

EFK를 활용한 로그 수집과 분석 (3일)

(Elasticsearch + Fluentd + Kibana)

2022 – 09

커리큘럼 상세

[Cloud Computing] EFK, Fluentd 로그 수집과 분석 - 2일차

교육 목표	<ul style="list-style-type: none"> Fluentd의 장점을 알아보고, 파이프라인 작성에 대해 배울 수 있습니다. Fluentd plugin 활용 방법을 배울 수 있습니다. 서버, DB, Web 서버 등 다양한 로그를 수집하여 데이터를 시각화하고 분석 할 수 있습니다. 		
교육 기간	1일, 7시간	선수 지식 (수강 요건)	<ul style="list-style-type: none"> Python 프로그램 기초 지식 EFK 설치 및 기본 지식 (사전학습 자료) https://www.youtube.com/watch?v=Gp0-7oVOtPw Web 서버 및 Linux 기본 지식 (사전학습 자료) https://www.youtube.com/watch?v=dsUyFss2Sh4
교육 내용	주제	시간	학습 내용
	Fluentd 의 구조와 기능	1H	<ul style="list-style-type: none"> Fluentd 개요 및 로그 수집 방법(data source & output)
	Fluentd를 통한 외부 로그 및 메트릭 수집 및 분석	1H	<ul style="list-style-type: none"> Fluentd 파이프라인 구성 (Fluentd Configuration)
		1H	<ul style="list-style-type: none"> Fluent Input/Output plugin의 종류 및 데이터 수집과 분석
	서버, DB, Web 서버 등 다양한 로그 수집 Pipeline 구성 및 실습	1H	<ul style="list-style-type: none"> [실습1] Fluentd(td-agent) 설치 및 td-agent.conf 파일 설정
		1H	<ul style="list-style-type: none"> [실습2] Nginx log 수집 및 분석
		1H	<ul style="list-style-type: none"> [실습3] 시스템 로그(rsyslogd) 수집 및 분석
		1H	<ul style="list-style-type: none"> [실습4] MySQL DB log 연동
실습 환경	<ul style="list-style-type: none"> Windows 환경(WSL2 – Ubuntu 20.04), Docker Desktop, SSD 60G 이상, 16G 이상 메모리 		

2 일차 일차 : EFK, Fluentd

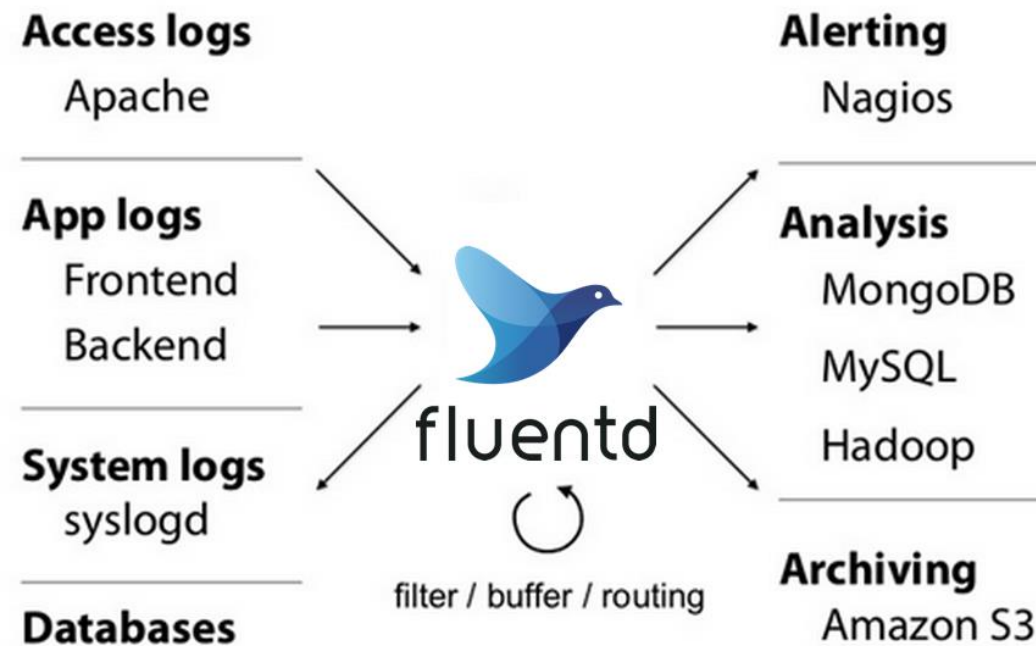
- Fluentd를 활용한 Log 수집 분석
- Fluentd Input/Output plugin의 종류 및 데이터 수집과 분석
- 다양한 Log 수집 Pipeline 구축(실습)
 - 1) Ngix Log
 - 2) Syslogd
 - 3) Database와 Files log

