



Elastalert: implementing rich monitoring with Elasticsearch



Hi, dear readers! Welcome to my blog. On this post, we will take a tour on a open source project developed by Yelp, called Elastalert. Focused on enriching Elasticsearch's role as a monitoring tool, it allow us to query Elasticsearch, sending alerts to different types of tools, such as e-mail boxes, Telegram chats, JIRA issues and more. So, without further delay, let's go deep on the tool!

Set up

In order to set up Elastalert, we need to clone the project's Git repository and install it with Python. If the reader doesn't have Python or Git installed, I recommend following the instructions [here](https://www.python.org/downloads/) (<https://www.python.org/downloads/>) for Python and [here](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git) (<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>) for Git. For this tutorial, I am using a Unix OS, but the instructions are similar for other environments such as Linux. Also, on this tutorial I am using virtualenv, in order to keep my Python interpreter "clean". The reader can find instructions to install virtualenv [here](https://virtualenv.pypa.io/en/latest/installation.html) (<https://virtualenv.pypa.io/en/latest/installation.html>).

To display the alerts, we will use a Telegram channel, which will receive alerts sent by a Telegram bot. In order to prepare the bot, we need a Telegram account and use the Bot Father (@BotFather) to create the bot, then create a public channel on telegram and associate the bot on the channel's admins. The instructions to make this configurations can be found [here](https://core.telegram.org/bots) (<https://core.telegram.org/bots>). In order to easy the steps for the reader, I leave the bot created for this lab (@elastalerthandson) published for anyone who wants to use this bot on his own telegram channels for testing!

With all the tools installed and ready, let's begin by cloning the Elastalert Git repository. To do this, we run the following command, on the folder of our choice:

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

After running the command, we will see that a folder called "elastalert" was created. Before we proceed, we will also create a virtualenv environment, where we will install Elastalert. We do this by running:

```
virtualenv virtualenvelastalert
```

After creating the virtual environment – which will create a folder called “virtualenvelastalert” -, we need to activate it before we proceed with the install. To do this, we run the following command, assuming the reader is on the same folder of the previous command:

```
source virtualenvelastalert/bin/activate
```

After activating, we will notice that the name of our virtual environment is now written as a prefix on the shell, meaning that it is activated. Now, to install elastalert, we navigate to the folder created previously by our git clone command and type the following:

```
python setup.py install  
sudo pip install -r requirements.txt
```

