Logstash实现mysql和ES的同步实现成功：

docker容器下安装logstash：

1、下载镜像，将容器文件复制一份到本地文件夹里,然后删除容器

docker run -d --name=logstash1 logstash:7.7.0

docker exec -it logstash1 /bin/bash  
cd /bin/

logstash-plugin install logstash-input-jdbc

logstash-plugin install logstash-output-elasticsearch

logstash-plugin install logstash-codec-json\_lines //可以不用

docker stop logstash1

docker rm logstash1

1. 重新跑一个容器挂载目录到本地目录下

docker run --name=logstash1 \

-v /home/logstash/config:/usr/share/logstash/config \

-v /home/logstash/data:/usr/share/logstash/data \

-v /home/logstash/pipeline:/usr/share/logstash/pipeline \

-d logstash:7.7.0

chmod 777 -R /home/logstash

1. 进入到容器内部进行插件安装：

docker exec -it logstash1 /bin/bash  
cd /bin/

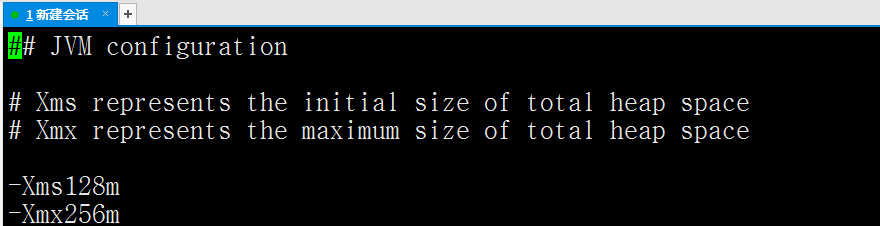
logstash-plugin install logstash-input-jdbc

logstash-plugin install logstash-output-elasticsearch

logstash-plugin install logstash-codec-json\_lines

Ps：要踩的坑，提示各种内存不足，类似于error='Not enough space'，查看jvm大小然后修改大小

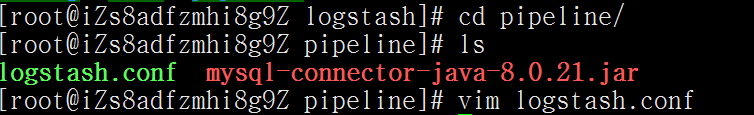
find / -name jvm.options



1. 下载jar包放到pipeline下



1. 到pipeline里面编辑logstash.conf



内容如下：这里的jar包需要自己去下载一下。这里特别注意下指定的id不论你数据库是大写还是小写，统一为小写，其他的符号跟上即可，document\_id => "%{id\_}"。

input {

stdin { }

jdbc {

#注意mysql连接地址一定要用ip，不能使用localhost等

jdbc\_connection\_string => "jdbc:mysql://8.129.217.148:3310/zspoa\_member"

jdbc\_user => "root"

jdbc\_password => "123456"

#这个jar包的地址是容器内的地址

jdbc\_driver\_library => "/usr/share/logstash/pipeline/mysql-connector-java-8.0.21.jar"

jdbc\_driver\_class => "com.mysql.cj.jdbc.Driver"

jdbc\_paging\_enabled => "true"

jdbc\_page\_size => "50000"

statement => "SELECT \* FROM member"

schedule => "\* \* \* \* \*"

}

}

output {

stdout {

codec => json\_lines

}

elasticsearch {

#注意mysql连接地址一定要用ip，不能使用localhost等

hosts => "121.199.2.132:9200"

index => "member"

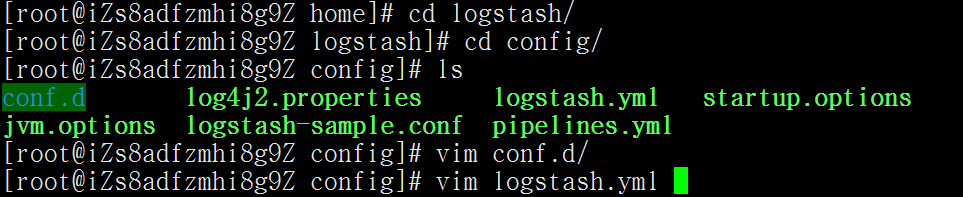
document\_type => "\_doc"

document\_id => "%{id}"

}

}

6、修改logstash.yml



logstash.yml全部内容改成：

http.host: "0.0.0.0"

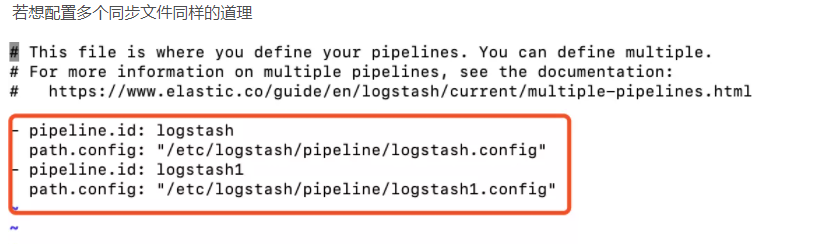
xpack.monitoring.elasticsearch.hosts: [ "http://121.199.2.132:9200" ]