교육 내용 및 일정 소개

교육 내용	주제	시간	학습 내용
	Fluentd 의 구조와 기능	1H	<ul style="list-style-type: none"> Fluentd 개요 및 로그 수집 방법(data source & output)
	Fluentd를 통한 외부 로그 및 메트릭 수집 및 분석	1H	<ul style="list-style-type: none"> Fluentd 파이프라인 구성 (Fluentd Configuration)
		1H	<ul style="list-style-type: none"> Fluent Input/Output plugin의 종류 및 데이터 수집과 분석
	서버, DB, Web 서버 등 다양한 로그 수집 Pipeline 구성 및 실습	1H	<ul style="list-style-type: none"> [실습] Fluentd(td-agent) 설치 및 td-agent.conf 파일 설정
		1H	<ul style="list-style-type: none"> [실습] Nginx log 수집 및 분석
		1H	<ul style="list-style-type: none"> [실습] 시스템 로그(rsyslogd) 수집 및 분석
		1H	<ul style="list-style-type: none"> [실습] MySQL DB log 연동

Fluentd: Open-Source Log 수집기

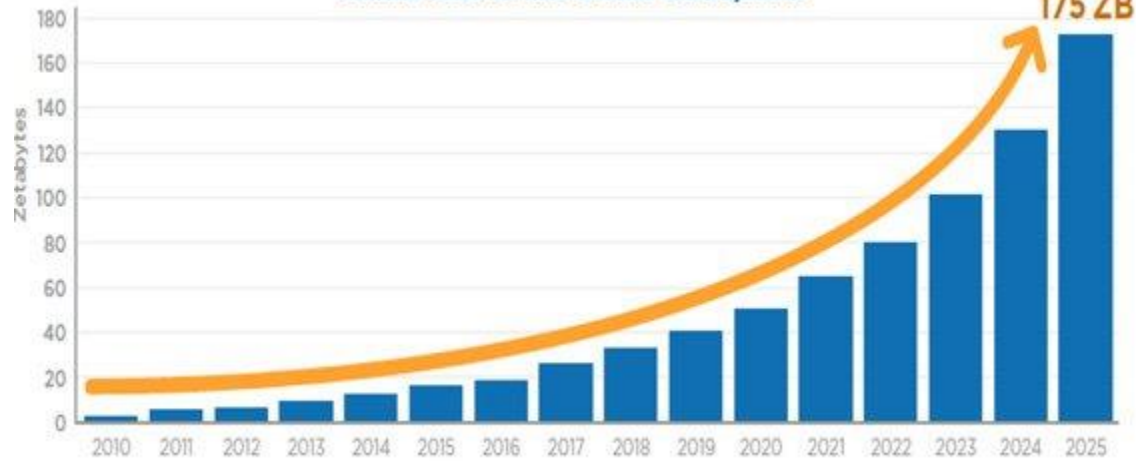
- RDBMS, NoSQL, IaaS, SaaS, Hadoop 수집된 로그를 거의 실시간으로 수집기 Fluentd로 안전하게 전송



<https://github.com/fluent/fluentd>

Digital Data의 증가량

Annual Size of the Global Datasphere



Source: Data Age 2025, sponsored by Seagate with data from IDC Global DataSphere, Nov 2018



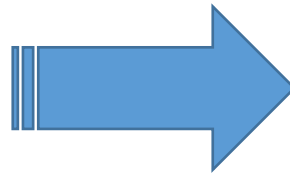
출처: IDC, 2025년 전 세계 데이터 용량의 예상 규모

현금(Money)의 추상화(Abstraction) 사례

Data



현금(Money)



