2일차. 플루언트디를 통한 파일 데이터 수집 - Fluentd

> *플루언트디를 통해 다양한 수집 예제를 실습합니다 이번 장에서 사용하는 외부 오픈 포트는 22, 80, 5601, 8080, 9880, 50070 입니다*

- 목차

예제 1 웹 서버를 통해서 전송 받은 데이터를 표준 출력으로 전달

1. 도커 컨테이너 기동

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex1
docker-compose up -d
docker logs -f fluentd
```

2. HTTP 로 Fluentd 서버 동작 유무 확인

```
docker ps --filter name=fluentd
curl -i -X POST -d 'json={"action":"login","user":2}' http://localhost:9880/test
```

3. Fluentd 구성 파일을 분석합니다

- fluent.conf

  ```
  <source>
    @type http
    port 9880
    bind 0.0.0.0
  </source>
  ```

@type stdout ``` * docker-compose.yml ```yml version: "3" services: fluentd: container_name: fluentd image: psyoblade/data-engineer-intermediate-day2-fluentd user: root tty: true ports: - 9880:9880 volumes: - ./fluent.conf:/fluentd/etc/fluent.conf ``` ### 4. 기동된 Fluentd 를 종료합니다 ```bash docker-compose down docker ps -a ```

예제 2 더미 에이전트를 통해 생성된 이벤트를 로컬 저장소에 저장

1. 도커 컨테이너 기동

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex2
docker-compose up -d
docker logs -f fluentd
```

2. 로컬 경로에 파일이 저장되는 지 확인

- target 경로에 로그가 1분 단위로 플러시 됩니다

```
ls -al target
tree target
```

3. Fluentd 구성 파일을 분석합니다

- fluent.conf

```
<source>
  @type dummy
  tag dummy.info
  size 5
  rate 1
  auto_increment_key seq
  dummy {"info":"hello-world"}
</source>
```

@type dummy tag dummy.debug size 3 rate 1 dummy {"debug":"hello-world"} @type record_transformer table_name ${tag_parts[0]} @type file path_suffix .log path /fluentd/target/${table_name}/%Y%m%d/part-%Y%m%d.%H%M timekey 1m timekey_wait 10s timekey_use_utc false timekey_zone +0900 ``` * docker-compose.yml ```yml version: "3" services: fluentd: container_name: fluentd image: psyoblade/data-engineer-intermediate-day2-fluentd user: root tty: true volumes: - ./fluent.conf:/fluentd/etc/fluent.conf - ./source:/fluentd/source - ./target:/fluentd/target ``` ### 4. 기동된 Fluentd 를 종료합니다 ```bash docker-compose down docker ps -a ```

예제 3 시스템 로그를 테일링 하면서 표준 출력으로 전달

1. 도커 컨테이너 기동 및 수집해야 할 로그폴더(source)를 생성합니다

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex3
mkdir source
docker-compose up -d
docker logs -f fluentd
```

2. 시스템 로그를 임의로 생성

- 로그 생성기를 통해 accesslog 파일을 계속 source 경로에 append 하고, target 경로에서는 수집되는지 확인합니다

```
python flush_logs.py
tree source
tree target
```

- 임의 로그 생성기 (flush_logs.py) 코드를 분석합니다

```
#!/usr/bin/env python
import sys, time, os, shutil
```

1. read apache_logs flush every 100 lines until 1000 lines

2. every 1000 lines file close & rename file with seq

3. create new accesslogs and goto 1.

def readlines(fp, num_of_lines): lines = "" for line in fp: lines += line num_of_lines = num_of_lines - 1 if num_of_lines == 0: break return lines

fr = open("apache_logs", "r") for x in range(0, 10): fw = open("source/accesslogs", "w+") for y in range(0, 10): lines = readlines(fr, 100) fw.write(lines) fw.flush() time.sleep(0.1) sys.stdout.write(".") sys.stdout.flush() fw.close() print("file flushed ... sleep 10 secs") time.sleep(10) shutil.move("source/accesslogs", "source/accesslogs.%d" % x) print("renamed accesslogs.%d" % x) fr.close()

