Elasticsearch介绍

•Elasticsearch 是一个实时的分布式搜索分析引擎， 它能让你以一个之前从未有过的速度和规模，去探索你的数据。 它被用作全文检索、结构化搜索、分析以及这三个功能的组合

•Elasticsearch是一个基于Apache Lucene的开源搜索引擎。无论在开源还是专有领域，Lucene可以被认为是迄今为止最先进、性能最好的、功能最全的搜索引擎库。 但是，Lucene只是一个库。想要使用它，你必须使用Java来作为开发语言并将其直接集成到你的应用中，更糟糕的是，Lucene非常复杂，你需要深入了解检索的相关知识来理解它是如何工作的。

•Elasticsearch也使用Java开发并使用Lucene作为其核心来实现所有索引和搜索的功能，但是它的目的是通过简单的RESTful API来隐藏Lucene的复杂性，从而让全文搜索变得简单。

Elasticsearch可以做什么-问老大!问开发!

mysql虽然也可以搜索，比如查询某个字符串%，需要全表扫描
- Elasticsearch非常适合全文检索
- Elasticsrarch 可以灵活的存储不同类型的数据

应用场景
- 商城的商品搜索
- 所有产品的评论
- 高亮显示搜索内容
- 收集展示各种日志

Elasticsearch数据格式

- Elasticsearch 使用JavaScript Object Notation 或者JSON作为文档的序列化格式。JSON序列化被大多数编程语言所支持，并且已经成为 NoSQL领域的标准格式。 它简单、简洁、易于阅读。

- 考虑一下这个 JSON 文档，它代表了一个user对象:

```
{
"email": "john@smith.com",
"first_name": "John",
"last_name": "Smith",
"info":
    { "bio": "Eco-warrior and defender of the weak",
    "age": 25,
    "interests": [ "dolphins", "whales" ]
    },
"join_date": "2014/05/01"
}
```

Elasticsearch安装部署

安装方式

安装方式	优点	缺点
docker	1.部署方便 2.开箱即用 3.启动迅速	1.需要有docker的知识 2.修改配置麻烦，需要重新生成镜像 3.数据存储需要挂载目录
tar	1.部署灵活 2.对系统的侵占性小	1.需要自己写启动管理文件 2.目录提前需要规划好
Rpm|deb	1.部署方便 2.启动脚本安装即用 3.存放目录标准化	1.软件各个组件分散在不同的目录 2.卸载可能不干净 3.默认配置需要修改
ansible	1.极其的灵活 2.你想要的功能都有 3.批量部署速度快	1.需要学习ansible语法和规则 2.需要提前规划好所有的标准 3.需要专人维护

Elasticsearch安装部署-rpm安装

```
### 安装java
yum install -y java-1.8.0-openjdk.x86_64

### 下载安装软件
mkdir  -p /data/es_soft/
cd /data/es_soft/
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.6.0.rpm
rpm -ivh elasticsearch-6.6.0.rpm

### 配置启动
systemctl daemon-reload
systemctl enable elasticsearch.service
systemctl start elasticsearch.service
systemctl status elasticsearch.service

### 检查是否启动成功
ps -ef|grep elastic
lsof -i:9200
```

Elasticsearch目录文件说明

```
rpm -ql elasticsearch                          #查看elasticsearch软件安装了哪些目录
rpm -qc elasticsearch                          #查看elasticsearch的所有配置文件

/etc/elasticsearch/elasticsearch.yml     #配置文件
/etc/elasticsearch/jvm.options.           #jvm虚拟机配置文件
/etc/init.d/elasticsearch                      #init启动文件
/etc/sysconfig/elasticsearch                #环境变量配置文件
/usr/lib/sysctl.d/elasticsearch.conf       #sysctl变量文件，修改最大描述符
/usr/lib/systemd/system/elasticsearch.service   #systemd启动文件
/var/lib/elasticsearch                         # 数据目录
/var/log/elasticsearch                         #日志目录
/var/run/elasticsearch                         #pid目录
```

Elasticsearch配置说明-1

Elasticsearch 已经有了很好的默认值，特别是涉及到性能相关的配置或者选项,其它数据库可能需要调优，但总得来说，Elasticsearch不需要。如果你遇到了性能问题，解决方法通常是更好的数据布局或者更多的节点。

egrep -v "^#" /etc/elasticsearch/elasticsearch.yml

```
cluster.name: dba5                                  #集群名称
node.name: node-1                                   #节点名称
path.data: /data/elasticsearch                      #数据目录
path.logs: /var/log/elasticsearch                   #日志目录
bootstrap.memory_lock: true                         #锁定内存
network.host: localhost                             #绑定IP地址
http.port: 9200                                     #端口号
discovery.zen.ping.unicast.hosts: [ “localhost” ]           #集群发现的通讯节点
discovery.zen.minimum_master_nodes: 2                       #最小主节点数
```

Elasticsearch配置说明-3

修改完配置文件后我们需要重启一下
mkdir /data/elasticsearch
chown -R elasticsearch:elasticsearch /data/elasticsearch/
systemctl restart elasticsearch
systemctl status elasticsearch

这个时候可能会启动失败，查看日志可能会发现是锁定内存失败
官方解决方案
https://www.elastic.co/guide/en/elasticsearch/reference/6.6/setup-configuration-memory.html
https://www.elastic.co/guide/en/elasticsearch/reference/6.6/setting-system-settings.html#sysconfig

修改启动配置文件或创建新配置文件
方法1: systemctl edit elasticsearch
方法2: vim /usr/lib/systemd/system/elasticsearch.service
增加如下参数
[Service]
LimitMEMLOCK=infinity

重新启动
systemctl daemon-reload
systemctl restart elasticsearch

Elasticsearch术语及概念

Elasticsearch术语及概念

索引词
在elastiasearch中索引词（term）是一个能够被索引的精确值。foo,Foo,FOO几个单词是不同的索引词。索引词（term）是可以通过term查询进行准确的搜索。

文本(text)
文本是一段普通的非结构化文字。通常，文本会被分拆成一个个的索引词，存储在elasticsearch的索引库中。为了让文本能够进行搜索，文本字段需要事先进行分析了；当对文本中的关键词进行查询的时候，搜索引擎应该根据搜索条件搜索出原文本。

