咕泡学院 JavaVIP 高级课程教案

ELK 分布式日志应用实战

关于本文档

主题	咕泡学院 Java VIP 高级课程教案--ELK 分布式日志应用实战
主讲	Tom 老师
适用对象	咕泡学院 Java 高级 VIP 学员及 VIP 授课老师
软件版本	ElasticSearch 6.5.1，Logstash 6.5.1，Kibana 6.5.1
IDE 版本	IntelliJ IDEA 2017.1.4

一、进入 ElasticSearch 的世界

1.1、ElasticSearch 版本选择及分布式环境搭建

1.1.1 版本问题

因为 ElasticSearch 是 ELK 组合中的一部分，都是 Elastic 产品中的组成部分，在 ElasticSearch 2.x 以前，ELK 中的各个中间件的版本不一致，如：ElasticSearch2.3.4，而 Kibana 对应的版本是 4.5.3。2016 年秋季，为了方便各中间件方便配合使用，ElasticSearch 直接从 2.x 升级到了 5.x，保持了和各个中间件版本一致。

因此，ElasticSearch 的版本历史是：1.x -> 2.x -> 5.x。

ElasticSearch 5.5.x 相对以前的 2.x 版本，是基于 Lucene 6 来构建的，它增加了 36%的查询速度，增加了 71%的索引速度，并且减少了 66%的硬盘空间占用，还较少了 85%的内存使用，同时还新增 IP 字段，以支持 IP4 和 IP6，在各方面超越了以往的历史版本。

1.1.2 下载和安装

打开网址：https://www.elastic.co/downloads/past-releases，选择 6.5.1 版本，获取下载链接。大家在学习过程中，尽量保持版本一致。

下载之后，直接解压即可。运行 bin 目录下的 elasticsearch.bat 文件（linux 下需要在 root 用户下运行 elasticsearch 文件），出现如下界面即启动成功。

```
[2018-11-06T13:55:14,113][INFO ][o.e.n.Node               ] [master] version[5.5.2], pid[2028], build[b2f0c09/2017-08-14T
12:33:14.154Z], OS[Windows 10/10.0/amd64], JVM[Oracle Corporation/Java HotSpot(TM) 64-Bit Server VM/1.8.0_131/25.131-b11]

[2018-11-06T13:55:14,113][INFO ][o.e.n.Node               ] [master] JVM arguments [-Xms2g, -Xmx2g, -XX:+UseConcMarkSweep
GC, -XX:CMSInitiatingOccupancyFraction=75, -XX:+UseCMSInitiatingOccupancyOnly, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.he
adless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -Djdk.io.permissionsUseCanonicalPath=true, -Dio.netty.noUnsafe=true
, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dlog4j.shutdownHookEnabled=false, -D
log4j2.disable.jmx=true, -Dlog4j.skipJansi=true, -XX:+HeapDumpOnOutOfMemoryError, -Delasticsearch, -Des.path.home=D:\elas
ticsearch-5.5.2-master]
[2018-11-06T13:55:15,467][INFO ][o.e.p.PluginsService     ] [master] loaded module [aggs-matrix-stats]
[2018-11-06T13:55:15,467][INFO ][o.e.p.PluginsService     ] [master] loaded module [ingest-common]
[2018-11-06T13:55:15,468][INFO ][o.e.p.PluginsService     ] [master] loaded module [lang-expression]
[2018-11-06T13:55:15,468][INFO ][o.e.p.PluginsService     ] [master] loaded module [lang-groovy]
[2018-11-06T13:55:15,468][INFO ][o.e.p.PluginsService     ] [master] loaded module [lang-mustache]
[2018-11-06T13:55:15,468][INFO ][o.e.p.PluginsService     ] [master] loaded module [lang-painless]
[2018-11-06T13:55:15,468][INFO ][o.e.p.PluginsService     ] [master] loaded module [parent-join]
[2018-11-06T13:55:15,468][INFO ][o.e.p.PluginsService     ] [master] loaded module [percolator]
[2018-11-06T13:55:15,468][INFO ][o.e.p.PluginsService     ] [master] loaded module [reindex]
[2018-11-06T13:55:15,469][INFO ][o.e.p.PluginsService     ] [master] loaded module [transport-netty3]
[2018-11-06T13:55:15,469][INFO ][o.e.p.PluginsService     ] [master] loaded module [transport-netty4]
[2018-11-06T13:55:15,469][INFO ][o.e.p.PluginsService     ] [master] no plugins loaded
[2018-11-06T13:55:17,954][INFO ][o.e.d.DiscoveryModule    ] [master] using discovery type [zen]
[2018-11-06T13:55:18,603][INFO ][o.e.n.Node               ] [master] initialized
[2018-11-06T13:55:18,603][INFO ][o.e.n.Node               ] [master] starting ...
[2018-11-06T13:55:19,104][INFO ][o.e.t.TransportService   ] [master] publish_address {127.0.0.1:9300}, bound_addresses {1
27.0.0.1:9300}
[2018-11-06T13:55:22,190][INFO ][o.e.c.s.ClusterService   ] [master] new_master {master}{n1y8ITICRJqFv3LJTu013g}{uVRdH6B7
R7yxJUCQUiGSyw}{127.0.0.1}{127.0.0.1:9300}, reason: zen-disco-elected-as-master ([0] nodes joined)
[2018-11-06T13:55:22,263][INFO ][o.e.h.n.Netty4HttpServerTransport] [master] publish_address {127.0.0.1:9200}, bound_addr
esses {127.0.0.1:9200}
[2018-11-06T13:55:22,264][INFO ][o.e.n.Node               ] [master] started
[2018-11-06T13:55:22,669][INFO ][o.e.g.GatewayService     ] [master] recovered [2] indices into cluster_state
```

如果在 CentOS 中，要做以下设置：

1）系统参数修改脚本

注意 要以 root 身份执行下面的脚本，执行后要重新登录普通账户启动 ES

```
#!/bin/bash
echo "* soft    nofile  65536" >> /etc/security/limits.conf
echo "* hard    nofile  65536" >> /etc/security/limits.conf
echo "* soft memlock unlimited" >> /etc/security/limits.conf
echo "* hard memlock unlimited" >> /etc/security/limits.conf
echo "vm.max_map_count = 262144" >> /etc/sysctl.conf
sysctl -p
 ulimit -l unlimited
```

2）JVM 参数调整

ES 5.5.x 的 JVM 参数配置方式和以往 2.x 等版本不同，它独立出了一个 jvm.options 的文件在 config 目录下，我们可以通过修改 jvm.options 里的参数来指定 ES 启动需要的 JVM 环境，比如 ES 默认是 2G 堆内存，当内存不足时，我们可以进行提升，如下：

```
-Xms4g
-Xmx4g
```

除此之外，我们还可以针对性的做各种调整，这里就不再赘述 JVM 的各参数用法。

启动报错问题

在 CentOS 下运行 ES 5.5.x 版本时，可能会遇到如下报错信息：

```
bound or publishing to a non-loopback or non-link-local address, enforcing bootstrap checks
ERROR: [1] bootstrap checks failed...
```

遇到这些错误信息不要慌，这是因为 CentOS 内核不支持 SecComp，而 ES 5.5.x 默认是要执行检测命令的，所以这里我们把这个环境检测禁掉，就可以正常运行了，参数如下：

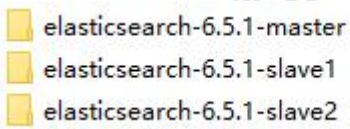
```
bootstrap.system_call_filter: false
```

在配置文件 elasticsearch.yml 追加即可。

到此，单实例安装就完成了。

1.1.3 分布式安装

1）将刚下载的 elasticsearch-5.5.2 复制两份，将三个目录分别做如下命名：



2）修改主节点的配置，打开 elasticsearch-6.5.1-master\config 下的 elasticsearch.yml 文件，在底部追加如下内容：

```
cluster.name: tom-test      #集群名称
node.name: master           #节点 ID，保证唯一
node.master: true           #标记是否为主节点

network.host: 127.0.0.1     #对外公开的 IP 地址，如果自动识别配置为 0.0.0.0
```

2）配置 slave-1 节点，打开 elasticsearch-6.5.1-slave-1\config 下的 elasticsearch.yml 文件，在底部追加如下内容：

```
cluster.name: tom-test      #集群名称三个节点保持一致
node.name: slave-1          #从节点 ID，保证唯一

network.host: 127.0.0.1     #对外公开的 IP 地址，如果自动识别配置为 0.0.0.0
http.port: 8200             #默认端口为 9200，因为我的环境是在同一台机器，因此，指定服务端口号

discovery.zen.ping.unicast.hosts: ["127.0.0.1"]    #集群的 IP 组，配置主节点 IP 即可
```

3）配置 slave-2 节点，打开 elasticsearch-5.5.1-slave-2\config 下的 elasticsearch.yml 文件，在底部追加如下内容：

```
cluster.name: tom-test      #集群名称三个节点保持一致
node.name: slave-2          #从节点 ID，保证唯一

network.host: 127.0.0.1     #对外公开的 IP 地址，如果自动识别配置为 0.0.0.0
http.port: 8000             #默认端口为 9200，因为我的环境是在同一台机器，因此，指定服务端口号

discovery.zen.ping.unicast.hosts: ["127.0.0.1"]    #集群的 IP 组，配置主节点 IP 即可
```

