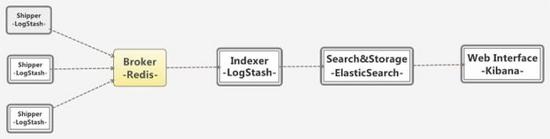
ElasticSearch+LogStash+Kibana+Redis 搭建日志管理服务

# 日志平台的结构示意图



说明：

* 多个独立的agent(Shipper)负责收集不同来源的数据，一个中心agent(Indexer)负责汇总和分析数据，在中心agent前的Broker(使用redis实现)作为缓冲区，中心agent后的ElasticSearch用于存储和搜索数据，前端的Kibana提供丰富的图表展示。
* Shipper表示日志收集，使用LogStash收集各种来源的日志数据，可以是系统日志、文件、redis、mq等等；
* Broker作为远程agent与中心agent之间的缓冲区，使用redis实现，一是可以提高系统的性能，二是可以提高系统的可靠性，当中心agent提取数据失败时，数据保存在redis中，而不至于丢失；
* 中心agent也是LogStash，从Broker中提取数据，可以执行相关的分析和处理(Filter)；
* ElasticSearch用于存储最终的数据，并提供搜索功能；
* Kibana提供一个简单、丰富的web界面，数据来自于ElasticSearch，支持各种查询、统计和展示；

# 搭建部署

## 部署Redis

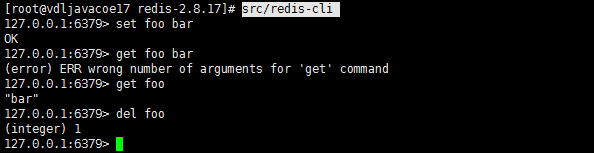
### 下载安装

|  |
| --- |
| #下载安装  yum install gcc -y  wget <http://download.redis.io/releases/redis-2.8.17.tar.gz>  tar -zxvf redis-2.8.17.tar.gz -C /usr/local  cd /usr/local/redis-2.8.17  #make命令执行完成后，会在src目录下生成5个可执行文件redis-server、  #redis-cli、redis-benchmark、redis-check-aof、redis-check-dump  make && make install  #以后台形式运行Redis需要读取配置文件方式启动  #注意，默认复制过去的redis.conf文件的daemonize参数为no，我们可以修改 #redis.conf文件，这个文件就是解压后的redis根目录下  #后台运行程序  daemonize yes  #密码认证  requirepass k12\_log\_prod |

### 运行Redis与测试

服务器启动：/usr/local/redis-2.8.17/src/redis-server redis.conf

客户端程序：/usr/local/redis-2.8.17/src/redis-cli



### 设置redis服务

vi /etc/rc.d/init.d/redis

|  |
| --- |
| #!/bin/sh  #chkconfig: 2345 90 10  #description:Redis is a persistent key-value database  #这里替换成实际的配置项  REDISPORT=6379  EXEC=/usr/local/bin/redis-server  CLIEXEC=/usr/local/bin/redis-cli  PIDFILE=/var/run/redis.pid  CONF="/usr/local/redis-2.8.17/redis.conf"  PASSWD="k12\_log\_prod"    case "$1" in  start)  if [ -f $PIDFILE ]  then  echo " PIDFILE exists, process is already running or crashed"  else  echo " Starting Redis Server ..."  $EXEC $CONF  fi  ;;    stop)  if [ ! -f $PIDFILE ]  then  echo "PIDFILE does not exist, process is not running"  else  PID=$(cat $PIDFILE)  echo "Stopping ..."  $CLIEXEC -p $REDISPORT -a $PASSWD shutdown    while [ -x "/proc/{$PID}" ]  do  echo "Waiting for Redis to shutdown ..."  sleep 1  done    echo "Redis stopped"  fi  ;;    \*)  echo "Please use start or stop as first argument"  ;;  esac |

--开机启动设置

|  |
| --- |
| chmod +x /etc/rc.d/init.d/redis #添加脚本执行权限  chkconfig --add redis  chkconfig redis on #设置开机启动  service redis start #启动redis |