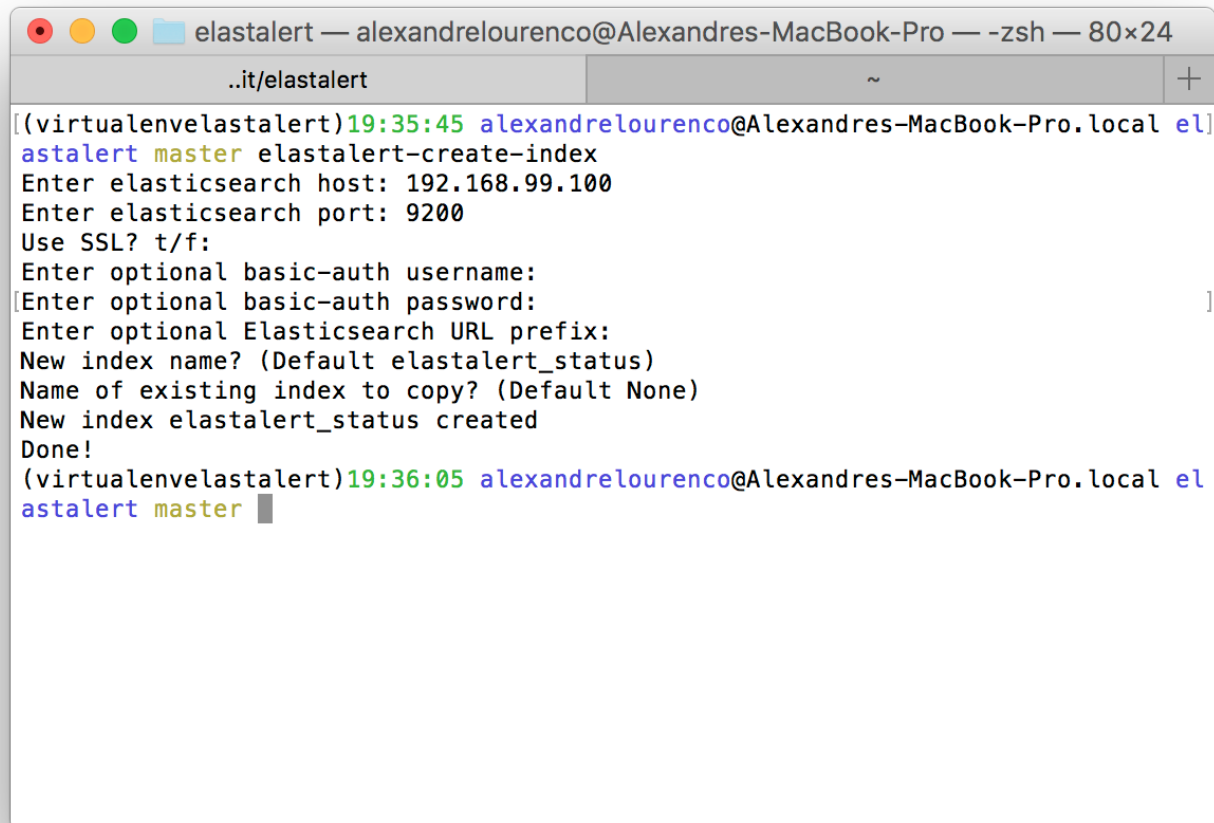
That’s it! Now that we have Elastalert installed, let’s continue the setup by creating the Elasticsearch index that it will be used as a metadata repository by Elastalert.

Creating the metadata index

In order to create Elastalert’s index, we run the command:

```
elastalert-create-index
```

The command-line tool will ask us some settings such as the name we want for the index and the ip/port of our Elasticsearch’s cluster. After providing the settings, the tool will create the index, like we can see on the picture bellow:



```
elastalert — alexandrelourenco@Alexandres-MacBook-Pro — -zsh — 80x24
..it/elastalert
~
+
[(virtualenv)19:35:45 alexandrelourenco@Alexandres-MacBook-Pro.local el]
astalert master elastalert-create-index
Enter elasticsearch host: 192.168.99.100
Enter elasticsearch port: 9200
Use SSL? t/f:
Enter optional basic-auth username:
[Enter optional basic-auth password: ]
Enter optional Elasticsearch URL prefix:
New index name? (Default elastalert_status)
Name of existing index to copy? (Default None)
New index elastalert_status created
Done!
(virtualenv)19:36:05 alexandrelourenco@Alexandres-MacBook-Pro.local el
astalert master
```

<https://dl.dropboxusercontent.com/s/778dvupvr2ushnd/elastalert1.png?dl=0> Creating the configuration files

All the configuration on Elastalert is made by YAML files. The main configuration file for the tool is called by default as “config.yaml” and is located on the same folder where we start Elastalert – which we will do in some moments. For our main configuration file, let’s create a file called “config.yaml” like the following:

```
rules_folder: rules_folder

run_every:
  seconds: 40

buffer_time:
  minutes: 15

es_host: 192.168.99.100

es_port: 9200

writeback_index: elastalert_status

alert_time_limit:
  days: 2
```

On the config above, we defined:

- The `rules_folder` property which defines the folder where our rules will be (all YAML files on the folder will be processed);
- The `run_every` property will make Elastalert to run all the rules on a 40 seconds frequency;
- The `buffer_time` property will make Elastalert cache the last period of time defined by the range of the property. This approach is used when the queries made on Elasticsearch are not on real time data;
- The host ip of the Elasticsearch's node used to query the alerts;
- The host port of the Elasticsearch's node used to query the alerts;
- The index used to store the metadata, that we created on the previous section;
- The maximum period of time Elastalert will hold a alert that the delivery has failed, making retries during the period;

