

# ELK环境基础部署及优化配置

## 前言

- ELK中logstash及elasticsearch组件依赖java环境，安装前请先确认java环境
- 相关参考文档：
  - 官方文档：<https://www.elastic.co/guide/index.html>
  - Elasticsearch权威指南：[http://www.learnes.net/getting\\_started/README.html](http://www.learnes.net/getting_started/README.html)
  - ELKstack权威指南：<http://kibana.logstash.es/content/>
  - kibana4官方展示：<http://demo.elastic.co/packetbeat/>

## Filebeat组件

### 1. 部署filebeat

下载filebeat安装包 `curl -L -O https://download.elastic.co/beats/filebeat/filebeat-1.3.1-x86_64.rpm`  
安装RPM包 `rpm -vi filebeat-1.3.1-x86_64.rpm`  
安装完成后，默认配置文件路径在 `/etc/filebeat/filebeat.yml`  
启动指令为 `filebeat -c /etc/filebeat/filebeat.yml -e &`

### 2. 配置解析

```
filebeat:
  spool_size: 1024                                # 最大可以攒够 1024 条数据一起发送出去
  idle_timeout: "5s"                              # 否则每 5 秒钟也得发送一次
  registry_file: ".filebeat"                      # 文件读取位置记录文件，会放在当前工作目录下。所以如果你换一个工作目录执行 filebeat
  config_dir: "path/to/configs/contains/many/yaml" # 如果配置过长，可以通过目录加载方式拆分配置
  prospectors:
    -
      fields:
        ownfield: "mac"                            # 类似 logstash 的 add_fields
      paths:
        - /var/log/system.log                       # 指明读取文件的位置
        - /var/log/wifi.log
      include_lines: ["^ERR", "^WARN"]              # 只发送包含这些字样的日志
      exclude_lines: ["^OK"]                        # 不发送包含这些字样的日志
    -
      document_type: "apache"                       # 定义写入 ES 时的 _type 值
      ignore_older: "24h"                           # 超过 24 小时没更新内容的文件不再监听。在windows上另外有一个配置叫force_close_file
      scan_frequency: "10s"                         # 每 10 秒钟扫描一次目录，更新通配符匹配上的文件列表
      tail_files: false                             # 是否从文件末尾开始读取
      harvester_buffer_size: 16384                   # 实际读取文件时，每次读取 16384 字节
      backoff: "1s"                                 # 每 1 秒检测一次文件是否有新的一行内容需要读取
      paths:
        - "/var/log/apache/*"                       # 可以使用通配符
      exclude_files: ["/var/log/apache/error.log"]
    -
      input_type: "stdin"                           # 除了 "log"，还有 "stdin"
      multiline:
        pattern: '^[[:space:]]'                    # 多行合并
        negate: false
        match: after
  output:
    logstash:
      host: ["127.0.0.1:5044"]
```

## logstash组件

### 1. 使用yum源部署logstash

认证公钥`rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch`  
在`/etc/yum.repo/`下添加yum源的子域文件，文件内容：

```
[logstash-2.4]
```

```
name=Logstash repository for 2.4.x packages
baseurl=https://packages.elastic.co/logstash/2.4/centos
gpgcheck=1
gpgkey=https://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

## 2. 使用二进制包安装logstash

使用指令 `wget https://download.elastic.co/logstash/logstash/logstash-2.4.0.tar.gz`

解压缩文件 `tar -zxvf logstash-2.4.0.tar.gz`

1.4版本依旧提供全量插件的二进制包，之后版本更新不再提供全量插件的部署包，建议使用yum源安装

## 3. 指令说明：

在使用配置文件启动logstash进程之前使用指令 `logstash -t -f /etc/logstash/logstash.json` 测试配置文件格式是否正确

启动logstash `logstash -f /etc/logstash/logstash.json`

## 4. input配置说明：

```
input {
  file {
    path => ["/var/log/*.log", "/var/log/message"]
    type => "system"                                #设置文件录入的type类型
    start_position => "beginning"                    #设置文件读取的起始位置
    discover_interval => 3                           #设置检查文件变化的时间间隔
    exclude => ["/var/log/other.log"]                 #设置排除不想检查的文件
    close_older => 3600                               #一个已经监听中的文件，如果超过这个值的时间内没有更新内容，就关闭监听它的文件
    ignore_older => 86400                             #在每次检查文件列表的时候，如果一个文件的最后修改时间超过这个值，就忽略这个文件
  }
  stdin {
    add_field => {"key" => "value"}                   #为输入的字符串增加一组键值对应关系
    codec => "rubydebug"                             #设置编码格式
    tags => ["add"]                                   #增加键值的tag字段，相当于列表的append
    type => "std"                                     #为输入的日志设置type类型字段
  }
}
```

## 5. codec配置说明：

