

Centralized Logging

Raúl Estrada

Octubre 2020

Chapter 7. Metrics, Log Collection, and Monitoring

That's it. This chapter could well have ended here but I shall carry on for the benefit of those amongst us who would prefer things in more detail.

A great deal of the DevOps practice(s) evolve around the idea of being able to review and react to the state of your infrastructure at any given time – should you need to.

That is not to say, setup e-mail notifications for every time the date changes on your host, but a stream of sensible, usable amount of event data which would allow an operator to make a reasonably informed decision under stress and/or uncertainty.

If you have been paying attention in life so far, you would have noticed many a wise man talking about **balance, the golden middle**.

You should aim to configure your monitoring system in a way that you are notified of events of potential interest and in a timely manner. The notifications should arrive in a format that is hard to overlook, and should provide enough detail for an operator to be able to make an informed guess at what is going on.

At the same time, the said monitoring system must cause the least amount of alert fatigue (as outlined in this concise Datadog article: <https://www.datadoghq.com/blog/monitoring-101-alerting>).

Unfortunately for our friendship, finding that middle ground which suits your case (your infrastructure and the people looking after it) is an adventure which you will have to go on alone. We could however spend some quality time together, discussing a few of the tools that could make it even more enjoyable!

Checklists are sophisticated, so here is one:

➤ Centralized logging:

- Ingesting and storing logs with **Logstash** and **Elasticsearch**
- Collecting logs with Elasticsearch **Filebeat**
- Visualizing logs with **Kibana**

➤ Metrics:

- Ingesting and storing metrics with **Prometheus**
- Gathering OS and application metrics with **Telegraf**
- Visualizing metrics with **Grafana**

➤ Monitoring:

- Alerting with **Prometheus**
- Self-remediation with **Prometheus** and **Jenkins**

Naturally, we would require a few hosts to form our playground for all of the preceding checklist. There has been sufficient practice in deploying VPC EC2 instances on AWS in previous chapters, thus I hereby exercise the great power of delegation and assume the existence of:

- A VPC with an IGW, NAT gateway, 2x private and 2x public subnets
- 2x standalone, vanilla Amazon Linux EC2 instances (say `t2.small`) within the public subnets
- 1x Auto Scale Group (`t2.nano`) within the private subnets
- 1x Internet-facing ELB passing HTTP traffic to the Auto Scale Group

Centralized logging

Since the olden days, mankind has strived to use its limited attention span only on what really matters in life, and without having to look for it too hard – if possible. So we started with copying log files around, evolution brought us centralized (r)syslog and today (we learn from our mistakes) we have Logstash and Elasticsearch.

Elasticsearch is a distributed, open source search and analytics engine, designed for horizontal scalability, reliability, and easy management. It combines the speed of search with the power of analytics via a sophisticated, developer-friendly query language covering structured, unstructured, and time-series data.

Logstash is a flexible, open source data collection, enrichment, and transportation pipeline. With connectors to common infrastructure for easy integration, Logstash is designed to efficiently process a growing list of log, event, and unstructured data sources for distribution into a variety of outputs, including Elasticsearch.

--<https://www.elastic.co/products>

Ingesting and storing logs with Logstash and Elasticsearch

We will be using Logstash to receive, process and then store log events into Elasticsearch.

For the purposes of the demos in this chapter, we'll be installing and configuring services manually, directly on the hosts. When done experimenting, you should, of course, use configuration management instead (wink).

Let us start by installing the two services on one of the standalone EC2 instances (we shall call it ELK):

```
# yum -y install https://download.elastic.co/elasticsearch/release/org/elasticsearch/distribution/rpm/elast
```

Copy

Edit `/etc/elasticsearch/elasticsearch.yml` :

Note

Please refer to: https://github.com/PacktPublishing/Implementing-DevOps-on-AWS/blob/master/5585_07_CodeFiles/elk/etc/elasticsearch/elasticsearch.yml

[Copy](#)

```
cluster.name: wonga-bonga
index.number_of_shards: 1
index.number_of_replicas: 0
index :
  refresh_interval: 5s
```

It is important to select a unique name for the Elasticsearch cluster, so that the node does not join somebody else's inadvertently, should there be any on your LAN. For development, we only ask for a single shard and no replicas. Impatience dictates a five second refresh rate on any ES indices.