인터넷 뱅킹

Fetures: 교환가치 + 은행의 기능



신용카드

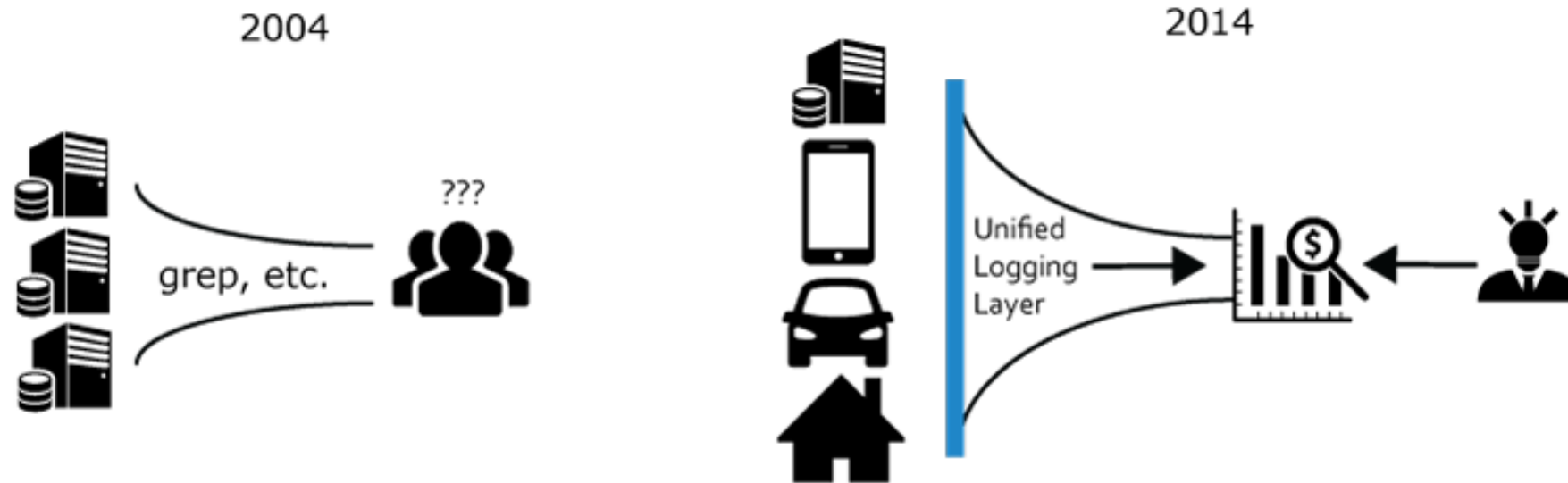
Fetures: 교환가치 + 신용



티머니

Fetures: 교환가치 +
버스 지하철 갈아탈 수 있다

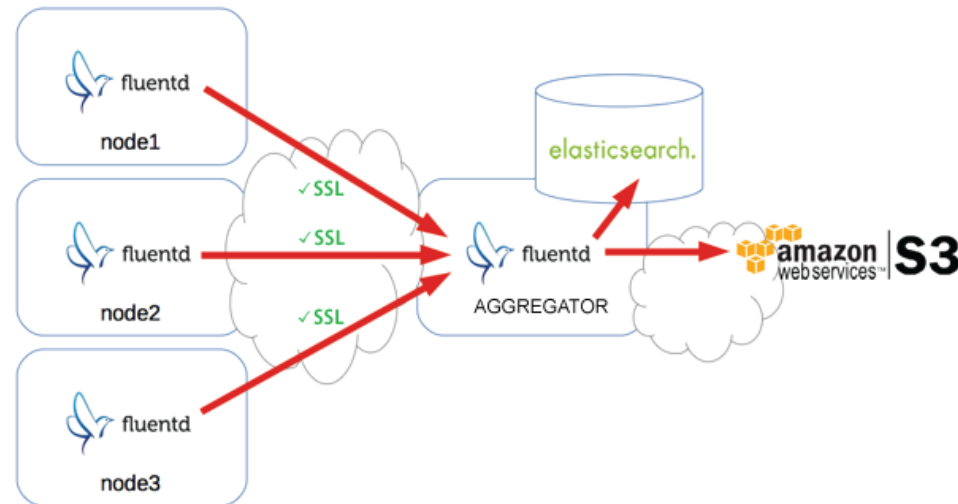
Logs from Machines



Log Data is for Machines

Fluentd를 활용한 Log 수집 분석 사례 실습

- 웹 서버에서 Apache httpd 로그 및 syslog를 수집
- 수집된 로그를 거의 실시간으로 수집기 Fluentd로 안전하게 전송
- 수집된 로그를 Elasticsearch 및 Log 저장
- 실시간으로 Kibana로 데이터를 시각화



플러그인 활용

Fluentd의 코어 부분은 최소한의 기능만 갖추어 경량화하였고, 많은 기능이 플러그인으로 구현되어 있습니다. 그렇기 때문에 플러그인 개발을 위한 인터페이스가 잘 정리되어 있고 써드파티의 플러그인 개발도 활발합니다.

Fluentd의 코어에 기본적인 플러그인은 번들되어 있지만 td-agent의 패키지에는 엔터프라이즈급에서 자주 사용되는 사실상 표준이 된 플러그인도 번들되어 있습니다

<https://www.fluentd.org/plugins>

플러그인의 리스트는 위 URL에 정리되어 있습니다

플러그인의 설치(Install)

datacounter 라는 단위시간별로 집계하는 플러그인을 Fluentd v0.12에서 설치하는 사례

```
$ td-agent-gem install fluent-plugin-datacounter -version="8.5.0m
```

특정 버전을 설치하려고 할 때는 gem 커맨드와 같이 --version 또는 - 파라미터를 사용합니다. 생략하면 가장 최신의 버전을 설치하게 됩니다. 서버를 구축한 타이밍에 따라 플러그인의 버전이 달라지지 않도록 서버의 초기화 스크립트 안에서는 버전을 지정하도록 합니다.

플러그인의 Uninstall

플러그인을 삭제하려고 할 때는 아래와 같이 td-agent-gem uninstall 커맨드를 사용합니다.
mongoDB 플러그인을 패키지를 지정해서 삭제하는 예시는 아래와 같습니다.

```
$ td-agent-gem uninstall fluent-plugin-mongodb --version="1.0.0"
```

.

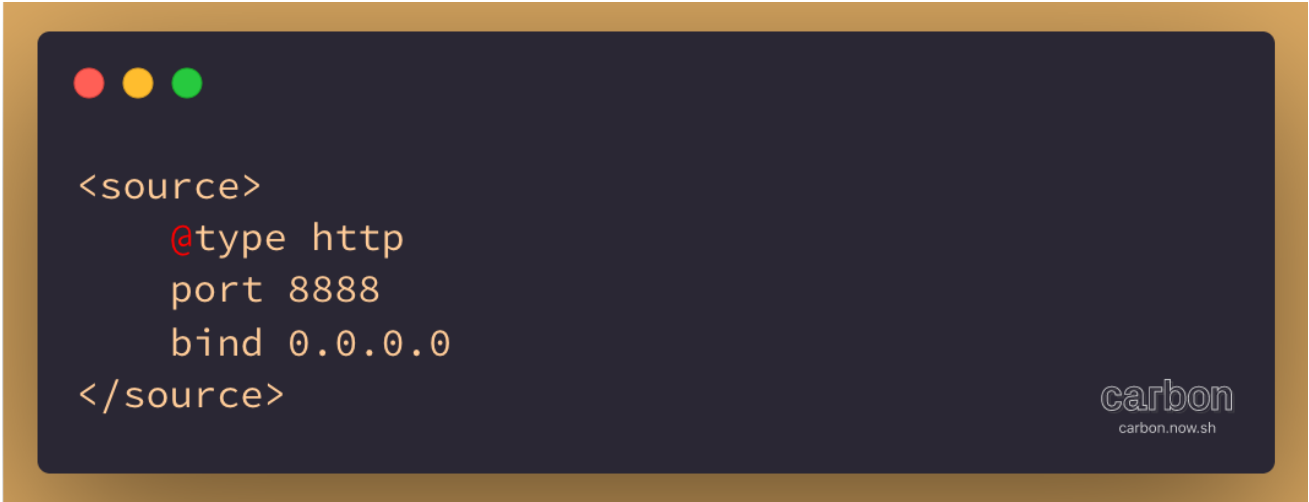
Fluentd 설정 커스터마이징(List of Directives)