### 自动化安装redis

|  |
| --- |
| #!/bin/bash  #安装redis  yum install gcc -y  wget http://download.redis.io/releases/redis-2.8.17.tar.gz  tar -zxvf redis-2.8.17.tar.gz -C /usr/local  cd /usr/local/redis-2.8.17  make && make install  #修改redis.conf  sed -i 's/daemonize no/daemonize yes/' /usr/local/redis-2.8.17/redis.conf  sed -i 's/# requirepass foobared/requirepass k12\_log\_prod/' /usr/local/redis-2.8.17/redis.conf  #配置redis服务  touch /etc/rc.d/init.d/redis  cat > /etc/rc.d/init.d/redis <<EOF  #!/bin/sh  #chkconfig: 2345 90 10  #description:Redis is a persistent key-value database  #这里替换成实际的配置项  REDISPORT=6379  EXEC=/usr/local/bin/redis-server  CLIEXEC=/usr/local/bin/redis-cli  PIDFILE=/var/run/redis.pid  CONF="/usr/local/redis-2.8.17/redis.conf"  PASSWD="k12\_log\_prod"    case "\$1" in  start)  if [ -f \$PIDFILE ]  then  echo " PIDFILE exists, process is already running or crashed"  else  echo " Starting Redis Server ..."  \$EXEC \$CONF  fi  ;;    stop)  if [ ! -f \$PIDFILE ]  then  echo "PIDFILE does not exist, process is not running"  else  PID=\$(cat \$PIDFILE)  echo "Stopping ..."  \$CLIEXEC -p \$REDISPORT -a \$PASSWD shutdown    while [ -x "/proc/{\$PID}" ]  do  echo "Waiting for Redis to shutdown ..."  sleep 1  done    echo "Redis stopped"  fi  ;;    \*)  echo "Please use start or stop as first argument"  ;;  esac  EOF  #开机启动  chmod +x /etc/rc.d/init.d/redis  chkconfig --add redis  chkconfig redis on  service redis start |

## Jdk1.8安装

rpm -ivh jdk-8u65-linux-x64.rpm

## 部署ElasticSearch

|  |
| --- |
| #下载和安装  wget https://download.elasticsearch.org/elasticsearch/elasticsearch/elasticsearch-1.3.4.tar.gz  tar zxvf elasticsearch-1.3.4.tar.gz -C /usr/local  #后台运行  /usr/local/elasticsearch-1.3.4/bin/elasticsearch –d  #访问默认的9200端口  curl -X GET <http://localhost:9200> |

## 部署中心LogStash

|  |
| --- |
| #安装  wget <https://download.elasticsearch.org/logstash/logstash/logstash-1.4.2.tar.gz>  tar zxvf logstash-1.4.2.tar.gz -C /usr/local/  cd /usr/local/logstash-1.4.2  mkdir conf logs  #配置文件conf/central.conf  input {  redis {  host => "127.0.0.1"  port => 6379  type => "redis-input"  data\_type => "list"  key => "key\_count"  }  }  output {  stdout {}  elasticsearch {  cluster => "elasticsearch"  codec => "json"  protocol => "http"  }  }  #启动  bin/logstash agent --verbose --config conf/central.conf --log logs/stdout.log &  input {  file {  path => ["/usr/local/nginx/logs/access.log"]  }  }  filter {  grok {  match => {  "message" => "%{IPORHOST:clientip} \[%{HTTPDATE:time}\] \"%{WORD:verb} %{URIPATHPARAM:request} HTTP/%{NUMBER:httpversion}\" %{NUMBER:http\_status\_code} %{NUMBER:bytes} \"(?<http\_referer>\S+)\" \"(?<http\_user\_agent>\S+)\" "  }  }  }  output {  elasticsearch {  hosts => ["127.0.0.1:9200"]  index => "logstash-nginx-%{+YYYY.MM.dd}"  workers => 1  flush\_size => 1  idle\_flush\_time => 1  template\_overwrite => true  }  stdout{codec => rubydebug}  } |

