一、search template

搜索模板，search template，高级功能，就可以将我们的一些搜索进行模板化，然后的话，每次执行这个搜索，就直接调用模板，给传入一些参数就可以了

1 template入门案例

简单定义参数并传递

```
GET /cars/_search/template
{
"source" : {
"query" : {
"match" : {
"remark" : "{{kw}}"
}
},
"size" : "{{size}}"
 },
 "params": {
 "kw" : "大众",
 "size" : 2
 }
 }
```

toJson方式传递参数

```
GET cars/_search/template
{
"source": "{ \"query\": { \"match\": {{#toJson}}parameter{{/toJson}} }}",
"params": {
"parameter" : {
"remark" : "大众"
}
}
}
```

join方式传递参数

```
GET cars/_search/template
{
"source" : {
```

```
"query" : {
"match" : {
"remark" : "{{#join delimiter=' '}}kw{{/join delimiter=' '}}"
}
}
},
 "params": {
"kw" : ["大众", "标致"]
 }
 }
```

default value定义：

```
GET cars/_search/template
{
"source" : {
"query" : {
"range" : {
"price" : {
"gte" : "{{start}}",
"lte" : "{{end}}{{^end}}200000{{/end}}"
}
 }
 }
 },
 "params": {
"start" : 100000
 }
 }
```

2 记录template实现重复调用

可以使用Mustache语言作为搜索请求的预处理，它提供了模板，然后通过键值对来替换模板中的变量。把脚本存储在本地磁盘中，默认的位置为：elasticsearch\config\scripts，通过引用脚本名称进行使用

2.1 保存template到ES

```
POST _scripts/test
{
"script": {
"lang": "mustache",
"source": {
```

```
"query": {
"match" : {
"remark" : "{{kw}}"
}
 }
 }
 }
 }
```

2.2 调用template执行搜索

```
GET cars/_search/template
{
"id": "test",
"params": {
"kw": "大众"
}
}
```

2.3 查询已定义的template

```
GET _scripts/test
```

2.4 删除已定义的template

```
DELETE _scripts/test
```

二、suggest search(completion suggest)

suggest search（completion suggest）：就是建议搜索或称为搜索建议，也可以叫做自动完成-auto completion。类似百度中的搜索联想提示功能。

ES实现suggest的时候，性能非常高，其构建的不是倒排索引，也不是正排索引，就是纯的用于进行前缀搜索的一种特殊的数据结构，而且会全部放在内存中，所以suggest search进行的前缀搜索提示，性能是非常高。

需要使用suggest的时候，必须在定义index时，为其mapping指定开启suggest。具体如下：

```
PUT /movie
{
"mappings": {
"properties" : {
"title" : {
```

```
"type": "text",
"analyzer": "ik_max_word",
"fields": {
"suggest" : {
 "type" : "completion",
 "analyzer": "ik_max_word"
 }
 }
 },
 "content": {
 "type": "text",
 "analyzer": "ik_max_word"
 }
 }
 }
 }

PUT /movie/_doc/1
{
"title": "西游记电影系列",
"content": "西游记之月光宝盒将与2021年进行......"
}

PUT /movie/_doc/2
{
"title": "西游记文学系列",
"content": "某知名网络小说作家已经完成了大话西游同名小说的出版"
}

PUT /movie/_doc/3
{
"title": "西游记之大话西游手游",
"content": "网易游戏近日出品了大话西游经典IP的手游，正在火爆内测中"
}

```

suggest 搜索：

```
GET /movie/_search
{
"suggest": {
```

```
"my-suggest" : {
"prefix" : "西游记",
"completion" : {
"field" : "title.suggest"
}
}
}
}
```

