Hyperledger Fabric + ELK Stack

2019-07-15 블록체인기술팀 강재구

개요

- Fabric에서 발생하는 트랜잭션들을 필터링하여 ELK Stack에 저장해 보았다.

  향후 편의를 위한 단순 데이터 조회와 같은 기능은 ELK로 따로 분리할 수 있을 것 같다.

  또한, 저장된 데이터들을 다양한 각도로 모니터링 할 수 있다는 장점을 가지고 있다.

Data Flow

```
Fabric Network      1. Block Commit
      |
      |
      V
 EventListener      2. Block Log & Filtered Log 저장
      |
      |
      V
   Filebeat         3. Log파일의 변경을 확인하고 Logstash에게 전달
      |
      |
      V
   Logstash         4. Filebeat의 Output을 필터링 하여 Elaticsearch에게 전달
      |
      |
      V
Elasticsearch       5. 데이터를 저장
      |
      |
      V
    Kibana          6. 저장된 데이터들을 모니터링
```

실습 환경

- Virtual Box 환경 구성

노드	OS	HOST	MEMORY	CPU	SERVICES
Node 1	Ubuntu	192.168.56.10	4096 MB	2	ELK Stack(Elastic, Logstash, Kibana)
Node 2	Ubuntu	192.168.56.20	2048 MB	2	Fabric, Event Listener, File Beat

- 버전

이름	버전
Ubuntu	16.04.6 LTS
Docker	18.09.6
Docker-compose	1.23.2
Fabric	1.4.0
Elasticsearch	7.2.0
Logstash	7.2.0
Kibana	7.2.0
Filebeat	7.2.0

목차

1. Fabric Network

- 생략 (fabric-samples 사용)

2. EventListener

Edit eventsclient.go

- ROOT 변수에 configtx.yaml의 파일 명 설정

```
// eventsclient.go

const (
        OLDEST = -2
        NEWEST = -1

        ROOT = "configtx"
)
```

- readCLInputs() 변경

```
// eventsclient.go

// 이벤트 수신 PEER 설정 host:port
//    +   /etc/hosts 파일에 PEER 추가

flag.StringVar(&serverAddr, "server", "peer0.org1.example.com:7051", "The RPC se
rver to connect to.")

// 이벤트 수신 CHANNEL 설정
```

```
flag.StringVar(&channelID, "channelID", "mychannel", "The channel ID to deliver
from.")

// server.key, server.crt, ca.crt||ca.pem 경로 설정
flag.StringVar(&clientKeyPath, "clientKey", "/crypto-config/peerOrganizations/or
g1.example.com/peers/peer0.org1.example.com/tls/server.key", "Specify path to th
e client TLS key")
flag.StringVar(&clientCertPath, "clientCert", "/crypto-config/peerOrganizations/
org1.example.com/peers/peer0.org1.example.com/tls/server.crt", "Specify path to
the client TLS certificate")
flag.StringVar(&serverRootCAPath, "rootCert", "/crypto-config/peerOrganizations/
org1.example.com/peers/peer0.org1.example.com/tls/ca.crt", "Specify path to the
server root CA certificate")
```

- initMSP() 변경

```
// eventclient.go

var mspMgrConfigDir = "/crypto-
config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp"
var mspID = "Org1MSP"
var mspType = "bccsp"
```

FABRIC_CFG_PATH 설정

```
export FABRIC_CFG_PATH=~/fabric-samples/first-network
```

> 위 까지 진행한 후 실행하면, PEER의 commit 발생 시 BLOCK 정보가 console에 나타난다.
>
> BLOCK에서 사용하고자 하는 데이터를 추출해서 사용할 것이다. (write set)
>
> **saveFilteredLog()** : JSON Parsing function
>
> 위 메소드는 JSON을 Parsing하여 .log 파일로 저장한다.
>
> 본 글에서는 Docker를 이용하여 EventListener를 활성화 시킬 것이다.

EventListener to Docker container

> EventListener를 Docker container화 시키는 작업이다.

```
## docker-eventlistener.yaml

version: '2'

services:
  eventlistener:
    container_name: eventlistener
    image: eventlistener:1.0
    command: go run /eventsclient.go
    volumes:
      - ./crypto-config:/crypto-config
      - ./eventdir/log:/log
      - ./eventdir/eventsclient.go:/eventsclient.go
    extra_hosts:
      - peer0.org1.example.com:192.168.56.20
```

Run EventListener

```
docker-compose -f ./docker-eventlistener.yaml up -d
docker logs -f eventlistener
```

다음과 같은 로그가 남는다.

```
{
        "data": {
                "data": [
                        {
                                "payload": {
                                        "data": {
                                                "actions": [
                                                        {
                                                                ...
                                                                },
```

또한 docker-eventlistener.yaml에서 지정한 경로로 log파일 2개가 남는다.

```
cd $PATH/eventdir/log
ls
blockhistory.log  filteredblockhistory.log
```

3. ELK Stack

> 본 글에서는 Docker를 이용하여 ELK Stack을 활성화 시킬 것이다.
>
> ELK의 configuration을 손수 작성 할수도 있지만 편의를 위해 ELK 전부를 install 하고,
>
> 제공된 config에서 일부를 수정 한 후, docker volume을 지정해 주는 방식을 사용한다.

