## 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。

### 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。

### 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）。

### 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

### 使用FileBeat采集Kafka日志到Elasticsearch

#### 需求分析

在资料中有一个kafka\_server.log.tar.gz压缩包，里面包含了很多的Kafka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Kafka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。

#### 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。

##### 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。

##### output配置

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。

#### 配置文件

1. 创建配置文件

|  |
| --- |
| cd /export/server/es/filebeat-7.6.1-linux-x86\_64  vim filebeat\_kafka\_log.yml |

1. 复制一下到配置文件中

|  |
| --- |
| filebeat.inputs:  - type: log  enabled: true  paths:  - /export/server/es/data/kafka/server.log.\*  output.elasticsearch:  hosts: ["node1:9200", "node2:9200", "node3:9200"] |

#### 运行FileBeat

1. 运行FileBeat

./filebeat -c filebeat\_kafka\_log.yml -e

1. 将日志数据上传到/var/kafka/log，并解压

mkdir -p /export/server/es/data/kafka/

tar -xvzf kafka\_server.log.tar.gz

注意: 文件权限的报错

如果在启动fileBeat的时候, 报了一个配置文件权限的错误, 请修改其权限为 -rw-r--r--

#### 查询数据

1. 查看索引信息

GET /\_cat/indices?v

|  |
| --- |
| {          "health": "green",          "status": "open",          "index": "filebeat-7.6.1-2020.05.29-000001",          "uuid": "dplqB\_hTQq2XeSk6S4tccQ",          "pri": "1",          "rep": "1",          "docs.count": "213780",          "docs.deleted": "0",          "store.size": "71.9mb",          "pri.store.size": "35.8mb"      } |

1. 查询索引库中的数据

GET /filebeat-7.6.1-2020.05.29-000001/\_search

|  |
| --- |
| {                  "\_index": "filebeat-7.6.1-2020.05.29-000001",                  "\_type": "\_doc",                  "\_id": "-72pX3IBjTeClvZff0CB",                  "\_score": 1,                  "\_source": {                      "@timestamp": "2020-05-29T09:00:40.041Z",                      "log": {                          "offset": 55433,                          "file": {                              "path": "/var/kafka/log/server.log.2020-05-02-16"                          }                      },                      "message": "[2020-05-02 16:01:30,682] INFO Socket connection established, initiating session, client: /192.168.88.100:46762, server: node1 /192.168.88.100:2181 (org.apache.zookeeper.ClientCnxn)",                      "input": {                          "type": "log"                      },                      "ecs": {                          "version": "1.4.0"                      },                      "host": {                          "name": "node1 "                      },                      "agent": {                          "id": "b4c5c4dc-03c3-4ba4-9400-dc6afcb36d64",                          "version": "7.6.1",                          "type": "filebeat",                          "ephemeral\_id": "b8fbf7ab-bc37-46dd-86c7-fa7d74d36f63",                          "hostname": "node1 "                      }                  }              } |

FileBeat自动给我们添加了一些关于日志、采集类型、Host各种字段。

#### 解决一个日志涉及到多行问题

我们在日常日志的处理中，经常会碰到日志中出现异常的情况。类似下面的情况：

|  |
| --- |
| [2020-04-30 14:00:05,725] WARN [ReplicaFetcher replicaId=0, leaderId=1, fetcherId=0] Error when sending leader epoch request for Map(test\_10m-2 -> (currentLeaderEpoch=Optional[161], leaderEpoch=158)) (kafka.server.ReplicaFetcherThread)  java.io.IOException: Connection to node2:9092 (id: 1 rack: null) failed.  at org.apache.kafka.clients.NetworkClientUtils.awaitReady(NetworkClientUtils.java:71)  at kafka.server.ReplicaFetcherBlockingSend.sendRequest(ReplicaFetcherBlockingSend.scala:102)  at kafka.server.ReplicaFetcherThread.fetchEpochEndOffsets(ReplicaFetcherThread.scala:310)  at kafka.server.AbstractFetcherThread.truncateToEpochEndOffsets(AbstractFetcherThread.scala:208)  at kafka.server.AbstractFetcherThread.maybeTruncate(AbstractFetcherThread.scala:173)  at kafka.server.AbstractFetcherThread.doWork(AbstractFetcherThread.scala:113)  at kafka.utils.ShutdownableThread.run(ShutdownableThread.scala:96)  [2020-04-30 14:00:05,725] INFO [ReplicaFetcher replicaId=0, leaderId=1, fetcherId=0] Retrying leaderEpoch request for partition test\_10m-2 as the leader reported an error: UNKNOWN\_SERVER\_ERROR (kafka.server.ReplicaFetcherThread)  [2020-04-30 14:00:08,731] WARN [ReplicaFetcher replicaId=0, leaderId=1, fetcherId=0] Connection to node 1 (node2/192.168.88.101:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient) |

在FileBeat中，Harvest是逐行读取日志文件的。但上述的日志会出现一条日志，跨多行的情况。有异常信息时，肯定会出现多行。我们先来看一下，如果默认不处理这种情况会出现什么问题。

##### 导入错误日志

1. 在/export/server/es/data/kafka/中创建名为server.log.2020-09-10的日志文件
2. 将资料中的err.txt日志文本贴入到该文件中

观察FileBeat，发现FileBeat已经针对该日志文件启动了Harvester，并读取到数据数据。

2020-05-29T19:11:01.236+0800 INFO log/harvester.go:297 Harvester started for file: /var/kafka/log/server.log.2020-09-10

1. 在Elasticsearch检索该文件。

|  |
| --- |
| GET /filebeat-7.6.1-2020.05.29-000001/\_search  {      "query": {          "match": {              "log.file.path": "/var/kafka/log/server.log.2020-09-10"          }      }  } |

我们发现，原本是一条日志中的异常信息，都被作为一条单独的消息来处理了~



这明显是不符合我们的预期的，我们想要的是将所有的异常消息合并到一条日志中。那针对这种情况该如何处理呢？

##### 问题分析

每条日志都是有统一格式的开头的，就拿Kafka的日志消息来说，[2020-04-30 14:00:05,725]这是一个统一的格式，如果不是以这样的形式开头，说明这一行肯定是属于某一条日志，而不是独立的一条日志。所以，我们可以通过日志的开头来判断某一行是否为新的一条日志。

##### FileBeat多行配置选项

在FileBeat的配置中，专门有一个解决一条日志跨多行问题的配置。主要为以下三个配置：

|  |
| --- |
| multiline.pattern: ^\[  multiline.negate: false  multiline.match: after |

multiline.pattern表示能够匹配一条日志的模式，默认配置的是以[开头的才认为是一条新的日志。

multiline.negate:配置该模式是否生效，默认为false。

multiline.match:表示是否将未匹配到的行追加到上一日志，还是追加到下一个日志。

##### 重新配置FileBeat

1. 修改filebeat.yml，并添加以下内容

|  |
| --- |
| filebeat.inputs:  - type: log  enabled: true  paths:  - /var/kafka/log/server.log.\*  multiline.pattern: '^\['  multiline.negate: true  multiline.match: after  output.elasticsearch:  hosts: ["node1:9200", "node2:9200", "node3:9200"] |

1. 修改「注册表」/data.json，将server.log.2020-09-10对应的offset设置为0

cd /export/server/es/filebeat-7.6.1-linux-x86\_64/data/registry/filebeat

vim data.json

1. 删除之前创建的文档

|  |
| --- |
| // 删除指定文件的文档  POST /filebeat-7.6.1-2020.05.29-000001/\_delete\_by\_query  {      "query": {          "match": {              "log.file.path": "/var/kafka/log/server.log.2020-09-10"          }      }  } |

1. 重新启动FileBeat

./filebeat -e

### 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。

#### input和harvester

##### 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。

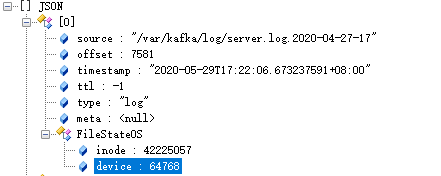
##### Harvesters（收割机）

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，并将读取到的内容发送给输出，每个文件都会启动一个Harvester。但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。

