参考：<http://blog.oldboyedu.com/elk/>

**最好用的日志分析工具ELK**

* **1年前 (2016-02-01)**
* [**老男孩**](http://blog.oldboyedu.com/author/oldboyoldboy/)
* [**Linux服务**](http://blog.oldboyedu.com/category/linux-service/)**,**[**学员风采**](http://blog.oldboyedu.com/category/student-style/)**,**[**日志**](http://blog.oldboyedu.com/category/linux-service/log/)
* [**3评论**](http://blog.oldboyedu.com/elk/#comments)
* **来源：本站原创**
* **2476℃**
* **字体：**[**小**](javascript:doZoom(16))[**中**](javascript:doZoom(20))[**大**](javascript:doZoom(24))

**一、ELK介绍**

**1.1 elasticsearch**

**1.1.1 elasticsearch介绍**

ElasticSearch是一个基于Lucene的搜索服务器。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，是第二流行的企业搜索引擎。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。

**1.1.2 elasticsearch几个重要术语**

* NRT（nearly real time）   
  elasticsearch是一个近似实时的搜索平台，从索引文档到可搜索有些延迟，通常为1秒。
* 集群   
  集群就是一个或多个节点存储数据，其中一个节点为主节点，这个主节点是可以通过选举产生的，并提供跨节点的联合索引和搜索的功能。集群有一个唯一性标示的名字，默认是elasticsearch，集群名字很重要，每个节点是基于集群名字加入到其集群中的。因此，确保在不同环境中使用不同的集群名字。一个集群可以只有一个节点。强烈建议在配置elasticsearch时，配置成集群模式。
* 节点   
  节点就是一台单一的服务器，是集群的一部分，存储数据并参与集群的索引和搜索功能。像集群一样，节点也是通过名字来标识，默认是在节点启动时随机分配的字符名。当然啦，你可以自己定义。该名字也蛮重要的，在集群中用于识别服务器对应的节点。   
  节点可以通过指定集群名字来加入到集群中。默认情况下，每个节点被设置成加入到elasticsearch集群。如果启动了多个节点，假设能自动发现对方，他们将会自动组建一个名为elasticsearch的集群。
* 索引   
  索引是有几分相似属性的一系列文档的集合。如nginx日志索引、syslog索引等等。索引是由名字标识，名字必须全部小写。这个名字用来进行索引、搜索、更新和删除文档的操作。   
  索引相对于关系型数据库的库。
* 类型   
  在一个索引中，可以定义一个或多个类型。类型是一个逻辑类别还是分区完全取决于你。通常情况下，一个类型被定义成具有一组共同字段的文档。如ttlsa运维生成时间所有的数据存入在一个单一的名为logstash-ttlsa的索引中，同时，定义了用户数据类型，帖子数据类型和评论类型（me：便于搜索和统计）。   
  类型相对于关系型数据库的表。
* 文档   
  文档是信息的基本单元，可以被索引的。文档是以JSON格式表现的。   
  在类型中，可以根据需求存储多个文档。   
  虽然一个文档在物理上位于一个索引，实际上一个文档必须在一个索引内被索引和分配一个类型。   
  文档相对于关系型数据库的列。
* 分片和副本   
  在实际情况下，索引存储的数据可能超过单个节点的硬件限制。如一个十亿文档需1TB空间可能不适合存储在单个节点的磁盘上，或者从单个节点搜索请求太慢了。为了解决这个问题，elasticsearch提供将索引分成多个分片的功能。当在创建索引时，可以定义想要分片的数量。每一个分片就是一个全功能的独立的索引，可以位于集群中任何节点上。   
  分片的两个最主要原因：   
  a、水平分割扩展，增大存储量   
  b、分布式并行跨分片操作，提高性能和吞吐量   
  分布式分片的机制和搜索请求的文档如何汇总完全是有elasticsearch控制的，这些对用户而言是透明的。   
  网络问题等等其它问题可以在任何时候不期而至，为了健壮性，强烈建议要有一个故障切换机制，无论何种故障以防止分片或者节点不可用。   
  为此，elasticsearch让我们将索引分片复制一份或多份，称之为分片副本或副本（me：副本也是分片的副本，es是基于分片来查找的）。   
  副本也有两个最主要原因：   
  1、高可用性，以应对分片或者节点故障。出于这个原因，分片副本要在不同的节点上。   
  2、提供性能，增大吞吐量，搜索可以并行在所有副本上执行。   
   总之，每一个索引可以被分成多个分片。索引也可以有0个或多个副本。复制后，每个索引都有主分片(母分片)和复制分片(复制于母分片)。分片和副本数量可以在每个索引被创建时定义。索引创建后，可以在任何时候动态的更改副本数量，但是，不能改变分片数。   
  默认情况下，elasticsearch为每个索引分片5个主分片和1个副本，这就意味着集群至少需要2个节点。索引将会有5个主分片和5个副本(1个完整副本)，每个索引总共有10个分片。   
  每个elasticsearch分片是一个Lucene索引。一个单个Lucene索引有最大的文档数LUCENE-5843, 文档数限制为2147483519(MAX\_VALUE – 128)。 可通过\_cat/shards来监控分片大小。

**1.2 logstash**

**1.2.1 logstash 介绍**

LogStash由JRuby语言编写，基于消息（message-based）的简单架构，并运行在Java虚拟机（JVM）上。不同于分离的代理端（agent）或主机端（server），LogStash可配置单一的代理端（agent）与其它开源软件结合，以实现不同的功能。

**1.2.2 logStash的四大组件**

* Shipper：发送事件（events）至LogStash；通常，远程代理端（agent）只需要运行这个组件即可；
* Broker and Indexer：接收并索引化事件；
* Search and Storage：允许对事件进行搜索和存储；
* Web Interface：基于Web的展示界面   
  正是由于以上组件在LogStash架构中可独立部署，才提供了更好的集群扩展性。