分析(analysis)
分析是将文本转换为索引词的过程，分析的结果依赖于分词器。比如：FOO BAR，Foo-Bar和foo bar这几个词有可能会被分析成相同的索引词foo和bar，这些索引词存储在Elasticsearch的索引库中。

集群(cluster)
集群由一个或多个节点组成，对外提供服务，对外提供索引和搜索功能。在所有节点，一个集群有一个唯一的名称默认为“elasticsearch”.此名称是很重要的，因为每个节点只能是集群的一部分，当该节点被设置为相同的集群名称时，就会自动加入集群。当需要有多个集群的时候，要确保每个集群的名称不能重复，，否则节点可能会加入到错误的集群。请注意，一个节点只能加入到一个集群。此外，你还可以拥有多个独立的集群，每个集群都有其不同的集群名称。

Elasticsearch术语及概念

节点(node)

一个节点是一个逻辑上独立的服务,它是集群的一部分,可以存储数据,并参与集群的索引和搜索功能。就像集群一样,节点也有唯一的名字,在启动的时候分配。如果你不想要默认名称,你可以定义任何你想要的节点名.这个名字在理中很重要,在Elasticsearch集群通过节点名称进行管理和通信.一个节点可以被配置加入到一个特定的集群。默认情况下,每个节点会加人名为Elasticsearch 的集祥中,这意味着如果你在网热动多个节点,如果网络畅通,他们能彼此发现井自动加人名为Elasticsearch 的一个集群中,你可以拥有多个你想要的节点。当网络没有集祥运行的时候,只要启动一个节点,这个节点会默认生成一个新的集群,这个集群会有一个节点。

分片(shard)

分片是单个Lucene 实例,这是Elasticsearch管理的比较底层的功能。索引是指向主分片和副本分片的逻辑空间。对于使用,只需要指定分片的数量,其他不需要做过多的事情。在开发使用的过程中,我们对应的对象都是索引,Elasticsearch 会自动管理集群中所有的分片,当发生故障的时候,Elasticsearch 会把分片移动到不同的节点或者添加新的节点。

一个索引可以存储很大的数据,这些空间可以超过一个节点的物理存储的限制。例如,十亿个文档占用磁盘空间为1TB。仅从单个节点搜索可能会很慢,还有一台物理机器也不一定能存储这么多的数据。为了解决这一问题,Elasticsearch将索引分解成多个分片。当你创建一个索引,你可以简单地定义你想要的分片数量。每个分片本身是一个全功能的、独立的单元,可以托管在集群中的任何节点。

主分片

每个文档都存储在一个分片中,当你存储一个文档的时候,系统会首先存储在主分片中,然后会复制到不同的副本中。默认情况下,一个索引有5个主分片。 你可以事先制定分片的数量,当分片一旦建立,则分片的数量不能修改。

Elasticsearch术语及概念

副本分片

每一个分片有零个或多个副本。副本主要是主分片的复制,其中有两个目的:
- 增加高可用性:当主分片失败的时候,可以从副本分片中选择一个作为主分片。
- 提高性能:当查询的时候可以到主分片或者副本分片中进行查询。默认情况下,一个主分片配有一个副本,但副本的数量可以在后面动态地配置增加。副本分片必部署在不同的节点上,不能部署在和主分片相同的节点上。

分片主要有两个很重要的原因是:
- 允许水平分割扩展数据。
- 允许分配和并行操作(可能在多个节点上)从而提高性能和吞吐量。

这些很强大的功能对用户来说是透明的,你不需要做什么操作,系统会自动处理。

索引(index)

索引是具有相同结构的文档集合。例如,可以有一个客户信息的索引,包括一个产品目录的索引,一个订单数据的索引。 在系统上索引的名字全部小写,通过这个名字可以用来执行索引、搜索、更新和删除操作等。在单个集群中,可以定义多个你想要的索引。

类型(type)

在索引中,可以定义一个或多个类型,类型是索引的逻辑分区。在一般情况下,一种类型被定义为具有一组公共字段的文档。例如,让我们假设你运行一个博客平台,并把所有的数据存储在一个索引中。在这个索引中,你可以定义一种类型为用户数据,一种类型为博客数据,另一种类型为评论数据。

文档(doc)

文档是存储在Elasticsearch中的一个JSON格式的字符串。它就像在关系数据库中表的一行。每个存储在索引中的一个文档都有一个类型和一个ID,每个文档都是一个JSON对象,存储了零个或者多个字段,或者键值对。原始的JSON 文档假存储在一个叫作Sour的字段中。当搜索文档的时候默认返回的就是这个字段。

Elasticsearch术语及概念

映射
映射像关系数据库中的表结构,每一个索引都有一个映射,它定义了索引中的每一个字段类型,以及一个索引范围内的设置。一个映射可以事先被定义,或者在第一次存储文档的时候自动识别。

字段
文档中包含零个或者多个字段,字段可以是一个简单的值(例如字符串、整数、日期),也可以是一个数组或对象的嵌套结构。字段类似于关系数据库中表的列。每个字段都对应一个字段类型,例如整数、字符串、对象等。字段还可以指定如何分析该字段的值。

主键
ID是一个文件的唯一标识,如果在存库的时候没有提供ID,系统会自动生成一个ID,文档的index/type/id必须是唯一的。

Elasticsearch术语及概念

复制

复制是一个非常有用的功能,不然会有单点问题。 当网络中的某个节点出现问题的时候,复制可以对故障进行转移,保证系统的高可用。因此,Elasticsearch 允许你创建一个或多个拷贝,你的索引分片就形成了所谓的副本或副本分片。

复制是重要的,主要的原因有:
- 它提供了高可用性,当节点失败的时候不受影响。需要注意的是,一个复制的分片不会存储在同一个节点中。
-它允许你扩展搜索量,提高并发量,因为搜索可以在所有副本上并行执行。
每个索引可以拆分成多个分片。索引可以复制零个或者多个分片.一旦复制,每个索引就有了主分片和副本分片.分片的数量和副本的数量可以在创建索引时定义.当创建索引后,你可以随时改变副本的数量,但你不能改变分片的数量.
默认情况下,每个索引分配5个分片和一个副本,这意味着你的集群节点至少要有两个节点,你将拥有5个主要的分片和5个副本分片共计10个分片.
每个Elasticsearch分片是一个Lucene 的索引。有文档存储数量限制,你可以在一个单一的Lucene索引中存储的最大值为lucene-5843,极限是2147483519(=integer.max_value-128)个文档。你可以使用cat/shards API监控分片的大小。

