# Evaluate ElastAlert for IT-DB use cases

## August 2019

**AUTHOR:**
Dimitra Chatzichrysou
Infrastructure & Automation (IT-DB-IA)

**SUPERVISOR:**
Aimilios Tsouvelekakis
Infrastructure & Automation (IT-DB-IA)

CERN openlab

# ABSTRACT

The Database Services Group (IT-DB) is responsible for providing database and middleware services to the laboratory. For these services, it is necessary to provide proper monitoring solutions to different user communities. The goal of this project is the evaluation and deployment of ElastAlert framework on IT-DB infrastructure, aiming to help the group and its user groups to have a better alerting over the stored data in the ElasticSearch cluster.

# TABLE OF CONTENTS

# 1. Introduction

The logging infrastructure consists of several components. These components collect essential information about servers and processes running on them, database instances, application servers and applications. For this functionality, the Elastic Stack (ElasticSearch, Logstash, Kibana, Beats) is used for log management.

To be more specific, logs from all the aforementioned data sources are collected from Filebeat. Filebeat is a lightweight shipper for forwarding and centralizing log data. It monitors the log files, collects log events, and forwards them to Logstash.

Logstash is used to aggregate and process data. Logstash is an open-source, server-side data processing pipeline that filters the data before it is indexed into Elasticsearch.

During the indexing process, ElasticSearch stores documents and builds an inverted index to make the document data searchable in near real-time and allow fast full-text searches. Elasticsearch stores data as JSON documents and an ElasticSearch index is a collection of documents that are related to each other. Each document correlates a set of keys (names of fields or properties) with their corresponding values.

Since the logs are indexed in ElasticSearch, users can run queries and use aggregations to retrieve summaries of the data. On top of that, Kibana is used for creating visualizations of the data, share dashboards and manage the Elastic Stack.

With so much data being generated every day, it is significant to have a full picture of systems' status at all times and be alerted for inconsistencies in those data. ElastAlert can be used as a companion tool with the Elastic Stack and trigger alerts when data matches certain patterns of interest.

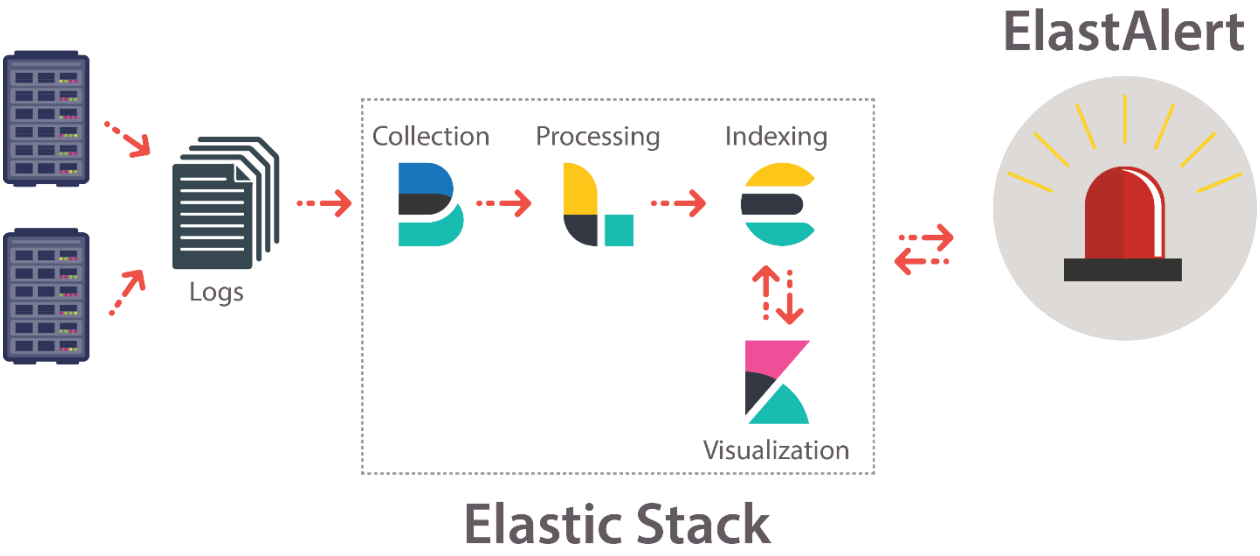


*Figure 1. Log management in IT-DB and monitoring with ElastAlert*

# 2. ElastAlert

ElastAlert is a framework written in Python, for real-time alerting on anomalies, spikes, or other patterns of interest from data in ElasticSearch. It combines two types of components: rule types and alerts.