三、geo point - 地理位置搜索和聚合分析

ES支持地理位置的搜索和聚合分析，可实现在指定区域内搜索数据、搜索指定地点附近的数据、聚合分析指定地点附近的数据等操作。

ES中如果使用地理位置搜索的话，必须提供一个特殊的字段类型。GEO - geo_point。地理位置的坐标点。

1、定义geo point mapping

如果需要使用地址坐标，则需要定义一个指定的mapping类型。具体如下：

使用什么数据可以确定，地球上的一个具体的点？经纬度。

```
PUT /hotel_app
{
"mappings": {
"properties": {
"pin": {
"type": "geo_point"
},
"name" : {
"type" : "text",
"analyzer": "ik_max_word"
}
}
}
}
```

2、录入数据

新增一个基于geo point类型的数据，可以使用多种方式。

多种类型描述geo_point类型字段的时候，在搜索数据的时候，显示的格式和录入的格式是统一的。不影响搜索。任何数据描述的geo_point类型字段，都

适用地理位置搜索。

数据范围要求：纬度范围是-90~90之间，经度范围是-180~180之间。经纬度数据都是浮点数或数字串（数字组成的字符串），最大精度：小数点后7位。（常用小数点后6位即可。）

基于对象：latitude：纬度、longitude：经度。*语义清晰，建议使用。*

```
PUT /hotel_app/_doc/1
{
"name": "七天连锁酒店",
"pin" : {
"lat" : 40.12,
"lon" : -71.34
}
}
```

基于字符串：依次定义纬度、经度。不推荐使用

```
PUT /hotel_app/_doc/2
{
"name": "维多利亚大酒店",
"pin" : "40.99, -70.81"
}
```

基于数组：依次定义经度、纬度。不推荐使用

```
PUT /hotel_app/_doc/3
{
"name": " 红树林宾馆",
"pin" : [40, -73.81]
}
```

3、搜索指定区域范围内的数据

总结：

矩形范围搜索：传入的top_left和bottom_right坐标点是有固定要求的。地图中以北作为top，南作为bottom，西作为left，东作为right。也就是top_left应该从西北向东南。Bottom_right应该从东南向西北。Top_left的纬度应该大于bottom_right的纬度，top_left的经度应该小于bottom_right的经度。

多边形范围搜索：对传入的若干点的坐标顺序没有任何的要求。只要传入若干地理位置坐标点，即可形成多边形。

搜索矩形范围内的数据

```
GET /hotel_app/_doc/_search
{
"query": {
"bool": {
"must": [
{
"match_all": {}
}
],
 "filter": {
 "geo_bounding_box": {
 "pin": {
 "top_left" : {
 "lat" : 41.73,
 "lon" : -74.1
 },
 "bottom_right" : {
 "lat" : 40.01,
 "lon" : -70.12
 }
 }
 }
 }
 }
 }
 }

GET /hotel_app/_doc/_search
{
"query": {
"constant_score": {
"filter": {
"geo_bounding_box": {
"pin": {
"top_left": {
"lat": -70,
```

```
"lon": 39
},
"bottom_right": {
"lat": -75,
"lon": 41
}
}
}
}
}
}
}
```

搜索多边形范围内的数据

```
GET /hotel_app/_doc/_search
{
"query": {
"bool": {
"must": [
{
"match_all": {}
}
],
 "filter": {
 "geo_polygon": {
 "pin": {
 "points": [
 {"lat" : 40.73, "lon" : -74.1},
 {"lat" : 40.01, "lon" : -71.12},
 {"lat" : 50.56, "lon" : -90.58}
 ]
 }
 }
 }
 }
 }
```

4、搜索某地点附近的数据

这个搜索在项目中更加常用。类似附近搜索功能。

Distance距离的单位，常用的有米（m）和千米（km）。

建议使用filter来过滤geo_point数据。因为geo_point数据相关度评分计算比较耗时。使用query来搜索geo_point数据效率相对会慢一些。建议使用filter。

```
GET /hotel_app/_doc/_search
{
"query": {
"bool": {
"must": [
{
"match_all": {}
}
],
 "filter": {
 "geo_distance": {
 "distance": "200km",
 "pin": {
 "lat": 40,
 "lon": -70
 }
 }
 }
 }
 }

 GET hotel_app/_search
 {
 "query": {
 "geo_distance" : {
 "distance" : "90km",
 "pin" : {
 "lat" : 40.55,
 "lon" : -71.12
 }
 }
 }
 }
```

