

# ELK 实时日志分析平台 环境搭建手册

2019 年 3 月

## 文档说明

文档名称	《ELK 实时日志分析平台环境搭建手册》		
文档编号		版本号	1.0
文档类型	<input type="radio"/> 原型稿 <input checked="" type="radio"/> 初稿 <input type="radio"/> 征求意见稿 <input type="radio"/> 最终稿		
编制		日期	2019/04/08
审核		日期	
备注			

## 修订记录

修订人	修订内容摘要	产生版本	修订日期	审核人

# 1 ELK 配置

## 1.1 组件介绍

### Elasticsearch:

是一个基于 Lucene 的搜索服务器。提供搜集、分析、存储数据三大功能。它提供了一个分布式多用户能力的全文搜索引擎，基于 RESTful web 接口。Elasticsearch 是用 Java 开发的，并作为 Apache 许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

### Logstash:

主要是用来日志的搜集、分析、过滤日志的工具。用于管理日志和事件的工具，你可以用它去收集日志、转换日志、解析日志并将他们作为数据提供给其它模块调用，例如搜索、存储等。

### Kibana:

是一个优秀的前端日志展示框架，它可以非常详细的将日志转化为各种图表，为用户提供强大的数据可视化支持。

### Filebeat: 隶属于 Beats。目前 Beats 包含四种工具:

#### 1.Packetbeat（搜集网络流量数据）

2.Metricbeat（搜集系统、进程和文件系统级别的 CPU 和内存使用情况等数据。通过从操作系统和服务收集指标，帮助您监控服务器及其托管的服务。）

3.Filebeat（搜集文件数据）

4.Winlogbeat（搜集 Windows 事件日志数据）

## Kafka:

- 1.发布和订阅记录流，类似于消息队列或企业消息传递系统。
- 2.以容错持久的方式存储记录流。
- 3.处理记录发生的流。

## 1.2 环境需求

- （1）VMware
- （2）Centos 6.5
- （3）JDK 1.8
- （4）nginx

**jdk** 官方下载地址 <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

## 1.3 JDK 安装

### 1.3.1 卸载旧版本 JDK

查看是否安装过 java

```
rpm -qa | grep java
```

`rpm -e --nodeps` #要卸载的包（包通过上面的指令可以获取到）

### 1.3.2 选择安装目录

选择安装 JDK 的位置/opt/install，如果存在这个目录无需创建，一般新到的机器是没有这个目录的，这个我们创建这个目录。指令 需要输入密码

管理员：mkdir /opt/install

非管理员：sudo mkdir /opt/install

### 1.3.3 上传和解压

将 jdk-8u131-linux-x64.tar.gz 上传到服务器的 /opt/install，使用 SecureFXPortable(或者 filezilla)将文件上传到服务器解压：

cd /opt/install      #进入 install 目录

tar -zxvf jdk-8u131-linux-x64.tar.gz      #解压到当前目录

```
[root@localhost ~]# rpm -qa | grep java      查询是否安装过java
[root@localhost ~]# mkdir /opt/install      创建目录
[root@localhost ~]# cd /opt/install      切换到目录
[root@localhost install]# ls
jdk-8u161-linux-x64.tar.gz
[root@localhost install]# tar -zxvf jdk-8u161-linux-x64.tar.gz      解压压缩包
```

### 1.3.4 环境变量配置

```
vi /etc/profile    #打开配置文件
```

在文档的最后面添加如下内容，记住不要带空格：

```
export JAVA_HOME=/opt/install/jdk1.8.0_131
```

```
export JRE_HOME=$JAVA_HOME/jre
```

```
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
```

```
export PATH=$PATH:$JAVA_HOME/bin
```

```
source /etc/profile    #让配置生效
```

```
java -version          #验证是否安装成功
```

```
[root@localhost install]# java -version
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)
[root@localhost install]#
```

## 2. Elasticsearch

### 2.1 安装 Elasticsearch

将 `elasticsearch-6.5.3.tar.gz` 上传到服务器的 `/opt/install`，使用 `SecureFXPortable`(或者 `filezilla`)将文件上传到服务器解压：

```
cd /opt/install          #进入 install 目录
```

```
tar -zxvf elasticsearch-6.5.3.tar.gz    #解压到当前目录
```

## 2.2 运行 Elasticsearch

切换至 elasticsearch-6.5.3 目录下的 bin 目录，使用

`./ elasticsearch` 运行

```
[root@localhost ~]# cd /opt/install/elasticsearch-6.5.3/bin
[root@localhost bin]# ./elasticsearch
```

## 2.3 处理异常

### 2.3.1 uncaught exception in thread [main]org.elasticsearch.bootstrap.StartupException:

**java.lang.RuntimeException: can not run elasticsearch as root**

原因是 elasticsearch 默认是不支持用 root 用户来启动的。

解决方案一：Des.insecure.allow.root=true

修改/usr/local/elasticsearch-2.4.0/bin/elasticsearch，

添加 ES\_JAVA\_OPTS="-Des.insecure.allow.root=true"

