目录

[ElK简介 2](#_Toc525818591)

[ELK技术栈 3](#_Toc525818592)

[Logstash 3](#_Toc525818593)

[Input plugin 4](#_Toc525818594)

[filter plugin 6](#_Toc525818595)

[Output plugin 6](#_Toc525818596)

[Elasticsearch 7](#_Toc525818597)

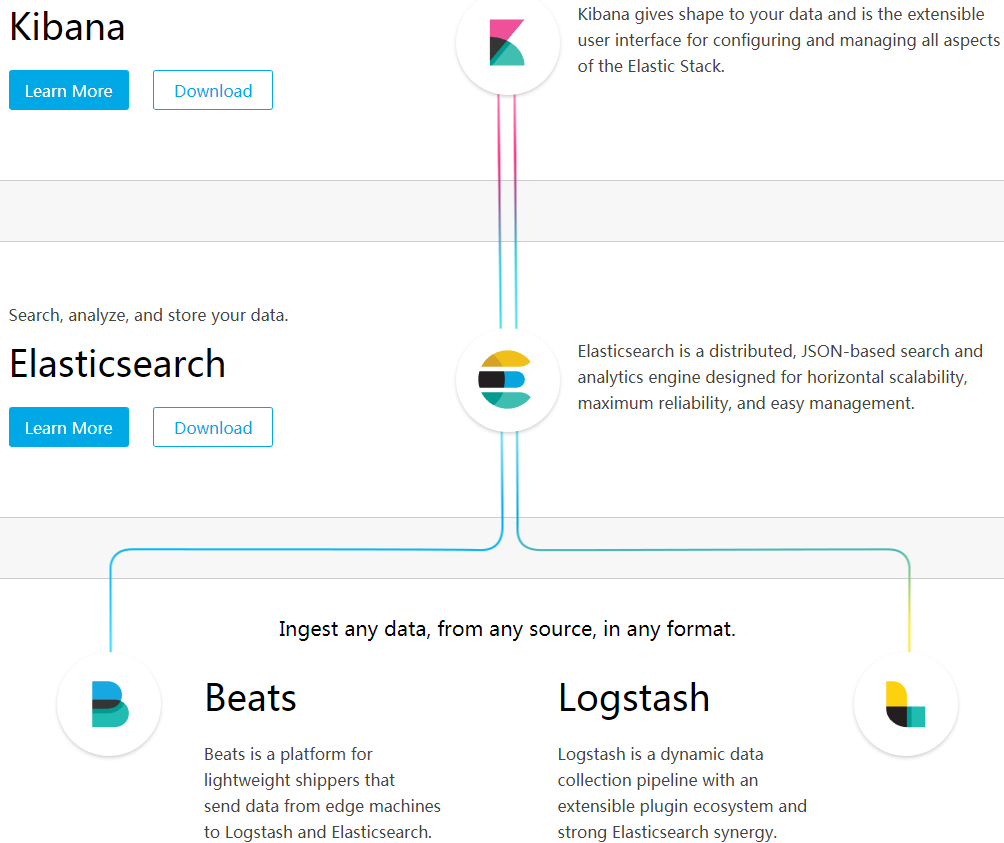
[基本概念 7](#_Toc525818598)

[常规操作 8](#_Toc525818599)

[附件导入 11](#_Toc525818600)

[Kibana 12](#_Toc525818601)

# ElK简介



ELK实际上是三个工具的集合（Elasticsearch + Logstash + Kibana），这三个工具组合形成了一套实用、易用的监控架构，很多公司利用它来搭建可视化的海量日志分析平台。