````
### 3. Fluentd 구성 파일을 분석합니다
* fluent.conf
```conf
<source>
    @type tail
    @log_level info
    path /fluentd/source/accesslogs
    pos_file /fluentd/source/accesslogs.pos
    refresh_interval 5
    multiline_flush_interval 5
    rotate_wait 5
    open_on_every_update true
    emit_unmatched_lines true
    read_from_head false
    tag weblog.info
    <parse>
        @type apache2
    </parse>
</source>

<match weblog.info>
    @type file
    @log_level info
    add_path_suffix true
    path_suffix .log
    path /fluentd/target/${tag}/%Y%m%d/accesslog.%Y%m%d.%H
    <buffer time,tag>
        timekey 1h
        timekey_use_utc false
        timekey_wait 10s
        timekey_zone +0900
        flush_mode immediate
        flush_thread_count 8
````
````

```
    </buffer>
</match>

<match weblog.debug>
    @type stdout
    @log_level debug
</match>
```

- docker-compose.yml

```
version: "3"
services:
fluentd:
  container_name: fluentd
  image: psyoblade/data-engineer-intermediate-day2-fluentd
  user: root
  tty: true
  volumes:
    - ./fluent.conf:/fluentd/etc/fluent.conf
    - ./source:/fluentd/source
    - ./target:/fluentd/target
```

4. 기동된 Fluentd 를 종료합니다

```
docker-compose down
docker ps -a
```

예제 4 트랜스포머를 통한 시간 데이터 변환

1. 도커 컨테이너 기동

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex4
docker-compose up -d
docker logs -f fluentd
```

2. http://student#.lgebigdata.com:8080/test 위치로 POST 데이터를 전송합니다

- ARC Rest Client 를 사용해도 되고 curl 을 사용해도 됩니다
  - Epoch 시간은 https://www.epochconverter.com/ 사이트를 이용합니다

```
POST: http://student{no}.lgebigdata.com:8080/test
BODY: { "column1":"1", "column2":"hello-world", "logtime": 1593379470 }
```

curl -X POST -d '{ "column1":"1", "column2":"hello-world", "logtime": 1593379470 }'
http://student{no}.lgebigdata.com:8080/test

```
### 3. 수신된 데이터가 로그에 정상 출력됨을 확인합니다
### 4. 기동된 Fluentd 를 종료합니다
```bash
docker-compose down
docker ps -a
```
```

예제 5 컨테이너 환경에서의 로그 전송

> *도커 어플리케이션에서 발생하는 로그를 플루언트디로 적재합니다*

1. 도커 로그 수집 컨테이너를 기동합니다

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex5
./start_fluentd.sh
```

- 위와 같이 명령을 수행하고 Ctrl+P,Q 를 누르고 컨테이너를 종료하지 않고 밖으로 빠져나옵니다

2. 더미 어플리케이션을 기동하여 도커로그 수집기로 로그를 전송합니다

```
./start_application.sh
```

3. 다시 로그 수집기(aggregator)에 로그가 정상적으로 수신되는지 확인합니다

```
docker logs -f aggregator
```

3. Fluentd 구성 파일을 분석합니다

- aggregator.conf

```
<source>
  @type forward
  port 24224
  bind 0.0.0.0
</source>
```

<filter docker.*> @type parser key_name log reserve_data true @type json

<filter docker.*> @type record_transformer table_name ${tag_parts[1]}

<match docker.*> @type stdout

```
* start\_fluentd.sh
```bash
#!/bin/bash
if [ -z $PROJECT_HOME ]; then
    echo "\$PROJECT_HOME 이 지정되지 않았습니다"
    exit 1
fi
docker run --name aggregator -it -p 24224:24224 -v
$PROJECT_HOME/aggregator.conf:/fluentd/etc/fluent.conf fluent/fluentd
```

