

1. Filebeat 설정

Filebeat 란? beats 중에 하나

Filebeat 의 역할은? 로그를 수집하는 것

> 그러므로 우리는 수집 경로(input)와 수집한 파일을 내보낼 경로(output)만 정해주면 된다

Filebeat 의 설정 파일 위치: /etc/filebeat/filebeat.yml

1.1 input

```
# ===== Filebeat inputs =====  
  
filebeat.inputs:  
- type: log  
  enabled: true  
  paths:  
    - /root/elasticTest/covid19_korea.csv
```

Input 플러그인은 우리가 사용해본 log 뿐 아니라 kafka 나 aws, http json 등 다양하게 정해줄 수 있으며 그에 따라서 옵션도 달라진다. 자세한 내용은 아래 링크의 Input tpyes 항목을 참조하자(영어).

<https://www.elastic.co/guide/en/beats/filebeat/current/configuration-filebeat-options.html>

여기서는 대표적인 log 플러그인의 옵션을 살펴보자

Log 옵션	
path: ~	데이터를 받아올 경로
enabled: true	이 input 경로를 사용할지 (true = 사용하겠다)
encoding: plain	데이터를 받아올 때 encoding 여부 (utf-8, euc-kr 등 사용가능)
include_lines: [^'WARN']	꼭 포함할 줄(줄 단위로 불러오니까) 명시, 정규표현식으로 표현, 여기의 경우 WARN 으로 시작하는 것은 무조건 포함

exclude_lines: [^'WARN']	반대로 포함하지 않을 줄 명시
exclude_files: ['*.log\$']	여러 파일을 수집할 경우 포함하지 않을 파일 명시 이 경우 확장자가 log 인 파일은 제외하고 수집
keep_null	빈 값도 수집할지 여부를 정해주는 옵션 기본 값은 false 이다

옵션은 다양하니까 자세한 내용은 다음 링크를 참조하자(영어)

<https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-log.html>

아래처럼 여러 개 작성해서 여러 경로로 수집도 가능하다

```
filebeat.inputs:
- type : log
  path:
    - /첫번째 수집 경로
  fields:
    service: apache (*얘는 나중에 경로에 따라 구분해서 처리하기 위해 구분자를
    달아주는 것, 주로 받아오는 경로를 대표할 수 있는 이름을 적어줌)

- type : log
  path:
    - /두번째 수집 경로
  fields:
    service: postgres
```

위처럼 fields 의 service 를 정해줄 경우 logstash 에서 아래처럼 구분해서 받아줄 수 있다.

```
[예시]
input {
  Beats {
    Port => "5044"
  }
}
```

```

filter {
  if [service] == "apache" { # filebeat 가 수집할 때 service 가 apache 인 데이터
    아파치용 필터 설정
  }
  else if [service] == "postgres" {
    postgres 용 필터 설정
  }
}
output {
  if [service] == "apache" {
    output 설정도 service 에 따라 설정 가능
  }
  else if [service] == "postgres" {
    postgres 용 output 설정
  }
}

```

1.2 output

output 도 다양한 플러그인으로 내보낼수 있는데 대표적으로 Elasticsearch 와 Logstash 로 내보낼 수 있다.

[Elasticsearch 로 내보내고 싶은 경우]

```

# ----- Elasticsearch Output -----
# output.elasticsearch
# Array of hosts to connect to.
# hosts: ["localhost:9200"]

# Protocol - either `http` (default) or `https`.
#protocol: "https"

# Authentication credentials - either API key or username/password.
#api_key: "id:api_key"
#username: "elastic"
#password: "changeme"

```

노란색으로 표시된 곳의 주석을 해제해준다

[Logstash 로 내보내고 싶은 경우]

```
# ----- Logstash Output -----
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]

  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"
```

노란색으로 표시된 곳의 주석을 해제해준다

2. logstash 설정

Logstash 란? filebeat 에서 불러온 데이터를 elasticsearch(DB)에 넣기 적절하게 가공하는 프로그램이다.

> 그러므로 우리가 설정해야 하는 건 데이터를 불러오는 input, 불러온 데이터를 가공하는 filter, 데이터를 내보내는 output 총 세가지이다.