1. <system> 디렉티브가 있으면 거기서 지정된 설정으로 코어 부분의 동작이 바뀐다.
2. @include 디렉티브가 있으면 로컬 파일이나 HTTP로 받은 설정을 가지고 Fluentd 프로세스를 작동한다.
3. <source> 디렉티브에서 지정한 플러그인을 경유해서 로그 수집을 시작한다.
4. 지정한 태그 패턴의 <label> 디렉티브 안에 있는 <filter>나 <match>의 내용을 처리한다.
5. 필요에 따라 중복해서 기술된 <filter> 디렉티브의 대상인 태그의 레코드를 가공한다.
6. 정한 태그 패턴의 <match> 디렉티브에서 태그의 변환과 외부로의 데이터 출력을 처리한다.

<https://docs.fluentd.org/configuration/config-file>

플러그인의 종류: Input Plugin

- Input Plugin : 데이터가 어디로부터 오는지 설정
- input 플러그인은 source 태그 안에 정의한다.
 - 예로 http source의 경우
 - HTTP 8888 포트에 바인딩되어, 해당 포트의 Server를 지속해서 리스닝하고 있음을 표현



```
<source>  
  @type http  
  port 8888  
  bind 0.0.0.0  
</source>
```

carbon
carbon.now.sh

Filter Plugin : 이벤트를 어떤 프로세스로 처리하는지

- 아래의 설정을 예로 들면 test.cycle이라는 태그를 가진 이벤트 중에서, action key에서 logout이라는 패턴을 제외시키고, 제외되지 않은 메시지만을 match 태그로 전달할 수 있도록 함
- 실제로 http에서 json={"action":"login", "user":2} 형태의 이벤트가 전달됨

```
<source>
  @type http
  port 8888
  bind 0.0.0.0
</source>

<filter test.cycle>
  @type grep
  <exclude>
    key action
    pattern ^logout$
  </exclude>
</filter>

<match test.cycle>
  @type stdout
</match>
```

Wildcards, Expansions and other tips

- `*` matches a single tag part.

For example, the pattern `a.*` matches `a.b`, but does not match `a` or `a.b.c`

- `**` matches zero or more tag parts.

For example, the pattern `a.**` matches `a`, `a.b` and `a.b.c`

- `{X,Y,Z}` matches `X`, `Y`, or `Z`, where `X`, `Y`, and `Z` are match patterns.

For example, the pattern `{a,b}` matches `a` and `b`, but does not match `c`

This can be used in combination with `*` or `**` patterns. Examples include `a.{b,c}.*` and `a.{b,c.**}`

- `./regular expression/` is for complex patterns

For example, the pattern `/(?!aW.).*/` matches non-`a`. started tags like `b.xxx`

This feature is supported since fluentd v1.11.2

Wildcards, Expansions and other tips

- `#{...}` evaluates the string inside brackets as a Ruby expression. (See Embedding Ruby Expressions section below).
- When multiple patterns are listed inside a single tag (delimited by one or more whitespaces), it matches any of the listed patterns.

For example: The patterns `<match a b>` match a and b. The patterns `<match a.** b.*>` match a, a.b, a.b.c (from the first pattern) and b.d (from the second pattern).

Output Plugin : 이벤트를 어디로 전달하는지 설정

- output 플러그인은 match 태그 안에 정의
- tag와 매치되는 이벤트들을 다른 시스템으로 전달할 때 주로 사용
- v1.0부터 Buffering과 Flushing에 대한 설정을 match 태그 내부에 buffer 태그로 따로 정의하도록 변경

