Elastic Stack을 활용한 Data Dashboard 만들기

Week 5 - Logstash로 데이터를 전처리하고 전송하자

Fast Campus

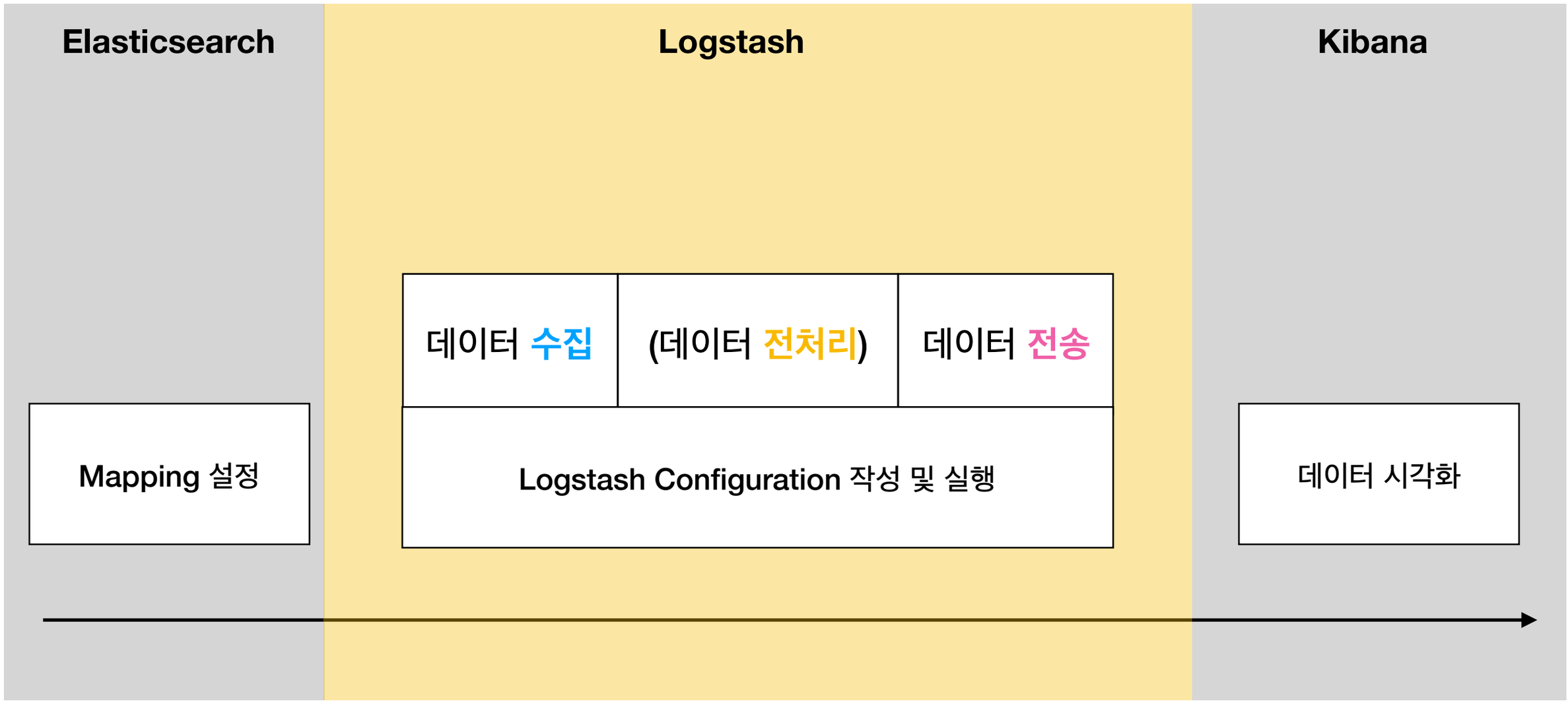
지금까지 시각화도 했고, 검색도 해봤는데 뭔가 2% 부족하다

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)

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**	데이터에 mutate 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 `$ docker exec -u 0 -it logstash bash`

2. conf 생성 Logstash conf 파일 생성 `$ vi 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 `$ docker exec -u 0 -it logstash bash`

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

(파일 이름 예시 : exercise.conf)

편집모드 Logstash conf 파일 편집모드 `i`

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

저장 Logstash conf 파일 저장 `ESC 입력 후 :wq`

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

conf 작성이 logstash 핵심

전반적인 흐름을 기억하자.

큰 흐름은 일정하고 conf 작성하는 부분만 바꾼다.

(오늘 실습은 이 과정의 반복이다.)

```
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]}"
    }
  }
}
```

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`

그 전에...

1. AWS EC2에 접속하자

`Windows(♕)는 putty를 통해서, Mac(♕)은 terminal에서 직접`

2. 지난 시간에 clone 받은 repo 데이터를 삭제하자

```
$ rm -rf elastic
```

3. repo를 새로 clone 받자

```
$ git clone -b class3 https://github.com/higee/elastic.git
```

4. elastic stack를 detached mode로 실행하자

```
$ cd elastic/Week4_Elasticsearch/code/install
$ docker-compose up -d
```

5. (약 3분 후) Logstash container에 접속하자

```
$ docker exec -u 0 -it logstash bash
```

Input - stdin

- logstash 실행 후 console에 입력한 데이터 수집
- 간단한 logstash 테스트 할 때 혹은 디버깅 할 때 주로 사용

logstash configuration 파일 (stdin.conf) 생성하자

```
$ vi stdin.conf
```

logstash conf 작성 👑

```
input {
  stdin {}
}

output {
  stdout {}
}
```

- Logstash는 input plugin과 output plugin이 최소 1개씩 있어야 한다
- 그러므로 input 학습 실습이지만 기본 output인 stdout을 설정했다
- Filter는 필수가 아니다

logstash를 실행하자

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

(약 1분 후) 아래와 같이 나오면...

```
bash-4.2$ bin/logstash -f stdin.conf
WARNING: Default JAVA_OPTS will be overridden by the JAVA_OPTS defined in the environment. Environment JAVA_OPTS are -Xms128m -Xmx128m -XX:-AssumeMP
Sending Logstash's logs to /usr/share/logstash/logs which is now configured via log4j2.properties
[2018-04-23T17:25:04,056][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"fb_apache", :directory=>"/usr/share/logstash/modules/fb_apache/configuration"}
[2018-04-23T17:25:04,063][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"netflow", :directory=>"/usr/share/logstash/modules/netflow/configuration"}
[2018-04-23T17:25:04,343][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"arcsight", :directory=>"/usr/share/logstash/vendor/bundle/jruby/1.9/gems/x-pack-5.6.4-java/modules/arcsight/configuration"}
[2018-04-23T17:25:05,287][INFO ][logstash.outputs.elasticsearch] Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://logstash_system:xxxxxx@elasticsearch:9200/]}}
[2018-04-23T17:25:05,288][INFO ][logstash.outputs.elasticsearch] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://logstash_system:xxxxxx@elasticsearch:9200/, :path=>"/"}
[2018-04-23T17:25:05,556][WARN ][logstash.outputs.elasticsearch] Restored connection to ES instance {:url=>"http://logstash_system:xxxxxx@elasticsearch:9200/"}
[2018-04-23T17:25:05,639][INFO ][logstash.outputs.elasticsearch] New Elasticsearch output {:class=>"LogStash::Outputs::ElasticSearch", :hosts=>["http://elasticsearch:9200"]}
[2018-04-23T17:25:05,639][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>".monitoring-logstash", "pipeline.workers"=>1, "pipeline.batch.size"=>2, "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>2}
[2018-04-23T17:25:05,671][INFO ][logstash.licensechecker.licensereader] Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://logstash_system:xxxxxx@elasticsearch:9200/]}}
[2018-04-23T17:25:05,675][INFO ][logstash.licensechecker.licensereader] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://logstash_system:xxxxxx@elasticsearch:9200/, :path=>"/"}
[2018-04-23T17:25:05,687][WARN ][logstash.licensechecker.licensereader] Restored connection to ES instance {:url=>"http://logstash_system:xxxxxx@elasticsearch:9200/"}
[2018-04-23T17:25:05,771][INFO ][logstash.pipeline        ] Pipeline .monitoring-logstash started
[2018-04-23T17:25:05,796][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>"main", "pipeline.workers"=>1, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>125}
[2018-04-23T17:25:05,815][INFO ][logstash.pipeline        ] Pipeline main started
The stdin plugin is now waiting for input:
[2018-04-23T17:25:05,958][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
```

```
The stdin plugin is now waiting for input:
[2018-04-23T17:25:05,958][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
[2018-04-23T17:25:15,775][INFO ][logstash.inputs.metrics  ] Monitoring License OK
hi
```

