실시간 분석

|  |
| --- |
| 엘라스틱 서치에 저장된 데이터 검색에 사용되기 위해 별도의 재시작이나 상태의 갱신이 필요하지 않다  데이터는 색인작업이 완료됨에 동시에 바로 검색할수 있다. 실시간 (real time)분석이라고 부른다 |

분산시스템

|  |
| --- |
| 엘라스텍서치는 여러개의 노드로 구성되는 분산시스템 이다 노드는 데이터 색인하고 검색기능을 수행하는 엘라스틱 서치의 단위의 프로세스이다. 소규모의 시스템에서 적은 수의 노드로 시스템을 구성후 시스템 규모가 늘어나면 기조논드에서 새노드를 실행하여 연결을 쉽게할수 있다. 각노드에 분선 저장되고 복사본을 유지해 각종 충돌로부터 유실을 방지한다. |

높은 가용성

|  |
| --- |
| 하나이상 노드로 구성돼 있으며 원본과 복사본을 가지고 있어 서로 다른 위치에 나누어 저장한다  노드가 종료되거나 실행에 실패 하는경우 데이터를 다른 노드로 옮기는 작업을 수행한다. 높은 가용성과 안정성을 제공한다 |

멀타 텐넌시

|  |
| --- |
| 데이터 색인을 이요한 전문검색 (풀 텍스트 서치)를 지원한다 |

JSON 문서기반

|  |
| --- |
| 기본적으로 JSON구조로 저장된다 |

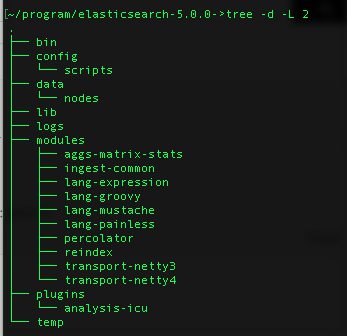
RESTFul API

|  |
| --- |
| URI를 사용한 동작이 가능하다 HTTP프로토콜로 JSON문서의 입출력과 다양한 제어를 하룻 있다. |

엘라스틱서치 설치

https://www.elastic.co/kr/downloads

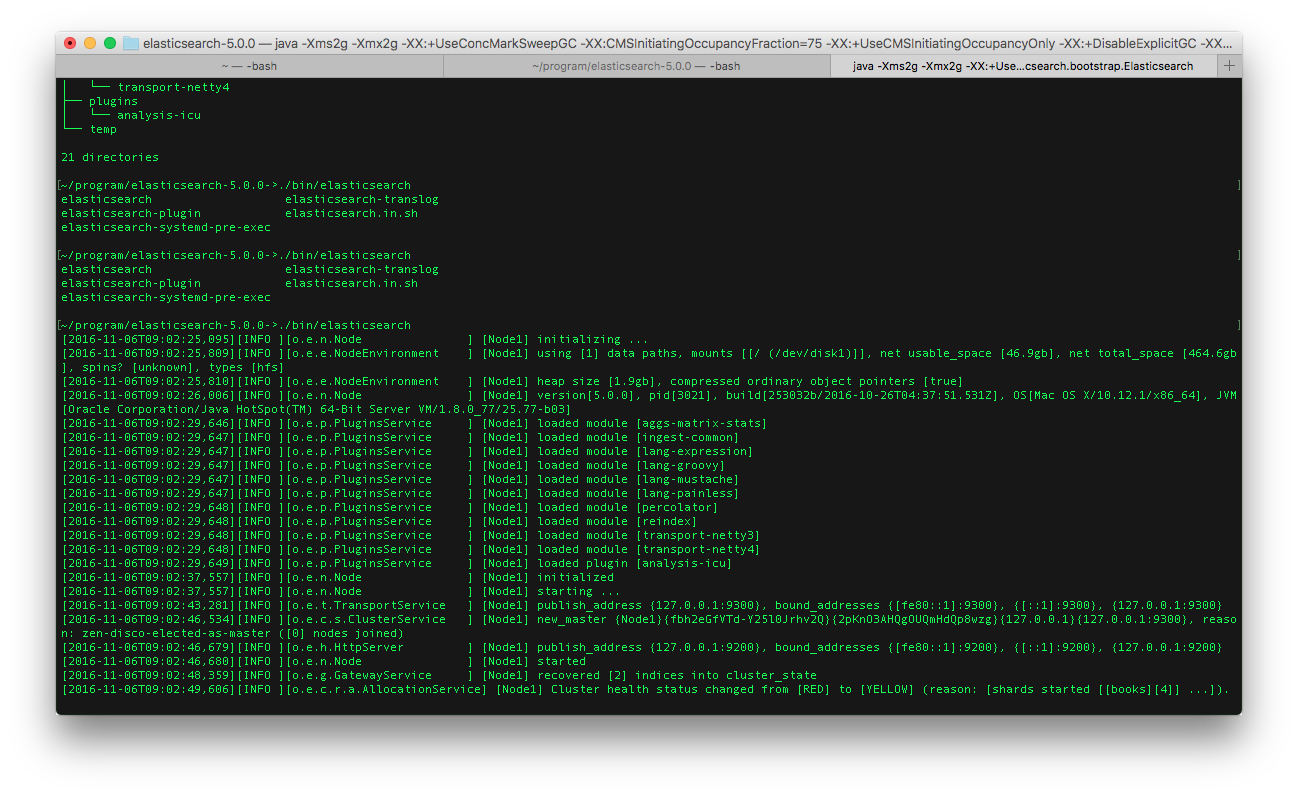
자바로 만들어졌기때문에 자바 JVM있어야한다.



실행 : /bin/elasticsearch

설치및 실행 끝`~ 넘쉽다잉`~

여기서 별도의 노드명을 지정하지 않으면 실행할때마다 임의 노드명으로 실행된다



여기서는 노드명이 Node1이다

|  |  |
| --- | --- |
| -d | 백그라운드에서 돌리기 |
| -p 파일명 | 프로세스 ID를 파일명으로 저장된 파일에 저장 |
| -f | 포그라운드에서 돌리가 |

로그 확인은

logs/ekastcsearch.log

콜해보자

~->curl -XGET http://127.0.0.1:9200

{

"name" : "fbh2eGf",

"cluster\_name" : "elasticsearch",

"cluster\_uuid" : "1cP8r8MWQIaO72fyCGcajQ",

"version" : {

"number" : "5.0.0",

"build\_hash" : "253032b",

"build\_date" : "2016-10-26T04:37:51.531Z",

"build\_snapshot" : false,

"lucene\_version" : "6.2.0"

},

"tagline" : "You Know, for Search"

}

~->

Config

config/elasticsearcg.yml

클러스터

#cluster.name: my-application

클러스터 이름을 지정할수 있다 별도 지정하지않으면 elasticsearch

반듯이 : 뒤에 공백이 하나있어야한다.. 주의

예제 2.28 클러스터 설정 확인

```

curl -XGET localhost:9200/\_cluster/stats?pretty=true

노드

#node.name: node-1

엘라스틱서치는 하나 이상 노드로 구성된다 실행된 하나으 엘리스틱서치 프로세스를 하논드라고 하고 각노드 연결된 전체 시스템을 엘리스틱 클러스터라고 한다.

인덱스

인덱스를 구성하는 샤드(Shard)와 복사본 (Replica)에 대한 설정이다

본적으로 엘라스틱 서치에 색인되는 데이터들은 5개의 샤드와 1개의 복사본으로 구성돼 저장된다 다음 내용을 변경하면 기본으로 구성되는 샤드와 복제본의 개수를 변경할수 있다.

index.number\_of\_shards: 5

샤드 개수

index.number\_of\_replica: 1

복사본 개수

경로

path.conf: /path/to/conf

Config/ 설정파일 위치

path.data: /path/to/data

데이터 파일경로

path.logs: /path/to/logs

로그경로

path.work:

임시작업경로

path.plugins:

플로그인설치경로

플러그인

elastcsearch.yml에서 plugin.mandatory: 옵션뒤에 플러그인이름을 명시할슀다 ,로 구분해서 여러개 플러그인지정가능하다

plugin.mandatory: mapper-attachents,lang-groovy

메모리

bootstrap.mlockall: true 하면 메모리를 고정한다 swap을 방지할수있다.

네트워크

network.host: 192.168.0.1

내부IP

network.bind\_host: 192.168.0.1

서배 내부 IP

network.publish\_host: 192.168.0.1

공개IP

등등

게이트웨이, 리커버리제한, 디스커버리, 슬로우로그, 여러가지 옵션이 존재한다..

플러그인설치

https://www.elastic.co/guide/en/elasticsearch/plugins/current/analysis-icu.html

sudo bin/elasticsearch-plugin install [plugin\_name]

For instance, to install the core ICU plugin, just run the following command:

sudo bin/elasticsearch-plugin install analysis-icu

Removaledit

The plugin can be removed with the following command:

sudo bin/elasticsearch-plugin remove analysis-icu

구조 용어

관계형DB

엘라스틱서치

데이터베이스(database)

인덱스(index)

테이블(table)

타입(type)

열(row)

도큐먼트(document)

행(column)

필드(field)

스키마(schema)

매핑(mapping)

데이터입력

/books/book/1 도큐먼트 데이터 입력

요청

응답

curl -XPUT http://localhost:9200/books/book/1 -d '

{

"title" : "Elasticsearch Guide",

"author" : "Kim",

"date" : "2014-05-01",

"pages" : 250

}'

{"\_index":"books","\_type":"book","\_id":"1","\_version":1,"result":"created","\_shards":{"total":2,"successful":1,"failed":0},"created":true}

주소체계

/{\_index}/{\_type}/{\_id}

id값을 지정하지 않고 처리한다면 임의 id값이 지정된다

curl -XPOST localhost:9200/books/book -d '

{

"title" : "Elasticsearch Guide",

"author" : "Kim",

"started" : "2014-05-1",

"pages" : 250

}'

{"\_index":"books","\_type":"book","\_id":"AVgzVPYTmwBZ\_om2Cmil","\_version":1,"result":"created","\_shards":{"total":2,"successful":1,"failed":0},"created":true}

위내용은 처음 실행되어 들어간거라 created가 true 이다.

만약에 같은 id값으로 한다면 false이되면서 \_version이 한단계 높아진다.

{"\_index":"books","\_type":"book","\_id":"1","\_version":2,"result":"updated","\_shards":{"total":2,"successful":1,"failed":0},"created":false}

데이터 확인

/books/book/1 도큐먼트 데이터 확인

curl -XGET http://localhost:9200/books/book/1

{"\_index":"books","\_type":"book","\_id":"1","\_version":2,"found":true,"\_source":

{

"title" : "Elasticsearch Guide",

"author" : "Kim",

"date" : "2014-05-01",

"pages" : 250

}}

\_source필드값에 입력한 JSON값이 들어가있다.

?pretty=true 옵션을주면 출력값을 줄바꿈해서 보여준다

curl -XGET http://localhost:9200/books/book/1?pretty=true

데이터 삭제

예제 3.7 /books/book/1/\_source 데이터 확인

```

curl -XGET http://localhost:9200/books/book/1/\_source

```

{

"title" : "Elasticsearch Guide",

"author" : "Kim",

"date" : "2014-05-01",

"pages" : 250

}

###3.2.2 데이터 삭제

예제 3.8 /books/book/1/ 도큐먼트 삭제

```

curl -XDELETE http://localhost:9200/books/book/1

```

{"found":true,"\_index":"books","\_type":"book","\_id":"1","\_version":3,"result":"deleted","\_shards":{"total":2,"successful":1,"failed":0}}

예제 3.9 /books/book/1/ 도큐먼트 삭제 후 확인

```

curl -XGET http://localhost:9200/books/book/1

```

{"\_index":"books","\_type":"book","\_id":"1","found":false}

found값이 false 나오며 찾을수 없음 나온다.

예제 3.10 /books/book/1 도큐먼트 삭제 후 다시 데이터 입력

```

curl -XPOST http://localhost:9200/books/book/1 -d '

