■ Apache Flume

- 앞서 실행했던 것처럼, HDFS에 데이터를 입력할 때 아래와 같은 명령으로 간단히 처리 가능 -> $ hdfs dfs –put [디렉토리]
- 이런 경우는 미리 잘 준비된 데이터를 업로드할 때 유용할 것임
- 현실에서는 서비스의 로그가 계속 유입이 되고, 즉시/대량으로 HDFS같은 DataStore에 저장되어 분석해야 함
- 데이터를 유실 없이 안정적이게 전송하기 위해 다양한 옵션이 필요함
- Hadoop으로 데이터를 입력하기 위해 간단하고 유연하며 확장이 가능한 솔루션으로서 Apache Flume이 적합함
- Apache Flume은 2011년에 Cloudera CDH3에 처음으로 소개되었으며, 현재는 Apache의 Top-Level Project로 이전 CDH3에서의 Version인 0.9.x버전은 Flume-OG라 명명하고 1.0.0 이후의 버전부터 Flume-NG(Next Generation)이라 명명함

■ Flume의 개념

1) Flume의 중요 개념은 크게 Event와 Agent가 있음

2) Event : Flume을 통해 전달되는 데이터 단위

3) Event는 Header와 body로 나뉨

- Header는 key/value 형태이고 라우팅(데이터를 보낼 경로 선택)을 결정하거나 구조화된 정보 (예: 이벤트가 발생된 서버의 호스트명, timestamp 등 추가 가능)를 운반할 때 활용
- Body는 전달되는 데이터를 확인 가능

4) Agent : 이벤트를 전달하는 컨테이너, Source, Channel, Sink로 구성해 흐름을

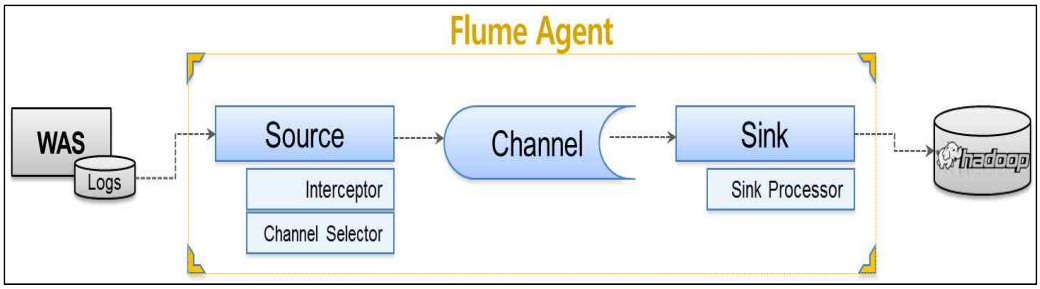
제어함

5) 에이전트끼리의 데이터 이동이 가능해서,(예: 1개의 에이전트가 다수의 에이전트와 연결 가능) 플룸은 목적에 따라 에이전트 간의 연결 및 확장이 가능

6) Source

- 이벤트를 수집하여 채널로 전달

- Interceptor : Source와 Channel 사이에서 데이터 필터링 및 가공



① Timestamp(이벤트 헤더에 현재 시간 값 추가),

② Static Interceptor(이벤트 헤더에 지정한 값 추가),

③ Regex Filtering Interceptor (정규표현식에 일치하는지에 따라 이벤트를 버릴지 결정)