或执行时添加：`sh /usr/local/elasticsearch-2.4.0/bin/elasticsearch -d -`

`Des.insecure.allow.root=true`

注意：正式环境用 root 运行可能会有安全风险，不建议用 root 来跑。

解决方案二：添加专门的用户

`useradd elastic`

`chown -R elastic:elastic elasticsearch-2.4.0`

`su elastic`

```
sh /usr/local/elasticsearch-2.4.0/bin/elasticsearch -d
```

### **2.3.2 Could not register mbeans java.security.AccessControlException: access denied ("javax.management.MBeanTrustPermission" "register")**

`sudo chown -R noroot:noroot elasticsearch` 改变 elasticsearch 文件夹所有者到当前用户

这是因为 elasticsearch 需要读写配置文件，我们需要给予 config 文件夹权限，需要新建了 elsearch 用户，elsearch 用户不具备读写权限，因此还是会报错，解决方法是切换到管理员账户，赋予权限即可：

`sudo -root` 切换至管理员权限

`chmod -R 775 config` 赋予权限

### **2.3.3 bootstrap checks failed[1]: max file descriptors [4096] for elasticsearch process is too low, increase to at least [65536]**

原因：无法创建本地文件问题,用户最大可创建文件数太小

解决方案：切换到 root 用户，编辑 limits.conf 配置文件，添加类似如下内容：

`vim /etc/security/limits.conf`

添加如下内容：



```
* soft nfile 65536
* hard nfile 131072
* soft nproc 2048
* hard nproc 4096
```

### **2.3.4 max number of threads [1024] for user [tzs] is too low, increase to at least [2048]**

原因：无法创建本地线程问题,用户最大可创建线程数太小

解决方案：切换到 root 用户，进入 limits.d 目录下，修改 90-nproc.conf 配置文件。

```
vim /etc/security/limits.d/90-nproc.conf
```

找到如下内容：

```
soft nproc 1024
```

修改为

```
soft nproc 2048
```

### **2.3.5 max virtual memory areas vm.max\_map\_count [65530] is too low, increase to at least [262144]**

原因：最大虚拟内存太小

root 用户执行命令：

```
sysctl -w vm.max_map_count=262144
```

或者修改 `/etc/sysctl.conf` 文件，添加 “`vm.max_map_count`” 设置 设置后，可以使用 `$ sysctl -p`

## 2.3.6 java.lang.IllegalArgumentException:property [elasticsearch.version] is missing for plugin [head]

在 es 的配置文件中加：

```
http.cors.enabled: true
```

```
http.cors.allow-origin: "*" 
```

## 2.4 集群部署

### 2.4.1 配置节点

采用三台 CentOS7.3 部署 Elasticsearch 集群，需要不同的节点名和 IP

系统	节点名	IP 地址
CentOs6.5	node-1	192.168.127.122
CentOs6.5	node-2	192.168.127.123
CentOs6.5	node-3	192.168.127.124

### 2.4.2 修改配置文件

使用命令 `vim /opt/install/elasticsearch-6.5.3/config/elasticsearch.yml`

下面增加以下内容

```

bootstrap.memory_lock: false

bootstrap.system_call_filter: false

#绑定的ip地址
network.host: 192.168.127.122
##设置对外服务的http端口, 默认为9200
http.port: 9200
## 设置节点间交互的tcp端口,默认是9300
transport.tcp.port: 9300

#集群的名称
cluster.name: es6.2
##节点名称,其余两个节点分别为node-2 和node-3
node.name: node-1
##指定该节点是否有资格被选举成为master节点, 默认是true, es是默认集群中的第一台机器为master, 如果这台机挂了就会重新选举master
node.master: true
##允许该节点存储数据(默认开启)
node.data: true
##索引数据的存储路径
path.data: /opt/install/elasticsearch-6.5.3/data
##日志文件的存储路径
path.logs: /opt/install/elasticsearch-6.5.3/logs

discovery.zen.ping.unicast.hosts: ["192.168.127.122:9300", "192.168.127.123:9300", "192.168.127.124:9300"]
##如果没有这种设置,遭受网络故障的集群就有可能将集群分成两个独立的集群 - 分裂的大脑 - 这将导致数据丢失
discovery.zen.minimum_master_nodes: 3

```

### 2.4.3 集群配置

将该文件复制到不同虚拟机下, 设置对应的 ip 和节点名称 (不推荐)

按照单机配置依次安装 (推荐)

### 2.4.4 集群异常

```

[2019-03-11T01:44:14,104][INFO ][o.e.d.z.ZenDiscovery      ] [node-2] failed to send join request to master [{node-1}{TxXDw2xTROG37YTtmUno4Q}{_Nwfjv0cQY-yLSlcfgfmhaA}{192.168.127.122}{192.168.127.122:9300}{ml.machine_memory=1028517888, ml.max_open_jobs=20, xpack.installed=true, ml.enabled=true}], reason [RemoteTransportException[[node-1][192.168.127.122:9300][internal:discovery/zen/join]]; nested: NotMasterException[Node [{node-1}{TxXDw2xTROG37YTtmUno4Q}{_Nwfjv0cQY-yLSlcfgfmhaA}{192.168.127.122}{192.168.127.122:9300}{ml.machine_memory=1028517888, xpack.installed=true, ml.max_open_jobs=20, ml.enabled=true}] not master for join request]; ], tried [3] times