```
input {
  stdin {
    codec => multiline {
      pattern => "^[\""
      negate => true
      what => "previous"
    }
  }
}
```

## 6. filter配置说明：

```
filter {
  if [type] == 'nginx_access_log' {
    grok {
      match => { "message" => "\"(?<datetime>.+?)\" \| (?<domain>.+?) \| (?<status_code>\d+?) \| (?<requests_time>.+?)" }
    }
    if [http_referer] != '-' {
      if '?' in [http_referer] {
        grok {
          match => { "http_referer" => "(?<referer_url>.+?)\?(?<referer_parameters>.*)" }
        }
        if '&' in [referer_parameters] {
          ruby {
            code => "
              new_event = LogStash::Event.new(Hash[event['referer_parameters'].split('&').map{|l| l.split('=')}]
              new_event.remove('@timestamp')
              event.append(new_event)
            "
          }
        }
      } else {

```

```

        grok {
            match => { "referer_parameters" => "(?<key>.+?)\=(?<value>.+?)" }
        }
        mutate {
            rename => { "value" => "%{key}" }          #取出参数名称替换临时value键名称
            remove_field => ["value"]                 #剔除临时的value类型
        }
    }
    mutate {
        remove_field => ["referer_parameters"]
    }
} else {
    mutate {
        rename => { "http_referer" => "referer_url" }
    }
}
}
if [requests] != '-' {
    grok {
        match => { "requests" => "(?<requests_method>\w+?) (?<requests_info>.+?) (?<http_version>.*)" }
    }
    if '?' in [requests_info] {
        grok {
            match => { "requests_info" => "(?<requests_url>.+?)\?(?<requests_parameters>.*)" }
        }
        if '&' in [requests_parameters] {
            ruby {
                code => "
                    new_event = LogStash::Event.new(Hash[event['requests_parameters'].split('&').map{|l| l.split('=')}])
                    new_event.remove('@timestamp')
                    event.append(new_event)
                "
            }
        } else {
            grok {
                match => { "requests_parameters" => "(?<key>.+?)=(?<value>.+?)" }
            }
            mutate {
                rename => { "value" => "%{key}" }
                remove_field => ["value"]
            }
        }
        mutate {
            remove_field => ["requests_parameters"]
        }
    } else {
        mutate {
            rename => { "requests_info" => "requests_url" }
        }
    }
}
}
date {
    match => ["datetime", "dd/MMM/yyyy:HH:mm:ss Z"]          #定义时间转化格式，设置偏移时间
    target => "log_time"                                       #赋值新的时间戳键名称
    timezone => "Asia/Chongqing"                               #设置时区，确认偏移时间
}
mutate {
    remove_field => ["message"]
    convert => [                                                #filter插件强制转化数据类型
        "requests_time", "float",
        "upstream_response_time", "float",
        "body_bytes_sent", "integer"
    ]
}
}
}
}
}

```

## 7. output配置说明:

```
output {
```

```

    stdout {
      codec => rubydebug {          #设置输出的编码格式
      }
      workers => 2                  #设置进程数
    }
  elasticsearch {
    hosts => ["192.168.0.2:9200"]
    index => "logstash-%{type}~%{+YYYY.MM.dd}"          #设置插入索引的名称，支持所有通用型变量插值
    document_type => "%{type}"          #设置索引的一级检索type类型
    workers => 1          #设置进程数
    flush_size => 20000          #设置最大缓存条数
    idle_flush_time => 10          #设置最大缓存时间，最小写入周期
    template_overwrite => true          #设置操作elasticsearch的template方式
  }
  email {
    to => "admin@website.com,root@website.com"
    cc => "other@website.com"
    via => "smtp"
    subject => "Warning: %{title}"
    options => {
      smtpIpOrHost      => "localhost",
      port              => 25,
      domain            => 'localhost.localdomain',
      userName          => nil,
      password          => nil,
      authenticationType => nil, # (plain, login and cram_md5)
      starttls          => true
    }
    htmlbody => ""
    body => ""
    attachments => ["/path/to/filename"]
  }
}

```

## elasticsearch组件

### 1. 部署elasticsearch:

使用指令

wget <https://download.elastic.co/elasticsearch/release/org/elasticsearch/distribution/tar/elasticsearch/2.4.1/elasticsearch-2.4.1.tar.gz> 下载

载二进制包

解压缩 `tar -zxvf elasticsearch-2.4.1.tar.gz`

配置文件在ES\_HOME下的config/elasticsearch.yml里

启动指令 `ES_HOME/bin/elasticsearch -c ES_HOME/config/elasticsearch.yml`

### 2. 配置说明:

```

cluster.name: JP_log          #设置集群名称

index:          #索引相关配置
  number_of_shards: 3          #设置单个索引的最大分片数量
  max_local_storage_nodes: 3    #设置最大存储节点数量
  translog:          #translog配置
    flush_threshold_period: 1h  #设置translog刷新时间
    flush_threshold_size: 2048mb #设置translog刷新大小
    durability: async          #设置translog的更新时间
  refresh:
    interval: 10s          #设置刷新时间间隔

node:
  name: es_node1          #设置节点名称
  master: true            #是否可以成为主节点
  data: true              #是否作为数据节点

path:          #设置相关存储路径
  data: /data/ELK/JP_log/
  log: /data/log/elasticsearch

network:          #设置绑定IP地址
  host: 10.205.140.40

```

```

http:
  port: 9200                                #设置绑定端口

discovery:
  zen:
    ping:
      unicast:
        hosts: ["es_node1", "es_node2", "es_node3"]    #配置集群节点
        minimum_master_nodes: 1                        #保证最小主节点数量时鸡群正常

gateway:
  recover_after_nodes: 1                      #数据恢复时的丢失节点数量

indices:
  cache:
    filter:
      size: 1024mb                             #设置查询数据的缓存池大小

```

### 3. 相关API说明

查看Elasticsearch状态 `http://127.0.0.1:9200/_cat/health?v`

查看所有node节点状态 `http://127.0.0.1:9200/_cat/nodes?v`

查看所有index状态 `http://127.0.0.1:9200/_cat/indices?v`

创建索引 PUT方法 `'127.0.0.1:9200/索引名称?pretty'`

删除索引 DELETE方法 `'127.0.0.1:9200/索引名称?pretty'`

搜索全部index中的内容

`curl -XPOST 127.0.0.1:9200/索引名称/_search?q=*&pretty '127.0.0.1:9200/索引名称/_search?pretty'-d '{ "query": { "match_all": { } } }`

## supervisor守护进程

### 1. 配置说明:

```

[program:jp_es]                                #配置项目名称
directory = /data/ELK/elasticsearch-2.4.1/bin/ #配置程序执行路径
command = /data/ELK/elasticsearch-2.4.1/bin/elasticsearch #配置程序执行指令
autostart = true                               #程序是否自启动
autorestart = true                             #是否自动重启
startsecs = 5                                  #自动重启检测时间
redirect_stderr = true                         #错误输出重定向
stdout_logfile = /data/log/supervisor/es.log    #程序屏幕打印的输出路径

```

### 2. 指令说明:

更新项目配置文件 `supervisorctl -c /etc/supervisord.conf update`

启动/停止/重启项目 `supervisorctl -c /etc/supervisord.conf start/stop/restart project/all`

## Kibana组件

### 1. 部署kibana

使用指令 `wget https://download.elastic.co/kibana/kibana/kibana-4.6.2-linux-x86_64.tar.gz` 下载二进制包

配置文件路径 `../config/kibana.yml`

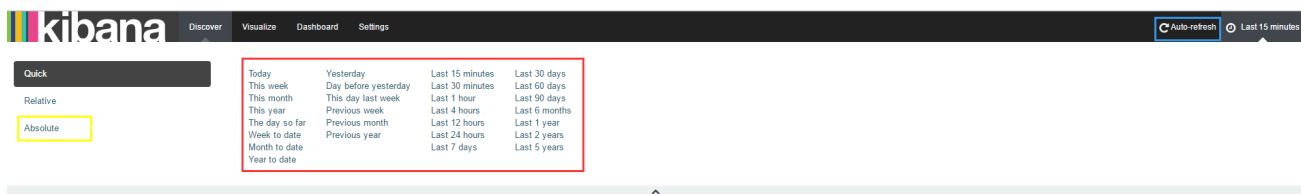
启动指令 `./kibana`

### 2. 使用说明

`[jp-log-]YYYY.MM.DD`

`[jp-log-php-log-]YYYY.MM.DD`

当前索引分为两部分，其中 `jp-log-YYYY.MM.DD` 为查询nginx访问日志索引，`jp-log-php-log` 为查询phpslow日志索引



蓝色框为设置自动刷新时间，红色框为预定义的时间范围，黄色框为自定义时间范围的选择

Quick

Relative

Absolute

From:2016-11-01 10:14:42.622  
YYYY-MM-DD HH:mm:ss SSS

To:2016-11-01 10:29:42.622  
YYYY-MM-DD HH:mm:ss SSS

Go

<November 2016>

Sun

Mon

Tue

Wed

Thu

Fri

Sat

30

31

01

02

03

04

05

06

07

08

09

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

01

02

03

04

05

06

07

08

09

10

<November 2016>

Sun

Mon

Tue

Wed

Thu

Fri

Sat

30

31

01

02

03

04

05

06

07

08

09

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

01

02

03

04

05

06

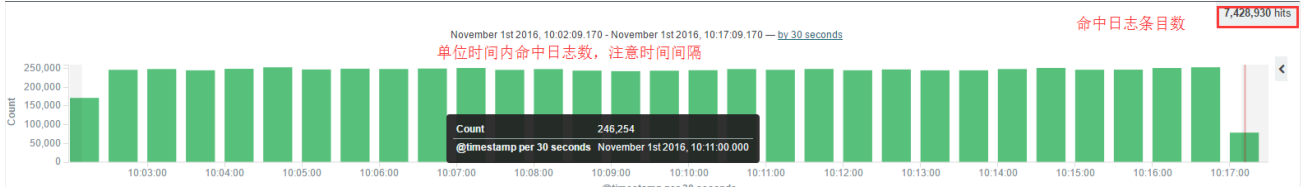
07

08

09

10

自定义时间选择



日志条目数说明

Time ▾	datetime	domain	host	requests_method	requests_info
--------	----------	--------	------	-----------------	---------------

日志字段筛选

Selected Fields

datetime

host

requests\_info

requests\_method

domain

Available Fields

Popular

beat.hostname

http\_referer

remote\_addr

requests

status\_code

tags

upstream\_addr

upstream\_response\_time

可在Select Fields移除已显示细节的字段

在Available Fields添加要显示细节的字段

Time ▾	datetime	domain	host	requests_method	requests_info
▶ November 1st 2016, 10:17:09.176	01/Nov/2016:10:17:07 +0800	pay.juanpi.com	GZ-JSQ-JP-PHP-NGINX-002.jp	POST	/paygate/gateway_pay_async_push/m_sdk_xnpay/1790483
▶ November 1st 2016, 10:17:09.168	01/Nov/2016:10:17:09 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	POST	/gateway/pay
▶ November 1st 2016, 10:17:09.168	01/Nov/2016:10:17:09 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	POST	/wallet/walletInfo
▶ November 1st 2016, 10:17:09.168	01/Nov/2016:10:17:09 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	POST	/wallet/getMiniWallet
▶ November 1st 2016, 10:17:09.168	01/Nov/2016:10:17:09 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	POST	/wallet/setPassword
▶ November 1st 2016, 10:17:09.168	01/Nov/2016:10:17:09 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	POST	/wallet/walletInfo
