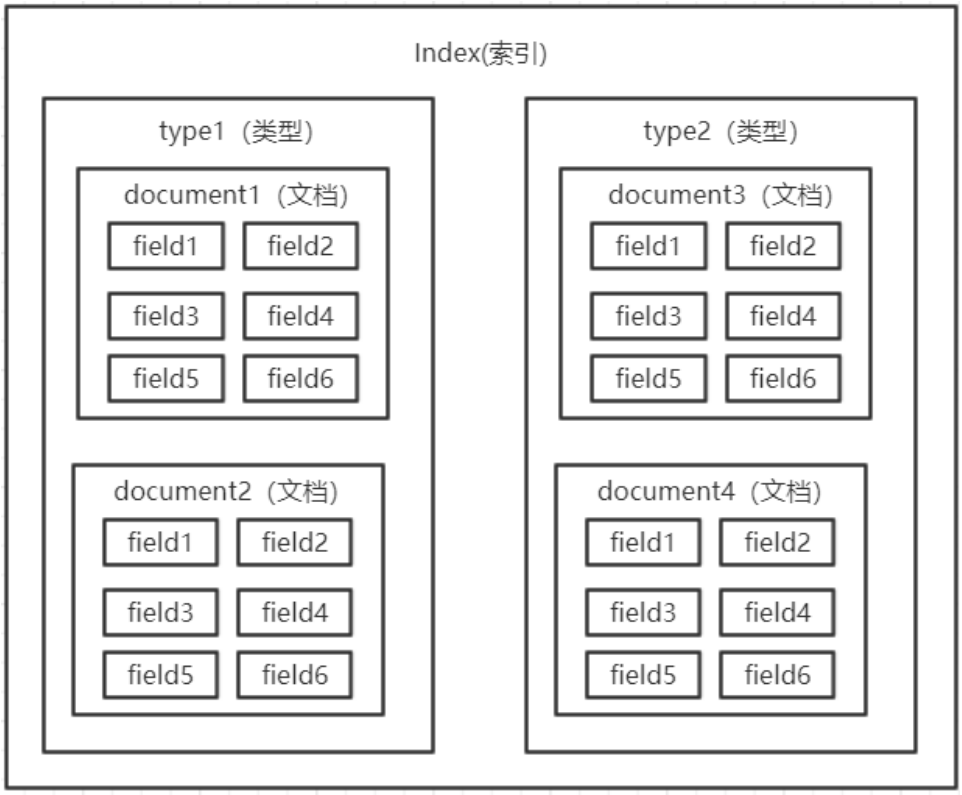
ELK

1.1、ElasticSearch

1.1.1、名词解释

1. 基本术语：
   之前版本的结构
   Es6.0 之后：一个索引中只有一个 type



- 文档 document
  用户存储在 es 中的数据文档
  - 元数据
    _index:文档所在索引名称
    _type:文档所在类型名称
    _id:文档唯一 id
    _uid:组合 id，由_type 和_id 组成（6.x 后，_type 不再起作用，同_id）
    _source:文档的原始 Json 数据，包括每个字段的内容
    _all:将所有字段内容整合起来，默认禁用（用于对所有字段内容检索）
- 索引 Index
  由具有相同字段的文档列表组成，用于定义字段名和字段值，一个集群或

elasticsearch 由多个索引组成，例如可以按照日期生成多个索引，方便数据搜索

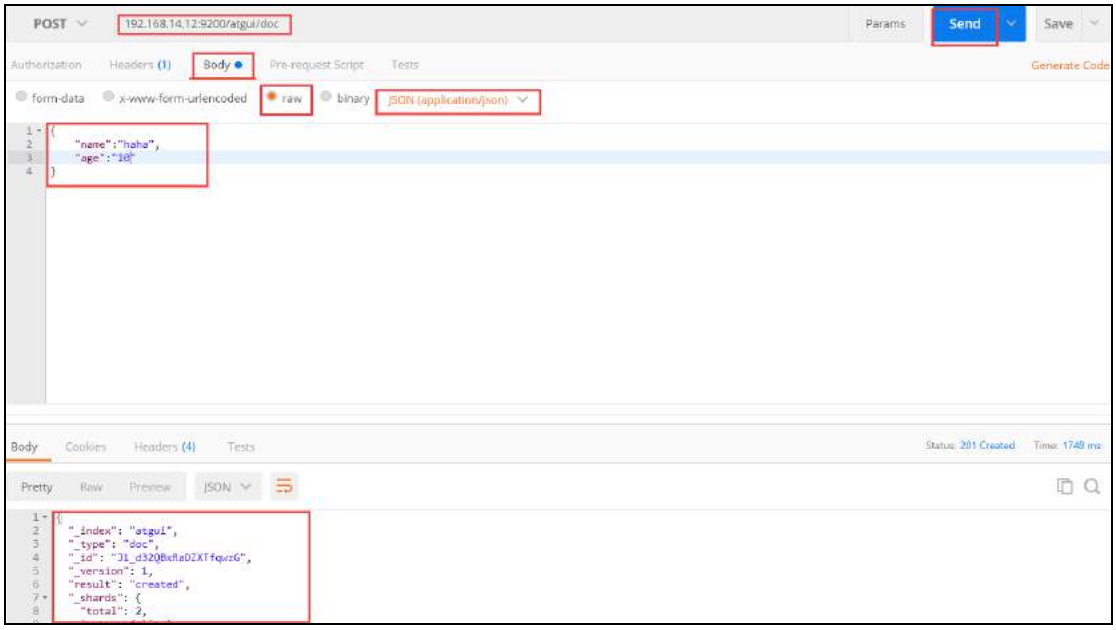
- 类型 Type
  具有相同特征文档的集合（ES6 之后一个索引中只能定义一个 type）
- 字段 Field
  具有相同特性数据名称

类型名称	组成
字符串	text、keyword（不分词）
数值型	long、integer、short、byte、double、float、half_float、scaled_float(长度短)
布尔	boolean
日期	Date
二进制	binary
范围类型	Integer_range、float_range、long_range、double_range、date_range(做数据范围查询)
坐标	附近的人

2. 集群术语：
   - 节点 Node
     一个 Elasticsearch 的运行实例，时集群构成的基本单元
   - 集群 cluster
     由一个或多个节点组成，对外提供服务

1.1.2、Elasticsearch 的 Rest API

- REST 访问 ES 方式（需要 Http Method、URI）
  1. 浏览器（postman）



2. Linux 命令行

   请求：

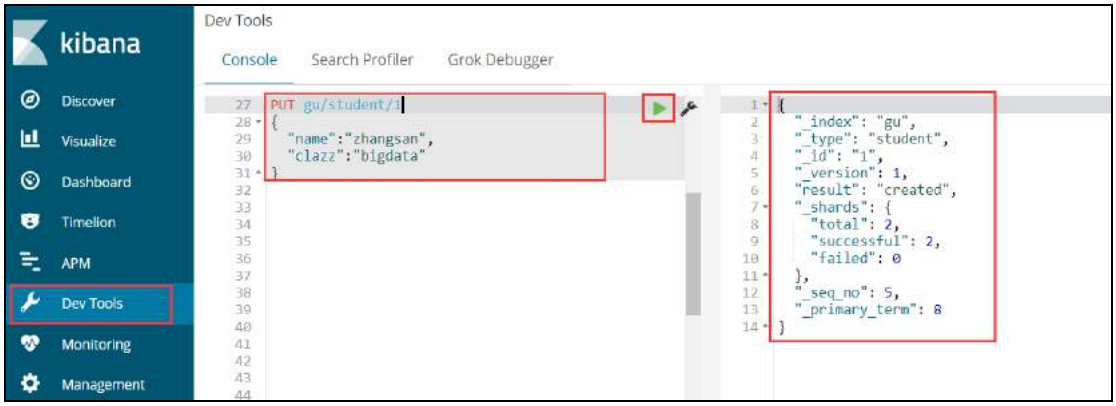
   [root@localcurl -XPOST 'http://192.168.14.12:9200/atguig/doc'
   -i -H
   "Content-Type:application/json"
   -d
   '{"name":"haha","age":"10"}'

   相应：

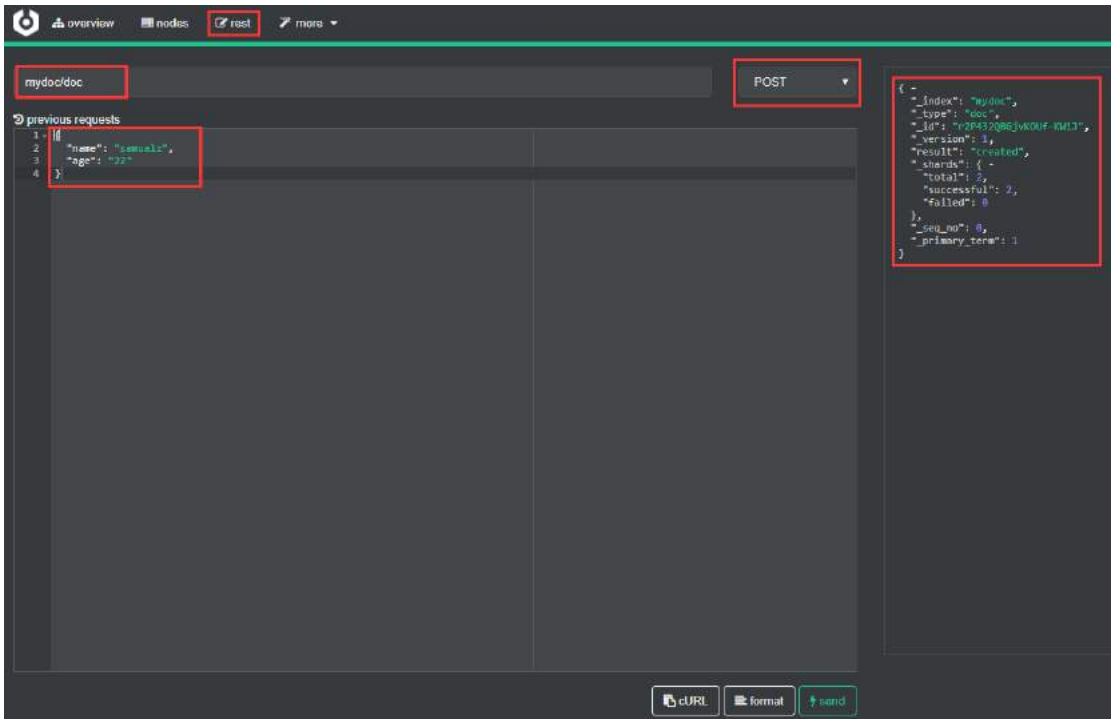
   HTTP/1.1 201 Created
   Location: /atguig/doc/KF_t32QBxRaDZXTftAxg
   content-type: application/json; charset=UTF-8
   content-length: 172

   {"_index":"atguig","_type":"doc","_id":"KF_t32QBxRaDZXTftAxg","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":0,"_primary_term":1}

3. Kibana 的 Dev Tools



4. Cerebro 插件



- ES 状态查看命令

  语法：ip:post/_cat/[args](?v|?format=json&pretty)

  （?v 表示显示字段说明,?format=json&pretty 表示显示成 json 格式）

  1、查看所有索引

  GET _cat/indices?v

  2、查看 es 集群状态

  GET _cat/health?v

  3、集群节点健康查看

  GET _cat/nodes?v

  4、列出倒叙索引

  GET _cat/segment?v

- 查看集群的状态

语法：GET _cluster/(args)[?v | ?format=json&pretty)

（?v 表示显示字段说明,?format=json&pretty 表示显示成 json 格式）

- 索引操作

1、添加

语法：PUT　index 名称

2、查看索引信息

语法：GET　index 名称

3、删除索引

语法：DELETE　　index 名称

4、查看索引状态

语法：HEAD　index 名称

语法：GET　index 名称/_status

- 文档操作

1、添加和修改

语法：(PUT|POST)　　index 名称/type 名称/[id]?