- start dummy docker container

```
./start_fluentd.sh
```

- execute my container and generate my message with --log-driver
  - By default, the logging driver connects to localhost:24224
```

```
docker run --rm --log-driver=fluentd ubuntu echo '{"message":"null tag message"}'
docker run --rm --log-driver=fluentd --log-opt tag=docker.{{.ID}} ubuntu echo '{"message":"send message with id"}'
docker run --rm --log-driver=fluentd --log-opt tag=docker.{{.Name}} ubuntu echo '{"message":"send message with name"}'
docker run --rm --log-driver=fluentd --log-opt tag=docker.{{.FullID}} ubuntu echo '{"message":"send message with full-id"}'
```

in case of connect other container

aggregator_address=`docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' aggregator` docker run --rm --log-driver=fluentd --log-opt tag=docker.helloworld --log-opt fluentd-address=$aggregator_address:24224 ubuntu echo '{"message":"exact ip send message"}'

```
* stop and remove container
```bash
docker stop aggregator
docker rm aggregator
```

4. 기동된 Fluentd 를 종료합니다

```
docker stop aggregator
docker rm aggregator
docker ps -a
```

예제 6 도커 컴포즈를 통한 로그 전송 구성

1. 도커 로그 수집 컨테이너를 기동하고, web, kibana, elasticsearch 모두 떠 있는지 확인합니다

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex6
docker-compose up -d
docker ps
```

2. 키바나를 통해 엘라스틱 서치를 구성합니다

- 키바나 사이트에 접속하여 색인을 생성
  - 1. http://student{id}.lgebigdata.com:5601 사이트에 접속 (모든 컴포넌트 기동에 약 3~5분 정도 소요됨)
  - 2. Explorer on my Own 선택 후, 좌측 "Discover" 메뉴 선택
  - 3. Step1 of 2: Create Index with 'fluentd-*' 까지 치면 아래에 색인이 뜨고 "Next step" 클릭
  - 4. Step2 of 2: Configure settings 에서 @timestamp 필드를 선택하고 "Create index pattern" 클릭
```

5. Discover 메뉴로 이동하면 전송되는 로그를 실시간으로 확인할 수 있음

- 웹 사이트에 접속을 시도
  - 1. http://student{id}.lgebigdata.com 사이트에 접속하면 It works! 가 뜨면 정상
  - 2. 다시 Kibana 에서 Refresh 버튼을 누르면 접속 로그가 전송됨을 확인

3. Fluentd 구성 파일을 분석합니다

- docker-compose.yml

```
version: "3"
services:
web:
  container_name: web
  image: httpd
  ports:
    - 80:80
  links:
    - fluentd
  logging:
    driver: fluentd
    options:
      fluentd-address: localhost:24224
      tag: httpd.access
fluentd:
  container_name: fluentd
  image: psyoblade/data-engineer-intermediate-day2-fluentd
  volumes:
    - ./fluentd/fluent.conf:/fluentd/etc/fluent.conf
  links:
    - elasticsearch
  ports:
    - 24224:24224
    - 24224:24224/udp
elasticsearch:
  container_name: elasticsearch
  image: elasticsearch:7.8.0
  command: elasticsearch
  environment:
    - cluster.name=demo-es
    - discovery.type=single-node
    - http.cors.enabled=true
    - http.cors.allow-credentials=true
    - http.cors.allow-headers=X-Requested-With,X-Auth-Token,Content-
Type,Content-Length,Authorization
    - http.cors.allow-origin=/https?:\/\/localhost(:[0-9]+)?/
    - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
  ulimits:
    memlock:
      soft: -1
      hard: -1
```

```
  expose:
    - 9200
  ports:
    - 9200:9200
  volumes:
    - ./elasticsearch/data:/usr/share/elasticsearch/data
    - ./elasticsearch/logs:/usr/share/elasticsearch/logs
  healthcheck:
    test: curl -s https://localhost:9200 >/dev/null; if [[ $$? == 52 ]]; then
echo 0; else echo 1; fi
    interval: 30s
    timeout: 10s
    retries: 5
kibana:
  container_name: kibana
  image: kibana:7.8.0
  links:
    - elasticsearch
  ports:
    - 5601:5601
```