데이터에 따라 input, output, filter 가 모두 달라지므로 elasticsearch 의 설정 파일은 하나로 고정되어 있지 않고 우리가 생성할 수 있다.

logstash 아래의 conf.d 파일 안에 있고 conf 확장자를 가지고 있다면 상관없다.

(우리 경로: /etc/logstash/conf.d)

설정 파일을 여러 개 만들어서 명령어로 교체해가며 사용할 수도 있다.

[참고] 다른 conf 파일로 교체하는 명령어

```
/usr/share/logstash/bin/logstash -r -f "/root/etc/logstash/conf.d/사용하려는파일.conf"
```

- f : 경로 알려주는 명령어

- r : 구성이 변경될 때 logstash 를 다시 시작하라는 명령어

우리는 처음 생성할 때 블로그를 따라 만들었던 first-pipeline.conf 파일을 사용해보자

2.1 input

input 플러그인은 file, jdbc, beats 등 다양하게 설정할 수 있지만 우리는 beats 의 일종인 filebeat 를 사용할 것이기 때문에 beats 로 설정한다.

다른 플러그인이 궁금하다면 아래 문서를 참조하자(영어)

<https://www.elastic.co/guide/en/logstash/current/input-plugins.htm>

[input 설정 예시 1 - beats]

```
input {
  beats {
    port => 5044
    host => "0.0.0.0"
  }
}
```

[input 설정 예시 2 - file]

```
input {
  file {
    path => "/var/log/messages"
    type => "syslog"
  }
}
```

옵션은 이밖에도 다양하지만 지금은 위의 port 나 host, path 정도만 알아둬도 좋을 것 같다

더 자세한 옵션은 다음 링크를 참조하자 (영어)

<https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-log.html>

2.2 filter

filter 가 바로 logstash 의 꽃이라고 할 수 있다.

DB 기능을 하는 elasticsearch 에 넣을 수 있도록 데이터를 가공하고 elasticsearch 에서 검색이나 계산할 수 있도록 맵핑해주는 역할을 한다.

filter 는 다양한 플러그인을 조합해서 사용한다

플러그인은 함수같은거라고 볼 수 있는데 글자를 잘라준다거나, 특정 글자만 치환한다거나, 데이터 타입을 정해준다거나(맵핑) 다양한 기능을 지원한다

한 플러그인만 쓰기보다는 여러 플러그인을 조합해서 사용하는 경우가 많다

플러그인의 구조는

플러그인명 1 {

 옵션 => 값

}

이런식으로 이뤄지며

플러그인명 1 {

 플러그인명 2 {

 플러그인명 2 의 옵션 => 값

 }

 플러그인명 1 의 옵션 => 값

}

이런식으로 중첩해서 사용도 가능하다

여기서는 일부 플러그인만 다룰 예정이라 자세한 내용은 아래 공식 문서를 참조하자 (영어)

<https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>

2.2.1 핵심 플러그인

들여가기 전에 알아둬야 할 용어!

message : filebeat 가 수집해서 logstash 로 넘어오는 데이터는 줄 단위로 읽어오는 데 이 줄단위의 데이터를 message 라고 한다

date

날짜 데이터 전용 플러그인

[예시]

```
date {
  match => [ "date", "yyyyMMdd" ]
  timezone => "Asia/Seoul"
  locale => "ko"
  target => "convert_date"
}
```

date 플러그인 옵션

locale 날짜를 한국어로 표현해주는 옵션
월의 형식을 mmm 으로 설정할 경우 월의 전체 이름이 나오는데 en 으로 하면 april 이 ko 로 하면 4 월이 나온다.