```

解决方法: 删除 data 目录下的 nodes 文件

```
rm -rf nodes
```

```
org.elasticsearch.bootstrap.StartupException: BindTransportException[Failed to bind to [9300]]; nested: BindException[Address already in use];
    at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:140) ~[elasticsearch-6.5.3.jar:6.5.3]
    at org.elasticsearch.bootstrap.Elasticsearch.execute(Elasticsearch.java:127) ~[elasticsearch-6.5.3.jar:6.5.3]
    at org.elasticsearch.cli.EnvironmentAwareCommand.execute(EnvironmentAwareCommand.java:86) ~[elasticsearch-6.5.3.jar:6.5.3]
    at org.elasticsearch.cli.Command.mainWithoutErrorHandling(Command.java:124) ~[elasticsearch-cli-6.5.3.jar:6.5.3]
```

原因：端口占用

解决方法：

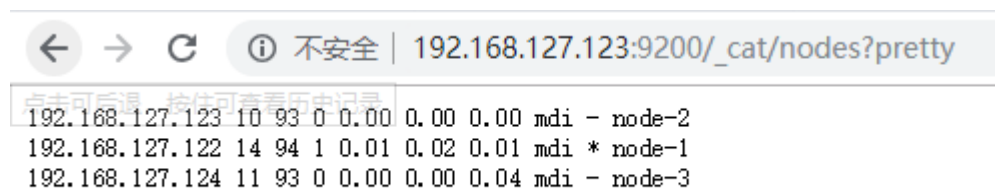
```
[root@localhost logs]# jps
2904 Jps
2796 Elasticsearch
[root@localhost logs]# kill -9 2796
```

## 2.5 测试

curl -X GET <http://192.168.127.122:9200/> （输入自己的 IP）

```
[root@localhost ~]# curl -X GET http://192.168.127.122:9200/
{
  "name" : "node-1",
  "cluster_name" : "es6.2",
  "cluster_uuid" : "GD1wh4zNTkqnGR-2VrcUbQ",
  "version" : {
    "number" : "6.5.3",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "159a78a",
    "build_date" : "2018-12-06T20:11:28.826501Z",
    "build_snapshot" : false,
    "lucene_version" : "7.5.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

浏览器输入: [http://192.168.127.122:9200/\\_cat/nodes?pretty](http://192.168.127.122:9200/_cat/nodes?pretty)



## 3.Kibana

### 3.1 安装 Kibana

注意: **ELK 下整套环境必须统一版本!!!**

将安装压缩包复制至/opt/install 目录下解压缩, 然后修改配置文件

kibana-6.5.3/config/kibana.yml

添加:

server.host: "192.168.127.122"

server.name: "your-hostname"

elasticsearch.url: <http://192.168.127.122:9200>

kibana.index: ".kibana"

```

# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "192.168.127.122"

# Enables you to specify a path to mount Kibana at if you are running behind a proxy.
# Use the `server.rewriteBasePath` setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#server.basePath: ""

# Specifies whether Kibana should rewrite requests that are prefixed with
# `server.basePath` or require that they are rewritten by your reverse proxy.
# This setting was effectively always `false` before Kibana 6.3 and will
# default to `true` starting in Kibana 7.0.
#server.rewriteBasePath: false

# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name. This is used for display purposes.
server.name: "your-hostname"

# The URL of the Elasticsearch instance to use for all your queries.
elasticsearch.url: "http://192.168.127.122:9200"

# When this setting's value is true Kibana uses the hostname specified in the server.host
# setting. When the value of this setting is false, Kibana uses the hostname of the host
# that connects to this Kibana instance.
#elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved searches, visualizations and
# dashboards. Kibana creates a new index if the index doesn't already exist.
kibana.index: ".kibana"

```

## 3.2 运行 Kibana

在 `/usr/local/kibana-5.5.2/bin` 目录下运行: `./kibana`

## 3.3 连接测试

Web 界面访问: <http://192.168.127.122:5601> , 如果此时需要输入用户名和密码登录,默认分别是 `elastic` 和 `changeme`



## 4.kafka

### 4.1 安装

### 4.2 配置 zookeeper

4.2.1 先通过配置文件，建立 zookeeper 集群，修改 config 下的 zookeeper.properties 文件：

```
dataDir=/opt/install/kafka_2.11-1.1.0/zookeeper
dataLogDir=/opt/install/kafka_2.11-1.1.0/log/zookeeper/
# the port at which the clients will connect
clientPort=2181
# disable the per-ip limit on the number of connections since this is a non-production config
maxClientCnxns=100
tickTime=2000
initLimit=10
syncLimit=5

server.1=192.168.127.122:2888:3888
server.2=192.168.127.123:2888:3888
server.3=192.168.127.124:2888:3888
```

注意：dataDir 路径与 dataLogDir 路径需要手动创建，配置文件不会自动生成

**4.2.2 进入 /opt/install/kafka\_2.11-1.1.0/zookeeper 创建 myid 文件，在三个服务器下分别写入 1，2，3.**

-----myid 是 zk 集群用来发现彼此的标识，必须创建，且不能相同；

## 4.3 搭建 kafka 集群

修改 server.properties 配置文件

