Elastic Stack을 활용한 Data Dashboard 만들기

Week 4 - Logstash로 데이터를 전처리하고 전송하자

Fast Campus

목차

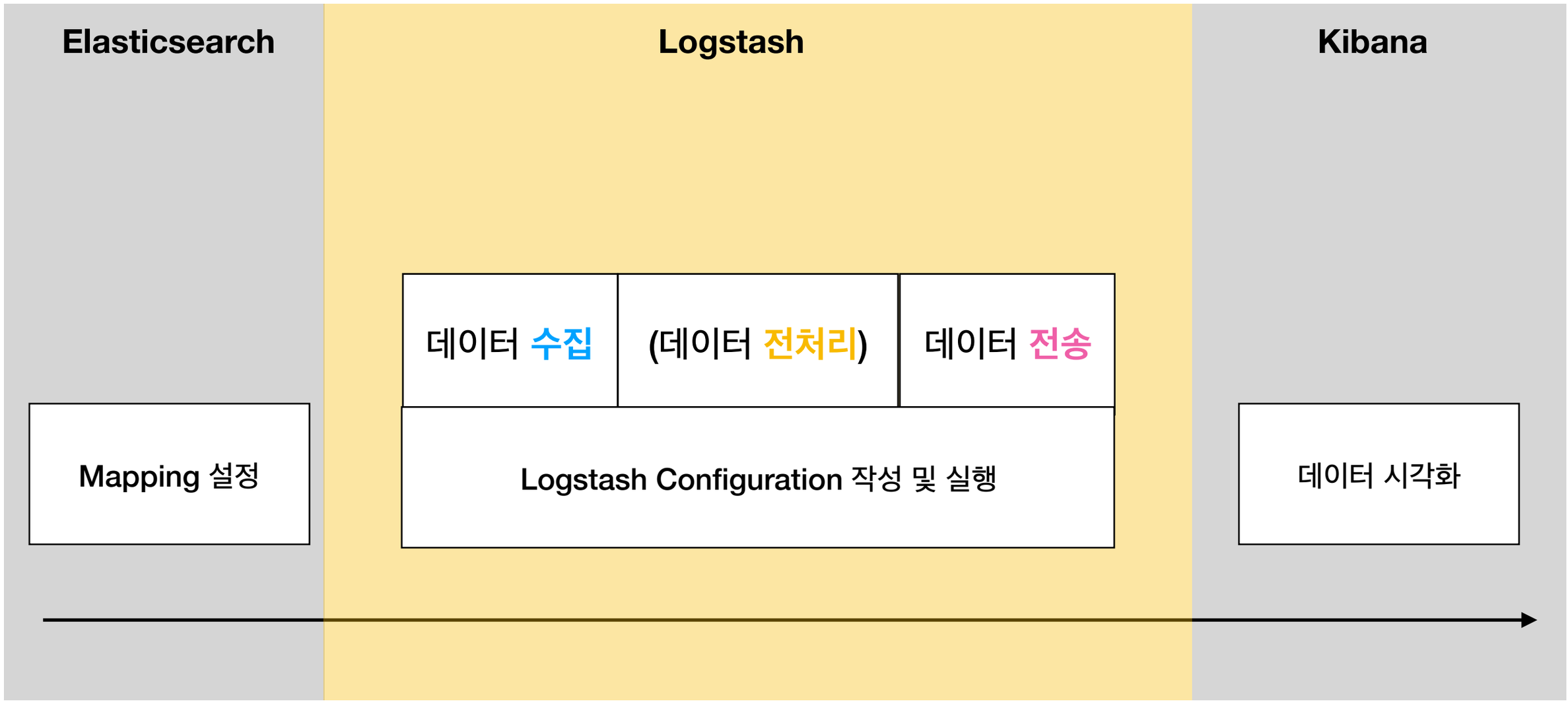
지금까지 시각화도 했고, 검색도 해봤는데 뭔가 2% 부족하다

이번에는 내가 가지고 있는 데이터를 직접 해보고 싶다

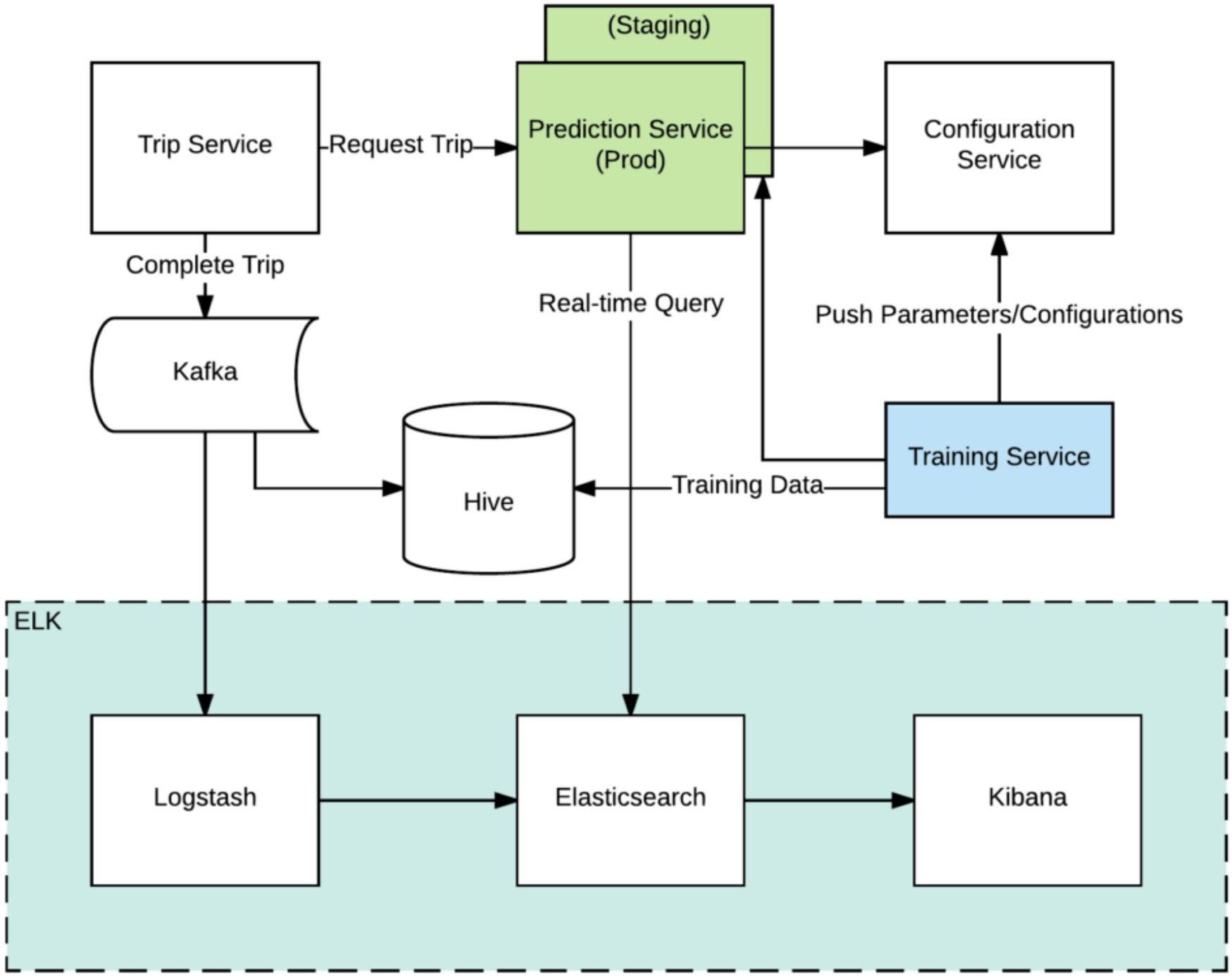
내가 가진 모든 데이터를 Elastic Stack으로 분석할 수 있나?

원본 데이터를 변형할 수도 있나?

이 때 데이터를 필요한 게 Logstash!



(분석가 workflow)



source : Uber ♕ *(data flow)*

logstash 5.6이 공식적으로 지원하는 plugins 중에서...

<-> community plugins 예) mongodb input plugin

용어	종류	예시
수집 ♕	input plugin	54	stdin, file(log, csv …), jdbc(mysql, …), elasticsearch, s3, …
전처리 ♕	filter plugin	47	mutate, csv, date, ruby, if, grok, drop, kv, range, …
전송 ♕	output plugin	55	stdout, elasticsearch, csv, kafka, mongodb, redis, s3, …

… 목적에 맞게 선택해서 사용하면 된다

example)

예시		해석
input plugin 👑	**jdbc**(mysql)	mysql database에서 데이터를 수집해서…
filter plugin 👑	**mutate**, **if**	데이터에 mutate, if filter를 적용하고…
output plugin 👑	**elasticsearch**	elasticsearch에 데이터를 전송해라

input

filter

output

product	quantity	sales
Onepiece	2	39000
Cardigan	1	37000
Knit	3	69000
Jeans	1	78000
T-Shirt	1	89000

다음 상품 ’10% off’ tag 추가

- Onepiece
- Cardigan


```
"hits": {
  "total": 2,
  "max_score": 1,
  "hits": [
    {
      "_index": "week5_higee",
      "_type": "week5_higee",
      "_id": "AWFaqPo-PloSIAlpN8yg",
      "_score": 1,
      "_source": {
        "product": "Onepiece",
        "quantity": 2,
        "sales": 39000,
        "tags": [
          "10% off"
        ]
      }
    },
    {
      "_index": "week5_higee",
      "_type": "week5_higee",
      "_id": "AWFaqQIiPloSIAlpN8yh",
      "_score": 1,
      "_source": {
        "product": "Knit",
        "quantity": 3,
        "sales": 69000
      }
    }
  ]
}
```

그렇다면 logstash를 실제로 활용할 경우 어떤 flow로 생각해야할까?