#### 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/export/server/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。



## Logstash

### 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、Kafka、RabbitMQ等整合。

#### 经典架构

#### 对比Flume

1. Apache Flume是一个通用型的数据采集平台，它通过配置source、channel、sink来实现数据的采集，支持的平台也非常多。而Logstash结合Elastic Stack的其他组件配合使用，开发、应用都会简单很多
2. Logstash比较关注数据的预处理，而Flume跟偏重数据的传输，几乎没有太多的数据解析预处理，仅仅是数据的产生，封装成Event然后传输。

#### 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，kafka中然后logstash去获取，利用filter功能过滤分析，然后存储到elasticsearch中

### 安装Logstash

1. 切换到roy用户
2. 下载Logstash

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

1. 解压Logstash到指定目录
2. 运行测试

|  |
| --- |
| cd /export/server/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 /export/server/es/logstash-7.6.1/logs which is now configured via log4j2.properties  [2020-05-28T16:31:44,159][WARN ][logstash.config.source.multilocal] Ignoring the 'pipelines.yml' file because modules or command line options are specified  [2020-05-28T16:31:44,264][INFO ][logstash.runner ] Starting Logstash {"logstash.version"=>"7.6.1"}  [2020-05-28T16:31:45,631][INFO ][org.reflections.Reflections] Reflections took 37 ms to scan 1 urls, producing 20 keys and 40 values  [2020-05-28T16:31:46,532][WARN ][org.logstash.instrument.metrics.gauge.LazyDelegatingGauge][main] A gauge metric of an unknown type (org.jruby.RubyArray) has been create for key: cluster\_uuids. This may result in invalid serialization. It is recommended to log an issue to the responsible developer/development team.  [2020-05-28T16:31:46,560][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.sources"=>["config string"], :thread=>"#<Thread:0x3ccbc15b run>"}  [2020-05-28T16:31:47,268][INFO ][logstash.javapipeline ][main] Pipeline started {"pipeline.id"=>"main"}  The stdin plugin is now waiting for input:  [2020-05-28T16:31:47,348][INFO ][logstash.agent ] Pipelines running {:count=>1, :running\_pipelines=>[:main], :non\_running\_pipelines=>[]}  [2020-05-28T16:31:47,550][INFO ][logstash.agent ] Successfully started 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" => "node1",  "message" => "hello world",  "@version" => "1",  "@timestamp" => 2020-05-28T08:32:31.007Z  } |

ps：

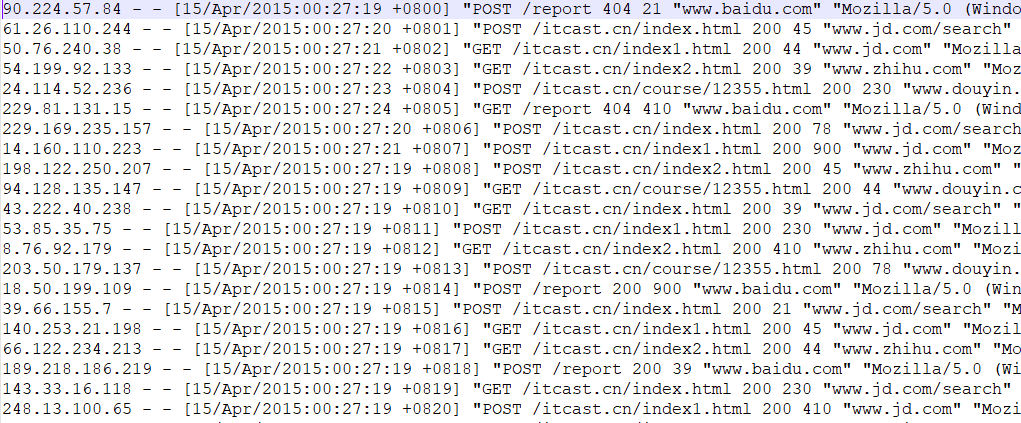
-e选项表示，直接把配置放在命令中，这样可以有效快速进行测试

### 采集Apache Web服务器日志

#### 需求

Apache的Web Server会产生大量日志，当我们想要对这些日志检索分析。就需要先把这些日志导入到Elasticsearch中。此处，我们就可以使用Logstash来实现日志的采集。

打开这个文件，如下图所示。我们发现，是一个纯文本格式的日志。如下图所示：



这个日志其实由一个个的字段拼接而成，参考以下表格。

|  |  |
| --- | --- |
| 字段名 | 说明 |
| client IP | 浏览器端IP |
| timestamp | 请求的时间戳 |
| method | 请求方式（GET/POST） |
| uri | 请求的链接地址 |
| status | 服务器端响应状态 |
| length | 响应的数据长度 |
| reference | 从哪个URL跳转而来 |
| browser | 浏览器 |

因为最终我们需要将这些日志数据存储在Elasticsearch中，而Elasticsearch是有模式（schema）的，而不是一个大文本存储所有的消息，而是需要将字段一个个的保存在Elasticsearch中。所以，我们需要在Logstash中，提前将数据解析好，将日志文本行解析成一个个的字段，然后再将字段保存到Elasticsearch中。

#### 准备日志数据

将Apache服务器日志上传到 /export/server/es/data/apache/ 目录

mkdir -p /export/server/es/data/apache/

#### 使用FileBeats将日志发送到Logstash

在使用Logstash进行数据解析之前，我们需要使用FileBeat将采集到的数据发送到Logstash。之前，我们使用的FileBeat是通过FileBeat的Harvester组件监控日志文件，然后将日志以一定的格式保存到Elasticsearch中，而现在我们需要配置FileBeats将数据发送到Logstash。FileBeat这一端配置以下即可：

|  |
| --- |
| #----------------------------- Logstash output ---------------------------------  #output.logstash:  # Boolean flag to enable or disable the output module.  #enabled: true  # The Logstash hosts  #hosts: ["localhost:5044"] |

hosts配置的是Logstash监听的IP地址/机器名以及端口号。

114.113.220.255

**准备FileBeat配置文件**

cd /export/server/es/filebeat-7.6.1-linux-x86\_64

vim filebeat-logstash.yml

因为Apache的web log日志都是以IP地址开头的，所以我们需要修改下匹配字段。

|  |
| --- |
| filebeat.inputs:  - type: log  enabled: true  paths:  - /var/apache/log/access.\*  multiline.pattern: '^\d+\.\d+\.\d+\.\d+ '  multiline.negate: true  multiline.match: after  output.logstash:  enabled: true  hosts: ["node1:5044"] |

**启动FileBeat，并指定使用新的配置文件**

./filebeat -e -c filebeat-logstash.yml

FileBeat将尝试建立与Logstash监听的IP和端口号进行连接。但此时，我们并没有开启并配置Logstash，所以FileBeat是无法连接到Logstash的。

|  |
| --- |
| 2020-06-01T11:28:47.585+0800 ERROR pipeline/output.go:100 Failed to connect to backoff(async(tcp://node1:5044)): dial tcp 192.168.88.100:5044: connect: connection refused |

#### 配置Logstash接收FileBeat数据并打印

Logstash的配置文件和FileBeat类似，它也需要有一个input、和output。基本格式如下：

|  |
| --- |
| # #号表示添加注释  # input表示要接收的数据  input {  }  # file表示对接收到的数据进行过滤处理  filter {  }  # output表示将数据输出到其他位置  output {  } |

**配置从FileBeat接收数据**

cd /export/server/es/logstash-7.6.1/config

vim filebeat-print.conf

|  |
| --- |
| input {  beats {  port => 5044  }  }  output {  stdout {  codec => rubydebug  }  } |

