## ES安装

1. wget <https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.5.1.tar.gz>

或者上官网下载

1. **tar -zxvf -C** xxx
2. **sysctl -w vm.max\_map\_count=262144** 设置单个JVM最大线程数
3. **vim /etc/sysctl.conf** 写入**vm.max\_map\_count=262144**，重启也生效
4. **vim /etc/security/limits.conf** 写入**es hard nofile 65536**和**es soft nofile 65536**
5. **useradd -g root es** 创建root组用户es
6. **chown -R es:root xxx** 将es目录下的所有文件的所有者设为es用户
7. **su es** 切换到es用户
8. **./bin/elasticsearch -d** -d表示后台运行
9. **curl localhost:9200** 返回Json，表示安装成功
10. **jps** 命令可以查询到正在运行的elasticSearch
11. **ps -ef | grep elastic** 查看elasticsearch进程是否启动
12. **lsof -i:9200** 查看elasticsearch的9200端口是否被占用
13. **netstat -tunlp | grep 9200**  查看elasticsearch的9200端口是否被占用
14. 下载ik分词器插件，解压后放到**./plugins/ik**下面
15. **elasticsearch-plugin list** 命令查看所有的插件

## KIBANA安装

1. 官网下载
2. 修改**./config/kibana.yml**，server.port、server.host、elasticsearch.hosts。
3. **./bin/kibana** 后台运行kibana
4. **iptables -I INPUT -p tcp -m multiport --dports 9200,5601 -s 10.32.18.181 -j ACCEPT**
5. **./config/kibana.yml**中配置 i18n.locale:=”zh-CN”
6. **./bin/kibana** 运行

在浏览器中访问ip:5601访问

## Logstash安装

1. 下载，解压
2. 在config目录下新建**demo-pipeline.conf**

|  |
| --- |
| input {  stdin{} #从命令行获取输入  }  output {  stdout{} #在命令行输出  } |

1. **./bin/logstash -f ./config/demo-logstash.config** 然后输入hello logstash，随后就会输出：

|  |
| --- |
| {  "@timestamp" => 2020-06-28T05:51:02.703Z,  "@version" => "1",  "host" => "dy0",  "message" => "hello logstash"  } |

## Kibana控制台中操作es命令

### PUT

|  |
| --- |
| PUT /test/**\_doc**/1  {  “name” : “nike”,  “price” : 1000,  “tags” : [“yundong”, “xiuxian”]  } |
| PUT /test2 创建索引的时候设置分片数、备份数、映射关系  {  "**settings**": {  "**number\_of\_shards**": 3,  "**number\_of\_replicas**": 1  },  “**mapping**” : {  “**properties**” : {  “name” : {  “type”:”**text**”  },  “age” : {  “type”:”**long**”  },  “birthday” : {  “type”:”**date**”  }  }  }  } |

### POST

|  |
| --- |
| POST /test/**\_update**/1  {  “**doc**”:  “tags” : [“shangwu”]  } |

### DELETE

|  |
| --- |
| DELETE /test/**\_doc**/1 |

### GET

|  |
| --- |
| **GET /test #返回该索引的mappings和settings** |
| **GET /test/\_doc/1** |
| **GET /test/\_search**  **“query”、”aggs”、”sort”、”from”、”size”、”\_source”**  {  ###查询  **"query"**:{  **"match"**: {  "name":"nike" #分词查找nike  },  **"match\_phrase"**: {  "name": "nike puma" #不要分词，精确查找  },  **"bool"**: {  "**must**": [ #must，和查询，must下面可以加多个match  {  "**match**": {  "name": "aaa"  }  }  ],  "**should**": [ #should，或查询，must下面可以加多个match  {  "**term**": { #term表示精确查询  "name": "aaa"  }  }  ],  "**filter**": {  "**range**": {  "price": {  "gte": 1000,  "lte": 1200  }  }  }  }  },  ##聚合  **"aggs"**: {  "group\_by\_tags": { #自定义操作名  "terms": {  "field": "name", #按照名字聚合  "order": {  "avg\_price": "desc" #按price平均值排序  }  },  "aggs": {  "avg\_price": {  "avg": {  "field": "price" #名字聚合后，price平均值  }  }  }  }  },  ###排序  **"sort"**: [  {  "price": {  "order": "desc"  }  }  ],  ###分页  **"from"**:2,  **"size"**:2,  ###返回字段  **"\_source"**: ["name"],  "**highlight**": { #查询出来的name字段高亮显示  "**pre\_tags**": "<p class='key' style='color:red'>", #自定义高亮显示  "**fields**": {  "name": {}  },  "**post\_tags**": "<p class='key' style='color:red'>"  }  } |

