Logstash

오늘의 학습내용

- Logstash 기본 개념
- Logstash 구성요소 이해
- Config file
- 데이터 파이프라인 설정

Logstash 기본 개념

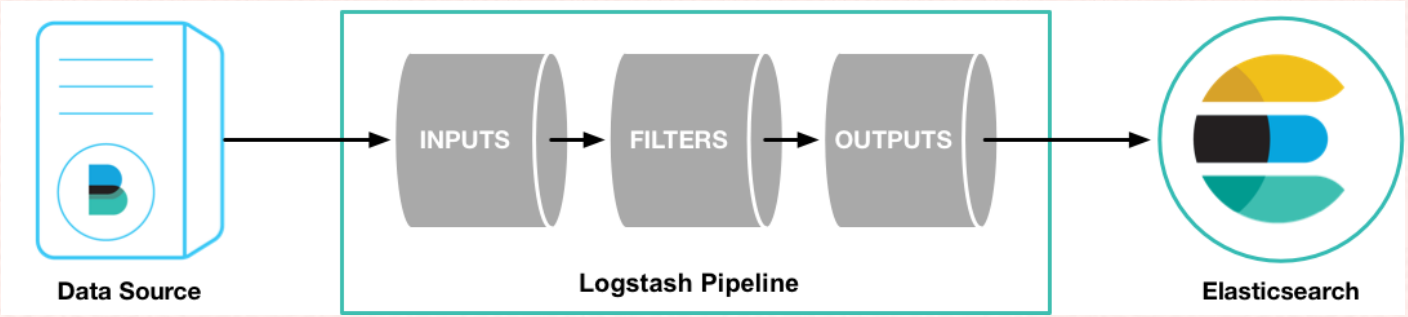
Logstash 기본 개념

- 실시간 파이프라인 기능을 가진 오픈소스 데이터 수집 엔진(DataFlow Engine).
- 광범위한 플러그인이 구비되어 다양한 아키텍처에서 손쉽게 데이터를 수집, 처리, 전달 가능.
- 프로세싱은 하나 이상의 파이프라인으로 구성.
- 각 파이프라인에서 하나 이상의 입력 플러그인이 내부 대기열에 배치된 데이터를 수신하거나 수집.
- 기본적으로 작고, 메모리에 보관되지만 안정성과 복원력을 향상시키도록 디스크에서 더 크고 영구적으로 구성 가능.
- 다양한 데이터를 통합하고 정규화해서 필요한 부분의 데이터를 Filtering 및 Streaming.

Logstash 기본 개념

Logstash 기본 개념

- Logstash는 플러그인 기반의 데이터 수집 및 처리 엔진
- 서로 다른 소스의 데이터를 탄력적으로 통합하고 사용자가 선택한 목적지로 데이터를 정규화
- 포맷이나 스키마에 관계 없이, 모든 데이터를 수집하고 강화 및 통합하기 위한 Elastic Stack의 중앙 데이터 플로우 엔진



Logstash 구성요소 이해

Pipeline

- Logstash 파이프라인은 하나 이상의 구성 파일을 기반으로 생성
- 단일 구성 파일을 사용한 단일 파이프라인 : Logstash가 -f 명령 줄 매개변수를 통해 지정하는 단일 구성 파일을 기반으로 단일 파이프라인 구성
- 여러 구성 파일을 사용한 단일 파이프라인 : Logstash는 특정 디렉터리의 모든 파일을 구성 파일로 사용하도록 구성 가능, logstash.yml 파일을 통해 설정
- 여러 파이프라인 사용 : Logstash 내에서 여러 파이프라인을 사용하려면 Logstash와 함께 제공되는 pipelines.yml 파일 편집 필요
- 데이터 처리를 위한 logstash 설정. 3가지 processor (Input, Filter, Output)로 구성되어있으며 순차적으로 실행됨

Logstash 구성요소 이해

Plugin

- 각 단계(Input, Filter, Output)는 여러 플러그인 조합으로 구성하며, Input -> Filter -> Output이 순차적으로 실행됨
- 예를 들어, Input은 File Plugin으로, Filter는 JSON plugin 그리고 Output에는 Elasticsearch plugin으로 구성한다면, 파일로부터 데이터를 가져와서 Elasticsearch에 저장 가능한 JSON형태로 변환한 후 Elasticsearch로 Indexing하게 됨
- 각 단계를 조합하는 Plugin의 사용법은 유사
- 원하는 plugin도 만들어서 사용 가능

Logstash 구성요소 이해

Input

- 데이터가 유입되는 근원지. Input Plugins을 사용해 특정 소스의 이벤트(데이터)를 가져옴
- logs & Files, Metrics, Wire Data, Web Apps, Data Stores, Data Streams 등의 수집을 지원
- Beats와 결합해서 쓰는 것도 가능
- Input plugin 종류 : azure_event_hubs, beats, couchdb_changes, elasticsearch, file, ganglia, google_cloud_storage, graphite, heartbeat, http, http_poller, jdbc, kafka, log4j, pipe, rabbitmq, redis, rss, s3, snmp, snmptrap, sqlite, stdin, syslog, tcp, udp, unix, websocket, xmpp 등

Logstash 구성요소 이해

Filter

- log parsing, 데이터 확장, 태그 추가 등 의 데이터 변형 작업을 함. Filtering 과정은 필수가 아님.
- 자주 쓰이는 Filter plugin