4）分别启动三个节点。

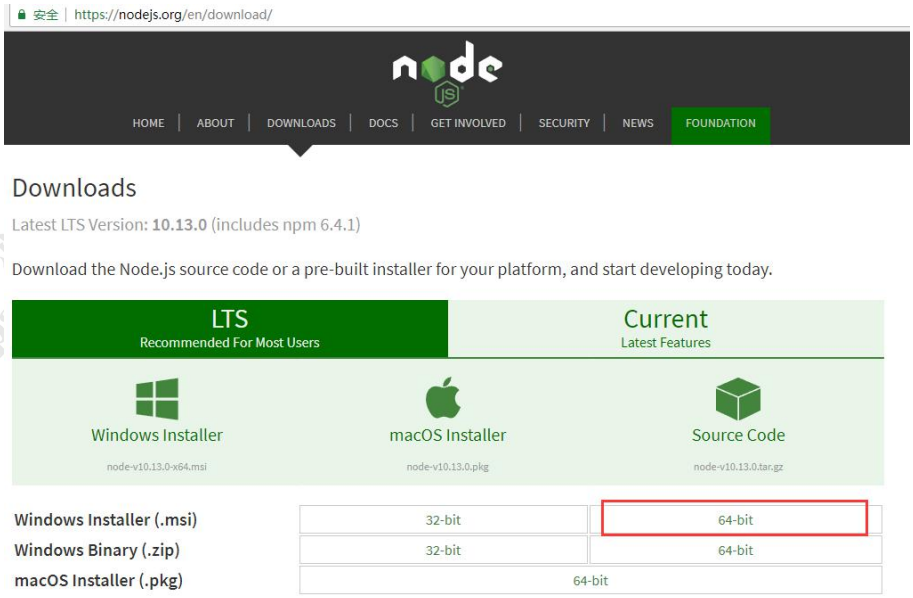
```
[2018-11-06T13:55:15,468][INFO ][o.e.p.PluginsService     ] [master] loaded module [reindex]
[2018-11-06T13:55:15,469][INFO ][o.e.p.PluginsService     ] [master] loaded module [transport-netty3]
[2018-11-06T13:55:15,469][INFO ][o.e.p.PluginsService     ] [master] loaded module [transport-netty4]
[2018-11-06T13:55:15,469][INFO ][o.e.p.PluginsService     ] [master] no plugins loaded
[2018-11-06T13:55:17,954][INFO ][o.e.d.DiscoveryModule    ] [master] using discovery type [zen]
[2018-11-06T13:55:18,603][INFO ][o.e.n.Node               ] [master] initialized
[2018-11-06T13:55:18,603][INFO ][o.e.n.Node               ] [master] starting ...
[2018-11-06T13:55:19,104][INFO ][o.e.t.TransportService   ] [master] publish_address {127.0.0.1:9300}, bound_addresses {1
27.0.0.1:9300}
[2018-11-06T13:55:22,190][INFO ][o.e.c.s.ClusterService   ] [master] new_master {master}{n1y8ITICRJqFv3LJTu013g}{uVRdH6B7
R7yxJUCQUiGSyw}{127.0.0.1}{127.0.0.1:9300}, reason: zen-disco-elected-as-master ([0] nodes joined)
[2018-11-06T13:55:22,263][INFO ][o.e.h.n.Netty4HttpServerTransport] [master] publish_address {127.0.0.1:9200}, bound_addr
esses {127.0.0.1:9200}
[2018-11-06T13:55:22,264][INFO ][o.e.n.Node               ] [master] started
```

```
[2018-11-06T14:59:58,873][INFO ][o.e.p.PluginsService     ] [slave-1] loaded module [transport-netty4]
[2018-11-06T14:59:58,874][INFO ][o.e.p.PluginsService     ] [slave-1] no plugins loaded
[2018-11-06T15:00:01,012][INFO ][o.e.d.DiscoveryModule    ] [slave-1] using discovery type [zen]
[2018-11-06T15:00:01,491][INFO ][o.e.n.Node               ] [slave-1] initialized
[2018-11-06T15:00:01,491][INFO ][o.e.n.Node               ] [slave-1] starting ...
[2018-11-06T15:00:01,821][INFO ][o.e.t.TransportService   ] [slave-1] publish_address {127.0.0.1:9301}, bound_addresses
{127.0.0.1:9301}
[2018-11-06T15:00:05,117][INFO ][o.e.c.s.ClusterService   ] [slave-1] detected_master {master}{n1y8ITICRJqFv3LJTu013g}{u
VRdH6B7R7yxJUCQUiGSyw}{127.0.0.1}{127.0.0.1:9300}, added {{master}{n1y8ITICRJqFv3LJTu013g}{uVRdH6B7R7yxJUCQUiGSyw}{127.0
.0.1}{127.0.0.1:9300},}, reason: zen-disco-receive(from master [master {master}{n1y8ITICRJqFv3LJTu013g}{uVRdH6B7R7yxJUCQ
UiGSyw}{127.0.0.1}{127.0.0.1:9300} committed version [6]])
[2018-11-06T15:00:05,430][INFO ][o.e.h.n.Netty4HttpServerTransport] [slave-1] publish_address {127.0.0.1:8200}, bound_ad
dresses {127.0.0.1:8200}
[2018-11-06T15:00:05,430][INFO ][o.e.n.Node               ] [slave-1] started
```

```
[2018-11-06T15:00:06,959][INFO ][o.e.p.PluginsService     ] [slave-2] no plugins loaded
[2018-11-06T15:00:16,132][INFO ][o.e.d.DiscoveryModule    ] [slave-2] using discovery type [zen]
[2018-11-06T15:00:16,614][INFO ][o.e.n.Node               ] [slave-2] initialized
[2018-11-06T15:00:16,615][INFO ][o.e.n.Node               ] [slave-2] starting ...
[2018-11-06T15:00:16,926][INFO ][o.e.t.TransportService   ] [slave-2] publish_address {127.0.0.1:9302}, bound_addresses
{127.0.0.1:9302}
[2018-11-06T15:00:20,174][INFO ][o.e.c.s.ClusterService   ] [slave-2] detected_master {master}{n1y8ITICRJqFv3LJTu013g}{u
VRdH6B7R7yxJUCQUiGSyw}{127.0.0.1}{127.0.0.1:9300}, added {{master}{n1y8ITICRJqFv3LJTu013g}{uVRdH6B7R7yxJUCQUiGSyw}{127.0
.0.1}{127.0.0.1:9300}, {slave-1}{qP8n_PkvSoC12fswgCe1Kg}{JvPrGgIfSBmEOGZaJsP9Bw}{127.0.0.1}{127.0.0.1:9301},}, reason: ze
n-disco-receive(from master [master {master}{n1y8ITICRJqFv3LJTu013g}{uVRdH6B7R7yxJUCQUiGSyw}{127.0.0.1}{127.0.0.1:9300}
committed version [15]])
[2018-11-06T15:00:20,818][INFO ][o.e.h.n.Netty4HttpServerTransport] [slave-2] publish_address {127.0.0.1:8000}, bound_ad
dresses {127.0.0.1:8000}
[2018-11-06T15:00:20,818][INFO ][o.e.n.Node               ] [slave-2] started
```

1.1.4 可视化插件安装

1）下载 NodeJS 环境，打开官网 https://nodejs.org/en/download/

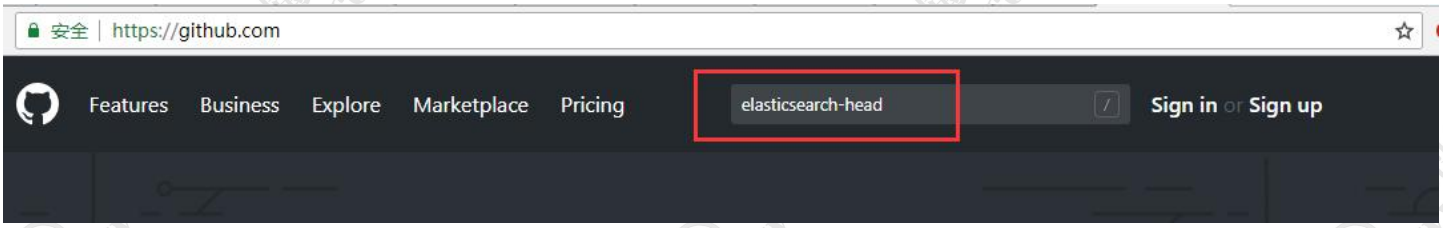


2）安装 NodeJS

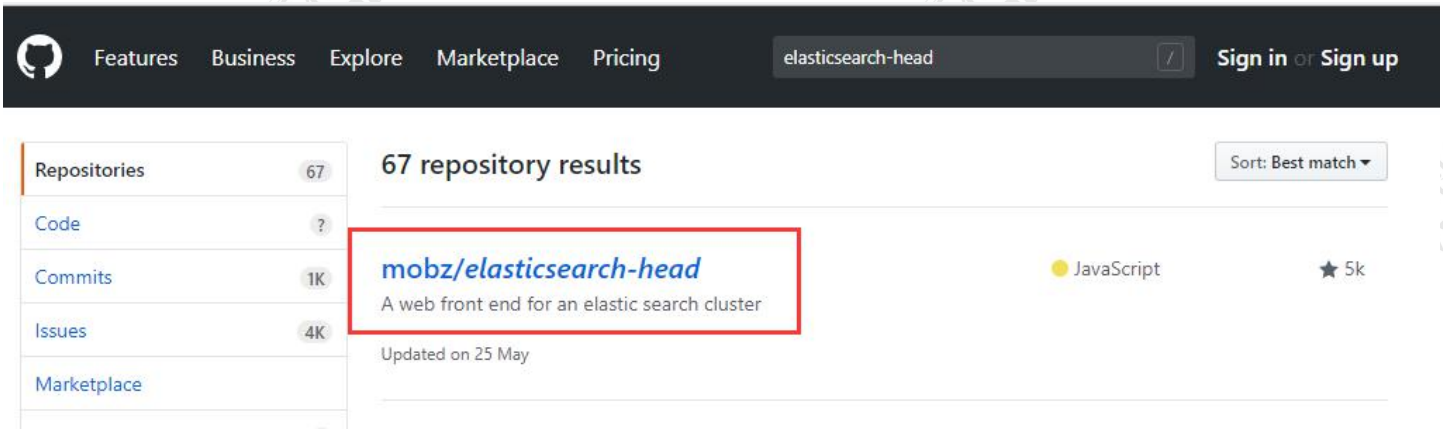
运行 node-v8.11.3-x64.msi 之后，配置系统环境变量。然后，检查输入 node -v 检查 node 是否安装成功。

```
C:\Users\Tom>node -v
v8.11.3
```