### GET \_cat/

|  |
| --- |
| **GET \_cat/master** |
| 查看es集群的主节点 |
|  |

|  |
| --- |
| **GET \_cat/indices?v** |
| 查看当前节点上各个索引的信息 |
| health status index uuid pri rep docs.count docs.deleted store.size pri.store.size  yellow open ecomerce u~w 1 1 3 1 15.7kb 15.7kb  green open .kibana\_1 u~Q 1 0 54 6 102.1kb 102.1kb |

|  |
| --- |
| **GET \_cat/health** |
| 查看当前节点健康信息 |

|  |
| --- |
| **GET \_cat/shards?v** |
| index shard prirep state docs store ip node  .apm-agent-configuration 0 p STARTED 0 283b 127.0.0.1 dy0  ecomerce 0 p STARTED 3 15.7kb 127.0.0.1 dy0  ecomerce 0 r UNASSIGNED  test 0 p(rimary) STARTED 5 11.4kb 127.0.0.1 dy0  test 0 r(replice) UNASSIGNED  .kibana\_1 0 p STARTED 54 151.9kb 127.0.0.1 dy0  .kibana\_task\_manager\_1 0 p STARTED 2 31.1kb 127.0.0.1 dy0  kibana\_sample\_data\_logs 0 p STARTED 14074 10.8mb 127.0.0.1 dy0 |

## ES核心概念

集群

一个节点默认就是一个集群，集群名默认为“elasticsearch”。

|  |
| --- |
| curl -X GET <http://dy0:9200> |
| {  **"name" : "dy0",**  **"cluster\_name" : "elasticsearch",**  "cluster\_uuid" : "VypmqrNRSKOWdbmcIHxN7w",  "version" : {  "number" : "7.4.1",  "build\_flavor" : "default",  "build\_type" : "tar",  "build\_hash" : "fc0eeb6e2c25915d63d871d344e3d0b45ea0ea1e",  "build\_date" : "2019-10-22T17:16:35.176724Z",  "build\_snapshot" : false,  "lucene\_version" : "8.2.0",  "minimum\_wire\_compatibility\_version" : "6.8.0",  "minimum\_index\_compatibility\_version" : "6.0.0-beta1"  },  "tagline" : "You Know, for Search"  } |

节点

单个ES实例，一个节点运行在一个隔离的容器或虚拟机中。

索引

一个elasticsearch的索引会被分成多个片，每个片都是一个Lucene的倒排索引，所以一个es所以是由多个Lucene的倒排索引构成的。

类型

弃用

文档

elasticsearch中每一条数据就是一个文档，ES是面向文档的

分片（shard）

ES是分布式的搜索引擎，所以**索引**通常会被分解成不同部分，而这些分布在不同节点的数据就是分片，而一个分片就是一个Lucene索引，一个包含倒排索引的文件目录，倒排索引使得elasticsearch在不扫描全部文档的情况下，得出哪些文档包含了特定的关键字。所以，**一个ES索引是由位于不同节点的Lucene倒排索引构成的**。