**1.2.2 LogStash主机分类**

* 代理主机（agent host）：作为事件的传递者（shipper），将各种日志数据发送至中心主机；只需运行Logstash 代理（agent）程序；
* 中心主机（central host，me：应该指的是包含elasticsearch 和 kibana的套件）：可运行包括中间转发器（Broker）、索引器（Indexer）、搜索和存储器（Search and Storage）、Web界面端（Web Interface）在内的各个组件，以实现对日志数据的接收、处理和存储。

**1.3 kibana**

Logstash是一个完全开源的工具，他可以对你的日志进行收集、分析，并将其存储供以后使用（如，搜索），您可以使用它。说到搜索，logstash带有一个web界面，搜索和展示所有日志。

**二、使用ELK必要性（解决运维痛点）**

* 开发人员不能登录线上服务器查看详细日志
* 各个系统都有日志，日至数据分散难以查找
* 日志数据量大，查询速度慢，或者数据不够实时

**三、elk部署之环境准备**

**3.1 机器准备**

两台虚拟机：   
hostname：linux-node1和linux-node2   
ip地址：192.168.56.11和192.168.56.22

**3.2 系统环境（两台完全一致）**

1. [root@linux-node2 ~]# cat /etc/redhat-release
2. CentOS Linux release 7.1.1503 (Core)
3. [root@linux-node2 ~]# uname -a
4. Linux linux-node2 3.10.0-229.el7.x86\_64 #1 SMP Fri Mar 6 11:36:42 UTC 2015 x86\_64 x86\_64 x86\_64 GNU/Linux
5. [root@linux-node2 ~]# cat /etc/hosts
6. 127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
7. ::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
8. 192.168.56.11 linux-node1.oldboyedu.com linux-node1
9. 192.168.56.12 linux-node2.oldboyedu.com linux-node2

**3.3 elk准备环境（两台完全一致）**

**3.3.1 elasticsearch安装**

下载并安装GPG key

1. [root@linux-node2 ~]# rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch

添加yum仓库

1. [root@linux-node2 ~]# vim /etc/yum.repos.d/elasticsearch.repo
2. [elasticsearch-2.x]
3. name=Elasticsearch repository for 2.x packages
4. baseurl=http://packages.elastic.co/elasticsearch/2.x/centos
5. gpgcheck=1
6. gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
7. enabled=1

安装elasticsearch

1. [root@hadoop-node2 ~]# yum install -y elasticsearch

**3.3.2 logstash安装**

下载并安装GPG key

1. [root@linux-node2 ~]# rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch

添加yum仓库

1. [root@linux-node2 ~]# vim /etc/yum.repos.d/logstash.repo
2. [logstash-2.1]
3. name=Logstash repository for 2.1.x packages
4. baseurl=http://packages.elastic.co/logstash/2.1/centos
5. gpgcheck=1
6. gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
7. enabled=1

安装logstash

1. [root@linux-node2 ~]# yum install -y logstash

安装kibana

1. [root@linux-node2 ~]#cd /usr/local/src
2. [root@linux-node2 ~]#wget https://download.elastic.co/kibana/kibana/kibana-4.3.1-linux-x64.tar.gz
3. tar zxf kibana-4.3.1-linux-x64.tar.gz
4. [root@linux-node1 src]# mv kibana-4.3.1-linux-x64 /usr/local/
5. [root@linux-node2 src]# ln -s /usr/local/kibana-4.3.1-linux-x64/ /usr/local/kibana

安装Redis，nginx和java

1. [root@linux-node2 ~]#yum install -y redis nginx java

**四、管理配置elasticsearch**

**4.1 管理linux-node1的elasticsearch**

修改elasticsearch配置文件，并授权

1. [root@linux-node1 src]# grep -n '^[a-Z]' /etc/elasticsearch/elasticsearch.yml
2. 17:cluster.name: chuck-cluster 判别节点是否是统一集群
3. 23:node.name: linux-node1 节点的hostname
4. 33:path.data: /data/es-data 数据存放路径
5. 37:path.logs: /var/log/elasticsearch/ 日志路径
6. 43:bootstrap.mlockall: true 锁住内存，使内存不会再swap中使用
7. 54:network.host: 0.0.0.0 允许访问的ip
8. 58:http.port: 9200 端口
9. [root@linux-node1 ~]# mkdir -p /data/es-data
10. [root@linux-node1 src]# chown elasticsearch.elasticsearch /data/es-data/

启动elasticsearch

1. [root@linux-node1 src]# systemctl start elasticsearch
2. [root@linux-node1 src]# systemctl enable elasticsearch
3. ln -s '/usr/lib/systemd/system/elasticsearch.service' '/etc/systemd/system/multi-user.target.wants/elasticsearch.service'
4. [root@linux-node1 src]# systemctl status elasticsearch
5. elasticsearch.service - Elasticsearch
6. Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled)
7. Active: active (running) since Thu 2016-01-14 09:30:25 CST; 14s ago
8. Docs: http://www.elastic.co
9. Main PID: 37954 (java)
10. CGroup: /system.slice/elasticsearch.service
11. └─37954 /bin/java -Xms256m -Xmx1g -Djava.awt.headless=true -XX:+UseParNewGC -XX:+UseConc...
12. Jan 14 09:30:25 linux-node1 systemd[1]: Starting Elasticsearch...
13. Jan 14 09:30:25 linux-node1 systemd[1]: Started Elasticsearch.
14. [root@linux-node1 src]# netstat -lntup|grep 9200
15. tcp6 0 0 :::9200 :::\* LISTEN 37954/java

访问9200端口，会把信息显示出来 

**4.2 elasticsearch进行交互**

**4.2.1 交互的两种方法**

* Java API ：   
  node client   
  Transport client
* RESTful API   
  Javascript   
  .NET   
  php   
  Perl   
  Python   
  Ruby

**4.2.2使用RESTful API进行交互**

查看当前索引和分片情况，稍后会有插件展示

1. [root@linux-node1 src]# curl -i -XGET 'http://192.168.56.11:9200/\_count?pretty' -d '{
2. "query" {
3. "match\_all": {}
4. }
5. }'
6. HTTP/1.1 200 OK
7. Content-Type: application/json; charset=UTF-8
8. Content-Length: 95
9. {
10. "count" : 0, 索引0个
11. "\_shards" : { 分区0个
12. "total" : 0,
13. "successful" : 0, 成功0个
14. "failed" : 0 失败0个
15. }
16. }