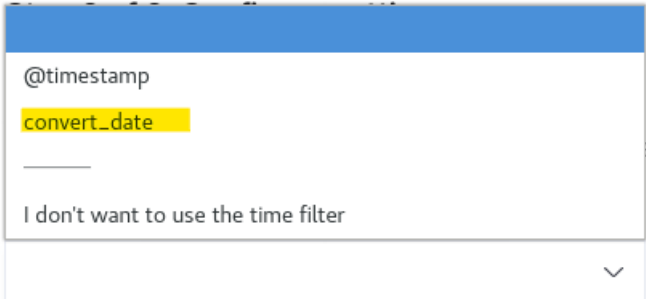
[예시]

```
date {
  locale => "ko"
}
```

match 날짜 데이터의 형식을 정해준다
match => ["날짜데이터필드명", "날짜형식"] 으로 작성한다

[예시]

```
date {
```

	<pre>match => ["date", "yyyyMMdd"] }</pre>
timezone	<p>시간대를 설정한다</p> <p>[예시]</p> <pre>date { timezone => "Asia/Seoul" }</pre>
target	<p>Index 를 설정할 때 드롭다운 메뉴에서 보일 이름을 설정한다 기본 값은 @timestamp</p> <p>[예시]</p> <pre>date { target => "convert_date" }</pre> <p>이 경우 kibana 에서 index 를 설정할 때 보이는 창</p> 
add_tag	2.3.3 공통옵션 참고
add_field	2.3.3 공통옵션 참고
id	2.3.3 공통옵션 참고
remove_field	2.3.3 공통옵션 참고
remove_tag	2.3.3 공통옵션 참고

더 자세한 옵션은 공식 문서를 참조하자(영어)

<https://www.elastic.co/guide/en/logstash/current/plugins-filters-date.html>

mutate

데이터를 변형시키는 플러그인 (예: 대문자 > 소문자 등)

[예시]

```
mutate {  
  lowercase => "message"  
}
```

이 필터를 통하면 모든 message 가 소문자가 된다

mutate 옵션

copy	특정 필드의 값을 다른 필드에 복사한다 <pre>mutate { copy => {"복사될_필드명" => "복사해서_넣을_필드명"} }</pre>
-------------	---

gsub	정규식을 사용해서 문자열을 교체하는 옵션
-------------	------------------------

[예시] - /를 _로 바꾼다

```
mutate {  
  gsub => [  
    "필드명", "/", "_",  
  ]  
}
```

[예시 2] - ₩, ?, #, -를 .으로 바꾼다

```
mutate {  
  gsub => [  
    "필드명", "[₩₩?#-]", "."  
  ]  
}
```

join	배열 자료형(리스트 등)이 들어있는 필드의 데이터를 join 한다
-------------	--------------------------------------

```
mutate {  
  join => {"필드명" => "구분자"}  
}
```

[예시]

날씨

	<div>['2018', '11', '2']</div> <pre>mutate { join => { "날씨" => "-" } }</pre> <p>결과값 2018-11-2</p>
lowercase	<p>데이터를 모두 소문자로 바꾼다</p> <pre>mutate { lowercase => ["필드명"] }</pre>
uppercase	<p>데이터를 모두 대문자로 바꾼다</p> <pre>mutate { uppercase => ["필드명"] }</pre>
capitalize	<p>필드 중 문장의 첫글자만 대문자로 하고 나머지는 소문자로 바꾼다.</p> <pre>mutate { capitalize => ["필드명"] }</pre>
merge	<p>두 필드의 값을 합친다 배열 + 스트링 => 가능 스트링 + 스트링 => 배열로 반환 배열 + 해시(키:값) => 불가능</p> <pre>mutate { merge => {"필드명 1" => "필드명 2"} }</pre>
coerce	<p>기본값을 설정한다 (만약 이 필드값이 비어서 들어오면 이 값을 대신 사용)</p> <pre>mutate { coerce => {"필드명" => "기본값"} }</pre>

rename	<p>필드명을 바꾼다</p> <pre>mutate { rename => {"기존필드명" => "변경할필드명"} }</pre>
split	<p>구분자를 기준으로 데이터를 자른다</p> <p>[예시]</p> <pre>mutate { split => ["message", ","] }</pre> <p># message 를 쉼표(.)를 기준으로 자른다 즉 "서울,부산,대구,대전" 이라고 되어있는 경우 [서울, 부산, 대구, 대전]으로 나뉜다 이 경우 [message][0] 이런식으로 인덱스를 써서 '서울'이라는 값을 호출할 수 있다</p>
update	<p>필드의 값을 지정값으로 바꾼다 (* update 와 replace 의 차이점은 만약 필드가 없으면 update 는 작업을 수행하지 않고 replace 는 새로운 필드를 생성한다)</p> <pre>mutate { rename => {"필드명" => "변경값"} }</pre>
replace	<p>필드의 값을 지정값으로 바꾼다 (* update 와 replace 의 차이점은 만약 필드가 없으면 update 는 작업을 수행하지 않고 replace 는 새로운 필드를 생성한다)</p> <pre>mutate { rename => {"필드명" => "변경값"} }</pre>
convert	2.3.3 공통옵션 참고
add_tag	2.3.3 공통옵션 참고
add_field	2.3.3 공통옵션 참고
id	2.3.3 공통옵션 참고
remove_field	2.3.3 공통옵션 참고
remove_tag	2.3.3 공통옵션 참고