**测试logstash配置是否正确**

bin/logstash -f config/filebeat-print.conf --config.test\_and\_exit

|  |
| --- |
| [2020-06-01T11:46:33,940][INFO ][logstash.runner ] Using config.test\_and\_exit mode. Config Validation Result: OK. Exiting Logstash |

**启动logstash**

bin/logstash -f config/filebeat-print.conf --config.reload.automatic

reload.automatic：修改配置文件时自动重新加载

**测试**

创建一个access.log.1文件，使用cat test >> access.log.1往日志文件中追加内容。

test文件中只保存一条日志：

|  |
| --- |
| [root@node1 log]# cat test  235.9.200.242 - - [15/Apr/2015:00:27:19 +0849] "POST /node1/bigdata.html 200 45 "www.baidu.com" "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4549.400 QQBrowser/9.7.12900 144.180.122.249 |

当我们启动Logstash之后，就可以发现Logstash会打印出来从FileBeat接收到的数据：

|  |
| --- |
| {  "log" => {  "file" => {  "path" => "/var/apache/log/access.log.1"  },  "offset" => 825  },  "input" => {  "type" => "log"  },  "agent" => {  "ephemeral\_id" => "d4c3b652-4533-4ebf-81f9-a0b78c0d4b05",  "version" => "7.6.1",  "type" => "filebeat",  "id" => "b4c5c4dc-03c3-4ba4-9400-dc6afcb36d64",  "hostname" => "node1 "  },  "@timestamp" => 2020-06-01T09:07:55.236Z,  "ecs" => {  "version" => "1.4.0"  },  "host" => {  "name" => "node1 "  },  "tags" => [  [0] "beats\_input\_codec\_plain\_applied"  ],  "message" => "235.9.200.242 - - [15/Apr/2015:00:27:19 +0849] \"POST /node1/bigdata.html 200 45 \"www.baidu.com\" \"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4549.400 QQBrowser/9.7.12900 144.180.122.249",  "@version" => "1"  } |

#### Logstash输出数据到Elasticsearch

通过控制台，我们发现Logstash input接收到的数据没有经过任何处理就发送给了output组件。而其实我们需要将数据输出到Elasticsearch。所以，我们修改Logstash的output配置。配置输出Elasticsearch只需要配置以下就可以了：

|  |
| --- |
| output {  elasticsearch {  hosts => [ "localhost:9200" ]  }} |

操作步骤：

1. 重新拷贝一份配置文件

cp filebeat-print.conf filebeat-es.conf

1. 将output修改为Elasticsearch

|  |
| --- |
| input {  beats {  port => 5044  }  }  output {  elasticsearch {  hosts => [ "node1:9200","node2:9200","node3:9200"]  }  } |

1. 重新启动Logstash

bin/logstash -f config/filebeat-es.conf --config.reload.automatic

1. 追加一条日志到监控的文件中，并查看Elasticsearch中的索引、文档

cat test >> access.log.1

// 查看索引数据

GET /\_cat/indices?v

我们在Elasticsearch中发现一个以logstash开头的索引。

|  |
| --- |
| {          "health": "green",          "status": "open",          "index": "logstash-2020.06.01-000001",          "uuid": "147Uwl1LRb-HMFERUyNEBw",          "pri": "1",          "rep": "1",          "docs.count": "2",          "docs.deleted": "0",          "store.size": "44.8kb",          "pri.store.size": "22.4kb"      } |

// 查看索引库的数据

GET /logstash-2020.06.01-000001/\_search?format=txt

{

    "from": 0,

    "size": 1

}

我们可以获取到以下数据：

|  |
| --- |
| "@timestamp": "2020-06-01T09:38:00.402Z",                      "tags": [                          "beats\_input\_codec\_plain\_applied"                      ],                      "host": {                          "name": "node1 "                      },                      "@version": "1",                      "log": {                          "file": {                              "path": "/var/apache/log/access.log.1"                          },                          "offset": 1343                      },                      "agent": {                          "version": "7.6.1",                          "ephemeral\_id": "d4c3b652-4533-4ebf-81f9-a0b78c0d4b05",                          "id": "b4c5c4dc-03c3-4ba4-9400-dc6afcb36d64",                          "hostname": "node1 ",                          "type": "filebeat"                      },                      "input": {                          "type": "log"                      },                      "message": "235.9.200.242 - - [15/Apr/2015:00:27:19 +0849] \"POST /node1/bigdata.html 200 45 \"www.baidu.com\" \"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4549.400 QQBrowser/9.7.12900 144.180.122.249",                      "ecs": {                          "version": "1.4.0"                      } |

从输出返回结果，我们可以看到，日志确实已经保存到了Elasticsearch中，而且我们看到消息数据是封装在名为**message**中的，其他的数据也封装在一个个的字段中。我们其实更想要把消息解析成一个个的字段。例如：IP字段、时间、请求方式、请求URL、响应结果，这样。

#### Logstash过滤器

在Logstash中可以配置过滤器Filter对采集到的数据进行中间处理，在Logstash中，有大量的插件供我们使用。参考官网：

<https://www.elastic.co/guide/en/logstash/7.6/filter-plugins.html>

此处，我们重点来讲解Grok插件。

##### 查看Logstash已经安装的插件

bin/logstash-plugin list

##### Grok插件

Grok是一种将非结构化日志解析为结构化的插件。这个工具非常适合用来解析系统日志、Web服务器日志、MySQL或者是任意其他的日志格式。

Grok官网：<https://www.elastic.co/guide/en/logstash/7.6/plugins-filters-grok.html>

##### Grok语法

Grok是通过模式匹配的方式来识别日志中的数据,可以把Grok插件简单理解为升级版本的**正则表达式**。它拥有更多的模式，默认，Logstash拥有120个模式。如果这些模式不满足我们解析日志的需求，我们可以直接使用正则表达式来进行匹配。

官网：<https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns>

grok模式的语法是：%{SYNTAX:SEMANTIC}

SYNTAX指的是Grok模式名称，SEMANTIC是给模式匹配到的文本字段名。例如：

|  |
| --- |
| %{NUMBER:duration} %{IP:client}  duration表示：匹配一个数字，client表示匹配一个IP地址。 |

默认在Grok中，所有匹配到的的数据类型都是字符串，如果要转换成int类型（目前只支持int和float），可以这样：%{NUMBER:duration:int} %{IP:client}

以下是常用的Grok模式：

|  |  |
| --- | --- |
| NUMBER | 匹配数字（包含：小数） |
| INT | 匹配整形数字 |
| POSINT | 匹配正整数 |
| WORD | 匹配单词 |
| DATA | 匹配所有字符 |
| IP | 匹配IP地址 |
| PATH | 匹配路径 |

##### 用法

|  |
| --- |
| filter {  grok {  match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }  }  } |

#### 匹配日志中的IP、日期并打印

|  |
| --- |
| 235.9.200.242 - - [15/Apr/2015:00:27:19 +0849] "POST /node1/bigdata.html 200 45 "www.baidu.com" "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4549.400 QQBrowser/9.7.12900 144.180.122.249 |

我们使用IP就可以把前面的IP字段匹配出来，使用HTTPDATE可以将后面的日期匹配出来。

**配置Grok过滤插件**

1. 配置Logstash

|  |
| --- |
| input {  beats {  port => 5044  }  }  filter {  grok {  match => {  "message" => "%{IP:ip} - - \[%{HTTPDATE:date}\]"  }  }  }  output {  stdout {  codec => rubydebug  }  } |

1. 启动Logstash

bin/logstash -f config/filebeat-filter-print.conf --config.reload.automatic

|  |
| --- |
| {  "log" => {  "offset" => 1861,  "file" => {  "path" => "/var/apache/log/access.log.1"  }  },  "input" => {  "type" => "log"  },  "tags" => [  [0] "beats\_input\_codec\_plain\_applied"  ],  "date" => "15/Apr/2015:00:27:19 +0849",  "ecs" => {  "version" => "1.4.0"  },  "@timestamp" => 2020-06-01T11:02:05.809Z,  "message" => "235.9.200.242 - - [15/Apr/2015:00:27:19 +0849] \"POST /node1/bigdata.html 200 45 \"www.baidu.com\" \"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4549.400 QQBrowser/9.7.12900 144.180.122.249",  "host" => {  "name" => "node1 "  },  "ip" => "235.9.200.242",  "agent" => {  "hostname" => "node1 ",  "version" => "7.6.1",  "ephemeral\_id" => "d4c3b652-4533-4ebf-81f9-a0b78c0d4b05",  "id" => "b4c5c4dc-03c3-4ba4-9400-dc6afcb36d64",  "type" => "filebeat"  },  "@version" => "1"  } |