1. HOME Logstash Home Directory 이동 `$ cd /home/ec2-user/fc/logstash-5.6.4/`

2. conf 생성 Logstash conf 파일 생성 `$ vim exercise.conf` (파일 이름 예시 : exercise.conf)

3. conf 편집 Logstash conf 파일 편집모드 `i`

4. conf 작성 Logstash conf 파일 편집

문제 정의
1. 데이터가 어디에 있는지 (=**input**) 명확히 한다.
2. 데이터를 어디로 전송할지 (=**output**) 명확히 한다.
3. 데이터를 어떻게 변형할지 (=**filter**) 명확히 한다.

코드 작성
4. 적절한 **input** plugin 선택
5. 적절한 **output** plugin 선택
6. 적절한 **filter** plugin 선택

5. conf 저장 Logstash conf 파일 저장 `ESC 입력 후 :wq`

conf 작성이 logstash 핵심

6. 실행 Logstash (conf 파일) 실행 `$ bin/logstash -f exercise.conf`

HOME Logstash Home Directory 이동 `$ cd /home/ec2-user/fc/logstash-5.6.4/`

conf 생성 Logstash conf 파일 생성 `$ vim exercise.conf`

(파일 이름 예시 : exercise.conf)

전반적인 흐름을 기억하자.

편집모드 Logstash conf 파일 편집모드 `i`

큰 흐름은 일정하고 conf 작성하는 부분만 바꾼다.

conf 작성 Logstash conf 파일 편집

문제 정의

2. 데이터를 어디로 전송할지 (=output) 명확히 한다.
3. 데이터를 어떻게 변형할지 (=filter) 명확히 한다.

코드 작성

5. 적절한 output plugin 선택
6. 적절한 filter plugin 선택

(오늘 실습은 이 과정의 반복이다.)

저장 Logstash conf 파일 저장 `ESC 입력 후 :wq`

conf 작성이 logstash 핵심

실행 Logstash (conf 파일) 실행 `$ bin/logstash -f exercise.conf`

```
1  input {
2    stdin {}
3  }
4
5  output {
6    stdout {
7      codec => rubydebug
8    }
9  }
10
11 filter {
12   mutate {
13     split => {"message" => ","}
14     add_field => {
15       "first" => "%{message[0]}"
16       "second" => "%{message[1]}"
17     }
18   }
19 }
```

Input

Output

Filter

```
input {
  stdin {}
}

output {
  stdout {
    codec => rubydebug
  }
}

filter {
  mutate {
    split => {"message" => ","}
    add_field => {
      "first" => "%{message[0]}"
      "second" => "%{message[1]}"
    }
  }
}
```

(p13과 p14의 색깔은 의미가 다릅니다)

1. 데이터가 어디에 있는지 (=**input**) 명확히 한다. `stdin`
2. 데이터를 어디로 전송할지 (=**output**) 명확히 한다. `stdout`
3. 데이터를 어떻게 변형할지 (=**filter**) 명확히 한다. `add_field` `split`
4. 적절한 **input** plugin 선택 `stdin`
5. 적절한 **output** plugin 선택 `stdout`
6. 적절한 **filter** plugin 선택 `mutate`

Input 학습

Output 학습

Filter 학습

구분	목적	input	output	filter
1	logstash 기본	stdin	stdout	
2	option 사용	stdin	stdout	
3	file 데이터 수집	file	stdout	
4	db 데이터 수집	jdbc	stdout	
5	es 데이터 수집	es	stdout	
6	es 데이터 csv 출력	es	csv	
7	(db) 데이터 es 전송	jdbc	es	
8	multiple output plugins	jdbc	stdout, es	
9	csv 필터 적용	jdbc	stdout, es	csv
10	mutate 필터 적용	jdbc	stdout, es	mutate
11	grok 필터 적용	jdbc	stdout, es	grok

Input - stdin

- logstash 실행 후 console에 입력한 값을 받아서 그대로 출력
- 간단한 logstash 테스트 할 때 혹은 디버깅 할 때 주로 사용

Logstash Home Directory로 가자 👑

```
1   $ cd /home/ec2-user/fc/logstash-5.6.4
```

logstash configuration 파일 (stdin.conf) 생성하자 👑

```
1   $ vim stdin.conf
```

Input - stdin

logstash conf 작성 👑

```
1   input {
2     stdin {}
3   }
4
5   output {
6     stdout {}
7   }
```

☝ - Logstash는 input과 output이 최소 1개씩 있어야 한다

- 그러므로 input 학습 실습이지만 기본 output인 stdout을 설정했다
- (Filter는 필수가 아니다)

Input - stdin

logstash를 실행하자 👑

```
1   $ bin/logstash -f stdin.conf
```

Input - stdin

- stdin 입력값
- input을 stdin으로 설정했기에 “hi” 입력하자 결과 출력
- event 1개

```
hi
2018-02-04T07:42:29.894Z ip-172-31-21-251 hi
hello
2018-02-04T07:42:33.009Z ip-172-31-21-251 hello
hello world
2018-02-04T07:42:38.553Z ip-172-31-21-251 hello world
```

- stdout 출력값
- stdin 입력값인 “hello world” 이외의 값은 시스템에서 제공

Input - stdin

logstash conf 작성 👑

- 출력된 결과가 가독성이 좋지 않아 가독성 좋게 변경
- option 사용법 학습 👑

```
1   input {
2     stdin {}
3   }
4
5   output {
6     stdout {
7       codec => rubydebug
8     }
9   }
```

☝ Console 창에서 보기 편하게 하기 위해서 설정

(ruby “awesome_print” library)

Input - stdin

logstash를 실행하면...

```
hi
{
      "@version" => "1",
          "host" => "ip-172-31-21-251",
    "@timestamp" => 2018-02-03T17:37:54.754Z,
       "message" => "hi"
}
hello
{
      "@version" => "1",
          "host" => "ip-172-31-21-251",
    "@timestamp" => 2018-02-03T17:37:57.676Z,
       "message" => "hello"
}
```

☜

- stdout 출력값
- 출력값이 key, value 형식 출력

Input - stdin

매우 단순하지만 stdin / stdout 을 구현하기 위해 거쳤던 단계를 잘 기억하자.

다양한 input / output / filter plugin을 사용해도 큰 틀은 동일하다

1 `$ cd /home/ec2-user/fc/logstash-5.6.4/`

2 `$ vim exercise.conf`

```
input {
  stdin {}
}

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

3 `$ bin/logstash -f exercise.conf`

Input - file

- file 형태 데이터를 수집할 때 사용하는 plugin
- test.log 나 test.csv 등 일반적인 파일형태는 모두 file plugin을 사용한다
- 우선 실습에 사용할 데이터를 다운 받자 👑

```
1  $ cd /home/ec2-user/fc/logstash-5.6.4/
2  $ wget https://raw.githubusercontent.com/higee/elastic/class2/Week4_Logstash/data/titanic.csv
```

Input - file

logstash conf 작성 👑

```
1   input {
2     file {  👉 file에서 데이터를 읽을 것이기에 file 입력
3       path => "/home/ec2-user/fc/logstash-5.6.4/titanic.csv"
4     }  ☝ 파일 경로 입력
5   }
6
7   output {
8     stdout {
9       codec => rubydebug
10    }
11  }
```

Input - file

logstash를 실행하면... 아무 것도 안 나온다.

```
[ec2-user@ip-172-31-21-251 logstash-5.6.4]$ bin/logstash -f file.conf
Sending Logstash's logs to /home/ec2-user/fc/logstash-5.6.4/logs which is now configured via log4j2.properties
[2018-02-05T20:33:24,633][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"netflow", :directory=>"/home/ec2-user/fc/logstash-5.6.4/modules/netflow/configuration"}
[2018-02-05T20:33:24,636][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"fb_apache", :directory=>"/home/ec2-user/fc/logstash-5.6.4/modules/fb_apache/configuration"}
[2018-02-05T20:33:24,822][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>"main", "pipeline.workers"=>2, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>250}
[2018-02-05T20:33:24,976][INFO ][logstash.pipeline        ] Pipeline main started
[2018-02-05T20:33:25,027][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
```

왜 ?

Input - file

- **start_position**이라는 옵션이 default로 **end**로 설정되어 있기 때문이다.
- 즉, 새로 추가되는 데이터만 보겠다는 것인데 이걸 **beginning**으로 해줘야 된다.
- (p35에서 배울 sincedb로 인해) 다른 데이터를 다운 받고 다시 시도하자 👑

```
$ cd /home/ec2-user/fc/logstash-5.6.4/
$ wget https://raw.githubusercontent.com/higee/elastic/class2/Week4_Logstash/data/titanic2.csv
```

Input - file

logstash conf 작성 👑