stdin으로 적당히 **hi** 입력하고 **Enter** 를 누르자

logstash 실행 결과

- stdin 입력값 (= 사용자가 직접 입력)
- 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 (= 화면에 직접 출력)
- stdin 입력값인 "hello world" 이외의 값은 시스템에서 제공

stdout 가독성 좋게 변경 👑

```
input {
  stdin {}
}
```

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

option 추가 하는 방법 확인 👑

logstash를 실행하면...

codec 설정 전

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

codec 설정 후

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

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

다양한 input / output / filter plugin을 사용해도 큰 틀은 동일하다

Logstash를 설치한 방법에 따라 상이할 수 있다 ♕

1	Logstash Home Directory 이동	`$ docker exec -u 0 -it logstash bash`

2	Logstash Configuration File 생성	`$ vi exercise.conf`
		1 input {
		2 stdin {}
		3 }
		4
		5 output {
		6 stdout {
		7 codec => rubydebug
		8 }
		9 }

3	Logstash 실행	`$ 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

logstash conf 작성 👑

```
input {
  file {
    path => "/usr/share/logstash/data/titanic.csv"
  }
}
```

file에서 데이터를 읽을 것이기에 file 입력

파일 경로 입력

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

logstash를 실행하면... 아무 것도 안 나온다.

```
bash-4.2$ vi titaninc.conf
bash-4.2$ bin/logstash -f titaninc.conf
WARNING: Default JAVA_OPTS will be overridden by the JAVA_OPTS defined in the environment. Environment JAVA_OPTS are -Xms128m -Xmx128m -XX:-AssumeMP
Sending Logstash's logs to /usr/share/logstash/logs which is now configured via log4j2.properties
[2018-04-22T17:18:53,176][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"fb_apache", :directory=>"/usr/share/logstash/modules/fb_apache/configuration"}
[2018-04-22T17:18:53,185][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"netflow", :directory=>"/usr/share/logstash/modules/netflow/configuration"}
[2018-04-22T17:18:53,464][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"arcsight", :directory=>"/usr/share/logstash/vendor/bundle/jruby/1.9/gems/x-pack-5.6.4-java/modules/arcsight/configuration"}
[2018-04-22T17:18:53,479][INFO ][logstash.setting.writabledirectory] Creating directory {:setting=>"path.queue", :path=>"/usr/share/logstash/data/queue"}
[2018-04-22T17:18:53,480][INFO ][logstash.setting.writabledirectory] Creating directory {:setting=>"path.dead_letter_queue", :path=>"/usr/share/logstash/data/dead_letter_queue"}
[2018-04-22T17:18:53,517][INFO ][logstash.agent           ] No persistent UUID file found. Generating new UUID {:uuid=>"71602167-19ea-47e7-bbac-328b375135f3", :path=>"/usr/share/logstash/data/uuid"}
[2018-04-22T17:18:54,937][INFO ][logstash.outputs.elasticsearch] Elasticsearch pool URLs updated {                oved=>[], :added=>[http://logstash_system:xxxxxx@elasticsearch:9200/]}}
[2018-04-22T17:18:54,938][INFO ][logstash.outputs.elasticsearch] Running health check to see if an                connection is working {:healthcheck_url=>http://logstash_system:xxxxxx@elasticsearch:9200/, :path=>"/"}
[2018-04-22T17:18:55,328][WARN ][logstash.outputs.elasticsearch] Restored con                //logstash_system:xxxxxx@                200/"}
[2018-04-22T17:18:55,479][INFO ][logstash.outputs.elasticsearch] New Elasti                uts::ElasticSearch",                icsearch:9200"]}
[2018-04-22T17:18:55,480][INFO ][logstash.pipeline        ] Starting pipel                eline.workers"=>1, "p                batch.s                "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>2}
[2018-04-22T17:18:55,540][INFO ][logstash.licensechecker.licensereader] El                earch pool                {:ch                =>{:removed=>[], :ad                tp://logst                em:xxxxxx@elasticsearch:9200/]}}
[2018-04-22T17:18:55,547][INFO ][logstash.licensechecker.licensereader] Ru                ealth che                se                n El                search connection is                {:healthc                =>http://logstash_system:xxxxxx@elasticsearch:9200/, :path=>"/"}
[2018-04-22T17:18:55,563][WARN ][logstash.licensechecker.licensereader] Rest                ES                >"http://logstash_system:xxxxxx@ela                :9200/"}
[2018-04-22T17:18:55,732][INFO ][logstash.pipeline        ] Pipeline .monitori                arted
[2018-04-22T17:18:55,757][INFO ][logstash.pipeline        ] Starting pipeline {"id                in", "pipe                orke                1, "pipeline.batch.size"=>125,                e.batch.delay"=>5, "pipeline.max_inflight"=>125}
[2018-04-22T17:18:56,230][INFO ][logstash.pipeline        ] Pipeline main started
[2018-04-22T17:18:56,338][INFO ][logstash.agent           ] Successfully started L                h API endp                :por                00}
[2018-04-22T17:19:05,736][INFO ][logstash.inputs.metrics  ] Monitoring Li
```

왜?

