Sjoerd Smink · Follow
Jul 20, 2019 · 5 min read

Trying out Open Distro for Elasticsearch with Logstash

Elasticsearch, Kibana, Logstash - maybe one of these names ring a bell. These products from the company Elastic are used a lot to search through logging data, files, or other big data. Searching through terabytes of data can be done in milliseconds, while also showing results in beautiful dashboards. And of course it can do much more.

Image source: another Medium article explaining the Elastic modules

Elastic offers an Open Source and Basic subscription, however the modules for authentication, alerting and logstash/beats (for log or data collection) are only available in the paid plan. There is a SaaS option that do have this, but if you're bound to an on-premise installation, you have to pay up (exact pricing is only available on request).

In March 2019 AWS made an announcement that they started with Open Distro for Elasticsearch because *"some of the more recent additions to Elasticsearch are proprietary"*. It has an Apache 2.0 license and they mention that it is not simply a fork of Elastic: *"we will continue to send our contributions and patches upstream to advance these projects"*. But the biggest advantage is the availability of Security, Alerting and Performance Analysis.

Security, Alerting and Performance Analysis.

Now version 1.0 is released, it's time to give it a try! The best way to install it, is using Docker compose. It's quite well documented, although I personally ran into some problems when installing Logstash.

Start with creating a new Virtual Machine or AWS, Azure or Google Cloud. This time I used Azure with Ubuntu, which required the installation of Docker CE and Docker compose first. Elasticsearch needs a high mmap count, which must be set (on the host!) with

```
sudo sysctl -w vm.max_map_count=262144
```

Create a file docker-compose.yml, which details the Docker compose with Logstash configuration:

```
version: '3'
services:
  odfe-node1:
    image: amazon/opendistro-for-elasticsearch:1.11.0
    container_name: odfe-node1
    environment:
      - cluster.name=odfe-cluster
      - node.name=odfe-node1
      - discovery.seed_hosts=odfe-node1,odfe-node2
      - cluster.initial_master_nodes=odfe-node1,odfe-node2
      - bootstrap.memory_lock=true # along with the memlock settings
below, disables swapping
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m" # minimum and maximum Java
heap size, recommend setting both to 50% of system RAM
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536 # maximum number of open files for the
Elasticsearch user, set to at least 65536 on modern systems
        hard: 65536
    volumes:
      - odfe-data1:/usr/share/elasticsearch/data
```

```
      - odfe-net
  odfe-node2:
    image: amazon/opendistro-for-elasticsearch:1.11.0
    container_name: odfe-node2
    environment:
      - cluster.name=odfe-cluster
      - node.name=odfe-node2
      - discovery.seed_hosts=odfe-node1,odfe-node2
      - cluster.initial_master_nodes=odfe-node1,odfe-node2
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    volumes:
      - odfe-data2:/usr/share/elasticsearch/data
      #- ./custom-
elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml
    networks:
      - odfe-net
  kibana:
    image: amazon/opendistro-for-elasticsearch-kibana:1.11.0
    container_name: odfe-kibana
    ports:
      - 5601:5601
    expose:
      - "5601"
    environment:
      ELASTICSEARCH_URL: https://odfe-node1:9200
      ELASTICSEARCH_HOSTS: https://odfe-node1:9200
    networks:
      - odfe-net

logstash:
    image: docker.elastic.co/logstash/logstash-oss:7.9.1
    container_name: logstash
    volumes:
    - ./logstash.conf:/usr/share/logstash/pipeline/logstash.conf
    ports:
      - "5011:5011"
      - "5012:5012"
      - "5013:5013"
    networks:
      - odfe-net
    depends_on:
      - odfe-node1

volumes:
```

This follows to a great extend the documentation, with the addition of Logstash. Keep in mind that you need the same Logstash version as Elasticsearch and Kibana (in this case 7.9.1); you can find the correct version on the website. OSS is the Open Source version of Elastic.

You also need to create a file called logstash.conf, which contains the following:

```
input {
  http {
    port => 5011
    codec => "line"
  }
  udp {
    port => 5012
    codec => "json"
  }
  tcp {
    port => 5013
    codec => "json_lines"
  }
}
output {
  elasticsearch {
    hosts => ["https://odfe-node1:9200"]
    ssl => true
    ssl_certificate_verification => false
    user => logstash
    password => logstash
    ilm_enabled => false
    index => "logstash"
  }
  stdout {
  }
}
```

In the above logstash.conf file are some particular things that were not well documented. The host needs to direct to the Elasticsearch container, using an https connection. Authentication with Elasticsearch is required, and for that there is a default user logstash/logstash (keep this in mind for improving security later!). Setting ilm_enabled to false is because Logstash version 7.0.1 has a bug that is uses the proprietary X-Pack, which causes issues when starting up. And setting the index to

For debugging purposes we're printing Logstash messages to stdout. And we have three input options, which we'll use for testing the pipeline.