```
1   input {
2     file {
3       path => "/home/ec2-user/fc/logstash-5.6.4/titanic2.csv"
4       start_position => "beginning"
5     }
6   }
7
8   output {
9     stdout {
10      codec => rubydebug
11    }
12  }
```

Input - file

logstash를 실행하면...

(파일을 처음부터 읽으므로 결과가 제대로 출력)

```
          "host" => "ip-172-31-21-251",
          "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic2.csv",
    "@timestamp" => 2018-02-03T18:29:17.844Z,
       "message" => "95,Coxon,0,3,male,59,0,0,364500,7.25,S"
}
{
      "@version" => "1",
          "host" => "ip-172-31-21-251",
          "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic2.csv",
    "@timestamp" => 2018-02-03T18:29:17.845Z,
       "message" => "96,Shorney,0,3,male,29.69911765,0,0,374910,8.05,S"
}
{
      "@version" => "1",
          "host" => "ip-172-31-21-251",
          "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic2.csv",
    "@timestamp" => 2018-02-03T18:29:17.845Z,
       "message" => "97,Goldschmidt,0,1,male,71,0,0,PC 17754,34.6542,C"
}
```

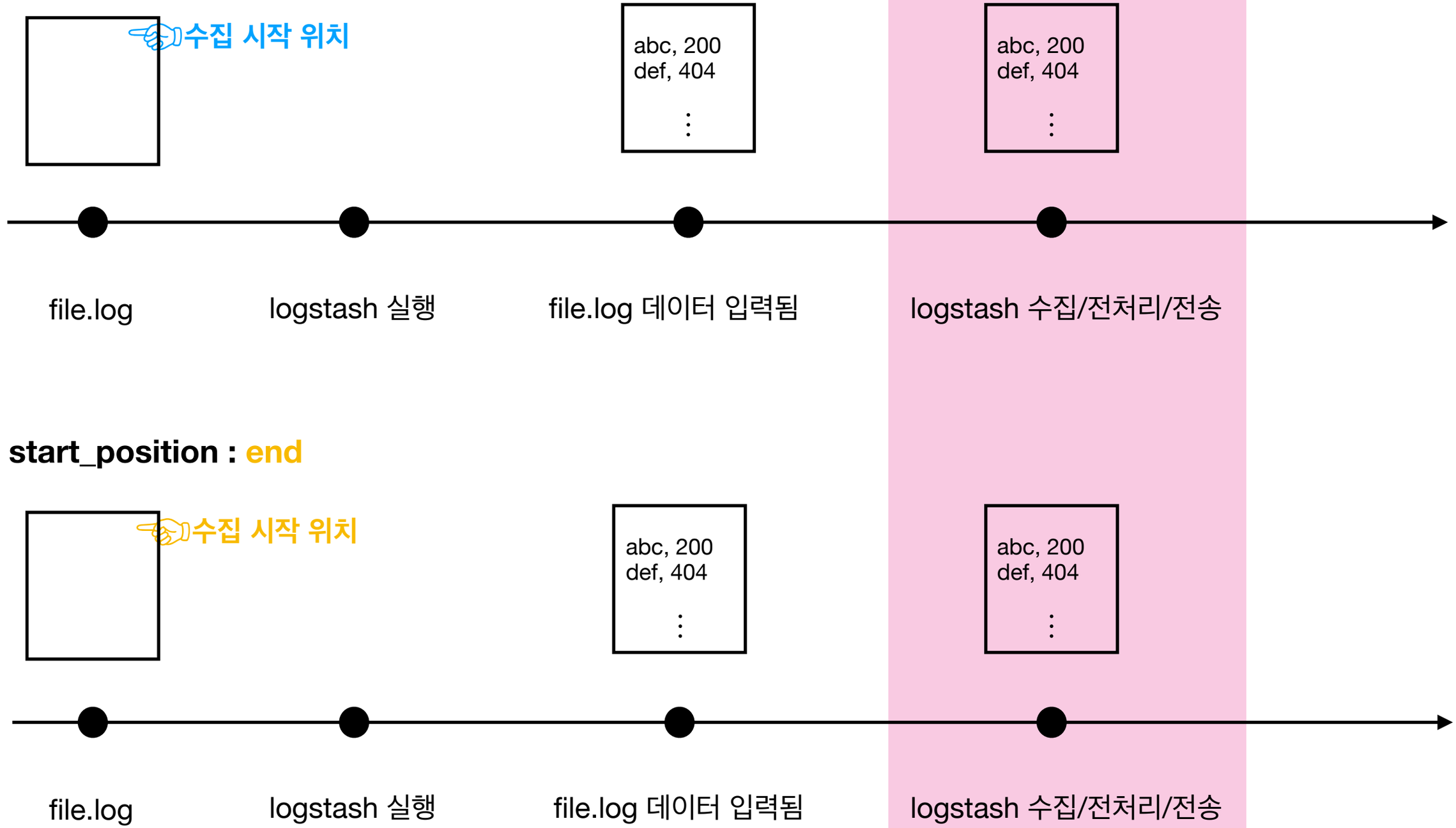
Q. 그래서 start_position option를 어떻게 하라는건지? 👑

A. logstash 실행 전에 file에 이미 데이터가 있는지 아닌지가 중요하다

Input - file 심화

logstash 실행 전 path file에 데이터가 없는 경우 : 차이가 없다

start_position : beginning

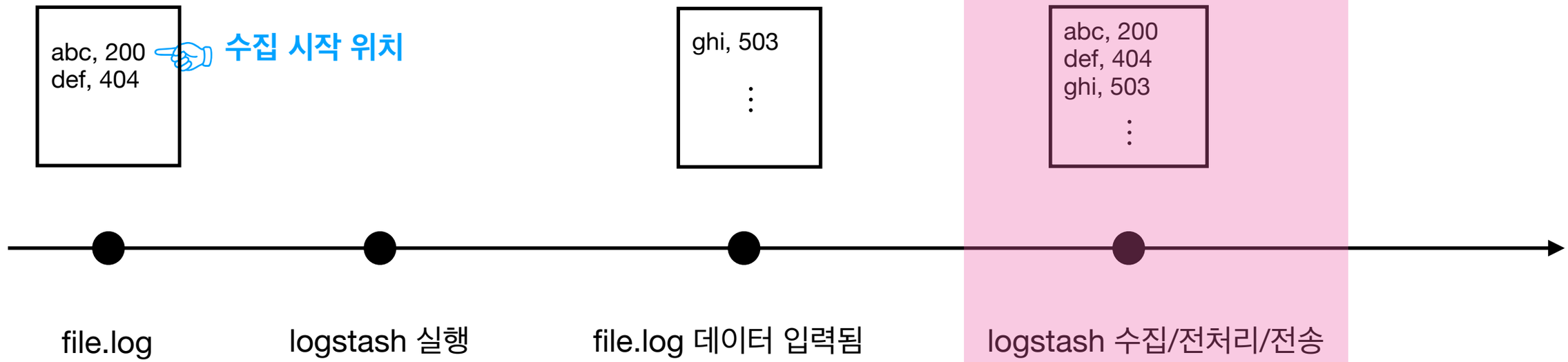


start_position : end

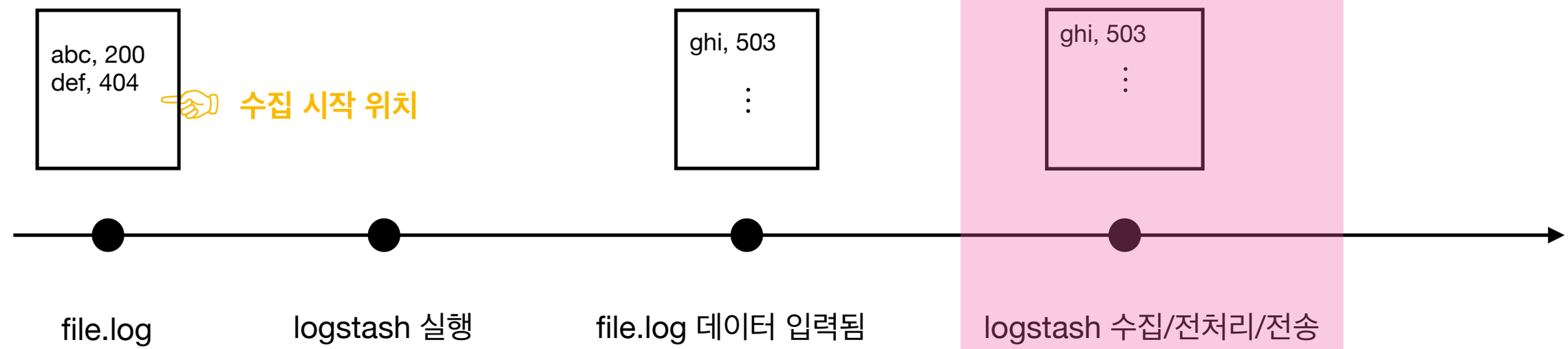
Input - file 심화

logstash 실행 전 path file에 데이터가 있는 경우 : 차이가 있다

start_position : beginning



start_position : end



Input - file 심화

단, start_position은 특정 파일을 처음 조회할 때에만 한 번 효과가 있다.

그 이후에는 sincedb에 기록이 되어 효과가 없다.

Input - file **심화**

Q. 왜 titanic.csv를 다운 받은 상황에서 다시 titanic2.csv를 다운 받아야 했나?

A. logstash는 기본적으로, 같은 데이터는 한 번만 조회하는 정책을 가지고 있기 때문이다

Input - file 심화

- test.log를 logstash로 수집/전처리/전송하는 상황에서 실수로 logstash를 중단했다고 하자.
- 사용자는 다시 logstash를 실행할텐데, 이 때 logstash는 test.log 어느 위치부터 다시 조회해야할까?
- 이런 정보를 담고 있는 게 **sincedb**다. 👑
- sincedb 기본
  - path
    - 기본 : <path.data>/plugins/inputs/file
    - 수업