[참고]

mutate 플러그인은 위에서부터 순차적으로 수행되는 것이 아니라 특정 순서에 따라 수행된다. 만약 순서를 바꾸고 싶다면 mutate 자체를 따로 선언해줘야 한다.

[예시] – split 다음에 rename 을 하고 싶을 때

```
mutate {  
  split => { "hostname" => "." }  
  add_field => { "shortHostname" => "%{[hostname][0]}" }  
}  
mutate {  
  rename => ["shortHostname", "hostname" ]  
}
```

mutate 수행 순서

- coerce
- rename
- update
- replace
- convert
- gsub
- uppercase
- capitalize
- lowercase
- strip
- remove
- split
- join
- merge
- copy

자세한 옵션을 참고 할 수 있는 공식문서(영어)

<https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html>

split

긴 데이터를 자르는 플러그인

주로 짧은 데이터(한줄단위)를 자를 때는 mutate 를 사용하고

만약 데이터단위가 여러줄로 이뤄져있을 경우 split 플러그인을 사용한다

```
split {  
  field => "자르려는 데이터가 있는 필드명"  
  target => "자른 데이터가 들어갈 필드명"  
  terminate => "구분자"  
}
```

[예시]

```
split {  
  field => "date"  
  target => "date_seperated"  
  terminate => "-"  
}
```

"2020-10-22" 데이터가 들어있는 "date" 필드를 이 split 필터에 통과시키면 "-"를 구분자로 데이터가 쪼개져서 "date_seperated" 필드에 [2020, 10, 22] 데이터가 들어가게 된다.

split 옵션	
field	목표가 되는 필드명
target	결과값이 들어갈 필드명
terminate	구분자
add_tag	2.3.3 공통옵션 참고
add_field	2.3.3 공통옵션 참고
id	2.3.3 공통옵션 참고
remove_field	2.3.3 공통옵션 참고
remove_tag	2.3.3 공통옵션 참고

자세한 설명을 확인할 수 있는 공식문서 링크!(영어)

<https://www.elastic.co/guide/en/logstash/current/plugins-filters-split.html>

grok

정규 표현식을 사용해서 비정형 데이터(예: html 같은 것)를 정형 데이터로 변환하고 분류하는 플러그인

grok 옵션	
match	<p>정규표현식으로 원하는 데이터를 추출해서 정형 데이터로 변환한다</p> <p>[예시]</p> <p>다음 문장에서 IP 만 추출하고 싶다. '당신의 IP 는 127.0.0.1 입니다'</p> <p>이 경우는 아래와 같이 match 옵션을 설정해서 추출할 수 있다.</p> <pre>filter { grok { match => { "message" => ".+(?<client>Wd+W.Wd+W.Wd+W.Wd).+" } } }</pre> <p>여기서 사용된 형식은 형식은 (?<필드명>정규표현식) 이다 %{정규표현식:필드명}으로도 작성할 수 있다.</p>
overwrite	<p>지정한 필드에 덮어쓰기를 한다 아래와 같이 설정하면 message 를 변환한 뒤 message 에 덮어쓰기해서 정형 데이터만 남긴다</p> <pre>grok { match => { "message" => "%{SYSLOGBASE} %{DATA:message}" } overwrite => ["message"] }</pre>
target	결과값을 입력할 필드를 지정해준다
add_tag	2.3.3 공통옵션 참고
add_field	2.3.3 공통옵션 참고

id	2.3.3 공통옵션 참고
remove_field	2.3.3 공통옵션 참고
remove_tag	2.3.3 공통옵션 참고

자세한 옵션은 공식 문서를 참조하자(영어)