我们看到，经过Grok过滤器插件处理之后，我们已经获取到了ip和date两个字段。接下来，我们就可以继续解析其他的字段。

#### 解析所有字段

将日志解析成以下字段：

|  |  |
| --- | --- |
| 字段名 | 说明 |
| client IP | 浏览器端IP |
| timestamp | 请求的时间戳 |
| method | 请求方式（GET/POST） |
| uri | 请求的链接地址 |
| status | 服务器端响应状态 |
| length | 响应的数据长度 |
| reference | 从哪个URL跳转而来 |
| browser | 浏览器 |

1. 修改Logstash配置文件

|  |
| --- |
| input {  beats {  port => 5044  }  }  filter {  grok {  match => {  "message" => "%{IP:ip} - - \[%{HTTPDATE:date}\] \"%{WORD:method} %{PATH:uri} %{DATA}\" %{INT:status} %{INT:length} \"%{DATA:reference}\" \"%{DATA:browser}\""  }  }  }  output {  stdout {  codec => rubydebug  }  } |

1. 测试并启动Logstash

我们可以看到，8个字段都已经成功解析。

|  |
| --- |
| {  "reference" => "www.baidu.com",  "@version" => "1",  "ecs" => {  "version" => "1.4.0"  },  "@timestamp" => 2020-06-02T03:30:10.048Z,  "ip" => "235.9.200.241",  "method" => "POST",  "uri" => "/node1/bigdata.html",  "agent" => {  "id" => "b4c5c4dc-03c3-4ba4-9400-dc6afcb36d64",  "ephemeral\_id" => "734ae9d8-bcdc-4be6-8f97-34387fcde972",  "version" => "7.6.1",  "hostname" => "node1 ",  "type" => "filebeat"  },  "length" => "45",  "status" => "200",  "log" => {  "file" => {  "path" => "/var/apache/log/access.log"  },  "offset" => 1  },  "input" => {  "type" => "log"  },  "host" => {  "name" => "node1 "  },  "tags" => [  [0] "beats\_input\_codec\_plain\_applied"  ],  "browser" => "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4549.400 QQBrowser/9.7.12900",  "date" => "15/Apr/2015:00:27:19 +0849",  "message" => "235.9.200.241 - - [15/Apr/2015:00:27:19 +0849] \"POST /node1 /bigdata.html HTTP/1.1\" 200 45 \"www.baidu.com\" \"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4549.400 QQBrowser/9.7.12900\""  } |

#### 将数据输出到Elasticsearch

到目前为止，我们已经通过了Grok Filter可以将日志消息解析成一个一个的字段，那现在我们需要将这些字段保存到Elasticsearch中。我们看到了Logstash的输出中，有大量的字段，但如果我们只需要保存我们需要的8个，该如何处理呢？而且，如果我们需要将日期的格式进行转换，我们又该如何处理呢？

##### 过滤出来需要的字段

要过滤出来我们需要的字段。我们需要使用mutate插件。mutate插件主要是作用在字段上，例如：它可以对字段进行重命名、删除、替换或者修改结构。

官方文档：<https://www.elastic.co/guide/en/logstash/7.6/plugins-filters-mutate.html>

例如，mutate插件可以支持以下常用操作

配置：

注意：此处为了方便进行类型的处理，将status、length指定为int类型。

|  |
| --- |
| input {  beats {  port => 5044  }  }  filter {  grok {  match => {  "message" => "%{IP:ip} - - \[%{HTTPDATE:date}\] \"%{WORD:method} %{PATH:uri} %{DATA}\" %{INT:status:int} %{INT:length:int} \"%{DATA:reference}\" \"%{DATA:browser}\""  }  }  mutate {  split => ["hostName", "."]  add\_field => {"shortHostName" => "%{hostname[0]}"}  enable\_metric => "false"  remove\_field => ["message", "log", "tags", "@timestamp", "input", "agent", "host", "ecs", "@version"]  munate {  rename => ["shortHostName", "hostName"]  }  }  }  output {  stdout {  codec => rubydebug  }  } |
|  |

##### 转换日期格式

要将日期格式进行转换，我们可以使用Date插件来实现。该插件专门用来解析字段中的日期，官方说明文档：<https://www.elastic.co/guide/en/logstash/7.6/plugins-filters-date.html>

用法如下：

将date字段转换为「年月日 时分秒」格式。默认字段经过date插件处理后，会输出到@timestamp字段，所以，我们可以通过修改target属性来重新定义输出字段。

Logstash配置修改为如下：

|  |
| --- |
| input {  beats {  port => 5044  }  }  filter {  grok {  match => {  "message" => "%{IP:ip} - - \[%{HTTPDATE:date}\] \"%{WORD:method} %{PATH:uri} %{DATA}\" %{INT:status:int} %{INT:length:int} \"%{DATA:reference}\" \"%{DATA:browser}\""  }  }  mutate {  enable\_metric => "false"  remove\_field => ["message", "log", "tags", "@timestamp", "input", "agent", "host", "ecs", "@version"]  }  date {  match => ["date","dd/MMM/yyyy:HH:mm:ss Z","yyyy-MM-dd HH:mm:ss"]  target => "date"  }  }  output {  stdout {  codec => rubydebug  }  } |

启动Logstash：

bin/logstash -f config/filebeat-filter-print.conf --config.reload.automatic

|  |
| --- |
| {  "status" => "200",  "reference" => "www.baidu.com",  "method" => "POST",  "browser" => "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.104 Safari/537.36 Core/1.53.4549.400 QQBrowser/9.7.12900",  "ip" => "235.9.200.241",  "length" => "45",  "uri" => "/node1 /bigdata.html",  "date" => 2015-04-14T15:38:19.000Z  } |

##### 输出到Elasticsearch指定索引

我们可以通过

elasticsearch {

hosts => ["node1:9200" ,"node2:9200" ,"node3:9200"]

index => "xxx"

}

index来指定索引名称，默认输出的index名称为：logstash-%{+yyyy.MM.dd}。但注意，要在index中使用时间格式化，filter的输出必须包含 @timestamp字段，否则将无法解析日期。

|  |
| --- |
| input {  beats {  port => 5044  }  }  filter {  grok {  match => {  "message" => "%{IP:ip} - - \[%{HTTPDATE:date}\] \"%{WORD:method} %{PATH:uri} %{DATA}\" %{INT:status:int} %{INT:length:int} \"%{DATA:reference}\" \"%{DATA:browser}\""  }  }  mutate {  enable\_metric => "false"  remove\_field => ["message", "log", "tags", "input", "agent", "host", "ecs", "@version"]  }  date {  match => ["date","dd/MMM/yyyy:HH:mm:ss Z","yyyy-MM-dd HH:mm:ss"]  target => "date"  }  }  output {  stdout {  codec => rubydebug  }  elasticsearch {  hosts => ["node1:9200" ,"node2:9200" ,"node3:9200"]  index => "apache\_web\_log\_%{+YYYY-MM}"  }  } |