ELK的优势

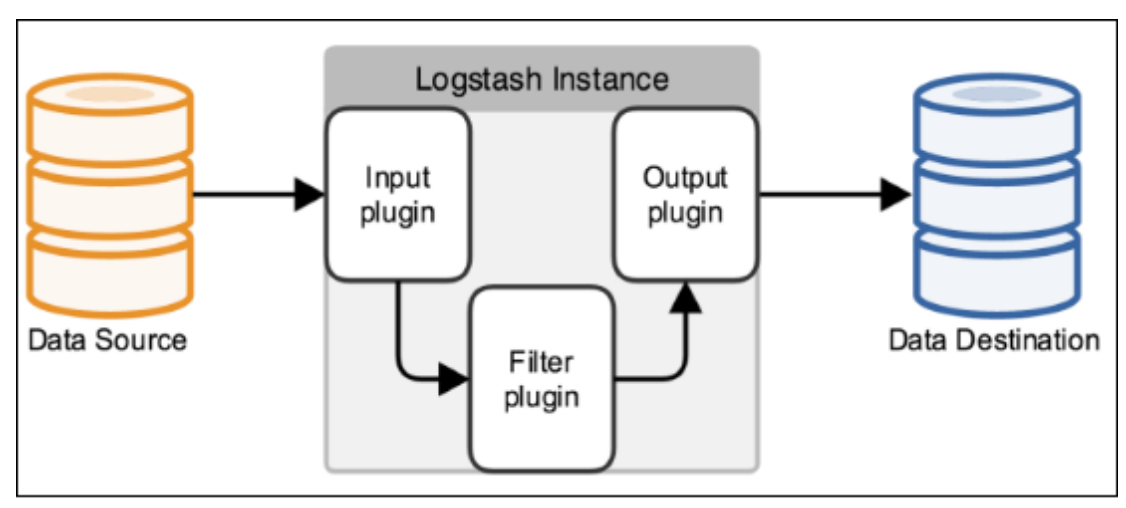
1. 强大的搜索功能

2. 完美的展示功能

3. 分布式功能。

# ELK技术栈

## Logstash



Logstash 是开源的服务器端数据处理管道，能够同时从多个来源采集数据，转换数据，然后将数据发送到您最喜欢的 “存储库” 中

Logstash要求Java8，目前不支持Java9.（Logstash6.4）

不需要安装，下载解压后，进入相关目录，在命令窗口可以直接启动使用。简单示例：

logstash -e "input { stdin { } } output { stdout { codec => rubydebug } }"

其中stdin、stdout表示控制台输入和输出。

为了适用于不同的场景，需要编写管道配置文件，下面为管道配置文件的基本框架：

input {

}

# The filter part of this file is commented out to indicate that it is

# optional.

# filter {

# }

output {

}

### Input plugin

#### logstash-input-jdbc

Jdbc input plugin作为logstash的一个插件，貌似已经在下载解压后的logstash中存在了，并不需要再按照，可以使用Jdbc input plugin同步数据库中的数据。

input {

jdbc {

#驱动，布置在了文件夹logstash-6.4.0\logstash-core\lib\jars

jdbc\_driver\_library => "postgresql-42.2.2.jar"

jdbc\_driver\_class => "org.postgresql.Driver"

jdbc\_connection\_string => "jdbc:postgresql://localhost:5432/equip"

jdbc\_user => "postgres"

jdbc\_password => "Abc12345"

#执行的sql语句，也可以以statement\_filepath属性代替，赋值sql文件路径+名称

statement => "SELECT d.id, u.user\_name, t.type\_name, m.device\_name from tb\_device\_detail d left join tb\_user u on d.user\_id=u.id left join tb\_device\_map m on d.device\_id = m.id left join tb\_device\_type t on m.type\_code = t.type\_code"

}

}

在同步数据时也可以实现增量新增数据，需要设置以下几个字段的值：

使用其他字段追踪，而不是默认的时间: use\_column\_value => true;

设置追踪的字段：tracking\_column => id

记录追踪字段最后的值和记录的文件位置：record\_last\_run => true;last\_run\_metadata\_path => “……”

设置监听间隔，分时天月年： schedule => “\*\*\*\*\*”

具体可以参考附件jdbc-pipeline.conf

若要验证配置，请运行以下命令：

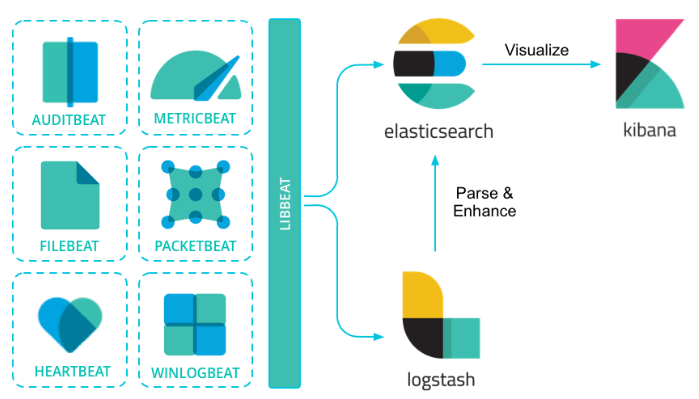
bin/logstash -f jdbc-pipeline.conf --config.test\_and\_exit

