover in Kibana"
```

`rocket_chat_alert_fields`: You can add additional fields to your Rocket.Chat alerts using this field. Specify the title using *title* and a value for the field using *value*. Additionally you can specify whether or not this field should be a *short* field using *short: true*.

Example rocket_chat_alert_fields:

```
rocket_chat_alert_fields:
  - title: Host
    value: monitor.host
    short: true
  - title: Status
    value: monitor.status
    short: true
  - title: Zone
    value: beat.name
    short: true
```

3.5.28 Squadcast

Alerts can be sent to Squadcast using the *http post* method described above and Squadcast will process it and send Phone, SMS, Email and Push notifications to the relevant person(s) and let them take actions.

Configuration variables in rules YAML file:

```
alert: post
http_post_url: <ElastAlert 2 Webhook URL copied from Squadcast dashboard>
http_post_static_payload:
  Title: <Incident Title>
http_post_all_values: true
```

For more details, you can refer the Squadcast documentation.

3.5.29 ServiceNow

The ServiceNow alerter will create a ne Incident in ServiceNow. The body of the notification is formatted the same as with other alerters.

The alerter requires the following options:

`servicenow_rest_url`: The ServiceNow RestApi url, this will look like TableAPI.

`username`: The ServiceNow Username to access the api.

`password`: The ServiceNow password to access the api.

`short_description`: The ServiceNow password to access the api.

`comments`: Comments to be attached to the incident, this is the equivilant of work notes.

`assignment_group`: The group to assign the incident to.

`category`: The category to attach the incident to, use an existing category.

`subcategory`: The subcategory to attach the incident to, use an existing subcategory.

`cmdb_ci`: The configuration item to attach the incident to.

`caller_id`: The caller id (email address) of the user that created the incident (elastalert@somewhere.com).

Optional:

`servicenow_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to ServiceNow. Set this option using `hostname:port` if you need to use a proxy. only supports https.

`servicenow_impact`: An integer 1, 2, or 3 representing high, medium, and low respectively. This measures the effect of an incident on business processes.

`servicenow_urgency`: An integer 1, 2, or 3 representing high, medium, and low respecitvely. This measures how long this incident can be delayed until there is a significant business impact.

Example usage:

```
alert:
  - "servicenow"
servicenow_rest_url: "servicenow rest url"
username: "user"
password: "password"
short_description: "xxxxxx"
```

(continues on next page)

(continued from previous page)

```
comments: "xxxxxx"
assignment_group: "xxxxxx"
category: "xxxxxx"
subcategory: "xxxxxx"
cmdb_ci: "xxxxxx"
caller_id: "xxxxxx"
servicenow_impact: 1
servicenow_urgenc: 3
```

3.5.30 Slack

Slack alerter will send a notification to a predefined Slack channel. The body of the notification is formatted the same as with other alerters.

The alerter requires the following option:

`slack_webhook_url`: The webhook URL that includes your auth data and the ID of the channel (room) you want to post to. Go to the Incoming Webhooks section in your Slack account https://XXXXX.slack.com/services/new/incoming-webhook , choose the channel, click 'Add Incoming Webhooks Integration' and copy the resulting URL. You can use a list of URLs to send to multiple channels.

Optional:

`slack_username_override`: By default Slack will use your username when posting to the channel. Use this option to change it (free text).

`slack_channel_override`: Incoming webhooks have a default channel, but it can be overridden. A public channel can be specified "#other-channel", and a Direct Message with "@username".

`slack_emoji_override`: By default ElastAlert 2 will use the `:ghost:` emoji when posting to the channel. You can use a different emoji per ElastAlert 2 rule. Any Apple emoji can be used, see http://emojipedia.org/apple/ . If slack_icon_url_override parameter is provided, emoji is ignored.

`slack_icon_url_override`: By default ElastAlert 2 will use the `:ghost:` emoji when posting to the channel. You can provide icon_url to use custom image. Provide absolute address of the pciture.

`slack_msg_color`: By default the alert will be posted with the 'danger' color. You can also use 'good' or 'warning' colors.

`slack_parse_override`: By default the notification message is escaped 'none'. You can also use 'full'.

`slack_text_string`: Notification message you want to add.

`slack_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to Slack. Set this option using `hostname:port` if you need to use a proxy. only supports https.

`slack_alert_fields`: You can add additional fields to your slack alerts using this field. Specify the title using *title* and a value for the field using *value*. Additionally you can specify whether or not this field should be a *short* field using *short: true*.

Example slack_alert_fields:

```
slack_alert_fields:
  - title: Host
    value: monitor.host
    short: true
  - title: Status
```

(continues on next page)

(continued from previous page)

```
    value: monitor.status
    short: true
  - title: Zone
    value: beat.name
    short: true
```

slack_ignore_ssl_errors: By default ElastAlert 2 will verify SSL certificate. Set this option to True if you want to ignore SSL errors.

slack_title: Sets a title for the message, this shows up as a blue text at the start of the message

slack_title_link: You can add a link in your Slack notification by setting this to a valid URL. Requires slack_title to be set.

slack_timeout: You can specify a timeout value, in seconds, for making communicating with Slack. The default is 10. If a timeout occurs, the alert will be retried next time ElastAlert 2 cycles.

slack_attach_kibana_discover_url: Enables the attachment of the kibana_discover_url to the slack notification. The config generate_kibana_discover_url must also be True in order to generate the url. Defaults to False.

slack_kibana_discover_color: The color of the Kibana Discover url attachment. Defaults to #ec4b98.

slack_kibana_discover_title: The title of the Kibana Discover url attachment. Defaults to Discover in Kibana.

Example slack_attach_kibana_discover_url, slack_kibana_discover_color, slack_kibana_discover_title:

```
# (Required)
generate_kibana_discover_url: True
kibana_discover_app_url: "http://localhost:5601/app/discover#/"
kibana_discover_index_pattern_id: "4babf380-c3b1-11eb-b616-1b59c2feec54"
kibana_discover_version: "7.15"