Fluentd가 내부에서 처리하는 데이터의 특징

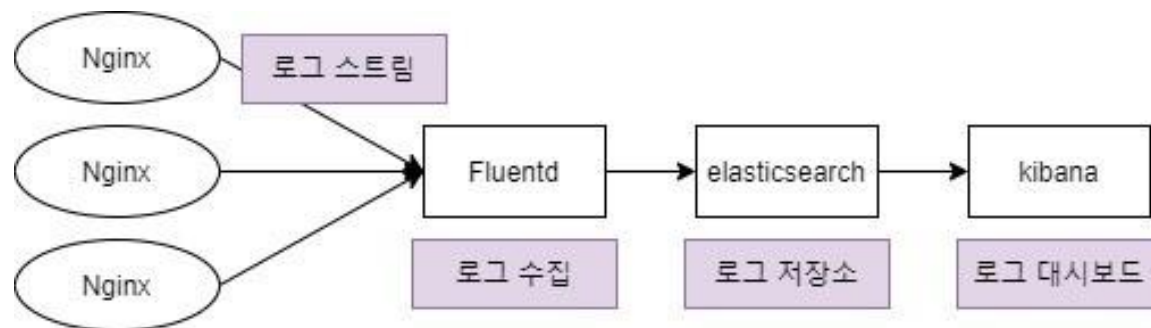
- Event의 구성
- Fluentd가 읽어들이는 데이터는 tag, time, record 로 구성된 이벤트(Event) 로 처리
 - tag: 이벤트를 어디로 보낼지 결정하기 위한 구분값
 - time: 이벤트가 발생한 시간
 - record: 데이터 (JSON)
- 이벤트 컴포넌트 중에서 tag는 Fluentd 내부 Routing의 기본으로, 각 이벤트들이 설정된 tag에 맞는 태그로 흘러 들어갈 수 있게끔 구분해 주는 역할 수행

Fluentd 설정 파일의 확인

- Fluentd를 실행하기 위해, 각 라이프사이클에 해당하는 지정자들을 설정 파일(Configuration File)에 작성. Fluentd 에이전트가 해당 설정 파일을 물고 실행
- 기본적으로 Ubuntu의 경우, 설정 파일은 /etc/td-agent 경로 아래 td-agent.conf라는 파일로 설정
- Ubuntu에서 fluentd를 설치하는 경우, 관련 패키지 에러가 발생할 수 있음
 - 패키지 관련 에러가 발생하면, apt-get을 이용하여 에러 로그에서 나타내고 있는 해당 패키지를 설치
 - Ubuntu 버전에 따라 제공하는 Fluentd 버전이 다르므로 반드시 확인
- td-agent 커맨드로 fluentd agent를 실행
- 백그라운드에서 데몬으로 수행하려면, -d 옵션과 --log 옵션을 추가하여 로그를 확인

Fluentd를 활용한 Log 수집 분석과 데몬셋(Daemonset)

- Nginx를 컨테이너 이미지로 POD에 배포, 그러면 nginx의 로그는 스트림으로 계속 출력되고
- 이때 로그들을 모아줄 수집기가 필요하다.
- 그게 Fluentd !
- Fluentd는 로그를 수집하는 툴이기 때문에 모든 node에 동일하게 배포되어야 하며, 그래서 DaemonSet으로 배포를 함