如果配置文件通过配置测试，则使用以下命令启动logstash：

bin/logstash -f jdbc-pipeline.conf --config.reload.automatic

#### logstash-input-beats

Beats 平台集合了多种单一用途数据采集器。这些采集器安装后可用作轻量型代理，从成百上千或成千上万台机器向 Logstash 或 Elasticsearch 发送数据



不同的数据类型、特点会有不同的beat与其对应。本文主要学习了filebeat。

filebeat会监控日志目录或者指定的日志文件，追踪读取这些文件（追踪文件的变化，不停的读），并且转发这些信息到elasticsearch或者logstarsh中存放。

下载解压，改写filebeat.yml配置文件后，通过命令窗口可以直接使用。

Filebeat.inputs.paths指定要收集的日志的路径，将enabled设为true来使input配置生效。

将结果输出到logstash：output.logstash.hosts: [“localhost:5044”]

具体见附件filebeat.yml.

在命令窗口输入以下内容启动filebeat：

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

filebeat根据.yml文件中的配置内容，将采集到的数据发送到了logstash中。在logstash管道配置文件中添加：input { beats {port => "5044"} }即可。

### filter plugin

过滤器插件在事件执行的中间过程中对结果进行处理。过滤器通常根据事件的特性有条件地应用。

#### Grok filter plugin

Grok提供了一种将非结构化日志数据解析成结构化、可查询的数据的方式。该工具非常适合于syslog日志、apache和其他web服务器日志、mysql日志，以及通常为开发人员自定义、而不利于计算机识别的任何日志格式。

Logstash提供了120种默认的patterns，参见：<https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns>. 。也可以通过patters\_dir自定义。

对于日志中的内容“127.0.0.1 GET /auth-control/user/login 200 1527”，可以通过：

grok{match=>{"message"=> "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{HOSTNAME:porot} %{NUMBER:duration}"}}

与其进行匹配。文本中的内容必须与grok中的正则式匹配对应，否则过滤器不起作用。Grok发挥作用时，client被赋值127.0.0.1，method赋值GET……

### Output plugin

#### File output plugin

通过此插件可以将logstash处理后的结果以文件的形式保存到指定路径。

output {

file {

path => "/path/log"

}

}

#### Elasticsearch output plugin

可以使用ElasticSearch output plugin将采集处理后发送到ElasticSeatch。

output {

elasticsearch {

hosts => [ "localhost:9200" ]

index => "mycase"

document\_id => "%{id}"

}

}

字段index指定了数据存储到的index，document\_id指定了每条记录（document）的唯一标识。如果不指定index，默认是logstash-${yyyy.mm.dd}的形式；document\_id会是随机序列数（在做通过jdbc与数据库全量同步时，因为生成的document\_id不同，相同的记录被当成了不同的）。

## Elasticsearch

ElasticSearch是一个高度可扩展的开源全文搜索和分析引擎。它允许您快速、实时地存储、搜索和分析大量数据。它通常用作底层引擎/技术，为具有复杂搜索特征和要求的应用程序提供动力

### 基本概念

Near Realtime

弹性搜索是一种近实时搜索平台。这意味着从索引文档到可搜索文档，会有轻微的延迟（通常是1秒）

Index

Index是具有相似特性的文档集合。索引由名称（必须全部小写）标识，该名称用于在对索引中的文档执行搜索、更新和删除操作时引用。在单个群集中，可以根据需要定义尽可能多的索引。

Type

Type属对Index里面的内容进行逻辑分区，以便于存储在同一个index中的不同类型数据。在索引中不再可能创建多个类型，并且在稍后的版本中将删除类型的全部概念（and the whole concept of types will be removed in a later version.）。

Document

Document是可被索引的信息的基本单位，以JSON表示。

在index/type中，可以按需要存储多个document。注意，尽管document物理上驻留在index中，但是document实际上必须被索引/分配给index中的type。

Shards&replicas

ElasticSearch可以将一个index的数据细分到不同的区片中——shards。当创建index时，可以自定义shards的数量。每一个shards本身是一个完全有效的、独立的“index”，可以在集群中的任何节点上托管。（分布式）。

因为会有不可预料的宕机情况，为了保证数据的安全、可用，每一个shard都至少有一个与其对应的备份——replica shards。默认情况下，会有5个primary shards和1个replica（每一个index有10个shards）。

### 常规操作

查看elasticsearch的集群、节点、index状况，可以调用\_cat API.