- **start_position**이라는 옵션이 default로 **end**로 설정되어 있기 때문이다.
- 즉, 새로 추가되는 데이터만 보겠다는 것인데 이걸 **beginning**으로 해줘야 된다.
- **다른 파일**로 다시 실행해보자

logstash conf 작성 👑

```
input {
  file {
    path => "/usr/share/logstash/data/titanic2.csv"
    start_position => "beginning"
  }
}
```

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

logstash를 실행하면... 파일을 처음부터 읽으므로 결과가 제대로 출력

```
{
      "@version" => "1",
          "host" => "b601c1c512e1",
          "path" => "/usr/share/logstash/data/titanic2.csv",
    "@timestamp" => 2018-04-22T17:34:20.678Z,
       "message" => "94,Dean,0,3,male,26,1,2,C.A. 2315,20.575,S"
}
{
      "@version" => "1",
          "host" => "b601c1c512e1",
          "path" => "/usr/share/logstash/data/titanic2.csv",
    "@timestamp" => 2018-04-22T17:34:20.679Z,
       "message" => "95,Coxon,0,3,male,59,0,0,364500,7.25,S"
}
{
      "@version" => "1",
          "host" => "b601c1c512e1",
          "path" => "/usr/share/logstash/data/titanic2.csv",
    "@timestamp" => 2018-04-22T17:34:20.679Z,
       "message" => "96,Shorney,0,3,male,29.69911765,0,0,374910,8.05,S"
}
```

심화

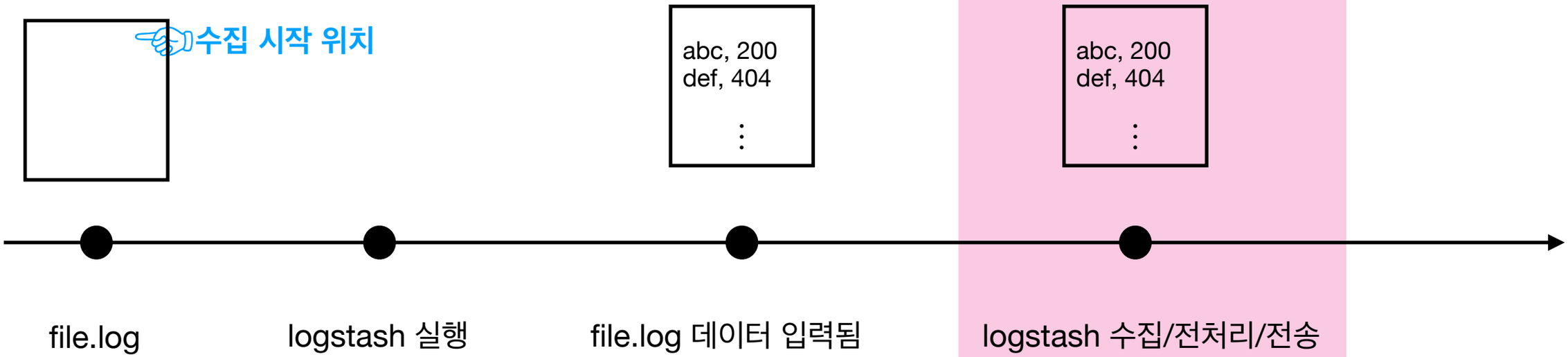
Q. 그래서 start_position option를 어떻게 하라는건지? 👑

A. logstash 실행 전에 file에 이미 데이터가 있는지 아닌지가 중요하다

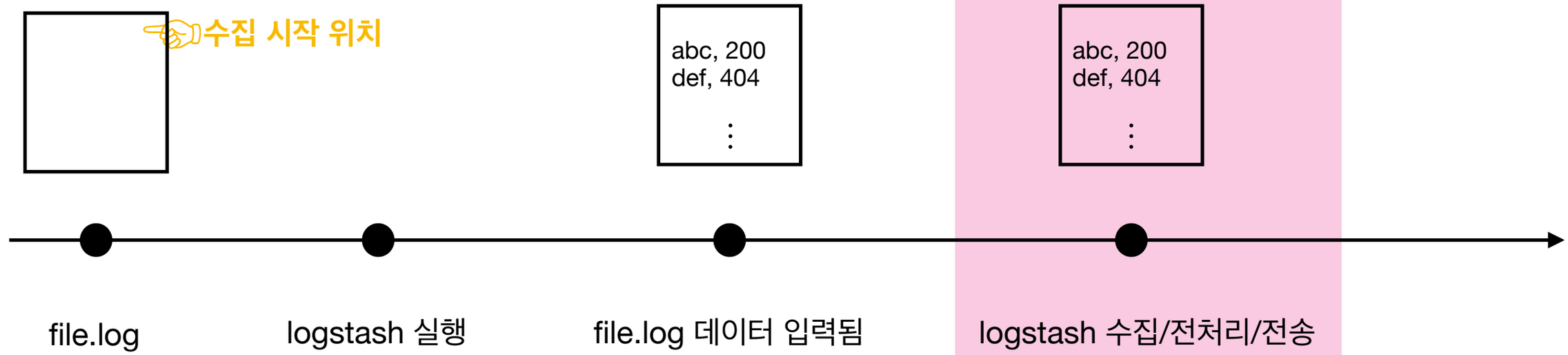
심화

logstash 실행 전 path file에 데이터가 없는 경우 : 차이가 없다

start_position : beginning



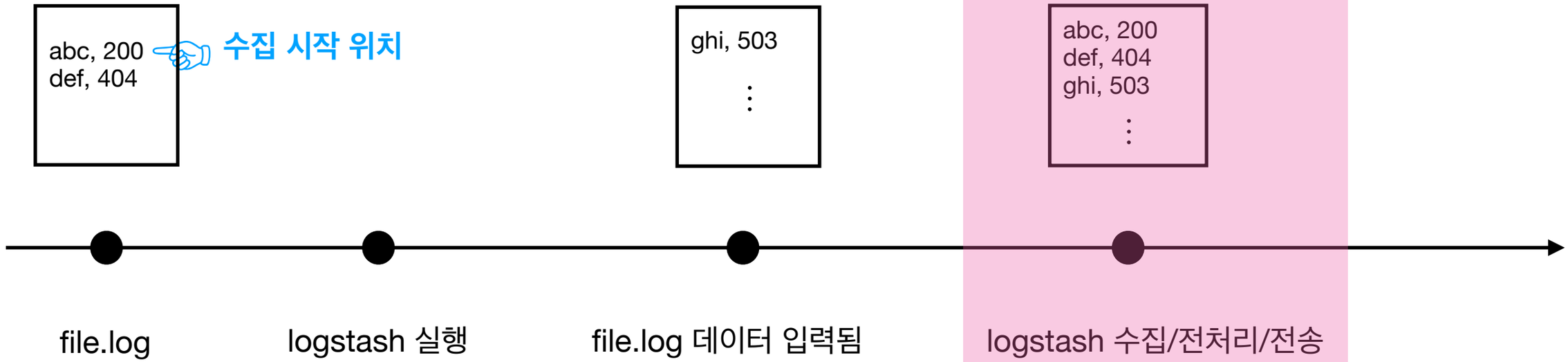
start_position : end



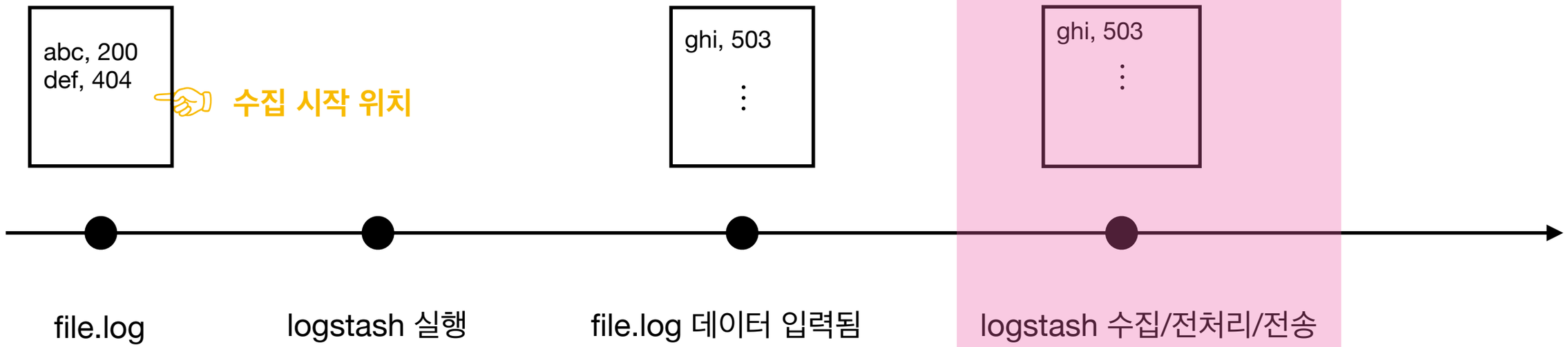
심화

logstash 실행 전 path file에 데이터가 있는 경우 : 차이가 있다

start_position : beginning



start_position : end



심화

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

그 이후에는 sincedb에 기록이 되어 효과가 없다.