Now, let's create the "rules_folder" folder and create 3 YAML files, which will hold our rules:

- `twitter_flatline.yaml`
- `twitter_frequency.yaml`
- `twitter-blacklist.yaml`

On this rules, we will test 3 types of rules Elastalert can manage:

- The flatline rule, which will alert when the number of documents find for a search drop bellow a threshold;
- The frequency rule, which will alert when a number of documents for a certain period of time is reached;
- The blacklist rule, which will alert when any document containing a list of words is found on the timeframe collected by the tool;

Of course, there's other rule types alongside those that we will cover on this lab, like the spike rule that can detect abnormal grows or shrinks on data across a time period, or the whitelist rule, which alert on any documents that contain any words from a list. More information about rules and their types can be found at the references on the end of this post.

For this lab, we will use a elasticsearch index with twitter data. The reader can found more information about how to set up a ELK environment on [my ELK series \(https://alexandreel.com/2015/01/26/elk-using-a-centralized-logging-architecture-part-1/\)](https://alexandreel.com/2015/01/26/elk-using-a-centralized-logging-architecture-part-1/). The Logstash configuration file used on this lab is as follows:

```
input {
  twitter {
    consumer_key => "XXXXXXXXXXXXXXXXXXXX"
    consumer_secret => "XXXXXXXXXXXXXXXXXXXX"
    keywords => ["coca cola", "java", "elasticsearch", "amazon"]
    oauth_token => "XXXXXXXXXXXXXXXXXXXX"
    oauth_token_secret => "XXXXXXXXXXXXXXXXXXXX"
  }
}

output {
  stdout { codec => rubydebug }
  elasticsearch {
    hosts => [ "192.168.99.100:9200" ]
    index => "twitter-%{+YYYY.MM.dd}"
  }
}
```

With our ELK stack set up and running, let's begin creating the rules. First, we create the frequency rule, by configuring the respective YAML file with the following code:

```
name: Twitter frequency rule

type: frequency

index: twitter-*

num_events: 3

timeframe:
  minutes: 15

realert:
  hours: 2

filter:
- query:
  query_string:
    query: "message:amazon"

alert:
- "telegram"

telegram_bot_token: 184186982:AAGpJRyWQ2Rb_RcFXncGrJrBrSK7BzoVFU8

telegram_room_id: "@elastalerthandson"
```

On the following file we configure a frequency rule. The rule is configured by setting the following properties:

- name: This property defines the rule's name. This property acts as the rule ID;
- type: This property defines the type of rule we are creating;
- index: This property defines the index on Elasticsearch where we want to make the searches;
- num_events: The number of documents necessary to be found in order to fire the alert;
- timeframe: The time period which will be queried to check the rule;
- realert: This property defines the time period that Elastalert will stop realerting the rule after the first match, preventing the users to be flooded with alerts;
- filter: This property is where we configure the query that will be send to Elasticsearch in order to check the rule;
- alert: This property is a list of targets which we want our alerts to be send. On our case, we just defined the telegram target;
- telegram_bot_token: On this property we set the access token from our bot, as received by the Botfather;
- telegram_room_id: On this property we define the id of the channel we want the alerts to be sent;

As we can see, is a very straightforward and simple configuration file. For the flatline config, we configure our respective YAML as follows:

```
name: Twitter flatline rule

type: flatline

index: twitter-*

threshold: 30

timeframe:
  minutes: 5

realert:
  minutes: 30

use_count_query: true

doc_type: logs

alert:
- "telegram"

telegram_bot_token: 184186982:AAGpJRyWQ2Rb_RcFXncGrJrBrSK7BzoVFU8

telegram_room_id: "@elastalerthandson"
```

The configuration is pretty much the same of the previous file, with the exception of 3 new properties:

