|  |
| --- |
| Neusoft |
| Kibana&Logstash |
| Kibana&logstash 安装配置 |

|  |
| --- |
| 成阳阳  2015/10/14 |
|  |

中文指南

http://kibana.logstash.es/

# 所需环境

安装jdk 1.7

[oracle java主页](http://www.oracle.com/technetwork/java" \t "_blank)

安装redis

[redis主页](http://redis.io/" \t "_blank)

安装Elasticsearch

[Elasticsearch主页](http://www.elasticsearch.org/" \t "_blank)

# logstash

<https://www.elastic.co/products/logstash>

Logstash 从 1.5 版本开始，将核心代码和插件代码完全剥离，并重构了插件架构逻辑，所有插件都以标准的 Ruby Gem 包形式发布。不过，为了方便大家从 1.4 版本过度，目前，官方依然发布打包有所有官方维护的插件在内的软件包。只是不再发布类似原先 1.4 时代的 logstash-contrib.tar.gz 这样的软件包了。依然要使用社区插件的读者，请阅读稍后插件安装章节。

下载官方软件包的方式有以下几种：

压缩包方式

wget https://download.elastic.co/logstash/logstash/logstash-1.5.1.tar.gz

Debian 平台

wget https://download.elastic.co/logstash/logstash/packages/debian/logstash\_1.5.1-1\_all.deb

Redhat 平台

wget https://download.elastic.co/logstash/logstash/packages/centos/logstash-1.5.1-1.noarch.rpm

## Hello World

和绝大多数 IT 技术介绍一样，我们以一个输出 "hello world" 的形式开始我们的 logstash 学习。

运行

在终端中，像下面这样运行命令来启动 Logstash 进程：

|  |
| --- |
| bin/logstash -e 'input{stdin{}}output{stdout{codec=>rubydebug}}' |

然后你会发现终端在等待你的输入。没问题，敲入 Hello World，回车，然后看看会返回什么结果！

结果

{

"message" => "Hello World",

"@version" => "1",

"@timestamp" => "2014-08-07T10:30:59.937Z",

"host" => "raochenlindeMacBook-Air.local",

}

## plugin的安装

从 logstash 1.5.0 版本开始，logstash 将所有的插件都独立拆分成 gem 包。这样，每个插件都可以独立更新，不用等待 logstash 自身做整体更新的时候才能使用了。

为了达到这个目标，logstash 配置了专门的 plugins 管理命令。

plugin 用法说明

|  |
| --- |
| Usage:  bin/plugin [OPTIONS] SUBCOMMAND [ARG] ...  Parameters:  SUBCOMMAND subcommand  [ARG] ... subcommand arguments  Subcommands:  install Install a plugin  uninstall Uninstall a plugin  update Install a plugin  list List all installed plugins  Options:  -h, --help print help |

首先，你可以通过命令查看本机现在有多少插件可用

|  |
| --- |
| bin/plugin list |

<https://github.com/logstash-plugins/> 查找插件

安装命令

|  |
| --- |
| bin/plugin install logstash-output-webhdfs |

同样，假如是升级，只需要运行：

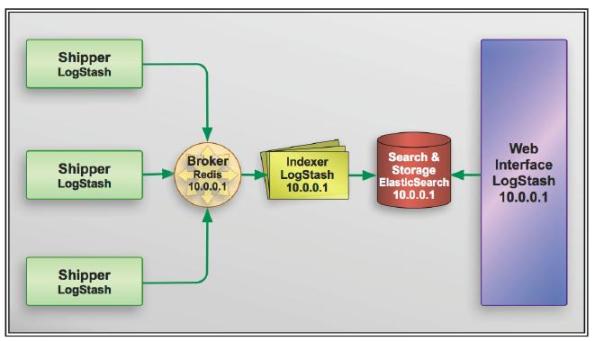
|  |
| --- |
| bin/plugin update logstash-input-tcp |

本地安装插件

|  |
| --- |
| bin/plugin install /path/to/logstash-filter-crash.gem |

## 配置语法

Logstash 社区通常习惯用 shipper，broker 和 indexer 来描述数据流中不同进程各自的角色。如下图：



logstash arch

不过我见过很多运用场景里都没有用 logstash 作为 shipper，或者说没有用 elasticsearch 作为数据存储也就是说也没有 indexer。所以，我们其实不需要这些概念。只需要学好怎么使用和配置 logstash 进程，然后把它运用到你的日志管理架构中最合适它的位置就够了。