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

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

```
bash-4.2$ pwd
/usr/share/logstash/data/plugins/inputs/file
bash-4.2$ ls -lah
total 8.0K
drwxr-xr-x 2 logstash logstash 104 Apr 22 17:37 .
drwxr-xr-x 3 logstash logstash  18 Apr 22 17:18 ..
-rw-r--r-- 1 logstash logstash  22 Apr 22 17:24 .sincedb_b503666ae29e95ab8024b1fd6cbc694c
-rw-r--r-- 1 logstash logstash  22 Apr 22 17:37 .sincedb_dae92a8984be95dcb3377c3b9394a87f
```

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

```
bash-4.2$ cat .sincedb_b503666ae29e95ab8024b1fd6cbc694c
14495087 0 51713 2310
```

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. 👑

한마디로, 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
    - 수정

```
bash-4.2$ pwd
/usr/share/logstash/data/plugins/inputs/file
bash-4.2$ ls -lah
total 20K
drwxr-xr-x 2 logstash logstash 104 Apr 22 17:37 .
drwxr-xr-x 3 logstash logstash  18 Apr 22 17:18 ..
-rw-r--r-- 1 logstash logstash  22 Apr 22 17:24 .sincedb_b503666ae29e95ab8024b1fd6cbc694c
-rw-r--r-- 1 logstash logstash  22 Apr 22 17:37 .sincedb_dae92a8984be95dcb3377c3b9394a87f
```

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

```
bash-4.2$ cat .sincedb_b503666ae29e95ab8024b1fd6cbc694c
14495087 0 51713 2310
```

심화

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.

심화

logstash conf 작성 👑

```
input {
  file {
    path => "/usr/share/logstash/data/titanic.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}
```

이렇게 설정하면

a) 이미 조회한 데이터이지만 재수집된다.

b) 단, 이번에 실행한 결과에 대한 기록은 sincedb에 저장하지 않는다.

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

심화

logstash를 실행하면...

```
{
      "@version" => "1",
          "host" => "b601c1c512e1",
          "path" => "/usr/share/logstash/data/titanic.csv",
    "@timestamp" => 2018-04-22T17:44:51.223Z,
       "message" => "46,Rogers,0,3,male,29.69911765,0,0,S.C./A.4. 23567,8.05,S"
}
{
      "@version" => "1",
          "host" => "b601c1c512e1",
          "path" => "/usr/share/logstash/data/titanic.csv",
    "@timestamp" => 2018-04-22T17:44:51.227Z,
       "message" => "47,Lennon,0,3,male,29.69911765,1,0,370371,15.5,Q"
}
{
      "@version" => "1",
          "host" => "b601c1c512e1",
          "path" => "/usr/share/logstash/data/titanic.csv",
    "@timestamp" => 2018-04-22T17:44:51.228Z,
       "message" => "48,O'Driscoll,1,3,female,29.69911765,0,0,14311,7.75,Q"
}
```

Input - jdbc

- jdbc를 지원하는 database 데이터를 수집할 때 사용하는 plugin
- 실습 database 정보 (mysql)
  - uri : mysql://13.125.153.139:3306/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        |
```

⋮

logstash conf 작성 👑

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.125.153.139:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/usr/share/logstash/driver/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
  }
}
```

☞ mysql uri

☝ jdbc driver library path
(docker로 설치할 때 함께 설치)

☝ 데이터 조회를 위한 sql 작성부
=> fc라는 table의 모든 data를 조회하라는 의미이다

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

logstash를 실행하면...

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

⋮

심화

Q. jdbc input도 logstash가 어디까지 조회했는지 기록을 하나?

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

(sql_last_value)

심화

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

심화

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

심화

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

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.125.153.139:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/usr/share/logstash/driver/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 => "/usr/share/logstash/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

```
bash-4.2$ cat id-under-5.db
--- 4
```

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

심화

Step3) sql_last_value를 이용해서 데이터를 조회하자 👑

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.125.153.139:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/usr/share/logstash/driver/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 => "/usr/share/logstash/id-under-5.db"
    statement => "
      SELECT *
      FROM fc
      WHERE id > :sql_last_value
    "
 }
}

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

이 부분을 수정했다

심화

Step4) logstash를 실행하면.. id > 4 데이터가 조회됐다

```
[2018-04-22T18:13:02,388][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"fb_apache", :directory=>"/usr/share/logstash/modules/fb_apache/configuration"}
[2018-04-22T18:13:02,397][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"netflow", :directory=>"/usr/share/logstash/modules/netflow/configuration"}
[2018-04-22T18:13:02,672][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"arcsight", :directory=>"/usr/share/logstash/vendor/bundle/jruby/1.9/gems/x-pack-5.6.4-java/modules/arcsight/configuration"}
[2018-04-22T18:13:03,803][INFO ][logstash.outputs.elasticsearch] Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://logstash_system:xxxxxx@elasticsearch:9200/]}}
[2018-04-22T18:13:03,805][INFO ][logstash.outputs.elasticsearch] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://logstash_system:xxxxxx@elasticsearch:9200/, :path=>"/"}
[2018-04-22T18:13:04,153][WARN ][logstash.outputs.elasticsearch] Restored connection to ES instance {:url=>"http://logstash_system:xxxxxx@elasticsearch:9200/"}
[2018-04-22T18:13:04,285][INFO ][logstash.outputs.elasticsearch] New Elasticsearch output {:class=>"LogStash::Outputs::ElasticSearch", :hosts=>["http://elasticsearch:9200"]}
[2018-04-22T18:13:04,286][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>".monitoring-logstash", "pipeline.workers"=>1, "pipeline.batch.size"=>2, "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>2}
[2018-04-22T18:13:04,336][INFO ][logstash.licensechecker.licensereader] Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://logstash_system:xxxxxx@elasticsearch:9200/]}}
[2018-04-22T18:13:04,343][INFO ][logstash.licensechecker.licensereader] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://logstash_system:xxxxxx@elasticsearch:9200/, :path=>"/"}
[2018-04-22T18:13:04,358][WARN ][logstash.licensechecker.licensereader] Restored connection to ES instance {:url=>"http://logstash_system:xxxxxx@elasticsearch:9200/"}
[2018-04-22T18:13:04,437][INFO ][logstash.pipeline        ] Pipeline .monitoring-logstash started
[2018-04-22T18:13:04,467][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>"main", "pipeline.workers"=>1, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>125}
[2018-04-22T18:13:04,795][INFO ][logstash.pipeline        ] Pipeline main started
[2018-04-22T18:13:04,910][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
[2018-04-22T18:13:06,734][INFO ][logstash.inputs.jdbc     ] (0.043000s)
    SELECT *
    FROM fc
    WHERE id > 4
```