## 部署远程LogStash

|  |
| --- |
| #安装  wget <https://download.elasticsearch.org/logstash/logstash/logstash-1.4.2.tar.gz>  tar zxvf logstash-1.4.2.tar.gz -C /usr/local/  cd /usr/local/logstash-1.4.2  mkdir conf logs  vi conf/shipper.conf  #配置文件表示输入来自于目录/data/logs/count/下的stdout.log和stderr.log两#个文件，且排除该目录下所有.gz文件和access.log；(这里因为path没有使用通  #配符，所以exclude是没有效果的)；输出表示将监听到的event发送到redis服务  #器，使用redis的list保存，key为”key\_count”，这里的 data\_type 属性和  #key 属性应该与中心agent的配置一致；  input {  file {  type => "type\_count"  path => ["/data/logs/count/stdout.log", "/data/logs/count/stderr.log"]  exclude => ["\*.gz", "access.log"]  }  }  output {  stdout {}  redis {  host => " 172.16.130.55"  port => 6379  data\_type => "list"  key => "key\_count"  }  }  #启动  bin/logstash agent --verbose --config conf/shipper.conf --log logs/stdout.log & |

## 部署Kibana

|  |
| --- |
| #下载加压kibaba  wget <https://download.elasticsearch.org/kibana/kibana/kibana-3.1.1.tar.gz>  tar zxvf kibana-3.1.1.tar.gz  #修改配置文件config.js，仅需要配置elasticsearch的地址即可：  elasticsearch: "http:// 172.16.130.55:9200"  #将目录kibana-3.1.1拷贝到tomcat的webapp目录下，并启动tomcat：  mv src/kibana-3.1.1 tomcat7/webapps/  tomcat7/bin/startup.sh  #访问  <http://172.16.130.55:8080/kibana-3.1.1/> |
| nohup /usr/local/kibana-4.0.2-linux-x86/bin/kibana &  http://139.224.33.13:5601/ |

# 简单测试

打开LogStash的远程agent和中心agent的日志：tail -f logs/stdout.log

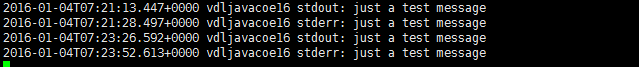
远程agent的数据是以 rpush 操作将event推送到redis的list中，中心agent通过 blpop 命令从redis的list中提取数据，因此，测试时由于数据量小，通过命令 llen key\_count 的返回结果很可能为空，因此为了观察redis中数据流的变化，可以使用 monitor 命令：redis-cli -p 6379 monitor

我们向/data/logs/count目录下的stdout.log和stderr.log各发送一条数据：

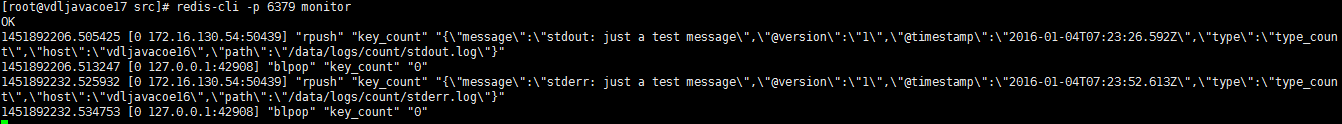
$ echo "stdout: just a test message" >> stdout.log

