文章转载自：https://mp.weixin.qq.com/s/W9b28CFBEmxBPz5bGd1-hw

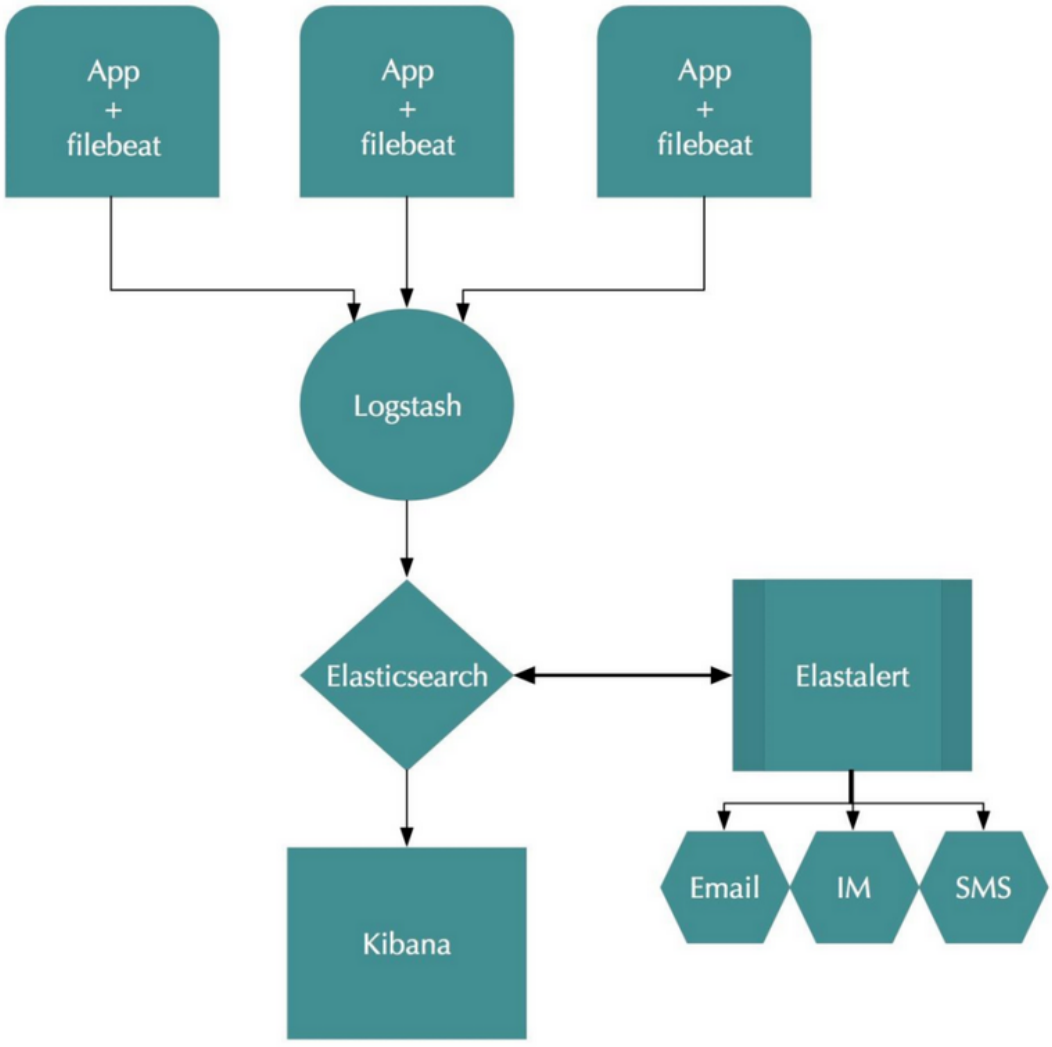
注意事项：
1.文章中凡是用python的均表示的是使用python3的版本，但是不能使用python3，因为安装的elastalert模块中好多文件开头用的是#!/usr/bin/env python，或者没有添加这个
也就是说假若系统使用双版本的python，python表示的是2版本，python3表示的是3版本，需要把python表示的2的版本链接给删除，换成python3的，最终结果是执行python出现的是3的版本信息。
2.关于elastalert_modules文件夹，并不是放在elastalert在python3的安装路径下，放在`~/elastalert`路径下了，然后进入到`~/elastalert`路径下，执行命令：python -m elastalert.elastalert --verbose --config /app/elastalert/config.yaml --rule /app/elastalert/example_rules/sms-applog.yaml

一、ElastAlert介绍

在日志管理上我们使用Elasticsearch，Logstash和Kibana技术栈来管理不断增长的数据和日志，但是对于错误日志的监控ELK架构并没有提供，所以我们需要使用到第三方工具ElastAlert，来帮助我们及时发现业务中存在的问题。

ElastAlert通过定期查询Elasticsearch，并将数据传递到规则类型，该规则类型确定何时找到匹配项。发生匹配时，将为该警报提供一个或多个警报，这些警报将根据匹配采取行动。

这是由一组规则配置的，每个规则定义一个查询，一个规则类型和一组警报。



ElastAlert支持以下方式报警

- Command （可调用短信接口）
- Email
- JIRA
- OpsGenie
- SNS
- HipChat
- Slack
- Telegram
- Debug
- Stomp