<https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html>

dissect

정규 표현식을 사용하지 않고 필드를 분류하는 플러그인

정규식을 쓰지 않는건 좋지만 grok 이 좀더 정교하고 자세한 분류가 가능하다

그래서 dissect 를 사용해서 크게 분류해준 뒤에 grok 을 중첩 사용해서 상세하게 분류하는 방식도 사용한다

[예제]

아래와 같은 문장이 있다고 하자

"John Smith,BigOaks,WoodLane,Hambledown,Canterbury,CB34RY"

```
dissect {
  mapping => {
    "message" => %{name},%{addr1},%{addr2},%{addr3},%{city},%{zip}
  }
}
```

문장을 위의 dissect 필터로 아래와 같이 맵핑할 수 있다.

```
"name": "Jane Doe",
"addr1": "4321 Fifth Avenue",
"addr2": "BigOaks",
"addr3": " WoodLane ",
"city": "New York"
"zip": "87432"
```

dissect 옵션

convert_datatype dissect 로 분류한 데이터의 형식을 정해줄 수 있다

	<pre> [예시] dissect { convert_datatype => { "name" => "text" "age" => "int" } } </pre>
mapping	Dissect 로 분류한 데이터를 맵핑한다 <pre> [예시] dissect { mapping => { "message" => %{name},%{addr1},%{addr2},%{addr3},%{city},%{zip} } } </pre>
add_tag	2.3.3 공통옵션 참고
add_field	2.3.3 공통옵션 참고
id	2.3.3 공통옵션 참고
remove_field	2.3.3 공통옵션 참고
remove_tag	2.3.3 공통옵션 참고

더 자세한 옵션은 공식 문서를 참조하자(영어)

<https://www.elastic.co/guide/en/logstash/current/plugins-filters-dissect.html>

translate

말그대로 문자열을 번역해주는 플러그인

yaml 이나 json, csv 처럼 형태가 정해져 있는 데이터에 사용한다

[예시]
예를들어 아래와 같은 파일이 있다고 가정해보자

품명	종류
----	----

딸기우유	우유
초코우유	우유
비타 500	음료

여기서 번역이라는 말은 원래의 뜻인 한국어를 영어로, 이런식으로 번역을 한다는 게 아니라 a 를 입력받으면 b 라는 값이 나오도록 정해주고 싶다는 말이다(마치 기존의 번역에서 사과를 넣으면 apple 이 나오는 것 처럼)

그럼 이제 종류에 따라 유통기한을 정해주고 싶다고 해보자 그러니까 종류라는 말을 넣으면 유통기한이 나오게 하고싶다

```
translate {
  field => "[종류]" # 기준이 되는 필드명
  destination => "[유통기한]" # 번역한 데이터를 입력할 필드명
  dictionary => [
    "우유" => "3 일"
    "음료" => "4 달"
  ]
}
```

이렇게 하면 다음처럼 번역이 되어서 데이터가 들어갈 것이다

종류가 우유면 유통기한은 3 일로 번역

종류가 음료면 유통기한은 4 달로 번역..

품명	종류	유통기한
딸기우유	우유	3 일
초코우유	우유	3 일
비타 500	음료	4 달

만약 destination 을 기존이랑 같게 하면 데이터 덮어쓰기도 가능하다
대신 이경우 override 옵션을 true 로 해줘야 한다

```
translate {
  field => "[유통기한]" # 기준이 되는 필드명
  destination => "[유통기한]" # 번역한 데이터를 입력할 필드명
```

자세한 설명은 아래 공식 문서에서 확인! (영어)

<https://www.elastic.co/guide/en/logstash/current/plugins-filters-translate.html>

drop

데이터를 버리는 플러그인, 주로 if 문과 함께 사용한다

[예시]

```
Filter {  
  if "test" in [message] {  
    drop {  
    }  
  }  
}
```

예시는 if 문을 사용하여 "test"라고 적힌 메시지는 버린다고 옵션을 설정한 것

Drop 공식문서 : <https://www.elastic.co/guide/en/logstash/current/plugins-filters-drop.html>