5、统计某位置附近区域内的数据

聚合统计分别距离某位置80英里，300英里，1000英里范围内的数据数量。

其中unit是距离单位，常用单位有：米（m），千米（km），英里（mi）

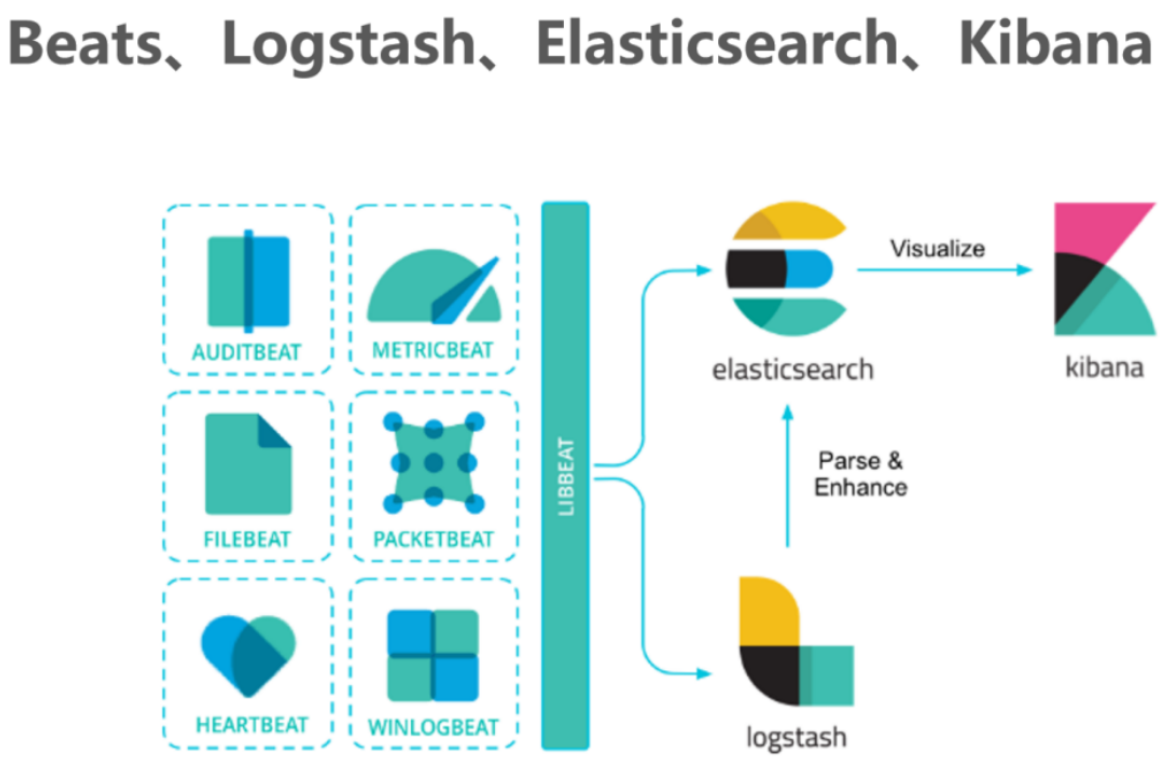
distance_type是统计算法：sloppy_arc默认算法、arc最高精度、plane最高效率

```
GET /hotel_app/_doc/_search
{
"size": 0,
"aggs": {
"agg_by_pin" : {
"geo_distance": {
"distance_type": "arc",
"field": "pin",
"origin": {
 "lat": 40,
 "lon": -70
 },
 "unit": "mi",
 "ranges": [
 {
 "to": 80
 },
 {
 "from": 80,
 "to": 300
 },
 {
 "from": 300,
 "to": 1000
 }
 ]
 }
 }
 }
 }
```