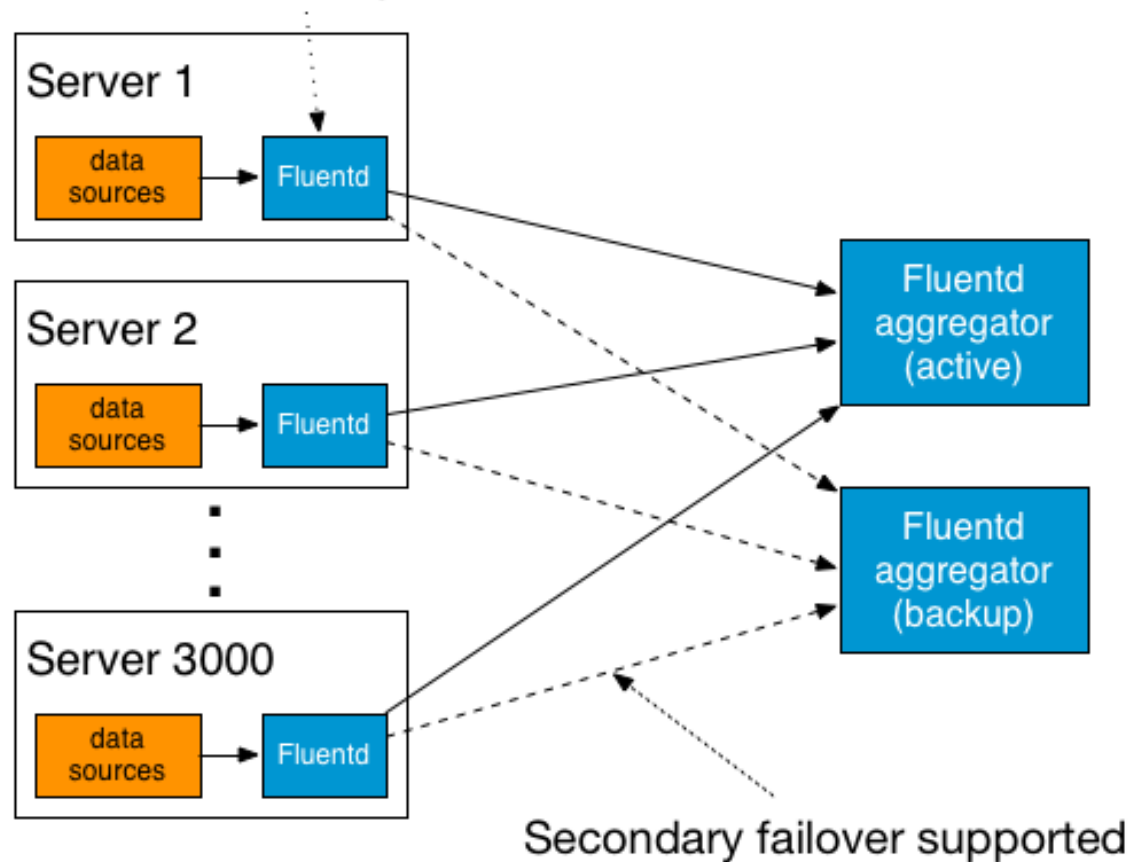


Log 누락을 막기 위한 8가지 Tips

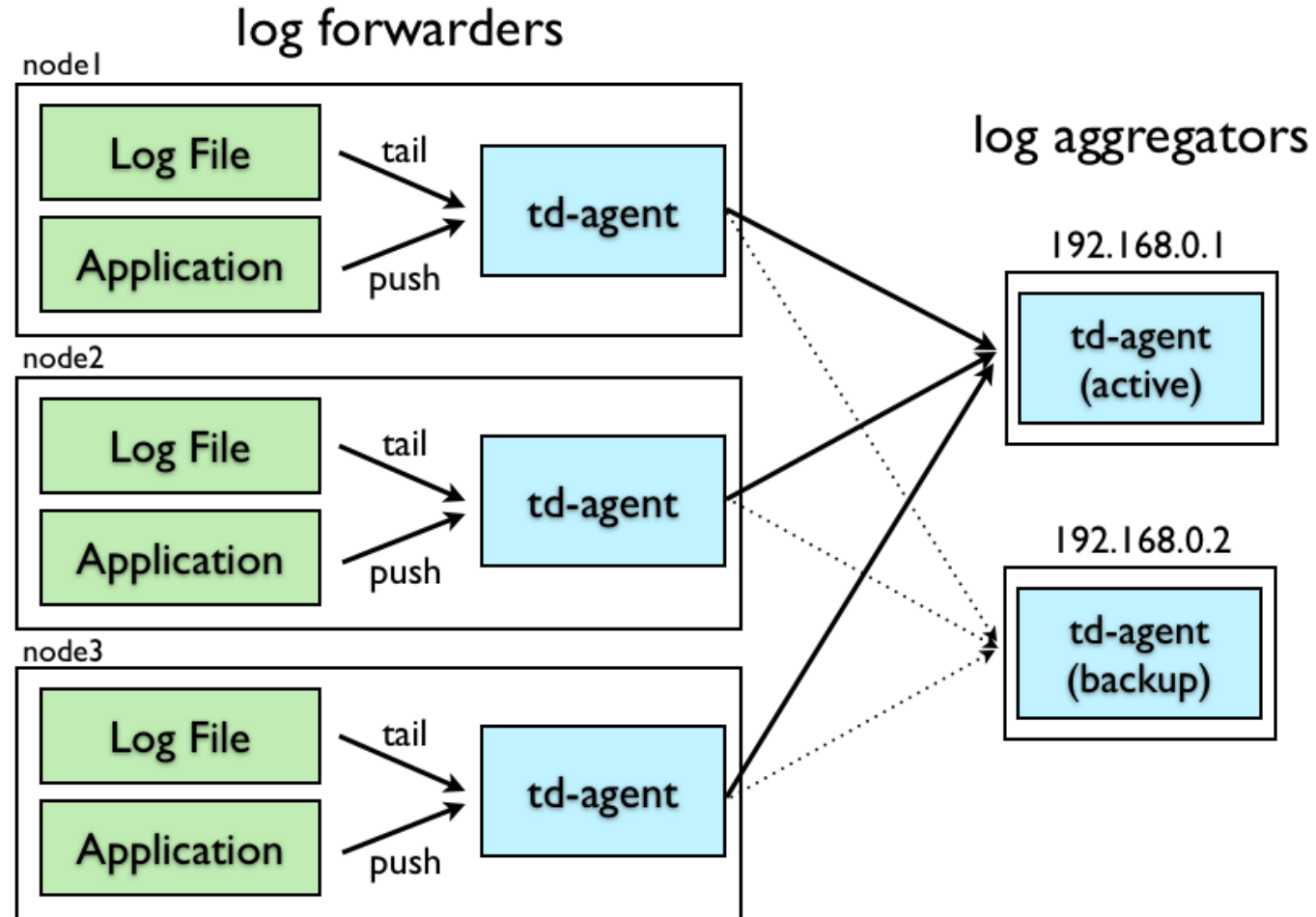
- 시스템 설정으로 로그 누락 막기
- 적절한 버퍼 설정하기
- 네트워크 단절에 대비하기
- 프로세스 다운에 대비하기
- 로그 누락을 막기 위한 forward 플러그인의 설정
- 로그 Aggregator의 이중 구성
- 종료 전에 버퍼를 플러시하기
- 메모리 버퍼가 있는 언어를 이용하기