{

"title" : "Elasticsearch Guide",

"author" : [ "Kim", "Lee" ],

"date" : "2014-05-01",

"pages" : 300

}'

```

예제 3.11 books/book 필드 삭제 후 도큐먼트 데이터 입력

```

curl -XDELETE http://localhost:9200/books/book

curl -XPOST http://localhost:9200/books/book/1 -d '

{

"title" : "Elasticsearch Guide",

"author" : [ "Kim", "Lee" ],

"date" : "2014-05-01",

"pages" : 300

}'

```

예제 3.12 /books 인덱스 삭제 후 /books/book/1 도큐먼트 확인

```

curl -XDELETE http://localhost:9200/books

{"acknowledged":true}

데이터 확인

curl -XGET http://localhost:9200/books/book/1

{"error":{"root\_cause":[{"type":"index\_not\_found\_exception","reason":"no such index","resource.type":"index\_expression","resource.id":"books","index\_uuid":"\_na\_","index":"books"}],"type":"index\_not\_found\_exception","reason":"no such index","resource.type":"index\_expression","resource.id":"books","index\_uuid":"\_na\_","index":"books"},"status":404}

데이터 업데이트(\_update) API

###3.2.3 데이터 업데이트(\_update) API

예제 3.13 /books/book/1 도큐먼트 확인

```

curl -XGET localhost:9200/books/book/1

```

{"\_index":"books","\_type":"book","\_id":"1","\_version":1,"found":true,"\_source":

{

"title" : "Elasticsearch Guide",

"author" : [ "Kim", "Lee" ],

"date" : "2014-05-01",

"pages" : 300

}}

예제 3.14 books/book/1 도큐먼트에 category 필드 추가

```

curl -XPOST localhost:9200/books/book/1/\_update -d '

{

"doc" : {

"category" : "ICT"

}

}'

{"\_index":"books","\_type":"book","\_id":"1","\_version":2,"result":"updated","\_shards":{"total":2,"successful":1,"failed":0}}

버전올라감

curl -XGET localhost:9200/books/book/1?pretty

{

"\_index" : "books",

"\_type" : "book",

"\_id" : "1",

"\_version" : 2,

"found" : true,

"\_source" : {

"title" : "Elasticsearch Guide",

"author" : [

"Kim",

"Lee"

],

"date" : "2014-05-01",

"pages" : 300,

"category" : "ICT"

}

}

예제 3.15 books/book/1 도큐먼트의 author 필드를 Lee로 변경

```

curl -XPOST localhost:9200/books/book/1/\_update -d '

{

"doc" : {

"author" : "Lee"

}

}'

{"\_index":"books","\_type":"book","\_id":"1","\_version":3,"result":"updated","\_shards":{"total":2,"successful":1,"failed":0}}

확인

curl -XGET localhost:9200/books/book/1?pretty

{

"\_index" : "books",

"\_type" : "book",

"\_id" : "1",

"\_version" : 3,

"found" : true,

"\_source" : {

"title" : "Elasticsearch Guide",

"author" : "Lee",

"date" : "2014-05-01",

"pages" : 300,

"category" : "ICT"

}

}

예제 3.16 books/book/1 도큐먼트의 pages 필드에 50을 더함

```

curl -XPOST localhost:9200/books/book/1/\_update -d '

{

"script" : "ctx.\_source.pages += 50"

}'

{"\_index":"books","\_type":"book","\_id":"1","\_version":4,"result":"updated","\_shards":{"total":2,"successful":1,"failed":0}}

curl -XGET localhost:9200/books/book/1?pretty

{

"\_index" : "books",

"\_type" : "book",

"\_id" : "1",

"\_version" : 4,

"found" : true,

"\_source" : {

"title" : "Elasticsearch Guide",

"author" : "Lee",

"date" : "2014-05-01",

"pages" : 350,

"category" : "ICT"

}

}

예제 3.17 author 필드의 값 "Lee"를 배열 ["Lee"]로 변경

```

curl -XPOST localhost:9200/books/book/1/\_update -d '

{

"doc" : {

"author" : ["Lee"]

}

}'

{"\_index":"books","\_type":"book","\_id":"1","\_version":5,"result":"updated","\_shards":{"total":2,"successful":1,"failed":0}}

curl -XGET localhost:9200/books/book/1?pretty

{

"\_index" : "books",

"\_type" : "book",

"\_id" : "1",

"\_version" : 5,

"found" : true,

"\_source" : {

"title" : "Elasticsearch Guide",

"author" : [

"Lee"

],

"date" : "2014-05-01",

"pages" : 350,

"category" : "ICT"

}

}

curl -XGET localhost:9200/books/book/1?pretty

예제 3.19 author 필드가 "Kim"을 포함하는 경우 pages 필드값을 100으로 변경

```

curl -XPOST localhost:9200/books/book/1/\_update -d '

{

"script" : "if(ctx.\_source.author.contains(auth)) { ctx.\_source.pages = 100 } else { ctx.\_source.pages = 200 }",

"params" : { "auth" : "Kim" }

}'

curl -XGET localhost:9200/books/book/1?pretty

예제 3.20 pages 필드가 100 이하일 때 도큐먼트 삭제

```

curl -XPOST localhost:9200/books/book/1/\_update -d '

{

"script" : "ctx.\_source.pages <= page\_cnt ? ctx.op = \"delete\" : ctx.op = \"none\"",

"params" : { "page\_cnt" : 100 }

}'

curl -XGET localhost:9200/books/book/1?pretty

```

예제 3.21 도큐먼트가 없다면 도큐먼트 생성. 도큐먼트가 있다면 counter 필드 1 증가

```

curl -XGET localhost:9200/books/book/1?pretty

curl -XPOST localhost:9200/books/book/1/\_update -d '

{

"script" : "ctx.\_source.counter += count",

"params" : { "count" : 1 },

"upsert" : { "counter" : 0 }

}'

curl -XGET localhost:9200/books/book/1?pretty

curl -XPOST localhost:9200/books/book/1/\_update -d '

{

"script" : "ctx.\_source.counter += count",

"params" : { "count" : 1 },

"upsert" : { "counter" : 0 }

}'

curl -XGET localhost:9200/books/book/1?pretty

```

예제 3.23 book\_1 파일의 내용을 읽어 들여 도큐먼트 데이터로 저장

```

curl -XPUT localhost:9200/books/book/1 -d @book\_1

curl -XGET localhost:9200/books/book/1?pretty

```

-d 옵션으로 body부분을 넣었다 하지만 파일내용으로 넣어보자 @파일명

예제 3.24 update\_1 파일의 내용을 읽어 들여 업데이트 스크립트 실행

```

echo '{

"doc" : {

"category" : "ICT"

}

}' > update\_1

curl -XPOST localhost:9200/books/book/1/\_update -d @update\_1

curl -XGET localhost:9200/books/book/1?pretty

```

벌크(\_bulk) API를 이용한 배치작업

입력데이터를 한꺼번에 모아 처리하므로 데이터를 각각 처리하고 겨로가를 반환할떄보다 속도가 빠르다

예제 3.25 벌크 API를 이용한 /books/book/1, 2 도큐먼트 입력

```

curl -XPOST localhost:9200/\_bulk -d '

{ "index" : { "\_index" : "books", "\_type" : "book", "\_id" : "1" } }

{ "title" : "Elasticsearch Guide", "author" : "Kim", "pages" : 250 }

{ "index" : { "\_index" : "books", "\_type" : "book", "\_id" : "2" } }

{ "title" : "Elasticsearch Easy Guide", "author" : "Lee", "pages" : 300 }

'

{"took":269,"errors":false,"items":[{"index":{"\_index":"books","\_type":"book","\_id":"1","\_version":6,"result":"updated","\_shards":{"total":2,"successful":1,"failed":0},"created":false,"status":200}},{"index":{"\_index":"books","\_type":"book","\_id":"2","\_version":1,"result":"created","\_shards":{"total":2,"successful":1,"failed":0},"created":true,"status":201}}]}

curl -XGET localhost:9200/books/book/1?pretty

{

"\_index" : "books",

"\_type" : "book",

"\_id" : "1",

"\_version" : 6,

"found" : true,

"\_source" : {

"title" : "Elasticsearch Guide",

"author" : "Kim",

"pages" : 250

}

}

curl -XGET localhost:9200/books/book/2?pretty

{

"\_index" : "books",

"\_type" : "book",

"\_id" : "2",

"\_version" : 1,

"found" : true,

"\_source" : {

"title" : "Elasticsearch Easy Guide",

"author" : "Lee",

"pages" : 300

}

}

<bulk\_1> 파일 내용

```

{ "delete" : { "\_index" : "books", "\_type" : "book", "\_id" : "1" } }

{ "update" : { "\_index" : "books", "\_type" : "book", "\_id" : "2" } }

{ "doc" : {"date" : "2014-05-01"} }

{ "create" : { "\_index" : "books", "\_type" : "book", "\_id" : "3" } }

{ "title" : "Elasticsearch Guide II", "author" : "Park", "pages" : 400 }

```

예제 3.26 bulk\_1 파일의 내용을 벌크로 실행

```

curl -XPOST localhost:9200/\_bulk?pretty --data-binary @bulk\_1

```

벌크UDP API

UDP프로토컬을통해 벌크처리할 데이터를 입력받는 기능을 제공한다.

설정값에서 bulk.udp.enable: true

예제 3.30 bulk\_udp 파일에 벌크 입력 데이터 저장

```

curl -XDELETE localhost:9200/books

echo '{ "index" : { "\_index" : "books", "\_type" : "book", "\_id" : "1" } }

{ "title" : "Elasticsearch Guide", "author" : "Kim", "pages" : 250 }

{ "index" : { "\_index" : "books", "\_type" : "book", "\_id" : "2" } }

{ "title" : "Elasticsearch Easy Guide", "author" : "Lee", "pages" : 300 }' > bulk\_udp

cat bulk\_udp

```

예제 3.31 nc 명령으로 벌크 UDP 입력을 통한 도큐먼트 생성

```

cat bulk\_udp | nc -w 0 -u localhost 9700

curl -XGET localhost:9200/books/book/1?pretty

curl -XGET localhost:9200/books/book/2?pretty

```

시스템구조

클러스터와 노드

클러스터 > +노드 하나의엘라스틱서치 프로세스는 하나의 노드를 담당한다

처럼 이루어지며 클러스터는 데이터의 접근이나 교환을 할수 없는 독립적인 시스템으로 유지된다

서버가 하나의 클러스터를 구성할수 있으며 반대로 하나의 물리적인 서버에 여러개의 클러스터가 존재할수도 있다.

클러스터는 하나이상의 노드로 이루어져있으며 각노드는 각각 하나의 엘라스틱서치 프로세스로써 실행된다

별도설정없이 엘라스틱서치실행하면 임의의 이름의 노드 하나가 생성된다

같은 시스템 또는 네트워크 바인딩이 되도록 설정한 다른 시스템에서

다시 한번 같은 클러스터명으로 설정된 엘라스틱서치를 실행하면 또다시 임의의 이름노드가 생성되면서 두개의 노드는 하나의 클러스터로 바인딩되어 묶인다 config/elasticsearch.yml 설정파일의 cluster.name: name 서로 다른 이름으로 설정한 노드 두개를 실행하면 바인딩되지 않고 각각 다른 클러스터를 구성하게된다.

스케일 아웃에 유용하다.