Create a Logstash **patterns** folder:

Copy

```
# mkdir /opt/logstash/patterns
```

Create a sample NGINX pattern `/opt/logstash/patterns/nginx` (ref: <https://www.digitalocean.com/community/tutorials/adding-logstash-filters-to-improve-centralized-logging>):

Note

Please refer to: https://github.com/PacktPublishing/Implementing-DevOps-on-AWS/blob/master/5585_07_CodeFiles/elk/opt/logstash/patterns/nginx

Copy

```
NGUSERNAME [a-zA-Z\.\@\-\+\_%]+
NGUSER %{NGUSERNAME}
NGINXACCESS %{IPORHOST:clientip} %{NGUSER:ident} %{NGUSER:auth} \[%{HTTPDATE:timestamp}\] "%{WORD:verb} %{L
```

Create `/etc/logstash/conf.d/main.conf` :

Note

Please refer to: https://github.com/PacktPublishing/Implementing-DevOps-on-AWS/blob/master/5585_07_CodeFiles/elk/etc/logstash/conf.d/main.conf

Copy

```
input {
  beats {
    port => 5044
  }
}

filter {
  if [type] == "nginx-access" {
    grok {
      match => { "message" => "%{NGINXACCESS}" }
    }
  }
}

output {
  elasticsearch {
    hosts => "localhost:9200"
    manage_template => false
    index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
    document_type => "%{[@metadata][type]}"
  }
}
```

Logstash allows us to configure one or more listeners (inputs) in order to receive data, filters to help us process it and outputs specifying where that data should be forwarded once processed.

We expect logs to be delivered by Elasticsearch Filebeat on **TCP: 5044** . If the log event happens to be of type **nginx-access** , we have it modified according to the **NGINXACCESS** pattern then shipped to Elasticsearch on localhost **TCP: 9200** for storage.

Finally, let us start the services:

```
# service elasticsearch start  
# service logstash start
```

Copy

Collecting logs with Elasticsearch Filebeat

We have the systems in place; let us push some from the ELK node that we are on.

We will use Filebeat to collect local logs of interest and forward those to Logstash (incidentally also local in this case):

Filebeat is a log data shipper. Installed as an agent on your servers, Filebeat monitors the log directories or specific log files, tails the files, and forwards them either to Elasticsearch or Logstash for indexing.

--<https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html>

Installation:

Copy

```
# yum -y install https://download.elastic.co/beats/filebeat/filebeat-1.3.1-x86_64.rpm
```

While functionality is provided to ship directly to ES, we are planning to use Logstash so we need to disable the Elasticsearch output and enable the logstash one in `/etc/filebeat/filebeat.yml` :

Note

Please refer to: https://github.com/PacktPublishing/Implementing-DevOps-on-AWS/blob/master/5585_07_CodeFiles/elk/etc/filebeat/filebeat.yml

[Copy](#)

```
output:
  #elasticsearch:
  # hosts: ["localhost:9200"]
  logstash:
    hosts: ["localhost:5044"]
```

We could also list a few more log files to collect:

[Copy](#)

```
filebeat:
  prospectors:
    -
      paths:
        - /var/log/*.log
        - /var/log/messages
        - /var/log/secure
```

Then start the service:

```
# service filebeat start
```

Copy

Fun, but let us launch a few other EC2 instances for even more of it!

We shall use the Auto Scale Group we mentioned earlier. We will install Filebeat on each instance and configure it to forward selected logs to our Logstash node.

First, ensure that the security group of the Logstash instance allows inbound connections from the Auto Scale Group (**TCP: 5044**).

Next, we use an EC2 User Data script to bootstrap the Filebeat binary and configuration onto each of the EC2 instances in our Auto Scale Group (we will call them webserver):

Note

Please refer to: https://github.com/PacktPublishing/Implementing-DevOps-on-AWS/blob/master/5585_07_CodeFiles/webserver/user_data.sh

With that in place, go ahead and scale the group up. The new web server instances, should start streaming logs promptly.

```
#!/bin/bash
```

```
yum -y install https://download.elastic.co/beats/filebeat/filebeat-1.3.1-x86_64.rpm
```

```
yum -y install nginx
```

```
cat << EOF > /etc/filebeat/filebeat.yml
```

```
filebeat:
```

```
  prospectors:
```

```
    -
```

```
      paths:
```

- /var/log/*.log
- /var/log/messages
- /var/log/secure

```
    -
```

```
      paths:
```

- /var/log/nginx/access.log

```
      document_type: nginx-access
```

```
  registry_file: /var/lib/filebeat/registry
```

```
output:
```

```
  logstash:
```

```
    hosts: ["10.0.1.132:5044"]
```

```
EOF
```

```
service nginx start
```

```
service filebeat start
```

Visualizing logs with Kibana

We have our logs collected by Filebeat and stored in Elasticsearch, how about browsing them?