ElastAlert periodically queries ElasticSearch and data are passed to the rule type. The rule type determines when a match is found and a match activates one or more alerts. This process is configured by a set of rules, where each rule defines a query, a rule type, and a set of alerts.


*Figure 2. How ElastAlert works*

## 2.1 ElastAlert Features

The main ElastAlert features are:

- Reliability: ElastAlert uses ElasticSearch to store information about its state on the writeback_index. This allows auditing and debugging of ElastAlert's operation, as well as avoiding loss of data when ElastAlert is restarted, or crashes. As a result, ElastAlert will resume where it previously stopped.
- High modularity: The main components of ElastAlert are the rule types and alerts. They can be imported as a module or customized.
- Easy configuration: It is easy to set up and configure ElastAlert, given that only a global configuration file and a set of rules have to be specified.

## 2.2 Rule Types

Rule types are responsible for processing the data returned from ElasticSearch and outputs the matches based on these data. ElastAlert supports different types of rules. Regarding the DB group's needs for monitoring, the implemented types of rules are:

- flatline: Match when there are less than X events in Y time.
- frequency: Match where there are X events in Y time.
- any: Match on any event based on a given filter.

The rule configuration file is in charge of initializing the rule type. Each rule defines a query to perform, parameters on what triggers a match, and types of alerts. *Figure 3* shows an example of a rule configuration file.

```
# Rule for ElastAlert

name: itdb_dia_logstash_errors

type: frequency

index: itdb_dia-logstash-*

num_events: 50

timeframe:
  minutes: 30

use_count_query: true

doc_type: doc

use_terms_query: false

realert:
  hours: 0

alert:
- servicenow
servicenow_rest_url: https://cerntraining.service-now.com/api/now/v1/table/incident
alert_subject: "ElastAlert: Rule {0} has been triggered in index {1}"
alert_subject_args:
- name
- index
assignment_group: "IT-DB Automation Tools 4th Line Support"
u_business_element: "Weblogic, Tomcat Java application servers and 3rd party packages"
u_functional_element: "Weblogic, Tomcat Java application servers and 3rd party packages
support"
```

*Figure 3. Rule configuration file itdb_dia_logstash_errors*

## 2.3 Alert Types

Alerts are responsible for taking action based on a match. A match is generally a dictionary containing values from a document in ElasticSearch. When a match is found, it is given to one or more alerters, which will perform an action based on the match.

ElastAlert supports different alert types (e.g. Email, Jira, Mattermost, Slack, ServiceNow, HTTP requests, etc.) and offers the option of creating a custom alerter.

For this specific project, Email and ServiceNow alert types are used.

ServiceNow is used at CERN as a service management application for maintenance and support of services. When an alert is triggered, an email is sent or an incident is reported at ServiceNow. In this way, the accountable group will be notified of the service's status. *Figure 4* shows an incident at ServiceNow created by ElastAlert.