#4. 시스템 구조

##4.1 클러스터와 노드

###4.1.1 노드 바인딩

예제 4.1 헤드 플러그인 설치

```

bin/plugin --install mobz/elasticsearch-head

```

예제 4.2 클러스터명을 es\_test로 설정

```

cluster.name: es\_test

```

예제 4.3 엘라스틱서치 실행

```

bin/elasticsearch

```

예제 4.4 새로운 엘라스틱서치 실행

```

bin/elasticsearch

```

예제 4.6 9200, 9201 포트의 REST API 상태 확인

```

curl -XGET localhost:9200

curl -XGET localhost:9201

```

예제 4.7 9200 포트로 데이터 입력 후 9201 포트로 조회

```

curl -XPUT http://localhost:9200/books/book/1 -d '

{

"title" : "Elasticsearch Guide",

"author" : "Kim",

"date" : "2014-05-01",

"pages" : 250

}'

curl -XGET http://localhost:9201/books/book/1

```

예제 4.8 클러스터명 es\_test\_2로 변경 후 엘라스틱서치 실행

```

bin/elasticsearch

```

###4.1.2 마스터 노드와 데이터 노드

예제 4.9 /es\_test/es\_1의 config/elasticsearch.yml 파일 설정

```

cluster.name: es\_test

node.name: "Node1"

node.master: true

node.data: false

```

예제 4.10 /es\_test/es\_2의 config/elasticsearch.yml 파일 설정

```

cluster.name: es\_test

node.name: "Node2"

node.master: false

node.data: true

```

예제 4.11 /es\_test/es\_1, /es\_test/es\_2 엘라스틱서치 실행

```

bin/elasticsearch

bin/elasticsearch

```

예제 4.12 /es\_test/es\_2에 -- node.name=Node3 옵션으로 엘라스틱서치 실행

```

bin/elasticsearch --node.name=Node3

```

##4.2 샤드와 복사본

예제 4.13 magazines 인덱스의 샤드와 복사본 설정

```

curl -XPUT localhost:9200/magazines -d '

{

"settings" : {

"number\_of\_shards" : 2,

"number\_of\_replicas" : 0

}

}'

```

예제 4.14 config/elasticsearch.yml 파일에서 샤드와 복사본 개수 설정

```

index.number\_of\_shards: 5

index.number\_of\_replicas: 1

```

##4.3 네트워크 바인딩과 디스커버리

###4.3.1 젠 디스커버리

예제 4.15 192.168.1.10 서버에서 엘라스틱서치 실행

```

bin/elasticsearch

```

예제 4.16 config/elasticsearch.yml 파일에서 젠 디스커버리 유니캐스트 설정

```

discovery.zen.ping.multicast.enabled: false

discovery.zen.ping.unicast.hosts: ["192.168.1.10", "192.168.1.11"]

```

예제 4.17 192.168.1.10, 192.168.1.11 서버에서 유니캐스트로 바인딩

```

bin/elasticsearch

bin/elasticsearch

```

예제 4.18 서버 1의 config/elasticsearch.yml 파일 설정

```

cluster.name: es\_test

network.bind\_host: 192.168.0.40

network.publish\_host: 121.131.44.50

discovery.zen.ping.multicast.enabled: false

discovery.zen.ping.unicast.hosts: ["192.168.0.40", "54.199.179.186"]

```

예제 4.19 서버 2의 config/elasticsearch.yml 파일 설정

```

cluster.name: es\_test

network.bind\_host: 172.31.3.238

network.publish\_host: 54.199.179.186

discovery.zen.ping.multicast.enabled: false

discovery.zen.ping.unicast.hosts: ["172.31.3.238", "121.131.44.50"]

```

예제 4.20 서버 1 엘라스틱서치 노드 실행

```

bin/elasticsearch

```

예제 4.21 서버 2 엘라스틱서치 노드 실행

```

bin/elasticsearch

```

###4.3.2 아마존 EC2 디스커버리

예제 4.22 EC2에 엘라스틱서치 설치

```

wget https://download.elasticsearch.org/elasticsearch/elasticsearch/elasticsearch-1.1.1.tar.gz

tar xvfz elasticsearch-1.1.1.tar.gz

```

예제 4.23 aws 플러그인 설치.

```

bin/plugin -install elasticsearch/elasticsearch-cloud-aws/2.1.1

```

예제 4.24 EC2 인스턴스에서의 config/elasticsearch.yml 파일 설정

```

cluster.name: es\_test

cloud:

aws:

access\_key: AKIAIEKJMRZYRL43P52A

secret\_key: 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

region: ap-northeast-1

discovery:

type: ec2

```

예제 4.25 AWS EC2에서 엘라스틱서치 실행 - 1

```

bin/elasticsearch

```

예제 4.26 AWS EC2에서 엘라스틱서치 실행 - 2

```

bin/elasticsearch

```

검색 API

q매개변수의 값으로 검색값을 입력한다.

#5. 검색

예제 5.1 5\_1\_books.json 파일 내용을 벌크 API로 입력

```

curl -XPOST localhost:9200/\_bulk --data-binary @5\_1\_books.json

curl -XPOST localhost:9200/\_bulk --data-binary @5\_2\_magazines.json

```

##5.1 검색(\_search) API

예제 5.2 books 인덱스, book 타입에서 hamlet 검색

```

curl 'localhost:9200/books/book/\_search?q=hamlet&pretty'

```

예제 5.3 books 인덱스에서 hamlet 검색

```

curl 'localhost:9200/books/\_search?q=hamlet&pretty'

```

예제 5.4 books, magazines 인덱스에서 time 검색

```

curl 'localhost:9200/books,magazines/\_search?q=time&pretty'

```

예제 5.5 \_all을 사용해 전체 인덱스에서 time 검색

```

curl 'localhost:9200/\_all/\_search?q=time&pretty'

```

예제 5.6 인덱스 지정을 생략해 전체 인덱스에서 time 검색

```

curl 'localhost:9200/\_search?q=time&pretty'

```

URI 검색

###5.2.1 q(query)

예제 5.7 전체 인덱스의 title 필드에서 time 검색

```

curl 'localhost:9200/\_search?q=title:time&pretty'

```

예제 5.8 title 필드에 검색어 time과 machine을 AND 조건으로 검색

```

curl 'localhost:9200/\_search?q=title:time%20AND%20machine&pretty'

```

###5.2.2 df(default field)

예제 5.9 df 매개변수를 사용해서 title 필드에서

time 검색

```

curl 'localhost:9200/\_search?q=time&df=title&pretty'

```

df매개변수를 사용해서 검색할 필드를 지정할수 있다

###5.2.3 default\_operator

예제 5.10 default\_operator 매개변수를 사용해서 기본 조건 명령어를 AND로 지정

```

curl 'localhost:9200/\_search?q=title:time%20machine&default\_operator=AND&pretty'

```

기본조건 명령얼,ㄹ OR에서 AND로 변경할수있다.

###5.2.4 explain

상세값이 출력된다

예제 5.11 explain 매개변수를 사용해서 검색 처리 결과 표시

```

curl 'localhost:9200/\_search?q=title:time&explain&pretty'

```

###5.2.5 \_source

예제 5.12 \_source 매개변수를 false로 설정해 도큐먼트 내용을 배제하고 검색

```

curl 'localhost:9200/\_search?q=title:time&\_source=false&pretty'

```

###5.2.6 fields

예제 5.13 fields 매개변수를 사용해 title, author, category 필드만 출력

```

curl 'localhost:9200/\_search?q=title:time&fields=title,author,category&pretty'

```

특정 필드만 보이게.

###5.2.7 sort

예제 5.14 author 필드가 jules인 도큐먼트를 pages 필드를 기준으로 오름차순 정렬

```

curl 'localhost:9200/books/\_search?q=author:jules&sort=pages&pretty'

```

기본적인 순서는 \_score값을 기준으로 정렬된다.

변경하려며 sort사용하면된다.

pages필드기준으로 오름차순으로 검색한 결과

예제 5.15 author 필드가 jules인 도큐먼트를 pages 필드를 기준으로 내림차순 정렬

```

curl 'localhost:9200/books/\_search?q=author:jules&sort=pages:desc&pretty'

```

예제 5.16 author 필드가 jules인 도큐먼트를 title 필드를 기준으로 오름차순 정렬

```

curl 'localhost:9200/books/\_search?q=author:jules&fields=title&sort=title&pretty'

```

예제 5.17 author 필드가 jules인 도큐먼트를 title 필드를 기준으로 내림차순 정렬

```

curl 'localhost:9200/books/\_search?q=author:jules&fields=title&sort=title:desc&pretty'

```

###5.2.8 timeout

검색이 실행되는동안 타임아웃을 줄수있다. 지정하지 않으면 제한시간없이 전체 결과나올때까지 기다린다.

단위는 밀리초이며 예를들어 timeout=3000지정하면 검색시작하고 3초후에 검색이 강제 종료하고 검색한 결과를 표시한다

###5.2.9 from

검색된결과를 몇번째 값부터 출력할지 지정한다 지정하지않으면 기본값은 0이며 카운트 역시 0부터시작한다

예제 5.18 from 매개변수를 사용해서 2번째 결과부터 표시

```

curl 'localhost:9200/books/\_search?q=author:jules&fields=title&from=1&pretty'

```

---

###5.2.10 size

몇개의 도큐먼트를 표시할지 나타낸다 10이면 100개가 검색되어도 10개만 출력한다.

Config의 http.max\_content\_length: 지정가능

###5.2.11 search\_type

SEARCH TYPE

search\_type 옵션을 사용

검색을 수행하는 방법을 지정

search\_type에 지정할 수 있는 값들

query\_then\_fetch : 전체 샤드의 검색이 모두 수행된 후 결과 출력, 전체 취합된 결과를 size 매개변수에서 지정한 만큼 출력

query\_and\_fetch : 샤드별로 검색되는 대로 결과 출력, size가 10이고 샤드의 개수가 5라면 출력 결과는 샤드당 10개씩 총 50개

dfs\_query\_then\_fetch : 검색 방식은 query\_then\_fetch와 같으며 정확한 스코어링을 위해 검색어들을 사전 처리

dfs\_query\_and\_fetch : 검색 방식은 query\_and\_fetch와 같으며 정확한 스코어링을 위해 검색어들을 사전 처리

count : 검색된 도큐먼트를 배제하고 전체 hits 수만 출력. (가장 빠른 속도)

scan : 검색 결과를 바로 보여주지 않고 저장했다가 \_scroll\_id를 사용해서 나중에 결과를 출력

예제 5.19 search\_type=query\_then\_fetch로 검색

```

curl 'localhost:9200/books/\_search?size=1&q=author:William&search\_type=query\_then\_fetch&fields=title,author&pretty'

```

예제 5.20 search\_type=query\_and\_fetch로 검색

```

curl 'localhost:9200/books/\_search?size=1&q=author:William&search\_type=query\_and\_fetch&fields=title,author&pretty'

```

예제 5.21 search\_type=scan으로 검색

```

curl 'localhost:9200/books/\_search?q=author:william&fields=title,author&search\_type=scan&scroll=10m&pretty'

```

예제 5.22 \_scroll\_id로 검색한 결과 출력

```

curl 'localhost:9200/\_search/scroll/c2Nhbjs1OzUxOnpnazlUWENRUTJ5TURiU3BfUGlLQXc7NTI6emdrOVRYQ1FRMnlNRGJTcF9QaUtBdzs1Mzp6Z2s5VFhDUVEyeU1EYlNwX1BpS0F3OzUxOjhsZlJpemZIVDQ2R1JvZjR3SHdWcmc7NTI6OGxmUml6ZkhUNDZHUm9mNHdId1ZyZzsxO3RvdGFsX2hpdHM6Njs=?pretty'

```

##5.3 리퀘스트 바디 검색

URI검색보다 더복잡한 형싱으로 검색할수있다

예제 5.23 리퀘스트 바디로 author 값이 william인 값 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"term" : { "author" : "william" }

}

}'

```

-------

###5.3.1 size, from, fields

예제 5.24 from:1, size:2, fields:[“title”,”category”] 조건으로 전체 필드에서 time 검색

```

curl 'localhost:9200/\_search?pretty' -d '

{

from : 1,

size : 2,

fields : ["title","category"],

"query" : {

"term" : { "\_all" : "time" }

}

}'

```

매겨변수를 옵션으로 지정할수 있다 참고로 쿼리에서 전체 필드를 검색하려면 필드명에 \_all을 입력한다.

전체인덱스에서 전체 필드를 대상으로 검색어 time으로 검색한다. from을 1옵션으로 검색된 두번째 값부터 size2로 2까지 데이터 출력하며 fields옵션을 이용해 title, category필드만 출력하게 한다

###5.3.2 sort

예제 5.25 category - 내림차순, pages, title - 오름차순 순서로 검색 결과 정렬

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"fields" : ["title","author","category","pages"],

"sort" : [{"category":"desc"},"pages","title"],

"query" : {

"term" : { "\_all" : "time" }

}

}'

```

예제 5.26 category mode: min으로 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"fields": ["title","author","category","pages"],