```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0
```

对应三个服务器分别改成0, 1, 2

```
##### Socket Server Settings #####

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#   listeners = listener_name://host_name:port
#   EXAMPLE:
#   listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://192.168.127.122:9092
```

改成对应服务器的地址

```
##### Log Basics #####

# A comma separated list of directories under which to store log files
log.dirs=/opt/install/kafka_2.11-1.1.0/log/kafka
```

手动创建对应路径，日志生成地址

```
##### Zookeeper #####

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=192.168.127.122:2181,192.168.127.123:2181,192.168.127.124:2181
```

设置zookeeper集群路径

```
# Timeout in ms for connecting to zookeeper
zookeeper.connection.timeout.ms=6000
```

## 4.4 启动 kafka

启动 kafka 需要先启动 zookeeper，kafka 目录下输入

`./bin/zookeeper-server-start.sh config/zookeeper.properties &`

将三台服务器的 zookeeper 全部启动。



然后 kafka 目录下输入

```
./bin/kafka-server-start.sh config/server.properties &
```

三台服务器全部启动无报错即成功

## 4.5 创建 topic

创建一个叫做“test”的 topic，它只有一个分区，一个副本。

分区数量影响可以同时消费 topic 的用户数量，如果想要两个 logstash 同时消费同样的 topic 数据，要建立消费者组

```
bin/kafka-topics.sh --create --zookeeper 192.168.127.122:2181 --replication-factor 3 --partitions 3 --topic test
```

可以通过 list 命令查看创建的 topic:

```
bin/kafka-topics.sh --list --zookeeper 192.168.127.122:2181
```

## 4.6 生产者与消费者

模拟 kafka 的输入与输出功能，可以使用用来测试数据的联通

生产者: `bin/kafka-console-producer.sh --broker-list 192.168.127.122:9092 --topic test`

```
[root@localhost kafka_2.11-1.1.0]# bin/kafka-console-producer.sh --broker-list 192.168.127.122:9092 --topic test
>hello, is kafka
>this is test
>{"name":"nihao"}
>
```

消费者: `bin/kafka-console-consumer.sh --bootstrap-server 192.168.127.122:9092 --topic test`

```
[root@localhost kafka_2.11-1.1.0]# bin/kafka-console-consumer.sh --bootstrap-server 192.168.127.122:9092 --topic test
hello, is kafka
this is test
{"name":"nihao"}
```

## 5.Logstash

### 5.1 安装 Logstash

将安装压缩包复制至/opt/install 目录下解压缩，然后进入 bin 目录下 新增配置文件 logstash.conf，文件内容：

```
input {
  kafka {
    #kafka集群地址
    bootstrap_servers => "192.168.127.122:9092,192.168.127.123:9092,192.168.127.124:9092"
    #收集的topic的信息
    topics => ["test"]
    #不同的logstash配置不同的消费者组，实现一收多发
    group_id => "test-consumer-group"
    #消费线程数
    consumer_threads => 3
    #是否将kafka信息传输
    decorate_events => false
    #不存储为message，而是json
    codec => json
  }
}
```

```
filter{
}

output {
  elasticsearch {
    #ES的集群地址
    hosts => ["192.168.127.122:9200","192.168.127.123:9200","192.168.127.124:9200"]
    #传入的index名称格式
    index => "logstash-%{+YYYY.MM.dd}"
    #每个输出插件的工作work数量
    workers => 1
  }
  #输出到控制台
  stdout{
    codec => rubydebug
  }
}
```

注意：配置文件中不要写注释

## 5.2 运行 Logstash

在 bin 目录下运行 `./logstash -f logstash.conf`

配置 `stdout{codec => rubydebug}`, 所有通过 logstash 的数据都会在控制台显示

```
[2019-04-07T18:26:23,642][ERROR][logstash.codecs.json      ] JSON parse error, original data now in message field {:error:zed token 'hello': was expecting ('true', 'false' or 'null')}
at [Source: (String)"hello, is kafka"; line: 1, column: 6]>, :data=>"hello, is kafka"}
{
  "tags" => [
    [0] "_jsonparsefailure"
  ],
  "message" => "hello, is kafka",
  "@version" => "1",
  "@timestamp" => 2019-04-08T01:26:23.642Z
}
[2019-04-07T18:26:32,685][ERROR][logstash.codecs.json      ] JSON parse error, original data now in message field {:error:zed token 'this': was expecting 'null', 'true', 'false' or NaN
at [Source: (String)"this is test"; line: 1, column: 5]>, :data=>"this is test"}
{
  "tags" => [
    [0] "_jsonparsefailure"
  ],
  "message" => "this is test",
  "@version" => "1",
  "@timestamp" => 2019-04-08T01:26:32.686Z
}
{
  "name" => "nihao",
  "@version" => "1",
  "@timestamp" => 2019-04-08T01:28:36.223Z
}
```

注：配合 kafka 生产者来测试 logstash 联通

## 6.nginx (用来代替数据源测试)

### 6.1nginx 是什么

Nginx (engine x) 是一个高性能的 HTTP 和反向代理服务器，也是一个 IMAP/POP3/SMTP 服务器。Nginx 是由伊戈尔·赛索耶夫为俄罗斯访问量第二的 Rambler.ru 站点（俄文：Рамблер）开发的，第一个公开版本 0.1.0 发布于 2004 年 10 月 4 日。它解决了服务器的 C10K（就是在一秒之内连接客户端的数目为 10k 即 1 万）问题。它的设计不像传统的服务器那样使用线程处理请求，而是一个更加高级的机制—事件驱动机制，是一种异步事件驱动结构。

## 6.2 nginx 下载和安装

### 6.2.1 下载源文件