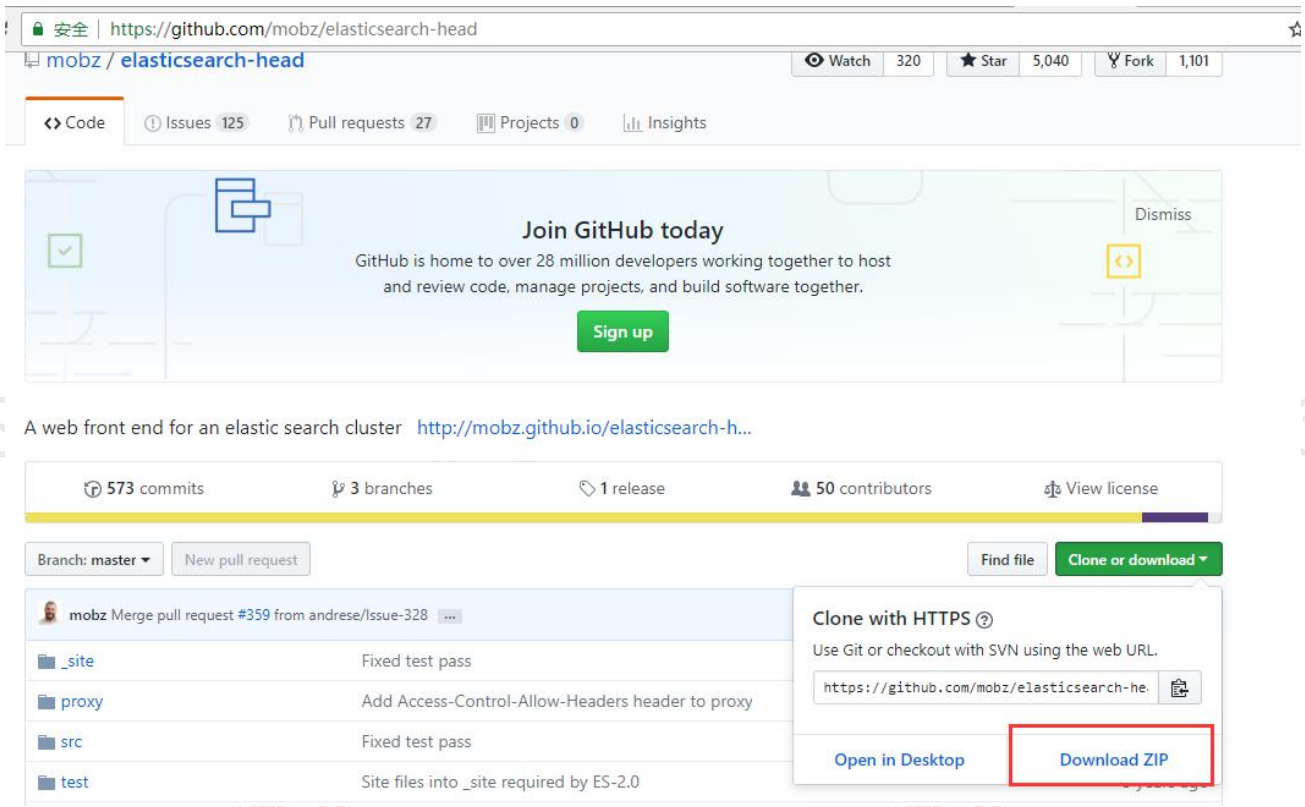
3）下载 ElasticSearch，打开 https://github.com，搜索 elasticsearch-head 关键字。



4）搜索结果，选择 mobz/elasticsearch-head



5）下载 elasticsearch-head-master.zip 包。



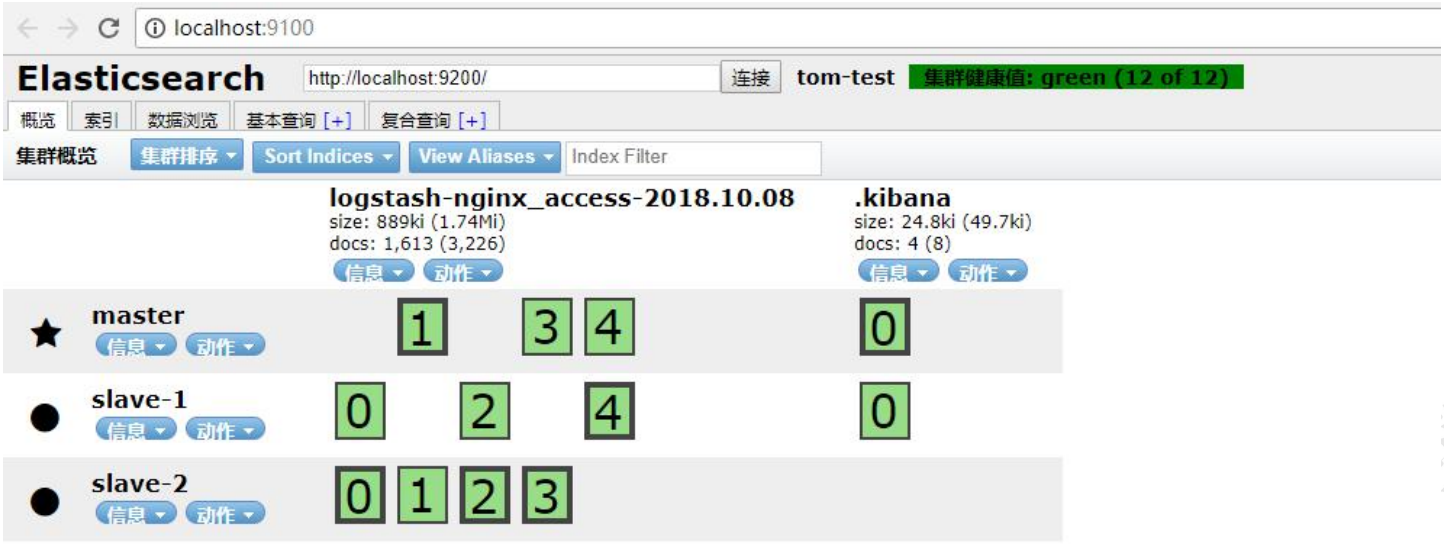
6）修改 master 节点的跨域配置，在 elasticsearch.yml 中追加以下内容。

```
http.cors.enabled: true     #允许跨域
http.cors.allow-origin: "*"
```

7）启动 head 插件

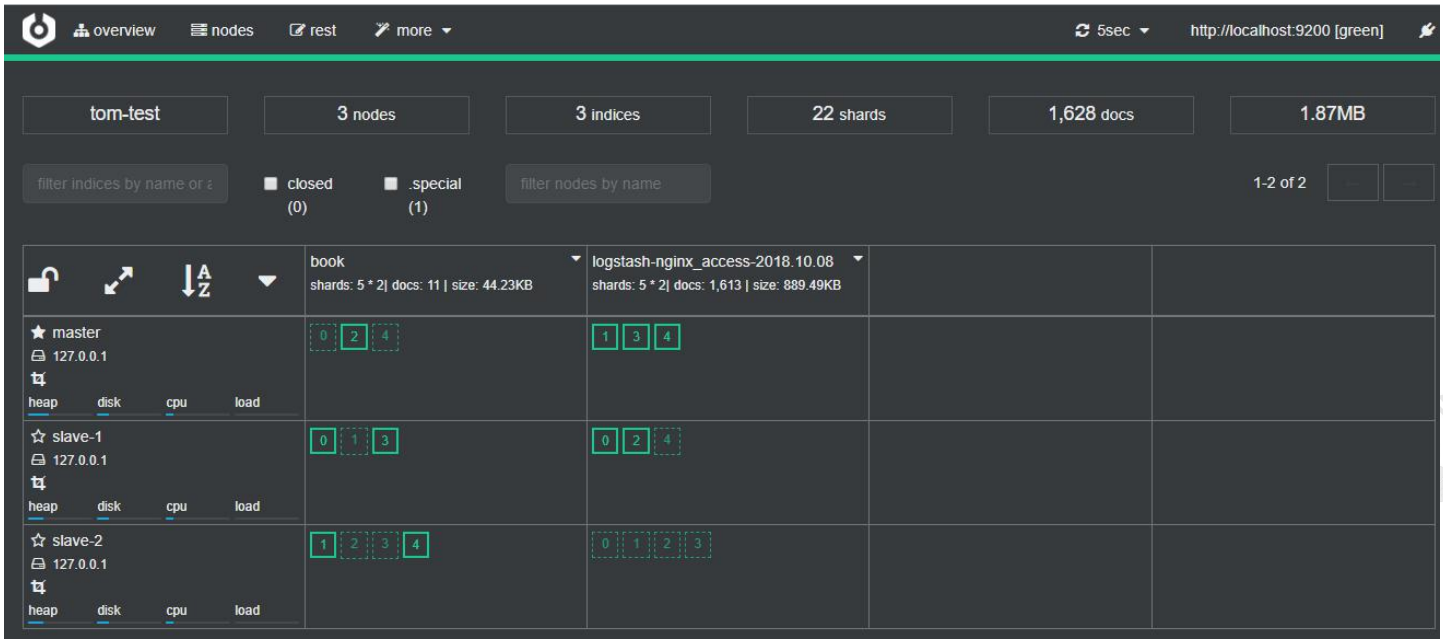
```
D:\>cd elasticsearch-head-master

D:\elasticsearch-head-master>npm run start

> elasticsearch-head@0.0.0 start D:\elasticsearch-head-master
> grunt server

(node:13768) ExperimentalWarning: The http2 module is an experimental API.
Running "connect:server" (connect) task
Waiting forever...
Started connect web server on http://localhost:9100
```