除了这种基本用法外，还有许多其他功能使警报更加有用：

- 警报链接到Kibana仪表板
- 任意字段的合计计数
- 将警报合并为定期报告
- 通过使用唯一键字段来分隔警报
- 拦截并增强比赛数据

二、部署ElastAlert

. 1. 部署所需环境

ELK 环境部署

EFK6.3+kafka+logstash日志分析平台集群

安装依赖包

```
yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-
devel readline-devel tk-devel gdbm-devel db4-devel libpcap-devel xz-devel
```

部署python3.6

```
mkdir -p /usr/local/python3
cd /usr/local/python3
wget https://www.python.org/ftp/python/3.6.9/Python-3.6.9.tgz
tar xf Python-3.6.9.tgz
cd Python-3.6.9
./configure --prefix=/usr/local/python3
make && make install
```

配置环境变量

```
# 将python2.7 软链删除，换成python3.6
rm /usr/bin/python
ln -s /usr/local/python3/bin/python3.6 /usr/bin/python
rm /usr/bin/pip
ln -s /usr/local/python3/bin/pip3 /usr/bin/pip

# 最终效果
[root@docker Python-3.6.9]# ll /usr/bin/python*
lrwxrwxrwx  1 root root   32 Sep  4 09:57 /usr/bin/python ->
/usr/local/python3/bin/python3.6
```

```
lrwxrwxrwx. 1 root root    9 Oct 10  2019 /usr/bin/python2 -> python2.7
-rwxr-xr-x. 1 root root 7216 Aug  7  2019 /usr/bin/python2.7
[root@docker Python-3.6.9]# ll /usr/bin/pip*
lrwxrwxrwx 1 root root  27 Sep  4 09:57 /usr/bin/pip -> /usr/local/python3/bin/pip3
-rwxr-xr-x 1 root root 284 Jan 29  2020 /usr/bin/pip2
-rwxr-xr-x 1 root root 288 Jan 29  2020 /usr/bin/pip2.7
```

验证版本

```
# python
Python 3.6.9 (default, Jun  2 2020, 12:12:43)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-18)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
# pip -V
pip 18.1 from /usr/local/python3/lib/python3.6/site-packages/pip (python 3.6)
```

. 2. 部署ElastAlert

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

安装模块：

```
cd elastalert
pip install "setuptools>=11.3"
python setup.py install
pip install -r requirements.txt
```

安装完后，会在 /usr/local/bin/ 下生成4个elastalert命令

```

$ ll /usr/local/bin/elastalert*
-rwxr-xr-x 1 root root 396 2月  14 10:03 /usr/local/bin/elastalert
-rwxr-xr-x 1 root root 422 2月  14 10:03 /usr/local/bin/elastalert-create-index
-rwxr-xr-x 1 root root 430 2月  14 10:03 /usr/local/bin/elastalert-rule-from-kibana
-rwxr-xr-x 1 root root 416 2月  14 10:03 /usr/local/bin/elastalert-test-rule
```

根据Elasticsearch的版本，您可能需要手动安装正确版本的elasticsearch-py。

Elasticsearch 5.0+：

```
pip install "elasticsearch>=5.0.0"
```

Elasticsearch 2.X：

```
pip install "elasticsearch<3.0.0"
```

. 3. 配置ElastAlert

配置config.yaml 文件

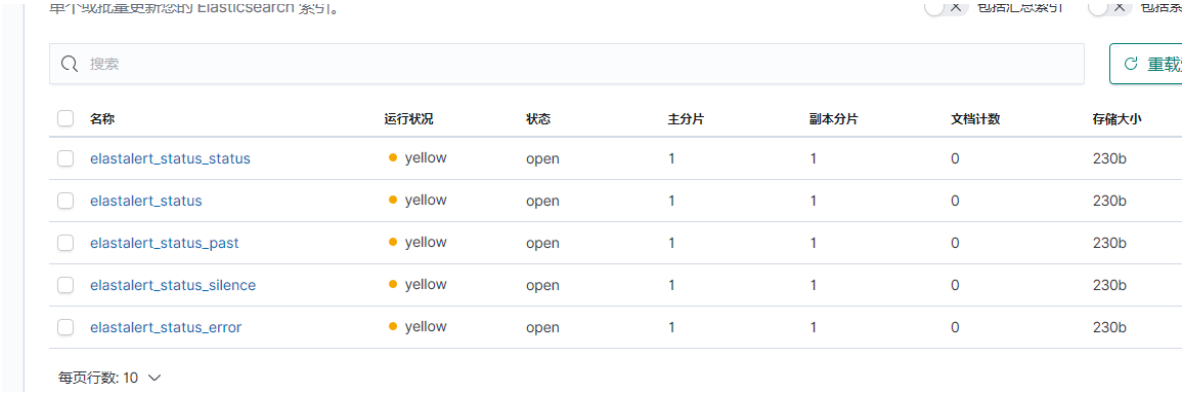
```
# cp config.yaml.example  config.yaml
# cat config.yaml
rules_folder: example_rules
run_every:
  seconds: 10
buffer_time:
  minutes: 15
es_host: 10.1.144.208
es_port: 9201
#es_username: elastic
#es_password: 123456
writeback_index: elastalert_status
alert_time_limit:
  days: 2
```

