



elasticsearch

엘라스틱서치 입문

TONY EOM





elasticsearch

## 오늘 우리가 함께 보게 될 내용

1. 사용 사례
2. 엘라스틱서치의 특징
3. 시스템 구조
4. 설치와 간단한 설정
5. 데이터 처리
6. 검색
7. QUERY DSL (질의)
8. 어그리게이션
9. 분석기
10. LOG STASH와의 결합
11. KIBANA



elasticsearch

## 사용 사례



**WIKIPEDIA**  
*The Free Encyclopedia*

## 위키피디아 (WIKIPEDIA)

- 전문검색(FULL TEXT SEARCH)를 수행
- 실시간 타이핑 검색
- 추천 검색어 기능

Best for reviews Paul McCartney ★★★ Le Week-end ★★★ The Fifth Estate ★★★

Friday 11 October  
Published in London  
and Manchester  
£1.40 (Ch Islands £1.80)



# the guardian

theguardian.com

'Comedies are filthy'  
Interview by Hadley Freeman

IN C2  
FILM &  
MUSIC



PLUS



GARY NUMAN  
Electro, kittens and trauma  
LORDE  
Chart-topping teen star

## Tom Hanks

## Spies to go under spotlight

● Review of surveillance powers  
in wake of Guardian revelations

● Public accountability and trust  
will be examined, says Clegg

Patrick Wintour  
Rowan Mason  
Dan Roberts/Washington

Mr Clegg's latest conversation again gives overviews about how to update the legal framework of the British state to reflect the right of disclosure by the Guardian that powerful new techniques appear to have been used by the secret services in legislative and political oversight.

The deputy prime minister's aides said the government would also be anxious to make sure the WikiLeaks leak to discuss the implications of the new surveillance techniques to the public. "It's important that the public trust that the law now reflects what the government has done," said Mr Clegg. "That's why we've highlighted concerns about the accountability of the security services."

Clegg was asked whether WikiLeaks' leak of classified US documents, apparently obtained from Edward Snowden, might give him the right to ask whether there is anything more we can do to make sure the security services are held to account. "I think it's right to ask whether there is anything we can do to make sure the security services are held to account. That's a basic principle of democracy. There is a totally legitimate debate about the power of these techniques and how they are used. I think it's right to ask, 'How you make sure these techniques are used in accordance with the rule of law?'

But Clegg also stressed that the only power he has is that of the Minister of State. As Andrew Parker, is the editor of the Guardian, he has no authority to instruct the paper to publish information that is not given to him. "I think it's right to ask whether there is anything we can do to make sure the security services are held to account. That's a basic principle of democracy. There is a totally legitimate debate about the power of these techniques and how they are used. I think it's right to ask, 'How you make sure these techniques are used in accordance with the rule of law?'

Clegg also stressed that the specific examples of State published by the Guardian that would give this key to terrorism.

Comments also weighed into the debate,



Editors speak out »  
Pages 9-13

● A debate about the proper perimeters for eavesdropping by intelligence agencies is healthy and necessary

Jill Abramson, executive editor, New York Times

● Without sight of the facts, how can democracy chart its course?

Marcus Brauchli, vice-president, Washington Post Company

● It is really striking to accuse journalists of being allies of terrorism simply for performing their professional responsibilities

Ricardo Kirschbaum, executive editor, Clarín, Argentina

● Ralph Miliband

## 더 가디언 (THE GUARDIAN)

- 방문객의 로그 분석
- SOCIAL 데이터 생성 및 분석으로 실시간 응대
- 기사에 대한 반응 분석



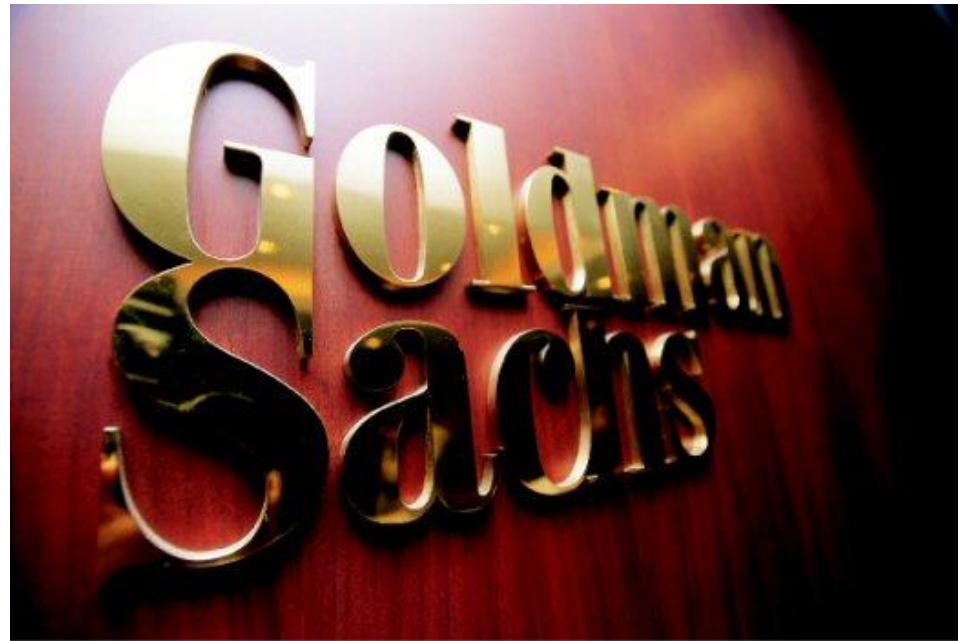
## **스택 오버플로우 (STACK OVERFLOW)**

검색내용과 결과를 통합해 유사한 질문과 해답을 연결



## 깃허브 (GITHUB)

1,300억 줄이 넘는 소스 코드를 검색하는데 사용



## 골드만 삭스 (GOLDMAN SACHS)

- 매일 5TB가 넘는 데이터를 저장
- 주식 시장의 변동 분석에 사용



elasticsearch

## 엘라스틱서치의 특징

# 아파치 루씬 기반

## 루씬의 특징

자바 언어로 개발

사용자 위치 정보 이용 가능

다국어 검색 지원

자동 완성 지원

미리 보기 지원

철자 수정 기능 지원



엘라스틱 서치는 루씬을 기반으로 만들어져 루씬의 기능을 대부분 지원



## 분산 시스템

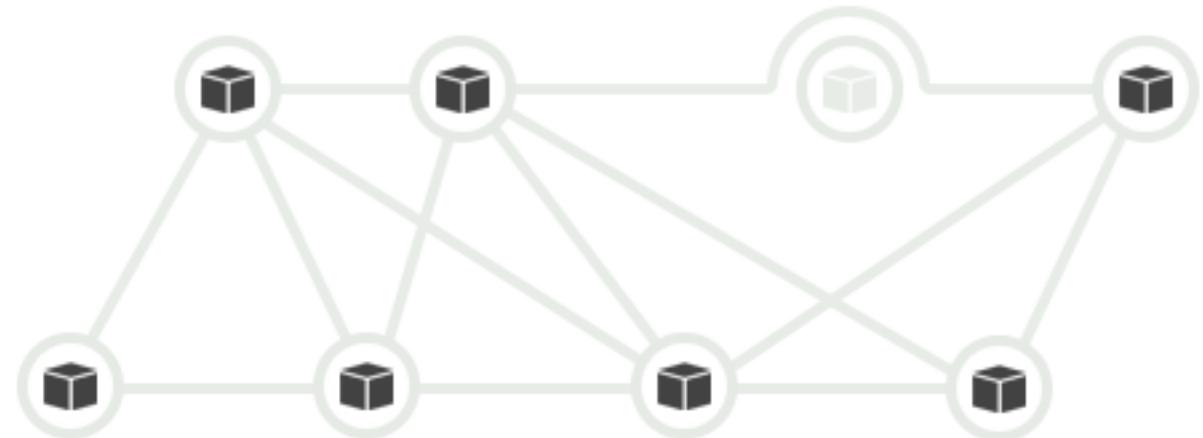
- 엘라스틱 서치는 여러 개의 노드로 구성되는 **분산 시스템**
- **노드**는 데이터를 색인하고 검색을 수행하는 단위 프로세스
- 기존 노드에 새 노드를 실행하여 연결하는 것만으로 확장 가능
- 데이터는 각 노드에 **분산 저장**
- 복사본을 유지하여 각종 충돌로부터 **노드 데이터 보호**
- DISCOVERY를 내장하여 별도의 분산 시스템 관리자 불필요

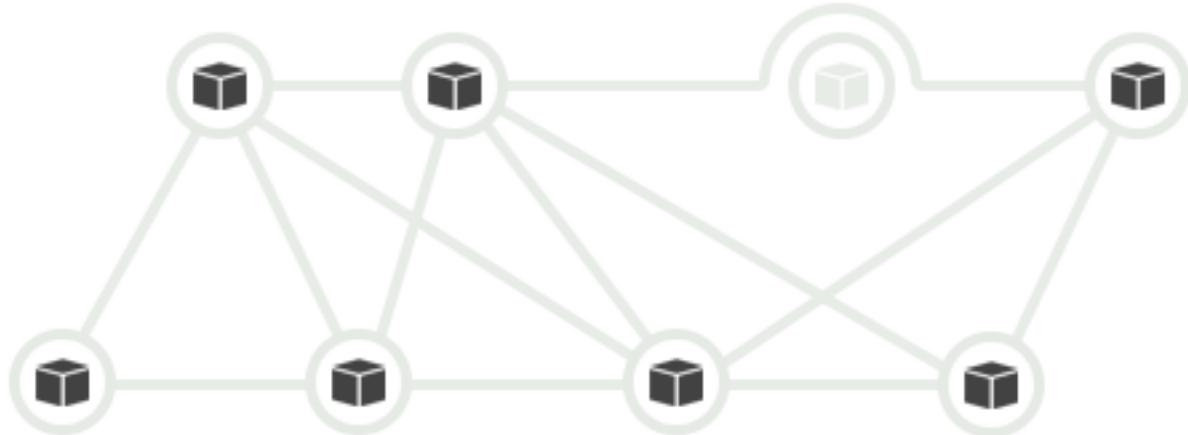
## 높은 가용성 (HIGH AVAILABILITY)

- 엘라스틱 서치는 하나 이상의 노드로 구성
- 각 노드는 1개 이상의 데이터 원본과 복사본을 서로 다른 위치에 나누어 저장
- 노드가 종료되거나 실행에 실패할 경우 다른 노드로 데이터 이동



항상 일정한 데이터 복사본의 개수를 유지하여 높은 가용성과 안정성 보장



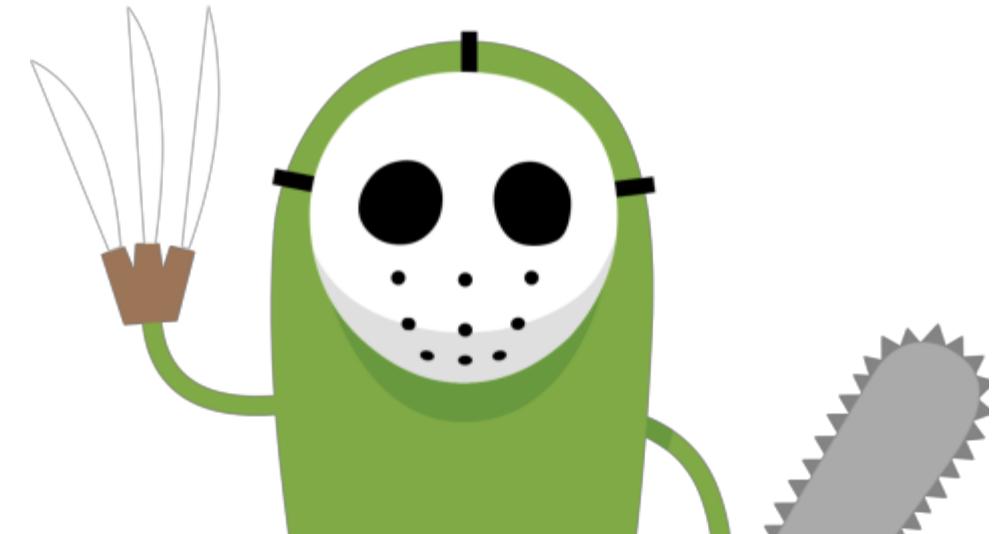


## 멀티 테넌시 (MULTY TENANCY)

- 데이터는 여러 개로 분리된 인덱스들에 그룹으로 저장 (인덱스 = RDBMS의 데이터베이스에 대응)
- 서로 다른 인덱스의 데이터를 **하나의 질의로** 검색하여 **하나의 출력으로** 도출 가능

## JSON DOCUMENT / RESTFUL API

- 기본적으로 모든 필드를 색인 후 JSON 구조로 저장
- JSON 구조로 인해 모든 레벨의 필드에 접근이 쉽고, 빠른 속도로 검색 가능
- 사전 매핑 없이 JSON 문서 형식으로 데이터를 입력하면 바로 색인 작업 수행
- REST 자원은 색인된 데이터 및 질의, 검색되어 JSON 형식으로 출력된 문서를 의미
- JSON 문서를 URI로 명시, 이 문서를 처리하기 위해 HTTP METHOD 이용





## 실시간 분석 (REAL TIME)

- 저장된 데이터는 검색에 사용되기 위해 별도의 재시작 / 갱신이 불필요
- 색인 작업이 완료됨과 동시에 바로 검색 가능
- 실시간 분석 / 검색은 데이터 증가량에 구애 받지 않음

## 솔라와 엘라스틱서치의 비교

	솔라 4.7.0	엘라스틱서치 1.0
문서 형식	XML, CSV, JSON	JSON
REST API	지원	지원
라이브러리	PHP, 루비, 펄, 스칼라, 파이썬, .NET, 자바스크립트	PHP, 루비, 펄, 스칼라, 파이썬, .NET, 자바스크립트, Erlang, Clojure
데이터 불러오기	JDBC, CSV, XML, Tika, URL, Flat File	Rivers modules (플러그인 형식의 모듈) - ActiceMQ, Amazon SQS, CouchDB, Dropbox, DynamoDB, FileSystem, Git, Github, Hazelcast, JDBC, JMS, Kafka, LDAP, MongoDB, neo4j, OAI, RabbitMQ, Redis, RSS, Sofa, Solr, St9, Subversion, Twitter, Wikipedia
다중 스키마 문서	지원하지 않음	지원
조인	지원하지 않음	Parent_type/child_type을 이용하여 구현
분산시스템의 노드 연결	주키퍼	젠 디스커버리, 주키퍼

## 몽고 DB와 엘라스틱서치의 비교

	몽고 DB 4.7.0	엘라스틱서치 1.0
구분	데이터 저장소	검색 엔진 어플리케이션
개발언어	C++	자바
운영체제	리눅스, OS X, 솔라리스, 윈도우	자바 가상 머신이 설치된 모든 OS
접속 방식	자체 프로토콜	RESTful/HTTP API
문서 구조	JSON	JSON

저장이 중요한 시스템에는 몽고 DB를 검색이 중요한 시스템에는 엘라스틱서치를 사용하도록 권장



elasticsearch

## 설치와 간단한 설정

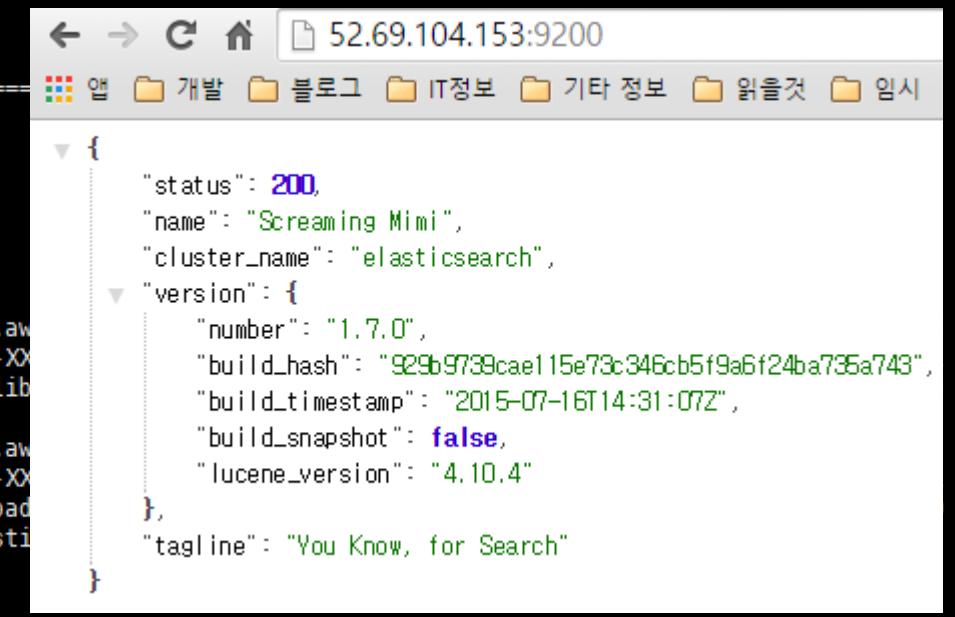
# 설치

```
[ec2-user@ip-172-31-14-124 Download]$ wget https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-1.7.0.tar.gz
--2015-07-28 15:46:57-- https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-1.7.0.tar.gz
Resolving download.elastic.co (download.elastic.co)... 107.20.197.72, 107.22.211.180, 23.23.105.243, ...
Connecting to download.elastic.co (download.elastic.co)|107.20.197.72|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28501532 (27M) [application/x-tar]
Saving to: 'elasticsearch-1.7.0.tar.gz'

elasticsearch-1.7.0.tar.gz          100%[=====]   2.63 MB/s   in 0.00s

2015-07-28 15:47:08 (2.63 MB/s) - 'elasticsearch-1.7.0.tar.gz' saved [28501532/28501532]

[ec2-user@ip-172-31-14-124 Download]$ tar xfz elasticsearch-1.7.0.tar.gz
[ec2-user@ip-172-31-14-124 Download]$ cd elasticsearch-1.7.0/bin/
[ec2-user@ip-172-31-14-124 bin]$ ./elasticsearch -d -p es.pid
[ec2-user@ip-172-31-14-124 bin]$ ps -ef | grep elasticsearch
ec2-user 6442 1 0 Jul19 ? 00:20:18 /usr/lib/jvm/java/bin/java -Xms256m -Xmx1g -Djava.awt
initiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -XX:+HeapDumpOnOutOfMemoryError -XX
dfile=es.pid -Des.path.home=/usr/local/server/elasticsearch -cp :/usr/local/server/elasticsearch/lib
sr/local/server/elasticsearch/lib/sigar/* org.elasticsearch.bootstrap.Elasticsearch
ec2-user 29740 1 84 15:48 pts/0 00:00:08 /usr/lib/jvm/java/bin/java -Xms256m -Xmx1g -Djava.awt
initiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -XX:+HeapDumpOnOutOfMemoryError -XX
dfile=es.pid -Des.path.home=/home/ec2-user/Download/elasticsearch-1.7.0 -cp :/home/ec2-user/Download
ownload/elasticsearch-1.7.0/lib/*:/home/ec2-user/Download/elasticsearch-1.7.0/lib/sigar/* org.elasticsearch
ec2-user 29771 29680 10 15:48 pts/0 00:00:00 grep elasticsearch
[ec2-user@ip-172-31-14-124 bin]$
```