```
[ec2-user@ip-172-31-21-251 file]$ pwd
/home/ec2-user/fc/logstash-5.6.4/data/plugins/inputs/file
[ec2-user@ip-172-31-21-251 file]$ ls -lah
total 28K
drwxrwxr-x 2 ec2-user ec2-user 4.0K Feb  4 11:16 .
drwxrwxr-x 3 ec2-user ec2-user 4.0K Dec 11 13:23 ..
-rw-rw-r-- 1 ec2-user ec2-user   20 Feb  3 18:22 .sincedb_8870f88565f4422dd26edfad0aee573a
-rw-rw-r-- 1 ec2-user ec2-user   20 Feb  3 18:29 .sincedb_980f17f4149d21a8b43538ac48210e08
-rw-rw-r-- 1 ec2-user ec2-user   18 Feb  4 10:54 .sincedb_a3994b8c55f173d7b3080ae74a8b5b91
-rw-rw-r-- 1 ec2-user ec2-user   18 Feb  4 10:06 .sincedb_c196e3a924cc2abcca1bbd65e527c986
-rw-rw-r-- 1 ec2-user ec2-user    1 Dec 11 13:24 .sincedb_fcea8bb6e7c7833f719f1b5b4e069e93
```

  - data
    - column : inode, major device number, minor device number, current byte offset
    - 예시

```
[ec2-user@ip-172-31-21-251 file]$ cat .sincedb_8870f88565f4422dd26edfad0aee573a
155232 0 51713 2310
```

- test.log를 logstash로 수집/전처리/전송하는 상황에서 실수로 logstash를 중단했다고 하자.
- 사용자는 다시 logstash를 실행할텐데, 이 때 logstash는 test.log 어느 위치부터 다시 조회해야할까?
- 이런 정보를 담고 있는 게 sincedb다.

한 마디로, logstash는 sincedb를 이용해서

1) 바라보는 데이터 별로 (logstash input path)

2) 어디까지 조회했는지

기록하고 있다고 알고 넘어가자

- sincedb 기본
  - path
    - 기본 : <path.data>/plugins/inputs/file
    - 수업

```
[ec2-user@ip-172-31-21-251 file]$ ls -lah
total 28K
drwxrwxr-x 3 ec2-user ec2-user 4.0K Feb  4 11:16 .
drwxrwxr-x 3 ec2-user ec2-user 4.0K Dec 11 13:23 ..
-rw-rw-r-- 1 ec2-user ec2-user   20 Feb  3 18:22 .sincedb_8870f88565f4422dd26edfad0aee573a
-rw-rw-r-- 1 ec2-user ec2-user   20 Feb  3 18:29 .sincedb_980f17f4149d21a8b43538ac48210e08
-rw-rw-r-- 1 ec2-user ec2-user   18 Feb  4 10:54 .sincedb_a3994b8c55f173d7b3080ae74a8b5b91
-rw-rw-r-- 1 ec2-user ec2-user   20 Feb  3 18:05 .sincedb_c196e3a924cc2abcca1bbd65e527c986
-rw-rw-r-- 1 ec2-user ec2-user    1 Dec 11 13:24 .sincedb_fcea8bb6e7c7833f719f1b5b4e069e93
```

  - data
    - column : inode, major device number, minor device number, current byte offset
    - 예시

```
[ec2-user@ip-172-31-21-251 file]$ cat .sincedb_8870f88565f4422dd26edfad0aee573a
155232 0 51713 2310
```

그러므로, 한 번 조회한 데이터를 logstash로 재실행해도 반응이 없는 이유는

logstash는 이미 해당 파일에 대해 끝까지 봤다는 기록이 남아있기 때문이다.

다시 조회하고 싶으면 sincedb_path option을 변경하면 된다.

logstash conf 작성 ♕

```
1   input {
2     file {
3       path => "/home/ec2-user/fc/logstash-5.6.4/titanic.csv"
4       start_position => "beginning"
5       sincedb_path => "/dev/null"
6     }
7   }
8
9   output {
10    stdout {
11      codec => rubydebug
12    }
13  }
```

이렇게 설정하면

a) 이미 조회한 데이터이지만 재수집된다.

b) 단, 이번에 실행해서 어디까지 조회했는지에 대한 데이터는 sincedb에 저장하지 않는다.

a) 이미 조회한 데이터도 다시 수집하면서

b) 그 조회 이력도 저장하고 싶으면, sincedb path를 임의로 입력하면 된다 ♕

```
sincedb_path => "/home/ec2-user/fc/logstash-5.6.4/titanic.db"
```

Input - file **심화**

logstash를 실행하면...

```
          "host" => "ip-172-31-21-251",
          "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic.csv",
    "@timestamp" => 2018-02-03T18:34:57.941Z,
       "message" => "45,Devaney,1,3,female,19,0,0,330958,7.8792,Q"
}
{
      "@version" => "1",
          "host" => "ip-172-31-21-251",
          "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic.csv",
    "@timestamp" => 2018-02-03T18:34:57.941Z,
       "message" => "46,Rogers,0,3,male,29.69911765,0,0,S.C./A.4. 23567,8.05,S"
}
{
      "@version" => "1",
          "host" => "ip-172-31-21-251",
          "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic.csv",
    "@timestamp" => 2018-02-03T18:34:57.941Z,
       "message" => "47,Lennon,0,3,male,29.69911765,1,0,370371,15.5,Q"
}
```

Input - jdbc

- database에서 데이터를 수집할 때 사용하는 plugin
- 우선 관련 driver를 설치하자 👑

```
1   $ cd /home/ec2-user/
2   $ wget https://downloads.mysql.com/archives/get/file/mysql-connector-java-5.1.36.tar.gz
3   $ tar -xzvf mysql-connector-java-5.1.36.tar.gz
```

Input - jdbc

- mongo uri : mysql://13.124.230.195:3306/fc
- database : fc
- table : fc
- user : fc
- password : fc

원본 데이터 (총 33 row)

```
mysql> use fc;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from fc;
+----+------+--------+------------+----------+-------------------+
| id | age  | salary | name       | location | employment_status |
+----+------+--------+------------+----------+-------------------+
|  1 |   33 |   4000 | Josh       | US       | employed          |
|  2 |   33 |  45000 | Tom        | US       | employed          |
|  3 |   23 |   3000 | Kirk       | US       | employed          |
|  4 |   27 |   3500 | Ken        | US       | employed          |
|  5 |   38 |   4500 | Jessie     | US       | employed          |
|  6 |   31 |   5200 | Jennifer   | US       | unemployed        |
|  7 |   33 |  22200 | Bob        | US       | unemployed        |
```

⋮

Input - jdbc

logstash conf 작성 👑

```
1   input {
2     jdbc {
3       jdbc_validate_connection => true
4       jdbc_connection_string => "jdbc:mysql://13.124.230.195:3306/fc"
5       jdbc_user => "fc"
6       jdbc_password => "fc"
7       jdbc_driver_library => "/home/ec2-user/mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar"
8       jdbc_driver_class => "com.mysql.jdbc.Driver"
9       statement => "SELECT * FROM fc"
10    }
11  }
12
13  output {
14    stdout {
15      codec => rubydebug
16    }
17  }
```

mysql uri

앞에서 설치한 jdbc driver library path

데이터 조회를 위한 sql 작성부

=> fc라는 table의 모든 data를 조회하라는 의미이다

logstash를 실행하면...

```
{
          "@timestamp" => 2018-02-03T19:03:01.883Z,
                "name" => "Kwon",
            "@version" => "1",
            "location" => "KR",
                  "id" => 29,
    "employment_status" => "unemployed",
              "salary" => 3500,
                 "age" => 48
}
{
          "@timestamp" => 2018-02-03T19:03:01.884Z,
                "name" => "Kang",
            "@version" => "1",
            "location" => "KR",
                  "id" => 30,
    "employment_status" => "employed",
              "salary" => 3300,
                 "age" => 28
}
```

⋮

Q. jdbc input도 logstash가 어디까지 조회했는지 기록을 하나?

A. sql로 조회한 마지막 row의 특정 column 데이터를 기록하는 방식으로 한다.

(sql_last_value)

다음 중 기준 column으로 적합한 column은?