Pull ELK Stack

```
# ELK (Elasticsearch, Logstash, Kibana) Download
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.2.0-
linux-x86_64.tar.gz
wget https://artifacts.elastic.co/downloads/logstash/logstash-7.2.0.tar.gz
wget https://artifacts.elastic.co/downloads/kibana/kibana-7.2.0-linux-
x86_64.tar.gz
tar -zcvf elasticsearch-7.2.0-linux-x86_64.tar.gz
tar -zcvf logstash-7.2.0.tar.gz
tar -zcvf kibana-7.2.0-linux-x86_64.tar.gz

# Pull ELK Image
docker pull docker.elastic.co/elasticsearch/elasticsearch:7.2.0
docker pull docker.elastic.co/logstash/logstash:7.2.0
docker pull docker.elastic.co/kibana/kibana:7.2.0
```

Edit ELK's configuration

```
# $*_HOME은 *의 설치경로
cd $ELASTIC_HOME/config
vim elasticsearch.yml
```

```
# elasticsearch.yml
network.host: 0.0.0.0
http.port: 9200
```

```
cd $LOGSTASH_HOME/config
vim logstash.conf
```

```
input {
  beats {
    port => 5044
  }
  stdin { }
```

```
}

filter {
        json {
                source => "message"
                add_field => {
                        "%{key}" => "%{value}"
                }
        }
}

output {
  elasticsearch {
    hosts => ["192.168.56.10:9200"]
  }

  stdout {
    codec => rubydebug
  }
}
```

```
cd $KIBANA_HOME/config
vim kibana.yml
```

```
server.port: 5601
server.host: "0.0.0.0"
elasticsearch.hosts: ["http://192.168.56.10:9200"]
```

Write docker-compose.yaml

```
version: '2.2'

services:
  elasticsearch01:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.2.0
    container_name: elasticsearch01
    environment:
      - node.name=elasticsearch01
      - cluster.initial_master_nodes=elasticsearch01
    volumes:
      - ./elasticsearch/config/:/usr/share/elasticsearch/config/
      - ./elasticsearch/data/:/usr/share/elasticsearch/data
    ports:
      - 9200:9200

  kibana:
```

```
    image: docker.elastic.co/kibana/kibana:7.2.0
    container_name: kibana
    environment:
      SERVER_NAME: kibana.example.com
    volumes:
      - ./kibana/config/:/usr/share/kibana/config/
    ports:
      - 5601:5601
    depends_on:
      - elasticsearch01

  logstash:
    image: docker.elastic.co/logstash/logstash:7.2.0
    container_name: logstash
    volumes:
      - ./logstash/:/usr/share/logstash/
    command: ./bin/logstash -f /usr/share/logstash/config/logstash.conf
    ports:
      - 5044:5044
      - 9600:9600
```

Run ELK Stack

```
sudo sysctl vm.max_map_count=262144
docker-compose -f ./docker-compose.yaml up -d
```

> ELK Stack과 Fabric, EventListener를 실행했다.
>
> Event발생 시, 새로 작성되는 .log파일을 logstash에게 넘겨주는 매개체가 필요하다.
>
> 우리는 Filebeat를 사용하여 ELK Stack과 Fabric을 연결 시켜 줄 것이다.

4. **Pull Filebeat**

```
docker pull docker.elastic.co/beats/filebeat:7.2.0
```

Filebeat configuration

- Write Filebeat Configuration

```
touch filebeat.docker.yaml
vim filebeat.docker.yml
```

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/filteredblockhistory.log
  multiline.pattern: "{\"write\":\"generated\"}"
  multiline.negate: true
  multiline.match: after

filebeat.config:
  modules:
    path: ${path.config}/modules.d/*.yml
    reload.enabled: false

filebeat.autodiscover:
  providers:
    - type: docker
      hints.enabled: true

processors:
- add_cloud_metadata: ~

output.logstash:
  hosts: "192.168.56.10:5044"
```

- Filebeat to docker

```
touch docker-filebeat.yaml
vim docker-filebeat.yaml
```

```
version: '2.2'

services:
  filebeat:
    image: docker.elastic.co/beats/filebeat:7.2.0
    container_name: filebeat
    command: filebeat -e -strict.perms=false
    volumes:
      - ./filebeat.docker.yml:/usr/share/filebeat/filebeat.yml
      - /var/run/docker.sock:/var/run/docker.sock
      - /home/jack2/fabric-samples/first-network/eventdir/log:/var/log
```

Run Filebeat

```
docker-compose -f docker-filebeat.yml up -d;
```

5. Check

정리하자면,

예제(mycc)에서는 총 2개의 Key(a, b)가 변경된다.

Block의 Read/Write Set에는 다음과 같은 기록을 확인할 수 있다.

(value는 base64 encoded 되어있다.)