> `rules_folder`：ElastAlert从中加载规则配置文件的位置。它将尝试加载文件夹中的每个.yaml文件。没有任何有效规则，ElastAlert将无法启动。
> `run_every`：ElastAlert多久查询一次Elasticsearch的时间。
> `buffer_time`：查询窗口的大小，从运行每个查询的时间开始向后延伸。对于其中use_count_query或use_terms_query设置为true的规则，将忽略此值。
> `es_host`：是Elasticsearch群集的地址，ElastAlert将在其中存储有关其状态，查询运行，警报和错误的数据。
> `es_port`：es对应的端口。
> `es_username`：可选的; 用于连接的basic-auth用户名es_host。
> `es_password`：可选的; 用于连接的basic-auth密码es_host。
> `es_send_get_body_as`：可选的; 方法查询Elasticsearch - GET，POST或source。默认是GET
> `writeback_index`：ElastAlert将在其中存储数据的索引的名称。我们稍后将创建此索引。
> `alert_time_limit`：失败警报的重试窗口。

创建elastalert-create-index索引

```
# elastalert-create-index
New index name (Default elastalert_status)
Name of existing index to copy (Default None)
New index elastalert_status created
Done!
```

具体效果



三、使用微信报警

由于ElastAlert没有内置企业微信的报警方式，我们还需要使用一个开源插件elastalert-wechat-plugin来实现微信的报警，Github项目地址

. 1. 下载项目文件

```
mkdir -p ~/elastalert/elastalert_modules
# 下载该文件的地址有问题，导致无法下载，因此这一步就不下载了，直接使用下一步修改好的内容复制粘贴创建这个文件即可
# wget -P ~/elastalert/elastalert_modules/ wget https://raw.githubusercontent.com/anjia0532/elastalert-wechat-plugin/master/elastalert_modules/wechat_qiye_alert.py
touch ~/elastalert/elastalert_modules/__init__.py
```

. 2. 修改插件源码

由于这个插件是基于python2.x版本开发的，而ElastAlert的最新版本使用的是python3.6版本开发，所以需要改一些代码，以便正常运行，另外还添添加了转中文字符功能。
wechat_qiye_alert.py修改后如下：

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import json
import datetime
from elastalert.alerts import Alerter, BasicMatchString
from requests.exceptions import RequestException
from elastalert.util import elastalert_logger,EAException #[感谢minminmsn分享](https://github.com/anjia0532/elastalert-wechat-plugin/issues/2#issuecomment-311014492)
import requests
from elastalert_modules.MyEncoder import MyEncoder

'''
#####################################################################
# 微信企业号推送消息                                                  #
#                                                                   #
# 作者: AnJia <anjia0532@gmail.com>                                  #
# 作者博客: https://anjia.ml/                                        #
# Github: https://github.com/anjia0532/elastalert-wechat-plugin     #
#                                                                   #
#####################################################################
'''
class WeChatAlerter(Alerter):

    #企业号id，secret，应用id必填

    required_options = frozenset(['corp_id','secret','agent_id'])

    def __init__(self, *args):
        super(WeChatAlerter, self).__init__(*args)
        self.corp_id = self.rule.get('corp_id', '')      #企业号id
        self.secret = self.rule.get('secret', '')        #secret
```

```
        self.agent_id = self.rule.get('agent_id', '')   #应用id

        self.party_id = self.rule.get('party_id')       #部门id
        self.user_id = self.rule.get('user_id', '')     #用户id，多人用 | 分割，全部用 @all
        self.tag_id = self.rule.get('tag_id', '')       #标签id
        self.access_token = ''                          #微信身份令牌
        self.expires_in=datetime.datetime.now() - datetime.timedelta(seconds=60)

    def create_default_title(self, matches):
        subject = 'ElastAlert: %s' % (self.rule['name'])
        return subject

    def alert(self, matches):

        if not self.party_id and not self.user_id and not self.tag_id:
            elastalert_logger.warn("All touser & toparty & totag invalid")

        # 参考elastalert的写法
        # https://github.com/Yelp/elastalert/blob/master/elastalert/alerts.py#L236-L243
        body = self.create_alert_body(matches)

        #matches 是json格式
        #self.create_alert_body(matches)是String格式,详见 [create_alert_body函数](https://github.com/Yelp/elastalert/blob/master/elastalert/alerts.py)

        # 微信企业号获取Token文档
        # http://qydev.weixin.qq.com/wiki/index.php?title=AccessToken
        self.get_token()

        self.senddata(body)

        elastalert_logger.info("send message to %s" % (self.corp_id))

    def get_token(self):

        #获取token是有次数限制的,本想本地缓存过期时间和token，但是elastalert每次调用都是一次性的，不能全局缓存
        if self.expires_in >= datetime.datetime.now() and self.access_token:
            return self.access_token

        #构建获取token的url
        get_token_url = 'https://qyapi.weixin.qq.com/cgi-bin/gettoken?corpid=%s&corpsecret=%s' %(self.corp_id,self.secret)

        try:
            response = requests.get(get_token_url)
            response.raise_for_status()
        except RequestException as e:
            raise EAException("get access_token failed , stacktrace:%s" % e)
            #sys.exit("get access_token failed, system exit")