```
cd /opt/install    切换到指定目录中
```

```
wget http://nginx.org/download/nginx-1.10.0.tar.gz    下载 nginx
```

```
tar -zxvf nginx-1.10.0.tar.gz    解压
```

### 6.2.2 安装依赖项

```
yum -y install gcc pcre pcre-devel zlib zlib-devel openssl openssl-devel    安装依赖项
```

### 6.2.3 配置 nginx 安装选项

这里只配置安装到/opt/install 目录下，其它选项可执行./configuration - help  
查看

```
cd /opt/install/nginx-1.10.0    跳转到解压后的目录
```

```
./configure --prefix=/opt/install
```

### 6.2.4 编译并安装

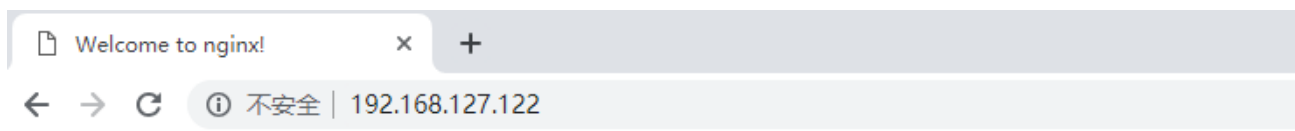
```
make && make install
```

## 6.3 启动并访问

```
./nginx 开启 nginx,
```

192.168.127.122:80 登录对应 IP，端口号默认为 80

出现页面



## 7.filebeat

### 7.1 安装

安装解压方式同 Logstash，然后修改配置文件

### 7.2 修改配置文件

```
#===== Filebeat inputs =====  
  
filebeat.inputs:  
  
# Each - is an input. Most options can be set at the input level, so  
# you can use different inputs for various configurations.  
# Below are the input specific configurations.  
  
- type: log  
  
  # Change to true to enable this input configuration.  
  enabled: true  
  
  # Paths that should be crawled and fetched. Glob based paths.  
  paths:  
    - /usr/local/nginx/logs/access.log  
  
  fields_under_root: true
```

Paths 路径为 nginx 日志存储路径

```
#===== Kibana =====
# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:
  host: "192.168.127.122:5601"
# Kibana Host

#----- kafka output -----
output.kafka:
# initial brokers for reading cluster metadata
  hosts: ["192.168.127.122:9092", "192.168.127.122:9092", "192.168.127.122:9092"]

# message topic selection + partitioning
  topic: '%[type]%'
  partition.round_robin:
    reachable_only: false
  required_acks: 1
  compression: gzip
  max_message_bytes: 1000000
```

输出为 kafka 集群

注意: topic: :test'(可以统一设置)

## 7.3 运行

在 filebeat 的安装目录下 执行

```
./filebeat -e -c filebeat.yml -d "publish"
```

## 7.4 测试

当 nginx 打开并且页面刷新时

控制台会接收到 nginx 的日志并且打印出来

```

2019-03-11T23:57:04.659-0700 DEBUG [publish] pipeline/processor.go:308 Publish event: {
  "@timestamp": "2019-03-12T06:57:04.650Z",
  "@metadata": {
    "beat": "filebeat",
    "type": "doc",
    "version": "6.5.3"
  },
  "source": "/usr/local/nginx/logs/access.log",
  "offset": 2351,
  "message": "192.168.127.1 - - [11/Mar/2019:23:57:02 -0700] \"GET / HTTP/1.1\" 304 0 \"-\" \"Mozilla/5.0 (Windows
  Gecko) Chrome/72.0.3626.121 Safari/537.36\"",
  "input": {
    "type": "log"
  },
  "prospector": {
    "type": "log"
  },
  "beat": {
    "name": "localhost.localdomain",
    "hostname": "localhost.localdomain",
    "version": "6.5.3"
  },
  "host": {
    "name": "localhost.localdomain",
    "architecture": "x86_64",
    "os": {
      "platform": "centos",
      "version": "6.5 (Final)",
      "family": "redhat",
      "codename": "Final"
    },
    "containerized": true
  }
}

```

## 8.fluentd 的安装与交互 kafka

**注意：**Fluentd 在架构中与 filebeat 起到同样的收集传输作用

### 8.1 安装 fluentd

```
curl -L https://toolbelt.treasuredata.com/sh/install-redhat-td-agent2.sh | sh
```

### 8.2 安装中异常

#### 8.2.1 culr: (35) SSL connect error