使用head插件显示索引和分片情况

1. [root@linux-node1 src]# /usr/share/elasticsearch/bin/plugin install mobz/elasticsearch-head

在插件中添加一个index-demo/test的索引，提交请求   
   
发送一个GET（当然可以使用其他类型请求）请求，查询上述索引id   
  
在基本查询中查看所建索引 

**4.2管理linux-node2的elasticsearch**

将linux-node1的配置文件拷贝到linux-node2中,并修改配置文件并授权   
配置文件中cluster.name的名字一定要一致，当集群内节点启动的时候，默认使用组播（多播），寻找集群中的节点

1. [root@linux-node1 src]# scp /etc/elasticsearch/elasticsearch.yml 192.168.56.12:/etc/elasticsearch/elasticsearch.yml
2. [root@linux-node2 elasticsearch]# sed -i '23s#node.name: linux-node1#node.name: linux-node2#g' elasticsearch.yml
3. [root@linux-node2 elasticsearch]# mkdir -p /data/es-data
4. [root@linux-node2 elasticsearch]# chown elasticsearch.elasticsearch /data/es-data/

启动elasticsearch

1. [root@linux-node2 elasticsearch]# systemctl enable elasticsearch.service
2. ln -s '/usr/lib/systemd/system/elasticsearch.service' '/etc/systemd/system/multi-user.target.wants/elasticsearch.service'
3. [root@linux-node2 elasticsearch]# systemctl start elasticsearch.service
4. [root@linux-node2 elasticsearch]# systemctl status elasticsearch.service
5. elasticsearch.service - Elasticsearch
6. Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled)
7. Active: active (running) since Thu 2016-01-14 02:56:35 CST; 4s ago
8. Docs: http://www.elastic.co
9. Process: 38519 ExecStartPre=/usr/share/elasticsearch/bin/elasticsearch-systemd-pre-exec (code=exited, status=0/SUCCESS)
10. Main PID: 38520 (java)
11. CGroup: /system.slice/elasticsearch.service
12. └─38520 /bin/java -Xms256m -Xmx1g -Djava.awt.headless=true -XX:+UseParNewGC -XX:+UseConc...
13. Jan 14 02:56:35 linux-node2 systemd[1]: Starting Elasticsearch...
14. Jan 14 02:56:35 linux-node2 systemd[1]: Started Elasticsearch.

在linux-node2配置中添加如下内容，使用单播模式(尝试了使用组播，但是不生效)

1. [root@linux-node1 ~]# grep -n "^discovery" /etc/elasticsearch/elasticsearch.yml
2. 79:discovery.zen.ping.unicast.hosts: ["linux-node1", "linux-node2"]
3. [root@linux-node1 ~]# systemctl restart elasticsearch.service

在浏览器中查看分片信息，一个索引默认被分成了5个分片，每份数据被分成了五个分片（可以调节分片数量），下图中外围带绿色框的为主分片，不带框的为副本分片，主分片丢失，副本分片会复制一份成为主分片，起到了高可用的作用，主副分片也可以使用负载均衡加快查询速度，但是如果主副本分片都丢失，则索引就是彻底丢失。 

**4.3使用kopf插件监控elasticsearch**

1. [root@linux-node1 bin]# /usr/share/elasticsearch/bin/plugin install lmenezes/elasticsearch-kopf

从下图可以看出节点的负载，cpu适应情况，java对内存的使用（heap usage），磁盘使用，启动时间   
  
除此之外，kopf插件还提供了REST API 等，类似kopf插件的还有bigdesk，但是bigdesk目前还不支持2.1!！！安装bigdesk的方法如下

1. /usr/share/elasticsearch/bin/plugin install lukas-vlcek/bigdesk

**4.4node间组播通信和分片**

当第一个节点启动，它会组播发现其他节点，发现集群名字一样的时候，就会自动加入集群。随便一个节点都是可以连接的，并不是主节点才可以连接，连接的节点起到的作用只是汇总信息展示   
   
最初可以自定义设置分片的个数，分片一旦设置好，就不可以改变。主分片和副本分片都丢失，数据即丢失，无法恢复，可以将无用索引删除。有些老索引或者不常用的索引需要定期删除，否则会导致es资源剩余有限，占用磁盘大，搜索慢等。如果暂时不想删除有些索引，可以在插件中关闭索引，就不会占用内存了。 

**五、配置logstash**

**5.1循序渐进学习logstash**

启动一个logstash,-e：在命令行执行；input输入，stdin标准输入，是一个插件；output输出，stdout：标准输出

1. [root@linux-node1 bin]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { stdout{} }' Settings: Default filter workers: 1
2. Logstash startup completed
3. chuck ==>输入
4. 2016-01-14T06:01:07.184Z linux-node1 chuck ==>输出
5. www.chuck-blog.com ==>输入
6. 2016-01-14T06:01:18.581Z linux-node1 www.chuck-blog.com ==>输出

使用rubudebug显示详细输出，codec为一种编解码器

1. [root@linux-node1 bin]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { stdout{ codec => rubydebug} }'
2. Settings: Default filter workers: 1
3. Logstash startup completed
4. chuck ==>输入
5. {
6. "message" => "chuck",
7. "@version" => "1",
8. "@timestamp" => "2016-01-14T06:07:50.117Z",
9. "host" => "linux-node1"
10. } ==>使用rubydebug输出

上述每一条输出的内容称为一个事件，多个相同的输出的内容合并到一起称为一个事件（举例：日志中连续相同的日志输出称为一个事件）！   
使用logstash将信息写入到elasticsearch

1. [root@linux-node1 bin]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { elasticsearch { hosts => ["192.168.56.11:9200"] } }'
2. Settings: Default filter workers: 1
3. Logstash startup completed
4. maliang
5. chuck
6. chuck-blog.com
7. www.chuck-bllog.com