Kibana, right on time:

Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. You use Kibana to search, view, and interact with data stored in Elasticsearch indices. You can easily perform advanced data analysis and visualize your data in a variety of charts, tables, and maps.

--<https://www.elastic.co/guide/en/kibana/current/introduction.html>

Install the package:

Copy

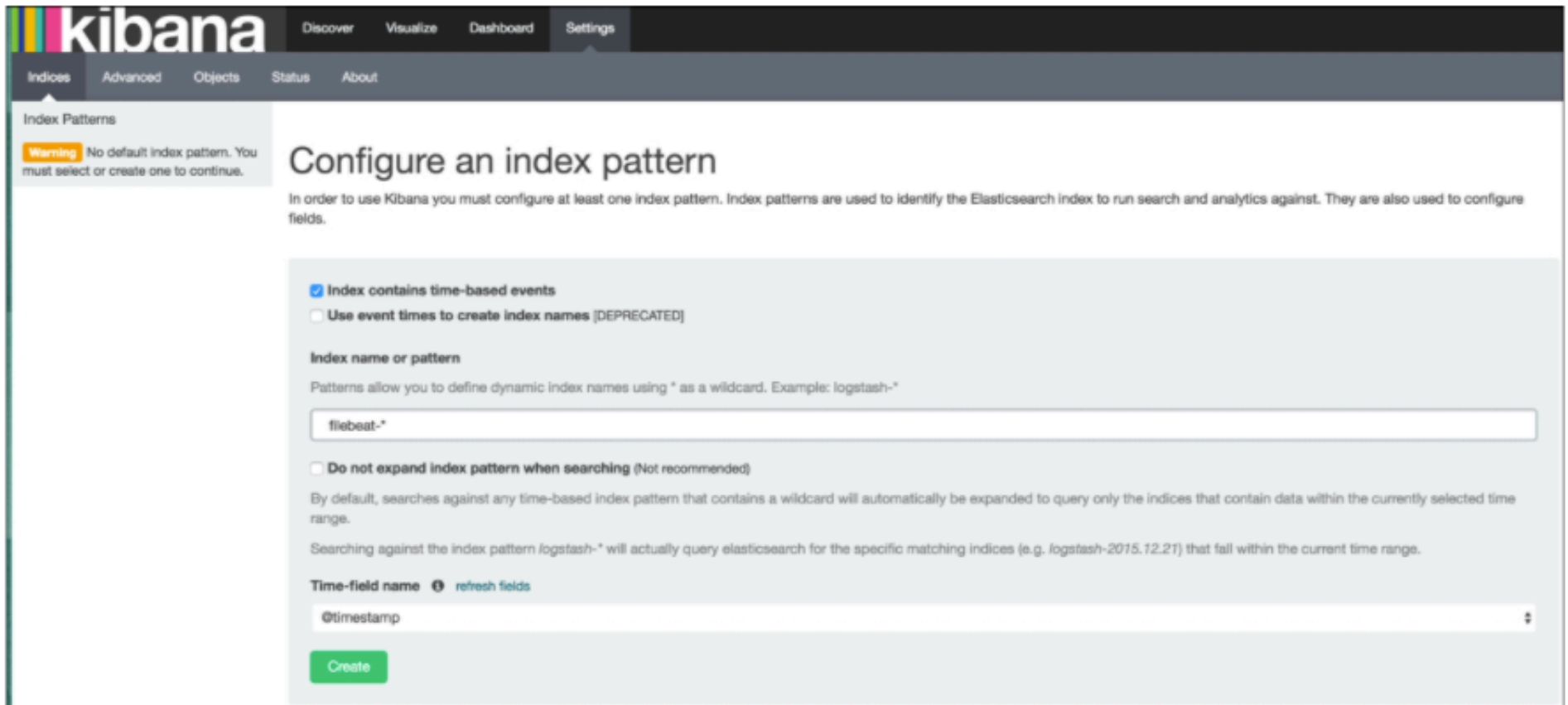
```
# yum -y install https://download.elastic.co/kibana/kibana/kibana-4.6.1-x86_64.rpm
```