备份（replica）

备份就是是针对每个分片，在不同的节点上的复制品

分片（shard）和备份（replice）

|  |
| --- |
| **查看索引的分片和备份** |
| get http://dy0:9200/test/\_settings |
| {      "test": {          "settings": {              "index": {                  "creation\_date": "1592642592409",                  "number\_of\_shards": "1",                  "number\_of\_replicas": "1",                  "uuid": "FtZtln26TMaRf9EW1-TZnw",                  "version": {                      "created": "7040199"                  },                  "provided\_name": "test"              }          }      }  } |

|  |
| --- |
| **创建索引时指定分区和备份** |
| put <http://dy0:9200/test{>  settings:{  “number\_of\_shards” : 3, #该分片数只对当前索引有效  “number\_of\_replica” : 3  }  } |

|  |
| --- |
| **索引已经被创建后，修改备份** |
| put <http://dy0:9200/test/_settings{>  “number\_of\_replica” : 6  } |

映射

字段的类型

倒排索引

能够过滤掉无关数据，提高搜索效率。

## IK分词器

IK的两种分词算法：ik\_smart（最少切分）和ik\_max\_word（最细粒度划分）

### ik\_smart（最少切分）

一段文本，按照断点打开来分词，不会重复。

“超级喜欢狂神说Java”会被分成：超级、喜欢、狂、神、说、Java

|  |
| --- |
| **GET \_analyze**  {  **“analyzer” : “ik\_smart”**, #分词器种类  “text” : “xxxxxx” #要进行分词的文本  } |

### ik\_max\_word（最细粒度划分）

|  |
| --- |
| **GET \_analyze**  {  **“analyzer” : “ik\_max\_word”**,  “text” : “xxxxx”  } |

### 分词器配置扩展字典

在plugins/ik/config下添加kaung.dic（文件名自定义），里面写上自定义的分词。然后在同级目录下的IKAnalyzer.cfg.xml配置该.dic文件，然后重启ES。

|  |
| --- |
| <properties>  <comment>IK Analyzer 扩展配置</comment>  <!--用户可以在这里配置自己的扩展字典 -->  **<entry key="ext\_dict">kuang.dic</entry>**  <!--用户可以在这里配置自己的扩展停止词字典-->  <entry key="ext\_stopwords"></entry>  <!--用户可以在这里配置远程扩展字典 -->  <!-- <entry key="remote\_ext\_dict">words\_location</entry> -->  <!--用户可以在这里配置远程扩展停止词字典-->  <!-- <entry key="remote\_ext\_stopwords">words\_location</entry> -->  </properties> |