"sort": [{"category":{"order":"desc","mode":"min"}},"pages","title"],

"query": {

"term": { "category" : "science" }

}

}'

```

예제 5.27 category mode: max로 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"fields": ["title","author","category","pages"],

"sort": [{"category":{"order":"desc","mode":"max"}},"pages","title"],

"query": {

"term": { "category" : "science" }

}

}'

```

예제 5.28 title, author 필드로 정렬. author 필드가 없어서 검색 실패

```

curl 'localhost:9200/\_search?pretty' -d '

{

"fields" : ["title","author","category"],

"sort" : ["title","author"],

"query" : {

"term" : { "title" : "time" }

}

}'

```

예제 5.29 ignore\_unmapped를 true로 설정. author 필드가 없이도 검색 성공

```

curl 'localhost:9200/\_search?pretty' -d '

{

"fields" : ["title","author","category"],

"sort" : ["title",{"author":{"ignore\_unmapped" : true}}],

"query" : {

"term" : { "title" : "time" }

}

}'

```

예제 5.30 track\_scores를 true로 설정. 점수 정보 표시

```

curl 'localhost:9200/\_search?pretty' -d '

{

"fields" : ["title","author","category"],

"sort" : ["title",{"author":{"ignore\_unmapped" : true}}],

"track\_scores": true,

"query" : {

"term" : { "title" : "time" }

}

}'

```

###5.3.3 \_source

예제 5.31 \_source: false로 설정해 도큐먼트 내용은 보이지 않게 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"\_source": false,

"query": {

"term": { "author": "william" }

}

}'

```

예제 5.32 \_source를 이용해서 title과 c로 시작하는 필드명의 값 표시

```

curl 'localhost:9200/magazines/\_search?pretty' -d '

{

"\_source": ["title","c\*"]

}'

```

예제 5.33 c로 시작하면서 ry로 끝나지 않는 필드 표시

```

curl 'localhost:9200/magazines/\_search?pretty' -d '

{

"\_source": {

"include": "c\*",

"exclude" : "\*ry"

}

}'

```

###5.3.4 partial\_fields, fielddata\_fields

예제 5.34 title, category 필드 출력

```

curl 'localhost:9200/magazines/\_search?pretty' -d '

{

"fields":["title","category"]

}'

```

예제 5.35 c로 시작하면서 ry로 끝나지 않는 필드 표시

```

curl 'localhost:9200/magazines/\_search?pretty' -d '

{

"partial\_fields" : {

"partial\_1" : {

"include" : "c\*",

"exclude" : "\*ry"

}

}

}'

```

예제 5.36 fielddata\_fields를 사용해서 title, category 필드 출력

```

curl 'localhost:9200/magazines/\_search?pretty' -d '

{

"fielddata\_fields" : ["title", "category"]

}'

```

###5.3.5 highlight

예제 5.37 author 필드의 검색어 william 강조

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"term" : { "author" : "william" }

},

"highlight" : {

"fields" : { "author" : {} }

}

}'

```

예제 5.38 strong 태그를 이용해 author 필드의 검색어 william 강조

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"term" : { "author" : "william" }

},

"highlight" : {

"pre\_tags" : ["<strong>"],

"post\_tags" : ["</strong>"],

"fields" : { "author" : { } }

}

}'

```

등등..

파이셋과 어그리게이션

데이터를 저장하고 불러오는 기능만 있는것이 아니라.

값의 카운트, 합계, 평균등 처리 기능이 있다 DBMS도 검색조건을 group by 해서 그룹함수를사용하듯 할수있다.

다양한 연산을 적용해서 출력하는것을 파이셋이라고 한다

더욱 다양한 기능을 제공하는 어그리게시녀 모듈이 추가되었다.

엘라스틱 서치에서 검색을 수행하면 검색된 도큐먼트가 hits 필드에 도큐먼트 배열로 나타난다 페이셋으로 검색을하면 검색시 입력한 조건에 대한 각 결과의 개수를 확인할수 있으며 이에 대한 내용은 facets필드에 출력된다.

데이타터를 입력 하기전에 맵핑을 우선 해줘야한다.

|  |  |
| --- | --- |
| curl -XPUT http://localhost:9200/hotels/ -d '  {  "mappings" : {  "hotel" : {  "properties" : {  "name" : { "type" : "string" },  "stars" : { "type" : "long" },  "rooms" : { "type" : "long" },  "location" : { "type" : "geo\_point" },  "city" : { "type" : "string" },  "address" : { "type" : "string" },  "internet" : { "type" : "boolean" },  "service" : { "type" : "string", "index" : "not\_analyzed" },  "checkin": { "type" : "date" , "format" : "dateOptionalTime"}  }  }  }  }' | {"acknowledged":true,"shards\_acknowledged":true} |

벌크이용하여 데이터입력

예제 6.2 hotels 인덱스 데이터 입력

```

curl -XPOST localhost:9200/\_bulk --data-binary @6\_1\_hotels.json

```

###6.1.1 텀 페이셋

예제 6.3 service 필드의 텀 페이셋 검색

```

$ curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"query" : {

"term" : { "name" : "seoul" }

},

"facets" : {

"term\_service" : {

"terms" : {

"field" : "service"

}

}

}

}'

```

예제 6.4 상위 3개의 텀 페이셋 검색

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"query" : {

"term" : { "name" : "seoul" }

},

"facets" : {

"term\_service" : {

"terms" : {

"field" : "service",

"size" : 3

}

}

}

}'

```

예제 6.5 페이셋을 텀의 알파벳 오름차순으로 표시

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"query" : {

"term" : { "name" : "seoul" }

},

"facets" : {

"term\_service" : {

"terms" : { "field" : "service", "order" : "term" }

}

}

}'

```

###6.1.2 범위 페이셋

예제 6.6 stars 필드의 값을 범위별로 페이셋으로 표시

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"range\_stars" : {

"range" : {

"field" : "stars",

"ranges" : [ {"to": 3}, {"from": 3, "to": 5}, {"from": 5} ]

}

}

}

}'

```

예제 6.6 stars 필드의 값을 범위별로 페이셋으로 표시 - 2

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"range\_service" : {

"range" : {

"stars" : [ {"to": 3}, {"from": 3, "to": 5}, {"from": 5} ]

}

}

}

}'

```

예제 6.7 stars 필드로 간격을 구분하고 price 값을 집계한 페이셋

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"range\_stars" : {

"range" : {

"key\_field" : "stars",

"value\_field" : "price",

"ranges" : [ {"to": 3}, {"from": 3, "to": 5}, {"from": 5} ]

}

}

}

}'

```

###6.1.3 히스토그램 페이셋

예제 6.8 rooms 필드의 값을 100 간격으로 구분한 페이셋

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"histo\_rooms" : {

"histogram" : {

"field" : "rooms",

"interval" : 100

}

}

}

}'

```

예제 6.9 rooms 필드를 100 간격으로 구분하고 price 값을 집계한 페이셋

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"histo\_rooms" : {

"histogram" : {

"key\_field" : "rooms",

"value\_field" : "price",

"interval" : 100

}

}

}

}'

```

###6.1.4 날짜 히스토그램 페이셋

예제 6.10 checkin 필드의 값을 1개월 간격으로 구분한 페이셋

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"histo\_checkin" : {

"date\_histogram" : {

"field" : "checkin",

"interval" : "month"

}

}

}

}'

```

###6.1.5 필터와 질의 페이셋

예제 6.11 텀 페이셋 결과에 service : Spa 필터 적용

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"term\_name" : {

"terms" : { "field" : "service" },

"facet\_filter" : {

"term" : { "service" : "Spa" }

}

}

}

}'

```

예제 6.12 필터 페이셋을 사용해서 service : Spa인 텀 검색

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"term\_filter" : {

"filter" : {

"term" : { "service" : "Spa" }

}

}

}

}'

```

예제 6.13 질의 페이셋을 사용해서 name이 seoul 또는 hotel인 값 검색

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"term\_query" : {

"query" : {

"match" : { "name" : "seoul OR hotel" }

}

}

}

}'

```

###6.1.6 통계 페이셋

예제 6.14 통계 페이셋을 사용해서 price 필드 정보 출력

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"my\_facet" : {

"statistical" : { "field" : "price" }

}

}

}'

```

예제 6.15 stars 필드별로 price 필드 정보 출력

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"my\_facet" : {

"terms\_stats" : {

"key\_field" : "stars",

"value\_field" : "price"

}

}

}

}'

```

###6.1.7 위치 거리 페이셋

예제 6.16 geo\_distance 페이셋을 이용해 3km, 6km 간격으로 호텔 분리

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"geo" : {

"geo\_distance" : {

"location" : {

"lat" : 37.52,

"lon" : 126.98

},

"ranges" : [

{ "to" : 3000 },

{ "from" : 3000, "to" : 6000 },

{ "from" : 6000, "to" : 9000 },

{ "from" : 9000 }

]

}

}

}

}'

```

예제 6.17 거리 간격별 price 필드 값 계산

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"geo" : {

"geo\_distance" : {

"location" : {

"lat" : 37.52,

"lon" : 126.98

},

"value\_field" : "price",

"ranges" : [

{ "to" : 3000 },

{ "from" : 3000, "to" : 6000 },

{ "from" : 6000, "to" : 9000 },

{ "from" : 9000 }

]

}

}

}

}'

```

예제 6.18 km 단위를 이용해서 거리 간격별 price 필드 값 계산

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"facets" : {

"geo" : {

"geo\_distance" : {

"location" : {

"lat" : 37.52,

"lon" : 126.98

},

"value\_field" : "price",

"ranges" : [

{ "to" : 3 },

{ "from" : 3, "to" : 6 },

{ "from" : 6, "to" : 9 },

{ "from" : 9 }

],

"unit" : "km"

}

}

}

}'

```

##6.2 어그리게이션

###6.2.1 최소, 최대, 합, 평균, 개수 어그리게이션

예제 6.19 price 필드의 최소(min) 값

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"price\_min" : {

"min" : { "field" : "price" }

}

}

}'

```

예제 6.20 price 필드의 최대(max) 값

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"price\_max" : {

"max" : { "field" : "price" }

}

}

}'

```

예제 6.21 price 필드의 합(sum)

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"price\_sum" : {

"sum" : { "field" : "price" }

}

}

}'

```

예제 6.22 price 필드의 평균(avg)

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"price\_avg" : {

"avg" : { "field" : "price" }

}

}

}'

```

예제 6.23 price 필드의 개수(value\_count)

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"price\_cnt" : {

"value\_count" : { "field" : "price" }

}

}

}'

```

###6.2.2 상태, 확장 상태 어그리게이션

예제 6.24 price 필드의 상태 정보

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"price\_stats" : {

"value\_stats" : { "field" : "price" }

}

}

}'

```

예제 6.25 price 필드의 확장 상태 정보

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"price\_ex\_stats" : {

"extended\_stats" : { "field" : "price" }

}

}

}'

```

###6.2.3 글로벌 어그리게이션

예제 6.26 name 필드에 seoul을 포함한 도큐먼트의 평균 price

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"query" : {

"term" : { "name" : "seoul" }

},

"aggs" : {

"avg\_price" : {

"avg" : { "field" : "price" }

}

}

}'

```

예제 6.27 global 어그리게이션 버킷에 담긴 도큐먼트의 평균 price

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"query" : {

"term" : { "name" : "seoul" }

},

"aggs" : {

"all\_price" : {

"global" : {},

"aggs" : {

"avg\_price" : {

"avg" : { "field" : "price" }

}

}

}

}

}'

```

###6.2.4 필터, 누락 어그리게이션

예제 6.28 필터 어그리게이션 버킷에 담긴 도큐먼트의 price 평균

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"filter\_name" : {

"filter" : {

"term" : { "name" : "seoul" }

},

"aggs" : {

"avg\_price" : {

"avg" : { "field" : "price" }

}

}

}

}

}'

```

예제 6.29 service 필드가 존재하지 않는 도큐먼트의 price 평균

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"missing\_service" : {

"missing" : { "field" : "service" },

"aggs" : {

"avg\_price" : {

"avg" : { "field" : "price" }

}

}

}

}

}'