Elasticsearch交互-交互方式

所有其他语言可以使用RESTful API通过端口9200和Elasticsearch进行通信，你可以用你最喜爱的web客户端访问Elasticsearch.事实上,你甚至可以使用curl命令和Elasticsearch交互 。

一个 Elasticsearch 请求和任何 HTTP 请求一样由若干相同的部件组成:
curl -X<VERB> '<PROTOCOL>://<HOST>:<PORT>/<PATH>?<QUERY_STRING>' -d '<BODY>'

VERB： 适当的 HTTP 方法 或 谓词 : GET`、 `POST`、 `PUT`、 `HEAD 或者 `DELETE`。 PROTOCOL http 或者 https`(如果你在 Elasticsearch 前面有一个 `https 代理)

HOST：Elasticsearch 集群中任意节点的主机名，或者用 localhost 代表本地机器上的节点。

PORT：运行 Elasticsearch HTTP 服务的端口号，默认是 9200 。

PATH： API 的终端路径(例如 _count 将返回集群中文档数量)。Path 可能包含多个组件，例如: _cluster/stats 和 _nodes/stats/jvm 。

QUERY_STRING: 任意可选的查询字符串参数 (例如 ?pretty 将格式化地输出 JSON 返回值，使其更容易阅读)

BODY：一个 JSON 格式的请求体 (如果请求需要的话)

Elasticsearch-交互

Elasticsearch交互- curl命令行交互

三种交互方式

curl命令：

- 最繁琐
- 最复杂
- 最容易出错
- 不需要安装任何软件，只需要有curl命令

es-head插件

- 查看数据方便
- 操作相对容易
- 需要node环境

kibana

- 查看数据以及报表格式丰富
- 操作很简单
- 需要java环境和安装配置kibana

Elasticsearch交互- head插件交互

Head插件在5.0以后安装方式发生了改变，需要nodejs环境支持，或者直接使用别人封装好的docker镜像
插件官方地址
https://github.com/mobz/elasticsearch-head

使用docker部署elasticsearch-head
docker pull alivv/elasticsearch-head
docker run --name es-head -p 9100:9100 -dit elivv/elasticsearch-head

使用nodejs编译安装elasticsearch-head
yum install nodejs npm openssl screen -y
node -v
npm -v
npm install -g cnpm --registry=https://registry.npm.taobao.org
cd /opt/
git clone git://github.com/mobz/elasticsearch-head.git
cd elasticsearch-head/
cnpm install
screen -S es-head
cnpm run start
Ctrl+A+D

修改ES配置文件支持跨域
http.cors.enabled: true
http.cors.allow-origin: "*"

Elasticsearch API

Elasticsearch API

创建索引

```
curl -XPUT '192.168.47.178:9200/vipinfo?pretty'
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "  vipinfo"
}
```

插入文档数据

```
curl -XPUT '192.168.47.178:9200/vipinfo/user/1?pretty' -H 'Content-Type: application/json' -d'
{
    "first_name" : "John",
    "last_name": "Smith",
    "age" : 25,
    "about" : "I love to go rock climbing", "interests": [ "sports", "music" ]
}
'

curl -XPUT  'localhost:9200/vipinfo/user/2?pretty' -H 'Content-Type: application/json' -d' {
"first_name": "Jane",
"last_name" : "Smith",
"age" : 32,
"about" : "I like to collect rock albums", "interests": [ "music" ]
}'

curl –XPUT  'localhost:9200/vipinfo/user/3?pretty' -H 'Content-Type: application/json' -d' {
"first_name": "Douglas", "last_name" : "Fir",
"age" : 35,
"about": "I like to build cabinets", "interests": [ "forestry" ]
}'
```

Elasticsearch 文档元数据

一个文档不仅仅包含它的数据 ，也包含元数据 —— 有关文档的信息。
三个必须的元数据元素如下:

_index 文档在哪存放

_type 文档表示的对象类别

_id 文档唯一标识

_index
一个 索引 应该是因共同的特性被分组到一起的文档集合。 例如，你可能存储所有的产品在索引 products 中，而存储所有销售的交易到索引 sales 中

_type
数据可能在索引中只是松散的组合在一起，但是通常明确定义一些数据中的子分区是很有用的。 例如，所 有的产品都放在一个索引中，但是你有许多不同的产品类别，比如 "electronics" 、 "kitchen" 和 "lawn- care" 。 这些文档共享一种相同的(或非常相似)的模式:他们有一个标题、描述、产品代码和价格。他们只是正 好属于“产品”下的一些子类。
Elasticsearch 公开了一个称为 types (类型)的特性，它允许您在索引中对数据进行逻辑分区。不同 types 的文档可能有不同的字段，但最好能够非常相似。

_id
ID 是一个字符串， 当它和 _index 以及 _type 组合就可以唯一确定 Elasticsearch 中的一个文档。 当你创 建一个新的文档，要么提供自己的 _id ，要么让 Elasticsearch 帮你生成

Elasticsearch API

查询索引中所有的
curl -XGET localhost:9200/vipinfo/user/_search?pretty

查询指定文档数据
curl -XGET 'localhost:9200/vipinfo/user/1?pretty'
curl -XGET 'localhost:9200/vipinfo/user/2?pretty'

按条件查询文档数据
查询索引中符合条件的数据:搜索姓氏为Smith的雇员
curl -XGET 'localhost:9200/vipinfo/user/_search?q=last_name:Smith&pretty'

使用Query-string查询
curl -XGET 'localhost:9200/vipinfo/user/_search?pretty' -H 'Content-Type: application/json'
```
{
  "query" : {
    "match" : {
        "last_name" : "Smith"
    }
  }
}
'
```

Elasticsearch API

使用过滤器查询

搜索姓氏为 Smith 的雇员，但这次我们只需要年龄大于 30 的。
查询需要稍作调整，使用过滤器 filter ，它支持高效地执行一个结构化查询

```
curl -XGET 'localhost:9200/vipinfo/user/_search?pretty' -H 'Content-Type: application/json' -d'{
  "query" : {
    "bool": {
      "must": {
        "match" : {
          "last_name" : "smith"
          }
      },
      "filter": {
        "range" : {"age" : { "gt" : 30 }
          }
        }
      }
    }
}'
```