④ Chennel Selector : Source가 받은 이벤트를 전달할 채널을 선택

- Source의 종류

① avro : Avro 클라이언트에서 전송하는 이벤트를 입력으로 사용, Agent와 Agent를 연결해줄 때 유용

② netcat : TCP로 라인 단위 수집

③ seq : 0부터 1씩 증가하는 EVENT 생성

④ exec : 명령행을 실행하고 콘솔 출력 내용을 데이터 입력으로 사용

⑤ syslogtcp : System 로그를 입력으로 사용

⑥ spooldir : 디렉토리에 새롭게 추가되는 파일을 데이터로 사용

⑦ thirft : Thrift 클라이언트에서 전송하는 이벤트를 입력으로 사용

⑧ jms : JMS 메시지 수집

8) Channel

- 이벤트를 Source와 Sink로 전달하는 통로

- 이벤트를 임시로 보관

- 종류: 메모리 채널, 파일 채널, JDBC채널

① 메모리 채널: Source에서 받은 이벤트를 Memory에 잠시 저장, 간편하고 빠른 고성능을 제공하지만 프로세스가 비정상적으로 종료시 이벤트 유실 가능성 있음

② 파일 채널: 속도는 메모리 채널에 비해 느리지만, 프로세스가 비정상적으로 죽더라도 프로세스를 재처리하여 이벤트 유실이 없음

③ JDBC 채널: JDBC로 저장

9) Sink

- Channel로 부터 받은 이벤트를 저장 또는 전달
- Sink Processor: 한 채널에 여러 Sink를 그룹으로 묶을 때 사용
- Sink Processor를 통해 Sink할 대상을 다중 선택하거나 여러개의 Sink를 하나의 그룹으로 관리하여 Failover(장애 극복)에 대응
- 종류

① null : 이벤트를 버림

② logger : 테스트 또는 디버깅을 위한 로깅

③ avro : 다른 Avro 서버(Avro Source)로 이벤트 전달

④ hdfs : HDFS에 저장

⑤ hbase : HBase에 저장

⑥ elasticsearch : 이벤트를 변환해서 ElasticSearch에 저장

⑦ file_roll : 로컬 파일에 저장

⑧ thrift : 다른 Thrift 서버(Thrift Source)로 이벤트 전달

■ Flume 설치

- server01에 bigdata 계정으로 접속 후 bigdata 홈디렉터리에 flume 파일 다운

```
$ wget http://mirror.navercorp.com/apache/flume/1.8.0/apache-flume-1.8.0-bin.tar.gz
```

- 다운 받은 'apache-flume-1.8.0-bin.tar.gz' 파일 압축 풀기

```
$ tar xvfz apache-flume-1.8.0-bin.tar.gz
```

- server01 세션을 하나 더 열어서 root로 로그인 후 /etc/profile 열기

```
# vi /etc/profile
```

- /etc/profile에 아래와 같이 플룸 바이너리 경로를 입력후 저장

```
export FLUME_HOME=/home/bigdata/apache-flume-1.8.0-bin
export PATH=$PATH:$FLUME_HOME/bin:$JAVA_HOME/bin
```

- source로 /etc/profile 적용

```
# source /etc/profile
```

- bigdata 계정으로 접속한 server01 세션을 열고 플룸의 conf 디렉터리로 이동

```
$ cd /home/bigdata/apache-flume-1.8.0-bin/conf
```

- flume-env.sh.template 파일을 flume-env.sh로 복사

```
$ cp flume-env.sh.template flume-env.sh
```

- flume-env.sh 파일을 열어 아래와 같이 자바 힙 메모리 설정

```
export JAVA_OPTS= Xms100m Xmx2000m
```

```
# Give Flume more memory and pre-allocate, enab
export JAVA_OPTS= Xms100m Xmx2000m
```