```

###6.2.5 텀 어그리게이션

예제 6.30 stars 필드값의 텀별로 price 필드 값 평균 계산

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"term\_stars" : {

"terms" : { "field" : "stars" },

"aggs" : {

"avg\_price" : {

"avg" : { "field" : "price" }

}

}

}

}

}'

```

예제 6.31 텀 명으로 내림차순 정렬

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"term\_stars" : {

"terms" : {

"field" : "stars",

"order" : { "\_term" : "desc" }

},

"aggs" : {

"avg\_price" : {

"avg" : { "field" : "price" }

}

}

}

}

}'

```

예제 6.32 avg\_price 하위 어그리게이션 값으로 오름차순 정렬

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"term\_stars" : {

"terms" : {

"field" : "stars",

"order" : { "avg\_price" : "asc" }

},

"aggs" : {

"avg\_price" : {

"avg" : { "field" : "price" }

}

}

}

}

}'

```

###6.2.6 범위, 날짜 범위 어그리게이션

예제 6.33 rooms 필드의 0~500, 500~1000, 1000~ 간격으로 price 평균값 계산

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"range\_room" : {

"range" : {

"field" : "rooms",

"ranges" : [{"to":500}, {"from":500, "to":1000}, {"from":1000}]

},

"aggs" : {

"avg\_price" : {

"avg" : { "field" : "price" }

}

}

}

}

}'

```

예제 6.34 range 어그리게이션 버킷에 키 속성 추가

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"range\_room" : {

"range" : {

"field" : "rooms",

"keyed" : true,

"ranges" : [{"to":500}, {"from":500, "to":1000}, {"from":1000}]

},

"aggs" : {

"avg\_price" : {

"avg" : { "field" : "price" }

}

}

}

}

}'

```

예제 6.35 date\_range 어그리게이션을 이용해서 날짜 범위 구분

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"date\_r\_checkin" : {

"date\_range" : {

"field" : "checkin",

"format" : "yyyy-MM-dd",

"ranges" : [ { "to" : "now-4M" }, { "from" : "now-4M" } ]

}

}

}

}'

```

예제 6.36 date\_range 어그리게이션을 이용해서 날짜 범위 구분

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"date\_r\_checkin" : {

"date\_range" : {

"field" : "checkin",

"format" : "yyyy-MM-dd",

"ranges" : [ { "to" : "2014-03-05T12:30:45" }, { "from" : "2014-03-05T12:30:45" } ]

}

}

}

}'

```

###6.2.7 히스토그램, 날짜 히스토그램 어그리게이션

예제 6.37 rooms 필드를 500 간격으로 구분한 히스토그램 어그리게이션

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"histo\_rooms" : {

"histogram" : {

"field" : "rooms",

"interval" : 500

}

}

}

}'

```

예제 6.38 min\_doc\_count: 0으로 지정해서 값이 없는 구간 표시

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"histo\_rooms" : {

"histogram" : {

"field" : "rooms",

"interval" : 300,

"min\_doc\_count" : 0

}

}

}

}'

```

예제 6.39 stats 하위 어그리게이션의 sum 값을 오름차순으로 정렬

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"histo\_rooms" : {

"histogram" : {

"field" : "rooms",

"interval" : 300,

"order" : { "state\_rooms.sum" : "asc" }

},

"aggs" : {

"state\_rooms" : {

"stats" : { }

}

}

}

}

}'

```

예제 6.40 날짜 히스토그램을 이용해 1개월 간격으로 checkin 필드 구분

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"histo\_checkin" : {

"date\_histogram" : {

"field" : "checkin",

"interval" : "1M",

"format" : "yyyy-MM-dd E"

}

}

}

}'

```

###6.2.8 위치 거리, 위치 해시 그리드 어그리게이션

예제 6.41 3km 간격으로 위치 거리 어그리게이션 버킷 구분

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"geo\_location" : {

"geo\_distance" : {

"field" : "location",

"origin" : "37.52, 126.98",

"distance\_type" : "plane",

"unit" : "km",

"ranges" : [{"to":3},{"from":3,"to":6},{"from":6,"to":9},{"from":9}]

}

}

}

}'

```

예제 6.42 precision: 5 정밀도 값으로 위치 해시 그리드 구분

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"geohash\_location" : {

"geohash\_grid" : {

"field" : "location",

"precision" : 5

}

}

}

}'

```

예제 6.43 위도: 37~38 / 경도: 126~128에 속하는 도큐먼트 필터링

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"aggs" : {

"geohash\_filter" : {

"filter" : {

"geo\_bounding\_box" : {

"location" : {

"top\_left" : "38.00, 126.00",

"bottom\_right" : "37.00, 127.00"

}

}

},

"aggs" : {

"geoghash\_location" : {

"geohash\_grid" : {

"field" : "location",

"precision" : 5

}

}

}

}

}

}'

```

질의queryDSL

#7. 질의(QueryDSL)

##7.1 쿼리

###7.1.1 텀, 텀즈 쿼리

예제 7.1 title 필드 값이 prince인 도큐먼트를 텀 쿼리로 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"term" : {

"title" : "prince"

}

}

}'

```

예제 7.2 Prince로 검색하면 저장된 텀과 다르므로 결과에 나타나지 않음

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"term" : {

"title" : "Prince"

}

}

}'

```

예제 7.3 텀즈 쿼리를 이용해서 prince, king 텀 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"terms" : {

"title" : ["prince","king"]

}

}

}'

```

예제 7.4 the, and, of 3개의 텀 중 2개 이상을 포함하는 도큐먼트 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"terms" : {

"title" : ["the","and","of"],

"minimum\_should\_match" : 2

}

}

}'

```

###7.1.2 매치, 다중 매치(multi match) 쿼리

예제 7.5 title 필드를 The And로 매치 쿼리 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"match" : {

"title" : "The And"

}

}

}'

```

예제 7.6 operator를 이용해서 검색 조건을 and로 변경

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"match" : {

"title" : {

"query" : "The And",

"operator" : "and"

}

}

}

}'

```

예제 7.7 analyzer를 이용해서 질의문에 공백 형태소 적용

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"match" : {

"title" : {

"query" : "prince king",

"analyzer" : "whitespace"

}

}

}

}'

```

예제 7.8 type: phrase를 사용해 and the 구문을 포함하는 도큐먼트 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"match" : {

"title" : {

"query" : "and the",

"type" : "phrase"

}

}

}

}'

```

예제 7.9 다중 매치 쿼리를 이용해서 prince king을 title, plot 필드에서 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"multi\_match" : {

"fields" : [ "title" , "plot" ],

"query" : "prince king"

}

}

}'

```

###7.1.3 불 쿼리

예제 7.10 불 쿼리를 이용한 다중 조건 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"bool" : {

"must" : {

"term" : { "title" : "the" }

},

"must\_not" : {

"term" : { "plot" : "prince" }

},

"should" : [

{ "term" : { "title" : "time" } },

{ "term" : { "title" : "world" } }

]

}

}

}'

```

###7.1.4 문자열 쿼리

예제 7.11 문자열 쿼리로 title 필드에서 prince 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"query\_string" : {

"query" : "title:prince"

}

}

}'

```

예제 7.12 default\_filed, default\_operator로 문자열 쿼리 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"query\_string" : {

"query" : "prince king",

"default\_field" : "plot",

"default\_operator" : "and"

}

}

}'

```

###7.1.5 접두어 쿼리

예제 7.13 접두어 쿼리로 질의어 prin 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"prefix" : {

"title" : "prin"

}

}

}'

```

###7.1.6 범위 쿼리

예제 7.14 범위 쿼리로 pages 필드가 50~149인 값 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"range" : {

"pages" : { "gte" : 50, "lt" : 150 }

}

}

}'

```

예제 7.15 범위 쿼리로 written 필드가 1600-01-01 ~ 1699-12-31인 값 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"range" : {

"written" : {

"gte" : "1600-01-01",

"lt" : "1699-12-31"

}

}

}

}'

```

###7.1.7 전체 매치 쿼리

예제 7.16 전체 매치 쿼리

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"match\_all" : {}

}

}'

```

###7.1.8 퍼지 쿼리

예제 7.17 퍼지 쿼리를 이용한 질의어 tree 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"fuzzy" : {

"title" : "tree"

}

}

}'

```

예제 7.18 퍼지 쿼리를 이용해 page 필드가 80~120인 값 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"fuzzy" : {

"pages" : {

"value" : 100,

"fuzziness" : 20

}

}

}

}'

```

##7.2 필터

###7.2.1 텀, 텀즈 필터

예제 7.19 텀 필터를 이용해서 title 필드가 prince인 값 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"filter" : {

"term" : {

"title" : "prince"

}

}

}'

```

예제 7.20 텀즈 필터를 이용해서 title 필드가 prince, king 인 값 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"filter" : {

"terms" : {

"title" : ["prince", "king"]

}

}

}'

```

예제 7.21 title 필드가 and, the인 값을 execution: and로 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"filter" : {

"terms" : {

"title" : ["and", "the"],

"execution" : "and"

}

}

}'

```

###7.2.2 범위 필터

예제 7.22 범위 필터로 pages 필드가 50~149인 값 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"filter" : {

"range" : {

"pages" : { "gte" : 50, "lt" : 150 }

}

}

}'

```

###7.2.3 and, or, not 필터

예제 7.23 not 필터로 pages 필드가 50~149가 아닌 도큐먼트 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"filter" : {

"not" : {

"range" : {

"pages" : { "gte" : 50, "lt" : 150 }

}

}

}

}'

```

예제 7.24 and 필터로 pages 필드가 50~149이면서 title: the인 값 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"filter" : {

"and" : [

{

"range" : {

"pages" : { "gte" : 50, "lt" : 150 }

}

},

{

"term" : { "title" : "the" }

}

]

}

}'

```

###7.2.4 불 필터

예제 7.25 불 필터를 이용한 다중 조건 검색

```

curl 'localhost:9200/books/\_search?pretty' -d '

{

"filter" : {

"bool" : {

"must" : {

"term" : { "title" : "the" }

},

"must\_not" : {

"term" : { "plot" : "prince" }

},

"should" : [

{ "term" : { "title" : "time" } },

{ "term" : { "title" : "world" } }

]

}

}

}'

```

###7.2.5 위치 필터

예제 7.26 geo\_bounding\_box 필터로 사각형 영역의 값을 검색

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"filter" : {

"geo\_bounding\_box" : {

"location" : {

"top\_left" : { "lat" : 38.00, "lon" : 126.00 },

"bottom\_right" : { "lat" : 37.00, "lon" : 127.00 }

}

}

}

}'

```

예제 7.27 geo\_distance 필터로 위도 경도 37.52, 126.98을 중심으로 5km 이내에 있는 도큐먼트 검색

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"filter" : {

"geo\_distance" : {

"distance" : "5km",

"location" : { "lat" : 37.52, "lon" : 126.98 }

}

}

}'

```

예제 7.28 geo\_distance\_range 필터로 5km~10km 사이에 있는 도큐먼트 검색

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"filter" : {

"geo\_distance\_range" : {

"from" : "5km",

"to" : "10km",

"location" : { "lat" : 37.52, "lon" : 126.98 }

}

}

}'

```

예제 7.29 geo\_polygon 필터로 삼각형 영역 사이의 값 검색

```

curl 'localhost:9200/hotels/\_search?pretty' -d '

{

"filter" : {

"geo\_polygon" : {

"location" : {

"points" : [

{ "lat" : 38.00, "lon" : 127.00 },

{ "lat" : 37.00, "lon" : 127.00 },

{ "lat" : 38.00, "lon" : 128.00 }

]