GET /\_cat/health?v

GET /\_cat/nodes?v

GET /\_cat/indices?v

#### 查询

有两种方式可以来执行查询：一种是通过REST request URI传输查询参数，一种是通过REST request body。下面两种方式得到的结果相同。

GET /bank/\_search?q=\*&sort=account\_number:asc&pretty

GET /bank/\_search

{

"query": { "match\_all": {} },

"sort": [

{ "account\_number": "asc" }

]

}

可以通过设置一些字段名来筛选查询结果。比如from、size指定数据的的起始位置和个数，\_source指定显示的数据字段。

模糊匹配，返回的结果是包含“mill”或“lane”的数据

GET /bank/\_search

{"query": { "match": { "address": "mill lane" } } }

短语精确匹配，返回的结果是address为“mill lane”的数据

GET /bank/\_search

{ "query": { "match\_phrase": { "address": "mill lane" } } }

返回结果包含“mill”和“lane”的数据

GET /bank/\_search

{ "query": {

"bool": {

"must": [{ "match": { "address": "mill" } },{ "match": { "address": "lane" } }]

}}}

类似的还有should、must\_not。对于多字段查询，可以使用“multi\_match”。

也可以对查询结果添加filter进行过滤。

GET /bank/\_search

{

"query": {

"bool": {

"must": { "match\_all": {} },

"filter": {

"range": {

"balance": { "gte": 20000, "lte": 30000

} } } } }

}

#### 添加

除了通过logstash中的elasticsearch output plugin添加外，还可以通过put指令。

PUT /customer/\_doc/1?pretty

{ "name": "John Doe" }

在没有命名为customer的index时，此指令会创建一个名为customer的index，并且在以\_doc为名的type下，指定id为1的数据中name存储“Johe Doe”。

#### 更新

对于数据的更新有两种方式。

1.和赋值操作一样，使用put，所有字段必须全部列出。更新后\_version加1（与乐观锁相关）。

2.使用\_update api。可以只列出要更新的字段。

POST /customer/\_doc/1/\_update?pretty

{ "doc": { "name": "Jane Doe" } }

#### 删除

DELETE /index\_name?pretty

DELETE /index\_name/type\_name /id?pretty

删除document不会立即将文档从磁盘中删除，只是将document标记为已删除状态。随着不断的索引更多的数据，Elasticsearch 将会在后台清理标记为已删除的document。

#### 聚合函数

Elasticsearch同样支持聚合函数

GET /bank/\_search

{

"size": 0,

"aggs": {

"group\_by\_state": {

"terms": {

"field": "state.keyword"

}

}

}

}

Size设置为0是为了出聚合结果外，让其他数据结果不显示。

因为在Kibana中默认显示10条数据，所以上述语句在SQL中从概念上类似于：

SELECT state, COUNT(\*) FROM bank GROUP BY state ORDER BY COUNT(\*) DESC LIMIT 10;

另外，可以通过avg查询平均值

### 附件导入

ElasticSearch只能处理文本，不能直接处理文档。要实现 ElasticSearch 的附件导入需要对多种主流格式的文档进行文本抽取并将抽取出来的文本内容导入 ElasticSearch。

使用Ingest Attachment Processor Plugin可以实现对（PDF,DOC等）主流格式文件的文本抽取及自动导入。

安装：bin/elasticsearch-plugin install ingest-attachment

安装后重启elasticsearch即可使用。

首先要建立自己的文本抽取管道

PUT \_ingest/pipeline/attachment

{ "description" : "Extract attachment information",

"processors" : [

{ "attachment" : { "field" : "data" }

} ]

}

然后在创建document时指定到此管道：

PUT my\_index/\_doc/my\_id?pipeline=attachment

{

"data": "e1xydGYxXGFuc2kNCkxvcmVtIGlwc3VtIGRvbG9yIHNpdCBhbWV0DQpccGFyIH0="