启动Logstash

bin/logstash -f config/filebeat-apache-weblog.conf --config.test\_and\_exit

bin/logstash -f config/filebeat-apache-weblog.conf --config.reload.automatic

注意：

* index名称中，不能出现大写字符

## Kibana

### 简介

通过上面的这张图就可以看到，Kibana可以用来展示丰富的图表。

* Kibana是一个开源的数据分析和可视化平台，使用Kibana可以用来搜索Elasticsearch中的数据，构建漂亮的可视化图形、以及制作一些好看的仪表盘
* Kibana是用来管理Elastic stack组件的可视化平台。例如：使用Kibana可以进行一些安全设置、用户角色设置、对Elasticsearch进行快照等等
* Kibana提供统一的访问入口，不管是日志分析、还是查找文档，Kibana提供了一个使用这些功能的统一访问入口
* Kibana使用的是Elasticsearch数据源，Elasticsearch是存储和处理数据的引擎，而Kibana就是基于Elasticsearch之上的可视化平台

主要功能：

* 探索和查询Elasticsearch中的数据
* 可视化与分析

### 安装Kibana

在Linux下安装Kibana，可以使用Elastic stack提供 tar.gz压缩包。官方下载地址：

<https://www.elastic.co/cn/downloads/past-releases/kibana-7-6-1>

1. 解压Kibana gz压缩包

tar -xzf kibana-7.6.2-linux-x86\_64.tar.gz

1. 进入到Kibana目录

cd kibana-7.6.2-linux-x86\_64/

1. 配置Kibana

|  |
| --- |
| # The Kibana server's name. This is used for display purposes.  server.name: "kibana"  # The URLs of the Elasticsearch instances to use for all your queries.  elasticsearch.hosts: ["http://node1:9200","http://node2:9200","http://node3:9200"] |

1. 运行Kibana

./bin/kibana

#### 查看Kibana状态

输入以下网址，可以查看到Kibana的运行状态：

[http://node1:5601/status](http://node1.itcast.cn:5601/status)

#### 查看Elasticsearch的状态

点击  按钮，再点击 「Index Management」，可以查看到Elasticsearch集群中的索引状态。

点击索引的名字，可以进一步查看索引更多的信息。

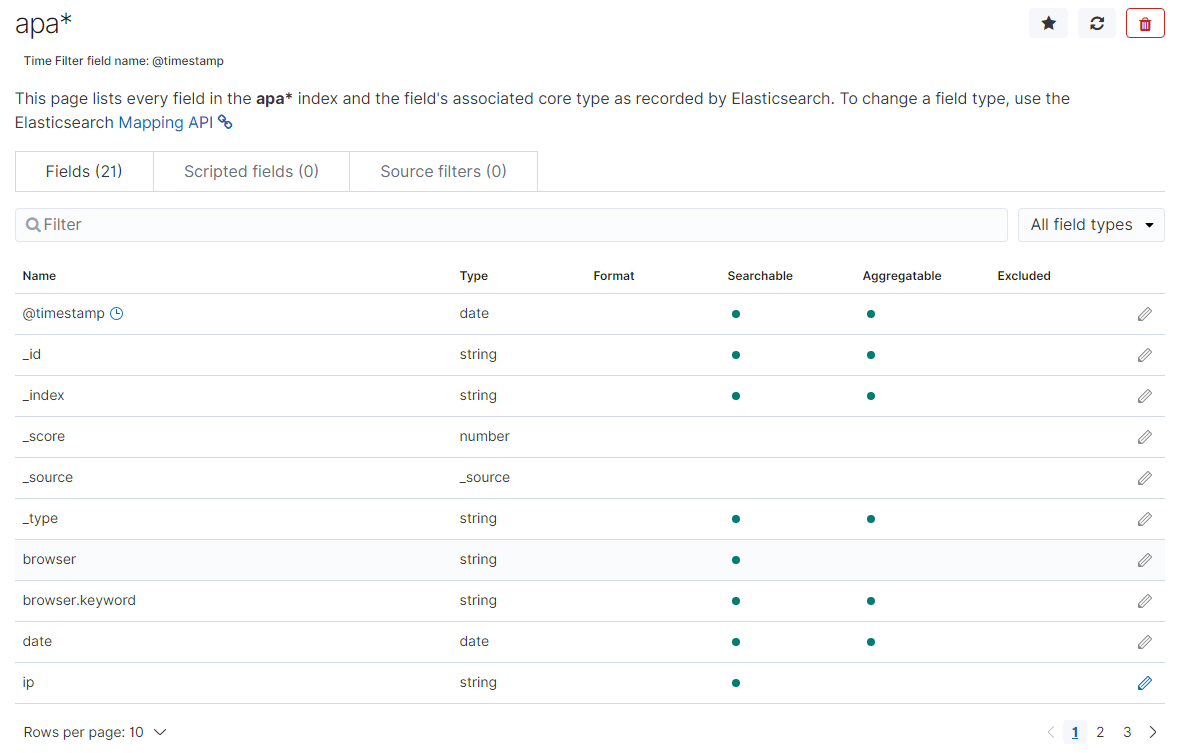
点击「Manage」按钮，还可以用来管理索引。

### 添加Elasticsearch数据源

#### Kibana索引模式

在开始使用Kibana之前，我们需要指定想要对哪些Elasticsearch索引进行处理、分析。在Kibana中，可以通过定义索引模式（Index Patterns）来对应匹配Elasticsearch索引。在第一次访问Kibana的时候，系统会提示我们定义一个索引模式。或者我们可以通过点击按钮，再点击Kibana下方的Index Patterns，来创建索引模式。参考下图：

1. 定义索引模式，用于匹配哪些Elasticsearch中的索引。点击「Next step」
2. 选择用于进行时间过滤的字段
3. 点击「Create Index Pattern」按钮，创建索引模式。创建索引模式成功后，可以看到显示了该索引模式对应的字段。里面描述了哪些可以用于搜索、哪些可以用来进行聚合计算等。



### 探索数据（Discovery）

通过Kibana中的Discovery组件，我们可以快速地进行数据的检索、查询。

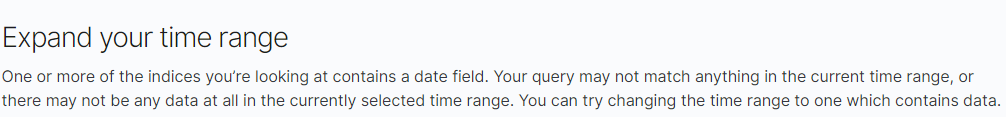
#### 使用探索数据功能

点击按钮可以打开Discovery页面。

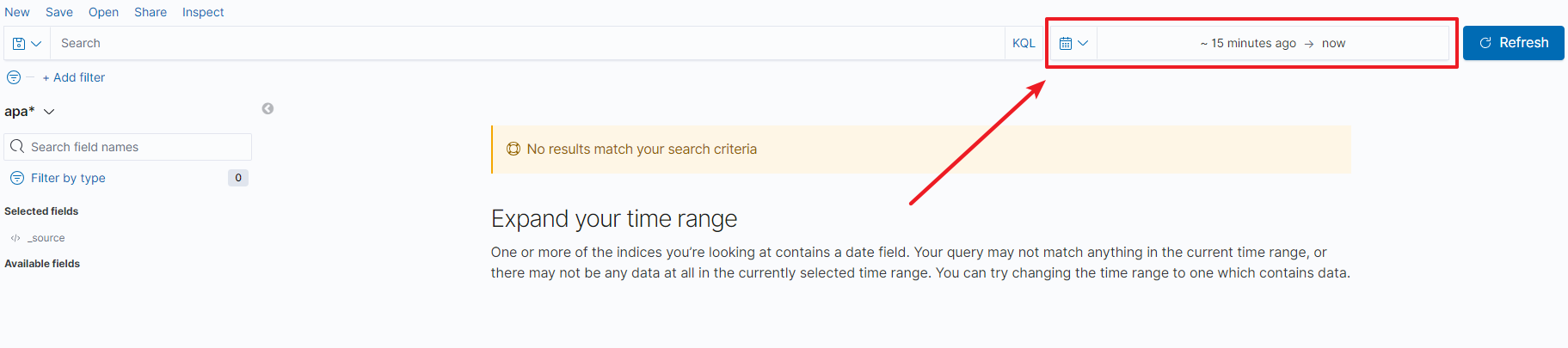
我们发现没有展示任何的数据。但我们之前已经把数据导入到Elasticsearch中了。