不添加 id 会自动生成 id

2、删除

语法：DELETE　　index 名称/type 名称/[id]

4、查看

语法：GET　index 名称/type 名称/[id]

1.1.3、正排索引和倒排索引

- 正排索引

记录文档 Id 到文档内容、单词的关联关系

docid	content
1	尚硅谷是最好的培训机构
2	php 是世界上最好的语言
3	尚硅谷是如何诞生的

- 倒排索引

记录单词到文档 id 的关联关系，包含：

单词词典（Term DicTionary）：记录所有文档的单词，一般比较大

倒排索引（Posting List）：记录单词倒排列表的关联信息

例如：尚硅谷

1、Term Dictionary

尚硅谷

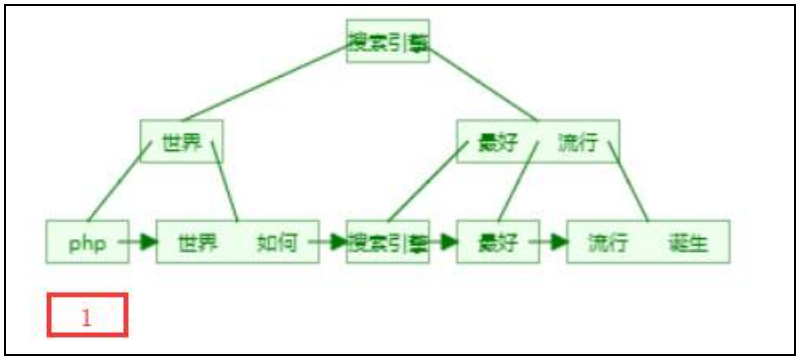
2、Posting List

DocId	TF	Position	Offset
1	1	0	<0,2>
3	1	0	<0,2>

DocId：文档 id，文档的原始信息
TF：单词频率，记录该词再文档中出现的次数，用于后续相关性算分
Position：位置，记录 Field 分词后，单词所在的位置，从 0 开始
Offset：偏移量，记录单词在文档中开始和结束位置，用于高亮显示等

3、内存结构

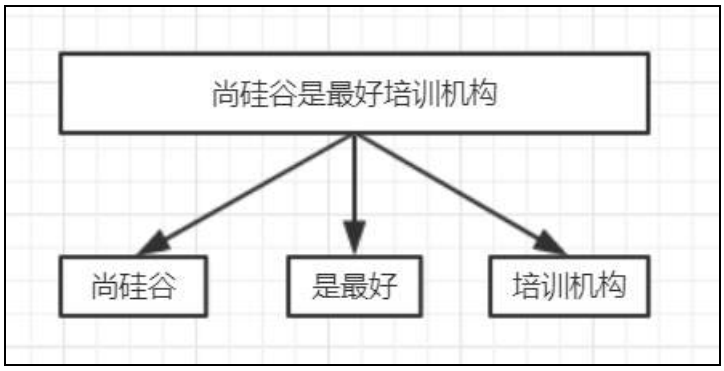
B+Tree



每个文档字段都有自己的倒排索引

1.1.4、分词

分词是指将文本转换成一系列单词（term or token）的过程，也可以叫做文本分析，在 es 里面称为 Analysis



- 分词机制

Character Filter	对原始文本进行处理	例：去除 html 标签、特殊字符等
Tokenizer	将原始文本进行分词	例：培训机构-->培训，机构
Token Filters	分词后的关键字进行加工	例：转小写、删除语气词、近义词和同义词等

- 分词 API

1、直接指定测试（指定分词器）

Request:

```
POST _analyze
{
  "analyzer": "standard",
  "text":"hello 1111"
}
```

Response:

```
{
  "tokens": [
    {
      "token": "hello",                #分词
      "start_offset": 0,               #开始偏移
      "end_offset": 5,                 #结束偏移
      "type": "<ALPHANUM>",            #单词类型
      "position": 0                    #位置
    },
    {
      "token": "world",
      "start_offset": 6,
      "end_offset": 11,
      "type": "<NUM>",
      "position": 1
    }
  ]
}
```

2、针对索引的字段进行分词测试（利用该字段的分词器）

Request：

```
POST atguigu/_analyze
{
  "field": "name",
  "text":"hello world"
}
```

Response:

```
{
  "tokens": [
    {
      "token": "hello",
      "start_offset": 0,
      "end_offset": 5,
      "type": "<ALPHANUM>",
      "position": 0
    },
    {
```

```
        "token": "world",
        "start_offset": 6,
        "end_offset": 11,
        "type": "<ALPHANUM>",
        "position": 1
      }
   ]
}
```

3、自定义分词器

Request:

```
POST _analyze
{
   "tokenizer": "standard",
   "filter": ["lowercase"],
   "text":"Hello WORLD"
}
```

Response:

```
{
   "tokens": [
      {
         "token": "hello",
         "start_offset": 0,
         "end_offset": 5,
         "type": "<ALPHANUM>",
         "position": 0
      },
      {
         "token": "world",
         "start_offset": 6,
         "end_offset": 11,
         "type": "<ALPHANUM>",
         "position": 1
      }
   ]
}
```

- Elasticsearch 自带的分词器

分词器（Analyzer）	特点
Standard（es 默认）	支持多语言，按词切分并做小写处理
Simple	按照非字母切分，小写处理
Whitespace	按照空格来切分
Stop	去除语气助词，如 the、an、的、这等

Keyword	不分词
Pattern	正则分词，默认\w+,即非字词符号做分割符
Language	常见语言的分词器（30+）

- 中文分词

分词器名称	介绍	特点	地址
IK	实现中英文单词切分	自定义词库	https://github.com/medcl/elasticsearch-analysis-ik
Jieba	python 流行分词系统，支持分词和词性标注	支持繁体、自定义、并行分词	http://github.com/sing1ee/elasticsearch-jieba-plugin
Hanlp	由一系列模型于算法组成的 java 工具包	普及自然语言处理在生产环境中的应用	https://github.com/hankcs/HanLP
THULAC	清华大学中文词法分析工具包	具有中文分词和词性标注功能	https://github.com/microbun/elasticsearch-thulac-plugin

- Character Filters

  在进行 Tokenizer 之前对原始文本进行处理，如增加、删除或替换字符等

HTML Strip	去除 html 标签和转换 html 实体
Mapping	字符串替换操作
Pattern Replace	正则匹配替换

注意：进行处理后，会影响后续 tokenizer 解析的 position 和 offset

Request：

```
POST _analyze
{
  "tokenizer": "keyword",
  "char_filter": ["html_strip"],
  "text":"<div><h1>B<sup>+</sup>Trees</h1></div>"
}
```

Response:

```
{
  "tokens": [
    {
      "token": """


B+Trees


""",
```

```
      "start_offset": 0,
      "end_offset": 38,
      "type": "word",
      "position": 0
    }
  ]
}
```

- Token Filter

  对输出的单词（term）进行增加、删除、修改等操作

Lowercase	将所有 term 转换为小写
stop	删除 stop words
NGram	和 Edge NGram 连词分割
Synonym	添加近义词的 term

Request:

```
POST _analyze
{
  "tokenizer": "standard",
  "text":"a Hello World",
  "filter": [
      "stop",
      "lowercase",
      {
        "type":"ngram",
        "min_gram":3,
        "max_gram":4

      }
    ]
}
```

Response:

```
{
  "tokens": [
    {
      "token": "hel",
      "start_offset": 2,
      "end_offset": 7,
      "type": "<ALPHANUM>",
      "position": 1
    },
    {
      "token": "hell",
```

```
      "start_offset": 2,
      "end_offset": 7,
      "type": "<ALPHANUM>",
      "position": 1
    },
    {
      "token": "ell",
      "start_offset": 2,
      "end_offset": 7,
      "type": "<ALPHANUM>",
      "position": 1
    },
    {
      "token": "ello",
      "start_offset": 2,
      "end_offset": 7,
      "type": "<ALPHANUM>",
      "position": 1
    },
    {
      "token": "llo",
      "start_offset": 2,
      "end_offset": 7,
      "type": "<ALPHANUM>",
      "position": 1
    },
    {
      "token": "wor",
      "start_offset": 8,
      "end_offset": 13,
      "type": "<ALPHANUM>",
      "position": 2
    },
    {
      "token": "worl",
      "start_offset": 8,
      "end_offset": 13,
      "type": "<ALPHANUM>",
      "position": 2
    },
    {
      "token": "orl",
      "start_offset": 8,
      "end_offset": 13,
```

```
            "type": "<ALPHANUM>",
            "position": 2
        },
        {
            "token": "orld",
            "start_offset": 8,
            "end_offset": 13,
            "type": "<ALPHANUM>",
            "position": 2
        },
        {
            "token": "rld",
            "start_offset": 8,
            "end_offset": 13,
            "type": "<ALPHANUM>",
            "position": 2
        }
    ]
}
```

- 自定义分词 api

Request:

```
PUT my_analyzer
{
   "settings": {
      "analysis": {
         "analyzer": {
            "my":{
               "tokenizer":"punctuation",
               "type":"custom",
               "char_filter":["emoticons"],
               "filter":["lowercase","english_stop"]
            }
         },
         "tokenizer": {
            "punctuation":{
               "type":"pattern",
               "pattern":"[.,!?]"
            }
         },
         "char_filter": {
            "emoticons":{
               "type":"mapping",
```

```
                "mappings":[
                    ":)=>_happy_",
                    ":(=>_sad_"
                ]
            }
        },
        "filter": {
            "english_stop":{
                "type":"stop",
                "stopwords":"_english_"
            }
        }
    }
}
```

测试：

```
POST my_analyzer/_analyze
{
  "analyzer": "my",
  "text":"I'm a :) person,and you?"
}
```

```
{
  "tokens": [
    {
      "token": "I'm a _happy_ person",
      "start_offset": 0,
      "end_offset": 15,
      "type": "word",
      "position": 0
    },
    {
      "token": "and you",
      "start_offset": 16,
      "end_offset": 23,
      "type": "word",
      "position": 1
    }
  ]
}
```

- 分词使用场景

  1、索引时分词：创建或更新文档时，会对相应得文档进行分词(指定字段分词)

```
PUT my_test
{
“mappings”:{
“doc”:{
“properties”:{
“title”:{
“type”:”text”,
“analyzer”:”whitespace”
}
}
}
}
}
```

2、查询时分词：查询时会对查询语句进行分词

```
POST my_test/_search
{
“query”:{
“match”:{
“message”:{
“query”:”hello”,
“analyzer”:”standard”
}
}
}
}
```

```
PUT my_test
{
“mappings”:{
“doc”:{
“properties”:{
“title”:{
“type”:”text”,
“analyzer”:”whitespace”,
“search_analyzer”:”standard”                    #查询指定分词器
}
}
}
}
}
```

一般不需要特别指定查询时分词器，直接使用索引时分词器即可，否则会出现无法匹配得情况，如果不需要分词将字段 type 设置成 keyword，可以节省空间

1.1.5、Mapping

- 作用：
  定义数据库中的表的结构的定义，通过 mapping 来控制索引存储数据的设置
  a. 定义 Index 下的字段名（Field Name）
  b. 定义字段的类型，比如数值型、字符串型、布尔型等
  c. 定义倒排索引相关的配置，比如 documentId、记录 position、打分等
- 获取索引 mapping
  不进行配置时，自动创建的 mapping
  请求：
  GET /atguigu/_mapping
  响应：

```
{
  "atguigu": {                                    #索引名称
    "mappings": {                                 #mapping 设置
      "student": {                                #type 名称
        "properties": {                           #字段属性
          "clazz": {
            "type": "text",                       #字段类型，字符串默认类型
            "fields": {                           #子字段属性设置
              "keyword": {                        #分词类型（不分词）
                "type": "keyword",
                "ignore_above": 256
              }
            }
          },
          "description": {
            "type": "text",
            "fields": {
              "keyword": {
                "type": "keyword",
                "ignore_above": 256
              }
            }
          },
          "name": {
            "type": "text",
            "fields": {
              "keyword": {
                "type": "keyword",
                "ignore_above": 256
              }
```

```
                }
            }
        }
    }
  }
}
```

- 自定义 mapping

  请求：

```
PUT my_index                                #索引名称
{
  "mappings":{
    "doc":{                                 #类型名称
      "dynamic":false,
      "properties":{
        "title":{
          "type":"text"                     #字段类型
        },
        "name":{
          "type":"keyword"
        },
        "age":{
          "type":"integer"
        }
      }
    }
  }
}
```

  响应：

```
{
  "acknowledged": true,
  "shards_acknowledged": true,
  "index": "my_index"
}
```

- Dynamic Mapping

  es 依靠 json 文档字段类型来实现自动识别字段类型，支持的类型

JSON 类型	es 类型
null	忽略
boolean	boolean
浮点类型	float
整数	long

object	object
array	由第一个非 null 值的类型决定
string	匹配为日期则设为 data 类型（默认开启） 匹配为数字的话设为 float 或 long 类型（默认关闭） 设为 text 类型，并附带 keyword 的子字段

- 注意：
  mapping 中的字段类型一旦设定后，禁止修改
  原因：Lucene 实现的倒排索引生成后不允许修改(提高效率)
  如果要修改字段的类型，需要从新建立索引，然后做 reindex 操作
- dynamic 设置
  a. true：允许自动新增字段（默认的配置）
  b. False：不允许自动新增字段，但是文档可以正常写入，无法对字段进行查询操作
  c. strict：文档不能写入（如果写入会报错）