```

[root@localhost opt]# curl -L https://toolbelt.treasuredata.com/sh/install-redhat-td-agent2.sh | sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total   Spent    Left   Speed
 0 <        0    0    0    0    0      0     0  --:--:-- --:--:-- --:--:--    0
curl: (35) SSL connect error

```

无法在服务器使用 curl 命令访问 https 域名,原因是 nss 版本有点旧了

yum -y update nss 更新一下

## 8.2.2 You could try using --skip-broken to work around the problem

```
Error: Package: gstreamer-plugins-bad-0.10.19-3.el6.rf.x86_64 (@rpmforge)
Requires: libmodplug.so.0()(64bit)
Removing: libmodplug-0.8.7-1.el6.rf.x86_64 (@rpmforge)
libmodplug.so.0()(64bit)
Updated By: 1:libmodplug-0.8.8.5-1.el6.x86_64 (epel)
Not found
You could try using --skip-broken to work around the problem
You could try running: rpm -Va --nofiles --nodigest
```

### 解决方法:

1. 进入 yum 源配置文件 `cd /etc/yum.repos.d`
2. 备份一下当前的源,以防出错后可以还原回来 `mv ./CentOS-Base.repo ./CentOS-Base.repo.bak`
3. 下载网易 163 的源 `wget http://mirrors.163.com/.help/CentOS7-Base-163.repo`
4. 清理一下旧包 `yum clean all`
5. 把下载下来文件 `CentOS7-Base-163.repo` 设置成为默认源 `mv CentOS7-Base-163.repo CentOS-Base.repo`
6. 生成缓存即可用 163 源了 `yum makecache`

```
[root@localhost ~]# yum install php
已加载插件: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
正在解决依赖关系
--> 正在检查事务
---> 软件包 php.x86_64.0.5.4.16-42.el7 将被 升级
---> 软件包 php.x86_64.0.5.4.16-43.el7_4 将被 更新
--> 正在处理依赖关系 php-common(x86-64) = 5.4.16-43.el7_4, 它被软件包 php-5.4.16-43.el7_4.x86_64 需要
--> 正在处理依赖关系 php-cli(x86-64) = 5.4.16-43.el7_4, 它被软件包 php-5.4.16-43.el7_4.x86_64 需要
--> 正在检查事务
---> 软件包 php-cli.x86_64.0.5.4.16-42.el7 将被 升级
---> 软件包 php-cli.x86_64.0.5.4.16-43.el7_4 将被 更新
---> 软件包 php-common.x86_64.0.5.4.16-42.el7 将被 升级
--> 正在处理依赖关系 php-common(x86-64) = 5.4.16-42.el7, 它被软件包 php-pdo-5.4.16-42.el7.x86_64 需要
--> 正在处理依赖关系 php-common(x86-64) = 5.4.16-42.el7, 它被软件包 php-process-5.4.16-42.el7.x86_64 需要
--> 正在处理依赖关系 php-common(x86-64) = 5.4.16-42.el7, 它被软件包 php-xml-5.4.16-42.el7.x86_64 需要
--> 正在处理依赖关系 php-common(x86-64) = 5.4.16-42.el7, 它被软件包 php-gd-5.4.16-42.el7.x86_64 需要
---> 软件包 php-common.x86_64.0.5.4.16-43.el7_4 将被 更新
--> 正在检查事务
---> 软件包 php-gd.x86_64.0.5.4.16-42.el7 将被 升级
---> 软件包 php-gd.x86_64.0.5.4.16-43.el7_4 将被 更新
---> 软件包 php-pdo.x86_64.0.5.4.16-42.el7 将被 升级
---> 软件包 php-pdo.x86_64.0.5.4.16-43.el7_4 将被 更新
---> 软件包 php-process.x86_64.0.5.4.16-42.el7 将被 升级
---> 软件包 php-process.x86_64.0.5.4.16-43.el7_4 将被 更新
---> 软件包 php-xml.x86_64.0.5.4.16-42.el7 将被 升级
---> 软件包 php-xml.x86_64.0.5.4.16-43.el7_4 将被 更新
--> 解决依赖关系完成
```



## 8.3 配置 kafka

注：<source>为配置读取，<match>为配置输出

配置 source 为读取 nginx 日志（测试用）

```
<source>
  type tail
  format /^(?<remote>[^\s]*) (?<host>[-.]) (?<user>[^\s]*) \[(?<time>[^\]]*)\] "(?<method>\S+)(?: +(?<path>[^\s]*) +\S*)?(?<code>[^\s]*) (?<size>[^\s]*) "(?<referer>[^\s]*)" "(?<agent>[^\s]*)"/
  path /usr/local/nginx/logs/access.log
  pos_file /usr/local/nginx/logs/access.log.pos
  tag nginx.tag
  time_key time
  time_format %d/%b/%Y:%H:%M:%S %z
</source>
```

```
<source>
  type tail
  format /^(?<remote>[^\s]*) (?<host>[-.]) (?<user>[^\s]*) \[(?<time>[^\]]*)\] "(?<method>\S+)(?: +(?<path>[^\s]*) +\S*)?(?<code>[^\s]*) (?<size>[^\s]*) "(?<referer>[^\s]*)" "(?<agent>[^\s]*)"/
  path /usr/local/nginx/logs/access.log
  pos_file /usr/local/nginx/logs/access.log.pos
  tag nginx.tag
  time_key time
  time_format %d/%b/%Y:%H:%M:%S %z
</source>
```

配置输出到 kafka