다운로드



압축해제



실행

# 프로그램 구조

```
bin
  - elasticsearch
    - elasticsearch.bat
    - elasticsearch.in.bat
    - elasticsearch.in.sh
    - elasticsearch-service-mgr.exe
    - elasticsearch-service-x64.exe
    - elasticsearch-service-x86.exe
    - es.pid
  - plugin
    - plugin.bat
    - service.bat
config
  - elasticsearch.yml
  - logging.yml
data
  - elasticsearch
    - nodes
lib
  - antlr-runtime-3.5.jar
  - apache-log4j-extras-1.2.17.jar
  - asm-4.1.jar
  - asm-commons-4.1.jar
  - elasticsearch-1.7.0.jar
  - groovy-all-2.4.4.jar
  - jna-4.1.0.jar
  - jts-1.13.jar
  - log4j-1.2.17.jar
  - lucene-analyzers-common-4.10.4.jar
  - lucene-core-4.10.4.jar
  - lucene-expressions-4.10.4.jar
  - lucene-grouping-4.10.4.jar
  - lucene-highlighter-4.10.4.jar
  - lucene-join-4.10.4.jar
  - lucene-memory-4.10.4.jar
  - lucene-misc-4.10.4.jar
  - lucene-queries-4.10.4.jar
  - lucene-queryparser-4.10.4.jar
  - lucene-sandbox-4.10.4.jar
  - lucene-spatial-4.10.4.jar
  - lucene-suggest-4.10.4.jar
  - sigar
    - libsigar-amd64-freebsd-6.so
    - libsigar-amd64-linux.so
    - libsigar-amd64-solaris.so
    - libsigar-ia64-linux.so
    - libsigar-sparc64-solaris.so
    - libsigar-sparc-solaris.so
    - libsigar-universal64-macosx.dylib
    - libsigar-universal-macosx.dylib
    - libsigar-x86-freebsd-5.so
    - libsigar-x86-freebsd-6.so
    - libsigar-x86-linux.so
    - libsigar-x86-solaris.so
    - sigar-1.6.4.jar
    - sigar-amd64-winnt.dll
    - sigar-x86-winnt.dll
    - sigar-x86-winnt.lib
  - spatial4j-0.4.1.jar
LICENSE.txt
logs
  - elasticsearch_index_indexing_slowlog.log
  - elasticsearch_index_search_slowlog.log
  - elasticsearch.log
NOTICE.txt
README.textile
```

자바 실행 변수 파일과 엘라스틱 서치 실행 파일, 플러그인 설치 프로그램

엘라스틱서치 실행 환경 및 로그 설정 파일

데이터 저장 공간, 클러스터 별로 하위 디렉토리 생성

엘라스틱서치 자바 라이브러리 파일

실행 로그와 슬로우 로그 파일

## 환경 설정

- elasticsearch.in.sh 파일을 수정하여 설정
- elasticsearch.yml 파일을 수정하여 설정
- 실행 시 -D\* 혹은 --\* 옵션을 사용하여 설정
- 실행 후 REST API를 이용

## 환경 설정 (elasticsearch.in.sh)

- 최소 최대 메모리 사용량을 지정 가능
- ES\_HEAP\_SIZE를 지정하여 동시 지정 가능
- 메모리 용량 변경에 들어가는 불필요한 오버헤드 방지 차원에서 최소, 최대 메모리를 동일하게 하는 것을 권장

[ elasticsearch.in.sh 파일을 수정하여 메모리 사용량 변경 ]

```
#!/bin/sh

ES_CLASSPATH=$ES_CLASSPATH:$ES_HOME/lib/elasticsearch-1.7.1.jar:$ES_HOME/lib/*:$ES_HOME/lib/sigar/*

if [ "x$ES_MIN_MEM" = "x" ]; then
    ES_MIN_MEM=256m
fi
if [ "x$ES_MAX_MEM" = "x" ]; then
    ES_MAX_MEM=lg
fi
if [ "x$ES_HEAP_SIZE" != "x" ]; then
    ES_MIN_MEM=$ES_HEAP_SIZE
    ES_MAX_MEM=$ES_HEAP_SIZE
fi

# min and max heap sizes should be set to the same value to avoid
# stop-the-world GC pauses during resize, and so that we can lock the
# heap in memory on startup to prevent any of it from being swapped
# out.
JAVA_OPTS="$JAVA_OPTS -Xms${ES_MIN_MEM}"
JAVA_OPTS="$JAVA_OPTS -Xmx${ES_MAX_MEM}"
```

## 환경 설정 (elasticsearch.yml)

- 엘라스틱 서치의 대부분의 환경설정을 가지고 있음
- 처음 설치 시 모든 옵션은 주석처리되어 기본값으로 실행
- 설정 문법은 YAML 문법을 사용하며 콜론 뒤에 공백이 없으면 오류 발생

## 자주 사용되는 설정

설정	내용
cluster.name	클러스터 이름을 설정 (기본값은 “elasticsearch”)
node.name	노드의 이름을 설정 (기본값은 임의의 이름이 부여)
bootstrap.mlockall	사용 메모리 고정 (JVM이 메모리를 다른 자바프로그램으로 돌리는 것을 방지)
index.number_of_shard	샤드 개수 지정 (기본값은 5)
index.number_of_replicas	복사본 개수 지정 (기본값은 1)

## 환경 설정 (elasticsearch.yml)

### HTTP 네트워크 설정

설정	내용
network.bind_host	서버 내부 IP 주소 (방화벽이나, 공유기 등을 사용 시)
network.publish_host	공개 IP 주소
network.host	내부 IP와 공개 IP가 동일할 경우 사용
tranport.tcp.port	다른 노드와 바인딩하여 통신 하기 위해 사용하는 포트 (기본값 9300)
transport.tcp.compress	통신 내용의 압축 여부
http.port	REST API 서비스 포트 (기본값 9200)
http.max_content_length	설정된 용량을 초과한 데이터는 전송하지 않음
http.enabled	REST API 서비스 제공 여부



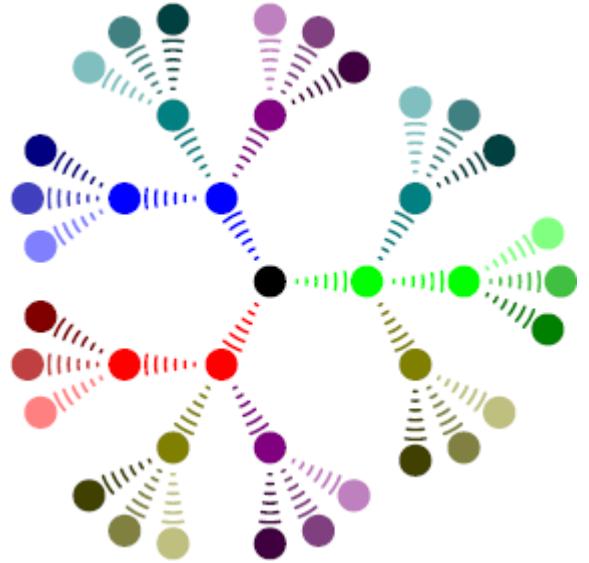
elasticsearch

## 시스템 구조



## 클러스터 (CLUSTER)

- 클러스터는 엘라스틱서치의 가장 큰 시스템 단위
- 하나의 클러스터는 여러 개의 노드로 이루어짐
- 여러 대의 서버가 하나의 클러스터를 구성할 수 있으며 그 반대도 가능
- 같은 클러스터의 이름으로 노드를 실행하는 것만으로 자동 확장

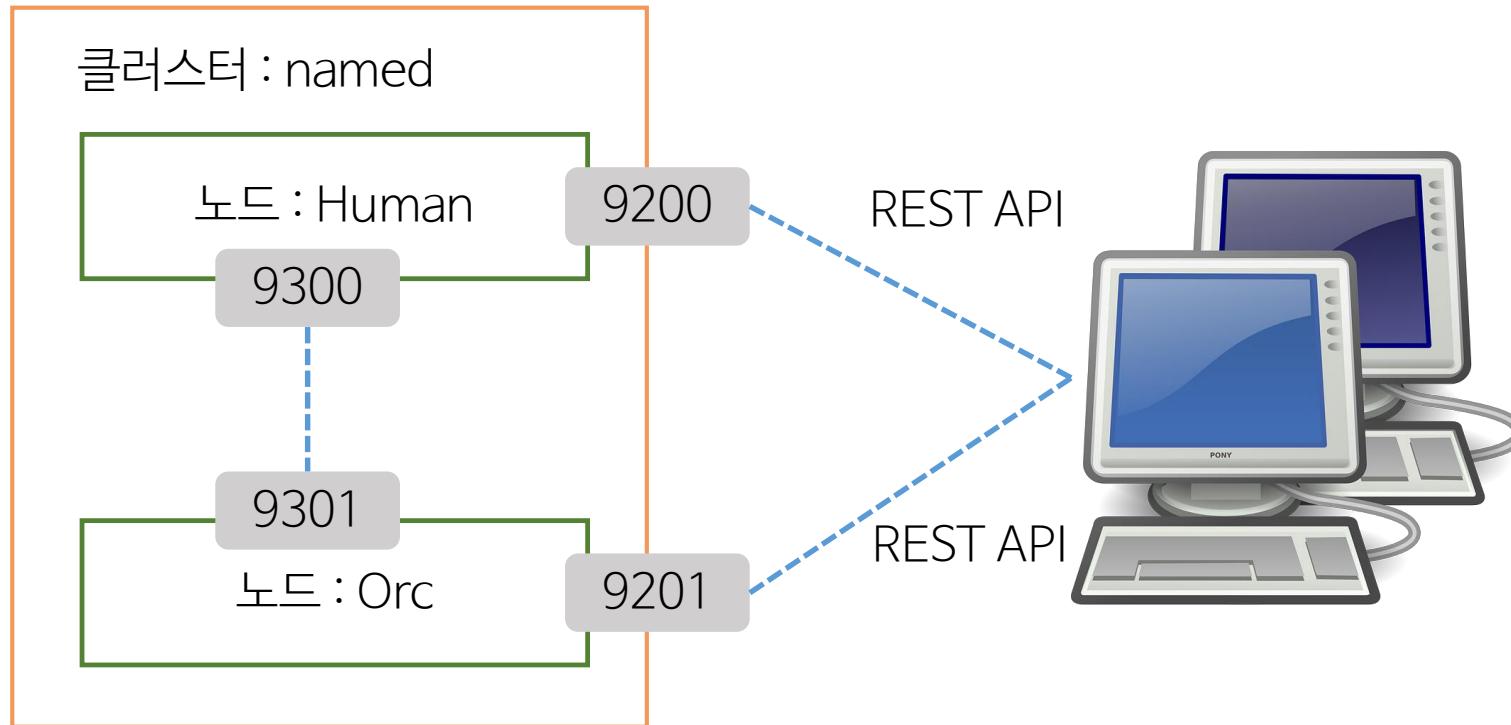


## 노드 (NODE)

- 노드는 마스터 노드와 데이터 노드로 구분
- 마스터 노드는 전체 클러스터 상태의 메타 정보를 관리
- 기존의 마스터 노드가 종료되는 경우 새로운 마스터 노드가 선출됨
- 데이터 노드는 실제 데이터가 저장되는 노드

## 노드 바인딩

- 같은 클러스터 이름을 가지고 실행된 노드는 자동으로 바인딩
- 9200번부터 REST API를 위한 HTTP 통신 포트가 할당
- 9300번 부터 노드간 바인딩을 위한 포트로 할당



# 노드 바인딩

## [ 두 개의 노드가 바인딩 ]

```
[root@localhost bin]# ./elasticsearch --node.name=Human
[2015-08-13 00:07:58,213][INFO ][node              ] [Human] version[1.7.1], pid[27596], build[b88f43f/2015-07-29T09:54:16Z]
[2015-08-13 00:07:58,214][INFO ][node              ] [Human] initializing ...
[2015-08-13 00:07:58,282][INFO ][plugins          ] [Human] loaded [], sites []
[2015-08-13 00:07:58,330][INFO ][env              ] [Human] using [1] data paths, mounts [[/ (/dev/sda2)]], net usable_space [41.6gb], net total_space [46.8gb], types [ext4]
[2015-08-13 00:08:00,181][INFO ][node              ] [Human] initialized
[2015-08-13 00:08:00,181][INFO ][node              ] [Human] starting ...
[2015-08-13 00:08:00,246][INFO ][transport        ] [Human] bound_address {inet[/0:0:0:0:0:0:0:9300]}, publish_address {inet[/192.168.153.130:9300]}
[2015-08-13 00:08:00,260][INFO ][discovery        ] [Human] named/PGPnp0M1SHWhjY7jdNThEQ
[2015-08-13 00:08:04,027][INFO ][cluster.service ] [Human] new_master [Human][PGPnp0M1SHWhjY7jdNThEQ][localhost.localdomain][inet[/192.168.153.130:9300]], reason: zen-disco-join (elected_as_master)
[2015-08-13 00:08:04,044][INFO ][http             ] [Human] bound_address {inet[/0:0:0:0:0:0:0:9200]}, publish_address {inet[/192.168.153.130:9200]}
[2015-08-13 00:08:04,044][INFO ][node              ] [Human] started
[2015-08-13 00:08:04,062][INFO ][gateway         ] [Human] recovered [0] indices into cluster_state
[2015-08-13 00:08:15,126][INFO ][cluster.service ] [Human] added {[Orc][_seEbgr1QIyEnW0_dnRdUg][localhost.localdomain][inet[/192.168.153.130:9301]]}, reason: zen-disco-receive(join from node[[Orc][_seEbgr1QIyEnW0_dnRdUg]][localhost.localdomain][inet[/192.168.153.130:9301]]])
```

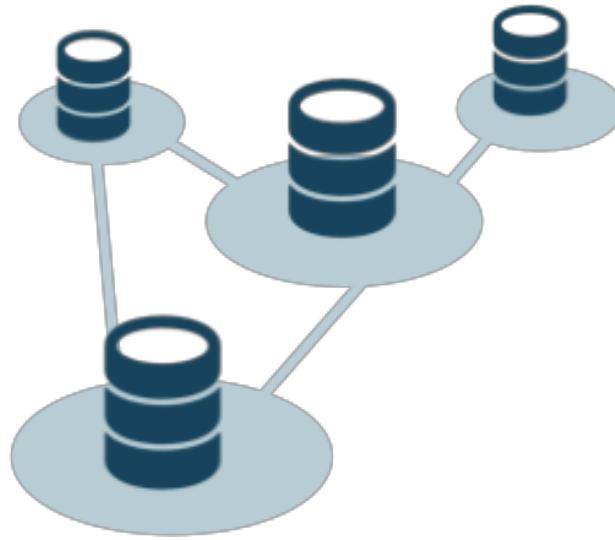
```
[root@localhost bin]# ./elasticsearch --node.name=Orc
[2015-08-13 00:08:10,096][INFO ][node              ] [Orc] version[1.7.1], pid[27660], build[b88f43f/2015-07-29T09:54:16Z]
[2015-08-13 00:08:10,096][INFO ][node              ] [Orc] initializing ...
[2015-08-13 00:08:10,166][INFO ][plugins          ] [Orc] loaded [], sites []
[2015-08-13 00:08:10,194][INFO ][env              ] [Orc] using [1] data paths, mounts [[/ (/dev/sda2)]], net usable_space [41.6gb], net total_space [46.8gb], types [ext4]
[2015-08-13 00:08:12,040][INFO ][node              ] [Orc] initialized
[2015-08-13 00:08:12,041][INFO ][node              ] [Orc] starting ...
[2015-08-13 00:08:12,088][INFO ][transport        ] [Orc] bound_address {inet[/0:0:0:0:0:0:0:9301]}, publish_address {inet[/192.168.153.130:9301]}
[2015-08-13 00:08:12,099][INFO ][discovery        ] [Orc] named/_seEbgr1QIyEnW0_dnRdUg
[2015-08-13 00:08:15,137][INFO ][cluster.service ] [Orc] detected_master [Human][PGPnp0M1SHWhjY7jdNThEQ][localhost.localdomain][inet[/192.168.153.130:9300]], added {[Human][PGPnp0M1SHWhjY7jdNThEQ][localhost.localdomain][inet[/192.168.153.130:9300]]}, reason: zen-disco-receive(from master [[Human][PGPnp0M1SHWhjY7jdNThEQ][localhost.localdomain][inet[/192.168.153.130:9300]]]))
[2015-08-13 00:08:15,172][INFO ][http             ] [Orc] bound_address {inet[/0:0:0:0:0:0:0:9201]}, publish_address {inet[/192.168.153.130:9201]}
[2015-08-13 00:08:15,172][INFO ][node              ] [Orc] started
```

## 마스터노드와 데이터 노드

- 기존 마스터노드가 종료되면 새로운 마스터노드가 선출
- 데이터노드가 하나밖에 없는 경우 복사본은 생성되지 않음
- 마스터노드와 데이터노드가 반드시 상호 배타적 관계는 아님
- 일반적으로 데이터노드는 외부 접근을 차단

### [ 노드 타입 설정 옵션 ]

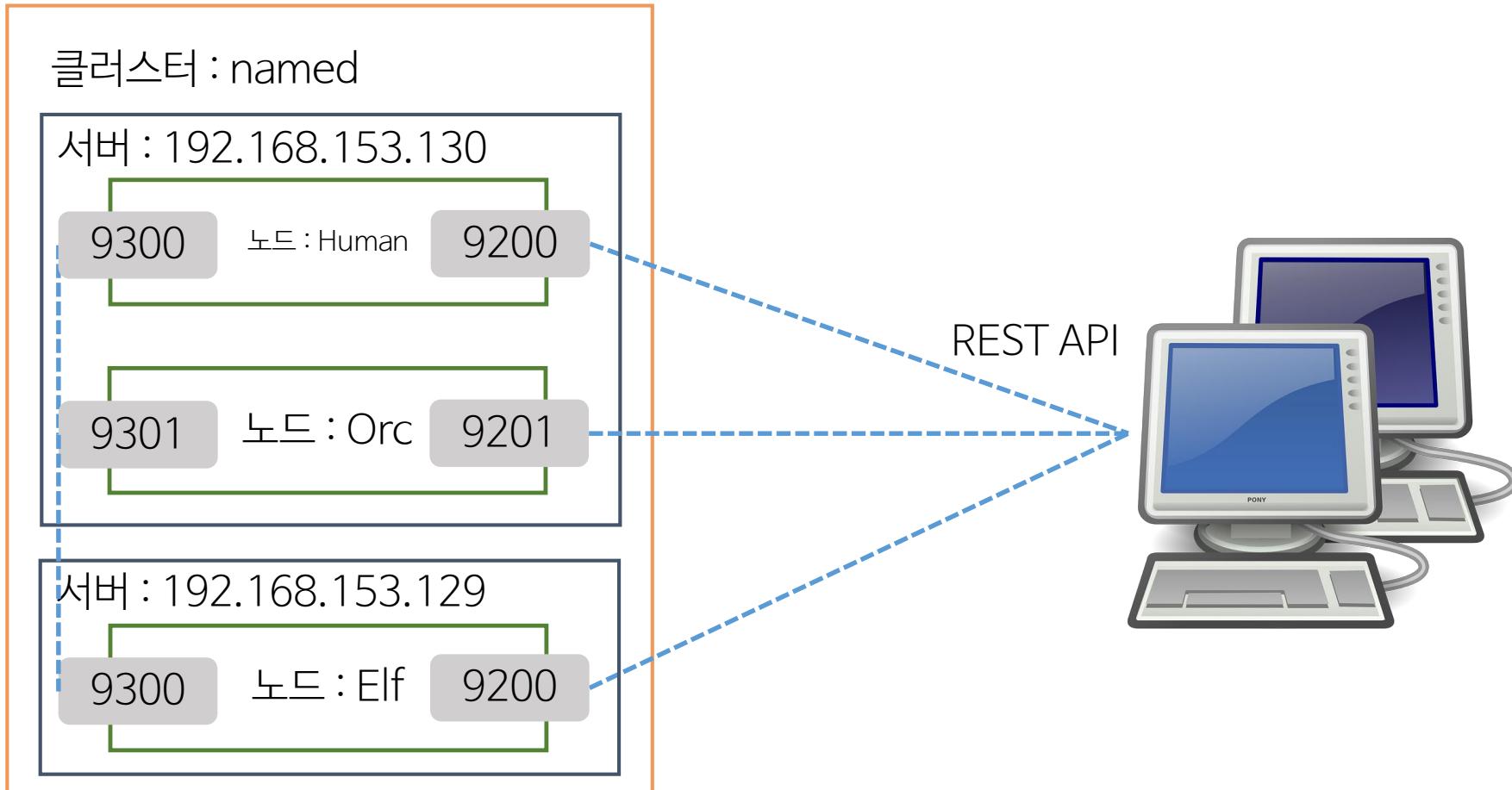
```
#node.master: true  
#  
# Allow this node to store data (enabled by default):  
#  
#node.data: true
```