}

}

}

}'

```

맵핑

데이터 저장 형태와 검색 엔진에서 해당하는 데이터에 어떻게 접근하고 처리하는지에 대한 명세이다.

맵핑API (\_mapping)

|  |  |
| --- | --- |
| curl 'http://localhost:9200/books/\_mapping?pretty' | {  "books" : {  "mappings" : {  "book" : {  "properties" : {  "author" : {  "type" : "text",  "fields" : {  "keyword" : {  "type" : "keyword",  "ignore\_above" : 256  }  }  },  "category" : {  "type" : "text",  "fields" : {  "keyword" : {  "type" : "keyword",  "ignore\_above" : 256  }  }  },  "date" : {  "type" : "date"  },  "pages" : {  "type" : "long"  },  "title" : {  "type" : "text",  "fields" : {  "keyword" : {  "type" : "keyword",  "ignore\_above" : 256  }  }  }  }  }  }  }  } |
| 예제 8.2 books/book 인덱스/타입에 read 필드 추가  ```  curl -XPUT 'http://localhost:9200/books/\_mapping/book' -d '  {  "book" : {  "properties" : {  "read" : { "type" : "boolean" }  }  }  }'  ``` | {"acknowledged":true} |

##8.2 내장필드

메타정보를 정의하고 데이터 처리 방법을 설정할수 있다.

ㅇ메타정보를 설정하기 위한 이 내부 기능들은 공식적으로는 필드 라고 불리고 있으며 관계형 데이터베이스의 컬럼에 대응되는 요소인 필드와 구분하기위해서 이책에서는 내장필드라고 말한다.

|  |  |
| --- | --- |
| 예제 8.4 인덱스 생성 시 내장필드 설정  ```  PUT <호스트>/<인덱스> -d '  {  "mappings" : {  "<타입명>" : {  "<내장필드명>" : {  … <필드 내용> …  }  }  }  }'  ``` | 예제 8.5 인덱스 생성 후 내장필드 설정  ```  PUT <호스트>/<인덱스>/\_mapping/<타입> -d '  {  "<타입명>" : {  "<내장필드명>" : {  … <필드 내용> …  }  }  }'  ``` |
|  |  |

맵핑변경하면 다른얘들 데이터 컬럼들 날라감

예제 8.7 books 인덱스의 id를 title 필드로 설정한 뒤 데이터 입력

```

curl -XDELETE 'http://localhost:9200/books'

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_id" : { "path" : "title" }

}

}

}'

curl -XPOST localhost:9200/\_bulk --data-binary @5\_1\_books.json

```

예제 8.8 새로 입력된 데이터 검색

```

curl 'http://localhost:9200/books/\_search?pretty'

```

예제 8.9 아이디를 이용해서 도큐먼트 조회

```

curl 'http://localhost:9200/books/book/King%20Lear?pretty'

```

###8.2.2 \_source

예제 8.10 books 인덱스의 \_source 필드를 enabled : false로 설정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_source" : {"enabled" : false}

}

}

}'

```

예제 8.11 검색어 prince로 books 인덱스 검색

```

curl 'http://localhost:9200/books/\_search?q=prince&pretty'

```

예제 8.12 title, author, category 필드만 저장하도록 지정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_source" : {

"includes" : ["title", "author", "category"]

}

}

}

}'

```

예제 8.13 검색어 prince로 검색

```

curl 'http://localhost:9200/books/\_search?q=prince&pretty'

```

예제 8.14 p로 시작하는 필드는 저장하지 않도록 지정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_source" : {

"excludes" : ["p\*"]

}

}

}

}'

```

예제 8.15 검색어 prince로 검색

```

curl 'http://localhost:9200/books/\_search?q=prince&pretty'

```

###8.2.3 \_all

예제 8.16 \_all 내장필드를 이용해서 plot 필드는 검색하지 않도록 지정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_all" : { "enabled" : true },

"properties" : {

"title" : {

"include\_in\_all" : true,

"type" : "string"

},

"plot" : {

"include\_in\_all" : false,

"type" : "string"

}

}

}

}

}'

예제 8.11 검색어 prince로 books 인덱스 검색

```

curl 'http://localhost:9200/books/\_search?q=prince&pretty'

```

예제 8.12 title, author, category 필드만 저장하도록 지정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_source" : {

"includes" : ["title", "author", "category"]

}

}

}

}'

```

예제 8.13 검색어 prince로 검색

```

curl 'http://localhost:9200/books/\_search?q=prince&pretty'

```

예제 8.14 p로 시작하는 필드는 저장하지 않도록 지정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_source" : {

"excludes" : ["p\*"]

}

}

}

}'

```

예제 8.15 검색어 prince로 검색

```

curl 'http://localhost:9200/books/\_search?q=prince&pretty'

```

###8.2.3 \_all

검색이 수행될때 검색할 대상 필드를 지정할수 있다 매핑을 설정할때 먼저 \_all 의 enabled옵션을 truen로 처리한다

예제 8.16 \_all 내장필드를 이용해서 plot 필드는 검색하지 않도록 지정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_all" : { "enabled" : true },

"properties" : {

"title" : {

"include\_in\_all" : true,

"type" : "string"

},

"plot" : {

"include\_in\_all" : false,

"type" : "string"

}

}

}

}

}'

```

예제 8.17 검색어 prince로 검색

```

curl 'http://localhost:9200/books/\_search?q=prince&pretty'

```

###8.2.4 \_analyzer

트정 필드에 저장된 값을 이용해서 해당 도큐먼트가 색인될때 사용할 분석기를 설정한다 path옵션을 사용해서 분서긱가 저장된 필드를 지정한다

예제 8.18 analyze\_value 필드의 값을 분석기로 설정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_analyzer" : { "path" : " analyze\_value" }

}

}

}'

```

###8.2.5 \_timestamp

데이터가 입력된 시간의 타임스탬프를 저장할수 있다. 타임스탬프를 저장하려면 enabled와 store옵션을 모두 true로

예제 8.19 \_timestamp 내장필드를 이용해서 타임스탬프 저장

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_timestamp" : {

"enabled" : true,

"store" : true

}

}

}

}'

```

예제 8.20 검색어 prince로 title 필드와 \_timestamp 필드 검색

```

curl -XGET 'http://localhost:9200/books/\_search?pretty' -d '

{

"fields" : [ "title" , "\_timestamp" ],

"query" : {

"term" : { "\_all" : "prince" }

}

}'

```

###8.2.6 \_ttl(time to live)

입력데이터 유지 시간을 설정한다.

예제 8.21 \_ttl 내장필드를 이용해서 2일 뒤 데이터가 삭제되도록 설정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"\_ttl" : {

"enabled" : true,

"default" : "2d"

}

}

}

}'

```

##8.3 데이터 타입

문자열, 정수 실수 불린과같은 기본타입과 객체 타입같은 확장타입이이 있다.

###8.3.1 문자열

예제 8.22 필드 옵션을 적용해서 books 인덱스 매핑 생성

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"properties" : {

"title" : { "type" : "string", "boost" : 2.0 },

"category" : { "type" : "string", "index" : "not\_analyzed" }

}

}

}

}'

```

예제 8.23 title 필드에 boost 2.0 적용 후 prince로 검색

```

curl 'http://localhost:9200/books/\_search?q=prince&pretty'

```

예제 8.24 category 필드를 science로 검색

```

curl 'http://localhost:9200/books/\_search?q=category:science&pretty'

```

예제 8.25 category 필드를 Science fiction으로 검색

```

curl 'http://localhost:9200/books/\_search?q=category:Science%20fiction&pretty'

```

###8.3.2 숫자

예제 8.26 num\_val 필드의 ignore\_malformed 옵션을 true로 설정

```

curl -XPUT 'http://localhost:9200/test\_nums' -d '

{

"mappings" : {

"test\_num" : {

"properties" : {

"num\_val" : { "type" : "integer", "ignore\_malformed" : true }

}

}

}

}'

```

예제 8.27 num\_val 필드 값에 hello 입력 후 확인

```

curl -XPUT 'http://localhost:9200/test\_nums/test\_num/1' -d '

{

"num\_val" : "hello"

}'

curl 'http://localhost:9200/test\_nums/test\_num/1'

```

예제 8.28 num\_val 필드의 통계 페이셋 확인

```

curl 'localhost:9200/test\_nums/\_search?pretty' -d '

{

"facets" : {

"num\_stat" : {

"statistical" : { "field" : "num\_val" }

}

}

}'

```

###8.3.3 날짜

###8.3.4 불린

###8.3.5 바이너리

###8.3.6 객체

예제 8.29 name, age, married 필드를 가지는 user 필드 값

```

{

"user" : {

"name" : "Kim",

"age" : 30,

"married" : true

}

}

```

예제 8.30 test\_users 인덱스에 객체 타입의 user 필드 매핑 설정

```

curl -XPUT 'http://localhost:9200/test\_users' -d '

{

"mappings" : {

"test\_user" : {

"properties" : {

"user" : {

"type" : "object",

"properties" : {

"name" : { "type" : "string" },

"age" : { "type" : "integer" },

"married" : { "type" : "boolean" }

}

}

}

}

}

}'

```

예제 8.31 user 필드에 gender 필드 값을 추가로 입력

```

curl -XPUT 'http://localhost:9200/test\_users/test\_user/1' -d '

{

"user" : {

"name" : "Kim", "age" : 30, "married" : true, "gender" : "male"

}

}'

```

예제 8.32 test\_users 인덱스의 매핑 확인

```

curl 'http://localhost:9200/test\_users/\_mapping?pretty'

```

예제 8.33 dynamic 옵션을 false로 설정

```

curl -XPUT 'http://localhost:9200/test\_users' -d '

{

"mappings" : {

"test\_user" : {

"properties" : {

"user" : {

"type" : "object",

"dynamic" : false,

"properties" : {

"name" : { "type" : "string" },

"age" : { "type" : "integer" },

"married" : { "type" : "boolean" }

}

}

}

}

}

}'

```

예제 8.34 test\_users 인덱스의 매핑 확인

```

curl 'http://localhost:9200/test\_users/\_mapping?pretty'

```

예제 8.35 test\_users 인덱스의 매핑 확인

```

curl 'localhost:9200/test\_users/\_search?pretty' -d '

{

"fields" : [ "user.name", "user.age", "user.married", "user.gender" ]

}'

```

###8.3.7 중첩

예제 8.36 user 필드의 name, age, married 하위 필드

```

{

"user" : [

{ "name" : "Kim", "age" : 30, "married" : true },

{ "name" : "Lee", "age" : 28, "married" : false }

]

}

```

예제 8.37 객제 필드의 하위 필드가 실제로 저장되는 형태

```

{

"user.name" : [ "Kim", "Lee" ],

"user.age" : [ 30, 28 ],

"user.married" : [ true, false ]

}

```

예제 8.38 test\_users 인덱스에 user 필드의 하위 필드 데이터 입력

```

curl -XPUT 'http://localhost:9200/test\_users/test\_user/1' -d '

{

"user" : [

{ "name" : "Kim", "age" : 30, "married" : true },

{ "name" : "Lee", "age" : 28, "married" : false }

]

}'

```

예제 8.39 must(AND) 조건으로 user.name : kim, user.age : 28 검색

```

curl 'http://localhost:9200/test\_users/\_search?pretty' -d '

{

"query" : {

"bool" : {

"must" : [

{ "match" : { "user.name" : "kim" } },

{ "match" : { "user.age" : 28 } }

]

}

}

}'

```

예제 8.40 중첩 타입으로 user 필드의 매핑 설정

```

curl -XPUT 'http://localhost:9200/test\_users' -d '

{

"mappings" : {

"test\_user" : {

"properties" : {

"user" : {

"type" : "nested",

"properties" : {

"name" : { "type" : "string" },

"age" : { "type" : "integer" },

"married" : { "type" : "boolean" }

}

}

}

}

}

}'

```

예제 8.41 중첩 필드의 하위 필드들이 실제로 저장되는 형태

```

{

"user.name" : "Kim",

"user.age" : 30,

"user.married" : true

},

{

"user.name" : "Lee",

"user.age" : 28,

"user.married" : false

}

```

예제 8.42 nested 질의문 방법

```

{

"nested" : {

"path" : <중첩 타입의 상위 필드명>,

"score\_mode" : <스코어 방식>,

"query" : {

<질의문>

}

}

}

```

예제 8.43 must(AND) 조건으로 user.name : kim, user.age : 30 검색

```

curl 'localhost:9200/test\_users/\_search?pretty' -d '

{

"query" : {

"nested" : {

"path" : "user",

"score\_mode" : "avg",

"query" : {

"bool" : {

"must" : [

{ "match" : { "user.name" : "kim" } },

{ "match" : { "user.age" : 30 } }

]