8）输入 http://localhost:9100/，可以看到所有节点的信息。



1.1.5 Cerebro 的安装

下载地址：https://github.com/lmenezes/cerebro/releases



1.2、ElasticSearch 基本原理及学习方法论

1.2.1 Lucene 工作原理

处理文本的最高效做法就是：正则匹配。

1.2.2 ElasticSearch 中的基本概念

索引： 含有相同属性的文档集合。
类型：索引可以定义一个或多个类型，文档必须属于一个类型。
文档：文档是可以被索引的基本数据单元。
分片：每个索引都有多个分片，每个分片是一个 Lucene 索引。
备份：拷贝一份分片就完成了分片的备份。

1.2.3 ElasticSearch API 命名风格

API 基本格式： http://<ip>:<port>/<索引>/<类型>/<文档 ID>

常用的 HTTP 动词：GET/PUT/POST/DELETE

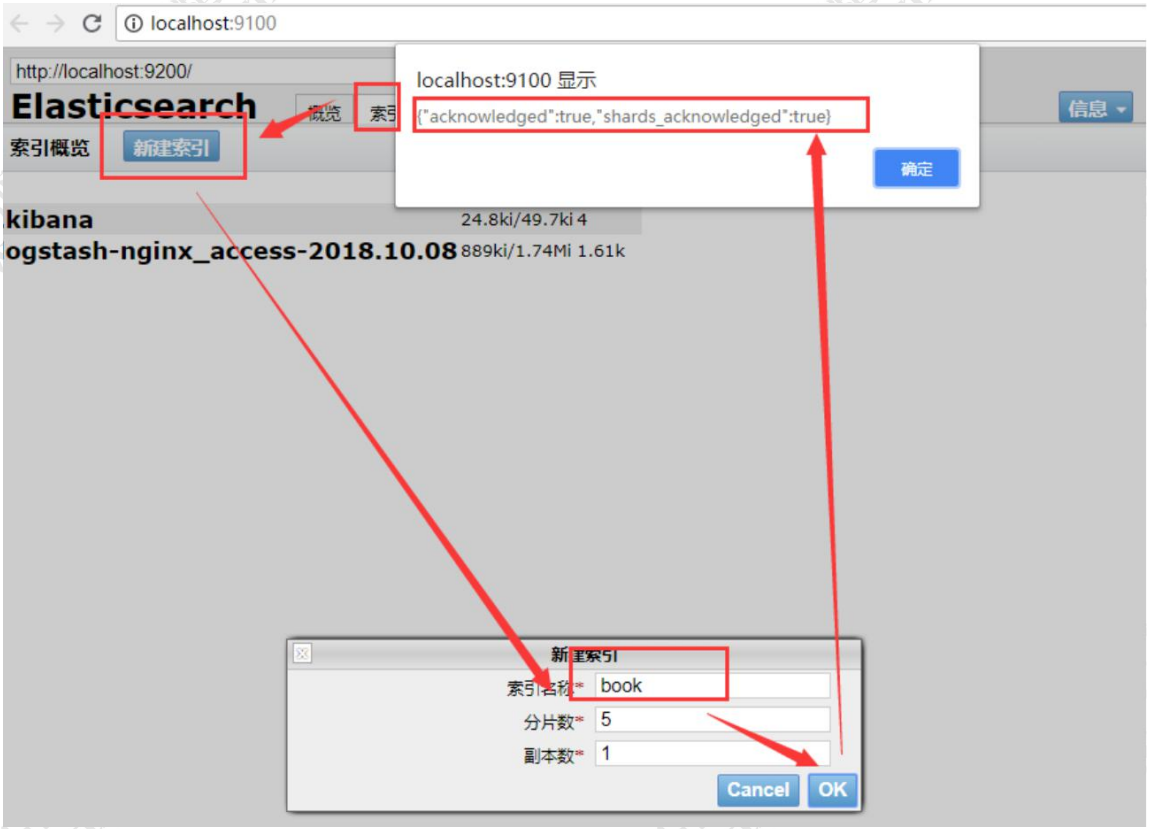
1.2.4 关系型数据库和 ElasticSearch 操作姿势对比

JDBC 操作	ElasticSearch Client 操作
1、加载驱动类(JDBC 驱动)	----
2、建立连接（Connection）	1、建立连接（TransportClient）
3、创建语句集（Statement）	2、条件构造（SearchRequestBuilder）
4、执行语句集 execute()	3、执行语句 execute()
5、获取结果集（ResultSet）	4、获取结果（SearchResponse）
6、关闭结果、语句、连接	5、关闭以上操作