Reliability and Scalability

file based buffering w/ retries

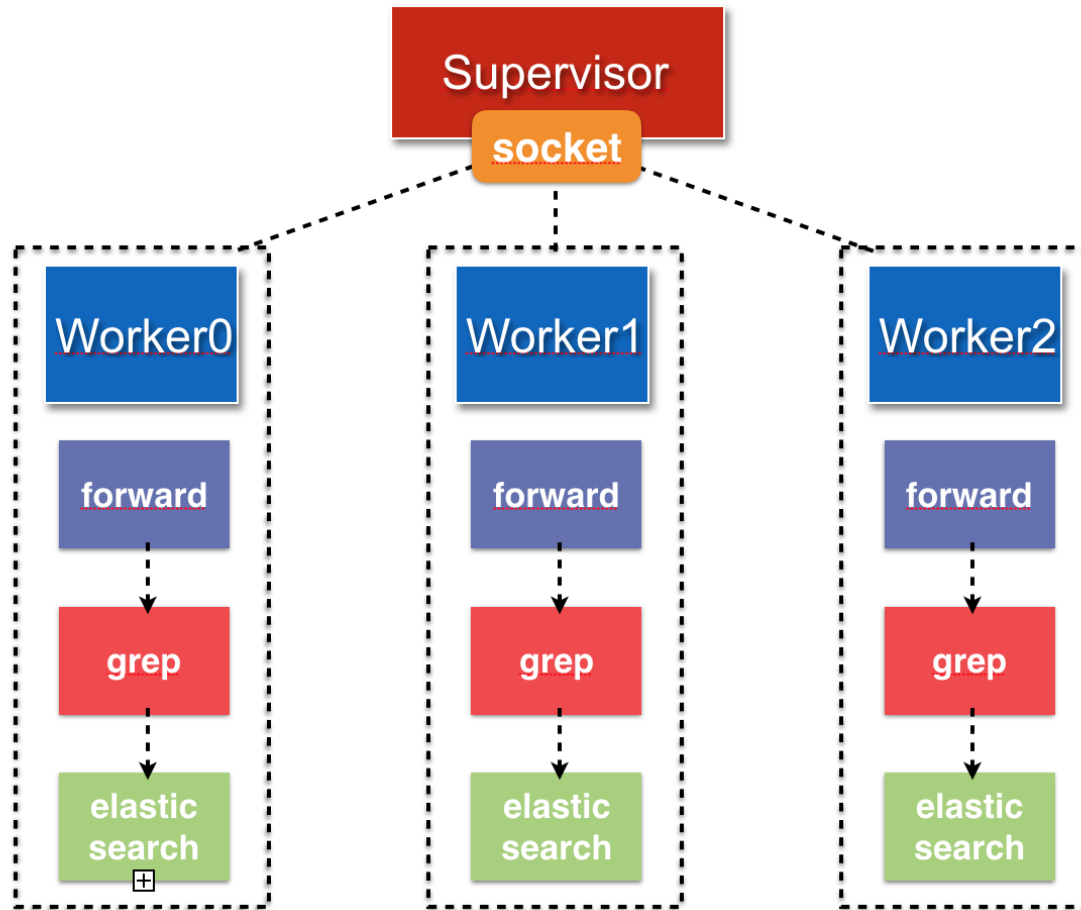


High Availability Config



Fluentd provides **"at most once"** and **"at least once"** transfers

Multi Process Workers



One instance of fluentd launches a supervisor and a worker.
A worker consists of input/filter/output plugins.

EFK 실습 예제 I : EFK를 이용한 간단한 Log 수집 및 시각화

- 간단한 Log 생성 Python 코드: `main.py`

```
$ tail -f log.json
```

```
$sudo service td-agent status
```

```
$ sudo ls /etc/td-agent/.
```

```
$ sudo vi /etc/td-agent/td-agent.conf
```

```
$ sudo service td-agent stop
```

```
$ sudo service td-agent start
```

```
$ python main.py
```

EFK 실습 예제 I : EFK를 이용한 간단한 Log 수집 및 시각화

- td-agent 설치

```
$ curl -L https://toolbelt.treasuredata.com/sh/install-ubuntu-xenial-td-agent2.sh | sh
```

- 데몬 실행

```
$ /etc/init.d/td-agent restart (재시작)
```

```
$ /etc/init.d/td-agent status (상태 확인)
```

- 시작시 자동 스크립트 실행

```
$ sudo update-rc.d td-agent defaults 95 10
```

```
$ sudo service td-agent status
```

Fluentd Forwarder 예제 II: Ngix Log 수집

- Fluentd (on Ubuntu Precise) 설치

```
$ sudo curl -L http://toolbelt.treasuredata.com/sh/install-ubuntu-precise.sh | sh
```

- Web Log 서버의 Fluentd 접속 권한 설정

```
$ sudo chmod og+rx /var/log/httpd
```

```
$ sudo chmod og+r /var/log/messages /var/log/secure /var/log/httpd/*
```