- cope_to
  作用:将字段的值赋值到目标字段，实现类似_all 的作用

  例如：
  1、创建 mapping，包含 copy_to 字段

```
PUT my_index
{
    "mappings":{
        "doc":{
            "properties":{
                "frist_name":{
                    "type":"text",
                    "copy_to":"full_name"
                },
                "last_name":{
                    "type":"text",
                    "copy_to":"full_name"
                },
                "full_name":{
                    "type":"text"
                }
            }
        }
    }
}
```

  2、创建文档

```
PUT my_index/doc/1
{
  "frist_name":"John",
  "last_name":"Smith"
}
```

3、查询文档

```
GET my_index/_search
{
  "query":{
    "match": {
      "full_name":{
        "query":"John Smith",
        "operator":"and"
      }
    }
  }
}
```

- Index 属性

Index 属性，控制当前字段是否索引，默认为 true，即记录索引，false 不记录，即不可以搜索，比如：手机号、身份证号等敏感信息，不希望被检索

例如：

1、创建 mapping

```
PUT my_index
{
  "mappings": {
    "doc":{
      "properties": {
        "cookie":{
          "type":"text",
          "index": false
        }
      }
    }
  }
}
```

2、创建文档

```
PUT my_index/doc/1
{
  "cookie":"123",
```

```
    "name":"home"
  }
```

3、查询

```
GET my_index/_search
{
  "query": {
    "match": {
      "cookie":"123"
    }
  }
}
#报错
GET my_index/_search
{
  "query": {
    "match": {
      "name":"home"
    }
  }
}
#有结果
```

- index_option 用于记录倒排索引记录的内容，有一下 4 种配置
  1、docs 只记录 doc id
  2、freqs 记录 docid 和 term frequencies
  3、positions 记录 docid、term frequencies 和 term position
  4、offsets 记录 docid、term frequencies、term position 和 character offsets
  Text 类型默认配置为 positions，其他默认为 docs
  记录的内容越多暂用的空间越大

```
PUT my_index
{
“doc”:{
“properties”:{
“cookie”:{
“type”:”text”,
“index_option”:”offsets”
}
}
}
}
```

- null_value

  当字段遇到 null 值时，默认为 null，即空值，此时而社会忽略该值。可以通过设定该值设定字段的默认值

1.1.6、数据类型

- 核心数据类型

  字符串型：text、keyword

  数值型：long、integer、short、byte、double、float、half_float、scaled_float

  日期类型：date

  布尔类型：boolean

  二进制类型：binary

  范围类型：integer_range、float_range、long_range、double_range、date_range
- 复杂数据类型

  数组类型：array

  对象类型：object

  嵌套类型：nested object
- 地理位置数据类型

  geo_point(点)、geo_shape(形状)
- 专用类型

  记录 IP 地址 ip

  实现自动补全 completion

  记录分词数：token_count

  记录字符串 hash 值母乳 murmur3
- 多字段特性 multi-fields

  允许对同一个字段采用不同的配置，比如分词，例如对人名实现拼音搜索，只需要在人名中新增一个子字段为 pinyin 即可

1、创建 mapping

```
PUT my_index1
{
  "mappings": {
    "doc":{
      "properties":{
        "username":{
          "type": "text",
          "fields": {
            "pinyin":{
              "type": "text"
            }
          }
        }
      }
    }
  }
```

```
}
```

2、创建文档

```
PUT my_index1/doc/1
{
  "username":"haha heihei"
}
```

3、查询

```
GET my_index1/_search
{
  "query": {
    "match": {
      "username.pinyin": "haha"
    }
  }
}
```

- Dynamic Mapping

  es 可以自动识别文档字段类型，从而降低用户使用成本

```
PUT /test_index/doc/1
{
  "username":"alfred",
  "age":1
}
```

```
{
  "test_index": {
    "mappings": {
      "doc": {
        "properties": {
          "age": {
            "type": "long"
          },
          "username": {
            "type": "text",
            "fields": {
              "keyword": {
                "type": "keyword",
                "ignore_above": 256
              }
            }
          }
        }
```

```
        }
    }
  }
}
```

age 自动识别为 long 类型，username 识别为 text 类型

```
PUT test_index/doc/1
{
  "username":"samualz",
  "age":14,
  "birth":"1991-12-15",
  "year":18,
  "tags":["boy","fashion"],
  "money":"100.1"
}
```

```
{
  "test_index": {
    "mappings": {
      "doc": {
        "properties": {
          "age": {
            "type": "long"
          },
          "birth": {
            "type": "date"
          },
          "money": {
            "type": "text",
            "fields": {
              "keyword": {
                "type": "keyword",
                "ignore_above": 256
              }
            }
          },
          "tags": {
            "type": "text",
            "fields": {
              "keyword": {
                "type": "keyword",
                "ignore_above": 256
              }
            }
```

```
            },
            "username": {
              "type": "text",
              "fields": {
                "keyword": {
                  "type": "keyword",
                  "ignore_above": 256
                }
              }
            },
            "year": {
              "type": "long"
            }
          }
        }
      }
    }
```

日期的自动识别可以自行配置日期格式，以满足各种需求

1、自定义日期识别格式

```
PUT my_index
{
  "mappings":{
    "doc":{
      "dynamic_date_formats": ["yyyy-MM-dd","yyyy/MM/dd"]
    }
  }
}
```

2、关闭日期自动识别

```
PUT my_index
{
  "mappings": {
    "doc": {
      "date_detection": false
    }
  }
}
```

字符串是数字时，默认不会自动识别为整形，因为字符串中出现数字时完全合理的

Numeric_datection 可以开启字符串中数字的自动识别

```
PUT my_index
{
```

```
    "mappings":{
      "doc":{
        "numeric_datection": true
      }
    }
  }
```

- Dynamic Templates

  允许根据 es 自动识别的数据类型、字段名等来自动设定字段类型

  -所有字符串类型都设定为 keyword 类型，即默认不分词

  -所有以 message 开头的字段都设定为 text 类型，即分词

  -所有以 long_开头的字段都设定为 long 类型

  -所有自动匹配为 double 类型的都设定为 float 类型，以节省空间

  1、匹配规则

  match_mapping_type：匹配 es 自动识别的字段类型，如 boolean，long，string 等

  match,unmatch：匹配字段名

  path_match,path_unmatch：匹配对象内部字段

  2、例子

  把所有字符串类型的匹配成 keyword 类型

```
PUT test_index
{
  "mappings": {
    "doc": {
      "dynamic_templates":[                    #数组，可指定多个模板
        {
          "strings":{                          #模板名称
            "match_mapping_type":"string",     #匹配规则
            "mapping":{                        #设置 mapping 信息
              "type":"keyword"
            }
          }
        }
      ]
    }
  }
}
```

  以 message 开头的字段都设置成 text 类型

```
PUT test_index
{
  "mappings": {
    "doc":{
      "dynamic_templates":[
```

```
            {
              "message_as_text":{
                "match_mapping_type":"string",
                "match":"message*",
                "mapping":{
                  "type":"text"
                }
              }
            }
          ]
        }
      }
    }
```

double 类型设定为 float，节省空间

```
PUT test_index
{
  "mappings": {
    "doc":{
      "dynamic_templates":[
        {
          "message_as_text":{
            "match_mapping_type":"double",
            "mapping":{
              "type":"float"
            }
          }
        }
      ]
    }
  }
}
```

- 索引模板
  用于在新建索引时自动应用预先设定的配置，简化索引创建的操作步骤
  1、可以设定索引的配置和 mapping
  2、可以有多个模板，根据 order 设置，order 大的覆盖小的配置
  3、

- 自定义 mapping 步骤
  1、写入一条文档到 es 的临时索引中，获取 es 自动生成的 mapping
  2、修改步骤 1 得到的 mapping，自定义相关配置
  3、使用步骤 2 中的 mapping 创建实际所需的索引

```
PUT _template/test_template
{
  "index_patterns":["te*","bar*"],
  "order":0,                          #order 越大，优先级越高，覆盖 order 小的模板
  "settings": {
    "number_of_shards": 1
  },
  "mappings": {
    "doc":{
      "_source":{
        "enabled":false
      },
      "properties":{
        "name":{
          "type":"keyword"
        }
      }
    }
  }
}
```

有时索引创建出问题，首先查看模板

1.1.7、Search API(URI)

```
GET /_search                        #查询所有索引文档
GET /my_index/_search               #查询指定索引文档
GET /my_index1,my_index2/_search    #多索引查询
GET /my_*/_search
```

- URI 查询方式（查询有限制，很多配置不能实现）

```
GET /my_index/_search?q=user:alfred     #指定字段查询
GET /my_index/_search?q=alfred&df=user&sort=age:asc&from=4&size=10&timeout=1s
```

q：指定查询的语句，例如 q=aa 或 q=user:aa
df:q 中不指定字段默认查询的字段，如果不指定，es 会查询所有字段
Sort：排序，asc 升序，desc 降序
timeout：指定超时时间，默认不超时
from，size：用于分页

- term 与 phrase
  term 相当于单词查询，phrase 相当于词语查询
  term：Alfred way 等效于 alfred or way
  phrase：”Alfred way” 词语查询，要求先后顺序
- 泛查询

Alfred 等效于在所有字段去匹配该 term(不指定字段查询)

- 指定字段

  name:alfred

- Group 分组设定（），使用括号指定匹配的规则

  （quick OR brown）AND fox：通过括号指定匹配的优先级

  status:(active OR pending) title:(full text search)：把关键词当成一个整体

- 查询案例及详解

1、批量创建文档

```
POST test_search_index/doc/_bulk
{
  "index":{
    "_id":1
  }
}
{
  "username":"alfred way",
  "job":"java engineer",
  "age":18,
  "birth":"1991-12-15",
  "isMarried":false
}
{
  "index":{
    "_id":2
  }
}
{
  "username":"alfred",
  "job":"java senior engineer and java specialist",
  "age":28,
  "birth":"1980-05-07",
  "isMarried":true
}
{
  "index":{
    "_id":3
  }
}
{
  "username":"lee",
  "job":"java and ruby engineer",
  "age":22,
  "birth":"1985-08-07",
  "isMarried":false
```

```
}
{
  "index":{
    "_id":4
  }
}
{
  "username":"lee junior way",
  "job":"ruby engineer",
  "age":23,
  "birth":"1986-08-07",
  "isMarried":false
}
```

2、泛查询

```
GET test_search_index/_search?q=alfred
```

3、查询语句执行计划查看

```
GET test_search_index/_search?q=alfred
{
  "profile":true
}
```

4、term 查询

```
GET test_search_index/_search?q=username:alfred way          #alfred OR way
```

5、phrase 查询

```
GET test_search_index/_search?q=username:"alfred way"
```

6、group 查询

```
GET test_search_index/_search?q=username:(alfred way)
```

7、布尔操作符

（1）AND(&&),OR(||),NOT(!)

例如：name:(tom NOT lee)

#表示 name 字段中可以包含 tom 但一定不包含 lee

（2）+、-分别对应 must 和 must_not

例如：name:(tom +lee -alfred)

#表示 name 字段中，一定包含 lee，一定不包含 alfred，可以包含 tom

注意：+在 url 中会被解析成空格，要使用 encode 后的结果才可以，为%2B

```
GET test_search_index/_search?q=username:(alfred %2Bway)
```

- 范围查询，支持数值和日期

1、区间：闭区间：[]，开区间:{}

age:[1 TO 10] #1<=age<=10

age:[1 TO 10} #1<=age<10

age:[1 TO] #1<=age

age:[* TO 10] #age<=10

2、算术符号写法

age:>=1

age:(>=1&&<=10)或者 age:(+>=1 +<=10)

- 通配符查询

?:1 个字符

*:0 或多个字符

例如：name:t?m

name:tom*

name:t*m

注意：通配符匹配执行效率低，且占用较多内存，不建议使用，如无特殊要求，不要讲?/*放在最前面

- 正则表达式

name:/[mb]oat/

- 模糊匹配 fuzzy query

name:roam~1

匹配与 roam 差 1 个 character 的词，比如 foam、roams 等

- 近似度查询 proximity search

“fox quick”~5

以 term 为单位进行差异比较，比如”quick fox” “quick brown fox”

1.1.8、Search API(Request Body Search)

- Match Query

对字段作全文检索，最基本和常用的查询类型

```
GET test_search_index/_search
{
  "profile":true,                # 显示执行计划
  "query":{
    "match": {
      "username": "alfred way"
    }
  }
}
```

通过 operator 参数可以控制单词间的匹配关系，可选项为 or 和 and

```
GET test_search_index/_search
{
```

```
    "query":{
      "match": {
        "username": {
          "query":"alfred way",
          "operator":"and"
        }
      }
    }
}
```