}

}

}

}

}'

```

예제 8.44 must(AND) 조건으로 user.name : kim, user.age : 28 검색

```

curl 'localhost:9200/test\_users/\_search?pretty' -d '

{

"query" : {

"nested" : {

"path" : "user",

"score\_mode" : "avg",

"query" : {

"bool" : {

"must" : [

{ "match" : { "user.name" : "kim" } },

{ "match" : { "user.age" : 28 } }

]

}

}

}

}

}'

```

###8.3.8 좌표

예제 8.45 하위 필드 lat, lon의 숫자 값으로 입력

```

{

"location" : {

"lat" : 37.525308,

"lon" : 126.926644

}

}

```

예제 8.46 문자열로 입력. 위도-경도 순으로 입력

```

{

"location" : "37.525308, 126.926644"

}

```

예제 8.47 배열로 입력. 경도-위도 순서로 입력

```

{

"location" : [ 126.926644, 37.525308 ]

}

```

예제 8.48 위치 해시(geohash) 값으로 입력

```

{

"location" : " wydm2r30v0mm"

}

```

예제 8.49 test\_geos 인덱스의 매핑 설정

```

curl -XPUT http://localhost:9200/test\_geos/ -d '

{

"mappings" : {

"test\_geo" : {

"properties" : {

"name" : { "type" : "string" },

"location" : { "type" : "geo\_point" }

}

}

}

}'

```

예제 8.50 name, location 값 입력

```

curl -XPUT 'http://localhost:9200/test\_geos/test\_geo/1' -d '

{

"name" : "Conrad Seoul",

"location" : "37.525308, 126.926644"

}'

```

예제 8.51 location 필드의 위도 37.53~37.52, 경도 126.92~126.93 범위의 값 검색

```

curl 'http://localhost:9200/test\_geos/\_search?pretty' -d '

{

"filter" : {

"geo\_bounding\_box" : {

"location" : {

"top\_left" : { "lat" : 37.53, "lon" : 126.92 },

"bottom\_right" : { "lat" : 37.52, "lon" : 126.93 }

}

}

}

}'

```

예제 8.52 name, location 값 입력

```

curl -XPUT 'http://localhost:9200/test\_geos/test\_geo/1' -d '

{

"name" : "Conrad Seoul",

"location" : "37.525308, 486.926644"

}'

```

예제 8.53 location 필드의 위도 37.53~37.52, 경도 126.92~126.93 범위의 값 검색

```

curl 'http://localhost:9200/test\_geos/\_search?pretty' -d '

{

"filter" : {

"geo\_bounding\_box" : {

"location" : {

"top\_left" : { "lat" : 37.53, "lon" : 126.92 },

"bottom\_right" : { "lat" : 37.52, "lon" : 126.93 }

}

}

}

}'

```

예제 8.54 normalize와 validate 옵션을 false로 매핑 설정

```

curl -XPUT http://localhost:9200/test\_geos/ -d '

{

"mappings" : {

"test\_geo" : {

"properties" : {

"name" : { "type" : "string" },

"location" : {

"type" : "geo\_point",

"normalize" : false,

"validate" : false

}

}

}

}

}'

```

예제 8.55 location 필드의 위도 37.53~37.52, 경도 126.92~126.93 범위의 값 검색

```

curl 'http://localhost:9200/test\_geos/\_search?pretty' -d '

{

"filter" : {

"geo\_bounding\_box" : {

"location" : {

"top\_left" : { "lat" : 37.53, "lon" : 126.92 },

"bottom\_right" : { "lat" : 37.52, "lon" : 126.93 }

}

}

}

}'

```

예제 8.56 normalize와 validate 옵션을 false로 매핑 설정

```

curl -XPUT http://localhost:9200/test\_geos/ -d '

{

"mappings" : {

"test\_geo" : {

"properties" : {

"location" : {

"type" : "geo\_point"

,"geohash\_prefix" : true

,"geohash\_precision" : 10

}

}

}

}

}'

```

예제 8.57 위치 해시값 wydm2r30v0mm으로 검색

```

curl 'localhost:9200/test\_geos/\_search?pretty' -d '

{

"query" : {

"filtered" : {

"query" : {

"match\_all" : {}

},

"filter" : {

"geohash\_cell": { "location": "wydm2r30v0mm" }

}

}

}

}'

```

예제 8.58 위치 해시값 wydm2r30v 로 검색

```

curl 'localhost:9200/test\_geos/\_search?pretty' -d '

{

"query" : {

"filtered" : {

"query" : {

"match\_all" : {}

},

"filter" : {

"geohash\_cell": { "location": "wydm2r30v" }

}

}

}

}'

```

###8.3.9 위치 모형

예제 8.59 location 필드를 geo\_shape 타입으로 매핑정보 설정

```

curl -XPUT http://localhost:9200/test\_geos/ -d '

{

"mappings" : {

"test\_geo" : {

"properties" : {

"location" : {

"type" : "geo\_shape",

"precision": 10

}

}

}

}

}'

```

예제 8.60 점. -77.03653, 38.387676 좌표의 점을 나타냄

```

{

"location" : {

"type" : "point",

"coordinates" : [-77.03653, 38.897676]

}

}

```

예제 8.61 선. 좌표 -77.03653, 38.387676에서부터 -77.009051, 38.889939까지 연결된 선을 나타냄

```

{

"location" : {

"type" : "linestring",

"coordinates" : [[-77.03653, 38.897676], [-77.009051, 38.889939]]

}

}

```

예제 8.62 다각형. corrdinates 값은 2차 배열로 되어 있으며 1번째 배열 값은 좌표들을 연결한 형태의 전체 다각형을, 그리고 2번째 배열 값은 좌표들을 연결한 형태의 구멍이 1번째 배열의 다각형 안에 있음을 나타냄

```

{

"location" : {

"type" : "polygon",

"coordinates" : [

[ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0], [100.0, 1.0], [100.0, 0.0] ],

[ [100.2, 0.2], [100.8, 0.2], [100.8, 0.8], [100.2, 0.8], [100.2, 0.2] ]

]

}

}

```

예제 8.63 다중점. 102.0, 2.0 좌표와 103.0, 2.0 좌표 두 개의 점을 나타냄

```

{

"location" : {

"type" : "multipoint",

"coordinates" : [

[102.0, 2.0], [103.0, 2.0]

]

}

}

```

예제 8.64 다중선. 선 배열을 값으로 가지고 있는 2차 배열

```

{

"location" : {

"type": "MultiLineString",

"coordinates": [

[ [100.0, 0.0], [101.0, 1.0] ],

[ [102.0, 2.0], [103.0, 3.0] ]

]

}

}

```

예제 8.65 다중다각형. 다각형 2차 배열을 값으로 가지고 있는 3차 배열

```

{

"location" : {

"type" : "multipolygon",

"coordinates" : [

[ [[102.0, 2.0], [103.0, 2.0], [103.0, 3.0], [102.0, 3.0], [102.0, 2.0]] ],

[

[[100.0, 0.0], [101.0, 0.0], [101.0, 1.0], [100.0, 1.0], [100.0, 0.0]],

[[100.2, 0.2], [100.8, 0.2], [100.8, 0.8], [100.2, 0.8], [100.2, 0.2]]

]

]

}

}

```

예제 8.66 사각형. 두 좌표를 대칭점으로 하는 사각형이다.

```

{

"location" : {

"type" : "envelope",

"coordinates" : [ [-45.0, 45.0], [45.0, -45.0] ]

}

}

```

예제 8.67 원. coordinates 좌표를 중심으로 radius 반지름만큼의 크기를 가진 원을 나타냄

```

{

"location" : {

"type" : "circle",

"coordinates" : [-45.0, 45.0],

"radius" : "100m"

}

}

```

예제 8.68 컬렉션. geometries 필드에 서로 다른 위치 형태의 값들을 배열로 입력

```

{

"location" : {

"type": "geometrycollection",

"geometries": [

{

"type": "point",

"coordinates": [100.0, 0.0]

},

{

"type": "linestring",

"coordinates": [ [101.0, 0.0], [102.0, 1.0] ]

}

]

}

}

```

##8.4 다중필드

###8.4.1 다중필드

예제 8.69 title 필드의 다중필드로 raw 필드 설정

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"properties" : {

"title" : {

"type" : "string", "index" : "analyzed",

"fields" : {

"raw" : { "type" : "string", "index" : "not\_analyzed"}

}

}

}

}

}

}'

```

예제 8.70 title.raw 필드 검색

```

curl 'http://localhost:9200/books/\_search?pretty' -d '

{

"query" : {

"term" : { "title.raw" : "The Prince and the Pauper" }

}

}'

```

###8.4.2 토큰 수

예제 8.71 title 필드의 토큰 수를 저장하는 tokens 다중 필드

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"properties" : {

"title" : {

"type" : "string", "index" : "analyzed",

"fields" : {

"tokens" : {

"type" : "token\_count",

"store" : true,

"analyzer" : "standard"

}

}

}

}

}

}

}'

```

예제 8.72 title 필드 검색어 the 로 title, title.tokens 필드 출력

```

curl 'http://localhost:9200/books/\_search?pretty' -d '

{

"fields" : ["title", "title.tokens"],

"query" : {

"term" : { "title" : "the" }

}

}'

```

##8.4 필드 복사

예제 8.73 title, author 필드의 값을 pk\_data 필드에 복사

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"properties" : {

"title" : { "type" : "string", "copy\_to" : "pk\_data" },

"author" : { "type" : "string", "copy\_to" : "pk\_data" },

"pk\_data" : { "type" : "string", "store" : true }

}

}

}

}'

```

예제 8.74 title, author, pk\_data 필드 값 확인

```

curl 'http://localhost:9200/books/\_search?pretty' -d '

{

"fields" : ["title", "author", "pk\_data"],

"query" : {

"term" : { "\_all" : "prince" }

}

}'

```

예제 8.75 title 필드의 값을 title\_1, title\_2 필드에 복사

```

curl -XPUT 'http://localhost:9200/books' -d '

{

"mappings" : {

"book" : {

"properties" : {

"title" : { "type" : "string", "copy\_to" : ["title\_1", "title\_2"] },

"title\_1" : { "type" : "string", "store" : true },

"title\_2" : { "type" : "string", "store" : true }

}

}

}

}'

```

예제 8.76 title, title\_1, title\_2 필드 값 확인

```

curl 'http://localhost:9200/books/\_search?pretty' -d '

{

"fields" : ["title", "title\_1", "title\_2"],

"query" : {

"term" : { "\_all" : "prince" }

}

}'

```

###8.2.1 \_id

최소 데이터 도큐먼트의 아이디를 의미

저장 하도록할수도 안할수도 있다 아래 옵션

예제 8.6 \_id 저장하도록 store 옵션 설정.

```

{

"books" : {

"\_id" : {

"index" : "not\_analyzed",

"store" : true

}

}

}

```

어그리게이션 (Aggregations)

* 관계형 데이터베이스의 그룹 처리와 비슷한 기능을 제공
* 기존 페이셋(Facet)의 단점을 보강한 모듈로 페이셋 대체 예정
* 버킷(Bucket) 어그리게이션과 메트릭(Metric) 어그리게이션으로 구분

|  |
| --- |
|  |

메트릭(Metric) 어그리게이션

* 주어진 조건으로 도큐먼트를 계산하여 결과값 도출
* min, max, sum, avg가 대표적

|  |
| --- |
|  |

상태, 확장 상태 어그리게이션

• 메트릭 어그리게이션  
• 상태타입을사용해서계산값을한번에표시  
• 상태정보외에제곱합,변위,표준편차값도확인가능

|  |
| --- |
|  |

버킷(Bucket) 어그리게이션

* 주어진 조건에 해당하는 도큐먼트를 버킷이라는 저장소 단위로 구분
* 새로운 데이터 집합 형성
* 버킷별로 하위 연산을 반복해서 수행 가능
* 레벨이깊어질수록메모리등의자원소모가심하므로주의필요
* 버킷 어그리게이션의 하위 어그리게이션으로는 버킷 혹은 메트릭 어그리게이션 사용 가능
* filter, missing, terms, range, histogram 등이 있음

