日志平台使用示例

笔记本:	ElasticSearch		
创建时间:	2020/6/9 14:42	**更新时间:**	2020/6/9 15:55
作者:	jin.zhou@definesys.com		
URL:	http://127.0.0.1:8705/swagger-ui.html#/%E6%97%A5%E5%BF%97%E5%B9...		

使用filebeat的方式读取日志文件的日志

本文将实现从日志平台申请分区并且配置filebeat进行日志发送的功能

1、系统管理员创建主题

在本平台里面主题代表着一个日志类型，比如说MySQL主题，那么这个主题下就会有所有的项目的MySQL日志，不同系统的MySQL日志对应着不同的分区。
在本次示例里面我们创建一个主题叫业务系统主题（businesstopic） 需要注意一点，主题名称不允许大写，必须使用小写,不能有符号，因为该主题名称后面会作为日志系统索引名称的一部分。

查看kafka进程

进入测试环境 172.16.161.51 使用root账户 执行 `docker ps`
如下图所示

```
root@ztcw1dev mpaas]# docker ps
CONTAINER ID        IMAGE                                COMMAND                  CREATE
RTS                                                      NAMES
96a0c0e9dc2         wurstmeister/kafka                   "start-kafka.sh"         6 days
0.0.0:9004->9092/tcp                                     kafka_kafka3_1
6634f058672         wurstmeister/kafka                   "start-kafka.sh"         6 days
0.0.0:9003->9092/tcp                                     kafka_kafka2_1
2e1bf5fe251         sheepkiller/kafka-manager            "./start-kafka-man..."   6 days
0.0.0:9000->9000/tcp                                     kafka_kafka-manager_1
e8c5aefd036         wurstmeister/kafka                   "start-kafka.sh"         6 days
0.0.0:9002->9092/tcp                                     kafka_kafka1_1
```

进入kafka创建主题

我们随便选择一个kafka的进程进行主题创建，则这里我选择第一个

```
docker exec -it 096a0c0e9dc2 bash
```

进入控制台之后在进入kafka的安装目录

```
cd $KAFKA_HOME/bin
```

进入之后如图所示

```
bash-4.4# cd $KAFKA_HOME/bin
bash-4.4# ls
connect-distributed.sh              kafka-delete-records.sh             kafka-server-stop.sh
connect-mirror-maker.sh             kafka-dump-log.sh                   kafka-streams-applicatio
connect-standalone.sh               kafka-leader-election.sh            kafka-topics.sh
kafka-acls.sh                       kafka-log-dirs.sh                   kafka-verifiable-consume
kafka-broker-api-versions.sh        kafka-mirror-maker.sh               kafka-verifiable-produce
kafka-configs.sh                    kafka-preferred-replica-election.sh trogdor.sh
kafka-console-consumer.sh           kafka-producer-perf-test.sh         windows
kafka-console-producer.sh           kafka-reassign-partitions.sh        zookeeper-security-migra
kafka-consumer-groups.sh            kafka-replica-verification.sh       zookeeper-server-start.s
kafka-consumer-perf-test.sh         kafka-run-class.sh                  zookeeper-server-stop.sh
kafka-delegation-tokens.sh          kafka-server-start.sh               zookeeper-shell.sh
bash-4.4#
```

在当前目录下面我们执行以下命令创建一个主题，在这个命令里面我们创建了一个名为 businesstopic的主题，创建了50个分区，当到后期如果分区数目不够了可以动态的扩容分区

```
    ./kafka-topics.sh --create -zookeeper zookeeper:2181 --replication-
factor 2 --partitions 50  --topic businesstopic
```

当出现下面的结果的时候，就表示创建成功了

```
Created topic businesstopic.
```

2、业务系统用户申请主题和分区

当其他系统需要对接我们日志系统的时候首先需要在我们的日志平台上面申请一个主题和分区，
由于还没有前端界面，所以我们通过日志平台的如下接口可以查询到目前系统里面有哪些可用的主题

```
http://127.0.0.1:8705/log-platform/listTopics?page=1&pageSize=20
```