```
"writes": [
        {
                "is_delete": false,
                "key": "a",
                "value": "OTA="
        },
        {
                "is_delete": false,
                "key": "b",
                "value": "MjEw"
        }
]
```

위 정보들이 Filtered 되어 filteredblockhistory.log로 저장된다.

```
{"write":"generated"}
{"is_delete":"false","key":"a","value":"90"}
{"write":"generated"}
{"is_delete":"false","key":"b","value":"210"}
```

Log 파일은 Filebeat를 통해 Logstash로 전달되고, 최종적으로 Elasticsearch에 적재된다.

> 이제, 정상적으로 Elasticsearch에 저장되었는지 확인 할 것이다.
>
> 확인은 Kibana(Elasticsearch의 모니터링 툴)을 이용한다.
>
> 인터넷 브라우저에서 Kibana가 활성화 되어 있는 port로 접근하면 된다.
>
> 본 예제를 그대로 따라했다면 192.168.56.10:5601 을 통해 Kibana를 사용할 수 있다.

아래와 같이 총 2개의 Key에 대한 변경이 저장된 것을 확인 할 수 있다.

	Time	_source
>	Jul 15, 2019 @ 11:17:34.737	is_delete: false log.offset: 67 log.file.path: /var/log/fi 11:17:34.737 input.type: log tags: beats_input_codec_plain_ agent.ephemeral_id: e7e0f77a-a512-4dd7-a2a4-1aeb5175dcde ag ecs.version: 1.0.0 _id: vdBs82sBijY85_taBvt- _type: _doc _
>	Jul 15, 2019 @ 11:17:34.737	is_delete: false host.name: 9d515d9a18af log.offset: 0 log 11:17:34.737 input.type: log tags: beats_input_codec_plain_ agent.ephemeral_id: e7e0f77a-a512-4dd7-a2a4-1aeb5175dcde ag ecs.version: 1.0.0 _id: vtBs82sBijY85_taBvuE _type: _doc _

각각의 txn을 눌러 확인해 본다.

Table JSON

	⊙	@timestamp	Jul 15, 2019 @ 11:17:34.737
	t	@version	1
	t	_id	vdBs82sBijY85_taBvt-
	t	_index	logstash-2019.07.15-000001
	#	_score	1
	t	_type	_doc
	t	agent.ephemeral_id	e7e0f77a-a512-4dd7-a2a4-1aeb5175dcde
	t	agent.hostname	9d515d9a18af
	t	agent.id	fd3b27ba-8b11-4423-90d2-1d0f55cd89d0
	t	agent.type	filebeat
	t	agent.version	7.2.0
	t	b	210
🔍 🔍 ◫ ✱	t	ecs.version	1.0.0
	t	host.name	9d515d9a18af
	t	input.type	log
	t	is_delete	false
	t	key	b
	t	log.file.path	/var/log/filteredblockhistory.log
	t	log.flags	multiline
	#	log.offset	67
	t	message	{"write":"generated"} {"is_delete":"false","key":"b","value":"210"}
	t	tags	beats_input_codec_plain_applied
	t	value	210

Table JSON

⊙	@timestamp	Jul 15, 2019 @ 11:17:34.737
t	@version	1
t	_id	vtBs82sBijY85_taBvuE
t	_index	logstash-2019.07.15-000001
#	_score	1
t	_type	_doc
t	a	90
t	agent.ephemeral_id	e7e0f77a-a512-4dd7-a2a4-1aeb5175dcde
t	agent.hostname	9d515d9a18af
t	agent.id	fd3b27ba-8b11-4423-90d2-1d0f55cd89d0
t	agent.type	filebeat
t	agent.version	7.2.0
t	ecs.version	1.0.0
t	host.name	9d515d9a18af
t	input.type	log
t	is_delete	false
t	key	a
t	log.file.path	/var/log/filteredblockhistory.log
t	log.flags	multiline
#	log.offset	0
t	message	{"write":"generated"} {"is_delete":"false","key":"a","value":"90"}
t	tags	beats_input_codec_plain_applied
t	value	90

> Kibana의 사용법을 익힌다면 효과적으로 데이터를 모니터링, 관리 할 수 있다.