- source로 flume-env.sh 파일 적용

```
$ source flume-env.sh
```

■ 플럼 수집 기능 구현

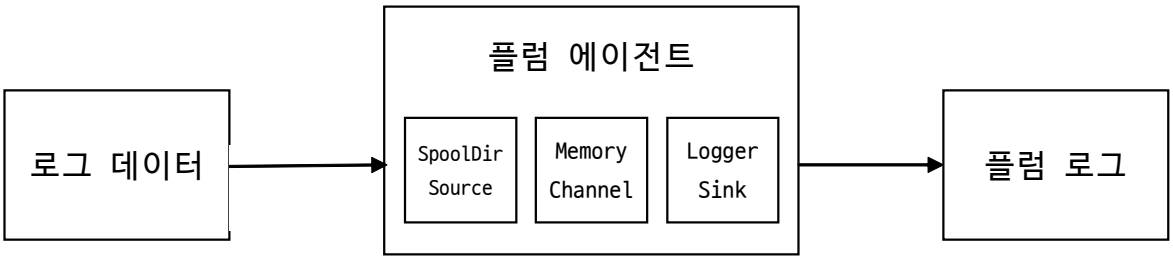
- HDFS로 적재하기 전, 플럼의 수집 기능을 테스트
- flume의 conf 디렉터리에서 test.conf 파일 생성

```
[bigdata@server01 conf]$ vi test.conf
```

- test.conf에 아래의 Agent 구성 입력

○ 플룸 Agent 구성 파일 설명

- Agent 이름: Test_Agent
- Test_Agent 구성: Source-Channel-Sink



```
Test_Agent.sources  = TestSource_SpoolSource
Test_Agent.channels = TestChannel_Channel
Test_Agent.sinks    = TestSink_LoggerSink
```

- 플럼의 에이전트에 사용할 Source, Channel, Sink의 각 리소스 변수 정의
- Test_Agent Source의 변수는 TestSource_SpoolSource
- Test_Agent Channel의 변수는 TestChannel_Channel
- Test_Agent Sink의 변수는 TestSink_LoggerSink

```
Test_Agent.sources.TestSource_SpoolSource.type = spooldir
Test_Agent.sources.TestSource_SpoolSource.spoolDir = /home/bigdata/working/jbm-batch-log
Test_Agent.sources.TestSource_SpoolSource.deletePolicy = immediate
Test_Agent.sources.TestSource_SpoolSource.batchSize = 1000
```

- Test_Agent의 Source 설정
- spooldir: 파일이 생성되면 그 파일의 내용을 수집
- “/home/bigdata/working/jbm-batch-log” 경로에서 생성되는 로그 파일 수집

- 경로 생성

```
[bigdata@server01 ~]$ mkdir working
[bigdata@server01 ~]$ mkdir working/jbm-batch-log
```

- batchSize 설정값 만큼 읽어서 Channel에 전송

```
Test_Agent.channels.TestChannel_Channel.type = memory
Test_Agent.channels.TestChannel_Channel.capacity = 100000
Test_Agent.channels.TestChannel_Channel.transactionCapacity  = 10000
```

- Test_Agent의 Channel 설정
- Channel의 종류를 “memory”로 설정
- Memory Channel: Source로부터 받은 데이터를 메모리상에 중간 적재. 성능은 높지만 메모리에 저장하는 거라 데이터가 유실될 위험이 있음

```
Test_Agent.sinks.TestSink_LoggerSink.type = logger
```

- Test_Agent의 Sink 설정
- Logger Sink는 테스트용으로 수집한 데이터를 플럼의 로그에 출력

```
Test_Agent.sources.TestSource_SpoolSource.channels = TestChannel_Channel
Test_Agent.sinks.TestSink_LoggerSink.channel = TestChannel_Channel
```

- Spooldir Source와 Memory Channel을 연결
- Logger Sink와 Memory Channel을 연결

- 작성후 플럼의 홈디렉터리(apache-flume-1.8.0-bin)로 이동

```
[bigdata@server01 conf]$ cd ..
```