- /etc/rsyslogd.conf 수정

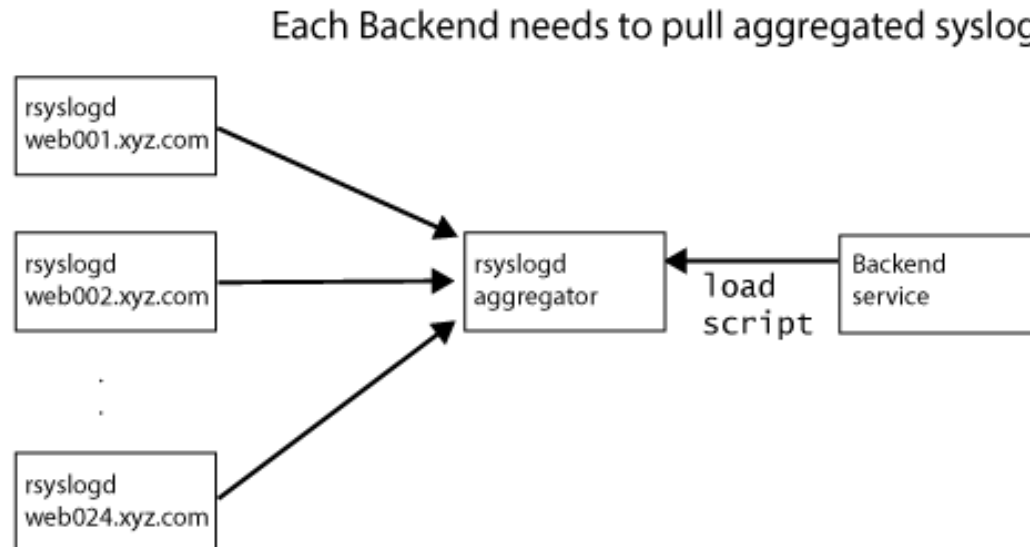
```
*.* @127.0.0.1:42185
```

- rsyslogd 재시작

```
$ sudo service rsyslog restart
```

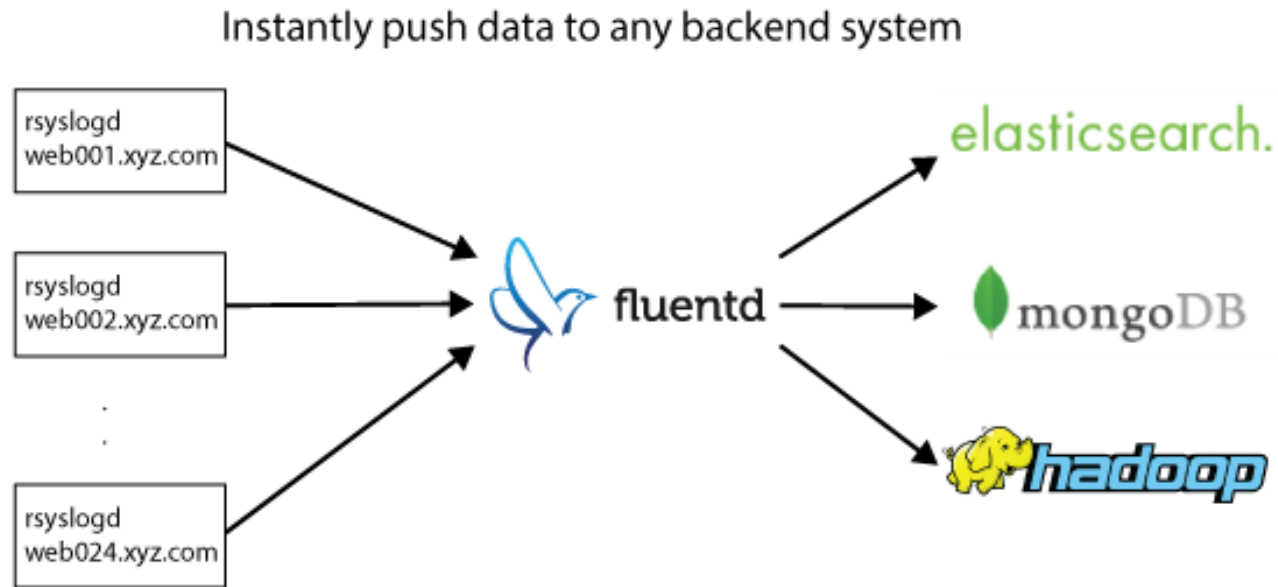
Aggregating Rsyslogd 예제 III : Rsyslogd Output 수집

- rsyslogd는 syslog를 수집하고 집계하는 검증된 진정한 미들웨어



<https://www.fluentd.org/guides/recipes/rsyslogd-aggregation>

- Fluentd는 즉시 사용 가능한 많은 데이터 소비자를 지원합니다. 적절한 출력 플러그인을 설치하면 몇 가지 구성 변경으로 새 데이터 소스를 추가
- Fluentd는 조정 가능한 횟수 및 버퍼링 설정을 사용하여 각 소비자에게 데이터를 PUSH



- Fluentd 설정

```
$ curl -L http://toolbelt.treasuredata.com/sh/install-ubuntu-precise.sh | sh
```

- Elasticsearch 출력 플러그인을 설치하고, 실행

```
$ /usr/sbin/td-agent-gem install fluent-plugin-elasticsearch
```

- /etc/td-agent/td-agent.conf 설정

```
<source>
  type syslog
  port 42185
  tag rsyslog
</source>

<match rsyslog.**>
  type copy
  <store>
    # for debug (see /var/log/td-agent.log)
    type stdout
  </store>
  <store>
    type elasticsearch
    logstash_format true
    flush_interval 10s # for testing.
    host YOUR_ES_HOST
    port YOUR_ES_PORT
  </store>
</match>
```

/etc/td-agent/td-agent.conf 설정

- Elasticsearch로의 데이터 흐름을 다시 시작하고 확인

\$ sudo service td-agent restart

Fluentd Forwarder 예제 IV: Fluentd MySQL slow log 연동

https://github.com/JSJeong-me/EFK/blob/main/Day_02.md