通过 minimum_should_match 参数可以控制需要匹配的单词数

如下例子，匹配 alfred way 分词后，其中的一个词即可

```
GET test_search_index/_search
{
  "query": {
    "match": {
      "username": {
        "query": "alfred way",
        "minimum_should_match":1
      }
    }
  }
}
```

1.1.9、相关性算分

相关性算分是指文档与查询语句间的相关度，英文为 relevance

本质就是搜索结果返回文档的排序问题

Term Frequency(TF)	词频	单词在该文档中出现的次数。词频越高，相关度越高
Document Frequency(DF)	文档频率	单词出现的文档数
Inverse Document Frequency(IDF)	逆向文档频率	
Field-length Norm		

ELK 课件

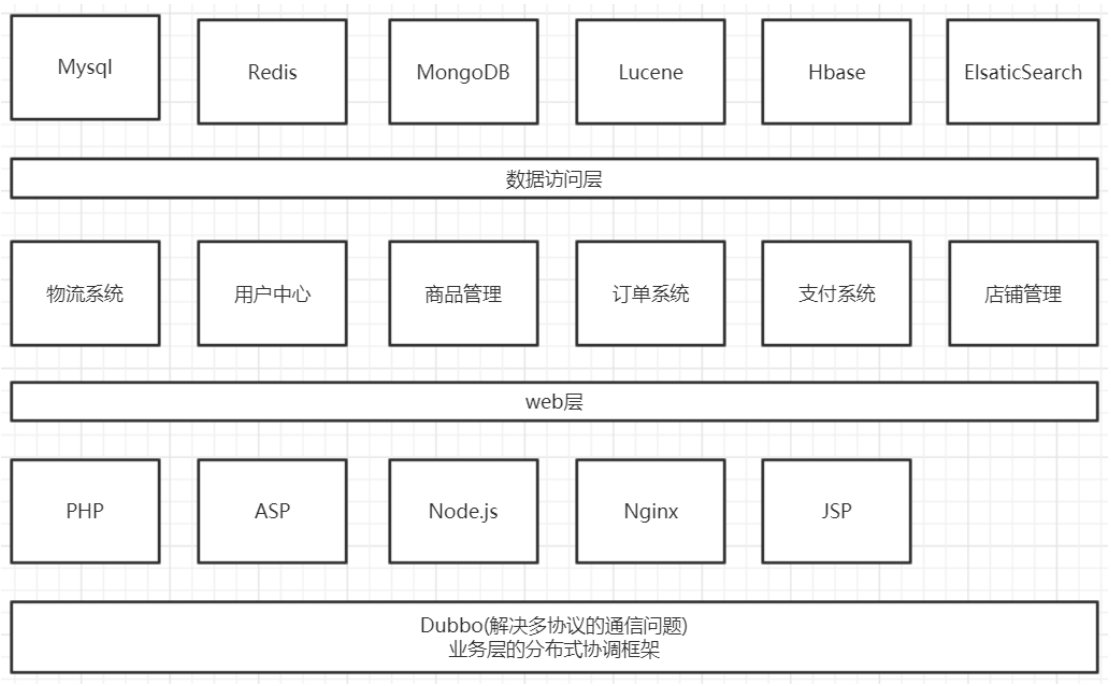
1、ELK 简介

1.1、ELK 是什么

- Elasticsearch 是个开源分布式搜索引擎，它的特点有：分布式，零配置，自动发现，索引自动分片，索引副本机制，restful 风格接口，多数据源，自动搜索负载等。
- Logstash 是一个完全开源的工具，他可以对你的日志进行收集、过滤，并将其存储供以后使用（如，搜索）。
- Kibana 也是一个开源和免费的工具，它 Kibana 可以为 Logstash 和 ElasticSearch 提供的日志分析友好的 Web 界面，可以帮助您汇总、分析和搜索重要数据日志。

1.2、ELK 应用

- 电商体系架构

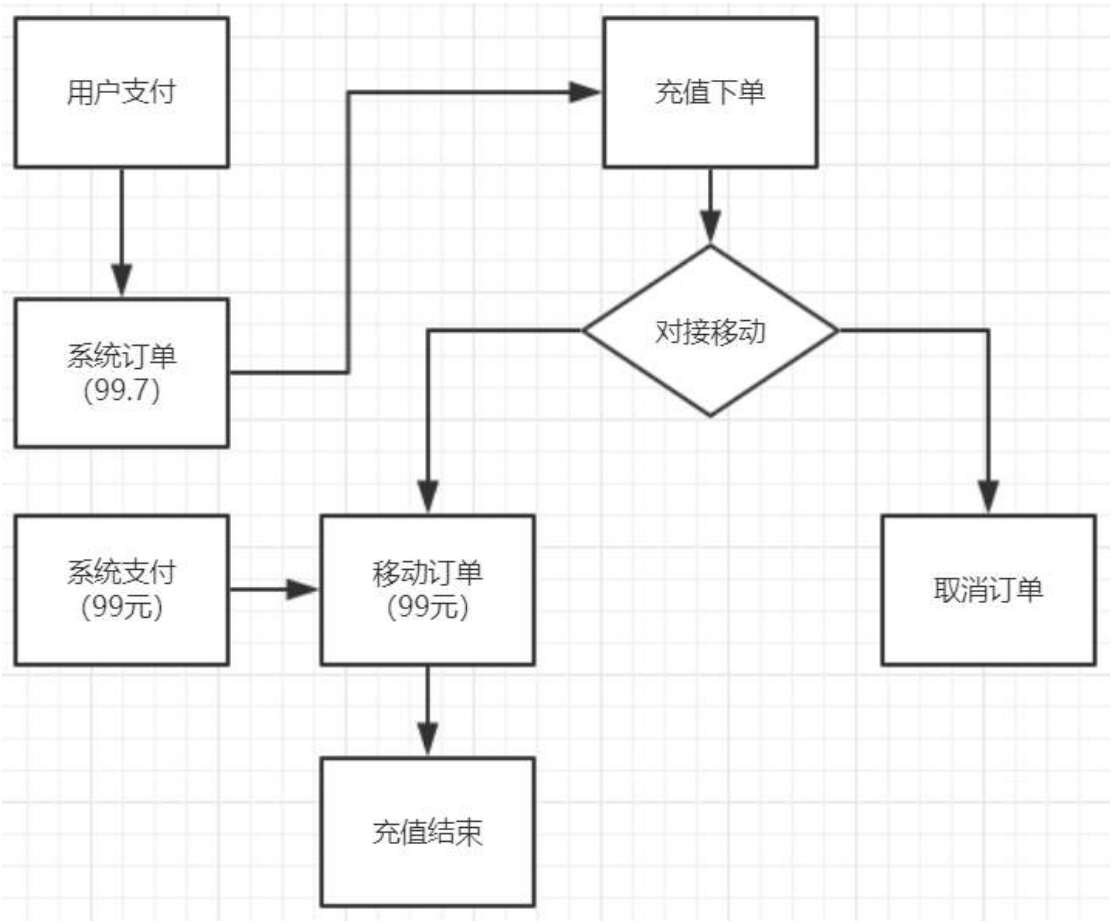


- 问题

  1、API 不一样，我们如何去整合？--》dubbo 定义统一的 api 规范
  2、各子系统之间会产生操作痕迹（用户行为轨迹）---》日志
  3、各个子系统都会生成各自的日志---日志整合--》logstash
  4、AOP 埋点，异步日志输出

- 具体场景 1

  通过第三方进行移动话费充值



日志输出：每次调用都会打印异步日志

分布式负载均衡：

很多太机器都可以充值（动态的去选择一台目前比较空闲的机器去执行这个任务）

问题：

A：兄弟，帮忙查一下今天手机号码 138001380000 充值日志记录（是否成功）

B：稍等

5 分钟后

A：怎么样了

B：稍等，还剩下 3 台机器没查完

结论：如果能把所有的日志整理在一起，就不会出现一台一台去查的问题

解决方案：

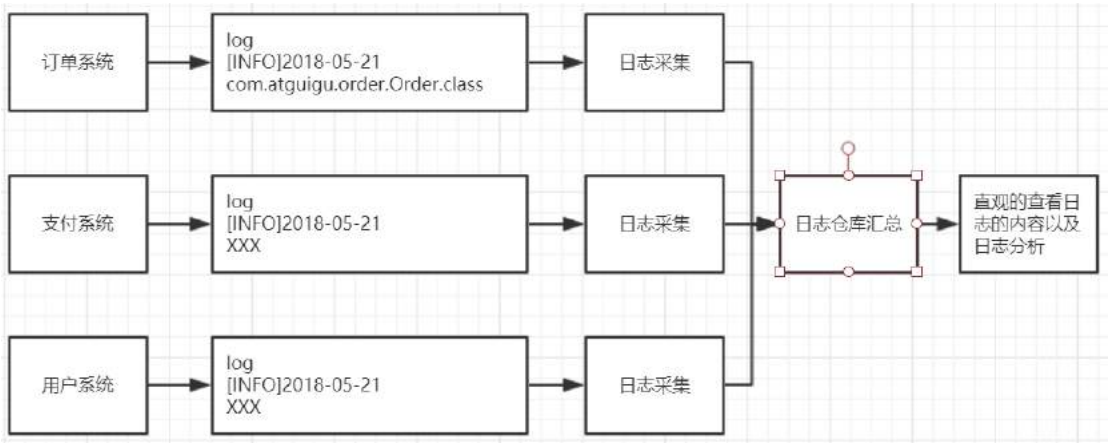
1、可不可以把日志放在数据库中。

数据量太大，且日志没有规范日志格式，数据库方案不太建议，且压力过大

2、采用大数据日志处理方案

成本太高，且分布式环境每个系统的日志规则不一样。

- 具体业务实践



日志收集：Logstash
日志存储：ElasticSearch
日志展示：Kibana
针对对台服务器日志不统一的问题，提供多种检索规则，方便可视化展示

- 案例总结

  分布式带来的问题：多节点、负载均衡、日志分散、运维成本高（需要人为跟踪）

1.3、集中式日志管理系统

当前主流的一些集中日志管理系统
1、简单的：Rsyslog
2、商业化：Splunk
3、开源的：Scribe（FaceBook），Chukwa（Apache）
4、ELK 最广泛的（Elastic Stack）(java 语言编写)
www.elastic.co/cn

1.4、ELK

ElasticSearch	Java	实时的分布式搜索和分析引擎，他可以用于全文检索，结构化搜索以及分析，lucene。Solr
Logstash	JRuby	具有实时渠道能力的数据收集引擎，包含输入、过滤、输出模块，一般在过滤模块中做日志格式化的解析工作
Kibana	JavaScript	为 ElasticSerach 提供分析平台和可视化的 Web 平台。他可以 ElasticSerach 的索引中查找，呼唤数据，并生成各种维度的表图

1.5、日志

日志：记录程序的运行轨迹---

级别：ERROR、INFO、DEBUG、WARN

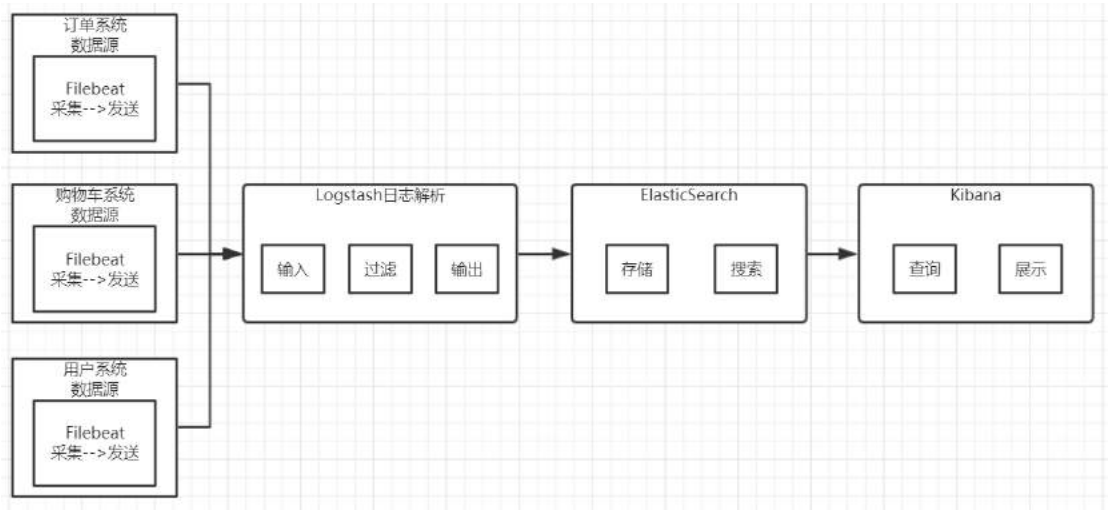
目的：方便定位和查找信息，记录除去业务外的附加的信息，链路

Filebeat 简介

当您要面对成百上千、甚至成千上万的服务器、虚拟机和容器生成的日志时，请告别 SSH 吧。Filebeat 将为您提供一种轻量型方法，用于转发和汇总日志与文件，让简单的事情不再繁杂。