  - grok : 문자열 데이터에서 필요한 데이터를 정규표현식으로 처리
  - mutate : 데이터 필드 단위로 변형, 이벤트 필드에서 일반적인 변환을 수행
  - date : String을 Data타입으로 변환(날짜)
  - drop : 이벤트를 삭제
  - clone : 이벤트의 복사본을 생성
  - json : input의 JSON 데이터 구조를 유지
  - geoip : ip 주소의 지리적 위치에 대한 정보 추가

Logstash 구성요소 이해

Output

❖ 데이터를 전송할 목적지. 가공된 데이터를 저장소에 적재할 경우 Output plugin으로 결정

- Elasticsearch : 이벤트 데이터를 elasticsearch에 전송
- file : 디스크 파일에 쓰기
- graphite : graphite에 전송 (graphite : 메트릭을 저장하고 그래프로 작성하는데 사용되는 오픈 소스 도구)
- statsd : 카운터 및 타이머와 같은 통계르 수신하고 UDP를 통해 전송되며, 하나 이상의 플러그 가능한 백엔드 서비스에 집계를 보내는 서비스

❖ Output plugin : boundary, csv, datadog, datadog_metrics, elastic_app_search, elasticsearch, email, file, ganglia, google_cloud_storage, graphite, graphtastic, http, influxdb, kafka, metriccatcher, mongodb, Nagios, opentsdb, pipe, rabbitmq, redis, redmine, riak, s3, 눈, solr_http, statsd, stdout, syslog, tcp, u에, webhdfs, websocket, xmpp, zabbix

Logstash 구성요소 이해

Codec

❖ 데이터 인코딩 & 디코딩. 메세지(들어오는 데이터)를 쉽게 구분하고 전송할 수 있도록 하는 input과 output 단계에서 사용되는 stream filter

- json : JSON 형식의 데이터를 인코딩하거나 디코딩, multiline : 여러 줄 텍스트 이벤트를 단일 이벤트로 병합

❖ Codec plugin : avro, cef, collectd, csv, dots, edn, edn_lines, es_bulk, fluent, graphite, gzip_lines, jdots, java_line, java_plain, json, json_lines, line, msgpack, multiline, netflow, nmap, plain, protobuf, rubydebug

Logstash 구성요소 이해

Pipeline

- Logstash 파이프라인은 하나 이상의 구성 파일을 기반으로 생성
- 단일 구성 파일을 사용한 단일 파이프라인 : Logstash가 -f 명령줄 매개변수를 통해 지정하는 단일 구성 파일을 기반으로 단일 파이프라인 구성
- 여러 구성 파일을 사용한 단일 파이프라인 : Logstash는 특정 디렉터리의 모든 파일을 구성 파일로 사용하도록 구성 가능, logstash.yml 파일을 통해 설정
- 여러 파이프라인 사용 : Logstash 내에서 여러 파이프라인을 사용하려면 Logstash와 함께 제공되는 pipelines.yml 파일을 편집 필요

Config file

Logstash Configuration file

- Logstash는 두가지 타입의 설정 파일
- 하나는 파이프라인을 정의하는 설정 파일
- 다른 하나는 Logstash 시작 및 실행과 관련된 옵션을 지정하는 설정 파일
- logstash.yml : Logstash 실행과 관련된 설정 파일
- pipeline.yml : 단일 Logstash 인스턴스에서 여러 개의 파이프라인을 실행할 때 사용
- jvm.option : JVM 설정. 힙 사이즈를 조절 및 GC 관련 옵션 등을 설정
- log4j2.properties : log4j 관련 설정 파일

Config file

Multiple Pipelines

❖ **pipeline.yml을 통해 단일 Logstash 인스턴스로 여러개의 파이프라인을 실행**

```
- pipeline.id: my-pipeline_1
  path.config: "/etc/path/to/p1.config"
  pipeline.workers: 3
- pipeline.id: my-other-pipeline
  path.config: "/etc/different/path/p2.cfg"
  queue.type: persisted
```

Config file

구성 파일(config/test.conf)

❖ **모든 Logstash 구성에는 적어도 하나의 입력 플러그인과 하나의 출력 플러그인 필요**

❖ **필터는 선택 사항**

```
input {
  file {
    path => ["/home/logstash/testdata.log"]
    sincedb_path => "/dev/null‟ ->각 입력 파일 내의 데이터를 추적
    start_position => "beginning‟ -> 새 파일이 발견될 때마다 처음부터 파일을 읽도록 지시
  }
}
filter {
}
output {
  stdout {  -> 콘솔 출력
     codec => rubydebug -> 디버깅
  }
}
```

데이터 파이프라인 설정

CSV File 분석

❖ csv 파일을 읽어서 Elasticsearch로 전송하는 실습

```
input {
  file {
    path => "/{your-path}/stock-data.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}
```

데이터 파이프라인 설정

CSV File 분석

❖ **csv 파일을 읽어서 Elasticsearch로 전송하는 실습**

```
filter {
  csv {
    separator => ","
    columns => ["Date","Open","High","Low","Close","Volume","Adj Close"]
  }
  mutate {
     convert => {
       "Open" => "float"
       "High" => "float"
       "Low" => "float"
       "Close" => "float"
       "Volume" => "float"
       "Adj Close" => "float"
  } ….
```

데이터 파이프라인 설정

CSV File 분석

❖ **csv 파일을 읽어서 Elasticsearch로 전송하는 실습**

```
output {
  elasticsearch {
    hosts => "127.0.0.1:9200"
    index => "stock-%{+YYYY.MM.dd}"
  }

  stdout {
  }
}
```