```
{
          "@timestamp" => 2018-02-03T19:03:01.883Z,
                "name" => "Kwon",
            "@version" => "1",
            "location" => "KR",
                  "id" => 29,
   "employment_status" => "unemployed",
              "salary" => 3500,
                 "age" => 48
}
{
          "@timestamp" => 2018-02-03T19:03:01.884Z,
                "name" => "Kang",
            "@version" => "1",
            "location" => "KR",
                  "id" => 30,
   "employment_status" => "employed",
              "salary" => 3300,
                 "age" => 28
}
{
          "@timestamp" => 2018-02-03T19:03:01.884Z,
                "name" => "Yoon",
            "@version" => "1",
            "location" => "KR",
                  "id" => 31,
   "employment_status" => "unemployed",
              "salary" => 3500,
                 "age" => 23
}
```

마지막 row 였다면

기준 column	sql_last_value
id	29
salary	3500
age	48

마지막 row 였다면

기준 column	sql_last_value
id	30
salary	3300
age	28

실제 마지막 row

기준 column	sql_last_value
id	**31**
salary	3500
age	23

정해진 답은 없지만, 다음 2가지는 잘 살펴보자

1. 기준 column 선정은 다음과 같은 성격을 가진 column을 권장

- **(일정하게) incremental** 하는 numeric type
- created date 또는 updated date 같은 date type

2. sql 작성할 때 기준으로 정한 column으로 **order by** 할 것

- sql last value는 가장 마지막 row만을 기억한다
- 그러므로 꼭 order by를 해서 **마지막 row에 최대값**이 (혹은 최소값) 오도록 설정

Step 1) id < 5 인 데이터를 조회하고 id column을 sql_last_value로 저장하자 👑

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.124.230.195:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/home/ec2-user/mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    use_column_value => true
    tracking_column => id
    last_run_metadata_path => "/home/ec2-user/fc/logstash-5.6.4/id-under-5.db"
    statement => "
      SELECT *
      FROM fc
      WHERE id < 5
    "
  }
}

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

어디까지 조회했는지를 id라는 column으로 추적

어디까지 조회했는지에 대한 기록을 id-under-5.db라는 파일에 저장

Step2) logstash를 실행하고 last_run_metadata_path 파일을 열어보자

```
[ec2-user@ip-172-31-21-251 logstash-5.6.4]$ cat id-under-5.db
--- 4
```

- id < 5 인 데이터를 조회했으므로 sql_last_value는 4가 된다
- (last_run_metadat_path를 변경하지 않는다면) 이 값을 이용해서 연이은 작업을 할 수 있다

Step3) sql_last_value를 이용해서 데이터를 조회하자 👑

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.124.230.195:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/home/ec2-user/mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    use_column_value => true
    tracking_column => id
    last_run_metadata_path => "/home/ec2-user/fc/logstash-5.6.4/id-under-5.db"
    statement => "
      SELECT *
      FROM fc
      WHERE id > :sql_last_value
    "
  }
}

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

이 부분을 수정했다

Step4) logstash를 실행하면.. id > 4 데이터가 조회됐다

```
[2018-02-04T14:57:41,976][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"netflow", :directory=>"/home/ec2-user/fc/logstash-5.6.4/module
s/netflow/configuration"}
[2018-02-04T14:57:41,979][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"fb_apache", :directory=>"/home/ec2-user/fc/logstash-5.6.4/modu
les/fb_apache/configuration"}
[2018-02-04T14:57:42,203][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>"main", "pipeline.workers"=>2, "pipeline.batch.size"=>125, "pipeline.batc
h.delay"=>5, "pipeline.max_inflight"=>250}
[2018-02-04T14:57:42,304][INFO ][logstash.pipeline        ] Pipeline main started
[2018-02-04T14:57:42,393][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
[2018-02-04T14:57:42,893][INFO ][logstash.inputs.jdbc     ] (0.005000s) SELECT * FROM fc WHERE id > 4
{
          "@timestamp" => 2018-02-04T14:57:42.907Z,
               "name" => "Jessie",
          "@version" => "1",
           "location" => "US",
                 "id" => 5,
  "employment_status" => "emp
             "salary" => 4500,
                "age" => 38
}
{
          "@timestamp" => 2018-02-04T14:57:42.910Z,
               "name" => "Jennifer",
          "@version" => "1",
           "location" => "US",
                 "id" => 6,
  "employment_status" => "unemployed",
             "salary" => 5200,
                "age" => 31
```

내부적으로 코드가 수정됐다

id = 5 부터 조회

•
•
•

Q. 데이터베이스에서 정기적으로 데이터를 조회할 수 없나?

A. scheduling을 이용해서 가능하다

Input - jdbc 심화

하루에 한 번 오전 6시에 데이터베이스에 데이터가 입력된다고 하자 ♕

```
input {
  jdbc  {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.124.230.195:3306/fc"
    jdbc_driver_library => "/home/ec2-user/driver/mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    jdbc_user => "fc"
    jdbc_password => "fc"
    statement => "SELECT * FROM fc"
    schedule => "0 6 * * *"
  }
}

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

☜ 스케줄 시간 설정 ♕

```
* * * * * *
| | | | | |
| | | | | +-- Year                (range: 1900-3000)
| | | | +---- Day of the Week     (range: 1-7, 1 standing for Monday)
| | | +------ Month of the Year   (range: 1-12)
| | +-------- Day of the Month    (range: 1-31)
| +---------- Hour                (range: 0-23)
+------------ Minute              (range: 0-59)
```

단, logstash는 가볍지 않아 서버 성능에 부담을 줄 수 있다.

그러므로 linux cron tab을 사용해서 특정한 시점에 한 번

logstash를 실행하고 종료하는 방법도 고려할만하다.

Input - elasticsearch

elastic stack을 활용해서 dashboard를 만드는데,

역으로 elasticsearch에서 데이터를 조회하는 게 필요한가?

Input - elasticsearch

1. elasticsearch의 데이터를 csv로 export 하고 싶은 경우
2. ~~elasticsearch의 데이터를 aggregation한 결과를 저장하고 싶은 경우~~ 👑

우선은 elasticsearch 데이터를 조회하는 법부터 보자

Input - elasticsearch

- host: http://13.124.230.195:9200
- index : week5

원본 데이터


```
"hits": {
  "total": 33,
  "max_score": 1,
  "hits": [
    {
      "_index": "week5",
      "_type": "week5",
      "_id": "AWFhmIfJPloSIAlpN80d",
      "_score": 1,
      "_source": {
        "@timestamp": "2018-02-04T16:14:01.456Z",
        "name": "Kirk",
        "@version": "1",
        "location": "US",
        "id": 3,
        "employment_status": "employed",
        "salary": 3000,
        "age": 23
      }
    },
```


⋮

Input - elasticsearch

logstash conf 작성 👑

```
1   input {
2     elasticsearch {
3       hosts => ["13.124.230.195:9200"]
4       index => "week5"
5       query => '{
6         "query": {
7           "match_all": {}
8         }
9       }'
10    }
11  }
12
13  output {
14    stdout {
15      codec => rubydebug
16    }
17  }
```

☝ Query DSL을 알면 좋은 이유가 하나 더 늘었다

Input - elasticsearch

logstash를 실행하면...

```
{
          "@timestamp" => 2018-02-03T19:03:01.883Z,
                "name" => "Kwon",
            "@version" => "1",
            "location" => "KR",
                  "id" => 29,
    "employment_status" => "unemployed",
              "salary" => 3500,
                 "age" => 48
}
{
          "@timestamp" => 2018-02-03T19:03:01.884Z,
                "name" => "Kang",
            "@version" => "1",
            "location" => "KR",
                  "id" => 30,
    "employment_status" => "employed",
              "salary" => 3300,
                 "age" => 28
}
```

⋮

Output - csv

- Input (Input + Filter)를 거친 데이터를 csv 형태로 저장하는 plugin
- elasticsearch 데이터를 csv로 export 하는 법을 배우자

Output - csv

원본 데이터

- host: http://13.124.230.195:9200
- index : week5


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 33,
    "max_score": 1,
    "hits": [
      {
        "_index": "week5",
        "_type": "week5",
        "_id": "AWFhmIfJPloSIAlpN80d",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-02-04T16:14:01.456Z",
          "name": "Kirk",
          "@version": "1",
          "location": "US",
          "id": 3,
          "employment_status": "employed",
          "salary": 3000,
          "age": 23
        }
      },
```


:
:
:

Output - csv

logstash conf 작성 👑

```
input {
  elasticsearch {
    hosts => ["13.124.230.195:9200"]
    index => "week5"
    query => '{
      "query": {
        "match_all": {}
      }
    }'
  }
}

output {
  csv {
    fields => ["name", "location", "employment_status", "salary", "age"]
    path => "/home/ec2-user/fc/logstash-5.6.4/week5.csv"
  }
}
```

csv 출력을 위해 csv output plugin 사용

csv를 저장할 경로 설정

- event에서 csv에 저장할 Field 설정
- csv 각 row의 column 또한 저 순서로 저장

Output - csv

logstash를 실행하고 week5.csv를 열어보자

```
[ec2-user@ip-172-31-21-251 logstash-5.6.4]$ cat week5.csv | head -5
Jennifer,US,unemployed,5200,31
Tim,UK,employed,47000,38
Tanaka,JP,employed,3500,38
Saeki,JP,employed,4500,44
Kang,KR,employed,3300,28
```

⋮

Output - elasticsearch

- 수집하고 전처리한 결과를 **elasticsearch**에 전송하는 **plugin**
- **dashboard**를 만드는 데 있어 가장 중요한 **output plugin**

Output - elasticsearch

logstash conf 작성 👑

```
input {
  jdbc  {
    jdbc_connection_string => "jdbc:mysql://13.124.230.195:3306/fc"
    jdbc_validate_connection => true
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/home/ec2-user/mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
  }
}

output {
  elasticsearch {
    hosts => ["13.124.230.195:9200"]
    index => "week5_higee"
    document_type => "week5_higee"
  }
}
```

☞ es에 데이터를 전송하기 위해 es output plugin 사용

Output - elasticsearch

실행해보면, 작업 후에 콘솔에 아무 표시가 없다