当将数据发送到 Logstash 或 Elasticsearch 时，Filebeat 使用背压敏感协议，以考虑更多的数据量。如果 Logstash 正在忙于处理数据，则可以让 Filebeat 知道减慢读取速度。一旦拥堵得到解决，Filebeat 就会恢复到原来的步伐并继续运行。

无论在任何环境中，随时都潜伏着应用程序中断的风险。Filebeat 能够读取并转发日志行，如果出现中断，还会在一切恢复正常后，从中断前停止的位置继续开始。



2、准备工作

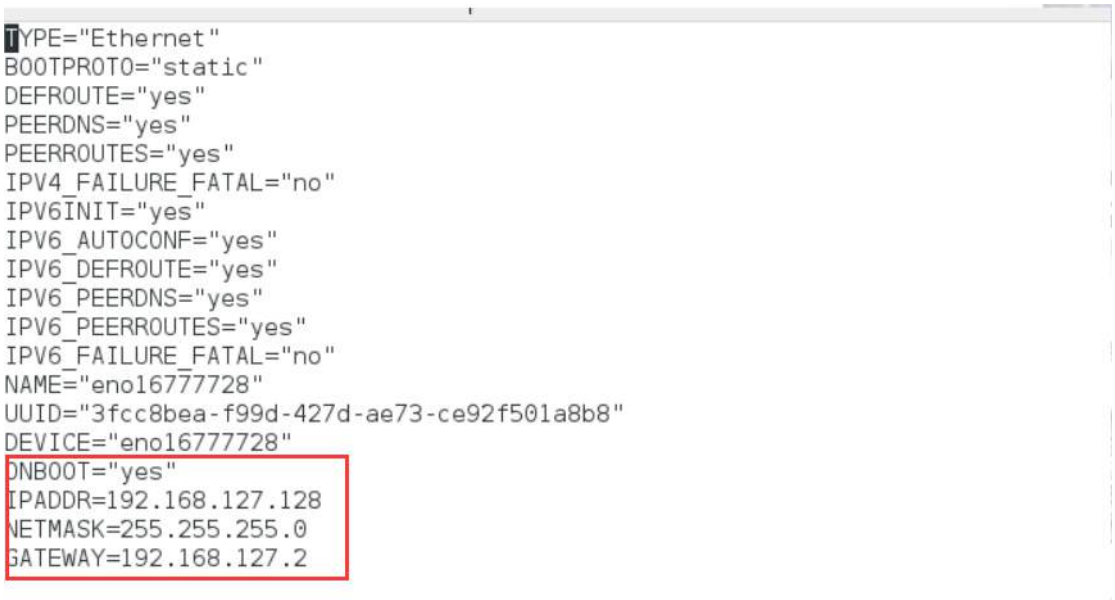
2.1、安装 Centos7

建议内存 2G 以上

2.2、基本配置

- 设置 IP 地址

vi /etc/sysconfig/network-scripts/ifcfg-eno33



service network restart

- 添加用户并授权

[root@localhost ~]# adduser elk1

[root@localhost ~]# passwd elk1

[root@localhost ~]# whereis sudoers

[root@localhost ~]# ls -l /etc/sudoers

[root@localhost ~]# chmod -v u+w /etc/sudoers

[root@localhost ~]# vi /etc/sudoers

```
## Allow root to run any commands anywher
root        ALL=(ALL)             ALL
linuxidc    ALL=(ALL)              ALL    #这个是新增的用户
```

[root@localhost ~]# chmod -v u-w /etc/sudoers

[root@localhost ~]# su elk1

3、ElasticSerach

3.1、Java 环境安装

- 解压安装包

[root@localhost jdk1.8]# tar -zxvf jdk-8u171-linux-x64.tar.gz

- 设置 Java 环境变量

[root@localhost jdk1.8.0_171]# vi /etc/profile

在文件最后添加

```
export JAVA_HOME=/home/elk1/jdk1.8/jdk1.8.0_171
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=.:$JAVA_HOME/LIB:$JRE_HOME/LIB:$CLASSPATH
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
```

[root@localhost jdk1.8.0_171]# source /etc/profile

[root@localhost jdk1.8.0_171]# java -version

java version "1.8.0_171"

Java(TM) SE Runtime Environment (build 1.8.0_171-b11)

Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)

3.2、ElasticSerach 单机安装

[root@localhost elasticserach]# tar -zxvf elasticsearch-6.3.1.tar.gz

[root@localhost elasticserach]# cd elasticsearch-6.3.1/bin

[root@localhost bin]# ./elasticsearch

```
[root@localhost bin]# ./elasticsearch
[2018-07-13T15:22:41,083][WARN ][o.e.b.ElasticsearchUncaughtExceptionHandler] [] uncaught exception in thread [main]
org.elasticsearch.bootstrap.StartupException: java.lang.RuntimeException: can not run elasticsearch as root
        at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:140) ~[elasticsearch-6.3.1.jar:6.3.1]
        at org.elasticsearch.bootstrap.Elasticsearch.execute(Elasticsearch.java:127) ~[elasticsearch-6.3.1.jar:6.3.1]
        at org.elasticsearch.cli.EnvironmentAwareCommand.execute(EnvironmentAwareCommand.java:86) ~[elasticsearch-6.3.1.jar:6.3.1]
        at org.elasticsearch.cli.Command.mainWithoutErrorHandling(Command.java:124) ~[elasticsearch-cli-6.3.1.jar:6.3.1]
        at org.elasticsearch.cli.Command.main(Command.java:90) ~[elasticsearch-cli-6.3.1.jar:6.3.1]
        at org.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:93) ~[elasticsearch-6.3.1.jar:6.3.1]
        at org.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:86) ~[elasticsearch-6.3.1.jar:6.3.1]
Caused by: java.lang.RuntimeException: can not run elasticsearch as root
        at org.elasticsearch.bootstrap.Bootstrap.initializeNatives(Bootstrap.java:104) ~[elasticsearch-6.3.1.jar:6.3.1]
        at org.elasticsearch.bootstrap.Bootstrap.setup(Bootstrap.java:171) ~[elasticsearch-6.3.1.jar:6.3.1]
        at org.elasticsearch.bootstrap.Bootstrap.init(Bootstrap.java:326) ~[elasticsearch-6.3.1.jar:6.3.1]
        at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:136) ~[elasticsearch-6.3.1.jar:6.3.1]
        ... 6 more
```

[root@localhost bin]# su elk1

[elk1@localhost bin]$ ./elasticsearch

```
[elk1@localhost bin]$ ./elasticsearch
Exception in thread "main" java.nio.file.AccessDeniedException: /home/elk1/elasticserach/elasticsearch-6.3.1/config/jvm.options
        at sun.nio.fs.UnixException.translateToIOException(UnixException.java:84)
        at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:102)
        at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:107)
        at sun.nio.fs.UnixFileSystemProvider.newByteChannel(UnixFileSystemProvider.java:214)
        at java.nio.file.Files.newByteChannel(Files.java:361)
        at java.nio.file.Files.newByteChannel(Files.java:407)
        at java.nio.file.spi.FileSystemProvider.newInputStream(FileSystemProvider.java:384)
        at java.nio.file.Files.newInputStream(Files.java:152)
        at org.elasticsearch.tools.launchers.JvmOptionsParser.main(JvmOptionsParser.java:58)
```

[root@localhost bin]# chown -R elk1:elk1 /home/elk1/elasticsearch

[elk1@localhost bin]$ ./elasticsearch

[elk1@localhost config]$ vi jvm.options

```
## See https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html
## for more information
##
################################################################

# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms2g
-Xmx2g
```

[elk1@localhost bin]$ ./elasticsearch

```
[2018-07-13T16:05:00,979][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [tribe]
[2018-07-13T16:05:00,979][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-core]
[2018-07-13T16:05:00,979][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-deprecation]
[2018-07-13T16:05:00,979][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-graph]
[2018-07-13T16:05:00,979][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-logstash]
[2018-07-13T16:05:00,980][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-ml]
[2018-07-13T16:05:00,980][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-monitoring]
[2018-07-13T16:05:00,980][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-rollup]
[2018-07-13T16:05:00,980][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-security]
[2018-07-13T16:05:00,980][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-sql]
[2018-07-13T16:05:00,981][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-upgrade]
[2018-07-13T16:05:00,981][INFO ][o.e.p.PluginsService     ] [_uHU_cC] loaded module [x-pack-watcher]
[2018-07-13T16:05:00,981][INFO ][o.e.p.PluginsService     ] [_uHU_cC] no plugins loaded
[2018-07-13T16:05:13,853][INFO ][o.e.x.s.a.s.FileRolesStore] [_uHU_cC] parsed [0] roles from file [/home/elk1/elasticserach/elasticsearch-6.3.1/config/roles.yml]
[2018-07-13T16:05:15,570][INFO ][o.e.x.m.j.p.l.CppLogMessageHandler] [controller/10016] [Main.cc@109] controller (64 bit): Version 6.3.1 (Build 4d0b8f0a0ef401) Co
[2018-07-13T16:05:17,231][DEBUG][o.e.a.ActionModule       ] Using REST wrapper from plugin org.elasticsearch.xpack.security.Security
[2018-07-13T16:05:17,756][INFO ][o.e.d.DiscoveryModule    ] [_uHU_cC] using discovery type [zen]
[2018-07-13T16:05:19,456][INFO ][o.e.n.Node               ] [_uHU_cC] initializing ...
[2018-07-13T16:05:19,456][INFO ][o.e.n.Node               ] [_uHU_cC] starting ...
[2018-07-13T16:05:20,082][INFO ][o.e.t.TransportService   ] [_uHU_cC] publish_address {127.0.0.1:9300}, bound_addresses {[::1]:9300}, {127.0.0.1:9300}
[2018-07-13T16:05:20,234][WARN ][o.e.b.BootstrapChecks    ] [_uHU_cC] max file descriptors [4096] for elasticsearch process is too low, increase to at least [655
[2018-07-13T16:05:20,234][WARN ][o.e.b.BootstrapChecks    ] [_uHU_cC] max number of threads [3818] for user [elk1] is too low, increase to at least [4096]
[2018-07-13T16:05:20,234][WARN ][o.e.b.BootstrapChecks    ] [_uHU_cC] max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]
```

[root@localhost jdk1.8.0_171]# curl 127.0.0.1:9200

```
[root@localhost jdk1.8.0_171]# curl 127.0.0.1:9200
{
  "name" : "_uHU_cC",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "mqFXQFsuSrKQpYtW8wWJYw",
  "version" : {
    "number" : "6.3.1",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "eb782d0",
    "build_date" : "2018-06-29T21:59:26.107521Z",
    "build_snapshot" : false,
    "lucene_version" : "7.3.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

#后台启动

[elk1@localhost bin]$ ./elasticsearch -d

#关闭程序

[elk1@localhost bin]$ ps -ef|grep elastic

```
[elk1@localhost bin]$ ps -ef|grep elastic
elk1      10097      1  8 16:07 pts/0    00:00:34 /home/elk1/jdk1.8/jdk1.8.0_171/bin/java -Xms2g -Xmx2g -XX:+UseConcMarkSwee
aysPreTouch -Xss1m -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djna.nosys=true -XX:-OmitStackTraceInFastThrow -Dio.netty
rThread=0 -Dlog4j.shutdownHookEnabled=false -Dlog4j2.disable.jmx=true -Djava.io.tmpdir=/tmp/elasticsearch.FJ7pocrL -XX:+Heap
+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintTenuringDistribution -XX:+PrintGCApplicationStoppedTime -Xloggc:logs/gc.log
home=/home/elk1/elasticserach/elasticsearch-6.3.1 -Des.path.conf=/home/elk1/elasticserach/elasticsearch-6.3.1/config -Des.di
sticsearch-6.3.1/lib/* org.elasticsearch.bootstrap.Elasticsearch -d
elk1      10348   2340  0 16:14 pts/0    00:00:00 grep --color=auto elastic
```

[elk1@localhost bin]$ kill 10097

#设置浏览器访问

[root@localhost bin]systemctl stop firewalld

[root@localhost bin]vi config/elasticsearch.yml

```
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- Network -----------------------------------
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
network.host: 192.168.14.13
#
# Set a custom port for HTTP:
#
#http.port: 9200
```

安装问题：