        token_json = response.json()
```

```
81
82          if 'access_token' not in token_json :
83              raise EAException("get access_token failed , , the response is
:%s" % response.text())
84              #sys.exit("get access_token failed, system exit")
85
86          #获取access_token和expires_in
87          self.access_token = token_json['access_token']
88          self.expires_in = datetime.datetime.now() +
datetime.timedelta(seconds=token_json['expires_in'])
89
90          return self.access_token
91
92      def senddata(self, content):
93
94          #如果需要原始json，需要传入matches
95
96          # http://qydev.weixin.qq.com/wiki/index.php?
title=%E6%B6%88%E6%81%AF%E7%B1%BB%E5%9E%8B%E5%8F%8A%E6%95%B0%E6%8D%AE%E6%A0
%BC%E5%BC%8F
97          # 微信企业号有字符长度限制（2048），超长自动截断
98
99          # 参考 http://blog.csdn.net/handsomekang/article/details/9397025
100         #len utf8 3字节，gbk2 字节，ascii 1字节
101         if len(content) > 512 :
102             content = content[:512] + "..."
103
104         # 微信发送消息文档
105         # http://qydev.weixin.qq.com/wiki/index.php?
title=%E6%B6%88%E6%81%AF%E7%B1%BB%E5%9E%8B%E5%8F%8A%E6%95%B0%E6%8D%AE%E6%A0
%BC%E5%BC%8F
106         send_url = 'https://qyapi.weixin.qq.com/cgi-bin/message/send?
access_token=%s' %( self.access_token)
107
108         headers = {'content-type': 'application/json'}
109
110         # 替换消息标题为中文,下面的字段为logstash切分的日志字段
111         title_dict = {
112            # "At least": "报警规则:At least",
113             "@timestamp": "报警时间",
114             "_index": "索引名称",
115             "_type": "索引类型",
116             "ServerIP": "报警主机",
117             "hostname": "报警机器",
118             "message": "报警内容",
119             "class": "报错类",
120             "lineNum": "报错行",
121            "num_hits": "文档命中数",
122            "num_matches": "文档匹配数"
123         }
124
125         #print(f"type:{type(content)}")
126         for k, v in title_dict.items():
127             content = content.replace(k, v, 1 )
128
129         # 最新微信企业号调整校验规则，tagid必须是string类型，如果是数字类型会报错，故
而使用str()函数进行转换
130         payload = {
```

```
            "touser": self.user_id and str(self.user_id) or '', #用户账户，建议使用tag
            "toparty": self.party_id and str(self.party_id) or '', #部门id，建议使用tag
            "totag": self.tag_id and str(self.tag_id) or '', #tag可以很灵活的控制发送群体细粒度。比较理想的推送应该是，在heartbeat或者其他elastic工具自定义字段，添加标签id。这边根据自定义的标签id，进行推送
            'msgtype': "text",
            "agentid": self.agent_id,
            "text":{
                "content": content.encode('UTF-8').decode("latin1") #避免中文字符发送失败
               },
            "safe":"0"
        }

        # set https proxy, if it was provided
        # 如果需要设置代理，可修改此参数并传入requests
        # proxies = {'https': self.pagerduty_proxy} if self.pagerduty_proxy else None
        try:
            #response = requests.post(send_url, data=json.dumps(payload, ensure_ascii=False), headers=headers)
            response = requests.post(send_url, data=json.dumps(payload, cls=MyEncoder, indent=4, ensure_ascii=False), headers=headers)
            response.raise_for_status()
        except RequestException as e:
            raise EAException("send message has error: %s" % e)

        elastalert_logger.info("send msg and response: %s" % response.text)


    def get_info(self):
        return {'type': 'WeChatAlerter'}
```

在同级目录下创建MyEncoder.py文件

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import json

class MyEncoder(json.JSONEncoder):
     def default(self, obj):
         if isinstance(obj, bytes):
            return str(obj, encoding='utf-8')
         return json.JSONEncoder.default(self, obj)
```

. 3. 申请企业微信账号

step 1: 访问网站 注册企业微信账号（不需要企业认证）。
step 2: 访问apps 创建第三方应用，点击创建应用按钮 -> 填写应用信息：
Step3: 创建部门，获取部门ID

步骤详见网址：https://www.cnblogs.com/sanduzxcvbnm/p/13612580.html

. 4. 配置报警规则

配置规则文件

```
$ cd /app/elastalert/example_rules/
$ cp example_frequency.yaml  sms-applog.yaml
$ cat sms-applog.yaml  | grep -v ^#
name: 【日志报警】
use_strftine_index: true
type: frequency

index: filebeat-*
num_events: 1
timeframe:
  minutes: 1
filter:
- query:
    query_string:
      query: '"\[ERROR\]" NOT "发送邮件失败"'

alert:
 - "elastalert_modules.wechat_qiye_alert.WeChatAlerter"
corp_id: wxdxxx40b4720f24
secret: xa4pWq63sxxtaZzzEg8X86OZBIoOkToCbh_oNc
agent_id: 1000002
party_id: 2
```

> `index`：要查询的索引的名称, ES中存在的索引。
> `num_events`：此参数特定于frequency类型，并且是触发警报时的阈值。
> `filter`：用于过滤结果的Elasticsearch过滤器列表，这里的规则定义是除了包含“发送邮件失败”的错误日志，其他所有ERROR的日志都会触发报警。
> `alert`：定义报警方式，我们这里采用企业微信报警。
> `corp_id`：企业微信的接口认证信息

. 5. 运行ElastAlert

```
# cd ~/elastalert
$ python -m elastalert.elastalert --verbose --config
/app/elastalert/config.yaml --rule /app/elastalert/example_rules/sms-
applog.yaml
1 rules loaded
INFO:elastalert:Starting up
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 9.999904 seconds
INFO:elastalert:Queried rule 【日志报警】 from 2020-06-05 17:47 CST to 2020-
06-05 17:47 CST: 0 / 0 hits
INFO:elastalert:Ran 【日志报警】 from 2020-06-05 17:47 CST to 2020-06-05 17:47
CST: 0 query hits (0 already seen), 0 matches, 0 alerts sent
后台运行
$ nohup python -m elastalert.elastalert --verbose --config
/app/elastalert/config.yaml --rule /app/elastalert/example_rules/sms-
applog.yaml > nohup.txt 2>&1 &
```