Elasticsearch API

更新数据的两种方式

#PUT更新，需要填写完整的信息

```
curl -XPUT 'localhost:9200/vipinfo/user/1?pretty' -H 'Content-Type: application/json' -d'
{
    "first_name" : "John",
    "last_name": "Smith",
    "age" : 27,
    "about" : "I love to go rock climbing", "interests": [ "sports", "music" ]
}
```

#POST更新，只需要填写需要更改的信息

```
curl -XPOST 'localhost:9200/vipinfo/user/1?pretty' -H 'Content-Type: application/json' -d'
{
    "age" : 29
}
```

Elasticsearch API

删除指定文档数据

```
curl -XDELETE 'localhost:9200/vipinfo/user/1?pretty'
{
  "_index" : "vipinfo",
  "_type" : "user",
  "_id" : "1",
  "_version" : 2,
  "result" : "deleted",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 1,
  "_primary_term" : 2
}
```

删除索引

```
curl -XDELETE 'localhost:9200/vipinfo?pretty'
{
  "acknowledged" : true
}
```

Elasticsearch 集群

Elasticsearch 集群

Elasticsearch 可以横向扩展至数百(甚至数千)的服务器节点，同时可以处理PB级数据
Elasticsearch 天生就是分布式的，并且在设计时屏蔽了分布式的复杂性。
Elasticsearch 尽可能地屏蔽了分布式系统的复杂性。
这里列举了一些在后台自动执行的操作:
分配文档到不同的容器 或 分片中,文档可以储存在一个或多个节点中
按集群节点来均衡分配这些分片，从而对索引和搜索过程进行负载均衡
复制每个分片以支持数据冗余，从而防止硬件故障导致的数据丢失
将集群中任一节点的请求路由到存有相关数据的节点
集群扩容时无缝整合新节点，重新分配分片以便从离群节点恢复

一个运行中的 Elasticsearch 实例称为一个 节点，而集群是由一个或者多个拥有相同 cluster.name 配置的节点组成,它们共同承担数据和负载的压力。
当有节点加入集群中或者从集群中移除节点时，集群将会重新平均分布所有的数据。

当一个节点被选举成为主节点时,它将负责管理集群范围内的所有变更,例如增加、删除索引,或者增加、删除节点等.而主节点并不需要涉及到文档级别的变更和搜索等操作,所以当集群只拥有一个主节点的情况下,即使流量的增加它也不会成为瓶颈.任何节点都可以成为主节点。我们的示例集群就只有一个节点,所以它同时也成为了主节点。

作为用户,我们可以将请求发送到 集群中的任何节点,包括主节点.每个节点都知道任意文档所处的位置,并且能够将我们的请求直接转发到存储我们所需文档的节点.无论我们将请求发送到哪个节点,它都能负责从各个包含我们所需文档的节点收集回数据,并将最终结果返回給客户端.
Elasticsearch 对这一切的管理都是透明的。

Elasticsearch 集群-安装部署

集群的安装部署和单机没有什么区别，区别在于配置文件里配置上集群的相关参数

```
cluster.name: dba6    #集群名称，一个集群内所有节点要一样
node.name: node-1
path.data: /data/elasticsearch
path.logs: /var/log/elasticsearch
bootstrap.memory_lock: true
network.host: localhost,localhost        #节点所在机器的IP地址
http.port: 9200
discovery.zen.ping.unicast.hosts: ["IP1","IP2"]  #集群节点发现IP
discovery.zen.minimum_master_nodes: 1  #最大master节点数
http.cors.enabled: true
http.cors.allow-origin: "*"
```

Elasticsearch 集群-查看集群信息

官网地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-health.html

```
curl -XGET 'http://localhost:9200/_cluster/health?pretty'
{
  "cluster_name" : "elasticsearch",
  "status" : "yellow",
  "timed_out" : false,
  "number_of_nodes" : 1,
  "number_of_data_nodes" : 1,
  "active_primary_shards" : 5,
  "active_shards" : 5,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 5,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 50.0
}
```

status 字段是我们最关心的。
green 所有的主分片和副本分片都正常运行。
yellow 所有的主分片都正常运行，但不是所有的副本分片都正常运行。
red 有主分片没能正常运行。

Elasticsearch 集群-查看集群信息

查看系统检索信息
Cluster Stats API允许从群集范围的角度检索统计信息。
官网地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-stats.html
操作命令：
curl -XGET 'http://localhost:9200/_cluster/stats?human&pretty'