## 네트워크 바인딩

- 효율적인 스케일아웃을 위해 네트워크에 있는 다른 서버의 노드와도 바인딩 가능
- 네트워크 바인딩을 위해 [젠 디스커버리\(ZEN DISCOVERY\)](#) 기능 내장
- 멀티캐스트와 유니캐스트 방식을 모두 지원
- 공식 운영 그룹에서는 [유니캐스트의 사용을 권장](#)
- 반드시 두 엘라스틱서치의 버전은 [동일해야 함](#)

# 네트워크 바인딩



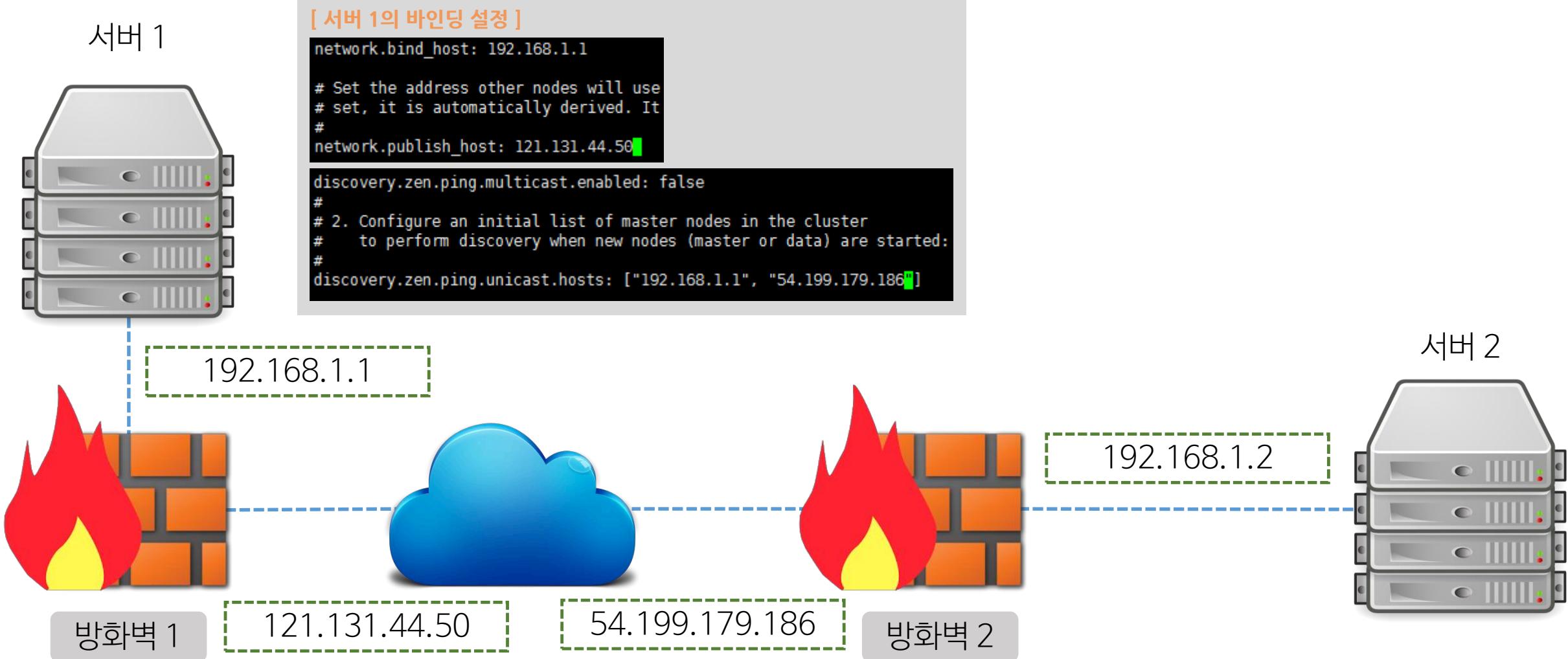
# 네트워크 바인딩

## [ 유니캐스트로 네트워크 바인딩 ]

```
#  
discovery.zen.ping.multicast.enabled: false  
#  
# 2. Configure an initial list of master nodes in the cluster  
#      to perform discovery when new nodes (master or data) are started:  
#  
discovery.zen.ping.unicast.hosts: ["192.168.153.130", "192.168.153.129"]
```

```
[root@localhost bin]# ./elasticsearch --node.name=Human  
[2015-08-13 01:20:41,587][INFO ][node          ] [Human] version[1.7.1], pid[28357], build[b88f43f/2015-07-29T09:54:16Z]  
[2015-08-13 01:20:41,588][INFO ][node          ] [Human] initializing ...  
[2015-08-13 01:20:41,655][INFO ][plugins       ] [Human] loaded [], sites []  
[2015-08-13 01:20:41,682][INFO ][env           ] [Human] using [1] data paths, mounts [[/ (/dev/sda2)]], net usable_space [41.6gb], net total_space [46.8gb], types [ext4]  
[2015-08-13 01:20:43,467][INFO ][node          ] [Human] initialized  
[2015-08-13 01:20:43,467][INFO ][node          ] [Human] starting ...  
[2015-08-13 01:20:43,510][INFO ][transport     ] [Human] bound_address {inet[/0:0:0:0:0:0:0:9300]}, publish_address {inet[/192.168.153.130:9300]}  
[2015-08-13 01:20:43,518][INFO ][discovery    ] [Human] named/CEAlzRICTfmsgE_0QaUytw  
[2015-08-13 01:20:46,535][INFO ][cluster.service] [Human] new_master [Human][CEAlzRICTfmsgE_0QaUytw][localhost.localdomain][inet[/192.168.153.130:9300]], reason: zen-disco-join (elected_as_master)  
[2015-08-13 01:20:46,551][INFO ][http          ] [Human] bound_address {inet[/0:0:0:0:0:0:0:9200]}, publish_address {inet[/192.168.153.130:9200]}  
[2015-08-13 01:20:46,551][INFO ][node          ] [Human] started  
[2015-08-13 01:20:46,559][INFO ][gateway      ] [Human] recovered [0] indices into cluster_state  
[2015-08-13 01:25:19,480][INFO ][cluster.service] [Human] added {[Elf][1PPqSbd3R5qv8rSeYzAexQ][localhost.localdomain][inet[/192.168.153.129:9300]]}, reason: zen-disco-receive(join from node[[Elf][1PPqSbd3R5qv8rSeYzAexQ][localhost.localdomain][inet[/192.168.153.129:9300]]])  
[root@localhost bin]# ./elasticsearch --node.name=Elf  
[2015-08-13 01:25:03,339][INFO ][node          ] [Elf] version[1.7.1], pid[5992], build[b88f43f/2015-07-29T09:54:16Z]  
[2015-08-13 01:25:03,339][INFO ][node          ] [Elf] initializing ...  
[2015-08-13 01:25:03,411][INFO ][plugins       ] [Elf] loaded [], sites []  
[2015-08-13 01:25:03,443][INFO ][env           ] [Elf] using [1] data paths, mounts [[/ (/dev/sda2)]], net usable_space [22.8gb], net total_space [27.2gb], types [ext4]  
[2015-08-13 01:25:05,225][INFO ][node          ] [Elf] initialized  
[2015-08-13 01:25:05,225][INFO ][node          ] [Elf] starting ...  
[2015-08-13 01:25:05,278][INFO ][transport     ] [Elf] bound_address {inet[/0:0:0:0:0:0:0:9300]}, publish_address {inet[/192.168.153.129:9300]}  
[2015-08-13 01:25:05,286][INFO ][discovery    ] [Elf] named/1PPqSbd3R5qv8rSeYzAexQ  
[2015-08-13 01:25:08,357][INFO ][cluster.service] [Elf] detected_master [Human][CEAlzRICTfmsgE_0QaUytw][localhost.localdomain][inet[/192.168.153.130:9300]], added {[Human][CEAlzRICTfmsgE_0QaUytw][localhost.localdomain][inet[/192.168.153.130:9300]]})  
[2015-08-13 01:25:08,377][INFO ][http          ] [Elf] bound_address {inet[/0:0:0:0:0:0:0:9200]}, publish_address {inet[/192.168.153.129:9200]}  
[2015-08-13 01:25:08,377][INFO ][node          ] [Elf] started
```

# 방화벽 외부 서버 사이의 네트워크 바인딩



## 샤드와 복사본

- 샤드는 데이터 검색을 위해 구분되는 최소 단위
- 색인된 데이터는 여러 개의 샤드로 분할돼 저장
- 기본적으로 인덱스당 5개의 샤드와 5개의 복사본으로 분리
- 개수 설정을 제외하면 사용자가 직접 샤드에 접근하는 경우는 없음
- 데이터가 색인돼 저장되는 공간을 최초 샤드(Primary Shard)라 함
- 최초 샤드에 데이터가 색인되면 동일한 수만큼 복사본을 생성
- 최초 샤드가 유실되는 경우 복사본을 최초 샤드로 승격
- 최초 샤드와 복사본은 동시 검색 대상이 되어 성능 향상에 이점이 있음
- 최초 샤드와 복사본은 서로 다른 노드에 저장
- 생성된 인덱스의 샤드 설정은 변경 불가능

### [ 샤드와 복사본 수 설정 ]

```
# Set the number of shards (splits) of an index (5 by default):
#
#index.number_of_shards: 5

# Set the number of replicas (additional copies) of an index (1 by default):
#
#index.number_of_replicas: 1
```

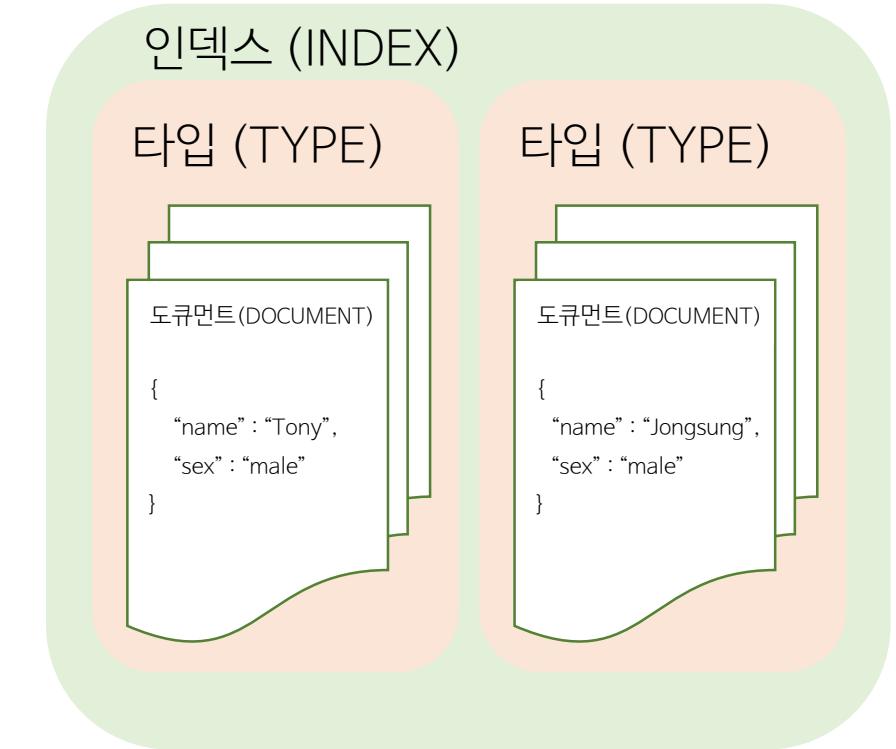


elasticsearch

**데이터 처리**

## 엘라스틱서치 데이터 구조

- 데이터 구조는 인덱스, 타입, 도큐먼트 단위로 이루어짐
- 서로 다른 인덱스 타입들은 서로 다른 매팅 구조로 구성
- 멀티 테넌시를 지원하므로 여러 개의 인덱스를 한꺼번에 검색 처리 가능



## RDBMS와 데이터 구조 비교



관계형 DB	엘라스틱서치
데이터베이스 (DATABASE)	인덱스 (INDEX)
테이블 (TABLE)	타입 (TYPE)
열 (ROW)	도큐먼트 (DOCUMENT)
행 (COLUMN)	필드 (FIELD)
スキ마 (SCHEMA)	매핑 (MAPPING)

## REST API

- 엘라스틱서치는 주로 REST API를 통하여 데이터를 처리
- HTTP METHOD는 GET, POST, PUT, DELETE, HEAD 등이 있음
- 엘라스틱서치의 REST API를 이용하기 위한 형태는 아래와 같음

```
curl -X{METHOD} http://host:port/{INDEX}/{TYPE}/{ID} -d '{DATA}'
```

## HTTP METHOD, CRUD, SQL 비교

HTTP METHOD	CRUD	SQL
GET	READ	SELECT
PUT	UPDATE	UPDATE
POST	CREATE	INSERT
DELETE	DELETE	DELETE

## 데이터 입력 (POST 메서드)

- POST나 PUT 메서드를 이용해 입력
- 처음 입력 시 created는 true
- 도큐먼트 ID를 생략하고 입력 가능
- 임의 ID 입력은 POST 메서드로만 가능

[ 도큐먼트 ID 부여 ]

POST http://52.69.104.153:9200/devteam/member

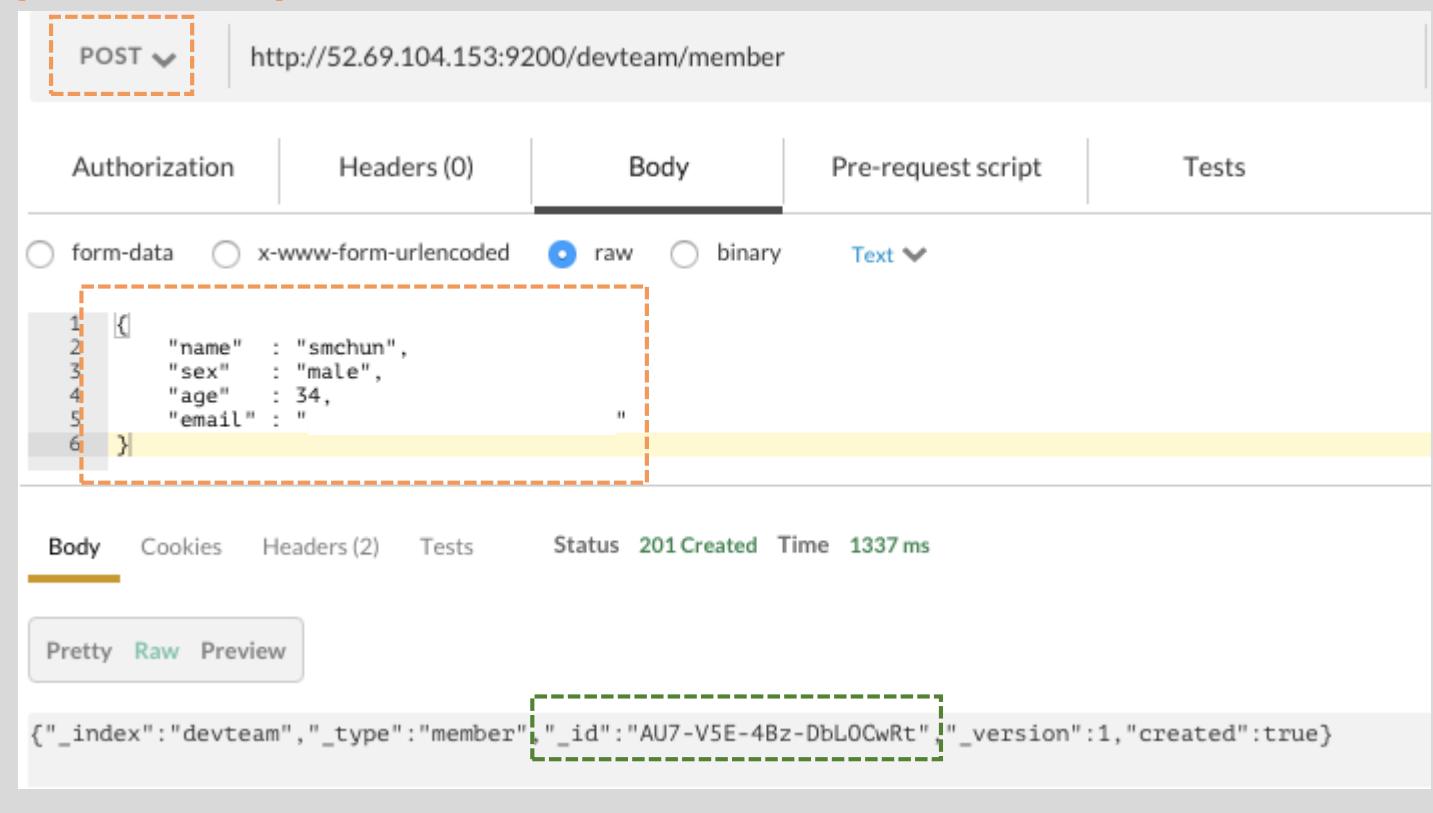
Authorization Headers (0) Body Pre-request script Tests

form-data x-www-form-urlencoded raw binary Text

```
1 {  
2   "name" : "smchun",  
3   "sex" : "male",  
4   "age" : 34,  
5   "email" : "  
6 }  
{"_index": "devteam", "_type": "member", "_id": "AU7-V5E-4Bz-DbLOCwRt", "_version": 1, "created": true}
```

Body Cookies Headers (2) Tests Status 201 Created Time 1337 ms

Pretty Raw Preview



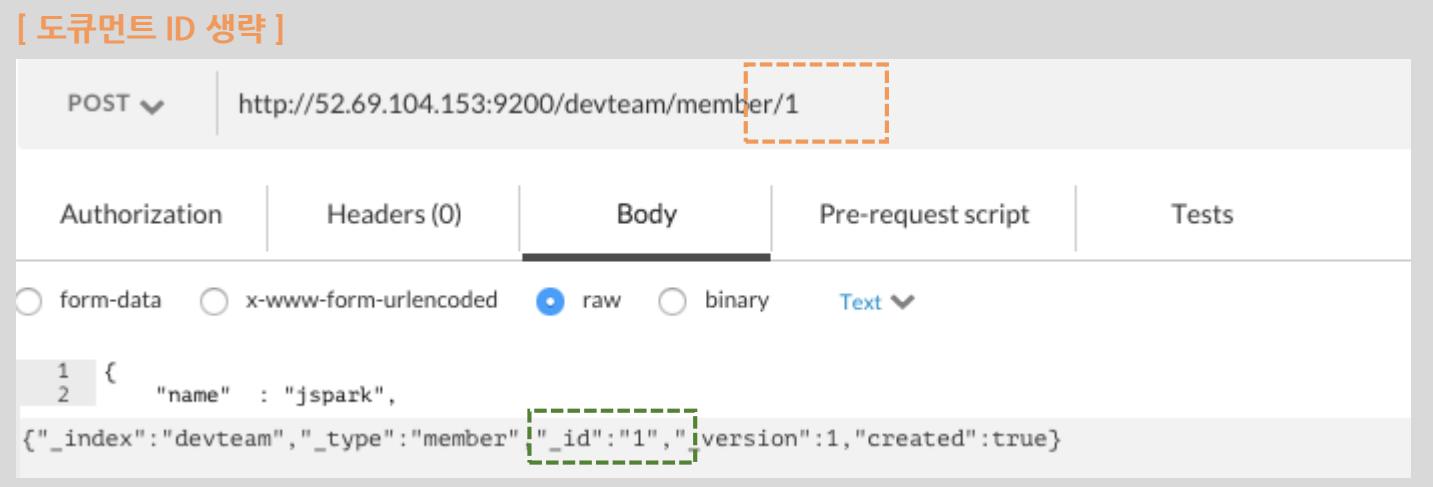
[ 도큐먼트 ID 생략 ]