- Test_Agent 실행

flume-ng agent -c conf -f conf파일 -n Agent이름 -Dflume.root.logger=INFO,console

```
[bigdata@server01 apache-flume-1.8.0-bin]$ ./bin/flume-ng agent -c conf -f conf/test.conf -n Test_Agent -Dflume.root.logger=INFO,console
```

```
2018-06-11 01:27:59,891 (lifecycleSupervisor-1-4) [INFO - org.apache.flume.source.SpoolDirectorySource.start(SpoolDirectorySource.java:83)] SpoolDirectorySource source starting with directory: /home/bigdata/working/jbm-batch-log
2018-06-11 01:27:59,989 (lifecycleSupervisor-1-4) [INFO - org.apache.flume.instrumentation.MonitoredCounterGroup.register(MonitoredCounterGroup.java:119)] Monitored counter group for type: SOURCE, name: TestSource_SpoolSource: Successfully registered new MBean.
2018-06-11 01:27:59,989 (lifecycleSupervisor-1-4) [INFO - org.apache.flume.instrumentation.MonitoredCounterGroup.start(MonitoredCounterGroup.java:95)] Component type: SOURCE, name: TestSource_SpoolSource started
```

- FileZilar를 통해 /home/bigdata/working/jbm-batch-log에 파일 업로드

- 파일 업로드와 동시에 Flume 수집기가 실행됨

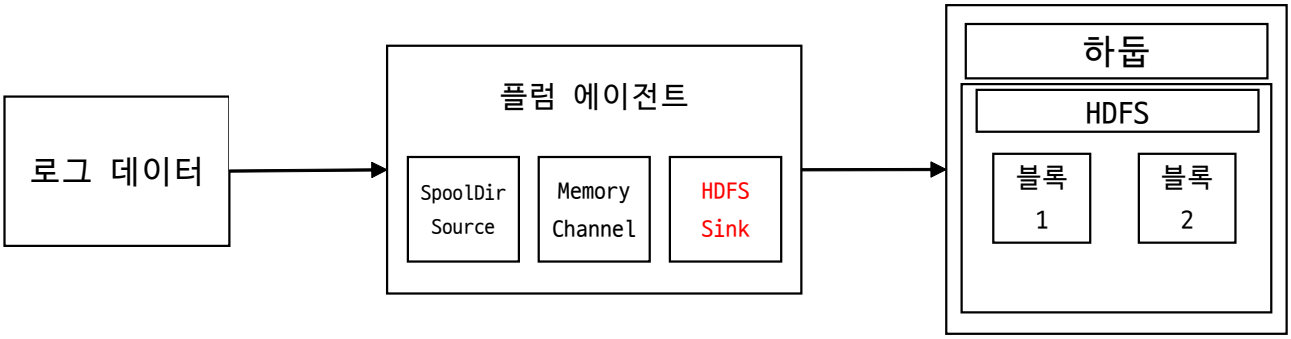
```
2018-06-11 01:42:30,866 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSink.java:95)] Event: { headers:{} body: 22 32 30 31 38 30 32 32 38 22 2C 22 EC 88 98 22 "20180228","..." }
2018-06-11 01:42:30,866 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSink.java:95)] Event: { headers:{} body: 22 32 30 31 38 30 32 32 38 22 2C 22 EC 88 98 22 "20180228","..." }
2018-06-11 01:42:30,866 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSink.java:95)] Event: { headers:{} body: 22 32 30 31 38 30 32 32 38 22 2C 22 EC 88 98 22 "20180228","..." }
2018-06-11 01:42:30,866 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.LoggerSink.process(LoggerSink.java:95)] Event: { headers:{} body: 22 32 30 31 38 30 32 32 38 22 2C 22 EC 88 98 22 "20180228","..." }
```

- 플럼의 이벤트가 header와 body로 구성됨을 확인
- 데이터의 이동을 바로 확인 가능

■ Agent 이름: JBM_Agent

JBM_Agent 구성: Source-Interceptor-Channel-Sink

1) HDFS Sink로 HDFS에 데이터를 적재



- flume의 conf 디렉터리에서 hdfs.conf 파일 생성

```
[bigdata@server01 conf]$ vi hdfs.conf
```