- `threshold`: This property defines the minimum amount of documents expected for the rule to receive in order that a alert is **not** needed to be sent;
- `use_count_query`: This property defines that Elastalert must use the count API from Elasticsearch. This API returns just the number of documents for the rule to be validated, eliminating the need to process the query data;
- `doc_type`: This property is needed by the count API aforementioned, in order to query the document count for a specific document type;

Finally, let's configure our final rule, coding the final YAML as follows:

```
name: Twitter blacklist rule

type: blacklist

index: twitter-*

compare_key: message

blacklist:
- "android"
- "java"

realert:
  hours: 4

filter:
- query:
    query_string:
      query: "*"

alert:
- "telegram"

telegram_bot_token: 184186982:AAGpJRyWQ2Rb_RcFXncGrJrBrSK7BzoVFU8

telegram_room_id: "@elastalerthandson"
```

On this file, the new properties that we needed to configure are:

- `compare_key`: This property defines the field on the documents that Elastalert will check the blacklist;
- `blacklist`: This property is a list of words which Elastalert will compare against the documents in order to check if any document has a blacklisted word;

And that concludes our configuration. Now, let's run Elastalert!

Running Elastalert

To run Elastalert, all we need to do is run a command like this, on the same folder of our YAML structure – where “config.yaml” is located:

```
elastalert --start NOW --verbose
```

On the command above, we set the flag “--start” to define that we want Elastalert to start the measurements from now up and the “--verbose” flag to print all the info log messages.

The simplest of the rules to test it out is the flatline rule. All we have to do is wait for about 5 minutes with Elasticsearch running and Logstash stopped – so no documents are streaming. After the wait, we can see on our channel that a alert is received on the channel:



Elasticalert hands-on

2 members



April 14

Channel created

April 15

Elasticalert hands-on

⚠ **Twitter flatline rule** ⚠ **Twitter flatline rule**

An abnormally low number of events occurred around 2016-04-15 18:49 BRT.
Between 2016-04-15 18:44 BRT and 2016-04-15 18:49 BRT, there were less than 30 events.

@timestamp:

2016-04-15T21:49:21.248082Z

count: 0

key: all

👁 2 18:58



Broadcast



<https://dl.dropboxusercontent.com/s/ds8crl7o419rylx/telegram1.png?dl=0>

And, as the time passes, we will receive other alerts as well, like the frequency alert:



Elasticalert hands-on



2 members



key: all

 2 18:58

Elasticalert hands-on

 **Twitter frequency rule**  Twitter frequency rule

At least 3 events occurred
between 2016-04-15 18:45 BRT
and 2016-04-15 19:00 BRT

@timestamp:

2016-04-15T22:00:56Z

@version: 1

_id: AVQb8Pa3rJVcBFur_oX8

_index: twitter-2016.04.15

_type: logs

client: <a href="https://

twitter.com/yuki_nagase"