```
Sending Logstash's logs to /home/ec2-user/fc/logstash-5.6.4/logs which is now configured via log4j2.properties
[2018-02-04T04:54:06,888][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"netflow", :directory=>"/home/ec2-user/fc/logstash-5.6.4/modules/netflow/configuration"}
[2018-02-04T04:54:06,891][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"fb_apache", :directory=>"/home/ec2-user/fc/logstash-5.6.4/modules/fb_apache/configuration"}
[2018-02-04T04:54:07,264][INFO ][logstash.outputs.elasticsearch] Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://13.124.230.195:9200/]}}
[2018-02-04T04:54:07,266][INFO ][logstash.outputs.elasticsearch] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://13.124.230.195:9200/, :path=>"/"}
[2018-02-04T04:54:07,342][WARN ][logstash.outputs.elasticsearch] Restored connection to ES instance {:url=>"http://13.124.230.195:9200/"}
[2018-02-04T04:54:07,376][INFO ][logstash.outputs.elasticsearch] Using mapping template from {:path=>nil}
[2018-02-04T04:54:07,380][INFO ][logstash.outputs.elasticsearch] Attempting to install template {:manage_template=>{"template"=>"logstash-*", "version"=>50001, "settings"=>{"index.refresh_interval"=>"5s"}, "mappings"=>{"_default_"=>{"_all"=>{"enabled"=>true, "norms"=>false}, "dynamic_templates"=>[{"message_field"=>{"path_match"=>"message", "match_mapping_type"=>"string", "mapping"=>{"type"=>"text", "norms"=>false}}}, {"string_fields"=>{"match"=>"*", "match_mapping_type"=>"string", "mapping"=>{"type"=>"text", "norms"=>false, "fields"=>{"keyword"=>{"type"=>"keyword", "ignore_above"=>256}}}}}], "properties"=>{"@timestamp"=>{"type"=>"date", "include_in_all"=>false}, "@version"=>{"type"=>"keyword", "include_in_all"=>false}, "geoip"=>{"dynamic"=>true, "properties"=>{"ip"=>{"type"=>"ip"}, "location"=>{"type"=>"geo_point"}, "latitude"=>{"type"=>"half_float"}, "longitude"=>{"type"=>"half_float"}}}}}}}
[2018-02-04T04:54:07,388][INFO ][logstash.outputs.elasticsearch] New Elasticsearch output {:class=>"LogStash::Outputs::ElasticSearch", :hosts=>["http://13.124.230.195:9200"]}
[2018-02-04T04:54:07,392][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>"main", "pipeline.workers"=>2, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>250}
[2018-02-04T04:54:07,482][INFO ][logstash.pipeline        ] Pipeline main started
[2018-02-04T04:54:07,549][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
[2018-02-04T04:54:08,113][INFO ][logstash.inputs.jdbc     ] (0.009000s) SELECT * FROM fc
[2018-02-04T04:54:10,511][WARN ][logstash.agent           ] stopping pipeline {:id=>"main"}
[ec2-user@ip-172-31-21-251 logstash-5.6.4]$
```

☝ **logstash 작업이 끝나고 다음 작업을 기다리고 있는 걸 볼 수 있다**

제대로 된 건지 확인하기 위해 kibana에서 확인하자

Output - elasticsearch

```
GET week5_higee/_search
{
  "query": {
    "match_all": {}
  }
}
```

```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 33,
    "max_score": 1,
    "hits": [
      {
        "_index": "week5_higee",
        "_type": "week5_higee",
        "_id": "AWFhu_EmPloSIAlpN814",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-02-04T16:52:42.149Z",
          "name": "Kirk",
          "@version": "1",
          "location": "US",
          "id": 3,
          "employment_status": "employed",
          "salary": 3000,
          "age": 23
        }
      },
      {
```

제대로 들어갔다!

es로 전송할 때 유용하게 사용할 수 있는 tip을 알고 넘어가자

- es index를 생성할 때 날짜 정보를 넣을 수 있나?

- es document id를 event의 특정 field data를 이용해서 설정할 수 있나?

Output - elasticsearch

1. es index를 생성할 때 날짜 정보를 넣을 수 있나?

```
GET week5_higee/_search
{
  "query": {
    "match_all": {}
  }
}
```


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 33,
    "max_score": 1,
    "hits": [
      {
        "_index": "week5_higee",
        "_type": "week5_higee",
        "_id": "AWFhu_EmPloSIAlpN814",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-02-04T16:52:42.149Z",
          "name": "Kirk",
          "@version": "1",
          "location": "US",
          "id": 3,
          "employment_status": "employed",
          "salary": 3000,
          "age": 23
        }
      },
```


원본 데이터에는 없던 field이다. @timestamp는 logstash가 특정 event를 처리한 시간을 기록하는데, 이 field를 이용해서 index 이름을 생성할 때 날짜 정보를 넣을 수 있다.

Output - elasticsearch

logstash conf 작성 👑

```
input {
  jdbc  {
    jdbc_connection_string => "jdbc:mysql://13.124.230.195:3306/fc"
    jdbc_validate_connection => true
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/home/ec2-user/mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
  }
}

output {
  elasticsearch {
    hosts => ["13.124.230.195:9200"]
    index => "week5_higee-%{+YYYY.MM.dd}"
    document_type => "week5_higee"
  }
}
```

@timestamp에서 연도, 월, 일을 추출해서

index 이름을 생성할 때 추가해준다.

Output - elasticsearch

logstash를 실행하고 Kibana에서 확인하자

```
GET week5_higee-2018.02.04/_search
{
  "query": {
    "match_all": {}
  }
}
```

다음과 같은 조합으로 index 이름 생성

- static : week5_higee
- dynamic : 2018.02.04


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 33,
    "max_score": 1,
    "hits": [
      {
        "_index": "week5_higee-2018.02.04",
        "_type": "week5_higee",
        "_id": "AWFh2rXqPloSIAlpN82L",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-02-04T17:26:18.594Z",
          "name": "Ken",
          "@version": "1",
          "location": "US",
          "id": 4,
          "employment_status": "employed",
          "salary": 3500,
          "age": 27
        }
      },
```

2. es document id를 event의 특정 field data를 이용해서 설정할 수 있나?

Output - elasticsearch

logstash conf 작성 👑

```
input {
  jdbc  {
    jdbc_connection_string => "jdbc:mysql://13.124.230.195:3306/fc"
    jdbc_validate_connection => true
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/home/ec2-user/mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
  }
}

output {
  elasticsearch {
    hosts => ["13.124.230.195:9200"]
    index => "week5_higee-%{+YYYY.MM.dd}"
    document_type => "week5_higee"
    document_id => "%{@timestamp}-%{name}"
  }
}
```

%{@timestamp}와 %{name} 부분에

각event에서 수집한 실제 value가 입력된다

Output - elasticsearch

logstash를 실행하고 Kibana에서 확인하자

```
GET week5_higee-2018.02.04/_search
{
  "query": {
    "match_all": {}
  }
}
```


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 66,
    "max_score": null,
    "hits": [
      {
        "_index": "week5_higee-2018.02.04",
        "_type": "week5_higee",
        "_id": "2018-02-04T17:35:32.207Z-Yang",
        "_score": null,
        "_source": {
          "@timestamp": "2018-02-04T17:35:32.207Z",
          "name": "Yang",
          "@version": "1",
          "location": "KR",
          "id": 32,
          "employment_status": "unemployed",
          "salary": 3600,
          "age": 23
        },
        "sort": [
          1517765732207
        ]
      },
```

Output - multiple output plugins

- 수집하고 전처리한 결과를 여러 **output**에 전송하는 방법
- 개발 단계에서 **stdout+main output**을 함께 쓰면 빠르게 확인할 수 있다

Output - multi

logstash conf 작성 👑

```
input {
  jdbc  {
    jdbc_connection_string => "jdbc:mysql://13.124.230.195:3306/fc"
    jdbc_validate_connection => true
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/home/ec2-user/mysql-connector-java-5.1.36/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
  }
}

output {
  elasticsearch {
    hosts => ["13.124.230.195:9200"]
    index => "week5_higee_multi"
    document_type => "week5_higee_multi"
  }
  stdout {
    codec => rubydebug
  }
}
```

- output plugin에 2개(elasticsearch, stdout) 사용
- input/output/filter 모두 복수개 사용 가능

Output - multi

logstash를 실행하면...

```
}
{
          "@timestamp" => 2018-02-04T17:52:15.799Z,
                "name" => "Kang",
            "@version" => "1",
            "location" => "KR",
                  "id" => 30,
    "employment_status" => "employed",
              "salary" => 3300,
                 "age" => 28
}
{
          "@timestamp" => 2018-02-04T17:52:15.800Z,
                "name" => "Yoon",
            "@version" => "1",
            "location" => "KR",
                  "id" => 31,
    "employment_status" => "unemployed",
              "salary" => 3500,
                 "age" => 23
}
```