- hdfs.conf에 아래와 같이 입력

```
JBM_Agent.sources  = JBM_SpoolSource
JBM_Agent.channels = JBM_MemChannel
JBM_Agent.sinks    = JBM_HdfsSink
```

• HdfsSink로 연결

```
JBM_Agent.sources.JBM_SpoolSource.interceptors = timeInterceptor
JBM_Agent.sources.JBM_SpoolSource.interceptors.timeInterceptor.type = timestamp
```

• “timeInterceptor” 설정

• 플럼의 헤더에 현재 타임 스탬프가 설정되어 필요하면 헤더로부터 타임스탬프값 가져와 활용

```
JBM_Agent.sources.JBM_SpoolSource.type = spooldir
JBM_Agent.sources.JBM_SpoolSource.spoolDir                                    =
/home/bigdata/working/jbm-batch-log
JBM_Agent.sources.JBM_SpoolSource.deletePolicy = immediate
JBM_Agent.sources.JBM_SpoolSource.batchSize = 1000
```

• JBM_Agent의 Source 설정

• spooldir: 파일이 생성되면 그 파일의 내용을 수집

• “/home/bigdata/working/jbm-batch-log” 경로에서 생성되는 로그 파일 수집

• batchSize 설정값 만큼 읽어서 Channel에 전송

```
JBM_Agent.channels.JBM_MemChannel.type = memory
JBM_Agent.channels.JBM_MemChannel.capacity  = 100000
JBM_Agent.channels.JBM_MemChannel.transactionCapacity  = 10000
```

• JBM_Agent의 Channel 설정

• Channel의 종류를 “memory”로 설정

• Memory Channel: Source로부터 받은 데이터를 메모리상에 중간 적재. 성능은 높지만 메모리에 저장하는 거라 데이터가 유실될 위험이 있음

```
JBM_Agent.sinks.JBM_HdfsSink.type = hdfs
JBM_Agent.sinks.JBM_HdfsSink.hdfs.path=/user/flume/%y/%m/%d
JBM_Agent.sinks.JBM_HdfsSink.hdfs.filePrefix=event
JBM_Agent.sinks.JBM_HdfsSink.hdfs.fileSuffix=.log
JBM_Agent.sinks.JBM_HdfsSink.hdfs.inUsePrefix=_
JBM_Agent.sinks.JBM_HdfsSink.hdfs.fileType=DataStream
JBM_Agent.sinks.JBM_HdfsSink.hdfs.writeFormat = Text
JBM_Agent.sinks.JBM_HdfsSink.hdfs.batchSize = 10000
JBM_Agent.sinks.JBM_HdfsSink.hdfs.rollInterval = 0
JBM_Agent.sinks.JBM_HdfsSink.hdfs.rollCount = 0
JBM_Agent.sinks.JBM_HdfsSink.hdfs.idleTimeout = 100
JBM_Agent.sinks.JBM_HdfsSink.hdfs.callTimeout = 600000
JBM_Agent.sinks.JBM_HdfsSink.hdfs.rollSize = 67108864
JBM_Agent.sinks.JBM_HdfsSink.hdfs.threadsPoolSize = 10
```

• HDFS sink 상세 설정값

• hdfs.path: 앞에 정의한 Interceptor의 값을 이용해 HDFS의 경로를 동적으로 나누는 “path” 설정

• hdfs.filePrefix : HDFS에 생성된 파일의 접두사에 붙을 이름

• hdfs.fileSuffix : 파일에 추가되는 접미사

• inUsePrefix : flume이 현재 쓰고 있는 임시 파일에 사용되는 접두사

• hdfs.rollSize : 파일크기(64MB) 정의

```
JBM_Agent.sources.JBM_SpoolSource.channels = JBM_MemChannel
JBM_Agent.sinks.JBM_HdfsSink.channel = JBM_MemChannel
```