▶ November 1st 2016, 10:17:09.167	01/Nov/2016:10:17:08 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	POST	/gateway/syncPayInfoFromRemote
▶ November 1st 2016, 10:17:09.167	01/Nov/2016:10:17:08 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	POST	/gateway/syncPayInfoFromRemote
▶ November 1st 2016, 10:17:09.167	01/Nov/2016:10:17:08 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	POST	/wallet/isWalletUser
▶ November 1st 2016, 10:17:09.167	01/Nov/2016:10:17:08 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	POST	/wallet/getMiniWallet
▶ November 1st 2016, 10:17:09.167	01/Nov/2016:10:17:08 +0800	api.pay.juanpi.jp	GZ-JSQ-JP-PHP-NGINX-001.jp	GET	/wallet/cancelOrderPay?apiKey=api_pay&sign=2ce406fcd9b04ea61653b487630dddb

效果展示

Search...

beat\_hostname:GZ-JSQ-JP-PHP-NGINX-00 AND domain:mapl.juanpi.com

domain:"mapl.juanpi.com"

beat\_hostname:GZ-JSQ-JP-PHP-NGINX-00

beat\_hostname:GZ-JSQ-JP-PHP-NGINX\*

查询索引栏及历史查询数据

全文检索直接在查询框输入查询字段

walle|

按字段查询以key:value的格式查询，可以使用简单匹配AND与OR字段，如

beat\_hostname:GZ-JSQ-JP-PHP-NGINX-00 AND domain:mapl.juanpi.com

注意，默认情况下value字段使用全文检索，如果要精确搜索加双引号"value"

manage dashboards

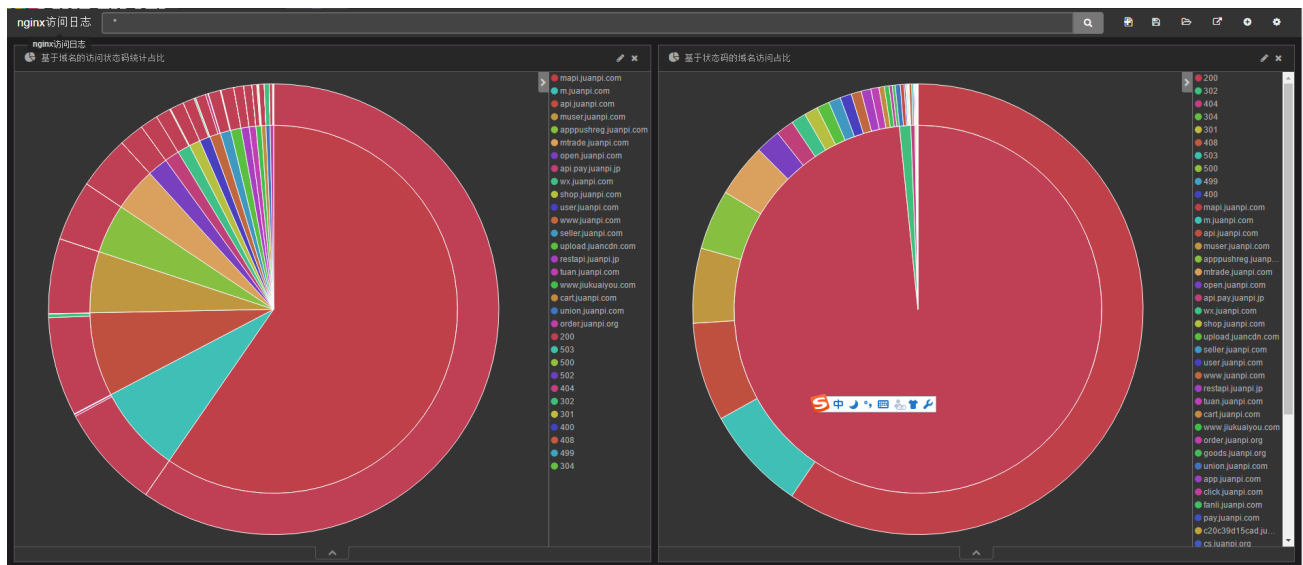
2 dashboards

Dashboard Filter

PHP 慢日志

nginx 访问日志

dashboard 导航页面，红色框选择已定义的 dashboard



自定义图表

## Create a new visualization

Step 1

Area chart	Great for stacked timelines in which the total of all series is more important than comparing any two or more series. Less useful for assessing the relative change of unrelated data points as changes in a series lower down the stack will have a difficult to gauge effect on the series above it.
Data table	The data table provides a detailed breakdown, in tabular format, of the results of a composed aggregation. Tip, a data table is available from many other charts by clicking grey bar at the bottom of the chart.
Line chart	Often the best chart for high density time series. Great for comparing one series to another. Be careful with sparse sets as the connection between points can be misleading.
Markdown widget	Useful for displaying explanations or instructions for dashboards.
Metric	One big number for all of your one big number needs. Perfect for showing a count of hits, or the exact average a numeric field.
Pie chart	Pie charts are ideal for displaying the parts of some whole. For example, sales percentages by department.Pro Tip: Pie charts are best used sparingly, and with no more than 7 slices per pie.
Tile map	Your source for geographic maps. Requires an elasticsearch geo_point field. More specifically, a field that is mapped as type:geo_point with latitude and longitude coordinates.
Vertical bar chart	The goto chart for oh-so-many needs. Great for time and non-time data. Stacked or grouped, exact numbers or percentages. If you are not sure which chart you need, you could do worse than to start here.