在elasticsearch中查看logstash新加的索引   
  
  
在elasticsearch中写一份，同时在本地输出一份，也就是在本地保留一份文本文件，也就不用在elasticsearch中再定时备份到远端一份了。此处使用的保留文本文件三大优势：1）文本最简单 2）文本可以二次加工 3）文本的压缩比最高

1. [root@linux-node1 bin]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { elasticsearch { hosts => ["192.168.56.11:9200"] } stdout{ codec => rubydebug } }'
2. Settings: Default filter workers: 1
3. Logstash startup completed
4. www.google.com
5. {
6. "message" => "www.google.com",
7. "@version" => "1",
8. "@timestamp" => "2016-01-14T06:27:49.014Z",
9. "host" => "linux-node1"
10. }
11. www.elastic.co
12. {
13. "message" => "www.elastic.co",
14. "@version" => "1",
15. "@timestamp" => "2016-01-14T06:27:58.058Z",
16. "host" => "linux-node1"
17. }

使用logstash启动一个配置文件，会在elasticsearch中写一份

1. [root@linux-node1 ~]# cat normal.conf
2. input { stdin { } }
3. output {
4. elasticsearch { hosts => ["localhost:9200"] }
5. stdout { codec => rubydebug }
6. }
7. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f normal.conf
8. Settings: Default filter workers: 1
9. Logstash startup completed
10. 123
11. {
12. "message" => "123",
13. "@version" => "1",
14. "@timestamp" => "2016-01-14T06:51:13.411Z",
15. "host" => "linux-node1

**5.2学习编写conf格式**

* 输入插件配置，此处以file为例，可以设置多个

1. input {
2. file {
3. path => "/var/log/messages"
4. type => "syslog"
5. }
6. file {
7. path => "/var/log/apache/access.log"
8. type => "apache"
9. }
10. }

* 介绍几种收集文件的方式，可以使用数组方式或者用\*匹配，也可以写多个path

1. path => ["/var/log/messages","/var/log/\*.log"]
2. path => ["/data/mysql/mysql.log"]

* 设置boolean值

1. ssl\_enable => true

* 文件大小单位

1. my\_bytes => "1113" # 1113 bytes
2. my\_bytes => "10MiB" # 10485760 bytes
3. my\_bytes => "100kib" # 102400 bytes
4. my\_bytes => "180 mb" # 180000000 bytes

* jason收集   
  codec => “json”
* hash收集

1. match => {
2. "field1" => "value1"
3. "field2" => "value2"
4. ...
5. }

* 端口

1. port => 33

* 密码

1. my\_password => "password"

**5.3 学习编写input的file插件**

**5.3.1 input插件之input**

sincedb\_path：记录logstash读取位置的路径   
start\_postion :包括beginning和end，指定收集的位置，默认是end，从尾部开始   
add\_field 加一个域   
discover\_internal 发现间隔，每隔多久收集一次，默认15秒

**5.4 学习编写output的file插件**

**5.5 通过input和output插件编写conf文件**

**5.5.1 收集系统日志的conf**

1. [root@linux-node1 ~]# cat system.conf
2. input {
3. file {
4. path => "/var/log/messages"
5. type => "system"
6. start\_position => "beginning"
7. }
8. }
9. output {
10. elasticsearch {
11. hosts => ["192.168.56.11:9200"]
12. index => "system-%{+YYYY.MM.dd}"
13. }
14. }
15. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f system.conf

**5.5.2 收集elasticsearch的error日志**

此处把上个system日志和这个error（java程序日志）日志，放在一起。使用if判断，两种日志分别写到不同索引中.此处的type（固定的就是type，不可更改）不可以和日志格式的任何一个域（可以理解为字段）的名称重复，也就是说日志的域不可以有type这个名称。

1. [root@linux-node1 ~]# cat all.conf
2. input {
3. file {
4. path => "/var/log/messages"
5. type => "system"
6. start\_position => "beginning"
7. }
8. file {
9. path => "/var/log/elasticsearch/chuck-cluster.log"
10. type => "es-error"
11. start\_position => "beginning"
12. }
13. }
14. output {
15. if [type] == "system" {
16. elasticsearch {
17. hosts => ["192.168.56.11:9200"]
18. index => "system-%{+YYYY.MM.dd}"
19. }
20. }
21. if [type] == "es-error" {
22. elasticsearch {
23. hosts => ["192.168.56.11:9200"]
24. index => "es-error-%{+YYYY.MM.dd}"
25. }
26. }
27. }
28. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f all.conf

**5.6 把多行整个报错收集到一个事件中**

**5.6.1举例说明**

以at.org开头的内容都属于同一个事件，但是显示在不同行，这样的日志格式看起来很不方便，所以需要把他们合并到一个事件中

**5.6.2引入codec的multiline插件**

官方文档提供

1. input {
2. stdin {
3. codec => multiline {
4. ` pattern => "pattern, a regexp"
5. negate => "true" or "false"
6. what => "previous" or "next"`
7. }
8. }
9. }

regrxp：使用正则，什么情况下把多行合并起来   
negate:正向匹配和反向匹配   
what:合并到当前行还是下一行   
在标准输入和标准输出中测试以证明多行收集到一个日志成功

1. [root@linux-node1 ~]# cat muliline.conf
2. input {
3. stdin {
4. codec => multiline {
5. pattern => "^\["
6. negate => true
7. what => "previous"
8. }
9. }
10. }
11. output {
12. stdout {
13. codec => "rubydebug"
14. }
15. }
16. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f muliline.conf
17. Settings: Default filter workers: 1
18. Logstash startup completed
19. [1
20. [2
21. {
22. "@timestamp" => "2016-01-15T06:46:10.712Z",
23. "message" => "[1",
24. "@version" => "1",
25. "host" => "linux-node1"
26. }
27. chuck
28. chuck-blog.com
29. 123456
30. [3
31. {
32. "@timestamp" => "2016-01-15T06:46:16.306Z",
33. "message" => "[2\nchuck\nchuck-bloh\nchuck-blog.com\n123456",
34. "@version" => "1",
35. "tags" => [
36. [0] "multiline"
37. ],
38. "host" => "linux-node1"