rel="nofollow">yuki_nagase

hashtags: [

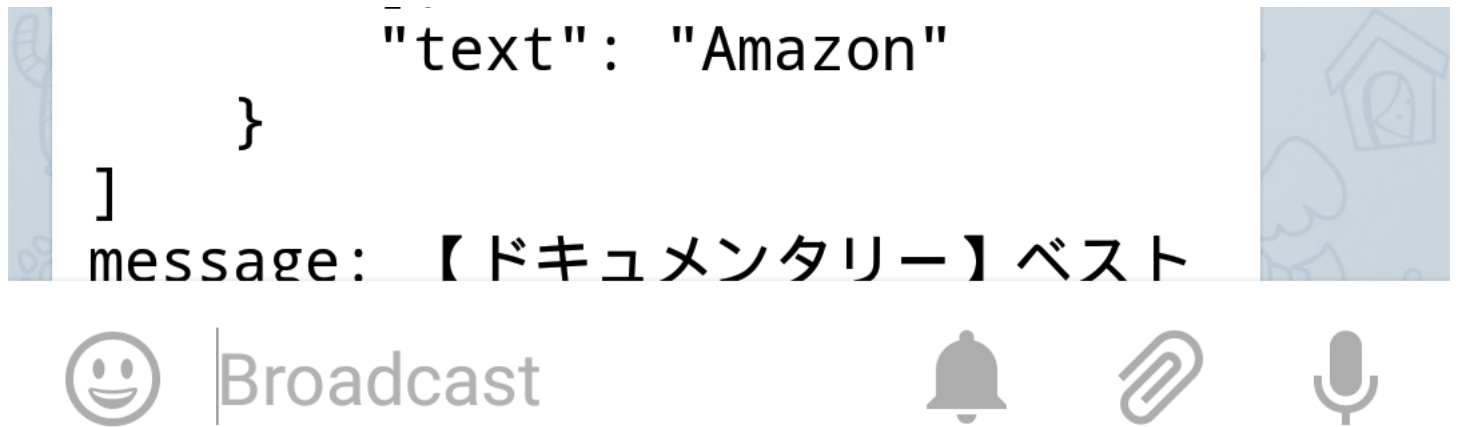
{

"indices": [

107,

114

],



(<https://dl.dropboxusercontent.com/s/tvud9yie74a71a4/telegram2.png?dl=0>)

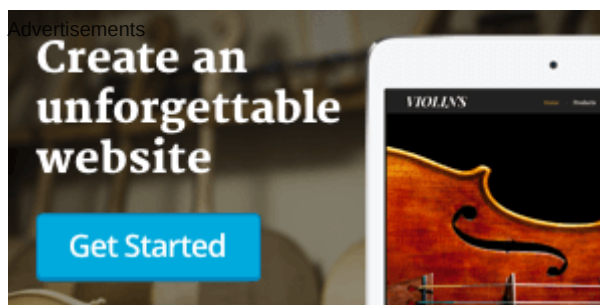
Conclusion

And so we conclude our tutorial about Elastalert. With a simple usage, we can see that we can construct really powerful alerts for our Elasticsearch system, enforcing the rule of the search engine on a monitoring ecosystem. Thank you for following me on another post, until next time.

Elastalert Git repository (<https://github.com/Yelp/elastalert>)

Elastalert documentation (<https://elastalert.readthedocs.org/en/latest/>)

Lab's source-code (<https://github.com/alexandreosl/elastalerthandson.git>)



15 DE APRIL DE 2016

ALERT, ELASTICSEARCH, YELP

22 COMMENTS

22 thoughts on “Elastalert: implementing rich monitoring with Elasticsearch”

E

Thanks for the blog. Easy to understand, and helped me a good deal.

28 DE APRIL DE 2016 AT 21:25

REPLY

ALEXANDREESL

Thank you!

18 DE JULY DE 2016 AT 00:35

REPLY

PRNASH

i tried to install elastalert but got the following error

```
blis/_blis.c: In function 'init_blist_types1':
blis/_blis.c:7376: error: expected expression before ')' token
blis/_blis.c:7378: error: 'PyBList_Type' undeclared (first use in this function)
blis/_blis.c:7378: error: 'PyType_Type' undeclared (first use in this function)
blis/_blis.c:7379: error: 'PyRootBList_Type' undeclared (first use in this function)
blis/_blis.c:7380: error: 'PyBListIter_Type' undeclared (first use in this function)
blis/_blis.c:7381: error: 'PyBListReverseIter_Type' undeclared (first use in this function)
blis/_blis.c: In function 'init_blist_types2':
blis/_blis.c:7394: warning: implicit declaration of function 'PyType_Ready'
blis/_blis.c:7394: error: 'PyRootBList_Type' undeclared (first use in this function)
blis/_blis.c:7395: error: 'PyBList_Type' undeclared (first use in this function)
blis/_blis.c:7396: error: 'PyBListIter_Type' undeclared (first use in this function)
blis/_blis.c:7397: error: 'PyBListReverseIter_Type' undeclared (first use in this function)
blis/_blis.c: At top level:
blis/_blis.c:7404: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'init_blist'
error: Setup script exited with error: command 'gcc' failed with exit status 1
```

11 DE JULY DE 2016 AT 07:39

REPLY

ALEXANDREESL

Hi, Prnash, everything fine with you? Please, provide some more information about your problem, which is your Python version, OS etc

thank you

18 DE JULY DE 2016 AT 00:28

REPLY

ELMAR

Hi,

You mentioned the following: "In order to prepare the bot, we need a Telegram account and use the Bot Father (@BotFather) to create the bot, then create a public channel on telegram and associate the bot on the channel's admins."