### 语法

Logstash 设计了自己的 DSL —— 有点像 Puppet 的 DSL，或许因为都是用 Ruby 语言写的吧 —— 包括有区域，注释，数据类型(布尔值，字符串，数值，数组，哈希)，条件判断，字段引用等。

区段(section)

Logstash 用 {} 来定义区域。区域内可以包括插件区域定义，你可以在一个区域内定义多个插件。插件区域内则可以定义键值对设置。示例如下：

|  |
| --- |
| input {  stdin {}  syslog {}  } |

### 数据类型

Logstash 支持少量的数据值类型：

bool

debug => true

string

host => "hostname"

number

port => 514

array

match => ["datetime", "UNIX", "ISO8601"]

hash

options => {

key1 => "value1",

key2 => "value2"

}

注意：如果你用的版本低于 1.2.0，哈希的语法跟数组是一样的，像下面这样写：

match => [ "field1", "pattern1", "field2", "pattern2" ]

### 字段引用(field reference)

字段是 Logstash::Event 对象的属性。我们之前提过事件就像一个哈希一样，所以你可以想象字段就像一个键值对。

小贴士：我们叫它字段，因为 Elasticsearch 里是这么叫的。

如果你想在 Logstash 配置中使用字段的值，只需要把字段的名字写在中括号 [] 里就行了，这就叫字段引用。

对于 嵌套字段(也就是多维哈希表，或者叫哈希的哈希)，每层的字段名都写在 [] 里就可以了。比如，你可以从 geoip 里这样获取 longitude 值(是的，这是个笨办法，实际上有单独的字段专门存这个数据的)：

[geoip][location][0]

小贴士：logstash 的数组也支持倒序下标，即 [geoip][location][-1] 可以获取数组最后一个元素的值。

Logstash 还支持变量内插，在字符串里使用字段引用的方法是这样：

"the longitude is %{[geoip][location][0]}"

### 条件判断(condition)

Logstash从 1.3.0 版开始支持条件判断和表达式。

表达式支持下面这些操作符：

|  |
| --- |
| equality, etc: ==, !=, <, >, <=, >=  regexp: =~, !~  inclusion: in, not in  boolean: and, or, nand, xor  unary: !() |

通常来说，你都会在表达式里用到字段引用。比如：

|  |
| --- |
| if "\_grokparsefailure" not in [tags] {  } else if [status] !~ /^2\d\d/ and [url] == "/noc.gif" {  } else {  } |

### 命令行参数

Logstash 提供了一个 shell 脚本叫 logstash 方便快速运行。它支持一下参数：

-e

意即执行。我们在 "Hello World" 的时候已经用过这个参数了。事实上你可以不写任何具体配置，直接运行 bin/logstash -e '' 达到相同效果。这个参数的默认值是下面这样：

|  |
| --- |
| input {  stdin { }  }  output {  stdout { }  } |

--config 或 -f

意即文件。真实运用中，我们会写很长的配置，甚至可能超过 shell 所能支持的 1024 个字符长度。所以我们必把配置固化到文件里，然后通过 bin/logstash -f agent.conf 这样的形式来运行。

此外，logstash 还提供一个方便我们规划和书写配置的小功能。你可以直接用 bin/logstash -f /etc/logstash.d/ 来运行。logstash 会自动读取 /etc/logstash.d/ 目录下所有的文本文件，然后在自己内存里拼接成一个完整的大配置文件，再去执行。

--configtest 或 -t

意即测试。用来测试 Logstash 读取到的配置文件语法是否能正常解析。Logstash 配置语法是用 grammar.treetop 定义的。尤其是使用了上一条提到的读取目录方式的读者，尤其要提前测试。

--log 或 -l

意即日志。Logstash 默认输出日志到标准错误。生产环境下你可以通过 bin/logstash -l logs/logstash.log 命令来统一存储日志。