> 注意：执行后提示找不到elastalert_modules模块的话，需要在`~/`目录下创建elastalert文件夹，然后再把elastalert_modules文件夹给放进去，而不是把elastalert_modules文件夹放在elastalert模块的安装路径下

```
    client = ElastAlerter(args)
  File "/usr/local/python3/lib/python3.6/site-packages/elastalert/elastalert.py", line 140, in __init__
    self.rules = self.rules_loader.load(self.conf, self.args)
  File "/usr/local/python3/lib/python3.6/site-packages/elastalert/loaders.py", line 126, in load
    raise EAException('Error loading file %s: %s' % (rule_file, e))
elastalert.util.EAException: Error loading file /app/elastalert/example_rules/sms-applog.yaml: Error initiating alert ['elastalert_modules.wechat_qiye_alert.WeChatAlerter']: Could not import module elastalert_modules.wechat_qiye_alert.WeChatAlerter: No module named 'elastalert_modules'
[root@kibana example_rules]# pip install elastalert_modules
```

出现这个提示:

是因为设置的接收报警的是一个报警组，但是应用中设置接收报警的是一个微信账户，只需要在应用中找到可接受范围修改成报警组就行了

```
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 9.999804 seconds
INFO:elastalert:Background configuration change check run at 2020-09-04 11:19 CST
INFO:elastalert:Background alerts thread 0 pending alerts sent at 2020-09-04 11:19 CST
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 9.999791 seconds
INFO:elastalert:Queried rule 【日志报警】 from 2020-09-04 11:04 CST to 2020-09-04 11:19 CST: 4 / 4 hits
INFO:elastalert:send msg and response: {"errcode":81013,"errmsg":"user & party & tag all invalid, hint: [1599189594_55_6535024b619172c96cb829b50b9849c4], from ip: 219.155.87.248, more info at https://open.work.weixin.qq.com/devtool/query?e=81013","invaliduser":"","invalidparty":"2"}
INFO:elastalert:send message to ww0b85c21458a13b12
INFO:elastalert:Ran 【日志报警】 from 2020-09-04 11:04 CST to 2020-09-04 11:19 CST: 4 query hits (3 already seen), 1 matches, 1 alerts sent
INFO:elastalert:Background configuration change check run at 2020-09-04 11:19 CST
INFO:elastalert:Background alerts thread 0 pending alerts sent at 2020-09-04 11:19 CST
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 9.998889 seconds
INFO:elastalert:Queried rule 【日志报警】 from 2020-09-04 11:04 CST to 2020-09-04 11:19 CST: 4 / 4 hits
INFO:elastalert:Ran 【日志报警】 from 2020-09-04 11:04 CST to 2020-09-04 11:19 CST: 4 query hits (4 already seen), 0 matches, 0 alerts sent
INFO:elastalert:Background configuration change check run at 2020-09-04 11:19 CST
INFO:elastalert:Background alerts thread 0 pending alerts sent at 2020-09-04 11:19 CST
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 9.999804 seconds
```