继续将上述实验结果放到all.conf的es-error索引中

1. [root@linux-node1 ~]# cat all.conf
2. input {
3. file {
4. path => "/var/log/messages"
5. type => "system"
6. start\_position => "beginning"
7. }
8. file {
9. path => "/var/log/elasticsearch/chuck-clueser.log"
10. type => "es-error"
11. start\_position => "beginning"
12. codec => multiline {
13. pattern => "^\["
14. negate => true
15. what => "previous"
16. }
17. }
18. }
19. output {
20. if [type] == "system" {
21. elasticsearch {
22. hosts => ["192.168.56.11:9200"]
23. index => "system-%{+YYYY.MM.dd}"
24. }
25. }
26. if [type] == "es-error" {
27. elasticsearch {
28. hosts => ["192.168.56.11:9200"]
29. index => "es-error-%{+YYYY.MM.dd}"
30. }
31. }
32. }

**六、熟悉kibana**

**6.1 编辑kinaba配置文件使之生效**

1. [root@linux-node1 ~]# grep '^[a-Z]' /usr/local/kibana/config/kibana.yml
2. server.port: 5601 kibana端口
3. server.host: "0.0.0.0" 对外服务的主机
4. elasticsearch.url: "http://192.168.56.11:9200" 和elasticsearch练习
5. kibana.index: ".kibana 在elasticsearch中添加.kibana索引

一个screen，并启动kibana

1. [root@linux-node1 ~]# screen
2. [root@linux-node1 ~]# /usr/local/kibana/bin/kibana
3. 使用crtl +a+d退出screen

使用浏览器打开192.168.56.11:5601

**6.2 验证error的muliline插件生效**

在kibana中添加一个es-error索引   
  
可以看到默认的字段   
  
选择discover查看   
  
验证error的muliline插件生效 

**七、logstash手机nginx、syslog和tcp日志**

**7.1收集nginx的访问日志**

在这里使用codec的json插件将日志的域进行分段，使用key-value的方式，使日志格式更清晰，易于搜索，还可以降低cpu的负载   
更改nginx的配置文件的日志格式，使用json

1. [root@linux-node1 ~]# sed -n '15,33p' /etc/nginx/nginx.conf
2. log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '
3. '$status $body\_bytes\_sent "$http\_referer" '
4. '"$http\_user\_agent" "$http\_x\_forwarded\_for"';
5. log\_format json '{ "@timestamp": "$time\_local", '
6. '"@fields": { '
7. '"remote\_addr": "$remote\_addr", '
8. '"remote\_user": "$remote\_user", '
9. '"body\_bytes\_sent": "$body\_bytes\_sent", '
10. '"request\_time": "$request\_time", '
11. '"status": "$status", '
12. '"request": "$request", '
13. '"request\_method": "$request\_method", '
14. '"http\_referrer": "$http\_referer", '
15. '"body\_bytes\_sent":"$body\_bytes\_sent", '
16. '"http\_x\_forwarded\_for": "$http\_x\_forwarded\_for", '
17. '"http\_user\_agent": "$http\_user\_agent" } }';
18. # access\_log /var/log/nginx/access\_json.log main;
19. access\_log /var/log/nginx/access.log json;

启动nginx

1. [root@linux-node1 ~]# nginx -t
2. nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
3. nginx: configuration file /etc/nginx/nginx.conf test is successful
4. [root@linux-node1 ~]# nginx
5. [root@linux-node1 ~]# netstat -lntup|grep 80
6. tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 43738/nginx: master
7. tcp6 0 0 :::80 :::\* LISTEN 43738/nginx: master

日志格式显示如下   
  
使用logstash将nginx访问日志收集起来，继续写到all.conf中   
  
将nginx-log加入kibana中并显示 

**7.2 收集系统syslog日志**

前文中已经使用文件file的形式收集了系统日志/var/log/messages，但是实际生产环境是需要使用syslog插件直接收集   
修改syslog的配置文件，把日志信息发送到514端口上

1. [root@linux-node1 ~]# vim /etc/rsyslog.conf
2. 90 \*.\* @@192.168.56.11:514

将system-syslog放到all.conf中，启动all.conf

1. [root@linux-node1 ~]# cat all.conf
2. input {
3. syslog {
4. type => "system-syslog"
5. host => "192.168.56.11"
6. port => "514"
7. }
8. file {
9. path => "/var/log/messages"
10. type => "system"
11. start\_position => "beginning"
12. }
13. file {
14. path => "/var/log/nginx/access\_json.log"
15. codec => json
16. start\_position => "beginning"
17. type => "nginx-log"
18. }
19. file {
20. path => "/var/log/elasticsearch/chuck-cluster.log"
21. type => "es-error"
22. start\_position => "beginning"
23. codec => multiline {
24. pattern => "^\["
25. negate => true
26. what => "previous"
27. }
28. }
29. }
30. output {
31. if [type] == "system" {
32. elasticsearch {
33. hosts => ["192.168.56.11:9200"]
34. index => "system-%{+YYYY.MM.dd}"
35. }
36. }
37. if [type] == "es-error" {
38. elasticsearch {
39. hosts => ["192.168.56.11:9200"]
40. index => "es-error-%{+YYYY.MM.dd}"
41. }
42. }
43. if [type] == "nginx-log" {
44. elasticsearch {
45. hosts => ["192.168.56.11:9200"]
46. index => "nginx-log-%{+YYYY.MM.dd}"
47. }
48. }
49. if [type] == "system-syslog" {
50. elasticsearch {
51. hosts => ["192.168.56.11:9200"]
52. index => "system-syslog-%{+YYYY.MM.dd}"
53. }
54. }
55. }
56. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f all.conf

在elasticsearch插件中就可见到增加的system-syslog索引 

**7.3 收集tcp日志**

编写tcp.conf

1. [root@linux-node1 ~]# cat tcp.conf
2. input {
3. tcp {
4. host => "192.168.56.11"
5. port => "6666"
6. }
7. }
8. output {
9. stdout {
10. codec => "rubydebug"
11. }
12. }