}

其中字段data的赋值为附件内容通过base64编码后的字符串。然后就可以通过get指令查看附件内容了。

GET my\_index/\_doc/my\_id

源字段必须是Base64编码二进制。如果不想引起在base64之间来回转换的开销，可以将字段指定为字节数组而不是字符串表示。处理器将跳过Base64解码。

对于文件，可以在后台使用java代码将其通过base64编码后存入data字段即可。

为了防止提取太多的字符和节点内存过载，用于提取的字符数量默认限制为100000。可以通过设置indexed\_chars来改变此值。使用-1没有限制，但是确保在设置该值时，节点将具有足够的HEAP来提取非常大的文档的内容

在提取的文本中查找内容，并将查找内容高亮，示例：

GET mypdf/\_doc/\_search

{"query": {

"bool": {

"must": [

{"match\_phrase": { "attachment.content":"违规庭审" }}

]

}

},

"\_source": "attachment",

"highlight": {

"fields": { "attachment.content": {} }

}

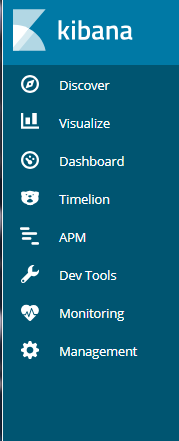
}

其中，“\_source”指定了只显示附件内容，而不显示data字段了。如果只关心要查找的数据（即高亮的内容），则可以将\_source设为“”。

## Kibana

Kibana 是一个开源的分析和可视化平台，旨在与 Elasticsearch 合作。Kibana 提供搜索、查看和与存储在 Elasticsearch 索引中的数据进行交互的功能。开发者或运维人员可以轻松地执行高级数据分析，并在各种图表、表格和地图中可视化数据。

在下载解压后，通过命令窗口/bin/kibana启动运行，通过浏览器访问localhost:5601进入Kibana界面



在界面左侧列出了Kibana提供的几个功能模块。

上文介绍的在ElasticSearch中的常规操作都是在Kibana中的Dev Tools模块Console下进行的。

下面简单介绍一下与数据可视化相关的一些操作

1）Management>>Kibana\_Index Patterns>>Create Index pattern。这里创建的index pattern是为了discovery、visualize服务的。

2）然后在Disvocer中就可以选择创建的Index pattern，可以点击Available fields下的字段后面add将其置为Seleted fields，这样主页面就会以列表形式显示被选择字段。同时可以通过add a filter对结果进行过滤。可以点击Save，将此设置和结果保存，提供给Visualize使用。

3）Visualize提供数据的可视化服务，点击Visualize，列表显示的是已经存在的图表。点击+，创建新的图表。——选择合适的Visualize type，然后在再选择数据的来源index 或者之前保存的search结果。将可以绘制图表了。将绘制后的图表进行保存，方便Dashboard使用

4）点击Dashboard，显示的是已经存在的dashboads。选择create new dashboard >> add,弹出的列表显示已经存在的visualize图表，选择后即被添加的dashboard中。也可以选择add new visualize，新建图表。选择完毕后点击空白处返回dashboard。可以认为dashboard提供了visualize图表的集中管理和展示。点击save保存。

## 为elasticsearch安装可视化插件head

参考：<https://blog.csdn.net/u012270682/article/details/72934270>

1.首先安装node

安装完成后，在安装目录下命令行执行node –v查看安装版本判断是否成功。

2.在同一目录下执行npm install –g grunt-cli ,安装grunt。用grunt –versin查看版本号。

3.安装head ：

修改elasticsearch.yml，在文件最后添加：

http.cors.enabled: true

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

node.master: true

node.data: true

放开network.host: 192.168.0.1的注释并改为network.host: 0.0.0.0

放开cluster.name；node.name；http.port的注释

双击elasticsearch.bat重启es。

<https://github.com/mobz/elasticsearch-head> 下载zip文件，解压到指定文件夹，我的文件夹是D:\elasticsearch-6.4.0\elasticsearch-head-master，（注意，不要放大plugin下，因为启动时会检索报错，没有安装这个插件）。

修改Gruntfile.js。在connect：{server：{options：{下添加hostname：‘\*’

在D:\elasticsearch-6.4.0\elasticsearch-head-master文件夹执行 npm install安装。安装完成执行grunt server或者npm runstart（以后每次）启动。

安装完成查看结果127.0.0.1:9100.