내부적으로 코드가 수정됐다

```
{
          "@timestamp" => 2018-04-22T18:13:06.790Z,
               "name" => "Jessie",
           "@version" => "1",
           "location" => "US",
                 "id" => 5,
    "employment_status" => "employed",
             "salary" => 4500,
                "age" => 38
}
{
          "@timestamp" => 2018-04-22T18:13:06.793Z,
               "name" => "Jennifer",
           "@version" => "1",
           "location" => "US",
                 "id" => 6,
    "employment_status" => "unemployed",
             "salary" => 5200,
                "age" => 31
}
{
          "@timestamp" => 2018-04-22T18:13:06.793Z,
               "name" => "Bob",
```

id = 5 부터 조회

심화

Q. 데이터베이스에서 정기적으로 데이터를 조회할 수 없나?

A. scheduling을 이용해서 가능하다

심화

하루에 한 번 오전 6시에 데이터베이스에 데이터가 입력된다고 하자 ♕

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.125.153.139:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/usr/share/logstash/driver/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
    schedule => "0 6 * * *"
  }
}

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

☞ 스케줄 시간 설정

```
 * * * * *
 | | | | |
 | | | | |
 | | | | +---- 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)
```

심화

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

- elasticsearch에 저장된 데이터를 수집할 때 사용하는 plugin
- 어떤 use case가 있을까?
  - 데이터를 csv로 export 할 경우
  - 서로 다른 cluster 상의 elasticsearch 간 데이터 전송
- 실습 elasticsearch 정보
  - host: http://13.125.153.139:9200
  - index : week5

원본 데이터

Dev Tools

Console

```
GET week5/_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",
        "_type": "week5",
        "_id": "AWLunwbNzMQVnr-9MyP7",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-04-22T18:30:21.900Z",
          "name": "Tom",
          "@version": "1",
          "location": "US",
          "id": 2,
          "employment_status": "employed",
          "salary": 45000,
          "age": 33
        }
      },
```

logstash conf 작성 👑

```
input {
  elasticsearch {
    hosts => ["13.125.153.139:9200"]
    index => "week5"
    query => '{
      "query" : {
        "match_all" : {}
      }
    }'
  }
}
```

Query DSL을 알면 좋은 이유가 하나 더 늘었다

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

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
- 실습 elasticsearch 정보
  - host: http://13.125.153.139:9200
  - index : week5

원본 데이터


```
{
  "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
        }
      },
```


⋮

logstash conf 작성 👑

```
input {
  elasticsearch {
    hosts => ["13.125.153.139:9200"]
    index => "week5"
    query => '{
      "query" : {
        "match_all" : {}
      }
    }'
  }
}
```

```
output {
  csv {
    fields => ["name", "location", "employment_status", "salary",  "age"]
    path => "/usr/share/logstash/data/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

logstash를 실행하고 week5.csv를 열어보자

```
bash-4.2$ cat week5.csv | head -5
Tom,US,employed,45000,33
Ken,US,employed,3500,27
Kelly,UK,employed,4500,28
Tim,UK,employed,47000,38
Saeki,JP,employed,4500,44
```

⋮

Output - elasticsearch

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

logstash conf 작성 👑

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.125.153.139:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/usr/share/logstash/driver/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
  }
}

output {
  elasticsearch {
    hosts => ["elasticsearch:9200"]
    index => "exercise1"
    document_type => "exercise1"
  }
}
```

☜ es에 데이터를 전송하기 위해 es output plugin 사용

☜ docker network로 묶여있기에 사용 가능.

그렇지 않으면 ["http://13.125.153.139:9200"] 와 같이 설정

실행해보면, 작업 후에 콘솔에 아무 표시가 없다

```
Sending Logstash's logs to /usr/share/logstash/logs which is now configured via log4j2.properties
[2018-04-23T04:07:01,005][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"fb_apache", :directory=>"/usr/share/logstash/modules/fb_apache/configuration"}
[2018-04-23T04:07:01,019][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"netflow", :directory=>"/usr/share/logstash/modules/netflow/configuration"}
[2018-04-23T04:07:01,292][INFO ][logstash.modules.scaffold] Initializing module {:module_name=>"arcsight", :directory=>"/usr/share/logstash/vendor/bundle/jruby/1.9/gems/x-pack-5.6.4
-java/modules/arcsight/configuration"}
[2018-04-23T04:07:02,653][INFO ][logstash.outputs.elasticsearch] Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://logstash_system:xxxxxx@elasticsearch:9200/
]}}
[2018-04-23T04:07:02,660][INFO ][logstash.outputs.elasticsearch] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://logstash_system:xxxx
xx@elasticsearch:9200/, :path=>"/"}
[2018-04-23T04:07:03,024][WARN ][logstash.outputs.elasticsearch] Restored connection to ES instance {:url=>"http://logstash_system:xxxxxx@elasticsearch:9200/"}
[2018-04-23T04:07:03,167][INFO ][logstash.outputs.elasticsearch] New Elasticsearch output {:class=>"LogStash::Outputs::ElasticSearch", :hosts=>["http://elasticsearch:9200"]}
[2018-04-23T04:07:03,168][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>".monitoring-logstash", "pipeline.workers"=>1, "pipeline.batch.size"=>2, "pipeline.batch.delay"=
>5, "pipeline.max_inflight"=>2}
[2018-04-23T04:07:03,223][INFO ][logstash.licensechecker.licensereader] Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://logstash_system:xxxxxx@elasticsearc
h:9200/]}}
[2018-04-23T04:07:03,226][INFO ][logstash.licensechecker.licensereader] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://logstash_syst
em:xxxxxx@elasticsearch:9200/, :path=>"/"}
[2018-04-23T04:07:03,243][WARN ][logstash.licensechecker.licensereader] Restored connection to ES instance {:url=>"http://logstash_system:xxxxxx@elasticsearch:9200/"}
[2018-04-23T04:07:03,314][INFO ][logstash.pipeline        ] Pipeline .monitoring-logstash started
[2018-04-23T04:07:03,354][INFO ][logstash.outputs.elasticsearch] Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://13.125.153.139:9200/]}}
[2018-04-23T04:07:03,360][INFO ][logstash.outputs.elasticsearch] Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://13.125.153.139:9200/
, :path=>"/"}
[2018-04-23T04:07:03,432][WARN ][logstash.outputs.elasticsearch] Restored connection to ES instance {:url=>"http://13.125.153.139:9200/"}
[2018-04-23T04:07:03,476][INFO ][logstash.outputs.elasticsearch] Using mapping template from {:path=>nil}
[2018-04-23T04:07:03,478][INFO ][logstash.outputs.elasticsearch] Attempting to install template {:manage_template=>{"template"=>"logstash-*", "version"=>50001, "settings"=>{"index.r
efresh_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"=>{"k
eyword"=>{"type"=>"keyword", "ignore_above"=>256}}}}}], "properties"=>{"@timestamp"=>{"type"=>"date", "include_in_all"=>false}, "@version"=>{"type"=>"keyword", "include_in_all"=>fal
se}, "geoip"=>{"dynamic"=>true, "properties"=>{"ip"=>{"type"=>"ip"}, "location"=>{"type"=>"geo_point"}, "latitude"=>{"type"=>"half_float"}, "longitude"=>{"type"=>"half_float"}}}}}}}
}
[2018-04-23T04:07:03,518][INFO ][logstash.outputs.elasticsearch] New Elasticsearch output {:class=>"LogStash::Outputs::ElasticSearch", :hosts=>["//13.125.153.139:9200"]}
[2018-04-23T04:07:03,526][INFO ][logstash.pipeline        ] Starting pipeline {"id"=>"main", "pipeline.workers"=>1, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.
max_inflight"=>125}
[2018-04-23T04:07:03,791][INFO ][logstash.pipeline        ] Pipeline main started
[2018-04-23T04:07:03,971][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
[2018-04-23T04:07:05,727][INFO ][logstash.inputs.jdbc     ] (0.046000s) SELECT * FROM fc
[2018-04-23T04:07:06,808][WARN ][logstash.agent           ] stopping pipeline {:id=>".monitoring-logstash"}
[2018-04-23T04:07:08,165][WARN ][logstash.agent           ] stopping pipeline {:id=>"main"}
bash-4.2$
```