글로벌(GLOBAL) 어그리게이션

* 모든 도큐먼트를 하나의 버킷에 모두 담는 버킷 어그리게이션
* 질의에 영향을 받지 않음  
  • 한번의검색으로질의내용과별도의어그리게이션을동시사용가능

|  |
| --- |
|  |

필터 어그리게이션

• 필터 어그리게이션은 주어진 필터에 해당하는 도큐먼트 버킷 생성

|  |
| --- |
|  |

누락 어그리게이션

지정한 필드가 존재하지 않거나 필드 값이 null인 도큐먼트 버킷 생성

|  |
| --- |
|  |

범위 어그리게이션

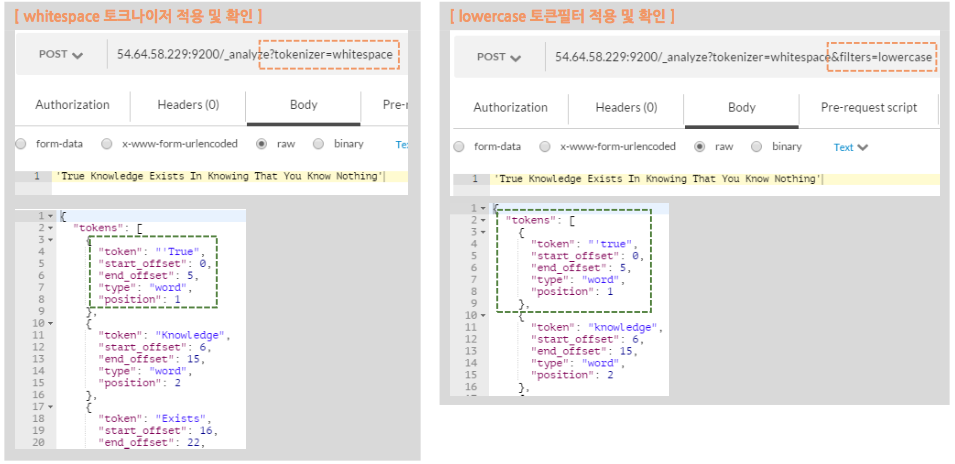
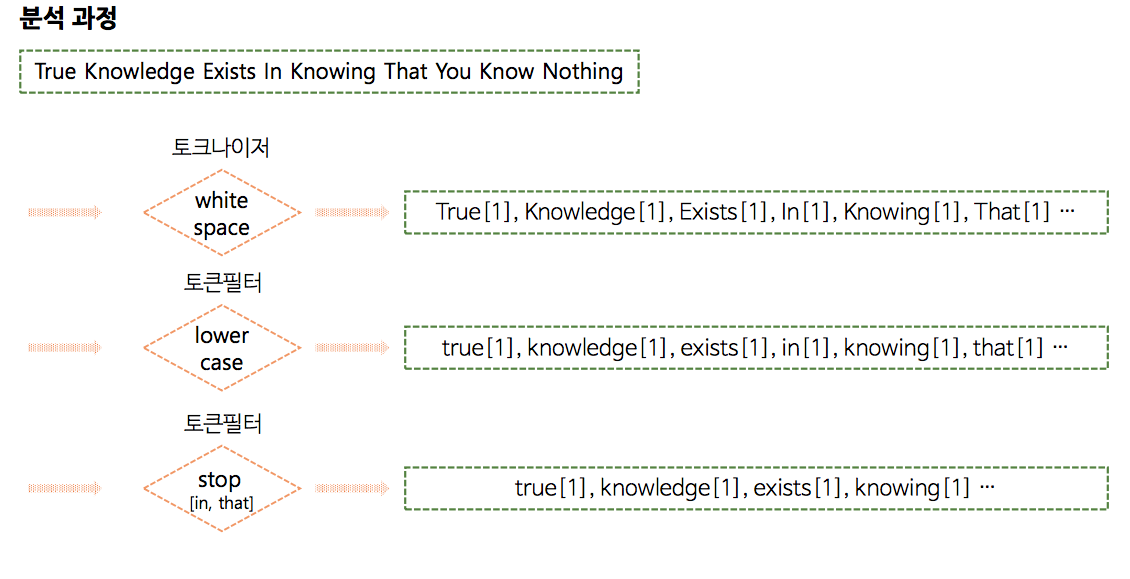
• 설정한값의범위별로버킷을생성할수있는버킷어그리게이션  
• 날짜형식필드값을사용하여날짜범위어그리게이션사용가능  
• 날짜 지정 형식은 반드시 JSON 표준 날짜-시간 입력 형식으로 입력

|  |
| --- |
|  |

분석기

분석

* 검색어를 추출하기 위한 프로세스를 거치는 과정(Analysis)
* 분석 과정에서 사용하는 프로그램을 분석기(Analyzer)라고 함
* 분석기는 토크나이저(Tokenizer)와 토큰필터(Token Filter)로 구성
* 토크나이저는 설정된 기준에 따라 검색어 토큰으로 분리
* 토큰필터는 분리된 토큰에 필터를 적용하여 검색에 쓰이는 검색어로 변환
* 엘라스틱서치는 내부적으로 다양한 분석기를 제공
* 사용자가 직접 분석기를 생성하여 적용 가능



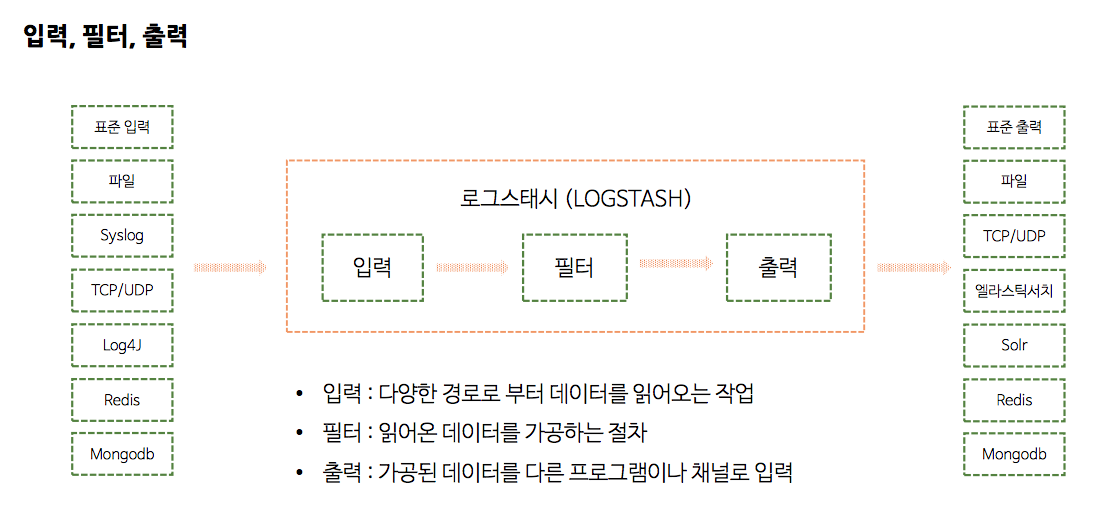
\_analyze API

• 입력한 데이터를 분석한 결과를 확인 가능

|  |
| --- |
|  |

LOGSTASH

* 데이터의 흐름을 관리하기 위해 개발된 오픈 소스 프로젝트
* 엘라스틱서치의 공식 패키지 구성 요소
* 아파치 라이선스 2.0 오픈소스
* JRuvy로 작성 (자바 런타임 환경 필수, 1.7 이상)
* 다양한 방식으로 데이터 입/출력 가능



설정 파일

• 설정은 별도의 파일로 저장  
• 실행시–f옵션과함께사용  
• input, filter, output 순서로 작성하며 내용은 중괄호 안에 입력  
• 입,출력경로가여러개일때에는줄바꿈으로구분,쉼표는입력하지않음

|  |
| --- |
|  |

입출력

* codec 옵션을 사용하여 데이터 형식 지정(기본값은 Plain)
* 입력 데이터의 변화를 로그스태시가 계속 감시

|  |
| --- |
|  |
|  |

필터

• 출력결과를변경할수있는다양한필터를제공 • 조건문을사용하여특정조건별필터링가능

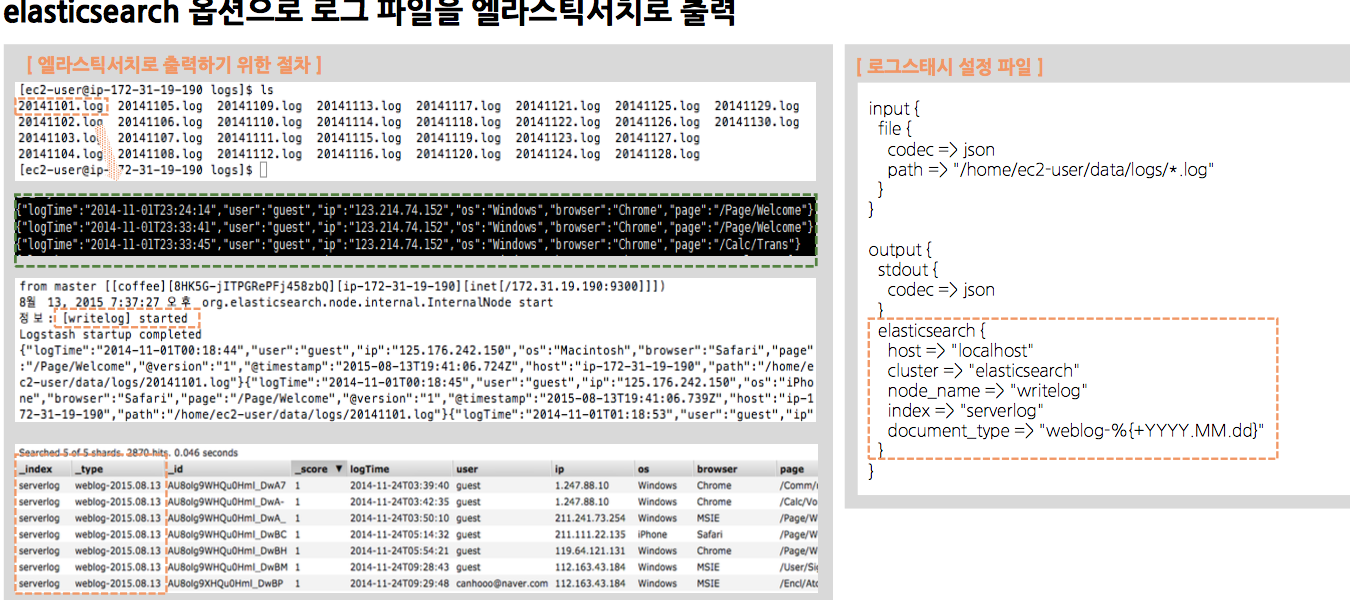
|  |  |
| --- | --- |
|  | input { stdin {  codec => json }  }  filter { if [name] == “sheom” {  mutate { add\_field => { "email" => "%{name}@okiconcession" } replace => [ "part", "Product %{part}" ]  } }  }  output { stdout {  codec => json }  file { codec => json path => "output.txt"  } } |
|  |  |

로그 파일을 엘라스틱서치로 출력

* 엘라스틱서치와 연결 시 JAVA 8 사용을 권장
* 엘라스틱서치로 출력하기 위한 옵션은 2가지를 제공

엘라스틱서치 출력 옵션

* + elasticsearch : 로그스태시를 하나의 엘라스틱서치 노드로 실행 후 시스템과 바인딩
  + elasticsearch\_http : 엘라스틱서치 서버 HTTP 포트로 직접 데이터 입력



키바나 kibana

KIBANA

ELK스택중하나

엘라스틱서치의 복잡한 질의를 편하게 입력 가능

입력된 질의를 간편하게 시각화

config.js 파일을 수정하여 간편하게 설정

node.js로 작성

일부 설정은 엘라스틱서치 인덱스에 저장

실행

./bin/kibana

Managerment 쪽에서 추가할 인덱스를 넣는다 위예제로는 books가될것이다.

Visualize에서 그래프로 확인해보자

또는 Dashboard로 대쉬보드또한 만들수있다.ㅋㅋ 좋으다~

-----