--filterworkers 或 -w

意即工作线程。Logstash 会运行多个线程。你可以用 bin/logstash -w 5 这样的方式强制 Logstash 为过滤插件运行 5 个线程。

注意：Logstash目前还不支持输入插件的多线程。而输出插件的多线程需要在配置内部设置，这个命令行参数只是用来设置过滤插件的！

提示：Logstash 目前不支持对过滤器线程的监测管理。如果 filterworker 挂掉，Logstash 会处于一个无 filter 的僵死状态。这种情况在使用 filter/ruby 自己写代码时非常需要注意，很容易碰上 NoMethodError: undefined method '\*' for nil:NilClass 错误。需要妥善处理，提前判断。

--pluginpath 或 -P

可以写自己的插件，然后用 bin/logstash --pluginpath /path/to/own/plugins 加载它们。

小贴士：如果你使用的 Logstash 版本高于 1.5.0-rc3，该参数已经被取消，请阅读插件开发章节，改成本地 gem 插件安装形式。

--verbose

输出一定的调试日志。

小贴士：如果你使用的 Logstash 版本低于 1.3.0，你只能用 bin/logstash -v 来代替。

--debug

输出更多的调试日志。

小贴士：如果你使用的 Logstash 版本低于 1.3.0，你只能用 bin/logstash -vv 来代替。

## 启动配置

1、根目录 /usr/local/logstash

2、在根目录中创建 conf logs两个目录

3、在conf中创建如下两个文件

logstash.input.conf

|  |
| --- |
| input {  file {  path => [  "/var/local/webresource/logs/web-admin/controller.log",  "/var/local/webresource/logs/web-admin/service.log",  "/var/local/webresource/logs/web-admin/system.log"  ]  type => "web-admin"  start\_position => "beginning"  codec => "json"  }  file {  path => [  "/var/local/webresource/logs/web-mall/controller.log",  "/var/local/webresource/logs/web-mall/service.log",  "/var/local/webresource/logs/web-mall/system.log"  ]  type => "web-mall"  start\_position => "beginning"  codec => "json"  }  } |

logstash.output.conf

|  |
| --- |
| output{  stdout{  codec=>rubydebug  }  } |

或者推送到Elasticsearch中

|  |
| --- |
| output{  elasticsearch {  host => "elasticsearch00.neusoft.com"  protocol => "http"  index => "logstash-%{type}-%{+YYYY.MM.dd}"  index\_type => "%{type}"  workers => 5  template\_overwrite => false  }  } |

4、添加logstash 启动脚本 bin/startLogstash.sh

|  |
| --- |
| echo "start logstash service"  #nohup ./logstash -f ../conf/ -l ../logs/logstash.log 1> ../logs/logstash.out 2> ../logs/logstash.err &  #tail -fn 100 ../logs/logstash.log  nohup ./logstash -f ../conf/ 1> ../logs/logstash.out 2> ../logs/logstash.err &  tail -fn 100 ../logs/logstash.out |

5、停止

使用 ps -ef|grep logstash

kill -9 [pid]

## 输入插件(Input)

读取文件(File)

分析网站访问日志应该是一个运维工程师最常见的工作了。所以我们先学习一下怎么用 logstash 来处理日志文件。

Logstash 使用一个名叫 FileWatch 的 Ruby Gem 库来监听文件变化。这个库支持 glob 展开文件路径，而且会记录一个叫 .sincedb 的数据库文件来跟踪被监听的日志文件的当前读取位置。所以，不要担心 logstash 会漏过你的数据。

sincedb 文件中记录了每个被监听的文件的 inode, major number, minor number 和 pos。

配置示例

input {

file {

path => ["/var/log/\*.log", "/var/log/message"]

type => "system"

start\_position => "beginning"

}

}

# Kibana 安装与配置

要安装启动 Kibana:

下载对应平台的 Kibana 4 二进制包

解压 .zip 或 tar.gz 压缩文件

在安装目录里运行: bin/kibana (Linux/MacOSX) 或 bin\kibana.bat (Windows)

完毕！Kibana 现在运行在 5601 端口了。

## 启动配置

建立 logs目录

启动文件目录 ./bin

添加startKibana.sh

|  |
| --- |
| echo "start kibana service"  nohup ./kibana 1> ../logs/kibana.out 2> ../logs/kibana.err &  tail -fn 100 ../logs/kibana.out |

停止： ps -ef|grep kibana

kill -9 [pid]

## Kibana链接Elasticsearch配置

配置文件目录

./config/ kibana.yml