POST http://52.69.104.153:9200/devteam/member/1

Authorization Headers (0) Body Pre-request script Tests

form-data x-www-form-urlencoded raw binary Text

```
1 {  
2   "name" : "jspark",  
{"_index": "devteam", "_type": "member", "_id": "1", "_version": 1, "created": true}
```



## 데이터 삭제

- DELETE 메서드를 이용해 삭제
- 검색을 이용해 선별적 삭제 가능
- 타입과 인덱스 단위로 삭제 가능
- 인덱스 단위로 삭제할 경우 타입, 도큐먼트가 모두 삭제됨
- 도큐먼트는 삭제 후에도 메타 데이터를 그대로 보존

[ 도큐먼트 삭제 ]

DELETE http://52.69.104.153:9200/devteam/member/1

Body Cookies Headers (2) Tests Status 200 OK Time 51ms

Pretty Raw Preview JSON

```
1 {  
2   "found": true,  
3   "_index": "devteam",  
4   "_type": "member",  
5   "_id": "1",  
6   "_version": 3  
7 }
```

[ 도큐먼트 삭제 후 조회 ]

GET http://52.69.104.153:9200/devteam/member/1

Body Cookies Headers (2) Tests Status 404 Not Found Time

Pretty Raw Preview JSON

```
1 {  
2   "_index": "devteam",  
3   "_type": "member",  
4   "_id": "1",  
5   "found": false  
6 }
```

## 데이터 갱신 (PUT 메서드)

- PUT 메서드를 이용해 갱신
- 갱신 시 `created`는 false
- `_version` 필드가 증가
- 기존 도큐먼트는 보관되지 않으며 삭제되고 새로 쓰임
- 이전 버전으로 변경 불가능

[ PUT 메서드를 이용해 데이터 갱신 ]

PUT http://52.69.104.153:9200/devteam/member/1

Authorization Headers (0) Body Pre-request script Tests

form-data x-www-form-urlencoded raw binary Text

```
1 {
2   "name" : "jspark",
3   "sex" : "female",
4   "age" : 33
5   "email" :
6   "lang" : [ "java", "php", "javascript", "html", "css" ]
7 }
```

Body Cookies Headers (2) Tests Status 200 OK Time 123 ms

Pretty Raw Preview

```
{"_index": "devteam", "_type": "member", "_id": "1", "_version": 2, "created": false}
```

[ 갱신 후 조회 ]

GET http://52.69.104.153:9200/devteam/member/1/\_source

Body Cookies Headers (2) Tests Status 200 OK Time 53 ms

```
1 {
2   "name": "jspark",
3   "sex": "female",
4   "age": 33,
5   "email": null,
6   "lang": [
7     "java",
8     "php",
9     "javascript",
10    "html",
11    "css"
12  ]
13 }
```

## 데이터 갱신 (\_update API - 1)

- POST 메서드를 사용
- doc과 script를 이용해서 데이터 제어
- 도큐먼트의 구조는 그대로 유지
- \_version 필드가 증가

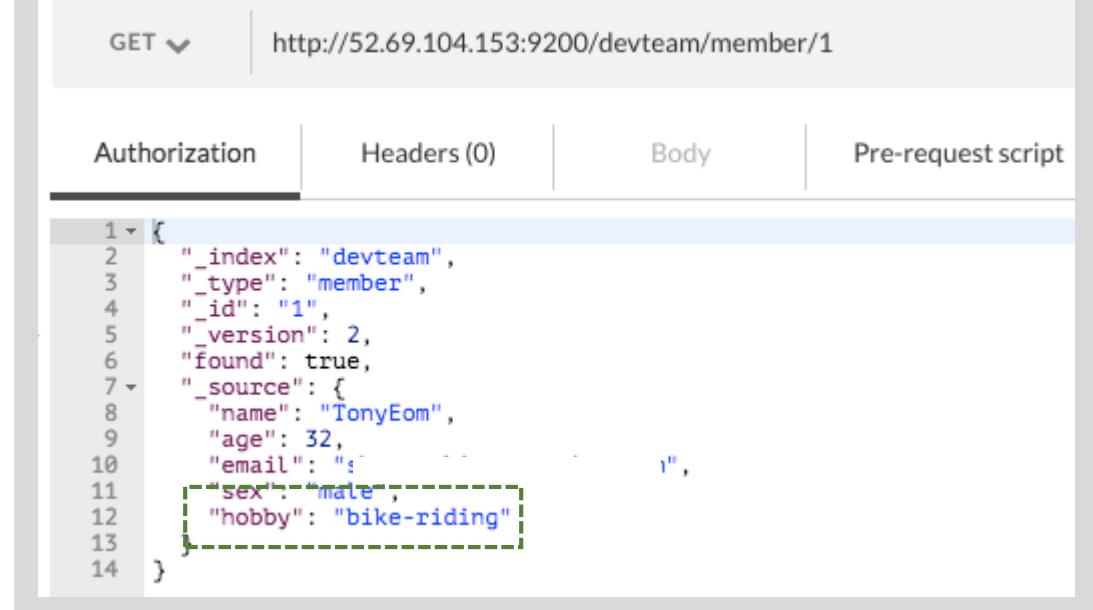
[ \_update API를 이용해 데이터 갱신 ]



```
POST http://52.69.104.153:9200/devteam/member/1/_update

1 {
2   "doc": {
3     "hobby": "bike-riding"
4   }
5 }
```

[ 갱신 후 조회 ]



```
GET http://52.69.104.153:9200/devteam/member/1

Authorization Headers (0) Body Pre-request script

1 {
2   "_index": "devteam",
3   "_type": "member",
4   "_id": "1",
5   "_version": 2,
6   "found": true,
7   "_source": {
8     "name": "TonyEom",
9     "age": 32,
10    "email": "tony.eom@naver.com",
11    "sex": "male",
12    "hobby": "bike-riding"
13  }
14 }
```

## 데이터 갱신 (\_update API - 2)

- script 매개변수를 이용하여 다양한 연산 적용 가능
- script는 MVEL 언어 문법을 사용
- 기본적으로는 사용 불가능하게 설정 됨

[ \_script 매개변수를 이용한 갱신 ]

```
POST http://52.69.104.153:9200/devteam/member/1/_update
```

```
1 {  
2   "script" : "if (ctx._source.age > 40) { ctx._source.age = u_age } else { ctx._source.age -= 10 }",  
3   "params" : { "u_age" : 32 }  
4 }
```

Body Cookies Headers(2) Tests Status 200 OK Time 121ms

Pretty Raw Preview JSON

```
1 {  
2   "_index": "devteam",  
3   "_type": "member",  
4   "_id": "1",  
5   "_version": 3,  
6 }
```

[ 갱신 전 ]

```
GET http://52.69.104.153:9200/devteam/member/1
```

```
1 {  
2   "_index": "devteam",  
3   "_type": "member",  
4   "_id": "1",  
5   "_version": 3,  
6   "found": true,  
7   "_source": {  
8     "name": "TonyEom",  
9     "age": 42,  
10    "email": "st...@n...",  
11    "sex": "male",  
12    "hobby": "bike-riding"  
13  }  
14 }
```

[ 갱신 후 ]

```
1 {  
2   "_index": "devteam",  
3   "_type": "member",  
4   "_id": "1",  
5   "_version": 4,  
6   "found": true,  
7   "_source": {  
8     "name": "TonyEom",  
9     "age": 32,  
10    "email": "st...@n...",  
11    "sex": "male",  
12    "hobby": "bike-riding"  
13  }  
14 }
```

## 데이터 갱신 (\_update API - 3)

- script 매개변수의 ctx.op 명령을 이용해 문서 제어 가능

### [ script 매개변수를 이용한 문서 삭제 ]

POST [http://52.69.104.153:9200/devteam/member/1/\\_update](http://52.69.104.153:9200/devteam/member/1/_update)

Authorization Headers (0) Body Pre-request script Tests

form-data  x-www-form-urlencoded  raw  binary Text

```
1 {  
2   "script" : "if (ctx._source.name.contains(param)) { ctx.op = 'delete' } else { ctx.op = 'none' }",  
3   "params" : { "param" : "Tony" }  
4 }
```

### [ 삭제 후 조회 ]

GET <http://52.69.104.153:9200/devteam/member/1>

Authorization Headers (0) Body Pre-reqes

No Auth

Body Cookies Headers (2) Tests Status 404 Not Found Time 48 ms

Pretty Raw Preview JSON

```
1 {  
2   "_index": "devteam",  
3   "_type": "member",  
4   "_id": "1",  
5   "found": false  
6 }
```

# 파일을 이용한 데이터 처리

- 입력 및 갱신 할 내용을 파일로 저장하여 처리 가능

## [ 파일을 이용한 도큐먼트 생성 ]

```
[ec2-user@ip-172-31-14-124 ~]$ echo '  
> {  
>   "name" : "grkim",  
>   "email" : "",  
>   "sex" : "male"  
> }' > new_member.txt  
[ec2-user@ip-172-31-14-124 ~]$ ls  
Download  new_member.txt  
[ec2-user@ip-172-31-14-124 ~]$ curl -XPOST localhost:9200/devteam/member [-d @new_member.txt  
{"_index": "devteam", "_type": "member", "_id": "AU7-o1trd8fdvaYykNVQ", "_version": 1, "created": true}
```

## [ 생성 후 조회 ]

The screenshot shows a Postman interface with a successful API call. The URL is `http://52.69.104.153:9200/devteam/member/AU7-o1trd8fdvaYykNVQ`. The request method is GET. The response status is 200 OK with a time of 106 ms. The response body is displayed in Pretty JSON format:

```
1 {  
2   "_index": "devteam",  
3   "_type": "member",  
4   "_id": "AU7-o1trd8fdvaYykNVQ",  
5   "_version": 1,  
6   "found": true,  
7   "_source": {  
8     "name": "grkim",  
9     "email": "",  
10    "sex": "male"  
11  }  
12 }
```

## 벌크 (\_bulk) API를 이용한 배치 작업

- index, create, delete, update의 4가지 동작 처리 가능
- delete를 제외한 동작은 메타 정보와 요청 데이터가 한 쌍
- 메타 정보와 요청 데이터는 한 줄로 입력, 입력이 후 줄 바꿈 필수
- 도큐먼트가 존재할 때 create 명령을 사용할 경우 에러 발생
- 도큐먼트가 존재할 때 index 명령을 사용할 경우 기존 도큐먼트 갱신
- 설정을 통해 UDP 프로토콜 사용 가능

# 벌크 (\_bulk) API를 이용한 데이터 생성

## [ magazines.json 파일 ]

```
{ "create" : { "_index": "books", "_type": "book"} }  
{"title": "The Merchant of Venice", "author": "William Shakespeare", "category": "Comedies", "written": "1596-02-01"}  
{ "create" : { "_index": "books", "_type": "book"} }  
{"title": "Romeo and Juliet", "author": "William Shakespeare", "category": "Tragedies", "written": "1562-12-01T20:00:00"}  
{ "create" : { "_index": "books", "_type": "book"} }  
{"title": "King Lear", "author": "William Shakespeare", "category": "Tragedies", "written": "1603-05-01T04:36:00"}  
{ "create" : { "_index": "books", "_type": "book"} }  
{"title": "Hamlet", "author": "William Shakespeare", "category": "Tragedies", "written": "1599-06-01T12:34:00", "id": "H1"}  
{ "create" : { "_index": "books", "_type": "book"} }  
{"title": "Othello", "author": "William Shakespeare", "category": "Tragedies", "written": "1603-07-01T13:34:00", "id": "O1"}  
{ "create" : { "_index": "books", "_type": "book"} }  
{"title": "The Adventures of Tom Sawyer", "author": "Mark Twain", "category": "Folk", "written": "1876-03-01T01:30:00", "id": "TS1"}
```

□ 메타 정보

□ 요청 데이터

## [ \_bulk API ]

```
[ec2-user@ip-172-31-19-190 search]$ curl -XPOST localhost:9200/_bulk --data-binary @magazines.json  
{  
  "took" : 536,  
  "errors" : false,  
  "items" : [ {  
    "create" : {  
      "_index" : "magazines",  
      "_type" : "magazine",  
      "_id" : "AU8XbRgh8agau8xyir3g",  
      "_version" : 1,  
      "status" : 201  
    }  
  }, {
```



elasticsearch

검색

## 검색

- 검색 기능은 질의(query) 명령어를 이용해 수행
- 문자열 형식의 매개변수를 이용한 URI 방식과 HTTP 데이터를 이용한 REQUEST BODY 방식이 있음
- 타입, 인덱스 범위로 질의 가능
- 여러 개의 인덱스를 묶어서 멀티 인덱스 범위로 질의 가능(멀티 테넌시)



## URI 방식

- \_search API를 사용
- 질의는 q 매개변수의 값으로 입력
- 검색결과는 hit 필드에 배열로 표시
- took은 실행에 소요된 시간 (1/1000 초)

[ hamlet을 포함하고 있는 인덱스 검색 ]

GET http://52.68.20.215:9200/books/book/\_search?q=hamlet&pretty Params

Authorization Headers (0) Body Pre-request script Tests

No Auth

Body Cookies Headers (2) Tests Status 200 OK Time 770 ms

Pretty Raw Preview JSON

```
1  {
2   "took": 253,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "failed": 0
8   },
9   "hits": {
10    "total": 1,
11    "max_score": 0.3074455,
12    "hits": [
13      {
14        "_index": "books",
15        "_type": "book",
16        "_id": "AU8Xa3uw8agau8xyir3V",
17        "_score": 0.3074455,
18        "_source": {
19          "title": "Hamlet",
20          "author": "William Shakespeare",
21          "category": "Tragedies",
22          "written": "1599-06-01T12:34:00",
23          "pages": 172,
24          "sell": 14610000,
25          "plot": "The protagonist of Hamlet is Prince Hamlet of Denmark, son of the recently deceased King Claudius, his father's brother and successor. Claudius hastily married King Hamlet's widow, Gertrude, has a long-standing feud with neighbouring Norway, and an invasion led by the Norwegian prince Fortinbras"
        }
      }
    ]
  }
}
```

# 멀티 테넌시 (Multi tenancy)

- 여러 인덱스를 동시에 검색
- 인덱스들을 쉼표로 구분하여 입력
- URI와 REQUEST BODY 방식 모두 사용 가능

[ books, magazines 인덱스에서 동시 검색 ]

GET http://52.68.20.215:9200/books,magazines/\_search?q=time&pretty

Authorization Headers (0) Body Pre-request script Tests

No Auth

Body Cookies Headers (2) Tests Status 200 OK Time 68 ms

Pretty Raw Preview JSON

```
1 {  
2   "took": 17,  
3   "timed_out": false,  
4   "_shards": {  
5     "total": 10,  
6     "successful": 10,  
7     "failed": 0  
8   },  
9   "hits": {  
10    "total": 6,  
11    "max_score": 0.22821909,  
12    "hits": [  
13      {  
14        "_index": "books",  
15        "_type": "book",  
16        "_id": "AU8Xa3uw8agau8xyir3b",  
17        "_score": 0.22821909,  
18        "_source": {  
19          "title": "The Time Machine",  
20          "author": "H. G. Wells",  
21          "category": "Science fiction novel",  
22          "written": "1895-11-01T05:01:00",  
23          "pages": 227,  
24          "sell": 22100000,  
25          "plot": "The book's protagonist is an English scientist and gentleman inventor living in England, and identified by a narrator simply as the Time Traveller. The narrator recounts the dinner guests that time is simply a fourth dimension, and his demonstration of a tabletop machine that can travel through time."  
26        }  
27      }  
28    }  
29  }  
30 }
```

## \_all

- 클러스터의 모든 인덱스 검색 가능
- 성능 저하를 가져옴
- 의도하지 않은 데이터 검색 등 결과 오염 발생
- 사용하지 않는 것을 권장
- 인덱스를 생략해도 같은 효과

[ 모든 인덱스에서 time 검색 ]

GET http://52.68.20.215:9200/\_all/\_search?q=time&pretty

Authorization Headers (0) Body Pre-request script Tests

No Auth

Body Cookies Headers (2) Tests Status 200 OK Time 61ms

Pretty Raw Preview JSON

```
1 {  
2   "took": 12,  
3   "timed_out": false,  
4   "_shards": {  
5     "total": 10,  
6     "successful": 10,  
7     "failed": 0  
8   },  
9   "hits": {  
10    "total": 6,  
11    "max_score": 0.22821909,  
12    "hits": [  
13      {  
14        "_index": "books",  
15        "_type": "book",  
16        "_id": "AU8Xa3uw8agau8xyir3b",  
17        "_score": 0.22821909,  
18        "_source": {  
19          "title": "The Time Machine",  
20          "author": "H. G. Wells",  
21          "category": "Science fiction novel",  
22          "written": "1895-11-01T05:01:00",  
23          "pages": 227,  
24          "sell": 22100000,  
25          "plot": "The book's protagonist is an English scientist and gentleman inventor living in England, and identified by a narrator simply as the Time Traveller. The narrator recounts the dinner guests that time is simply a fourth dimension, and his demonstration of a tabletop mo"}
```

## 검색 필드 지정

- 특정 필드만 지정하여 검색 가능
- 필드명 : 질의어 형태로 사용
- df 매개변수를 이용할 수 있음

[ title 필드에서 time 검색 ]

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://52.68.20.215:9200/\_search?q=title:time
- Authorization:** No Auth
- Headers:** (0)
- Body:** (Empty)
- Pre-request script:** (None)
- Tests:** (None)
- Status:** 200 OK Time 101 ms
- Pretty:** Selected
- Raw:** Raw JSON view
- Preview:** Preview tab
- JSON:** JSON tab

The response body is a JSON object representing the search results:

```
1 {  
2   "took": 7,  
3   "timed_out": false,  
4   "_shards": {  
5     "total": 10,  
6     "successful": 10,  
7     "failed": 0  
8   },  
9   "hits": {  
10    "total": 2,  
11    "max_score": 0.70273256,  
12    "hits": [  
13      {  
14        "_index": "books",  
15        "_type": "book",  
16        "_id": "AU8Xa3uw8agau8xyir3b",  
17        "_score": 0.70273256,  
18        "_source": {  
19          "title": "The Time Machine",  
20          "author": "H. G. Wells",  
21          "category": "Science fiction novel",  
22          "written": "1895-11-01T05:01:00",  
23          "pages": 227,  
24          "sell": 22100000,  
25          "plot": "The book's protagonist is an English scientist and gentleman inventor living in England, and identified by a narrator simply as the Time Traveller. The narrator recounts the dinner guests that time is simply a fourth dimension, and his demonstration of a tabletop model of a time machine."  
26        }  
27      }  
28    ]  
29  }  
30 }
```

The plot field contains a detailed summary of the book's content.