四、Beats

Beats是一个开放源代码的数据发送器。我们可以把Beats作为一种代理安装在我们的服务器上，这样就可以比较方便地将数据发送到Elasticsearch或者Logstash中。Elastic Stack提供了多种类型的Beats组件。

审计数据	AuditBeat
日志文件	FileBeat
云数据	FunctionBeat
可用性数据	HeartBeat
系统日志	JournalBeat
指标数据	MetricBeat
网络流量数据	PacketBeat
Windows事件日志	Winlogbeat



Beats可以直接将数据发送到Elasticsearch或者发送到Logstash，基于Logstash可以进一步地对数据进行处理，然后将处理后的数据存入到Elasticsearch，最后使用Kibana进行数据可视化。

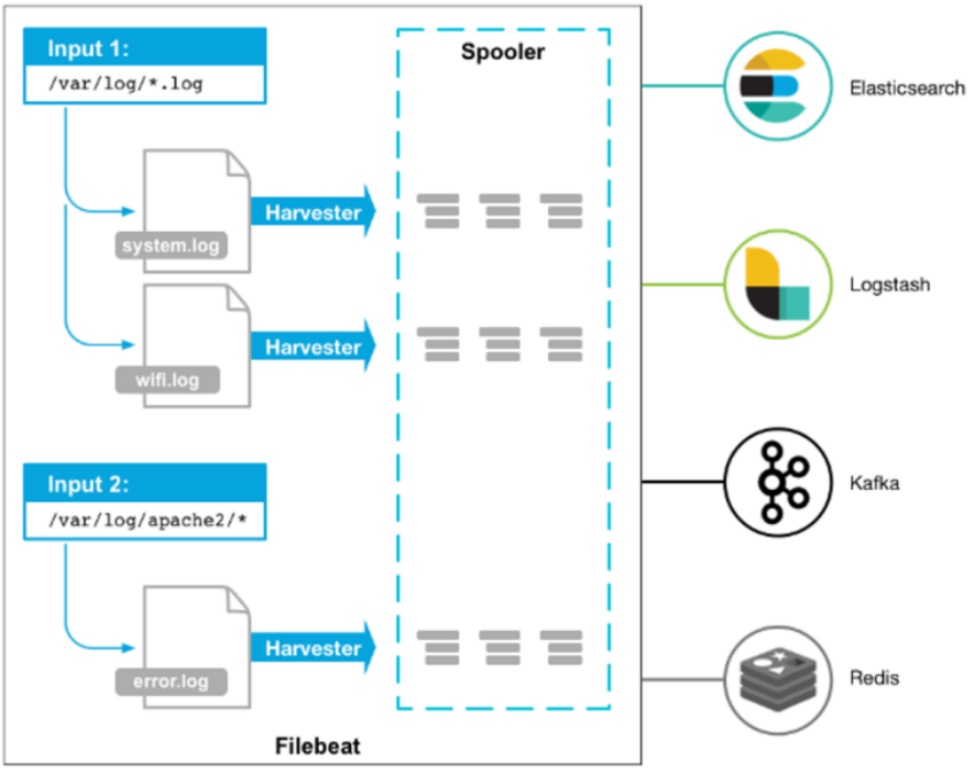
1、FileBeat简介

FileBeat专门用于转发和收集日志数据的轻量级采集工具。它可以为作为代理安装在服务器上，FileBeat监视指定路径的日志文件，收集日志数据，并将收集到的日志转发到Elasticsearch或者Logstash。

2、FileBeat的工作原理

启动FileBeat时，会启动一个或者多个输入（Input），这些Input监控指定的日志数据位置。FileBeat会针对每一个文件启动一个Harvester（收割机）。Harvester读取每一个文件的日志，将新的日志发送到libbeat，libbeat将数据收集到一起，并将数据发送给输出（Output）。