Kibana提示，让我们扩大我们的查询的时间范围。



默认Kibana是展示最近15分钟的数据。我们把时间范围调得更长一些，就可以看到数据了。



将时间范围选择为1年范围内的，我们就可以查看到Elasticsearch中的数据了。



#### 导入更多的Apache Web日志数据

1. 将资料中的 access.log 文件上传到Linux
2. 将access.log移动到/var/apache/log，并重命名为access.log.2

mv access.log /var/apache/log/access.log.2

1. 启动FileBeat

./filebeat -e -c filebeat-logstash.yml

1. 启动Logstash

bin/logstash -f config/filebeat-es.conf --config.reload.automatic

#### 基于时间过滤查询

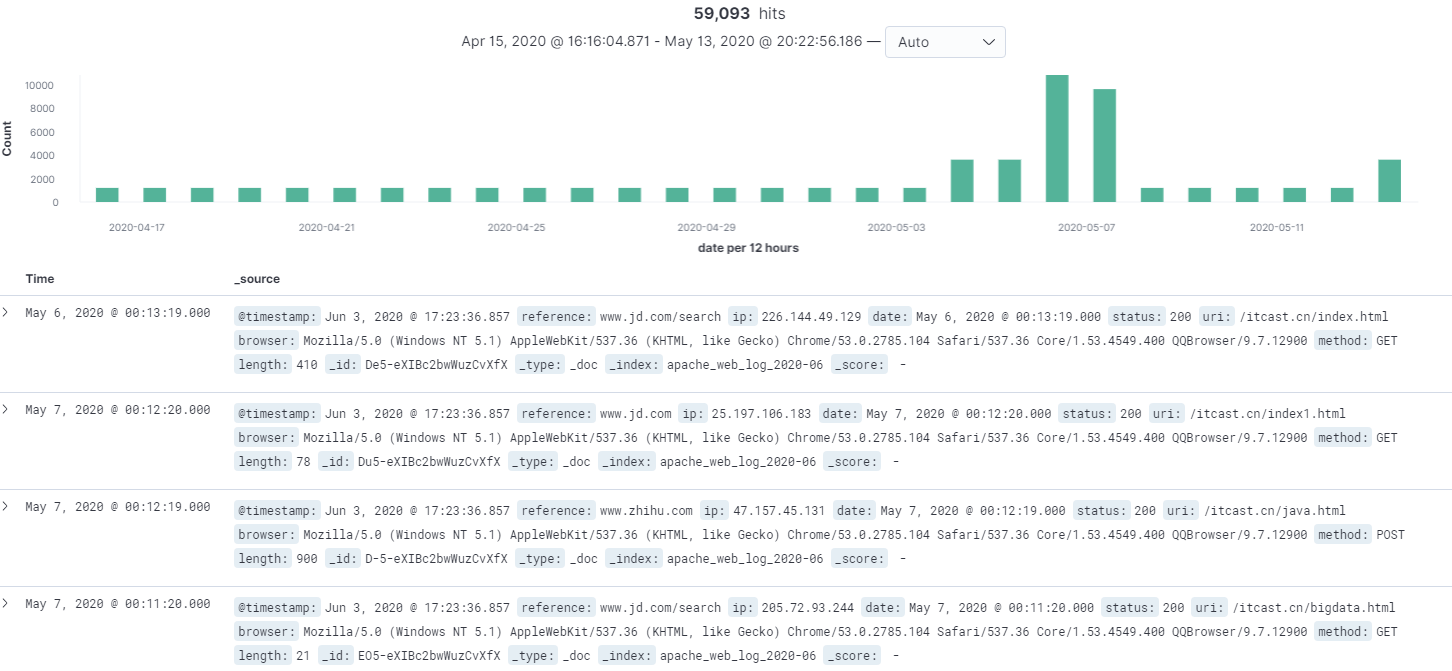
##### 选择时间范围

##### 指定查询某天的数据

查询2020年5月6日的所有日志数据。

##### 从直方图上选择日期更细粒度范围

如果要选择查看某一天的日志，上面这种方式会有一些麻烦，我们有更快更容易的方式。



#### 使用Kibana搜索数据

在Kibana的Discovery组件中，可以在查询栏中输入搜索条件。默认情况下，可以使用Kibana内置的标准查询语言，来进行快速查询。还有一种是遗留的基于Lucene的查询语法目前暂时可用，这种查询语法也可以使用基于JSON的Elasticsearch DSL也是可用的。当我们在Discovery搜索数据时，对应的直方图、文档列表都会随即更新。默认情况下，优先展示最新的文档，按照时间倒序排序的。

##### Kibana查询语言（KQL）

在7.0中，Kibana上线了新的查询语言。这种语言简洁、易用，有利于快速查询。

查询语法：

「字段:值」，如果值是字符串，可以用双引号括起来。

**查询包含zhihu的请求**

\*zhihu\*

**查询页面不存在的请求**

status : 404

**查询请求成功和不存在的请求**

status: (404 or 200)

**查询方式为POST请求，并请求成功的日志**

status: 200 and method: post

**查询方式为GET成功的请求，并且响应数据大于512的日志**

status: 200 and method: get and length > 512

**查询请求成功的且URL为「/node1」开头的日志**

uri: "\/node1\/\*"

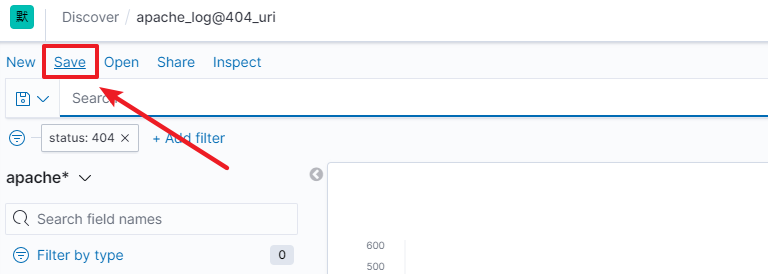
注意：因为/为特殊字符，需要使用反斜杠进行转义

##### 过滤字段

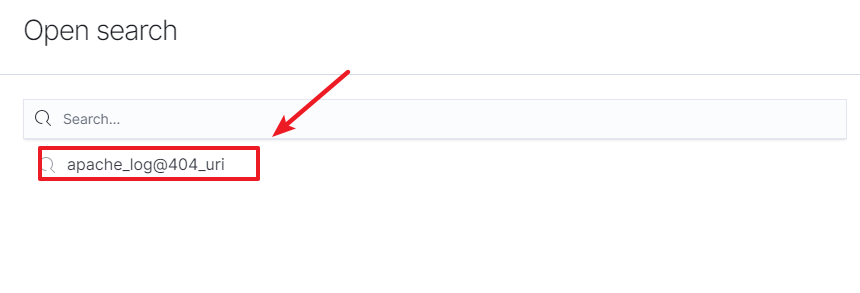
Kibana的Discovery组件提供各种各样的筛选器，这样可以筛选出来我们关注的数据上。例如：我们只想查询404的请求URI。