```
ERROR: [3] bootstrap checks failed
[1]: max file descriptors [4096] for elasticsearch process is too low, increase to at least [65536]
[2]: max number of threads [3818] for user [elk1] is too low, increase to at least [4096]
[3]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]
[2018-07-13T16:24:42,964][INFO ][o.e.n.Node               ] [_uHU_cC] stopping ...
[2018-07-13T16:24:43,183][INFO ][o.e.n.Node               ] [_uHU_cC] stopped
[2018-07-13T16:24:43,183][INFO ][o.e.n.Node               ] [_uHU_cC] closing ...
[2018-07-13T16:24:43,228][INFO ][o.e.n.Node               ] [_uHU_cC] closed
[2018-07-13T16:24:43,252][INFO ][o.e.x.m.j.p.NativeController] Native controller process has stopped -
```

[1] [2]解决方案

[root@localhost bin]# vi /etc/security/limits.conf

```
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#@student        -       maxlogins       4

* hard nofile 65536
* soft nofile 131072
* hard nproc 4096
* soft nproc 2048
End of file
```

*代表所有用户

[3] 解决方案

[root@localhost bin]# vi /etc/sysctl.conf
[root@localhost bin]# sysctl -p

```
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
vm.max_map_count=655360
fs.file-max=655360
```

3.3、ElasticSerach 集群安装

- 修改配置文件 elasticserach.yml
  vim /elasticsearch.yml

```
cluster.name: aubin-cluster#必须相同
# 集群名称（不能重复）
node.name: els1（必须不同）
# 节点名称，仅仅是描述名称，用于在日志中区分（自定义）
#指定了该节点可能成为 master 节点，还可以是数据节点
node.master: true
node.data: true
path.data: /var/lib/elasticsearch
# 数据的默认存放路径（自定义）
path.logs: /var/log/elasticsearch
# 日志的默认存放路径
network.host: 192.168.0.1
# 当前节点的 IP 地址
http.port: 9200
# 对外提供服务的端口
transport.tcp.port: 9300
#9300 为集群服务的端口
discovery.zen.ping.unicast.hosts: ["172.18.68.11", "172.18.68.12","172.18.68.13"]
# 集群个节点 IP 地址，也可以使用域名，需要各节点能够解析
discovery.zen.minimum_master_nodes: 2
# 为了避免脑裂，集群节点数最少为 半数+1
```

注意：清空 data 和 logs 数据

192.168.14.12:9200/_cat/nodes?v

3.4、安装 head 插件

- 下载 head 插件

  wget https://github.com/mobz/elasticsearch-head/archive/elasticsearch-head-master.zip

  也可以用 git 下载，前提 yum install git

  unzip elasticsearch-head-master.zip
- 安装 node.js

  wget https://npm.taobao.org/mirrors/node/latest-v4.x/node-v4.4.7-linux-x64.tar.gz

  tar -zxvf node-v9.9.0-linux-x64.tar.gz
- 添加 node.js 到环境变量

```
export JAVA_HOME=/home/elk1/jdk1.8/jdk1.8.0_171
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=.:$JAVA_HOME/LIB:$JRE_HOME/LIB:$CLASSPATH
export NODE_HOME=/home/elk1/elasticserach/head/node-v9.9.0-linux-x64
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$NODE_HOME/bin:$PATH
```

  source /etc/profile
- 测试

  node -v

  npm -v
- 安装 grunt（grunt 是一个很方便的构建工具，可以进行打包压缩、测试、执行等等的工作）

  进入到 elasticsearch-head-master

  npm install -g grunt-cli

  npm install

  (npm install -g cnpm --registry=https://registry.npm.taobao.org)

- 修改 Elasticsearch 配置文件

  编辑 elasticsearch-6.3.1/config/elasticsearch.yml,加入以下内容：

  http.cors.enabled: true

  http.cors.allow-origin: "*"

- 修改 Gruntfile.js（注意’，’）

打开 elasticsearch-head-master/Gruntfile.js，找到下面 connect 属性，新增 hostname:’*’:

```
connect: {
        server: {
            options: {
                hostname: '*',
                port: 9100,
```

```
                base: '.',
                keepalive: true
            }
        }
    }
```

- 启动 elasticsearch-head
  进入 elasticsearch-head 目录，执行命令：grunt server
- 后台启动 elasticsearch-head
  nohup grunt server &exit
- 关闭 head 插件
  ps -aux|grep head
  kill 进程号

3.5、ElasticSerach API

- elasticsearch rest api 遵循的格式为：
  curl -X<REST Verb> <Node>:<Port>/<Index>/<Type>/<ID>
- 检查 es 版本信息
  curl IP:9200
- 查看集群是否健康
  http://IP:9200/_cat/health?v
- 查看节点列表
  http://IP:9200/_cat/nodes?v
- 列出所有索引及存储大小
  http://IP:9200/_cat/indices?v
- 创建索引
  curl -XPUT 'IP:9200/XX?pretty'
- 添加一个类型
  curl -XPUT 'IP:9200/XX/external/2?pretty' -d '
  {
    "gwyy": "John"
  }'
- 更新一个类型
  curl -XPOST 'IP:9200/XX/external/1/_update?pretty' -d '
  {
    "doc": {"name": "Jaf"}
  }'
- 删除指定索引
  curl -XDELETE 'IP:9200/_index?pretty'

3.6、配置详情

- ElasticSearch.yml
  ES 的相关配置

```
# 集群的名字，以此作为是否同一集群的判断条件
cluster.name: elasticsearch
# 节点名字，以此作为集群中不同节点的区分条件
node.name: node-1
#设置当前节点既可以为主节点也可以为数据节点
node.master: true
node.data: true
# 索引分片个数，默认为 5 片
#index.number_of_shards: 5
# 索引副本个数，默认为 1 个副本
#index.number_of_replicas: 1
# 数据存储目录（多个路径用逗号分隔）
discovery.zen.ping.unicast.hosts: ["192.168.14.14","192.168.14.15"]
discovery.zen.minimum_master_nodes: 2
#数据目录
path.data: /home/elk1/elasticserach/data
# 日志目录
path.logs: /home/elk1/elasticserach/logs
# 修改一下 ES 的监听地址，这样别的机器才可以访问
network.host: 192.168.14.13
# 设置节点间交互的 tcp 端口（集群）,默认是 9300
transport.tcp.port: 9300
# 监听端口（默认的就好）
http.port: 9200
# 增加新的参数，这样 head 插件才可以访问 es
http.cors.enabled: true
http.cors.allow-origin: "*"
```

- Jvm.options
  JVM 的相关配置

- Log4j2.properties
  日志相关配置

3.7、Elasticserach 模式

- 分为 Development 和 Production 两种模式
  - 区分方式

以 transport 的地址是否绑定在 localhost 为标准（实际地址）
即：elasticserach.yml 文件中的 network.host 配置

- 模式区别
  （1）Development 模式下启动时会以 warning 的方式提示配置检查异常
  （2）Production 模式下在启动时会以 error 的方式提示配置检查异常并推出

3.8、elasticserach 操作

- 基本概念
  - Document:文档对象
  - Index:索引（库）
  - Type:索引中的数据类型（表）
  - Field:字段，文档的属性（字段）
  - Query DSL:查询语法（sql）
- CRUD 操作
  - 创建文档
    请求：

```
POST /atguigu/student/1
{
    "name":"zhangsan",
    "clazz":"0115bigdata",
    "description":"we are family"
}
```

    返回：

```
{
  "_index": "atguigu",
  "_type": "student",
  "_id": "1",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 2,
    "failed": 0
  },
  "_seq_no": 0,
  "_primary_term": 1
}
```

  - 获取文档
    请求：

```
GET atguigu/student/1
```

    返回：

```
{
  "_index": "atguigu",
  "_type": "student",
  "_id": "1",
  "_version": 1,
  "found": true,
  "_source": {
    "name": "zhangsan",
    "clazz": "0115bigdata",
    "description": "we are family"
  }
}
```

- 更新文档

请求：

```
POST /atguigu/student/1/_update
{
    "doc":{
        "description":"hello world"
    }
}
```

返回：

```
{
  "_index": "atguigu",
  "_type": "student",
  "_id": "1",
  "_version": 2,
  "result": "updated",
  "_shards": {
    "total": 2,
    "successful": 2,
    "failed": 0
  },
  "_seq_no": 1,
  "_primary_term": 1
}
```

- 删除文档

请求：

```
DELETE atguigu/student/1
```

查询结果：

```
{
  "_index": "atguigu",
  "_type": "student",
  "_id": "1",
```

```
    "found": false
}
```

- Elasticserach Query
  - Query String

    GET /atguigu/student/_sea'rch?q=关键字

    返回：

```
{
    "took": 8,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "max_score": 0.2876821,
        "hits": [
            {
                "_index": "atguigu",
                "_type": "student",
                "_id": "1",
                "_score": 0.2876821,
                "_source": {
                    "name": "zhangsan",
                    "clazz": "0115bigdata",
                    "description": "we are family"
                }
            }
        ]
    }
}
```

  - Query DSL

```
GET atguigu/student/_search
{
    "query":{
        "term":{
            "name":{
                    "value":"zhangsan"
```

```
        }
      }
    }
  }
```

4、Logstash

4.1、安装 logstash

[root@localhost logstash]# tar -zxvf logstash-6.3.1.tar.gz

[root@localhost logstash-6.3.1]# cd config

[root@localhost config]# vi log4j_to_es.conf

```
# For detail structure of this file
# Set: https://www.elastic.co/guide/en/logstash/current/configuration-file-structure.html
input {
  # For detail config for log4j as input,
  # See: https://www.elastic.co/guide/en/logstash/current/plugins-inputs-log4j.html
  log4j {
    mode => "server"
    host => "centos2"
    port => 4567
  }
}
filter {
  #Only matched data are send to output.
}
output {
  # For detail config for elasticsearch as output,
  # See: https://www.elastic.co/guide/en/logstash/current/plugins-outputs-elasticsearch.html
  elasticsearch {
    action => "index"          #The operation on ES
    hosts  => "centos2:9200"   #ElasticSearch host, can be array.
    index  => "applog"         #The index to write data to.
  }
}
```

```
input {
        file {
                path=>[""]
                type=>""
                start_position=>"beginning"
        }
}
output {
        stdout {
                codec=>rubydebug
        }
}
```

[root@localhost logstash-6.3.1]# ./bin/logstash -f config/log4j_to_es.conf

4.2、输入、输出、过滤

- 输入
  input{file{path=>”/tomcat/logs/abc.log”}}
- 输出
  output{stdout{codec=>rubydebug}}
- 过滤插件
  - Grok
    1、基于正则表达式提供了丰富可重用的模式（pattern）
    2、基于此可以将非结构化数据作结构化处理
  - Date
    将字符串类型的时间字段转换为时间戳类型，方便后续数据处理
  - Mutate
    进行增加、修改、删除、替换等字段相关处理

4.3、logstash 格式化 nginx 日志内容

- 创建 nginx_logstash.conf 文件

```
input {
   stdin { }
}

filter {
   grok {
      match => {
        "message" => '%{IPORHOST:remote_ip} - %{DATA:user_name} \[%{HTTPDATE:time}\] "%{WORD:request_action} %{DATA:request} HTTP/%{NUMBER:http_version}" %{NUMBER:response} %{NUMBER:bytes} "%{DATA:referrer}" "%{DATA:agent}"'
      }
   }

   date {
      match => [ "time", "dd/MMM/YYYY:HH:mm:ss Z" ]
      locale => en
   }

   geoip {
      source => "remote_ip"
      target => "geoip"
   }
```

```
  useragent {
     source => "agent"
     target => "user_agent"
  }
}

output {
stdout {
 codec => rubydebug
 }
}
```

- Logstash 启动解析 nginx 文件

head -n 2 /home/elk1/nginx_logs|./logstash -f ../config/nginx_logstash.conf

- 结果

```
{
              "user_name" => "-",
               "referrer" => "-",
             "@timestamp" => 2015-05-17T08:05:32.000Z,
                "request" => "/downloads/product_1",
                   "time" => "17/May/2015:08:05:32 +0000",
                  "geoip" => {
              "country_code3" => "NL",
                    "longitude" => 4.8995,
             "continent_code" => "EU",
                    "latitude" => 52.3824,
                    "timezone" => "Europe/Amsterdam",
              "country_code2" => "NL",
                         "ip" => "93.180.71.3",
               "country_name" => "Netherlands",
                    "location" => {
                 "lat" => 52.3824,
                 "lon" => 4.8995
             }
      },
               "@version" => "1",
          "http_version" => "1.1",
              "remote_ip" => "93.180.71.3",
                "message" => "93.180.71.3 - - [17/May/2015:08:05:32 +0000] \"GET /downloads/product_1 HTTP/1.1\" 304 0 \"-\" \"Debian APT-HTTP/1.3 (0.8.16~exp12ubuntu10.21)\"",
                  "bytes" => "0",
             "user_agent" => {
               "minor" => "3",
```