## docker-compose搭建ELK集群

三台虚拟机（100,101,102）分别安装好docker、docker-compose。

先在三个节点上搭建es集群，然后在100节点上添加kibana和logstash

### 搭建三节点es+单节点Kibana集群来存储和展示数据

分别在三台节点上docker-compose build、docker-compose up运行节点，然后登陆http://dy0:5601访问kibana。

#### 192.168.197.100上的配置：

**目录结构：**

|-docker-elk

|-**docker-compose.yml**

|-elasticsearch

|-config

|-**elasticsearch.yml**

|-**Dockerfile**

|-kibana

|-config

|-**kibana.yml**

|-**Dockerfile**

|-data

|-nodes

|-0

|-ik

|-.env

|  |
| --- |
| **docker-compose.yml** |
| version: '3.2'  services:  **elasticsearch:**  **build:**  **context:** elasticsearch/ #基于elasticsearch目录下的Dockerfile来构建镜像  **args:**  #镜像构建时指定环境变量，构建成功后取消。相当于执行docker build --build-arg  ELK\_VERSION: $ELK\_VERSION #Dockerfile在build过程中需要的参数  **volumes:**  #卷挂载路径  - type: bind  source: ./data  target: /usr/share/elasticsearch/data  - type: bind  source: ./elasticsearch/config/elasticsearch.yml  target: /usr/share/elasticsearch/config/elasticsearch.yml  read\_only: true  #- type: bind  # source: ./elasticsearch/config/jvm.options  # target: /usr/share/elasticsearch/config/jvm.options  # read\_only: true  - type: bind  source: ./ik/\*  target: /usr/share/elasticsearch/plugins/  **environment:** #环境变量配置，会保存在镜像和容器中  **ELASTIC\_PASSWORD:** changeme  **discovery.type:** '' ？  **node.name:** es-0  **discovery.seed\_hosts:** 192.168.197.100,192.168.197.101,192.168.197.102  **cluster.initial\_master\_nodes:** 192.168.197.101,192.168.197.100,192.168.197.102  **node.max\_local\_storage\_nodes:** '3' #每台机器能运行的最大节点数  **ES\_JAVA\_OPTS:** -Xms512m -Xmx512m  **network\_mode:** host  #容器和宿主机共用network spacename，即使用宿主机的IP和端口对外通信  **kibana:**  build:  context: kibana/  args:  ELK\_VERSION: $ELK\_VERSION  volumes:  - type: bind  source: ./kibana/config/kibana.yml  target: /usr/share/kibana/config/kibana.yml  read\_only: true  network\_mode: host  **depends\_on:**  #先启动（不需要完全启动）elasticsearch，再启动kibana  - elasticsearch |

|  |
| --- |
| elasticsearch/config/**elasticsearch.yml** |
| ---  # Default Elasticsearch configuration from Elasticsearch base image.  ## <https://github.com/elastic/elasticsearch/blob/master/distribution/docker/src/docker/config/>  ##elasticsearch.yml  #  **cluster.name:** "dy-cluster" #集群名  **network.host:** 0.0.0.0 #？  ## X-Pack settings  ## see https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-xpack.html  #  **xpack.license.self\_generated.type:** basic  **xpack.security.enabled:** false  **xpack.monitoring.collection.enabled:** true  ##es7中提供新的集群协调子系统  **discovery.seed\_hosts:** ["192.168.197.100","192.168.197.101","192.168.197.102"]  #可能成为master的节点  **cluster.initial\_master\_nodes:** ["192.168.197.100","192.168.197.101","192.168.197.102"]  #首次启动集群时指定可能的master节点  **network.publish\_host:** 192.168.197.100  #设置该节点和其他节点交互的ip地址 |

|  |
| --- |
| elasticsearch/**Dockerfile** |
| ARG ELK\_VERSION  #在镜像被构建、容器被**启动后无效**，可以接收docker-compose.yml中build args设置的参数  #ENV AAA BBB  #在镜像被构建、容器被**启动后仍然有效**，作为环境变量  # https://github.com/elastic/elasticsearch-docker  FROM docker.elastic.co/elasticsearch/elasticsearch:${ELK\_VERSION}  # Add your elasticsearch plugins setup here  # Example: RUN elasticsearch-plugin install analysis-icu |

|  |
| --- |
| kibana/config/**kibana.yml** |
| ---  ## Default Kibana configuration from Kibana base image.  ## https://github.com/elastic/kibana/blob/master/src/dev/build/tasks/os\_packages/docker\_generator/templates/kibana\_yml.template.js  #  server.name: kibana  server.host: "0"  elasticsearch.hosts: [ "http://192.168.197.101:9200" ] #es集群中的节点ip  xpack.monitoring.ui.container.elasticsearch.enabled: true  ## X-Pack security credentials  #  elasticsearch.username: elastic  elasticsearch.password: changeme  i18n.locale: "zh-CN" #使用中文 |

|  |
| --- |
| kibana/Dockerfile |
| ARG ELK\_VERSION  # https://github.com/elastic/kibana-docker  FROM docker.elastic.co/kibana/kibana:${ELK\_VERSION}  # Add your kibana plugins setup here  # Example: RUN kibana-plugin install <name|url> |