**指定过滤出来404以及请求的URI、从哪儿跳转来的日志**

**将查询保存下来，方便下次直接查看**



下次直接点击Open就可以直接打开之前保存的日志了

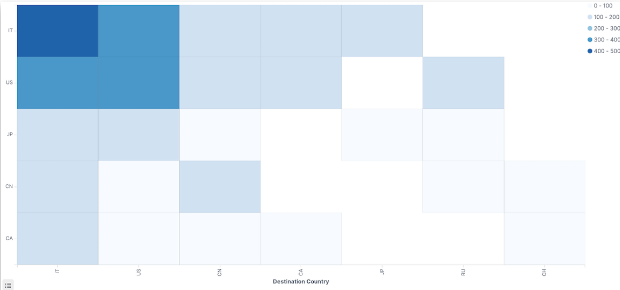


### 数据可视化（Visualize）

Kibana中的Visualize可以基于Elasticsearch中的索引进行数据可视化，然后将这些可视化图表添加到仪表盘中。

#### 数据可视化的类型

* Lens
  + 通过简单地拖拽数据字段，快速构建基本的可视化
* 常用的可视化对象
  + 线形图（Line）、面积图（Area）、条形图（Bar）：可以用这些带X/Y坐标的图形来进行不同分类的比较
  + 饼图（Pie）：可以用饼图来展示占比
  + 数据表（Data Table）：以数据表格的形式展示
  + 指标（Metrics）：以数字的方式展示
  + 目标和进度：显示带有进度指标的数字
  + 标签云/文字云（Tag Cloud）：以文字云方式展示标签，文字的大小与其重要性相关
* Timelion
  + 从多个时间序列数据集来展示数据
* 地图
  + 展示地理位置数据
* 热图
  + 在矩阵的单元格展示数据



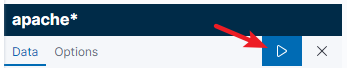
* 仪表盘工具
  + Markdown部件：显示一些MD格式的说明
  + 控件：在仪表盘中添加一些可以用来交互的组件
* Vega

#### 以饼图展示404与200的占比

效果图：

操作步骤：

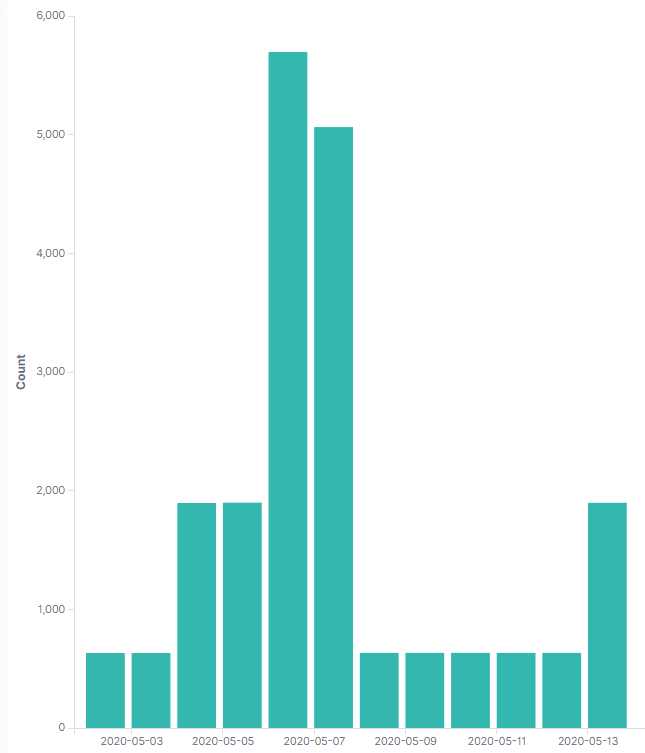
1. 创建可视化
2. 选择要进行可视化图形类型，此处我们选择Pie（饼图类型）
3. 选择数据源
4. 添加分桶、分片（还记得吗？我们在Elasticsearch进行分组聚合都是以分桶方式进行的，可以把它理解为分组）
5. 配置分桶以及指标计算方式
6. 点击蓝色播放按钮执行。



1. 保存图形（取名为：apache\_log@404\_200）

#### 以条形图方式展示2020年5月每日请求数

效果如下：



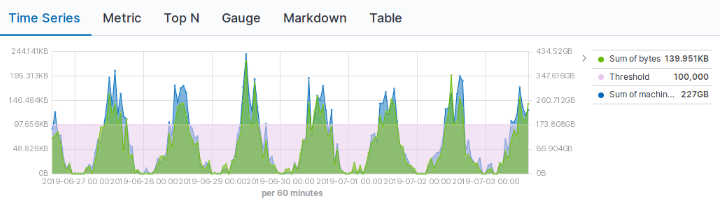
开发步骤：

我们还可以修改图形的样式，例如：以曲线、面积图的方式展示。

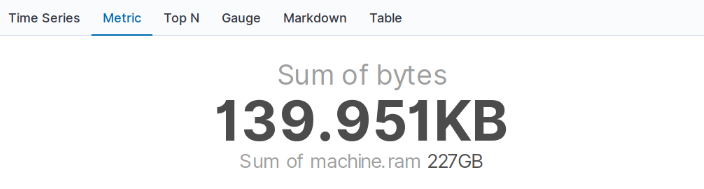
#### 以TSVB可视化不同访问来源的数据

TSVB是一个基于时间序列的数据可视化工具，它可以使用Elasticsearch聚合的所有功能。使用TSVB，我们可以轻松地完成任意聚合方式来展示复杂的数据。它可以让我们快速制作效果的图表：

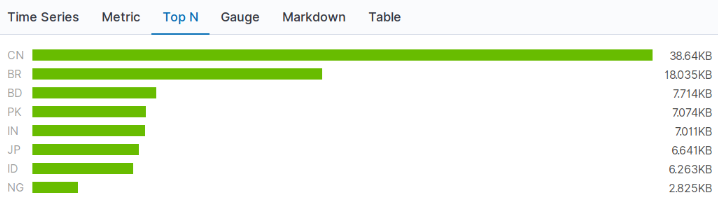
1. 基于时间序列的图形展示



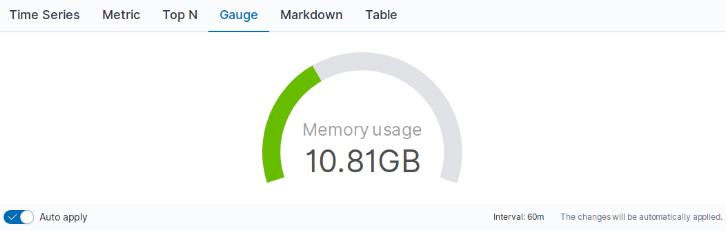
1. 展示指标数据



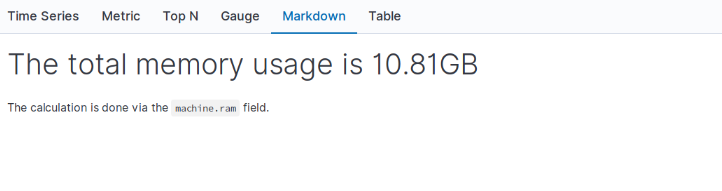
1. TopN

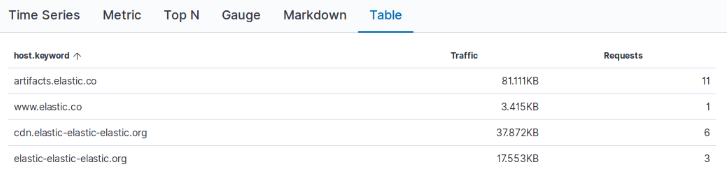


1. 类似油量表的展示



1. Markdown自定义数据展示



1. 以表格方式展示数据

操作步骤：

1. 创建TSVB可视化对象
2. 配置Time Series数据源分组条件
3. 配置Metric



1. TopN



#### 制作用户选择请求方式、响应字节大小控制组件

##### 控制组件

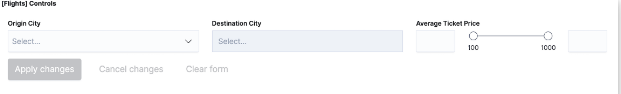
在Kibana中，我们可以使用控件来控制图表的展示。例如：提供一个下列列表，供查看图表的用户只展示比较关注的数据。我们可以添加两个类型的控制组件：