这是pipelines.yml的文档修改

- pipeline.id: table1

path.config: "/usr/share/logstash/pipeline/logstash.conf"

如果需要配置多个表同步则可以多写点配置：

7、重启容器：

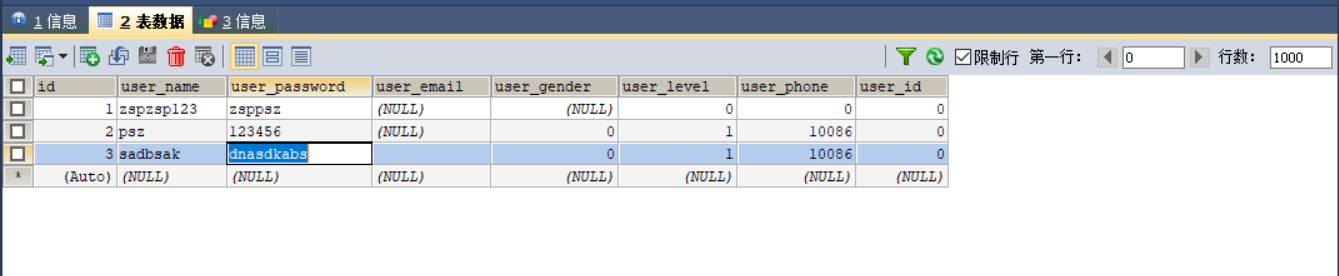
docker restart logstash

查看执行结果：

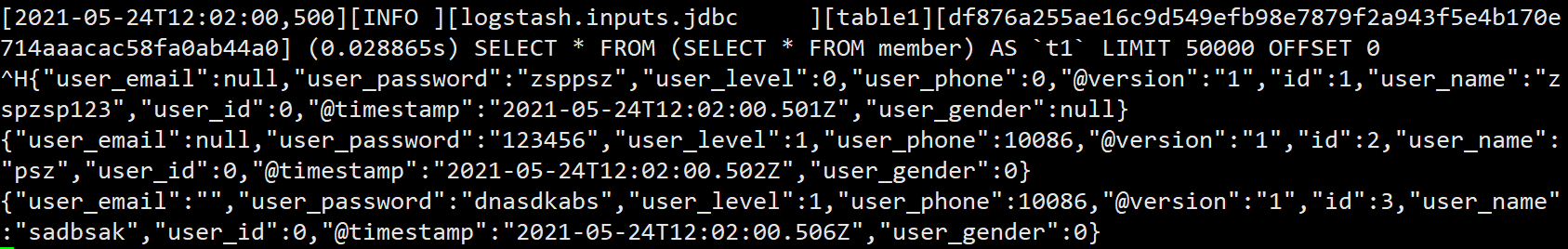
docker logs -f --tail=30 logstash1

演示结果：

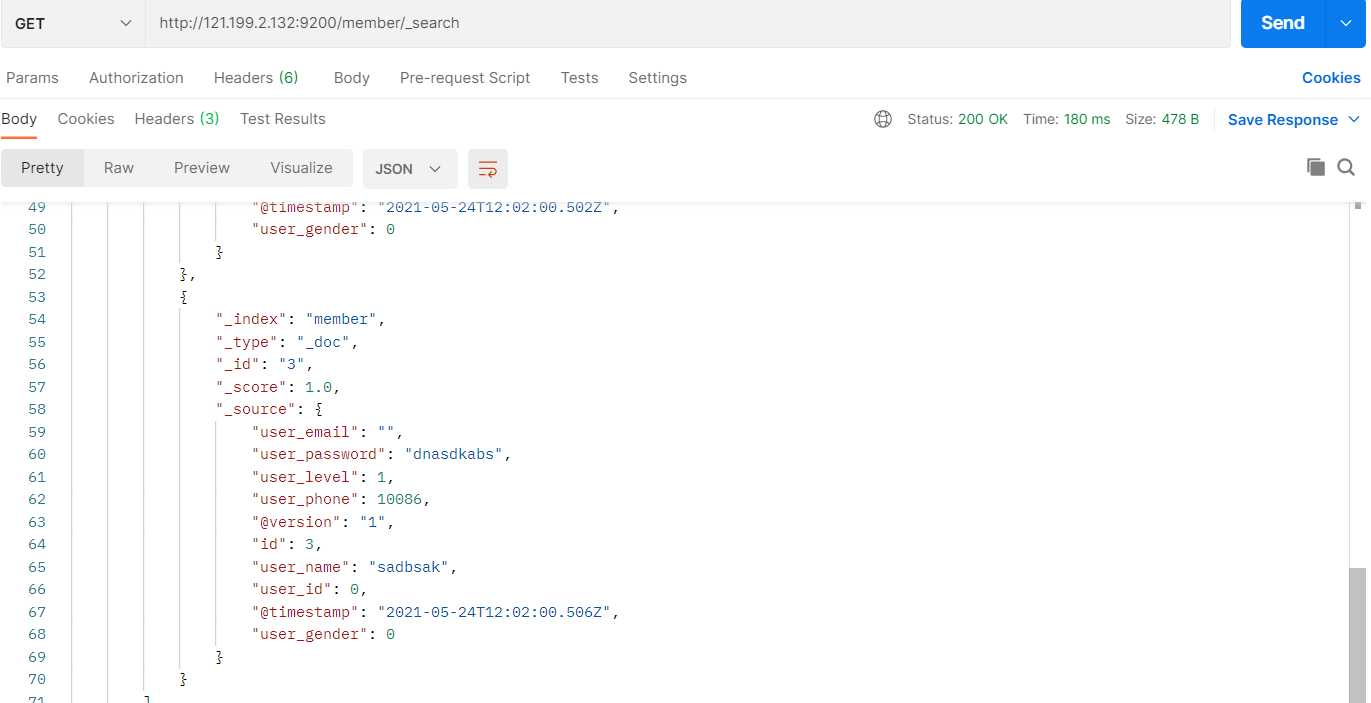
1. 更新数据库



1. 等待logstash执行补充

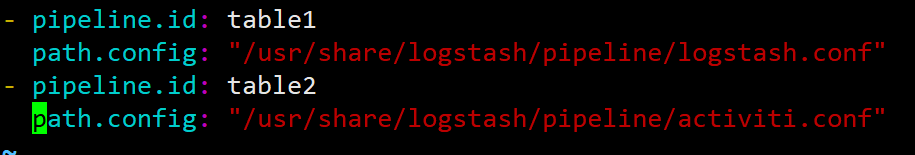


1. Postman查询



多表两个方法：

1. 直接在刚才的pipelines.yml指定新的table然后给予配置文件位置



activiti.conf的内容为：

#这个jar包的地址是容器内的地址

jdbc\_driver\_library => "/usr/share/logstash/pipeline/mysql-connector-java-8.0.21.jar"

jdbc\_driver\_class => "com.mysql.cj.jdbc.Driver"

jdbc\_paging\_enabled => "true"

jdbc\_page\_size => "50000"

statement => "SELECT \* FROM act\_hi\_actinst"

schedule => "\* \* \* \* \*"

}

}

output {

stdout {

codec => json\_lines

}

elasticsearch {

#注意mysql连接地址一定要用ip，不能使用localhost等

hosts => "121.199.2.132:9200"

index => "act\_actinst"

document\_type => "\_doc"

document\_id => "%{id\_}"

}

}

重新启动容器即可，这里特别注意下指定的id不论你数据库是大写还是小写，统一为小写，其他的符号跟上即可，document\_id => "%{id\_}"。

1. 第二个方法在一个配置类里面写上多表同步即可，（不过我没有使用过。

多表同步：

input {

stdin {

}

jdbc {

type => "zspoa\_member"

#注意mysql连接地址一定要用ip，不能使用localhost等

jdbc\_connection\_string => "jdbc:mysql://8.129.217.148:3310/zspoa\_member"

jdbc\_user => "root"

jdbc\_password => "123456"

#这个jar包的地址是容器内的地址

jdbc\_driver\_library => "/usr/share/logstash/pipeline/mysql-connector-java-8.0.21.jar"