```
                    "os" => "Debian",
                  "name" => "Debian APT-HTTP",
             "os_name" => "Debian",
                 "build" => "",
                 "major" => "1",
                "device" => "Other"
       },
                    "agent" => "Debian APT-HTTP/1.3 (0.8.16~exp12ubuntu10.21)",
                     "host" => "localhost.localdomain",
                "response" => "304",
       "request_action" => "GET"
}
```

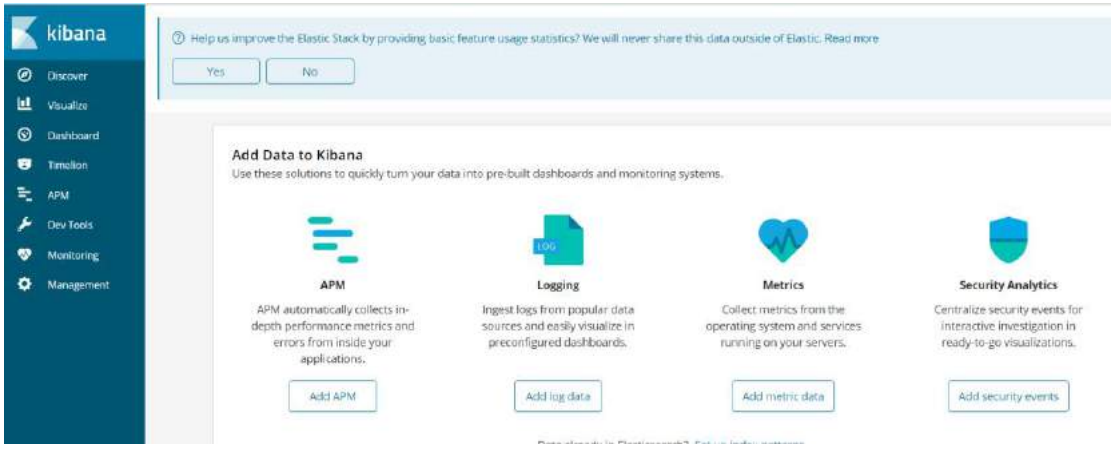
5、Kibana

5.1、Kibana 安装

[root@localhost kibana]# tar -zxvf kibana-6.3.1-linux-x86_64.tar.gz
[root@localhost kibana]# cd kibana-6.3.1-linux-x86_64/config
[root@localhost config]# vi kibana.yml

```
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and hos
# The default is 'localhost', which usually means remote machines will not be able
# To allow connections from remote users, set this parameter to a non-loopback add
server.host: "192.168.14.15"

# Enables you to specify a path to mount Kibana at if you are running behind a pro
# Use the `server.rewriteBasePath` setting to tell Kibana if it should remove the
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#server.basePath: ""

# Specifies whether Kibana should rewrite requests that are prefixed with
# `server.basePath` or require that they are rewritten by your reverse proxy.
# This setting was effectively always `false` before Kibana 6.3 and will
# default to `true` starting in Kibana 7.0.
#server.rewriteBasePath: false

# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name.  This is used for display purposes.
#server.name: "your-hostname"

# The URL of the Elasticsearch instance to use for all your queries.
elasticsearch.url: "http://192.168.14.14:9200"

# When this setting's value is true Kibana uses the hostname specified in the serv
# setting. When the value of this setting is false, Kibana uses the hostname of th
# that connects to this Kibana instance.
#elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved searches, visualizations an
# dashboards. Kibana creates a new index if the index doesn't already exist.
kibana.index: ".kibana"
```

[root@localhost bin]# ./kibana



5.2、kibana 配置

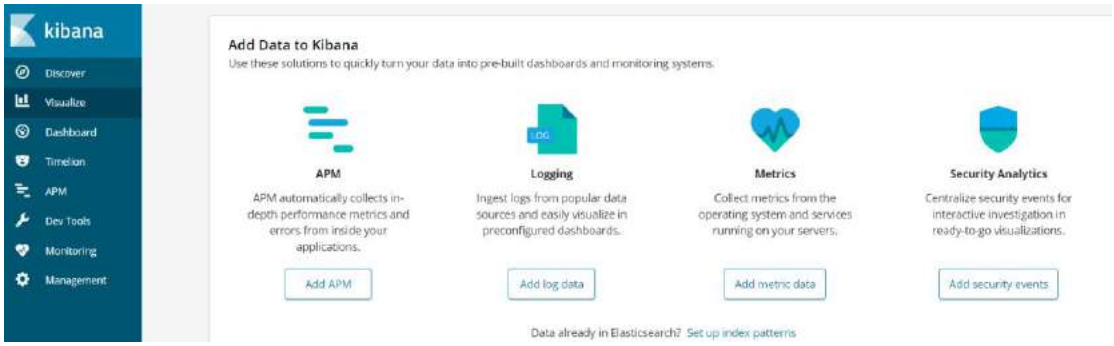
- 配置文件在 config 文件夹下
- Kibana.yml 常用配置说明

```
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "192.168.14.15"

# Enables you to specify a path to mount Kibana at if you are running behind a proxy.
# Use the `server.rewriteBasePath` setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#server.basePath: ""

# Specifies whether Kibana should rewrite requests that are prefixed with
# `server.basePath` or require that they are rewritten by your reverse proxy.
# This setting was effectively always `false` before Kibana 6.3 and will
# default to `true` starting in Kibana 7.0.
#server.rewriteBasePath: false

# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name.  This is used for display purposes.
#server.name: "your-hostname"

# The URL of the Elasticsearch instance to use for all your queries.
elasticsearch.url: "http://192.168.14.14:9200"

# When this setting's value is true Kibana uses the hostname specified in the server.host
# setting. When the value of this setting is false, Kibana uses the hostname of the host
# that connects to this Kibana instance.
#elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved searches, visualizations and
# dashboards. Kibana creates a new index if the index doesn't already exist.
kibana.index: ".kibana"
```

Server.host/server.port:访问的端口号和地址(地址设置后才能被外网访问)

Elasticsearch.url:访问 elasticserach 的地址

5.3、kibana 功能简介



Discover:数据搜索查看

Visualize:图标制作

Dashboard:仪表盘制作

Timeline:时序数据的高级可视化分析

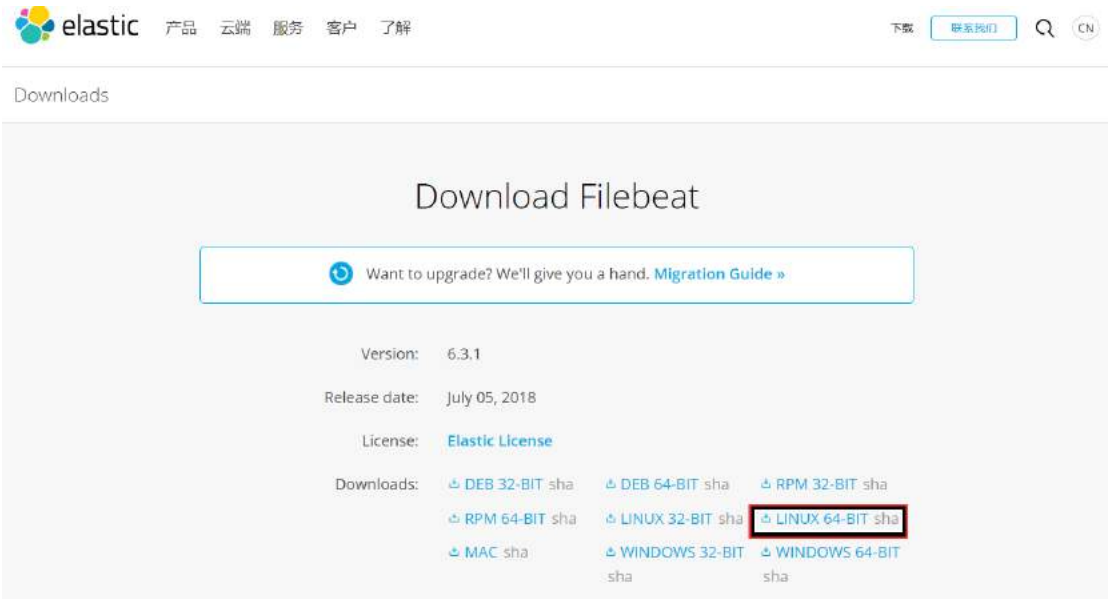
DevTools:开发者工具

Management:kibana 相关配置

6、Filebeat 和 packetbeat

2.1、Filebeat

- 下载 Filebeat
  https://www.elastic.co/cn/downloads/beats/filebeat
  查看系统位数：getconf LONG_BIT



2.2、Packetbeat

- Packetbeat 简介
  （1）实时抓取网络包
  （2）自动解析应用层协议（抓包）
  DNS、Http、Redis、Mysql 等
- Packetbeat 抓取 elasticserach 请求数据
  （1）进入 packetbeat 目录，创建 es.yml 文件
  （2）编辑 es.yml 文件

```
packetbeat.interfaces.device: ens33#网卡

packetbeat.protocols.http:
    ports: [9200]#es 端口
    send_request: true#抓取请求信息
    include_body_for: ["application/json", "x-www-form-urlencoded"]#包含内容
output.console:
    pretty: true#控制台输出
```

（3）启动 packetbeat

sudo ./packetbeat -e -c es.yml -strict.perms=false

7、Nginx

- 安装 nginx

#安装依赖环境
yum install gcc-c++
yum install pcre-devel
yum install zlib zlib-devel
yum install openssl openssl-deve
#//一键安装上面四个依赖
#yum -y install gcc zlib zlib-devel pcre-devel openssl openssl-devel

#解压
tar -xvf nginx-1.13.7.tar.gz

#进入 nginx 目录
cd /usr/local/nginx　#执行命令

./configure

#执行 make 命令 make//执行 make install 命令
make
make install
//启动命令
nginx/sbin/nginx
//停止命令
nginx/sbin/nginx -s stop 或者 : nginx -s quit
//重启命令
nginx -s reload

8、数据可视化演示实战

8.1、实战说明

- 需求：

收集 Elasticserach 集群的查询语句
分析查询语句的常用语句、响应时长等

- 方案

数据收集：Packetbeat+logstash
数据分析：Kibana+Elasticsearch

8.2、前期准备

- Production Cluster(生产环境)
  1、Elasticsearch 192.168.14.13:9200
  2、Kibana 192.168.14.15:5601
- Monitoring Cluster(监控环境)
  1、Elasticsearch 192.168.14.16:8200
  2、Kibana 192.168.14.16:8601
- Logstash\packetbeat

8.3、实战

- 启动数据采集集群
  启动 ES：
  ./elasticsearch

```
# ======================== Elasticsearch Configuration =========================
#
# NOTE: Elasticsearch comes with reasonable defaults for most settings.
#       Before you set out to tweak and tune the configuration, make sure you
#       understand what are you trying to accomplish and the consequences.
#
# The primary way of configuring a node is via this file. This template lists
# the most important settings you may want to configure for a production cluster.
#
# Please consult the documentation for further information on configuration options:
# https://www.elastic.co/guide/en/elasticsearch/reference/index.html
#
# 集群的名字
cluster.name: elasticsearch1
# 节点名字
node.name: node-1
# 索引分片个数，默认为5片
#index.number_of_shards: 5
# 索引副本个数，默认为1个副本
#index.number_of_replicas: 1
#discovery.zen.ping.unicast.hosts: ["192.168.14.13","192.168.14.14"]
# 集群个节点IP地址，也可以使用els、els.shuaiguoxia.com等名称，需要各节点能够解析
discovery.zen.minimum_master_nodes: 2
# 为了避免脑裂，集群节点数最少为 半数+1
# 数据存储目录（多个路径用逗号分隔）
path.data: /home/elk1/elasticserach/data
# 日志目录
path.logs: /home/elk1/elasticserach/logs
# 修改一下ES的监听地址，这样别的机器才可以访问
network.host: 192.168.14.15
# 设置节点间交互的tcp端口（集群）,默认是9300
transport.tcp.port: 9300
# 监听端口（默认的就好）
http.port: 9200
# 增加新的参数，这样head插件才可以访问es
http.cors.enabled: true
http.cors.allow-origin: "*"
```

修改 kibana 配置

```
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "192.168.14.15"

# Enables you to specify a path to mount Kibana at if you are running behind a proxy.
# Use the `server.rewriteBasePath` setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#server.basePath: ""

# Specifies whether Kibana should rewrite requests that are prefixed with
# `server.basePath` or require that they are rewritten by your reverse proxy.
# This setting was effectively always `false` before Kibana 6.3 and will
# default to `true` starting in Kibana 7.0.
#server.rewriteBasePath: false

# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name.  This is used for display purposes.
#server.name: "your-hostname"