# (Optional)
kibana_discover_from_timedelta:
  minutes: 10
kibana_discover_to_timedelta:
  minutes: 10

# (Required)
slack_attach_kibana_discover_url: True

# (Optional)
slack_kibana_discover_color: "#ec4b98"
slack_kibana_discover_title: "Discover in Kibana"
```

slack_ca_certs: Set this option to True if you want to validate the SSL certificate.

slack_footer: Add a static footer text for alert. Defaults to “”.

slack_footer_icon: A Public Url for a footer icon. Defaults to “”.

slack_image_url: An optional URL to an image file (GIF, JPEG, PNG, BMP, or SVG). Defaults to “”.

slack_thumb_url: An optional URL to an image file (GIF, JPEG, PNG, BMP, or SVG) that is displayed as thumbnail. Defaults to “”.

slack_author_name: An optional name used to identify the author. Defaults to “”.

`slack_author_link`: An optional URL used to hyperlink the author_name. Defaults to "".

`slack_author_icon`: An optional URL used to display a 16x16 pixel icon beside the author_name. Defaults to "".

`slack_msg_pretext`: You can set the message attachment pretext using this option. Defaults to "".

`slack_attach_jira_ticket_url`: Add url to the jira ticket created. Only works if the Jira alert runs before Slack alert. Set the field to `True` in order to generate the url. Defaults to `False`.

`slack_jira_ticket_color`: The color of the Jira Ticket url attachment. Defaults to `#ec4b98`.

`slack_jira_ticket_title`: The title of the Jira Ticket url attachment. Defaults to `Jira Ticket`.

3.5.31 Splunk On-Call (Formerly VictorOps)

Splunk On-Call (Formerly VictorOps) alerter will trigger an incident to a predefined Splunk On-Call (Formerly VictorOps) routing key. The body of the notification is formatted the same as with other alerters.

The alerter requires the following options:

`victorops_api_key`: API key generated under the 'REST Endpoint' in the Integrations settings.

`victorops_routing_key`: Splunk On-Call (Formerly VictorOps) routing key to route the alert to.

`victorops_message_type`: Splunk On-Call (Formerly VictorOps) field to specify severity level. Must be one of the following: INFO, WARNING, ACKNOWLEDGEMENT, CRITICAL, RECOVERY

Optional:

`victorops_entity_id`: The identity of the incident used by Splunk On-Call (Formerly VictorOps) to correlate incidents throughout the alert lifecycle. If not defined, Splunk On-Call (Formerly VictorOps) will assign a random string to each alert.

`victorops_entity_display_name`: Human-readable name of alerting entity to summarize incidents without affecting the life-cycle workflow. Will use `alert_subject` if not set.

`victorops_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to Splunk On-Call (Formerly VictorOps). Set this option using `hostname:port` if you need to use a proxy. only supports https.

Example usage:

```
alert:
  - "victorops"
victorops_api_key: "VictorOps API Key"
victorops_routing_key: "VictorOps routing Key"
victorops_message_type: "INFO"
```

3.5.32 Stomp

This alert type will use the STOMP protocol in order to push a message to a broker like ActiveMQ or RabbitMQ. The message body is a JSON string containing the alert details. The default values will work with a pristine ActiveMQ installation.

The alerter requires the following options:

`stomp_hostname`: The STOMP host to use, defaults to `localhost`.

`stomp_hostport`: The STOMP port to use, defaults to `61613`.

`stomp_login`: The STOMP login to use, defaults to `admin`.

`stomp_password`: The STOMP password to use, defaults to `admin`.

Optional:

`stomp_destination`: The STOMP destination to use, defaults to `/queue/ALERT`

The stomp_destination field depends on the broker, the /queue/ALERT example is the nomenclature used by ActiveMQ. Each broker has its own logic.

Example usage:

```
alert:
  - "stomp"
stomp_hostname: "localhost"
stomp_hostport: "61613"
stomp_login: "admin"
stomp_password: "admin"
stomp_destination: "/queue/ALERT"
```

3.5.33 Telegram

Telegram alerter will send a notification to a predefined Telegram username or channel. The body of the notification is formatted the same as with other alerters.

The alerter requires the following two options:

`telegram_bot_token`: The token is a string along the lines of `110201543:AAHdqTcvCH1vGWJxfSeofSAs0K5PALDsaw` that will be required to authorize the bot and send requests to the Bot API. You can learn about obtaining tokens and generating new ones in this document https://core.telegram.org/bots#6-botfather

`telegram_room_id`: Unique identifier for the target chat or username of the target channel using telegram chat_id (in the format "-xxxxxxxx")

Optional:

`telegram_api_url`: Custom domain to call Telegram Bot API. Default to api.telegram.org

`telegram_proxy`: By default ElastAlert 2 will not use a network proxy to send notifications to Telegram. Set this option using `hostname:port` if you need to use a proxy. only supports https.

`telegram_proxy_login`: The Telegram proxy auth username.

`telegram_proxy_pass`: The Telegram proxy auth password.

`telegram_parse_mode`: The Telegram parsing mode, which determines the format of the alert text body. Possible values are `markdown`, `markdownV2`, `html`. Defaults to `markdown`.

Example usage:

```
alert:
  - "telegram"
telegram_bot_token: "bot_token"
telegram_room_id: "chat_id"
```

3.5.34 Tencent SMS

Required:

`tencent_sms_secret_id`: `SecretID` is used to identify the API caller.

`tencent_sms_secret_key`: `SecretKey` is used to encrypt the string to sign that can be verified on the server. You should keep it private and avoid disclosure.

`tencent_sms_sdk_appid`: SMS application ID, which is the *SdkAppId* generated after an application is added in the SMS console, such as 1400006666

`tencent_sms_to_number`: Target mobile number in the E.164 standard (+[country/region code][mobile number])

Example: +8613711112222, which has a + sign followed by 86 (country/region code) and then by 13711112222 (mobile number). Up to 200 mobile numbers are supported

`tencent_sms_template_id`: Template ID. You must enter the ID of an approved template, which can be viewed in the SMS console.

If you need to send SMS messages to global mobile numbers, you can only use a Global SMS template.

Optional:

`tencent_sms_sign_name`: Content of the SMS signature, which should be encoded in UTF-8. You must enter an approved signature, such as Tencent Cloud. The signature information can be viewed in the SMS console. Note: this parameter is required for Mainland China SMS.

`tencent_sms_region`: Region parameter, which is used to identify the region(Mainland China or Global) to which the data you want to work with belongs.

`tencent_sms_template_parm`: The number of template parameters needs to be consistent with the number of variables of the template corresponding to TemplateId. this value format by rfc6901


```
{
  "_index" : "tmec"
  "_type" : "fluentd",
  "_id" : "PeXLrnsBvusb3d0w6dUl",
  "_score" : 1.0,
  "_source" : {
    "kubernetes" : {
      "host" : "9.134.191.187",
      "pod_id" : "66ba4e5a-1ad2-4655-9a8e-cffb6b942559",
      "labels" : {
        "release" : "nginx",
        "pod-template-hash" : "6bd96d6f74"
      },
      "namespace_name" : "app",
      "pod_name" : "app.nginx-6bd96d6f74-2ts4x"
    },
    "time" : "2021-09-04T03:13:24.192875Z",
    "message" : "2021-09-03T14:34:08+0000|INFO|vector eps : 192.168.0.2:10000,",
  }
}
```


```
tencent_sms_template_id: "1123835"
tencent_sms_template_parm:
  - "/kubernetes/pod_name"
```

3.5.35 TheHive

TheHive alerter can be used to create a new alert in TheHive. The alerter supports adding tags, custom fields, and observables from the alert matches and rule data.

Required:

`hive_connection`: The connection details to your instance (see example below for the required syntax). Only `hive_apikey` is required, `hive_host` and `hive_port` default to `http://localhost` and `9000` respectively.

`hive_alert_config`: Configuration options for the alert, see example below for structure.

If not supplied, the alert title and description will be populated from the ElastAlert 2 default `title` and `alert_text` fields, including any defined `alert_text_args`.

Optional:

`tags` can be populated from the matched record, using the same syntax used in `alert_text_args`. If a record doesn't contain the specified value, the rule itself will be examined for the tag. If this doesn't contain the tag either, the tag is attached without modification to the alert. For aggregated alerts, all matches are examined individually, and tags generated for each one. All tags are then attached to the same alert.

`customFields` can also be populated from rule fields as well as matched results. Custom fields are only populated once. If an alert is an aggregated alert, the custom field values will be populated using the first matched record, before checking the rule. If neither matches, the `customField.value` will be used directly.

`hive_observable_data_mapping`: If needed, matched data fields can be mapped to TheHive observable types using the same syntax as `customFields`, described above. The algorithm used to populate the observable value is similar to the one used to populate the `tags`, including the behaviour for aggregated alerts. The tlp, message, and tags fields are optional for each observable. If not specified, the tlp field is given a default value of 2.

`hive_proxies`: Proxy configuration.

`hive_verify`: Whether or not to enable SSL certificate validation. Defaults to False.

`description_args`: can be used to call rule and match fileds in the description of the alert in TheHive `description_missing_value`: Text to replace any match field not found when formating the `description`. Defaults to `<MISSING VALUE>`.

Example usage:

```
alert: hivealerter

hive_connection:
  hive_host: http://localhost
  hive_port: <hive_port>
  hive_apikey: <hive_apikey>
  hive_proxies:
    http: ''
    https: ''

hive_alert_config:
  customFields:
    - name: example
      type: string
      value: example
  follow: True
  severity: 2
  status: 'New'
```

(continues on next page)

(continued from previous page)

```
  source: 'elastalert'
  description_args: [ name, description]
  description: '{0} : {1}'
  tags: ['tag1', 'tag2']
  title: 'Title'
  tlp: 3
  type: 'external'

hive_observable_data_mapping:
  - domain: agent.hostname
    tlp: 1
    tags: ['tag1', 'tag2']
    message: 'agent hostname'
  - domain: response.domain
    tlp: 2
    tags: ['tag3']
  - ip: client.ip
```

3.5.36 Twilio

The Twilio alerter will send an alert to a mobile phone as an SMS from your Twilio phone number. The SMS will contain the alert name. You may use either Twilio SMS or Twilio Copilot to send the message, controlled by the `twilio_use_copilot` option.

Note that when Twilio Copilot *is* used the `twilio_message_service_sid` option is required. Likewise, when *not* using Twilio Copilot, the `twilio_from_number` option is required.

The alerter requires the following options:

`twilio_account_sid`: The SID of your Twilio account.

`twilio_auth_token`: Auth token associated with your Twilio account.

`twilio_to_number`: The phone number where you would like to send the alert.

Either one of

- `twilio_from_number`: The Twilio phone number from which the alert will be sent.
- `twilio_message_service_sid`: The SID of your Twilio message service.

Optional:

`twilio_use_copilot`: Whether or not to use Twilio Copilot, False by default.

Example with Copilot usage:

```
alert:
  - "twilio"
twilio_use_copilot: True
twilio_to_number: "0123456789"
twilio_auth_token: "abcdefghijklmnopqrstuvwxyz012345"
twilio_account_sid: "ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567"
twilio_message_service_sid: "ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567"
```

Example with SMS usage:

```
alert:
  - "twilio"
twilio_to_number: "0123456789"
twilio_from_number: "9876543210"
twilio_auth_token: "abcdefghijklmnopqrstuvwxyz012345"
twilio_account_sid: "ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567"
```

3.5.37 Zabbix

Zabbix will send notification to a Zabbix server. The item in the host specified receive a 1 value for each hit. For example, if the elastic query produce 3 hits in the last execution of ElastAlert 2, three '1' (integer) values will be send from elastalert to Zabbix Server. If the query have 0 hits, any value will be sent.

Required:

`zbx_sender_host`: The address where zabbix server is running, defaults to `'localhost'`.

`zbx_sender_port`: The port where zabbix server is listenning, defaults to `10051`.

`zbx_host_from_field`: This field allows to specify `zbx_host` value from the available terms. Defaults to `False`.

`zbx_host`: This field setup the host in zabbix that receives the value sent by ElastAlert 2.

`zbx_key`: This field setup the key in the host that receives the value sent by ElastAlert 2.

Example usage:

```
alert:
  - "zabbix"
zbx_sender_host: "zabbix-server"
zbx_sender_port: 10051
zbx_host: "test001"
zbx_key: "sender_load1"
```

To specify `zbx_host` depending on the available elasticsearch field, zabbix alerter has `zbx_host_from_field` option.

Example usage:

```
alert:
  - "zabbix"
zbx_sender_host: "zabbix-server"
zbx_sender_port: 10051
zbx_host_from_field: True
zbx_host: "hostname"
zbx_key: "sender_load1"
```

where `hostname` is the available elasticsearch field.

CHAPTER

FOUR

ELASTALERT 2 METADATA INDEX

ElastAlert 2 uses Elasticsearch to store various information about its state. This not only allows for some level of auditing and debugging of ElastAlert 2's operation, but also to avoid loss of data or duplication of alerts when ElastAlert 2 is shut down, restarted, or crashes. This cluster and index information is defined in the global config file with `es_host`, `es_port` and `writeback_index`. ElastAlert 2 must be able to write to this index. The script, `elastalert-create-index` will create the index with the correct mapping for you, and optionally copy the documents from an existing ElastAlert 2 writeback index. Run it and it will prompt you for the cluster information.

ElastAlert 2 will create three different types of documents in the writeback index:

4.1 elastalert_status

`elastalert_status` is a log of the queries performed for a given rule and contains:

- `@timestamp`: The time when the document was uploaded to Elasticsearch. This is after a query has been run and the results have been processed.
- `rule_name`: The name of the corresponding rule.
- `starttime`: The beginning of the timestamp range the query searched.
- `endtime`: The end of the timestamp range the query searched.
- `hits`: The number of results from the query.
- `matches`: The number of matches that the rule returned after processing the hits. Note that this does not necessarily mean that alerts were triggered.
- `time_taken`: The number of seconds it took for this query to run.

`elastalert_status` is what ElastAlert 2 will use to determine what time range to query when it first starts to avoid duplicating queries. For each rule, it will start querying from the most recent endtime. If ElastAlert 2 is running in debug mode, it will still attempt to base its start time by looking for the most recent search performed, but it will not write the results of any query back to Elasticsearch.

4.2 elastalert

`elastalert` is a log of information about every alert triggered and contains:

- `@timestamp`: The time when the document was uploaded to Elasticsearch. This is not the same as when the alert was sent, but rather when the rule outputs a match.
- `rule_name`: The name of the corresponding rule.
- `alert_info`: This contains the output of Alert.get_info, a function that alerts implement to give some relevant context to the alert type. This may contain alert_info.type, alert_info.recipient, or any number of other sub fields.
- `alert_sent`: A boolean value as to whether this alert was actually sent or not. It may be false in the case of an exception or if it is part of an aggregated alert.
- `alert_time`: The time that the alert was or will be sent. Usually, this is the same as @timestamp, but may be some time in the future, indicating when an aggregated alert will be sent.
- `match_body`: This is the contents of the match dictionary that is used to create the alert. The subfields may include a number of things containing information about the alert.
- `alert_exception`: This field is only present when the alert failed because of an exception occurring, and will contain the exception information.
- `aggregate_id`: This field is only present when the rule is configured to use aggregation. The first alert of the aggregation period will contain an alert_time set to the aggregation time into the future, and subsequent alerts will contain the document ID of the first. When the alert_time is reached, all alerts with that aggregate_id will be sent together.

4.3 elastalert_error

When an error occurs in ElastAlert 2, it is written to both Elasticsearch and to stderr. The `elastalert_error` type contains:

- `@timestamp`: The time when the error occurred.
- `message`: The error or exception message.
- `traceback`: The traceback from when the error occurred.
- `data`: Extra information about the error. This often contains the name of the rule which caused the error.

4.4 silence

`silence` is a record of when alerts for a given rule will be suppressed, either because of a `realert` setting or from using –silence. When an alert with `realert` is triggered, a `silence` record will be written with `until` set to the alert time plus `realert`.

- `@timestamp`: The time when the document was uploaded to Elasticsearch.
- `rule_name`: The name of the corresponding rule.
- `until`: The timestamp when alerts will begin being sent again.
- `exponent`: The exponential factor which multiplies `realert`. The length of this silence is equal to `realert` * 2**exponent. This will be 0 unless `exponential_realert` is set.

Whenever an alert is triggered, ElastAlert 2 will check for a matching `silence` document, and if the `until` timestamp is in the future, it will ignore the alert completely. See the *Running ElastAlert 2* section for information on how to silence an alert.

CHAPTER

FIVE

ELASTICSEARCH SECURITY PRIVILEGES

While ElastAlert 2 will just work out-of-the-box for unsecured Elasticsearch, it will need a user with a certain set of permissions to work on secure Elasticseach that allow it to read the documents, check the cluster status etc.

5.1 SearchGuard Permissions

The permissions in Elasticsearch are specific to the plugin being used for RBAC. However, the permissions mentioned here can be mapped easily to different plugins other than Searchguard.

Details about SearchGuard Action Groups: https://docs.search-guard.com/latest/action-groups

5.1.1 Writeback Permissions

For the global config (which writes to the writeback index), you would need to give all permissions on the writeback indices. In addition, some permissions related to Cluster Monitor Access are required.

`Cluster Permissions`: CLUSTER_MONITOR, indices:data/read/scroll*

`Index Permissions` (Over Writeback Indices): INDICES_ALL

5.1.2 Per Rule Permissions

For per rule Elasticsearch config, you would need at least the read permissions on the index you want to query. Detailed SearchGuard Permissions:

`Cluster Permissions`: CLUSTER_COMPOSITE_OPS_RO

`Index Permissions` (Over the index the rule is querying on): READ, indices:data/read/scroll*

CHAPTER

SIX

ADDING A NEW RULE TYPE

This document describes how to create a new rule type. Built in rule types live in `elastalert/ruletypes.py` and are subclasses of `RuleType`. At the minimum, your rule needs to implement `add_data`.

Your class may implement several functions from `RuleType`:

```
class AwesomeNewRule(RuleType):
    # ...
    def add_data(self, data):
        # ...
    def get_match_str(self, match):
        # ...
    def garbage_collect(self, timestamp):
        # ...
```

You can import new rule types by specifying the type as `module.file.RuleName`, where module is the name of a Python module, or folder containing `__init__.py`, and file is the name of the Python file containing a `RuleType` subclass named `RuleName`.

6.1 Basics

The `RuleType` instance remains in memory while ElastAlert is running, receives data, keeps track of its state, and generates matches. Several important member properties are created in the `__init__` method of `RuleType`:

`self.rules`: This dictionary is loaded from the rule configuration file. If there is a `timeframe` configuration option, this will be automatically converted to a `datetime.timedelta` object when the rules are loaded.

`self.matches`: This is where ElastAlert 2 checks for matches from the rule. Whatever information is relevant to the match (generally coming from the fields in Elasticsearch) should be put into a dictionary object and added to `self.matches`. ElastAlert 2 will pop items out periodically and send alerts based on these objects. It is recommended that you use `self.add_match(match)` to add matches. In addition to appending to `self.matches`, `self.add_match` will convert the datetime `@timestamp` back into an ISO8601 timestamp.

`self.required_options`: This is a set of options that must exist in the configuration file. ElastAlert 2 will ensure that all of these fields exist before trying to instantiate a `RuleType` instance.

6.2 add_data(self, data):

When ElastAlert 2 queries Elasticsearch, it will pass all of the hits to the rule type by calling `add_data`. `data` is a list of dictionary objects which contain all of the fields in `include`, `query_key` and `compare_key` if they exist, and `@timestamp` as a datetime object. They will always come in chronological order sorted by '@timestamp'.

6.3 get_match_str(self, match):

Alerts will call this function to get a human readable string about a match for an alert. Match will be the same object that was added to `self.matches`, and `rules` the same as `self.rules`. The `RuleType` base implementation will return an empty string. Note that by default, the alert text will already contain the key-value pairs from the match. This should return a string that gives some information about the match in the context of this specific RuleType.

6.4 garbage_collect(self, timestamp):

This will be called after ElastAlert 2 has run over a time period ending in `timestamp` and should be used to clear any state that may be obsolete as of `timestamp`. `timestamp` is a datetime object.

6.5 Tutorial

As an example, we are going to create a rule type for detecting suspicious logins. Let's imagine the data we are querying is login events that contains IP address, username and a timestamp. Our configuration will take a list of usernames and a time range and alert if a login occurs in the time range. First, let's create a modules folder in the base ElastAlert 2 folder:

```
$ mkdir elastalert_modules
$ cd elastalert_modules
$ touch __init__.py
```

Now, in a file named `my_rules.py`, add

```
import dateutil.parser

from elastalert.ruletypes import RuleType

# elastalert.util includes useful utility functions
# such as converting from timestamp to datetime obj
from elastalert.util import ts_to_dt

class AwesomeRule(RuleType):

    # By setting required_options to a set of strings
    # You can ensure that the rule config file specifies all
    # of the options. Otherwise, ElastAlert 2 will throw an exception
    # when trying to load the rule.
    required_options = set(['time_start', 'time_end', 'usernames'])

    # add_data will be called each time Elasticsearch is queried.
```

(continues on next page)

(continued from previous page)

```
    # data is a list of documents from Elasticsearch, sorted by timestamp,
    # including all the fields that the config specifies with "include"
    def add_data(self, data):
        for document in data:

            # To access config options, use self.rules
            if document['username'] in self.rules['usernames']:

                # Convert the timestamp to a time object
                login_time = document['@timestamp'].time()

                # Convert time_start and time_end to time objects
                time_start = dateutil.parser.parse(self.rules['time_start']).time()
                time_end = dateutil.parser.parse(self.rules['time_end']).time()

                # If the time falls between start and end
                if login_time > time_start and login_time < time_end:

                    # To add a match, use self.add_match
                    self.add_match(document)

    # The results of get_match_str will appear in the alert text
    def get_match_str(self, match):
        return "%s logged in between %s and %s" % (match['username'],
                                                   self.rules['time_start'],
                                                   self.rules['time_end'])

    # garbage_collect is called indicating that ElastAlert 2 has already been run up to␣
↪timestamp
    # It is useful for knowing that there were no query results from Elasticsearch␣
↪because
    # add_data will not be called with an empty list
    def garbage_collect(self, timestamp):
        pass
```

In the rule configuration file, `examples/rules/example_login_rule.yaml`, we are going to specify this rule by writing