⋮

Output - multi

elasticsearch에도 들어간 걸 확인할 수 있다

```
GET week5_higee_multi/_search
{
  "query": {
    "match_all": {}
  }
}
```


```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 33,
    "max_score": 1,
    "hits": [
      {
        "_index": "week5_higee_multi",
        "_type": "week5_higee_multi",
        "_id": "AWFh8nh5PloSIAlpN82z",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-02-04T17:52:15.768Z",
          "name": "Josh",
          "@version": "1",
          "location": "US",
          "id": 1,
          "employment_status": "employed",
          "salary": 4000,
          "age": 33
        }
      },
```


:

Filter - csv

- 특정한 separator로 연결된 데이터 (예: csv) 파일을 parsing할 때 유용
- 우선 데이터를 다운 받자 👑

```
$ cd /home/ec2-user/fc/logstash-5.6.4
$ wget https://raw.githubusercontent.com/higee/elastic/class2/Week4_Logstash/data/titanic-header.csv
```

Filter - csv

이런 데이터가 있다고 하자

```
1 Index, Name, Survival, Pclass, Sex, Age, SibSp, Parch, Ticket, Fare, Embarked
2 1,Braund,0,3,male,22,1,0,A/5 21171,7.25,S
3 2,Cumings,1,1,female,38,1,0,PC 17599,71.2833,C
4 3,Heikkinen,1,3,female,26,0,0,STON/O2. 3101282,7.925,S
```

Filter - csv

아래와 같이 key, value 형태로 elasticsearch에 색인하려면 어떻게 해야 할까?

```
{
  "Index" : 1,
  "Name" : "Braud",
  "Survival" : 0,
  "Pclass" : 3,
  "Sex" : "male",
  "Age" : 22,
  "SibSp" : 1,
  "Parch" : 0,
  "Ticket" : "A/5 21171",
  "Fare" : 7.25,
  "Embarked" : "S"
}
```

Filter - csv

logstash conf 작성 👑

```
input {
  file  {
    path => "/home/ec2-user/fc/logstash-5.6.4/titanic-header.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  csv {
    separator => ","
  }
}

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

☜ 계속 실험할 것이기에 설정

☜ ,로 구분된 데이터이기에 ‘,' 입력

☜

- filter plugin의 전반적인 구조는 input/output과 유사
- plugin을 선택한 후 option을 입력하는 형태

logstash를 실행하면...

Header에 해당하는 데이터

잘 구분되었으나 column 이름으로는

어떤 데이터를 나타내는지 파악이 어렵다

```
{
      "column1" => "Index",
     "column11" => " Embarked",
     "column10" => " Fare",
      "column5" => " Sex",
      "column4" => " Pclass",
      "column3" => " Survival",
      "column2" => " Name",
      "message" => "Index, Name, Survival, Pclass, Sex, Age, SibSp, Parch, Ticket, Fare, Embarked",
         "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic-header.csv",
   "@timestamp" => 2018-02-05T14:51:04.716Z,
     "@version" => "1",
         "host" => "ip-172-31-21-251",
      "column9" => " Ticket",
      "column8" => " Parch",
      "column7" => " SibSp",
      "column6" => " Age"
}
{
      "column1" => "1",
     "column11" => "S",
     "column10" => "7.25",
      "column5" => "male",
      "column4" => "3",
      "column3" => "0",
      "column2" => "Braund",
      "message" => "1,Braund,0,3,male,22,1,0,A/5 21171,7.25,S",
         "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic-header.csv",
   "@timestamp" => 2018-02-05T14:51:04.718Z,
     "@version" => "1",
         "host" => "ip-172-31-21-251",
      "column9" => "A/5 21171",
      "column8" => "0",
      "column7" => "1",
      "column6" => "22"
}
```

logstash conf 작성 👑

```
input {
  file  {
    path => "/home/ec2-user/fc/logstash-5.6.4/titanic-header.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  csv {
    separator => ","
    autodetect_column_names => true
  }
}

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

이 옵션을 이용해서 첫번째 row의 데이터를 column으로 사용하도록 설정

Filter - csv

logstash를 실행하면...

column name이 올바르게 제공되었다

```
{
          " Sex" => "male",
        "Index" => "1",
         " Fare" => "7.25",
      "message" => "1,Braund,0,3,male,22,1,0,A/5 21171,7.25,S",
         " Name" => "Braund",
          " Age" => "22",
         "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic-header.csv",
     " Embarked" => "S",
    "@timestamp" => 2018-02-05T15:06:31.628Z,
        " SibSp" => "1",
        " Parch" => "0",
     "@version" => "1",
         "host" => "ip-172-31-21-251",
       " Pclass" => "3",
       " Ticket" => "A/5 21171",
     " Survival" => "0"
}
```

Filter - csv

그런데 모두 string 처리가 되었다!

```
{
          " Sex" => "male",
        "Index" => "1",
         " Fare" => "7.25",
      "message" => "1,Braund,0,3,male,22,1,0,A/5 21171,7.25,S",
         " Name" => "Braund",
          " Age" => "22",
         "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic-header.csv",
     " Embarked" => "S",
   "@timestamp" => 2018-02-05T15:06:31.628Z,
        " SibSp" => "1",
        " Parch" => "0",
     "@version" => "1",
         "host" => "ip-172-31-21-251",
       " Pclass" => "3",
       " Ticket" => "A/5 21171",
     " Survival" => "0"
}
```

Filter - csv

logstash conf 작성 ♕

```
1   input {
2     file  {
3       path => "/home/ec2-user/fc/logstash-5.6.4/titanic-header.csv"
4       start_position => "beginning"
5       sincedb_path => "/dev/null"
6     }
7   }
8
9   filter {
10    csv {
11      separator => ","
12      autodetect_column_names => true
13      convert  => {
14        "Pclass" => "integer"
15        "Index" => "integer"
16        "Survival" => "integer"
17        "Fare" => "float"
18      }
19    }
20  }
21
22  output {
23    stdout {
24      codec => rubydebug
25    }
26  }
```

“Pclass” Field의 Type을 integer로

“Index” Field의 Type을 integer로

“Survival” Field의 Type을 integer로

logstash를 실행하면...

```
{
      "Embarked" => "C",
        "Pclass" => 1,
        "Ticket" => "PC 17599",
           "Sex" => "female",
         "Index" => 2,
         "SibSp" => "1",
       "message" => "2,Cumings,1,1,female,38,1,0,PC 17599,71.2833,C",
      "Survival" => 1,
          "Name" => "Cumings",
          "Fare" => 71.2833,
          "path" => "/home/ec2-user/fc/logstash-5.6.4/titanic-header.csv",
    "@timestamp" => 2018-02-05T15:34:17.190Z,
         "Parch" => "0",
      "@version" => "1",
          "host" => "ip-172-31-21-251",
           "Age" => "38"
}
```

Filter - csv

지금까지 만든 filter를 적용해서 elasticsearch로 전송해보자 👑

```
input {
  file  {
    path => "/home/ec2-user/fc/logstash-5.6.4/titanic-header.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  csv {
    separator => ","
    autodetect_column_names => true
    convert  => {
      "Pclass" => "integer"
      "Index" => "integer"
      "Survival" => "integer"
      "Fare" => "float"
    }
  }
}

output {
  elasticsearch {
    hosts => ["13.124.230.195:9200"]
    index => "week5_higee_csv_filter"
    document_type => "week5_higee_csv_filter"
  }
}
```

Filter - mutate

- Field data의 값을 변형할 때 사용하는 강력한 plugin
- 여러가지 option을 혼합해서 같이 사용한다
- 우선 데이터를 다운 받자 👑

```
1   $ cd /home/ec2-user/fc/logstash-5.6.4
2   $ wget https://raw.githubusercontent.com/higee/elastic/class2/Week4_Logstash/data/ip-address.log
```

Filter - mutate (split)

이런 데이터 (ip-address.log)가 있다고 하자

```
1    13.124.230.195:5601
2    13.124.230.195:3306
3    13.124.230.195:9200
4    13.124.230.195:9300
```

Filter - mutate (split)

아래와 같이 elasicsearch에 저장하고 싶으면?

```
{
  "ip" : 13.124.230.195,
  "port" : 5601
},
{
  "ip" : 13.124.230.195,
  "port" : 3306
},
{
  "ip" : 13.124.230.195,
  "port" : 9200
},
{
  "ip" : 13.124.230.195,
  "port" : 9300
}
```

logstash conf 작성 👑

```
1   input {
2     file  {
3       path => "/home/ec2-user/fc/logstash-5.6.4/ip-address.log"
4       start_position => "beginning"
5       sincedb_path => "/dev/null"
6     }
7   }
8
9   filter {
10    mutate {
11      split => { "message" => ":" }
12    }
13  }
14
15  output {
16    stdout {
17      codec => rubydebug
18    }
19  }
```

데이터를 :를 기준으로 split 해보자

Filter - mutate

logstash를 실행하면...

```
{
      "@version" => "1",
          "host" => "ip-172-31-21-251",
          "path" => "/home/ec2-user/fc/logstash-5.6.4/ip-address.log",
    "@timestamp" => 2018-02-05T17:15:52.962Z,
       "message" => [
       [0] "13.124.230.195"
       [1] "5601"
    ]
}
```

원래 하나였던 것이 설정한 separator를 기준으로 두 개로 split되었다.