# The URL of the Elasticsearch instance to use for all your queries.
elasticsearch.url: "http://192.168.14.14:9200"

# When this setting's value is true Kibana uses the hostname specified in the server.host
# setting. When the value of this setting is false, Kibana uses the hostname of the host
# that connects to this Kibana instance.
#elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved searches, visualizations and
# dashboards. Kibana creates a new index if the index doesn't already exist.
kibana.index: ".kibana"
```

./kibana　　#启动

- 启动数据分析集群

（1）启动 ES

同上

（2）启动 logstash

```
input {
        beats {
                port => 5044
        }
}
filter {
        if "search" in [request]{
                grok {
                        match => { "request" => ".*\n\{(?<query_body>.*)"}
                }
                grok {
                        match => { "path" => "\/(?<index>.*)\/_search"}
                }
          if [index] {
            } else {
                        mutate {
                           add_field    => { "index" => "All" }
                }
            }

              mutate {
```

```
                        update    => { "query_body" => "{%{query_body}"}}
            }

        #       mutate {
        #             remove_field => [ "[http][response][body]" ]
        #       }
    }

    output {
        #stdout{codec=>rubydebug}

        if "search" in [request]{
                elasticsearch {
                hosts => "127.0.0.1:9200"
                }
         }
    }
```

（3）启动

./bin/logstash -f config/log4j_to_es.conf

附录：防火墙配置

1、firewalld 的基本使用

启动： systemctl start firewalld

关闭： systemctl stop firewalld

查看状态： systemctl status firewalld

开机禁用 ： systemctl disable firewalld

开机启用 ： systemctl enable firewalld

2.systemctl 是 CentOS7 的服务管理工具中主要的工具，它融合之前 service 和 chkconfig 的功能于一体。

启动一个服务：systemctl start firewalld.service

关闭一个服务：systemctl stop firewalld.service

重启一个服务：systemctl restart firewalld.service

显示一个服务的状态：systemctl status firewalld.service

在开机时启用一个服务：systemctl enable firewalld.service
在开机时禁用一个服务：systemctl disable firewalld.service
查看服务是否开机启动：systemctl is-enabled firewalld.service
查看已启动的服务列表：systemctl list-unit-files|grep enabled
查看启动失败的服务列表：systemctl --failed

3.配置 firewalld-cmd

查看版本： firewall-cmd --version
查看帮助： firewall-cmd --help
显示状态： firewall-cmd --state
查看所有打开的端口： firewall-cmd --zone=public --list-ports
更新防火墙规则： firewall-cmd --reload
查看区域信息: firewall-cmd --get-active-zones
查看指定接口所属区域： firewall-cmd --get-zone-of-interface=eth0
拒绝所有包：firewall-cmd --panic-on
取消拒绝状态： firewall-cmd --panic-off
查看是否拒绝： firewall-cmd --query-panic

4.那怎么开启一个端口呢

添加

firewall-cmd --zone=public --add-port=80/tcp --permanent （--permanent 永久生效，没有此参数重启后失效）

重新载入

firewall-cmd --reload

查看

firewall-cmd --zone= public --query-port=80/tcp

删除

firewall-cmd --zone= public --remove-port=80/tcp --permanent

Nginx 安装手册

1 nginx 安装环境

nginx 是 C 语言开发，建议在 linux 上运行，本教程使用 Centos6.5 作为安装环境。

- gcc

安装 nginx 需要先将官网下载的源码进行编译，编译依赖 gcc 环境，如果没有 gcc 环境，需要安装 gcc：yum install gcc-c++

- PCRE

PCRE(Perl Compatible Regular Expressions)是一个 Perl 库，包括 perl 兼容的正则表达式库。nginx 的 http 模块使用 pcre 来解析正则表达式，所以需要在 linux 上安装 pcre 库。

yum install -y pcre pcre-devel

注：pcre-devel 是使用 pcre 开发的一个二次开发库。nginx 也需要此库。

- zlib

zlib 库提供了很多种压缩和解压缩的方式，nginx 使用 zlib 对 http 包的内容进行 gzip，所以需要在 linux 上安装 zlib 库。

yum install -y zlib zlib-devel

- openssl

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及 SSL 协议，并提供丰富的应用程序供测试或其它目的使用。

nginx 不仅支持 http 协议，还支持 https（即在 ssl 协议上传输 http），所以需要在 linux 安装 openssl 库。

yum install -y openssl openssl-devel

2 编译安装

将 nginx-1.8.0.tar.gz 拷贝至 linux 服务器。

解压：

tar -zxvf nginx-1.8.0.tar.gz

cd nginx-1.8.0

1、 configure

./configure --help 查询详细参数（参考本教程附录部分：nginx 编译参数）

注意：上边将临时文件目录指定为/var/temp/nginx，需要在/var 下创建 temp 及 nginx 目录

参数设置如下：

./configure \
--prefix=/usr/local/nginx \
--pid-path=/var/run/nginx/nginx.pid \
--lock-path=/var/lock/nginx.lock \
--error-log-path=/var/log/nginx/error.log \
--http-log-path=/var/log/nginx/access.log \
--with-http_gzip_static_module \
--http-client-body-temp-path=/var/temp/nginx/client \
--http-proxy-temp-path=/var/temp/nginx/proxy \
--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \
--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \
--http-scgi-temp-path=/var/temp/nginx/scgi

#注意：上边将临时文件目录指定为/var/temp/nginx，需要在/var 下创建 temp 及 nginx 目录

2、 编译安装
make
make　install

安装成功查看安装目录 ：

```
[root@server01 nginx-1.8.0]# cd ../nginx
[root@server01 nginx]# ll
总用量 12
drwxr-xr-x. 2 root root 4096 5月  10 19:54 conf
drwxr-xr-x. 2 root root 4096 5月  10 19:54 html
drwxr-xr-x. 2 root root 4096 5月  10 19:54 sbin
```

3 启动 nginx

cd /usr/local/nginx/sbin/
./nginx

查询 nginx 进程：

```
[root@server01 sbin]# ps aux|grep nginx
root     15098  0.0  0.0   3568   528 ?        Ss   20:15   0:00 nginx: master process ./nginx
nobody   15099  0.0  0.0   3768   884 ?        S    20:15   0:00 nginx: worker process
```

15098 是 nginx 主进程的进程 id，15099 是 nginx 工作进程的进程 id

注意：执行./nginx 启动 nginx，这里可以-c 指定加载的 nginx 配置文件，如下：
./nginx -c /usr/local/nginx/conf/nginx.conf
如果不指定-c，nginx 在启动时默认加载 conf/nginx.conf 文件，此文件的地址也可以在编译安装 nginx 时指定./configure 的参数（--conf-path= 指向配置文件（nginx.conf））

4 停止 nginx

方式 1，快速停止：
cd /usr/local/nginx/sbin
./nginx -s stop
此方式相当于先查出 nginx 进程 id 再使用 kill 命令强制杀掉进程。

方式 2，完整停止(建议使用)：
cd /usr/local/nginx/sbin
./nginx -s quit
此方式停止步骤是待 nginx 进程处理任务完毕进行停止。

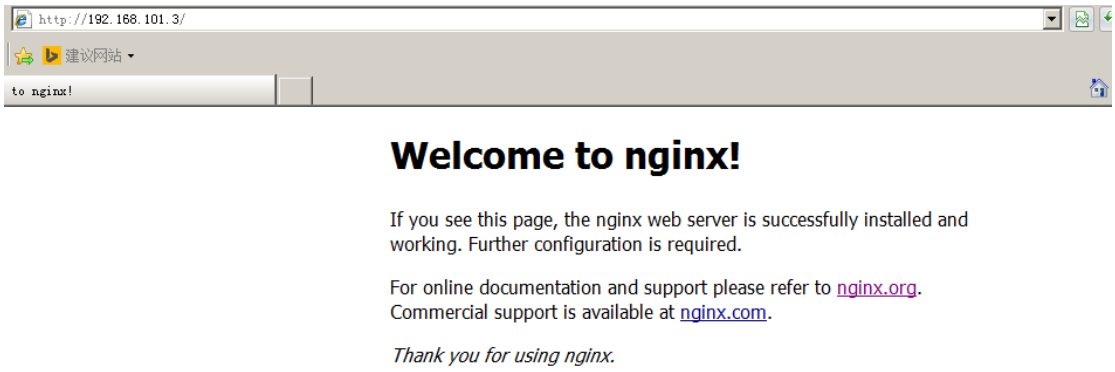
5 重启 nginx

方式 1，先停止再启动（建议使用）：
对 nginx 进行重启相当于先停止 nginx 再启动 nginx，即先执行停止命令再执行启动命令。
如下：
./nginx -s quit
./nginx

方式 2，重新加载配置文件：
当 nginx 的配置文件 nginx.conf 修改后，要想让配置生效需要重启 nginx，使用-s reload 不用先停止 nginx 再启动 nginx 即可将配置信息在 nginx 中生效，如下：
./nginx -s reload

6 测试

nginx 安装成功，启动 nginx，即可访问虚拟机上的 nginx：



到这说明 nginx 上安装成功。

7 开机自启动 nginx

7.1编写 shell 脚本

这里使用的是编写 shell 脚本的方式来处理

vi /etc/init.d/nginx　(输入下面的代码)

```
#!/bin/bash
# nginx Startup script for the Nginx HTTP Server
# it is v.0.0.2 version.
# chkconfig: - 85 15
# description: Nginx is a high-performance web and proxy server.
#              It has a lot of features, but it's not for everyone.
# processname: nginx
# pidfile: /var/run/nginx.pid
# config: /usr/local/nginx/conf/nginx.conf
nginxd=/usr/local/nginx/sbin/nginx
nginx_config=/usr/local/nginx/conf/nginx.conf
nginx_pid=/var/run/nginx.pid
RETVAL=0
prog="nginx"
# Source function library.
. /etc/rc.d/init.d/functions
# Source networking configuration.
. /etc/sysconfig/network
# Check that networking is up.
```

```
[ ${NETWORKING} = "no" ] && exit 0
[ -x $nginxd ] || exit 0
# Start nginx daemons functions.
start() {
if [ -e $nginx_pid ];then
    echo "nginx already running...."
    exit 1
fi
    echo -n $"Starting $prog: "
    daemon $nginxd -c ${nginx_config}
    RETVAL=$?
    echo
    [ $RETVAL = 0 ] && touch /var/lock/subsys/nginx
    return $RETVAL
}
# Stop nginx daemons functions.
stop() {
        echo -n $"Stopping $prog: "
        killproc $nginxd
        RETVAL=$?
        echo
        [ $RETVAL = 0 ] && rm -f /var/lock/subsys/nginx /var/run/nginx.pid
}
# reload nginx service functions.
reload() {
    echo -n $"Reloading $prog: "
    #kill -HUP `cat ${nginx_pid}`
    killproc $nginxd -HUP
    RETVAL=$?
    echo
}
# See how we were called.
case "$1" in
start)
        start
        ;;
stop)
        stop
        ;;
reload)
        reload
        ;;
restart)
        stop
```

```
        start
        ;;
status)
        status $prog
        RETVAL=$?
        ;;
*)
        echo $"Usage: $prog {start|stop|restart|reload|status|help}"
        exit 1
esac
exit $RETVAL
```

:wq　保存并退出

7.2设置文件的访问权限

chmod a+x /etc/init.d/nginx　　(a+x ==> all user can execute　所有用户可执行)

这样在控制台就很容易的操作 nginx 了：查看 Nginx 当前状态、启动 Nginx、停止 Nginx、重启 Nginx…

```
[root@DevelopServer nginx-0.8.54]# /etc/init.d/nginx status
nginx is stopped
[root@DevelopServer nginx-0.8.54]# /etc/init.d/nginx start
Starting nginx:                                            [  OK  ]
[root@DevelopServer nginx-0.8.54]# /etc/init.d/nginx stop
Stopping nginx:                                            [  OK  ]
[root@DevelopServer nginx-0.8.54]# /etc/init.d/nginx restart
Stopping nginx:                                            [FAILED]
Starting nginx:                                            [  OK  ]
[root@DevelopServer nginx-0.8.54]# /etc/init.d/nginx stop
Stopping nginx:                                            [  OK  ]
[root@DevelopServer nginx-0.8.54]# /etc/init.d/nginx restart
Stopping nginx:                                            [FAILED]
Starting nginx:                                            [  OK  ]
[root@DevelopServer nginx-0.8.54]#
```

如果修改了 nginx 的配置文件 nginx.conf，也可以使用上面的命令重新加载新的配置文件并运行，可以将此命令加入到 rc.local 文件中，这样开机的时候 nginx 就默认启动了

7.3加入到 rc.local 文件中

vi /etc/rc.local

加入一行　/etc/init.d/nginx start　　保存并退出，下次重启会生效。