使用nc对6666端口写入数据

1. [root@linux-node1 ~]# nc 192.168.56.11 6666 </var/log/yum.log

将信息输入到tcp的伪设备中

1. [root@linux-node1 ~]# echo "chuck" >/dev/tcp/192.168.56.11/6666

**八、logstash解耦之消息队列**

**8.1 图解使用消息队列架构**

　　数据源Datasource把数据写到input插件中，output插件使用消息队列把消息写入到消息队列Message Queue中，Logstash indexing Instance启动logstash使用input插件读取消息队列中的信息，Fliter插件过滤后在使用output写入到elasticsearch中。   
　　如果生产环境中不适用正则grok匹配，可以写Python脚本从消息队列中读取信息，输出到elasticsearch中

**8.2 上图架构的优点**

* 解耦，松耦合
* 解除了由于网络原因不能直接连elasticsearch的情况
* 方便架构演变，增加新内容
* 消息队列可以使用rabbitmq，zeromq等，也可以使用redis，kafka（消息不删除，但是比较重量级）等

**九、引入redis到架构中**

**9.1 使用redis收集logstash的信息**

修改redis的配置文件并启动redis

1. [root@linux-node1 ~]# vim /etc/redis.conf
2. 37 daemonize yes
3. 65 bind 192.168.56.11
4. [root@linux-node1 ~]# systemctl start redis
5. [root@linux-node1 ~]# netstat -lntup|grep 6379
6. tcp 0 0 192.168.56.11:6379 0.0.0.0:\* LISTEN 45270/redis-server

编写redis.conf

1. [root@linux-node1 ~]# cat redis-out.conf
2. input{
3. stdin{
4. }
5. }
6. output{
7. redis{
8. host => "192.168.56.11"
9. port => "6379"
10. db => "6"
11. data\_type => "list" # 数据类型为list
12. key => "demo"
13. }

启动配置文件输入信息

1. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f redis-out.conf
2. Settings: Default filter workers: 1
3. Logstash startup completed
4. chuck
5. chuck-blog

使用redis-cli连接到redis并查看输入的信息

1. [root@linux-node1 ~]# redis-cli -h 192.168.56.11
2. 192.168.56.11:6379> info #输入info查看信息
3. # Server
4. redis\_version:2.8.19
5. redis\_git\_sha1:00000000
6. redis\_git\_dirty:0
7. redis\_build\_id:c0359e7aa3798aa2
8. redis\_mode:standalone
9. os:Linux 3.10.0-229.el7.x86\_64 x86\_64
10. arch\_bits:64
11. multiplexing\_api:epoll
12. gcc\_version:4.8.3
13. process\_id:45270
14. run\_id:83f428b96e87b7354249fe42bd19ee8a8643c94e
15. tcp\_port:6379
16. uptime\_in\_seconds:1111
17. uptime\_in\_days:0
18. hz:10
19. lru\_clock:10271973
20. config\_file:/etc/redis.conf
21. # Clients
22. connected\_clients:2
23. client\_longest\_output\_list:0
24. client\_biggest\_input\_buf:0
25. blocked\_clients:0
26. # Memory
27. used\_memory:832048
28. used\_memory\_human:812.55K
29. used\_memory\_rss:5193728
30. used\_memory\_peak:832048
31. used\_memory\_peak\_human:812.55K
32. used\_memory\_lua:35840
33. mem\_fragmentation\_ratio:6.24
34. mem\_allocator:jemalloc-3.6.0
35. # Persistence
36. loading:0
37. rdb\_changes\_since\_last\_save:0
38. rdb\_bgsave\_in\_progress:0
39. rdb\_last\_save\_time:1453112484
40. rdb\_last\_bgsave\_status:ok
41. rdb\_last\_bgsave\_time\_sec:0
42. rdb\_current\_bgsave\_time\_sec:-1
43. aof\_enabled:0
44. aof\_rewrite\_in\_progress:0
45. aof\_rewrite\_scheduled:0
46. aof\_last\_rewrite\_time\_sec:-1
47. aof\_current\_rewrite\_time\_sec:-1
48. aof\_last\_bgrewrite\_status:ok
49. aof\_last\_write\_status:ok
50. # Stats
51. total\_connections\_received:2
52. total\_commands\_processed:2
53. instantaneous\_ops\_per\_sec:0
54. total\_net\_input\_bytes:164
55. total\_net\_output\_bytes:9
56. instantaneous\_input\_kbps:0.00
57. instantaneous\_output\_kbps:0.00
58. rejected\_connections:0
59. sync\_full:0
60. sync\_partial\_ok:0
61. sync\_partial\_err:0
62. expired\_keys:0
63. evicted\_keys:0
64. keyspace\_hits:0
65. keyspace\_misses:0
66. pubsub\_channels:0
67. pubsub\_patterns:0
68. latest\_fork\_usec:9722
69. # Replication
70. role:master
71. connected\_slaves:0
72. master\_repl\_offset:0
73. repl\_backlog\_active:0
74. repl\_backlog\_size:1048576
75. repl\_backlog\_first\_byte\_offset:0
76. repl\_backlog\_histlen:0
77. # CPU
78. used\_cpu\_sys:1.95
79. used\_cpu\_user:0.40
80. used\_cpu\_sys\_children:0.00
81. used\_cpu\_user\_children:0.00
82. # Keyspace
83. db6:keys=1,expires=0,avg\_ttl=0
84. 192.168.56.11:6379> select 6 #选择db6
85. OK
86. 192.168.56.11:6379[6]> keys \* #选择demo这个key
87. 1) "demo"
88. 192.168.56.11:6379[6]> LINDEX demo -2 #查看消息
89. "{\"message\":\"chuck\",\"@version\":\"1\",\"@timestamp\":\"2016-01-18T10:21:23.583Z\",\"host\":\"linux-node1\"}"
90. 192.168.56.11:6379[6]> LINDEX demo -1 #查看消息
91. "{\"message\":\"chuck-blog\",\"@version\":\"1\",\"@timestamp\":\"2016-01-18T10:25:54.523Z\",\"host\":\"linux-node1\"}"

