**简要概述**

1. 所有的系统日志收集都采用rsyslog，rsyslog通过omkafka模块将系统日志格式化写入kafka。**[rsyslog的配置模板](http://219.83.161.15:9999/zentao/doc-view-89.html" \t "_self)实例**
2. 启动的单独的logstash实例，设置topic，组名和消费者id以及线程数，logstash启动后从kafka主题分区中按顺序消费数据
3. logstash将解析过滤格式化后的非结构化（json格式）的数据写入elasticsearch

**配置文件**

在/etc/logstash/conf.d/目录下创建test.conf文件内容如下：

# Beats -> Logstash -> Elasticsearch pipeline.  
input {  
    kafka{  
        bootstrap\_servers => ["192.168.123.231:9092,192.168.123.232:9092,192.168.123.233:9092"]  
        client\_id => "es-node1-log"  
        group\_id => "es-nodes-log"  
        auto\_offset\_reset => "latest"  
        consumer\_threads => 3  
        decorate\_events => "true"  
        topics => ["test\_syslog"]  
        type => "syslog\_type"  
      }  
}  
filter {  
    if [type] == "syslog\_type" {  
        grok {  
                patterns\_dir => ["/etc/logstash/pattern\_dir/patterns"]  
                match => {"message" => "%{SSH\_MSG}"}  
                match => {"message" => "%{SYS\_MSG}"}  
        }  
  
        date {  
                match => ["[@metadata][datetime]", "yyyy-MM-dd HH:mm:ss.SSS", "ISO8601"]  
                target => "@timestamp"  
                timezone => "Asia/Shanghai"  
        }  
  
        mutate {  
                remove\_field => ["@version","message"]  
                add\_field => ["partion", "%{[@metadata][kafka][partition]}"]  
        }  
    }else {  
        drop{}  
    }  
}  
output {  
    if [progname] == "sshd" {  
        elasticsearch {  
            hosts => ["192.168.123.231:9200","192.168.123.232:9200","192.168.123.233:9200"]  
            index => "syslog-test-%{+YYYY.MM.dd}"  
        }  
    }  
    stdout {  
        codec => rubydebug {  
        #metadata => true  
        }  
    }  
}

在上述配置中，将解析匹配的正则表达式都写入到了正则模板文件中,模板文件路径:

/etc/logstash/pattern\_dir/patterns

该文件配置内容查看：**[正则模板文件内容](http://219.83.161.15:9999/zentao/doc-view-92.html" \t "_self)**

**解析处理流程**

上面logstash的配置文件test.conf中主要有input，filter，output三大核心块，其配置语法采用的是ruby的DSL。

其中每个核心快区域中都有相关的处理插件，input是用来接收数据源的，可以通过监听tcp/udp端口，编解码，序列化，添加类型，添加标签等常用的插件

这里只接受我们需要的插件，对其他处理数据的插件可以参考官网地址: <https://www.elastic.co/guide/en/logstash/6.4/input-plugins.html>

由于logstash是作为消费者使用，日志消息是存放在kafka中的，需要logstash能够和kafka进行交互，官网提供了input-kafka插件。

其主要参数配置如下:

input {  
    kafka{  
            #指定kafka集群所有节点ip  
        bootstrap\_servers => ["192.168.123.231:9092,192.168.123.232:9092,192.168.123.233:9092"]  
        #给每个年消费者线程设置唯一的id，  
        client\_id => "es-node1-log"  
        #给每个组设置唯一的id（注：消费者线程都是在一个组中，有共同的组id，但是每个分区只能指定唯一的组内的单个消费者线程）  
        group\_id => "es-nodes-log"  
        #消费事件记录，记录当前读取的消息位置  
        auto\_offset\_reset => "latest"  
        #设置消费者线程数  
        consumer\_threads => 3  
        #开启该选项后，会附加一些元数据信息比如topic，分区，offset，时间戳到日志消息事件，但不会添加到日志消息中，可以通过设置@metadata => 查看附加的字段信息  
        decorate\_events => "true"  
        #指定要消费的主题(topic)名  
        topics => ["test\_syslog"]  
        #添加字段类型方便对日志消息进行分类处理  
        type => "syslog\_type"  
      }  
     kafka{  
          ...（还可以设置多个kafka连接不同的topic）  
      }  
}