However, it is not entirely clear how to do this and what you mean.

I have created a Telegram account. Using @FatherBot created a another bot and received the API token.

In the rule yaml I have configured the following:

```
telegram_bot_token: tokenwithoutquotes
telegram_room_id: "@my_telegram_user_id"
```

I expected to receive alerts on @my_telegram_user_id

Is this correct? Or what is wrong?

Thanks

Elmar

29 DE JULY DE 2016 AT 05:48

REPLY

ALEXANDREESL

Hi, Elmar! Elmar, after you create the bot using the @FatherBot, on the telegram_room_id, is not your user, you need to create a channel on telegram, associate your bot to the channel with admin rights, and then use the channel's ID on the telegram_room_id, this way the alerts will be pushed to the channel, ok?

Thank you

13 DE AUGUST DE 2016 AT 15:15

REPLY

ZHOU RU DONG

how to use mail send elters?

17 DE AUGUST DE 2016 AT 07:39

REPLY

ALEXANDREESL

Hi, Zhou! I think you meant how to send e-mails as alerts, right? All you have to do is to configure the YAML to point to a SMTP server, as you can see bellow:

<https://elastalert.readthedocs.io/en/latest/elastalert.html#alerts>

Thank you

20 DE AUGUST DE 2016 AT 19:22

REPLY

ELASTAUSER

I got the following error while running the python setup.py install

blist/_blist.c:38:20: fatal error: Python.h: No such file or directory

I solved it by running the following command

yum -y install python-devel

I spent a bit finding so I hope this helps someone.

1 DE SEPTEMBER DE 2016 AT 21:05

REPLY

ALEXANDREESL

Thank you for your help!

4 DE SEPTEMBER DE 2016 AT 00:38

REPLY

MEIA

i got error,

[root@ELK-5 elastalert]# elastalert-create-index

Enter Elasticsearch host: localhost

Enter Elasticsearch port: 9200

Use SSL? t/f:

Enter optional basic-auth username (or leave blank):

Enter optional basic-auth password (or leave blank):

Enter optional Elasticsearch URL prefix (prepends a string to the URL of every request):

New index name? (Default elastalert_status)

Name of existing index to copy? (Default None)

Traceback (most recent call last):

File "/root/elastalert/virtualenv/elastalert/bin/elastalert-create-index", line 11, in

load_entry_point('elastalert==0.1.6', 'console_scripts', 'elastalert-create-index')()

File "/root/elastalert/virtualenv/elastalert/lib/python2.7/site-packages/elastalert-0.1.6-py2.7.egg/elastalert/create_index.py", line 120, in main

if es_index.exists(index):

File "build/bdist.linux-x86_64/egg/elasticsearch/client/utils.py", line 69, in _wrapped

File "build/bdist.linux-x86_64/egg/elasticsearch/client/indices.py", line 225, in exists

File "build/bdist.linux-x86_64/egg/elasticsearch/transport.py", line 327, in perform_request

File "build/bdist.linux-x86_64/egg/elasticsearch/connection/http_requests.py", line 79, in perform_request

elasticsearch.exceptions.ConnectionError: ConnectionError(HTTPConnectionPool(host='localhost', port=9200): Max retries exceeded with url: /elastalert_status (Caused by NewConnectionError(': Failed to establish a new connection: [Errno 111] Connection refused',))) caused by:

ConnectionError(HTTPConnectionPool(host='localhost', port=9200): Max retries exceeded with url: /elastalert_status (Caused by NewConnectionError(': Failed to establish a new connection: [Errno 111] Connection refused',)))

2 DE FEBRUARY DE 2017 AT 17:06

REPLY

ALEXANDREESL

Hi mea! From the error, it appears your Elasticsearch is not running or it is not running on localhost:9200. Could you please show your Elasticsearch configuration and check if it is running properly before the command is executed?