CSV

csv 전용 플러그인

csv 옵션

autodetect_column_names true 로 설정하면 첫번째 열을 칼럼명으로 설정한다
기본값은 false

[예시]

```
csv {  
  autodetect_column_names => true  
}
```

Autogenerate_column_names true 로 설정하면 칼럼명을 명시해주지 않았을 경우
자동으로 칼럼명을 생성한다 기본값은 true. 만약 이

	설정도 false 고 칼럼명을 명시해주지도 않으면 파싱 자체가 안된다
columns	칼럼명을 설정해준다 만약 칼럼의 일부만 설정해줬을 경우(총 칼럼은 3 개인데 칼럼명은 2 개만 설정했을 경우) 나머지는 자동으로 붙는다 [예시] csv { columns => ["title", " column1", " column2", "description"] }
separator	구분자를 명시해준다. 기본 값은 ',' [예시] csv { separator => "," }
skip_empty_rows	빈 열을 스킵할지 설정한다. 기본값은 false
skip_header	헤더를 스킵할지 설정한다. 기본값은 false
convert	2.3.3 공통옵션 참고
add_tag	2.3.3 공통옵션 참고
add_field	2.3.3 공통옵션 참고
id	2.3.3 공통옵션 참고
remove_field	2.3.3 공통옵션 참고
remove_tag	2.3.3 공통옵션 참고

더 자세한 옵션을 보고 싶으면 공식 문서를 참고하자

<https://www.elastic.co/guide/en/logstash/current/plugins-filters-csv.htm>

2.2.2 기타플러그인

기타 플러그인	
http	http 전용 플러그인 참고 – 자세한 옵션 (영어) :

	https://www.elastic.co/guide/en/logstash/current/plugins-filters-http.html
Jdbc_static	jdbc 전용 플러그인 참고 – 자세한 옵션 (영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-jdbc_static.html
jdbc_streaming	jdbc 전용 플러그인 참고 – 자세한 옵션 (영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-jdbc_streaming.html
xml	xml 전용 플러그인 참고 – 자세한 옵션 (영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-xml.html
elasticsearch	기존 elasticsearch 에 등록되어있던 index(DB)의 정보 사용 참고 – 자세한 옵션 (영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-elasticsearch.html
geoip	Ip 의 지역정보 이용 참고 – 자세한 옵션 (영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-geoip.html
kv	키와 값으로 이뤄진 데이터를 처리하는 플러그인 참고(한글) : http://kangmyounghun.blogspot.com/2018/09/logstash-kv.html 참고 – 자세한 옵션 (영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-kv.html
elapsed	경과 시간을 측정하고 싶을 때 사용하는 플러그인 참고(한글) : https://kangmyounghun.blogspot.com/2020/02/logstash-elapsed.html 참고 – 자세한 옵션 (영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-elapsed.html
urldecode	decode 하는 플러그인이라는 데 어떻게 쓰는지 모르겠다.. 일단 알아두면 좋을 것 같아서 적어놓고 좀 더 공부해봄 참고 – 자세한 옵션 (영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-urldecode.html

clone	Index 를 복제해서 두개의 index 를 만들 때 사용 참고 – 자세한 옵션(영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-clone.html
alter	mutate 플러그인으로 해결하지 못하는 문자열 치환 작업 수행 플러그인 참고 – 자세한 옵션(영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-alter.html
sleep	작업 사이에 간격을 주고 싶을 때 사용하는 플러그인 참고 – 자세한 옵션(영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-sleep.html
throttle	데이터 유입량을 조절하는 플러그인 참고 – 자세한 옵션(영어) : https://www.elastic.co/guide/en/logstash/current/plugins-filters-throttle.html
truncate	들어오는 데이터의 크기(글자수)를 제한하는 플러그인 참고 – 자세한 옵션(영어) https://www.elastic.co/guide/en/logstash/current/plugins-filters-truncate.html

2.2.3 공통 옵션

공통옵션	
add_field	<p>“필드 => 값” 해서 필드를 생성하고 값을 할당함</p> <p>[예시]</p> <pre>mutate { split => ["message", ","] add_field => { "date" => "%{[message][0]}" "region" => "%{[message][1]}" "confirmed" => "%{[message][2]}" "death" => "%{[message][3]}" "released" => "%{[message][4]}" } }</pre>