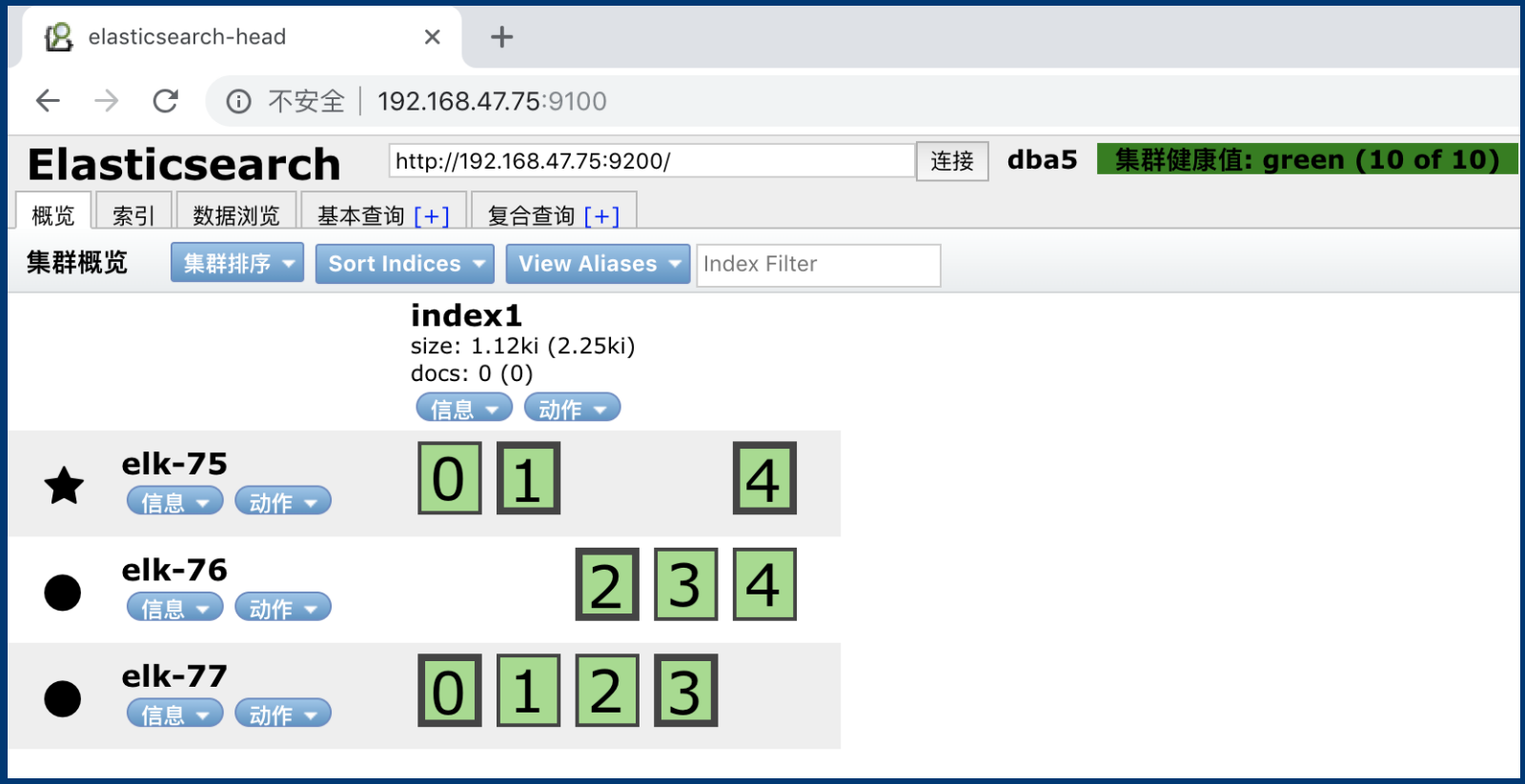
查看集群的设置
官方地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-get-settings.html
操作命令：
curl -XGET 'http://localhost:9200/_cluster/settings?include_defaults=true&human&pretty'

查询节点的状态
官网地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-nodes-info.html
操作命令：
curl -XGET 'http://localhost:9200/_nodes/procese?human&pretty'
curl -XGET 'http://localhost:9200/_nodes/_all/info/jvm,process?human&pretty'
curl -XGET 'http://localhost:9200/_cat/nodes?human&pretty'

Elasticsearch 集群-分片与复制

默认ES会创建5分片1副本的配置

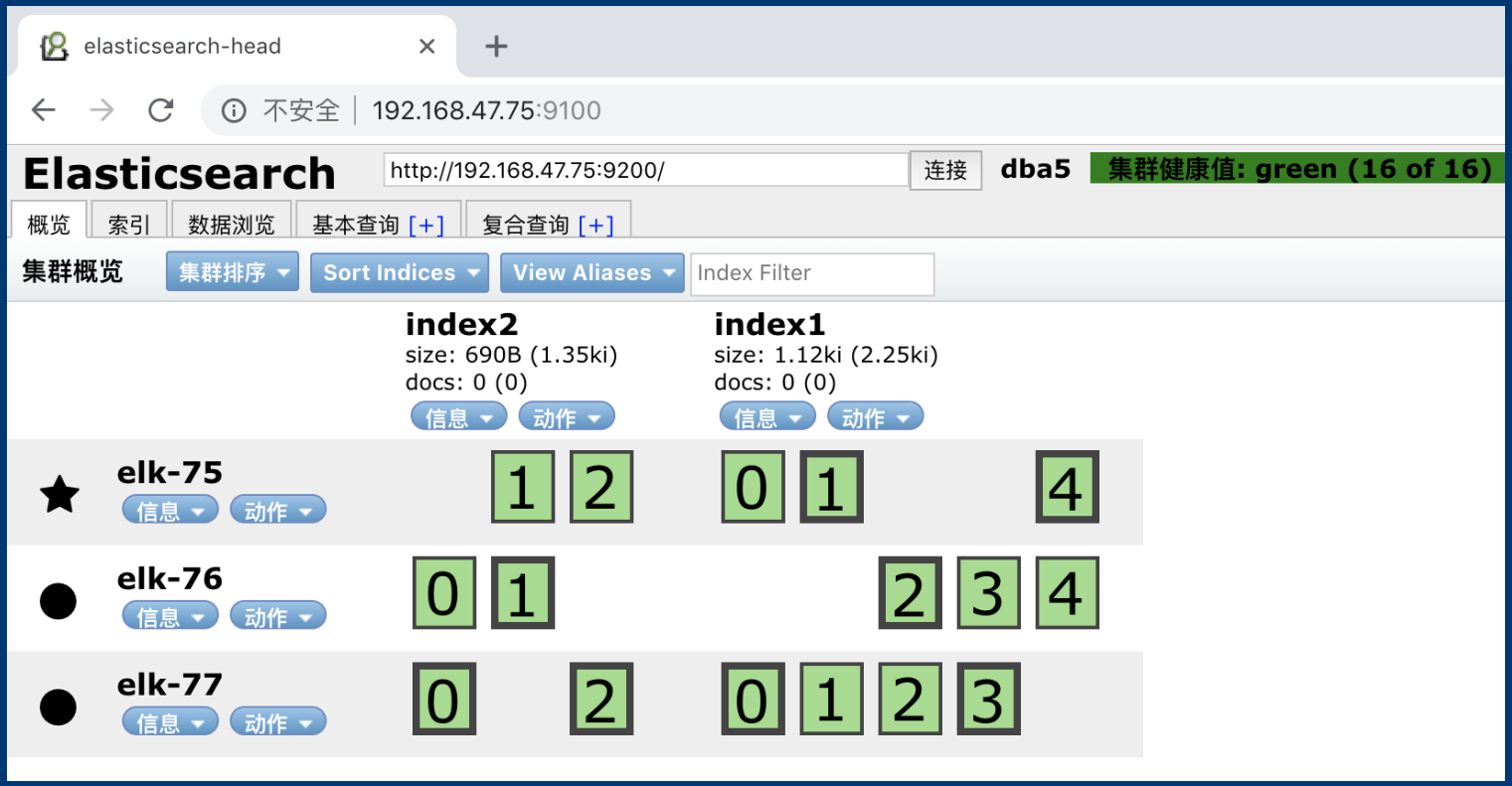
```
curl -XPUT 'localhost:9200/index1?pretty'
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "index1"
}
```