## 검색 연산자

- AND와 OR를 이용하여 조건 명령 가능
- 앞 뒤에 공백을 넣어 지정 (%20)
- default\_operator로 기본 지정 가능

[ title 필드에서 time과 machine으로 AND 검색 ]

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://52.68.20.215:9200/\_search?q=title%20time%20AND%20machine
- Authorization:** No Auth
- Headers (0):** None
- Tests:** q=time machine&default\_operator=AND
- Status:** 200 OK Time 68 ms
- Body:** Pretty (selected), Raw, Preview, JSON
- Response Body (Pretty JSON):**

```
1 {  
2   "took": 17,  
3   "timed_out": false,  
4   "_shards": {  
5     "total": 10,  
6     "successful": 10,  
7     "failed": 0  
8   },  
9   "hits": {  
10    "total": 1,  
11    "max_score": 0.6582823,  
12    "hits": [  
13      {  
14        "_index": "books",  
15        "_type": "book",  
16        "_id": "AU8Xa3uw8agau8xyir3b",  
17        "_score": 0.6582823,  
18        "_source": {  
19          "title": "The Time Machine",  
20          "author": "H. G. Wells",  
21          "category": "Science fiction novel",  
22          "written": "1895-11-01T05:01:00",  
23          "pages": 227,  
24          "sell": 22100000,  
25          "plot": "The book's protagonist is an English scientist and gentleman inventor living in Victorian London who invents a machine that transports him through time to the year 802,705 A.D."  
26        }  
27      }  
28    }  
29  }  
30 }
```

# Explain

- explain 매개변수로 사용
  - 해당 검색 결과의 점수 계산에 사용된 값 출력
  - 점수는 해당하는 데이터의 정확도를 계산한 값
  - 기본적으로 점수가 높을 수록 상위 노출

## [ explain 매개변수를 사용하여 계산 값 출력 ]

GET [http://52.68.20.215:9200/\\_search?q=title:time&explain](http://52.68.20.215:9200/_search?q=title:time&explain)

Authorization Headers (0) Body Pre-request script Tests

No Auth

Body Cookies Headers (2) Tests Status 200 OK Time 64 ms

Pretty Raw Preview JSON ▾

```
28
29  . He reveals that he has built a machine capable of carrying a person, and returns at
30 remarkable tale, becoming the new narrator."
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

The screenshot shows a POST request to the Elasticsearch \_search endpoint with the query `q=title:time&explain`. The response body is displayed in Pretty mode, highlighting the '\_explanation' field which provides a detailed breakdown of the search weight calculation. The explanation for the term 'time' includes its field weight (tf(freq=1.0)), term frequency (termFreq=1.0), and document frequency (idf(docFreq=1, maxDocs=3)). The overall weight for the term is 0.70273256.

## \_source

- \_source 매개변수의 값을 true, false로 지정
- 기본값은 true
- false로 지정할 경우 메타 정보만 출력

[ \_source 매개변수를 사용하여 메타정보만 출력 ]

GET http://52.68.20.215:9200/\_search?q=title:time&\_source=false

Body Cookies Headers (2) Tests Status 200 OK Time 61ms

Pretty Raw Preview JSON

```
1. {
2.   "took": 9,
3.   "timed_out": false,
4.   "_shards": {
5.     "total": 10,
6.     "successful": 10,
7.     "failed": 0
8.   }
9.   "hits": {
10.    "total": 2,
11.    "max_score": 0.70273256,
12.    "hits": [
13.      {
14.        "_index": "books",
15.        "_type": "book",
16.        "_id": "AU8Xa3uw8agau8xyir3b",
17.        "_score": 0.70273256
18.      },
19.      {
20.        "_index": "magazines",
21.        "_type": "magazine",
22.        "_id": "AU8XbRgh8agau8xyir3h",
23.        "_score": 0.30685282
24.      }
25.    ]
26.  }
27. }
```

## 출력 필드 지정

- \_fields 매개변수를 사용
- 각 필드는 쉼표를 사용하여 구분
- 출력 결과에 표시할 필드를 지정할 경우 사용

[ title, author, category 필드만 출력 ]

GET ▾

http://52.68.20.215:9200/\_search?q=title:time&fields=title,author,category

Body Cookies Headers (2) Tests Status 200 OK Time 62 ms

Pretty Raw Preview

JSON ▾



```
12 ▾
13 ▾
14
15
16
17
18 ▾
19 ▾
20
21
22 ▾
23
24
25 ▾
26
27
28
29
30 ▾
31
32
33
34
35 ▾
36 ▾
37
38
39 ▾
40
41
42
43
44
45 }
```

```
"hits": [
  {
    "_index": "books",
    "_type": "book",
    "_id": "AU8Xa3uw8agau8xyir3b",
    "score": 0.70273256,
    "fields": {
      "author": [
        "H. G. Wells"
      ],
      "category": [
        "Science fiction novel"
      ],
      "title": [
        "The Time Machine"
      ]
    }
  },
  {
    "_index": "magazines",
    "_type": "magazine",
    "_id": "AU8XbRgh8agau8xyir3h",
    "score": 0.30685282,
    "fields": {
      "category": [
        "News magazine"
      ],
      "title": [
        "Time"
      ]
    }
  }
]
```

# 정렬

- sort 매개변수를 사용
- 기본적으로 결과는 점수 값을 기준으로 정렬
- 기본적인 정렬 방식은 오름차순
- 검색 정렬 기준을 변경
- sort={필드명}:{정렬방식} 형태로 지정

[ sort 매개변수를 사용하여 정렬 ]

GET http://52.68.20.215:9200/books/\_search?q=author:jules&fields=pages&sort=pages:desc&pretty

Body Cookies Headers (2) Tests Status 200 OK Time 55 ms

Pretty Raw Preview JSON

```
18 ▾
19 ▾
20
21
22
23 ▾
24
25
26
27 ▾
28
29
30
31
32 ▾
33 ▾
34
35
36
37 ▾
38
39
40
41 ▾
42
43
44
45
46 ▾
47 ▾
48
49
50
51 ▾
52
```

```
{
  "fields": {
    "pages": [
      304
    ],
    "sort": [
      304
    ]
  },
  "_index": "books",
  "_type": "book",
  "_id": "A8Xa3uw8agau8xyir3e",
  "_score": null,
  "fields": {
    "pages": [
      212
    ],
    "sort": [
      212
    ]
  },
  "_index": "books",
  "_type": "book",
  "_id": "A8Xa3uw8agau8xyir3f",
  "_score": null,
  "fields": {
    "pages": [
      189
    ],
    "sort": [
      189
    ]
  }
}
```

## 출력 구간 지정 (from, size)

- from은 검색 결과 출력 시작 값
- 미 지정 시 from의 기본값은 0
- size는 from을 시작으로 출력할 도큐먼트 수
- 미 지정 시 size의 기본값은 10
- 너무 많은 도큐먼트를 출력할 시 max\_content\_length

설정의 영향을 받을 수 있으므로 적절한 조정 필요

### [ from, size 매개변수를 이용한 출력 구간 지정 ]

GET [http://52.68.20.215:9200/books/\\_search?q=time&from=1&size=5](http://52.68.20.215:9200/books/_search?q=time&from=1&size=5)

Body Cookies Headers (2) Tests Status 200 OK Time 118 ms

Pretty Raw Preview JSON

```
1 {  
2   "took": 30,  
3   "timed_out": false,  
4   "_shards": {  
5     "total": 5,  
6     "successful": 5,  
7     "failed": 0  
8   },  
9   "hits": {  
10    "total": 4,  
11    "max_score": 0.22821909,  
12    "hits": [  
13      {  
14        "_index": "books",  
15        "_type": "book",  
16        "_id": "AU8Xa3uw8agau8xyir3f",  
17        "_score": 0.22448172,  
18        "source": {  
19          "title": "Around the World in Eighty Days",  
20          "author": "Jules Verne",  
21          "category": "adventure novel",  
22          "written": "1873-07-01T10:30:00",  
23          "pages": 189,  
24          "sell": 27200000,  
25          "plot": "Fogg and Passepartout reach Suez in time. While  
detective named Fix, who has been dispatched from London in search  
robber, Fix mistakes Fogg for the criminal. Since he cannot secure  
travellers to Bombay. Fix becomes acquainted with Passepartout with  
large reward if he gets them to Bombay early. They dock two days at  
26      }  
27    },  
28    {  
29      "_index": "books",  
30      "_type": "book",  
31      "_id": "AU8Xa3uw8aqau8xyir3d".  
32    }  
33  }  
34 }  
35 }
```

## SEARCH TYPE

- search\_type 옵션을 사용
- 검색을 수행하는 방법을 지정

### search\_type에 지정할 수 있는 값들

- query\_then\_fetch : 전체 색드의 검색이 모두 수행된 후 결과 출력, 전체 취합된 결과를 size 매개변수에서 지정한 만큼 출력
- query\_and\_fetch : 색드별로 검색되는 대로 결과 출력, size가 10이고 색드의 개수가 5라면 출력 결과는 색드당 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

## REQUEST BODY 검색

- 검색 조건을 JSON 데이터 형식의 질의로 입력
- URI 검색보다 복잡한 형식으로 검색 가능
- 엘라스틱서치의 질의 언어(QueryDSL) 사용

[ term 쿼리를 이용한 기본적인 검색 ]

POST [http://52.68.20.215:9200/books/\\_search](http://52.68.20.215:9200/books/_search)

Authorization Headers (0) Body Pre-request script

form-data  x-www-form-urlencoded  raw  binary Text ▾

```
1 {  
2   "fields" : [ "title", "author", "category" ],  
3   "sort" : [ { "category" : "desc" } ],  
4   "query" : {  
5     "term" : { "_all" : "time" }  
6   }  
7 }
```

# HIGHLIGHT

- Highlight 옵션을 사용하여 설정
- 기본적으로 <em> 태그를 사용
- pre/post\_tags 옵션으로 태그 변경 가능

[ 문자열 time을 검색하여 highlight 처리 ]

POST [http://52.68.20.215:9200/books/\\_search](http://52.68.20.215:9200/books/_search)

Authorization Headers (0) Body Pre-request script

form-data  x-www-form-urlencoded  raw  binary [Text](#)

```
1 {  
2   "fields" : [ "title", "author", "category" ],  
3   "sort" : [ { "category" : "desc" } ],  
4   "query" : {  
5     "term" : { "_all" : "time" }  
6   },  
7   "highlight" : {  
8     "pre_tags" : [ "<strong>" ],  
9     "post_tags" : [ "</strong>" ],  
10    "fields" : { "title" : {} }  
11  }  
12 }
```

```
56   "_index": "books",  
57   "_type": "book",  
58   "_id": "AU8Xa3uw8agau8xyir3b",  
59   "_score": null,  
60   "fields": {  
61     "author": [  
62       "H. G. Wells"  
63     ],  
64     "category": [  
65       "Science fiction novel"  
66     ],  
67     "title": [  
68       "The Time Machine"  
69     ]  
70   },  
71   "highlight": {  
72     "title": [  
73       "The <strong>Time</strong> Machine"  
74     ]  
75   },  
76   "sort": [  
77     "science"  
78   ]  
79 }
```



elasticsearch

**QueryDSL (질의)**

## 쿼리(Query)와 필터(Filter)

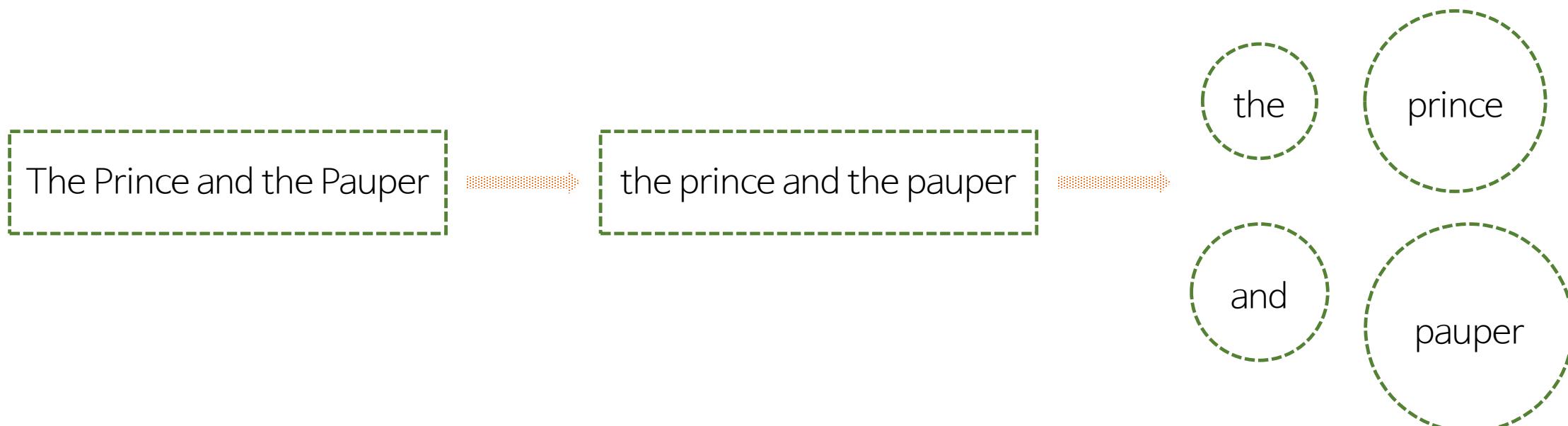
- 엘라스틱서치에서 데이터를 찾고 구분하는 기능은 크게 쿼리와 필터를 이용한 방법으로 나뉨
- 쿼리와 필터는 모두 JSON 형식으로 구현하며 이를 QueryDSL(Domain Specific Language)이라고 함
- 쿼리와 필터는 각각 독립적인 요소로 실행
- 쿼리와 필터 문법은 어그리게이션(Aggregation)과 같은 API의 내부에서도 사용

### 쿼리와 필터의 비교

	쿼리 (Query)	필터 (Filter)
검색 대상 (일반적)	전문검색 (Full Text)	바이너리 구분 (Y/N)
점수 계산	O	X
캐싱	X	O
응답 속도 (상대적)	느림	빠름

## 기본적인 형태소 분석 과정

- 대문자는 모두 소문자로 변환
- 중복 단어 제거
- 분석 과정을 거치고 저장된 토큰을 텀(term)이라 칭함



## 텀(term) 쿼리

- term 옵션을 사용
- 질의문이 저장된 템과 정확히 일치하는 내용을 찾음

[ tile 필드에서 “Prince” 검색 ]

The screenshot shows a search interface with the following details:

- Request URL:** POST http://52.68.89.159:9200/books/\_search
- Authorization:** None
- Headers:** None
- Body:** raw
- Search Query (Body):**

```
1 {  
2   "query": {  
3     "term": {  
4       "title": "Prince"  
5     }  
6   }  
7 }
```

The query part of the body is highlighted with a red dashed box.- Response (Right Panel):**

```
1 {  
2   "took": 3,  
3   "timed_out": false,  
4   "_shards": {  
5     "total": 5,  
6     "successful": 5,  
7     "failed": 0  
8   },  
9   "hits": {  
10    "total": 0,  
11    "max_score": null,  
12    "hits": []  
13  }  
14 }
```

The entire hits section is highlighted with a green dashed box.

## 텀즈(terms) 쿼리

- terms 옵션을 사용
- 2개 이상의 템을 같이 검색할 경우 사용
- 템즈 쿼리의 값은 항상 배열 형식으로 입력
- minimum\_should\_match 옵션을 사용하여 최소 일치 수 지정 가능

[ title 필드에서 the, end, of 템 중 2개 이상 일치하는 도큐먼트 검색 ]

```
POST http://52.68.89.159:9200/books/_search
```

Authorization	Headers (0)	Body
<input type="radio"/> form-data	<input type="radio"/> x-www-form-urlencoded	<input checked="" type="radio"/> raw
		<pre>1 { 2     "query" : { 3         "terms" : { 4             "title" : [ "the", "and", "of" ], 5             "minimum_should_match" : 2 6         } 7     } 8 }</pre>

```
1 {
2     "took": 11,
3     "timed_out": false,
4     "_shards": {
5         "total": 5,
6         "successful": 5,
7         "failed": 0
8     },
9     "hits": {
10        "total": 5,
11        "max_score": 0.31945422,
12        "hits": [
13            {
14                "_index": "books",
15                "_type": "book",
16                "_id": "AU8Xa3uw8agau8xyir3X",
17                "_score": 0.31945422,
18                "_source": {
19                    "title": "The Adventures of Tom Sawyer",
20                    "author": "Mark Twain",
21                    "category": "Folk",
22                    "written": "1876-03-01T01:34:00",
23                    "pages": 220,
24                    "sell": 83200000,
```

## 매치 (Match), 멀티 매치 쿼리

- match 옵션을 사용
- 질의문 또한 형태소 분석을 거친 뒤 사용
- multi\_match 옵션을 사용하여 여러 필드에서 검색 가능
- analyzer 옵션으로 원하는 형태소 분석 적용 가능
- operator 옵션으로 검색 연산자 지정 가능

[ 기본적인 매치(Match) 쿼리 ]

```
POST http://52.68.89.159:9200/books/_search
{
  "query": {
    "match": {
      "title": {
        "query": "The And",
        "operator": "and"
      }
    }
  }
}
```

[ 멀티 매치(Multi match) 쿼리 ]

```
POST http://52.68.89.159:9200/books/_search
{
  "query": {
    "multi_match": {
      "fields": [ "title", "plot" ],
      "query": "prine king"
    }
  }
}
```

## [ 기본적인 불(Bool) 쿼리 형태 ]

# 불(Bool) 쿼리

- 내부의 질의로 다른 쿼리를 포함시켜 사용
- 쿼리를 조건문인 불 조합으로 적용하여 최종 검색 결과를 나타냄

## 불 쿼리에 사용할 수 있는 조건문

- must : 반드시 해당해야 하는 조건 (AND)
- must\_not : 해당돼서는 안됨(NOT)
- Should : 반드시 해당될 필요는 없으나 해당될 경우 더 높은 스코어를 가짐

```
POST http://52.68.89.159:9200/books/_search

1  "query": {
2    "bool": {
3      "must": [
4        {"term": {"title": "the"}},
5      ],
6      "must_not": [
7        {"term": {"plot": "prince"}}
8      ],
9      "should": [
10        {"term": {"title": "time"}},
11        {"term": {"title": "world"}}
12      ]
13    }
14  }
15
16
17

9  "hits": {
10   "total": 10,
11   "max_score": 0.36509055,
12   "hits": [
13     {
14       "_index": "books",
15       "_type": "book",
16       "_id": "AU8Xa3uw8agau8xyir3b",
17       "_score": 0.36509055,
18       "_source": {
19         "title": "The Time Machine",
20         "author": "H. G. Wells",
21         "category": "Science fiction novel",
22         "written": "1895-11-01T05:01:00",
23         "pages": 227,
24         "sell": 22100000,
25         "plot": "The book's protagonist is an Englishman, and identified by a narrator simply as the dinner guests that time is simply a fourth dimension. He reveals that he has built a machine capable of remarkable tale, becoming the new narrator."
26       }
27     },
28     ...
29   ]
30 }
```

## 문자열 쿼리

- URI 검색에서 `q` 매개변수에 사용하는 질의문 형태 사용 가능
- {필드명} : {질의문} 형식으로 필드 지정 가능
- 검색 연산자 사용 가능
- 와일드 카드 사용 가능 (`?`, `*` 등)
- 비교적 가장 다양한 질의 구현 가능

### [ 와일드 카드를 이용한 문자열 쿼리 ]

POST http://52.68.89.159:9200/books/\_search

```
1 {  
2   "query": {  
3     "query_string": {  
4       "query": "title:prin*"  
5     }  
6   }  
7 }
```

```
"hits": {  
  "total": 1,  
  "max_score": 1,  
  "hits": [  
    {  
      "_index": "books",  
      "_type": "book",  
      "_id": "AU8Xa3uw8agau8xyir3Y",  
      "_score": 1,  
      "_source": {  
        "title": "The Prince and the Pauper",  
        "author": "Mark Twain",  
        "category": "Children's literature",  
        "written": "1881-08-01T10:34:00",  
        "pages": 79,  
        "sell": 112100000,  
        "plot": "Tom Carty (youngest son of a family living  
better life, encouraged by the local priest (who has taught h  
sees a prince (the Prince of Wales - Edward VI). Tom is nearl  
and invites Tom into his palace chamber. There the two boys g  
uncanny resemblance. They decide to switch clothes \"temporar  
y change their lives. Tom becomes a prince and the pauper becomes a  
prince."}  
      }  
    }  
  ]  
}
```

## [ pages 필드 값이 50이상, 150미만인 도큐먼트 검색 ]

POST [http://52.68.89.159:9200/books/\\_search](http://52.68.89.159:9200/books/_search)

```
1 {  
2   "query": {  
3     "range": {  
4       "pages": {  
5         "gte": 50, "lt": 150 }  
6     }  
7   }  
8 }  
  