```
<match *.*>
@type kafka
brokers 192.168.127.122:9092,192.168.127.124:9092,192.168.127.123:9092
#zookeeper 192.168.127.122:2181
default_topic test
output_data_type json
</match>
```

```
<match *.*>
@type kafka
brokers 192.168.127.122:9092,192.168.127.124:9092,192.168.127.123:9092
#zookeeper 192.168.127.122:2181
default_topic test
output_data_type json
</match>
```

因为没有下载 zookeeper 插件 所以直接输出到 kafka

## 8.4 重要：fluentd 的多发 copy 插件

```
<match myevent.file_and_elasticsearch>
  @type copy
  <store>
    @type file
    path /var/log/fluent/myapp
    compress gzip
    <format>
      localtime false
    </format>
    <buffer time>
      timekey_wait 10m
      timekey 86400
      timekey_use_utc true
      path /var/log/fluent/myapp
    </buffer>
    <inject>
      time_format %Y%m%dT%H%M%S%z
      localtime false
    </inject>
  </store>
  <store>
    @type elasticsearch
    host fluentd
    port 9200
    index_name fluentd
    type_name fluentd
  </store>
</match>
```

使用 store 标签分割，是内置的插件，不需要下载，在同一 match 标签下配置不同的 store，多次发送

## 8.5 测试 kafka 消费者

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 -
--topic test --from-beginning
```

创建消费者并且刷新 nginx 产生日志

```
{
  "remote": "192.168.127.1",
  "host": "-",
  "user": "-",
  "method": "GET",
  "path": "/",
  "code": "304",
  "size": "0",
  "referer": "-",
  "agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36"
}
{"remote": "192.168.127.1",
 "host": "-",
 "user": "-",
 "method": "GET",
 "path": "/",
 "code": "304",
 "size": "0",
 "referer": "-",
 "agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36"
}
{"remote": "192.168.127.1",
 "host": "-",
 "user": "-",
 "method": "GET",
 "path": "/",
 "code": "304",
 "size": "0",
 "referer": "-",
 "agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36"
}
{"remote": "192.168.127.1",
 "host": "-",
 "user": "-",
 "method": "GET",
 "path": "/",
 "code": "304",
 "size": "0",
 "referer": "-",
 "agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36"
}
{"remote": "192.168.127.1",
 "host": "-",
 "user": "-",
 "method": "GET",
 "path": "/",
 "code": "304",
 "size": "0",
 "referer": "-",
 "agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36"
}
{"remote": "192.168.127.1",
 "host": "-",
 "user": "-",
 "method": "GET",
 "path": "/",
 "code": "304",
 "size": "0",
 "referer": "-",
 "agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36"
}
{"remote": "192.168.127.1",
 "host": "-",
 "user": "-",
 "method": "GET",
 "path": "/",
 "code": "304",
 "size": "0",
 "referer": "-",
 "agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36"
}
```

获取日志传输成功！

## 9.测试完整数据联通

数据流向:nginx(应用产生日志)

=> filebeat/fluend(收集日志并传输)

=> kafka(消息队列，高并发高可用)

=> logstash (抓取 kafka 中的日志，导入 ELK 系统)

=> elasticsearch (储存数据，提供查询)

=> kibana/elasticsearch-head (提供 ES 的可视化)

### 9.1 nginx 访问

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

## 9.2 filebeat 接收成功

```
2019-04-07T19:09:18.366-0700 DEBUG [publish] pipeline/processor.go:308 Publish event: {
  "@timestamp": "2019-04-08T02:09:18.307Z",
  "@metadata": {
    "beat": "filebeat",
    "type": "doc",
    "version": "6.5.3"
  },
  "message": "192.168.127.1 - - [07/Apr/2019:19:09:15 -0700] \"GET / HTTP/1.1\" 304 0 \"-\" \"Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36\"",
  "input": {
    "type": "log"
  },
  "prospector": {
    "type": "log"
  },
  "host": {
    "name": "localhost.localdomain",
    "containerized": true,
    "architecture": "x86_64",
    "os": {
      "version": "6.5 (Final)",
      "family": "redhat",
      "codename": "Final",
      "platform": "centos"
    }
  },
  "beat": {
    "version": "6.5.3",
    "name": "localhost.localdomain",
    "hostname": "localhost.localdomain"
  },
  "source": "/usr/local/nginx/logs/access.log",
  "offset": 28543
}
```