$ echo "stderr: just a test message" >> stderr.log

## 远程agent和中心agent都会收到event消息

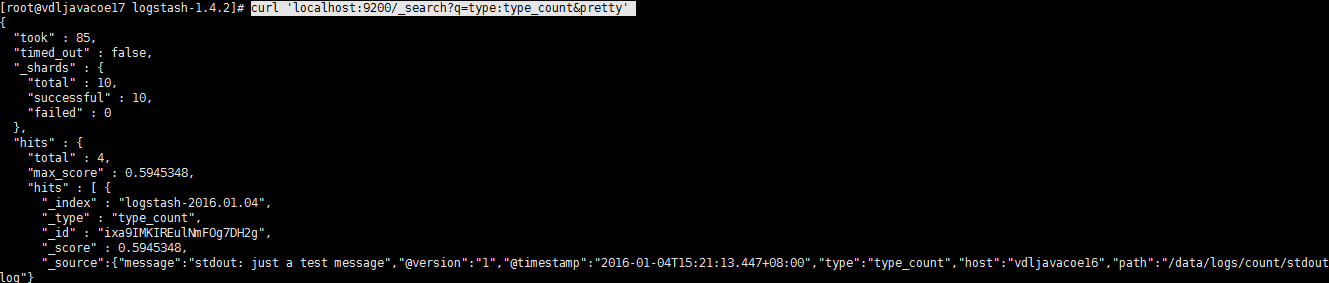


## 我们可以观察到redis的输出

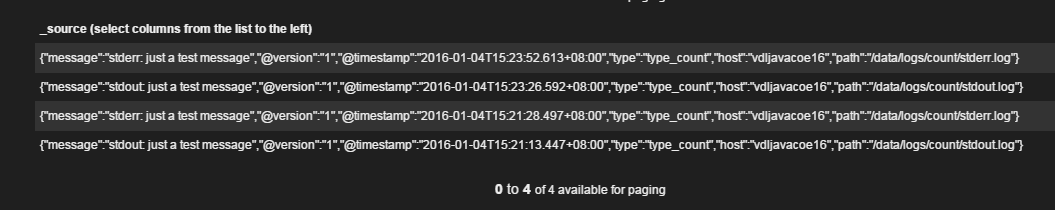


## elasticsearch中执行如下的简单查询

curl 'localhost:9200/\_search?q=type:type\_count&pretty'



<http://172.16.130.55:8080/kibana-3.1.1/>



## 后续工作

* 使用LogStash的Filter对日志数据进行过滤和分析；
* 使用Redis的Cluster模式替换单机模式；
* 在elasticsearch中对数据进行高级的分析和查询；
* 熟悉Kibana的展示组件以及查询语法；

# Logstash使用grok过滤nginx日志

在生产环境中，nginx日志格式往往使用的是自定义的格式，我们需要把logstash中的message结构化后再存储，方便kibana的搜索和统计，因此需要对message进行解析。本文采用grok过滤器，使用match正则表达式解析，根据自己的log\_format定制。

## nginx日志格式

|  |
| --- |
| #log\_format配置如下  log\_format main '$remote\_addr - $remote\_user [$time\_local] $http\_host $request\_method "$uri" "$query\_string" '  '$status $body\_bytes\_sent "$http\_referer" $upstream\_status $upstream\_addr $request\_time $upstream\_response\_time '  '"$http\_user\_agent" "$http\_cdn\_src\_ip" "$http\_x\_forwarded\_for"' ; |
| #日志如下  1.1.1.1 - - [06/Jun/2016:00:00:01 +0800] www.test.com GET "/api/index" "?cms=0&rnd=1692442321" 200 4 "http://www.test.com/?cp=sfwefsc" 200 192.168.0.122:80 0.004 0.004 "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36" "2.2.2.2" "-" |

## 编写正则表达式

　　logstash中默认存在一部分正则让我们来使用，可以访问Grok Debugger来查看，可以在 $logstash/vendor/bundle/jruby/1.9/gems/logstash-patterns-core-4.0.0/patterns/ 目录里面查看。基本定义在grok-patterns中，我们可以使用其中的正则，当然并不是所有的都适合nginx字段，这时就需要我们自定义正则，然后通过指定patterns\_dir来调用。

·同时在写正则的时候可以使用Grok Debugger或者Grok Comstructor工具来帮助我们更快的调试。在不知道如何使用logstash中的正则的时候也可使用Grok Debugger的Descover来自动匹配。