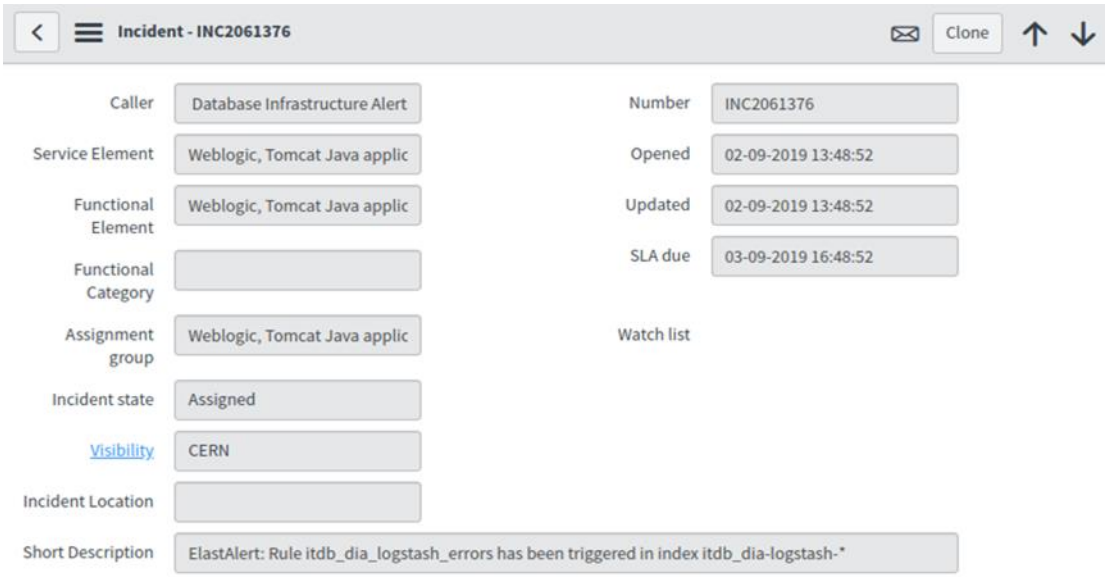


*Figure 4. Incident at ServiceNow*

## 2.4 Global configuration file

ElastAlert has a global configuration file, which defines several aspects of its operation and common settings to all rules.

*Figure 5* shows an example of a global ElastAlert configuration file:

```
rules_folder: /etc/elastalert/rules

scan_subdirectories: true

run_every:
  minutes: 5

buffer_time:
  hours: 1

es_host: <host>

es_port: 443

es_url_prefix: es

use_ssl: True

verify_certs: True

ca_certs: /etc/pki/tls/certs/CERN-bundle.pem

es_username: <username>
es_password: "{{password}}"

writeback_index: itdb_elastalert

alert_time_limit:
  days: 2
```

```
snow_username: <username>
snow_password: "{{password}}"
```

*Figure 5. Global configuration file config.yaml*

The global configuration file defines:

| Field | Usage |
|---|---|
| **rules_folder** | The name of the directory which contains rule configuration files. ElastAlert will load all files in this directory, and all its subdirectories, that end in **.yaml**. If the contents of this directory change, ElastAlert will load, reload or remove rules based on their respective config files. |
| **scan_subdirectories** | Sets whether or not ElastAlert should recursively descend the rules directory. |
| **run_every** | How often ElastAlert should query Elasticsearch. ElastAlert will remember the last time it ran the query for a given rule, and periodically query from that time until the present. |
| **es_host** | The host name of the Elasticsearch cluster. |
| **es_url_prefix** | URL prefix for the Elasticsearch endpoint. |
| **use_ssl** | Connect to es_host using TLS |
| **verify_certs** | Verify TLS certificates. |
| **ca_certs** | Path to a PEM certificate to use as the client certificate. |
| **es_username** | Basic-auth username for connecting to es_host. |
| **es_password** | Basic-auth password for connecting to es_host. |
| **writeback_index** | The index on es_host to use. |
| **alert_time_limit** | Alerts which throw errors may be automatically retried for a period of time. |
| **snow_username** | The ServiceNow Username to access the api. |
| **snow_password** | The ServiceNow password to access the api. |

## 2.5 Contributions

Incidents are created at ServiceNow via its REST API. Business element and Functional element are required parameters for the creation of the incident, that the ServiceNow Alerter did not support. Further, some of the required options had to be optional and the “Short Description” field on ServiceNow had to be configured for each incident from the rule configuration file. To keep it as simple as possible for the users, the credentials for the ServiceNow Alerter had to be declared on the global configuration file, and not in each rule file.

As a result, the source code of the framework was modified by:

- refactoring the ServiceNow Alerter,
- redefining required and optional parameters, and
- adding required credentials at the global configuration file.

## 3. Deployment

An overview of ElastAlert's deployment is given in *Figure 6*.