4. 기동된 Fluentd 를 종료합니다

```
docker-compose down
docker ps -a
```

예제 7 멀티 프로세스를 통한 성능 향상

1. 서비스를 기동하고 별도의 터미널을 통해서 멀티프로세스 기능을 확인합니다 (반드시 source 경로를 호스트에서 생성합니다)

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex7
mkdir source
./startup.sh
```

2. 새로운 터미널에서 다시 아래의 명령으로 2가지 테스트를 수행합니다.

- 첫 번째 프로세스가 파일로 받은 입력을 표준 출력으로 내보내는 프로세스입니다

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex7
for x in $(seq 1 1000); do echo "{\"hello\":\"world\"}" >> source/start.log;
done
```

- 두 번째 프로세스는 HTTP 로 입력 받은 내용을 표준 출력으로 내보내는 프로세스입니다

```
curl -XPOST -d "json={\"hello\":\"world\"}" http://localhost:9880/test
```

3. Fluentd 구성 파일을 분석합니다

- fluent.conf

```
<system>
  workers 2
  root_dir /fluentd/log
</system>
<worker 0>
  <source>
      @type tail
      path /fluentd/source/*.log
      pos_file /fluentd/source/local.pos
      tag worker.tail
      <parse>
          @type json
      </parse>
  </source>
  <match>
      @type stdout
  </match>
</worker>
<worker 1>
  <source>
      @type http
      port 9880
      bind 0.0.0.0
  </source>
  <match>
      @type stdout
  </match>
</worker>
```

- startup.sh

```
#!/bin/bash
export PROJECT_HOME=`pwd`
name="multi-process"
echo "docker run --name $name -u root -p 9880:9880 -v
$PROJECT_HOME/fluent.conf:/fluentd/etc/fluent.conf -v
$PROJECT_HOME/source:/fluentd/source -v $PROJECT_HOME/target:/fluentd/target -
it psyoblade/data-engineer-intermediate-day2-fluentd"
docker run --name $name -u root -p 9880:9880 -v
$PROJECT_HOME/fluent.conf:/fluentd/etc/fluent.conf -v
$PROJECT_HOME/source:/fluentd/source -v $PROJECT_HOME/target:/fluentd/target -
it psyoblade/data-engineer-intermediate-day2-fluentd
```

- shutdown.sh

```
#!/bin/bash
name="multi-process"
container_name=`docker ps -a --filter name=$name | grep -v 'CONTAINER' | awk '{
print $1 }'`
docker rm -f $container_name
```

4. 기동된 Fluentd 를 종료합니다

```
./shutdown.sh
docker ps -a
```

예제 8 멀티 프로세스를 통해 하나의 위치에 저장

1. 서비스를 기동하고 별도의 터미널을 통해서 멀티프로세스 기능을 확인합니다 (반드시 source 경로를 호스트에서 생성합니다)

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex8
mkdir source
./startup.sh
```

2. 별도의 터미널에서 아래의 명령으로 멀티 프로세스를 통해 하나의 위치에 저장되는 것을 확인합니다

```
tree target
```

3. Fluentd 구성 파일을 분석합니다

- fuent.conf

```
<system>
  workers 2
  root_dir /fluentd/log
</system>
```

<worker 0> @type http port 9880 bind 0.0.0.0 @type stdout @type file @id out_file_0 path "/fluentd/target/#{worker_id}/${tag}/%Y/%m/%d/testlog.%H%M" <buffer time,tag> timekey 1m timekey_use_utc false timekey_wait 10s

<worker 1> @type http port 9881 bind 0.0.0.0 @type stdout @type file @id out_file_1 path "/fluentd/target/#{worker_id}/${tag}/%Y/%m/%d/testlog.%H%M" <buffer time,tag> timekey 1m timekey_use_utc false timekey_wait 10s