99 _index: "books",  
60   "_type": "book",  
61   "_id": "AU8Xa3uv8agau8xyir3T",  
62   "_score": 1,  
63   "_source": {  
64     "title": "Romeo and Juliet",  
65     "author": "William Shakespeare",  
66     "category": "Tragedies",  
67     "written": "1562-12-01T20:40:00",  
68     "pages": 125,  
69     "sell": 182700000,  
70     "plot": "Meanwhile, Benvolio talks with  
disovers that it stems from unrequited infatuation.  
Mercutio, Romeo attends the ball at the Capulet's  
love with Juliet. Juliet's cousin, Tybalt, is  
by Juliet's father, who doesn't wish to shed blood.  
Romeo sneaks into the Capulet orchard and overhears  
of the Montagues. Romeo makes himself known to  
reconcile the two families through their children."  
71   },  
72   {  
73     "_index": "books",  
74     "_type": "book",  
75     "_id": "AU8Xa3uw8agau8xyir3W",  
76     "_score": 1,  
77     "_source": {  
78       "title": "Othello",  
79       "author": "William Shakespeare",  
80       "category": "Tragedies",  
81       "written": "1603-07-01T13:34:00",  
82       "pages": 100,  
83       "sell": 141200000,  
84       "plot": "Before Brabantio reaches Othello,  
Othello is summoned to advise the senators. Brabantio  
accuses Othello of having seduced his daughter,  
Desdemona. Othello denies the charge, but Desdemona  
confesses that she has fallen in love with Othello.  
Othello promises to marry Desdemona and  
she becomes the new wife of Othello."}  
85   }  
86 }  
87 }  
88 }  
89 }  
90 }
```

## 범위 쿼리

- range 옵션을 사용
- 필드는 숫자 또는 날짜/시간 형식이어야 함
- 날짜/시간은 JSON의 표준 날짜/시간 입력 형식이어야 함

## 범위 쿼리에 사용되는 값

- gte(greater than or equal) : 주어진 값보다 크거나 같음
- gt(greater than) : 주어진 값보다 큼
- lte(less than or equal) : 주어진 값보다 작거나 같음
- lt(less than) : 주어진 값보다 작음

# 퍼지 쿼리

- 레벤슈타인 거리 알고리즘을 기반으로 유사 단어 검색 지원
  - 숫자, 날짜 형식을 대상으로 범위 검색으로 응용 가능
  - fuzziness 값으로 날짜를 입력할 경우 3d, 3y와 같은 날짜 심볼 사용

[ pages 필드의 값을 100을 기준으로 +/- 20 범위 검색 ]

POST ▼ http://52.68.89.159:9200/books/\_search

## 필터

- 스코어를 계산하지 않으므로 쿼리에 비해 월등히 빠름
- 결과가 메모리에 캐싱 됨
- \_cache 옵션을 사용하여 캐싱 여부 설정 가능
- 다른 필터나 쿼리, 어그리게이션 등의 처리에 사용
- 대부분의 옵션은 쿼리와 동일

### [ 필터를 이용한 템즈 쿼리 ]

POST [http://52.68.89.159:9200/books/\\_search](http://52.68.89.159:9200/books/_search)

```
1 {  
2   "filter": {  
3     "terms": {  
4       "title": [ "prince", "king" ]  
5     }  
6   }  
7 }
```

```
1 {  
2   "took": 7,  
3   "timed_out": false,  
4   "_shards": {  
5     "total": 5,  
6     "successful": 5,  
7     "failed": 0  
8   },  
9   "hits": {  
10    "total": 1,  
11    "max_score": 1,  
12    "hits": [  
13      {  
14        "_index": "books",  
15        "_type": "book",  
16        "_id": "AU8Xa3uw8agau8xyir3Y",  
17        "_score": 1,  
18        "source": {  
19          "title": "The Prince and the Pauper",  
20          "author": "Mark Twain",  
21          "category": "Children's literature",  
22          "written": "1881-08-01T10:34:00",  
23          "pages": 79,  
24          "sell": 112100000,  
25          "plot": "Tom Canty (youngest son of a f  
better life, encouraged by the local priest (who  
sees a prince (the Prince of Wales - Edward VI).  
and invites Tom into his palace chamber. There the  
uncanny resemblance. They decide to switch clothe  
game, snatching up an article of national importa  
Edward is trying to escape the brutality of Tom's  
customs and manners. His fellow nobles and palace  
he will go mad. They repeatedly ask him about the  
"}  
26      }  
27    }  
28  }  
29 }
```



elasticsearch

## 어그리게이션 (Aggregations)

# 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

## [ 어그리게이션 문법 구조 ]

```
{  
  "aggs" (또는 aggregations) : {  
    "<어그리게이션 명 1>" : {  
      "<어그리게이션 타입>" : {  
        ... 어그리게이션 문법 ...  
      }  
      , "aggs" : 하위 어그리게이션  
    },  
    "<어그리게이션 명 2>" : {  
      ... 어그리게이션 ...  
    }  
  }  
}
```

"Member\_Name" : "candy boy",  
"Member\_Position" : "crew"  
},  
이렇게 하나의 값이 나왔습니다. 그런대 배열 안에 모든 Member\_Position  
이 crew인 값을 가져올수가 없드라구요 . ㅠㅠ 아시는분 제발 알려주세요

좋아요 공유하기

이정현님이 좋아합니다.

 승현엄 db.testdb.aggregate(  
 { \$match: {Project\_Name: "daum"}},  
 { \$unwind: "\$Member"},  
 { \$match: {Member.Member\_Position: "crew"}},  
 { \$group: { \_id: "\$\_id", member: { \$push: "\$Member.Member\_Name" }}})  
 { "\_id" : ObjectId("54d74609c4b80e6df9a1614a"), "member" : [ "candy boy",  
 "mango boy" ] }

이거 좀 지지고 볶고하면...  
2월 9일 오전 3:27 · 좋아요

단어 자체로 집합을 의미, 다양한 곳에서 지원하는 구조

## 메트릭(Metric) 어그리게이션

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

[ 최소, 최대, 평균, 합계를 구하는 메트릭 어그리게이션 ]

POST http://52.68.20.215:9200/hotels/\_search

Authorization Headers (0) Body

form-data x-www-form-urlencoded raw binary

```
1 {  
2   "aggs" : {  
3     "price_min": {  
4       "min" : { "field" : "price" }  
5     },  
6     "price_max": {  
7       "max" : { "field" : "price" }  
8     },  
9     "price_avg": {  
10      "avg" : { "field" : "price" }  
11    },  
12    "price_sum": {  
13      "sum" : { "field" : "sum" }  
14    }  
15  }  
16 }
```

```
]  
"aggregations": {  
  "price_sum": {  
    "value": 0  
  },  
  "price_min": {  
    "value": 54  
  },  
  "price_avg": {  
    "value": 179.85  
  },  
  "price_max": {  
    "value": 380  
  }  
}
```

## 상태, 확장 상태 어그리게이션

- 메트릭 어그리게이션
- 상태 타입을 사용해서 계산 값을 한번에 표시
- 상태 정보 외에 제곱 합, 변위, 표준편차 값도 확인 가능

[ 상태, 확장 상태 어그리게이션 ]

POST http://52.68.20.215:9200/hotels/\_search

Authorization Headers (0) Body

form-data x-www-form-urlencoded raw binary

```
1 {  
2   "aggs" : {  
3     "price_ex_stats" : {  
4       "extended_stats" : {  
5         "field" : "price"  
6       }  
7     }  
8   }  
9 }
```

```
242   ],  
243   "checkin": "2014-03-04T11:00:00"  
244   }  
245   ]  
246 },  
247 },  
248 "aggregations": {  
249   "price_ex_stats": {  
250     "count": 20,  
251     "min": 54,  
252     "max": 380,  
253     "avg": 179.85,  
254     "sum": 3597,  
255     "sum_of_squares": 833373,  
256     "variance": 9322.627500000002,  
257     "std_deviation": 96.5537544583327,  
258     "std_deviation_bounds": {  
259       "upper": 372.9575089166654,  
260       "lower": -13.257508916665415  
261     }  
262   }  
263 }  
264 }
```

## 버킷(Bucket) 어그리게이션

- 주어진 조건에 해당하는 도큐먼트를 버킷이라는 저장소 단위로 구분
- 새로운 데이터 집합 형성
- 버킷별로 하위 연산을 반복해서 수행 가능
- 레벨이 깊어질 수록 메모리 등의 자원 소모가 심하므로 주의 필요
- 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
- 한 번의 검색으로 질의 내용과 별도의 어그리게이션을 동시 사용 가능

[ 쿼리 결과에 대해 평균 어그리게이션만 사용 ]

```
POST http://52.68.20.215:9200/hotels/_search
Authorization Headers (0) Body
form-data x-www-form-urlencoded raw binary
1 {
2   "query": {
3     "term": { "name": "seoul" }
4   },
5   "aggs": {
6     "avg_price": {
7       "avg": { "field": "price" }
8     }
9   }
10 }

234 "aggregations": {
235   "avg_price": {
236     "value": 241.44444444444446
237   }
238 }
239 }
```

[ 쿼리 결과에 대해 글로벌 어그리게이션으로 생성된 버킷의 하위 어그리게이션으로 평균 계산 ]

```
POST http://52.68.20.215:9200/hotels/_search
Authorization Headers (0) Body
form-data x-www-form-urlencoded raw binary
1 {
2   "query": {
3     "term": { "name": "seoul" }
4   },
5   "aggs": {
6     "all_price": {
7       "global": {},
8       "aggs": [
9         "avg_price": {
10           "avg": { "field": "price" }
11         }
12       ]
13     }
14   }
15 }

234 "aggregations": {
235   "all_price": {
236     "doc_count": 20,
237     "avg_price": {
238       "value": 179.85
239     }
240   }
241 }
242
229
230
231
232
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549
55410
55411
55412
55413
55414
55415
55416
55417
55418
55419
55420
55421
55422
55423
55424
55425
55426
55427
55428
55429
55430
55431
55432
55433
55434
55435
55436
55437
55438
55439
55440
55441
55442
55443
55444
55445
55446
55447
55448
55449
55450
55451
55452
55453
55454
55455
55456
55457
55458
55459
55460
55461
55462
55463
55464
55465
55466
55467
55468
55469
55470
55471
55472
55473
55474
55475
55476
55477
55478
55479
55480
55481
55482
55483
55484
55485
55486
55487
55488
55489
55490
55491
55492
55493
55494
55495
55496
55497
55498
55499
554100
554101
554102
554103
554104
554105
554106
554107
554108
554109
554110
554111
554112
554113
554114
554115
554116
554117
554118
554119
554120
554121
554122
554123
554124
554125
554126
554127
554128
554129
554130
554131
554132
554133
554134
554135
554136
554137
554138
554139
554140
554141
554142
554143
554144
554145
554146
554147
554148
554149
554150
554151
554152
554153
554154
554155
554156
554157
554158
554159
554160
554161
554162
554163
554164
554165
554166
554167
554168
554169
554170
554171
554172
554173
554174
554175
554176
554177
554178
554179
554180
554181
554182
554183
554184
554185
554186
554187
554188
554189
554190
554191
554192
554193
554194
554195
554196
554197
554198
554199
554200
554201
554202
554203
554204
554205
554206
554207
554208
554209
554210
554211
554212
554213
554214
554215
554216
554217
554218
554219
554220
554221
554222
554223
554224
554225
554226
554227
554228
554229
5542210
5542211
5542212
5542213
5542214
5542215
5542216
5542217
5542218
5542219
55422110
55422111
55422112
55422113
55422114
55422115
55422116
55422117
55422118
55422119
554221110
554221111
554221112
554221113
554221114
554221115
554221116
554221117
554221118
554221119
5542211110
5542211111
5542211112
5542211113
5542211114
5542211115
5542211116
5542211117
5542211118
5542211119
55422111110
55422111111
55422111112
55422111113
55422111114
55422111115
55422111116
55422111117
55422111118
55422111119
554221111110
554221111111
554221111112
554221111113
554221111114
554221111115
554221111116
554221111117
554221111118
554221111119
5542211111110
5542211111111
5542211111112
5542211111113
5542211111114
5542211111115
5542211111116
5542211111117
5542211111118
5542211111119
55422111111110
55422111111111
55422111111112
55422111111113
55422111111114
55422111111115
55422111111116
55422111111117
55422111111118
55422111111119
554221111111110
554221111111111
554221111111112
554221111111113
554221111111114
554221111111115
554221111111116
554221111111117
554221111111118
554221111111119
5542211111111110
5542211111111111
5542211111111112
5542211111111113
5542211111111114
5542211111111115
5542211111111116
5542211111111117
5542211111111118
5542211111111119
55422111111111110
55422111111111111
55422111111111112
55422111111111113
55422111111111114
55422111111111115
55422111111111116
55422111111111117
55422111111111118
55422111111111119
554221111111111110
554221111111111111
554221111111111112
554221111111111113
554221111111111114
554221111111111115
554221111111111116
554221111111111117
554221111111111118
554221111111111119
5542211111111111110
5542211111111111111
5542211111111111112
5542211111111111113
5542211111111111114
5542211111111111115
5542211111111111116
5542211111111111117
5542211111111111118
5542211111111111119
55422111111111111110
55422111111111111111
55422111111111111112
55422111111111111113
55422111111111111114
55422111111111111115
55422111111111111116
55422111111111111117
55422111111111111118
55422111111111111119
554221111111111111110
554221111111111111111
554221111111111111112
554221111111111111113
554221111111111111114
554221111111111111115
554221111111111111116
554221111111111111117
554221111111111111118
554221111111111111119
5542211111111111111110
5542211111111111111111
5542211111111111111112
5542211111111111111113
5542211111111111111114
5542211111111111111115
5542211111111111111116
5542211111111111111117
5542211111111111111118
5542211111111111111119
55422111111111111111110
55422111111111111111111
55422111111111111111112
55422111111111111111113
55422111111111111111114
55422111111111111111115
55422111111111111111116
55422111111111111111117
55422111111111111111118
55422111111111111111119
554221111111111111111110
554221111111111111111111
554221111111111111111112
554221111111111111111113
554221111111111111111114
554221111111111111111115
554221111111111111111116
554221111111111111111117
554221111111111111111118
554221111111111111111119
5542211111111111111111110
5542211111111111111111111
5542211111111111111111112
5542211111111111111111113
5542211111111111111111114
5542211111111111111111115
5542211111111111111111116
5542211111111111111111117
5542211111111111111111118
5542211111111111111111119
55422111111111111111111110
55422111111111111111111111
55422111111111111111111112
55422111111111111111111113
55422111111111111111111114
55422111111111111111111115
55422111111111111111111116
55422111111111111111111117
55422111111111111111111118
55422111111111111111111119
554221111111111111111111110
554221111111111111111111111
554221111111111111111111112
554221111111111111111111113
554221111111111111111111114
554221111111111111111111115
554221111111111111111111116
554221111111111111111111117
554221111111111111111111118
554221111111111111111111119
5542211111111111111111111110
5542211111111111111111111111
5542211111111111111111111112
5542211111111111111111111113
5542211111111111111111111114
5542211111111111111111111115
5542211111111111111111111116
5542211111111111111111111117
5542211111111111111111111118
5542211111111111111111111119
55422111111111111111111111110
55422111111111111111111111111
55422111111111111111111111112
55422111111111111111111111113
55422111111111111111111111114
55422111111111111111111111115
55422111111111111111111111116
55422111111111111111111111117
55422111111111111111111111118
55422111111111111111111111119
554221111111111111111111111110
554221111111111111111111111111
554221111111111111111111111112
554221111111111111111111111113
554221111111111111111111111114
554221111111111111111111111115
554221111111111111111111111116
554221111111111111111111111117
554221111111111111111111111118
554221111111111111111111111119
5542211111111111111111111111110
5542211111111111111111111111111
5542211111111111111111111111112
5542211111111111111111111111113
5542211111111111111111111111114
5542211111111111111111111111115
5542211111111111111111111111116
5542211111111111111111111111117
5542211111111111111111111111118
5542211111111111111111111111119
55422111111111111111111111111110
55422111111111111111111111111111
55422111111111111111111111111112
55422111111111111111111111111113
55422111111111111111111111111114
55422111111111111111111111111115
55422111111111111111111111111116
55422111111111111111111111111117
55422111111111111111111111111118
55422111111111111111111111111119
554221111111111111111111111111110
5542211111111111111111111111111111
5542211111111111111111111111111112
5542211111111111111111111111111113
5542211111111111111111111111111114
5542211111111111111111111111111115
5542211111111111111111111111111116
5542211111111111111111111111111117
5542211111111111111111111111111118
5542211111111111111111111111111119
55422111111111111111111111111111110
554221111111111111111111111111111111
554221111111111111111111111111111112
554221111111111111111111111111111113
554221111111111111111111111111111114
554221111111111111111111111111111115
554221111111111111111111111111111116
554221111111111111111111111111111117
554221111111111111111111111111111118
554221111111111111111111111111111119
5542211111111111111111111111111111110
5542211111111111111111111111111111111
5542211111111111111111111111111111112
5542211111111111111111111111111111113
5542211111111111111111111111111111114
5542211111111111111111111111111111115
5542211111111111111111111111111111116
5542211111111111111111111111111111117
5542211111111111111111111111111111118
5542211111111111111111111111111111119
55422111111111111111111111111111111110
554221111111111111111111111111111111111
554221111111111111111111111111111111112
554221111111111111111111111111111111113
554221111111111111111111111111111111114
554221111111111111111111111111111111115
554221111111111111111111111111111111116
554221111111111111111111111111111111117
554221111111111111111111111111111111118
554221111111111111111111111111111111119
5542211111111111111111111111111111111110
5542211111111111111111111111111111111111
5542211111111111111111111111111111111112
5542211111111111111111111111111111111113
5542211111111111111111111111111111111114
5542211111111111111111111111111111111115
5542211111111111111111111111111111111116
5542211111111111111111111111111111111117
5542211111111111111111111111111111111118
5542211111111111111111111111111111111119
55422111111111111111111111111111111111110
55422111111111111111111111111111111111111
55422111111111111111111111111111111111112
554221111111111
```

## 필터 어그리게이션

- 필터 어그리게이션은 주어진 필터에 해당하는 도큐먼트 버킷 생성

[ term 필터를 사용하여 만들어진 버킷으로 평균값 계산]

```
POST http://52.68.89.159:9200/hotels/_search
Authorization: 
Headers (0)
Body
form-data x-www-form-urlencoded raw binary
1 {
2   "aggs": {
3     "filter_name": {
4       "filter": {
5         "term": { "name": "seoul" }
6       },
7       "aggs": {
8         "avg_price": {
9           "avg": { "field": "price" }
10        }
11      }
12    }
13  }
14 }
```