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

제대로 된 건지 확인하기 위해 kibana에서 확인하자

```
GET exercise1/_search
```


```
{
  "took": 5,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 33,
    "max_score": 1,
    "hits": [
      {
        "_index": "exercise1",
        "_type": "exercise1",
        "_id": "AWLymFrnLP4i7fZOrfgd",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-04-23T13:01:33.511Z",
          "name": "Ken",
          "@version": "1",
          "location": "US",
          "id": 4,
          "employment_status": "employed",
          "salary": 3500,
          "age": 27
        }
      },
      {
```


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?

1. es index를 생성할 때 날짜 정보를 넣을 수 있나?

```
GET exercise1/_search
```


```
{
  "took": 5,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 33,
    "max_score": 1,
    "hits": [
      {
        "_index": "exercise1",
        "_type": "exercise1",
        "_id": "AWLymFrnLP4i7fZOrfgd",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-04-23T13:01:33.511Z",
          "name": "Ken",
          "@version": "1",
          "location": "US",
          "id": 4,
          "employment_status": "employed",
          "salary": 3500,
          "age": 27
        }
      },
```


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

logstash conf 작성 👑

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.125.153.139:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/usr/share/logstash/driver/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
 }
}
```

```
output {
  elasticsearch {
    hosts => ["elasticsearch:9200"]
    index => "exercise1-%{+YYYY.MM.DD}"
    document_type => "exercise1"
  }
}
```

@timestamp에서 연도, 월, 일을 추출해서 index이름을 생성할 때 추가해준다.

logstash를 실행하고 Kibana에서 확인하자

Dev Tools

Console Search Profiler Grok Debugger

```
GET exercise1-2018.04.23/_search
{
  "query": {
    "match_all": {}
  }
}
```

다음과 같은 조합으로 index 이름 생성

- static : exercise
- dynamic : 2018.04.23


```
{
  "took": 2,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 33,
    "max_score": 1,
    "hits": [
      {
        "_index": "exercise1-2018.04.23",
        "_type": "exercise1",
        "_id": "AWLyp5_ULP4i7fZOrf2I",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-04-23T13:18:14.164Z",
          "name": "Kirk",
          "@version": "1",
          "location": "US",
          "id": 3,
          "employment_status": "employed",
          "salary": 3000,
          "age": 23
        }
      },
```

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

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.125.153.139:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/usr/share/logstash/driver/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
 }
}
```

```
output {
  elasticsearch {
    hosts => ["elasticsearch:9200"]
    index => "exercise2-%{+YYYY.MM.DD}"
    document_type => "exercise2"
    document_id => "%{@timestamp}-%{name}"
  }
}
```

%{@timestamp}와 %{name} 부분에 각event에서 수집한 실제 value가 입력된다

logstash를 실행하고 Kibana에서 확인하자

Dev Tools

Console | Search Profiler | Grok Debugger

```
GET exercise2-2018.04.23/_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": "exercise2-2018.04.23",
        "_type": "exercise2",
        "_id": "2018-04-23T13:22:05.856Z-Tim",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-04-23T13:22:05.856Z",
          "name": "Tim",
          "@version": "1",
          "location": "UK",
          "id": 16,
          "employment_status": "employed",
          "salary": 47000,
          "age": 38
        }
      },
```

Output - multiple output plugins

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

logstash conf 작성 👑

```
input {
  jdbc {
    jdbc_validate_connection => true
    jdbc_connection_string => "jdbc:mysql://13.125.153.139:3306/fc"
    jdbc_user => "fc"
    jdbc_password => "fc"
    jdbc_driver_library => "/usr/share/logstash/driver/mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    statement => "SELECT * FROM fc"
 }
}
```

```
output {
  elasticsearch {
    hosts => ["elasticsearch:9200"]
    index => "exercise3"
    document_type => "exercise3"
  }
  stdout {
    codec => rubydebug
  }
}
```

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

logstash를 실행하면

stdout

```
{
          "@timestamp" => 2018-04-23T13:26:04.747Z,
                "name" => "Yoon",
            "@version" => "1",
            "location" => "KR",
                  "id" => 31,
    "employment_status" => "unemployed",
              "salary" => 3500,
                 "age" => 23
}
{
          "@timestamp" => 2018-04-23T13:26:04.747Z,
                "name" => "Yang",
            "@version" => "1",
            "location" => "KR",
                  "id" => 32,
    "employment_status" => "unemployed",
              "salary" => 3600,
                 "age" => 23
}
```

⋮

elasticsearch

Dev Tools

Console | Search Profiler | Grok Debugger

```
GET exercise3/_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": "exercise3",
        "_type": "exercise3",
        "_id": "AWLyrs32LP4i7fZOrgB5",
        "_score": 1,
        "_source": {
          "@timestamp": "2018-04-23T13:26:04.705Z",
          "name": "Josh",
          "@version": "1",
          "location": "US",
          "id": 1,
          "employment_status": "employed",
          "salary": 4000,
          "age": 33
        }
      },
```

⋮

Conditionals ♕

- 특정한 조건에 따라 **filter** 또는 **output**을 선별적으로 적용하고 싶을 때
- 분기는 아래와 같이 가능
  - **if**
  - **else if**
  - **else**

logstash conf 작성 👑

```
input {
  stdin {}
}

output {
  if [message] in ["Seoul", "Busan"] {
    elasticsearch {
      hosts => ["elasticsearch:9200"]
      index => "korea"
    }
  }
  else if [message] == "Osaka" {
    elasticsearch {
      hosts => ["elasticsearch:9200"]
      index => "japan"
    }
  }
  else {
    stdout {
      codec => rubydebug
    }
  }
}
```

logstash를 실행하고 다음을 입력해보자

stdout

```
hi
{
      "@version" => "1",
          "host" => "b601c1c512e1",
    "@timestamp" => 2018-04-23T13:43:48.041Z,
       "message" => "hi"
}
Osaka
Seoul
```

elasticsearch

Dev Tools

Console Search Profiler Grok Debugger

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

```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 1,
    "hits": [
      {
        "_index": "korea",
        "_type": "logs",
        "_id": "AWLyvx2nLP4i7fZOrgfH",
        "_score": 1,
        "_source": {
          "@version": "1",
          "host": "b601c1c512e1",
          "@timestamp": "2018-04-23T13:43:53.899Z",
          "message": "Seoul"
        }
      }
```