FileBeat工作原理



3、安装FileBeat

安装FileBeat只需要将FileBeat Linux安装包上传到Linux系统，并将压缩包解压到系统就可以了。

FileBeat官方下载地址：

https://www.elastic.co/cn/downloads/past-releases/filebeat-7-6-1

上传FileBeat安装到Linux，并解压。

```
tar -xvzf filebeat-7.6.1-linux-x86_64.tar.gz -C ../usr/local/es/
```

4、使用FileBeat采集MQ日志到Elasticsearch

4.1、需求分析

在资料中有一个mq_server.log.tar.gz压缩包，里面包含了很多的MQ服务器日志，现在我们为了通过在Elasticsearch中快速查询这些日志，定位问题。我们需要用FileBeats将日志数据上传到Elasticsearch中。

问题：

首先，我们要指定FileBeat采集哪些MQ日志，因为FileBeats中必须知道采集存放在哪儿的日志，才能进行采集。

其次，采集到这些数据后，还需要指定FileBeats将采集到的日志输出到Elasticsearch，那么Elasticsearch的地址也必须指定。

4.2、配置FileBeats

FileBeats配置文件主要分为两个部分。

1. inputs
2. output

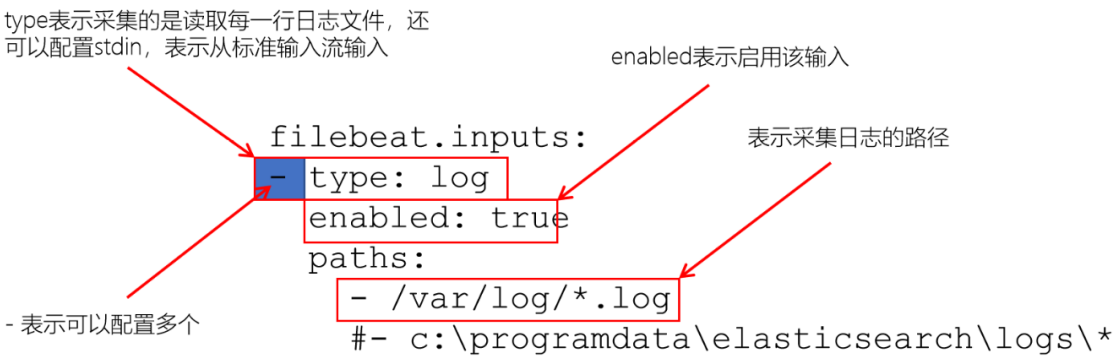
从名字就能看出来，一个是用来输入数据的，一个是用来输出数据的。

4.2.1、input配置

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/*.log
    #- c:\programdata\elasticsearch\logs\*
```

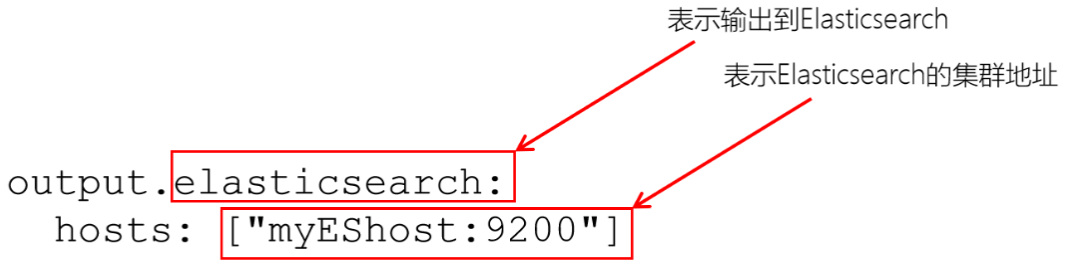
在FileBeats中，可以读取一个或多个数据源。

FileBeats配置文件 - input



4.2.2、output配置

FileBeat配置文件 - output



默认FileBeat会将日志数据放入到名称为：filebeat-%filebeat版本号%-yyyy.MM.dd 的索引中。

PS：

FileBeats中的filebeat.reference.yml包含了FileBeats所有支持的配置选项。

4.3、配置文件

1. 创建配置文件