错误码查询工具

API错误码 81013

查询

错误说明： UserID、部门ID、标签ID全部非法或无权限

排查方法： **错误码：81013**

UserID、部门ID、标签ID全部非法或无权限。一般有以下两种原因：

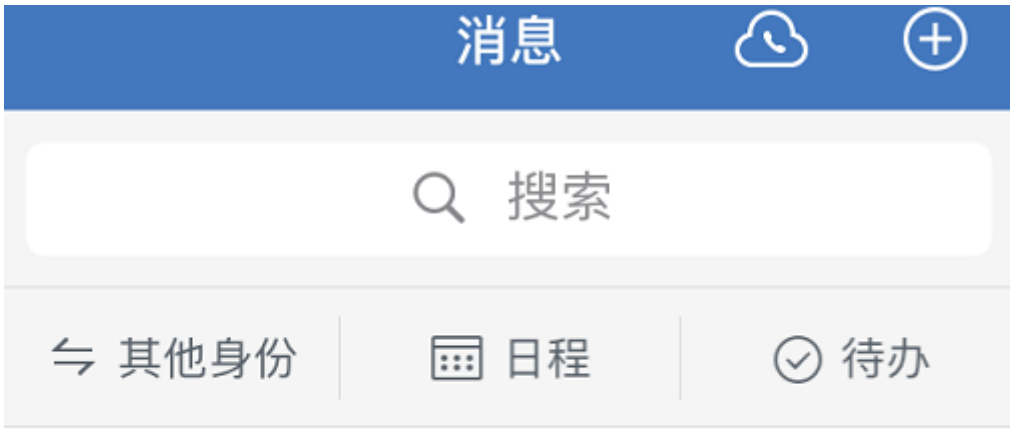
1）成员、部门或标签已被删除，此种情况需要调整调用接口的接收人参数。

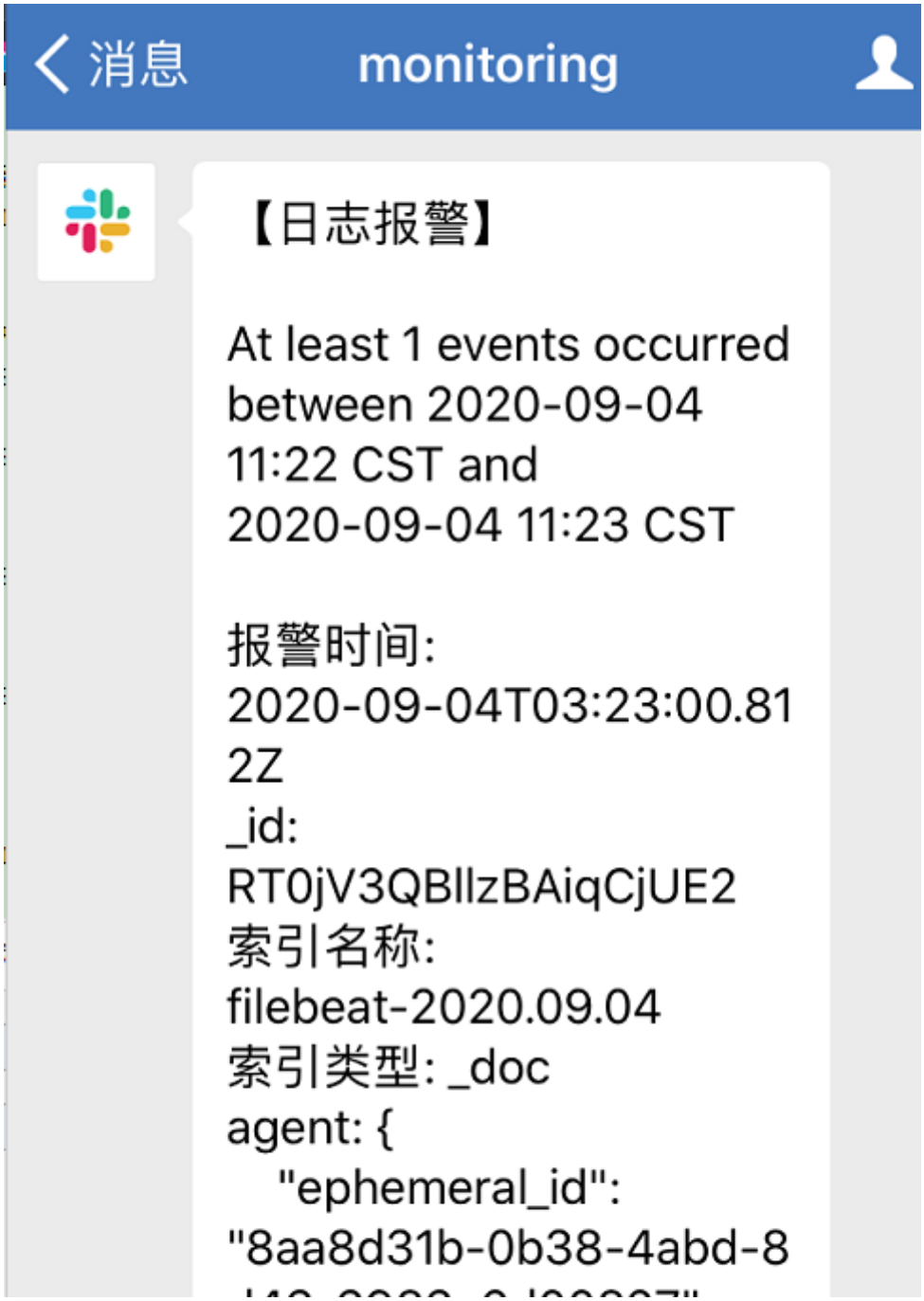
2）成员、部门或标签被移出应用的可见范围，可在管理端将接收人添加到应用的可见范围内。