```
* startup.sh
```bash
#!/bin/bash
export PROJECT_HOME=`pwd`
name="multi-process-ex"
echo "docker run --name $name -u root -p 9880:9880 -p 9881:9881 -v
$PROJECT_HOME/fluent.conf:/fluentd/etc/fluent.conf -v
$PROJECT_HOME/source:/fluentd/source -v $PROJECT_HOME/target:/fluentd/target -it
psyoblade/data-engineer-intermediate-day2-fluentd"
docker run --name $name -u root -p 9880:9880 -p 9881:9881 -v
$PROJECT_HOME/fluent.conf:/fluentd/etc/fluent.conf -v
$PROJECT_HOME/source:/fluentd/source -v $PROJECT_HOME/target:/fluentd/target -it
psyoblade/data-engineer-intermediate-day2-fluentd
```

- progress.sh

```
#!/bin/bash
max=60
```
```

```
dot="."
for number in $(seq 0 $max); do
  for port in $(seq 9880 9881); do
      # echo curl -XPOST -d "json={\"hello\":\"$number\"}"
http://localhost:$port/test
      curl -XPOST -d "json={\"hello\":\"$number\"}" http://localhost:$port/test
  done
  echo -ne "[$number/$max] $dot\r"
  sleep 1
  dot="$dot."
done
tree
```

- shutdown.sh

```
#!/bin/bash
name="multi-process-ex"
container_name=`docker ps -a --filter name=$name | grep -v 'CONTAINER' | awk '{
print $1 }'`
docker rm -f $container_name
```

4. 기동된 Fluentd 를 종료합니다

```
Ctrl+C
docker-compose down
docker ps -a
```

예제 9 전송되는 데이터를 분산 저장소에 저장

1. 서비스를 기동합니다

```
cd /home/ubuntu/work/data-engineer-intermediate-training/day2/ex9
./startup.sh
```

2. 별도의 터미널에서 모든 서비스(fluentd, namenode, datanode)가 떠 있는지 확인합니다

```
docker ps
```

3. HTTP 로 전송하고 해당 데이터가 하둡에 저장되는지 확인합니다

- http://student#.lgebigdata.com:50070/explorer.html 에 접속하여 확인합니다
- 혹은 namenode 에 설치된 hadoop client 로 확인 합니다
- WARN: 현재 노드수가 1개밖에 없어서 Replication 오류가 나고 있습니다.

```
./progress.sh
docker exec -it namenode hadoop fs -ls /user/fluent/webhdfs/
```

4. Fluentd 구성 파일을 분석합니다

- fluent.conf

```
version: '2'
services:
namenode:
container_name: namenode
image: bde2020/hadoop-namenode:1.1.0-hadoop2.8-java8
volumes:
- ./hadoop/namenode:/hadoop/dfs/name
environment:
- CLUSTER_NAME=test
env_file:
- ./hadoop/hadoop.env
ports:
- 50070:50070
datanode:
container_name: datanode
image: bde2020/hadoop-datanode:1.1.0-hadoop2.8-java8
depends_on:
- namenode
volumes:
- ./hadoop/datanode:/hadoop/dfs/data
env_file:
- ./hadoop/hadoop.env
ports:
- 50075:50075
fluentd:
container_name: fluentd
image: psyoblade/data-engineer-intermediate-day2-fluentd
depends_on:
- namenode
- datanode
user: root
ports:
- 9880:9880
tty: true
volumes:
- ./fluentd/fluent.conf:/fluentd/etc/fluent.conf
volumes:
namenode:
datanode:
```

- start container

```
docker-compose up -d
```

- generate logs with progress.sh

```
#!/bin/bash
max=60
dot="."
for number in $(seq 0 $max); do
curl -XPOST -d "json={\"hello\":\"$number\"}"
http://localhost:9880/test.info
echo -ne "[$number/$max] $dot\r"
```

```
sleep 1
dot="$dot."
done
```

4. 기동된 Fluentd 를 종료합니다

```
Ctrl+C
docker-compose down
docker ps -a
```