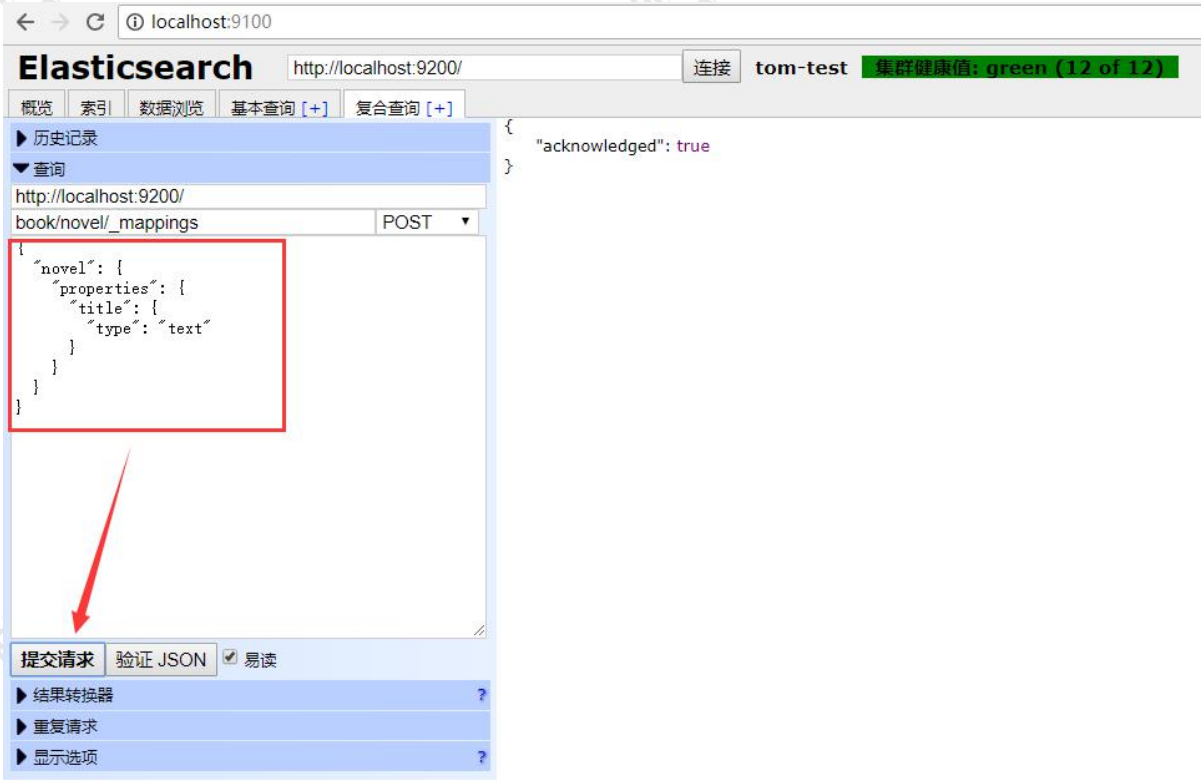
1.3、ElasticSearch 基本操作

1.3.1 创建索引

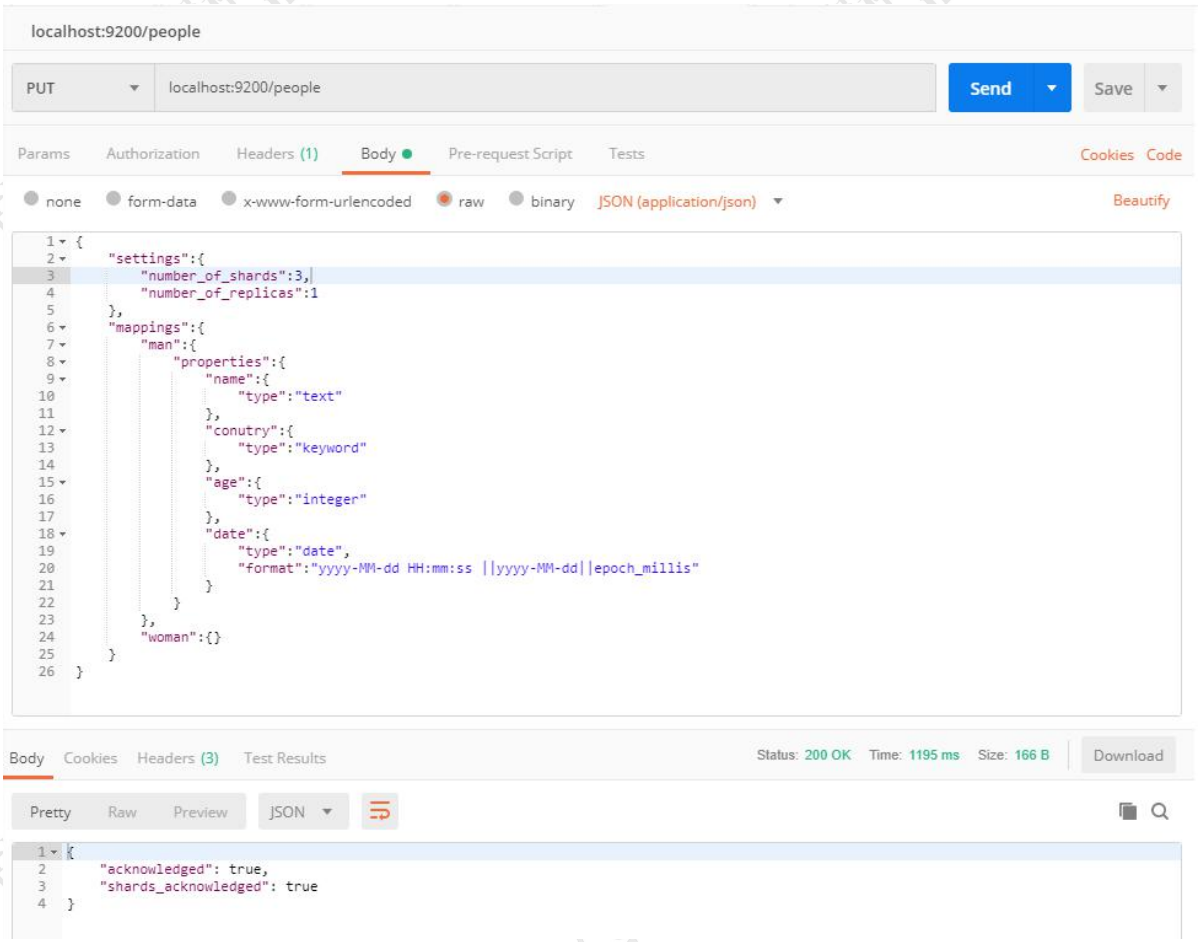
方式一：创建非结构化的索引（如下图所示）。



方式二：创建结构化的索引，输入 book/novel/_mappings（如下图所示），



方式三：可以在 Postman 中选择 PUT 方法，输入 localhost:9200/people，然后在 raw 中编辑一下 json 信息（如下图）：



输入的 json 内容如下：

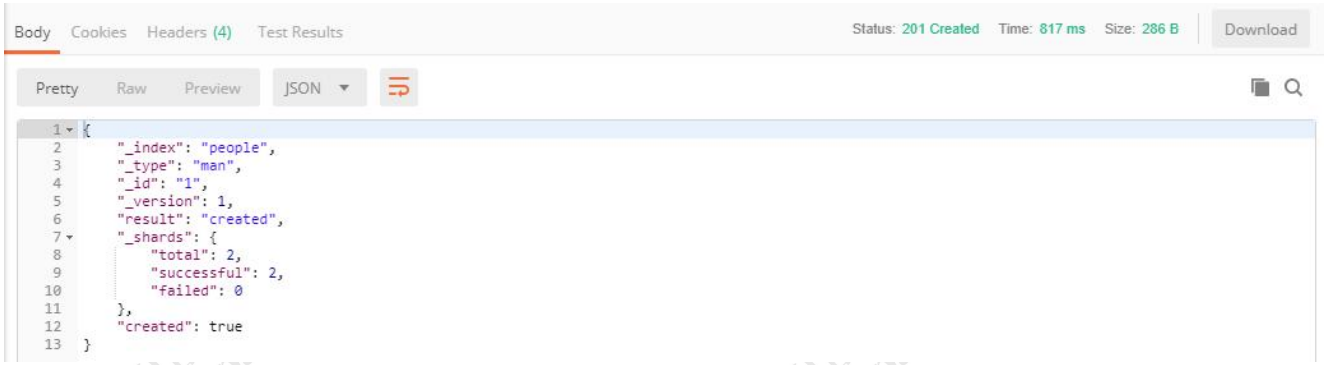
```
{
    "settings":{
        "number_of_shards":3,
        "number_of_replicas":1
    },
    "mappings":{
        "man":{
            "properties":{
                "name":{ "type":"text" },
                "conutry":{ "type":"keyword" },
                "age":{ "type":"integer" },
                "date":{
                    "type":"date",
                    "format":"yyyy-MM-dd HH:mm:ss ||yyyy-MM-dd||epoch_millis"
                }
            }
        },
        "woman":{}
    }
}
```

1.3.2 插入数据

方式一：指定文档 ID 插入，在 Postman 中使用 PUT 方法，输入 localhost:9200/people/man/1,在 raw 区域输入:

```
{
    "name":"Tom",
    "country":"China",
    "age":18,
   "date":"2000-10-11"
}
```

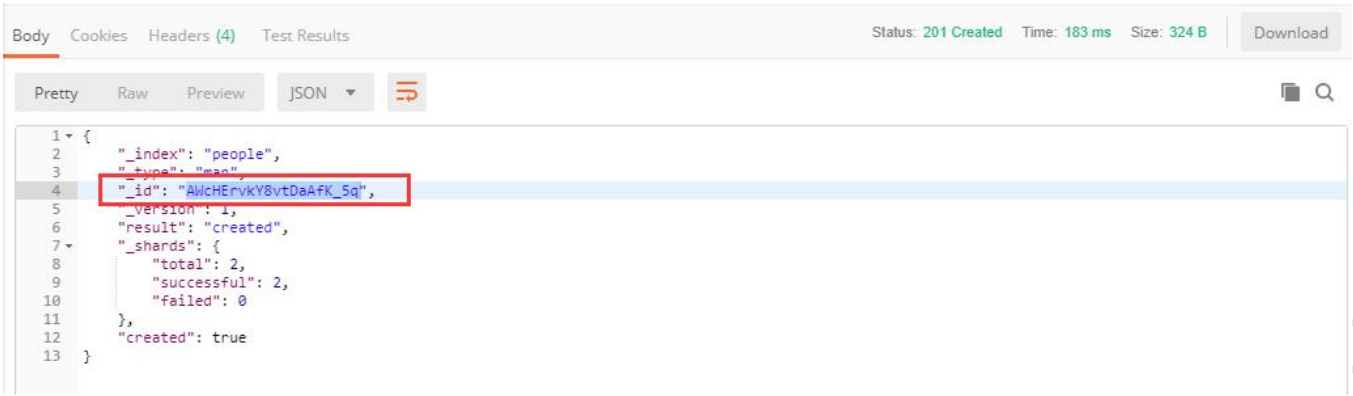
执行结果如下：



方式二：自动生成文档 ID 插入，在 Postman 中使用 POST 方法，输入 localhost:9200/people/man，在 raw 区域输入：

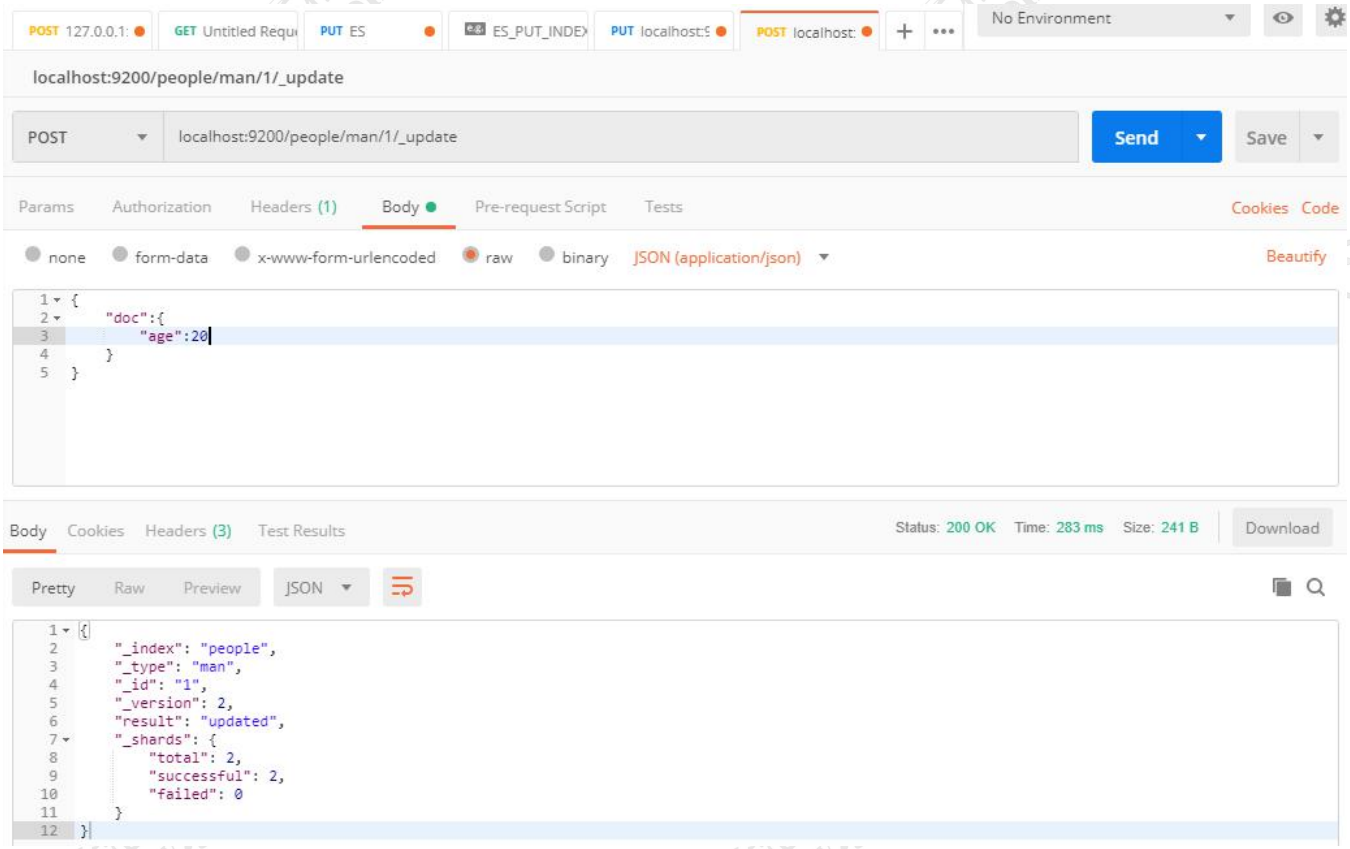
```
{
    "name":"Tom 老师",
    "country":"China",
    "age":19,
   "date":"1999-10-11"
}
```