|  |
| --- |
| # nginx标准日志格式  logstash自带的grok正则中有Apache的标准日志格式  COMMONAPACHELOG %{IPORHOST:clientip} %{HTTPDUSER:ident} %{USER:auth} \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})" %{NUMBER:response} (?:%{NUMBER:bytes}|-)  COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}  对于nginx标准日志格式只是最后多了一个 $http\_x\_forwarded\_for 变量  AINNGINXLOG %{COMBINEDAPACHELOG} %{QS:x\_forwarded\_for} |
| 自定义格式  %{IPORHOST:remote\_addr} - (%{USERNAME:user}|-) \[%{HTTPDATE:log\_timestamp}\] %{HOSTNAME:http\_host} %{WORD:request\_method} \"%{URIPATH1:uri}\" \"%{URIPARM1:param}\" %{BASE10NUM:http\_status} (?:%{BASE10NUM:body\_bytes\_sent}|-) \"(?:%{URI1:http\_referrer}|-)\" (%{BASE10NUM:upstream\_status}|-) (?:%{HOSTPORT:upstream\_addr}|-) (%{BASE16FLOAT:upstream\_response\_time}|-) (%{BASE16FLOAT:request\_time}|-) (?:%{QUOTEDSTRING:user\_agent}|-) \"(%{IPV4:client\_ip}|-)\" \"(%{WORD:x\_forword\_for}|-)\"  #自定义的正则  URIPARM1 [A-Za-z0-9$.+!\*'|(){},~@#%&/=:;\_?\-\[\]]\*  URIPATH1 (?:/[A-Za-z0-9$.+!\*'(){},~:;=@#%&\_\- ]\*)+  URI1 (%{URIPROTO}://)?(?:%{USER}(?::[^@]\*)?@)?(?:%{URIHOST})?(?:%{URIPATHPARAM})? HOSTPORT %{IPORHOST}:%{POSINT}  注意：message是每段读进来的日志，IPORHOST、USERNAME、HTTPDATE等都是patterns/grok-patterns中定义好的正则格式名称，对照日志进行编写。grok pattren的语法为：%{SYNTAX:semantic}，":" 前面是grok-pattrens中定义的变量，后面可以自定义变量的名称。(?:%{SYNTAX:semantic}|-)这种形式是条件判断。 |

## logstash的配置文件

|  |
| --- |
| # mkdir -p /usr/local/logstash/patterns  # vi /usr/local/logstash/patterns/nginx  URIPARM1 [A-Za-z0-9$.+!\*'|(){},~@#%&/=:;\_?\-\[\]]\*  URIPATH1 (?:/[A-Za-z0-9$.+!\*'(){},~:;=@#%&\_\- ]\*)+  URI1 (%{URIPROTO}://)?(?:%{USER}(?::[^@]\*)?@)?(?:%{URIHOST})?(?:%{URIPATHPARAM})?  NGINXACCESS %{IPORHOST:remote\_addr} - (%{USERNAME:user}|-) \[%{HTTPDATE:log\_timestamp}\] %{HOSTNAME:http\_host} %{WORD:request\_method} \"%{URIPATH1:uri}\" \"%{URIPARM1:param}\" %{BASE10NUM:http\_status} (?:%{BASE10NUM:body\_bytes\_sent}|-) \"(?:%{URI1:http\_referrer}|-)\" (%{BASE10NUM:upstream\_status}|-) (?:%{HOSTPORT:upstream\_addr}|-) (%{BASE16FLOAT:upstream\_response\_time}|-) (%{BASE16FLOAT:request\_time}|-) (?:%{QUOTEDSTRING:user\_agent}|-) \"(%{IPV4:client\_ip}|-)\" \"(%{WORD:x\_forword\_for}|-)\" |
| logstash.conf配置文件内容  input {  file {  path => "/usr/local/nginx/logs/access.log"  type => "nginx-access"  }  }  filter {  if [type] == "nginx-access" {  grok {  patterns\_dir => "/usr/local/logstash/patterns" //设置自定义正则路径  match => {  "message" => "%{NGINXACCESS}"  }  }  date {  match => [ "log\_timestamp" , "dd/MMM/YYYY:HH:mm:ss Z" ]  }  　　　　　　　　　urldecode {  　　　　　　　　　all\_fields => true  　　　　　　　}  　　　　　　　　　//把所有字段进行urldecode（显示中文）  　　　　　　 mutate {  　　　　　　　　　　gsub => ["user\_agent","[\"]",""] //将user\_agent中的 " 换成空  　　　　　　　}  }  }  output {  if [type] == "nginx-access" {  elasticsearch {  hosts => ["127.0.0.1:9200"]  manage\_template => true  index => "logstash-nginx-%{+YYYY.MM.dd}"  }  }  } |