```
cd /usr/local/es/filebeat-7.6.1-linux-x86_64
touch filebeat_mq_log.yml
vim filebeat_mq_log.yml
```

2. 复制以下到配置文件中

```
filebeat.inputs:
- type: log
  enabled: true
```

```
  paths:
    - /var/mq/log/server.log.*

output.elasticsearch:
    hosts: ["192.168.21.130:9200", "192.168.21.131:9200", "192.168.21.132:9200"]
```

4.4、运行FileBeat

1. 启动Elasticsearch

在每个节点上执行以下命令，启动Elasticsearch集群：

```
nohup /usr/local/es/elasticsearch-7.6.1/bin/elasticsearch 2>&1 &
```

2. 运行FileBeat

```
./filebeat -c filebeat_mq_log.yml -e
```

3. 将日志数据上传到/var/mq/log，并解压

```
mkdir -p /var/mq/log
cd /var/mq/log
tar -zxvf mq_server.log.tar.gz
```

4.5、查询数据

通过head插件，我们可以看到filebeat采集了日志消息，并写入到Elasticsearch集群中。

五、FileBeat是如何工作的

FileBeat主要由input和harvesters（收割机）组成。这两个组件协同工作，并将数据发送到指定的输出。

1、input和harvester

1.1、inputs（输入）

input是负责管理Harvesters和查找所有要读取的文件的组件

如果输入类型是 log，input组件会查找磁盘上与路径描述的所有文件，并为每个文件启动一个Harvester，每个输入都独立地运行

1.2、Harvesters（收割机）

Harvesters负责读取单个文件的内容，它负责打开/关闭文件，并逐行读取每个文件的内容，将读取到的内容发送给输出

每个文件都会启动一个Harvester

Harvester运行时，文件将处于打开状态。如果文件在读取时，被移除或者重命名，FileBeat将继续读取该文件