```
name: "Example login rule"
es_host: elasticsearch.example.com
es_port: 14900
type: "elastalert_modules.my_rules.AwesomeRule"
# Alert if admin, userXYZ or foobaz log in between 8 PM and midnight
time_start: "20:00"
time_end: "24:00"
usernames:
- "admin"
- "userXYZ"
- "foobaz"
# We require the username field from documents
include:
- "username"
```

(continues on next page)

(continued from previous page)

```
alert:
- debug
```

ElastAlert 2 will attempt to import the rule with `from elastalert_modules.my_rules import AwesomeRule`. This means that the folder must be in a location where it can be imported as a Python module.

An alert from this rule will look something like:

```
Example login rule

userXYZ logged in between 20:00 and 24:00

@timestamp: 2015-03-02T22:23:24Z
username: userXYZ
```

CHAPTER

SEVEN

ADDING A NEW ALERTER

Alerters are subclasses of `Alerter`, found in `elastalert/alerts.py`. They are given matches and perform some action based on that. Your alerter needs to implement two member functions, and will look something like this:

```
class AwesomeNewAlerter(Alerter):
    required_options = set(['some_config_option'])
    def alert(self, matches):
        ...
    def get_info(self):
        ...
```

You can import alert types by specifying the type as `module.file.AlertName`, where module is the name of a python module, and file is the name of the python file containing a `Alerter` subclass named `AlertName`.

7.1 Basics

The alerter class will be instantiated when ElastAlert 2 starts, and be periodically passed matches through the `alert` method. ElastAlert 2 also writes back info about the alert into Elasticsearch that it obtains through `get_info`. Several important member properties:

`self.required_options`: This is a set containing names of configuration options that must be present. ElastAlert 2 will not instantiate the alert if any are missing.

`self.rule`: The dictionary containing the rule configuration. All options specific to the alert should be in the rule configuration file and can be accessed here.

`self.pipeline`: This is a dictionary object that serves to transfer information between alerts. When an alert is triggered, a new empty pipeline object will be created and each alerter can add or receive information from it. Note that alerters are called in the order they are defined in the rule file. For example, the Jira alerter will add its ticket number to the pipeline and the email alerter will add that link if it's present in the pipeline.

7.2 alert(self, match):

ElastAlert 2 will call this function to send an alert. `matches` is a list of dictionary objects with information about the match. You can get a nice string representation of the match by calling `self.rule['type'].get_match_str(match, self.rule)`. If this method raises an exception, it will be caught by ElastAlert 2 and the alert will be marked as unsent and saved for later.

7.3 get_info(self):

This function is called to get information about the alert to save back to Elasticsearch. It should return a dictionary, which is uploaded directly to Elasticsearch, and should contain useful information about the alert such as the type, recipients, parameters, etc.

7.4 Tutorial

Let's create a new alert that will write alerts to a local output file. First, create a modules folder in the base ElastAlert 2 folder:

```
$ mkdir elastalert_modules
$ cd elastalert_modules
$ touch __init__.py
```

Now, in a file named `my_alerts.py`, add

```
from elastalert.alerts import Alerter, BasicMatchString

class AwesomeNewAlerter(Alerter):

    # By setting required_options to a set of strings
    # You can ensure that the rule config file specifies all
    # of the options. Otherwise, ElastAlert 2 will throw an exception
    # when trying to load the rule.
    required_options = set(['output_file_path'])

    # Alert is called
    def alert(self, matches):

        # Matches is a list of match dictionaries.
        # It contains more than one match when the alert has
        # the aggregation option set
        for match in matches:

            # Config options can be accessed with self.rule
            with open(self.rule['output_file_path'], "a") as output_file:

                # basic_match_string will transform the match into the default
                # human readable string format
                match_string = str(BasicMatchString(self.rule, match))

                output_file.write(match_string)

    # get_info is called after an alert is sent to get data that is written back
    # to Elasticsearch in the field "alert_info"
    # It should return a dict of information relevant to what the alert does
    def get_info(self):
        return {'type': 'Awesome Alerter',
                'output_file': self.rule['output_file_path']}
```

In the rule configuration file, we are going to specify the alert by writing

```
alert: "elastalert_modules.my_alerts.AwesomeNewAlerter"
output_file_path: "/tmp/alerts.log"
```

ElastAlert 2 will attempt to import the alert with `from elastalert_modules.my_alerts import AwesomeNewAlerter`. This means that the folder must be in a location where it can be imported as a python module.

CHAPTER

EIGHT

WRITING FILTERS FOR RULES

This document describes how to create a filter section for your rule config file.

The filters used in rules are part of the Elasticsearch query DSL, further documentation for which can be found at https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html This document contains a small subset of particularly useful filters.

The filter section is passed to Elasticsearch exactly as follows:

```
filter:
  and:
    filters:
      - [filters from rule.yaml]
```

Every result that matches these filters will be passed to the rule for processing.

8.1 Common Filter Types:

8.1.1 query_string

The query_string type follows the Lucene query format and can be used for partial or full matches to multiple fields. See http://lucene.apache.org/core/2_9_4/queryparsersyntax.html for more information:

```
filter:
- query:
    query_string:
      query: "username: bob"
- query:
    query_string:
      query: "_type: login_logs"
- query:
    query_string:
      query: "field: value OR otherfield: othervalue"
- query:
    query_string:
       query: "this: that AND these: those"
```

8.1.2 term

The term type allows for exact field matches:

```
filter:
- term:
    name_field: "bob"
- term:
    _type: "login_logs"
```

Note that a term query may not behave as expected if a field is analyzed. By default, many string fields will be tokenized by whitespace, and a term query for "foo bar" may not match a field that appears to have the value "foo bar", unless it is not analyzed. Conversely, a term query for "foo" will match analyzed strings "foo bar" and "foo baz". For full text matching on analyzed fields, use query_string. See https://www.elastic.co/guide/en/elasticsearch/guide/current/term-vs-full-text.html

8.1.3 terms

Terms allows for easy combination of multiple term filters:

```
filter:
- terms:
    field: ["value1", "value2"] # value1 OR value2
```

You can also match on multiple fields (All terms must match at least one of the given values):

```
- terms:
    fieldX: ["value1", "value2"]
- terms:
    fieldY: ["something", "something_else"]
- terms:
    fieldZ: ["foo", "bar", "baz"]
```

8.1.4 wildcard

For wildcard matches:

```
filter:
- query:
    wildcard:
      field: "foo*bar"
```

8.1.5 range

For ranges on fields:

```
filter:
- range:
    status_code:
      from: 500
      to: 599
```

8.1.6 Negation, and, or

Below is a more complex example for Elasticsearch 7.x, provided by a community user.:

```
filter:
- term:
    action: order
- terms:
    dining:
        - pickup
        - delivery
- bool:
    #exclude common/expected orders
    must_not:
        #Alice usually gets a pizza
        - bool:
            must: [ {term: {uid: alice}}, {term: {menu_item: pizza}} ]
        #Bob loves his hoagies
        - bool:
            must: [ {term: {uid: bob}}, {term: {menu_item: sandwich}} ]
        #Charlie has a few favorites
        - bool:
            must:
               - term:
                   uid: charlie
               - match:
                   menu_item: "burrito pasta salad pizza"
```

CHAPTER

NINE

ENHANCEMENTS

Enhancements are modules which let you modify a match before an alert is sent. They should subclass `BaseEnhancement`, found in `elastalert/enhancements.py`. They can be added to rules using the `match_enhancements` option:

```
match_enhancements:
- module.file.MyEnhancement
```

where module is the name of a Python module, or folder containing `__init__.py`, and file is the name of the Python file containing a `BaseEnhancement` subclass named `MyEnhancement`.

A special exception class `DropMatchException` can be used in enhancements to drop matches if custom conditions are met. For example:

```
class MyEnhancement(BaseEnhancement):
    def process(self, match):
        # Drops a match if "field_1" == "field_2"
        if match['field_1'] == match['field_2']:
            raise DropMatchException()
```

9.1 Example

As an example enhancement, let's add a link to a whois website. The match must contain a field named domain and it will add an entry named domain_whois_link. First, create a modules folder for the enhancement in the ElastAlert 2 directory.