- Spooldir Source와 Memory Channel을 연결
- HDFS Sink와 Memory Channel을 연결

- 작성후 플럼의 홈디렉터리(apache-flume-1.8.0-bin)로 이동

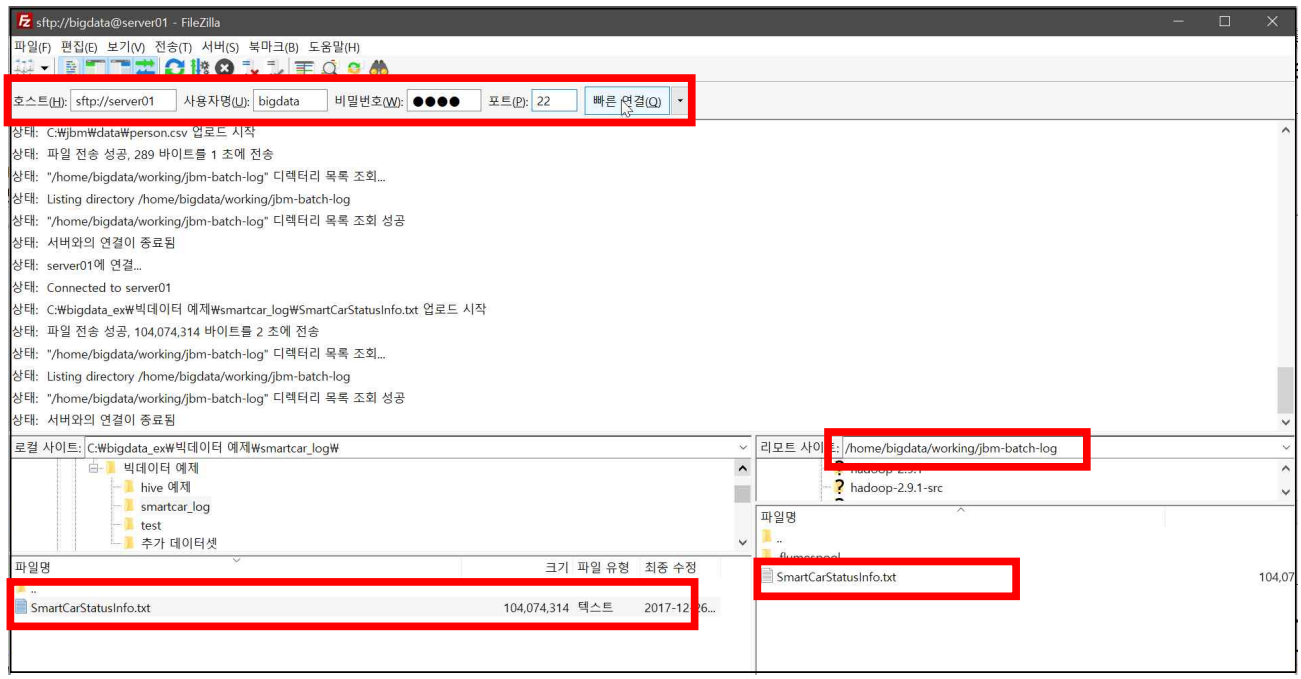
```
[bigdata@server01 conf]$ cd ..
```

- JBM_Agent 실행

```
[bigdata@server01 apache-flume-1.8.0-bin]$ ./bin/flume-ng agent -c conf -f conf/hdfs.conf -n JBM_Agent -Dflume.root.logger=INFO,console
```