1. 选项列表

* 根据一个或多个指定选项来筛选内容。例如：我们先筛选某个城市的数据，就可以通过选项列表来选择该城市

1. 范围选择滑块

* 筛选出来指定范围的数据。例如：我们筛选某个价格区间的商品等。

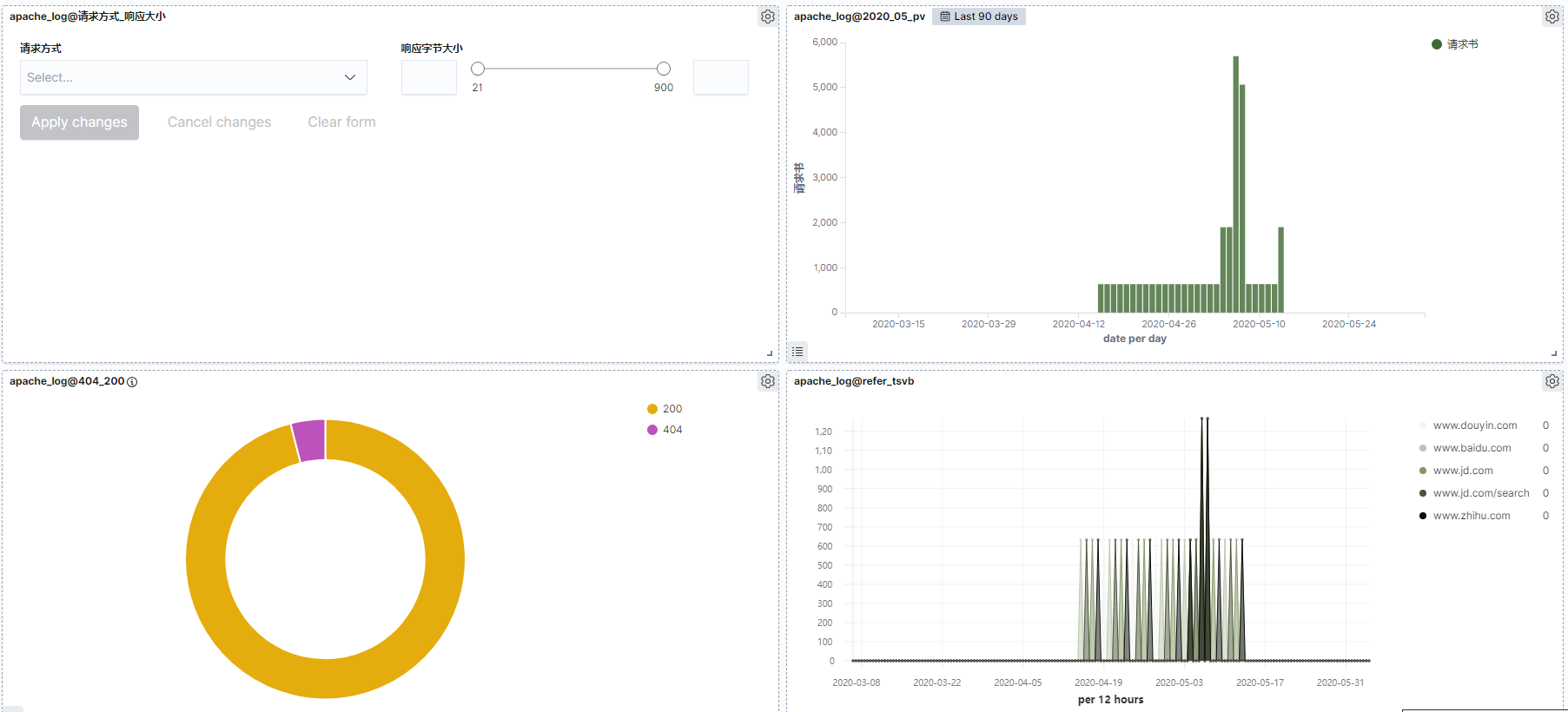


##### Kibana开发

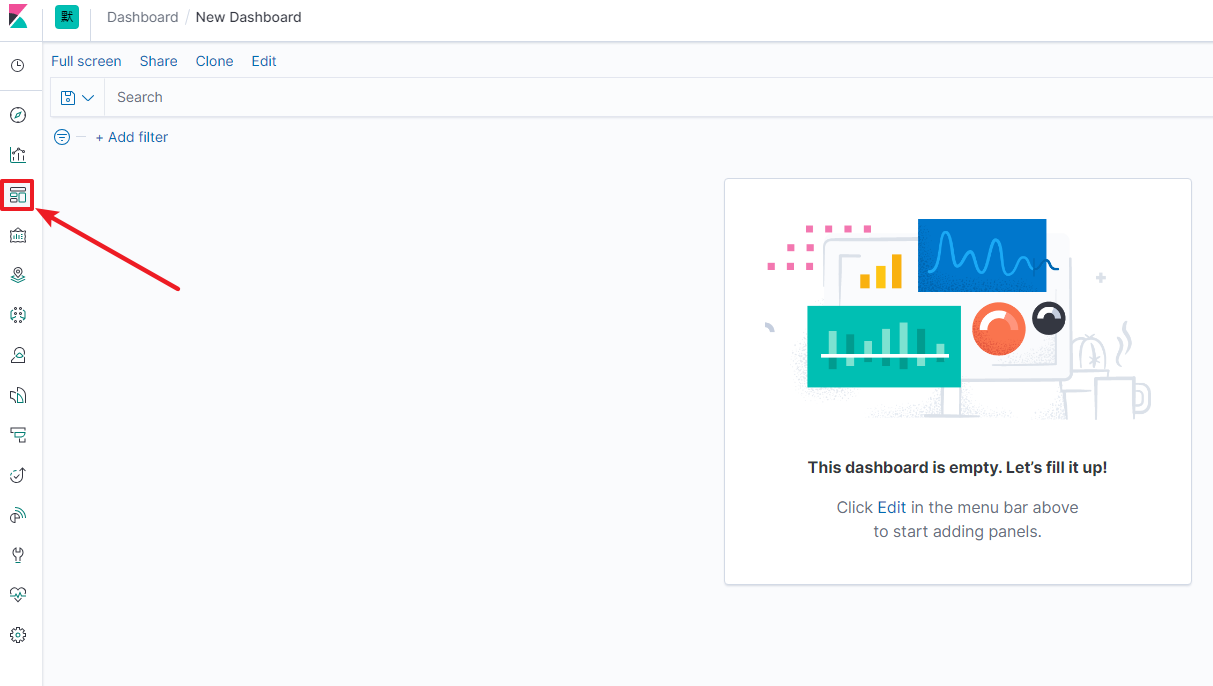


### 制作Dashboard

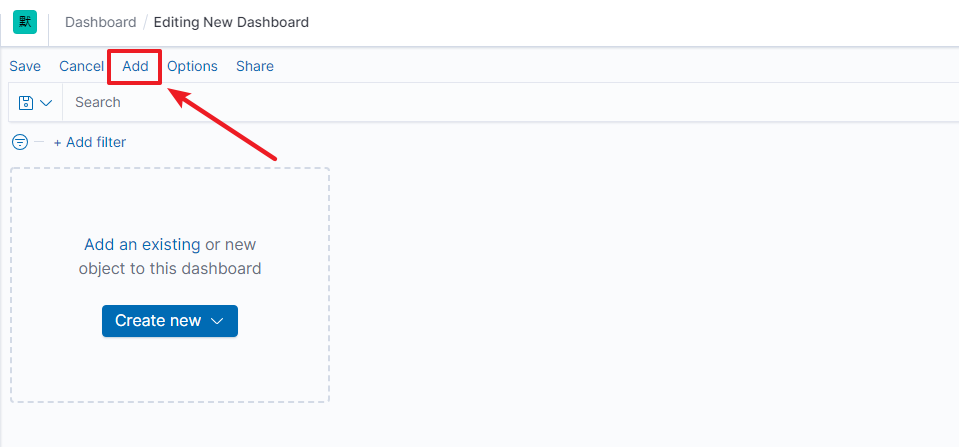
接下来，我们把前面的几个图形放到一个看板中。这样，我们就可以在一个看板中，浏览各类数据了。



1. 点击第三个组件图标，并创建一个新的Dashboard。



1. 点击Edit编辑Dashboard。



1. 依次添加我们之前制作好的图表。