激活 Windows

## 9.3 kafka 消费者成功消费

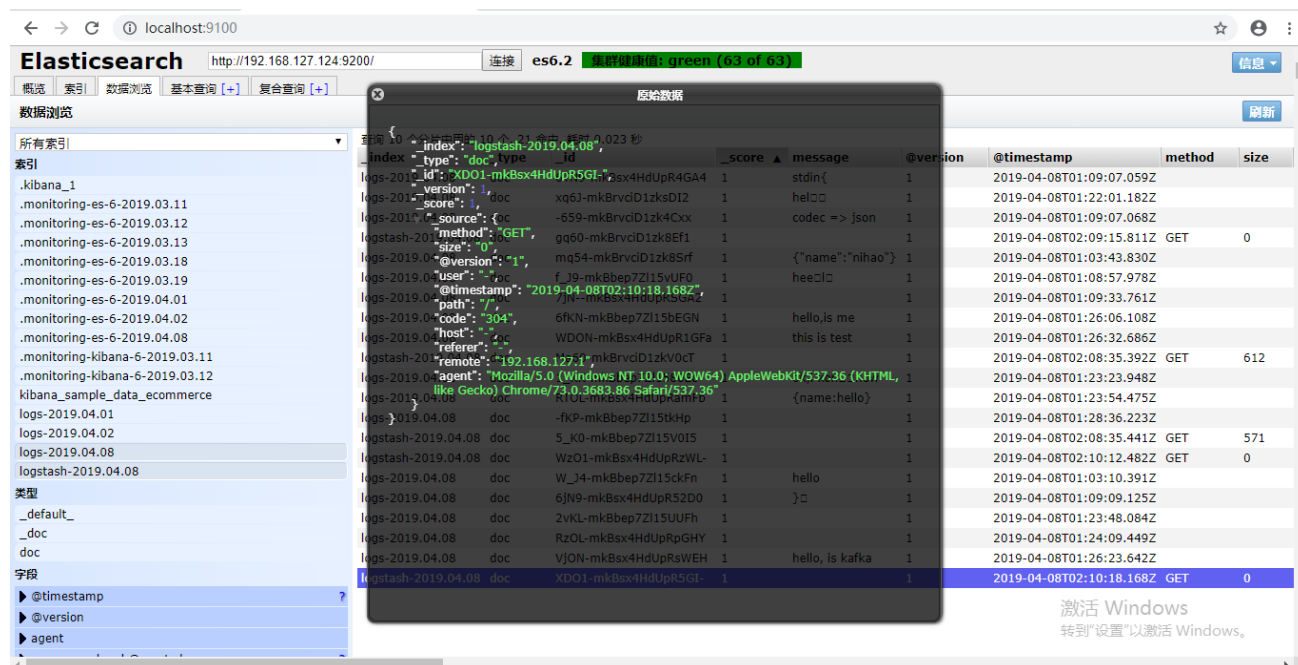
```
{
  "remote": "192.168.127.1",
  "host": ".",
  "user": "-",
  "method": "GET",
  "path": "/",
  "code": "304",
  "size": "0",
  "referer": "-",
  "agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36"
}
{"@timestamp":"2019-04-08T02:10:20.377Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.5.3","topic":"test"},"message":"192.168.127.1 - - [07/Apr/2019:19:10:18 -0700] \"GET / HTTP/1.1\" 304 0 \"-\" \"Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36\"",
  "prospector":{"type":"log"},"input":{"type":"log"},"beat":{"name":"localhost.localdomain","hostname":"localhost.localdomain","version":"6.5.3"},"host":{"name":"localhost.localdomain","architecture":"x86_64","os":{"platform":"centos","version":"6.5 (Final)","family":"redhat","codename":"Final","containerized":true},"source":"/usr/local/nginx/logs/access.log","offset":28915}}
```

## 9.4 logstash 输出成功

```
[2019-04-07T19:10:21.527][WARN ][logstash.outputs.elasticsearch] Could not index event to Elasticsearch. {:status=>400, :action=>[\"index\", {:_id=>nil, :_index=>\"logstash-2019.04.08\", :_type=>\"doc\", :routing=>nil}, #<LogStash::Event:0x49a3a5a1>], :response=>{\"index=>{\"_index\"=>\"logstash-2019.04.08\", \"_type\"=>\"doc\", \"_id\"=>\"8_k1-mkBep7z1L58ELD\", \"status\"=>400, \"error\"=>{\"type\"=>\"mapper_parsing_exception\", \"reason\"=>\"failed to parse field [host] of type [text]\", \"caused_by\"=>{\"type\"=>\"illegal_state_exception\", \"reason\"=>\"Can't get text on a START_OBJECT at 1:53\"}}}}}]
{
  "source" => "/usr/local/nginx/logs/access.log",
  "host" => {
    "architecture" => "x86_64",
    "containerized" => true,
    "name" => "localhost.localdomain",
    "os" => {
      "codename" => "Final",
      "version" => "6.5 (Final)",
      "platform" => "centos",
      "family" => "redhat"
    }
  },
  "offset" => 28915,
  "@version" => "1",
  "prospector" => {
    "type" => "log"
  },
  "input" => {
    "type" => "log"
  },
  "@timestamp" => 2019-04-08T02:10:20.377Z,
  "message" => "192.168.127.1 - - [07/Apr/2019:19:10:18 -0700] \"GET / HTTP/1.1\" 304 0 \"-\" \"Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36\"",
  "beat" => {
    "name" => "localhost.localdomain",
    "hostname" => "localhost.localdomain",
    "version" => "6.5.3"
  }
}
```

激活 Windows  
转到“设置”以激活 Windows

## 9.5 可视化界面查询 ES 中数据



注：界面为 elasticsearch-head 插件，文档中使用为 kibana