Start the service:

Copy

```
# service kibana start
```

The default port is **TCP:5601** , if allowed in the relevant security group, you should be able to see the Kibana dashboard:



kibana Discover Visualize Dashboard **Settings**

Indices Advanced Objects Status About

Index Patterns

Warning No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☒ Index contains time-based events

☐ Use event times to create index names [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

filebeat-*

☐ Do not expand index pattern when searching (Not recommended)

By default, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.

Searching against the index pattern *logstash-** will actually query elasticsearch for the specific matching indices (e.g. *logstash-2015.12.21*) that fall within the current time range.

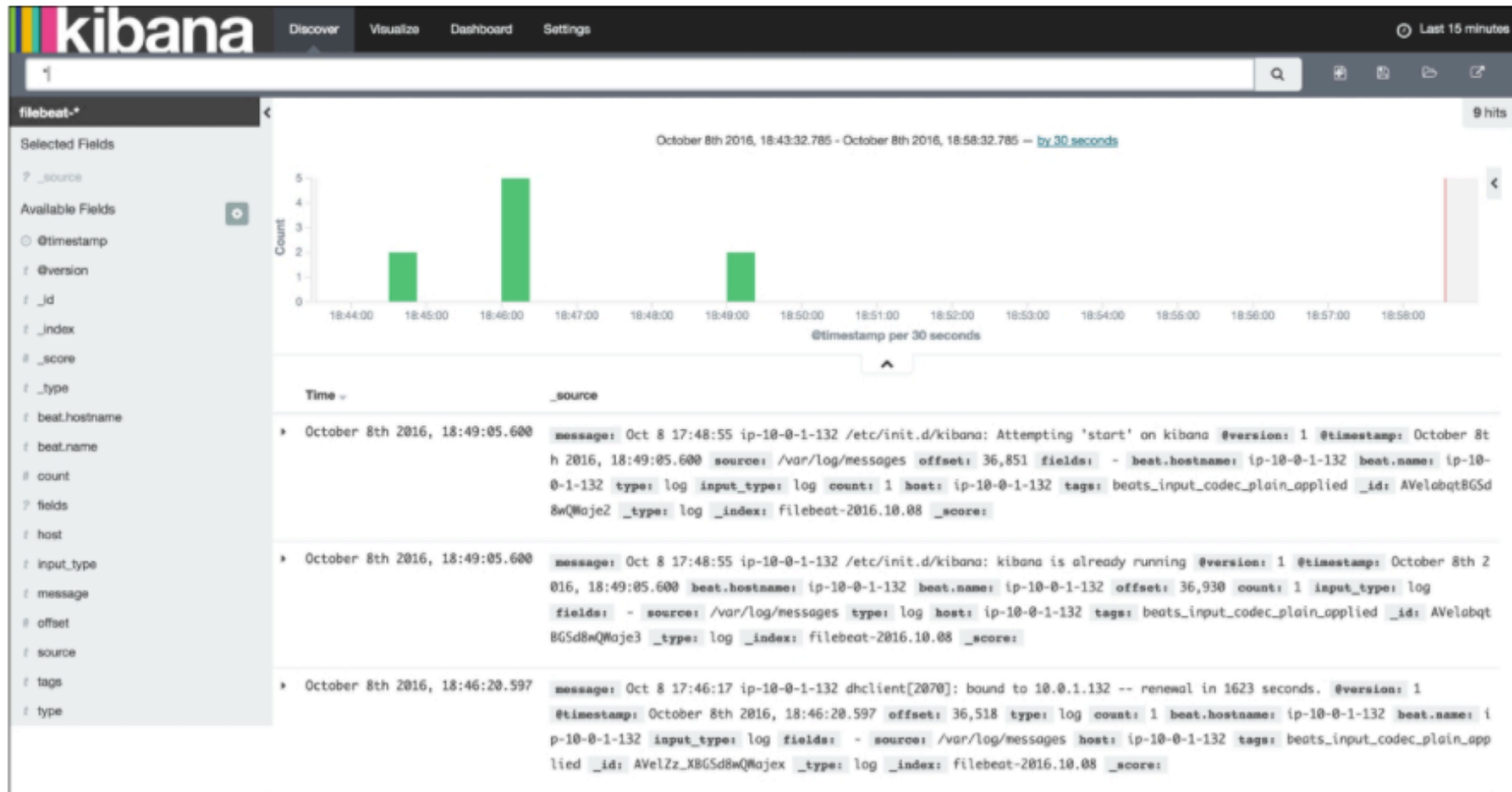
Time-field name ⓘ [refresh fields](#)