```
$ mkdir elastalert_modules
$ cd elastalert_modules
$ touch __init__.py
```

Now, in a file named `my_enhancements.py`, add

```
from elastalert.enhancements import BaseEnhancement

class MyEnhancement(BaseEnhancement):

    # The enhancement is run against every match
    # The match is passed to the process function where it can be modified in any way
    # ElastAlert 2 will do this for each enhancement linked to a rule
    def process(self, match):
```

(continues on next page)

(continued from previous page)

```
        if 'domain' in match:
            url = "http://who.is/whois/%s" % (match['domain'])
            match['domain_whois_link'] = url
```

Enhancements will not automatically be run. Inside the rule configuration file, you need to point it to the enhancement(s) that it should run by setting the `match_enhancements` option:

```
match_enhancements:
- "elastalert_modules.my_enhancements.MyEnhancement"
```

CHAPTER

TEN

RULES LOADERS

RulesLoaders are subclasses of `RulesLoader`, found in `elastalert/loaders.py`. They are used to gather rules for a particular source. Your RulesLoader needs to implement three member functions, and will look something like this:

```
class AwesomeNewRulesLoader(RulesLoader):
    def get_names(self, conf, use_rule=None):
        ...
    def get_hashes(self, conf, use_rule=None):
        ...
    def get_yaml(self, rule):
        ...
```

You can import loaders by specifying the type as `module.file.RulesLoaderName`, where module is the name of a python module, and file is the name of the python file containing a `RulesLoader` subclass named `RulesLoaderName`.

10.1 Example

As an example loader, let's retrieve rules from a database rather than from the local file system. First, create a modules folder for the loader in the ElastAlert 2 directory.

```
$ mkdir elastalert_modules
$ cd elastalert_modules
$ touch __init__.py
```

Now, in a file named `mongo_loader.py`, add

```
from pymongo import MongoClient
from elastalert.loaders import RulesLoader
import yaml

class MongoRulesLoader(RulesLoader):
    def __init__(self, conf):
        super(MongoRulesLoader, self).__init__(conf)
        self.client = MongoClient(conf['mongo_url'])
        self.db = self.client[conf['mongo_db']]
        self.cache = {}

    def get_names(self, conf, use_rule=None):
        if use_rule:
            return [use_rule]
```

(continues on next page)

(continued from previous page)

```
        rules = []
        self.cache = {}
        for rule in self.db.rules.find():
            self.cache[rule['name']] = yaml.load(rule['yaml'])
            rules.append(rule['name'])

        return rules

    def get_hashes(self, conf, use_rule=None):
        if use_rule:
            return [use_rule]

        hashes = {}
        self.cache = {}
        for rule in self.db.rules.find():
            self.cache[rule['name']] = rule['yaml']
            hashes[rule['name']] = rule['hash']

        return hashes

    def get_yaml(self, rule):
        if rule in self.cache:
            return self.cache[rule]

        self.cache[rule] = yaml.load(self.db.rules.find_one({'name': rule})['yaml'])
        return self.cache[rule]
```

Finally, you need to specify in your ElastAlert 2 configuration file that MongoRulesLoader should be used instead of the default FileRulesLoader, so in your `elastalert.conf` file:

```
rules_loader: "elastalert_modules.mongo_loader.MongoRulesLoader"
```

CHAPTER

ELEVEN

EXPOSING RULE METRICS

11.1 Configuration

Running ElastAlert with `--prometheus_port` configuration flag will expose ElastAlert 2 Prometheus metrics on the specified port. Prometheus metrics are disabled by default.

To expose ElastAlert rule metrics on port `9979` run the following command:

```
$ elastalert --config config.yaml --prometheus_port 9979
```

11.2 Rule Metrics

The metrics being exposed are related to the ElastAlert metadata indices. The exposed metrics are in the Prometheus text-based format. Metrics are of the metric type counter or gauge and follow the Prometheus metric naming.

In the standard metric definition, the metric names are structured as follows:

```
elastalert_{metric}_{unit}
```

Where:

- `{metric}` is a unique name of the metric. For example, `hits`.
- `{unit}` is the unit of measurement of the metric value. For example, `total` is a counter type metric and `created` is a gauge type metric.

All metrics except `elastalert_errors_{unit}` have values that apply to a particular rule name. In the exported metrics, these can be identified using the `rule_name` Prometheus label.

Find below all available metrics:

METRIC	Type	Description	Label
`elastalert_scrapes_{unit}`	Counter, Gauge	Number of scrapes	`rule_name`
`elastalert_hits_{unit}`	Counter, Gauge	Number of hits	`rule_name`
`elastalert_matches_{unit}`	Counter, Gauge	Number of matches	`rule_name`
`elastalert_time_taken_{unit}`	Counter, Gauge	Time taken in seconds	`rule_name`
`elastalert_alerts_sent_{unir}`	Counter, Gauge	Number of alerts sent	`rule_name`
`elastalert_alerts_not_sent_{unit}`	Counter, Gauge	Number of alerts not sent	`rule_name`
`elastalert_alerts_silenced_{unit}`	Counter, Gauge	Number of silenced alerts	`rule_name`
`elastalert_errors_{unit}`	Counter, Gauge	Number of errors	

CHAPTER

TWELVE

SIGNING REQUESTS TO AMAZON OPENSEARCH SERVICE

When using Amazon OpenSearch Service, you need to secure your Elasticsearch from the outside. Currently, there is no way to secure your Elasticsearch using network firewall rules, so the only way is to signing the requests using the access key and secret key for a role or user with permissions on the Elasticsearch service.

You can sign requests to AWS using any of the standard AWS methods of providing credentials. - Environment Variables, `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` - AWS Config or Credential Files, `~/.aws/config` and `~/.aws/credentials` - AWS Instance Profiles, uses the EC2 Metadata service

12.1 Using an Instance Profile

Typically, you'll deploy ElastAlert 2 on a running EC2 instance on AWS. You can assign a role to this instance that gives it permissions to read from and write to the Elasticsearch service. When using an Instance Profile, you will need to specify the `aws_region` in the configuration file or set the `AWS_DEFAULT_REGION` environment variable.

12.2 Using AWS profiles

You can also create a user with permissions on the Elasticsearch service and tell ElastAlert 2 to authenticate itself using that user. First, create an AWS profile in the machine where you'd like to run ElastAlert 2 for the user with permissions.

You can use the environment variables `AWS_DEFAULT_PROFILE` and `AWS_DEFAULT_REGION` or add two options to the configuration file: - `aws_region`: The AWS region where you want to operate. - `profile`: The name of the AWS profile to use to sign the requests.

CHAPTER

THIRTEEN

FREQUENTLY ASKED QUESTIONS

13.1 My rule is not getting any hits?

So you've managed to set up ElastAlert 2, write a rule, and run it, but nothing happens, or it says `0 query hits`. First of all, we recommend using the command `elastalert-test-rule rule.yaml` to debug. It will show you how many documents match your filters for the last 24 hours (or more, see `--help`), and then shows you if any alerts would have fired. If you have a filter in your rule, remove it and try again. This will show you if the index is correct and that you have at least some documents. If you have a filter in Kibana and want to recreate it in ElastAlert 2, you probably want to use a query string. Your filter will look like