	<pre>} 위의 태그를 해설해보자면 mutate 플러그인에서 split 옵션을 사용해서 , 를 기준으로 message 를 자르고 각각의 date, region, confirmed, death, released 필드 생성 후 자른 값을 할당하는 필터링이다.</pre>
convert	<p>필드의 형을 정해주는 옵션(맵핑)</p> <p>[예시]</p> <pre>convert => { "confirmed" => "integer" "death" => "integer" "released" => "integer" }</pre>
add_tag	<p>필터링을 끝낸 데이터에 tag 를 붙인다. 주로 if 문과 함께 사용한다.</p> <p>[예시]</p> <pre>csv{ add_tag => ['csv'] }</pre> <p>이러면 csv 필터를 통과한 데이터에는 csv 라는 tag 가 붙게된다 이렇게 붙인 태그는 아래처럼 if 문과 결합해서 사용한다</p> <pre>if "csv" in [tags] grok { csv 필터를 통과한 데이터에 추가로 필터링할 내용 작성 }</pre>
id	<p>id 를 부여한다</p> <p>별도로 부여해주지 않아도 자동으로 생기지만 데이터를 분류할 때 유용하게 쓰이기 때문에 직접 지정해주는 편이 좋다.</p> <p>예를 들어 csv 필터를 두개 이상 사용하고 각 필터별로 데이터를 구분해야 할 필요가 있을 때 id 를 정해주면 로그스태시 과정에서 구분해서 처리할 수 있다.</p>

	<p>* id 와 tag 의 차이점: tag 는 중복값을 허용하지만(csv 를 여기저기 붙일 수 있음) id 는 중복값을 허용하지 않는다. 한 id 는 한 필터만 사용할 수 있다</p>
remove_field	<p>불필요한 field 를 제거한다</p> <p>[예제]</p> <pre>mutate { remove_field => ["ecs", "host", "@version", "agent", "log", "tags", "input", "message"] }</pre>
remove_tag	<p>불필요한 태그를 제거한다</p> <p>예를 들어 csv 필터를 통과한 데이터만 grok 로 처리하고 싶어서 tag 를 사용했다면 grok 필터 처리가 끝나면 더 이상 tag 가 필요 없다. 이 경우 grok 필터의 끝에서 이 옵션을 사용해서 tag 를 제거한다.</p> <pre>grok { remove_tag => ["csv"] }</pre>

2.3 output

내보내는 플러그인도 email, file, kafka, csv 등 다양한 플러그인이 있지만 우리는 elasticsearch 로 내보낼 거니까 elasticsearch 에 대해서만 알아본다.

다른 플러그인도 알고 싶다면 아래 링크를 참조하자(영어).

<https://www.elastic.co/guide/en/logstash/current/output-plugins.html>

[output 작성 예시]

```
output {
  stdout {
    codec => rubydebug
  }
  elasticsearch {
    hosts => "localhost:9200"
    manage_template => false
    index => "%[@metadata][beat]} - %[@metadata][version]} - % +YYYY.MM.dd} "
  }
}
```

Elasticsearch 대표 옵션	
host => "localhost:9200"	elastic 호스트 명시
manage_template => false	인덱스를 logstash 로 붙이는 것 기본 값이 true 이기 때문에 false 로 해줘야 내 마음대로 index 를 설정할 수 있다
index => ~	인덱스 값 설정 위에 설정된 값은 %[@metadata][beat] : 지금 사용하고 있는 beats 명, 이경우 filebeat %[@metadata][version] : 지금 사용하고 있는 beats 의 버전 %{+YYYY.MM.dd} : 시간 다른 값으로 설정할 수도 있다
id => ~	Elasticsearch 의 id 설정 필수 조건은 아니지만 편의를 위해 설정하는 것을 추천

옵션은 이 밖에도 다양하므로 더 자세한 옵션을 보고싶다면 아래 링크를 참조하자.

<https://www.elastic.co/guide/en/logstash/current/plugins-outputs-elasticsearch.html>

[참고]

stdout {}

출력값을 정하는 module 이라 생략 가능하다

(참고: rubydebug 는 예쁘게 출력하기 위해 ruby 의 print 옵션을 사용한다는 뜻