Thank you

3 DE MARCH DE 2017 AT 18:39

REPLY

JANOO BHANDARI

while running "elastalert-create-index" command, Iam getting the following error.

root@practice1:/home/ubuntu/elastalert# elastalert-create-index

Traceback (most recent call last):

File "/usr/local/bin/elastalert-create-index", line 5, in

from pkg_resources import load_entry_point

File "/usr/lib/python2.7/dist-packages/pkg_resources.py", line 2749, in

working_set = WorkingSet._build_master()

File "/usr/lib/python2.7/dist-packages/pkg_resources.py", line 446, in _build_master

return cls._build_from_requirements(__requires__)

File "/usr/lib/python2.7/dist-packages/pkg_resources.py", line 459, in _build_from_requirements

dists = ws.resolve(reqs, Environment())

File "/usr/lib/python2.7/dist-packages/pkg_resources.py", line 628, in resolve

raise DistributionNotFound(req)

pkg_resources.DistributionNotFound: elasticsearch<3.0.0

Could you please help?

14 DE FEBRUARY DE 2017 AT 11:23

REPLY**ALEXANDREESL**

Hi, janoo! Please, could you provide more info on your problem? Did you run the pip install command previously? What it was the output?

Thanks

3 DE MARCH DE 2017 AT 18:38**REPLY****ZANU**

i have same problem, i run the pip like this pip install “elasticsearch>=5.0.0”.
can you help me?

19 DE MARCH DE 2017 AT 17:33**ALEXANDREESL**

Hi, Zanu! Of course I can, can you please send the output from your pip? Did you tried install with “pip install -r requirements.txt”?

thank you

19 DE MARCH DE 2017 AT 20:36**ZANU**

hi alexandreesl, ys i tried install with “pip install -r requirements.txt”.

here's the output:

```
(virtualenvvelastalert) [root@localhost elastalert]# sudo pip install -r requirements.txt
Collecting argparse==1.3.0 (from -r requirements.txt (line 1))
Using cached argparse-1.3.0-py2.py3-none-any.whl
Collecting aws-requests-auth==0.2.5 (from -r requirements.txt (line 2))
Using cached aws-requests-auth-0.2.5.tar.gz
Complete output from command python setup.py egg_info:
/usr/lib64/python2.7/distutils/dist.py:267: UserWarning: Unknown distribution option:
'install_requires'
warnings.warn(msg)
usage: -c [global_opts] cmd1 [cmd1_opts] [cmd2 [cmd2_opts] ...]
or: -c -help [cmd1 cmd2 ...]
or: -c -help-commands
or: -c cmd -help
```

error: invalid command 'egg_info'

Command “python setup.py egg_info” failed with error code 1 in /tmp/pip-build-BYmYHM/aws-requests-auth/

20 DE MARCH DE 2017 AT 05:37**ALEXANDREESL**

I did some research and appears to me that you have a old pip, try upgrading with this command and try again please:

pip install –upgrade setuptools

Thank you

20 DE MARCH DE 2017 AT 10:21

ZANU

thank you alexandreessl it's work.

btw what do you do in config bot telegram? can you tell me to make bot telegram?

24 DE MARCH DE 2017 AT 15:10

ALEXANDREESL

Hi, Zanu! Have you tried the procedure on the documentation on this link:

<https://core.telegram.org/bots> ?

You can also use the bot I created just for testing your configuration, all you have to do is enter the channel “Elasticalert hands-on” on Telegram as a member and use the same YAML configuration I provided.

Thank you!

25 DE MARCH DE 2017 AT 11:45

ZANU

Thank you alexandreessl for your help,
its work 😊

20 DE APRIL DE 2017 AT 14:58

ALEXANDREESL

Great, Zanu! Thank you =)

20 DE APRIL DE 2017 AT 23:15