我们可以看到我们刚刚创建的主题已经存在在系统里面了

Response body


```
{
  "responseCode": "200",
  "msg": "请求成功",
  "data": {
    "pageSize": 20,
    "pageNum": 0,
    "count": 16,
    "result": [
      {
        "topicId": null,
        "topicKey": "businesstopic",
        "topicName": null,
        "topicNote": null,
        "creationDate": null,
        "lastUpdateDate": null,
        "createdBy": null,
        "lastUpdateBy": null,
        "objectVersionNumber": null
      },
      {
        "topicId": null,
```


申请一个可用的分区

由于我们已经得到了需要使用的主题，我们可以通过接口向该主题申请一个分区

```
http://127.0.0.1:8705/log-platform/applyPartitionKey?
topicKey=businesstopic
```

返回结果如图，可以知道申请的分区编号是 0

Response body


```
{
  "responseCode": "200",
  "msg": "请求成功",
  "data": "0"
}
```


保存分区配置

如若有前端界面，我们可以在界面上填写如下的信息
接口地址是 http://127.0.0.1:8705/log-platform/savePartitionInfo
下面的参数里面 partitionKey代表之前申请的分区，partitionTopic代表之前申请分区的topic

```
{
  "partitionKey": "0",
  "partitionNote": "业务系统A的一个分区",
  "partitionNumber": "",
  "partitionTopic": "businesstopic",
  "partitionUser": "jenkin"
}
```

设置日志记录的配置信息

这一步十分关键，必须要配置，在这一步里面可以配置日志预警的方式，以及日志预警的消息模板，同时还可以配置日志归档的方式，日志索引的名称等等
在下面的配置里面我们配置了 索引的名称后缀为log_platform，日志预警的关键字为error，日志预警之后进行通知的模板【系统发生了异常，请及时处理】，日志归档的方式是每天一次，**日志系统的编码是businesstopic^0"，这是一个固定的格式，一定不能更改，日志系统编码就是用主题^分区这种格式**

```
{

  "indexSuffix": "log_platform",
  "keyWords": "error",
  "needLogContent": 1,
  "noticeTemplate": "系统发生了异常，请及时处理",
  "placeOnFileTime": 1,
  "placeOnFileTimeUnit": "DAY",
  "placeOnFileType": "DELETE",
  "systemCode": "businesstopic^0",
  "targetEmails": "1394046585@qq.com"
}
```

调用接口：http://127.0.0.1:8705/log-platform/saveSystemNodeConfig
在这一步保存成功之后就已经可以使用了，这里看一下效果

在业务系统上安装filebeat

这里我们使用可以两种方式安装filebeat 7.1.1
1、使用rpm包管理器安装

```
curl -L -O
https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.1.1-
x86_64.rpm
sudo rpm -vi filebeat-7.1.1-x86_64.rpm
```

2、解压安装

```
curl -L -O
https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.1.1-
darwin-x86_64.tar.gz
tar xzvf filebeat-7.1.1-darwin-x86_64.tar.gz
cd filebeat-7.1.1-darwin-x86_64/
```

安装好之后可以进入目录 /etc/filebeat/ 看到配置文件 filebeat.yml

修改filebeat配置文件

1、input部分的配置

我们编辑filebeat的配置文件，在inputs模块里面配置如下内容：
在下面的配置里面我们把主题和分区都配置为我们刚刚申请的就可以了

```
- type: log
  enabled: true
  paths:
    - /data/business-system-logs.log
  tail_files: true
  encoding: UTF-8
  fields:
    partition: "0"
    log_topic: "businesstopic"
  fields_under_root: true
  multiline.pattern: '^[[:space:]]+(at|\.{3})\b|^Caused by:'
  multiline.negate: false
  multiline.match: after
```

2、输出部分的配置

输出部分我们直接输出到kafka里面，如果是测试环境，那么直接复制下面的配置不用改动其他任何代码

```
#------------------------------ Kafka output ------------------------------
----
output.kafka:
  hosts:
```

```
["172.16.161.51:9002","172.16.161.51:9003","172.16.161.51:9004"]
  topic: '%{[log_topic]}'
  partition.hash:
    reachable_only: false
    hash: ['partition']
```