#### 192.168.197.101上的配置：（102类似）

**目录结构：**

|-docker-elk

|-**docker-compose.yml**

|-elasticsearch

|-config

|-**elasticsearch.yml**

|-**Dockerfile**

|-data

|-nodes

|-0

|-ik

|-.env

|  |
| --- |
| **docker-compose.yml** |
| version: '3.2'  services:  elasticsearch:  build:  context: elasticsearch/  args:  ELK\_VERSION: $ELK\_VERSION  volumes:  - type: bind  source: ./data  target: /usr/share/elasticsearch/data  - type: bind  source: ./elasticsearch/config/elasticsearch.yml  target: /usr/share/elasticsearch/config/elasticsearch.yml  read\_only: true  #- type: bind  # source: ./elasticsearch/config/jvm.options  # target: /usr/share/elasticsearch/config/jvm.options  # read\_only: true  - type: bind  source: ./ik/\*  target: /usr/share/elasticsearch/plugins/  environment:  ELASTIC\_PASSWORD: changeme  discovery.type: ''  node.name: es-1  node.max\_local\_storage\_nodes: '3'  ES\_JAVA\_OPTS: -Xms512m -Xmx512m  network\_mode: host |

|  |
| --- |
| elasticsearch/config/**elasticsearch.yml** |
| ---  ## Default Elasticsearch configuration from Elasticsearch base image.  ### https://github.com/elastic/elasticsearch/blob/master/distribution/docker/src/docker/config/elasticsearch.yml  #  cluster.name: "dy-cluster"  network.host: 0.0.0.0  ## X-Pack settings  ## see https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-xpack.html  #  xpack.license.self\_generated.type: basic  xpack.security.enabled: false  xpack.monitoring.collection.enabled: true  discovery.seed\_hosts: ["192.168.197.100","192.168.197.101","192.168.197.102"]  cluster.initial\_master\_nodes: ["192.168.197.101"]  network.publish\_host: 192.168.197.101 |

|  |
| --- |
| Elasticsearch/**Dockerfile** |
| ARG ELK\_VERSION  # https://github.com/elastic/elasticsearch-docker  FROM docker.elastic.co/elasticsearch/elasticsearch:${ELK\_VERSION}  # Add your elasticsearch plugins setup here  # Example: RUN elasticsearch-plugin install analysis-icu |

cluster.initial\_master\_nodes：集群初始化的提供的master候选地址，第一次启动时将从该列表中获取master

discovery.seed\_hosts：配置该节点会与哪些候选地址进行通信，hostname,ip ,ip+port

### 添加logstash来获取数据

#### 192.168.197.100上的配置：

**目录结构：**

|-docker-elk

|-**docker-compose.yml**

|-elasticsearch

|-**Dockerfile**

|-config

|-**elasticsearch.yml**

|-kibana

|-**Dockerfile**

|-config

|-**kibana.yml**

|-logstash

|-**Dockerfile**

|-config

|-**logstash.yml**

|-**pipeline**

|-**logstash-demo.conf**

|-**template**

|-**logstash-demo-template.json**

|-data

|-nodes

|-0

|-ik

|-.env

1、修改docker-elk下的docker-compose.yml，添加logstash部分的配置。

|  |
| --- |
| version: '3.2'  services:  **elasticsearch:**  build:  context: elasticsearch/  args:  ELK\_VERSION: $ELK\_VERSION  volumes:  - type: bind  source: ./data  target: /usr/share/elasticsearch/data  - type: bind  source: ./elasticsearch/config/elasticsearch.yml  target: /usr/share/elasticsearch/config/elasticsearch.yml  read\_only: true  #- type: bind  # source: ./elasticsearch/config/jvm.options  # target: /usr/share/elasticsearch/config/jvm.options  # read\_only: true  - type: bind  source: ./ik/\*  target: /usr/share/elasticsearch/plugins/  environment:  ELASTIC\_PASSWORD: changeme  discovery.type: ''  node.name: es-0  discovery.seed\_hosts: 192.168.197.100,192.168.197.101,192.168.197.102  cluster.initial\_master\_nodes: 192.168.197.101,192.168.197.100,192.168.197.102  node.max\_local\_storage\_nodes: '3'  ES\_JAVA\_OPTS: -Xms512m -Xmx512m  network\_mode: host  **logstash:**  build:  context: logstash/  args:  ELK\_VERSION: $ELK\_VERSION  volumes:  - type: bind  source: ./logstash/config/**logstash.yml**  target: /usr/share/logstash/config/logstash.yml  read\_only: true  - type: bind  source: ./logstash/**pipeline**  target: /usr/share/logstash/pipeline  read\_only: true  - type: bind  source: ./logstash/**template**  target: /usr/share/logstash/template  read\_only: true  environment:  LS\_JAVA\_OPTS: "-Xmx512m -Xms512m"  network\_mode: host  depends\_on:  - elasticsearch  **kibana:**  build:  context: kibana/  args:  ELK\_VERSION: $ELK\_VERSION  volumes:  - type: bind  source: ./kibana/config/kibana.yml  target: /usr/share/kibana/config/kibana.yml  read\_only: true  network\_mode: host  depends\_on:  - elasticsearch |