2、FileBeats如何保持文件状态

FileBeat保存每个文件的状态，并定时将状态信息保存在磁盘的「注册表」文件中

该状态记录Harvester读取的最后一次偏移量，并确保发送所有的日志数据

如果输出（Elasticsearch或者Logstash）无法访问，FileBeat会记录成功发送的最后一行，并在输出（Elasticsearch或者Logstash）可用时，继续读取文件发送数据

在运行FileBeat时，每个input的状态信息也会保存在内存中，重新启动FileBeat时，会从「注册表」文件中读取数据来重新构建状态。

'

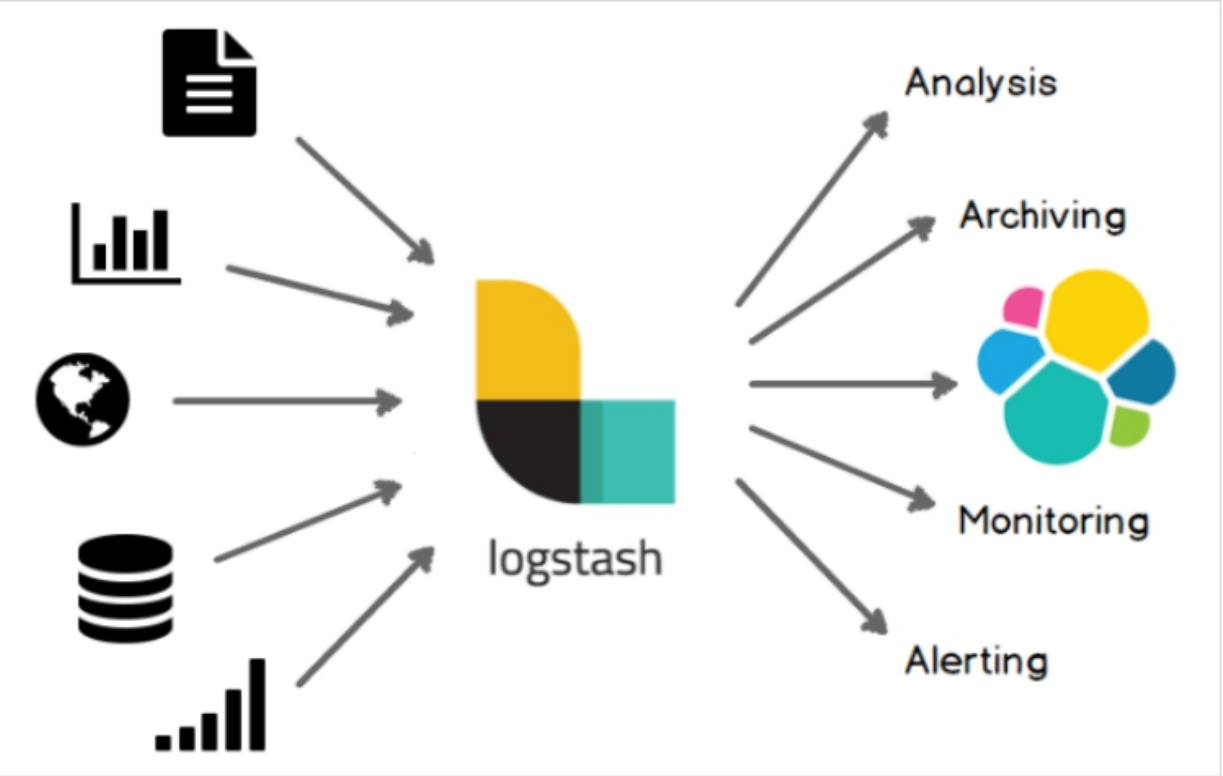
在/usr/local/es/filebeat-7.6.1-linux-x86_64/data目录中有一个Registry文件夹，里面有一个data.json，该文件中记录了Harvester读取日志的offset。

六. Logstash

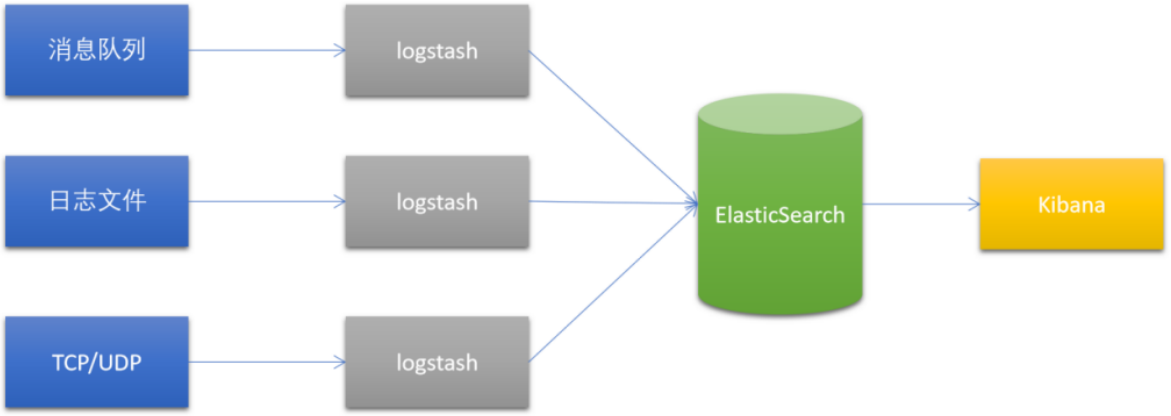
1、简介

Logstash是一个开源的数据采集引擎。它可以动态地将不同来源的数据统一采集，并按照指定的数据格式进行处理后，将数据加载到其他的目的地。最开始，Logstash主要是针对日志采集，但后来Logstash开发了大量丰富的插件，所以，它可以做更多的海量数据的采集。

它可以处理各种类型的日志数据，例如：Apache的web log、Java的log4j日志数据，或者是系统、网络、防火墙的日志等等。它也可以很容易的和Elastic Stack的Beats组件整合，也可以很方便的和关系型数据库、NoSQL数据库、MQ等整合。



1.1 经典架构



1.2 对比FileBeat

logstash是jvm跑的，资源消耗比较大

而FileBeat是基于golang编写的，功能较少但资源消耗也比较小，更轻量级

logstash 和filebeat都具有日志收集功能，Filebeat更轻量，占用资源更少

logstash 具有filter功能，能过滤分析日志

一般结构都是filebeat采集日志，然后发送到消息队列，redis，MQ中然后logstash去获取，利用filter功能过滤分析，然后存储到elasticsearch中

FileBeat和Logstash配合，实现背压机制

2 安装Logstash和Kibana

2.1 安装Logstash

1. 下载Logstash

