Signals: Enterprise Alerting for Elasticsearch for Free

Jochen Kressin · 2024-10-15

Search Guard enhances Elasticsearch and Kibana by adding features and functionalities that simplify enterprise-level operations. This article introduces Signals, our professional Alerting solution for the ELK stack. The best part: Signals provides Elasticsearch Alerting for free!

Watch the video in the online version of this article.

Signals is a sophisticated monitoring and notification system for Elasticsearch and Kibana, designed to ingest, monitor, analyze, and correlate data from various sources. Including but not limited to data in Elasticsearch indices.

By setting up *Watches* and *Conditions*, Signals can alert you when anomalies are detected in your data. Notifications can be sent via Email, Slack, PagerDuty, and more. You can also establish advanced escalation models, using different channels depending on the issue's severity.

For setting up and configuring alerts, you can either use the Signals Alerting REST API or the Signals Alerting Kibana UI.

In contrast to Kibana Alerts, the offering from Elastic, Signals works as a fully-fledged plugin for Elasticsearch. This means you can run it independently of Kibana, and you can scale Signals nearly infinitely. By default, Signals uniformly distributes its watches over all nodes of an Elasticsearch cluster, so it automatically scales with your system. We have seen real-life use cases of customers who run many hundreds of Watches in parallel.

However, if you want to limit or control in which nodes Signals is running, we have administration options that let you do exactly that.

Enough theory ... let's delve into the core components of Signals Alerting and demonstrate how to set up a common use case: log file monitoring.

Use Case: Application Log File Monitoring

In this scenario, we use Elasticsearch to ingest log files from a customer-facing web application. Each log line is transformed into a document with fields like timestamp, log message, and HTTP response code. Our goal is to be send notifications from our Elasticsearch cluster if anomalies in the application's error rate are detected.

Watches

First, we set up a watch. A watch fetches data from one or more input sources at defined intervals and sends notifications if certain conditions are met. It can also perform calculations, transformations, and correlations on the data. Here's the basic structure of a watch:

```
{
  "trigger": {
     ...
  },
  "checks": [
    ...
  ],
  "actions": [
  ],
}
```

This might seem a bit abstract, so let's break it down.

To check the application logs regularly, we set up a *trigger* that runs every 10 minutes. Signals offers a variety of predefined triggers, such as hourly, daily, or cron-based triggers. For our case, we use a 10-minute interval trigger:

```
"trigger": {
  "schedule": {
    "interval": "10m"
   }
}
```

Next, we need to access the data to be examined. Since our data is stored in Elasticsearch, we define a Search Input for our watch. A search input runs an Elasticsearch query on one or more indices and makes the results available for subsequent steps of the Watch. We create a query that selects all log entries with an error code 500 from the past 10 minutes in an index named "logs":

```
"checks": [
  {
    "type": "search",
    ...
    "request": {
        "indices": [
            "logs"
        ],
        "body": {
          "query": {
            "bool": {
              "must": [
                {
                  "range": {
                    "timestamp": {
                      "gte": "now-10m"
                    }
                  }
                },
                {
                  "match": {
                    "response_code": 500
                  }
                }
              ]
            }
          }
        }
      }
    }
  ]
```

Conditions

Conditions control the execution flow of a Watch. A condition decides whether the Watch should proceed based on whether a specific value or threshold is reached. We want to be notified if the number of errors exceeds 5, which corresponds to the number of documents found by our query:

```
{
  "type": "condition",
  "name": "error rate above 5",
  "source": "data.searchresult.hits.total.value > 5"
}
```

If fewer than 5 documents are found, the execution stops here. Otherwise, Signals performs the defined actions.

Actions

Actions are used to send notifications via email or messaging services like Slack and PagerDuty, or to write data back to Elasticsearch indices. A general-purpose action is the *Webhook* action, which makes HTTP requests to configurable endpoints.

Signals supports these action types:

- Email
- Slack
- PagerDuty
- JIRA
- Webhooks
- Index (for writing data back to Elasticsearch)

The free (as in free beer) Community Edition includes Email Alerts, Slack, Webhooks and Index actions.

Escalation Model: Severity Levels

In real-world scenarios, you need fine-grained control over notifications. Not all anomalies are equal. A minor increase in error rates might warrant an email to the DevOps team, while a major spike might require immediate PagerDuty notfications.

Signals allows you to associate any metric observed by a watch with a severity level and assign actions to these levels. Actions execute only if these severity levels are reached.

In our use case, we define two severity levels: one for error rates between 5 and 10, and one for error rates above 10.

```
"severity": {
    "value": "data.searchresult.hits.total.value",
    "order": "ascending",
    "mapping": [
        {
            "threshold": 5,
            "level": "warning"
        },
        {
            "threshold": 10,
            "level": "error"
        }
    ]
}
```

We can then map actions to these severity levels:

```
"actions": [
    {
        "type": "email",
        "severity": ["warning"],
        ...
    },
    {
        "type": "pagerduty",
        "severity": ["error"],
        ...
    }
]
```

Throttling and Acknowledgement

It might take time to identify and fix issues. Signals allows throttling or acknowledging actions to prevent notification flooding.

A throttle limits the execution frequency of an action within a specified period:

```
{
  "actions": [
      {
          "type": "email",
          "throttle_period": "1h",
        ...
      }
  ]
}
```

Acknowledging an action silences it until it is unacknowledged. This can be done via the REST API or the Signals UI. Acknowledgement links can also be added to notifications using template variables.

Resolve Actions

Receiving notifications when issues are resolved is as important as knowing when they arise. Resolve actions are executed when a previously active severity level is no longer active.

For instance, to send a Slack message when the error level decreases from "error" to "warning", we define a resolve action like this:

```
 "resolve_actions": [
     {
         "type": "slack",
         "resolves_severity": [ "error" ],
         "text": "Severity level is now {{severity.level}}; it was before:
{{resolved.severity.level}}. The error rate has decreased from {{resolved.severity.value}} to
{{severity.value}}"
     }
 ]
```

Access Control: Multi-Tenancy Integration

Signals integrates seamlessly with the Search Guard Security features, allowing you to control which users can access watches and notification channels. This ensures that you can manage who can create, modify, or delete watches and notification channels.

Next Steps

We will follow up this overview with detailed posts on each Signals feature. In the meantime, download and install the Search Guard and explore its capabilities. We welcome any questions or feedback on the Search Guard Forum.

This article was published on the Search Guard blog: https://search-guard.com/blog/enterprise-alerting-elasticsearch-kibana-free/