为了下一步写input插件到把消息发送到elasticsearch中，多在redis中写入写数据

1. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f redis-out.conf
2. Settings: Default filter workers: 1
3. Logstash startup completed
4. chuck
5. chuck-blog
6. a
7. b
8. c
9. d
10. e
11. f
12. g
13. h
14. i
15. j
16. k
17. l
18. m
19. n
20. o
21. p
22. q
23. r
24. s
25. t
26. u
27. v
28. w
29. x
30. y
31. z

查看redis中名字为demo的key长度

1. 192.168.56.11:6379[6]> llen demo
2. (integer) 28

**9.2 使用redis发送消息到elasticsearch中**

编写redis-in.conf

1. [root@linux-node1 ~]# cat redis-in.conf
2. input{
3. redis {
4. host => "192.168.56.11"
5. port => "6379"
6. db => "6"
7. data\_type => "list"
8. key => "demo"
9. }
10. }
11. output{
12. elasticsearch {
13. hosts => ["192.168.56.11:9200"]
14. index => "redis-demo-%{+YYYY.MM.dd}"
15. }
16. }

启动配置文件

1. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f redis-in.conf
2. Settings: Default filter workers: 1
3. Logstash startup completed

不断刷新demo这个key的长度（读取很快，刷新一定要速度）

1. 192.168.56.11:6379[6]> llen demo
2. (integer) 28
3. 192.168.56.11:6379[6]> llen demo
4. (integer) 28
5. 192.168.56.11:6379[6]> llen demo
6. (integer) 19 #可以看到redis的消息正在写入到elasticsearch中
7. 192.168.56.11:6379[6]> llen demo
8. (integer) 7 #可以看到redis的消息正在写入到elasticsearch中
9. 192.168.56.11:6379[6]> llen demo
10. (integer) 0

在elasticsearch中查看增加了redis-demo 

**9.3 将all.conf的内容改为经由redis**

编写shipper.conf作为redis收集logstash配置文件

1. [root@linux-node1 ~]# cp all.conf shipper.conf
2. [root@linux-node1 ~]# vim shipper.conf
3. input {
4. syslog {
5. type => "system-syslog"
6. host => "192.168.56.11"
7. port => "514"
8. }
9. tcp {
10. type => "tcp-6666"
11. host => "192.168.56.11"
12. port => "6666"
13. }
14. file {
15. path => "/var/log/messages"
16. type => "system"
17. start\_position => "beginning"
18. }
19. file {
20. path => "/var/log/nginx/access\_json.log"
21. codec => json
22. start\_position => "beginning"
23. type => "nginx-log"
24. }
25. file {
26. path => "/var/log/elasticsearch/chuck-cluster.log"
27. type => "es-error"
28. start\_position => "beginning"
29. codec => multiline {
30. pattern => "^\["
31. negate => true
32. what => "previous"
33. }
34. }
35. }
36. output {
37. if [type] == "system" {
38. redis {
39. host => "192.168.56.11"
40. port => "6379"
41. db => "6"
42. data\_type => "list"
43. key => "system"
44. }
45. }
46. if [type] == "es-error" {
47. redis {
48. host => "192.168.56.11"
49. port => "6379"
50. db => "6"
51. data\_type => "list"
52. key => "es-error"
53. }
54. }
55. if [type] == "nginx-log" {
56. redis {
57. host => "192.168.56.11"
58. port => "6379"
59. db => "6"
60. data\_type => "list"
61. key => "nginx-log"
62. }
63. }
64. if [type] == "system-syslog" {
65. redis {
66. host => "192.168.56.11"
67. port => "6379"
68. db => "6"
69. data\_type => "list"
70. key => "system-syslog"
71. }
72. }
73. if [type] == "tcp-6666" {
74. redis {
75. host => "192.168.56.11"
76. port => "6379"
77. db => "6"
78. data\_type => "list"
79. key => "tcp-6666"
80. }
81. }
82. }

在redis中查看keys

1. 192.168.56.11:6379[6]> select 6
2. OK
3. 192.168.56.11:6379[6]> keys \*
4. 1) "system"
5. 2) "nginx-log"
6. 3) "tcp-6666"

编写indexer.conf作为redis发送elasticsearch配置文件

1. [root@linux-node1 ~]# cat indexer.conf
2. input {
3. redis {
4. type => "system-syslog"
5. host => "192.168.56.11"
6. port => "6379"
7. db => "6"
8. data\_type => "list"
9. key => "system-syslog"
10. }
11. redis {
12. type => "tcp-6666"
13. host => "192.168.56.11"
14. port => "6379"
15. db => "6"
16. data\_type => "list"
17. key => "tcp-6666"
18. }
19. redis {
20. type => "system"
21. host => "192.168.56.11"
22. port => "6379"
23. db => "6"
24. data\_type => "list"
25. key => "system"
26. }
27. redis {
28. type => "nginx-log"
29. host => "192.168.56.11"
30. port => "6379"
31. db => "6"
32. data\_type => "list"
33. key => "nginx-log"
34. }
35. redis {
36. type => "es-error"
37. host => "192.168.56.11"
38. port => "6379"
39. db => "6"
40. data\_type => "list"
41. key => "es-error"
42. }
43. }
44. output {
45. if [type] == "system" {
46. elasticsearch {
47. hosts => "192.168.56.11"
48. index => "system-%{+YYYY.MM.dd}"
49. }
50. }
51. if [type] == "es-error" {
52. elasticsearch {
53. hosts => "192.168.56.11"
54. index => "es-error-%{+YYYY.MM.dd}"
55. }
56. }
57. if [type] == "nginx-log" {
58. elasticsearch {
59. hosts => "192.168.56.11"
60. index => "nginx-log-%{+YYYY.MM.dd}"
61. }
62. }
63. if [type] == "system-syslog" {
64. elasticsearch {
65. hosts => "192.168.56.11"
66. index => "system-syslog-%{+YYYY.MM.dd}"
67. }
68. }
69. if [type] == "tcp-6666" {
70. elasticsearch {
71. hosts => "192.168.56.11"
72. index => "tcp-6666-%{+YYYY.MM.dd}"
73. }
74. }
75. }