3、重启filebeat

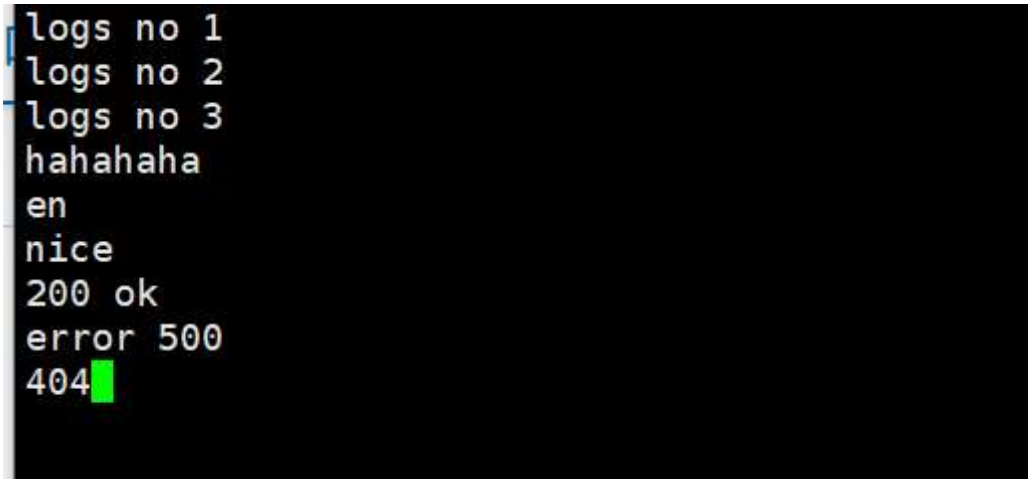
在配置好上面的配置文件之后我们需要重启一下filebeat好让配置生效

```
service filebeat restart
```

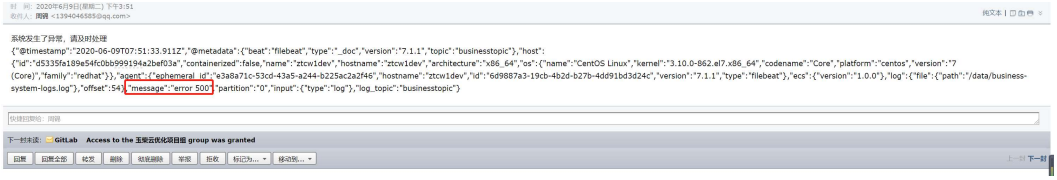
4、验证

进入我们刚刚配置的日志文件目录里面（/data/business-system-logs.log）

编辑这个日志文件，我们添加了如下的几行日志



可以看到我们已经收到了邮件，邮件内容如下：



这时候我们再kibana里面也是能够查到所有的日志的

在kibana控制面板里面我们执行以下的查询

```
GET /businesstopic_0_log_platform_2020_06_09/_search
{
  "query": {
    "match_all": {}
  }
}
```

可以看到结果，一共有9条日志记录


```
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 9,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "businesstopic_0_log_platform_2020_06_09",
        "_type" : "_doc",
        "_id" : "WpcQmHIBu7rhztw6KFKE",
        "_score" : 1.0,
        "_source" : {
          "@timestamp" : "2020-06-09T07:51:33.901Z",
          "@metadata" : {
            "beat" : "filebeat",
            "type" : "_doc",
            "version" : "7.1.1",
            "topic" : "businesstopic"
          },
          "agent" : {
            "id" : "6d9887a3-19cb-4b2d-b27b-4dd91bd3d24c",
            "version" : "7.1.1",
            "type" : "filebeat",
            "ephemeral_id" : "e3a8a71c-53cd-43a5-a244-b225ac2a2f46",
            "hostname" : "ztcw1dev"
          },
          "log" : {
            "offset" : 0,
            "file" : {
              "path" : "/data/business-system-logs.log"
            }
          },
          "message" : "logs no 1",
          "log_topic" : "businesstopic",
          "partition" : "0",
          "input" : {
            "type" : "log"
          },
          "ecs" : {
            "version" : "1.0.0"
          },
          "host" : {
            "id" : "d5335fa189e54fc0bb999194a2bef03a",
            "containerized" : false,
            "hostname" : "ztcw1dev",
            "architecture" : "x86_64",
            "name" : "ztcw1dev",
            "os" : {
              "platform" : "centos",
              "version" : "7 (Core)",
              "family" : "redhat",
              "name" : "CentOS Linux",
```