Dev Tools

Console Search Profiler Grok Debugger

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

```
{
  "took": 0,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 1,
    "hits": [
      {
        "_index": "japan",
        "_type": "logs",
        "_id": "AWLyvw7uLP4i7fZOrge0",
        "_score": 1,
        "_source": {
          "@version": "1",
          "host": "b601c1c512e1",
          "@timestamp": "2018-04-23T13:43:50.148Z",
          "message": "Osaka"
        }
      }
    ]
  }
}
```

Filter - csv

- 특정한 separator로 연결된 데이터 (예: csv) 파일을 parsing할 때 유용

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
```

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"
}
```

logstash conf 작성 👑

```
input {
  file {
    path => "/usr/share/logstash/data/titanic-header.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}
```

☜ 계속 실험할 것이기에 설정

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

☝ ,로 구분된 데이터이기에 ‘,' 입력

☜

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

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

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 => "/usr/share/logstash/data/titanic-header.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}
```

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

☝ 이 옵션을 이용해서 첫번째 row의 데이터를 column으로 사용하도록 설정

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

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"
}
```

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"
}
```

logstash conf 작성 👑

```
input {
  file {
    path => "/usr/share/logstash/data/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 {
  stdout {
    codec => rubydebug
  }
}
```

☞ "Pclass" Field의 Type을 integer로

"Index" Field의 Type을 integer로

"Survival" Field의 Type을 integer로

"Fare" Field의 Type을 float로

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 - mutate

- Field data의 값을 변형할 때 사용하는 강력한 plugin
- 여러가지 option을 혼합해서 같이 사용한다

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
```

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 👑

```
input {
  file {
    path => "/usr/share/logstash/data/ip-address.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  mutate {
    split => {
      "message" => ":"
    }
  }
}

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

데이터를 :를 기준으로 split 해보자

logstash를 실행하면...

```
{
       "@version" => "1",
           "host" => "b601c1c512e1",
           "path" => "/usr/share/logstash/data/ip-address.log",
     "@timestamp" => 2018-04-23T20:06:50.278Z,
        "message" => [
         [0] "13.124.230.195",
         [1] "5601"
    ]
}
```

기존 message field "13.124.230.195:5601"

split된 데이터로 각각 ip, port라는 field를 생성해보자

```
{
       "@version" => "1",
           "host" => "b601c1c512e1",
           "path" => "/usr/share/logstash/data/ip-address.log",
     "@timestamp" => 2018-04-23T20:06:50.278Z,
        "message" => [
         [0] "13.124.230.195",
         [1] "5601"
    ]
}
```

활용

ip : "13.124.230.195"

port : "5601"

logstash conf 작성 👑

```
input {
  file {
    path => "/usr/share/logstash/data/ip-address.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}
```

```
filter {
  mutate {
    split => {
      "message" => ":"
    }
    add_field => {
      "ip" => "%{message[0]}"
      "port" => "%{message[1]}"
    }
  }
}
```

☞ split된 message의 첫 번째 데이터

☞ split된 message의 두 번째 데이터

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

logstash를 실행하면...

```
{
          "path" => "/usr/share/logstash/data/ip-address.log",
    "@timestamp" => 2018-04-23T20:15:57.244Z,
          "port" => "5601",
            "ip" => "13.124.230.195",
      "@version" => "1",
          "host" => "b601c1c512e1",
       "message" => [
        [0] "13.124.230.195",
        [1] "5601"
    ]
}
```

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

logstash conf 작성 👑

```
input {
  file {
    path => "/usr/share/logstash/data/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"]
  }
}
```

☝ 제거할 field 이름을 직접 나열

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

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 - grok

- 구조화되어 있지 않은 데이터를 처리하는데 강력한 **plugin**
- 실제 **production**에서 생기는 **log**를 분석할 때 **grok + 다른 plugin** 조합 사용

아래와 같은 로그가 생긴다고 하자

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"
```

res_time req

심화

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

심화

1. 가지고 있는 로그 파일 패턴을 발견한다 (~~최소한 노력한다~~)

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"
```

res_time req

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

심화

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

(각 pattern이 무엇을 의미하는지 사전 학습 필요)

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"
```

res_time req

- time : TIMESTAMP_ISO8601
- client_ip : IPORHOST
- res_time : NUMBER
- status_code : NUMBER
- req : QS

심화

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

심화

4. logstash conf 작성한다 👑

```
input {
  file {
    path => "/usr/share/logstash/data/access.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

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

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

☝ grok debugger에서 작성했던 코드를 입력한다

심화

logstash를 실행하면...

```
{
          "path" => "/usr/share/logstash/data/access.log",
    "@timestamp" => 2018-04-23T20:35:16.498Z,
   "status_code" => "200",
      "res_time" => "0.079",
      "clientip" => "172.30.39.133",
      "@version" => "1",
          "host" => "b601c1c512e1",
          "time" => "2018-02-05T05:49:53.859060Z",
       "message" => "2018-02-05T05:49:53.859060Z 172.30.39.133 0.079 200 \"GET https://helloworld.com/users/1 HTTP/1.1\"",
           "req" => "\"GET https://helloworld.com/users/1 HTTP/1.1\""
}
{
          "path" => "/usr/share/logstash/data/access.log",
    "@timestamp" => 2018-04-23T20:35:16.507Z,
   "status_code" => "404",
      "res_time" => "0.1",
      "clientip" => "172.29.43.253",
      "@version" => "1",
          "host" => "b601c1c512e1",
          "time" => "2018-02-05T06:49:53.859060Z",
       "message" => "2018-02-05T06:49:53.859060Z 172.29.43.253 0.1 404 \"GET https://helloworld.com/users/3 HTTP/1.1\"",
           "req" => "\"GET https://helloworld.com/users/3 HTTP/1.1\""
}
```

심화

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

```
{
    "status_code" => "200",
       "res_time" => "0.079",
           "verb" => "\"GET",
            "uri" => "https://helloworld.com/users/1",
       "clientip" => "172.30.39.133",
           "http" => "HTTP/1.1\"",
           "time" => "2018-02-05T05:49:53.859060Z"
}
{
    "status_code" => "404",
       "res_time" => "0.1",
           "verb" => "\"GET",
            "uri" => "https://helloworld.com/users/3",
       "clientip" => "172.29.43.253",
           "http" => "HTTP/1.1\"",
           "time" => "2018-02-05T06:49:53.859060Z"
}
```

심화

아래와 같은 로그가 생긴다고 하자