正常发送报警的消息：

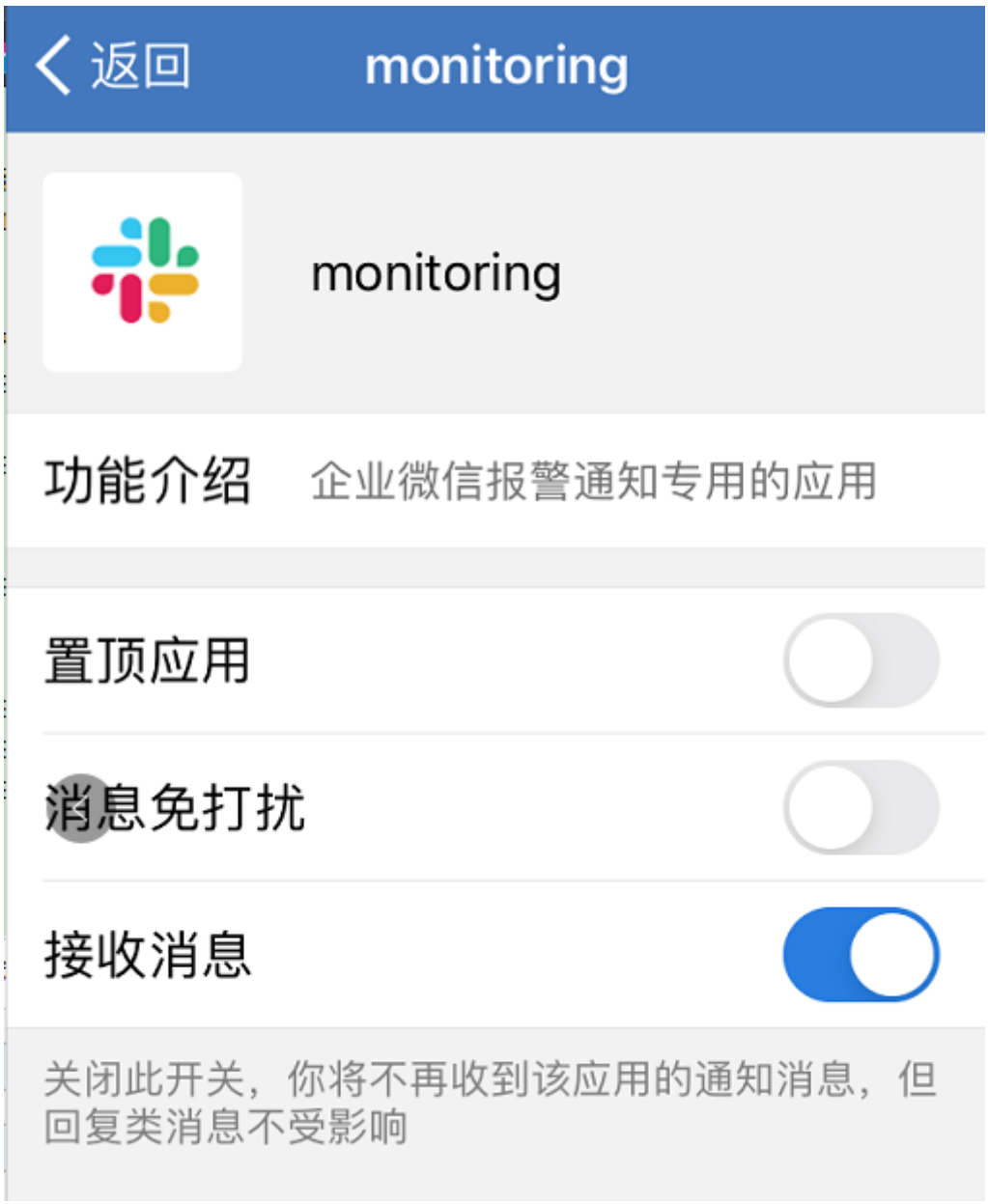
```
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 9.999808 seconds
INFO:elastalert:Queried rule 【日志报警】 from 2020-09-04 11:08 CST to 2020-09-04 11:23 CST: 5 / 5 hits
INFO:elastalert:send msg and response: {"errcode":0,"errmsg":"ok","invaliduser":""}
INFO:elastalert:send message to ww0b85c21458a13b12
INFO:elastalert:Ran 【日志报警】 from 2020-09-04 11:08 CST to 2020-09-04 11:23 CST: 5 query hits (4 already seen), 1 matches, 1 alerts sent
INFO:elastalert:Background configuration change check run at 2020-09-04 11:23 CST
INFO:elastalert:Background alerts thread 0 pending alerts sent at 2020-09-04 11:23 CST
INFO:elastalert:Queried rule 【日志报警】 from 2020-09-04 11:08 CST to 2020-09-04 11:23 CST: 5 / 5 hits
INFO:elastalert:Ran 【日志报警】 from 2020-09-04 11:08 CST to 2020-09-04 11:23 CST: 5 query hits (5 already seen), 0 matches, 0 alerts sent
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 9.999817 seconds
```

个人企业微信中收到的消息截图：






d43-3983e2d00827",
"报警机器":
"docker.domain.com",
消息
monitoring
agent: {
"ephemeral_id":
"8aa8d31b-0b38-4abd-8
d43-3983e2d00827",
"报警机器":
"docker.domain.com",
"id":
"147d9456-23f4-470e-9
4cb-fdddab45f5a6",
"type": "filebeat",
"version": "7.5.0"
}
ecs: {
"version": "1.1.0"
}
fields: {
"type": "test"
}
host: {
"architecture":
"x86_64",
"containe...


返回
monitoring
monitoring
功能介绍
企业微信报警通知专用的应用
置顶应用
消息免打扰
接收消息
关闭此开关，你将不再收到该应用的通知消息，但回复类消息不受影响

PDF文件下载地址

延伸：

1.监控规则中监控的日志索引是：`filebeat-*`,也就是说在elk中要有这个日志索引存在

2.监控规则如下：

```
filter:
- query:
    query_string:
      query: '"\[ERROR\]" NOT "发送邮件失败"'
```