```
filter:
- query:
    query_string:
      query: "foo: bar AND baz: abc*"
```

If you receive an error that Elasticsearch is unable to parse it, it's likely the YAML is not spaced correctly, and the filter is not in the right format. If you are using other types of filters, like `term`, a common pitfall is not realizing that you may need to use the analyzed token. This is the default if you are using Logstash. For example,

```
filter:
- term:
    foo: "Test Document"
```

will not match even if the original value for `foo` was exactly "Test Document". Instead, you want to use `foo.raw`. If you are still having trouble troubleshooting why your documents do not match, try running ElastAlert 2 with `--es_debug_trace /path/to/file.log`. This will log the queries made to Elasticsearch in full so that you can see exactly what is happening.

13.2 I got hits, why didn't I get an alert?

If you got logs that had `X query hits, 0 matches, 0 alerts sent`, it depends on the `type` why you didn't get any alerts. If `type: any`, a match will occur for every hit. If you are using `type: frequency`, `num_events` must occur within `timeframe` of each other for a match to occur. Different rules apply for different rule types.

If you see `X matches, 0 alerts sent`, this may occur for several reasons. If you set `aggregation`, the alert will not be sent until after that time has elapsed. If you have gotten an alert for this same rule before, that rule may be silenced for a period of time. The default is one minute between alerts. If a rule is silenced, you will see `Ignoring match for silenced rule` in the logs.

If you see `X alerts sent` but didn't get any alert, it's probably related to the alert configuration. If you are using the `--debug` flag, you will not receive any alerts. Instead, the alert text will be written to the console. Use `--verbose` to

achieve the same affects without preventing alerts. If you are using email alert, make sure you have it configured for an SMTP server. By default, it will connect to localhost on port 25. It will also use the word "elastalert" as the "From:" address. Some SMTP servers will reject this because it does not have a domain while others will add their own domain automatically. See the email section in the documentation for how to configure this.

13.3 Why did I only get one alert when I expected to get several?

There is a setting called `realert` which is the minimum time between two alerts for the same rule. Any alert that occurs within this time will simply be dropped. The default value for this is one minute. If you want to receive an alert for every single match, even if they occur right after each other, use

```
realert:
  minutes: 0
```

You can of course set it higher as well.

13.4 How can I prevent duplicate alerts?

By setting `realert`, you will prevent the same rule from alerting twice in an amount of time.

```
realert:
  days: 1
```

You can also prevent duplicates based on a certain field by using `query_key`. For example, to prevent multiple alerts for the same user, you might use

```
realert:
  hours: 8
query_key: user
```

Note that this will also affect the way many rule types work. If you are using `type: frequency` for example, `num_events` for a single value of `query_key` must occur before an alert will be sent. You can also use a compound of multiple fields for this key. For example, if you only wanted to receieve an alert once for a specific error and hostname, you could use

```
query_key: [error, hostname]
```

You can also write in the following way.

```
query_key:
  - error
  - hostname
```

Internally, this works by creating a new field for each document called `field1,field2` with a value of `value1, value2` and using that as the `query_key`.

The data for when an alert will fire again is stored in Elasticsearch in the `elastalert_status` index, with a `_type\ ``ofsilence`` and also cached in memory.

13.5 How can I change what's in the alert?

You can use the field `alert_text` to add custom text to an alert. By setting `alert_text_type: alert_text_only` Or `alert_text_type: alert_text_jinja`, it will be the entirety of the alert. You can also add different fields from the alert:

With `alert_text_type: alert_text_jinja` by using Jinja2 Template.

```
alert_text_type: alert_text_jinja

alert_text: |
  Alert triggered! *({{num_hits}} Matches!)*
  Something happened with {{username}} ({{email}})
  {{description|truncate}}
```

- Top fields are accessible via `{{field_name}}` or `{{_data['field_name']}}`, `_data` is useful when accessing *fields with dots in their keys*, as Jinja treat dot as a nested field.
- If `_data` conflicts with your top level data, use `jinja_root_name` to change its name.

With `alert_text_type: alert_text_only` by using Python style string formatting and `alert_text_args`. For example

```
alert_text: "Something happened with {0} at {1}"
alert_text_type: alert_text_only
alert_text_args: ["username", "@timestamp"]
```

You can also limit the alert to only containing certain fields from the document by using `include`.

```
include: ["ip_address", "hostname", "status"]
```

13.6 My alert only contains data for one event, how can I see more?

If you are using `type: frequency`, you can set the option `attach_related: true` and every document will be included in the alert. An alternative, which works for every type, is `top_count_keys`. This will show the top counts for each value for certain fields. For example, if you have

```
top_count_keys: ["ip_address", "status"]
```

and 10 documents matched your alert, it may contain something like

```
ip_address:
127.0.0.1: 7
10.0.0.1: 2
192.168.0.1: 1

status:
200: 9
500: 1
```

13.7 How can I make the alert come at a certain time?

The `aggregation` feature will take every alert that has occured over a period of time and send them together in one alert. You can use cron style syntax to send all alerts that have occured since the last once by using

```
aggregation:
  schedule: '2 4 * * mon,fri'
```

13.8 I have lots of documents and it's really slow, how can I speed it up?

There are several ways to potentially speed up queries. If you are using `index: logstash-*`, Elasticsearch will query all shards, even if they do not possibly contain data with the correct timestamp. Instead, you can use Python time format strings and set `use_strftime_index`

```
index: logstash-%Y.%m
use_strftime_index: true
```

Another thing you could change is `buffer_time`. By default, ElastAlert 2 will query large overlapping windows in order to ensure that it does not miss any events, even if they are indexed in real time. In config.yaml, you can adjust `buffer_time` to a smaller number to only query the most recent few minutes.