@timestamp

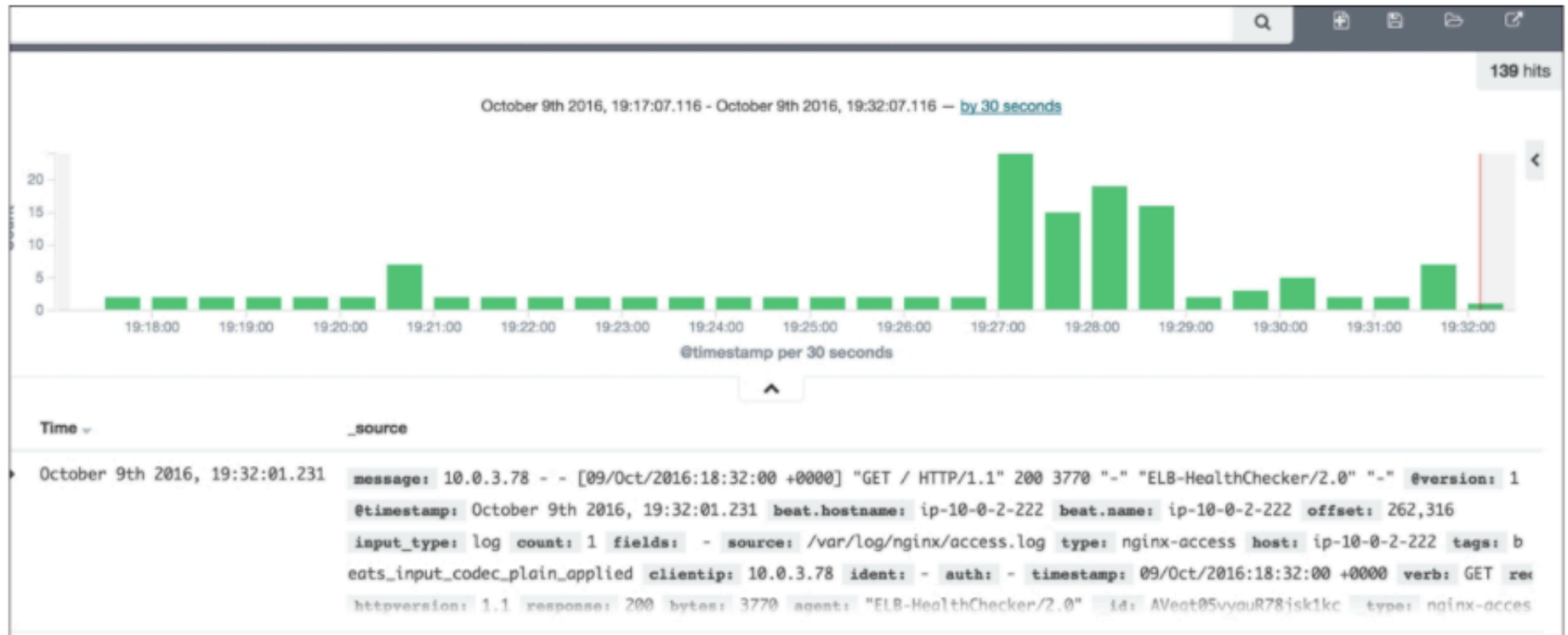
Create

Set the **index pattern** to **filebeat-*** and click **Create**.

Kibana is now ready to display our Filebeat data. Switch to the **Discover** tab to see the list of recent events:



In addition to the standard **Syslog** messages, you will also notice some **NGINX access-log** entries, with various fields populated as per the filter we specified earlier:



Logs: done. Now, how about some metrics?