```
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
  "Bar",
  "checkin": "2014-03-04T11:00:00"
}
],
"aggregations": {
  "filter_name": {
    "doc_count": 9,
    "avg_price": {
      "value": 241.44444444444446
    }
  }
}
```

## 누락 어그리게이션

- 지정한 필드가 존재하지 않거나 필드 값이 null인 도큐먼트 버킷 생성

[ term 필터를 사용하여 만들어진 버킷으로 평균값 계산]

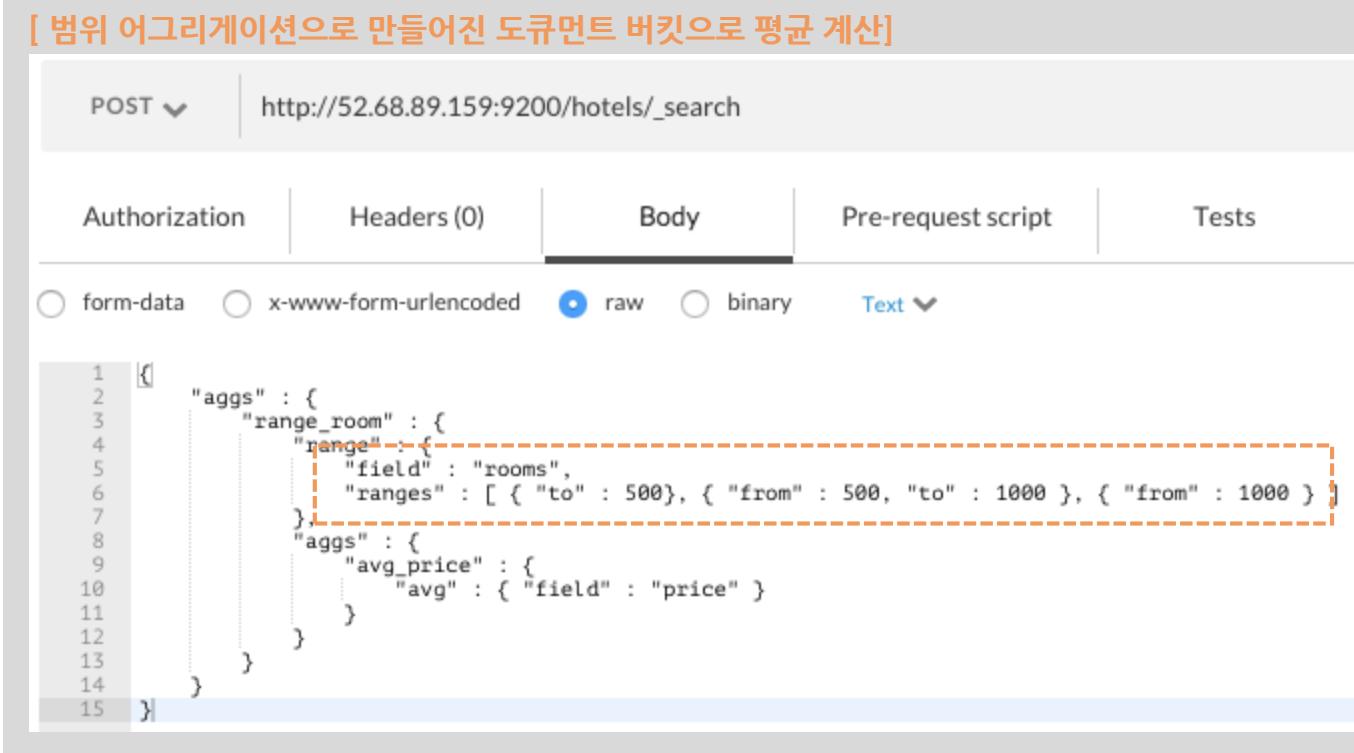
```
POST http://52.68.89.159:9200/hotels/_search
Authorization Headers (0) Body
form-data x-www-form-urlencoded raw binary
1 {
2   "aggs": {
3     "mission_service": {
4       "missing": { "field": "service" },
5       "aggs": {
6         "avg_price": {
7           "avg": { "field": "price" }
8         }
9       }
10      }
11    }
12 }
```

```
235 "city": "Seoul",
236 "address": "72, Digital-ro 32-gil",
237 "price": 189,
238 "internet": true,
239 "service": [
240   "Conference space",
241   "Bar"
242 ],
243 "checkin": "2014-03-04T11:00:00"
244
245
246
247
248 "aggregations": {
249   "mission_service": {
250     "doc_count": 5,
251     "avg_price": {
252       "value": 76
253     }
254   }
255 }
256 }
```

## 범위 어그리게이션

- 설정한 값의 범위 별로 버킷을 생성할 수 있는 버킷 어그리게이션
- 날짜 형식 필드 값을 사용하여 날짜 범위 어그리게이션 사용 가능
- 날짜 지정 형식은 반드시 JSON 표준 날짜-시간 입력 형식으로 입력

[ 범위 어그리게이션으로 만들어진 도큐먼트 버킷으로 평균 계산]



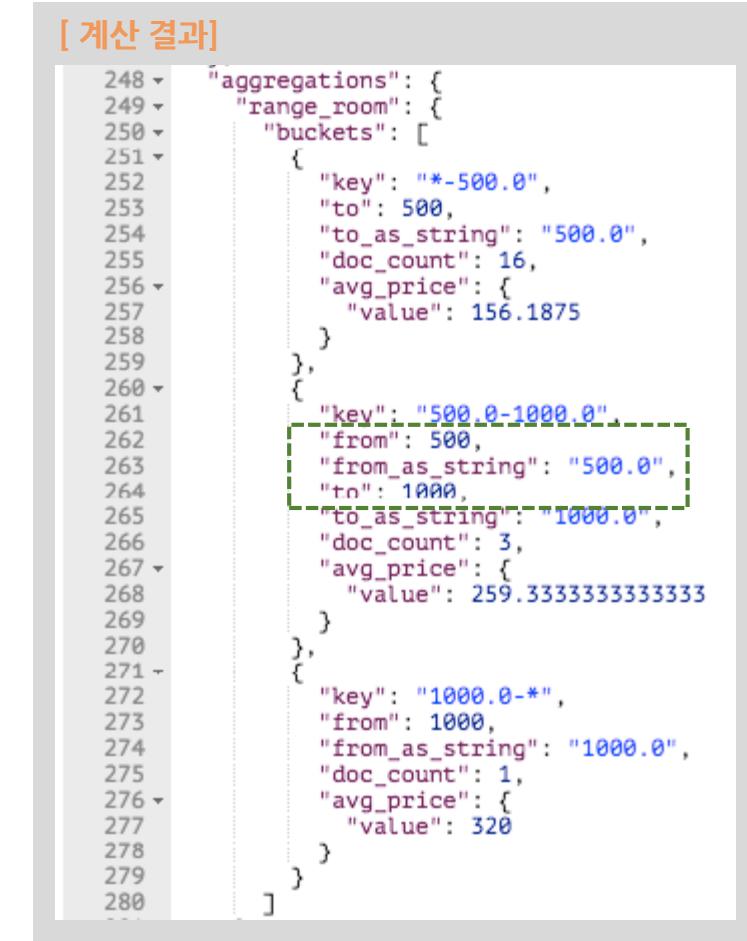
POST [http://52.68.89.159:9200/hotels/\\_search](http://52.68.89.159:9200/hotels/_search)

Authorization Headers (0) Body Pre-request script Tests

form-data x-www-form-urlencoded raw binary Text

```
1 {  
2   "aggs" : {  
3     "range_room" : {  
4       "range" : {  
5         "field" : "rooms",  
6         "ranges" : [ { "to" : 500}, { "from" : 500, "to" : 1000 }, { "from" : 1000 } ]  
7       },  
8       "aggs" : {  
9         "avg_price" : {  
10           "avg" : { "field" : "price" }  
11         }  
12       }  
13     }  
14   }  
15 }
```

[ 계산 결과]



```
248 "aggregations": {  
249   "range_room": {  
250     "buckets": [  
251       {  
252         "key": "*-500.0",  
253         "to": 500,  
254         "to_as_string": "500.0",  
255         "doc_count": 16,  
256         "avg_price": {  
257           "value": 156.1875  
258         }  
259       },  
260       {  
261         "key": "500.0-1000.0",  
262         "from": 500,  
263         "from_as_string": "500.0",  
264         "to": 1000,  
265         "to_as_string": "1000.0",  
266         "doc_count": 3,  
267         "avg_price": {  
268           "value": 259.3333333333333  
269         }  
270       },  
271       {  
272         "key": "1000.0-*",  
273         "from": 1000,  
274         "from_as_string": "1000.0",  
275         "doc_count": 1,  
276         "avg_price": {  
277           "value": 320  
278         }  
279       }  
280     ]  
281   }  
282 }
```



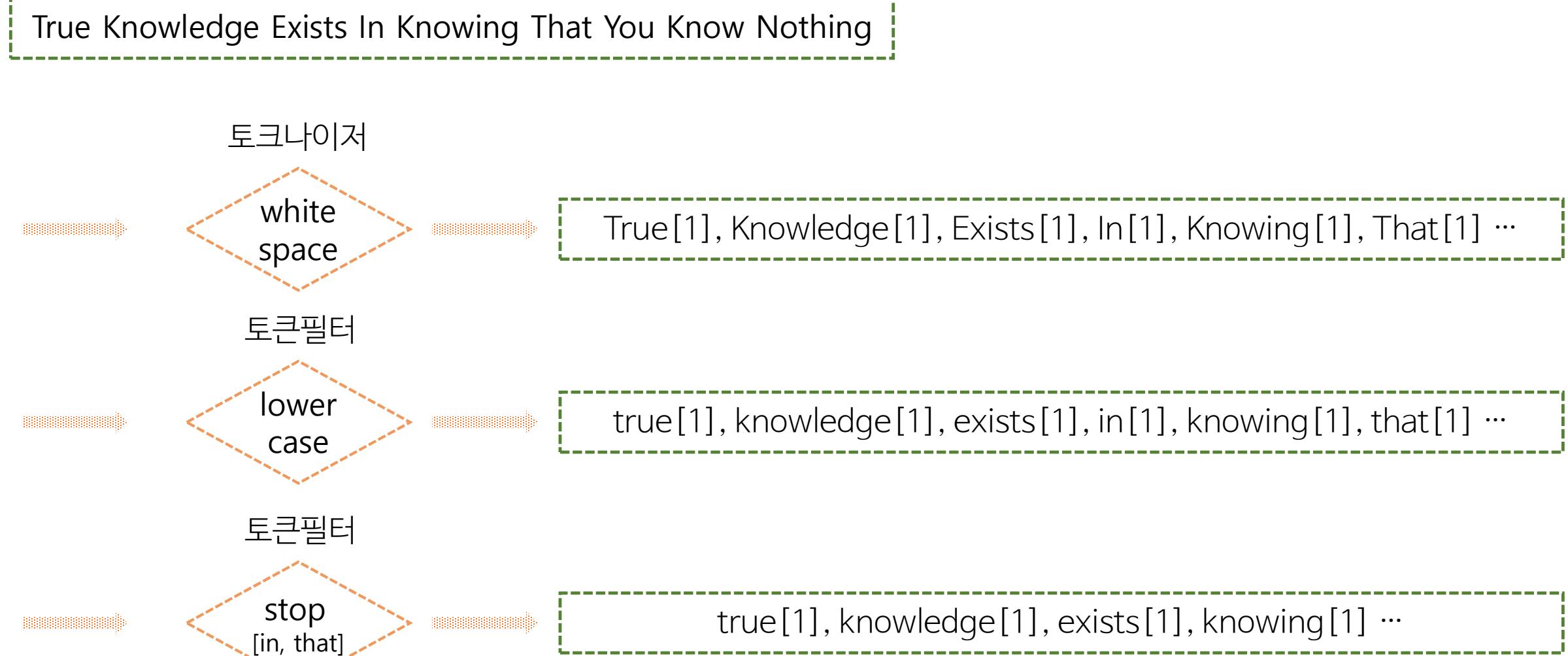
elasticsearch

분석기

## 분석

- 검색어를 추출하기 위한 **프로세스**를 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

## 분석 과정



## \_analyze API

- 입력한 데이터를 분석한 결과를 확인 가능

[ whitespace 토크나이저 적용 및 확인 ]

POST 54.64.58.229:9200/\_analyze?tokenizer=whitespace

Body

Text

```
1 'True Knowledge Exists In Knowing That You Know Nothing'
```

tokens

```
1 "tokens": [
2   {
3     "token": "True",
4     "start_offset": 0,
5     "end_offset": 5,
6     "type": "word",
7     "position": 1
8   },
9   {
10    "token": "Knowledge",
11    "start_offset": 6,
12    "end_offset": 15,
13    "type": "word",
14    "position": 2
15  },
16  {
17    "token": "Exists",
18    "start_offset": 16,
19    "end_offset": 22,
20  }
```

[ lowercase 토큰필터 적용 및 확인 ]

POST 54.64.58.229:9200/\_analyze?tokenizer=whitespace&filters=lowercase

Body

Text

```
1 'True Knowledge Exists In Knowing That You Know Nothing'
```

tokens

```
1 "tokens": [
2   {
3     "token": "true",
4     "start_offset": 0,
5     "end_offset": 5,
6     "type": "word",
7     "position": 1
8   },
9   {
10    "token": "knowledge",
11    "start_offset": 6,
12    "end_offset": 15,
13    "type": "word",
14    "position": 2
15  }
```

## 한글 형태소 분석기

- 영어가 아닌 우리말로 된 문장을 분석하려면 기본적으로 제공하는 분석기만으로는 부족
- 한글을 의미로 해석해서 분리할 수 있는 별도의 한글 형태소 분석기 필요
- 엘라스틱서치나 아파치 루씬에서는 [한글을 위한 별도의 분석기](#)는 제공하지 않음
- 한글 분석기는 오픈소스로 개발되어 공개되고 있으며 대표적으로는 [아리랑](#), [은전한닢](#) 분석기가 있음
- 은전한닢 프로젝트는 2015년 8월 20일 JVM버전 공개 (Scala로 작성)

[ 은전한닢 프로젝트 ]

카카오토픽에서 은전한닢 프로젝트를 사용합니다.



KakaoTopic

카카오토픽 클러스터 시스템에서 은전한닢 프로젝트를 사용합니다.

작성자: 이용운 시간: 오후 1:40 댓글 1개:



라벨: 사용자, 은전한닢

[이 게시물에 대한 링크](#)



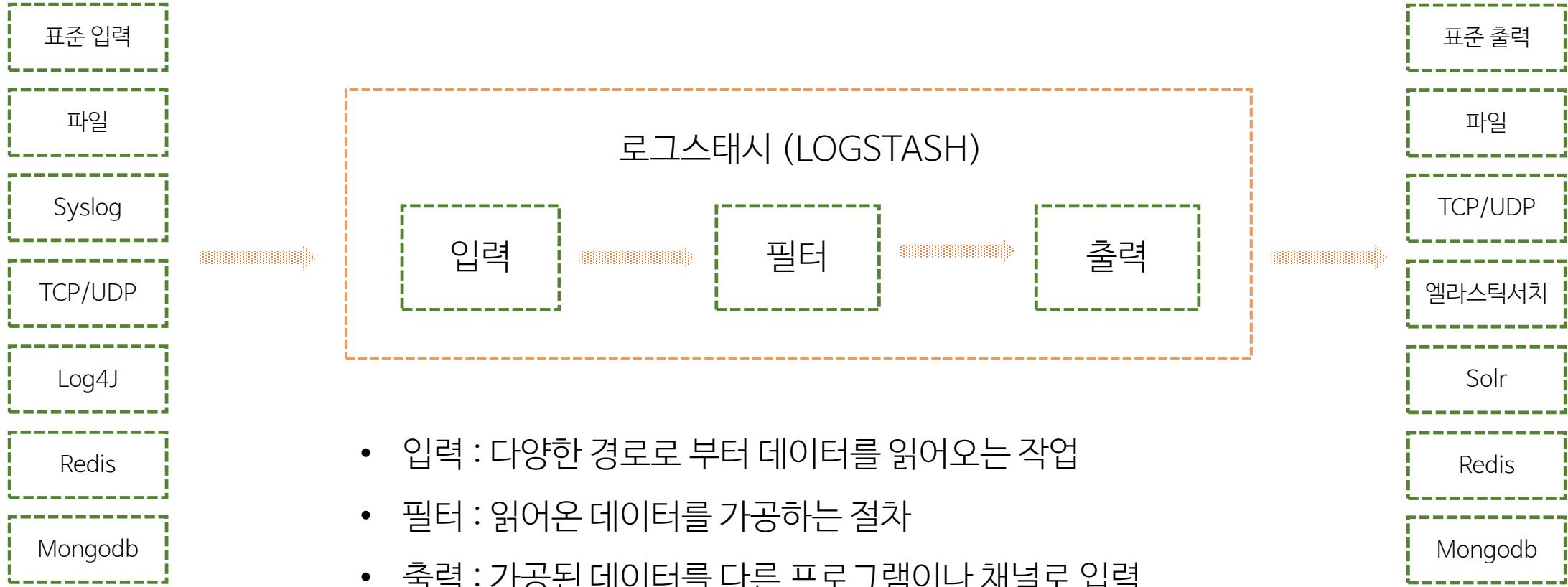
elasticsearch

**LOG STASH와 결합**

## LOGSTASH

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

# 입력, 필터, 출력



# 입력, 필터, 출력 지원 현황

2015년 8월 LOGSTASH 사이트 참고

inputs	codecs	filters	outputs
• collectd	• cloudtrail	• advisor	• boundary
• drupal_dblog	• collectd	• alter	• circonus
• elasticsearch	• compress_spooler	• anonymize	• cloudwatch
• eventlog	• dots	• checksum	• csv
• exec	• edn	• cidr	• datadog
• file	• edn_lines	• cipher	• datadog_metrics
• ganglia	• fluent	• clone	• elasticsearch
• gelf	• graphite	• collate	• elasticsearch_http
• gemfire	• json	• csv	• elasticsearch_river
• generator	• json_lines	• date	• email
• graphite	• json_spooler	• dns	• exec
• heroku	• line	• drop	• file
• imap	• msgpack	• elapsed	• ganglia
• invalid_input	• multiline	• elasticsearch	• gelf
• irc	• netflow	• environment	• gemfire
• jmx	• noop	• extractnumbers	• google_bigquery
• log4j	• oldlogstashjson	• fingerprint	• google_cloud_storage
• lumberjack	• plain	• gelfify	• graphite
• pipe	• rubydebug	• geoip	• graptastic
• puppet_facter	• spool	• grep	• hipchat
• rabbitmq		• grok	• http
• rackspace		• grokdiscovery	• irc
• redis		• i18n	• jira
• relp		• json	• juggernaut
• s3		• json_encode	• librato
• snmptrap		• kv	• loggly
• sqlite		• metaevent	• lumberjack
• sqs		• metrics	• metriccatcher
• stdin		• multiline	• mongodb
• stomp		• mutate	• nagios
• syslog		• noop	• nagios_nsca
• tcp		• prune	• null
• twitter		• punct	• opentsdb
• udp		• railsparallelrequest	• pagerduty
• unix		• range	• pipe
• varnishlog		• ruby	• rabbitmq
• websocket		• sleep	• rackspace
• wmi		• split	• redis
• xmpp		• sumnumbers	• redmine
• zenoss		• syslog_pri	• riak
• zeromq		• throttle	• riemann
		• translate	• s3
		• unique	• sns
		• urldecode	• solr_http
		• useragent	• sqs
		• uuid	• statsd
		• wms	• stdout
		• wmts	• stomp
		• xml	• syslog

## 설정 파일

- 설정은 별도의 파일로 저장
- 실행 시 -f 옵션과 함께 사용
- input, filter, output 순서로 작성하며 내용은 중괄호 안에 입력
- 입, 출력 경로가 여러 개일 때에는 줄 바꿈으로 구분, 쉼표는 입력하지 않음

### [ 로그스태시 설정 파일 구조 ]

```
input {
    <입력경로> {
        <옵션명> => <옵션값>
    }
}

filter {
    <필터> {
        <옵션명> > <옵션값>
    }
    if <필드명> = <필드값> {
        <필터> {
            <옵션명> => <옵션값>
        }
    }
}

output {
    <출력경로> {
        <옵션명> => <옵션값>
    }
}
```

# 입출력

- codec 옵션을 사용하여 데이터 형식 지정(기본값은 Plain)
- 입력 데이터의 변화를 로그스타시가 계속 감시

## [ 로그스타시 설정 파일 ]

```
input {
    stdin {}
}

output {
    stdout {
        codec => json
    }
    file {
        codec => json
        path => "output.txt"
    }
}
```

## [ JSON 형식의 표준 출력과 파일 출력 ]

```
[ec2-user@ip-172-31-19-190 bin]$ ./logstash -f standard.conf
Logstash startup completed
Hello Logstash
{"message": "Hello Logstash", "@version": "1", "@timestamp": "2015-08-13T06:38:08.880Z", "host": "ip-172-31-19-190"}Hello ElasticSearch
{"message": "Hello ElasticSearch", "@version": "1", "@timestamp": "2015-08-13T06:38:29.999Z", "host": "ip-172-31-19-190"}
```

```
[ec2-user@ip-172-31-19-190 bin]$ cat output.txt
{"message": "Hello Logstash", "@version": "1", "@timestamp": "2015-08-13T06:38:08.880Z", "host": "ip-172-31-19-190"}
{"message": "Hello ElasticSearch", "@version": "1", "@timestamp": "2015-08-13T06:38:29.999Z", "host": "ip-172-31-19-190"}
```

## 필터

- 출력 결과를 변경할 수 있는 다양한 필터를 제공
- 조건문을 사용하여 특정 조건 별 필터링 가능

### [ 로그스태시 설정 파일 ]

