Running ElastAlert for the First Time

Requirements

- Elasticsearch
- ISO8601 or Unix timestamped data
- Python 2.7
- pip, see requirements.txt
- Packages on Ubuntu 14.x: python-pip python-dev libffi-dev libssl-dev

Downloading and Configuring

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

```
$ pip install elastalert
```

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

```
$ git clone https://github.com/Yelp/elastalert.git
```

Install the module:

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

Depending on the version of Elasticsearch, you may need to manually install the correct version of elasticsearch-py.

Elasticsearch 5.0+:

```
$ pip install "elasticsearch>=5.0.0"
```

Elasticsearch 2.X:

```
$ pip install "elasticsearch<3.0.0"
```

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

`rules_folder` is where ElastAlert will load rule configuration files from. It will attempt to load every .yaml file in the folder. Without any valid rules, ElastAlert will not start. ElastAlert 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 example_rules folder.

`run_every` is how often ElastAlert 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 address of an Elasticsearch cluster where ElastAlert will store data about its state, queries run, alerts, and errors. Each rule may also use a different Elasticsearch host to query against.

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

`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`

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

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

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

`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 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`

Setting Up Elasticsearch

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

First, we need to create an index for ElastAlert to write to by running `elastalert-create-index` and following the instructions:

```
$ 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 Metadata Index.

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 `example_rules/example_frequency.yaml` as a template:

```
# From example_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 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 will try to use `@timestamp` by default, but this can be changed with the `timestamp_field` option. By default, ElastAlert 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."

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 example_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> example_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

Running ElastAlert

There are two ways of invoking ElastAlert. 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 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 will attempt to load the other rules in the example_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 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 reques
```

This line showing that ElastAlert 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 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 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 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 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`.