- FileZilla를 통해 /home/bigdata/working/jbm-batch-log에 파일 업로드

- 호스트: server01
- 사용자명: bigdata
- 비밀번호: 1111
- 포트: 22



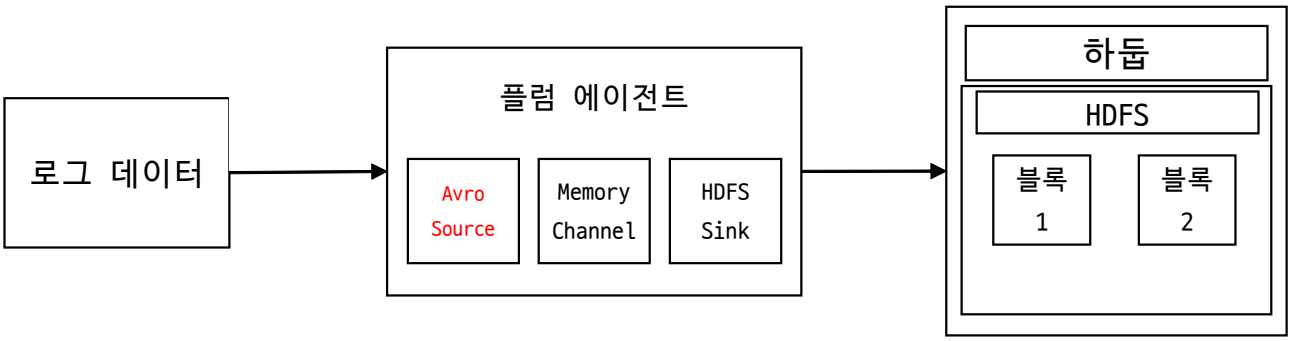
- 파일 업로드와 동시에 플룸 수집기 실행

```
2018-06-11 06:06:37,897 (hdfs-JBMSink_HdfsSink-roll-timer-0) [INFO - org.apache.flume.sink.hdfs.BucketWriter$5.ca
ll(BucketWriter.java:456)] Closing idle bucketWriter /user/flume/18/06/11/_event.1528664638791.log.tmp at 1528664
797897
2018-06-11 06:06:37,898 (hdfs-JBMSink_HdfsSink-roll-timer-0) [INFO - org.apache.flume.sink.hdfs.BucketWriter.clos
e(BucketWriter.java:393)] Closing /user/flume/18/06/11/_event.1528664638791.log.tmp
2018-06-11 06:06:37,951 (hdfs-JBMSink_HdfsSink-call-runner-8) [INFO - org.apache.flume.sink.hdfs.BucketWriter$8.c
all(BucketWriter.java:655)] Renaming /user/flume/18/06/11/_event.1528664638791.log.tmp to /user/flume/18/06/11/ev
ent.1528664638791.log
2018-06-11 06:06:37,961 (hdfs-JBMSink_HdfsSink-roll-timer-0) [INFO - org.apache.flume.sink.hdfs.HDFSEventSink$1.r
un(HDFSEventSink.java:382)] Writer callback called.
```

- HDFS의 /user/flume 에서 파일 확인 가능

```
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -ls -R /user/flume
drwxr-xr-x   - bigdata supergroup          0 2018-06-11 05:45 /user/flume/18
drwxr-xr-x   - bigdata supergroup          0 2018-06-11 05:45 /user/flume/18/06
drwxr-xr-x   - bigdata supergroup          0 2018-06-11 06:10 /user/flume/18/06/11
-rw-r--r--   3 bigdata supergroup   68531514 2018-06-11 06:04 /user/flume/18/06/11/event.1528664638790.log
-rw-r--r--   3 bigdata supergroup   35542800 2018-06-11 06:06 /user/flume/18/06/11/event.1528664638791.log
```

2) Avro Source의 RPC 통신방식으로 데이터를 받아 하둡으로 적재



- flume의 conf 디렉터리에서 avro.conf 파일 생성

```
[bigdata@server01 conf]$ vi avro.conf
```

- avro.conf에 아래와 같이 입력

```
JBM_Agent.sources.JBM_AvroSource.interceptors =
timeInterceptor typeInterceptor
```

```
JBM_Agent.sources.JBM_AvroSource.interceptors.timeInterceptor.type = timestamp
JBM_Agent.sources.JBM_AvroSource.interceptors.typeInterceptor.type = static
JBM_Agent.sources.JBM_AvroSource.interceptors.typeInterceptor.key = logType
JBM_Agent.sources.JBM_AvroSource.interceptors.typeInterceptor.value          =
cafe-batch-log
```