```
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"  
    }  
}
```

### [ 필터가 적용된 출력 ]

```
[ec2-user@ip-172-31-19-190 bin]$ cat output.txt  
{"name": "sheom", "mobile": "010-1111-2222", "part": "Product Developer", "@version": "1", "@timestamp": "2015-08-13T07:19:21.203Z", "host": "ip-172-31-19-190", "email": "sheom@okiconcession"}
```

## 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

# elasticsearch 옵션으로 로그 파일을 엘라스틱서치로 출력

## [ 엘라스틱서치로 출력하기 위한 절차 ]

```
[ec2-user@ip-172-31-19-190 logs]$ ls  
20141101.log 20141105.log 20141109.log 20141113.log 20141117.log 20141121.log 20141125.log 20141129.log  
20141102.log 20141106.log 20141110.log 20141114.log 20141118.log 20141122.log 20141126.log 20141130.log  
20141103.log 20141107.log 20141111.log 20141115.log 20141119.log 20141123.log 20141127.log  
20141104.log 20141108.log 20141112.log 20141116.log 20141120.log 20141124.log 20141128.log  
[ec2-user@ip-172-31-19-190 logs]$
```

```
{"logTime":"2014-11-01T23:24:14","user":"guest","ip":"123.214.74.152","os":"Windows","browser":"Chrome","page":"/Page/Welcome"}  
{"logTime":"2014-11-01T23:33:41","user":"guest","ip":"123.214.74.152","os":"Windows","browser":"Chrome","page":"/Page/Welcome"}  
{"logTime":"2014-11-01T23:33:45","user":"guest","ip":"123.214.74.152","os":"Windows","browser":"Chrome","page":"/Calc/Trans"}
```

```
from master [[coffee][8HK5G-jITPGRePFj458zbQ][ip-172-31-19-190][inet[/172.31.19.190:9300]]]  
8월 13, 2015 7:37:27 오후 org.elasticsearch.node.internal.InternalNode start  
정보 : [writelog] started  
Logstash startup completed  
{"logTime":"2014-11-01T00:18:44","user":"guest","ip":"125.176.242.150","os":"Macintosh","browser":"Safari","page":  
"/Page/Welcome","@version":"1","@timestamp":"2015-08-13T19:41:06.724Z","host": "ip-172-31-19-190","path": "/home/e  
c2-user/data/logs/20141101.log"} {"logTime":"2014-11-01T00:18:45","user":"guest","ip":"125.176.242.150","os": "iPho  
ne","browser": "Safari","page": "/Page/Welcome","@version": "1","@timestamp": "2015-08-13T19:41:06.739Z","host": "ip-1  
72-31-19-190","path": "/home/ec2-user/data/logs/20141101.log"} {"logTime": "2014-11-01T01:18:53","user": "guest","ip":
```

Searched 5 of 5 shards. 2870 hits. 0.046 seconds									
_index	_type	_id	_score	logTime	user	ip	os	browser	page
serverlog	weblog-2015.08.13	AU8olg9WHQu0Hml_DwA7	1	2014-11-24T03:39:40	guest	1.247.88.10	Windows	Chrome	/Comm/
serverlog	weblog-2015.08.13	AU8olg9WHQu0Hml_DwA-	1	2014-11-24T03:42:35	guest	1.247.88.10	Windows	Chrome	/Calc/Vo
serverlog	weblog-2015.08.13	AU8olg9WHQu0Hml_DwA_	1	2014-11-24T03:50:10	guest	211.241.73.254	Windows	MSIE	/Page/W
serverlog	weblog-2015.08.13	AU8olg9WHQu0Hml_DwBC	1	2014-11-24T05:14:32	guest	211.111.22.135	iPhone	Safari	/Page/W
serverlog	weblog-2015.08.13	AU8olg9WHQu0Hml_DwBH	1	2014-11-24T05:54:21	guest	119.64.121.131	Windows	Chrome	/Page/W
serverlog	weblog-2015.08.13	AU8olg9WHQu0Hml_DwBM	1	2014-11-24T09:28:43	guest	112.163.43.184	Windows	MSIE	/User/Si
serverlog	weblog-2015.08.13	AU8olg9XHQu0Hml_DwBP	1	2014-11-24T09:29:48	canhooo@naver.com	112.163.43.184	Windows	MSIE	/Encl/Atc

## [ 로그스태시 설정 파일 ]

```
input {  
  file {  
    codec => json  
    path => "/home/ec2-user/data/logs/*.log"  
  }  
}  
  
output {  
  stdout {  
    codec => json  
  }  
}  
  
elasticsearch {  
  host => "localhost"  
  cluster => "elasticsearch"  
  node_name => "writelog"  
  index => "serverlog"  
  document_type => "weblog-%{+YYYY.MM.dd}"  
}
```



elasticsearch

**KIBANA**

# KIBANA

- ELK 스택 중 하나
- 엘라스틱서치의 복잡한 질의를 편하게 입력 가능
- 입력된 질의를 간편하게 시각화
- config.js 파일을 수정하여 간편하게 설정
- node.js로 작성
- 일부 설정은 엘라스틱서치 인덱스에 저장



# KIBANA와 엘라스틱서치 연결

[ kibana.yaml ]

```
# Kibana is served by a back end server. This controls which port to use.
port: 5601

# The host to bind the server to.
host: "0.0.0.0"

# The Elasticsearch instance to use for all your queries.
elasticsearch_url: "http://localhost:9200"
-----
# preserve_elasticsearch_host true will send the hostname specified in `elasticsearch`. If you set it to false,
# then the host you use to connect to *this* Kibana instance will be sent.
elasticsearch_preserve_host: true

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

# KIBANA 실행

## [ kibana 실행 파일 ]

```
#!/bin/sh
SCRIPT=$0

# SCRIPT may be an arbitrarily deep series of symlinks. Loop until we have the concrete path.
while [ -h "$SCRIPT" ] ; do
    ls=$(ls -ld "$SCRIPT")
    # Drop everything prior to -
    link=$(expr "$ls" : '.*-> \(.*\$)')
    if expr "$link" : '/.*' > /dev/null; then
        SCRIPT="$link"
    else
        SCRIPT=$(dirname "$SCRIPT")/"$link"
    fi
done

DIR=$(dirname "${SCRIPT}")/..
NODE=${DIR}/node/bin/node
SERVER=${DIR}/src/bin/kibana.js

CONFIG_PATH="${DIR}/config/kibana.yml" NODE_ENV="production" exec "${NODE}" "${SERVER}" ${@}
```

## [ kibana 실행 ]

```
[ec2-user@ip-172-31-19-190 bin]$ sudo ./kibana
{"name": "Kibana", "hostname": "ip-172-31-19-190", "pid": 1742, "level": 30, "msg": "Found kibana index", "time": "2015-08-13T23:53:30.868Z", "v": 0}
{"name": "Kibana", "hostname": "ip-172-31-19-190", "pid": 1742, "level": 30, "msg": "Listening on 0.0.0.0:5601", "time": "2015-08-13T23:53:33.358Z", "v": 0}
```

# KIBANA 연결

[ kibana 연결 확인 ]

The screenshot displays the Kibana Discover interface within a web browser. The URL in the address bar is highlighted with a red box. The browser's title bar shows "Discover - Kibana 4". The main content area shows a search bar with a placeholder of "\*", a time range selector set to "Last 15 minutes", and a result count of "0 hits". On the left, there's a sidebar titled "serverlog" with sections for "Selected Fields" (containing "? \_source") and "Available Fields". The main body of the page displays an error message: "No results found 😞". Below this, it says: "Unfortunately I could not find any results matching your search. I tried really hard. I looked all over the place and frankly, I just couldn't find anything good. Help me, help you. Here's some ideas:". It then provides three examples: 1) "Expand your time range" (with a note about date fields and time pickers). 2) "Refine your query" (with a note about Elasticsearch Query String syntax and web server logs). 3) "Examples:" followed by four search queries: "Find requests that contain the number 200, in any field:", "Or we can search in a specific field. Find 200 in the status field:", "Find all status codes between 400-499:", and "Find status codes 400-499 with the extension php:". Each example has a corresponding code snippet in a text input field.

Selected Fields  
? \_source

Available Fields

No results found 😞

Unfortunately I could not find any results matching your search. I tried really hard. I looked all over the place and frankly, I just couldn't find anything good. Help me, help you. Here's some ideas:

Expand your time range

I see you are looking at an index with a date field. It is possible your query does not match anything in the current time range, or that there is no data at all in the currently selected time range. Click the button below to open the time picker. For future reference you can open the time picker by clicking the [time picker](#) in the top right corner of your screen.

Refine your query

The search bar at the top uses Elasticsearch's support for Lucene Query String syntax. Let's say we're searching web server logs that have been parsed into a few fields.

Examples:

Find requests that contain the number 200, in any field:  
200

Or we can search in a specific field. Find 200 in the status field:  
status:200

Find all status codes between 400-499:  
status:[400 TO 499]

Find status codes 400-499 with the extension php:  
status:[400 TO 499] AND extension:PHP

Or HTML  
status:[400 TO 499] AND (extension:php OR extension:html)

# KIBANA 연결

[ Index pattern 설정 ]

Discover Visualize Dashboard Settings

Indices Advanced Objects About

Index Patterns

serverlog

## Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

Index contains time-based events  
 Use event times to create index names

**Index name or pattern**  
Patterns allow you to define dynamic index names using \* as a wildcard. Example: logstash-\*

serverlog

Time-field name i refresh fields  
logTime

Create

# KIBANA 시각화 종류

## [ kibana 시각화 종류 ]

### Area chart

Great for stacked timelines in which the total of all series is more important than comparing any two or more series. Less useful for assessing the relative change of unrelated data points as changes in a series lower down the stack will have a difficult to gauge effect on the series above it.

### Data table

The data table provides a detailed breakdown, in tabular format, of the results of a composed aggregation. Tip, a data table is available from many other charts by clicking grey bar at the bottom of the chart.

### Line chart

Often the best chart for high density time series. Great for comparing one series to another. Be careful with sparse sets as the connection between points can be misleading.

### Markdown widget

Useful for displaying explanations or instructions for dashboards.

### Metric

One big number for all of your one big number needs. Perfect for show a count of hits, or the exact average a numeric field.

### Pie chart

Pie charts are ideal for displaying the parts of some whole. For example, sales percentages by department. Pro Tip: Pie charts are best used sparingly, and with no more than 7 slices per pie.

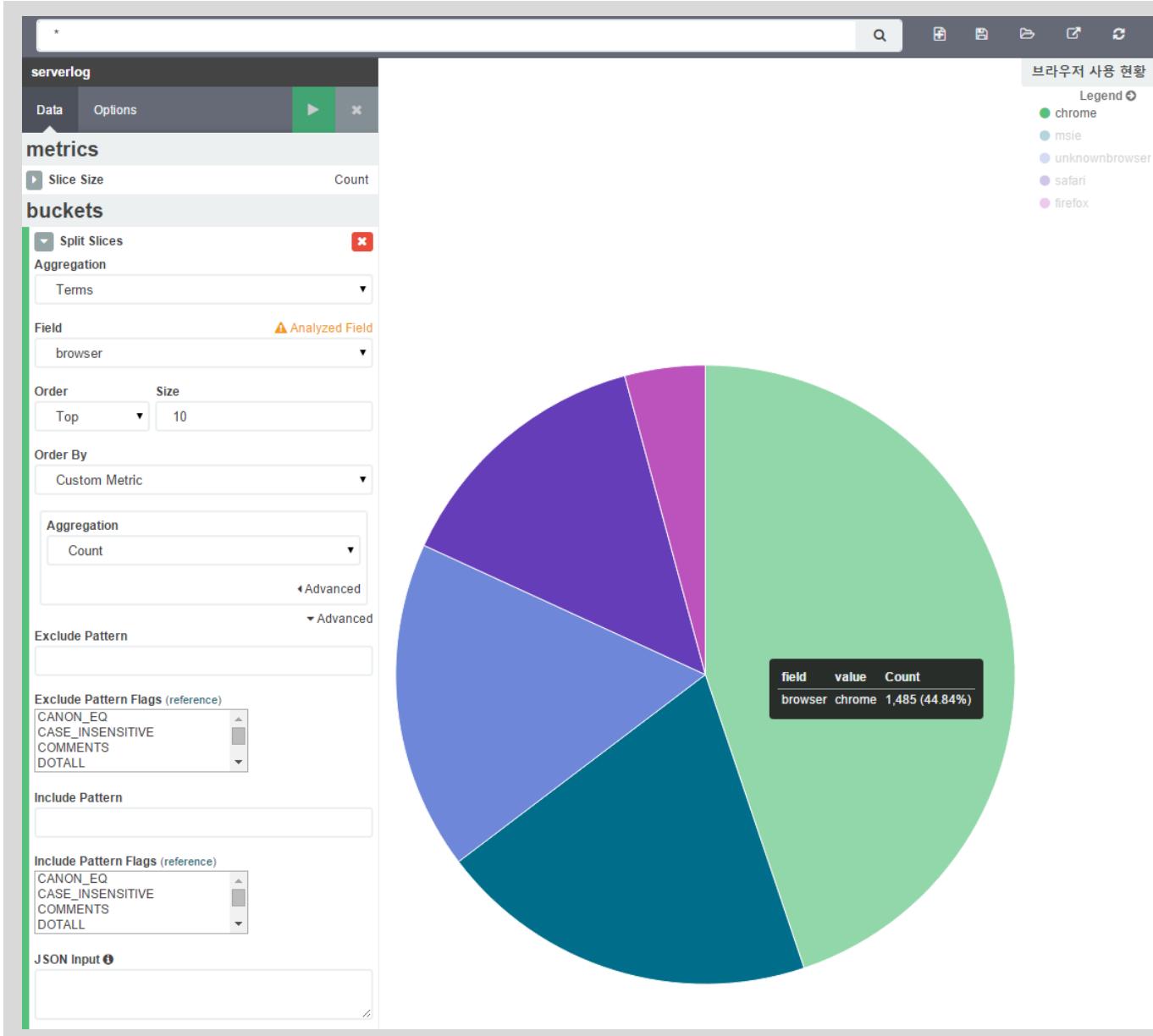
### Tile map

Your source for geographic maps. Requires an elasticsearch geo\_point field. More specifically, a field that is mapped as type:geo\_point with latitude and longitude coordinates.

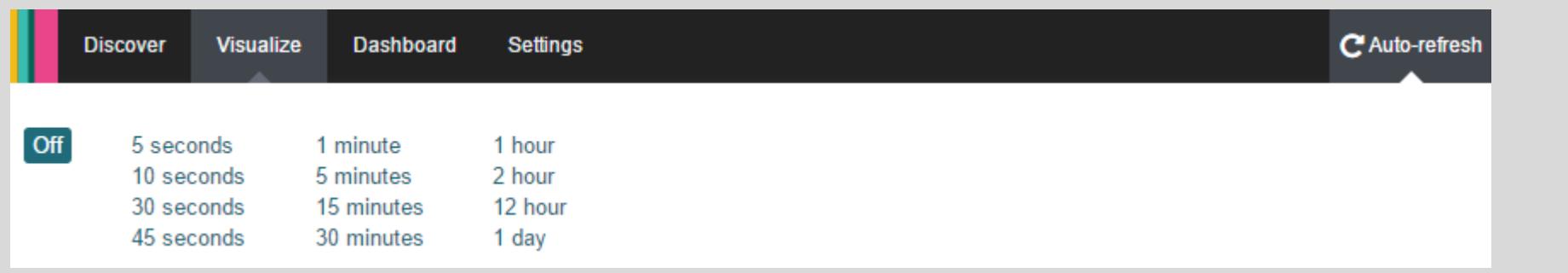
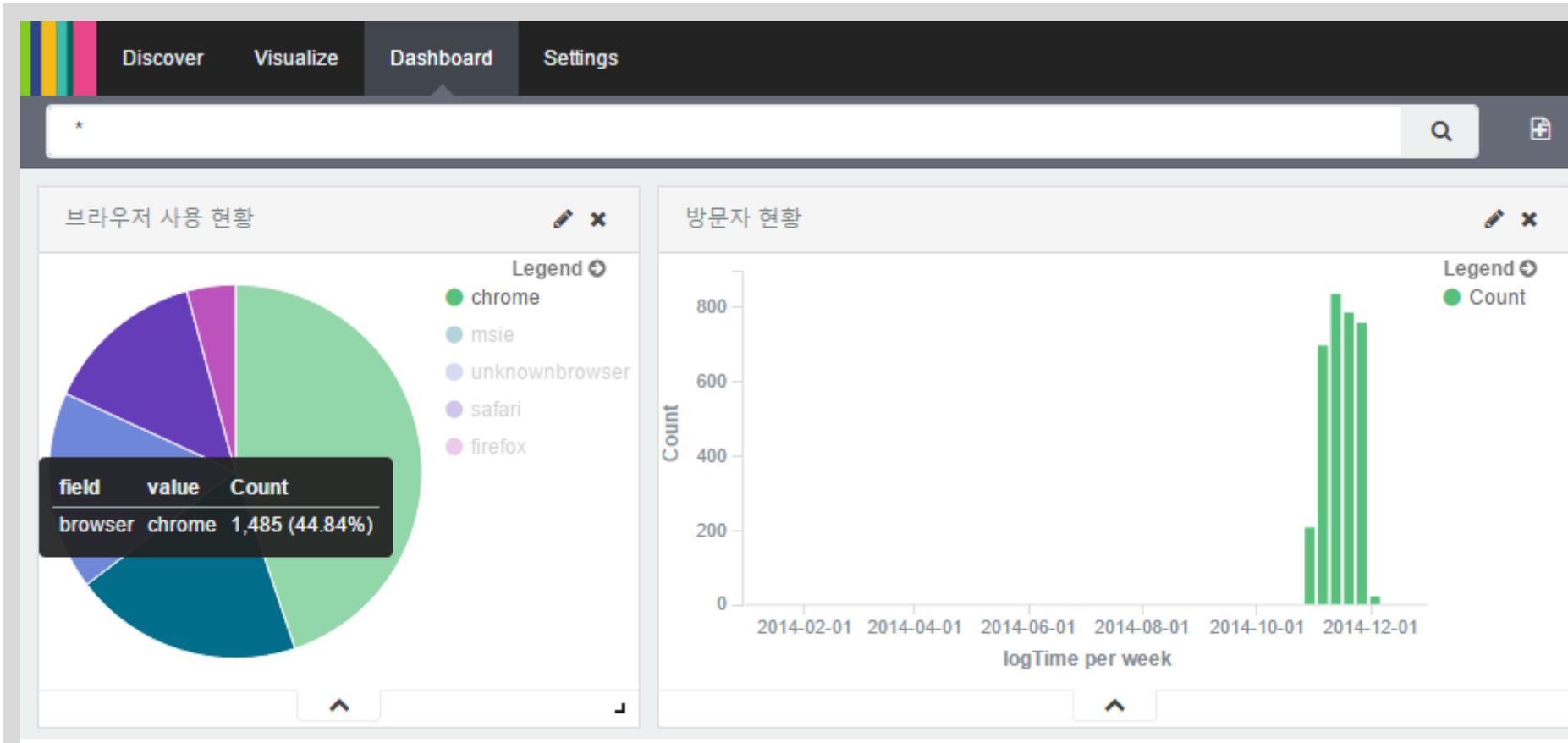
### Vertical bar chart

The goto chart for oh-so-many needs. Great for time and non-time data. Stacked or grouped, exact numbers or percentages. If you are not sure which chart your need, you could do worse than to start here.

# 시각화 생성 (VISUALIZE MENU)



# 대시보드 (DASHBOARD)





# elasticsearch

## 참고자료

- 시작하세요! 엘라스틱서치 ([김종민, 위키북스](#))
- 엘라스틱 웹사이트 (<https://www.elastic.co>)
- 스택오버플로우 (<http://stackoverflow.com>)