jdbc\_driver\_class => "com.mysql.cj.jdbc.Driver"

jdbc\_paging\_enabled => "true"

jdbc\_page\_size => "50000"

statement => "SELECT \* FROM member"

schedule => "\* \* \* \* \*"

}

jdbc {

type => "zspoa\_act\_actinst"

#注意mysql连接地址一定要用ip，不能使用localhost等

jdbc\_connection\_string => "jdbc:mysql://8.129.217.148:3310/zspoa\_activiti"

jdbc\_user => "root"

jdbc\_password => "123456"

#这个jar包的地址是容器内的地址

jdbc\_driver\_library => "/usr/share/logstash/pipeline/mysql-connector-java-8.0.21.jar"

jdbc\_driver\_class => "com.mysql.cj.jdbc.Driver"

jdbc\_paging\_enabled => "true"

jdbc\_page\_size => "50000"

statement => "SELECT \* FROM act\_hi\_actinst"

schedule => "\* \* \* \* \*"

}

}

filter {

mutate {

convert => [ "publish\_time", "string" ]

}

date {

timezone => "Europe/Berlin"

match => ["publish\_time" , "ISO8601", "yyyy-MM-dd HH:mm:ss"]

}

#date {

# match => [ "publish\_time", "yyyy-MM-dd HH:mm:ss,SSS" ]

# remove\_field => [ "publish\_time" ]

# }

json {

source => "message"

remove\_field => ["message"]

}

}

output {

if [type]=="zspoa\_member" {

elasticsearch {

#ESIP地址与端口

hosts => "121.199.2.132:9200"

#ES索引名称（自己定义的）

index => "zspoa\_member"

#自增ID编号

# document\_id => "%{id}"

}

}

if [type]=="zspoa\_act\_actinst" {

elasticsearch {

#ESIP地址与端口

hosts => "121.199.2.132:9200"

#ES索引名称（自己定义的）

index => "zspoa\_act\_actinst"

#自增ID编号

# document\_id => "%{ID\_}"

}

}

}

使用docker logs -f --tail=30 logstash1查看结果