• 플럼의 해당 이벤트 내에 사용할 상수 값 설정

• “logType” 이라는 상수를 선언

• 값은 “cafe-batch-log”

• “cafe-batch-log” 값은 HDFS에 적재된 데이터들을 구분 할 수 있게 사용

```
JBM_Agent.sources  = JBM_AvroSource
JBM_Agent.channels = JBM_MemChannel
JBM_Agent.sinks    = JBM_HdfsSink
```

• AvroSource로 연결

```
JBM_Agent.sources.JBM_AvroSource.type = avro
JBM_Agent.sources.JBM_AvroSource.channels = JBM_MemChannel
JBM_Agent.sources.JBM_AvroSource.bind = 0.0.0.0
JBM_Agent.sources.JBM_AvroSource.port = 65111
```

• bind : 수신할 호스트 이름 또는 IP주소를 입력

• port : 바인딩할 포트번호

```
JBM_Agent.channels.JBM_MemChannel.type = memory
JBM_Agent.channels.JBM_MemChannel.capacity  = 100000
JBM_Agent.channels.JBM_MemChannel.transactionCapacity  = 10000
```

• JBM_Agent의 Channel 설정

```
JBM_Agent.sinks.JBM_HdfsSink.type = hdfs
JBM_Agent.sinks.JBM_HdfsSink.hdfs.path=/user/flume/%y/%m/%d
JBM_Agent.sinks.JBM_HdfsSink.hdfs.filePrefix= %{logType}
JBM_Agent.sinks.JBM_HdfsSink.hdfs.fileSuffix=.log
JBM_Agent.sinks.JBM_HdfsSink.hdfs.inUsePrefix=_
JBM_Agent.sinks.JBM_HdfsSink.hdfs.fileType=DataStream
JBM_Agent.sinks.JBM_HdfsSink.hdfs.writeFormat = Text
JBM_Agent.sinks.JBM_HdfsSink.hdfs.batchSize = 10000
JBM_Agent.sinks.JBM_HdfsSink.hdfs.rollInterval = 0
JBM_Agent.sinks.JBM_HdfsSink.hdfs.rollCount = 0
JBM_Agent.sinks.JBM_HdfsSink.hdfs.idleTimeout = 100
JBM_Agent.sinks.JBM_HdfsSink.hdfs.callTimeout = 600000
JBM_Agent.sinks.JBM_HdfsSink.hdfs.rollSize = 67108864
JBM_Agent.sinks.JBM_HdfsSink.hdfs.threadsPoolSize = 10
```

• HDFS sink 상세 설정값

• %{logType}: 파일 접두사에 interceptor에서 정의한 명칭이 붙음

```
JBM_Agent.sources.JBM_AvroSource.channels = JBM_MemChannel
JBM_Agent.sinks.JBM_HdfsSink.channel = JBM_MemChannel
```

• Avro Source와 Memory Channel을 연결

• HDFS Sink와 Memory Channel을 연결

- hadoop으로 이동 후 파일 확인

```
$ cd /home/bigdata/hadoop-2.9.1
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -ls -R /user/flume
```

```
[bigdata@server01 hadoop-2.9.1]$ ./bin/hdfs dfs -ls -R /user/flume
drwxr-xr-x   - bigdata supergroup          0 2018-06-11 05:45 /user/flume/18
drwxr-xr-x   - bigdata supergroup          0 2018-06-11 05:45 /user/flume/18/06
drwxr-xr-x   - bigdata supergroup          0 2018-06-11 09:34 /user/flume/18/06/11
-rw-r--r--   3 bigdata supergroup    1001324 2018-06-11 09:34 /user/flume/18/06/11/cafe-batch-log.1528675569111.log
```