启动shipper.conf

1. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f shipper.conf
2. Settings: Default filter workers: 1

由于日志量小，很快就会全部被发送到elasticsearch，key也就没了，所以多写写数据到日志中

1. [root@linux-node1 ~]# for n in `seq 10000` ;do echo $n >>/var/log/elasticsearch/chuck-cluster.log;done
2. [root@linux-node1 ~]# for n in `seq 10000` ;do echo $n >>/var/log/nginx/access\_json.log;done
3. [root@linux-node1 ~]# for n in `seq 10000` ;do echo $n >>/var/log/messages;done

查看key的长度看到key在增长

1. (integer) 2481
2. 192.168.56.11:6379[6]> llen system
3. (integer) 2613
4. 192.168.56.11:6379[6]> llen system
5. (integer) 2795
6. 192.168.56.11:6379[6]> llen system
7. (integer) 2960

启动indexer.conf

1. [root@linux-node1 ~]# /opt/logstash/bin/logstash -f indexer.conf
2. Settings: Default filter workers: 1
3. Logstash startup completed

查看key的长度看到key在减小

1. 192.168.56.11:6379[6]> llen nginx-log
2. (integer) 9680
3. 192.168.56.11:6379[6]> llen nginx-log
4. (integer) 9661
5. 192.168.56.11:6379[6]> llen nginx-log
6. (integer) 9661
7. 192.168.56.11:6379[6]> llen system
8. (integer) 9591
9. 192.168.56.11:6379[6]> llen system
10. (integer) 9572
11. 192.168.56.11:6379[6]> llen system
12. (integer) 9562

kibana查看nginx-log索引 

**十、学习logstash的fliter插件**

**10.1 熟悉grok**

前文学习了input和output插件，在这里学习fliter插件   
  
filter插件有很多，在这里就学习grok插件，使用正则匹配日志里的域来拆分。在实际生产中，apache日志不支持jason，就只能使用grok插件匹配；mysql慢查询日志也是无法拆分，只能石油grok正则表达式匹配拆分。   
在如下链接，github上有很多写好的grok模板，可以直接引用   
<https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns>   
在装好的logstash中也会有grok匹配规则，直接可以引用，路径如下

1. [root@linux-node1 patterns]# pwd
2. /opt/logstash/vendor/bundle/jruby/1.9/gems/logstash-patterns-core-2.0.2/patterns

**10.2 根据官方文档提供的编写grok.conf**

1. [root@linux-node1 ~]# cat grok.conf
2. input {
3. stdin {}
4. }
5. filter {
6. grok {
7. match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }
8. }
9. }
10. output {
11. stdout {
12. codec => "rubydebug"
13. }
14. }

启动logstash，并根据官方文档提供输入，可得到拆分结果如下显示 

**10.3 使用logstash收集mysql慢查询日志**

倒入生产中mysql的slow日志，示例格式如下：

1. # Time: 160108 15:46:14
2. # User@Host: dev\_select\_user[dev\_select\_user] @ [192.168.97.86] Id: 714519
3. # Query\_time: 1.638396 Lock\_time: 0.000163 Rows\_sent: 40 Rows\_examined: 939155
4. SET timestamp=1452239174;
5. SELECT DATE(create\_time) as day,HOUR(create\_time) as h,round(avg(low\_price),2) as low\_price
6. FROM t\_actual\_ad\_num\_log WHERE create\_time>='2016-01-07' and ad\_num<=10
7. GROUP BY DATE(create\_time),HOUR(create\_time);

使用multiline处理，并编写slow.conf

1. [root@linux-node1 ~]# cat mysql-slow.conf
2. input{
3. file {
4. path => "/root/slow.log"
5. type => "mysql-slow-log"
6. start\_position => "beginning"
7. codec => multiline {
8. pattern => "^# User@Host:"
9. negate => true
10. what => "previous"
11. }
12. }
13. }
14. filter {
15. # drop sleep events
16. grok {
17. match => { "message" =>"SELECT SLEEP" }
18. add\_tag => [ "sleep\_drop" ]
19. tag\_on\_failure => [] # prevent default \_grokparsefailure tag on real records
20. }
21. if "sleep\_drop" in [tags] {
22. drop {}
23. }
24. grok {
25. match => [ "message", "(?m)^# User@Host: %{USER:user}\[[^\]]+\] @ (?:(?<clienthost>\S\*) )?\[(?:%{IP:clientip})?\]\s+Id: %{NUMBER:row\_id:int}\s\*# Query\_time: %{NUMBER:query\_time:float}\s+Lock\_time: %{NUMBER:lock\_time:float}\s+Rows\_sent: %{NUMBER:rows\_sent:int}\s+Rows\_examined: %{NUMBER:rows\_examined:int}\s\*(?:use %{DATA:database};\s\*)?SET timestamp=%{NUMBER:timestamp};\s\*(?<query>(?<action>\w+)\s+.\*)\n#\s\*" ]
26. }
27. date {
28. match => [ "timestamp", "UNIX" ]
29. remove\_field => [ "timestamp" ]
30. }
31. }
32. output {
33. stdout{
34. codec => "rubydebug"
35. }
36. }

执行该配置文件，查看grok正则匹配结果 

**十一、生产如何上线ELK。**

**10.1日志分类**

系统日志 rsyslog logstash syslog插件

访问日志 nginx logstash codec json

错误日志 file logstash file+ mulitline

运行日志 file logstash codec json

设备日志 syslog logstash syslog插件

debug日志 file logstash json or mulitline

**10.2 日志标准化**

1）路径固定标准化

2）格式尽量使用json

**10.3日志收集步骤**

**系统日志开始->错误日志->运行日志->访问日志**   
原文链接:<http://www.chuck-blog.com/chuck/201.html>

[来自为知笔记(Wiz)](http://www.wiz.cn/i/43d0e194)

AD：[官方群:运维交流09群385168604 Linux交流QQ群339128815](http://oldboy.blog.51cto.com/)