```
https://www.elastic.co/cn/downloads/past-releases/logstash-7-6-1
```

此处：我们可以选择资料中的logstash-7.6.1.zip安装包。

2. 解压Logstash到指定目录

```
unzip logstash-7.6.1 -d /usr/local/es/
```

3. 运行测试

```
cd /usr/local/es/logstash-7.6.1/
bin/logstash -e 'input { stdin { } } output { stdout {} }'
```

等待一会，让Logstash启动完毕。

```
Sending Logstash logs to /usr/local/es/logstash-7.6.1/logs which is now co
nfigured via log4j2.properties
[2021-02-28T16:31:44,159][WARN ][logstash.config.source.multilocal] Ignori
ng the 'pipelines.yml' file because modules or command line options are spec
fied
[2021-02-28T16:31:44,264][INFO ][logstash.runner          ] Starting Logst
ash {"logstash.version"=>"7.6.1"}
[2021-02-28T16:31:45,631][INFO ][org.reflections.Reflections] Reflections
took 37 ms to scan 1 urls, producing 20 keys and 40 values
[2021-02-28T16:31:46,532][WARN ][org.logstash.instrument.metrics.gauge.Laz
yDelegatingGauge][main] A gauge metric of an unknown type (org.jruby.RubyArra
y) has been create for key: cluster_uuids. This may result in invalid seriali
zation.  It is recommended to log an issue to the responsible developer/devel
opment team.
[2021-02-28T16:31:46,560][INFO ][logstash.javapipeline    ][main] Starting
pipeline {:pipeline_id=>"main", "pipeline.workers"=>2, "pipeline.batch.size"=
>125, "pipeline.batch.delay"=>50, "pipeline.max_inflight"=>250, "pipeline.sou
rces"=>["config string"], :thread=>"#<Thread:0x3ccbc15b run>"}
[2021-02-28T16:31:47,268][INFO ][logstash.javapipeline    ][main] Pipeline
started {"pipeline.id"=>"main"}
The stdin plugin is now waiting for input:
[2021-02-28T16:31:47,348][INFO ][logstash.agent           ] Pipelines runn
ing {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[]}
[2021-02-28T16:31:47,550][INFO ][logstash.agent           ] Successfully s
tarted Logstash API endpoint {:port=>9600}
```

然后，随便在控制台中输入内容，等待Logstash的输出。

```
{
"host" => "127.0.0.1",
"message" => "hello logstash",
"@version" => "1",
"@timestamp" => 2021-02-28:01:01.007Z
}
```

ps:

-e选项表示，直接把配置放在命令中，这样可以有效快速进行测试

文档：06 ElasticSearch笔记.note

链接：http://note.youdao.com/noteshare?id=f8813b31f33964c3acd84494fea22e46&sub=D1415954CB354F14A42A5C6FAA44ADFB