Elasticsearch 集群-分片与复制

同时我们也可以手动分配分片数和副本数

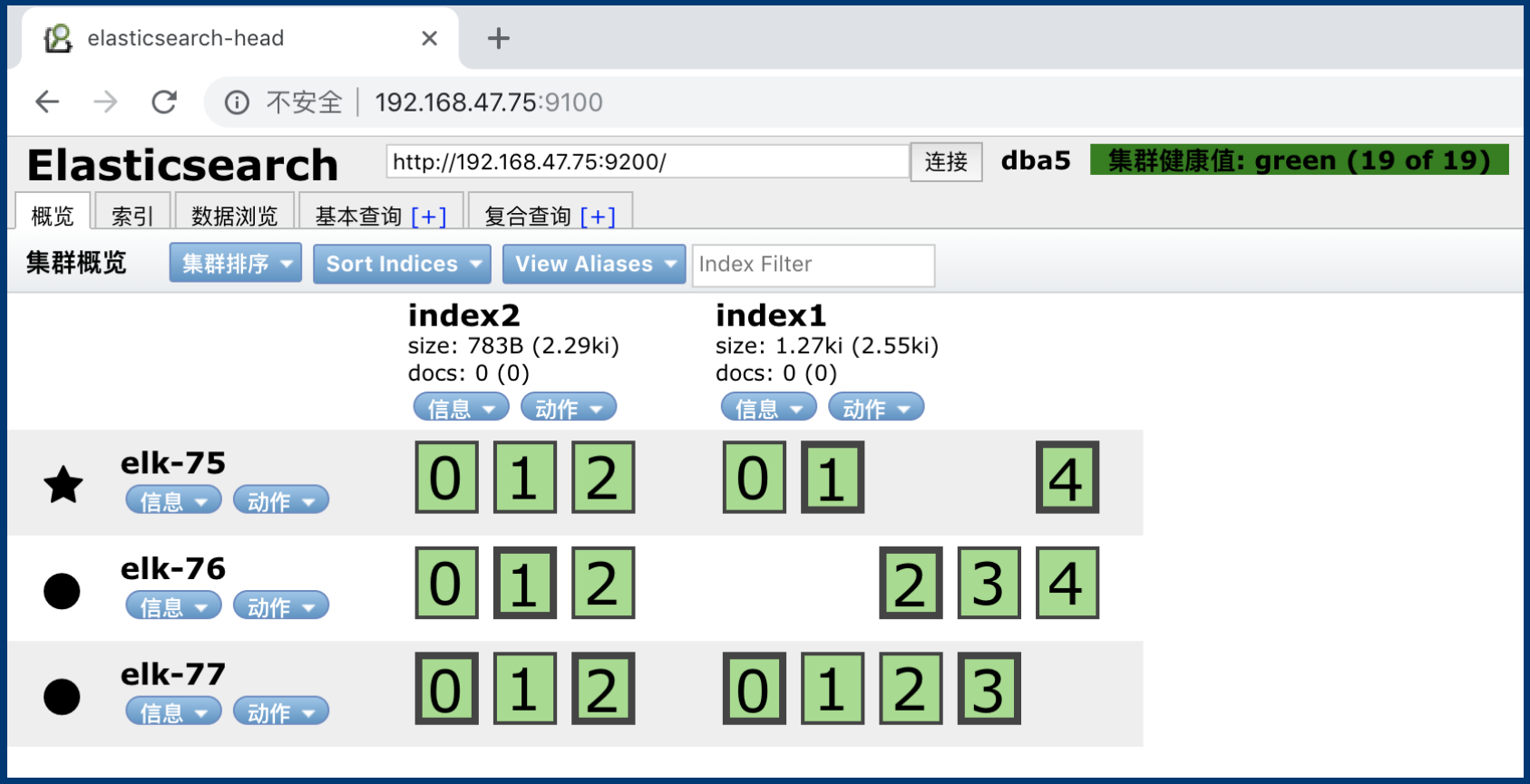
```
curl -XPUT 'localhost:9200/index2?pretty' -H 'Content-Type: application/json' -d'
{
"settings" : {
  "number_of_shards" : 3,
  "number_of_replicas" : 1
 }
}'
```



Elasticsearch 集群-分片与复制

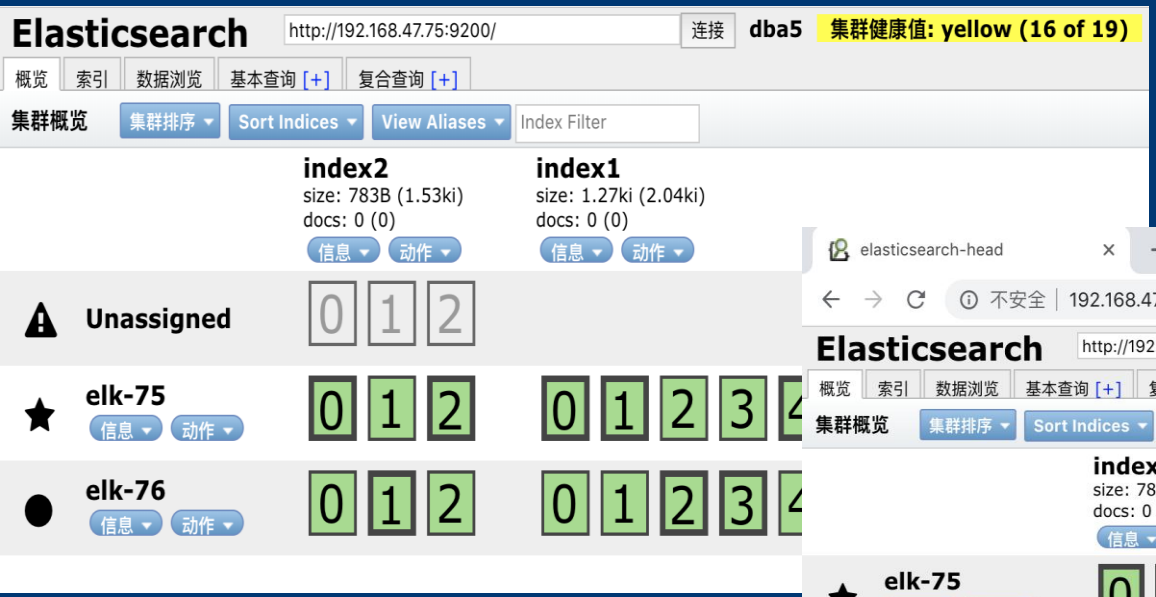
调整福本数
分片数一旦创建就不能再更改了，但是我们可以调整副本数

```
curl -XPUT 'localhost:9200/index2/_settings?pretty' -H 'Content-Type: application/json' -d'
{
"settings" : {
  "number_of_replicas" : 2
 }
}'
```