消息经过kafka插件处理后，会放入logstash的队列中,如果需要过滤的话则应该在filter中进行处理(也可以不处理直接输出)

filter 提供了各种过滤插件用来解析，匹配，增加，修改，删除，类型转换等插件来对消息进行处理。（具体可以参考官网）

这里只列出需要处理日志信息的插件，filter区域配置如下:

filter {  
    if [type] == "syslog\_type" {  
        grok {  
                        #加载正则配置模板文件，文件必须以类似name pattern的模式  
                patterns\_dir => ["/etc/logstash/pattern\_dir/patterns"]  
                #SSH\_MS和SYS\_MSG是在正则模板配置建中设置的变量通过%{}来引用。可以查看模板文件内容看些变量对应的模式匹配  
                match => {"message" => "%{SSH\_MSG}"}  
                match => {"message" => "%{SYS\_MSG}"}  
        }  
                  
                #日期修改插件，可以设置时区吗，对时间字符串进行格式化  
        date {  
                match => ["[@metadata][datetime]", "yyyy-MM-dd HH:mm:ss.SSS", "ISO8601"]  
                target => "@timestamp"  
                timezone => "Asia/Shanghai"  
        }  
                  
                #更改插件，可以增加，删除，转换字段  
        mutate {  
                remove\_field => ["@version","message"]  
                add\_field => ["partion", "%{[@metadata][kafka][partition]}"]  
        }  
    }else {  
            #删除掉信息  
        drop{}  
    }  
}

output 内容：

output {  
    if [progname] == "sshd" {  
        #格式化处理后的日志信息输入到elasticserach  
        elasticsearch {  
            hosts => ["192.168.123.231:9200","192.168.123.232:9200","192.168.123.233:9200"]  
            index => "syslog-test-%{+YYYY.MM.dd}"  
        }  
    }  
    #将格式化处理后的日志输出到控制台  
    stdout {  
        codec => rubydebug {  
        #metadata => true  
        }  
    }  
}

**启动logstash**

命令：

lostash -f /etc/logstash/conf.d/test.conf --config.reload.automatic

解释：

* -f: 指定配置文件
* --config.reload.automatic： 修改配置文件时不需要重新启动logstash

**控制台信息**

模拟ssh登录服务器观察控制台输出的信息（前提是rsyslog一定要配置好，参考**[rsyslog的配置模板](http://219.83.161.15:9999/zentao/doc-view-89.html" \t "_self)实例**）

比如我通过ssh连接ip地址192.168.123.231这台服务节点，控制台输出信息如下

{  
          "type" => "syslog\_type",  
      "facility" => "daemon",  
      "messages" => "Started Session 317 of user root.",  
      "loglevel" => "info",  
       "partion" => "1",  
            "ip" => "192.168.123.231",  
    "@timestamp" => 2019-03-18T09:37:16.702Z,  
      "progname" => "systemd",  
        "procid" => "1"  
}  
{  
          "type" => "syslog\_type",  
      "facility" => "authpriv",  
      "messages" => "pam\_unix(sshd:session): session opened for user root by (uid=0)",  
      "loglevel" => "info",  
       "partion" => "2",  
            "ip" => "192.168.123.231",  
    "@timestamp" => 2019-03-18T09:37:16.703Z,  
      "progname" => "sshd",  
        "procid" => "29235"  
}  
{  
          "type" => "syslog\_type",  
      "facility" => "authpriv",  
      "messages" => "Accepted password for root from 192.168.123.244 port 60395 ssh2",  
      "loglevel" => "info",  
     "client\_ip" => "192.168.123.244",  
       "partion" => "2",  
          "user" => "root",  
            "ip" => "192.168.123.231",  
    "@timestamp" => 2019-03-18T09:37:16.696Z,  
      "progname" => "sshd",  
        "procid" => "29235"  
}

**kibana查看信息**

