#!/usr/bin/env python

from time import sleep

from datetime import datetime

from random import randint

from elasticsearch import Elasticsearch

from elasticsearch\_watcher import WatcherClient

# initialize the standard client as usual

es = Elasticsearch(['elastic:changeme@bb16:9200'])

# add the .watcher namespace to it

WatcherClient.infect\_client(es)

# clear the index first

es.indices.delete(

index=['alerts', 'test'], ignore=404)

# get the watcher plugin version

print('Using watcher', es.watcher.info()['version']['number'])

# Register a new watch

es.watcher.put\_watch(

id='error\_500',

body={

# label the watch

'metadata': {'tags': ['python']},

# Run the watch every 1 minute

'trigger': { 'schedule': { 'interval': '1m' } },

# Search for at least 3 documents matching the condition

'condition': { 'script': { 'inline': 'ctx.payload.hits.total > 3' } },

# Throttle the watch execution for 30 seconds

'throttle\_period': '1m',

# The search request to execute

'input': {

'search': {

'request': {

'indices': ['test'],

'body': {

'query': {

'filtered': {

'query': { 'match': { 'severity': 'Error'} },

'filter': { 'range': { 'timestamp': { 'from': '{{ctx.trigger.scheduled\_time}}||-5m', 'to': '{{ctx.trigger.triggered\_time}}' } } }

}

},

# Return statistics about different hosts

'aggregations': {

'hosts': { 'terms': { 'field': 'host' } }

}

}}}},

# The actions to perform

'actions': {

'send\_email': {

'transform': {

# Transform the data for the template

'script': '''return [

total: ctx.payload.hits.total,

hosts: ctx.payload.aggregations.hosts.buckets.collect { [ host: it.key, errors: it.doc\_count ] },

errors: ctx.payload.hits.hits.collect { it.\_source }

];'''

},

'email': {

'to': 'ian.t.robertson.ctr@mail.mil',

'subject': '[ALERT] {{ctx.watch\_id}}',

'attach\_data': True,

'body': '''

Received {{ctx.payload.total}} error documents in the last 5 minutes.

Hosts:

{{#ctx.payload.hosts}}\* {{host}} ({{errors}})

{{/ctx.payload.hosts}}'''.replace('\n'+' '\*24, '\n').strip(),

}

},

'index\_payload': {

# Transform the data to be stored

'transform': { 'script': 'return [ watch\_id: ctx.watch\_id, payload: ctx.payload ]' },

'index': { 'index': 'alerts', 'doc\_type': 'alert' }

},

'ping\_webhook': {

'webhook': {

'method': 'POST',

'host' : 'bigbrother3.arl.army.mil',

'port': 9612,

'body': '{"watch\_id" : "{{ctx.watch\_id}}", "payload" : "{{ctx.payload}}"}'

}

}

}

}

)

# index documents to trigger the watch

for \_ in range(5):

es.index(

index='test',

doc\_type='d',

body={

'timestamp': datetime.utcnow(),

'status': 500,

'host': '10.0.0.%d' % randint(1, 3)

}

)

# wait a bit...

for \_ in range(30):

sleep(1)

print('.', sep='', end='', flush=True)

print()

# display information about watch execution

print('=' \* 80)

s = es.search(

index='.watch\_history\*',

q='watch\_id:error\_500',

sort='trigger\_event.schedule.triggered\_time:asc'

)

for hit in s['hits']['hits']:

print('%s: %s' % (hit['\_id'], hit['\_source']['state']))

# delete the watch

es.watcher.delete\_watch(id='error\_500', force=True)