执行结果如下：



1.3.3 修改文档

方式一：直接修改文档，打开 Postmain，选择 POST 方法，输入 localhost:9200/people/man/1/_update，运行结果如下：



方式二：通过脚本修改文档，在 raw 区输入以下内容：

所有年龄增加一岁。

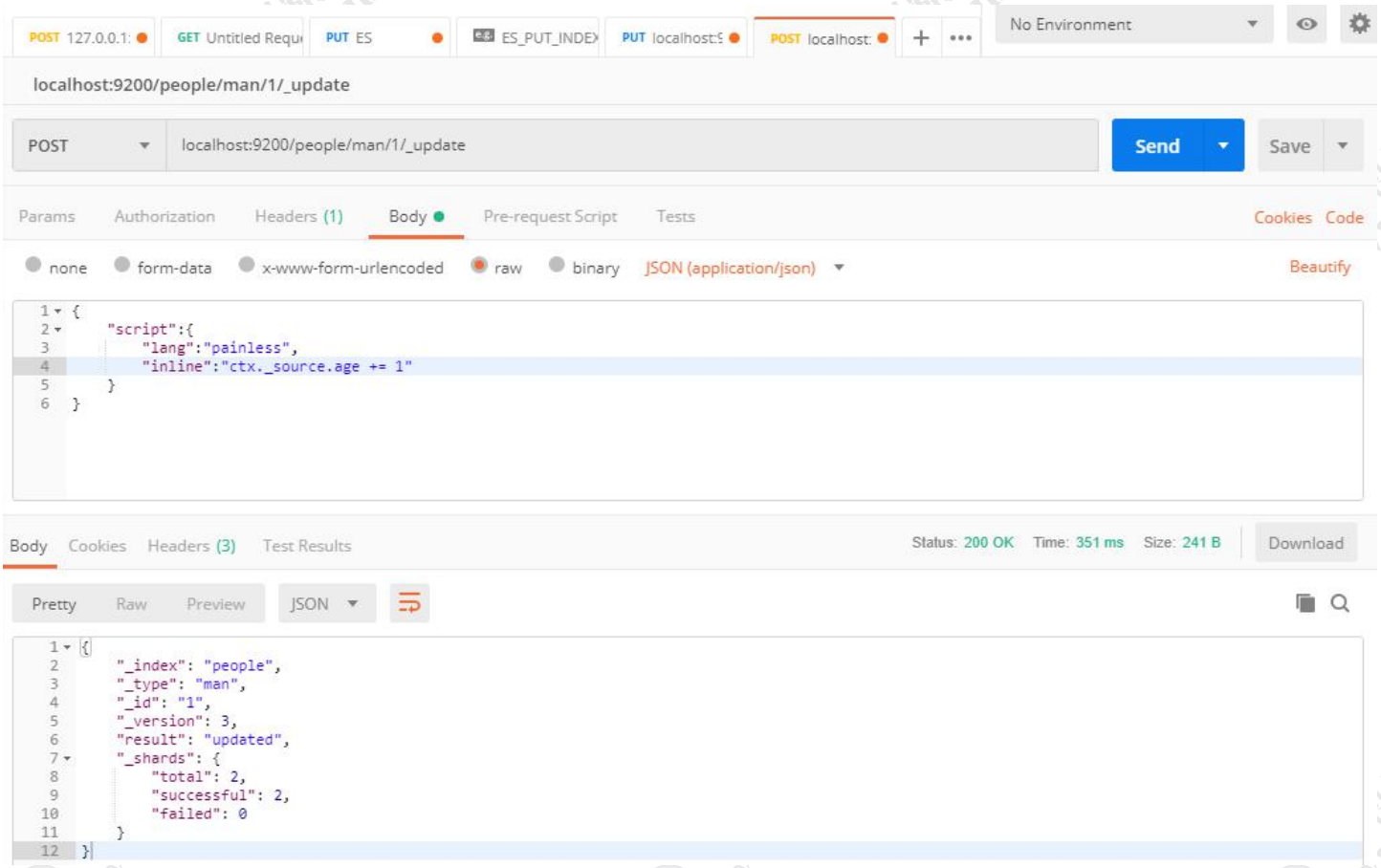
```
{
    "script":{
        "lang":"painless",
        "inline":"ctx._source.age += 1",
    }
}
```

或者输入：

修改年龄为 30 岁。

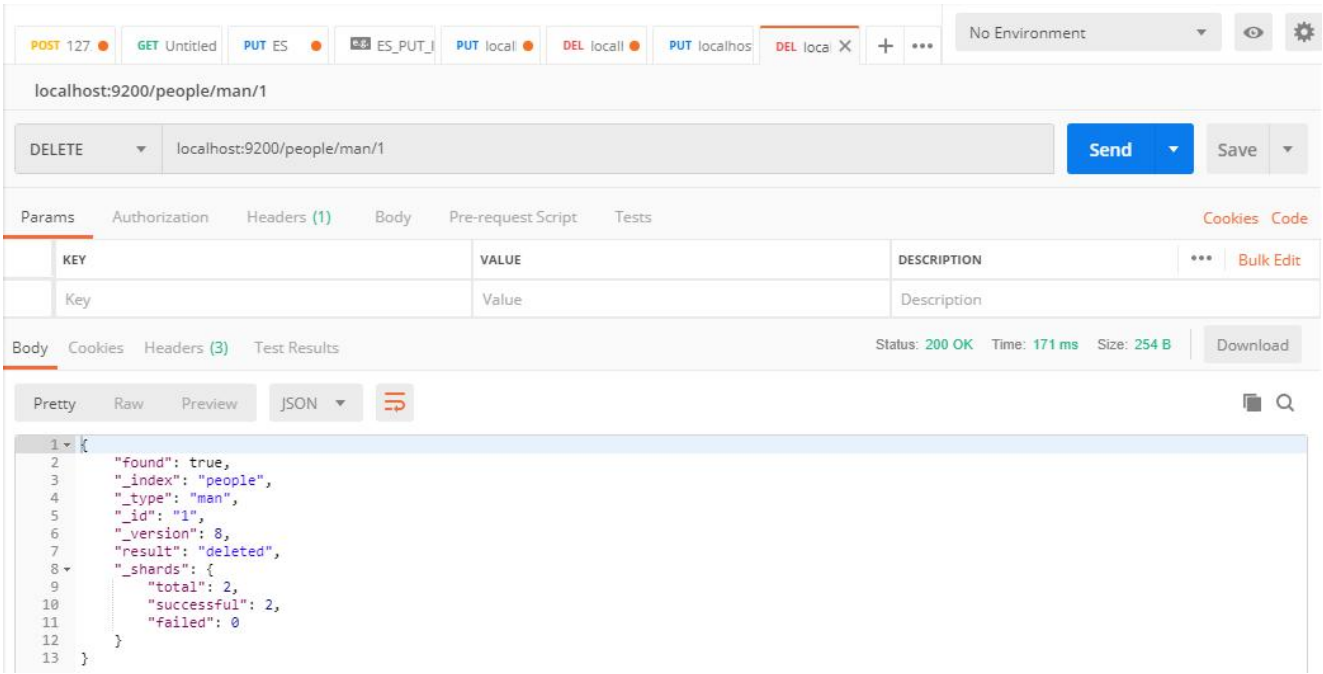
```
{
    "script":{
        "lang":"painless",
        "inline":"ctx._source.age = params.age",
        "params":{
            "age":30
        }
    }
}
```