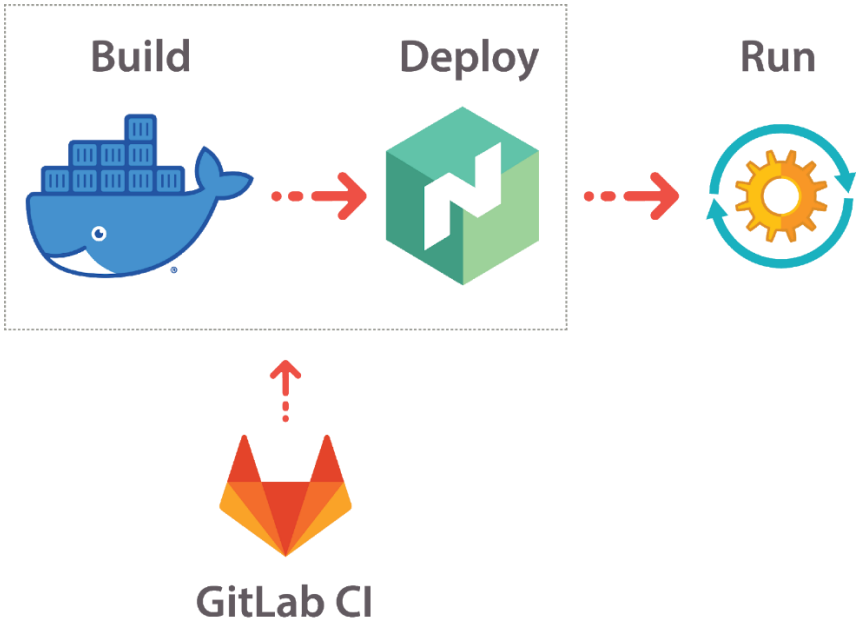


*Figure 6. How ElastAlert runs*

### 3.1 GitLab CI Pipeline

GitLab CI is used for the Continuous Integration of the project and allows us to run several tasks whenever new code is pushed to the GitLab repo. GitLab CI pipelines are configured using a YAML file (i.e. `.gitlab-ci.yml`) within the project. A pipeline is a group of steps that are grouped by similar characteristics. These steps are called jobs and each job belongs to a single stage.

The following three stages have been defined:

- docker-build
- lint
- deploy

For each stage the following jobs have been defined:

- build-dev: the Docker Image is built and uploaded to the Container registry
- yaml-lint-dev: Validate syntax of each rule configuration file in the rules directory
- deploy-dev: Run ElastAlert as a service

*Figure 7* shows an example pipeline.

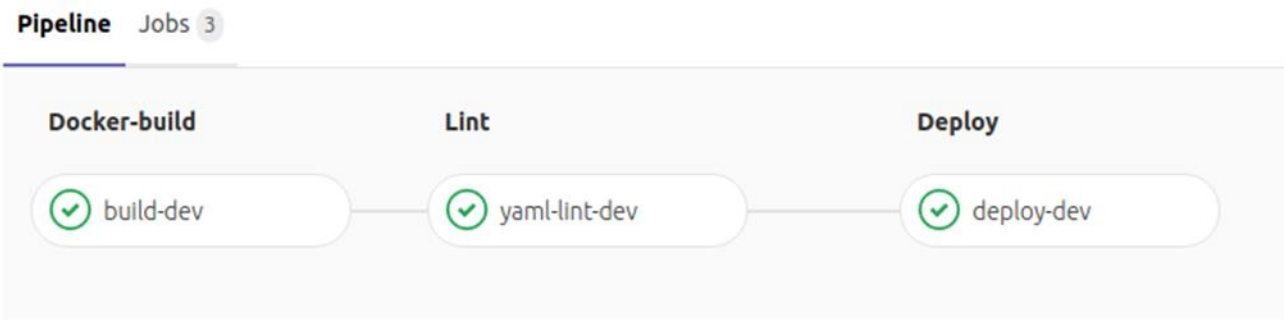


*Figure 7. Triggered pipeline at GitLab CI*

## 3.2 DOCKER

ElastAlert is running as a containerized service. A Dockerfile in the project's repository defines what goes on in the environment inside the container. In the Dockerfile, the required packages for ElastAlert, ElastAlert, and various configurations are specified. Whenever the Dockerfile is updated, a pipeline is triggered to build the new image and push it to CERN's Gitlab Container Registry.

## 3.3 NOMAD

ElastAlert is deployed to CERN's clusters using Nomad. Nomad is a flexible workload orchestrator that enables easy deployment and management of any containerized application. Nomad automatically handles application, node, and driver failures. Declarative job files are used for scheduling virtualized, containerized and standalone applications. Jobs are the primary configuration that users interact with when using Nomad.