Elasticsearch 集群-分片与复制

此时如果我们关闭一个节点，会发现集群状态发生了改变
在我们关闭 Node 1 的同时也失去了主分片 1 和 2，并且在缺失主分片的时候索引也不能正常工作。 如果 此时来检查集群的状况，我们看到的状态将会为 red :不是所有主分片都在正常工作。 幸运的是，在其它节点上存在着这两个主分片的完整副本， 所以新的主节点立即将这些分片在 Node 2 和 Node 3 上对应的副本分片提升为主分片， 此时集群的状态将会为 yellow 。 这个提升主分片的过程是瞬间 发生的，如同按下一个开关一般。
为什么我们集群状态是 yellow 而不是 green 呢? 虽然我们拥有所有的三个主分片，但是同时设置了每个主 分片需要对应2份副本分片，而此时只存在一份副本分片。 所以集群不能为 green 的状态，不过我们不必 过于担心:如果我们同样关闭了 Node 2 ，我们的程序 依然 可以保持在不丢任何数据的情况下运行，因为 Node 3 为每一个分片都保留着一份副本。
如果我们重新启动 Node 1 ，集群可以将缺失的副本分片再次进行分配



Elasticsearch x-pack监控功能

Elasticsearch x-pack 监控功能

x-pack为elasticsearch, logstash, kibana提供了监控，报警，用户认证等功能，属于一个集成的插件。

6.x以前是以插件形式存在
6.x以后默认安装的时候就集成在软件包里了

不过这些功能并不全是免费的，具体分为四个版本：开源，基础，黄金，铂金四个版本

默认安装的时候是基础版

不同版本功能对比网址：
https://www.elastic.co/subscriptions

由于x-pack存在可以破解的方法
索性，官方在6.3以后直接将监控功能免费了

也就是说，不用破解，默认的基础班就可以使用x-pack的官方监控，为官方点赞！

Elasticsearch x-pack 监控功能

#架构说明：
https://www.elastic.co/guide/en/elastic-stack-overview/6.6/how-monitoring-works.html

#ES配置说明
https://www.elastic.co/guide/en/kibana/current/monitoring-kibana.html

#开启数据监控参数
GET _cluster/settings
PUT _cluster/settings { "persistent": { "xpack.monitoring.collection.enabled": true } }

#修改ES配置文件，最后添加2行
xpack.monitoring.exporters.my_local:
  type: local

Elasticsearch Search Guard 安装配置

Elasticsearch 权限认证Search Guard

由于x-pack里关于安全认证是收费功能，而我们又不想去破解
那么替代的方案就是Search Guard

相关网址：
https://docs.search-guard.com/latest/main-concepts
https://docs.search-guard.com/latest/demo-installer
https://docs.search-guard.com/latest/search-guard-versions
https://docs.search-guard.com/latest/search-guard-installation

Elasticsearch运维工具

Elasticsearch 优化

https://www.elastic.co/guide/en/elasticsearch/reference/current/system-config.html

1.JVM最大最小内存调整
2.关闭SWAP分区
3. vm.max_map_count调整
4.ulimit调整
5.磁盘性能

Elasticsearch-dump

https://github.com/taskrabbit/elasticsearch-dump

Elasticsearch python操作

https://pypi.org/project/elasticsearch/

https://elasticsearch-py.readthedocs.io/en/master/

Elasticsearch 防脑裂配置

https://www.elastic.co/guide/en/elasticsearch/reference/6.6/modules-discovery-zen.html

discovery.zen.minimum_master_nodes: 2
discovery.zen.fd.ping_interval: 10s
discovery.zen.fd.ping_timeout: 60s
discovery.zen.fd.ping_retries: 6

Elasticsearch项目-中文分词器

官方地址
https://github.com/medcl/elasticsearch-analysis-ik

分词器安装
cd /usr/share/elasticsearch/bin
./elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v6.4.2/elasticsearch-analysis-ik-6.4.2.zip

分词器测试
创建索引
curl -XPUT http://localhost:9200/index

创建映射
```
curl -XPOST http://localhost:9200/index/fulltext/_mapping -H 'Content-Type:application/json' -d'
{
        "properties": {
            "content": {
                "type": "text",
                "analyzer": "ik_max_word",
                "search_analyzer": "ik_max_word"
            }
        }

}'
```