We're now ready to give it a try. Start the container using

```
sudo docker-compose up
```

You can later stop it using Ctrl-C and

```
sudo docker-compose down -v
```

The -v is only required when you make changes to the Docker compose script. This will also erase all data and settings.

After Elasticsearch, Kibana and Logstash have started, you should be able to go to http://your-vm-ip:5601. Of course port 5601 should be made accessible for that in AWS / Azure / Google Cloud. Login with *admin/admin*. You can add some sample data to Elasticsearch, to visualise it in Kibana. But it's more fun to add data ourselves.

If ports 5011, 5012 and 5013 are open to the outside world, the following commands can be executed from anywhere. If you don't want to open these ports, you can also execute it from your VM as localhost. To start with testing the HTTP input, use

```
wget -S --post-data 'test message' http://your-vm-ip:5011/
```

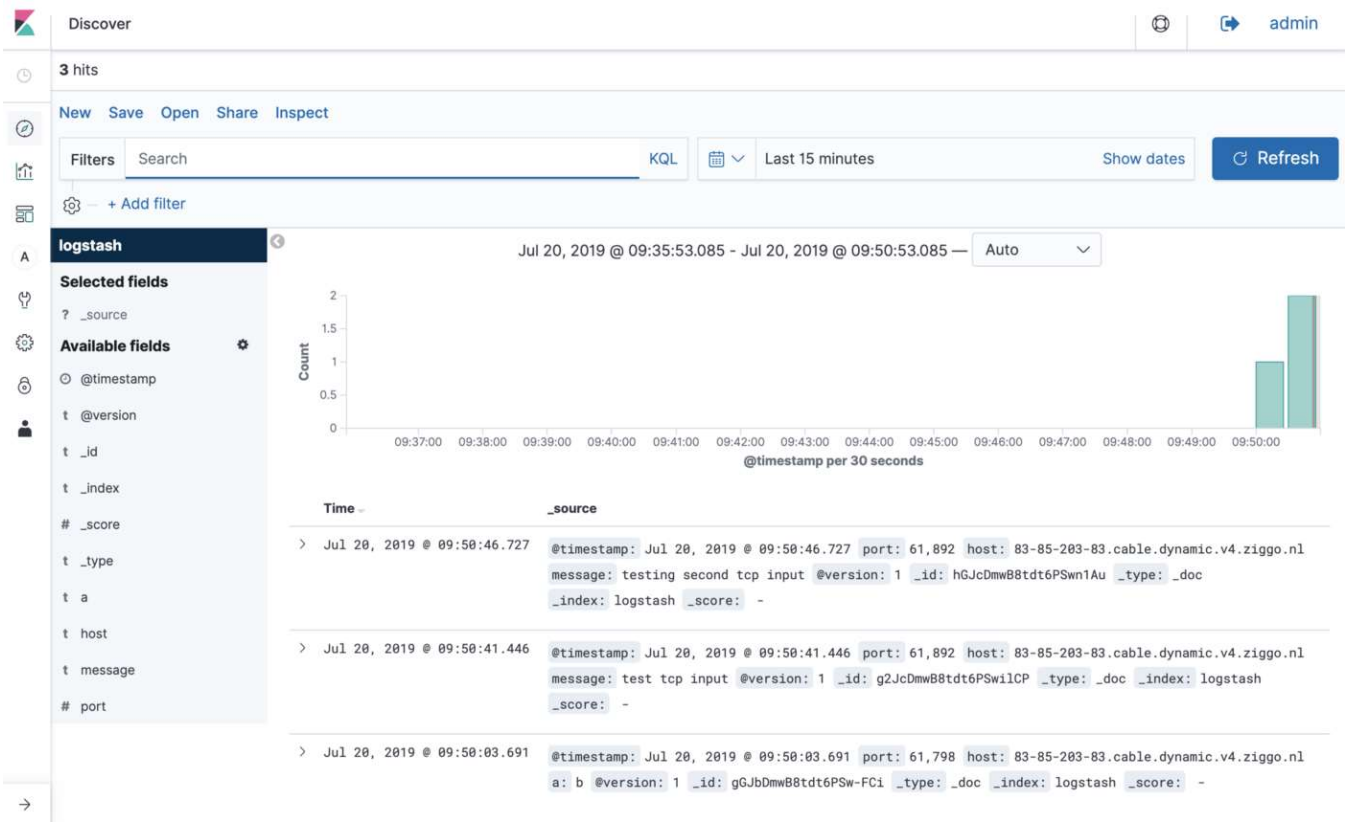
This should add your first log line in Elasticsearch! To use UDP, you can

```
telnet your-vm-ip 5011
{"a":"b"}
```



```
netcat your-vm-ip 5011
{"message":"test tcp input"}
{"message":"testing second tcp input"}
```

You should now be able to see the test data in Kibana. If not, take a look at the terminal where logstash is outputting debug data.



It should be said that this configuration is mainly meant for testing purposes. If you want to use it for production, you should take a look at the security documentation.

That's it! Enjoy your fully automated data pipeline.