都可以得到以下结果：

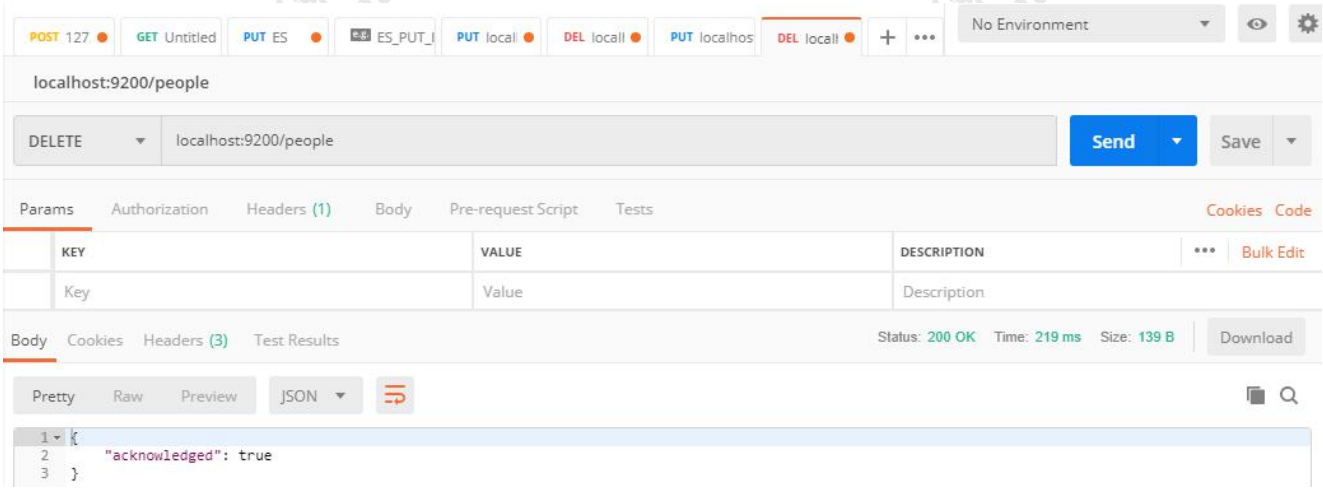


1.3.4 删除文档

1）删除文档：打开 Postman，选择 DELETE 方法，输入 localhost:9200/people/man/1，执行结果如下：

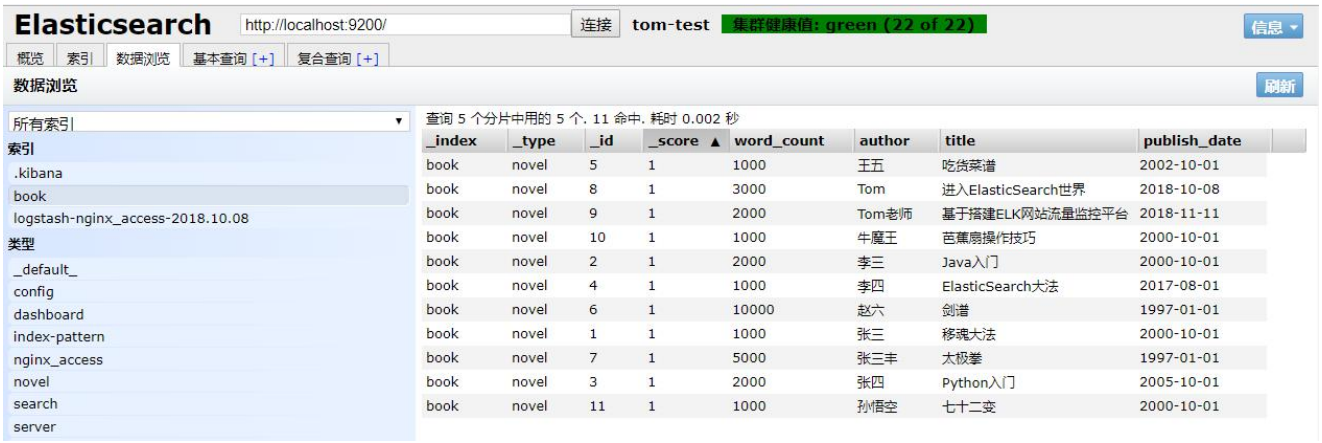


3）删除索引，打开 Postman 输入 localhost:9200/people，执行结果如下：

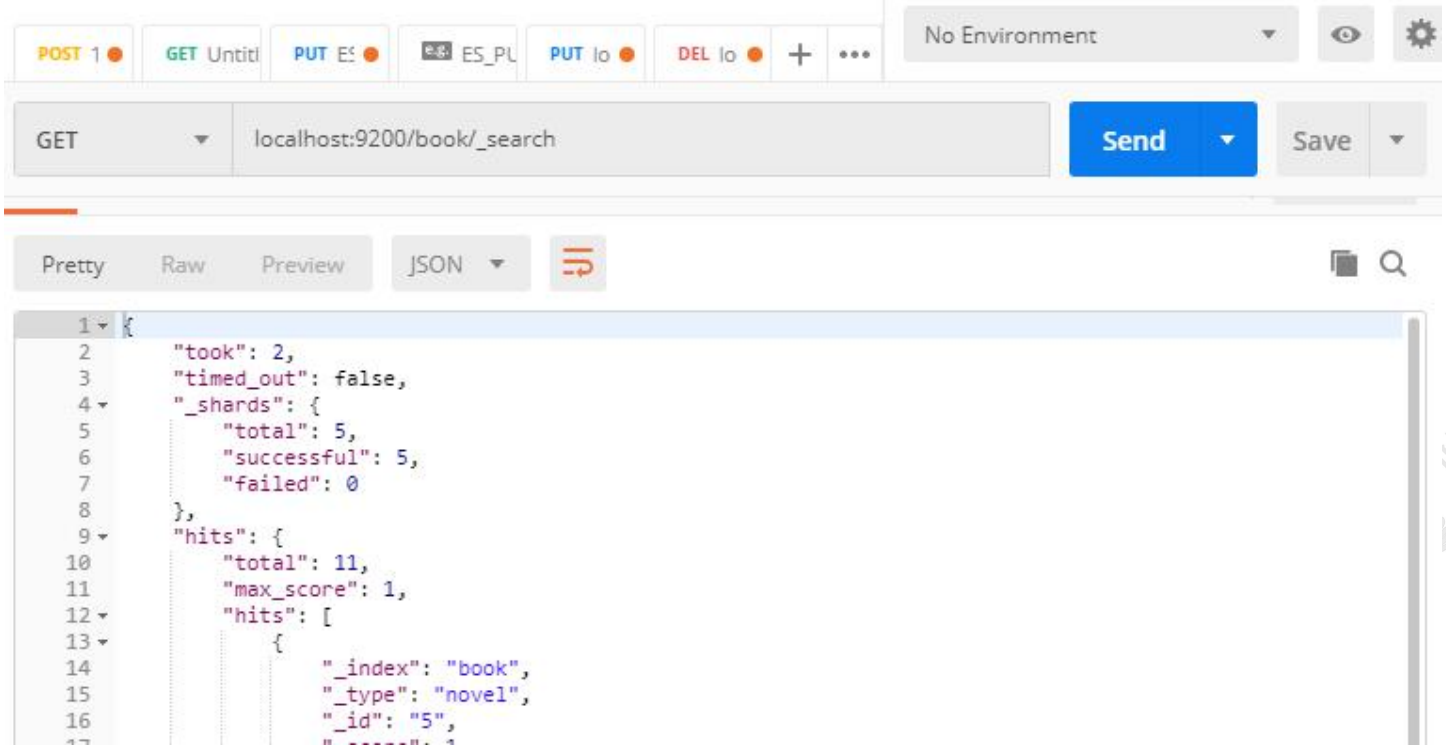


1.3.5 查询语法

在查询操作之前，我已经初始化了一些数据，如下图：



1) 全表查询:在 Postman 中选择 GET 方法，输入 localhost:9200/book/_search 得到以下结果：



2) 条件查询:在 Postman 中选择 GET 方法，输入 localhost:9200/book/_search，然后在 raw 区域中编辑如下内容：查询书籍中包含 ElasticSearch 关键字，且按发版日期降序排序。

```
{
    "query":{
        "match":{
            "title":"ElasticSearch"
        }
    },
    "sort":[
        {"publish_date":{"order":"desc"}}
        ]
}
```

执行得到以下结果：

3) 聚合查询:在 Postman 中选择 GET 方法，输入 localhost:9200/book/_search，然后在 raw 区域中编辑如下内容：根据书籍字数和发版日期进行分组

```
{
    "aggs":{
        "group_by_word_count":{
            "terms":{
                "field":"word_count"
            }
        },
        "group_by_publish_date":{
            "terms":{
                "field":"publish_date"
            }
        }
    }
}
```

运行结果：

```
                "key_as_string": "2018-11-11 00:00:00",
                "doc_count": 1
            }
        ]
    },
    "group_by_word_count": {
        "doc_count_error_upper_bound": 0,
        "sum_other_doc_count": 0,
        "buckets": [
            {
                "key": 1000,
                "doc_count": 5
            },
            {
                "key": 2000,
                "doc_count": 3
            },
            {
                "key": 3000,
                "doc_count": 1
            },
            {
                "key": 5000,
                "doc_count": 1
            },
            {
                "key": 10000,
                "doc_count": 1
            }
        ]
    }
  }
}
```


4) 聚合统计:在 Postman 中选择 GET 方法，输入 localhost:9200/book/_search，然后在 raw 区域中编辑如下内容：根据书籍的字数进行聚合统计。

```
{
    "aggs":{
        "grades_word_count":{
            "stats":{ "field":"word_count" }
        }
    }
}
```

运行结果如下：


```
    },
    "aggregations": {
        "grades_word_count": {
            "count": 11,
            "min": 1000,
            "max": 10000,
            "avg": 2636.3636363636365,
            "sum": 29000
        }
    }
}
```


1.4、ElasticSearch 高级查询

1.4.1 query 条件：

方式一：模糊匹配，在 Postman 中选择 GET 方法，输入 localhost:9200/book/_search，然后在 raw 区域中编辑如下内容：

查询标题中包含”ElasticSearch”和”入门”关键字的书籍

```
{
    "query":{
        "match":{
            "title":"ElasticSearch 入门"
        }
    }
}
```

方式二：习语匹配

查询标题中包含”ElasticSearch”的书籍

```
{
    "query":{
        "match_phrase":{
            "title":"ElasticSearch"
        }
    }
}
```

方式三：多字段匹配

查询作者和标题中都包含”Tom”的书籍

```
{
    "query":{
        "multi_match":{
            "query":"Tom",
            "fields":["author","title"]
        }
    }
}
```

方式三：Query 语法查询

查询标题和作者中同时包含 ElasticSearch 和大法，或者包含 Python 的书籍。

```
{
    "query":{
        "query_string":{
            "query":"(ElasticSearch AND 大法) OR Python",
            "fields":["title","author"]
        }
    }
}
```

方式四：结构化数据查询

1）查询字数在 1000 到 2000 之间的数据

```
{
    "query":{
        "range":{
            "word_count":{
                "gt":1000,
                "lte":2000
            }
        }
    }
}
```

2）查询 2018-01-01 至今发版的所有书籍

```
{
    "query":{
        "range":{
            "publish_date":{
                "gt":"2018-01-01",
                "lte":"now"
            }
        }
    }
}
```

1.4.2 filter 条件：

```
{
    "query":{
        "bool":{
            "filter":{
                "term":{
                    "word_count":1000
                }
            }
        }
    }
}
```