```
buffer_time:
  minutes: 5
```

By default, ElastAlert 2 will download every document in full before processing them. Instead, you can have ElastAlert 2 simply get a count of the number of documents that have occured in between each query. To do this, set `use_count_query: true`. This cannot be used if you use `query_key`, because ElastAlert 2 will not know the contents of each documents, just the total number of them. This also reduces the precision of alerts, because all events that occur between each query will be rounded to a single timestamp.

If you are using `query_key` (a single key, not multiple keys) you can use `use_terms_query`. This will make ElastAlert 2 perform a terms aggregation to get the counts for each value of a certain field. May not be compatible with all rule types.

13.9 Can I perform aggregations?

The only aggregation supported currently is a terms aggregation, by setting `use_terms_query`.

13.10 I'm not using @timestamp, what do I do?

You can use `timestamp_field` to change which field ElastAlert 2 will use as the timestamp. You can use `timestamp_type` to change it between ISO 8601 and unix timestamps. You must have some kind of timestamp for ElastAlert 2 to work. If your events are not in real time, you can use `query_delay` and `buffer_time` to adjust when ElastAlert 2 will look for documents.

13.11 I'm using flatline but I don't see any alerts

When using `type: flatline`, ElastAlert 2 must see at least one document before it will alert you that it has stopped seeing them.

13.12 How can I get a "resolve" event?

ElastAlert 2 does not currently support stateful alerts or resolve events. However, if you have a rule alerting you that a condition has occurred, such as a service being down, then you can create a second rule that will monitor the first rule, and alert you when the first rule ceases to trigger.

For example, assuming you already have a rule named "Service is offline" that's working today, you can add a second rule as follows:

```
name: Service is back online
type: flatline
index: elastalert*
query_key: "rule_name"
filter:
- query:
    query_string:
      query: "rule_name:\"Service is offline\" AND matches:>0"
forget_keys: true
timeframe:
  minutes: 30
threshold: 1
```

This second rule will trigger after the timeframe of 30 minutes has elapsed with no further matches against the first rule.

13.13 Can I set a warning threshold?

Currently, the only way to set a warning threshold is by creating a second rule with a lower threshold.

13.14 Does it support Elastic Cloud's "Cloud ID"?

While Elastic Cloud is supported via the traditional URL connection method, connecting via Cloud ID is not currently supported.

13.15 I need to go through an http (s) proxy to connect to Elasticsearch. Does ElastAlert 2 support it?

Not supported.

13.16 About boolean value

You can use all lowercase letters or only uppercase letters at the beginning.

example

```
# OK
use_ssl: true
# OK
use_ssl: True
# OK
use_ssl: false
# OK
use_ssl: False
```

13.17 Is it possible to send an SNMP Trap with an alert notification?

- You need to additionally install snmp snmptrapd on the docker image. In other words, you need to modify the Dockerfile and recreate the Docker image with docker build.
- It is possible with the command Alerter.

example

```
name: "mariadb-error-log-warning"
type: "frequency"
index: "mariadb-*"
num_events: 1
timeframe:
    minutes: 5
realert:
  minutes: 1
filter:
  - query:
      query_string:
        query: "@log_name:mysqld.error AND message:Warning"
alert:
  - command
command: ["/usr/bin/snmptrap", "-IR", "-v", "2c", "-c", "public", "xxx.xxx.xxx.xxxxx:xxx
↪", "", "netSnmp.99999", "netSnmp.99999.1", "s", "Hello, World"]
is_enabled: true
timestamp_field: "@timestamp"
timestamp_type: "iso"
use_strftime_index: false
```

13.18 Is Email Alerter compatible with Microsoft 365 (formerly Office 365)?

Not supported.

13.19 Does Email Alerter support the Google Gmail API?

Not supported.

13.20 Can Email Alerter send emails via the Gmail sending server?

It is possible. However, you need to turn on (enable) the item "Access to insecure apps" in the "Security" settings of your Google account.

13.21 Is it possible to send a JPEG image encoded as base64 in elasticsearch as an image attachment with an Email Alerter?

Yes, this is possible if the base64 encoded bytes are available in the matched document, as shown in the example below:

```
include: [base64field]
alert_text_args: [base64field]
email_format: "html"
alert_text_type: alert_text_only
alert_text: |
  <html>
  <body>
  <div>
    <img src=" {}" alt="Image" />
  </div>
  </body>
  </html>
```

13.22 Does the alert notification destination support Alertmanager?

Now supported as of ElastAlert 2.2.3.

13.23 The es_host parameter seems to use only one host. Is it possible to specify multiple nodes?

There are two options:

1. Use haproxy in front of elasticsearch to support multiple hosts.
2. Use the new `es_hosts` parameter introduced in ElastAlert 2.2.3. See *Configuration*.

13.24 Is there any plan to implement a REST API into this project?

No plan.

13.25 An error occurred when trying to create a blacklist rule that parses a file with more than 1024 lines.

This is the default limit for ElasticSearch. Specifying more than 1024 items in the blacklist will result in an error. This is a known issue. Perhaps White List can have similar issues. See the following issues on the original yelp/elastalert for more information.

https://github.com/Yelp/elastalert/issues/1867 https://github.com/Yelp/elastalert/issues/2704

13.26 ElastAlert 2 doesn't have a listening port?

ElastAlert 2 does not have a network API. There is no listening port. You can monitor its activity by viewing the console output or Docker logs.

13.27 I've set `ssl_show_warn` but it doesn't seem to work.

Now supported as of ElastAlert 2.4.0.

13.28 How to write a query filter for phrases containing spaces?

To search for values containing spaces, or other special characters you will need to use escape characters. This is briefly mentioned at the bottom of the Lucene Query Parser Syntax documentation but does not go into extensive detail. Below are some examples to use in ElastAlert 2 rule filters.

Example 1 - Escaping double quotes within double quotes. Useful for embedded single quotes and double quotes in your search phrase:

```
filter:
 - query:
     query_string:
       query: "\"Women's Clothing\""
```

Example 2 - Avoiding escaping altogether by enclosing double quotes within single quotes:

```
filter:
 - query:
     query_string:
       query: '"Rabbia Al"'
```

13.29 Does ElastAlert 2 support Elasticsearch 8?

ElastAlert 2 supports Elasticsearch 8.

To upgrade an existing ElastAlert 2 installation to Elasticsearch 8 the following manual steps are required (note the important WARNING below):

- Shutdown ElastAlert 2.
- Delete the old `elastalert*` indices. See Elasticsearch documentation for instructions on how to delete via the API, or use the Kibana Index Management interface.
- Upgrade the Elastic cluster to Elasticsearch 8 following the Elastic 8 upgrade instructions.
- If NOT running ElastAlert 2 via Docker or Kubernetes, run elastalert-create-index to create the new indices. This is not needed when running via a container since the container always attempts to creates the indices at startup, if they're not yet created.
- Restart ElastAlert 2.

WARNING: Failure to remove the old ElastAlert indices can result in a non-working Elasticsearch cluster. This is because the ElastAlert indices contain deprecated features and the Elasticsearch 8 upgrade logic is currently flawed and does not correctly handle this situation. The Elasticsearch GitHub repository contains more information on this problem.

13.30 Support multiple sns_topic_arn in Alert Amazon SNS(Simple Notification Service)?

example

```
alert:
 - sns:
     sns_topic_arn: "aws-topic1"
 - sns:
     sns_topic_arn: "aws-topic2"
```

13.31 Support multiple telegram_room_id in Alert Telegram?

example

```
alert:
 - telegram:
     telegram_room_id: "AAA"
 - telegram:
```

(continues on next page)

(continued from previous page)

```
      telegram_room_id: "BBB"
telegram_bot_token: "XXX"
```

CHAPTER

FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search