日志索引中只要不包含"发送邮件失败"，凡是包含"[ERROR]"的信息都会触发报警

因此可以这样进行验证：

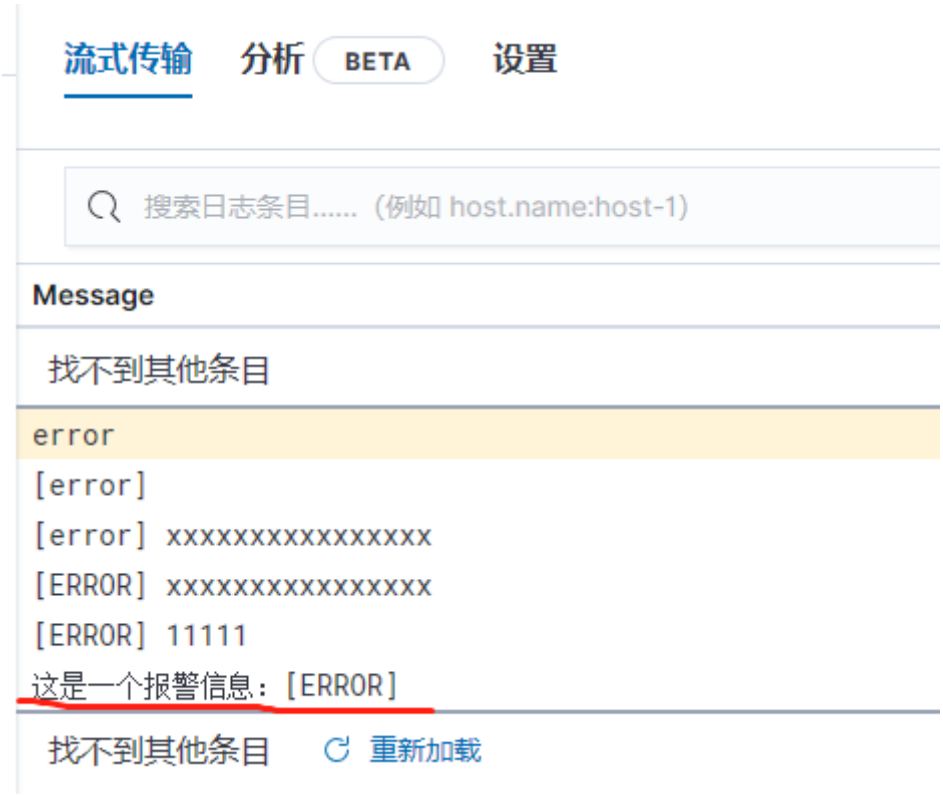
使用filebeat监控具体某一个日志文件，直接传输给es，但是传输给ES后创建的索引必须是以"filebeat-"开头的，具体可以看这个：https://www.cnblogs.com/sanduzxcvbnm/p/12350618.html

然后呢，手动往这个日志文件中写数据，在kibana中查看这个索引内容，然后查看是否触发报警：

1.手动往监控的日志文件中写数据，含有“[ERROR]”信息就行

```
[root@docker tmp]# echo "error" >> test.log
[root@docker tmp]# echo "[error]" >> test.log
[root@docker tmp]# echo "[error] xxxxxxxxxxxxxxx" >> test.log
[root@docker tmp]# echo "[ERROR] xxxxxxxxxxxxxxx" >> test.log
[root@docker tmp]# echo "[ERROR] 11111" >> test.log
[root@docker tmp]# echo "这是一个报警信息：[ERROR]" >> test.log
[root@docker tmp]#
```

2.在kibana中查看上一条数据是否写进来了



3.查看是否报警

```
INFO:elastalert:Background alerts thread 0 pending alerts sent at 2020-09-04 11:36 CST
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 9.999708 seconds
INFO:elastalert:Queried rule 【日志报警】 from 2020-09-04 11:22 CST to 2020-09-04 11:37 CST: 2 / 2 hits
INFO:elastalert:send msg and response: {"errcode":0,"errmsg":"ok","invaliduser":""}
INFO:elastalert:send message to ww0b85c21458a13b12
INFO:elastalert:Ran 【日志报警】 from 2020-09-04 11:22 CST to 2020-09-04 11:37 CST: 2 query hits (1 already seen), 1 matches, 1 alerts sent
INFO:elastalert:Background configuration change check run at 2020-09-04 11:37 CST
INFO:elastalert:Background alerts thread 0 pending alerts sent at 2020-09-04 11:37 CST
```

4.查看具体报警内容

3.若是想监控多个日志索引怎么办
可以再创建一个yaml报警规则的文件，然后再启动一个命令，指定使用这个报警规则文件

4.目录搞不明白的话可以看如下的目录结构
当前使用的目录结构如下：
目录结构：

```
/app/elastalert # 以下这些目录和文件是克隆git地址所提供的
└── build
└── changelog.md
└── config.yaml # 配置文件
└── config.yaml.example
```

```
└── dist
└── docker-compose.yml
└── Dockerfile-test
└── docs
└── elastalert
└── elastalert.egg-info
└── example_rules # 报警规则目录
└── LICENSE
└── Makefile
└── pytest.ini
└── README.md
└── requirements-dev.txt
└── requirements.txt
└── setup.cfg
└── setup.py
└── supervisord.conf.example
└── tests
└── tox.ini
```

```
~/elastalert # 这个是单独创建的目录
├── elastalert_modules
├     ├── __init__.py
├     ├── wechat_qiye_alert.py
```

使用的命令需要先进入到`~/elastalert`目录下执行,指定使用的配置文件，指定使用的报警规则

```
python -m elastalert.elastalert --verbose --config
/app/elastalert/config.yaml --rule /app/elastalert/example_rules/sms-
applog.yaml
# 后台运行
nohup python -m elastalert.elastalert --verbose --config
/app/elastalert/config.yaml --rule /app/elastalert/example_rules/sms-
applog.yaml > nohup.txt 2>&1 &
```

或者采用如下的这种目录结构：

```
~/elastalert
├── elastalert_modules # 这个是单独创建的目录，里面的这俩文件需要单独创建
├     ├── __init__.py
├     ├── wechat_qiye_alert.py
└── build # 以下这些目录是克隆git地址所提供的
└── changelog.md
└── config.yaml # 配置文件
└── config.yaml.example
└── dist
└── docker-compose.yml
└── Dockerfile-test
└── docs
└── elastalert
└── elastalert.egg-info
└── example_rules # 报警规则目录
└── LICENSE
└── Makefile
└── pytest.ini
└── README.md
└── requirements-dev.txt
```

```
└── requirements.txt
└── setup.cfg
└── setup.py
└── supervisord.conf.example
└── tests
└── tox.ini
```

使用的命令需要先进入到`~/elastalert`目录下执行,指定使用的报警规则

```
python -m elastalert.elastalert --verbose --rule
example_rules/example_frequency.yaml
```