Elasticsearch项目-中文分词器

创建一些文档

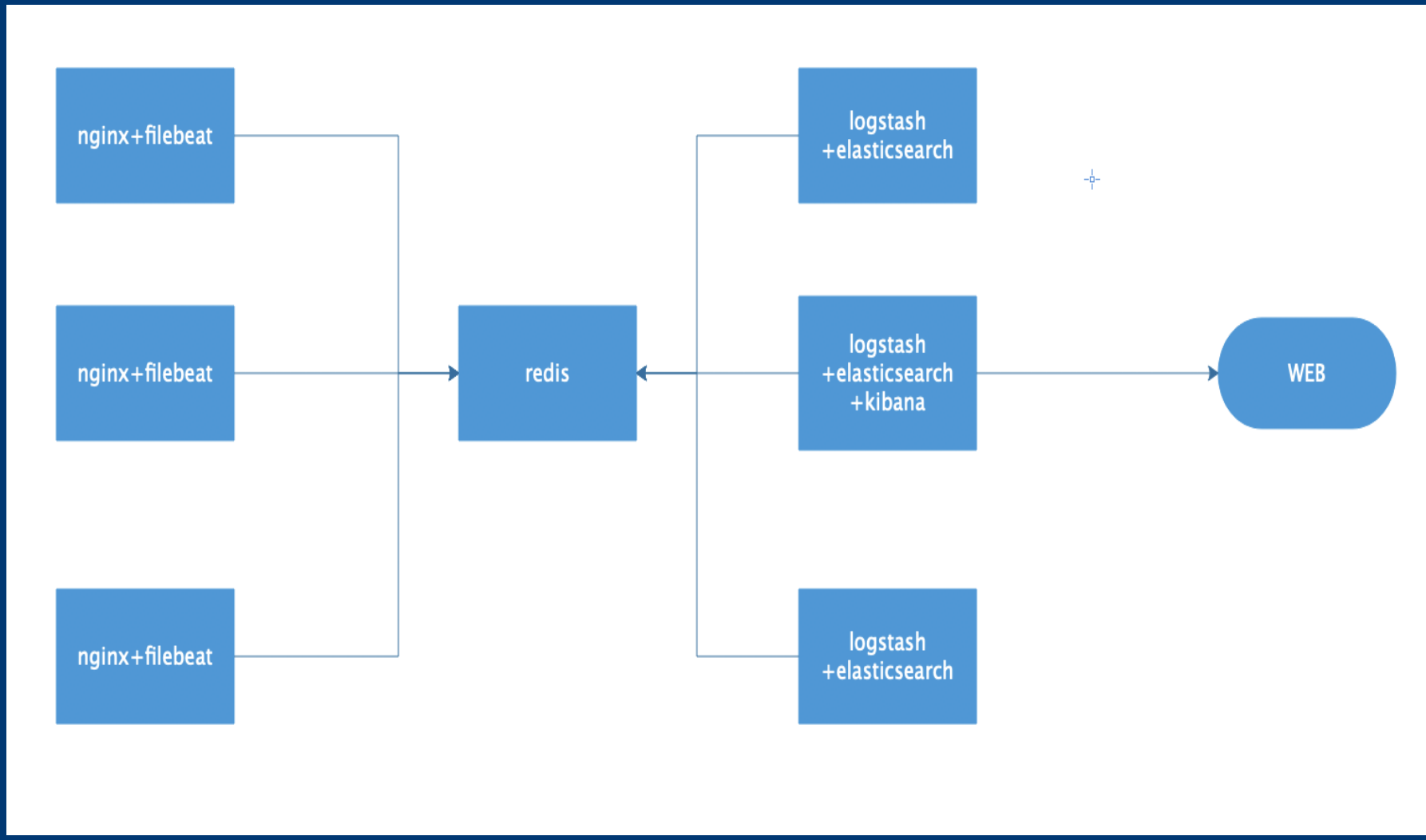
```
curl -XPOST http:// localhost:9200/index/fulltext/1 -H 'Content-Type:application/json' -d'
{"content":"美国留给伊拉克的是个烂摊子吗"}
'

curl -XPOST http:// localhost:9200/index/fulltext/2 -H 'Content-Type:application/json' -d'
{"content":"公安部：各地校车将享最高路权"}
'

curl -XPOST http:// localhost:9200/index/fulltext/3 -H 'Content-Type:application/json' -d'
{"content":"中韩渔警冲突调查：韩警平均每天扣1艘中国渔船"}
'

curl -XPOST http://localhost:9200/index/fulltext/4 -H 'Content-Type:application/json' -d'
{"content":"中国驻洛杉矶领事馆遭亚裔男子枪击 嫌犯已自首"}
'
```

查询

```
curl -XPOST http:// localhost:9200/index/fulltext/_search?pretty  -H 'Content-Type:application/json' -d'
{
    "query" : { "match" : { "content" : "中国" }},
    "highlight" : {
        "pre_tags" : ["<tag1>", "<tag2>"],
        "post_tags" : ["</tag1>", "</tag2>"],
        "fields" : {
            "content" : {}
        }
    }
}
'
```

Elasticsearch项目-日志收集展示

架构图



Elasticsearch项目-日志收集展示

Nginx日志修改

```
log_format access_json '{"@timestamp":"$time_iso8601",'
        "'host':'$server_addr',"
        "'clientip':'$remote_addr',"
        "'size':$body_bytes_sent,"
        "'responsetime':$request_time,"
        "'upstreamtime':'$upstream_response_time',"
        "'upstreamhost':'$upstream_addr',"
        "'http_host':'$host',"
        "'url':'$uri',"
        "'domain':'$host',"
        "'xff':'$http_x_forwarded_for',"
        "'referer':'$http_referer',"
        "'status':'$status'}';
```

redis配置

```
daemonize yes
bind localhost
port 6380
pidfile /opt/redis_cluster/redis_6380/pid/redis_6380.pid
logfile /opt/redis_cluster/redis_6380/logs/redis_6380.log
databases 16
```

Elasticsearch项目-日志收集展示

filebeat配置

```
filebeat.prospectors:
- type: log
  enabled: true
  paths:
    - /usr/local/nginx/logs/*access.log
  json.keys_under_root: true
  json.overwrite_keys: true

output.redis:
    hosts: ["localhost"]
    key: "filebeat"
    db: 0
    timeout: 5
```

redis验证数据

```
keys *
LLEN filebeat
RPOP filebeat
```

Elasticsearch项目-日志收集展示

```
 input {
  redis {
    host => "localhost"
    port => "6380"
    db => "0"
    key => "filebeat"
    data_type => "list"
  }
}

filter {
  mutate {
    convert => ["upstream_time", "float"]
    convert => ["request_time", "float"]
  }
}

output {
  if [source] == "/usr/local/nginx/logs/act.goumin.com_access.log" {
    elasticsearch {
      hosts => "http://localhost:9200"
      manage_template => false
      index => "act-%{+YYYY.MM}"
    }
  }


  if [source] == "/usr/local/nginx/logs/app.goumin.com_access.log" {
    elasticsearch {
      hosts => "http:// localhost:9200"
      manage_template => false
      index => "app-%{+YYYY.MM}"
    }
  }
```

Elasticsearch 利用模版收集日志

https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-module-nginx.html