Filter - mutate

split된 데이터로 각각 ip, port라는 field를 생성해보자 (바로 뒤에서)

```
{
      "@version" => "1",
          "host" => "ip-172-31-21-251",
          "path" => "/home/ec2-user/fc/logstash-5.6.4/ip-address.log",
    "@timestamp" => 2018-02-05T17:15:52.962Z,
       "message" => [
       [0] "13.124.230.195"
       [1] "5601"
    ]
}
```

활용

ip : “13.124.230.195”

port : “5601”

logstash conf 작성 👑

```
1   input {
2     file  {
3       path => "/home/ec2-user/fc/logstash-5.6.4/ip-address.log"
4       start_position => "beginning"
5       sincedb_path => "/dev/null"
6     }
7   }
8
9   filter {
10    mutate {
11      split => { "message" => ":" }
12      add_field => {
13        "ip" => "%{message[0]}"
14        "port" => "%{message[1]}"
15      }
16    }
17  }
18
19  output {
20    stdout {
21      codec => rubydebug
22    }
23  }
```

☜ split된 message의 첫 번째 데이터

☜ split된 message의 두 번째 데이터

logstash를 실행하면...

```
{
          "path" => "/home/ec2-user/fc/logstash-5.6.4/ip-address.log",
    "@timestamp" => 2018-02-05T17:27:47.386Z,
          "port" => "5601",
            "ip" => "13.124.230.195",
      "@version" => "1",
          "host" => "ip-172-31-21-251",
       "message" => [
        [0] "13.124.230.195",
        [1] "5601"
    ]
}
```

- 필요한 데이터는 모두 생성했다
- elasticsearch로 전송하기 전에, 불필요한 field는 제거하자 (port, ip 외의 모든 field)

elasticsearch에 불필요한 데이터를 저장할 필요가 없으므로

이번에는 불필요한 Field를 제거하자

logstash conf 작성 👑

```
1   input {
2     file  {
3       path => "/home/ec2-user/fc/logstash-5.6.4/ip-address.log"
4       start_position => "beginning"
5       sincedb_path => "/dev/null"
6     }
7   }
8
9   filter {
10    mutate {
11      split => { "message" => ":" }
12      add_field => {
13        "ip" => "%{message[0]}"
14        "port" => "%{message[1]}"
15      }
16      remove_field => ["path", "@timestamp", "@version", "host", "message"]
17    }
18  }
19
20  output {
21    stdout {
22      codec => rubydebug
23    }
24  }
```

☝ 제거할 field 이름을 직접 나열

logstash를 실행하면...

```
{
    "port" => "5601",
      "ip" => "13.124.230.195"
}
{
    "port" => "3306",
      "ip" => "13.124.230.195"
}
{
    "port" => "9200",
      "ip" => "13.124.230.195"
}
{
    "port" => "9300",
      "ip" => "13.124.230.195"
}
```

Filter - mutate

앞에서 활용한 filter를 이용해 elasticsearch에 데이터를 전송하자 👑

```
input {
  file  {
    path => "/home/ec2-user/fc/logstash-5.6.4/ip-address.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  mutate {
    split => { "message" => ":" }
    add_field => {
      "ip" => "%{message[0]}"
      "port" => "%{message[1]}"
    }
    remove_field => ["path", "@timestamp", "@version", "host", "message"]
  }
}

output {
  elasticsearch {
    hosts => ["13.124.230.195:9200"]
    index => "week5_higee_ip_address"
    document_type => "week5_higee_ip_address"
  }
}
```

Filter - grok

- 구조화되어 있지 않은 데이터를 처리하는데 강력한 plugin
- 실제 production에서 생기는 log를 분석할 때 grok + 다른 plugin 조합 사용
- 우선 데이터를 다운 받자 👑

```
$ cd /home/ec2-user/fc/logstash-5.6.4
$ wget https://raw.githubusercontent.com/higee/elastic/class2/Week4_Logstash/data/access.log
```

아래와 같은 로그가 생긴다고 하자

request_time client_ip status_code

```
1  2018-02-05T05:49:53.859060Z 172.30.39.133 0.079 200 "GET https://helloworld.com/users/1 HTTP/1.1"
2  2018-02-05T06:49:53.859060Z 172.29.43.253 0.1 404 "GET https://helloworld.com/users/3 HTTP/1.1"
3  2018-02-05T07:49:53.859060Z 172.30.40.210 0.052 503 "GET https://helloworld.com/users/2 HTTP/1.1"
4  2018-02-05T08:49:53.859060Z 172.31.40.131 0.038 200 "POST https://helloworld.com/users/5 HTTP/1.1"
```

processing_time request

이런 로그를 분석하려면 다음의 과정을 거친다

- 가지고 있는 로그 파일 패턴을 발견한다 (~~최소한 노력한다~~)
- **grok-patterns** 👑에서 가서 match 할 수 있는 pattern을 찾는다
- **grok debugger** 👑를 이용해서 시도해본다
- grok debugger에서 발견한 pattern을 활용해서 logstash grok filter을 작성한다
- 필요한 경우 다른 filter plugin을 함께 사용한다

1. 가지고 있는 로그 파일 패턴을 발견한다 (최소한 노력한다)

request_time client_ip status_code

```
1 2018-02-05T05:49:53.859060Z 172.30.39.133 0.079 200 "GET https://helloworld.com/users/1 HTTP/1.1"
2 2018-02-05T06:49:53.859060Z 172.29.43.253 0.1 404 "GET https://helloworld.com/users/3 HTTP/1.1"
3 2018-02-05T07:49:53.859060Z 172.30.40.210 0.052 503 "GET https://helloworld.com/users/2 HTTP/1.1"
4 2018-02-05T08:49:53.859060Z 172.31.40.131 0.038 200 "POST https://helloworld.com/users/5 HTTP/1.1"
```

processing_time request

- request_time : 날짜/시간을 나타내는 특정한 형식
- client_ip : ip 주소
- processing_time : 0보다 큰 소수점
- status_code : 정수
- request : HTTP 메서드 + protocol://host:port/uri + HTTP 버전 형식

2. grok-patterns에서 가서 match 할 수 있는 pattern을 찾는다 👑

(각 pattern이 무엇을 의미하는지 사전 학습 필요)

request_time client_ip status_code

```
1  2018-02-05T05:49:53.859060Z 172.30.39.133 0.079 200 "GET https://helloworld.com/users/1 HTTP/1.1"
2  2018-02-05T06:49:53.859060Z 172.29.43.253 0.1 404 "GET https://helloworld.com/users/3 HTTP/1.1"
3  2018-02-05T07:49:53.859060Z 172.30.40.210 0.052 503 "GET https://helloworld.com/users/2 HTTP/1.1"
4  2018-02-05T08:49:53.859060Z 172.31.40.131 0.038 200 "POST https://helloworld.com/users/5 HTTP/1.1"
```

processing_time request

- request_time : TIMESTAMP_ISO8601
- client_ip : IPORHOST
- processing_time : NUMBER
- status_code : NUMBER
- request : QS

3. grok debugger를 이용해 시도해본다 👑

Grok Debugger | Debugger | Discover | Patterns

2018-02-05T05:49:53.859060Z 172.31.40.153 0.079 200 "GET https://helloworld.com/users/1 HTTP/1.1"

%{TIMESTAMP_ISO8601:time} %{IPORHOST:clientip} %{NUMBER:response_time} %{NUMBER:status_code} %{QS:request}

Add custom patterns | Keep Empty Captures | Named Captures Only | Singles | Autocomplete | Go


```
{
  "time": [
    [
      "2018-02-05T05:49:53.859060Z"
    ]
  ],
  "YEAR": [
    [
      "2018"
    ]
  ],
  "MONTHNUM": [
    [
      "02"
    ]
  ],
  "MONTHDAY": [
    [
      "05"
    ]
  ],
```


- 문법 : %{매칭하려는 패턴:임의의 Field 이름}
- 예시 : %{NUMBER:response_time}
- 의미 : NUMBER에 해당하는 값이 오면 response_time이라는 Field에 저장

☞ log 입력

☞ 패턴 입력

☞ 결과 표시

4. logstash conf 작성한다 👑

```
input {
  file  {
    path => "/home/ec2-user/fc/logstash-5.6.4/access.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  grok {
    match => {
     "message" => '%{TIMESTAMP_ISO8601:time} %{IPORHOST:clientip} %{NUMBER:response_time} %{NUMBER:status_code}'
    }
  }
}

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

grok debugger에서 작성했던 코드를 입력한다

logstash를 실행하면...

```
{
          "path" => "/home/ec2-user/fc/logstash-5.6.4/access.log",
       "request" => "\"GET https://helloworld.com/users/1 HTTP/1.1\"",
    "@timestamp" => 2018-02-05T18:46:22.457Z,
   "status_code" => "200",
      "clientip" => "172.31.40.153",
      "@version" => "1",
          "host" => "ip-172-31-21-251",
 "response_time" => "0.079",
          "time" => "2018-02-05T05:49:53.859060Z",
       "message" => "2018-02-05T05:49:53.859060Z 172.31.40.153 0.079 200 \"GET https://helloworld.com/users/1 HTTP/1.1\""
}
```

5. 다른 필터를 활용하면 좀 더 의미 있는 단위의 데이터 수집이 가능하다 👑

```
{
       "status_code" => "200",
              "verb" => "\"GET",
               "uri" => "https://helloworld.com/users/1",
        "@timestamp" => 2018-02-05T19:05:39.403Z,
          "clientip" => "172.31.40.153",
              "http" => "HTTP/1.1\"",
     "response_time" => "0.079",
              "time" => "2018-02-05T05:49:53.859060Z"
}
```