1.4.3 复合查询

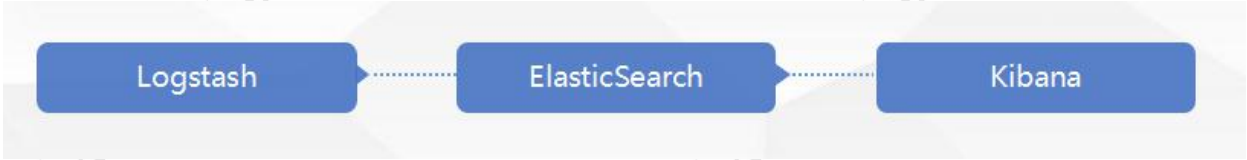
```
{
    "query":{
        "bool":{
            "must":[
                {
                    "match":{
                        "title":"ElasticSearch"
                    }
                },
                {
                 "match":{
                     "author":"Tom"
                 }

                }
            ],
            "filter":{
                "term":{
                    "word_count":3000
                }
            }
        }
    }
}
```

1.5、ElasticSearch 与 Spring API 集成

看直播代码演示

二、基于 ELK 搭建网站流量可视化监控平台

通过上图我们可以看到，ELK 是由三个 Elastic 的产品组合而成，分别是 ElasticSearch、Logstash 和 Kibana。三者之间是的部署关系如下图所示：



Logstash 就好比是挖矿工，将原料采集回来存放到 ElasticSearch 这个仓库中，Kibana 再将存放在 ElasticSearch 中的原料进行加工包装成产品，输出到 web 界面。基本工作原理如下图所示：

2.1、Logstash 原理分析及环境搭建

官网 https://www.elastic.co/cn/downloads/past-releases 下载 Logstash 6.5.1,解压即可。

启动方式一：命令行输入：

```
bin\logstash -e 'input { stdin {} } output { stdout {} }'
```

运行得到如下结果，说明 logstash 成果启动。

```
D:\logstash-5.5.2>bin\logstash -e 'input { stdin {} } output { stdout {} }'
ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.
Sending Logstash's logs to D:/logstash-5.5.2/logs which is now configured via log4j2.properties
[2018-11-13T16:40:41,203][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>"main", "pipeline.workers"=>8, "pi
eline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>1000}
[2018-11-13T16:40:41,258][INFO ][logstash.pipeline        ] Pipeline main started
The stdin plugin is now waiting for input:
[2018-11-13T16:40:41,459][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
```

启动方式二：在 config 目录下新建 logstash.conf 文件，编辑以下内容：

```
input {
    stdin {}
}
output {
    stdout {}
}
```

控制台输入以下命令：

```
bin\logstash -f config\logstash.conf
```

2.2、访问日志生产平台的搭建

为让演示效果更加真实，这里直接利用 Nginx 产生的访问日志作为流量监控的元数据。因此，自己要先搭建 Nginx 运行环境，并部署一个可以访问的 web 项目。然后，在 logstash 的安装目录新建一个 patterns 目录，在此目录下创建 nginx 空白文件，内容如下：

```
NGINXACCESS %{IPORHOST:clientip} %{HTTPDUSER:ident} %{USER:auth} \[%{HTTPDATE:timestamp}\]
"(?:%{WORD:verb} %{URIPATH:uri}%{URIPARAM:param}(?:
HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})" %{NUMBER:response} (?:%{NUMBER:bytes}|-)

NGINXACCESSLOG %{NGINXACCESS} %{QS:referrer} %{QS:agent} %{QS:x_forwarded_for}
```

最后，对 logstash.conf 中的内容进行修改：

```
    input {
        file {
            path => ["D:/nginx-1.14.0/logs/access.log"]
            type => "nginx_access"
            start_position => "beginning"
        }
    }
    filter {
        if [type] == "nginx_access" {
            grok {
                patterns_dir => "D:/logstash-5.5.2/config/patterns/"
                match => {
                    "message" => "%{NGINXACCESS}"
                }
            }
            date {
                match => ["timestamp","dd/MMM/YYYY:HH:mm:ss Z"]
            }
            if [param] {
                ruby {
                    init => "@kname = ['quote','url_args']"
                    code => "
                        new_event =
LogStash::Event.new(Hash[@kname.zip(event.get('param').split('?'))])
                        new_event.remove('@timestamp')
                        event.append(new_event)
                    "
                }
                if [url_args] {
                    ruby {
                        init => "@kanme = ['key','value']"
                        code =>
"event.set('nested_args',event.get('url_args').split('&').clllect {|i|
Hash[@kanme.zip(i.split('='))]})"
                        remove_field => ["url_args","param","quote"]
                    }
                }
                mutate {
                    convert => ["response","integer"]
                    remove_field => "timestamp"
                }
            }
        }
    }
    output {
        stdout {
            codec => rubydebug
        }
    }
```

启动 logstash 便可以将 Nginx 日志同步到 logstash 中来。gork 内置表达式查询地址：
https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns

2.3、Logstash 与 ElasticSearch 集成

在 logstash.config 追加以下内容，即可与 ElasticSearch 实现无缝集成：

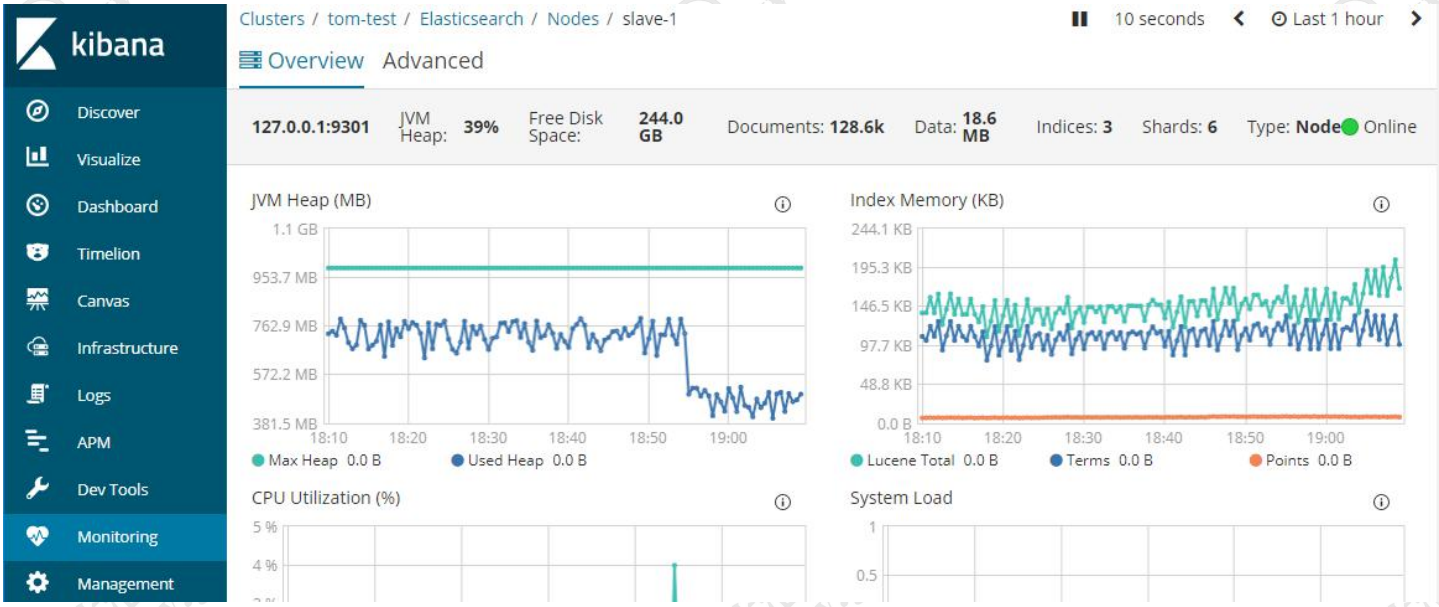
```
    elasticsearch {
            hosts => ["http://localhost:9200"]
            index => "logstash-%{type}-%{+YYYY.MM.dd}"
            document_type => "%{type}"
            sniffing => true
            #user => "tom"
            #password => "123456"
    }
```

2.4、利用 Kibana 实现网站流量可视化

官网 https://www.elastic.co/cn/downloads/past-releases 下载 Kibana6.5.1,解压即可。命令行启动 bin\kibana.bat 文件：

```
D:\kibana-5.5.2>bin\kibana.bat
  log   [09:00:57.914] [info][status][plugin:kibana@5.5.2] Status changed from uninitialized to green - Ready
  log   [09:00:58.006] [info][status][plugin:elasticsearch@5.5.2] Status changed from uninitialized to yellow - Waiting
for Elasticsearch
  log   [09:00:58.029] [info][status][plugin:console@5.5.2] Status changed from uninitialized to green - Ready
  log   [09:00:58.047] [info][status][plugin:metrics@5.5.2] Status changed from uninitialized to green - Ready
  log   [09:00:58.617] [info][status][plugin:timelion@5.5.2] Status changed from uninitialized to green - Ready
  log   [09:00:58.624] [info][listening] Server running at http://localhost:5601
  log   [09:00:58.625] [info][status][ui settings] Status changed from uninitialized to yellow - Elasticsearch plugin is
 yellow
  log   [09:00:58.793] [info][status][plugin:elasticsearch@5.5.2] Status changed from yellow to green - Kibana index rea
dy
  log   [09:00:58.795] [info][status][ui settings] Status changed from yellow to green - Ready
```

然后在浏览器输入 http://localhost:5601,即可看到可视化界面：



此时，只要 web 程序产生访问日志，就会被 Logstash 同步到 ElasticSearch 中来，同时，会被 Kibana 拉取到同时以可视化的界面展现出来，是不是很神奇呢？

此文档中的内容作为预习资料，大家可以提前预热一遍。具体的骚操作，大家来直播课堂听 Tom 老师分享。