1. 创建./config/logstash.yml

|  |
| --- |
| ---  ## Default Logstash configuration from Logstash base image.  ## https://github.com/elastic/logstash/blob/master/docker/data/logstash/config/logstash-full.yml  #  http.host: "0.0.0.0"  **xpack.monitoring.elasticsearch.hosts: [ "http://192.168.197.101:9200" ]** #可以监控es，作为输出目的地  ### X-Pack security credentials  #  xpack.monitoring.enabled: true  xpack.monitoring.elasticsearch.username: elastic  xpack.monitoring.elasticsearch.password: changeme |

1. 创建./pipeline/logstash-demo.conf

|  |
| --- |
| input {  **kafka** { #kafka作为数据来源  bootstrap\_servers => "192.168.197.102:9092" #102上的kafka作为数据来源  topics => ["logstash\_demo"] #kafka的某条主题  consumer\_threads => 2 #kafka消费者线程  group\_id => "logstash"  codec => "json"  decorate\_events => true  }  }  output {  **elasticsearch** {  hosts => ["http://192.168.197.101:9200"] #101上的es作为数据目的地  index => "logstash-demo-%{+YYYY.MM.dd}" #根据日期新建索引  user => "elastic"  password => "changeme"  template\_overwrite => true  template\_name => "logstash\_demo.json" #模板名  template => "/usr/share/logstash/template/logstash\_demo\_template.json"  #模板的定义，logstash-demo.conf中配置的数据将会按照logstash\_demo\_template.json中定义的格式进行存储。  }  } |

1. 创建./template/logstash\_demo\_template.json

|  |
| --- |
| {  "order" : 10,  "version" : 60001,  "index\_patterns" : [  "logstash-demo-\*"  ],  "settings" : {  "index" : {  "refresh\_interval" : "10s",  "max\_result\_window" : "100000"  }  },  "mappings" : {  "properties": {  "@timestamp": {  "type": "date"  },  "@version": {  "type": "keyword"  },  "name": {  "type": "text",  "fielddata": true,  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  },  "age": {  "type": "long",  "fielddata": true,  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  },  "address": {  "type": "text",  "fielddata": true,  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  }  }  }  } |

### 添加kafka作为数据来源

1. 在192.168.197.102下载解压kafka
2. 在./config/server.properties中配置kafka依赖的zookeeper的ip和端口

|  |
| --- |
| zookeeper.connect=192.168.197.102:2181 |

1. 在102上启动zookeeper和kafka

|  |
| --- |
| **zKServer.sh start**  **./bin/kafka-server-start.sh config/server.properties &** |

1. 创建一个名为logstash\_demo的主题

|  |
| --- |
| **./bin/kafka-topics.sh --create --zookeeper 192.168.197.102:2181 --replication-factor 1 --partitions 1 --topic logstash\_demo** |

1. 向logstash\_demo这个topic中生产数据

|  |
| --- |
| **./kafka-console-producer.sh --broker-list localhost:9092 --topic logstash\_demo** |

发送的数据需要符合logstash\_demo\_template.json中定义的格式，这样数据就会按照logstash-demo.con定义的pipeline存储来es中。