```
127.0.0.1 - - [13/Dec/2015:03:02:45 -0800] "GET /xampp/status.php HTTP/1.1" 200 891 "http://cadenza/xampp/navi.php" "Mozilla/5.0 (Macintosh; Intel Mac OSX 10.9; rv:25.0) Gecko/20100101 Firefox/25.0"
123.222.333.123 HOME - [01/Feb/1998:01:08:46 -0800] "GET /bannerad/ad.htm HTTP/1.0" 200 2808 "http://www.referrer.com/bannerad/ba_intro.htm" "Mozilla/4.01 (Macintosh; I; PPC)"
daum.net HOME - [01/Apr/1998:01:09:14 -0800] "GET /bannerad/click.htm HTTP/1.0" 200 2070 "http://www.referrer.com/bannerad/menu.htm" "Mozilla/4.01 (Macintosh; I; PPC)"
111.222.333.123 AWAY - [01/Apr/1998:01:09:14 -0800] "GET /bannerad/click.htm HTTP/1.0" 200 2070 "http://www.referrer.com/bannerad/menu.htm" "Mozilla/4.01 (Macintosh; I; PPC)"
unicomp6.unicomp.net AWAY - [01/Apr/1998:01:09:14 -0800] "GET /bannerad/click.htm HTTP/1.0" 200 2070 "http://www.referrer.com/bannerad/menu.htm" "Mozilla/4.01 (Macintosh; I; PPC)"
123.222.333.123 AWAY - [01/Feb/2003:01:08:53 -0800] "GET /bannerad/ad7.gif HTTP/1.0" 200 332 "http://www.referrer.com/bannerad/ba_ad.htm" "Mozilla/4.01 (Macintosh; I; PPC)"
```

심화

아래와 같은 결과를 얻으려면 어떻게 grok을 작성해야 할까?

```
{
        "request" => "/bannerad/click.htm",
          "agent" => "\"Mozilla/4.01 (Macintosh; I; PPC)\"",
           "auth" => "-",
          "ident" => "AWAY",
           "verb" => "GET",
        "message" => "unicomp6.unicomp.net AWAY - [01/Apr/1998:01:09:14 -0800] \
           "path" => "/usr/share/logstash/data/apache.log",
       "referrer" => "\"http://www.referrer.com/bannerad/menu.htm\"",
     "@timestamp" => 2018-04-23T20:43:49.278Z,
       "response" => "200",
          "bytes" => "2070",
       "clientip" => "unicomp6.unicomp.net",
       "@version" => "1",
           "host" => "b601c1c512e1",
    "httpversion" => "1.0",
      "timestamp" => "01/Apr/1998:01:09:14 -0800"
}
```

심화

많이 사용되는 패턴은 등록되어 있다 👑

```
input {
  file {
    path => "/usr/share/logstash/data/apache.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
}

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

Filter - date 👑

- elasticsearch는 기본적으로 입력된 시간을 utc로 생각한다
- 만약에 log data에 현지 시간으로 값이 입력되어 있으면 어떻게 될까?
- 이를 해결하기 위해 사용하는 게 date filter다

아래와 같은 로그가 생긴다고 하자

- 시간 : 한국 시간 기준
- 이름 : 특정 시간에 접속한 유저

```
1    2018-02-12T16:03:38 Ben
2    2018-02-13T03:25:31 John
3    2018-02-14T13:31:11 Leo
```

default

데이터를 전송할 index mapping을 잡아주고...

```
PUT exercise4
{
  "mappings" : {
    "exercise4" : {
      "properties" : {
        "time" : {
          "type" : "date"
        },
        "name" : {
          "type" : "keyword"
        }
      }
    }
  }
}
```

default

아래와 같은 logstash.conf로 데이터를 전송하자 👑

```
input {
  file {
    path => "/usr/share/logstash/data/test.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}

filter {
  grok {
    match => {
      "message" =>
        '%{TIMESTAMP_ISO8601:time} %{WORD:keyword}'
    }
    remove_field =>
      ["path", "@timestamp", "@version", "host", "message"]
  }
}

output {
  elasticsearch {
    hosts => ["elasticsearch:9200"]
    index => "exercise4"
    document_type => "exercise4"
  }
}
```

date filter

마찬가지로 데이터를 전송할 index mapping을 잡아주고...

```
PUT exercise5
{
  "mappings" : {
    "exercise5" : {
      "properties" : {
        "time" : {
          "type" : "date"
        },
        "name" : {
          "type" : "keyword"
        }
      }
    }
  }
}
```

date filter

아래와 같은 logstash.conf로 데이터를 전송하자 👑

```
input {
  file {
    path => "/usr/share/logstash/data/test.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}
```

```
filter {
  grok {
    match => {
      "message" =>
        '%{TIMESTAMP_ISO8601:time} %{WORD:name}'
    }
    remove_field =>
      ["path", "@timestamp", "@version", "host", "message"]
  }
 date {
    match => ["time", "YYYY-MM-dd'T'HH:mm:ss"]
    timezone => "Asia/Seoul"
    target => "time"
  }
}
```

```
output {
  elasticsearch {
    hosts => ["elasticsearch:9200"]
    index => "exercise5"
    document_type => "exercise5"
  }
}
```

date filter 적용 전/후 elasticserch 저장된 값을 비교하자

입력값	default	date filter
2018-02-12 16:03:38	 { "_index": "exercise4", "_type": "exercise4", "_id": "AWL0hVoszMQVnr-9MyRS", "_score": 1, "_source": { "name": "Ben", "time": "2018-02-12T16:03:38" } },	 { "_index": "exercise5", "_type": "exercise5", "_id": "AWL0hnY-zMQVnr-9MyRV", "_score": 1, "_source": { "name": "Ben", "time": "2018-02-12T07:03:38.000Z" } },
2018-02-13 03:25:31	 { "_index": "exercise4", "_type": "exercise4", "_id": "AWL0hVoszMQVnr-9MyRT", "_score": 1, "_source": { "name": "John", "time": "2018-02-13T03:25:31" } },	 { "_index": "exercise5", "_type": "exercise5", "_id": "AWL0hnZ9zMQVnr-9MyRW", "_score": 1, "_source": { "name": "John", "time": "2018-02-12T18:25:31.000Z" } },
2018-02-14 013:31:11	 { "_index": "exercise4", "_type": "exercise4", "_id": "AWL0hVoszMQVnr-9MyRU", "_score": 1, "_source": { "name": "Leo", "time": "2018-02-14T13:31:11" } }	 { "_index": "exercise5", "_type": "exercise5", "_id": "AWL0hnZ9zMQVnr-9MyRX", "_score": 1, "_source": { "name": "Leo", "time": "2018-02-14T04:31:11.000Z" } }

Filter - drop 👑

- 특정한 event가 들어올 경우 event 자체를 무시할 경우 사용
- 예를 들어 log를 수집할 때 INFO 레벨은 무시하고 싶을 때 사용 가능

logstash conf 작성하고 실행해보자 👑

```
input {
  stdin {}
}
```

```
filter {
  if [message] == "hello" {
    drop {}
  }
}
```

message가 "hello"인 event 무시

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

만약에 access log에서 status code가 200이 아닌 것만 수집하려면? 👑

```
input {
  file {
    path => "/usr/share/logstash/data/access.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
 }
}

filter {
  grok {
    match => {
      "message" =>
       '%{TIMESTAMP_ISO8601:time} %{IPORHOST:clientip} %{NUMBER:res_time} %{NUMBER:status_code} %{QS:req}'
    }
  }
  if [status_code] == "200" {
    drop {}
  }
}

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

status code가 200인, 즉 정상인 결과는 수집하지 않고 에러 로그만 수집

질문 및 Feedback은

gshock94@gmail.com로 주세요