A job is a declarative specification of tasks that Nomad should run. The hierarchy for a job is:

job
  _ group
      _ task

The Nomad job file elastalert.nomad is given in *Figure 8*.

Nomad job file **`elastalert.nomad`** defines the below:

- Ship the global configuration file with the template Stanza.
- Fetch and unpack rules directory from the GitLab repository with the artifact Stanza.
- Fetch ElastAlert's docker image from the GitLab Container Registry with Docker Driver.
- Define the container's entrypoint with args.
- Bind host paths to container paths for mounting the global configuration file and the rules directory.
- Instruct Nomad to register the task "elastalert" as a service with service Stanza.

```
job "elastalert" {
  datacenters = ["datacenter-1", "datacenter-2", "datacenter-3"]
  type = "service"

  group "monitoring" {
```

```
    count = 1
    restart {
      attempts = 2
      interval = "1h"
      delay = "15s"
      mode = "fail"
    }
    task "elastalert" {
      template {
        change_mode = "restart"
        destination = "local/config.yaml"
        data = <<EOH
          rules_folder: /etc/elastalert/rules

          scan_subdirectories: true

          run_every:
            minutes: 5

          buffer_time:
            hours: 1

          es_host: <host>

          es_port: 443

          es_url_prefix: es

          use_ssl: True

          verify_certs: True

          ca_certs: /etc/pki/tls/certs/CERN-bundle.pem

          es_username: <username
          es_password: "{{password}}"

          writeback_index: itdb_elastalert

          alert_time_limit:
            days: 2

          snow_username: <username>
          snow_password: "{{password}}"
        EOH
      }

      artifact {
        source = "git::https://<username>:{{password}}@gitlab.cern.ch/db/elasticsearch-
project/elastalert.git//rules"
        destination = "local/repo"
      }

      driver = "docker"
      config {
        image = "gitlab-registry.cern.ch/db/elasticsearch-
project/elastalert/elastalert:e429a9f0"
        auth {
          username = <username>
          password = "{{password}}"
        }
        {% if ci_ref == "master" %}
```

```
        args = ["/usr/bin/elastalert", "--config", "/etc/elastalert/config.yaml", "--
verbose"]
        {% else %}
        args = ["/usr/bin/elastalert", "--config", "/etc/elastalert/config.yaml", "--
debug"]
        {% endif %}
        volumes = [
          "local/config.yaml:/etc/elastalert/config.yaml",
          "local/repo:/etc/elastalert/rules"
        ]
      }

      resources {
        cpu    = 800
        memory = 1024
      }

      service {
        name = "elastalert"
      }
    }
  }
}
```

*Figure 8. Nomad job file elastalert.nomad*

## 4. Conclusion

The present work delivered the evaluation of ElastAlert for IT-DB use cases. ElastAlert can be used to provide better monitoring over data. A set of rules were defined for monitoring IT-DB services and ElastAlert was deployed on CERN's infrastructure. By combining Docker and Nomad, ElastAlert is already running as a highly available service on dev clusters. Minor changes should be done to the rules to run ElastAlert fully on production. Future work will include the upgrade of ElastAlert's version in Python 3.6 and the creation of dashboards in Grafana for analytics and visualization.

## 5. Bibliography


1. ElastAlert GitHub repository,
   https://github.com/Yelp/elastalert

2. ElastAlert Documentation,
   https://elastalert.readthedocs.io/

3. Elastic Stack Documentation,
   https://www.elastic.co/guide/index.html

4. GitLab CI for Continuous Integration of GitLab projects,
   https://cern.service-now.com/service-portal/article.do?n=KB0003690

5. GitLab CI/CD Pipeline Configuration Reference,
   https://gitlab.cern.ch/help/ci/yaml/README.md

6. yamllint Documentation,
   https://yamllint.readthedocs.io/

7. Docker overview,
   https://docs.docker.com/engine/docker-overview/

8. Practices for writing Dockerfiles,
   https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

9. Nomad Documentation,
   https://www.nomadproject.io/docs/index.html

10. go-getter Library GitHub repository,
    https://github.com/hashicorp/go-getter

11. Resources for design *Figures 1, 2 & 6*,
    https://www.freepik.com